Learning the Non-Coding Genome

Dissertation

zur Erlangung des Grades eines Doktors
der Naturwissenschaften (Dr. rer. nat.)
am Fachbereich für Mathematik und Informatik
der Freien Universität Berlin.

Vorgelegt von Max Schubach
Berlin, 2017

Betreuer & Erstgutachter: Prof. Dr. Peter N. Robinson

Zweitgutachter: Prof. Dr. Rosario M. Piro

Drittgutachter: Prof. Dr. Giorgio Valentini

Tag der Disputation: Donnerstag, 20. September, 2018

Datum der Veröffentlichung: Dienstag, 13. November, 2018

Abstract

The interpretation of the non-coding genome still constitutes a major challenge in the application of whole-genome sequencing. For example, disease and trait-associated variants represent a tiny minority of all known genetic variations, but millions of putatively neutral sites can be identified. In this context, machine learning (ML) methods for predicting disease-associated non-coding variants are faced with a chicken-and-egg problem – such variants cannot be easily found without ML, but ML cannot be applied efficiently until a sufficient number of instances have been found. Recent ML-based methods for variant prediction do not adopt specific imbalance-aware learning techniques to deal with imbalanced data that naturally arise in several genome-wide variant scoring problems, resulting in relatively poor performance with reduced sensitivity and precision.

In this work, I present a ML algorithm, called hyperSMURF, that adopts imbalance-aware learning strategies based on resampling techniques and a hyper-ensemble approach which is able to handle extremely imbalanced datasets. It outperforms previous methods in the context of non-coding variants associated with Mendelian diseases or complex diseases. I show that imbalance-aware ML is a key issue for the design of robust and accurate prediction algorithms. Open-source implementations of hyperSMURF are available in R and Java, such that it can be applied effectively in other scientific projects to discover disease-associated variants out of millions of neutral sites from whole-genome sequencing.

In addition the algorithm was used to create a new pathogenicity score for regulatory Mendelian mutations (ReMM score), which is significantly better than other commonly used scores to rank regulatory variants from rare genetic disorders. The score is integrated in Genomiser, an analysis framework that goes beyond scoring the relevance of variation in the non-coding genome. The tool is able to associate regulatory variants to specific Mendelian diseases. Genomiser scores variants through pathogenicity scores, like ReMM score for non-coding, and combines them with allele frequency, regulatory sequences, chromosomal topological domains, and phenotypic relevance to discover variants associated to specific Mendelian disorders. Overall, Genomiser is able to identify causal regulatory variants, allowing effective detection and discovery of regulatory variants in Mendelian disease.

Acknowledgements

At this point I would like to thank everybody who supported me during my doctoral candidate time. First of all I have to express my gratitude to my supervisor Prof. Dr. Peter N. Robinson. My research and work would not be possible without his thoughts, ideas, patient and helpful discussions. Thank you for your great support during this time.

My special thanks goes to my collaborates at the Dipartimento di Informatica at the University of Milano. Especially Prof. Dr. Giorgio Valentini for the constructive discussion and the productive and pleasant time together in Berlin. But also many thanks to Dr. Matteo Re and Marco Notaro from Giorgio's group.

Many thanks to all that helped in the development of Exomiser and Genomiser, in particular Prof. Dr. Damian Smedley and Dr. Jules OB. Jacobsen. Thanks to the core Jannovar crew Marten Jäger and Dr. Manuel Holtgrewe for developing Jannovar further to a well implemented and well documented software, which is often not the case when it comes to scientific software.

I also would like to thank Prof. Dr. Peter Krawitz, Prof. Dr. Uwe Kornak, Prof. Dr. Stefan Mundlos and all others from the Medical Genetics and Human Genetics department for the pleasant time at the Charité.

Finally, I would like to thank Linde Böhm and Dr. Manuel Holtgrewe for the helpful feedback on the writing part.

This work is dedicated to Dr. Hans Reinhard Schubach.

∗ 17.03.1950; † 08.10.2016

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Two buzzwords were omnipresent in the press over the last years: *big data* and *artificial intelligence (AI)*. All kinds of fields generate big data, for example medicine, social media, environmental and ecological research, process optimization, and lots of other areas. These data harbor important information for decision support in medicine, optimization of processes, for example in terms of power-saving, or an optimal embedding of the environment to preserve and use it for living, mobility and agriculture.

Challenges that come along with big data are its storage, effective accessibility of the data and its interpretation. All three sections are highly dependent of each other, but at the end the interpretation becomes a crucial part. The hypothesis is that the answer raised by questions of a big dataset is hidden within it and that by using the right assumptions solutions can be extracted. Therefore sophisticated algorithms for analysis are necessary to achieve the promise of big data.

One of the tools to extract knowledge out of big data might be AI. In AI a computer program is able to find logical connections within a big dataset and uses them to make assumptions out of millions of data points. Nowadays, the most popular AI idea is *deep learning*. Deep learning is a class of *artificial neural networks (ANNs)* whose design was inspired by a network of neurons in the brain. Raw information is processed through the different neurons of the network. Thereby different artificial neurons (edges) get activated or deactivated and information is combined together at synapses (vertices) until a final interpretable output is generated. Learning with ANNs is referred as 'deep' if the ANN design contains at least three layers of vertices between the input and the output layer, so called *hidden layer*s. Thus, a deep ANN receives at its input interface lots of data that will be processed trough a network with different types of layers, and at the end the important information is extracted and assumptions are made.

One famous example of deep learning algorithm is the *AlphaGo* machine from Google [1]. AlphaGo is able to play the board game *Go* which is supposed to be extremely complex for a computer and so far professional Go players were unbeatable by a computer. But recently AlphaGo

was able to beat the best Go players in the world. It is interesting to consider how AlphaGo achieved this. At the cutting edge there was an enormous amount of data the deep learning algorithm of AlphaGo was trained on. After Silver *et al.* [1] had a reasonable well running machine they let the software play random games against itself more than thousands of times to raise its power.

Generating data from self-play is an ideal environment for a software or algorithm to learn a specific task. Therefore self-play is a common approach for *reinforcement learning* in game AI [2]. In reinforcement learning the software tries to find a strategy by itself to maximize a cumulative reward. In game AI this will be the winning or losing of a game but the algorithm has to decide on its own which move will be the best. Thus, through self-play the algorithm will see millions of possible moves and so learns to relate which move is in which situation more likely to win or loose a game.

In other research fields the generation of data can be limited. If we apply the idea to medicine we easily can find data where it is not possible to create or to find new examples because of cost, ethical, technical, feasible or time reasons. For example if we want to predict the cause of a rare genetic disease we will be limited to the incidence of that disease. A genetic disease is defined as rare if the prevalence is lower than 1 in 2000. If we consider a population of over 2 million in Europe there will be an upper limitation of 350,000 on a rare disease. Some extremely rare diseases might affect less than 1000 people in Europe or even in the world. In this example we are faced with several problems. Data generation, for example using whole-genome sequencing (WGS) (see Section 1.2), will be time consuming and cost expensive. It will also not be possible to include everyone from our cohort because of logistic issues or if the disease cannot be reliably diagnosed in a sufficient number of patients. Consequently on the one hand we have small, maybe imperfect, examples for the rare genetic disease, and on the other hand a large cohort of healthy examples leading to natural imbalances within the collected data.

In the initial example AlphaGo has none of these problems because it is possible to generate an extremely large dataset using self-play. The outcome, win or loose, is easily measurable, always balanced, one player wins and the other one loses, and false positives (cheating) are unlikely. When it comes to genetic diseases, another problem is that the mechanism causing the disease might be extremely complex and different over the genome. In a board game the rules are relatively small. They can, like in Go, lead to complex patterns but the biology will always be more diverse than any present or future board game.

Structure of this Chapter This chapter gives short background information on the main important topics in this work. First, Section 1.1 wraps up the biological background with a focus on genetic processes. Second, Section 1.2 shows how genetic code can be determined and how genetic differences between individuals can be interpreted. Third, Section 1.3 introduces *machine learning (ML)*, a field in AI. Finally, this chapter concludes in Section 1.4 with an outline of this thesis.

1.1 Biological Background

The letters of life are the four nucleotides adenine (A), cytosine (C), thymine (T), and guanine (G). The deoxyribonucleic acid (DNA) is a polymeric macromolecule made from these four nucleotide monomers also known as bases. A and T can pair together using two hydrogen bonds and C pairs together with G using three hydrogen bonds. This makes the DNA double-stranded with complementary bases at each site (see Figure 1.1).

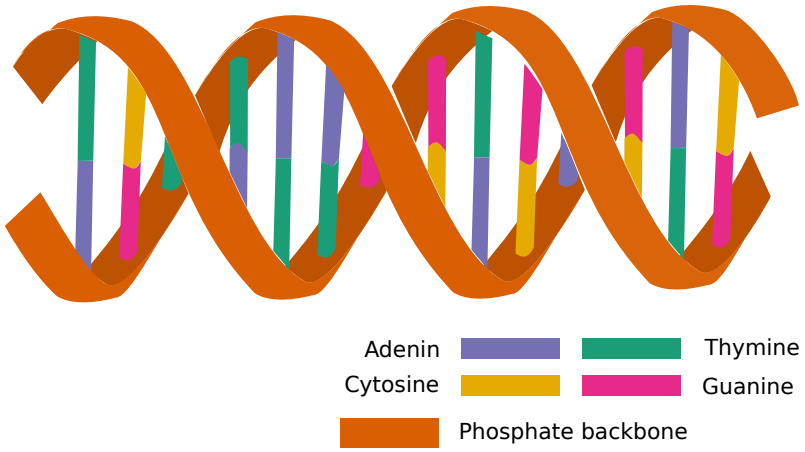


Figure 1.1: The structure of the DNA. The DNA is a double-stranded helix. Each strand has a phosphate backbone to which the nucleotides are connected. The two strands are paired through the complementary bases A with T and C with G.

The human genome is composed of two copies of the 22 *autosomes*, the two *allosomes* (two X-chromosomes for female and one X and one Y chromosome for male), and several copies of circular mitochondrial DNA. The DNA decodes proteins in their sequence information. Proteins perform a vast array of functions within organisms like catalyzing metabolic reactions. They are the engine of the cell. Proteins consist of one or more chains of amino acid (AA) residues. In eukaryotes there are 21 proteinogenic AAs. Each of the AA is encoded by at least one specific triplet (*codon*) of nucleotides.

The triplets decoding for a protein are structured in genes at the DNA level. The initial step of decoding the DNA into protein is to translate the sequence of codons from a gene into messenger ribonucleic acids (mRNAs), starting at a specific *start-codon* and ending at a *stop-codon*. The ribonucleic acid (RNA) is single stranded and has an uracil (U) instead of an A. The process of translation into mRNA is called *transcription* (see Figure 1.2).

Genes consist of coding or non-coding information. The coding information is used to generate the protein, and the non-coding parts have regulatory responsibilities for synthesizing the protein or they will be removed during the generation of the mRNA. Blocks of non-coding information, called *introns*, are removed from a pre-mRNA and the coding blocks, called *exons*, are spliced together (see Figure 1.2). This process is called *splicing* and results in a mature-mRNA that is used for the next step, the *translation*. Within the translation the codons are read by the ribosome complex and the corresponding AAs are concatenated. Then the chain is folded and sometimes multiple chains are grouped together resulting in the functional protein.

Regulatory Code Only around 3 % of the human DNA is coding sequence. The remaining non-coding part, previously referred to as "dark matter", has a regulatory purpose and positively or negatively regulates the transcriptional activity of genes. There are several known categories of functional elements in the non-coding genome.

Genes are flanked by untranslated regions (UTRs), the 5'UTR before the start-codon and the 3'UTR after the stop-codon, visualized in Figure 1.2. Both UTRs will be present in the mature-mRNA. To be precise a proportion of the 5'UTR in eukaryotes, which contain an *upstream open reading frame (uORF)* within the 5'UTR, is sometimes translated into a protein product but in some mRNAs the UTR has a major regulatory function: it forms complex secondary structures to regulate the expression of the gene and it is involved in the recruitment of ribosomal complex to start the translation. The 3'UTR contains regulatory regions that post-transcriptionally influence the gene expression. For example it contains microRNA (miRNA) binding sites to down-regulate the gene expression. During transcription proteins polyadenulate the transcript, which means that a poly-A tail is added at the end of the 3'UTR. This tail is important for the nuclear export, translation and stability of the mature-mRNA.

DNA elements that directly regulate the transcription are *promoter*s, *enhancer*s and *silencer*s. promoters are located near the transcription start site upstream of the gene. They are involved in the binding of proteins which recruit the RNA polymerase (transcription factors) and the binding of the RNA polymerase itself to start the transcriptional mechanism. The transcription factors have specific activator or repressor sequences to regulate the gene expression. Enhancers or silencers can also recruit transcription factors but in contrast to promoters, they can be several hundred kilo bases (Kb) upstream or downstream of the gene. Therefore the DNA has to be folded. A schematic overview of the regulatory elements is in Figure 1.2.

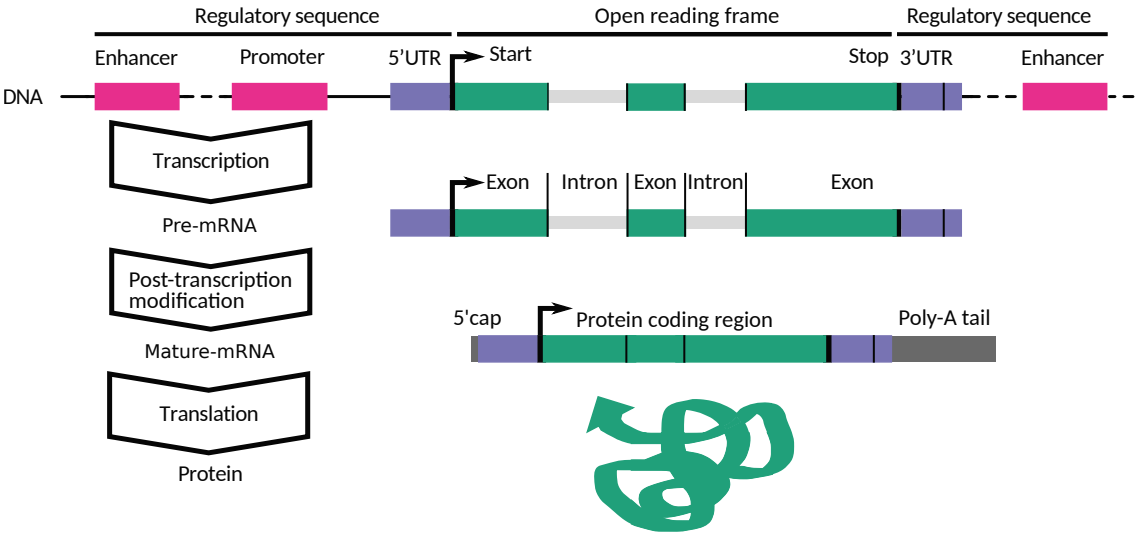


Figure 1.2: Schematic structure of an eukaryotic protein-coding gene. Translation as well as transcription is controlled by regulatory elements (pink and blue). During the post-transcription step the introns (light grey) will be spliced out and UTR regions (blue) will be modified with 5'cap and a poly-A tail (dark grey). At the translation stage the coding sequence (green) will be translated into AAs. Finally the protein gets folded into its specific 3D-structure (bottom).

Topological Confirmation Regulatory elements, like enhancers and silencers, have to come to close proximity to the promoter to get involved into gene regulation. Therefore the linear genome has to organize itself through folding to a 3D-structure. In the cells the folding is usually made by a complex of macromolecules called *chromatin*. For example, if a protein is transcribed, the chromatin structure at that site will be loose to allow access for transcription factors and polymerases to the DNA. Therefore the chromatin structure will give a hint if a gene is actively transcribed or not.

Several methods were developed like *Chromatin immunoprecipitation sequencing (ChIP-seq)* or *DNase I hypersensitive sites sequencing (DNase-seq)* to find out the states of chromatin. Using methods of chromosome conformation capture techniques [3] reveals that the genome is subdivided into *topologically-associated domains (TADs)*. This subdivision is highly conserved over cell types or even different species, so we can claim that TADs are involved in long-range regulatory interactions. Another important observation is that regulatory elements only interact with promoters and their genes within a TAD (intra-TAD) but not on elements outside of it (inter-TAD). Figure 1.3 shows an overview of TAD and enhancer regulation within it.

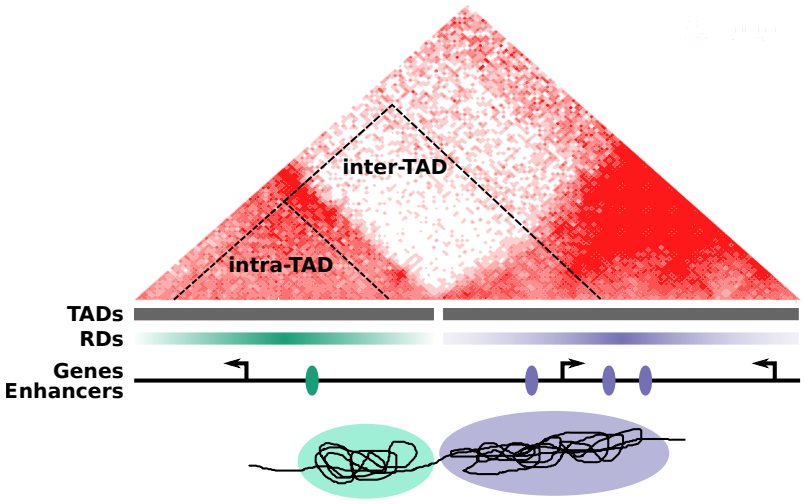


Figure 1.3: Topological confirmation of the genome. Representation of a hypothetical locus. Hi-C interaction frequencies are displayed as a two-dimensional heatmap (red), where intra-TAD contacts are more frequent than inter-TAD contacts. TADs and regulatory domains (RDs) are represented as bars; genes and enhancers are depicted as arrows and ovals, respectively. From the Hi-C heatmaps, one can envision TAD as entangled skeins of DNA (bottom).

1.2 Sequencing and Sequence Analysis

The process to obtain the sequence of nucleotides of an organism is called *sequencing*. The first method was developed by Sanger *et al.* [4] and Sanger & Coulson [5] and this method and further developments are known as *classical sequencing* or *Sanger sequencing*. In comparison to the classical approach the *next-generation sequencing (NGS)* methods have a much higher throughput and they enable the possibility to sequence all protein coding regions in the human genome (*exome*) or even the whole human genome under reasonable costs. These sequencing technologies are Illumina, 454/Roche, and IonTorrent, also referred to as second-generation technologies. The drawback of the second generation is that they can only sequence short fragments of DNA. The third-generation technologies, like PacBio or MinION, are able to sequence longer fragments but have a reduced throughput and a higher error rate compared to the second-generation.

The DNA output fragments (reads) of the sequencers have to be puzzled together to receive the complete nucleotide sequence of the target. The reads can be assembled *de-novo* into longer *contig*s or, if a *reference sequence* of the organism is available, they can be aligned against it. Mostly the reference sequence is a consensus sequence, obtained by a collection of sequenced individuals. The decision of using a reference alignment depends on the sequencing technology and on the species. If no reference alignment is present the *sequence assembly* approach has to be done. Here usually longer reads are factorized because it is easier to retain the global structure of the genome and repetitive DNA regions can be resolved. Mostly the long reads are combined with short reads from the second-generation to have a higher resolution. For reference *mapping* the second generation is used frequently because the genome structure is given by the reference and small differences can be detected with high reliability.

In addition to most approaches it is favorable to compare differences in sequences and therefore a mapping to a reference sequence results in a standardized and comparable output. This enables databases to be constructed and results to be shared among the community. The drawback with these technologies is that the resolution on complex structural rearrangement and repetitive regions is low.

Sequence Variation The diversity of a species is reflected by differences in their genome. These variations can be detected if the assembled sequences between two individuals are compared or if the genome of an individual is compared to a reference genome. Complete *chromosomal aberration*s are the largest but also the least frequent class of variations. Insertions, deletions or translocations, like inversions, are larger *structural variations (SVs).* The class of *small insertion or deletions (InDels)* can be detected by a single read from a NGS technology of the second-generation. Therefore sizes of InDels are often defined as smaller than 25 or 50 base pairs (bp). The smallest and most frequent variations are *single nucleotide variants (SNVs).* For example, compared to the human reference genome there will be over 3,000,000 single nucleotide positions that differ in a whole-genome sequence of a human individual.

By using a reference genome it is possible to compare variants between individuals in a population. This is enormously helpful to find the causative variant, also called *mutation*, in an individual with a rare genetic disease because we are able to separate common from rare vari-

ants. Frequent variants in population are called *polymorphism*s, defined as the allele different to the reference which occurs more than 1 % in a population. Thus a frequent SNV is called *single nucleotide polymorphism (SNP)*. Variants with a lower *allele frequency (AF)* than 1 % are rare variants. In order to find the cause of a rare disease this variant set is usually used to search for the causative pathogenic mutation.

By using the location of the variants and a database with genes, it is possible to discover the so called *functional class* of a variant. The functional class tells us something about the region of the variant, like if it is between two genes (intergenic), within an intron (intronic), an UTR or in the coding sequence of a gene. For coding sequences the classification is more detailed. Using the degenerated code, the functional class will describe if the SNV changes an AA in the resulting protein (*missense*) or not (*synonymous*) in comparison to the reference. Also an introduction of new stop or start codons, as well as destroying them, can be possible (*nonsense*). For InDels the functional class describes if it is *in-frame* or a *frameshift*. Nonsense, missense and frameshift variations are more likely disease causing than other variant classes.

To assess the potential deleteriousness of variants further, it is interesting how well conserved the site is, meaning if the same DNA or AA sequence is present in other species. If we have a highly conserved site it is likely that it has an important function and changes might cause a dysfunctional gene or protein. In terms of a missense variant it is important to know which new AA is introduced. A similar AA might be unproblematic but if the new AA is different in size, polarity or hydrophobicity the function of the protein might be influenced. To do this in an automated way, several conservation or pathogenicity scores were developed [6–13]. They integrate the presented ideas into one final score, which is much easier to handle instead of having thousands of variants to evaluate.

The final step of identifying a causal variant is linking the variant to the phenotype of the patient. Either the affected gene or at least the gene family or genes in the same biological pathway should be known to cause similar symptoms. Finally the pathomechanism should be proposed.

1.3 Data Mining and Machine Learning

Data is omnipresent in our society and normally we are awash with information. In addition there is a big growing gap between generating and understanding the data. *Data mining* is the process of discovering patterns in large datasets. These data are stored electronically and the search is done automated or augmented by a computer. Therefore data mining is used to extract knowledge out of large data.

ML evolved form the study of pattern recognition and computational learning theory in AI. In ML the focus is on the learning process. Learning enriches knowledge and the knowledge can be used to categorize or classify new observations. This is why ML has its focus on the technical learning process and the performance of it. A common example will be to observe the present behavior and compare it with the past behavior. In ML the learning is named as *training* and the performance evaluation as the *testing* step.

There are different styles of learning in data mining applications. Depending on the learning signal, ML tasks are typically classified into three broad categories. If the training example inputs and outputs are given we can learn the link between the input to the output. This is done by *supervised learning*. If the output is not given and the learning algorithm has to structure the input by its own we speak of *unsupervised learning*. In between these two categories there is *semisupervised learning* where the training input is incomplete and output classes are missing.

If we describe them on the desired output we can use the following categories. In *classification learning* we have a set of classified examples from which we want to learn a way to classify unseen examples. In *association learning* every association among the underlying data is sought, not a single prediction of a class value. There are numeric predictions, where the outcome is a numeric quantity and not a discrete class. The field of *clustering* tries to find groups that belong together.

1.4 Thesis Outline

In this thesis, I give an overview of ML techniques with a special focus on imbalanced data, explain important variant scores for evaluation of non-coding variations and describe my contribution to this field. The thesis is structured on the one hand in a technical part, containing the development of a new ML algorithm that can be used on highly imbalanced data. On the other hand there is an applied part where the invented algorithm is applied on genetic data in combination with other methods to discover mutations in whole genomes.

In this chapter, I gave an overview to the biological background, sequencing and variant evaluation in a Mendelian context and a short introduction into ML. Chapter 2 defines formal and mathematical preliminaries, as well as the main ML concepts, the used ML methods and classifiers, and ML performance validation strategies. At the end, Chapter 2 contains a detailed description of available non-coding deleteriousness scores that were used in this work, describing their ML strategy, their training data and performance.

In Chapter 3 I present *hyperSMURF*, a ML algorithm that is specifically designed for highly imbalanced (genetic) data. After that, I describe the experimental evaluation of hyperSMURF and its results and compare it to retrained learners that were used to generate other non-coding deleteriousness scores.

Chapter 4 introduces the software framework *Exomiser* along with the tool *Jannovar*. Exomiser filters and prioritizes variants for novel disease-gene discovery or differential diagnostics of Mendelian diseases. I explain the modifications on Exomiser that were made to extend the variant discovery to the non-coding part of the genome, called *Genomiser*. Alongside I introduce the pathogenicity score *ReMM* which was developed to asses non-coding Mendelian variants using the hyperSMURF approach. ReMM score and Genomiser are experimentally validated and compared to other scores and software. Finally, in Chapter 5 I give a conclusion of this thesis.

Chapter 2

Preliminaries

In this chapter I will define formal and mathematical preliminaries in Section 2.1. Then, in Section 2.2, I explain the main concepts of ML, give formal definitions in this field, and describe the important algorithms that were used in this work. Section 2.3 gives a brief introduction into *ontologies*. Finally, Section 2.4 contains a detailed description of available non-coding deleteriousness scores that were used in this work, including their ML strategy, their training data and performance.

2.1 Mathematical Preliminaries

In this section I describe the main mathematical concepts and notations that I will use later in this work. The notation and definitions are based on the work of Wolff *et al.* [14].

2.1.1 Sets and Matrices

Set A *set* S is a well-defined collection of distinct objects or elements. They are denoted by curly brackets. For example $S = \{0, 23, 42\}$ has the three elements 0, 23 and 42. Sequences in sets are denoted by dots, like $S = \{0, 1, \ldots, 42\}$. I will use the symbol $\mathbb{N}$ as the set of all positive, or natural numbers. $\mathbb{N}_0$ are all non-negative numbers and $\mathbb{Z}$ is the set of all integers.

The number of elements or the size of a set S is denoted by $|S|$. The cartesian product of two sets S_1 and S_2 is denoted as $S_1 \times S_2$. $S_1 \subset S_2$ denotes that S_1 is a subset of S_2 and if S_1 is a subset or equal to S_2 it is denoted by $S_1 \subseteq S_2$. The union of two sets S_1 and S_2 is denoted as $S_1 \cup S_2$ and the intersection as $S_1 \cap S_2$.

Matrix A *matrix* $\mathbf{M}$ is a rectangular array of elements, mostly numbers. It has m rows and n columns. For example the matrix $\mathbf{M} = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 23 & 24 \end{bmatrix}$ is a 2×3 matrix. A specific element is denoted by $a_{i,j}$ where $\max(i) = m$ and $\max(j) = n$. For example $a_{2,3}$ of the previous matrix $\mathbf{M}$ is $a_{2,3} = 42$. Matrices will be denoted as a bold capital letter. An $1 \times n$ or $m \times 1$ matrix is called *row vector* or *column vector* $\mathbf{v}$ and is denoted by a bold lower case letter. A $n \times n$ *square matrix* $\mathbf{M}$ has the same number of rows and columns. $\mathbf{I}$ is the *identity matrix*, a square matrix where $a_{i,j} = 1$ for all $i = j$ and $a_{i,j} = 0$ for all $i \neq j$. The *inverse matrix* of a square matrix $\mathbf{M}$, denoted by $\mathbf{M}^{-1}$, is a matrix $\mathbf{M}^{-1}$ such that $\mathbf{M}\mathbf{M}^{-1} = \mathbf{I}$. The rank of a matrix $\mathbf{M}$ is the dimension of the vector space spanned by its columns (or rows) and is denoted by $rank(\mathbf{M})$.

A number λ and a non-zero vector $\mathbf{v}$ satisfying $\mathbf{M}\mathbf{v} = \lambda\mathbf{v}$ are called an *eigenvalue* and an *eigenvector* of a $n \times n$ matrix $\mathbf{M}$, respectively. The number λ is an eigenvalue of $\mathbf{M}$ if and only if $\mathbf{M} - \lambda\mathbf{I}_n$ is not invertible. The *eigendecomposition* is the factorization of a diagonalizable matrix into a canonical form, whereby the matrix is represented in terms of its eigenvalues and eigenvectors $\mathbf{M}\mathbf{v} = \lambda\mathbf{v}$. The matrix is diagonalizable if there exists an invertible matrix $\mathbf{T}$ such that $\mathbf{T}^{-1}\mathbf{M}\mathbf{T}$ is a *diagonal matrix*.

Element $a_{i,j}$ of a *covariance matrix* Σ represents the covariance between the ith and jth elements of a random vector $\mathbf{X} = \begin{bmatrix} X_1 \\ \vdots \\ X_n \end{bmatrix}$ of length n ($\Sigma_{i,j} = \text{cov}(X_i, X_j)$). A random variable X_i can be interpreted as all different variables of an attribute within our instance (see Section 2.2).

2.1.2 Graphs

In this thesis only a broad description of a *graph* is need. So a graph is an ordered pair $G = (V, E)$ comprising a set of vertices V together with a set of edges E. An edge $z \in E$ is a subset of two vertices, e.g. $z = (x, y), x, y \in V$, which the edge connects. A graph can be undirected meaning that the edge $z = (x, y)$ is the same as the edge $z = (y, x)$. If the graph is directed the edges E, or arrows, are directed meaning that edge $z = (x, y)$ goes from vertice x to vertice y but not the other way round.

Directed Acyclic Graph A directed acyclic graph (DAG) is a directed graph that has no cycles. This characteristic results in a topological ordering of a DAG. So a DAG has only sequences of vertices such that every edge is directed from earlier to later in the sequence.

Tree and Forest A tree is an *undirected graph* in which any two vertices are connected by exactly one path. So every acyclic connected graph is a tree. In this work only rooted trees were used. I will speak of a forest when we have a disjoint union of trees.

Common Ancestor In rooted trees of DAGs it is possible to compute the common ancestors (CAs) of set of vertices. They can be observed by visiting the path from all vertices back to the

root in a tree or following the reverse path in a DAG. If a node is visited by all backtraces from the initial set it will be a CA. Sometimes the lowest common ancestor (LCA) is computed. It is the shared ancestor that is located farthest from the root.

2.2 Machine Learning Preliminaries

This section wraps up important definitions in ML (Section 2.2.1), a formal description of *classifier*s used in this work (Section 2.2.3) and strategies for training and testing including different performance measurements (Section 2.2.5). Because Chapter 3 presents a new ensemble method on imbalanced datasets, known possible classification strategies on such data are described in Section 2.2.4. The main concept behind the new method is ensemble ML and Section 2.2.2 describes this in more detail. The notation and definitions are based on the work of Witten *et al.* [15].

2.2.1 Instances and Attributes

The examples or observations of a classification problem are called *instance*s $\mathcal{I}$. These instances are used to train and test the classifier (see Section 2.2.5). Each instance is an individual and independent example of the concept to be learned. A dataset is a matrix in which every row vector represents an instance, a vector of *feature*s Θ also known as *attribute*s. Each instance has a fixed and predefined set of attributes. For every instance the value of an attribute is a measurement of the quantity to which the attribute refers. Different classes can be defined for attributes. The most common variables are continuous so that the quantity is numeric like real or integer values (note that integer are not continuous in a mathematical sense). Another important class are categorical variables where the quantities are nominal. A *nominal attribute* has one value out of a predefined finite set of possibilities. But there are other levels of measurement like ordinal, interval or ratio.

2.2.2 Ensemble Learning

Ensemble methods combine multiple ML algorithms to increase the predictive performance. Different training sets can learn a model by themselves and afterwards the trained models are combined to produce an ensemble of learned models. Often the different trained models have a weak predictive power but the combination of them can be very powerful. Important models are *bagging*, *boosting*, *randomization* and *stacking*.

The idea behind the methods is the so called *bias-variance decomposition*. In terms of classification the error of an algorithm can be divided in *bias* and *variance*. The bias is the error rate for a particular learning algorithm. It measures how well the learning method matches the problem. E.g. a high bias results in missing relations between features and target outputs (underfitting). This is because a learner is only trained by a finite number of instances and is therefore not

fully representative over the overall population of possible instances. The expected value of this component is called variance of a learning method for a particular problem.

Here, I describe bagging and randomization in more detail because the new ensemble method described in Chapter 3 uses the bagging principle together with a Random Forest (RF) classifier which uses bagging and randomization.

Bootstrap Aggregating The simplest way to combine different trained classifiers together is to make a majority vote for a final prediction using all classifiers (for classifiers see Section 2.2.3). If the prediction is numeric, for example if we receive probabilities, we can use an average vote. In bootstrap aggregating or bagging [16] every classifier has the same weight. So every vote is equal (e.g. the boosting approach uses a weighted vote). We can think of a bagging example by randomly choosing n training datasets of the same size from our problem domain (sampling with replacement). After that n different classifiers $C = \{C_1, C_2, \ldots, C_n\}$ are trained and finally we get the prediction p of all classifiers by using the average prediction $p = \frac{\sum_{i=1}^{n} p_i}{n}$. In Algorithm 2.1 the bagging pseudocode is shown.

With bagging it is possible to get an unbiased performance estimate during the training phase, which is known as *out-of-the-bag error* (for performance measurement see Section 2.2.5). The subsampling allows us to test every training instance $\mathbf{x}_j \in \mathcal{I}$ with all classifiers $\hat{C}$, $\hat{C} \subset C$, where $\mathbf{x}_j$ was not used to train the classifiers $\hat{C}$. The drawback is that the out-of-the-bag error often underestimates the actual performance of the bagged ensemble classifier.

Randomization Randomization can be used to increase the variance of a learning algorithm. Apart of using random input instances, an option is to run multiple classifiers on all training instances using different pseudorandom number generators to introduce a random variance and finally combine the results by majority or average vote. Another well known method is random subspaces. For every iteration of training a random subset of attributes is chosen. This concept was implemented in randomized decision trees and the popular classifier RF is a combination of *random tree*s and bagging [17]. The combination is often powerful because the introduced variance is different in bagging and randomization. Therefore the total error will be minimized.

2.2.3 Classifiers

The algorithm that is used in a classification problem is known as classifier or learner $h(\mathbf{x})$ where $\mathbf{x}$ is an input vector. It can consist of a single function or a complex concept that maps the input data to the categorial output. In *ensemble learning* (see Section 2.2.2) multiple classifiers are used for training. Here we can speak of one ensemble classifier that uses classifiers. Again these classifiers can be ensemble classifiers but the nuclear entity of learners is a base classifier. In this work a new algorithm is introduced (Section 3.1) which is based on an ensemble classifier called RF. RFs use as base classifier the concept of decision trees used as random trees which will be introduced in the next paragraphs.

Algorithm 2.1: Bootstrap Aggregating (Bagging)

Training phase

Input : $\mathcal{I} \leftarrow$ All training instances.
$n \leftarrow$ Ensemble size.

Output : C Ensemble of classifiers.

`// Initialize set of classifiers.`

1 $C = \emptyset$

`// Train` n `classifiers.`

2 **for** $i \leftarrow 1$ **to** n **do**
3 Take a bootstrap sample $\mathcal{I}_i$ from $\mathcal{I}$.
4 Build classifier C_i using $\mathcal{I}_i$ as training set.
5 Add classifier C_i to current Ensemble $C = C \cup C_i$.
end
6 **return** C

Classification phase

Input : $\mathbf{x} \leftarrow$ Single instance to test.
$C \leftarrow$ Ensemble of classifiers of size n.

Output : $p \leftarrow$ Average probability over all predictions of $\mathbf{x}$ on C.

1 **for** $i \leftarrow 1$ **to** n **do**
2 $p[i] \leftarrow$ prediction of $\mathbf{x}$ on C_i.
end
3 $p \leftarrow \frac{\sum_{i=1}^{n} p[i]}{\mathit{length}(p[\,])}$
4 **return** p

Decision Trees A decision tree takes advantage of a tree-like model to support decisions by *divide-and-conquer*. A node in a decision tree is for a specific test of an attribute, mostly an attribute value with a constant. To classify an unknown instance, it is routed down the tree according to the values of attributes tested at the nodes along the path. The leaf nodes apply a classification, set of classifications, or a probability distribution over all classifications to instances when they reache it. Nodes for decisions with numeric attributes usually have two or three children according to the operation *less than, greater than*, and potentially *equality* to a constant. Nominal attributes have usually as many children as possible values.

In a decision tree the missing values can be handled in a special way, if they were not treated as an attribute value. A possible, but simple solution, might be to record all branches at a node when testing a missing value and finally take the most commonly used branch. More sophisticated approaches split the instance and assign a weight to all parts, which is proportional to the number of training instances visiting that branch. Finally, the decisions at the leaf nodes must be recombined using the weights.

The automatic learning of decision trees can be expressed recursively. The set is split into two subsets using an argument and in both subsets the process is continued in a recursive manner. This ends if the instances of the subset at a node have the same class, or, when splitting, add no value to the predictions. In general the argument that splits the instances "best" is taken at a node.

Different algorithms are used to measure the best split. For example the algorithm C4.5 [18] implemented in the Weka library [15] uses information gain. It is based on the concept of entropy from information theory. Entropy $H(\mathcal{I})$ is defined as

$$H(\mathcal{I}) = I_E(p_1, p_2, ..., p_n) = -\sum_{i=1}^{n} p_i \log_2 p_i \tag{2.1}$$

where p_i represents the percentage of each class present in the child node that results from a split in the tree with $\sum_{i=1}^{n} p_i = 1$. A perfect split will be when every split consists only of one class resulting in zero entropy. A poor split with 50 % of class labels in both splits will have an entropy of one. The information gain IG is the entropy of the parent $H(\mathcal{I})$ minus the weighted sum of all children entropies of attribute a (see Definition 2.1).

Definition 2.1 (Information Gain). *Let $\mathcal{I}$ denote a set of training instances, each of the form $(\boldsymbol{x}, y) = (x_1, x_2, x_3, \ldots, x_k, y)$ where $x_a \in vals(a)$ is the value of the ath attribute of example $\boldsymbol{x}$ and y is the corresponding class label. The information gain for an attribute a is defined in terms of entropy $H()$ as follows:*

$$IG(\mathcal{I}, a) = H(\mathcal{I}) - H(\mathcal{I}|a) \tag{2.2}$$

$$= H(\mathcal{I}) - \sum_{v \in vals(a)} \frac{|\{\boldsymbol{x} \in \mathcal{I} | x_a = v\}|}{|\mathcal{I}|} \cdot H\left(\{\boldsymbol{x} \in \mathcal{I} | x_a = v\}\right) \tag{2.3}$$

The C4.5 algorithm [18] uses information gain as measurement and the pseudocode of the decision tree construction algorithm is shown in Algorithm 2.2.

Another popular decision tree algorithm is the classification and regression tree (CART). The regression concept is similar, but CART uses the gini impurity I_G instead of the information gain IG. The gini impurity measures how often a randomly chosen instance from the set would be incorrectly labeled if it was randomly labeled according to the distribution of classes J in the subset. So we define the probability f_i to label an instance correctly with label $i \in J$ in the subset and the probability $1 - f_i$ to label the same instance incorrectly. Both probabilities are multiplied and summed up over all labels in the (sub)set.

$$I_G(f) = \sum_{i=1}^{J} f_i \cdot (1 - f_i) \tag{2.4}$$

It reaches its minimum (zero) when all instances in the corresponding subset of a node fall into a single target category.

Algorithm 2.2: C4.5

Input : $\mathcal{I} \leftarrow$ All training instances.
Output : A decision tree classifier C created in a recursive manner.

$C \leftarrow$ *C4.5*$(\mathcal{I})$
return C

Function C4.5$(\mathcal{I})$
    $C \leftarrow \{\}$
    `// Leaf if no tree can be constructed.`
    if $y_i = y_j \forall$ *classes* $y_i, y_j \in \mathcal{I}$ **then** `// pure class`
        return *leaf with class* y_i
    else if $\mathcal{I} = \{\}$ **then** `// Class must be derived other than` $\mathcal{I}$
        return *leaf e.g. with majority class*
    end
    `// Find out the best attribute for a split.`
    $info \leftarrow [\,]$ `// Store information gain of each attribute.`
    forall $a_i \in$ attributes$(\mathcal{I})$ **do** `//` *attributes*$(\mathcal{I}) = \Theta$
        $info[i] \leftarrow IG(\mathcal{I}, a_i)$
    end
    $a_{best} \leftarrow$ *index*$(info,$ *max*$(info))$
    `// Constructing the (sub)tree.`
    $C \leftarrow$ Decision node for testing a_{best}
    $\hat{\mathcal{I}} \leftarrow$ *split*$(\mathcal{I}, a_{best})$
    forall $\hat{\mathcal{I}}_i \in \hat{\mathcal{I}}$ **do**
        $C_i \leftarrow$ *C4.5*$(\hat{\mathcal{I}}_i)$
        $C \leftarrow$ *append*(C, C_i)
    end
    return C

Random Forests The algorithm of RFs was introduced by Breiman [17]. A RF is an ensemble of k tree predictors and for the ith tree a random vector Θ_i is generated of individual sampled attributes, independent of the past random vectors $\Theta_1, \ldots, \Theta_{i-1}$, but with the same distribution. This type of decision trees are called a random trees. The reason for this ensemble approach is that if one or a few features are very strong predictors for target class, the different trees can become correlated and the variance is similar, which is not the goal of randomization. Finally, we can define the ith classifier as $h(\mathbf{x}, \Theta_i)$. In addition the k trees are generated using bagging, enabling the out-of-the-bag error method for RFs.

Definition 2.2 (Random Forest). *A Random Forest (RF) is an ensemble classifier consisting of an ensemble of k tree-structured classifiers $\{h(\mathbf{x}, \Theta_i) \,|i = 1, \ldots k\}$ where the Θ_i are independent identically distributed random vectors and each tree casts a unit vote for the most popular class at input* $\mathbf{x}$.

A final RF classifier uses each tree $h(\mathbf{x}, \Theta_i)$ to vote on the final class y. A typical number of random features used as Θ_i are $\left\lfloor \sqrt{|\Theta|} \right\rfloor$ or $\lfloor log_2 (|\Theta|) + 1 \rfloor$.

2.2.4 Classification on Imbalanced Datasets

The class imbalance problem means that we have, on the one hand, a class represented by a large number of examples. On the other hand a second class only represented by a few examples. This imbalance can cause a significant decrement in the performance achievable by standard learning methods which assume a balanced distribution of classes. Therefore these algorithms fail to properly represent the distributive characteristics of the data when presented with complex imbalanced datasets and as a result provide unfavorable accuracies across the classes of the data.

To handle the imbalance problem there are several approaches for training complex data with standard learning methods. In the last years three main proposals have emerged: (1) data sampling, (2) algorithm modification, and (3) cost-sensitive learning. The idea in data sampling is to modify the training data to generate more balanced data. Here it is possible to oversample the minority class to match the other class, e.g. by randomly oversampling the minority class without replacement [19]. Or the majority class can be downsampled to the same size of the minority class, e.g. by randomly eliminating instances of the majority class until it has the same size as the minority [20]. We speak of an algorithm modification if the classifier algorithm itself adapts for imbalanced data and the initial training set is not modified.

Finally, cost-sensitive learning combines the data structure with algorithm modifications. The general idea is to penalize the mistakes made by classification of minority classes to a greater extent than those of the majority classes. E.g. some learners can use weights of instances that portray the importance of these instances. Instances with lower weights have a smaller influence in training than instances with larger weights. Using this idea, every instance of the minority class can get a weight of one. The majority class will get the ratio that is needed for downsampling it to the same size as weight.

In this work a specific method was used to oversample the minority class. The algorithm will now be explained in detail.

Synthetic Minority Over-Sampling Technique Synthetic minority oversampling technique (SMOTE) is a technique to increase the minority class by generating new synthetic instances during the test phase [21]. Therefore for every minority instance $\mathbf{x}_i^{min}$ in the training set the k-neighbor instances $\hat{\mathcal{I}}^{min}$ (all minority classes) are defined by k-mer clustering ($k \in \mathbb{N}$) and a new instance $\mathbf{x}_{S_i}^{min}$ is generated by randomly swapping attributes between the actual instance $\mathbf{x}_i^{min}$ and one random other instance $\mathbf{x}_j^{min} \in \hat{\mathcal{I}}^{min}$ of the k-neighbors. This can be repeated until we reach an oversampling factor o so that the final number of minority instances are $|\mathcal{I}^{min}| + |\mathcal{I}_S^{min}| = |\mathcal{I}^{min}| + o \cdot |\mathcal{I}^{min}|$. Algorithm 2.3 shows the pseudocode of SMOTE.

Algorithm 2.3: Synthetic minority oversampling technique (SMOTE)

Input : $\mathcal{I}^{min} \leftarrow$ Minority class instances.
$N \leftarrow$ Amount of SMOTE in % or oversampling factor $o \times 100$.
$k \leftarrow$ Number of nearest neighbors

Output : $\left(\frac{N}{100}\right) \cdot T$ synthetic minority class samples $(Synthetic[\,][\,])$.

$T \leftarrow |\mathcal{I}^{min}|$ `// Number of minority class samples.`

`// If` N `is less than 100%, randomize the minority class samples as only a random percent of them will be SMOTEd.`

if $N < 100$ **then**
  Randomize$(\mathcal{I}^{min})$
  $T \leftarrow \left(\frac{N}{100}\right) \cdot T$
  $N \leftarrow 100$
end

`// Set necessary variables`

$N \leftarrow \left\lfloor \frac{N}{100} \right\rfloor$ `// The amount of SMOTE is assumed to be in integral multiples of 100.`

$numattrs[\,] \leftarrow$ Number of attributes

$idx\_synthetic \leftarrow 0$ `// Keep a count of number of synthetic samples generated.`

$\mathcal{I}_S^{min}[\,][\,]$ `// Array for synthetic samples`

`// Compute` k `nearest neighbors for each minority class sample only.`

for $i \leftarrow 1$ **to** T **do**
  $\hat{\mathcal{I}}^{min}[\,][\,] \leftarrow$ Compute k nearest neighbors for $\mathbf{x}_i^{min}$, and get the indices
  Populate$(N, \mathcal{I}^{min}, i, \mathcal{I}_S^{min}, idx\_synthetic, \hat{\mathcal{I}}^{min})$
end

`// Function to generate the synthetic samples.`

Function Populate$(N, \mathcal{I}, i, \mathcal{I}_S, idx, \hat{\mathcal{I}})$
  while $N \neq 0$ **do**
    $nn \leftarrow$ Choose a random number between 1 and k `// Select one of the nearest neighbors of` $\mathbf{x}_i^{min}$.
    for $attr \leftarrow 1$ **to** $numattrs$ **do**
      $diff \leftarrow \hat{\mathcal{I}}[nn][attr] - \mathcal{I}[i][attr]$
      $gap \leftarrow$ Random number between 0 and 1
      $\mathcal{I}_S[idx][attr] \leftarrow \mathcal{I}[i][attr] + gap \cdot diff$
    end
    $idx \leftarrow idx + 1$
    $N \leftarrow N - 1$
  end
  return

2.2.5 Performance Measurement

In most classification problems (supervised and sometimes unsupervised) we build the learner to classify new instances that we never have seen before. So the question to every trained classifier is always, how good it will predict totally new observations. To get an insight about the performance several testing strategies have been developed and different measurements can be used to rate the quality. Normally these methods are conservative, meaning that we can expect the same or an even higher performance on new data. But on the counterpart a classifier can also be overfitted, meaning that it represents the trained data nearly perfectly but new instances are a closed book. Now I explain briefly testing strategies and performance measurements used in this work.

Training and Testing The performance testing of a classifier has to be separated from the training phase. Training instances should not be used during testing. Otherwise a bias is introduced and usually the performance on new observations is worse. Therefore different strategies can be used for training and testing. The optimal approach depends on the dataset to be analyzed. Some of them are computationally expensive and some of them do not work well on small data.

A standard approach will be to split up the data into a training and a test set, like 60 % of all instances for training and the remaining 40 % for testing. The data should be divided randomly. Sometimes also stratified splits are made to ensure the same proportion between classes in the training and test set.

Another well-known approach is cross-validation (CV). With *CV* it is possible to test on all data but the test set is always disjoint from the training set. In *leave-one-out cross-validation (LOOCV)* for every instance $\mathbf{x}_i$ all other instances $\mathcal{I}/\mathbf{x}_i$ are used for training the classifier and it will be tested on $\mathbf{x}_i$. If more than one instance is held out it is called a *k-fold cross-validation (k-fold CV)*, where we partitioned all instances $\mathcal{I}$ into $k \in \mathbb{N}$ groups, mostly of equal size, so that $\mathcal{I} = \mathcal{I}_1 \triangle \mathcal{I}_2 \triangle \ldots \triangle \mathcal{I}_k$. Then training and testing will take place k-times. For the nth round the classifier will be trained on $\mathcal{I}/\mathcal{I}_n$ and then tested on $\mathcal{I}_n$. Classical k-fold training strategies are 5 and 10-fold CVs.

Receiver Operating Characteristic Receiver operating characteristic (ROC) curves were adopted from signal detection where the tradeoff was characterized between a hit rate and a false-alarm rate over a noisy channel. ROC curves are widely used in ML to show the performance of a classifier without regard to class distribution or error costs. On the vertical axis the ROC curve plots the true positive rate (TP rate) (also recall or sensitivity) and on the horizontal axis the false positive rate (FP rate). The TP rate is expressed as a percentage of the true positives (TP) included in the sample compared to all positives.

$$TPRate = 100 \cdot \frac{TP}{TP + FN} \tag{2.5}$$

The FP rate is the number of true negatives (TN) in the sample, expressed as a percentage of the total number of negatives.

$$FPRate = 100 \cdot \frac{FP}{FP + TN} \tag{2.6}$$

If the ROC curve lies close to the diagonal we have a similar TP rate and FP rate rate which can be interpreted as a random prediction. Ideally the curve climbs vertically so that the TP rate is close to 100 % and only after that the FP rate rises. ROC curves can be summarized in a single quantity using the area under the curve (AUC). If a classifier has an area under the receiver operating characteristic curve (AUROC) close to 0.5 it performs like random. An AUROC of 1.0 will be a perfect classification.

In imbalanced datasets ROC curves can be disadvantageous because they are independent of the class distribution. Therefore precision-recall (PR) is a better performance measurement and it can be shown that a good PR curve also has a good ROC curve [22–24].

Precision and Recall PR curves show the precision on the vertical axis, also called positive predictive value (PPV), and on the horizontal axis the recall (also sensitivity or TP rate). The area under the precision recall curve (AUPRC) is the summary statistic of PR curves and can range from zero to one where also larger is better. But the PR curve or its AUPRC value is closely related to the class distribution. For example if we have high precision on the first 20 %, but after that the curve falls off steeply, the classifier performs well when it reports high scores or probabilities.

$$precision = 100 \cdot \frac{TP}{TP + FP} \tag{2.7}$$

F-Score The F-score (also F_1-measure or F-measure) can be used for information retrieval. It can be interpreted as a weighted average of the PR curve and is mostly used as the harmonic mean of precision and recall.

$$F_1 = \frac{2 \cdot recall \cdot precision}{recall + precision} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN} \tag{2.8}$$

If the focus is more on the recall or precision the F-score can be generalized to

$$F_\alpha = (1 + \alpha^2) \cdot \frac{recall \cdot precision}{\alpha^2 \cdot precision + recall} = \frac{(1 + \alpha^2) \cdot TP}{(1 + \alpha^2) \cdot TP + \alpha^2 \cdot FN + FP} \tag{2.9}$$

For example for $\alpha = 2$ the recall is weighted 4 times more than the precision.

2.3 Ontologies

An ontology is a standardized way for knowledge representation, or in other words, an explicit specification of a conceptualization as defined by Gruber *et al.* [25]. So an ontology contains the relevant concepts, also called objects or *term*s, which are named and defined entities of the domain of interest. To specify a conceptualization we need to define statements, so called *axiom*s, that say what is true in the domain and give us a possible interpretation of the defined terms. Some of those axioms are relationships that can be defined in a computer-interpretable way. Two important relationships between terms are `is_a` and `part_of`.

Different types of ontologies exist. Ontologies that provide a controlled vocabulary for objects in a domain are called *domain ontologies*. For example the word "card" can be in the domain computer hardware a "graphic card". In the domain of party games a card can be a "playing card" of a card game. Another type of ontologies will be upper or *top-level ontologies*. They consist of very general terms that are common across all domains.

2.3.1 Semantic Similarity

A specific type of domain ontologies are *attribute ontologies*. These ontologies link terms to other objects. For example the Gene Ontology (GO) has an annotation relationship between terms and genes or proteins [26]. Thus, GO uses the annotations to assign biological functions, characteristics or attributes to the genes. Using attribute ontologies we can find annotated objects, like genes, that are similar to a set of terms using semantic similarity. In contrast to that, a domain ontology provides us only a controlled vocabulary and we are limited to search terms within the ontology using parent- or subclasses.

Information Content Resnik [27] defines the information content (IC) of a term t as the negative log likelihood,

$$IC\left(t\right) = -\log\left(p\left(t\right)\right), \tag{2.10}$$

where $p(t)$ is the probability of encountering an instance of term t in an object, like a gene. Intuitively this means if a term has a higher probability, for example many genes are annotated with it, the informativeness decreases. The root term r will have an IC of zero, $IC\left(p\left(r\right)\right) = IC(1) = 0$, because it will be shared by all objects. This intuitive behavior is also implied by the annotation propagation rule. So $p(t)$ is a monotonically increasing function when following the links from the leaves to the root of an ontology. Generally speaking, when terms t_1 `is_a` t_2, then $p(t_1) \leq p(t_2)$ and $IC\left(p\left(t_1\right)\right) \geq IC\left(p\left(t_2\right)\right)$.

Using the IC, Resnik [27] showed how to define a similarity between two terms. This is defined as the IC of the CA from both terms. Because an ontology is a DAG there might be multiple CAs. The most specific one is called the *most informative common ancestor*. So the similarity between two terms is

$$sim(t_1, t_2) = \max\left\{IC(a) | a \in CA\left(t_1, t_2\right)\right\}. \tag{2.11}$$

In attribute ontologies we might want to find similarities between objects, like how similar two genes are given an ontology. This results in a comparison between sets of terms. This is done using the previously defined term similarity and some modifications. For example we can simply do a weighted average of all comparisons using

$$sim(T_i, T_j) = \frac{1}{|T_i| \cdot |T_j|} \cdot \sum_{t_k \in T_i, t_l \in T_j} sim(t_k, t_l), \tag{2.12}$$

where $T = \{t_1, \ldots, t_n\}$ is a set of terms, maybe derived from an object [28].

Sometimes also the best-matched average (BMA) is used [29]. It can be asymmetric like

$$BMA_{asym}(T_i, T_j) = \frac{1}{|T_i|} \cdot \sum_{t_k \in T_i} \max_{t_l \in T_j} \{sim(t_k, t_l)\}, \tag{2.13}$$

or symmetric

$$BMA_{sym}(T_i, T_j) = \frac{BMA_{asym}(T_i, T_j)}{2} + \frac{BMA_{asym}(T_j, T_i)}{2}. \tag{2.14}$$

2.4 Genome-wide Pathogenicity Scores

To find out the pathogenicity or the functional relevance of SNVs or InDels several scores have been developed. They predict the impact of a variant to the designed purpose, e.g. how intolerant a mutation in a gene is in terms of the resulting change of the protein function. A few dozen such scores exist and in general they can assess the impact of a SNV on a protein in the coding regions.

One of the first scores was Sorting Intolerant From Tolerant (SIFT) [6] which scores variants according to conservation of the AA sequence in related proteins. SIFT calculates the probability that an AA at a position is tolerated conditional on the most frequent amino acid being tolerated. It normalizes the value and if it is less than a cutoff the variant is predicted to be deleterious. Other scores followed, like PolyPhen or MutationTaster [7–9].

All of these scores have been successfully used in Exome or other genetic studies [30–32]. But if we expand the target region to the full genome they are not able to score non-coding variants (NCVs). The two first pathogenicity scores for coding and non-coding variations are CADD form Kircher *et al.* [33] and GWAVA published by Ritchie *et al.* [34]. Several other methods were published afterwards, like the recent score LINSIGHT [35]. The common ground of the methods is that all of them use supervised machine-learning methods. Except the scores Eigen and Eigen-PC [36] where an unsupervised approach was applied.

The purpose and design of all non-coding or genome-wide scores differs slightly. We can roughly divide them into four groups: (1) ML classifiers that attempt to separate known disease variants from putatively benign variants using a variety of genomic features (for example GWAVA and FATHMM-MKL [34, 37]); (2) sequence- and motif-based predictors for the impact of NCVs on cell-type-specific molecular phenotypes, such as chromatin accessibility or histone modifications (for example DeepSEA [38]); (3) evolutionary methods that consider data on genetic variation together with functional genomic data with the aim of predicting the effects of NCVs on fitness (for example CADD and LINSIGHT [33, 35]); (4) unsupervised ML approaches based on correlation of features (Eigen and Eigen-PC [36]).

In this work several non-coding scores were used to compare, on the one hand, the base learners of these scores to a new ML approach (Chapter 3) and, on the other hand, the generated scores to a new developed non-coding score for regulatory Mendelian mutations (Chapter 4). Thus, the performance and learning strategies of the genome-wide scores CADD, GWAVA, DeepSEA, Eigen and Eigen-PC, FATHMM-MKL, and LINSIGHT are described in more detail in the next Sections 2.4.1 to 2.4.6.

2.4.1 CADD

The Combined Annotation–Dependent Depletion (CADD) score from Kircher *et al.* [33] is able to prioritize functional, deleterious and pathogenic variants across many functional categories. Thus it integrates multiple genome features per possible nucleotide change. CADD is specifically designed to score any possible human SNV or small InDel (limited to human reference genome hg19 release).

CADD uses an evolutionary model for training sets selection to ensure that the method is not limited to a small set of genetically or experimentally well-characterized mutations. So Kircher *et al.* [33] state that CADD measures the likelihood of deleteriousness and not the likelihood of pathogenicity, because no known pathogenic mutations from databases and also no known benign variants were used for training. But the property of deleteriousness should correlate with both molecular functionality and pathogenicity [39], so it can be used to discover pathogenic mutations.

Because of this thought Kircher *et al.* [33] used as non-deleterious set, positions that differ between the human genome and the inferred human-chimpanzee ancestral genome [40] at which humans carry a derived allele with an AF of at least 95 % compared to the 1000 Genomes Project (1KG) [41]. The resulting negative dataset contains 14.9 million SNVs and 1.7 million InDels. The positive variant set was simulated using an empirical model of sequence evaluation with CpG-specific mutation rates and simulation parameters from whole genome alignments of six primate species of the Enredo-Pecan-Ortheus (EPO) study [40, 42]. The final genome-wide substitution rate matrix contains local mutation rate estimates in blocks of 100 Kb together with the frequency and length distribution of insertion and deletion events. This matrix was used to simulate over 44 millions SNVs and over 5 million InDels on regions where EPO alignments were present. Kircher *et al.* [33] claimed that this strategy ensures the prediction of deleteriousness and not pathogenicity.

CADD relies mostly on features from annotations of the Variant Effect Predictor (VEP) [43], conservation scores and different chromatin features from Encyclopedia of DNA Elements (ENCODE). In total a vector of 63 features per variant was constructed. These features, together with the positive and negative set, were used to train 10 different models by a Support Vector Machine (SVM) with a linear kernel. In newer versions of CADD the SVM learner was replaced by a logistic regression. All 10 datasets were generated by randomly sampling 13,141,299 SNVs, 627,071 insertions and 926,968 deletions from the positive and the negative set, which results in 10 perfect balanced datasets. Train and test strategy was a simple split using 99 % for training and 1 % for testing. Kircher *et al.* [33] used a low generalization parameter $c = 0.0025$ to converge the SVM in a reasonable amount of time. Finally the average model was used as a final model and they computed the combined SVM scores (C-scores) of all possible substitutions in the human reference genome. In addition they introduced with the scaled C-score, a Phred-like metric ranging from 1 to 99 on the basis of their ranking relative to all possible substitutions in the human reference genome:

$$-10 \log_{10} \left(\frac{\text{rank}}{\text{total number of substitutions}} \right). \tag{2.15}$$

To assess the functionality of the score to different diseases, Kircher *et al.* [33] tested them to previously known mutations in Mendelian diseases, Genome Wide Association Studies (GWAS) hits or somatic variants. For example they used pathogenic and non-pathogenic variants of the ClinVar database [44, 45] and achieved an AUROC of 0.9164.

In sum, CADD is a deleteriousness rather than a pathogenicity score. This means that CADD do not use known pathogenic variants linked to a specific disease as well as known benign variants for training. The authors had a unique way to generate their training data and their results showed that training on simulated de-novo mutations portrays the pathogenicity of variations in different types of diseases. But because these variants are not exactly known to be pathogenic and cause a genetic disease they named them deleterious and non-deleterious. With these data they were able to build a general score that is able to score variants of the whole genome in an unbiased way, totally independent of databases, and consequently CADD can be used in different types of variant scoring approaches.

2.4.2 GWAVA

Genome-wide annotation of variants (GWAVA) was developed by Ritchie *et al.* [34] to interpret variants outside the genic regions. They integrated various genomic and epigenomic features and trained a modified RF. To be more precise Ritchie *et al.* [34] had the following feature groups: Open chromatin like DNase I hypersensitivity, transcription factor binding sites (TFBSs) and histone modifications as well as RNA polymerase binding from ChIP-seq experiments, CpG islands, Genome segmentation like calls from ChromHMM [46], conservation score like GERP [10, 11], human variation frequencies, distance to the nearest splice-site and other genetic contexts, and finally sequence context information such as the G/C content.

The training set contains as positive class 1614 SNVs, tagged as "regulatory mutation" in Human Gene Mutation Database (HGMD) [47]. As negative or control set Ritchie *et al.* [34] used 15,730,276 common variations (minor allele frequency (MAF) $\geq$1 %) in the 1KG [41]. With their AF cutoff they wanted to reduce the chance of including rare functional variants in their control set. They generated three subsets of their negative data. First, they sampled 100 times the size of the positive set (dataset D_1). Second, they computed the distance to the nearest transcription start site (TSS) and used all close 1KG SNVs until they reached 10 times the size of the positive set (dataset D_2). And finally, they used a selection window of 1 Kb around the positives, whereby 5027 negatives remained (dataset D_3).

For classification Ritchie *et al.* [34] modified a RF to deal with the imbalances in their datasets. The dataset D_1 is balanced but in D_2 the positive/negative ration is 1:10 and 1:3 for D_3. Thus, they constructed the RF so that the majority class is subsampled to the same size as the minority class before building every random tree in the forest. Consequently, if the forest size is large enough, the classifier used at least every training instance once. For all three datasets a forest size of $t = 100$ was used.

As performance measurement a 10-fold CV approach was applied and the AUROC measured. The AUROCs of this experiment were $\text{AUROC}_{D_1} = 0.97$, $\text{AUROC}_{D_2} = 0.88$ and $\text{AUROC}_{D_3} = 0.71$. In addition a gene-aware CV approach was tested, so that variants from the same gene appearing in the training and test sets do not inflate performance statistics. Therefore they randomly selected only one variation per gene of the positive set. This reduced the performance in terms of AUROCs slightly ($\text{AUROC}_{D_1} = 0.95$, $\text{AUROC}_{D_2} = 0.82$, and $\text{AUROC}_{D_3} = 0.64$).

They validated their final trained classifier on pathogenic non-coding mutations from ClinVar [44, 45], GWAS SNPs from GWAS catalog [48], chromosome 22 of the individual `NA06984` from the 1KG project (negatives) including regulatory HGMD variants of that chromosome as positives and finally, with non-coding somatic mutations from Catalogue of Somatic Mutations in Cancer (COSMIC) [49]. For all four datasets Ritchie *et al.* [34] did a retraining on the new data using CV for performance tests. For negatives the GWAVA authors used the previously described negative set from the 1KG data with close variants to TSS in the ClinVar analysis and excluded chromosome 22 in the genome individual analysis. For the GWAS data they generated a matched SNV control set from common GWAS genotyping arrays, and in the somatic analysis recurrent variations were used (occuring in multiple studies).

The gini importance was measured to show how important the features are. The gini importance is derived from the gini impurity (see Section 2.2.3) and is the mean decrease in impurity at each node in the tree owing to the feature of interest, weighted by the proportion of samples reaching that node. This analysis was done only on their initial setup but it shows that depending on the negatives the best features are conservation scores, distances to known TSS and the H3K4 trimethylation level.

To conclude Ritchie *et al.* [34] built an imbalance-aware classifier based on a RF that is able to predict non-coding regulatory variations of different categories like somatic, common or Mendelian variants. In all analysis the same features set was used for classification.

2.4.3 DeepSEA

The approach of the deep learning-based sequence analyzer (DeepSEA) is to learn the regulatory code of the human genome from large-scale chromatin-profiling data to prediction chromatin effects of sequence alterations on a single-nucleotide level [38]. It consists of two parts. First, allele-specific chromatin profiles were predicted though a deep convolutional neural network (CNN). And second, the chromatin profiles were combined with conservation scores and a boosted logistic regression classifier was trained. Compared to other approaches the focus of DeepSEA is more on the generation of features than on the training of NCVs.

For training their CNN they split the human genome into 200 bp bins. Then they used 919 chromatin features (125 DNase features, 690 transcription factor features from ChIP-seq, 104 histone features) and labeled the feature with 1 if more than half of the 200 bp bin is in the peak region. 0 otherwise. Then 1000 bp sequence fragments were used to train the network. Fragments were centered around a 200 bp bin so that there were flanking regions of 400 bp on each side. The nucleotides of the region were used in the feature matrix as columns so that finally a three dimensional matrix of size $1000 \times 4 \times 919$ was constructed as input.

DeepSEA used three convolutional layers with 320, 480 and the last layer with 960 kernels. Finally, they generated the fully connected layer, and the sigmoid output layer makes the predictions (range 0 to 1) for the 919 chromatin features. To evaluate their performance, chromosome 7 and 8 were excluded from training and the CNN was trained on the other autosomes. Finally 4000 regions on chromosome 7 were used for testing and they achieved a median AUROCs of 0.958 for TFBSs, 0.923 for DNase and 0.856 for histone modifications.

For functional SNV priorisation the feature scores were computed using the sequence with and without the variant. Then the absolute difference between the probability values was used,

$$\texttt{diff} = \left| P\left(\text{reference}\right) - P\left(\text{alternative}\right) \right|, \tag{2.16}$$

as well as the relative log fold changes of odd,

$$\texttt{logfold} = \left| \log \frac{P\left(\text{reference}\right)}{1 - P\left(\text{reference}\right)} - \log \frac{P\left(\text{alternative}\right)}{1 - P\left(\text{alternative}\right)} \right|. \tag{2.17}$$

Both different feature sets show different feature spaces and those are visualized in Figure 2.1.

Finally, the conservation scores PhastCons [12] (excluding human), PhyloP [13] (excluding human), and GERP++ [10, 11] were added to the initial feature set resulting in $2 \times 919 + 4 = 1842$ features training a boosted logistic regression classifier. Zhou & Troyanskaya [38] used regulatory mutations from HGMD [47], expression quantitative trait loci (eQTLs) and GWAS hits as positive class for training. Negative SNP groups were generated using random subsets from 1KG [41] with a MAF greater than 0.5 % that are in non-coding regions close to the positives (within 1200 bp, 21,000 bp and 24,000 bp). Using their method and datasets, they claimed that DeepSEA outperform previous published methods (CADD [33], GWAVA [34] and Funseq2 [50]).

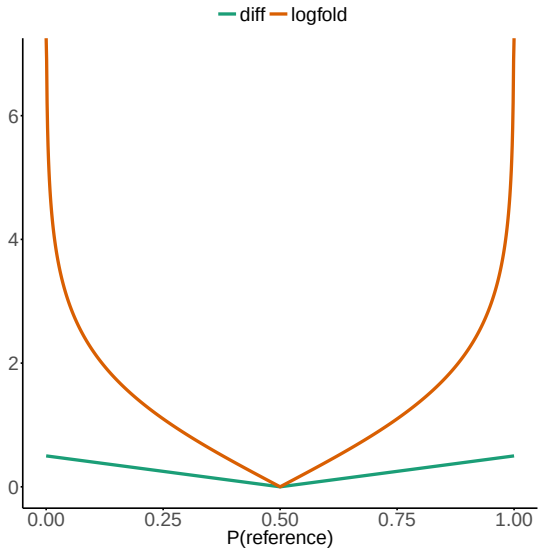


Figure 2.1: Feature space of DeepSEA. Feature set characteristics of DeepSEA. Visualization of the functions `diff` and `logfold` that were used to generate the two different feature spaces for DeepSEA. $P\left(\text{reference}\right)$ values between 0 and 1 are represented in abscissa. A fixed $P\left(\text{alternative}\right)$ of 0.5 is used.

2.4.4 Eigen

In 2016 Ionita-Laza *et al.* [36] published an unsupervised approach to integrate different annotations into one measure of functional importance (Eigen). They assume that the variants can be partitioned into functional and non-functional without any labeling. The idea is that there is a blockwise conditional independence between annotations given the true state of a variant, resulting in blocks or groups of annotations that differ in their mean between functional and non-functional variants. This correlation structure can be used to separate the two variant groups. In addition they introduced Eigen-PC, another meta score that is based on the eigendecomposition of annotation covariation matrix and uses the lead eigenvector to weight the individual annotations.

Ionita-Laza *et al.* [36] rely on similar features as other methods but they generated blocks (feature groups) such that predictors of different blocks were conditionally independent. Predictors within a block were allowed to be conditionally dependent. Forming of these blocks was done with a correlation matrix and they selected blocks by hand.

Two different input datasets were used to build the meta-scores. The *non-synonymous* dataset (missense and nonsense variants) integrates all non-synonymous variants that were present in the dbNSFP [51]. The other dataset for non-coding and synonymous variants was built with all variants in the 1KG [41] database without a match in dbNSFP and within 500 bp upstream of the gene start site (418,997 variants in total).

The authors identified three blocks for the NCVs. First, a conservation score block including GERP [10, 11], PhyloP [13] and PhastCons scores [12] (8 features in total). Second, a regulatory information block including open chromatin measures, TFBSs and histone modifications (17 features in total). And finally third, an AF block where they used AFs from different ethnicities of the 1KG [41] as features (4 features in total). For non-synonymous coding variants (missense and nonsense) the regulatory information block was replaced by protein function scores (SIFT, PolyPhen and Mutation Assessor [6, 7, 52]).

For validation of Eigen, Ionita-Laza *et al.* [36] used autosomal pathogenic and benign variant sets from ClinVar [44, 45]. Because of their two different functional scores these variants were subdivided into a non-synonymous coding set and a non-coding set, including synonymous variants. Eigen score achieved an AUROC of 0.868 (Eigen-PC $= 0.839$) on the missense and nonsense variants (16,545 non-synonymous pathogenic and 3482 benign variants; imbalance $\sim$1:5). CADD score version 1.0 achieved a similar AUROC of 0.861 but the AUROC of version 1.1 of CADD was only 0.776. For non-coding or synonymous pathogenic mutations they identified 111 pathogenic variants in ClinVar [44, 45]. As begin/negative set they randomly selected the same amount of variants that have the same set of functional classes (5'UTR, 3'UTR, etc.) determined by VEP [43]. Here, Eigen achieved an AUROC of 0.785 and Eigen-PC of 0.614. Again CADD has a similar performance of 0.777.

To strengthen the validation of non-codings, Ionita-Laza *et al.* [36] predicted the functional significance of GWAS hits from GWAS catalog [48]. GWAS hits that map to known regulatory elements achieved the highest Eigen scores. In addition 676 eQTL SNPs which map to known regulatory elements were tested and compared to CADD and GWAVA. A Wilcoxon rank-sum

test [53] was used to show that Eigen and Eigen-PC scores led to more significant results than CADD and GWAVA.

Finally, Eigen was used to compare recurrent and non-recurrent somatic non-coding mutations from COSMIC [49]. A Wilcoxon rank-sum showed that p-values from Eigen and Eigen-PC are orders of magnitude smaller than the scores of CADD. The method GWAVA that also claims to predict recurrent somatic mutations (see Section 2.4.2) was not used for comparison.

In sum, Ionita-Laza *et al.* [36] showed for the first time a non-coding score by unsupervised learning. Their meta-score depends only on the correlation of features of the two used training sets, non-synonymous variations from dbNSFP [51] and non-codings/synonymous variants from 1KG [41]. The features used were similar to those used in other methods, but Eigen is the only presented method here that uses AFs as predictive features. For Mendelian diseases AFs are mostly used for filtering rather than in a pathogenicity score because the prevalence of a disease is known [54–57].

2.4.5 FATHMM-MKL

FATHMM-MKL from Shihab *et al.* [37] is a ML approach that integrates functional annotations from ENCODE with nucleotide-based sequence conservation scores. This results in similar feature sets to CADD [33] or GWAVA [34]. The training set assembly was based on GWAVA because also HGMD variants [47] for positives and 1KG variants [41] with a MAF >1 % for negatives were used. The main difference is that all germline HGMD, not only regulatory mutations, were used in FATHMM-MKL. Positive and negative sets were further split according to whether or not the variant introduces an amino acid substitution. Only the non-coding set will be discussed here further.

Shihab *et al.* [37] used multiple kernel learning (MKL) to indicate weather a SNV is functional or not. Therefore they created 10 feature groups out of their features (coding and non-coding) and encoded them into kernel matrices. A final composite kernel matrix $\boldsymbol{K} = \sum_i^{10} = 1\lambda_i \boldsymbol{K_i}$ can be derived by each corresponding base kernel $\boldsymbol{K}_i$, and $\boldsymbol{K}$ was used to to train a SVM. For NCVs only a subset of four feature groups were used because they achieved the best results. These groups were conservation scores, histone modification based on ChIP-seq peak calls, TFBSs based on PeakSeq peak calls, and open chromatin based on DNase-seq peak calls. The other six groups were rejected.

The authors compared FATHMM-MKL to CADD [33] and GWAVA [34]. They modified their non-coding training set in several ways to avoid overfitting or reduced performance of a score because of incompleteness. First, they removed every variant where no C-Score was available in CADD. Second, the same negative selecting strategy as in GWAVA [34] was performed using window of 1 Kb around the positives (also explained in Section 2.4.2). And third, all training data included in the GWAVA classifier were removed so that there will be no bias by predicting variants that GWAVA is already trained on. In addition to the adaption of GWAVA and CADD every variant that did not have at least one value in all 10 feature groups was removed. These modifications reduced their training set from 6.7 million negative and 12,438 positive examples to 4591 negatives and 3032 positives which where used in a 5-fold CV.

FATHMM-MKL derived an AUROC of 0.91 closely followed by CADD with 0.87. GWAVA has only an AUROC of 0.69 on the non-coding dataset. The strongest feature group of FATHMM-MKL is the conservation group with an AUROC of 0.88. The AUROC of the other three groups is between 0.55 to 0.61. As additional performance test Shihab *et al.* [37] predicted a set of pathogenic and benign NCVs ($n = 647$) from ClinVar [44, 45] using their fully trained classifier on the previous data. With ClinVar, FATHMM-MKL results in an AUROC of 0.93, CADD of 0.89 and GWAVA 0.62.

To conclude the FATHMM-MKL score achieved slightly better performance than CADD on the tested NCVs in terms of the AUROC and it substantially outperforms GWAVA. The method used by FATHMM-MKL (MKL SVM) is similar to CADD (SVM with linear kernel) as well as the feature set. The major difference of FATHMM-MKL is the specific non-coding training set (HGMD and 1KG). CADD is more general in the training set by using evolutionary concepts (simulated mutations and derived ancestor alleles) and not specific curated dataset (e.g. HGMD). FATHMM-MKL might be slightly better in the actual specific tested case but it has to be shown whether it can also generalize as well as CADD on the whole genome.

2.4.6 LINSIGHT

Linear INSIGHT (LINSIGHT) uses evolutionary methods, similar to the idea of CADD, that consider data on genetic variation together with functional genomic data for predicting effects of NCVs on fitness [35]. In particular LINSIGHT combines a generalized linear model for functional genomic data with a probabilistic model of molecular evolution called Inference of Natural Selection from Interspersed Genomically coHerent elemenTs (INSIGHT) [58].

The idea behind INSIGHT is to estimate the probabilities that variants at each genomic site will have fitness consequences based on patterns of genetic polymorphism within a species (humans) and patterns of divergence from closely related outgroup species (apes: rhesus macaque, chimpanzee, and orangutan). Therefore INSIGHT compares the focal sites with those flanking neutral sites. To find out if these elements of interest are under selective pressure, INSIGHT predicts two hidden variables in their model, visualized in Figure 2.2: (1) the ancestral allele Z_i at a specific site i of closest outgroup species using a set of aligned bases O_i (here, apes) before it diverged to (2) the ancestral allele A_i from the humans at the site i before the polymorphism known in our population was introduced (given by the 1KG data [59]). Predicting these two variables will allow to measure the ratio of polymorphism rate γ at i to a local polymorphism rate. Parameter γ cannot be predicted by the population itself because we do not know if the major allele or the minor allele we see in humans at this site was the ancestral allele.

Related on the difference of Z_i and A_i the observed allele in the human population has three states. First, they are completely divergent and monomorphic in humans (D). Second, there is a low frequency-derived allele in humans based on a threshold (L). And third, using the frequency threshold there can be a high frequency-derived allele (H). The frequency threshold of LINSIGHT is set to $f = 0.15$.

With these thoughts we can classify non-coding sites into neutral drift, weak negative selection and strong negative selection. Variants at sites with strong negative selection are immediately

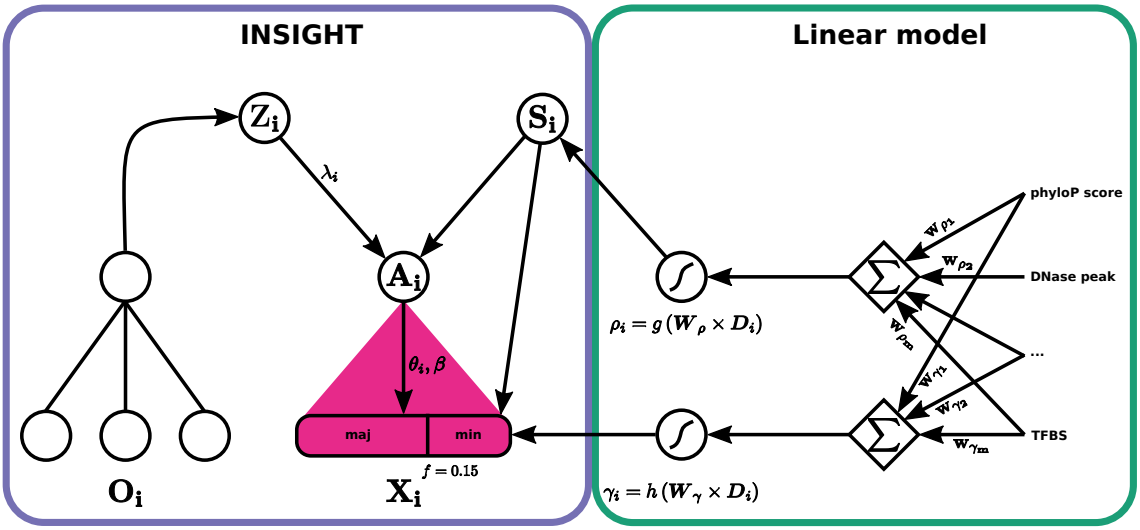


Figure 2.2: The LINSIGHT model. From the primate outgroup multi-alignment O_i at a focal site i INSIGHT predicts the ancestral allele hidden variable Z_i. Through a neutral substitution rate λ_i we came to the second hidden variable, the ancestral allele A_i of the human population. From A_i INSIGHT defines on the one hand the fraction of neutral polymorphisms $\beta = (\beta_1, \beta_2, \beta_3)$ with low- ($\beta_1 = (0, f)$), intermediate- ($\beta_2 = [f, 1-f]$), and high-frequency-derived ($\beta_2 = (1-f, 1)$) alleles. On the other hand there is the neutral polymorphism rate θ_i at site i. Then LINSIGHT combines this probabilistic graphical model with a generalized linear model. The parameters for the selection S_i from INSIGHT, ρ and γ, are defined by linear combinations of local genomic features, followed by sigmoid transformations.

removed from the population and cannot segregate or fix in human population. This will hold for the completely divergent sites D or conserved sites. Weak negatives sites can segregate under low frequency like the L sites. Mutation in neutral sites can segregate with low and high frequencies and can fix in the population, like H, L, D and conserved sites. In addition INSIGHT has the ability to model also positive selection. But in the model of LINSIGHT this was discarded because positive selection has a negligible importance when estimating genome-wide probabilities of fitness consequences [60].

The influence of negative selection at site i is summarized by ρ_i and the relative rate of low-frequency derived alleles γ_i. The parameter ρ_i is the probability that a site i is under negative selection (strong or weak) within functional elements. So this parameter gives the fitness consequences of a mutation at this site and, because of that, ρ_i is in the main focus of the LINSIGHT analysis. γ_i is used for fitting the model but ignored later. LINSIGHT then tries to predict selection parameters by a generalized linear regression model using local genomic features. It assumes that $\rho_i = g(\boldsymbol{W}_\rho \times \boldsymbol{D}_i)$ and $\gamma_i = h(\boldsymbol{W}_\gamma \times \boldsymbol{D}_i)$ were $\boldsymbol{W}_\rho$ and $\boldsymbol{W}_\gamma$ are row vectors of weights for the genomic features, $\boldsymbol{D}_i$ a column vector of genomic features, and $g()$ and $h()$ non-linear functions. Huang *et al.* [35] used two sigmoid functions, the Gomertz sigmoid function $g(x) = \exp(-3\exp(-x))$ [61] for $g()$ to avoid saturation at small values and the standard logistic function $h(x) = \frac{1}{1+e^x}$. Then LINSIGHT tries to estimate the parameters by maximum likelihood using online stochastic gradient-descent algorithm. Gradients of the feature weights are computed by backpropagation. Thus, the method is similar to a neural network without any hidden layer. Figure 2.2 shows an overview of the LINSIGHT model.

LINSIGHT uses 48 different features such as RNA expression levels, chromatin accessibility, histone modifications, bound transcription factors, conservation scores like PhyloP and phastCons [12, 13], distance to the nearest TSS or known TFBS motive. At the end, Huang *et al.* [35] precomputed their final derived fitness score for the whole genome (hg19 release). Then they extracted several median scores of different genomic areas to show that splice sites have the highest median scores (0.956) followed by annotated TFBSs with a median score of 0.240. Unannotated intronic and intergenic regions showed the lowest scores (0.044 to 0.048).

To compare LINSIGHT to other genome-wide deleteriousness scores (CADD, Eigen, DeepSEA, FunSeq2, GWAVA, and PhyloP [13, 33, 34, 36, 38, 50]), Huang *et al.* [35] used as positives non-coding nucleotide positions in HGMD [47] and ClinVar [44, 45] and common polymorphisms (MAF $>1\,\%$) as negatives. Three different test sets were defined from this data (all balanced). (1) all positives and a random sample of negatives; (2) negative examples matched by distance to the nearest TSS (balanced); (3) negative examples matched by specific genomic regions ("promoter" regions upstream 1 Kb of annotated TSS from all genes, "splice" regions within 20 bp of any annotated splice site, UTR regions, and all "other" regions).

Overall, Huang *et al.* [35] showed that LINSIGHT outperforms the other tested methods on all three datasets. LINSIGHT achieves an AUROC of 0.897 for the unmatched, 0.759 for the matched TSS and 0.661 for the matched region test set. All other methods have similar or decreased AUROCs, but they are always lower than LINSIGHT. So Huang *et al.* [35] concluded that their approach on characterizing NCVs based on natural selection in the past provides useful information about phenotypic importance in the present.

2.5 Chapter Conclusion

This preliminary chapter described important formal definitions and mathematical concepts on which the later chapters will be based on. I also introduced the main ML concepts, like RF, bagging, ensemble methods, and training/testing strategies which will be important for my work. Finally, different non-coding pathogenicity scores were introduced because these scores and their base learners will be used for benchmarking my work.

Summary of genome-wide Pathogenicity Scores I take advantage of this concluding section to summarize the introduced genome-wide pathogenicity scores. It is difficult to tell which score suits for which task. The original publications are difficult to use for performance comparison because different datasets or different dataset releases, as well as different versions of the scores, were used. Depending on the goal, further evaluations may be required before selecting and using a pathogenicity score. It might also be possible to use a combination of scores, which is often the case in non-synonymous variant predictions for Mendelian disease [54, 55].

Table 2.1 gives an overview of all introduced scores. The table summarizes the different learners, training data and features for every scores. This might help to select an appropriate score for a pathogenicity prediction task, especially the training data and the selected features will help in that case.

In theory, the evolutionary scores and the unsupervised Eigen approaches should be generalized best over different types of variants because they do not rely on a specific training class, like regulatory HGMD variants, and avoid the observation bias that exists in the databases. One example of observation bias is that the database ClinVar [44, 45] has a high fraction of BRCA2 and BRCA1 mutations [62]. When used for training or testing it might overfit or overestimate these two genes and have a reduced performance on other genes. But even if scores generalize well there will be specific prediction problems where these scores will have a poor performance. For example Mather *et al.* [63] showed that the CADD score has limited clinical validity for pathogenic germline variants in non-coding regions in cancer.

Maybe the best option to compare all scores is to look into scientific publications that compare scores with a standardized dataset. For example by using the work of Dong *et al.* [64] and Liu *et al.* [65]. But I have to notice that these publications might have an observation bias because variants in the testing set might be included in the training set of some pathogenicity scores. This leads to an overestimated performance of these scores. In theory that can be avoided, if pathogenicity scores replaced the outcome value of training variants by a score from an internal CV. But I am not aware that any of the scores have such a strategy implemented. In addition, when it comes to non-coding scores, we have the problem of missing data, and with the actual databases we are extremely limited in creating a comprehensive dataset for pathogenic NCVs.

In the following chapters I will describe my contribution to this field. The next Chapter 3 presents *hyperSMURF*, a ML algorithm that is specifically designed for highly imbalanced (genetic) data, as well as its experimental evaluation. Chapter 4 describes how I use *hyperSMURF* to generate a new pathogenicity score and how this score is integrated in a framework to discover the non-coding part of the genome. Then, the pathogenicity score as well as the framework are experimentally validated and compared to other scores and software. Finally, Chapter 5 concludes this thesis.

Table 2.1: Summary of genome-wide pathogenicity scores. Grouped by their classification task. Different feature groups were encoded by (A) sequence conservation, (B) ChIP-seq/Open chromatin, (C) population frequencies, (D) Sequence and genetic context. The asterisk (∗) at the CADD classifier denotes that the initial version of CADD 1.0 was built with an SVM. Version 1.1 and newer used a logistic regression.

Classification task					
Known disease from putatively benign variants					
Name	**Classifier**	**Positives**	**Negatives**	**Features**	**Literature**
GWAVA	Modified RF	Regulatory mutations in HGMD	Common 1KG variants (1 %)	A; B; C; D; ChromHMM	Ritchie *et al.* [34]
FATHMM-MKL	SVM with MKL	Germline HGMD variants	Common 1KG variants (1 %)	A; B	Shihab *et al.* [37]
DeepSEA	Logistic regression	HGMD variants, eQTL and GWAS hits	Common 1KG variants (0.5 %)	A; B; Extraction on sequence- and motif-based cell-type-specific molecular phenotypes through CNN	Zhou & Troyanskaya [38]
Predicting the effects on fitness derived from evolution					
Name	**Classifier**	**Positives**	**Negatives**	**Features**	**Literature**
CADD	SVM/Logistic regression*	Simulated variants	Human derived variants	A; B; D	Kircher *et al.* [33]
LINSIGHT	Logistic regression	Fit of INSIGHT model		A; B; D; RNA expression	Huang *et al.* [35] and Gronau *et al.* [58]
Unsupervised approach					
Name	**Classifier**	**Instances**		**Features**	**Literature**
Eigen	Correlation of features using predefined feature groups	1KG variants within 500 bp upstream of the gene start site		A; B; C	Ionita-Laza *et al.* [36]
Eigen-PC	Correlation of features based on the eigendecomposition of annotation covariation matrix	1KG variants within 500 bp upstream of the gene start site		A; B; C	Ionita-Laza *et al.* [36]

Chapter 3

Imbalanced Training Sets

I developed the ensemble machine learning method hyperSMURF that can deal with extremely imbalanced datasets in collaboration with Giorgio Valentini and Peter N. Robinson. Giorgio Valentini implemented the R version of hyperSMURF. Matteo Re did the feature extraction for GWAS data with DeepSEA as well as the training for it. I presented the hyperSMURF algorithm at the German Conference on Bioinformatics (GCB) 2016 and at ISMB/ECCB 2017. The hyperSMURF algorithm was published in Scientific Reports [66].

M. Schubach, M. Re, P. N. Robinson, and G. Valentini. (2017). Imbalance-Aware Machine Learning for Predicting Rare and Common Disease-Associated Non-Coding Variants. *Scientific Reports*, 7(1), 2959.

ML is a powerful method to explore biological processes with only little knowledge. Current scientific knowledge can be exploited to train a classifier to predict characteristics of new data. This provides insight into the biological mechanisms involved. To increase performance of the ML algorithm it is advantageous that the data for training is reliable. But nature teaches us that it is hard to find good and reliable data because validations of hypotheses can be laborious, costly or unfeasible with the current state of science. Therefore, ML may be applied in situations where there are only a few reliable candidates that achieve a certain expected behavior but there is a magnitude of observations with the opposite or neutral behavior. This leads to an imbalance between positive and negative observation. Such training sets for a ML classifier are suboptimal because classifiers tend to focus on the majority class and therefore may neglect the minority class.

Since WGS has become increasingly affordable, the number of genome projects with thousands of sequenced individuals has increased over the last years [49, 59, 67–69]. For example the gnomAD or ExAC study collects genomes or exomes to show heterogeneity of the population so that other projects can use them as a reference database. Other studies try to to find causative mutations for rare or common diseases (e.g. by de Ligt *et al.* [30] and Stahl *et al.* [70]). Here, the challenge is to find the one or the few causative changes within around three million differences compared to a reference genome.

As explained in Section 2.4 there are multiple approaches to predict the pathogenicity on variants mainly for coding regions [6–9]. But the protein coding part is only a small subset less than 1.5 % of the genome (according to National Center for Biotechnology Information Reference Sequence Database (RefSeq) annotation release 108 [71]). A few approaches are available for the non-coding genome but some of them did not have a major focus on training their methods on the non-coding part, or they followed different training concepts, like generalizing using evolutionary ideas or extracting non-coding features (compare with Section 2.4). Recent studies showed that non-coding scores do not perform well in predicting non-coding causative mutations for Mendelian diseases and cancer [56, 63].

Disease and trait-associated variants represent a tiny minority of all known genetic variation. For instance in non-coding regions the number of available positive examples for Mendelian diseases is of the order of several hundreds, while the number of negative examples is of the order of millions [56].

Similar conditions, with different levels of imbalance between positive and negative examples, may arise in other medically relevant contexts, e.g. in variants related to cancer [72] or in GWAS for complex diseases [73]. The vast majority of trait- or complex-disease-associated variants identified to date in GWAS have been found to be located outside of protein-coding sequences and in some cases localize to known gene regulatory elements such as promoters and enhancers. But it is difficult to judge if these common variants are causative or if they can be interpreted as "tags" for haplotypes on which functional variants reside, rather than necessarily being disease-causing themselves [74, 75].

Relatively few NCVs have been identified to date as causal for Mendelian disease. Although historically, variation in non-coding sequences has remained underinvestigated [76], mutations have been confidently identified in a wide range of non-coding function elements, including promoters, enhancers, 5' and 3' UTRs, RNA genes, and imprinting control region (ICR) elements [56]. A better understanding of regulatory variants will therefore be necessary to unravel the functional architecture of rare and common disease.

However, computational algorithms for the analysis of non-coding deleterious variants are faced with special challenges owing to the rarity of confirmed pathogenic mutations. In this setting, classical learning algorithms, such as SVM [77] or artificial neural networks [78] tend to generalize poorly, because they usually predict the minority class with very low sensitivity and precision [79]. In the context of the prediction of genetic variants associated with traits or diseases, this boils down to wrongly predicting most of the disease-associated variants as non-disease associated, thus significantly limiting the usefulness of machine learning methods for the prediction of novel disease-associated NCVs.

Structure of this Chapter To deal with the imbalance problem that naturally arises from the analysis of genetic variants in the human genome, I developed a ML method, Hyper SMOTE Undersampling with Random Forests (hyperSMURF), which is specifically designed for the analysis of (extreme) imbalanced (genomic) data. The topic of this chapter is the method hyperSMURF and its performance. In Section 3.1 I describe hyperSMURF in detail including the pseudo-code in Algorithm 3.1 together with implementation details. Datasets used for performance measurement together with other ML strategies to which I compare hyperSMURF are described in Section 3.3. Section 3.4 examines the features of the datasets in more detail. Section 3.5 presents the performance results of hyperSMURF compared to other ML strategies and finally Section 3.6 concludes this chapter.

3.1 HyperSMURF

Hyper SMOTE Undersampling with Random Forests (hyperSMURF) is a method specifically conceived to handle extremely imbalanced data. To deal with such problems and to achieve high coverage of the available input data as well as a high accuracy of the predictions, hyperSMURF is based on three complementary strategies: (1) sampling techniques; (2) ensemble methods; (3) a hyper-ensemble approach. Now these strategies are described briefly and Figure 3.1 contains a schema of hyperSMURF which concludes the overview.

Sampling Techniques The majority class is subdivided into $n \in \mathbb{N}$ non-overlapping partitions. Their instances are randomly subsampled to reduce the number of negative examples in each partition, reducing extreme imbalance. At the same time oversampling techniques are applied to the minority class to enlarge their number. For oversampling, SMOTE from Chawla *et al.* [21] is used. This allows the construction of new synthetic minority examples similar to the available original ones (see Section 2.2.4 for more details). Finally, for each subsampled partition of instances from the majority class, all minority class instances and a different oversampled set of minority instances are added, resulting in n balanced datasets.

Ensembling Methods Each balanced dataset $d_i, i \in \{1, 2, \ldots, n\}$ does not in itself assure a sufficient coverage of the available training data, since only a small subset of the majority class is included. To overcome this limitation, a different learning machine is trained on each partition d_i of the n different datasets and then the predictions of the n resulting models are combined according to ensemble techniques to obtain a "consensus" prediction of the ensemble. Note that each base learner is trained on different training data, thus assuring diversity between the base learners, a key concept for the success of ensemble methods [80]. At the same time a high coverage is guaranteed, since the n datasets include all or a significant part of the available training data. Section 2.2.2 contains more details on ensemble methods.

Hyper-Ensemble Approach Another key factor for the success of ensemble methods is the accuracy of the base learners [80]. As described in Section 2.2.2, ensembles usually improve the

accuracy and the robustness of the predictions of the learning machines [81, 82]. To this end, a RF [17] was used instead of training a single base learner for each dataset d_i. Then the predictions were combined by averaging across the probabilities estimated by each RF. Because each base learner is in turn an ensemble of decision trees, the hyperSMURF method results in a hyper-ensemble approach (an ensemble of ensembles). In principle any ensemble of learning machines can be used but RFs have been chosen because of their prooven effectiveness in the analysis of genetic variants [83, 84].

In addition hyperSMURF was trained on a feature set which contains groups of scores with similar features like conservation scores (see Section 3.4). The random feature selection of RFs results in different feature combinations between and within the groups, so that trees become uncorrelated from feature groups, resulting in different variances, which results in a better performance [17]. See Section 2.2.3 for more details on RFs.

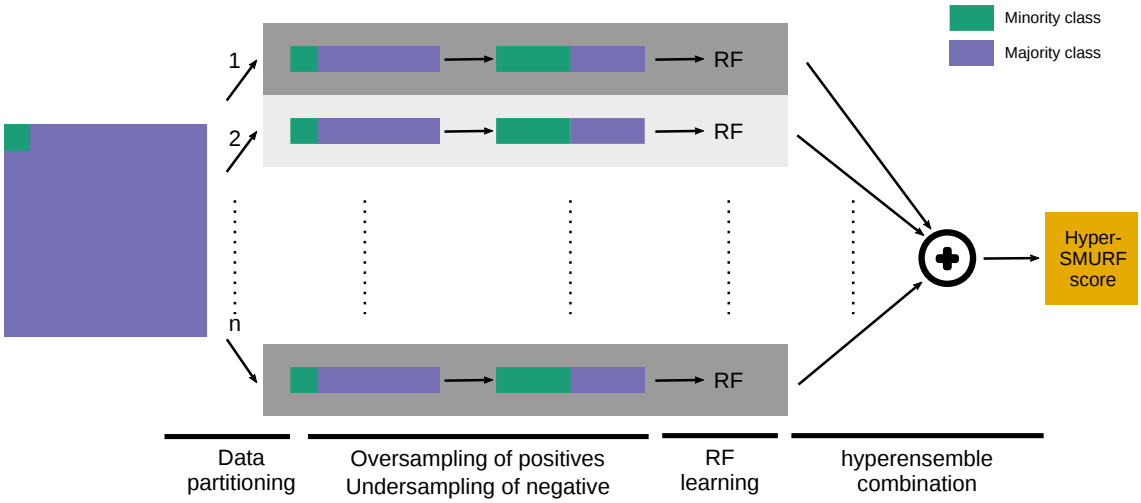


Figure 3.1: Schematic overview of hyperSMURF. This schema visualizes the steps processed in hyperSMURF. At the beginning we start with an imbalanced dataset for training (left-side). The training set is partitioned into n partitions using all minority class instances (green) and an equal split of the majority class instances (blue) for every partition. In every partition the minority instances are oversampled using SMOTE and the majority instances are subsampled to an appropriate size. Here minority and majority instances are equal after over- and undersampling. Then, for each partition, a RF is trained. When a new instance $\boldsymbol{x}$ should be predicted, the average prediction of all n RFs is used (hyper-ensemble combination) to generate the hyperSMURF score ($Hy_{score}(\boldsymbol{x})$; right-side).

3.1.1 Algorithm

This section explains the pseudocode of hyperSMURF in Algorithm 3.1 in more detail. $\mathcal{I}^{min}$ represents the instances of the minority class while $\mathcal{I}^{maj}$ are the instances of the majority class. More precisely $\mathcal{I}^{min} = \{(\boldsymbol{x}, y)|\boldsymbol{x} \in \Theta, y = 1\}$ and $\mathcal{I}^{maj} = \{(\boldsymbol{x}, y)|\boldsymbol{x} \in \Theta, y = 0\}$, where $y \in \{0, 1\}$ represents a binary classification and Θ is a set of real-valued vectors representing the features associated with each instance.

The first algorithm step (i) is the partitioning. Here, the algorithm subdivides the instances of the majority class $\mathcal{I}^{maj}$ into n partitions $\left\{\mathcal{I}_1^{maj}, \mathcal{I}_2^{maj}, \ldots, \mathcal{I}_n^{maj}\right\}$, such that $\bigcup_{i=1}^{n} \mathcal{I}_i^{maj} = \mathcal{I}^{maj}$ and $\bigcap_{i=1}^{n} \mathcal{I}_i^{maj} = \emptyset$ (line 1).

During the for-loop (lines 2 to 7), the algorithm iterates over the n partitions of the data, does over- and undersampling steps, and finally trains a different RF [17] at each iteration. At first within the loop – algorithm step (ii): oversampling – a set of new synthetic instances $\mathcal{I}_{S_i}^{min}$ is generated using all minority instances $\mathcal{I}^{min}$ through SMOTE [21] (line 3). This algorithm generates representative novel instances that are similar to the original instances by linear interpolation between randomly chosen pairs of close instances of the minority class. For more information about SMOTE see Section 2.2.4. The resulting "synthetic" dataset $\mathcal{I}_{S_i}^{min}$ has a cardinality $|\mathcal{I}_{S_i}^{min}| = o\left|\mathcal{I}^{min}\right|$. Thus, if o is set to $o = 0$, no oversampling is done. Note that the oversampling factor o is related to the factor N that is described in the SMOTE Algorithm 2.3 $\left(o = \frac{N}{100}\right)$.

In the second part within the loop – algorithm step (iii): undersampling – a subsample of majority instances $\widehat{\mathcal{I}}_i^{maj}$ is constructed for every partition i by randomly subsampling instances from $\mathcal{I}_i^{maj}$ using the undesampling factor u in order to obtain $\left|\widehat{\mathcal{I}}_i^{maj}\right| = u\left(\left|\mathcal{I}^{min}\right| + \left|\mathcal{I}_{S_i}^{min}\right|\right), u > 0$ (line 4). If u is set to $u = 0$, the undersampling step is skipped. A value of $u = 1$ will balance both classes during training but sometimes it might be favorable to retain some imbalance while training, e.g. using a three times larger majority set with $u = 3$. Thus, we can exploit a larger set of minority instances for training.

The algorithm step (iv) is the training set assembly. The training set $\mathcal{T}_i$ for every partition i is obtained by a simple union of the sampled majority $\hat{\mathcal{I}}^{maj}$, original minority $\mathcal{I}^{min}$, and synthetic minority instances $\mathcal{I}_{S_i}^{min}$ (line 5).

At each iteration of the for-loop the resulting training set $\mathcal{T}_i$ is used to train a RF model C_i (line 6) which is able to compute the probability $P(\boldsymbol{x} = 1|C_i)$ that a given instance $\boldsymbol{x}$ belongs to the positive class label – algorithm step (v): RF training. The output of the algorithm is a set $C = \{C_1, C_2, \ldots, C_n\}$ of RFs (line 8).

Finally, if C is constructed, the hyperSMURF score (Hy_{score}) for a new instance $\boldsymbol{x}$ can be computed by averaging across the probabilities estimated by each different base RF:

$$Hy_{score}(\boldsymbol{x}) := \frac{1}{n} \sum_{i=1}^{n} P(\boldsymbol{x} = 1|C_i) \tag{3.1}$$

In the standard hyperSMURF settings each base learner C_i is a RF, which is in turn an ensemble of decision trees, thus resulting in an ensemble of ensembles (hyper-ensemble). But in theory it can be replaced by any other learner.

In sum, by adopting both over- and undersampling techniques [21, 23, 85] to increase the cardinality of the minority class and to reduce the cardinality of the minority class, hyperSMURF obtains balanced datasets for training the RF. Moreover the ensembling approach assures a high coverage of the available training data with diverse base learners, since each RF is trained on different training data. Finally the hyper-ensemble approach uses ensembles as base learners, instead of single learning machines, which assures accurate and robust predictions.

Algorithm 3.1: Hyper SMOTE Undersampling with Random Forests (hyperSMURF).

Input : $\mathcal{I}^{maj} \leftarrow$ set of majority class
$\mathcal{I}^{min} \leftarrow$ set of minority class
$n \leftarrow$ number of partitions
$k \leftarrow$ number of nearest neighbors for *SMOTE*
$o \leftarrow$ oversampling factor
$u \leftarrow$ undersampling factor

Output : Set of random forests C with $|C| = n$

`// (i) Partitioning of` $\mathcal{N}$

$\left\{\mathcal{I}_1^{maj}, \mathcal{I}_2^{maj}, \ldots, \mathcal{I}_n^{maj}\right\} \leftarrow$ *partition* $\left(\mathcal{I}^{maj}, n\right)$

for $i \leftarrow 1$ **to** n **do**

`// (ii)` ***SMOTE*** `oversampling minority class`

$\mathcal{I}_{S_i}^{min} \leftarrow$ *SMOTE* $\left(\mathcal{I}^{min}, o, k\right)$

`// (iii) Undersampling majority class`

$\widehat{\mathcal{I}}_i^{maj} \leftarrow$ *Undersample* $\left(\mathcal{I}_i^{maj}, u, \left|\mathcal{I}_{S_i}^{min}\right|\right)$

`// (iv) Training set assembly`

$\mathcal{T}_i \leftarrow \mathcal{I}^{min} \bigcup \mathcal{I}_{S_i}^{min} \bigcup \widehat{\mathcal{I}}_i^{maj}$

`// (v) Random Forest training`

$C_i \leftarrow RF(\mathcal{T}_i)$

end

return $C \leftarrow \{C_1, C_2, \ldots, C_n\}$

Calculating the out-of-the-bag error (see Section 2.2.2) using the partitions is possible but not recommended. Of course every instance of the majority class is only used once and the $n-1$ partitions can be used for calculating. But all original instances of the minority class are present in all partitions. Therefore we cannot use them. It is possible to use the artificial minority instances generated by smote because they should differ between the partitions due to randomization in SMOTE. But they are synthetic and their benefit is only during the training step. In addition we cannot ensure that synthetic instances always have different feature vectors between the partitions and therefore there can be identical sampled synthetic instances. Because of these considerations the out-of-bag error is not implemented and not considered further.

3.1.2 Implementation

HyperSMURF is implemented in R as a library and in Java as a Weka plugin [15]. The R library is available via the Comprehensive R Archive Network (CRAN)[1] under `https://cran.r-project.org/package=hyperSMURF` including a reference manual and examples. The hyperSMURF Weka plugin is hosted on GitHub (`https://charite.github.io/hyperSMURF`) under GNU General Public License Version 3 (GNU GPLv3). The Java hyperSMURF implementation can be downloaded and included in other programs using the software project management and comprehension tool Maven, or it can be included directly into Weka using the plugin manager. Appendix E contains a detailed tutorial about the hyperSMURF Java package and how it can be integrated in other Java programs to train and test instances. Further, I created an extended manual with installation details and examples on `https://hypersmurf.readthedocs.io` and the Java application programming interface (API) is hosted on the website `https://javadoc.io/doc/de.charite.compbio/hyperSMURF`. Results presented in this work were created with the Java hyperSMURF implementation.

3.2 Genomic Data

To show the effectiveness of the proposed approach, I trained hyperSMURF with different genomic imbalanced datasets and performed an experimental comparison with learners of popular non-coding deleteriousness scores and other published approaches. The following three challenging and medically relevant data for the ML prediction problem were selected: first, the prediction of regulatory mutations underlying Mendelian diseases [56] (Section 3.2.1); second, the prediction of NCVs associated with GWAS regulatory hits from the GWAS-Catalog [48] (Section 3.2.2); and third, the prediction of the relationship between functional elements and miRNA regulation (Section 3.2.3).

3.2.1 Mendelian Data

For the Mendelian dataset we have a binary classification problem. On the one hand there are the non-coding Mendelian disease-associated mutations (positive class). On the other hand there are observed probably non-deleterious variant sites (negative class). In total hyperSMURF was trained with 406 positive and 14,755,199 negative instances resulting in an imbalance of ∼1:36,000. Each genomic position of a variant was annotated with 26 different genomic attributes including conservation scores and chromatin states. Now, positives and negatives for the Mendelian data along with their genomic attributes are described in detail.

Regulatory Mendelian Mutations For the positive variant set a comprehensive literature review was performed to identify NCVs that are convincingly associated with Mendelian disease. Only those variations were included for which the publications provide plausible evidence of

[1] `https://cran.r-project.org`

pathogenicity. The following lines describe the criteria and the judging approach NCVs. The phenotypic abnormalities of the individual carrying the variant were assessed and a variant was included only if the disease association was regarded as plausible on the basis of evidence such as familial co-segregation or experimental validation, using techniques such as luciferase reporter assays, electrophoretic mobility assay, or telomerase activity assay. In some cases pathogenicity was assigned based on curator judgment or computational predictions; for instance, mutations in RNA genes that affected RNA secondary structure elements such as stem loops were included.

To identify articles for biocuration, a number of review articles were consulted on non-coding mutations [73, 86–91] including 5' and 3' UTR mutations [92–97], enhancer mutations [76, 98, 99], promoter mutations [100, 101], and mutations affecting miRNA genes or miRNA recognition sites in mRNAs [102–105]. Additionally, locus-specific databases were reviewed for selected genes [106–108]. Variants that represent susceptibility loci for common, complex disease (for example GWAS hits) were excluded. Likewise, somatic variants associated with cancer were discarded.

A total of 453 unique non-coding Mendelian disease-associated variants were identified (see Appendix A and Tables A.1 to A.7). Mutations were manually mapped to the human genome reference GRCh37, if necessary. Each variant was cataloged according to its sequence variant type (Table 3.1). The disease associated with the variant was mapped to an Online Mendelian Inheritance in Man (OMIM) disease identifier and the affected gene was encoded with a National Center for Biotechnology Information (NCBI) Entrez Gene identifier. For hyperSMURF performance measurement only SNVs were used as positive dataset and therefore 47 InDels were removed, resulting in 406 positive variants for performance evaluation.

Observed Probably non-deleterious Variant Sites For negatives, single-nucleotide sites in the human genome were identified at which the human genome reference sequence differs from the inferred ancestral primate genome, based on the Ensembl EPO whole-genome alignments of six primate species [40, 42] (Ensembl Compara release e71). A file containing the inferred ancestral sequences was downloaded from the 1KG website [59, 68]. All positions of high-confidence alignments that differ from the human reference sequence (hg19) were selected. Low-confidence calls are defined in the EPO alignment file as those where the ancestral state is supported by one sequence only. The alignment file was compared with the human genome sequence (hg19) via an in-house Java program that cataloged the differences found according to location with respect to genomic annotations. Table A.8 shows a detailed summary of the inspected differences.

Nucleotide positions associated with variants present in the phase 3 1KG data at a frequency of higher than 5 % were excluded [59, 68]. This means that the derived allele in the human genome is present at a frequency of less than 95 % so that it is less certain that the allele has been exposed to many generations of natural selection. The Variant Call Format (VCF) file with integrated AFs from the 1KG phase 3 was downloaded from their FTP server on May 30, 2015 and the AF info field of the 1KG database VCF was used as the threshold. All variants were annotated with Jannovar v.0.14 [109] using the RefSeq [71] (annotation release 105) and only variants that were annotated with a non-coding variant effect were used as final non-deleterious variant sites (negative positions).

Table A.8 shows the distribution of variants extracted in this way and the variant categories selected for analysis are marked. This yielded to a total number of 14,755,199 sites. Because deleterious variants are depleted by natural selection in fixed or nearly fixed derived alleles, we can infer that variations at these sites are unlikely to be associated with Mendelian disease. Therefore the genomic sites from this set were chosen as negative instances for training.

Table 3.1: Overview of non-coding regulatory Mendelian mutations. A total of 453 unique, non-coding, regulatory Mendelian mutations were identified by manual biocuration (see Appendix A and Tables A.1 to A.7). The pathomechanism of a subset of the 5' and 3' UTR mutations was indicated in the original publications and is shown here. 406 of the 453 mutations were single-nucleotide variants and were used as positive instances for machine learning. One example of a disease caused by each pathomechanistic category is shown together with the affected gene and the OMIM number of the disease.

Category	**Example**			**Count**
	Disease	**Gene**	**OMIM ID**	
Enhancer	Triphalangeal thumb, type I	SHH	174500	42
Promoter	Hemophilia B	F9	306900	142
5' UTR				153
Transcription (core promoter)	Acute intermittent porphyria	HMBS	176000	52/153
uORF	Marie Unna hereditary hypotrichosis	HR	146550	37/153
Secondary structure	Hyperferritinemia cataract syndrome	FTL	600886	31/153
Kozak sequence	Beta thalassemia	HBB	613985	2/153
Unclassified	Thrombocytopenia 2	ANKRD26	188000	31/153
3' UTR				43
Polyadenylation	Permanent neonatal diabetes	INS	606176	14/43
miRNA binding	Autosomal-dominant spastic paraplegia 31	REEP1	610250	5/43
Other	Autosomal-dominant myopia 21	ZNF644	614167	24/43
Large non-coding RNA gene	Microcephalic osteodysplastic primordial dwarfism, type 1	RNU4ATAC	210710	65
MiRNA gene	Autosomal-dominant deafness 50	MIR96	613074	5
Imprinting control region	Beckwith-Wiedemann syndrome	H19	130650	3
Total				453
Total SNVs				406

Genomic Attributes Every position in the genome (hg19 release) was annotated with 26 numeric features. Conservation scores PhastCons and PhyloP [12, 13] were derived from University of California, Santa Cruz (UCSC) [110] using the multi-species alignment of 9 primates, 32 mammals and 45 vertebrates. GERP++ element scores and the corresponding p-values were downloaded from the GERP website on June 6, 2015 [10, 11]. CpG and G/C content as well as the observed to expected CpG ratio were downloaded directly from the UCSC Table Browser on June 6, 2015 [111]. In addition the GC content in the human genome (hg19 release) in a range of ±75 bp for every position was computed (Ns are not counted). Transcription and regulation annotations were downloaded from UCSC [110]. The maximum ENCODE H3K27 acetylation level along with the maximum ENCODE H3K4 methylation level and the maximum ENCODE H3K4 trimethylation level was used. DNase hypersensitive scores were derived from the UCSC ENCODE Regulation DNase Clusters track V3 along with the number of overlapping TFBSs conserved in the human/mouse/rat alignment. In addition, permissive and robust enhancers were taken from the FANTOM5 project [112]. Population-based features were computed by counting the number of rare ($\leq$0.5 % AF) and common (>0.5 % AF) 1KG variants [59, 68] (phase 3 version 5a of 05/02/2013) in a window of ±500 and using the ratio of rare variants ($\leq$0.5 % AF) and common variants (>0.5 % AF) (zero if common variants are zero). Finally, overlapping copy-number variations (CNVs) for every position in the human genome were counted for each position in the CNV databases Database of Genomic Variants (DGV) [113], dbVar [114] and ISCA [115] (study IDs nstd37, nstd45 and nstd75). All attributes are listed and briefly explained in Table A.9.

3.2.2 GWAS Data

To predict regulatory SNPs in the genome, 2115 regulatory GWAS hits were downloaded on May 13, 2016 from the location `https://xioniti01.u.hpc.mssm.edu/v1.0/EIGEN_TestDatasets/GWAS_eQTLs/GWAScatalog_EIGEN.txt` [36]. Regulatory GWAS hits were defined from the GWAS-Catalog [48] that overlap with a known regulatory element. These hits were also the variations that received the highest Eigen scores by Ionita-Laza *et al.* [36]. The same non-deleterious sites defined in the Mendelian data in Section 3.2.1 were used as negatives but every position overlapping with a regulatory GWAS hit was removed (remaining 1,475,505 negatives).

Variants were annotated with a different feature set to show that the hyperSMURF method is not dependent on the features. Here the deep CNN method for extracting functional annotations from DeepSEA was used. Therefore regulatory GWAS hits and negative variants were converted into VCF format. Then, by running DeepSEA [38], their chromatin effect features were extracted. For this task DeepSEA in version 0.94 together with the genome assembly GRCh37/hg19 were used. The code for the standalone version and additional evolutionary conservation score files were downloaded from `http://deepsea.princeton.edu/media/code/deepsea.v0.94.tar.gz` in order to produce a set of genomic attributes directly extracted from the DNA sequence through deep CNNs.

Using the DeepSEA feature extraction, the feature set is larger than the one used in the Mendelian dataset. The deep convolutional network at the core of the DeepSEA method makes predictions for 919 chromatin features (125 DNase features, 690 TFBSs features, 104 histone features). For

each of the considered variants DeepSEA predicts two score values by analyzing two 1000 bp FAST-All (FASTA) sequences centered on the variant based on the reference genome and carrying the reference and the alternative allele at the variant position. The DeepSEA method is described in Section 2.4.3 in more detail.

The predicted chromatin effect scores provided by DeepSEA were used to compute two set of features: 919 absolute differences of probabilities in Equation (2.16) and 919 relative log fold changes of odds in Equation (2.17), obtaining an initial set of 1838 features. In addition the four precomputed evolutionary scores were extracted from the DeepSEA intermediate output. The scores are the PhastCons score for primates [12] (excluding human), the PhyloP score for primates [13] (excluding human) and the two GERP++ scores [10, 11]. Substitution scores were not used. Finally a set of 1842 features was obtained for each variant. According to the protocol detailed by Zhou & Troyanskaya [38] all the extracted features were normalized to zero mean and unit variance prior to further analyses.

The processing of the features with deep convolutional networks, using an NVIDIA K20 device for Graphics Processing Unit (GPU) parallel computing and a server with 128 GB of RAM required about 115.6 hours of computation time for processing 1,475,911 variants. To reduce the computation time while maintaining a sufficiently large set of negative background variants, only 10 % of the negative data were taken randomly (1,475,505 variants in total), resulting in an imbalance of ∼1:700.

3.2.3 eQTL Data

To determine the relationship between functional elements and miRNA regulation, the raw data of Budach *et al.* [116] was downloaded from `https://github.molgen.mpg.de/budach/miRNA_eQTL` on January, 2017. The file *modelmatrix.txt.gz* contains 2,002,126 SNPs in total, 4785 of them are associated with miRNA-eQTLs and 1,997,341 with non-miRNA-eQTL observations. This leads to an imbalance of ∼1:400. Budach *et al.* [116] choose 18 features for their final model during *feature selection*. Features contain different transcription factors, ChromHMM states, miRNA promoters, and multiple separate parts of the miRNA primary transcript. HyperSMURF and their logistic regression model was trained on the complete imbalanced dataset instead of using random balanced datasets like in the initial publication [116].

3.3 HyperSMURF Performance Measurement

The performance of hyperSMURF was assessed using the three datasets described in the previous section. Special evaluation strategies were developed to generate an unbiased performance measurement (see Section 3.3.1). Finally, in Section 3.3.2, I performed an experimental comparison of hyperSMURF with learners of popular non-coding deleteriousness scores and other published approaches. This retraining and testing of the models was done on the same data and genomic features, instead of using pre-trained models or pre-computed scores as it has been usually done in previous work [34, 36, 38, 56, 83]. The other non-coding scores and their models are previously

introduced in Section 2.4. In this chapter the learners are named after the score for simplicity but the results listed here always show the retrained ML methods.

3.3.1 Performance Evaluation Strategies

Extensive performance measurements, tests on the influence of different parameters and the behavior on different imbalance settings were primary made with the Mendelian data (see Sections 3.2.1 and 3.5). The main purpose of GWAS (Sections 3.2.2 and 3.5.3) and eQTL data (Sections 3.2.3 and 3.5.4) was the comparison between hyperSMURF and other ML methods, to show that hyperSMURF can be used with different genetic data as well as a different types and sizes of features.

On the Mendelian and GWAS data the model performance was tested with a *cytogenetic band-aware 10-fold cross-validation (cytoband-aware 10-fold CV)* to ensure that positive or negative variants of the same location, gene, or disease do not occur in the training and test set. Therefore the mutations were partitioned into the chromosomal bands. Bands with at least one positive mutation were assigned to one of the ten folds so that each fold had similar numbers of positives (stratified CV). To be more precise the Mendelian data (Section 3.2.1) contains around 40 positives and the GWAS data (Section 3.2.2) around 212 positives per fold. The remaining bands were randomly assigned to the different folds and negative sites were added to the partition of their associated band.

For each round of the CV, the nine folds corresponding to the training set underwent the subdivision of hyperSMURF into n partitions and were over- and undersampled according to the procedures described in Section 3.1.1. The trained ensemble was then tested on the remaining held-out unchanged fold not used for training. In this way, across the ten rounds of the CV procedure, all available genomic positions were tested.

Negative variants that are genomically close to positives might have similar features, like GC content. Therefore those data might be more difficult to separate for the classifier. To test the capability of hyperSMURF to distinguish classes on those extreme settings, further experiments were performed by selecting as negatives those variants being within 100, 500 and 1000 Kb distance from a positive, or being in the same TAD. Then a *topologically-aware cross-validation (topologically-aware CV)* was performed by constructing a number of folds equal to the number of TADs or the number of 100, 500 and 1000 Kb genomic windows around each positive. Overlapping windows were merged together. In this way the performance evaluation metrics are calculated based on the ranking of each positive over its matched negatives rather than on the overall ranking of all positive variants over all negative variants.

Two evaluation strategies were realized with the eQTL data. On the one hand the initial approach by Budach *et al.* [116] was implemented. Here, three quarters of all instances were randomly sampled for training, using the rest for testing, and the analysis was repeated 50 times. For this analysis the performance results are shown in Section 3.5.4. On the other hand an approach similar to the cytoband-aware 10-fold CV, described previously, was taken. Thus, SNPs that affect multiple (predicted) recognition sites of the same miRNA might have similar behavior to unrelated SNPs. To ensure that SNPs of the same miRNA do not occur in the training and test

set, they were partitioned into their relative miRNA. Then each miRNA was randomly assigned to one of the 10 folds by assuring that the size of the eQTLs was similar in all 10 folds (around 478 per fold; stratified CV). Training and test sets for hyperSMURF and the logistic linear regression were equal for the repetitive training and the *microRNA-relation cross-validation (miRNA-relation CV)*. Performance results of the miRNA-relation CV experiment are presented in Section 3.5.4.

Automatic Tuning of Parameters HyperSMURF learning depends on several parameters including oversampling and undersampling ratios, the size of the hyper-ensemble (number of partitions), as well as the learning parameters of the RF that constitute the base learners of the hyperSMURF method. Therefore I developed a parameter search to find out characteristics of hyperSMURF and to gain the optimal performance. Parameter search was done using the cytogenetic band-aware CV strategy. For each round in the CV the nine partitions of the training set were used to make an internal 9-fold CV using different parameter settings. For every parameter combination the AUPRC was recorded. Finally, an optimal hyperSMURF was trained, using for every 10-fold CV step the best parameter setting in terms of the highest AUPRC from the internal 9-fold CV.

An extensive search was done using the Mendelian data (see Section 3.2.1). Because of performance issues the parameters were split into three analysis types. First, optimization was done using the number of partitions n, together with the oversampling factor o from SMOTE and the undersampling factor m. Parameters of the RF were kept fixed using $t = 10$ random trees and $d = 5$ random features. Second, the optimal k nearest neighbors of SMOTE per fold were searched using the optimal values of the previous analysis. And finally third, the RF settings were optimized with fixed partitions $n = 100$, over- and undersampling ($o = 2$, $m = 3$). HyperSMURF training on Mendelian data was repeated 100 times with standard settings and the optimal settings (see Tables 3.2 and B.3) using different *seed*s to get an average performance that is not dependent on randomization issues. Results of the parameter tuning can be found in Section 3.5.1.

3.3.2 Comparison with state-of-the-art Methods

The imbalance-aware hyperSMURF method was compared with the retrained learners of popular methods for scoring NCVs. This extensive comparison was done on the Mendelian and the GWAS data. Learners of the following scores were used: CADD, GWAVA, DeepSEA, Eigen and Eigen-PC [33, 34, 36, 38]. Apart from GWAVA, which uses a modified version of the RF algorithm by balancing the training data through undersampling of the majority class (see Section 2.4.2 for more details about GWAVA), the other learners are imbalance-unaware in the sense that they do not adopt specific learning techniques to deal with highly imbalanced data. All models were retrained and tested using the Mendelian and GWAS data (see Sections 3.2.1 and 3.2.2) instead of using pre-trained models or pre-computed scores as it is often done in previous work [34, 36, 38, 56, 83].

The SVM underlying the CADD score v1.0 was reimplemented by a more efficient C++ implementation using the LibLinear library v.2.01 [117]. The GWAVA learner was reimplemented

using Java and the Weka library [15]. The module is called weka-GWAVA (version 0.1) and source code is available from `https://charite.github.io/weka-GWAVA`. The R code for the Eigen and Eigen-PC methods was downloaded from the Eigen Website: `http://www.columbia.edu/~ii2135/Eigen_functions_112015.R`, while the original DeepSEA software in version 0.94 was downloaded from `http://deepsea.princeton.edu/media/code/deepsea.v0.94.tar.gz`.

All methods were used with the same input data and with exactly the same folds using the previously described cytoband-aware 10-fold CV. The settings of the individual methods were selected from their original publication. This means GWAVA was used with a forest-size of 100, but the number of random features was used as in hyperSMURF, because the feature sizes differ between the original publication and the data used in this work. For the SVM approach of the CADD method the same c of $c = 0.0025$ was used as by Kircher *et al.* [33] and finally the code of Eigen, Eigen-PC and DeepSEA remained untouched. For Eigen conditionally independent blocks were identified manually using feature correlation (see Section 3.4). In the Mendelian data three blocks were defined: conservation, population and regulatory features. The GWAS data contains the following three blocks: conservation, logfold and diff. The results of this extensive comparison are shown in Section 3.5.3.

3.4 Informative Features

Univariate analysis was performed to see the performance of single features in the different datasets. This gives an idea about top performing features by themselves. But it is worth pointing out that some features perform poorly alone but have a high information content when combined with another feature. For a robust prediction the average AUROC and AUPRC of an univariate logistic regression model repeated 100 times [118] implemented in Weka [15] was used. To measure the performance for the Mendelian and GWAS data the previously described cytoband-aware 10-fold CV was performed.

Budach *et al.* [116] did an extensive performance measurement of their eQTL features with an additional feature selection that results in 18 remaining features for training. The performed feature selection was similar to the univariate logistic regression analysis except that for every regression run the distance between SNP and miRNA was added as second feature. Therefore no univariate logistic regression analysis was performed here.

In addition to the univariate analysis, correlation plots of features using the Pearson correlation coefficient were made to inspect correlation between features and to find functional groups (see Figure 3.2). These groups are needed as input for Eigen to obtain the between-block correlations (see Section 2.4.4).

Mendelian Data Results of the 100 times repeated univariate logistic regression model of all 26 features are shown in Table B.1. Most of the features show an information gain on their own, except the two FANTOM5 features, the GERP++ element-score p-value and counts of rare variants as well as counts of overlapping CNVs in dbVAR, DGV, and ISCA.

The population features like CNVs or SNVs are expected to have a poor performance as singleton features. For example the number of common and rare variants in a ±500 window have AUROCs of 0.58627 and 0.50697. So there is no information gain if we know that a region has many or just a few rare variants. For common variants there is a small gain on highly polymorphic regions like the major histocompatibility complex (MHC) because these regions might not contribute to Mendelian disease. But of course, it is of high information if we know the fraction of rare and common variants in those regions, meaning that on polymorphic sites there might by chance also be more rare variants. But if this is not the case, this might be an indicator for a negative selection of rare variants at the site. This hypothesis underlines the performance of the fracRareCommon feature, a ratio of rare to common variants. The information gain in terms of the AUROC tremendously grows to 0.67170. So we can suggest similar results for the CNVs by using the ratio between pathogenic CNVs (dbVAR, ISCA) to polymorphic CNVs (DGV).

By inspecting the correlation of the features in Figure 3.2a we can define roughly four groups. One large group are the conservation scores. Another group are the chromatin features including also CpG and GC content. The third group are the population features like CNVs, rare or common variants and the last group are the FANTOM5 tracks which have only a tiny relation to DNase features and the H3K4 methylation level. For preselected Eigen blocks FANTOM5 were added to the chromatin feature group.

GWAS Data The GWAS data has a large set of 1838 features resulting in a total of 1,838,000 trainings of the logistic regression models because of the 10-folds and the 100 repetitions. The outstanding features are the two conservation scores PhyloP and PhastCons. PhyloP has a large AUPRC of 0.36978 $\pm$ 0.01004 (AUROC of 0.76382 $\pm$ 0.00019) and PhastCons is highly predictive because the AUROC is over 0.9 (AUROC $= 0.93689 \pm 0.00094$ and AUPRC $= 0.04262 \pm 0.00332$). All non-conservation features that were generated by the deep CNN of DeepSEA seem to be not informative by themselves. Most features have an AUROC around 0.5. The univarite results of the genomic features were ranked according to the estimated AUPRC and the ten best and ten less informative features are listed in Table B.2.

For the correlation matrix the four conservation scores and the eight best other features together with their logfold values were selected. The corresponding correlation matrix in Figure 3.2b shows two correlation groups: on the one hand the conservation scores and on the other hand the CNN features (chromatin, DNase, and TFBS features). In addition there is a clear difference between logfold and the absolute difference of the probability features. This shows that the different computation of features results in different feature space, like it is visualized in Figure 2.1. The highly correlated CNN features suggest that the feature set of the GWAS data might be overloaded because we can imagine a similar behavior for the remaining 1818 features. So it might be advantageous to use a single score over multiple cells like the maximum H3K4 methylation level in all cells used in the Mendelian dataset (see Table A.9).

Overall these results show that it will be important to select always one conservation score for training a classifier. Therefore in hyperSMURF I would suggest to increase the number of random features to 100 so that around 22 % of the random trees in the RF will use a conservation score. This results from the four conservation features in the 1838 large feature set as seen the equation

$\frac{4}{1838} \cdot 100 \cdot 100 = 21.76\,\%$. Another option will be to select the features in a stratified way by building groups. Then the algorithm must be adapted to always select a feature out of each group to ensure that a conservation score will be always selected. Another common option is to use weights for features.

eQTL Data As explained above no univariate logistic regression was done for the eQTL data. Only the correlation between the remaining features is visualized in Figure 3.2c. These results suggest that the remaining features, after the feature selection, were wisely chosen because no correlation is detected and every feature reflects a different feature space for the prediction.

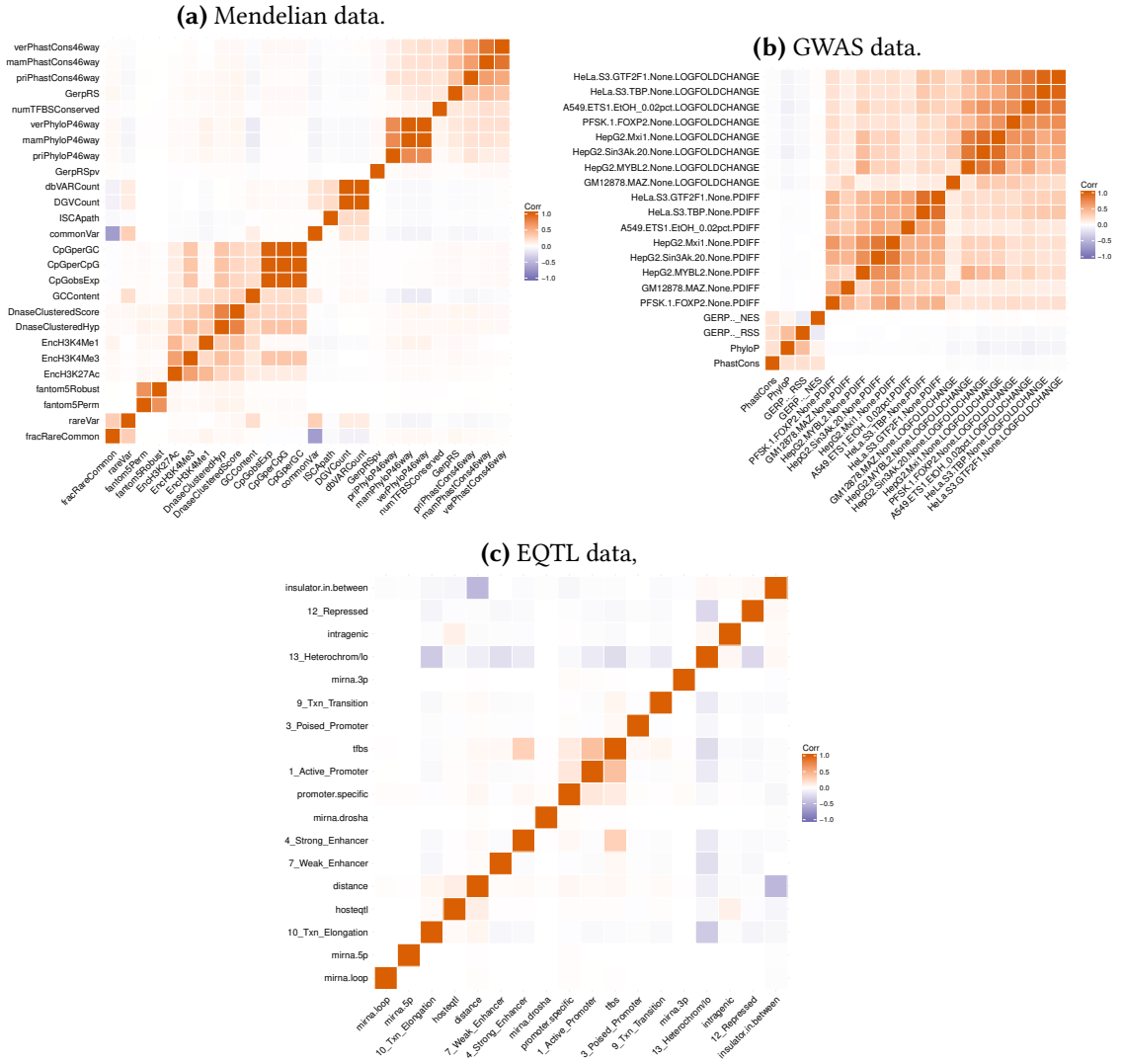


Figure 3.2: Feature correlation of Mendelian, GWAS and eQTL data. Visualization of correlation between the features within **(a)** Mendelian, **(b)** GWAS and **(c)** eQTL data. Correlation is computed by Pearson correlation coefficient. For GWAS features correlation **(b)** a subset of features is shown: conservation and eight other features (using logfold and the absolute difference of the probability).

3.5 HyperSMURF Performance Results

This section shows the main results of the ML algorithm hyperSMURF by applying it to the Mendelian, GWAS and eQTL data described in the previous Section 3.3. First, in Section 3.5.1 I explore the different parameters of hyperSMURF carefully and suggest standard settings as well as optimal parameters from automatic tuning experiments on the Mendelian data. Further, Section 3.5.2 will explore the performance of different functional elements within the Mendelian data. An important part is Section 3.5.3 where I compare the performance of retrained popular non-coding methods to hyperSMURF using GWAS and Mendelian data. Finally, Section 3.5.4 compares the performance results from Budach *et al.* [116] to hyperSMURF on the eQTL data.

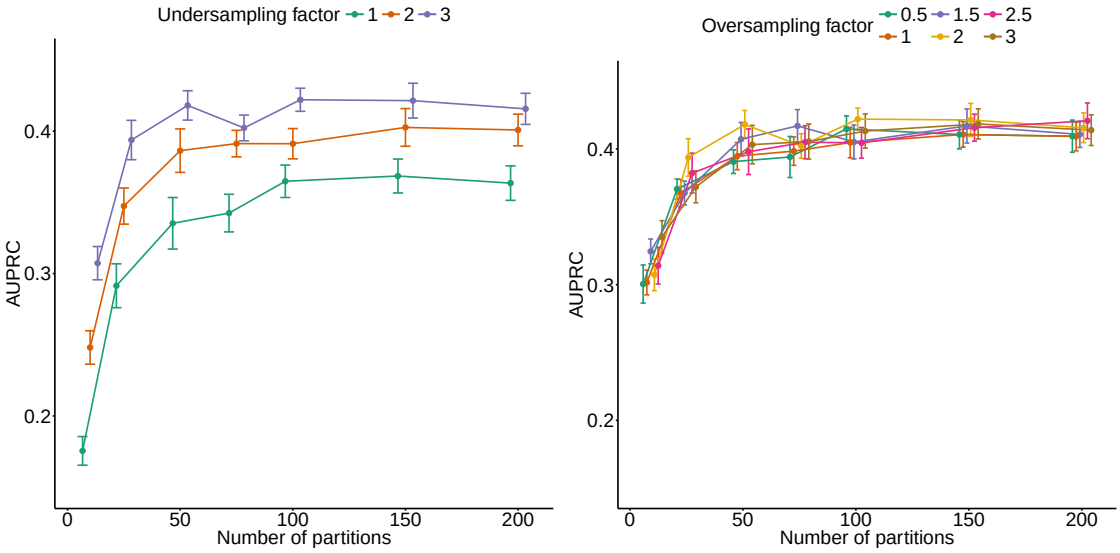
3.5.1 Optimal Parameters

To provide an idea about the impact of the different learning parameters on the overall hyperSMURF performance, an automatic tuning of parameters, described in Section 3.3.1, was performed with the Mendelian data. Figures 3.3, B.1 and B.2 show that with an increasing number of partitions n, the performance in terms of AUPRC increases, as expected, because a larger space of the negative variants is explored and the larger size of the hyper-ensemble provides more accurate and reliable results. Nevertheless we have a steep boost until $n = 100$ and for larger numbers of partitions the improvement is marginal. In terms of runtime this means that we can safely use $n = 100$ to obtain reasonably good performance.

In addition Figures 3.3a and B.2 show that the best selection of the undersampling factor m within a partition is important. The curves corresponding to $m = 3$ lie steadily over the other curves with lower undersampling ratio and show on average an AUPRC increment of about 0.05 independently of the selected oversampling factor o.

The oversampling factor o contributes to the overall performance too (Figures 3.3a and B.2) even if the improvement is less significant than that of the undersampling ratio. In contrast to that, the size of the k nearest neighbors used by SMOTE seems to play no performance role on the tested neighbor sizes $k = \{3, 4, 5, 6, 7, 8, 9, 10\}$. Here, Figure 3.4b shows at all neighbors sizes similar performance in terms of the AUPRC at the complete range of tested oversampling factors. Later we will see that using the optimized k per fold has the same performance as using the standard value $k = 5$ on the Mendelian data.

The learning parameters of the RF also play a role in the performance of the hyper-ensemble. Figure 3.4a shows that increasing the forest size leads to better performances, but the AUPRC curve reaches saturation for forest sizes larger than $t = 20$. Therefore there is no need to increase runtime by increasing the forest size. Moreover, the number of random selected features d is an important learning parameter because Figure 3.4a shows that the best curves correspond to $d = 5$ and $d = 6$. A relatively large decrement in performance can be registered when a suboptimal number of features d was selected. These results reveal that we can obtain good performances on Mendelian data with the standard settings listed in Table 3.2, while keeping the runtime to a minimum.



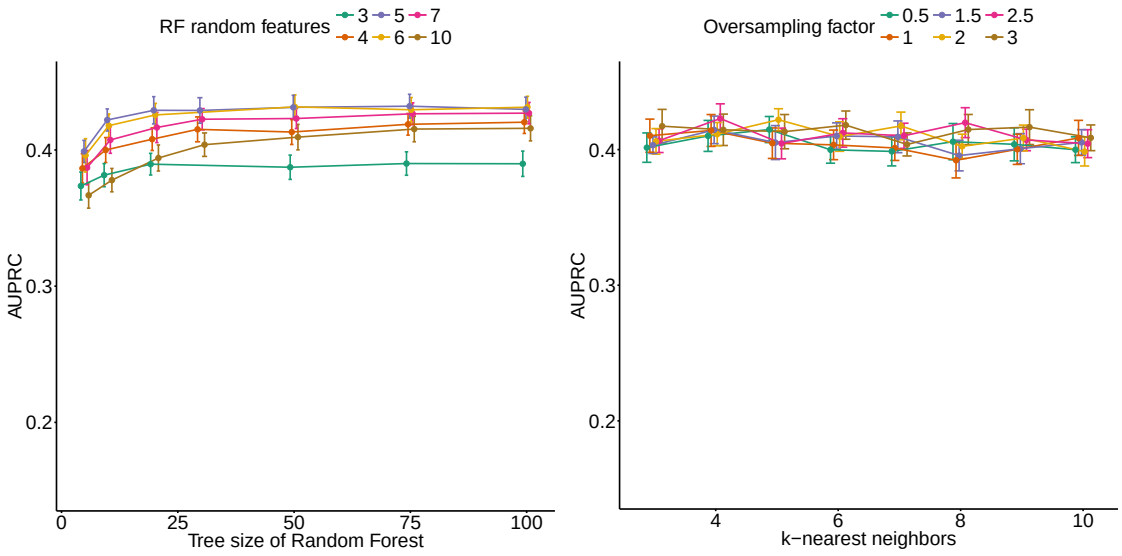
(a) Oversampling factor $o = 2$.

(b) Undersampling factor $m = 3$.

Figure 3.3: HyperSMURF parameter tuning. AUPRC as a function of different hyperSMURF partition sizes generated by internal 9-fold cytogenetic band-aware CV using Mendelian data. Error bars represent the standard deviation between ten repetitions of internal 9-fold CV using different folds. **(a)** Curves represent different undersampling factors with a fixed oversampling factor $o = 2$ (Figures with oversampling factor $o = 0.5$, $o = 1$, $o = 1.5$, $o = 2.5$, and $o = 3$ can be found in Figure B.2). The oversampling factor is the ratio of synthetic positive examples generated through the SMOTE algorithm with respect to the available number of positive examples (see Section 3.3.1 for details). **(b)** Curves represent different oversampling factors with a fixed undersampling factor $m = 3$ (Figures with undersampling factor $m = 1$ and $m = 2$ can be found in Figure B.1). The undersampling factor is the ratio of negative examples with respect to positives. Negative examples were randomly sampled without replacement from each partition of the data (see Section 3.3.1 for details).

Table 3.2: HyperSMURF default parameters. Default values and sets of parameter values explored for the automatic tuning of hyperSMURF. The item "Random tree features" refers to the number of randomly selected features at each step of the construction of the inductive trees that constitute the base learners of the RF. More details about the parameters of hyperSMURF can be found in Section 3.1. For random tree features the standard is always set to $d = \lfloor\sqrt{|\mathbf{x}|}\rfloor$ where $\mathbf{x}$ is a vector of all training features. Therefore the random tree features of $d = 5$ marked by an asterisk $*$ holds only for the Mendelian data because it has 26 features. The equation results in $d = 4$ for the eQTL data. On the GWAS data d is increased to $d = 100$ to raise the probability to include a conservation score in a random tree (see Section 3.4).

Parameter	Description	Default	Parameter Values for Optimization
n	Number of partitions	100	$10, 25, 50, 75, 100, 150, 200$
o	SMOTE oversampling factor	2	$0.5, 1, 1.5, 2, 2.5, 3$
k	SMOTE k-nearest neighbor	5	$3, 4, 5, 6, 7, 8, 9, 10$
m	Undersampling factor	3	$1, 2, 3$
t	Forest size	10	$5, 10, 20, 30, 50, 75, 100$
d	Random tree features	5^*	$3, 4, 5, 6, 7, 10$



(a) RF parameter tuning.

(b) SMOTE k-nearest neighbors parameter tuning.

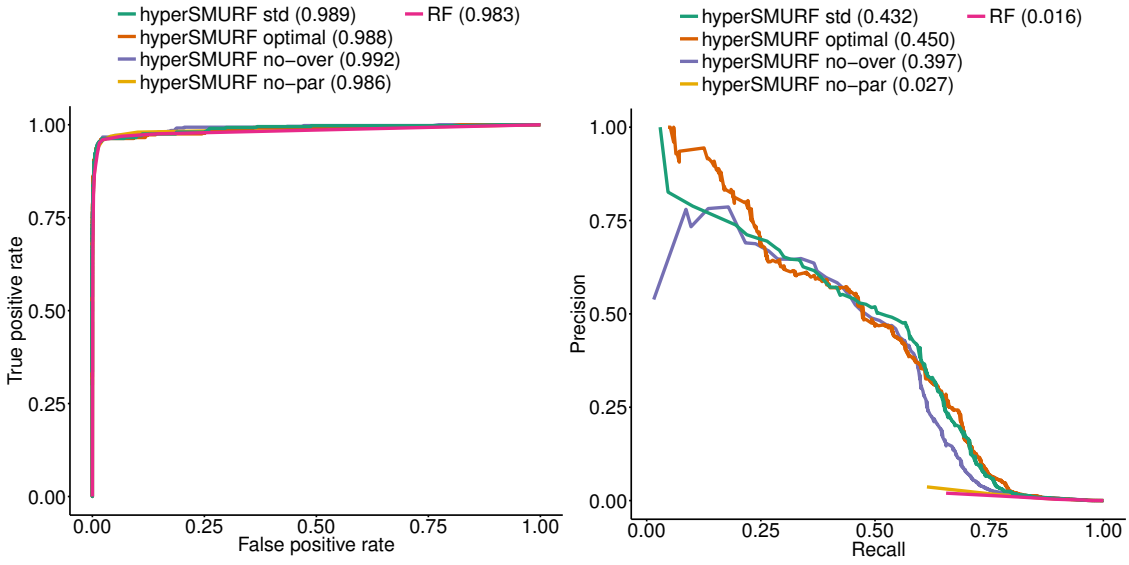
Figure 3.4: RF and SMOTE k-nearest neighbors parameter tuning. (a) AUPRC as a function of different RF sizes (number of decision trees) trained by internal 9-fold cytogenetic band-aware CV. The curves represent the number of random features $d = \{3, 4, 5, 6, 7, 10\}$ used to construct each decision tree of the RF. **(b)** AUPRC as a function of different k-nearest neighbors trained by internal 9-fold cytogenetic band-aware CV. The curves represent different oversampling factors $o = \{0.5, 1, 1.5, 2, 2.5, 3\}$ for SMOTE. All not described hyperSMURF parameters were set to standard values (Table 3.2). Error bars represent the standard deviation between ten repetitions of internal 9-fold CV using different folds.

To show how the different methods within hyperSMURF interact and contribute to the final result, only parts of hyperSMURF were run on the Mendelian data and the performance was compared to the standard settings. A complete overview of the performance result is shown in Table 3.3, and PR along with ROC curves are shown in Figure 3.5. All of these results show that each component of the algorithm (and in particular the hyper-ensemble approach) plays a key role in improving the performance of the method. Comparison with a standard RF (that constitutes the base learner of hyperSMURF) shows the huge improvements obtained by hyperSMURF with respect to this standard state-of-art ML algorithm.

Table 3.3: Impact of hyperSMURF components on its overall performance. AUROC, AUPRC and AUROC of the top 100, 500, and 1000 variants(AUROC_{100}, AUROC_{500} and AUROC_{1000}) with the Mendelian data. "HyperSMURF std" is the full hyperSMURF algorithm; "HyperSMURF optimal" uses the best automated selected parameters for every CV step (see Section 3.3.1); "HyperSMURF no-over" is hyperSMURF with no oversampling; "HyperSMURF no-par" is hyperSMURF with no partitioning and therefore no hyper-ensemble approach. "RF" is the classical Random Forest ensemble. In every setting a subsampling of the majority (negative) class to three times the cardinality of the minority class (positives) was performed. Other parameters were set to default (see Table 3.2). Every training was done 100 times with a different pseudorandom number generator by choosing a different seed. The mean AUCs and their standard deviation are shown here.

Algorithm	**AUPRC**	**AUROC**	**AUROC_{100}**	**AUROC_{500}**
HyperSMURF std	0.43 ±0.02	0.99 ±0.00	0.67 ±0.05	0.79 ±0.01
HyperSMURF optimal	0.45 ±0.01	0.99 ±0.00	0.69 ±0.03	0.78 ±0.01
HyperSMURF no-over	0.40 ±0.02	0.99 ±0.00	0.66 ±0.05	0.76 ±0.02
HyperSMURF no-par	0.03 ±0.00	0.98 ±0.00	0.00	0.14 ±0.23
RF	0.02 ±0.00	0.99 ±0.00	0.00	0.03 ±0.11

Finally, it is possible to use the best parameter setting in terms of the highest AUPRC generated by the internal 9-fold CV in a final and optimal training of a cytoband-aware 10-fold CV. Of course this is possible because the internal 9-fold CV uses only the training dataset and the test dataset remains untouched. The optimal settings of hyperSMURF can be found in Table B.3. To show the increase of performance the cytoband-aware 10-fold CV was repeated 100 times using the standard and the optimal hyperSMURF (Table 3.2). In every round a different pseudorandom number generator by choosing a different seed was used to get an average performance that is not dependent on the chain of the chosen pseudorandom number generator. The standard hyperSMURF achieved an AUPRC of 0.4319 ± 0.0189, while the optimized algorithm an AUPRC of 0.4503 ± 0.0189. This difference is significant according to the Wilcoxon rank sum test (p-value $< 2.2 \cdot 10^{-16}$). PR and ROC curves of the repeated experiment are visualized in Figure 3.5.



(a) ROC curves. **(b)** PR curves.

Figure 3.5: ROC and PR curves of the standard, optimal and only parts of hyperSMURF. "HyperSMURF std" is the full hyperSMURF algorithm; "HyperSMURF optimal" uses the best automated selected parameters for every CV step (see Section 3.3.1); "HyperSMURF no-over" is hyperSMURF with no oversampling; "HyperSMURF no-par" is hyperSMURF with no partitioning and therefore no hyper-ensemble approach. "RF" is the classical Random Forest ensemble. Here, only the experiment which has the closest AUPRC to the mean of the 100 times repeated experiment per group is shown.

The optimized k nearest neighbors of SMOTE have no contribution to the performance. When using the default of $k = 5$ the AUPRC performance is 0.4509 ± 0.0105. This is slightly better than using the optimized k. But the difference is not significant according to the Wilcoxon rank sum test (p-value $= 1$).

Figure 3.5 is an excellent example to show that the ROC curve is not at all informative on imbalanced datasets. There is no significant difference between the ROC curves and their AUROCs in Figure 3.5a. But in Figure 3.5b the PR curves and their AUPRCs are different, especially if we compare the RF approach with hyperSMURF.

3.5.2 Performance on genomically close Variations

Positive and negative variations that are in similar regions of the genome might have similar features. Therefore it can be difficult to separate them properly by the classifier. To see the performance on close variation Section 3.3 introduces the topologically-aware CV where different windows around positives were set and only negatives within the resulting regions were chosen. This approach also reduces the imbalance between the two classes because only a small fraction of negatives was selected. Now the imbalance ranges from ∼1:80 to ∼1:3000, dependent on which dataset or which approach is used (see Table 3.4). So the default parameters in Table 3.2 have to be adjusted.

For the Mendelian data, only the 100 Kb negative selection has a low imbalance of ∼1:300 and with default settings of partitioning and oversampling we get a balanced dataset in every partition. Consequently no undersampling is needed and the parameter is set to $m = 1$. For training the GWAS data sets I reduced the partitioning to $n = 50$ in all four experiments, changed the oversampling to $o = 1$ in the 100, 500 Kb and TAD selection, and finally used $m = 1$ for undersampling in the 100 Kb experiment. In theory other combinations of parameter settings can be used to balance the dataset. Also an internal CV can be done to select the best parameters. But in this analysis it is sufficient to have a good performance rather than optimal, because I want to demonstrate that hyperSMURF also works on genomically close variations.

Table 3.4 shows the imbalance, the resulting number of folds and the performance in terms of the AUROC and AUPRC of the different experiments. The results clearly demonstrate that hyperSMURF is able to distinguish disease variants genomically close to non-disease variants. The AUPRC achieved performances are even better than those obtained with the experimental setup over all negatives by the cytoband-aware 10-fold CV. The main reason for this is the reduced imbalance of the new experiments. But also the increased number of folds in the CV leads to better results, because more training data is available to test the remaining fold.

Table 3.4: Performance of hyperSMURF with different negative selection strategies. The first column represents the size of the "genomic window" used to select negatives around each positive or the "TAD-based" negative selection strategy; the second column reports the imbalance between positive and negative examples; the third is the number of folds of the topologically-aware CV, while the last two columns show the estimated AUPRC and AUROC. The experiment was done with **(a)** Mendelian and **(b)** GWAS data.

(a) Mendelian data

Negatives Selection	Imbalance Ratio	Folds	AUPRC	AUROC
±100 Kb	1:302	116	0.71	0.98
±500 Kb	1:1432	116	0.63	0.98
±1000 Kb	1:2765	111	0.62	0.98
TAD	1:1406	125	0.61	0.98

(b) GWAS data

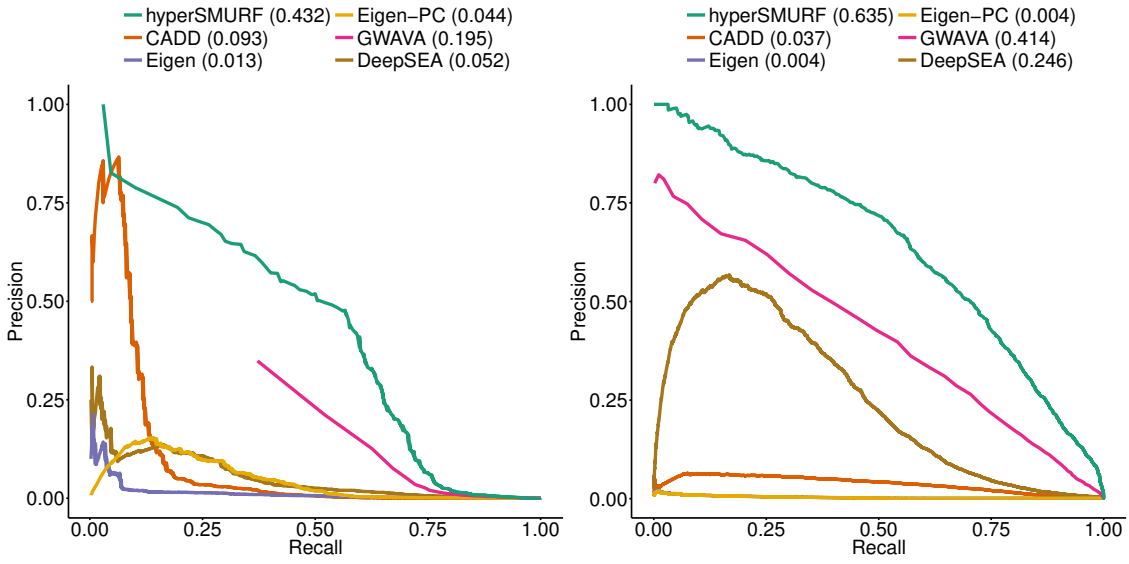
Negatives Selection	Imbalance Ratio	Folds	AUPRC	AUROC
±100 Kb	1:80	1402	0.6488	0.9840
±500 Kb	1:277	723	0.4796	0.9841
±1000 Kb	1:409	413	0.4213	0.9851
TAD	1:269	1196	0.4792	0.9838

These experiments show that hyperSMURF is able to distinguish between close variations. But it should be wisely considered wether a classifier is trained or tested on new or unknown variations. If the goal is a general classifier for the whole genome then it might be useful to include negatives

from unknown regions. Here I would strongly recommend the overall approach. If some genes or regions are of special interest where already known regulatory variations are present, the topologically-aware CV strategy can be of benefit because it might distinguish better between close variants.

3.5.3 State-of-the-art Methods Performance

As described in Section 3.3.2, the performance of hyperSMURF was compared to the retrained methods of CADD, GWAVA, Eigen, Eigen-PC and DeepSEA [33, 34, 36, 38] using the Mendelian and GWAS data. Different metrics were used to compare hyperSMURF with the other methods. The standard metrics AUROC, AUPRC, precision, recall, and F-measure at different score threshold levels as well as an analysis of the distribution of top-ranked variants associated with traits or diseases were applied.



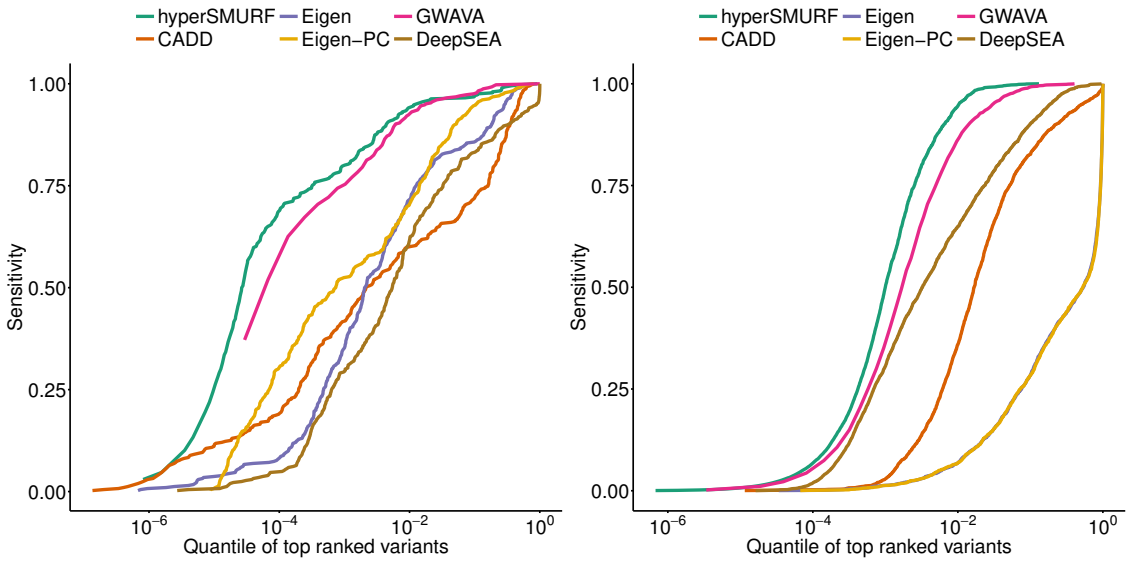
(a) Mendelian data precision-recall (PR) curves.

(b) GWAS data precision-recall (PR) curves.

Figure 3.6: Comparison of the PR curves across methods. **(a)** PR curves using the Mendelian data. **(b)** PR curves with GWAS data. Numbers in parentheses represent AUPRC values of the PR curves.

ROC and PR Curves The overall results in terms of PR curves, presented in Figure 3.6, show that hyperSMURF achieves significantly better results than the retrained state-of-the-art methods with both data, Mendelian and GWAS. At all recall levels hyperSMURF achieves larger values of precision and the AUPRC is significantly larger than with other methods. These results are confirmed by the analysis of the ROC curves in Figure B.4 and the sensitivity of the methods as a function of the quantiles of top ranked variants in Figure 3.7. Although the AUROCs between the methods are similar, the difference between the ROC curve of hyperSMURF compared to CADD and DeepSEA was statistically significant according to the DeLong test (significance level $\alpha = 0.05$; compare with Table B.5a, [119]). But it is well-known that with imbalanced data the AUPRC is more informative than the AUROC [7, 24].

Focusing on the retrained learners the imbalance-aware strategy of GWAVA works in both settings better than the other imbalance-unaware learners. Interestingly the third best method is different for the GWAS and the Mendelian data. For Mendelian the SVM of the CADD method scores reasonably well on low recall levels but does not display good performance on the GWAS data. In contrast to that, the DeepSEA approach has no good performance on Mendelian data but shows a reasonable performance on the GWAS data. These observations might result from the features that were used in both sets. The SVM seems to score better on smaller feature sizes. The convolutional networks from DeepSEA have a better performance on large features and the feature set of the GWAS data itself is generated directly for the DeepSEA method using deep convolutional networks.



(a) Mendelian data.

(b) GWAS data.

Figure 3.7: Comparison of the sensitivity across methods with respect to the quantiles of top ranked variants. **(a)** Sensitivity with respect to the quantiles of top ranked variants with Mendelian data. **(b)** Sensitivity with respect to the quantiles of top ranked variants with the regulatory GWAS hits. X-axis is $\log_{10}$ transformed.

Distribution of top ranked Positive Variants The curves in Figure 3.7 represent the sensitivity as a function of the quantile of top ranked positive variants. For instance a sensitivity equal to 0.1 at a quantile 10^{-3} means that one tenth of the positive variants lie in the top 0.001 quantile, the first top ranked thousandth of the overall variants. For example in Figure 3.6a the curve of hyperSMURF has a sensitivity equal to about 0.75 at a quantile 10^{-4}. This shows that about three quarters of the Mendelian mutations in non-coding regions are scored in the top ten thousandth of the overall variants. Therefore 300 Mendelian mutations are within the top scored 1700 of the over 14 million genetic variants analyzed. Similar results are achieved for the GWAS data shown in Figure 3.7b. Here, hyperSMURF gains a sensitivity of 0.75 at a quantile level of 10^{-3}. So we have around 1500 regulatory GWAS hits within the top scored 2900 of around 1.4 million variants.

Thus, these curves reflect the ability of the different methods to score the positive variants in the top positions of the ranking. From this standpoint the curves show that imbalance-aware methods achieve the best results because the curves of hyperSMURF and GWAVA lie significantly above the curves of the other imbalance-unaware methods with both analyzed datasets (compare with Figure 3.7). The vote for the third best method is similar as in the ROC and PR curves. DeepSEA works reasonably well on the GWAS data. For the Mendelian data no clear third best method is detectable.

Precision, Recall, and F-Score Results For a better understanding of the prediction behavior of non-coding common and rare disease associated variants of the different retrained methods the precision, recall, and F-score as a function of the score predicted by hyperSMURF and the other methods were inspected and visualized. In sum, hyperSMURF shows the best F-score for Mendelian and GWAS data. This is mainly the consequence of a better precision for hyperSMURF, while maintaining a recall comparable with the other methods. The second best method is GWAVA possibly because it is the only one among the compared methods that explicitly adopts imbalance-aware learning techniques. Detailed precision, recall and F-score results as a function of the score thresholds are presented in Figures B.5, B.7, B.8 and B.10 and are comprehensively discussed in Appendix B.

Table 3.6: Comparison of imbalance-unaware and imbalance-aware methods with progressively imbalanced data. Results of imbalance-unaware (SVM of CADD) and imbalance-aware (hyperSMURF and modified RF of GWAVA) learners in terms of AUPRC and AUROC using cytoband-aware 10-fold CV with Mendelian data. Imbalance ratio represents the ratio between positive examples and negatives using the Mendelian data.

Imbalance Ratio	AUPRC			AUROC		
	CADD	GWAVA	hyperSMURF	CADD	GWAVA	hyperSMURF
1	0.89	0.99	0.99	0.87	0.99	0.99
0.1	0.79	0.96	0.96	0.89	0.99	0.99
0.01	0.52	0.88	0.90	0.89	0.99	0.99
0.001	0.23	0.71	0.78	0.85	0.99	0.99

Progressively Imbalanced Data To show how imbalance-aware and imbalance-unaware methods perform on different imbalance levels, negative variants of the Mendelian data were subsampled to four different subsets with imbalance levels of 1:1, 1:10, 1:100, and 1:1000. Then a cytoband-aware 10-fold CV training was done using the imbalance-aware methods hyperSMURF and GWAVA, and the imbalance-unaware SVM classifier of CADD.

The results in Table 3.6 show that all methods have a good AUROC on all imbalance levels. But as described in Section 2.2.5 the ROC statistics are not convincing in imbalanced datasets. For the imbalance-unaware SVM the AUPRC reduces significantly when the imbalance is increased. In these settings AUPRC of the GWAVA and hyperSMURF methods decreases less dramatically.

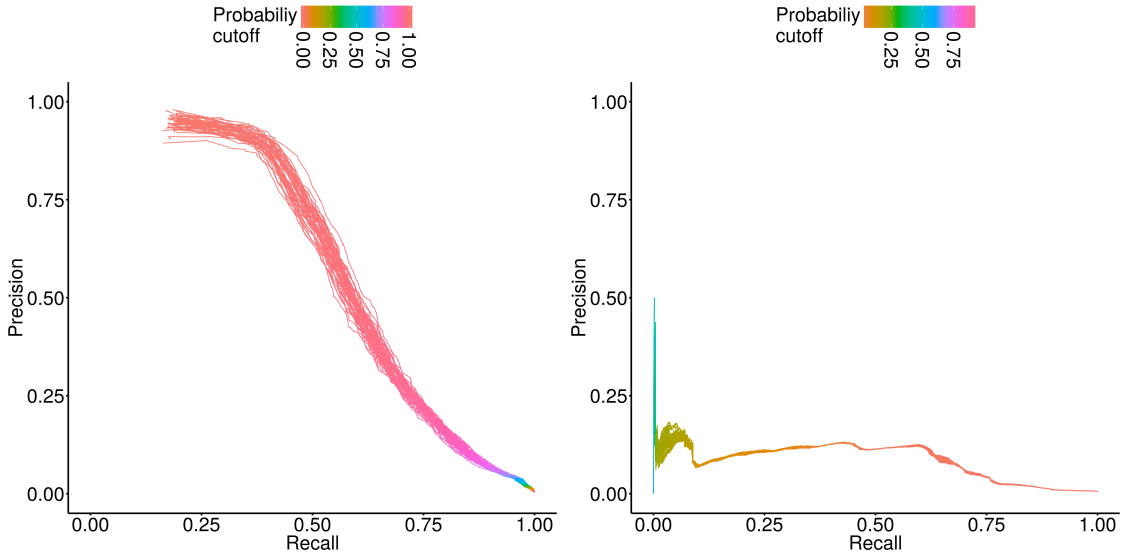
With 1:1000 they are still larger than 0.7. Comparing hyperSMURF and GWAVA it seems that hyperSMURF performs slightly better than GWAVA. This small performance boost can be due to the SMOTE oversampling because it is, of course, not present in GWAVA and there is a similar increase of around 0.05 of the AUPRC in other experiments, like the automated parameter selection in Figure 3.3b or the performances of the different components in hyperSMURF shown in Table 3.3.

3.5.4 eQTL Performance

The imbalance of the eQTL dataset is with ∼1:400 not as high as in the Mendelian dataset (see Section 3.2.1) and therefore the parameters of the hyperSMURF algorithm must be carefully chosen. I used the standard parameters of hyperSMURF already used in Mendelian training (see Section 3.2.1) but without undersampling of the majority class (see step (iii) in Algorithm 3.1). Standard values of hyperSMURF are in Table 3.2. This leads to a theoretical imbalance of ∼1:1.39 in every partition to train the RF. Further parameter search was not performed because of high performance on the standard settings.

Repetitive training of the eQTL dataset resulted in an average AUROC of 0.9860 ± 0.0010 for hyperSMURF and 0.9353 ± 0.0005 for the logistic regression model. The average AUPRC is 0.5837 ± 0.0109 for hyperSMURF and 0.0845 ± 0.0007 for the logistic regression model showing clearly that hyperSMURF has superior results on the given imbalance problem. The PR curves of all 50 repetitions are visualized in Figure 3.8 together with the probability cutoff (same results for ROC curves are in Figure B.11). Both methods have stable shapes of curves and stable AUCs. But there is a huge improvement in terms of the PR curve with hyperSMURF. Around 25 % of positives are at the top rank and more than 60 % of all positives have a precision over 0.5 using hyperSMURF. The regression has no significant precision at any recall. In addition the probability value seems to scale perfectly on the hyperSMURF approach (compare the probability cutoff in Figures 3.8 and B.11). Most positives have a probability larger than 0.9 (see PR curves in Figure 3.8) but in general the probability is nicely distributed over all data (see the ROC curves in Figure B.11). Therefore a reliable cutoff can be used to find new eQTLs.

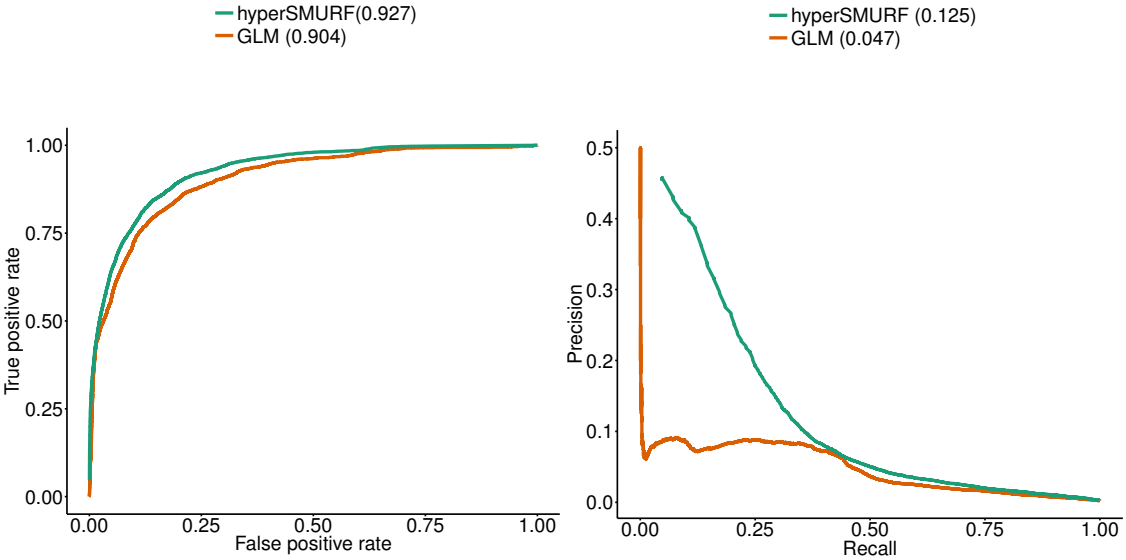
The PR and ROC curve performance of miRNA-relation CV are shown in Figure 3.9. ROC and PR statistics show lower performance than in the previous test where a random split of training and test data was used (experiment was repeated 50 times). Especially the AUPRC is dramatically reduced. This demonstrates that SNPs from the same miRNA tend to have similar characteristics and therefore boost the performance if they were divided in training and test. It might be valid to do so if the goal is to identify new eQTLs of miRNAs that are included in the data. But it will perform poorly on other miRNAs that are not in the training set or on trained miRNAs that do not have any positives included in the training set. According to the Budach *et al.* [116] training set the latter applies for the vast majority (92 %) of all miRNAs. In this case the approach of the miRNA-relation CV helps to give an unbiased performance for these and other unknown miRNAs.



(a) PR curves of hyperSMURF.

(b) PR curves of logistic linear regression.

Figure 3.8: Performance measurement of the eQTL data using splits. PR curves generated by sampling 50-times using random splits with three quarters for training and remaining instances for testing. The color of the line shows the probability at the given recall value. Training was done with **(a)** hyperSMURF and **(b)** a logistic linear regression model.



(a) ROC curves on eQTL data using CV.

(b) PR curves on eQTL data using CV.

Figure 3.9: Performance measurement of the eQTL data using 10-fold CV. PR and ROC curves using the miRNA-relation CV of the eQTL data (see Section 3.2.3). HyperSMURF was used with standard settings (see Table 3.2) without undersampling. The logistic regression model (GLM) was configured as described by Budach *et al.* [116] using all data and the special CV setting. Values in brackets after the algorithm show the AUC.

3.6 Discussion and Chapter Conclusion

In this chapter I introduced hyperSMURF, a ML algorithm that is specifically designed to handle large (genetic) imbalanced data. HyperSMURF uses over- and undersampling techniques as well as ensemble methods (hyper-ensemble) to generate small balanced subsets. Two software implementations, one in R and one in Java, are freely available together with an extended manual and tutorial (`https://hypersmurf.readthedocs.io` and Appendix E).

HyperSMURF was used with three different imbalanced datasets: a Mendelian dataset to predict regulatory NCVs, a GWAS dataset for the prediction of regulatory GWAS hits and finally, eQTL data to determine the relationship between functional elements and miRNA regulation. Mendelian and GWAS data have similar negatives, but of course both datasets have different positives and also different features were used. Especially the GWAS dataset contains a huge number of features (1838), most of them are chromatin effect scores generated by DeepSEA. But the univariate regression analysis in Section 3.4 suggests that only the conservation scores are predictive by themselves. Therefore a high number of random features per tree was used to increase the likelihood of selecting a conservation score.

An extensive unbiased parameter search was done with the Mendelian data to figure out the behavior of hyperSMURF and the contribution of its different components. The presented unbiased approach to find the best parameter combinations showed that the performance increases when good settings are chosen. Nevertheless one has to weigh up the performance with the runtime. Thus a standard hyperSMURF setting was determined with the genetic data in Table 3.2, which keeps the runtime to a minimum while still guaranteeing a good performance.

In addition experimental results showed that hyperSMURF is able to distinguish disease variants that are genomically close to non-disease variants: AUPRC results achieved with this experimental setting are sometimes also better than those obtained with the experimental set-up that includes all the negatives and the overall ranking of all positives versus all negatives. This is not surprising because "TAD-aware" and "genomic-window-aware" experimental settings lead to less imbalanced prediction tasks.

Compared to other ML methods used in NCVs prediction, hyperSMURF showed on all three genetic datasets a better performance than the other retrained learners. This is due to the fact that hyperSMURF is the only method explicitly designed for high imbalanced data. GWAVA, which also has an implemented imbalance-aware strategy, has consequently the second best performance. It can handle imbalanced data to some extent. But with increasing imbalance the performance in terms of the AUPRC declines faster (see Table 3.6).

Finally the results of the eQTL data show how important it is to think about imbalance-aware strategies in the context of predicting consequences of genetic data. This example was not constructed to fit into an imbalanced sample. It was directly taken out of the literature without any modifications. This shows that imbalanced sets are available in the literature and they have to be carefully treated. Of course Budach *et al.* [116] achieved an accuracy of 85 %, but they avoided the imbalance by sampling balanced subsets of the two classes for model selection and testing. Assuming that they wanted to test how good their features are, it is a reasonable approach. But if they wanted to use this training as a basis to determine new relationships between functional

elements and miRNA regulation, the training on the whole imbalanced data should be used together with an imbalance-aware classifier.

To conclude, the results showed that imbalance-aware machine learning strategies can successfully counteract the bias towards the majority class. Imbalance-aware strategies are essential for predicting disease-associated NCVs in genomic contexts because these datasets are characterized by a large imbalance between the known small set of available disease-associated variants and the huge set of all human genetic variations. HyperSMURF implements robust strategies using over- and undersampling techniques to handle imbalanced data. So the next step will be to use the insights we gained in this chapter about hyperSMURF, features and the genetic data to generate a new genome-wide score of any position in the human genome. Thus the next chapter will introduce a new genome-wide NCVs pathogenicity score for Mendelian disease. This score will be integrated in a complete framework, which analyzes variants from WGS together with phenotypes to find the causative mutation out of thousands of variants.

Chapter 4

Regulatory Variants

I developed the Regulatory Mendelian Mutation score (ReMM score) for pathogenicity prediction of NCVs in collaboration with Giorgio Valentini and Peter N. Robinson. The Genomiser framework was developed mainly in collaboration with Damian Smedley, Jules OB. Jacobsen and Peter N. Robinson. The initial performance measurements of Genomiser and Phen-Gen were done by Damian Smedley. I presented the Genomiser at the Annual Meeting of the German Society of Human Genetics 2016 and at the Annual Meeting of the Arbeitsgemeinschaft für Gen-Diagnostik e.V. (AGD) 2016. Genomiser and ReMM score are published together in the American Journal of Human Genetics [56].

D. Smedley, M. Schubach, J. OB. Jacobsen, S. Köhler, T. Zemojtel, M. Spielmann, M. Jäger. H. Hochheiser, N. L. Washington, J. A. McMurry, M. A. Haendel, C, J. Mungall, S. E. Lewis, T. Groza, G. Valentini, and P. N. Robinson. (2016). A Whole-Genome Analysis Framework for Effective Identification of Pathogenic Regulatory Variants in Mendelian Disease. *The American Journal of Human Genetics*, 99(3), 595–606.

The underlying framework of Genomiser is Exomiser. The development of Exomiser was lead by Damian Smedley and Peter N. Robinson and we published it in Nature protocols [55]. An important component of Exomiser is Jannovar which was originally published in 2014 by Jäger et al. *[109]. After that Manuel Holtgrewe and I developed Jannovar further.*

D. Smedley, J. OB. Jacobsen, M. Jäger, S. Köhler, M. Holtgrewe, M. Schubach, E. Siragusa, T. Zemojtel, O. J. Buske, N. L. Washington, W. P. Bone, M. A. Haendel, and P. N. Robinson. (2015). Next-generation diagnostics and disease-gene discovery with the Exomiser. *Nature protocols*, 10(12), 2004–15.

From the previous Chapter 3 we learned that with imbalance-aware ML methods, like hyperSMURF, it is possible to predict regulatory mutations out of thousands of non-deleterious variants

with high confidence. Because WGS is now affordable such methods are highly needed to find the needle in the haystack: the pathogenic mutation within thousands of benign variants. For Mendelian diseases common approaches, tools and standard diagnostics are focused on coding regions. This might introduce a long-standing observational bias toward coding sequences. The *regulatory code* that determines whether and how a given genetic variant affects the function of a regulatory element remains poorly understood for most classes of regulatory variation. Thus, given our lack of understanding and tooling, it is not surprising that so far very few disease-causing NCVs have been identified as causal in Mendelian disease. To address this gap we need on the one side a regulatory pathogenicity score and on the other side a fast framework that effectively finds the causative coding or non-coding mutation.

This chapter will introduce both: first, an integrative algorithm for ranking NCVs in WGS data and second, a ML method for scoring NCVs. The ML method rates each position of the non-coding genome based on predicted pathogenicity in Mendelian diseases using hyperSMURF and the previously introduced Mendelian data (see Sections 3.1 and 3.2.1). The integrative algorithm factors in multiple inputs: (1) phenotypes, (2) variants in coding regions, (3) variants in non-coding regions, and (4) existing published gene-phenotype associations.

Structure of this Chapter In this chapter I will present at the beginning, in Section 4.1, the framework Exomiser, an analysis framework that is able to score coding variants and associate them to specific Mendelian diseases. Then I will explain in Section 4.2 Genomiser, a further development of Exomiser, which extends the framework to NCVs. This analysis framework is able to not only score the relevance of variation in the non-coding genome, but also to associate regulatory variants to specific Mendelian diseases. In Section 4.3 I explain the ReMM score for relevance prediction of NCVs. Section 4.4 shows the performance of Genomiser by simulated whole-genomes using different genome-wide deleteriousness scores and Genomiser is compared to another genome-wide prioritization tool. Finally, Section 4.5 concludes this chapter.

4.1 Exomiser

Exomiser [55] is an application that filters and prioritizes variants together with genes in NGS projects for novel disease-gene discovery or differential diagnostics of Mendelian diseases. On the variant side Exomiser uses population databases like ExAC [67] to remove frequent variants, Jannovar [109] to find out the corresponding gene and the functional effect within it (see Section 4.1.1), and pathogenicity scores like MutationTaster [8, 9] to assess the potential deleteriousness of the variant. On the phenotypic side Exomiser comprises a suite of algorithms for prioritizing genes that are related to the input phenotype, using random-walk analysis of protein interaction networks, clinical relevance and cross-species phenotype comparison. For this approach Exomiser relies on similarity scores between sets of terms from the Human Phenotype Ontology (HPO). The preliminary Section 2.3 shows a general overview on ontologies and the semantic similarity approach. The HPO, its structure and applications are explained in Section 4.1.2 in more detail.

Now I will briefly explain the filtering and prioritization steps. For variants from WGS the filtering must be extended to NCVs but have to remain fast. So an optimal filter configuration and combination have to be created. Therefore I will take up the filtering approaches later in the Genomiser part, in Section 4.2, and describe them in more detail.

Filtering The variant input of Exomiser are the called or detected variants from an alignment against a reference sequence (hg19/GRCh37) stored in a VCF file, resulting from exome sequencing of a rare disease patient and, optionally, other affected and unaffected family members. Exomiser reduces the variants contained in the VCF file using several criteria and filters. The filtering step is critical in order to reduce the number of variants. For example a typical exome contains around ∼30,000 variants and the filtering step reduces the variants to around 1000 or less.

Functional Filter Exomiser annotates each variant relative to the UCSC hg19 transcript set using the Jannovar software library [109]. In the actual development version 7.3 or 8.0, Exomiser is able to also use RefSeq or Ensembl as transcript database. This annotation describes the location within or between transcripts, the type of variant (e.g. missense, nonsense or intergenic) and the predicted consequence of the variant on the protein-coding sequence. By default, Exomiser removes any variants that are synonymous and non-coding (intergenic, intronic, upstream, downstream or intronic).

Frequency Filter As Exomiser deals with rare genetic diseases, frequent variants can be removed using an upper threshold of the MAF for Exomiser's frequency filter. The population databases 1KG, Exome Server Project (ESP) 6500 and ExAC [59, 67, 68] are included.

Variant Pathogenicity Filter Pathogenicity predictions are assessed via SIFT, MutationTaster and PolyPhen2 [6–9] provided by dbNSFP [51]. During variant pathogenicity filtering variants that were not predicted as pathogenic from one of the three scores can be removed.

Pedigree Filter Finally variants can be filtered on expected inheritance patterns. This step is optional but frequently used because there are some expectations clinicians made out of the pedigree of the affected individual. The following inheritance filters are available: autosomal dominant (AD), autosomal recessive (AR) or X-linked recessive (XR). The approach is slightly different between a single individual or a multisample family-based analysis. Exomiser uses the inheritance annotation from Jannovar [109]. Jannovar together with the inheritance filtering approach will be described in Section 4.1.1 in more detail.

Prioritization The filtering approach effectively reduces the number of variants. However, this typically leaves us with more candidates (up to 1000) than can reasonably be manually assessed. So some sort of ranking by a prioritization algorithm is necessary. The Exomiser suite contains a number of different methods for variant prioritization based on protein-protein interactions and/or phenotype comparisons between a patient and existing human disease databases and model organisms. Exomiser calculates variant-based and method-specific, gene-based scores and combines them using a logistical regression model to generate a final combined score that is used for ranking. Variant scores are a combination of how rare the variant is as observed in

the population databases, together with its predicted pathogenicity. How the gene score is calculated varies, depending on which prioritization method is chosen by the user. Figure 4.1 gives an overview of the prioritization methods which I will describe now briefly:

PhenIX This method should be selected when Exomiser is used in clinical diagnostics. The phenotypic prioritization procedure analyzes only those genes that have been associated with a Mendelian disease [54]. The algorithm uses the semantic similarity approach of Phenomizer [120] for differential diagnostics. The PhenIX algorithm evaluates and ranks variants on the basis of pathogenicity and semantic similarity of patients' phenotypes as described by HPO-terms to those of Mendelian diseases whose molecular etiology has been clarified.

PHIVE This algorithm is a cross-species mouse-human phenotype comparison. The idea behind PHIVE is that if a mouse model exists for the gene containing the disease-associated mutation, then it is likely to exhibit phenotypic similarity to the clinical phenotypes. Here, Exomiser implements the PhenoDigm algorithm [121] to calculate the phenotypic similarity between a patient's clinical signs and symptoms and observed phenotypes in mouse mutants associated with each gene candidate in the exome.

ExomeWalker Mutations in a gene that is part of the same pathway or genes that interact closely with known disease genes may result in similar phenotypic manifestations. Therefore this approach uses protein-protein association networks together with random walks [122] to find associated genes from seeded genes or disease-gene families based on phenotypic similarity [123].

hiPHIVE The human/interactome-PHIVE (hiPHIVE) algorithm extends the PHIVE algorithm to include zebrafish and human phenotype data. For genes that have no phenotype data from any of these sources a random walk is performed in a protein-protein association to score close candidates network to genes with strong phenotypic similarity to the patient.

4.1.1 Jannovar

Jäger *et al.* [109] developed Jannovar, a Java framework for transcript-based annotation and pedigree analysis inspired by the transcript annotation tool ANNOVAR [124]. Jannovar is able to read variants, e.g. stored in a VCF file, and annotates them according to a transcript database. It supports multiple transcript databases and multiple genome releases. Transcript databases are RefSeq, Ensembl, and UCSC as well as user defined transcript databases using mouse (builds mm9 and mm10) and human genomes (builds hg18, hg19/GRCh37 and GRCh38). For definition of variation features Jannovar uses the standard Sequence Ontology (SO). Jannovar is fast, designed in a modular way and well documented so that it can easily be used as software library for variant annotation in other tools like in the Exomiser framework. The software manual is available at the website `https://jannovar.readthedocs.io` and the API documentation at `https://javadoc.io/doc/de.charite.compbio/jannovar-core`).

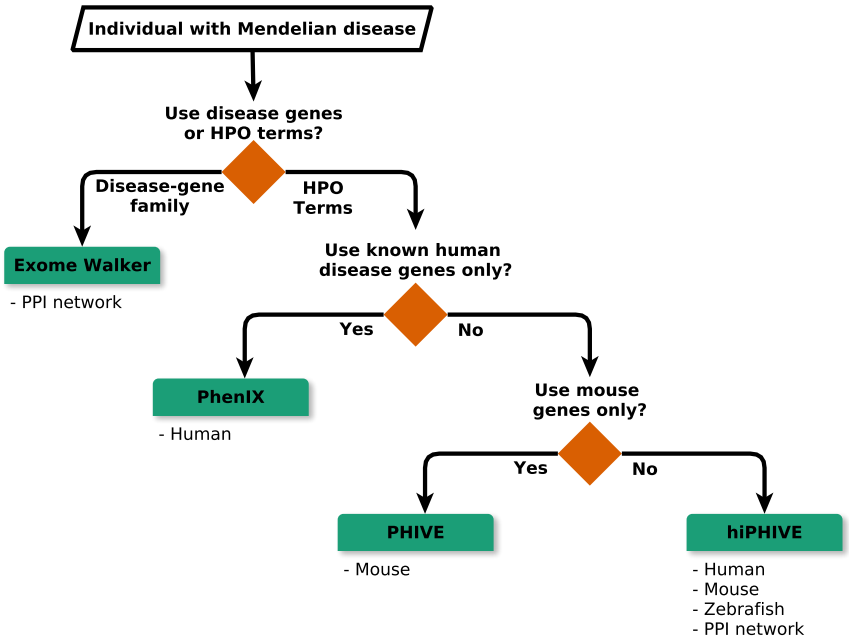


Figure 4.1: Phenotype prioritization algorithms of Exomiser. This decision tree gives an overview about the available phenotype prioritization methods implemented in Exomiser. The paths from the root to the leaves define possible solutions about which method should be selected depending on the goal and the available input.

Inheritance A function of Jannovar beyond the variant feature annotation is the inheritance annotation. It can use nuclear VCFs as well as multi VCFs from a family to find out the inheritance mode of each variant. In Mendelian studies this annotation can be used to effectively filter down variants that only follow an expected inheritance pattern. Exomiser takes the advantage of Jannovar's inheritance annotation to remove variants that are not in concordance with the input inheritance pattern of the user. The inheritance annotation is a rule-based system and it works on single individuals, as well as on small and large family pedigrees. The rule-based system can analyze the following inheritance annotations: autosomal dominant (AD), autosomal recessive (AR), X-linked recessive (XR), and X-linked dominant (XD).

The program will rewrite the genotypes included in the VCF to internal genotype objects. These genotypes can either be `NOCALL` (no genotype was determined for the person), `REF` (homozygous wild-type), `HET` (heterozygous alternative), or `HOM` (homozygous alternative). In general a caller calls a hemizygous mutation as homozygous. Therefore Jannovar uses `HOM` and in addition `HET` to increase the sensitivity as hemizygous on known males. The persons can either be affected, unaffected or their affection state is unknown.

Autosomal dominant (AD) For example this filter can be used to filter for de-novo mutations. If the pedigree contains a nuclear person then the variant call list must contain one `HET` call. If multiple individuals are available every affected person should have a `HET` call. To avoid problems on difficult or uncalled sites it is also possible that some affected or unaffected have a `NOCALL`. But it is important that every unaffected cannot have a `HET` or a `HOM` call at the observed site.

Autosomal recessive (AR) Here it can be either a homozygous or a compound heterozygous mode of inheritance. Homozygous has similar rules for affected as in AD, but all of them should have a `HOM` instead of a `HET` call (some `NOCALL`s are allowed). Every unaffected cannot be `HOM` and there is a special rule defined for the parents. The homozygous variant must be inherited with a `HET` call from both unaffected parents. Thus, parents from affected individuals must be `HET` for this site.

Compound heterozygous rules are more complex. For example the rules must run on a region within a chromosome or, to simplify it, to a list of variants. As default Jannovar uses all variants annotated to the same gene with a moderate (e.g. missense) or higher effect as such a list. In a nuclear family there should be at least two `HET` genotypes in the list. On larger pedigrees Jannovar tries to take the unaffected parents from affected children into account to figure out if two different `HET` variants are not located on the same allele. In particular one variant should be `HET` in the mother and `REF` the father. The other variant can only be `REF` in the mother and `HET` in the father. The rest is straight forward. Every affected should have both variants `HET`. If an unaffected has both variants `HET` then Jannovar tries to find out if they are also on different alleles using the parents (if available). If this holds then both variants will be discarded.

X-linked dominant (XD) All variants must be X-chromosomal and Jannovar uses different rules for males and females. Males can only be `REF` or hemizygous, but variant callers do not distinguish between males and females, so they will stick to their `REF`, `HET` and `HOM` calls. Thus, Jannovar will consider `HET` calls in males as `HOM` to increase the sensitivity. For XD every affected female can only have a `HET` variant and affected males a `HET` or `HOM` variant. For males XD will be a rare case because these mutations are mostly lethal. Unaffected persons can only be `HET` at this site.

X-linked recessive (XR) Here we have the same chromosomal restrictions and male/female differences like in XD. The simplest case is that affected persons are `HOM` (maybe some `NOCALL`) and all others are `HET`, `REF` or `NOCALL`. It is important that on an affected male child with unaffected parents the mutation can only be inherited by the mother. If the affected child is female it must be inherited from both parents but the father must also be affected.

For completeness Jannovar also has rules for XR compound heterozygous. This can only affect females but it is very unlikely because one variant has to be transmitted by an affected father or a de-novo mutation. The latter will be handled by XD rules.

4.1.2 The Human Phenotype Ontology

In 2008 Robinson *et al.* [125] presented the Human Phenotype Ontology (HPO), a controlled, standardized vocabulary for describing entities and the semantic relationships between phenotypic abnormalities encountered in human disease [125–127]. It is widely used for rare diseases but it is also possible to use it for common (complex) disease description. The HPO is organized as four independent subontologies. (1) Phenotypic abnormality, (2) Mode of inheritance with Mendelian or non-Mendelian inheritance modes, (3) Mortality/Aging subontology, and finally, (4) clinical modifier subontology to characterize phenotypic abnormalities, for example with the age of onset. Using this controlled vocabulary a comparison or clustering of phenotypes is easily possible and enables the integration of phenotype information across scientific fields and databases.

HPO is widely used in translational research and genomic medicine [128–130] and it is the standard terminology to describe phenotypic features of affected individuals in Exomiser. All phenotype prioritization algorithms, except ExomeWalker, uses HPO-terms and semantic similarity (Section 2.3.1) to find phenotypic similar genes to an input term query. Here the HPO provides annotation profiles from terms to rare disorders using OMIM, Orphanet and DatabasE of Chromosomal Imbalance and Phenotype in Humans using Ensembl Resources (DECIPHER) for disease entities [131]. In addition the HPO has annotation profiles for common diseases generated with a text-mining approach by analyzing the PubMed corpus [132]. But only the rare disorder profiles and not the common disease annotations are included in Exomiser.

4.2 Genomiser

The Genomiser implementation is based on Exomiser (see Section 4.1, [55]). Genomiser, like Exomiser, ranks variants in the human sequence according to their pathogenicity and contribution to a Mendelian disease. Also the analysis steps of Genomiser are similar to Exomiser (see Figure 4.2 for a schematic overview of the Genomiser steps). The major difference is that Genomiser can filter and rank NCVs in addition to coding variants.

To use the full functionality of Genomiser, the initial input is a VCF from a whole-genome along with phenotypes of the sequenced person, AF cutoff and maybe other configurations. The next steps are designed in a way that large VCF files are processed in a reasonable time frame using a minimal memory. Therefore the initial step is to find the genes with phenotype similarity to the input phenotype (set of HPO-terms). A cutoff for phenotype similarity is used (standard 0.5) to remove unrelated genes. This allows for variants to be directly removed if they do not relate to a phenotypically similar gene while extracting them from the input VCF. All phenotype similarity algorithms of Exomiser/Genomiser are described under Section 4.1 in detail. Now the variants are streamed through different annotation and filtering steps and they will be directly removed if they fail one filtering step. In addition the concept of streaming enables parallel computing, so in theory every variant can be processed in parallel up to the final step, except a compound heterozygous filtering is made by Jannovar.

The next filtering steps are processed per variant. For every variant a lookup of their population frequency is made in population databases and every frequent variant will be removed. The standard used cutoff is 1 %. As population databases Genomiser uses ExAC, ESP 6500 and frequencies from the 1KG [59, 67, 68, 133]. In addition the functional effect of the variants is annotated using Jannovar [109]. According to the position or the sequence effect it will be annotated with different variant scores. For coding missense, nonsense or InDel mutations the maximum of pathogenicity scores of Exomiser are used, namely MutationTaster, Polyphen and SIFT. Like in Exomiser, an additional score of 0.9 for essential splice-sites and 0.95 for nonsense as well as frameshift mutations is added.

For NCVs or synonymous variants Genomiser is able to use any precomputed score on the genome (hg19 release). This can be CADD [33] or Eigen [36] described in Sections 2.4.1 and 2.4.4 or the score described in the next Section 4.3, the ReMM score. Because Eigen is not available on allosomes, it is recommended to use Genomiser with ReMM or CADD to fully explore the genome. In addition, variants that are below a threshold (specific to the pathogenicity score) will be removed (always according to the highest pathogenicity score). It is important to notice that no variant will be removed in terms of its functional class. For example Exomiser will remove 3' or 5' UTRs, synonymous, intronic or intergenic variants.

At the end of the filtering it is possible to use an inheritance filter. This might be useful if more than one family member is sequenced or if a specific inheritance mode is expected for a dominant inheritance with a heterozygous de-novo mutation. The inheritance filter is implemented in Jannovar [109] and is described in Section 4.1.1 in more detail.

Finally, for the remaining variants, the pathogenicity scores and the phenotype score of the corresponding genes are combined to a final Genomiser score. In the final output the Genes with their variants are ranked according to their Genomiser score. Figure 4.2 shows a complete overview of the Genomiser annotation and filtering steps. In the next paragraph I will describe how Genomiser assigns NCVs to a gene.

Target Gene of Non-Coding Variants For NCVs the association of a variant to a gene is not as straight forward as for coding variants. Here, Genomiser takes advantage of the three-dimensional structure of the DNA to find the corresponding gene. This means that enhancers might come to close proximity to a promoter next to a gene using a loop of the DNA strand. Therefore the variant can be far away in terms of bases and multiple genes can lie in between. This is why Genomiser assigns NCVs, in particular distal variants more than 20 Kb away from a gene, to all possible genes within their TAD [134], the segment of the genome that is in close proximity. See Section 1.1 for more information about topological confirmation of the DNA. To be more stringent these variants must reside in predicted enhancers from the FANTOM5 consortium [135] or Ensembl regulatory feature build [136] and are only associated with the most phenotypically similar gene in the TAD. Proximal intergenic, intronic and 5' and 3'UTR variants are associated to the closest gene.

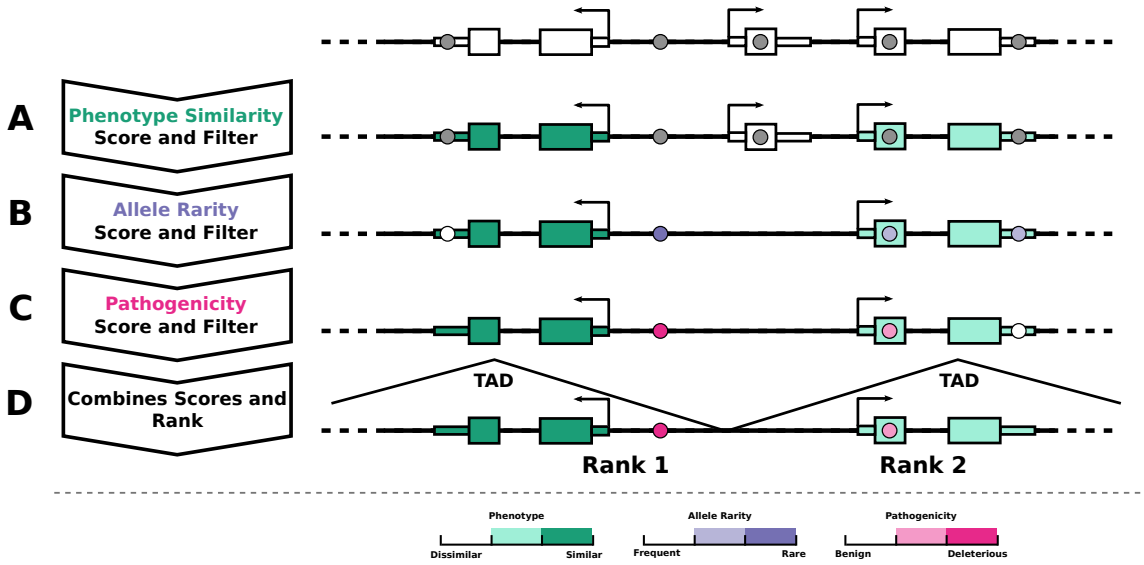


Figure 4.2: Schematic overview of variant processing in Genomiser. Genomiser uses a (multi-sample) VCF, a list of HPO-terms representing the clinical signs and symptoms observed in the individual being investigated by WGS, and optional user parameters that control the filtering and prioritization steps. **A**: Genes are scored and filtered according to their phenotype similarity with the input HPO-terms. Genes with low or no similarity together with their respective variants are removed (gene in the centre). **B**: Variants are annotated with allele frequencies from population databases and frequent variants will are filtered out (variant on the left side). **C**: Pathogenicity scores are assigned to variants and non-deleterious ones are discarded (variant on the right side). **D**: Variant pathogenicity and gene phenotype similarity scores are combined to a final Genomiser score. For NCVs TADs are used to assign the variant to the highest phenotypic similar gene in the same TAD (see Section 1.1 for an introduction into TADs). Finally, genes are ranked according to their Genomiser score (rank 1 left gene, rank 2 right gene).

4.3 Regulatory Mendelian Mutation Score

Genomiser was generated for the specific task of finding coding variants together with regulatory NCVs. For that reason ML techniques for building a scoring model of non-coding Mendelian variants would perform better with a highly reliable training set, consisting of mutations that had been validated by experimentation or co-segregation studies, or for which other convincing evidence of pathogenicity was available. Such catalog did not exist before a detailed and comprehensive biocuration was done from the medical literature to identify experimentally or otherwise validated NCVs (< 35 nucleotides) associated with Mendelian disease. The gained variants are described in the previous chapter under Section 3.2.1 and in Appendix A. In total 453 regulatory variants were identified including 406 SNVs which will be used as positive set. As negatives the same set of non-deleterious variant sites was used, like for hyperSMURF performance testing, derived from differences between the human genome reference sequence and the inferred ancestral primate genome based on the EPO whole-genome alignments of six primate species [40, 42] (see Section 3.2.1 for a detailed description of the 14,755,199 negatives). Also the 26 genomic attributes were used resulting in the same Mendelian data described in Section 3.2.1.

I showed in Section 3.5.3 that the ML method hyperSMURF performs superior to other learners of non-coding scores. Therefore it was chosen as learning strategy using the default hyperSMURF parameters in Table 3.2 as configuration. The combination of the Mendelian data and the hyperSMURF ML method will result in the Regulatory Mendelian Mutation score (ReMM score). This score makes a relevance prediction of regulatory variants, or to be more precise a relevance prediction of positions of the human genome in terms of a Mendelian disease.

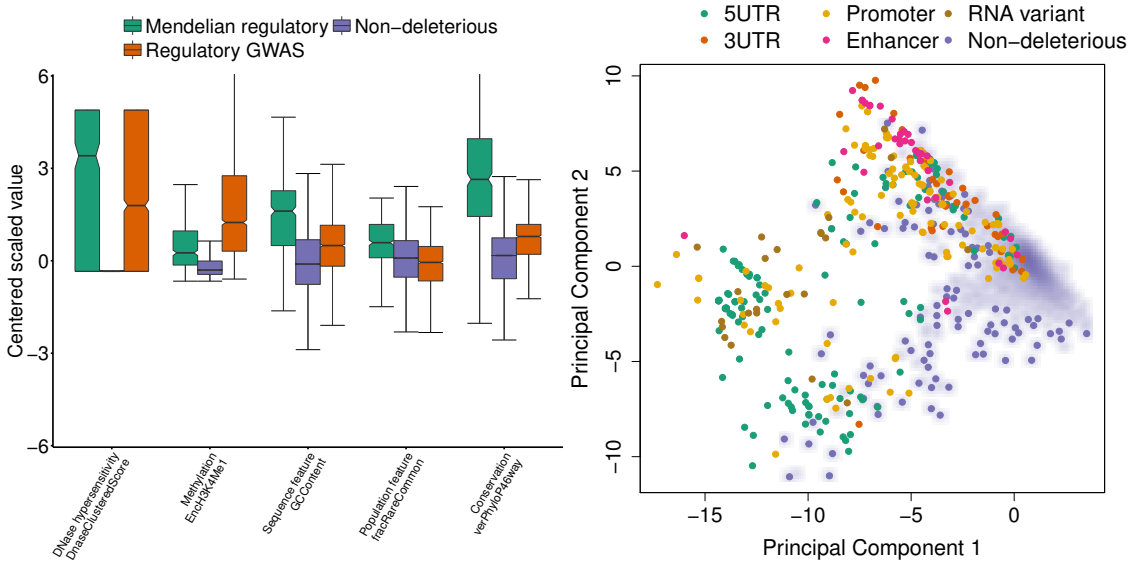
To build a precomputed score for every position in the human genome (reference genome release hg19) the Hy_{score} of the cytoband-aware 10-fold CV was used for all available training data (406 positives and 14,755,199 negatives). For all other positions in the human genome a global model was built using the complete negatives and positives. Therefore all remaining 2,845,135,389 unambiguous (i.e., not "N") positions of the human reference genome (release hg19) were annotated. This precomputed Hy_{score} using the Mendelian data over the whole human genome will be defined as the ReMM score and it is available as indexed tabix format from the website `https://charite.github.io/software-remm-score.html`.

Now only a simple lookup is needed to find out the ReMM score of a new observed variant. Here we need only the position and not the exchange of the variant because ReMM is constructed only on the positions in the human genome. This means also that ReMM is not directly aware of changes in sequence of motives. For example, to get the ReMM score of variant `chr16:209709T>C` we can use the tabix command to get the score of position 209,709 on chromosome 16: `tabix ReMM.v0.3.1.tsv.gz 16:209709-209709`. The result should be 0.5300. Small InDels were explicitly excluded from the training set (exactly 47 small InDels were present in the curated Mendelian data) but a ReMM score can also be computed for them. In order to predict the pathogenicity of deletions, it is possible to use the maximum ReMM score of any nucleotide affected by the deletion. E.g. the positions of three deleted nucleotides of the deletion `chr2:86564631CATG>C` have the scores 0.9230, 0.9250 and 0.9560 resulting in a final score of $\max\{0.9230, 0.9250, 0.9560\} = 0.9560$. For insertions, we can use the maximum ReMM score of the two nucleotides that surround the insertion. For example the position before the insertion `chr9:35657917G>GCA` has the score 1.0 and the position after the insertion has the score 0.9950, resulting in a final score of $\max\{1.0, 0.9950\} = 1.0$.

ReMM Score Feature Analysis To show that the attributes are informative between positive and negative instances several analyses were done to inspect the feature set. The Mendelian regulatory mutations displayed a number of substantial differences as compared to the neutral variants but also to other variant groups like GWAS hits. Figure 4.3a shows the centered mean and scaled genomic attributes of Mendelian regulatory mutations as compared with the derived non-deleterious positions as well as the 2115 regulatory GWAS hits from the GWAS data (see Section 3.2.2), annotated with the Mendelian features. Five highly informative attributes of different attribute groups are shown. In all shown attributes the Mendelian regulatory mutations are significantly different to the regulatory GWAS hits and to the negatives.

In addition, principal-component analysis (PCA) was performed on the two variant classes of the Mendelian dataset with all 26 features visualized in Figure 4.3b. The first two components show a certain separation between our Mendelian regulatory mutations and the negative variants. Both

principal components make up 27 % of the total variability. Using PCA we are able to discover challenging variant classes. Inspecting the different regulatory classes in Figure 4.3b we see that 5'UTR, promoter and RNA variants are the three classes that separate best with the non-deleterious variants. Enhancer and 3'UTR variants seem to be more challenging. So in the next section I will inspect the ReMM score performance of the different functional variant classes to see if we can detect a similar behavior.



(a) Higly informative attributes. **(b)** PCA.

Figure 4.3: Genomic attributes of regulatory Mendelian mutations. **(a)** Centered mean and scaled genomic attributes of Mendelian regulatory mutations compared with the derived non-deleterious positions as well as the 2115 regulatory GWAS hits from the GWAS data (see Section 3.2.2) annotated with Mendelian features. Five highly informative attributes of different attribute groups are shown. **(b)** PCA showing the first two principle components, which make up 27.2 % of the total variability.

ReMM Score Performance of different Functional Elements Table 4.1 shows the performance of the ReMM score considering variants located in different functional elements of non-coding regions. Performance values were generated by using the ReMM score of positives from a specific functional element, removing all other positives and using all negatives for comparison. Results highlight that ReMM achieves relatively good results for different categories of functional elements, although the performance was poorer for mutations in 3' UTRs and enhancers. This is in concordance with the insights gained by the PCA analysis (Figure 4.3b) in the previous paragraph.

The poor performance on the 3'UTR site was also detected by LINSIGHT [35]. Their median LINSIGHT score was only 0.076. For example 5'UTR have a median score of 0.128. Because LINSIGHT used similar feature sets and also the negative training is somehow similar to ReMM one can claim that the feature set is not sufficient for 3'UTR variants. 3'UTRs are not under the same rigid structural constraints as the coding or the 5'UTRs that have to accommodate the translational machinery. Evolutionary pressure may thus have taken advantage of the greater degree

of freedom of 3'UTRs to modulate mRNA molecules [137]. Therefore more diverse features are needed to portray the mechanisms of the 3'UTR.

Table 4.1: ReMM score performance with respect to specific functional elements. Performances considering variants located in different functional elements of non-coding regions. AUROC_{100} is the AUROC computed considering only the top ranked 100 variants. AUROC_{500} and AUROC_{1000} are computed in a similar way.

Functional Element	AUPRC	AUROC	AUROC_{100}	AUROC_{500}	AUROC_{1000}
5'UTR	0.31	1.00	0.62	0.82	0.89
3'UTR	0.01	0.95	0.60	0.79	0.78
Enhancer	0.05	1.00	0.36	0.66	0.83
Promoter	0.23	0.99	0.64	0.78	0.83
RNA component	0.51	1.00	0.91	0.85	0.89
MiRNA	0.23	1.00	0.41	0.88	0.94

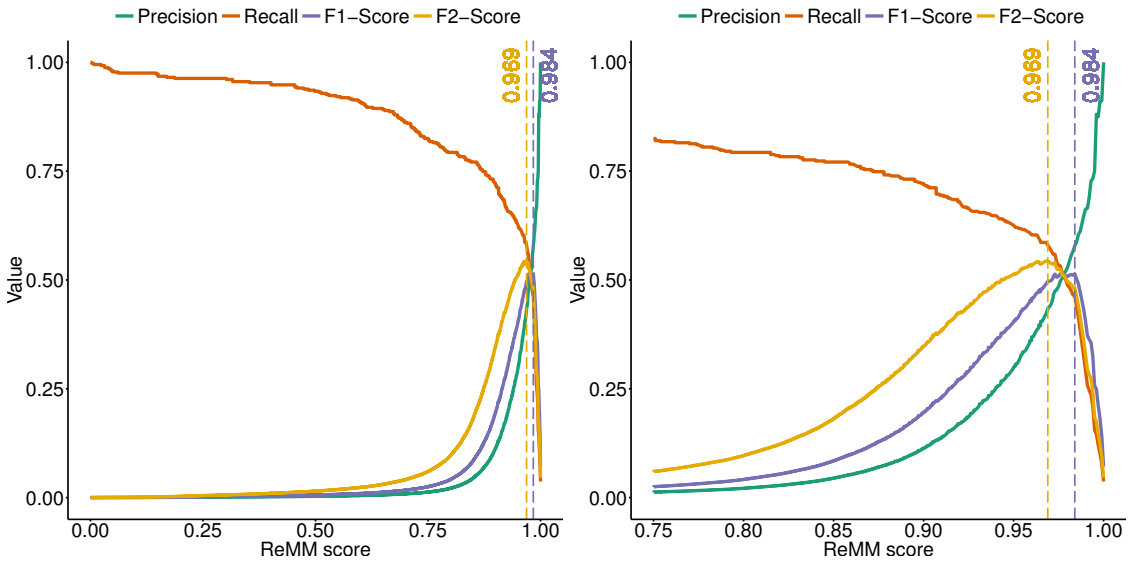
Probability Cutoff The question is upon which ReMM score we can achieve a reliable result if the variant is involved into a regulatory mechanism. In theory each RF of hyperSMURF returns a probability and the Hy_{score} is the average probability over the 100 partitions ranging from 0 to 1. So using cutoff $c = 0.5$ will return a good result in terms of positives because nearly every positive mutation should be predicted as pathogenic. But the problem is that the number of negatives might be extremely large compared to the positives and we will have an overestimation.

In Figure 4.4a the precision, recall and F-measures are plotted over the complete range of the ReMM score. As expected, if we choose $c = 0.5$ we will get a recall close to one, meaning that we will predict nearly all positives as pathogenic (379 out of 406; TP rate $= 0.934$). But the precision is close to 0 so we will predict a lot of false positives (FP) (124,756 out of 14,755,199; FP rate $= 0.008$). The F-measure (F_1score) shows that we will have the highest harmonic mean of recall and precision at $c = 0.984$. This results in a TP rate of 0.463 and a FP rate of 0.000009. If we want to predict more positives it might be good to use the maximum of the F_2score $= \frac{5 \cdot \text{precision} \cdot \text{recall}}{4 \cdot \text{precision} + \text{recall}}$. So we will get a recall four times larger than the precision. In Figure 4.4b we see that the cutoff will lie around $c = 0.969$ resulting in a TP rate of 0.583 and a FP rate of 0.000021.

ClinVar [44, 45] has two categories for deleterious Mendelian mutations: (1) pathogenic and (2) likeley pathogenic. So if we transfer this to ReMM we can define the group of pathogenic predictions with a score larger that 0.984 ($\max \text{F}_1$score). A likely pathogenic prediction will be if the ReMM score is larger than 0.969 ($\max \text{F}_2$score). Of course values lower the likely pathogenic score can have a regulatory influence in the genome but it is unlikely that this cause a Mendelian disease directly. Maybe those variants are involved in more common diseases or acting in an oligogenic fashion.

Depending on the approach where ReMM is integrated the cutoff c must be wisely chosen, and it might be advantageous to use the raw ReMM score rather than defining cutoffs. For example in Genomiser we use a cutoff of $c = 0.5$ to completely filter the variants below it. This ensures that

we miss only 21 SNVs and no InDels out of 453 variants during the performance measurement (see Section 4.4). Now we still have a lot of FPs but the powerful approach of Genomiser is to combine ReMM with phenotype predictions and other works have previously shown that phenotypic information can effectively boost the prioritization of disease-associated genes because it is very specific and selective [54, 57, 138]. Therefore it is fine to choose a lower threshold because the combination with another approach boosts the performance.



(a) Complete range of scores.

(b) Range $[0.75, 1]$ of scores.

Figure 4.4: Precision, recall, F_1 score (F-measure), and F_2 score of the ReMM score. (a) Complete range of ReMM score. **(b)** Details of ReMM score in the interval $[0.75, 1]$.

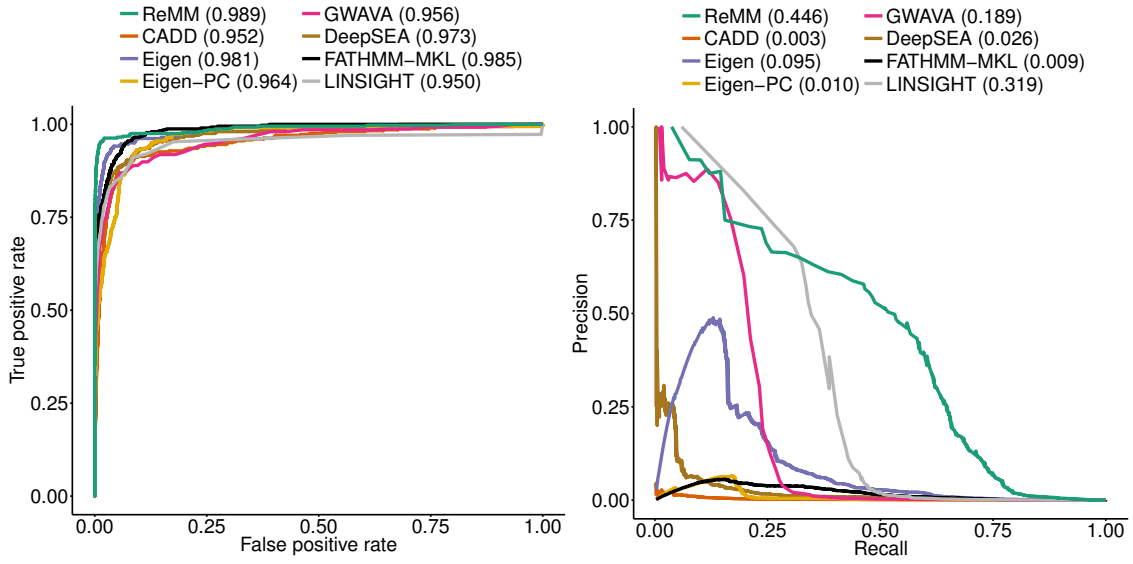
4.3.1 Comparison to genome-wide Pathogenicity Scores

The Mendelian data training set was used to show the performance of other genome-wide non-coding scores for relevance prediction in this context. The result will show which score might be a good choice to use within Genomiser to predict NCVs. The following other non-coding scores were used for comparison: CADD v.1.3, GWAVA v.1.0, FATHMM-MKL, Eigen, Eigen-PC, DeepSEA, and LINSIGHT [33–38]. CADD, Eigen, Eigen-PC and LINSIGHT scores were extracted from the provided precomputed genome-wide file. For Eigen and Eigen-PC all variants on allosomes were removed, because Eigen/Eigen-PC is available only on autosomes. Position scores of GWAVA, FATHMM-MKL[1] and DeepSEA were computed using the source code and trained models provided by the authors. For ReMM the scores of the cytoband-aware 10-fold CV were used (see Section 3.3).

Evaluation was mainly done using the ROC and PR curves which are visualized in Figure 4.5. More statistical comparison is described in Appendix C. The PR and ROC curves show that in the context of the prioritization of the Mendelian mutations, the ReMM score substantially out-

[1] commit `d4af576240fb872179805fb113e892597248441d`

performs other state-of-the-art non-coding scores. It is worth mentioning that in the context of extremely imbalanced data, the AUPRC curve is more informative than the area under the ROC (see Section 2.2.5 [22–24]) but even the small differences between the ROC curves, shown in Figure 4.5a, are in some cases statistically significant according to the Delong test [119] for the comparison of the areas under dependent ROC curves (Table C.1).



(a) ROC curves of pathogenictity scores.

(b) PR curves of pathogenictity scores.

Figure 4.5: Performance of non-coding variant scores on Mendelian data. **(a)** ROC and **(b)** PR curves of the different inspected genome-wide pathogenicity scores on the Mendelian data. Curves were generated using available precomputed scores or pre-trained models. For Eigen and Eigen-PC only the autosomes were used and all variants (positives and negatives) on allosomes were discarded. Numbers after the score name in the legend show the AUC.

If we take a closer look at the PR performance of the different scores (Figure 4.5b) we see that GWAVA works reasonably well for 20 % of the regulatory Mendelian mutations. Also Eigen and Eigen-PC have a small signal for those positives. I would suggest that there is a small fraction of around 20 % of positives that are very different to the negatives. If we go back to the PCA analysis there is a small cluster of regulatory Mendelian mutations that is far away from the negatives and no negatives lie inside the cluster.

Simulation of real Genomes Whichever scoring methodology is used, variant analysis alone is unlikely to be useful to identify Mendelian disease-associated variants in WGS data, which typically contain over 4 million variants, approximately 25,000 of which are located in protein coding sequences. Therefore an analysis on real genomes using the phase 3 1KG genomes [41] was done to see the performance of the scores in a more realistic environment. To do so, variants with a higher MAF than 1 % were removed to gain only rare variants that might be involved in a Mendelian disease. The remaining variants per genome were annotated with their ReMM, CADD, Eigen and Eigen-PC scores because they have precomputed scores over the whole genome. Only autosomes were used in that analysis because allosome scores are not available

for Eigen/Eigen-PC. All remaining variants in every sample of the 1KG were ordered according to their score value. Then every regulatory Mendelian mutation (345 NCVs on autosomes) was iteratively added to every genome (in total 2504) and the observed rank for every of these spiked-in mutations was recorded.

The MAF filtering of genomes from 1KG yields to an average number of 68,622.3 variants per genome. The 345 regulatory variants resulted in 708,630 of spike-ins. The percentage of observed top first ranks and the percentage of the regulatory variant falls into the 1, 10, 100 and 1000 positions as shown in Figure 4.6. It shows that after filtering out common variants the prioritization by CADD was not able to identify the causative variant as the top hit in any of the samples. ReMM achieves this in 4.41 %, Eigen-PC in 0.69 % and interestingly Eigen in 8.13 % of all genomes. Even looking at the top 10 and 100 variants, the causative variant was only seen in 0 % or 0.29 % of samples by CADD, 21.03 % or 41.86 % by Eigen, 15.84 % or 26.39 % by Eigen-PC and 8.51 % or 20.82 % by ReMM. Therefore analyzing complete genomes will still be a challenging question. But if WGS of the patient's parents are available, the analysis can be easier because the inheritance filter can limit the amount of variants to a manageable size. For example recent studies show that there are around 60 to 100 de-novo SNVs available per genome [139].

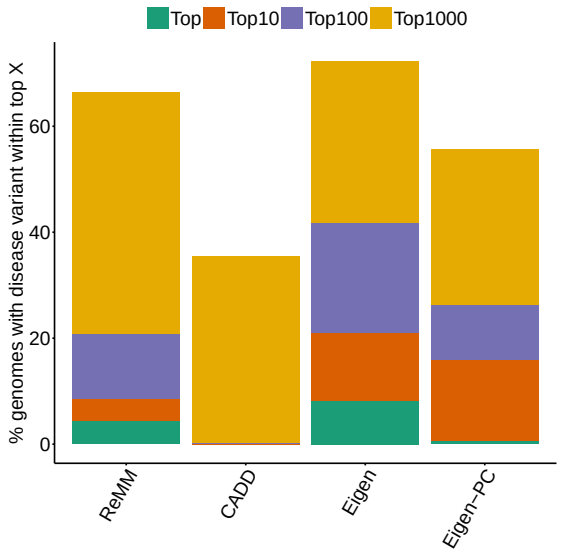


Figure 4.6: Ranking results of non-coding scores in simulated genomes. Every regulatory Mendelian mutation was spiked into all 1KG samples (only rare variants), annotated by ReMM, CADD, Eigen and Eigen-PC, and the observed rank was reported. This barplot shows how often the causative mutation was seen at the first position or within the top 10, top 100 or top 1000 positions of the simulated genomes. The average number of rare variants in the genomes is 68,622.3.

I remark that the better performance of Eigen compared to ReMM and the good performance of Eigen-PC is due to a bias in their feature selection in combination with this analysis. Eigen uses AFs from the 1KG. In this analysis the same genomes were used to spike-in our regulatory mutations. Therefore every variant in a genome, except the added regulatory variant, has an AF in the 1KG, and Ionita-Laza *et al.* [36] argued that high weights of Eigen correlate with no AFs in the 1KG. Therefore a fair comparison of Eigen between CADD and ReMM is not possible using

simulated whole-genomes of the 1KG. The reduced performance of Eigen-PC compared to Eigen is due to the lack of a clear distinct AF group for the eigendecomposition and the AF in the lead eigenvector of Eigen-PC can play a minor role.

These results show that we need additional approaches to find the causative mutation in WGS from persons with a Mendelian disease. Especially if we do not want to be restricted to a to a subclass of variants like de-novo mutations. Here, Genomiser comes into place because it builds a bridge from the genetic information to the phenotypes which always boosts the ranking [54, 57, 138]. Thus, the next Section 4.4 will present the performance of Genomiser on the Mendelian dataset with real genomes.

4.4 Genomiser Performance

For performance evaluation every Mendelian regulatory mutation (453 NCVs, see Table 3.1 and Appendix A) was inserted 23 times in one of the 1092 whole-genome VCF files from the phase 1 1KG [41] resulting in $453 \times 23 = 10{,}419$ simulated genomes. For dominant diseases, one heterozygous mutation was added, and for recessive diseases, either one homozygous mutation or two heterozygous mutations were added to the 1KG VCF file. In all experiments the phenotypic (HPO) annotations for the corresponding disease in OMIM were taken from the HPO annotation files. To measure the ability of Genomiser to detect known disease-gene associations, the analysis was repeated with incomplete (maximum of three HPO annotations), noisy (two random HPO-terms added) and imprecise (two of the original HPO annotations replaced by the more general parent terms in the ontology) annotations.

The simulated genomes were run through the default settings of Genomiser: $\leq$1 % MAF, ReMM or CADD score for NCVs, MutationTaster, Polyphen, and SIFT for coding and phenotype similarity with hiPHIVE. ReMM score position included in the ReMM training set were generated through the cytoband-aware 10-fold CV. For all other positions in the human genome a global model using the complete negative and positive positions was used. At the end the performance of Genomiser was measured with respect to the seeded regulatory Mendelian variant. The number of first ranked causative variants from all 10,419 experiments of the simulated genomes was counted.

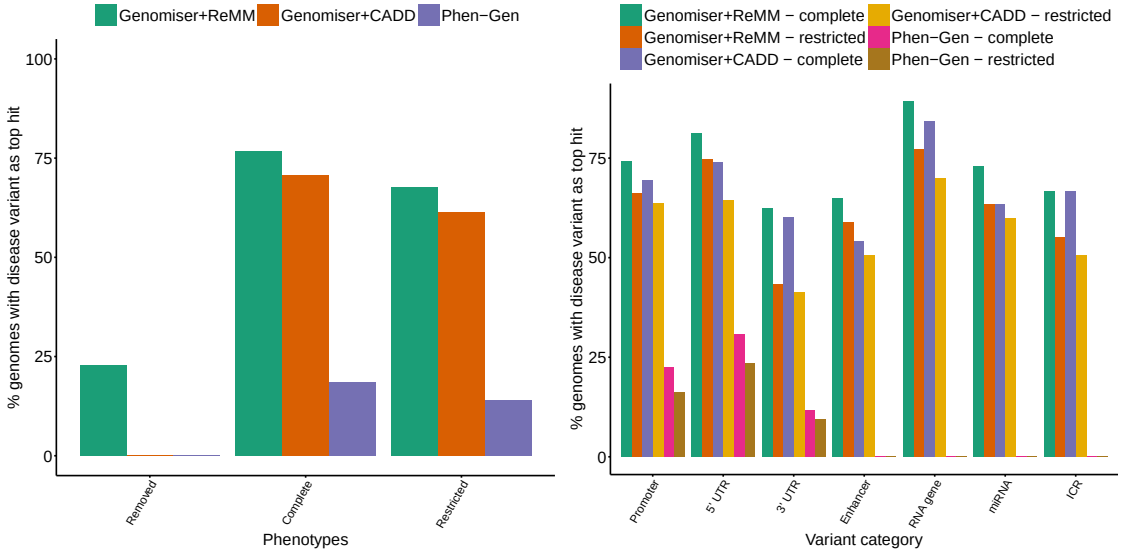
In 2014 Javed *et al.* [57] published the tool Phen-Gen. It has the capacity to process HPO-encoded phenotypic information and whole-genome data. Therefore Phen-Gen was used to compare its performance with the performance of Genomiser using the same genomic and phenotypic profiles and identical AF and inheritance model filtering.

Figure 4.7a shows Genomiser was able to prioritize the causative, regulatory variant as the top-scoring candidate in 77 % of the simulated Mendelian genomes when using the full phenotypic profile. There is a slight reduction in performance to 68 % when using the restricted phenotypic profile that is more likely to represent the type of phenotype annotations collected in realistic clinical settings.

Figure 4.7b shows that the performance varied according to variant category: the 5'UTR, RNA gene and miRNA gene mutations were the easiest to prioritize and the 3'UTR mutations were

particularly difficult followed by enhancer mutations. This is in accordance to the results of the ReMM score performance with respect to specific functional elements in Table 4.1. The table shows that ReMM has the lowest performance on 3'UTR and enhancer mutations. But also CADD has the same decrease in performance, suggesting that 3'UTR and enhancer mutations used in this work are in general difficult to detect.

Interestingly for 3'UTR mutations the performance drops significantly when using restricted phenotypes. This is mostly due to the fact that the used mutations in 3'UTR causing a rare disorder were only associated with two HPO-terms. For example there are 7 NCVs causing Beta-Thalassemia (OMIM-id 613985). This syndrome is only annotated with the terms `HP:0011906` and `HP:0004840`. By creating the restricted phenotypes we will add two random terms to the original terms. Thus we have an equal correct-to-noise term ratio resulting in a bad performance. Other examples for syndromes caused by mutations in the 3'UTR with only two annotated terms are the Alpha-Thalassemia, Myopia 21 or Deafness (autosomal recessive 39).



(a) Genomiser and Phen-Gen performance on simulated genomes.

(b) Genomiser and Phen-Gen performance by variant category.

Figure 4.7: Performance on Genomiser and Phen-Gen with simulated 1KG genomes using the 453 NCVs. Restricted phenotype means incomplete (maximum of three HPO annotations), noisy (two random HPO-terms added), and imprecise (two of the original HPO annotations replaced by the more general parent terms in the ontology) phenotype annotations.

Using CADD instead of the ReMM score the performance was 71 %, and 61 % when using the full or restricted phenotypic profiles, respectively. When CADD was used without any phenotype data the causative variant was not seen as the top scoring hit in any samples. Compared to ReMM without phenotype data the performance dropped substantially but 23 % were still seen at the first position in a whole genome (see Figure 4.7a).

In both scenarios, using complete and restricted phenotypes, the Genomiser results represent a substantial improvement over Phen-Gen [57], which achieved performances of 19 % and 14 % using the full or restricted phenotypes, respectively (Figure 4.7a). In addition, Phen-Gen was

not able to detect enhancer, miRNA, and ICR mutations. Therefore it can only be used to detect mutations close to a gene (Figure 4.7b).

To assess the performance of Genomiser on 22 published cases of compound heterozygosity, both mutations were spiked into a 1KG VCF file and ran the standard Genomiser analysis with ReMM. Here, one causal mutation is regulatory and the other is coding or at a splice site (see Table D.1). The causative gene was ranked at the top in 18 (84 %) of all simulations, demonstrating the ability of Genomiser to integrate information about coding and NCVs into the prioritization process.

Weaken the 20 kb Regulatory Region Constraint Genomiser has a limitation to detect variants that are over 20 Kb away from the next gene. In that case the variant must overlap with predicted enhancers from the FANTOM5 consortium [135] or regulatory regions from the Ensembl regulatory feature build [136]. This constraint speeds up the analysis of Genomiser because fewer variants have to be considered. For the actual test set this works substantially well because of the good performance of Genomiser (Figure 4.7). But on new genomes the constraint might be problematic and can filter out good candidates. In addition, it might not be needed because the ReMM score uses FANTOM5 and Ensemble regulatory features as part of its attribute set. Therefore it can be favorable to let ReMM decide if the variant is regulatory or not.

Another problem of the 20 Kb constraint is that variants, that are in the 20 Kb range of a gene, as well as intergenic variants, are always mapped to the closest gene. Of course it is likely that variants before the TSS are located in a promoter. But in general the literature uses a maximal distance of 2 Kb in front of the TSS [140–142]. The TAD is a conserved structure so we can see it as a regulatory compartment. Rudan *et al.* [143] showed that intra-TAD domain loops are divergent in evolution. In addition Mumbach *et al.* [144] found out that only a fraction (14 %) of enhancers with known GWAS hits target the next gene. Considering these insights, the target of intergenic or intronic variants should be every gene in its corresponding TAD, and Genomiser might assign variants to the false gene which might be phenotypically unimportant.

Because of these thoughts I modified the actual development version of Genomiser[1] to drop the 20 Kb constraint, use ReMM only for relevance prediction of intergenic or intronic variants and associate every intergenic, intronic, upstream, and downstream variant to the most phenotypically similar gene in the TAD. The main differences of the development to the stable Genomiser version 7.2 is that Jannovar v0.20 is integrated to assess the variant effect using the transcript database RefSeq, as well as that it includes minor enhancements and bugfixes.

For performance evaluation every 448 Mendelian regulatory mutation (see Table 3.1 and Appendix A, duplicated positions were removed) was inserted 23 times in one of the 2054 whole-genome VCF files from the phase 3 1KG [59, 68] resulting in $448 \times 23 = 10{,}304$ simulated genomes. Variants were always spiked in with a homozygous genotype. This should have no effect because the inheritance mode is set to unknown and no OMIM prioritizer is used. The function of the OMIM prioritizer is to reduce the phenotype score when the variant does not fit to the inheritance mode of the associated syndrome. E.g. the Exomiser scores of heterozygous variants that are linked to a recessive disease will be divided by two. Randomly two to ten HPO-terms were generated from the associated OMIM disease with a 40 % chance to return a more

[1] commit `2978b534737857fc7c6a7442b88fce9222bdddcb`

generalized term (using a term upwards in the DAG). Finally, 20 % random terms were added so that we finally get a noisy and maybe more realistic phenotype query for the patient. Then the standard and the modified (no-constraint) Genomiser were run on the inputs using a frequency cutoff of 1 % and the hiPHIVE prioritization algorithm. At the end the final score and the rank of the Mendelian regulatory mutation were observed. It is important to say that both versions of Exomiser were run with exactly the same input data (genomes and HPO-terms) to make the results comparable.

Table 4.2 shows how many times we see a Mendelian mutation at the first position or within the top 10 or top 100. Both analyses have really similar results. In the standard Genomiser approach we see the variant of interest 70.027 % at the first position and using the no-constraint approach in 69.501 % of all 10,304 tests. Therefore also the ROC curves in Figure D.1a as well as the AUROCs are nearly the same ($\text{AUROC}_{\text{standard}} = 0.915$ and $\text{AUROC}_{\text{no-constraint}} = 0.914$).

Another important observation is that both benchmark results are similar but slightly better (around 2 %) than the initial Genomiser benchmark runs using restricted phenotypes. This is on the one hand due to minor fixes and improvements of the frameworks. On the other hand, in this analysis the RefSeq instead of the UCSC transcript database is used. RefSeq has less transcripts, especially less predicted transcripts, so that the coverage of the genome is smaller. This results in smaller coding predictions by Jannovar and therefore the pathogenicity of a larger amount of variants will be due to ReMM and not the coding variant scores. As I showed before, ReMM is really restrictive in assigning high values to positions and therefore more variants will be sorted out. So actually both versions of Genomiser work really well on the tested regulatory Mendelian mutations. But the regulatory region constraint can be removed without any drawbacks.

Table 4.2: Genomiser performance comparison between standard settings and without limiting to only regulatory regions in TADs. 453 NCVs were spiked into genomes of the 1KG and Genomiser was run using noisy phenotype queries. Here we see the percentage of the gene associated to the variant at the rank one (Top), within the first 10 (Top10) and within the first 100 (Top100) genes. Genomiser was used in the standard version (Standard) and in a modified version (No-Constraint) where the assignment of 20 Kb to the next gene was removed and every intergenic, intronic, upstream, and downstream variant was associated to the most phenotypcially similar gene in the TAD. No overlap to a regulatory region defined by FANTOM5 [135] or the Ensembl regulatory feature build [136] is needed here.

Rank	No-Constraint	Standard
Top	69.5 %	70.7 %
Top 10	83.2 %	83.2 %
Top 100	89.2 %	89.2 %

4.5 Discussion and Chapter Conclusion

In this chapter I explained the coding variant prioritization tool Exomiser for Mendelian disease. Exomiser uses variants from a VCF file together with phenotypes, encoded in HPO-terms, further filters them on variant effect and allele frequency, and finally ranks the variants/genes according to phenotype similarity and variant pathogenicity. Two important components in Exomiser are on the one hand the HPO, a standard terminology to describe phenotypic features, which enables semantic similarities to phenotypic similar genes to an input term query; and on the other hand Exomiser uses Jannovar, a library that annotates the variant effect in SO-terms and can annotate variants with inheritance patterns.

Genomiser extends Exomiser to the non-coding part of the genome. On deep intergenic variants it uses the topological structure of the DNA to assign variants to the most phenotypically similar gene in the same TAD. For pathogenicity prediction Genomiser uses the precomputed non-coding score of the whole genome.

One of these pathogenicity scores is ReMM and it is built directly to assess the relevance of NCVs, including UTRs, in Mendelian diseases. ReMM is trained on 406 Mendelian mutations using the hyperSMURF algorithm. The ReMM score is defined per position in the human genome and ranges from 0 (no regulatory site) to 1 (regulatory site). On the Mendelian dataset ReMM works better than other non-coding scores, like LINSIGHT, CADD or Eigen. All included functional elements have a reasonably good performance. Only variants in the 3'UTR are difficult to detect, which is in concordance with the literature. It seems that 3'UTRs are not under the same rigid structural constraints as other elements so that evolutionary pressure may thus have taken advantage of the greater degree of freedom here to modulate mRNA molecules [137]. Therefore more diverse features and more diverse 3'UTR examples are needed to portray and learn the mechanisms of the 3'UTRs.

A good alternative score in terms of Mendelian regulatory variants might be LINSIGHT. In the performance measurements LINSIGHT shows the best performance of the compared scores, apart from ReMM. It has a slightly better performance than ReMM for around 40 % of the variants but then the performance has a steep decrease (compare with Figure 4.5b). On the one hand LINSIGHT uses similar features per positions, and I showed previously in Section 4.3 that the features are informative for the Mendelian dataset. On the other hand the evolutionary model adapted from INSIGHT seems to model a mendelian evolutionary behavior very well. Therefore the fitting of the selection parameter S_i per site i is really informative in terms of Mendelian pathogenic mutations.

The Eigen and Eigen-PC scores are the only scores generated from unsupervised learning. This idea is reasonable because most of the time it is difficult to create good training sets or we are too limited (maybe also biased) in training data using a supervised learning. But the other side of the coin is that Eigen includes allele frequencys (AFs) from the 1KG. This becomes problematic for recessive diseases, because it is likely that those variants have an AF, resulting in lower scores. This drawback was also discovered by benchmarking with simulated real genomes, as Figure 4.6 showed. Every negative variant became a low score because all of them had an AF from the 1KG. Most of the Mendelian mutations are unique and have no AF resulting in good Eigen scores. Now

we have to weigh if an AF cutoff might be a better option. For example the recent work from Whiffin *et al.* [145] showed that using high-resolution variant frequencies to empower clinical genome interpretation might be more powerful than including AFs in a training set.

The combination of Genomiser and ReMM score provides the first practical approach to effectively analyze variant data from WGS. It can find pathogenic NCVs as well as coding variants. Especially the phenotypic approach boosts the performance, and even with realistic restricted phenotype terms Genomiser is able to find the causative mutation in 68 % on rank one. There might be a small drawback for the detection of new pathogenic variants because of the previously discussed constraints and limitations on intergenic, upstream, downstream and intronic variants. But the last analysis showed that these specific constraints are not necessary and the phenotypic approach together with the ReMM score is stable enough. To conclude this I would recommend to remove the regulatory region constraint in further versions of Genomiser because it will be a limitation on novel genomes. The ReMM score by itself seems to be enough to assess the importance of a NCV and no further filters are needed.

Chapter 5

Discussion and Conclusion

In this thesis, I described my contributions to the field of ML on highly imbalanced data and that of assessing the regulatory genome for Mendelian diseases. First, I gave a brief introduction to the background in biology, data mining and ML, computer science and mathematics. Then I presented different well known pathogenicity or deleteriousness scores for the non-coding genome. After that I explained my contribution in this field, the ML algorithm hyperSMURF and the ReMM score together with the application Exomiser and Genomiser.

In this chapter, I will give a short summary of my contributions and discuss limitations. The thesis concludes with an outlook on future work and upcoming challenges.

Contribution Summary

In Chapter 3 I introduced hyperSMURF, a ML algorithm for highly imbalanced data that naturally arise in several genome-wide variant scoring problems. Five major characteristics can be highlighted.

1. HyperSMURF uses over- and undersampling techniques to allow the construction of balanced training sets, thus avoiding the bias towards the non-deleterious genetic variants. The partitioning combined with the over- and undersampling techniques results in diverse base learners, a well-known factor for the success of ensemble methods [80].
2. The relatively small size of the training data is counterbalanced by the partitioning of the data and by the ensemble approach. This allows the algorithm to learn from all or a large set of instances, which improves the coverage of the available training data.
3. Predictions are provided by an ensemble of RFs (hyper-ensemble), thus improving the robustness and the accuracy of the predictions.

4. The relatively small size of the training examples scales nicely with large amounts of data, as shown for genetic variants.

5. The ensemble approach allows a parallel implementation of the algorithm.

HyperSMURF was used in combination with genetic NCV data. Thus, in Section 3.3.1 I explained the cytogenetic band-aware CV, a strategy which allows an unbiased performance measurement. It guarantees that variants from the same gene, disease or locus are not divided between training and test set. Otherwise overfitting will be introduced. In addition, the prediction of positives and negatives in genomically close regions in Section 3.5.2, using similar motivated CV strategies, results in good performance. Even this is known to be a difficult challenge, because variants in the same region might share similar feature values.

The experimental results showed that imbalance-aware methods significantly outperform imbalance-unaware ML methods for the prediction of regulatory non-coding Mendelian and common disease associated variants, as well as SNPs associated with miRNA to eQTLs relationships. Apart from the GWAS and Mendelian negatives, training sets, features, feature sizes and imbalances were different between the analyzed datasets. This shows that hyperSMURF is robust in the prediction of imbalanced data of versatile genetic applications.

In Chapter 4 I explained the usage of hyperSMURF together with the Mendelian data to build the ReMM score for assessing the pathogenicity of NCVs in Mendelian diseases. Previous methods have been designed to detect functional NCVs in general, rather than solely detecting those NCVs that cause Mendelian disease. It is difficult to find a good pathogenic variant set for that purpose due to the fact that available databases contain errors [146] and conflate Mendelian and GWAS-associated variants. Therefore the manually curated 406 Mendelian SNVs were used as positive training set to build the ReMM score. This dataset was created by an extensive and detailed literature curation to identify mutations that are associated with Mendelian disease and whose pathogenicity was judged to be plausible based on co-segregation, experimental evidence, or similar considerations. Further inspection of the data, e.g. in Figure 4.3, showed that they do in fact substantially differ from background positions in the genome and also from GWAS hits.

For the prioritization of Mendelian mutations in WGS data, the ReMM score performed much better than other NCV scores. Here, ReMM was compared to CADD, GWAVA, Eigen, Eigen-PC, FATHMM-MKL, DeepSEA and LINSIGHT [33–37]. The ReMM score was precomputed for every unambiguous position of the human reference genome using all available training data. Afterwards the positions of the training data were replaced with scores from the cytogenetic band-aware CV to have a global score that can be used in other approaches and for unbiased comparisons to other scores on the full range of the human genome. Apart from the unsupervised Eigen/Eigen-PC approach, none of the other scores provides such an unbiased scoring file.

To find the causative mutations in rare genetic disorders, the Exomiser framework brings multiple phenotype and genotype strategies together to filter and rank variants according to the disease. Also in Chapter 4 I explained Exomiser, its implemented filters and phenotype prioritization algorithms together with important underlying concepts: the HPO and the software tool Jannovar for functional annotation of variations. Then I introduced Genomiser, a further development of Exomiser, to analyze variant data from whole genomes. Genomiser takes advantage

of the topological confirmation of the genome to assign NCVs to a related gene. The software was benchmarked using 10,419 simulated whole genomes with complete and restricted as well as without phenotypes, in combination with ReMM or CADD as pathogenicity scores for NCVs. I was able to show that the inclusion of phenotype data is critical for the effective prioritization of the regulatory variants because performance dropped from 68 % to 23 % when Genomiser was run with a consequent removal of phenotypic filtering and prioritization. The same analysis, but with a replacement of ReMM with CADD, failed completely on this task, because CADD is not specifically designed for regulatory Mendelian mutations and highly imbalanced data.

Challenges and Outlook

Finally I discuss how hyperSMURF, the ReMM score and Exomiser or Genomiser can be used in future applications and research. I will examine the limitations of the different approaches and present strategies to overcome them. I will end this work with some general words about imbalanced data, concluding the considerations in the introduction.

HyperSMURF Oversampling by SMOTE seems to be the weakest factor on imbalanced training, as shown in the analysis of the different compartments of hyperSMURF in Section 3.5.1. But in this work the SMOTE algorithm itself was used like in the initial publication and not further decomposed or analyzed. But several modifications are potentially advantageous on training data with specific characteristics. For example in the original publication of SMOTE by Chawla *et al.* [21] a new synthetic instance is generated between exactly two instances in a cluster. It is also conceivable that several or all instances within a cluster will be used for creating synthetic instances. Also, different strategies can be combined to generate different variances, a key component for performance improvement in ensemble learning.

From the analysis of different NCV classes using the ReMM score we know that 3' UTR variants are difficult to classify (compare with Section 4.3). Therefore it might be beneficial to shift the importance of the classification boundary to those minority classes which are difficult to learn during oversampling. An algorithm that combines this idea together with the general approach of SMOTE is the adaptive synthetic sampling approach (ADASYN) [147]. ADASYN generates a higher fraction of synthetic variants for minority classes that are more difficult to learn.

HyperSMURF has an advantageous algorithmic feature: it is highly parallelizable. Each training and testing per partition is independent from each other. All actual implementations, in R and Java, support parallel computing, but only on a single machine. On cluster environments multiple small jobs are preferred to a large one. Therefore a fast and parallel implementation with a high performance cluster support will assist the further usage of hyperSMURF in other fields of applications.

A limitation of the proposed approach is the large number of learning parameters. Many different parameters might act as a deterrent to the use of hyperSMURF in other projects. Indeed the proper choice of good setting values is not always straightforward and may have a certain impact on the accuracy of the algorithm. The provided default parameters in Table 3.2 worked well in

the experiments. Also an experimental procedure to automatically tune the parameters through an internal CV was introduced. In the future this can be implemented directly into the software to provide a decision support system to the user that helps to gain an optimal parameter set without overfitting. Again a fast, parallel and cluster support software will be favorable because many computational expensive experiments have to be done in the internal CV.

ReMM Score The ReMM score clearly falls in category one of genome-wide scores, described in Section 2.4. To recap it here, these scores use a ML classifier that attempts to separate known disease variants from putatively benign variants with a variety of genomic features. Therefore ReMM is specifically designed for regulatory Mendelian diseases. On the one hand this is the strength of ReMM: the good curated positive data, the features directly selected for that purpose and the special imbalance-aware strategy lead to a superior performance on Mendelian regulatory variants in comparison to other non-coding scores. Right now the number of known causative regulatory variants is still extremely low and therefore it is difficult to assess the performance on the full range of possible regulatory Mendelian mutations. Further work will be needed to determine whether the ReMM score, future versions of the ReMM score or one of the competing scores will be useful for the full spectrum of non-coding Mendelian variations, which could conceivably differ in many ways from the small set of currently known non-coding Mendelian variants.

On the other hand the performance of ReMM will be limited naturally on other groups of variants like somatic mutations or common variations. If other types of variant classes should be investigated, like coding, GWAS or somatic variants, the evolutionary driven scores like CADD or LINSIGHT, or even the unsupervised Eigen score, will model the data better because theses scores use a more general approach. But an important factor of evolutionary scores is sequence conservation. Berthelot *et al.* [148] and Villar *et al.* [149] showed that conserved regulatory regions are rare and tend to be promoters. So sequence conservation is not the most important factor of evolution in enhancers and therefore evolutionary scores are limited in modeling these regions.

In Section 2.5 I concluded that databases used for performance testing can have an observation bias. For example most of the ClinVar mutations are located in BRCA2 and BRCA1 [62], resulting in an underrepresentation of variants in other genes. Partly this is also true on the Mendelian data, the training data of the ReMM score. The 406 NCVs are associated with 132 different genes but the five genes TERC, HBB, FTL, RMRP, and F9 contain 30 % of all variants. In some subgroups the observation bias is much higher. For example 17 (46 %) of all 37 enhancer mutations are linked to the gene SHH. We can suppose that variants from one gene or element act in a similar way and can have similar features. This might lead to an overrepresentation of variants of this type in the overall training set.

The best possibility to remove the observation bias is to train multiple classifiers with a randomly sampled subset by selecting only a small fraction of variants per gene and variant class. New predictions will be made by an average or majority vote over all classifiers. For the ReMM score I discarded this strategy because the positive dataset is extremely small. A subsampling of variants will increase the imbalance which will cause other unintentional effects of decreasing

performance. But this idea should be considered if the number of regulatory Mendelian mutations increase. With the actual amount of data it might be better to integrate adaptive learning strategies for oversampling. This can be achieved through the previously discussed method ADASYN or a modified version of it, which takes the number of variants per gene into account during creation of synthetic instances.

In this work only unspecific regulatory features were used, like the maximum H3K4 methylation level from different cell lines. The advantage of these features is that they have a dense signal over the whole genome. Also the TAD is conserved over cell lines and can be seen as a general compartmentation of the DNA. But most regulatory mechanisms are really specific on the tissue type or the time point when they are active – broadly speaking what happens within a TAD is not always the same. This introduces several new problems. Tissue and maybe time point specific experiments have to be made, which introduces two totally new dimensions. Mostly the data generated from such experiments are sparse on the whole genome, resulting in not assigned or zero attributes for all other, not involved instances. Therefore we need a logic to integrate the data before we begin with the training. A first step into this direction might be DeepSEA because it integrates several sparse regulatory features into sequence-based dense scores. But still the tissue and time dimension is missing. In the future this gap might be closed by multiplex assays of variant effect (MAVEs) [62]. For example massive parallel reporter assays (MPRAs) are able to test every variant effect (SNVs and 1 bp deletions) of a regulatory region in a specific tissue [150].

Exomiser and Genomiser An application that integrates the ReMM score to find causative regulatory mutations in Mendelian diseases is the Genomiser framework. The performance of the simulated genomes showed that the software is able to find the causative mutation in around 70 % of all tests. So the software is ready for new unsolved genomes. The combination with Exomiser also enables the observations of coding variants, and maybe Genomiser can help to find the missing link in those cases where we have only one heterozygous coding mutation in a recessive gene. With the AR filter of Jannovar, Genomiser will detect the second non-coding mutation, e.g. a 5' UTR variant which disturbs the transcription of one allele, leading to transcripts that contain only the pathogenic coding mutation (like a homozygous variant). Genomiser can even be used on exome sequencing data because the exon enrichment often covers 5' and 3' UTRs.

One effect prediction is still missing in the framework: splice sites. Only essential splice sites, ±2 bases of the exon start or end, are predicted. But other splice sites exist like deep intronic splice elements or exonic splicing enhancers (ESEs). Prediction of splice elements will be crucial to assess the pathogenicity of the whole range of all possible SNVs or InDels that might cause a rare genetic disorder. Again MAVEs exist for revealing variant effects on mRNA processing [62]. With these splicing assays it will be possible to generate a large dataset for variant splice effects and this can be used to generate an accurate score for splicing prediction.

Because Exomiser and also Genomiser are used in multiple large projects and the Monarch Initiative together with the Queen Mary University of London push the development, Exomiser will be improved further – with more data being integrated and old data updated. If Genomiser is becoming a standard tool for analyzing genomes, constant further development would be needed.

In the actual development a representational state transfer (REST) web service is integrated into Exomiser and multiple features will sill be added, like using own frequency databases e.g. from an in-house cohort. In addition the Exomiser developer team plan to generate a version of the new genome GRCh38 release. Therefore also ReMM needs to be updated. A simple lift-over to the new genome might be possible. But the best way will be a new training directly on features and variants of the new release.

The central point of Genomiser are the phenotypes because they are the key component for a good performance. But for untrained users it might be difficult to detect the key phenotypes or whether separate phenotypes belong to the syndrome or to something else. The presented phenotype similarity algorithms can handle noise to some extent because when using restricted phenotypes the performance of Genomiser is still high. But other phenotype approaches might make more stable predictions irrespective of user input. We can think of using image analysis from faces, limbs or X-rays to detect phenotypic abnormalities [151]. This can be added as a third dimension, beside HPO and molecular deleteriousness scores, to predict the affected gene with a causative mutation.

So we see that for specific purposes in biology or medicine, like the relevance prediction of NCVs, we are limited to certain amount of data, mostly for positive examples. This can result in a huge imbalance between the effect we want to predict and what we consider as neutral or negative. To increase the number of examples we cannot use the concept of self-play and take another spin, like in the board game Go [1, 2]. Huge efforts have to be made to gain more observations. But where financial and technical limitations exist, this may not always be possible.

Smart but imbalance-unaware methods will mostly fail on highly imbalanced data because they tend to learn only the majority class. Therefore algorithms designed specifically for highly imbalanced data can help to fill that gap. Using such imbalance-aware ML techniques and a good curated training set with convincing features, it is possible to extract knowledge out of a handful examples and expand it to a large scale like the whole genome. In this work I successfully used this approach for relevance prediction on NCVs. Finally a clever combination with other approaches, like the ML on regulatory Mendelian mutations together with phenotypes, results in powerful applications.

In the future, approaches of combining multiple datasources will become an important topic. The general challenge will be to integrate different metadata and experiments of different tissues and time points, use AI to find the links within the data, and extract knowledge out of it. But we should always keep in mind that there might only be a few known examples at any given time point. That is where efficient imbalance-aware ML strategies can make a major contribution.

References

1. Silver, D. *et al.* Mastering the game of Go with deep neural networks and tree search. *Nature* **529**, 484–89 (Jan. 2016).
2. Littman, M. L. in *Machine Learning Proceedings 1994* (eds Cohen, W. W. & Hirsh, H.) 157–63 (Morgan Kaufmann Publishers Inc., San Francisco (CA), July 1994).
3. Simonis, M., Kooren, J. & de Laat, W. An evaluation of 3C-based methods to capture DNA interactions. *Nature Methods* **4**, 895–901 (Nov. 2007).
4. Sanger, F., Nicklen, S. & Coulson, A. R. DNA sequencing with chain-terminating inhibitors. *Proceedings of the National Academy of Sciences* **74**, 5463–7 (1977).
5. Sanger, F. & Coulson, A. R. A rapid method for determining sequences in DNA by primed synthesis with DNA polymerase. *Journal of Molecular Biology* **94**, 441–6 (May 1975).
6. Kumar, P., Henikoff, S. & Ng, P. C. Predicting the effects of coding non-synonymous variants on protein function using the SIFT algorithm. *Nature Protocols* **4**, 1073–81 (June 2009).
7. Adzhubei, I. A. *et al.* A method and server for predicting damaging missense mutations. *Nature Methods* **7**, 248–9 (Apr. 2010).
8. Schwarz, J. M., Rödelsperger, C., Schuelke, M. & Seelow, D. MutationTaster evaluates disease-causing potential of sequence alterations. *Nature Methods* **7**, 575–6 (Aug. 2010).
9. Schwarz, J. M., Cooper, D. N., Schuelke, M. & Seelow, D. MutationTaster2: mutation prediction for the deep-sequencing age. *Nature Methods* **11**, 361–2 (Mar. 2014).
10. Davydov, E. V. *et al.* Identifying a High Fraction of the Human Genome to be under Selective Constraint Using GERP++. *PLoS Computational Biology* **6**, e1001025 (Dec. 2010).
11. Cooper, G. M. *et al.* Distribution and intensity of constraint in mammalian genomic sequence. *Genome Research* **15**, 901–13 (July 2005).
12. Siepel, A. *et al.* Evolutionarily conserved elements in vertebrate, insect, worm, and yeast genomes. *Genome research* **15**, 1034–50 (Aug. 2005).
13. Pollard, K. S., Hubisz, M. J., Rosenbloom, K. R. & Siepel, A. Detection of nonneutral substitution rates on mammalian phylogenies. *Genome Research* **20**, 110–21 (Jan. 2010).
14. Wolff, M., Hauck, P. & Küchlin, W. *Mathematik für Informatik und Bioinformatik* (Springer-Verlag Berlin Heidelberg, 2004).

15. Witten, I. H., Frank, E., Hall, M. A. & Pal, C. J. *Data Mining: Practical machine learning tools and techniques* (Morgan Kaufmann Publishers Inc., 2016).
16. Breiman, L. Bagging predictors. *Machine Learning* **24**, 123–40 (Aug. 1996).
17. Breiman, L. Random Forests. *Machine Learning* **45**, 5–32 (Oct. 2001).
18. Quinlan, J. R. *C4.5: Programs for Machine Learning* (Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993).
19. Ling, C. X. & Li, C. *Data mining for direct marketing: Problems and solutions.* in *KDD* **98** (1998), 73–9.
20. Japkowicz, N. *et al. Learning from imbalanced data sets: a comparison of various strategies* in *AAAI workshop on learning from imbalanced data sets* **68** (Sept. 2000), 10–5.
21. Chawla, N. V., Bowyer, K. W., Hall, L. O. & Kegelmeyer, W. P. SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research,* 321–57 (Feb. 2002).
22. Saito, T. & Rehmsmeier, M. The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets. *PLOS ONE* **10**, 1–21 (Mar. 2015).
23. He, H. & Garcia, E. A. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering* **21**, 1263–84 (June 2009).
24. Davis, J. & Goadrich, M. *The Relationship Between Precision-Recall and ROC Curves* in *Proceedings of the 23rd International Conference on Machine Learning* (ACM, Pittsburgh, Pennsylvania, USA, 2006), 233–40.
25. Gruber, T. R. *et al.* A translation approach to portable ontology specifications. *Knowledge acquisition* **5**, 199–220 (June 1993).
26. Ashburner, M. *et al.* Gene Ontology: tool for the unification of biology. *Nature Genetics* **25**, 25–9 (May 2000).
27. Resnik, P. *Using Information Content to Evaluate Semantic Similarity in a Taxonomy* 448–53 (Morgan Kaufmann Publishers Inc., Montreal, Quebec, Canada, 1995).
28. Azuaje, F. & Bodenreider, O. Incorporating Ontology-Driven Similarity Knowledge into Functional Genomics: An Exploratory Study. *BIBE 2004 : proceedings : 19-21 May, 2004, Taichung, Taiwan, ROC. IEEE International Symposium on Bioinformatics and Bioengineering (4th : 2004 : Taichung, Taiwan)* **2004**, 317–24 (May 2004).
29. Pesquita, C., Faria, D., Falcão, A. O., Lord, P. & Couto, F. M. Semantic Similarity in Biomedical Ontologies. *PLoS Computational Biology* **5**, e1000443 (July 2009).
30. De Ligt, J. *et al.* Diagnostic Exome Sequencing in Persons with Severe Intellectual Disability. *New England Journal of Medicine* **367**. PMID: 23033978, 1921–9 (Nov. 2012).
31. Lemke, J. R. *et al.* Targeted next generation sequencing as a diagnostic tool in epileptic disorders. *Epilepsia* **53**, 1387–98 (Aug. 2012).
32. Glöckle, N. *et al.* Panel-based next generation sequencing as a reliable and efficient technique to detect mutations in unselected patients with retinal dystrophies. *European Journal of Human Genetics* **22**, 99 (Jan. 2014).

33. Kircher, M. *et al.* A general framework for estimating the relative pathogenicity of human genetic variants. *Nature Genetics* **46**, 310–5 (Mar. 2014).
34. Ritchie, G. R. S., Dunham, I., Zeggini, E. & Flicek, P. Functional annotation of noncoding sequence variants. *Nature Methods* **11**, 294–6 (Mar. 2014).
35. Huang, Y.-F., Gulko, B. & Siepel, A. Fast, scalable prediction of deleterious noncoding variants from functional and population genomic data. *Nature Genetics* **49**, 618–24 (Mar. 2017).
36. Ionita-Laza, I., McCallum, K., Xu, B. & Buxbaum, J. D. A spectral approach integrating functional genomic annotations for coding and noncoding variants. *Nature Genetics* **48**, 214–20 (Feb. 2016).
37. Shihab, H. A. *et al.* An integrative approach to predicting the functional effects of noncoding and coding sequence variation. *Bioinformatics* **31**, 1536–43 (May 2015).
38. Zhou, J. & Troyanskaya, O. G. Predicting effects of noncoding variants with deep learning-based sequence model. *Nature Methods* **12**, 931–4 (Oct. 2015).
39. Kimura, M. *The Neutral Theory of Molecular Evolution* 367 (Cambridge University Press, 1983).
40. Paten, B. *et al.* Genome-wide nucleotide-level mammalian ancestor reconstruction. *Genome Research* **18**, 1829–43 (Nov. 2008).
41. Consortium, T. 1. G. P. An integrated map of genetic variation from 1,092 human genomes. *Nature* **491**, 56–65 (Nov. 2012).
42. Paten, B., Herrero, J., Beal, K., Fitzgerald, S. & Birney, E. Enredo and Pecan: Genome-wide mammalian consistency-based multiple alignment with paralogs. *Genome Research* **18**, 1814–28 (Nov. 2008).
43. McLaren, W. *et al.* The Ensembl Variant Effect Predictor. *Genome Biology* **17**, 122 (Dec. 2016).
44. Landrum, M. J. *et al.* ClinVar: public archive of relationships among sequence variation and human phenotype. *Nucleic Acids Research* **42**, D980–5 (Jan. 2014).
45. Landrum, M. J. *et al.* ClinVar: public archive of interpretations of clinically relevant variants. *Nucleic Acids Research* **44**, D862–8 (Jan. 2016).
46. Ernst, J. & Kellis, M. ChromHMM: automating chromatin-state discovery and characterization. *Nature Methods* **9**, 215–6 (Feb. 2012).
47. Stenson, P. D. *et al.* The Human Gene Mutation Database: 2008 update. *Genome Medicine* **1**, 13 (Jan. 2009).
48. Welter, D. *et al.* The NHGRI GWAS Catalog, a curated resource of SNP-trait associations. *Nucleic Acids Research* **42**, D1001–6 (Jan. 2014).
49. Bamford, S. *et al.* The COSMIC (Catalogue of Somatic Mutations in Cancer) database and website. *British Journal of Cancer* **91**, 355–8 (June 2004).
50. Fu, Y. *et al.* FunSeq2: a framework for prioritizing noncoding regulatory variants in cancer. *Genome Biology* **15**, 480 (Oct. 2014).

51. Liu, X., Wu, C., Li, C. & Boerwinkle, E. dbNSFP v3.0: A One-Stop Database of Functional Predictions and Annotations for Human Nonsynonymous and Splice-Site SNVs. *Human Mutation* **37**, 235–41 (Mar. 2016).

52. Reva, B., Antipin, Y. & Sander, C. Predicting the functional impact of protein mutations: application to cancer genomics. *Nucleic Acids Research* **39**, e118 (Sept. 2011).

53. Mann, H. B. & Whitney, D. R. On a test of whether one of two random variables is stochastically larger than the other. *The Annals of Mathematical Statistics,* 50–60 (1947).

54. Zemojtel, T. *et al.* Effective diagnosis of genetic disease by computational phenotype analysis of the disease-associated genome. *Science Translational Medicine* **6**, 252ra123 (Sept. 2014).

55. Smedley, D. *et al.* Next-generation diagnostics and disease-gene discovery with the Exomiser. *Nature Protocols* **10**, 2004–15 (Nov. 2015).

56. Smedley, D. *et al.* A Whole-Genome Analysis Framework for Effective Identification of Pathogenic Regulatory Variants in Mendelian Disease. *The American Journal of Human Genetics* **99**, 595–606 (Sept. 2016).

57. Javed, A., Agrawal, S. & Ng, P. C. Phen-Gen: combining phenotype and genotype to analyze rare disorders. *Nature Methods* **11**, 935–7 (Sept. 2014).

58. Gronau, I., Arbiza, L., Mohammed, J. & Siepel, A. Inference of natural selection from interspersed genomic elements based on polymorphism and divergence. *Molecular Biology and Evolution* **30**, 1159–71 (May 2013).

59. Consortium, T. 1. G. P. A global reference for human genetic variation. *Nature* **526**, 68–74 (Oct. 2015).

60. Gulko, B., Hubisz, M. J., Gronau, I. & Siepel, A. A method for calculating probabilities of fitness consequences for point mutations across the human genome. *Nature Genetics* **47**, 276–83 (Mar. 2015).

61. Gompertz, B. On the nature of the function expressive of the law of human mortality, and on a new mode of determining the value of life contingencies. *Philosophical transactions of the Royal Society of London* **115**, 513–83 (1825).

62. Starita, L. M. *et al.* Variant Interpretation: Functional Assays to the Rescue. *The American Journal of Human Genetics* **101**, 315–25 (Sept. 2017).

63. Mather, C. A. *et al.* CADD score has limited clinical validity for the identification of pathogenic variants in noncoding regions in a hereditary cancer panel. *Genetics in Medicine* **18**, 1269–75 (May 2016).

64. Dong, C. *et al.* Comparison and integration of deleteriousness prediction methods for nonsynonymous SNVs in whole exome sequencing studies. *Human molecular genetics* **24**, 2125–37 (Dec. 2014).

65. Liu, X., Li, C. & Boerwinkle, E. The performance of deleteriousness prediction scores for rare non-protein-changing single nucleotide variants in human genes. *Journal of Medical Genetics* **54**, 134–44 (Feb. 2017).

66. Schubach, M., Re, M., Robinson, P. N. & Valentini, G. Imbalance-Aware Machine Learning for Predicting Rare and Common Disease-Associated Non-Coding Variants. *Scientific Reports* **7**, 2959 (June 2017).

67. Lek, M. *et al.* Analysis of protein-coding genetic variation in 60,706 humans. *Nature* **536**, 285–91 (Aug. 2016).

68. Sudmant, P. H. *et al.* An integrated map of structural variation in 2,504 human genomes. *Nature* **526**, 75–81 (Oct. 2015).

69. Sifrim, A. *et al.* Distinct genetic architectures for syndromic and nonsyndromic congenital heart defects identified by exome sequencing. *Nature Genetics* **48**, 1060–5 (Sept. 2016).

70. Stahl, E. A. *et al.* Genome-wide association study meta-analysis identifies seven new rheumatoid arthritis risk loci. *Nature Genetics* **42**, 508–14 (June 2010).

71. Pruitt, K. D. *et al.* RefSeq: an update on mammalian reference sequences. *Nucleic Acids Research* **42**, D756–63 (Jan. 2014).

72. Forbes, S. A. *et al.* COSMIC: exploring the world's knowledge of somatic mutations in human cancer. *Nucleic Acids Research* **43**, D805–11 (Jan. 2015).

73. Ma, M. *et al.* Disease-associated variants in different categories of disease located in distinct regulatory elements. *BMC Genomics* **16**, S3 (June 2015).

74. Visscher, P. M., Brown, M. A., McCarthy, M. I. & Yang, J. Five years of GWAS discovery. *The American Journal of Human Genetics* **90**, 7–24 (Jan. 2012).

75. Edwards, S. L., Beesley, J., French, J. D. & Dunning, A. M. Beyond GWASs: Illuminating the Dark Road from Association to Function. *The American Journal of Human Genetics* **93**, 779–97 (Nov. 2013).

76. Gordon, C. T. & Lyonnet, S. Enhancer mutations and phenotype modularity. *Nature Genetics* **46**, 3–4 (Jan. 2014).

77. Cortes, C. & Vapnik, V. Support-vector networks. *Machine Learning* **20**, 273–97 (Sept. 1995).

78. Bishop, C. *Neural networks for pattern recognition* 1st ed. (Clarendon Press; Oxford University Press, 1995).

79. Galar, M., Fernandez, A., Barrenechea, E., Bustince, H. & Herrera, F. A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based Approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* **42**, 463–84 (July 2012).

80. Kuncheva, L. I. *Diversity in Classifier Ensembles* 295–327 (John Wiley & Sons, Inc., 2004).

81. Kittler, J. & Roli, F. *Multiple Classifier Systems: First International Workshop, MCS 2000 Cagliari, Italy, June 21-23, 2000 Proceedings* (Springer Science & Business Media, 2000).

82. Srivastava, A. N. *Advances in machine learning and data mining for astronomy* (Chapman and Hall/CRC, 2012).

83. Ritchie, G. R. & Flicek, P. *Functional Annotation of Rare Genetic Variants* 57–70 (Springer, 2015).

84. Goldstein, B. A., Polley, E. C. & Briggs, F. Random forests for genetic association studies. *Statistical Applications in Genetics and Molecular Biology* **10** (June 2011).

85. Liu, X.-Y., Wu, J. & Zhou, Z.-H. Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* **39**, 539–50 (Dec. 2009).

86. Cazzola, M. & Skoda, R. C. Translational pathophysiology: a novel molecular mechanism of human disease. *Blood* **95**, 3280–88 (May 2000).

87. Scheper, G. C., van der Knaap, M. S. & Proud, C. G. Translation matters: protein synthesis defects in inherited disease. *Nature Reviews Genetics* **8**, 711–23 (July 2007).

88. Cooper, T. A., Wan, L. & Dreyfuss, G. RNA and Disease. *Cell* **136**, 777–93 (Feb. 2009).

89. Ward, L. D. & Kellis, M. Interpreting noncoding genetic variation in complex traits and human disease. *Nature Biotechnology* **30**, 1095–106 (Nov. 2012).

90. Jarinova, O. & Ekker, M. Regulatory variations in the era of next-generation sequencing: Implications for clinical molecular diagnostics. *Human Mutation* **33**, 1021–30 (Apr. 2012).

91. Jones, B. L. & Swallow, D. M. The impact of cis-acting polymorphisms on the human phenotype. *The HUGO journal* **5**, 13–23 (July 2011).

92. Chen, J.-M., Férec, C. & Cooper, D. N. A systematic analysis of disease-associated variants in the 3' regulatory regions of human protein-coding genes I: general principles and overview. *Human Genetics* **120**, 1–21 (Apr. 2006).

93. Chatterjee, S. & Pal, J. K. Role of 5'- and 3'-untranslated regions of mRNAs in human diseases. *Biology of the Cell* **101**, 251–62 (May 2009).

94. Chuzhanova, N., Cooper, D. N., Férec, C. & Chen, J.-M. Searching for potential microRNA-binding site mutations amongst known disease-associated 3' UTR variants. *Genomic Medicine* **1**, 29–33 (Jan. 2007).

95. Pickering, B. M. & Willis, A. E. The implications of structured 5' untranslated regions on translation and disease. *Seminars in Cell & Developmental Biology* **16**, 39–47 (Feb. 2005).

96. Calvo, S. E., Pagliarini, D. J. & Mootha, V. K. Upstream open reading frames cause widespread reduction of protein expression and are polymorphic among humans. *Proceedings of the National Academy of Sciences* **106**, 7507–12 (May 2009).

97. Wethmar, K., Smink, J. J. & Leutz, A. Upstream open reading frames: molecular switches in (patho)physiology. *Bioessays* **32**, 885–93 (Oct. 2010).

98. Epstein, D. J. Cis-regulatory mutations in human disease. *Brief Functional Genomics* **8**, 310–6 (July 2009).

99. Sakabe, N. J., Savic, D. & Nobrega, M. A. Transcriptional enhancers in development and disease. *Genome Biology* **13**, 238 (Jan. 2012).

100. Khan, I. A. *et al.* In silico discrimination of single nucleotide polymorphisms and pathological mutations in human gene promoter regions by means of local DNA sequence context and regularity. *In Silico Biology* **6**, 23–34 (June 2006).

101. Savinkova, L. K. *et al.* TATA box polymorphisms in human gene promoters and associated hereditary pathologies. *Biochemistry (Moscow)* **74**, 117–29 (Feb. 2009).

102. Cammaerts, S., Strazisar, M., De Rijk, P. & Del Favero, J. Genetic variants in microRNA genes: impact on microRNA expression, function, and disease. *Frontiers in Genetics* **6**, 186 (May 2015).

103. Hrdlickova, B., de Almeida, R. C., Borek, Z. & Withoff, S. Genetic variation in the non-coding genome: Involvement of micro-RNAs and long non-coding RNAs in disease. *Biochimica et Biophysica Acta (BBA) - Molecular Basis of Disease* **1842**, 1910–22 (Oct. 2014).

104. Kawahara, Y. Human diseases caused by germline and somatic abnormalities in microRNA and microRNA-related genes. *Congenital anomalies* **54**, 12–21 (Feb. 2014).

105. Meola, N., Gennarino, V. A. & Banfi, S. microRNAs and genetic diseases. *Pathogenetics* **2**, 7 (Nov. 2009).

106. Pinto, D. *et al.* Convergence of genes and cellular pathways dysregulated in autism spectrum disorders. *The American Journal of Human Genetics* **94**, 677–94 (May 2014).

107. Giardine, B. *et al.* HbVar database of human hemoglobin variants and thalassemia mutations: 2007 update. *Human Mutation* **28**, 206 (Feb. 2007).

108. Podlevsky, J. D., Bley, C. J., Omana, R. V., Qi, X. & Chen, J. J.-L. The telomerase database. *Nucleic Acids Research* **36**, D339–43 (Jan. 2008).

109. Jäger, M. *et al.* Jannovar: A Java Library for Exome Annotation. *Human Mutation* **35**, 548–55 (May 2014).

110. Rosenbloom, K. R. *et al.* The UCSC Genome Browser database: 2015 update. *Nucleic Acids Research* **43**, D670–81 (Jan. 2015).

111. Karolchik, D. *et al.* The UCSC Table Browser data retrieval tool. *Nucleic Acids Research* **32**, D493–46 (Jan. 2004).

112. Andersson, R. *et al.* An atlas of active enhancers across human cell types and tissues. *Nature* **507**, 455–61 (Mar. 2014).

113. MacDonald, J. R., Ziman, R., Yuen, R. K. C., Feuk, L. & Scherer, S. W. The Database of Genomic Variants: a curated collection of structural variation in the human genome. *Nucleic Acids Research* **42**, D986–92 (Jan. 2014).

114. Lappalainen, I. *et al.* dbVar and DGVa: public archives for genomic structural variation. *Nucleic Acids Research* **41**, D936–41 (Jan. 2013).

115. Riggs, E. R., Jackson, L., Miller, D. T. & Van Vooren, S. Phenotypic information in genomic variant databases enhances clinical care and research: The international standards for cytogenomic arrays consortium experience. *Human Mutation* **33**, 787–96 (May 2012).

116. Budach, S., Heinig, M. & Marsico, A. Principles of microRNA Regulation Revealed Through Modeling microRNA Expression Quantitative Trait Loci. *Genetics* **203**, 1629–40 (Aug. 2016).

117. Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R. & Lin, C.-J. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research* **9**, 1871–4 (Aug. 2008).

118. Cessie, S. L. & Houwelingen, J. C. V. Ridge Estimators in Logistic Regression. *Journal of the Royal Statistical Society. Series C (Applied Statistics)* **41**, 191–201 (1992).

119. DeLong, E., DeLong, D. & Clarke-Pearson, D. Comparing the Areas under Two or More Correlated Receiver Operating Characteristic Curves: A Nonparametric Approach. *Biometrics* **44**, 837–45 (1988).

120. Köhler, S. *et al.* Clinical Diagnostics in Human Genetics with Semantic Similarity Searches in Ontologies. *The American Journal of Human Genetics* **85**, 457–64 (Oct. 2009).

121. Smedley, D. *et al.* PhenoDigm: analyzing curated annotations to associate animal models with human diseases. *Database* **2013**, bat025 (May 2013).

122. Köhler, S., Bauer, S., Horn, D. & Robinson, P. N. Walking the interactome for prioritization of candidate disease genes. *The American Journal of Human Genetics* **82**, 949–58 (Apr. 2008).

123. Smedley, D. *et al.* Walking the interactome for candidate prioritization in exome sequencing studies of Mendelian diseases. *Bioinformatics* **30**, 3215–22 (July 2014).

124. Wang, K., Li, M. & Hakonarson, H. ANNOVAR: functional annotation of genetic variants from high-throughput sequencing data. *Nucleic Acids Research* **38**, e164 (July 2010).

125. Robinson, P. N. *et al.* The Human Phenotype Ontology: A Tool for Annotating and Analyzing Human Hereditary Disease. *The American Journal of Human Genetics* **83**, 610–5 (Nov. 2008).

126. Robinson, P. & Mundlos, S. The Human Phenotype Ontology. *Clinical Genetics* **77**, 525–34 (June 2010).

127. Köhler, S. *et al.* The Human Phenotype Ontology in 2017. *Nucleic Acids Research* **45**, D865–76 (Jan. 2017).

128. Robinson, P. N. Deep phenotyping for precision medicine. *Human Mutation* **33**, 777–80 (May 2012).

129. Robinson, P. N. & Webber, C. Phenotype ontologies and cross-species analysis for translational research. *PLoS Genetics* **10**, e1004268 (Apr. 2014).

130. Deans, A. R. *et al.* Finding our way through phenotypes. *PLoS Biology* **13**, e1002033 (Jan. 2015).

131. Köhler, S. *et al.* The Human Phenotype Ontology project: linking molecular biology and disease through phenotype data. *Nucleic Acids Research* **42**, D966–74 (Jan. 2013).

132. Groza, T. *et al.* Automatic concept recognition using the Human Phenotype Ontology reference and test suite corpora. *Database* **2015**, bav005 (Jan. 2015).

133. *Exome Variant Server* Seattle, WA: NHLBI GO Exome Sequencing Project (ESP).

134. Dixon, J. R. *et al.* Topological domains in mammalian genomes identified by analysis of chromatin interactions. *Nature* **485**, 376–80 (Apr. 2012).

135. Lizio, M. *et al.* Gateways to the FANTOM5 promoter level mammalian expression atlas. *Genome Biology* **16**, 22 (Jan. 2015).

136. Zerbino, D. R., Wilder, S. P., Johnson, N., Juettemann, T. & Flicek, P. R. The ensembl regulatory build. *Genome Biology* **16**, 56 (Mar. 2015).

137. Conne, B., Stutz, A. & Vassalli, J. D. The 3' untranslated region of messenger RNA: A molecular 'hotspot' for pathology? *Nature Medicine* **6**, 637–41 (June 2000).

138. Yang, H., Robinson, P. N. & Wang, K. Phenolyzer: phenotype-based prioritization of candidate genes for human diseases. *Nature Methods* **12**, 841–3 (Sept. 2015).

139. Kong, A. *et al.* Rate of de novo mutations and the importance of father's age to disease risk. *Nature* **488**, 471–5 (Aug. 2012).

140. Jung, I., Park, J., Choi, C. & Kim, D. Identification of novel trans-crosstalk between histone modifications via genome-wide analysis of maximal deletion effect. *Genes & Genomics* **37**, 693–701 (Aug. 2015).

141. Lee, D. *et al.* A method to predict the impact of regulatory variants from DNA sequence. *Nature Genetics* **47**, 955–61 (June 2015).

142. Whalen, S., Truty, R. M. & Pollard, K. S. Enhancer-promoter interactions are encoded by complex genomic signatures on looping chromatin. *Nature Genetics* **48**, 488–96 (Apr. 2016).

143. Rudan, M. V. *et al.* Comparative Hi-C Reveals that CTCF Underlies Evolution of Chromosomal Domain Architecture. *Cell Reports* **10**, 1297–1309 (Mar. 2015).

144. Mumbach, M. R. *et al.* Enhancer connectome in primary human cells identifies target genes of disease-associated DNA elements. *Nature Genetics* **49**, 1602–12 (Sept. 2017).

145. Whiffin, N. *et al.* Using high-resolution variant frequencies to empower clinical genome interpretation. *Genetics in Medicine* **19**, 1151–8 (May 2017).

146. Bell, C. J. *et al.* Carrier Testing for Severe Childhood Recessive Diseases by Next-Generation Sequencing. *Science Translational Medicine* **3**, 65ra4 (Jan. 2011).

147. He, H., Bai, Y., Garcia, E. A. & Li, S. *ADASYN: Adaptive synthetic sampling approach for imbalanced learning* in *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)* (June 2008), 1322–8.

148. Berthelot, C., Villar, D., Horvath, J. E., Odom, D. T. & Flicek, P. Complexity and conservation of regulatory landscapes underlie evolutionary resilience of mammalian gene expression. *bioRxiv* (Apr. 2017).

149. Villar, D. *et al.* Enhancer Evolution across 20 Mammalian Species. *Cell* **160**, 554–66 (Nov. 2017).

150. Inoue, F. & Ahituv, N. Decoding enhancers using massively parallel reporter assays. *Genomics* **106**, 159–64 (Sept. 2015).

151. Ferry, Q. *et al.* Diagnostically relevant facial gestalt information from ordinary photos. *eLife* **3**, e02020 (June 2014).

152. Eilbeck, K. *et al.* The Sequence Ontology: a tool for the unification of genome annotations. eng. *Genome Biology* **6**, R44 (Apr. 2005).

153. Hilbert, M. Big Data for Development: A Review of Promises and Challenges. *Development Policy Review* **34**, 135–74 (Jan. 2016).

Appendix A

Regulatory Mendelian Mutations

In this appendix all regulatory Mendelian mutations used for training and testing are listed. They are grouped by functional classes, each class in a separate table: the chromosomal position (GRCh37 assembly), the reference and alternative allele, the OMIM identifier of the syndrome caused by the mutation, the affected gene and the PubMed identifier (PMID) of the original publication where the mutation is linked to the disease. All listed positions of SNVs were used as positive set for the Mendelian data. The InDels were included in the test set of Genomiser.

Table A.1 contains the 42 identified enhancer mutations, Table A.2 the 142 promoter mutations, Table A.3 the 153 5' and Table A.4 the 43 3'UTR mutations, Table A.5 the 65 mutations in RNA genes, Table A.6 the 3 mutations in ICRs, and finally, Table A.7 the 5 mutations in miRNAs. All variants were identified by an extensive biocuration and only those variations were included where the publications provide plausible evidence of pathogenicity.

Table A.8 shows an overview of negative Mendelian data. Single-nucleotide sites were detected in the human genome that differs from the inferred ancestral primate genome based on the Ensembl EPO whole-genome alignments of six primate species [40, 42]. Low-confidence calls (ancestral state is supported by one sequence), marked by a lower case printing in the multiple sequence alignment, were removed. In addition variants with a frequency of higher than 5 % were filtered out. Finally the variants were annotated by Jannovar [109] v.0.14 using RefSeq [71] (annotation release 105) and only variants of NCV were used as final non-deleterious variant sites (negative positions). To sum up the extraction steps of negative sites, Table A.8 shows the distribution of variant categories for single nucleotide positions in human genome (reference release hg19) that differ from the inferred sequence of the last common primate ancestor.

The feature vector of the Mendelian data consists of 26 genomic attributes (see Section 3.2.1). Table A.9 describes all attributes together with the source.

Table A.1: Enhancer mutations. All coordinates refer to the GRCh37 assembly. A total of 42 enhancer mutations were identified from the medical literature.

Chr	Position	Ref	Alt	OMIM ID	Gene	PMID
chr1	21890663	G	A	241500	ALPL	10679946
chr1	209989478	C	CA	119300	IRF6	24442519
chr2	219524871	A	G	124000	BCS1L	19389488
chr7	156061506	C	T	142945	SHH	18836447
chr7	156583831	T	C	174500	SHH	17152067
chr7	156583949	G	C	174500	SHH	17152067
chr7	156583951	G	A	174500	SHH	22903933
chr7	156583968	T	TTAAGGAAGTGATT	174500	SHH	22495965
chr7	156584107	A	C	174500	SHH	20068592
chr7	156584153	T	C	174500	SHH	25382487
chr7	156584166	C	G	188740	SHH	19847792
chr7	156584166	C	T	188740	SHH	19847792
chr7	156584168	G	A	174500	SHH	24777739
chr7	156584174	G	A	174500	SHH	19519794
chr7	156584236	A	C	174500	SHH	20569257
chr7	156584241	A	G	174500	SHH	12837695
chr7	156584265	T	A	174500	SHH	12837695
chr7	156584273	C	T	174500	SHH	20569257
chr7	156584275	A	G	174500	SHH	18463159
chr7	156584465	G	C	174500	SHH	12837695
chr7	156584863	G	A	174500	SHH	17300748
chr8	11331747	G	T	613375	BLK	19667185
chr9	35657917	G	GCA	250250	RMRP	11207361
chr9	35657945	T	C	250250	RMRP	11207361
chr9	35658032	A	AGAGTAGT	250250	RMRP	21063072
chr10	23508305	A	G	615935	PTF1A	24212882
chr10	23508363	A	G	615935	PTF1A	24212882
chr10	23508365	A	G	615935	PTF1A	24212882
chr10	23508437	A	G	615935	PTF1A	24212882
chr10	23508446	A	C	615935	PTF1A	24212882
chr11	5271283	G	A	141749	HBG1	2423160
chr11	5276125	G	T	141749	HBG2	10335983
chr11	31685945	C	A	106210	PAX6	24290376
chr11	67250359	CCG	CAA	102200	AIP	20506337
chr12	114704515	G	T	142900	TBX5	22543974
chr16	209709	T	C	604131	HBA2	16728641
chr17	68676303	T	C	261800	SOX9	19234473
chrX	55054635	A	G	300751	ALAS2	23935018
chrX	70331494	G	A	300400	IL2RG	26525228
chrX	100641044	A	C	300755	BTK	9545398
chrX	105251968	G	A	314200	SERPINA7	25361180
chrY	2655719	C	T	400044	SRY	9452083

Table A.2: Promoter mutations. All coordinates refer to the GRCh37 assembly. A total of 142 promoter mutations were identified from the medical literature.

Chr	Position	Ref	Alt	OMIM ID	Gene	PMID
chr1	8021919	C	G	606324	PARK7	18722801
chr1	155271258	T	C	266200	PKLR	11054094
chr1	155271269	C	G	266200	PKLR	12393511
chr1	160001799	G	C	610293	PIGM	16767100

continued on the next page

Table A.2: Promoter mutations – *Continued from previous page*

Chr	Position	Ref	Alt	OMIM ID	Gene	PMID
chr1	171621877	A	C	137750	MYOC	15483649
chr1	173886568	G	C	613118	SERPINC1	22234719
chr1	228337561	A	G	608804	GJC2	20695017
chr2	47630106	G	C	120435	MSH2	11782355
chr2	128175983	A	G	612304	PROC	10942114
chr2	128175984	A	G	612304	PROC	10942114
chr2	128175988	T	A	176860	PROC	7592627
chr2	234668851	T	TCAT	606785	UGT1A1	26220753
chr3	12421189	A	G	604367	PPARG	15531525
chr3	48632780	G	A	226600	COL7A1	10980546
chr3	169482906	G	C	614743	TERC	22323451
chr3	169482947	G	C	127550	TERC	16670076
chr5	112073008	A	T	175100	APC	11606402
chr5	176836585	C	G	234000	F12	18832903
chr5	176836590	G	A	234000	F12	18832903
chr6	118869382	A	G	609909	PLN	12705874
chr6	118869423	A	C	609909	PLN	18241046
chr7	31003560	A	C	612781	GHRHR	11875102
chr7	117119337	T	G	219700	CFTR	7540587
chr7	117119923	G	T	219700	CFTR	10204861
chr7	117119984	G	A	277180	CFTR	10200050
chr8	11560787	T	C	614429	GATA4	22500510
chr8	11560864	C	A	614429	GATA4	22500510
chr8	11561283	C	T	614429	GATA4	22500510
chr8	11561369	G	T	614429	GATA4	22500510
chr8	11561399	AG	A	614429	GATA4	22500510
chr8	41655164	A	G	182900	ANK1	8640229
chr8	41655260	G	C	182900	ANK1	8640229
chr9	35658014	C	CCACGTCCTCAG CTTCA	250250	RMRP	17937437
chr9	104198194	C	T	229600	ALDOB	20882353
chr10	71075518	A	G	235700	HK1	19608687
chr10	89623365	G	T	158350	PTEN	17847000
chr10	89623373	C	G	158350	PTEN	17847000
chr10	89623462	G	A	158350	PTEN	17847000
chr10	127505271	A	G	263700	UROS	11254675
chr10	127505287	G	T	263700	UROS	11254675
chr10	127505291	G	T	263700	UROS	11254675
chr11	2182532	G	C	606176	INS	20133622
chr11	2182533	G	C	606176	INS	20133622
chr11	2182543	CAGATGGCGGGG GCTGAGGCTGCA	C	606176	INS	20133622
chr11	2193085	A	T	605407	TH	17696123
chr11	2193086	C	T	605407	TH	17696123
chr11	2193087	G	A	605407	TH	17696123
chr11	5248326	CTT	C	613985	HBB	16732578
chr11	5248329	T	G	613985	HBB	7076659
chr11	5248329	T	C	613985	HBB	26554738
chr11	5248330	T	C	613985	HBB	2458145
chr11	5248331	A	T	613985	HBB	3382401
chr11	5248331	A	G	613985	HBB	2741940
chr11	5248332	T	C	613985	HBB	3002527
chr11	5248333	G	T	613985	HBB	1729892
chr11	5248372	G	A	613985	HBB	21801233
chr11	5248374	T	A	613985	HBB	17516066
chr11	5248388	G	T	613985	HBB	1428943
chr11	5248388	G	C	613985	HBB	6280057
chr11	5248388	G	A	613985	HBB	2018842
chr11	5248389	G	A	613985	HBB	16732578

continued on the next page

Table A.2: Promoter mutations – *Continued from previous page*

Chr	Position	Ref	Alt	OMIM ID	Gene	PMID
chr11	5248389	G	T	613985	HBB	1986379
chr11	5248391	G	A	613985	HBB	1634236
chr11	5271201	G	A	141749	HBG1	1704803
chr11	5271204	C	T	141749	HBG1	3181130
chr11	5276186	A	G	141749	HBG2	2441598
chr11	5276213	G	C	141749	HBG2	6208955
chr11	17409772	C	A	601820	KCNJ11	15579781
chr11	17498513	G	C	256450	ABCC8	15579781
chr11	116708365	T	G	604091	APOA1	9974418
chr12	121416289	A	C	600496	HNF1A	9313764
chr12	121416354	T	C	600496	HNF1A	10649494
chr12	121416444	T	G	600496	HNF1A	22413961
chr12	121416446	TG	G	600496	HNF1A	10649494
chr12	121416448	G	C	600496	HNF1A	10333057
chr13	20767158	G	A	220290	GJB2	17660464
chr13	48877851	G	A	180200	RB1	1881452
chr13	48877856	T	G	180200	RB1	17096365
chr13	48877856	T	A	180200	RB1	17096365
chr13	48877860	G	T	180200	RB1	1881452
chr13	52585683	T	A	277900	ATP7B	20931554
chr13	60738072	C	T	609129	DIAPH3	20624953
chr13	113760062	C	G	227500	F7	9716591
chr13	113760095	T	G	227500	F7	8978290
chr13	113760097	T	G	227500	F7	12888866
chr13	113760101	C	T	227500	F7	11110717
chr13	113760124	A	C	227500	F7	12888866
chr14	73603081	GC	G	613694	PSEN1	20194882
chr16	56995796	G	A	143470	CETP	11397708
chr17	3539712	G	GT	219750	CTNS	11505338
chr17	3539712	G	T	219750	CTNS	11505338
chr17	3539720	G	C	219800	CTNS	11505338
chr17	4806453	C	T	608931	CHRNE	10382905
chr17	4806454	G	A	608931	CHRNE	10211467
chr17	42078968	C	A	237310	NAGS	21681857
chr18	77748580	ACGCCGTGCGTG CTGACGGCATGC GCGCGCGCTAG	A	608572	TXNL4A	25434003
chr19	11200031	ACTC	A	143890	LDLR	9610768
chr19	11200073	C	T	143890	LDLR	10484771
chr19	11200087	T	C	143890	LDLR	8589690
chr19	11200089	C	T	143890	LDLR	7937987
chr19	39137810	CA	C	603278	ACTN4	19666657
chr19	45449218	A	G	207750	APOC2	9017511
chr19	49468350	C	A	600886	FTL	19254706
chr20	42984264	G	A	125850	HNF4A	12235114
chr20	42984276	C	T	125850	HNF4A	20546279
chr20	42984299	T	C	125850	HNF4A	11590126
chr20	42984309	A	G	125850	HNF4A	20546279
chrX	37639262	A	C	306400	CYBB	8083361
chrX	37639264	T	C	306400	CYBB	8083361
chrX	37639266	C	T	306400	CYBB	9600921
chrX	37639267	C	T	306400	CYBB	9600921
chrX	38211584	A	G	311250	OTC	20127982
chrX	55057617	G	C	300751	ALAS2	12663458
chrX	70443029	T	G	302800	GJB1	8757034
chrX	70443029	T	C	302800	GJB1	16373087
chrX	70443031	G	C	302800	GJB1	15470753
chrX	70443099	C	T	302800	GJB1	8757034
chrX	70443185	G	A	302800	GJB1	21504505

continued on the next page

Table A.2: Promoter mutations – *Continued from previous page*

Chr	Position	Ref	Alt	OMIM ID	Gene	PMID
chrX	135730217	A	C	308230	CD40LG	17244160
chrX	138612869	G	C	306900	F9	1631558
chrX	138612869	G	T	306900	F9	24138812
chrX	138612869	G	A	306900	F9	24138812
chrX	138612871	A	G	306900	F9	24138812
chrX	138612872	C	G	306900	F9	24138812
chrX	138612872	C	T	306900	F9	7633432
chrX	138612874	T	G	306900	F9	8324220
chrX	138612875	T	A	306900	F9	3416069
chrX	138612875	T	C	306900	F9	24138812
chrX	138612876	G	C	306900	F9	7633432
chrX	138612889	G	A	306900	F9	23472758
chrX	138612889	G	C	306900	F9	24138812
chrX	138612890	A	T	306900	F9	1733855
chrX	138612890	A	G	306900	F9	23472758
chrX	146993366	G	C	300624	FMR1	20799337
chrX	146993405	T	C	300624	FMR1	20799337
chrX	146993444	A	G	300624	FMR1	20799337
chrX	153237261	A	G	309541	HCFC1	23000143
chrX	154251045	A	G	306700	F8	24372689
chrX	154251045	AG	G	306700	F8	24372689
chrX	154251048	A	T	306700	F8	24372689
chrX	154251082	T	C	306700	F8	22136525
chrX	154251084	A	C	306700	F8	19422439

Table A.3: 5'UTR mutations. All coordinates refer to the GRCh37 assembly. A total of 153 5'UTR mutations were identified from the medical literature.

Chr	Position	Ref	Alt	OMIM ID	Gene	PMID
chr1	10003560	A	T	608553	NMNAT1	26316326
chr1	10003561	C	T	608553	NMNAT1	26316326
chr1	21835920	C	T	241500	ALPL	10679946
chr1	55505180	C	A	603776	PCSK9	18559913
chr1	91487748	G	T	614167	ZNF644	24991186
chr1	113498814	C	T	610021	SLC16A1	17701893
chr1	171621759	G	A	137750	MYOC	15483649
chr1	209975361	T	A	119300	IRF6	12219090
chr2	25387652	G	T	609734	POMC	9620771
chr2	47630249	G	GA	120435	MSH2	11782355
chr2	73114549	G	A	612716	SPR	15241655
chr2	86564631	CATG	C	610250	REEP1	18321925
chr2	128176001	T	C	176860	PROC	7881411
chr2	203241250	GGC	GAT	178600	BMPR2	17641158
chr2	203241529	G	A	178600	BMPR2	19223935
chr3	37034932	C	G	609310	MLH1	17690979
chr3	37034997	C	T	609310	MLH1	12919137
chr3	37035012	C	A	609310	MLH1	21840485
chr3	49209095	C	T	236000	KLHDC8B	19706467
chr3	128598490	C	CTAAG	611126	ACAD9	17564966
chr3	184094078	C	A	187950	THPO	10583217
chr3	184094093	TC	T	187950	THPO	9694695
chr5	1295161	T	G	615134	TERT	23348503
chr5	14871567	G	A	118600	ANKH	12297987
chr5	36877039	CC	A	122470	NIPBL	16799922
chr5	63258025	CT	C	614674	HTR1A	21990073
chr5	149340544	T	C	222600	SLC26A2	10482955

continued on the next page

Table A.3: 5'UTR mutations – *Continued from previous page*

Chr	Position	Ref	Alt	OMIM ID	Gene	PMID
chr6	88299677	T	C	611523	RARS2	25809939
chr6	137143759	C	T	215100	PEX7	12325024
chr7	107301201	T	C	274600	SLC26A4	17503324
chr7	117120064	C	G	219700	CFTR	21847140
chr7	117120115	C	T	211400	CFTR	21837768
chr8	19796671	T	G	238600	LPL	9017514
chr8	19796711	G	C	238600	LPL	9017514
chr8	21988118	T	C	146550	HR	19122663
chr8	21988119	A	G	146550	HR	19122663
chr8	21988140	C	G	146550	HR	19122663
chr8	21988146	C	T	146550	HR	19122663
chr8	21988148	G	A	146550	HR	24261346
chr8	21988149	G	C	146550	HR	19122663
chr8	21988151	A	T	146550	HR	19122663
chr8	21988202	G	T	146550	HR	19122663
chr8	21988215	G	A	146550	HR	19122663
chr8	21988219	C	T	146550	HR	19122663
chr8	21988220	A	T	146550	HR	19122663
chr8	21988220	A	G	146550	HR	19122663
chr8	21988221	T	C	146550	HR	19122663
chr8	41655127	TCA	T	182900	ANK1	20479128
chr9	21974860	C	A	155601	CDKN2A	9916806
chr9	37422743	TGC	TAT	260000	GRHPR	25410531
chr9	116037909	TTGTCAGTGACG CACTTCC	T	615922	PRPF4	24419317
chr9	130616643	C	T	187300	ENG	22192717
chr9	130616644	G	A	187300	ENG	16752392
chr9	130616761	G	A	187300	ENG	22192717
chr10	27389371	G	A	188000	ANKRD26	21211618
chr10	27389373	G	A	188000	ANKRD26	21467542
chr10	27389374	G	T	188000	ANKRD26	21467542
chr10	27389376	T	G	188000	ANKRD26	21467542
chr10	27389380	A	C	188000	ANKRD26	21211618
chr10	27389381	A	C	188000	ANKRD26	21467542
chr10	27389382	T	C	188000	ANKRD26	21467542
chr10	27389383	C	T	188000	ANKRD26	21211618
chr10	27389389	C	T	188000	ANKRD26	21211618
chr10	71038467	G	A	605285	HK1	19536174
chr11	299504	G	A	610967	IFITM5	22863190
chr11	2182419	T	G	606176	INS	20133622
chr11	5248257	C	G	613985	HBB	14687034
chr11	5248269	G	C	613985	HBB	8562944
chr11	5248280	C	T	613985	HBB	1536956
chr11	5248291	GA	G	613985	HBB	7803275
chr11	5248294	G	A	613985	HBB	18473240
chr11	47470715	G	C	616326	RAPSN	12651869
chr11	47470726	T	C	616326	RAPSN	12651869
chr11	57365055	C	T	106100	SERPING1	8755917
chr11	61735061	T	A	615517	FTH1	11389486
chr11	88070895	G	T	245000	CTSC	23108224
chr11	118955588	CG	C	176000	HMBS	11071386
chr12	12870319	CTTCC	C	610755	CDKN1B	23555276
chr12	12870744	GAGAG	G	610755	CDKN1B	22129891
chr12	15130918	G	C	610024	PDE6H	15629837
chr13	29233225	TC	T	601952	POMP	20226437
chr13	48877899	G	C	180200	RB1	8570221
chr13	48877900	G	T	180200	RB1	25999316
chr13	52585551	T	G	277900	ATP7B	9199563
chr13	52585596	G	T	277900	ATP7B	9199563

continued on the next page

Table A.3: 5'UTR mutations – *Continued from previous page*

Chr	Position	Ref	Alt	OMIM ID	Gene	PMID
chr13	52585606	T	G	277900	ATP7B	20931554
chr14	37130036	G	A	604625	PAX9	21443745
chr14	55369403	G	A	128230	GCH1	10825351
chr14	76447266	C	T	107970	TGFB3	15639475
chr17	66508599	G	A	160980	PRKAR1A	12424709
chr17	75316275	G	C	162100	SEPT9	16186812
chr19	11200072	C	T	143890	LDLR	15303010
chr19	11200083	C	T	143890	LDLR	11792717
chr19	11200085	C	G	143890	LDLR	21538688
chr19	11200085	C	T	143890	LDLR	21538688
chr19	11200086	C	G	143890	LDLR	17625505
chr19	11200086	CT	C	143890	LDLR	14616764
chr19	11200089	C	G	143890	LDLR	21538688
chr19	11200104	T	C	143890	LDLR	25248394
chr19	11200105	C	T	143890	LDLR	15303010
chr19	11200124	T	C	143890	LDLR	25248394
chr19	35773456	G	A	613313	HAMP	15198949
chr19	39138352	C	T	603278	ACTN4	19666657
chr19	49468579	C	G	600886	FTL	11238302
chr19	49468589	T	C	600886	FTL	21410535
chr19	49468594	C	G	600886	FTL	15223007
chr19	49468597	G	C	600886	FTL	12200611
chr19	49468597	G	T	600886	FTL	12730114
chr19	49468597	G	A	600886	FTL	9226182
chr19	49468598	C	T	600886	FTL	16900584
chr19	49468599	T	C	600886	FTL	12730114
chr19	49468601	C	A	600886	FTL	9414313
chr19	49468601	C	T	600886	FTL	23421845
chr19	49468601	C	G	600886	FTL	12670350
chr19	49468602	AAC	A	600886	FTL	12730114
chr19	49468602	A	G	600886	FTL	12670350
chr19	49468604	C	G	600886	FTL	15234655
chr19	49468604	C	A	600886	FTL	14662596
chr19	49468604	C	T	600886	FTL	12730114
chr19	49468605	A	G	600886	FTL	12730114
chr19	49468606	GTGTTTGGACGG AACAG	G	600886	FTL	12730114
chr19	49468606	G	C	600886	FTL	7492760
chr19	49468608	G	A	600886	FTL	15690351
chr19	49468611	T	G	600886	FTL	19887780
chr19	49468612	G	A	600886	FTL	12730114
chr19	49468614	A	C	600886	FTL	20578964
chr19	49468616	G	C	600886	FTL	10759702
chr19	49468617	G	C	600886	FTL	23421845
chr19	49468621	A	T	600886	FTL	16395671
chr20	2451408	G	T	117650	SNRPB	25047197
chr20	6103422	T	C	173650	FERMT1	25156791
chr22	19710933	C	G	231200	GP1BB	8703016
chr22	40742514	T	C	103050	ADSL	12016589
chrX	49114969	C	A	304790	FOXP3	16371377
chrX	49834101	G	A	300009	CLCN5	19673950
chrX	68049525	T	G	304110	EFNB1	23335590
chrX	68049525	T	C	304110	EFNB1	23335590
chrX	100641212	T	C	300755	BTK	9445504
chrX	103031893	C	T	312080	PLP1	8723686
chrX	123480184	C	T	308240	SH2D1A	9771704
chrX	138612900	T	A	306900	F9	24138812
chrX	138612901	T	C	306900	F9	24138812
chrX	138612902	T	C	306900	F9	24138812

continued on the next page

Table A.3: 5'UTR mutations – *Continued from previous page*

Chr	Position	Ref	Alt	OMIM ID	Gene	PMID
chrX	138612903	C	G	306900	F9	24138812
chrX	138612905	CA	C	306900	F9	2917196
chrX	138612906	A	G	306900	F9	24138812
chrX	138612907	A	G	306900	F9	2917196
chrX	138612907	A	C	306900	F9	24138812
chrX	146993615	G	C	300624	FMR1	11897823
chrX	153363065	TCTCCTCCTCGC	T	312750	MECP2	15034579
chrX	153991099	C	G	305000	DKC1	11379875
chrX	154250939	C	T	306700	F8	16972227
chrX	154251046	G	A	306700	F8	16972227

Table A.4: 3'UTR mutations. All coordinates refer to the GRCh37 assembly. A total of 43 3'UTR mutations were identified from the medical literature.

Chr	Position	Ref	Alt	OMIM ID	Gene	PMID
chr1	11083408	G	A	612069	TARDBP	19618195
chr1	26143316	T	C	602771	SEPN1	16498447
chr1	45481173	G	T	176100	UROD	11295834
chr1	91381763	C	T	614167	ZNF644	21695231
chr1	91382343	G	C	614167	ZNF644	21695231
chr1	100661453	T	G	248600	DBT	20570198
chr1	156028185	C	A	610798	LAMTOR2	17195838
chr2	69553299	G	T	610542	GFPT1	25765662
chr2	71913729	T	A	253601	DYSF	16705711
chr2	86444173	C	T	610250	REEP1	16826527
chr2	86444180	C	A	610250	REEP1	16826527
chr2	86444209	G	A	610250	REEP1	18321925
chr7	81384514	CTTTCATCATC	C	608265	HGF	19576567
chr7	81384516	TTCA	T	608265	HGF	19576567
chr8	11422122	G	T	613375	BLK	19667185
chr8	22058957	T	C	614856	BMP1	25214535
chr11	2181023	T	C	606176	INS	20133622
chr11	5246714	TTTTATT	T	613985	HBB	1374896
chr11	5246715	T	C	613985	HBB	1374896
chr11	5246717	T	C	613985	HBB	1856830
chr11	5246718	A	G	613985	HBB	4018033
chr11	5246718	A	T	613985	HBB	15481893
chr11	5246720	T	G	613985	HBB	11722440
chr11	5246796	T	G	613985	HBB	22734587
chr11	46761055	G	A	188050	F2	8916933
chr12	102796022	A	T	608747	IGF1	14684690
chr13	100638514	T	A	609637	ZIC2	22859937
chr13	100638825	T	C	609637	ZIC2	22859937
chr13	100638890	T	A	609637	ZIC2	22859937
chr13	100638902	A	G	609637	ZIC2	22859937
chr14	76425035	G	A	107970	TGFB3	15639475
chr16	223690	TAA	T	604131	HBA2	26193977
chr16	223691	A	G	613978	HBA2	1581238
chr16	223693	A	G	613978	HBA2	11480787
chr16	31202807	G	A	608030	FUS	23847048
chr16	31202818	G	A	608030	FUS	23847048
chr16	31202867	C	T	608030	FUS	23847048
chr16	31202869	G	A	608030	FUS	23847048
chr21	27253648	AAT	A	605714	APP	25828868
chrX	22266301	A	G	307800	PHEX	18625346
chrX	48683304	A	T	300863	HDAC6	20181727

continued on the next page

Table A.4: 3'UTR mutations – *Continued from previous page*

Chr	Position	Ref	Alt	OMIM ID	Gene	PMID
chrX	49106917	T	C	304790	FOXP3	19471859
chrX	49106919	T	C	304790	FOXP3	11685453

Table A.5: RNA gene mutations. All coordinates refer to the GRCh37 assembly. A total of 65 RNA gene mutations were identified from the medical literature.

Chr	Position	Ref	Alt	OMIM ID	Gene	PMID
chr2	122288463	C	T	616651	RNU4ATAC	26522830
chr2	122288468	C	T	616651	RNU4ATAC	26522830
chr2	122288485	G	A	210710	RNU4ATAC	21474760
chr2	122288492	G	A	616651	RNU4ATAC	26522830
chr2	122288501	G	A	210710	RNU4ATAC	25735804
chr2	122288503	G	A	616651	RNU4ATAC	26522830
chr2	122288505	G	A	210710	RNU4ATAC	21474761
chr2	122288505	G	C	210710	RNU4ATAC	21474761
chr2	122288506	G	A	210710	RNU4ATAC	21474761
chr2	122288508	C	G	210710	RNU4ATAC	21474761
chr2	122288510	G	A	210710	RNU4ATAC	22581640
chr2	122288521	G	C	210710	RNU4ATAC	22581640
chr2	122288566	G	A	210710	RNU4ATAC	21474760
chr2	122288573	T	C	616651	RNU4ATAC	26522830
chr2	122288579	G	A	210710	RNU4ATAC	22581640
chr3	169482399	C	T	614743	TERC	12676774
chr3	169482441	G	C	127550	TERC	11574891
chr3	169482456	CGG	C	614743	TERC	15550482
chr3	169482472	T	C	614743	TERC	21931702
chr3	169482524	C	A	614743	TERC	18753630
chr3	169482526	G	A	MIM	TERC	17936651
chr3	169482527	C	T	614743	TERC	12676774
chr3	169482544	C	T	614743	TERC	12676774
chr3	169482592	C	T	614743	TERC	25346280
chr3	169482645	G	C	614743	TERC	14630445
chr3	169482667	C	T	614743	TERC	21931702
chr3	169482669	G	A	614743	TERC	17640862
chr3	169482671	C	T	614743	TERC	17640862
chr3	169482673	T	G	614743	TERC	21931702
chr3	169482706	C	T	614743	TERC	21436073
chr3	169482720	T	C	614743	TERC	21931702
chr3	169482732	T	G	614743	TERC	15550482
chr3	169482733	G	A	614743	TERC	14630445
chr3	169482735	AAGTC	A	614743	TERC	17640862
chr3	169482740	AGC	ACT	127550	TERC	11574891
chr3	169482742	C	A	614743	TERC	21931702
chr3	169482749	A	T	127550	TERC	18931339
chr3	169482751	CAG	C	127550	TERC	15319288
chr3	169482751	C	T	614743	TERC	17392301
chr3	169482769	A	C	614743	TERC	21931702
chr3	169482777	G	C	614743	TERC	15319288
chr3	169482782	C	T	614743	TERC	21931702
chr3	169482812	T	C	614743	TERC	17460043
chr3	169482814	G	A	127550	TERC	18931339
chr9	35657754	G	A	250250	RMRP	16244706
chr9	35657754	G	C	250250	RMRP	16244706
chr9	35657767	G	A	250250	RMRP	16244706
chr9	35657771	C	T	250250	RMRP	16244706
chr9	35657773	T	C	250250	RMRP	16244706

continued on the next page

Table A.5: RNA gene mutations – *Continued from previous page*

Chr	Position	Ref	Alt	OMIM ID	Gene	PMID
chr9	35657795	A	G	250250	RMRP	16244706
chr9	35657802	G	C	250250	RMRP	16244706
chr9	35657820	G	A	250250	RMRP	16244706
chr9	35657820	G	GA	250250	RMRP	14569125
chr9	35657822	C	T	250250	RMRP	16244706
chr9	35657833	C	A	250250	RMRP	16244706
chr9	35657847	C	T	250250	RMRP	17937437
chr9	35657869	C	G	250250	RMRP	16244706
chr9	35657888	C	T	250250	RMRP	16244706
chr9	35657889	G	A	250250	RMRP	12107819
chr9	35657918	C	T	250250	RMRP	26279652
chr9	35657951	A	G	250250	RMRP	16244706
chr9	35657952	G	A	250250	RMRP	16244706
chr9	35657975	C	T	250250	RMRP	16244706
chr9	35657988	C	G	250250	RMRP	26279652
chr9	35658011	G	A	250250	RMRP	16244706

Table A.6: Imprinting control region mutations. All coordinates refer to the GRCh37 assembly. A total of 3 ICR mutations were identified from the medical literature.

Chr	Position	Ref	Alt	OMIM ID	Gene	PMID
chr11	2020124	C	A	180860	H19	20007505
chr11	2023019	A	G	130650	H19	20007505
chr11	2023048	A	C	130650	H19	21863054

Table A.7: MiRNA gene mutations. All coordinates refer to the GRCh37 assembly. A total of 5 miRNA gene mutations were identified from the medical literature.

Chr	Position	Ref	Alt	OMIM ID	Gene	PMID
chr7	129414553	A	G	613074	MIR96	22038834
chr7	129414596	G	T	613074	MIR96	19363479
chr7	129414597	C	T	613074	MIR96	19363479
chr9	73424964	G	A	616722	MIR204	26056285
chr15	79502186	C	T	614303	MIR184	21996275

Table A.8: Detailed information of the negative Mendelian data. Distribution of variant categories for single nucleotide positions in *Homo sapiens* that differ from the inferred sequence of the last common primate ancestor. An asterisk (∗) marks variant categories that were used as negative dataset for hyperSMURF. Therefore only variants that match to the SO-terms SO:0001970, SO:0001969, SO:0001623, SO:0001624, SO:0001631, SO:0001632, SO:0001628, SO:0001782, SO:0001566, SO:0002018, SO:0002011, SO:0002017, and SO:0001627 were chosen [152]. All of the terms belong to non-coding regions. Fixed variants in the population, defined as a more than 5 % presence of the ancestral allele in the individuals of the 1KG, were rejected. Variants were annotated by Jannovar version 0.14 [109] with transcript definitions from the RefSeq [71] (annotation release 105).

Category	All	High quality	Fixed	High-quality & Fixed
CDS	49,599	44,885	43,420	38,706
CDS (syn)	57,708	52,656	52,189	47,137
Unclassified sequence variant	11,408	10,675	11,408	10,675
Splice	12,520	12,553	12,430	11,218
5'UTR	764,719	711,934	692,943	640,158*
3'UTR	121,014	112,740	109,034	100,760*
Intron	5,954,014	5,600,983	5,383,124	5,030,093*
Upstream/Downstream	224,128	198,554	203,737	178,163*
Noncoding (exon)	67,704	58,236	62,038	52,570
Noncoding (intron)	858,848	782,486	782,720	706,358*
Intergenic	9,908,106	8,989,024	9,018,749	8,099,667*
Total	18,029,768	16,574,726	16,371,792	14,915,505

Table A.9: Genomic attributes used for the Mendelian data. 26 genomic attributes used as feature vector for the Mendelian data (see Section 3.2.1) are listed. For each attribute a short description and the source (UCSC table identifier of the UCSC Table Browser [111] or web-link) is given.

Attribute	Description
GCContent	GC-content in a window of ±75 bp
CpGperGC	Percentage of island that is C or G. UCSC table `cpgIslandExt`
CpGperCpG	Percentage of island that is CpG. UCSC table `cpgIslandExt`
CpGobsExp	Ratio of observed to expected CpG in island. UCSC table `cpgIslandExt`
priPhyloP46way	Primate PhyloP score. `http://hgdownload.soe.ucsc.edu/goldenPath/hg19/phyloP46way/primates`
verPhyloP46way	Vertebrate PhyloP. `http://hgdownload.soe.ucsc.edu/goldenPath/hg19/phyloP46way/vertebrate`
mamPhyloP46way	Mammalian PhyloP score. `http://hgdownload.soe.ucsc.edu/goldenPath/hg19/phyloP46way/placentalMammals`
priPhastCons46way	Primate PhastCons conservation score `http://hgdownload.soe.ucsc.edu/goldenPath/hg19/phastCons46way/primates`
verPhastCons46way	Vertebrate PhastCons conservation score `http://hgdownload.soe.ucsc.edu/goldenPath/hg19/phastCons46way/vertebrate`
mamPhastCons46way	Mammalian PhastCons conservation score `http://hgdownload.soe.ucsc.edu/goldenPath/hg19/phastCons46way/placentalMammals`
GerpRS	GERP++ element score `http://mendel.stanford.edu/SidowLab/downloads/gerp/hg19.GERP_elements.tar.gz`
GerpRSpv	GERP++ element p-Value `http://mendel.stanford.edu/SidowLab/downloads/gerp/hg19.GERP_elements.tar.gz`
EncH3K27Ac	Maximum ENCODE H3K27 acetylation level `http://hgdownload.cse.ucsc.edu/goldenPath/hg19/encodeDCC/wgEncodeRegMarkH3k27ac`
EncH3K4Me1	Maximum ENCODE H3K4 methylation level `http://hgdownload.cse.ucsc.edu/goldenPath/hg19/encodeDCC/wgEncodeRegMarkH3k4me1`
EncH3K4Me3	Maximum ENCODE H3K4 trimethylation level `http://hgdownload.cse.ucsc.edu/goldenPath/hg19/encodeDCC/wgEncodeRegMarkH3k4me3`
DnaseClusteredHyp	DnaseClustered V3 hypersensitivity score `http://hgdownload.cse.ucsc.edu/goldenPath/hg19/encodeDCC/wgEncodeRegDnaseClustered`
DnaseClusteredScore	Number of DnaseClustered V3 hypersensitive cells `http://hgdownload.cse.ucsc.edu/goldenPath/hg19/encodeDCC/wgEncodeRegDnaseClustered`
fantom5Perm	FANTOM 5 permissive enhancers `http://enhancer.binf.ku.dk/presets/permissive_enhancers.bed`
fantom5Robust	FANTOM5 robust enhancers `http://enhancer.binf.ku.dk/presets/robust_enhancers.bed`
numTFBSConserved	Number of overlapping transcription factor binding sites. UCSC table `tfbsConsSites`
rareVar	Number of rare 1KG variants ($\leq$0.5 % AF) in a window of ±500 bp
commonVar	Number of common 1KG variants (>0.5 % AF) in a window of ±500 bp
fracRareCommon	Ratio rare to common variants

ISCApath	Overlapping ISCA CNVs
	`http://www.ncbi.nlm.nih.gov/dbvar/studies/nstd75`
	`http://www.ncbi.nlm.nih.gov/dbvar/studies/nstd46`
	`http://www.ncbi.nlm.nih.gov/dbvar/studies/nstd37`

dbVARCount	Overlapping dbVAR CNVs `ftp://ftp.ncbi.nlm.nih.gov/pub/dbVar/data/Homo_sapiensby_assembly/GRCh37.p13/gvf/GRCh37.p13.remap.all.germline.ucsc.gvf.gz`
DGVCount	Overlapping DGV CNVs `http://dgv.tcag.ca/dgv/docs/GRCh37_hg19_variants_2014-10-16.txt`

Appendix B

HyperSMURF Performance

In Appendix B I show additional results of hyperSMURF according to the three datasets: Mendelian, GWAS and eQTL. The first part shows the results of the univariate linear regression model on the Mendelian and GWAS data, discussed previously in Section 3.4. Afterwards all results of the optimal parameter search from Section 3.5.1 are shown and finally an in-depth comparison of the performance on the different datasets between hyperSMURF and the retrained learners is made.

Informative features

Table B.1 shows the results of the univariate linear regression model on the Mendelian data. Performance measurement was done using the cytoband-aware 10-fold CV. The experiment was repeated 100-times and AUROC and AUPRC were measured. The same table is available for the GWAS data (Table B.2). Here, only the ten best and ten less formative features in terms of the AUPRC were shown instead of all 1842 features. A detailed explanation about the results can be found in Section 3.4.

Table B.1: Informative Mendelian genomic features according to univariate logistic regression. Results are computed by cytoband-aware 10-fold CV using a logistic regression model with Mendelian data. The experiment is repeated 100 times, and AUPRC together with AUROC means and the standard deviations are listed. Genomic features were ranked according to the estimated AUPRC. The sources of the data for the considered genomic features are listed in Table A.9.

Genomic Feature	AUPRC	AUROC
mamPhyloP46way	0.24018 ± 0.01600	0.92077 ± 0.00025
verPhyloP46way	0.11705 ± 0.00635	0.92450 ± 0.00028
priPhyloP46way	0.02750 ± 0.00541	0.96333 ± 0.00036
priPhastCons46way	0.00951 ± 0.00965	0.89562 ± 0.00316
mamPhastCons46way	0.00157 ± 0.00020	0.85717 ± 0.00171
verPhastCons46way	0.00121 ± 0.00019	0.84922 ± 0.00177
DnaseClusteredHyp	0.00101 ± 0.00032	0.73808 ± 0.00414
CpGperCpG	0.00096 ± 0.00040	0.61934 ± 0.00257
CpGobsExp	0.00093 ± 0.00036	0.61912 ± 0.00286
CpGperGC	0.00081 ± 0.00019	0.61898 ± 0.00235
numTFBSConserved	0.00062 ± 0.00011	0.62967 ± 0.01099
EncH3K4Me3	0.00058 ± 0.00004	0.81133 ± 0.01060
GerpRS	0.00039 ± 0.00004	0.84027 ± 0.00724
DnaseClusteredScore	0.00032 ± 0.00005	0.73790 ± 0.00222
GCContent	0.00032 ± 0.00002	0.82201 ± 0.00091
rareVar	0.00024 ± 0.00010	0.50697 ± 0.00233
EncH3K27Ac	0.00024 ± 0.00001	0.79388 ± 0.00877
EncH3K4Me1	0.00006 ± 0.00000	0.73490 ± 0.00819
fracRareCommon	0.00005 ± 0.00000	0.67170 ± 0.00201
commonVar	0.00004 ± 0.00000	0.58627 ± 0.00563
ISCApath	0.00004 ± 0.00000	0.50986 ± 0.01101
GerpRSpv	0.00004 ± 0.00000	0.49735 ± 0.02040
fantom5Perm	0.00003 ± 0.00000	0.49468 ± 0.02366
fantom5Robust	0.00003 ± 0.00000	0.49362 ± 0.02293
dbVARCount	0.00002 ± 0.00000	0.44047 ± 0.03022
DGVCount	0.00002 ± 0.00000	0.44047 ± 0.03022

Table B.2: Informative GWAS genomic features according to univariate logistic regression. Results are computed by cytoband-aware 10-fold CV using a logistic regression model with GWAS data. The experiment is repeated 100 times. AUPRC together with AUROC means and the standard deviations are listed. Genomic features were ranked according to the estimated AUPRC and the ten best and ten less informative features are shown.

Genomic Feature	AUPRC	AUROC
PhyloP	0.36978 ± 0.01004	0.76382 ± 0.00019
PhastCons	0.04262 ± 0.00332	0.93689 ± 0.00094
HepG2|Sin3Ak-20|None|PDIFF	0.00415 ± 0.00006	0.51082 ± 0.00137
HeLa-S3|TBP|None|PDIFF	0.00414 ± 0.00010	0.50653 ± 0.00381
PFSK-1|FOXP2|None|PDIFF	0.00392 ± 0.00006	0.51070 ± 0.00329
HeLa-S3|GTF2F1|None|PDIFF	0.00391 ± 0.00009	0.50962 ± 0.00636
HepG2|Mxi1|None|PDIFF	0.00382 ± 0.00019	0.50738 ± 0.00276
A549|ETS1|EtOH_0.02pct|PDIFF	0.00381 ± 0.00012	0.51528 ± 0.00577
HepG2|MYBL2|None|PDIFF	0.00373 ± 0.00010	0.51412 ± 0.00084
GM12878|MAZ|None|PDIFF	0.00373 ± 0.00009	0.51408 ± 0.00264
⋮	⋮	⋮
NHDF-Ad|H3K9me3|None|PDIFF	0.00137 ± 0.00004	0.52076 ± 0.01015
NT2-D1|ZNF274|None|PDIFF	0.00137 ± 0.00003	0.50806 ± 0.00856
H1-hESC|H3K9me3|None|PDIFF	0.00136 ± 0.00003	0.51356 ± 0.00811
GM12878|ZNF274|None|PDIFF	0.00134 ± 0.00002	0.50384 ± 0.00432
HeLa-S3|BRF2|None|PDIFF	0.00132 ± 0.00004	0.49894 ± 0.00515
GM08714|ZNF274|None|PDIFF	0.00131 ± 0.00001	0.49587 ± 0.00388
HeLa-S3|SPT20|None|PDIFF	0.00130 ± 0.00003	0.50425 ± 0.00888
K562|KAP1|None|PDIFF	0.00129 ± 0.00001	0.50126 ± 0.00326
U2OS|SETDB1|None|PDIFF	0.00129 ± 0.00001	0.49855 ± 0.00371
K562|SETDB1|MNaseD|PDIFF	0.00127 ± 0.00001	0.48961 ± 0.00455

Optimal Parameters

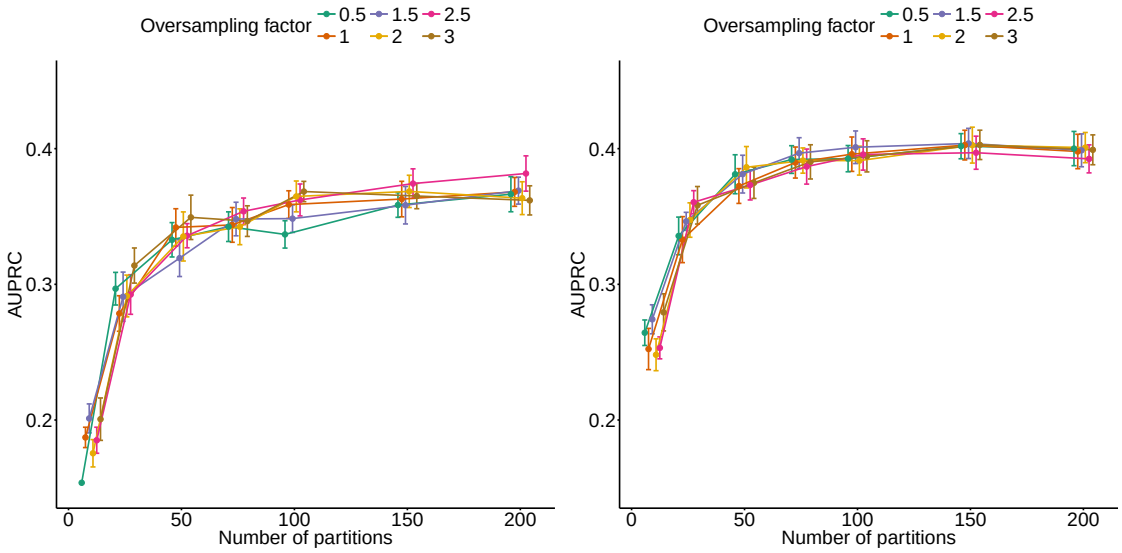
Several experiments were done to find optimal parameters for hyperSMURF on the Mendelian data. Therefore several settings per variable of hyperSMURF were chosen (see Table 3.2 for all tested parameters and values) and an internal 9-fold cytoband-aware CV was performed for each round of the cytoband-aware 10-fold CV. This procedure of automatic tuning of the parameters is described in Section 3.3.1 in more detail.

Figure B.1 shows the AUPRC of the parameter tuning with a fixed undersampling factor ($m =$ 1, 2 and 3), the number of partitions n and different fixed oversampling factors. The oversampling factor curves show clearly that an increase in partition size results in an increase of the AUPRC. But the performance increase saturates at around $n = 100$ partitions for all oversampling and undersampling factors.

The oversampling factor seems to be more diffuse and no optimal value can be determined clearly. But using oversampling (oversampling factor $>$0) will always increase the performance, as previously showed in Table 3.3. The best undersampling factor can be derived from Figure B.2. Here the same performance is shown but with a fixed oversampling factor of $o =$ 0.5, 1, 1.5, 2, 2.5 and 3 and the undersampling parameters used as curves. For all oversampling settings an undersampling factor of $m = 3$ gives the best AUPRC. It is important to say that larger values of m were not tested because this will result in larger datasets and increase the runtime. But a tendency can still be seen here. In all figures the curves with $m = 2$ and $m = 3$ are closer together than $m = 1$ to $m = 2$. This gives a hint of saturation on larger undersampling factors.

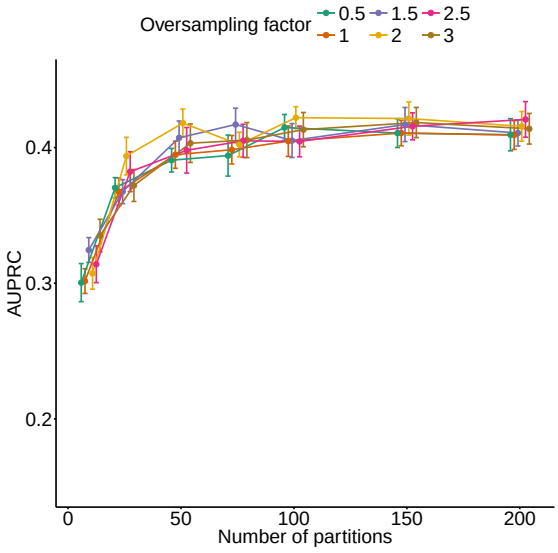
The number of nearest neighbors of SMOTE seems to play a minor role. No optimal k could be figured out in Figure 3.4b. Finally the experiments of the RF learner in Figure 3.4a suggest a lower forest size because no performance increase can be measured on forests larger than $t = 25$. Random feature of $d = 5$ or $d = 6$ seems to be a good choice.

Bringing all these observations together we can agree on a standard setting of hyperSMURF. Because we are dealing with huge datasets the used standard setting keeps the computational burden to a minimum while still remaining at high performance. This is why these standard hyperSMURF settings, listed in Table 3.2, were chosen. For completeness Table B.3 shows all optimal hyperSMURF settings after the automatic tuning process. The optimal hyperSMURF increases the AUPRC from 0.4319 to 0.4503.



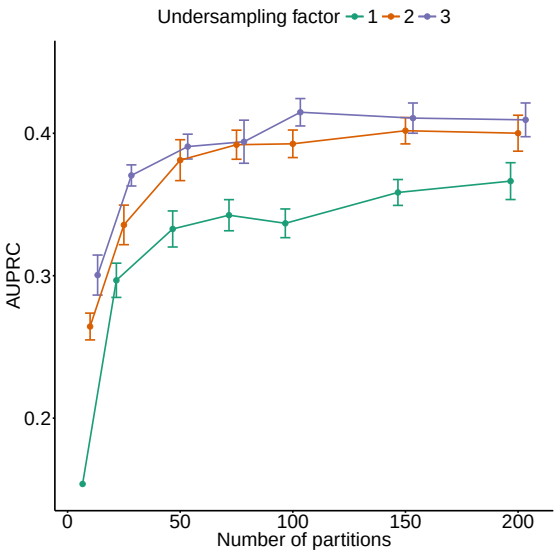
(a) Undersampling factor $m = 1$.

(b) Undersampling factor $m = 2$.

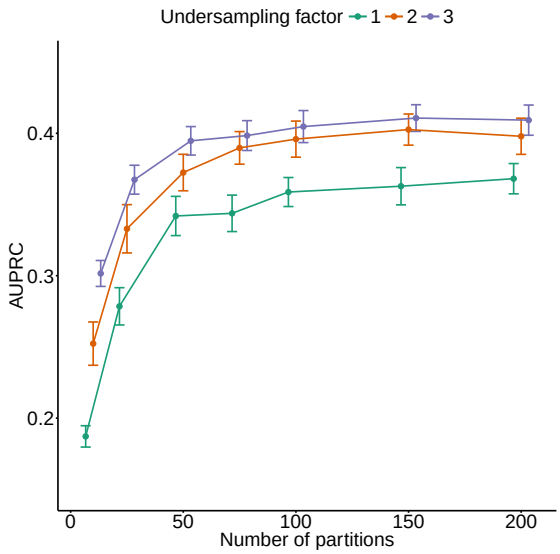


(c) Undersampling factor $m = 3$.

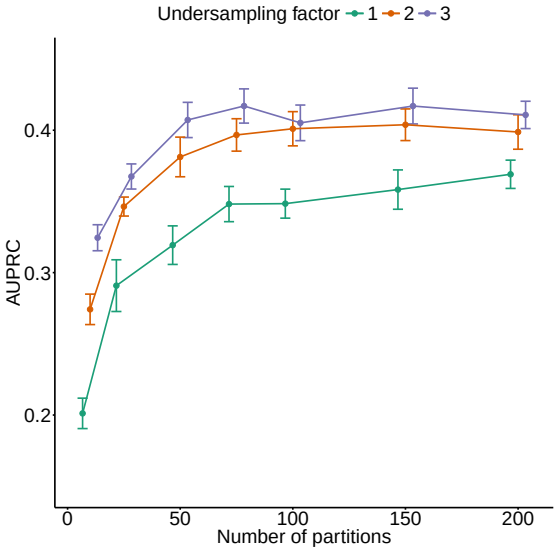
Figure B.1: HyperSMURF parameter tuning with fixed undersampling factor. AUPRC as a function of different hyperSMURF partition sizes generated by internal 9-fold cytogenetic band-aware CV using Mendelian data. Curves show different oversampling factors with a fixed undersampling factor m for each figure: **(a)** $m = 1$, **(b)** $m = 2$ and **(c)** $m = 3$. Error bars represent the standard deviation between ten repetitions of internal 9-fold CV. The undersampling factor is the ratio of negative examples with respect to positives. Negative examples were randomly sampled without replacement from each partition of the data (see Section 3.3.1 for details).



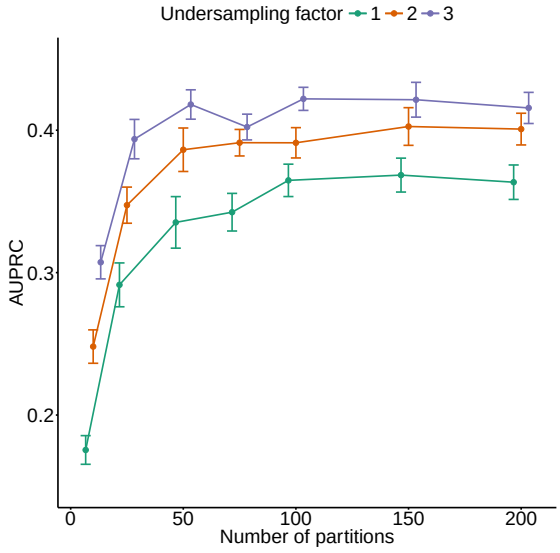
(a) Oversampling factor $f = 0.5$.



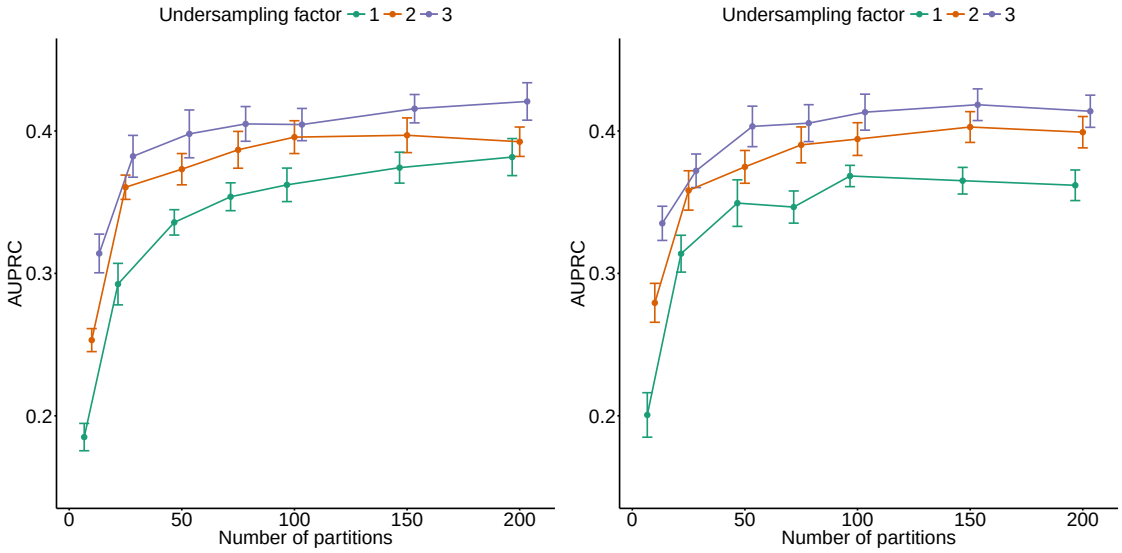
(b) Oversampling factor $f = 1$.



(c) Oversampling factor $f = 1.5$.



(d) Oversampling factor $f = 2$.



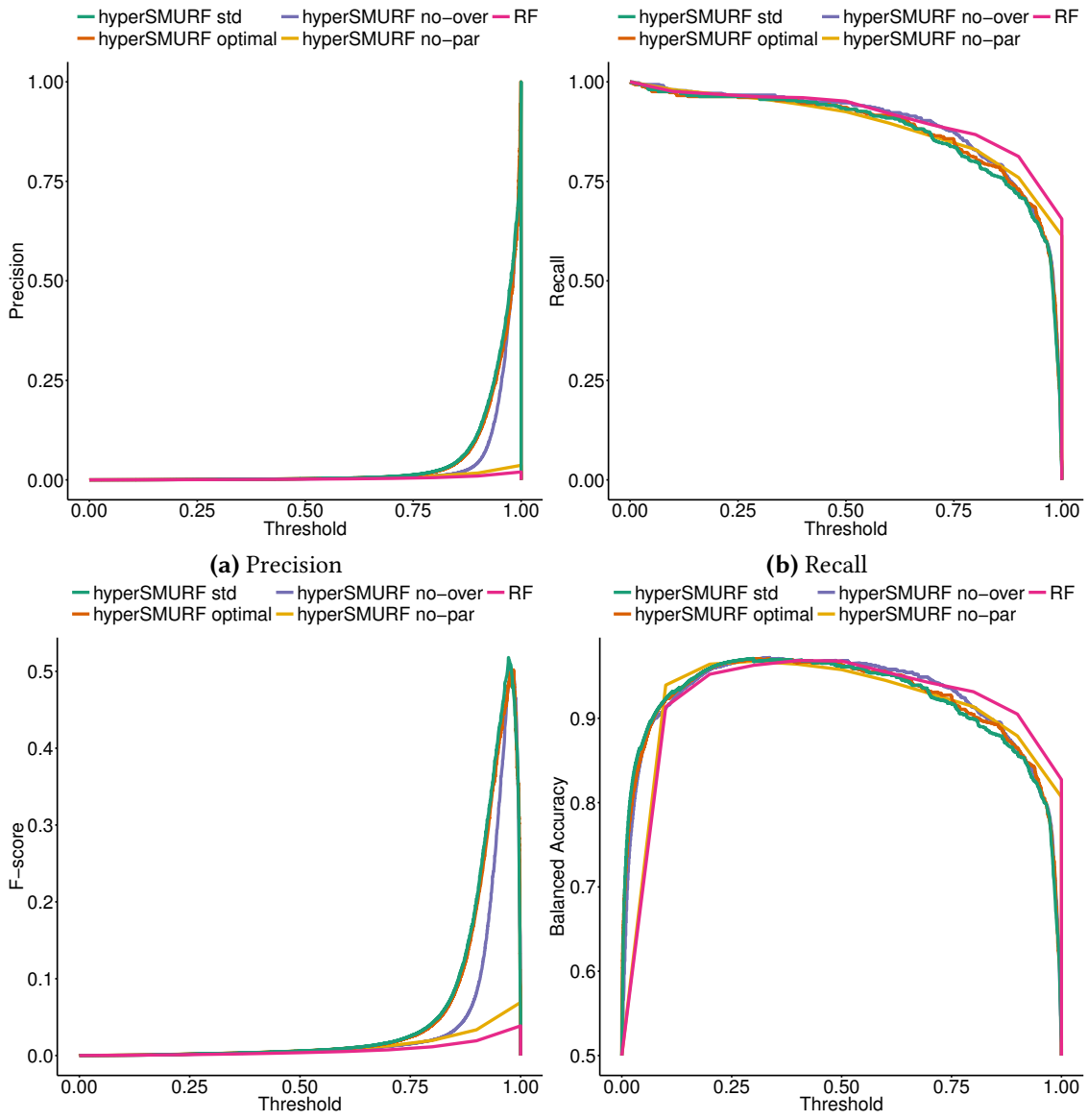
(e) Oversampling factor $f = 2.5$.

(f) Oversampling factor $f = 3$.

Figure B.2: HyperSMURF parameter tuning with fixed oversampling factor. AUPRC as a function of different hyperSMURF partition sizes generated by internal 9-fold cytogenetic band-aware CV using Mendelian data. Curves show different undersampling factors with a fixed oversampling factor f for each figure: **(a)**, $f = 0.5$, **(b)** $f = 1$, **(c)** $f = 1.5$, **(d)** $f = 2$, **(e)** $f = 2.5$, and **(f)** $f = 3$. Error bars represent the standard deviation between ten repetitions of internal 9-fold CV using different folds. The oversampling factor is the ratio of synthetic positive examples generated through the SMOTE algorithm with respect to the available number of positive examples (see Section 3.3.1 for details).

Table B.3: Optimal hyperSMURF parameters on the Mendelian data. For each fold, parameters were selected using internal 9-fold cytogenetic band-aware CV (see Section 3.3.1). The best parameters in terms of the AUPRC for each partitioning of the cytoband-aware 10-fold CV are shown.

		Fold									
Par.	**Description**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
n	Number of partitions	75	75	75	50	100	200	50	100	200	50
f	SMOTE oversampling factor	1.5	2.5	1.5	1	2	2	2.5	2.5	1	3
k	SMOTE k-nearest neighbor	9	5	4	6	5	3	4	5	6	10
m	Undersampling factor	3	3	3	3	3	3	3	3	3	3
t	Forest size	20	10	50	50	20	75	100	20	50	20
d	Random tree features	7	6	6	6	5	6	5	5	6	5



(a) Precision

(b) Recall

(c) F_1-score

(d) Balanced accuracy

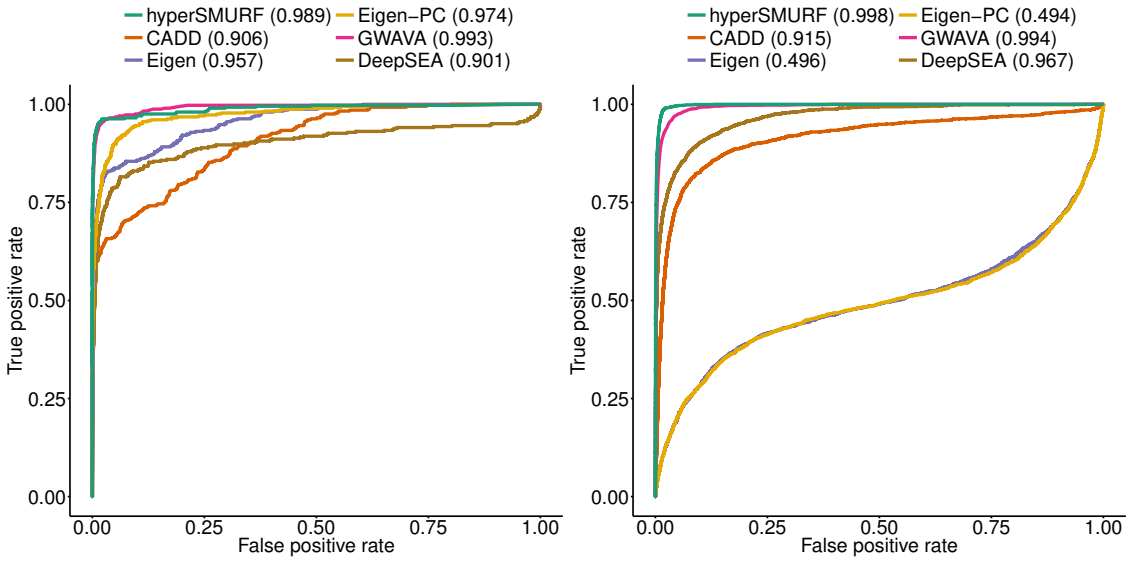
Figure B.3: Precision, recall and F-score comparison of hyperSMURF, an optimized hyperSMURF and only subparts of hyperSMURF. "HyperSMURF std" is the full hyperSMURF algorithm; "HyperSMURF optimal" uses the best automated selected parameters for every CV step (see Section 3.3.1); "HyperSMURF no-over" is hyperSMURF with no oversampling; "HyperSMURF no-par" is hyperSMURF with no partitioning and therefore no hyper-ensemble approach. "RF" is the classical Random Forest ensemble. In every setting a subsampling of the majority (negative) class to three times the cardinality of the minority class (positives) was performed. Other parameters were set to default (see Table 3.2).

Performance on Non-Coding Variants

This section contains some additional plots about the performance of hyperSMURF on Mendelian, GWAS and eQTL data. The main discussion about the results can be found in Sections 3.5.3 and 3.5.4. Here I present an in-depth discussion about performance in terms of precision, recall (sensitivity), balanced accuracy and F_1-score (F-Measure).

State-Of-The-Art Methods

Figure B.4 shows the ROC curves of hyperSMURF and the retrained learners from CADD, GWAVA, Eigen, Eigen-PC, and DeepSEA on the Mendelian and GWAS data. As expected GWAVA has a similar performance to hyperSMURF in both datasets because it has the same base-learner and, the main reason, it adopts an imbalance-aware strategy. Eigen and Eigen-PC have a reasonably good ROC curve on the Mendelian data but completely fail on the GWAS data. The reason for this might be the huge feature set where only two distinct feature groups are present. To show that the ROC curves are significantly different a one-sided DeLong test was performed and the results are listed in Table B.4. These results underline the previous observations that only GWAVA has a similar ROC performance to hyperSMURF.



(a) Mendelian data ROC curves.

(b) GWAS data ROC curves.

Figure B.4: Comparison across retrained learners from different non-coding scoring methods by ROC curves. Values in brackets show the AUROC. **(a)** ROC curves of the retrained learners from the different non-coding scoring methods on Mendelian data; **(b)** ROC curves of the retrained learners from the different non-coding scoring methods on regulatory GWAS data.

Table B.4: Statistical comparison of ROC curves. Comparison of the AUROC between hyperSMURF (Mendelian data AUC $= 0.99$; GWAS data AUC $= 1.00$) and the retrained learners from other scoring methods: CADD, GWAVA, Eigen, Eigen-PC, and DeepSEA. Tests were performed using the one-sided DeLong test [119]. An asterisk ($*$) marks a statistically significant difference (significance level $\alpha = 0.05$).

(a) Mendelian data

Score	AUROC	P-Value
HyperSMURF	0.99	–
CADD	0.91	0.04 *
GWAVA	0.99	1.00
Eigen	0.96	0.25
Eigen-PC	0.97	0.38
DeepSEA	0.90	9.13×10^{-96} *

(b) GWAS data

Score	AUROC	P-Value
HyperSMURF	1.00	–
CADD	0.92	9.83×10^{-177} *
GWAVA	0.99	0.46
Eigen	0.96	1.02×10^{-233} *
Eigen-PC	0.97	9.55×10^{-234} *
DeepSEA	0.90	6.52×10^{-162} *

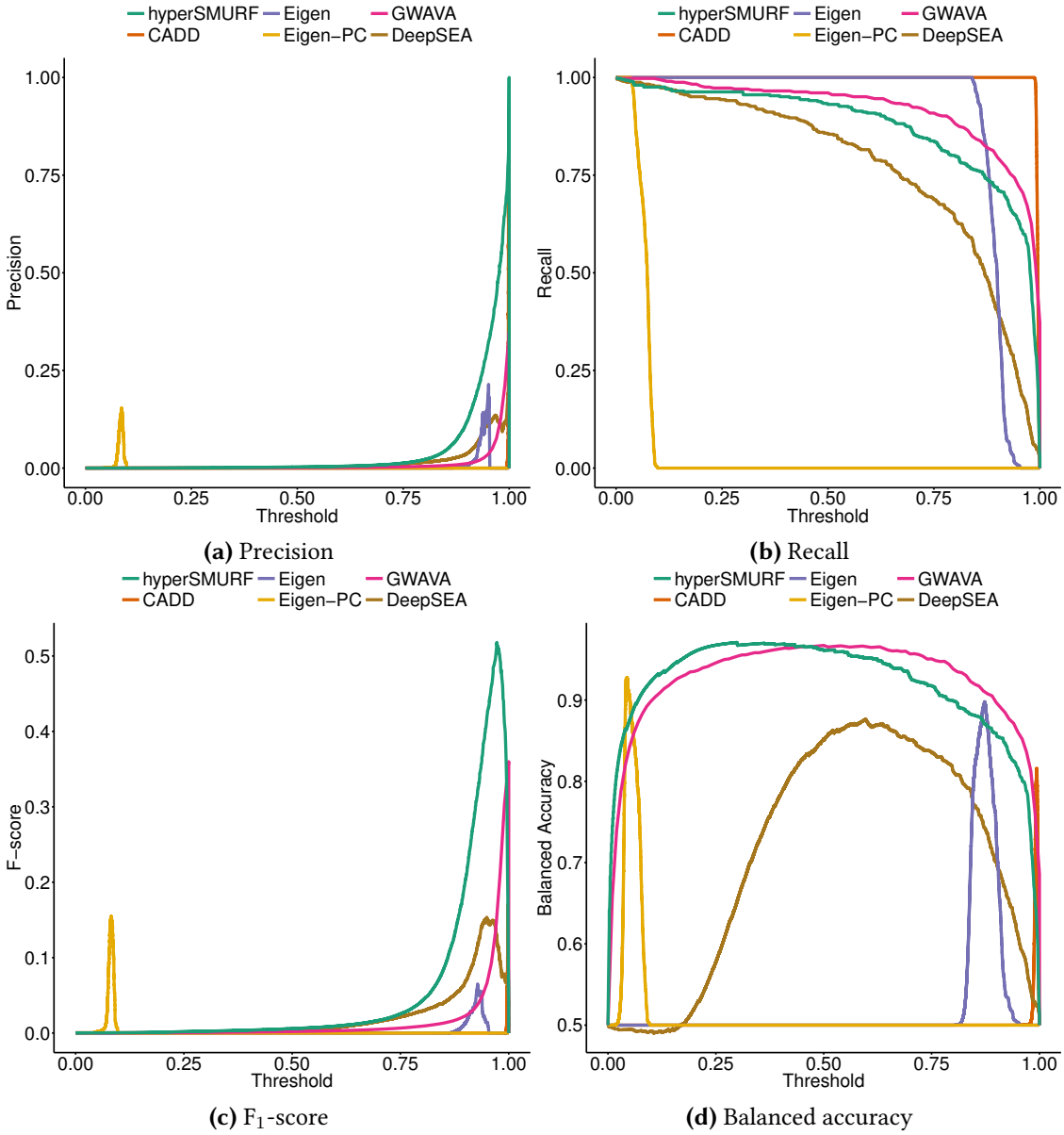
In order to obtain a common basis for the comparison, all scores were rescaled in the range $[0, 1]$ through a simple linear transformation (normalized score). On the Mendelian data hyperSMURF has the best precision across the normalized scores (Figure B.5a) and the retrained SVM of CADD the best recall (Figure B.5b). HyperSMURF is the only method that achieves both: a high precision and a high recall. Therefore it achieves the best F_1-score (Figure B.5c).

The second best method in terms of the F_1-score is GWAVA (Figure B.5c). It has a good sensitivity but only a high precision for normalized scores close to one (Figures B.5a and B.5b). Therefore the F_1-score has a maximum of around 0.35 gained by scores close to one (Figures B.6e and B.7e). Because GWAVA adopts a balancing approach the balanced accuracy is good over the complete range of scores (Figure B.5d).

The retrained SVM of CADD has no recall (Figure B.5b). Thus, the F_1-score is always at zero except of a final peek close to one (Figures B.6b and B.7b).

Eigen and Eigen-PC have a similar behavior. Both have a small peak in precision where at the same range the recall drops (Figures B.5a and B.5b). Of course at this range they gain the best F_1-score (Figure B.5c). But interestingly the range of the peeks between the methods are at different values (Figures B.6c, B.6d, B.7c and B.7d). Eigen has its peak at higher normalized scores and Eigen-PC at low scores. This means that for Eigen-PC several negative variants get an extremely high score in contrast to the regulatory mutations. Therefore it can only be used to remove negative variants instead of finding regulatory Mutations with high values.

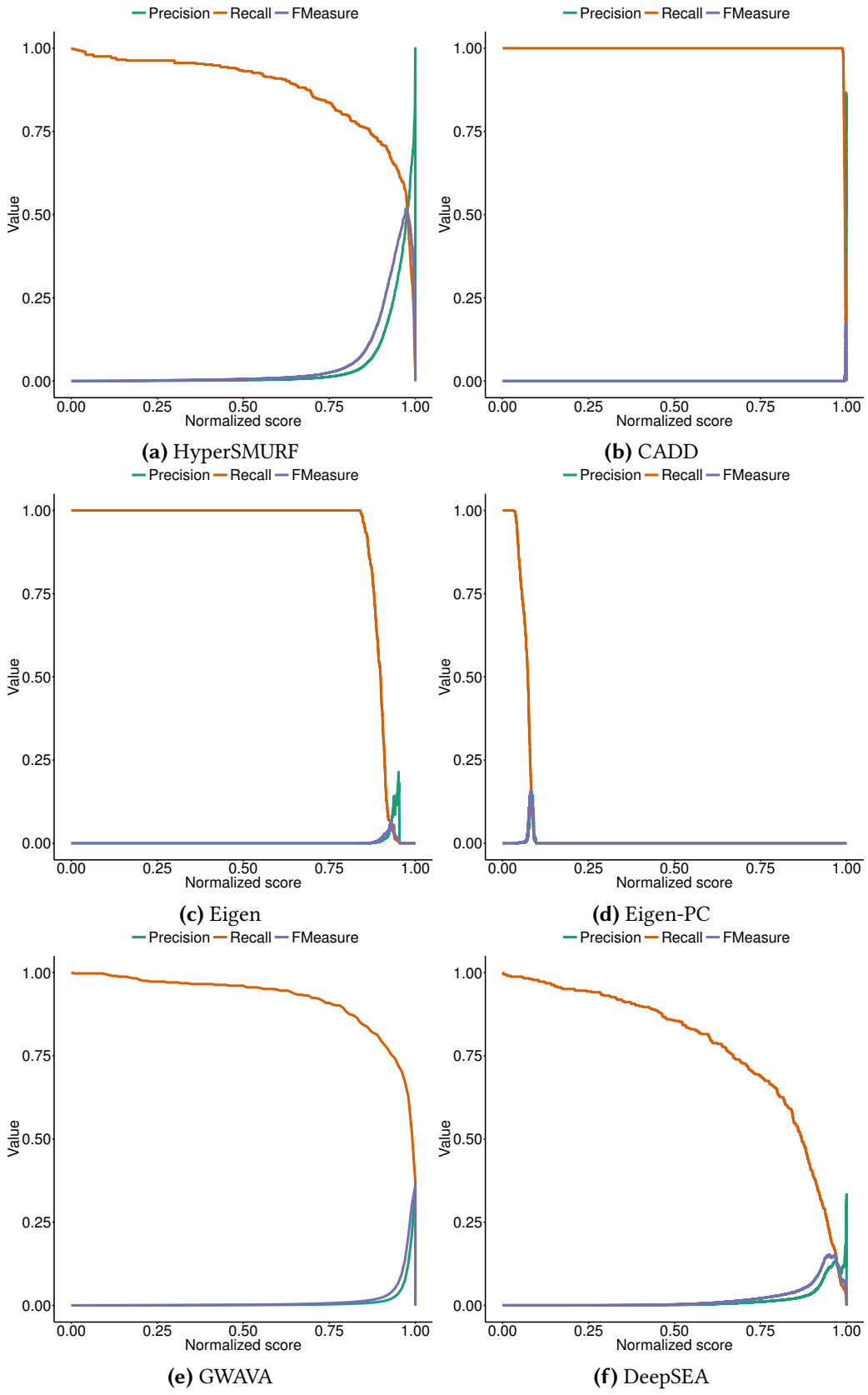
Finally DeepSEA has a lower recall with a low maximum precision of around 0.25 close to one (Figures B.6f and B.7f). In summary, this analysis shows that hyperSMURF has a substantially better performance than the other methods. Only the imbalance-aware learning method GWAVA shows that it is able to deal with the data. But the imbalance factor is too high to reach the hyperSMURF performance.



(a) Precision **(b)** Recall

(c) F_1-score **(d)** Balanced accuracy

Figure B.5: Precision, recall, and F-score comparison across retrained learners from different non-coding scoring with Mendelian data. HyperSMURF, CADD, Eigen, Eigen-PC, GWAVA, and DeepSEA performance by varying the normalized score threshold.

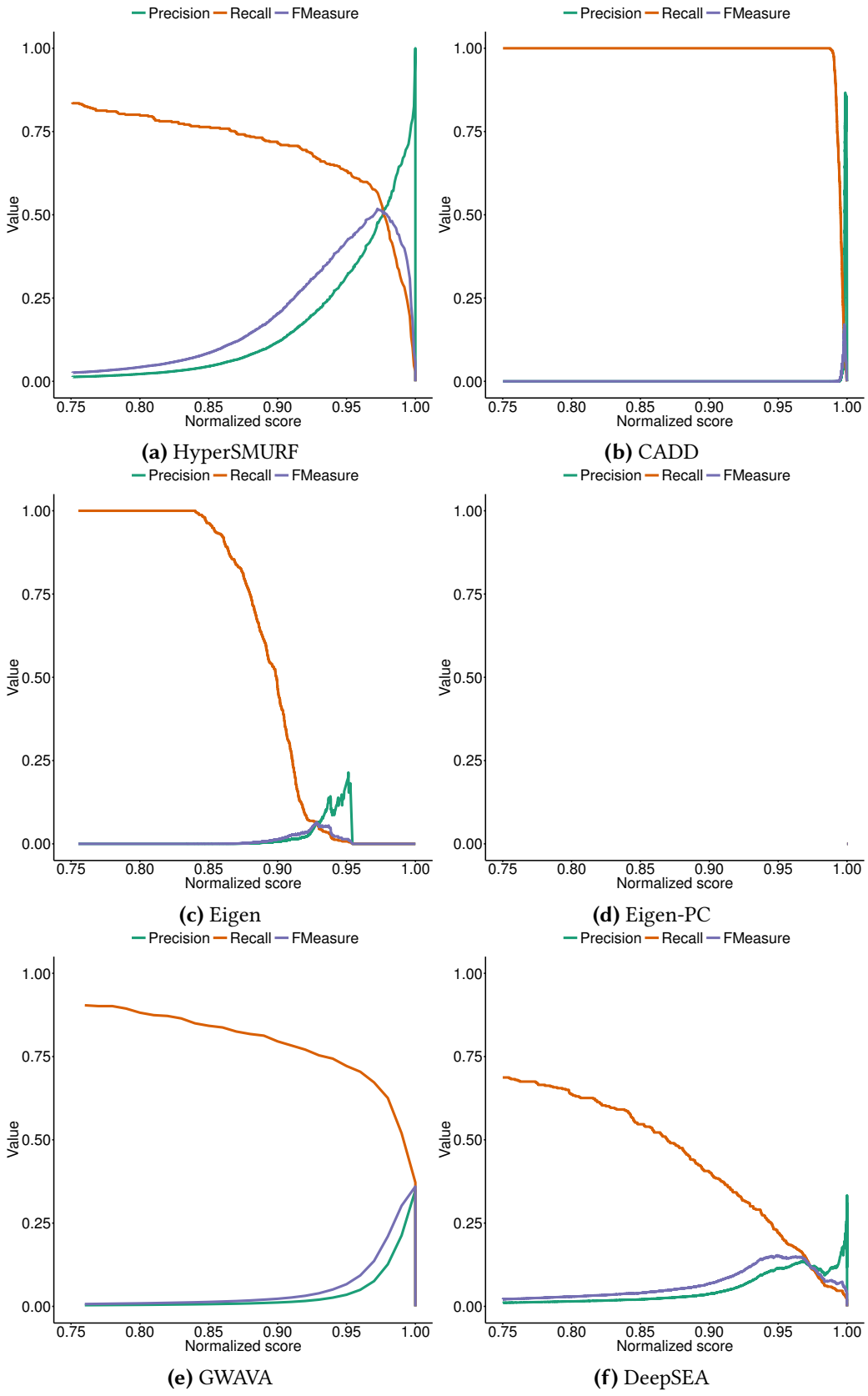


(a) HyperSMURF **(b)** CADD

(c) Eigen **(d)** Eigen-PC

(e) GWAVA **(f)** DeepSEA

Figure B.6: Precision, recall and F-score values of the normalized scores from retrained non-coding score learners on the Mendelian data.



(a) HyperSMURF **(b)** CADD

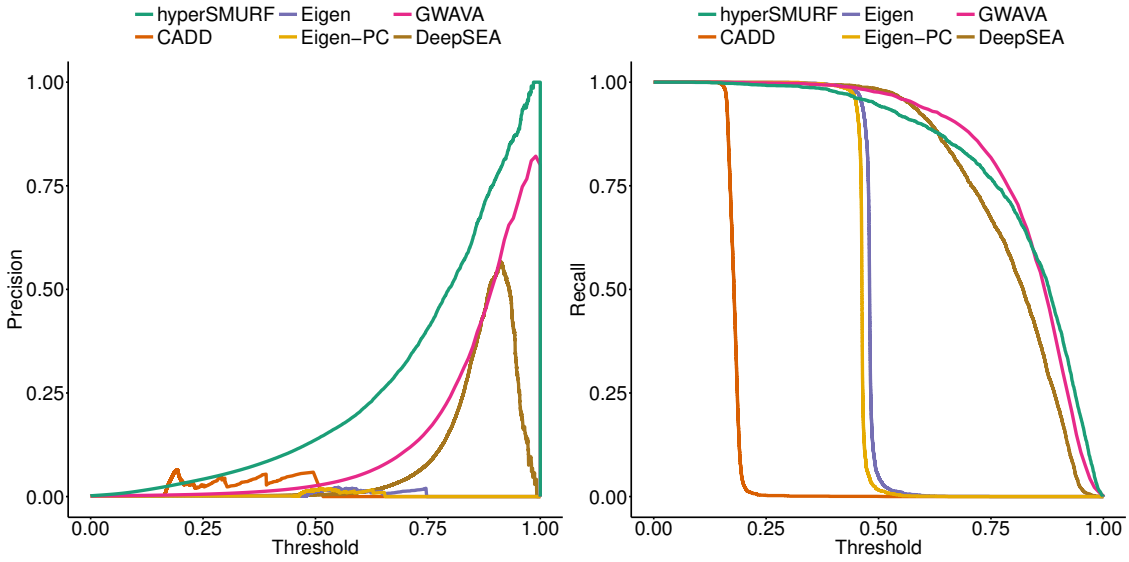
(c) Eigen **(d)** Eigen-PC

(e) GWAVA **(f)** DeepSEA

Figure B.7: Details of precision, recall and F-score within the interval $[0.75, 1]$ of the normalized scores from retrained non-coding score learners on the Mendelian data.

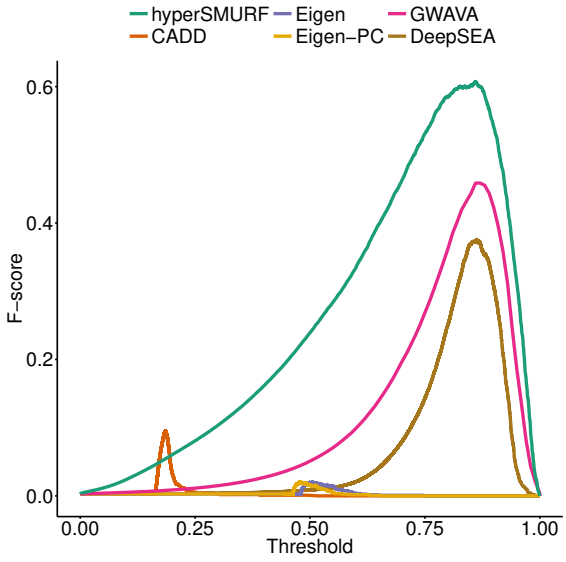
On the GWAS data we can define two groups of methods. The first group contains hyperSMURF, GWAVA and DeepSEA. All of them have similarly good recall values (Figure B.8b) and a good precision (descending order, Figure B.8a). All other scores, forming the second group, have precision values close to zero and reduced sensitivity.

All well performing methods have good F_1-scores with maximums ranging from 0.4 to 0.6 (Figure B.8). GWAVA and DeepSEA have very similar performance on precision, recall and F_1-score (compare with Figures B.9e, B.9f, B.10e and B.10f). But GWAVA has the best balanced accuracy because of the imbalance-aware method (Figure B.8d). For the GWAS data no different behavior of Eigen and Eigen-PC can be detected (compare with Figures B.9c, B.9d, B.10c and B.10d). Both methods seem to have problems with the huge amount of features in contrast to the low amount of correlated groups (conservation scores, logfold and diff features).

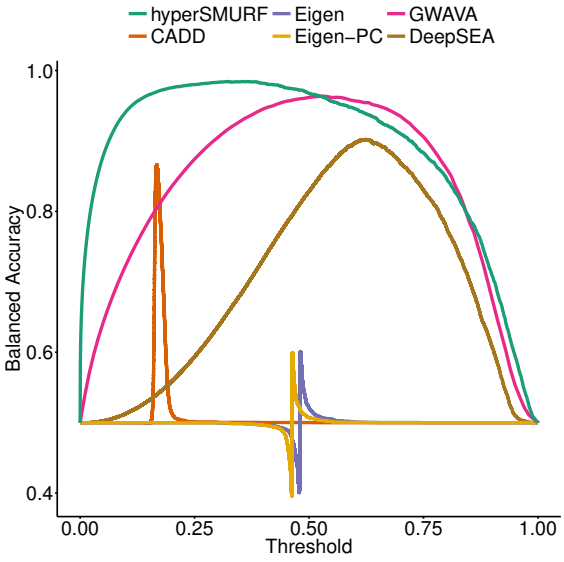


(a) Precision

(b) Recall

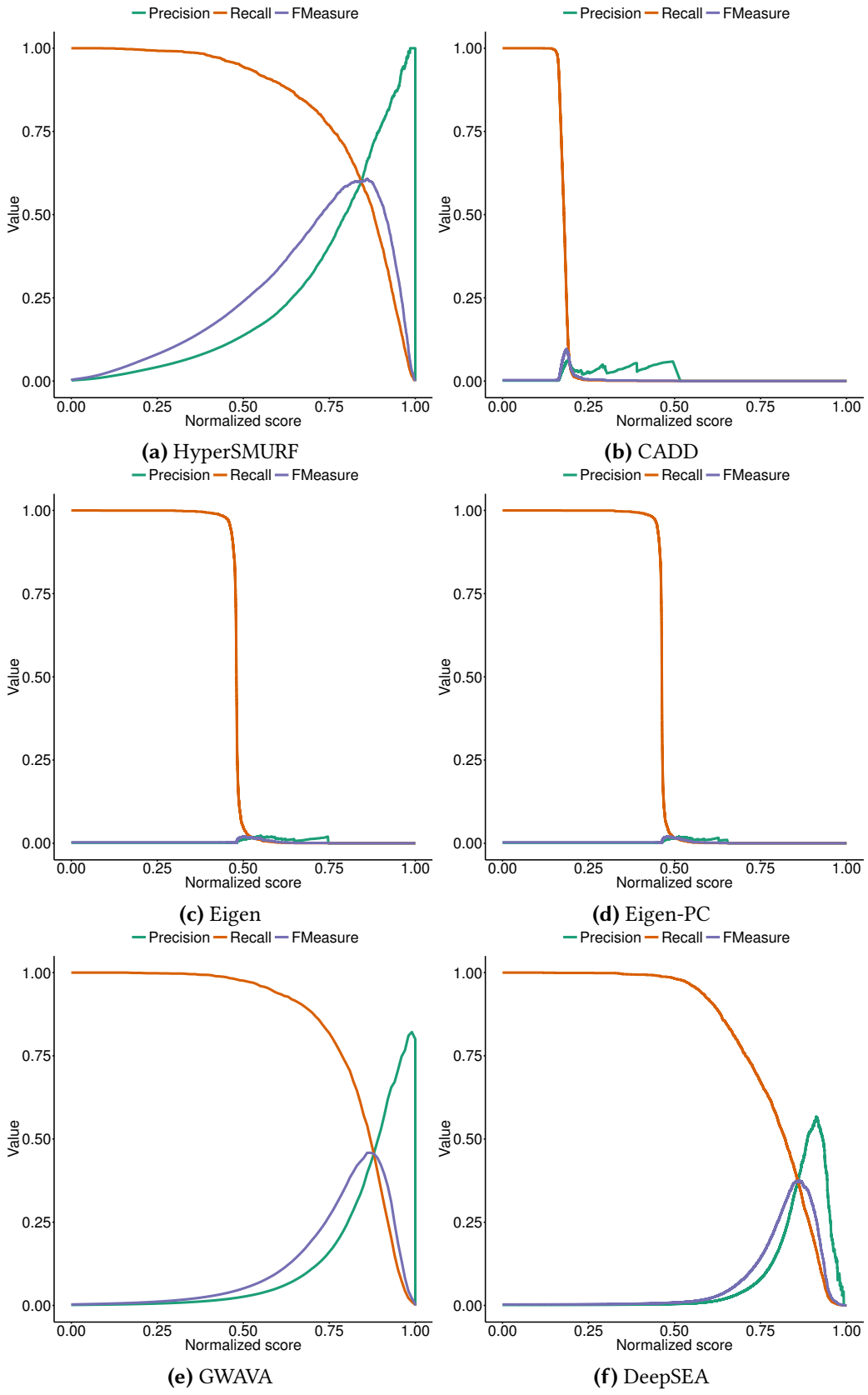


(c) F_1-score



(d) Balanced accuracy

Figure B.8: Precision, recall, and F-score comparison across retrained learners from different non-coding scoring with GWAS data. HyperSMURF, CADD, Eigen, Eigen-PC, GWAVA, and DeepSEA performance by varying the normalized score threshold.

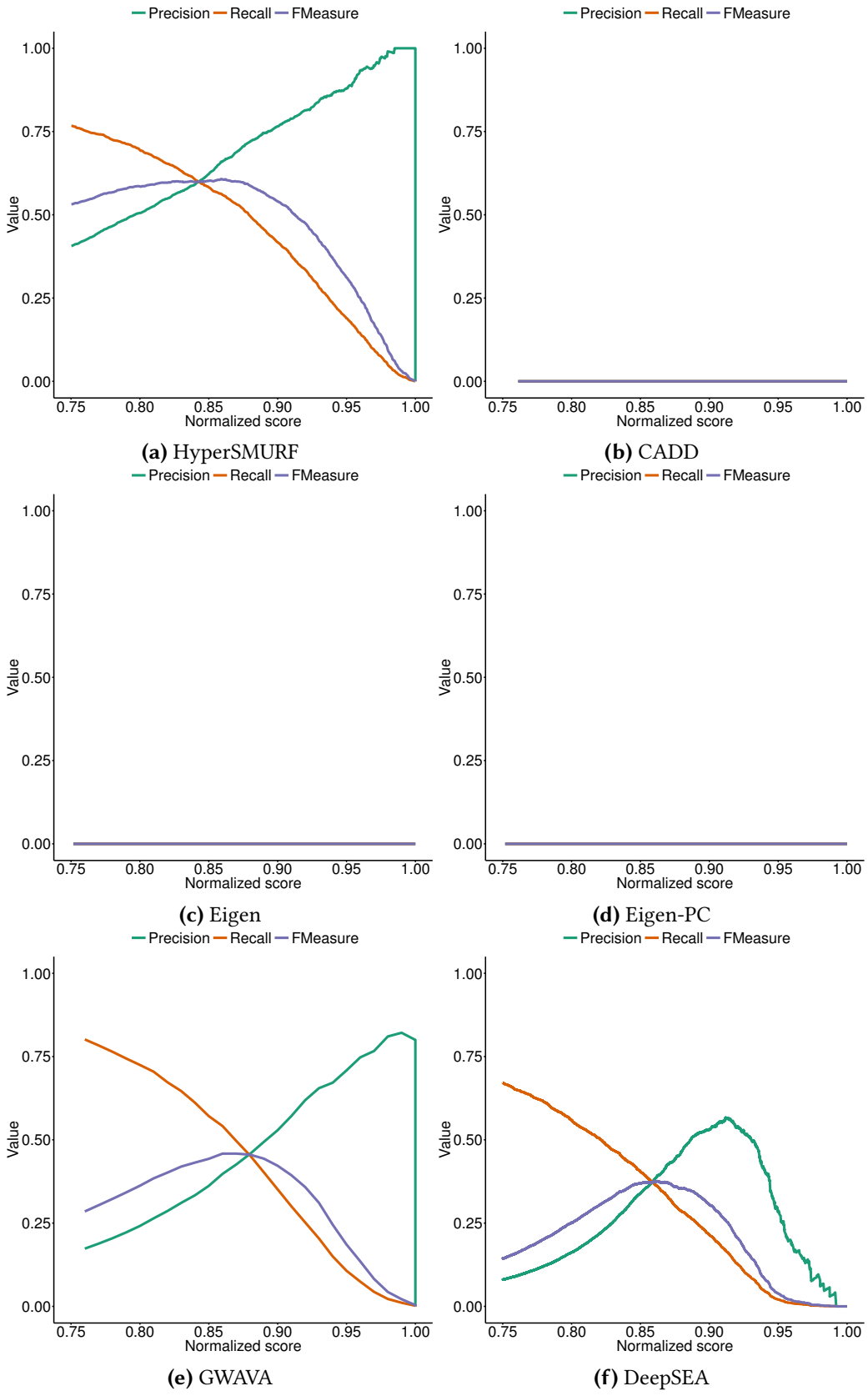


(a) HyperSMURF **(b)** CADD

(c) Eigen **(d)** Eigen-PC

(e) GWAVA **(f)** DeepSEA

Figure B.9: Precision, recall and F-score values of the normalized scores from retrained non-coding score learners on the GWAS data.



(a) HyperSMURF **(b)** CADD

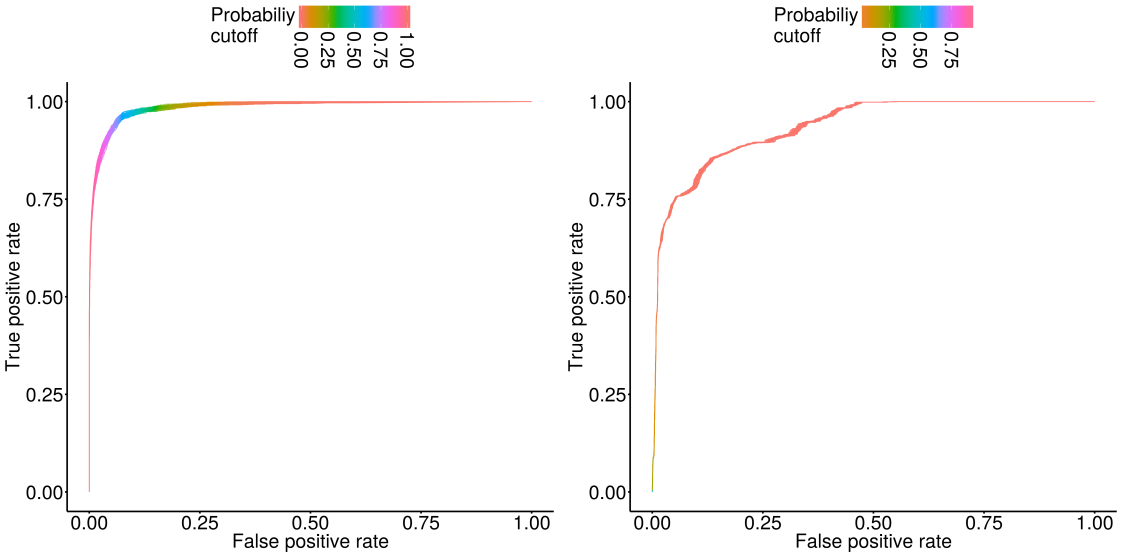
(c) Eigen **(d)** Eigen-PC

(e) GWAVA **(f)** DeepSEA

Figure B.10: Details of precision, recall and F-score within the interval $[0.75, 1]$ of the normalized scores from retrained non-coding score learners on the GWAS data.

eQTL

Finally I show the missing ROC curves of the 50-times-repeated experiment on the eQTL data. HyperSMURF as well as the logistic linear regression showed stable ROC predictions between all 50 repetitions (Figure B.11). Comparing the curves of both methods we clearly see that hyperSMURF shows a better ROC than the original logistic linear regression model.



(a) ROC curves of hyperSMURF.

(b) ROC curves of logistic linear regression.

Figure B.11: Performance measurement of the eQTL data using splits. ROC curves generated by a 50-times-repeated sampling using random splits with three quarters for training and remaining instances for testing. The color of the line shows the probability at the given recall value. Training was done with **(a)** hyperSMURF and **(b)** a logistic linear regression model.

Appendix C

ReMM Score Performance

This appendix contains an in-depth comparison of ReMM to the other non-coding scores CADD, Eigen, Eigen-PC, GWAVA, DeepSEA, FATHMM-MKL and LINSIGHT. In order to obtain a common basis for the comparison, all scores were rescaled in the range $[0, 1]$ through a simple linear transformation (normalized score).

GWAVA displayed the best precision across the normalized scores (Figure C.1a), whereas FATHMM-MKL and DeepSEA had the best recall (Figure C.1b). Nevertheless ReMM is the only method that achieves both a relatively high precision and recall (Figures C.1a, C.1b and C.2a), thus achieving the best F_1-score (Figure C.1c) and best balanced accuracy (Figure C.1d). Although GWAVA displayed the best precision, it showed a marked decrement of the recall as a function of the normalized score (Figure C.1c), and for the highest values of the precision the recall is close to zero (Figures C.2e and C.3e). Correspondingly, GWAVA showed a maximal F_1-score of only about 0.3, as compared to a maximum ReMM F_1-score larger than 0.5 (Figure C.1c).

In contrast to GWAVA, the recall of LINSIGHT is better on higher normalized scores. This results in a reasonable F_1-score with a maximum over 0.4. Even if LINSIGHT is trained on a totally different dataset with a generalized linear regression model that tends to generalize poorly in imbalanced datasets the result is fairly good. There is a subset or maybe a specific functional variant class where LINSIGHT has a good performance.

DeepSEA achieved a high sensitivity but a very low precision which was close to zero for the full range of the normalized score, with a peak close to one when the sensitivity declines close to zero (Figures C.2f and C.3f). Thus, it results in a F_1-score that is very close to zero in the full range of the normalized scores.

CADD performs poorly on this task, mainly due to a low precision, with a recall that is very close to zero for a normalized score larger than 0.5 (Figures C.2b and C.3b).

Eigen achieved the best F_1-score for normalized score close to 0.26 (Figures C.1c and C.2b). This is the result of a peak in precision (about 0.5) close to this value of the normalized score. Unfortunately the recall declines for normalized scores larger than 0.2, thus leading to poor F_1-scores just for score thresholds larger than 0.26. This is due to the fact that several negative variants get an extremely high score in contrast to the regulatory mutations. Therefore a cutoff at 0.26 (normalized score) or 4 (Eigen score) for Mendelian regulatory mutations could represent an appropriate threshold to improve the performance of Eigen. In sum, Eigen has comparable but slightly lower performances in comparison to GWAVA. Eigen-PC has a similar behavior in terms of performance but both precision and recall are lower than Eigen, leading to an even poorer F_1 curve (Figures C.2d and C.3d).

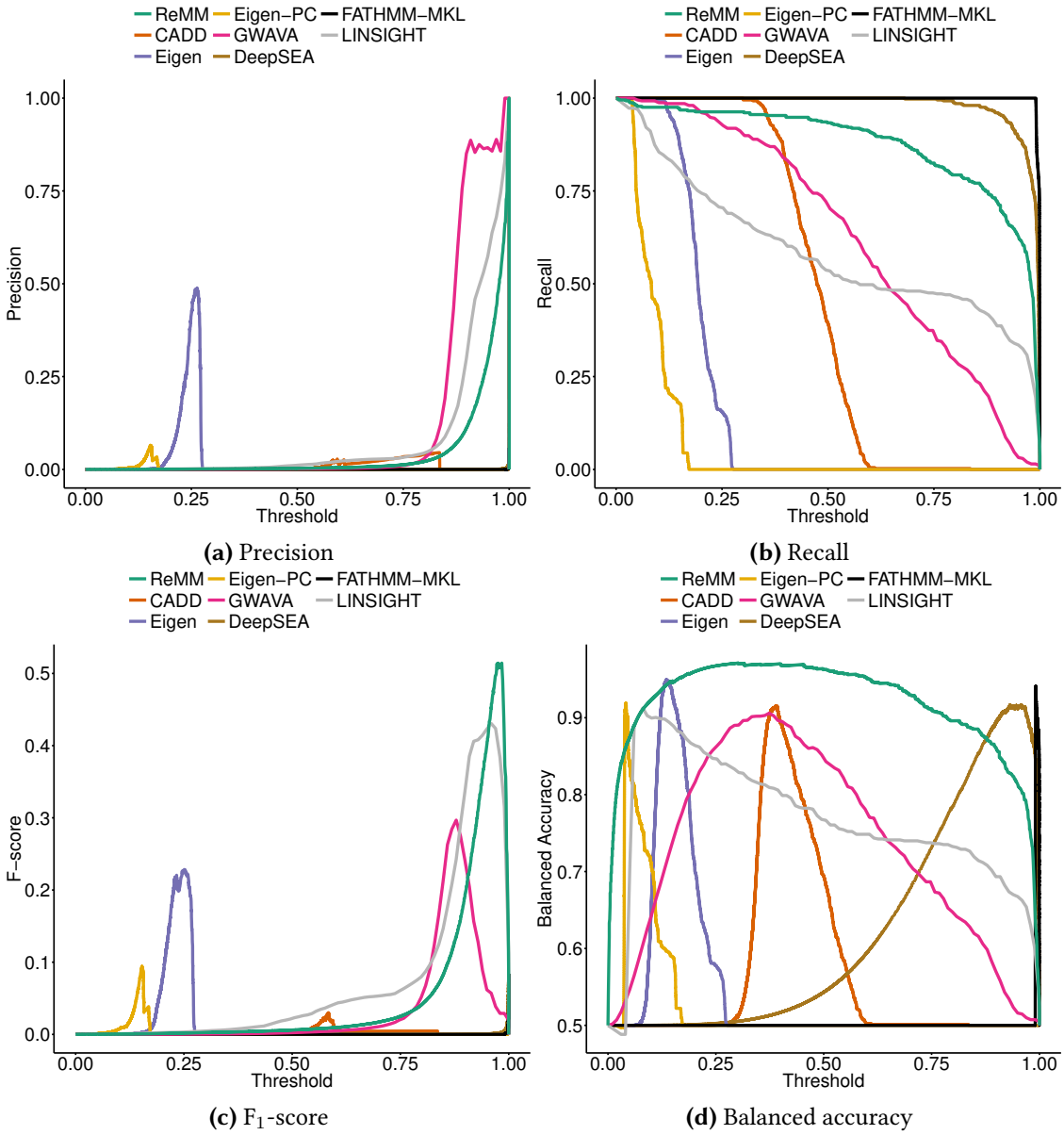
FATHMM-MKL showed a low precision but a high sensitivity with a significant decay only for normalized scores very close one. The resulting F_1-score is very low also for large values due to the poor performance in precision (Figures C.2g and C.3g).

Table C.1: Statistical comparison of ROC curves between ReMM and other NCV scores. One-sided DeLong test [119] was performed between ReMM (AUC $= 0.99$) and CADD, GWAVA, Eigen, Eigen-PC, DeepSEA, FATHMM-MKL, and LINSIGHT. An asterisk ($*$) marks a statistically significant difference (significance level $\alpha = 0.05$).

Score	AUROC	p-value
ReMM score	0.99	–
CADD	0.95	0.78
GWAVA	0.96	0.75
Eigen	0.98	1.27×10^{-39} *
Eigen-PC	0.96	2.01×10^{-58} *
DeepSEA	0.97	0.63
Fathmm-MKL	0.98	0.54
LINSIGHT	0.95	0.79

In summary, this analysis indicates that ReMM substantially outperforms the other methods in predicting non-coding regulatory Mendelian mutations. ReMM is the only method able to obtain both a relatively high precision and recall for the largest values of the normalized score (Figure C.2a). In particular, Figure C.3a indicates that ReMM, for score values higher than 0.97, can achieve an increasing precision from 0.5 to 1 while maintaining a relatively high recall between 0.3 and 0.5. This suggests that a threshold in the range of $[0.95, 1]$ may be most appropriate to search for novel Mendelian mutations in the non-coding genome. Similar considerations about a probability cutoff can be found in the corresponding Section 4.3. Note that for scores very close to one the sensitivity is very low, thus leading to an F_1-score close to zero.

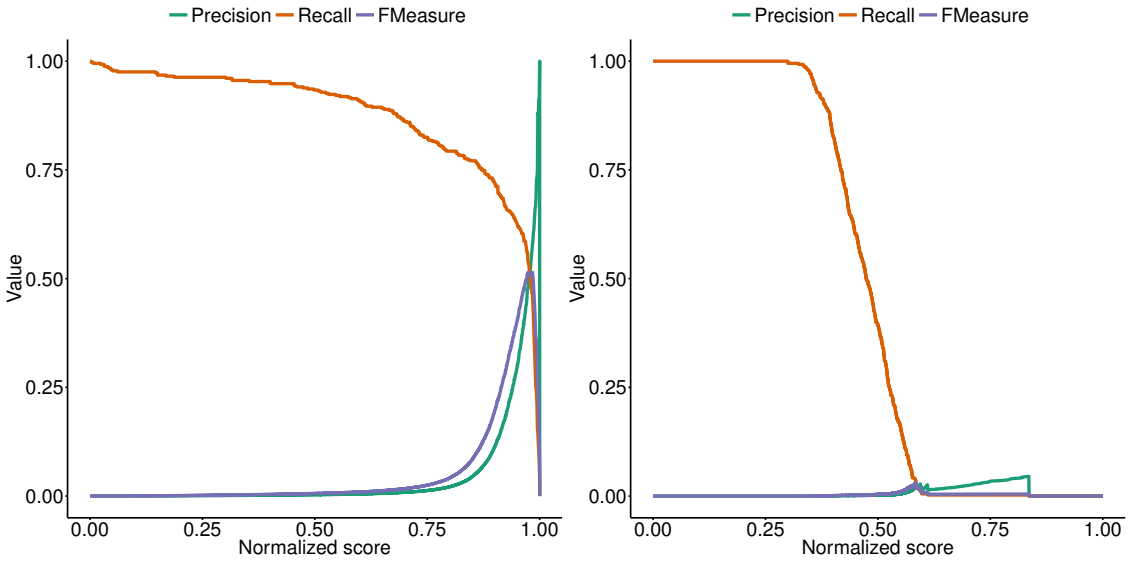
I remark that ReMM was specifically designed to deal with this extremely imbalanced task, with a small but highly reliable set of positive examples (manually curated Mendelian mutations). The five competing methods analyzed here were not specifically designed for Mendelian mutations and moreover, apart from GWAVA, they do not adopt learning strategies specifically devised to deal with extremely imbalanced data. These facts might explain their worse results with respect to ReMM in the prediction of Mendelian mutations.



(a) Precision **(b)** Recall

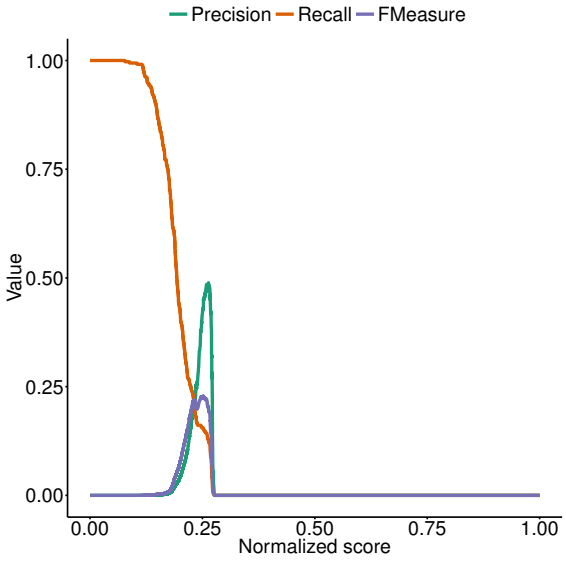
(c) F_1-score **(d)** Balanced accuracy

Figure C.1: Precision, recall, and F-score comparison across different genome-wide pathogenicity scores using the Mendelian data. ReMM, CADD, Eigen, Eigen-PC, GWAVA, DeepSEA, FATHMM-MKL, and LINSIGHT performance by varying the normalized score threshold.

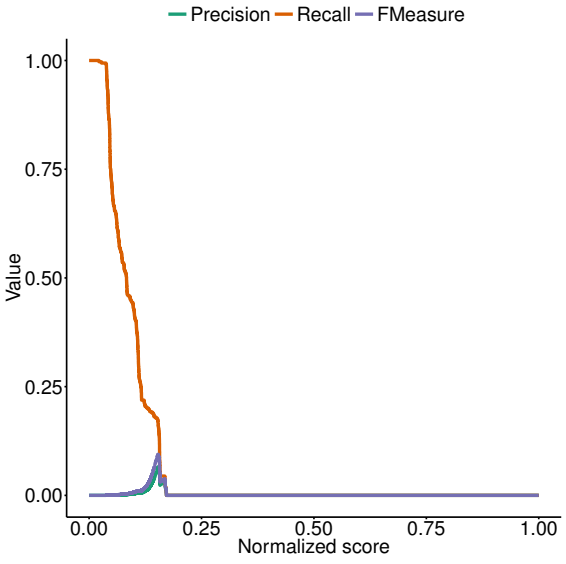


(a) ReMM score

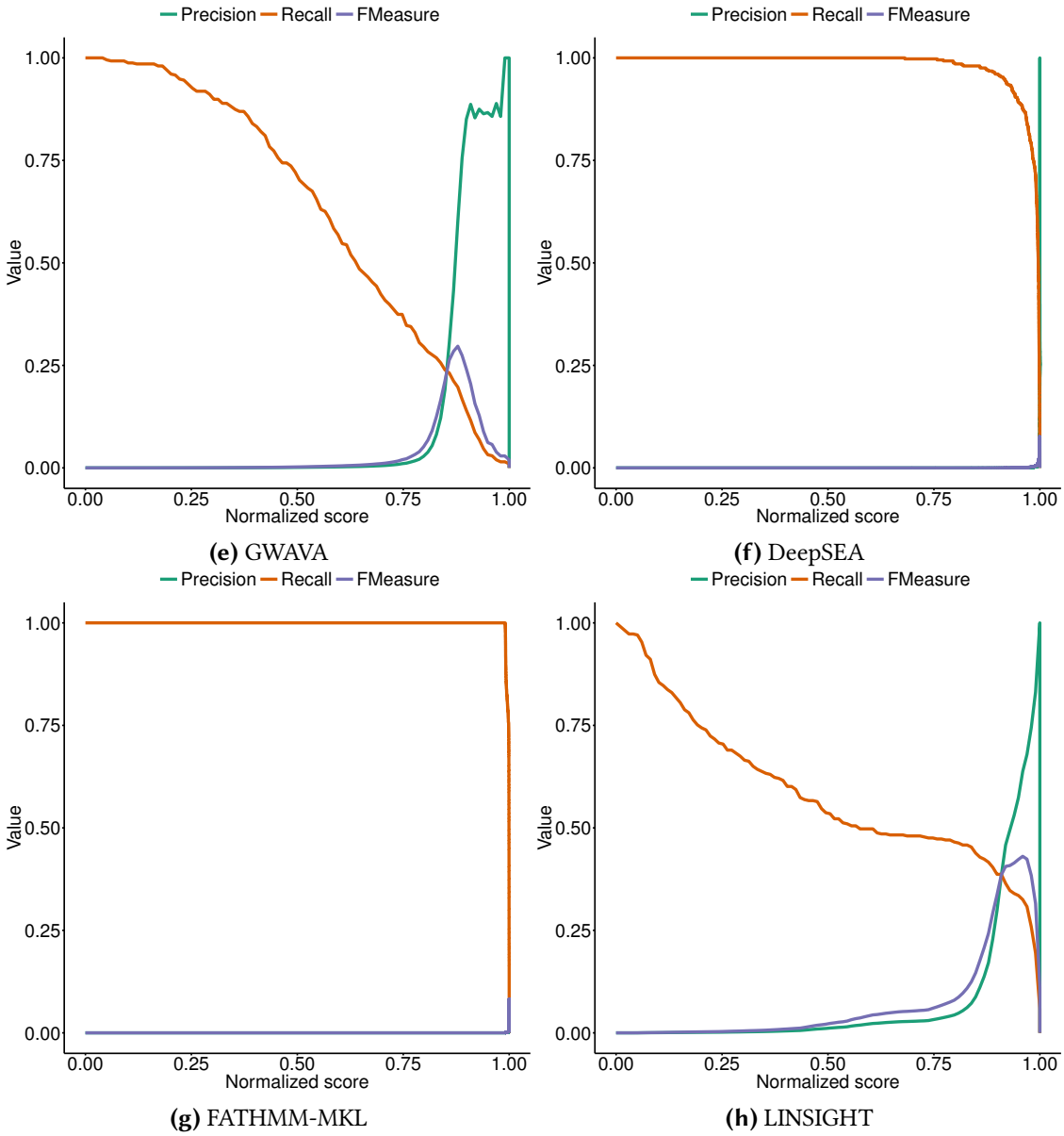
(b) CADD



(c) Eigen



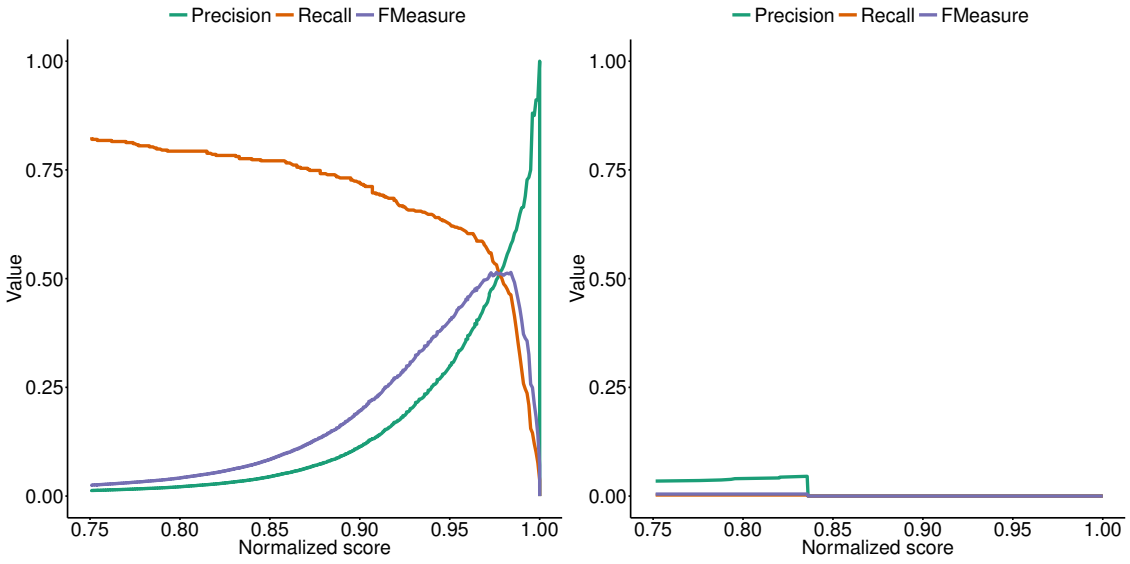
(d) Eigen-PC



(e) GWAVA **(f)** DeepSEA

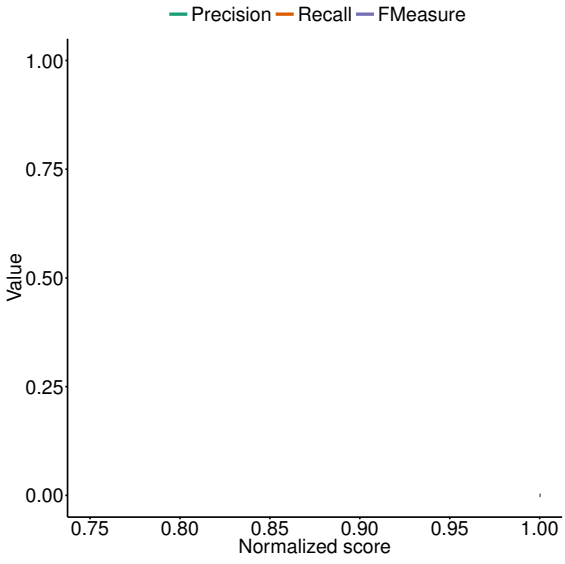
(g) FATHMM-MKL **(h)** LINSIGHT

Figure C.2: Precision, recall and F-score results of the normalized genome-wide pathogenicity scores with the Mendelian data.

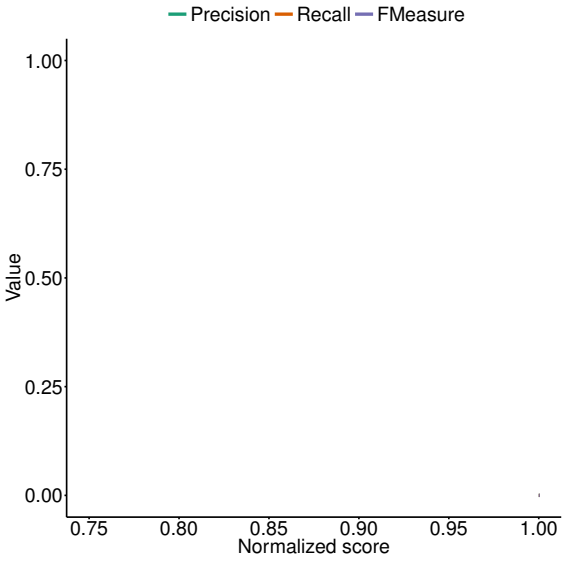


(a) ReMM score

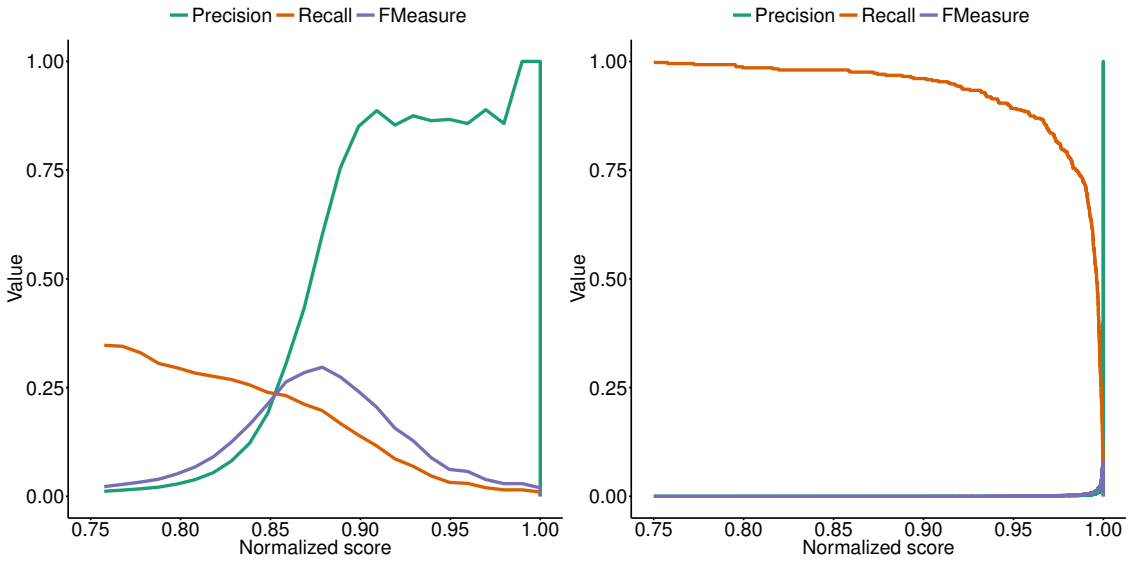
(b) GWAVA



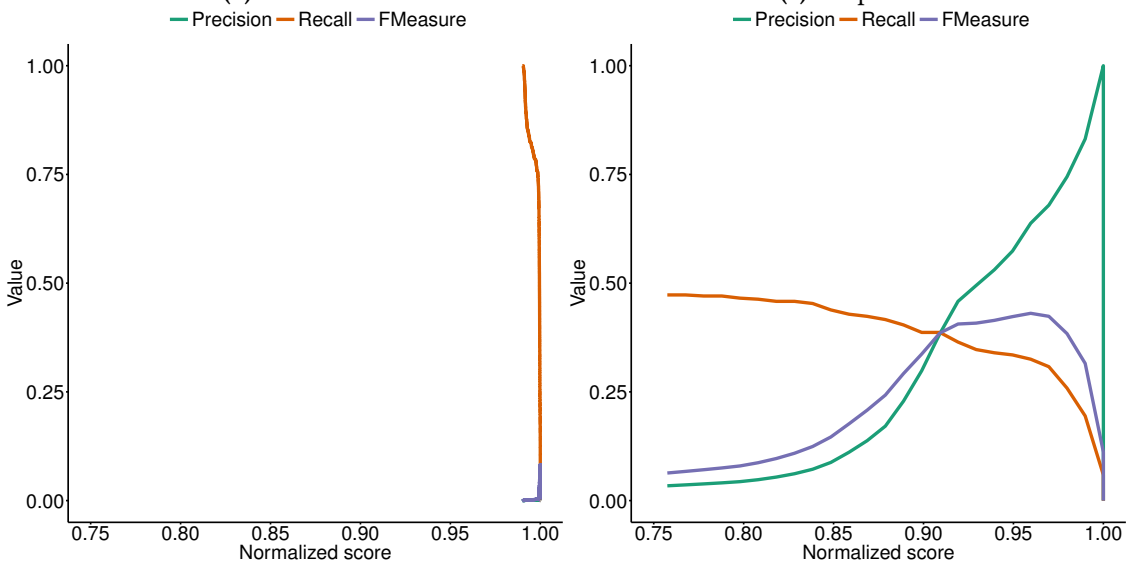
(c) Eigen



(d) Eigen-PC



(e) GWAVA **(f)** DeepSEA



(g) FATHMM-MKL **(h)** LINSIGHT

Figure C.3: Detailed precision, recall and F-score results of the normalized genome-wide pathogenicity scores in the interval $[0.75, 1]$ of the Mendelian data.

Appendix D

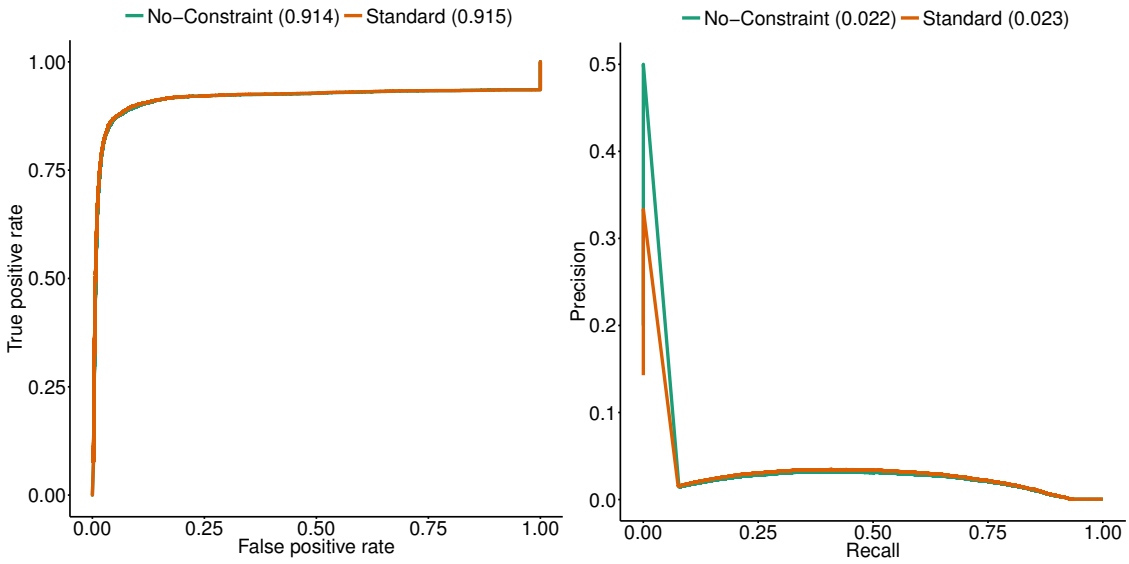
Genomiser Performance

Appendix D contains some additional material about the measurement of the performance of Genomiser and the results. Table D.1 lists all variants that were used for compound heterozygous analysis of Genomiser. Therefore each spike-in was done with a coding and a non-coding mutation identified in the literature. In total 22 cases were analyzed.

Figure D.1 shows the ROC and PR curves of the comparison between the standard Genomiser approach and the modified Genomiser, where all intronic, intergenic, upstream and downstream variants were assigned to the phenotypically closest gene, independent of an overlap with a regulatory region defined by FANTOM5 [135] or the Ensembl regulatory feature build [136].

Table D.1: Compound heterozygous mutations used for Genomiser performance measurement. In total 22 cases were identified in the literature with one coding or splice site mutation and one mutation in a non-coding sequence. All cases were used to test the performance of Genomiser on combinations of coding/non-coding mutations. Note that the 22 non-coding mutations were also included in the main test set of 453 NCVs.

Coding Variant	Non-coding Variant	Gene	PMID
chr22:40760969G>A	chr22:40742514T>C	ADSL	12016589
chr9:104189856C>G	chr9:104198194C>T	ALDOB	20882353
chr1:100680411G>A	chr1:100661453T>G	DBT	20570198
chr2:69583638C>A	chr2:69553299G>T	GFPT1	25765662
chr7:31016054A>G	chr7:31003560A>C	GHRHR	11875102
chr13:20763471C>T	chr13:20767158G>A	GJB2	17660464
chr9:37430601TC>T	chr9:37422744GC>AT	GRHPR	25410531
chr11:5248004G>A	chr11:5248291GA>G	HBB	7803275
chr11:5248225CTT>C	chr11:5248280C>T	HBB	19372376
chr11:5247992CAAAG>C	chr11:5248294G>A	HBB	18473240
chr11:5247992CAAAG>C	chr11:5246720T>G	HBB	11722440
chr11:5248223G>GTA	chr11:5246718A>T	HBB	5481893
chr11:5248004G>A	chr11:5248269G>C	HBB	8562944
chr11:5247992CAAAG>C	chr11:5248374T>A	HBB	17516066
chr11:5248232T>A	chr11:5248372G>A	HBB	21801233
chr10:71119707G>A	chr10:71075518A>G	HK1	19608687
chr2:128185950C>T	chr2:128175983A>G	PROC	10942114
chr11:47469631G>T	chr11:47470715G>C	RAPSN	12651869
chr11:2187270G>T	chr11:2193087G>A	TH	17696123
chr10:127503630A>G	chr10:127505271A>G	UROS	11254675
chr10:127477562C>T	chr10:127505291G>T	UROS	11254675
chr10:127505005C>T	chr10:127505287G>T	UROS	11254675



(a) ROC curve

(b) PR curve

Figure D.1: Performance curves of Genomiser with standard settings and without limiting to only regulatory regions in TAD. 453 NCVs were spiked into genomes of the 1KG and Genomiser was run using noisy phenotype queries. The Exomiser scores were recorded of every gene in the output and the **(a)** ROC and **(b)** the PR curves were plotted here. An Exomiser score of zero was assigned to the target gene of the NCV, if the variant was not present (filtered out) in the Genomiser output. Genomiser was used in the standard version (Standard) and in a modified version (No-Constraint) where the assignment of 20 Kb to the next gene was removed and every intergenic, intronic, upstream, and downstream variant was associated to the most phenotypcially similar gene in the TAD. In 'No-Constraint' variants did not necessarily overlap to a regulatory region defined by FANTOM5 [135] or the Ensembl regulatory feature build [136].

Appendix E

HyperSMURF Tutorial

This appendix chapter is a tutorial on how to use the hyperSMURF Java library for training and testing on imbalanced data. Here we will create a new Maven project, called `hyperSMURF-tutorial`, and use hyperSMURF together with Weka [15] to train some tasks. Therefore we first have to set up a Maven project. Maven will manage and import all necessary libraries. Then we will start with a synthetic example. Afterwards we use real genetic data for training. All files of this tutorial are available under `https://www.github.com/visze/hyperSMURF-tutorial`.

E.1 Requirements

First we have to build a maven project and include the hyperSMURF library into the `pom.xml` file. Therefore we generate a new folder (`hyperSMURF-tutorial`) with a new `pom.xml` file.

```
$ mkdir hyperSMURF-tutorial
$ cd hyperSMURF-tutorial
$ touch pom.xml
```

Then we open the `pom.xml` file in an editor and put in the following lines:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
 ↪  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 ↪  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
 ↪  http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>de.charite.compbio.hypersmurf</groupId>
  <artifactId>hyperSMURF-tutorial</artifactId>
  <packaging>jar</packaging>
  <version>0.3</version>
  <name>hyperSMURF-tutorial</name>
  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>
  <dependencies>
    <dependency>
      <groupId>nz.ac.waikato.cms.weka</groupId>
      <artifactId>weka-dev</artifactId>
      <version>3.9.0</version>
    </dependency>
    <dependency>
      <groupId>de.charite.compbio</groupId>
      <artifactId>hyperSMURF</artifactId>
      <version>0.3</version>
    </dependency>
  </dependencies>
  <build>
    <plugins>
      <plugin>
        <artifactId>maven-assembly-plugin</artifactId>
        <executions>
          <execution>
            <phase>package</phase>
            <goals>
              <goal>single</goal>
            </goals>
          </execution>
        </executions>
        <configuration>
          <descriptorRefs>
            <descriptorRef>jar-with-dependencies</descriptorRef>
          </descriptorRefs>
        </configuration>
      </plugin>
    </plugins>
  </build>
</project>
```

Now we can start writing Java files under the folder `src/main/java`. To build a final runnable jar-file we simply use the command `mvn clean package` to compile the jar file into the `target` folder. Then we can run the `hyperSMURF-tutorial-0.3-jar-with-dependencies.jar` jar in `target` folder by specifying our main class (here `SyntheticExample` from the next section):

```
$ java -cp target/hyperSMURF-tutorial-0.3-jar-with-dependencies.jar
 ↪  de.charite.compbio.hypersmurf.SyntheticExample
```

If you use Eclipse for developing a useful maven command to generate a project that can be imported into eclipse is `mvn eclipse:eclipse`.

E.2 Simple usage Examples with Synthetic Data

In this section we will add a new file `SyntheticExample.java` under the folder `src/main/java/de/charite/compbio/hypersmurf` to our maven project. In the file we will generate the `SyntheticExample` class which contains all functions to generate some imbalanced synthetic data, configure and run hyperSMURF and finally show some performance measurements of the training. The class has a main function to run it and two other functions:
(1) `generateSyntheticData` to generate synthetic imbalanced data and (2) `classify` to classify instances with a classifier using k-fold CV.

The outline of the Java class `SyntheticExample` looks like this:

```
package de.charite.compbio.hypersmurf;

public class SyntheticExample {

  /**
   * We need a seed for the random number generator to make consistent predictions.
   */
  private static int SEED = 42;

  public static void main(String[] args) throws Exception {

  }
}
```

So the class only defines a seed for the random number generator to make predictions consistent. Then we use the RDG1 data generator from Weka to generate synthetic data. For example we will generate 10,000 instances, each with 20 numeric attributes, and set the index to the last attribute which contains class c_0 and c_1 by default. Then we randomize the data using our predefined seed:

```
RDG1 dataGenerator = new RDG1();
dataGenerator.setRelationName("SyntheticData");
dataGenerator.setNumExamples(10000);
dataGenerator.setNumAttributes(20);
dataGenerator.setNumNumeric(20);
dataGenerator.setSeed(SEED);
dataGenerator.defineDataFormat();
Instances instances = dataGenerator.generateExamples();

// set the index to last attribute
instances.setClassIndex(instances.numAttributes() - 1);

// randomize the data
Random random = new Random(SEED);
instances.randomize(random);
```

The problem is that this data is not imbalanced. We can check this by writing a short helper function.

```
private static int[] countClasses(Instances instances) {
int[] counts = new int[instances.numClasses()];
for (Instance instance : instances) {
  if (instance.classIsMissing() == false) {
    counts[(int) instance.classValue()]++;
  }
}
return counts;
}
```

Now if we add `int[] counts = countClasses(instances);` to our instance generation and print it using `System.out.println("Before imbalancing: " + Arrays.toString(counts));` we will see that c_0 has 2599 and c_1 has 7401 instances.

To imbalance the data we will write some code. For example we want to use only 50 instances of c_0. So we have to generate a new `Instances` object and assign all c_1 class instances and only 50 c_0 class instances to it.

```
// imbalance data
int numberOfClassOne = 50;
Instances imbalancedInstances = new Instances(instances, counts[1] + numberOfClassOne);
for (int i = 0; i < instances.numInstances(); i++) {
  if (instances.get(i).classValue() == 0.0) {
    if (numberOfClassOne != 0) {
      imbalancedInstances.add(instances.get(i));
      numberOfClassOne--;
    }
  } else {
    imbalancedInstances.add(instances.get(i));
  }
}
imbalancedInstances.randomize(random);
counts = countClasses(imbalancedInstances);
System.out.println("After imbalancing: " + Arrays.toString(counts));
```

The last line prints out the new imbalance. Now c_0 has only 50 instances.

Now we have to set up our classifier. We will use hyperSMURF with 10 partitions, an oversampling factor of 2 (200 %), no undersampling, and each forest should have a size on 10.

```
// setup the hyperSMURF classifier
HyperSMURF clsHyperSMURF = new HyperSMURF();
clsHyperSMURF.setNumIterations(10);
clsHyperSMURF.setNumTrees(10);
clsHyperSMURF.setDistributionSpread(0);
clsHyperSMURF.setPercentage(200.0);
clsHyperSMURF.setSeed(SEED);
```

The next step will be the performance testing of hyperSMURF on the newly generated imbalanced dataset. Therefore we will use a 5-fold CV. To rerun this performance test using other classifiers we write everything into a new function `static void classify(AbstractClassifier cls, Instances instances, int folds){}`. The `classify` function will collect the predic-

tions over all 5 folds in the `Evaluation` object which then can be used to print out the overall performance results. Here is the complete `classify` function:

```
private static void classify(AbstractClassifier cls, Instances instances, int folds)
 ↪  throws Exception {
  // perform cross-validation and add predictions
  Instances predictedData = null;
  Evaluation eval = new Evaluation(instances);
  for (int n = 0; n < folds; n++) {
    System.out.println("Training fold " + n + " from " + folds + "...");
    Instances train = instances.trainCV(folds, n);
    Instances test = instances.testCV(folds, n);

    // build and evaluate classifier
    Classifier clsCopy = AbstractClassifier.makeCopy(cls);
    clsCopy.buildClassifier(train);
    eval.evaluateModel(clsCopy, test);

    // add predictions
    AddClassification filter = new AddClassification();
    filter.setClassifier(cls);
    filter.setOutputClassification(true);
    filter.setOutputDistribution(true);
    filter.setOutputErrorFlag(true);
    filter.setInputFormat(train);
    Filter.useFilter(train, filter); // trains the classifier
    // perform predictions on test set
    Instances pred = Filter.useFilter(test, filter);
    if (predictedData == null)
      predictedData = new Instances(pred, 0);
    for (int j = 0; j < pred.numInstances(); j++)
      predictedData.add(pred.instance(j));
  }

  // output evaluation
  System.out.println();
  System.out.println("=== Setup ===");
  System.out.println("Classifier: " + cls.getClass().getName() + " " +
 ↪  Utils.joinOptions(cls.getOptions()));
  System.out.println("Dataset: " + instances.relationName());
  System.out.println("Folds: " + folds);
  System.out.println("Seed: " + SEED);
  System.out.println();
  System.out.println(eval.toSummaryString("=== " + folds + "-fold Cross-validation ===",
 ↪  false));
  System.out.println();
  System.out.println(eval.toClassDetailsString("=== Details ==="));

}
```

Finally we can test hyperSMURF by calling the method using five folds: `classify(clsHyperSMURF, imbalancedInstances, 5);`. The output of the performance should be similar to the next text:

```
=== 5-fold Cross-validation ===
Correctly Classified Instances        7406               99.3961 %
Incorrectly Classified Instances        45                0.6039 %
Kappa statistic                          0.3809
Mean absolute error                      0.0858
Root mean squared error                  0.1278
Relative absolute error                637.5943 %
Root relative squared error            156.5741 %
Total Number of Instances             7451


=== Details ===
                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.280    0.001    0.609      0.280    0.384      0.410    0.895     0.337     c0
                 0.999    0.720    0.995      0.999    0.997      0.410    0.895     0.999     c1
Weighted Avg.    0.994    0.715    0.993      0.994    0.993      0.410    0.895     0.995
```

So we will get an AUROC of 0.895 and an AUPRC of 0.337 for our minority class c_0. We can also use a RF classifier using the same number of random trees to see the differences:

```
// setup a RF classifier
RandomForest clsRF = new RandomForest();
clsRF.setNumIterations(10);
clsRF.setSeed(SEED);
// classify RF
classify(clsRF, imbalancedInstances, 5);
```

Now we see that the RF is only able to get an AUROC of 0.706 and an AUPRC of 0.109.

E.3 Usage Examples with Genetic Data

HyperSMURF was designed to predict rare genomic variants when the available examples of such variants are substantially less than background examples. This is a typical situation with genetic variants. For instance, we have only a small set of available variants known to be associated with Mendelian diseases in non-coding regions (positive examples) against the sea of background variants, i.e. a ratio of about $1:36{,}000$ between positive and negative examples [56].

Here we show how to use hyperSMURF to detect these rare features using datasets obtained from the original large set of Mendelian data [56]. To provide usage examples that do not require more than 1 minute of computation time on a modern desktop computer, we considered datasets downsampled from the original Mendelian data. In particular we constructed Mendelian datasets with a progressively larger imbalance between Mendelian associated mutations and background genetic variants. We start with an artificially balanced dataset and then we consider progressively imbalanced datasets with ratio `positive:negative` varying from $1:10$, $1:100$ and $1:1000$. These datasets are downloadable as compressed `.arff` files, easily usable by Weka, from `https://www.github.com/charite/hyperSMURF-tutorial/data`.

The `Mendelian.balanced.arff.gz` file includes 26 features, a column `class` showing the belonging class (1 = positive, 0 = negative), and a column `fold`. This is a numeric attribute with the number of the fold in which each example will be included according to the cytoband-aware 10-fold CV procedure (0 to 9). In total the file contains 406 positives and 400 negatives.

Now we have to write the following code in our new Java file `MendelianExample.java` in folder `src/main/java/de/charite/compbio/hypersmurf`:

1. Loader of the Instances.
2. CV strategy that takes the column `fold` into account when partitioning and removing the column `fold` for training.
3. Setting up our hyperSMURF classifier.

So this will be the blank `MendelianExample.java` class:

```
public class MendelianExample {
  /**
   * We need a seed to make consistent predictions.
   */
   private static int SEED = 42;
   /**
   * The number of folds are predefined in the dataset
   */
   private static int FOLDS = 10;

  public static void main(String[] args) throws Exception {

  }
}
```

To read the data we can simply use the `ArffLoader` from Weka. We will use the first argument of the command-line arguments as our input file.

```
// read the file from the first argument of the command line input
ArffLoader reader = new ArffLoader();
reader.setFile(new File(args[0]));
Instances instances = reader.getDataSet();
```

Then we have to set the class attribute. This is the last attribute of our instances. So we write `instances.setClassIndex(instances.numAttributes() - 1);`. Because we have a balanced dataset of the Mendelian data we do not need to do over- or undersampling. So we simply run hyperSMURF with two partitions and a forest size of ten. Over- and undersampling settings have to be set to 0.

```
// setup the hyperSMURF classifier
HyperSMURF clsHyperSMURF = new HyperSMURF();
clsHyperSMURF.setNumIterations(2);
clsHyperSMURF.setNumTrees(10);
clsHyperSMURF.setDistributionSpread(0);
clsHyperSMURF.setPercentage(0.0);
clsHyperSMURF.setSeed(SEED);
```

Now we arrived at the special cytogenetic band-aware CV. The folds are predefined as attribute `fold` in the instances object. We have to select the instances on that fold but have to remove the fold attribute before training or testing a classifier. So we will write a small helper method that gives us a given fold for testing or the inverse for training. The blank method can be written like this:

```
private static Instances getFold(Instances instances, int fold, boolean invert) throws
 ↪  Exception {

}
```

We will use the filter `SubsetbyExpression` to get the instances with the fold and we can simply use the `Instances` method `deteleAttributeAt(int index)` to remove the fold attribute. For `SubsetbyExpression` filter we write a regular expression like `attribute = n` or `!(attribute = n)` to get the nth fold, or all other folds except fold n. The identifier `attribute` will be written like `ATT` with the index (one based) of the attribute. We can get it using `int indexFold = instances.attribute("fold").index();` (zero based) and we have to increment it by one for our filter method. So the content of our `getFold` method can look like:

```
// filter on fold variable
int indexFold = instances.attribute("fold").index();
SubsetByExpression filterFold = new SubsetByExpression();
if (invert)
  filterFold.setExpression("!(ATT" + (indexFold + 1) + " = " + fold + ")");
else
  filterFold.setExpression("ATT" + (indexFold + 1) + " = " + fold);
filterFold.setInputFormat(instances);
Instances filtered = Filter.useFilter(instances, filterFold);

// remove fold attribute
filtered.deleteAttributeAt(indexFold);

return filtered;
```

Now it is time for the CV. This is similar to the Synthetic Example in Appendix E.2, but we will use the `getFold` method to make the train/test partitioning.

```
// perform cross-validation and add predictions
Instances predictedData = null;
Evaluation eval = new Evaluation(instances);
for (int n = 0; n < FOLDS; n++) {
  System.out.println("Training fold " + (n+1) + " from " + FOLDS + "...");
  Instances train = getFold(instances, n, true);
  Instances test = getFold(instances, n, false);

  // build and evaluate classifier
  Classifier clsCopy = AbstractClassifier.makeCopy(cls);
  clsCopy.buildClassifier(train);
  eval.evaluateModel(clsCopy, test);

  // add predictions
  AddClassification filter = new AddClassification();
  filter.setClassifier(cls);
  filter.setOutputClassification(true);
  filter.setOutputDistribution(true);
  filter.setOutputErrorFlag(true);
  filter.setInputFormat(train);
  Filter.useFilter(train, filter); // trains the classifier
  // perform predictions on test set
  Instances pred = Filter.useFilter(test, filter);
  if (predictedData == null)
    predictedData = new Instances(pred, 0);
  for (int j = 0; j < pred.numInstances(); j++)
    predictedData.add(pred.instance(j));
}

// output evaluation
System.out.println();
System.out.println("=== Setup ===");
System.out.println("Classifier: " + cls.getClass().getName() + " " +
 ↪  Utils.joinOptions(cls.getOptions()));
System.out.println("Dataset: " + instances.relationName());
System.out.println("Folds: " + FOLDS);
System.out.println("Seed: " + SEED);
System.out.println();
System.out.println(eval.toSummaryString("=== " + FOLDS + "-fold Cross-validation ===",
 ↪  false));
System.out.println();
System.out.println(eval.toClassDetailsString("=== Details ==="));
```

If we run hyperSMURF with the settings above the command-line output will show an AUPRC of 0.989 and an AUROC of 0.989 of our class 1 which are the Mendelian regulatory mutations. This is the complete output:

```
=== 10-fold Cross-validation ===
Correctly Classified Instances         770               95.5335 %
Incorrectly Classified Instances        36                4.4665 %
Kappa statistic                          0.9107
Mean absolute error                      0.0898
Root mean squared error                  0.1925
Relative absolute error                 17.9538 %
Root relative squared error             38.4915 %
Total Number of Instances              806


=== Details ===
                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.985    0.074    0.929      0.985    0.956      0.912    0.989     0.983     0
                 0.926    0.015    0.984      0.926    0.954      0.912    0.989     0.989     1
Weighted Avg.    0.955    0.044    0.957      0.955    0.955      0.912    0.989     0.986
```

Then we can perform the same computation using the progressively imbalanced datasets: `Mendelian.1_10.arff.gz`, `Mendelian.1_100.arff.gz`, and `Mendelian.1_1000.arff.gz`. Of course every time we have to adapt the settings of hyperSMURF.

Using `Mendelian.1_10.arff.gz` together with hyperSMURF the output can look like this:

```
// setup the hyperSMURF classifier
clsHyperSMURF = new HyperSMURF();
clsHyperSMURF.setNumIterations(5);
clsHyperSMURF.setNumTrees(10);
clsHyperSMURF.setDistributionSpread(0);
clsHyperSMURF.setPercentage(100.0);
clsHyperSMURF.setSeed(SEED);
```

```
=== 10-fold Cross-validation ===
Correctly Classified Instances        4310               97.8212 %
Incorrectly Classified Instances        96                2.1788 %
Kappa statistic                          0.8779
Mean absolute error                      0.0577
Root mean squared error                  0.1427
Relative absolute error                 34.4437 %
Root relative squared error             49.3333 %
Total Number of Instances              4406


=== Details ===
                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.981    0.044    0.995      0.981    0.988      0.880    0.990     0.999     0
                 0.956    0.020    0.833      0.956    0.890      0.880    0.990     0.950     1
Weighted Avg.    0.978    0.042    0.980      0.978    0.979      0.880    0.990     0.994
```

Increasing the imbalance with `Mendelian.1_100.arff.gz`:

```
// setup the hyperSMURF classifier
clsHyperSMURF = new HyperSMURF();
clsHyperSMURF.setNumIterations(5);
clsHyperSMURF.setNumTrees(10);
clsHyperSMURF.setDistributionSpread(0);
clsHyperSMURF.setPercentage(100.0);
clsHyperSMURF.setSeed(SEED);
```

```
=== 10-fold Cross-validation ===
Correctly Classified Instances       39987               99.1348 %
Incorrectly Classified Instances       349                0.8652 %
Kappa statistic                          0.6795
Mean absolute error                      0.0249
Root mean squared error                  0.0851
Relative absolute error                124.7001 %
Root relative squared error             85.3023 %
Total Number of Instances            40336


=== Details ===
                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.992    0.071    0.999      0.992    0.996      0.705    0.991     1.000     0
                 0.929    0.008    0.541      0.929    0.684      0.705    0.991     0.900     1
Weighted Avg.    0.991    0.071    0.995      0.991    0.992      0.705    0.991     0.999
```

Again increasing the imbalance with `Mendelian.1_1000.arff.gz`:

```
// setup the hyperSMURF classifier
clsHyperSMURF = new HyperSMURF();
clsHyperSMURF.setNumIterations(10);
clsHyperSMURF.setNumTrees(10);
clsHyperSMURF.setDistributionSpread(3);
clsHyperSMURF.setPercentage(200.0);
clsHyperSMURF.setSeed(SEED);
```

```
=== 10-fold Cross-validation ===
Correctly Classified Instances      392436               99.2597 %
Incorrectly Classified Instances      2927                0.7403 %
Kappa statistic                          0.2021
Mean absolute error                      0.0233
Root mean squared error                  0.0805
Relative absolute error               1135.2254 %
Root relative squared error            251.4735 %
Total Number of Instances           395363


=== Details ===
                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.993    0.079    1.000      0.993    0.996      0.323    0.989     1.000     0
                 0.921    0.007    0.114      0.921    0.204      0.323    0.989     0.773     1
Weighted Avg.    0.993    0.079    0.999      0.993    0.995      0.323    0.989     1.000
```

As we can see, we have a certain decrement of performance when the imbalance increases. Indeed when we have perfectly balanced data the AUPRC is very close to 1, while by increasing the imbalance we have a progressive decrement of the AUPRC to 0.950, 0.900, down to 0.773 when we have a $1:1000$ imbalance ratio. Nevertheless this decline in performance is relatively small compared to other ML methods.

We can perform the same task using parallel computation. For instance, by using 4 cores with an Intel i7-2670QM CPU, 2.20GHz, we perform a full cytoband-aware 10-fold CV using 406 genetic variants known to be associated with Mendelian diseases and 400,000 background variants in less than 5 minutes. We get the best performance boost from this implementation if we do the training of the partitioning in parallel. So we can set the number of execution slots to 4 using `clsHyperSMURF.setNumExecutionSlots(4);`.

Of course the training and CV functions allow also to set the parameters of the RF ensembles, that constitute the base learners of the hyperSMURF hyper-ensemble, such as the number of decision trees to be used for each RF (`setNumTrees(int num)`) or the number of features to be randomly selected from the set of available input features at each step of the inductive learning of the decision tree (`setNumFeatures(int num)`). The full description of the `HyperSMURF` class can be found in the hyperSMURF Java API `https://javadoc.io/doc/de.charite.compbio/hyperSMURF`.

Abbreviations

1KG 1000 Genomes Project.

A adenine.
AA amino acid.
AD autosomal dominant.
ADASYN adaptive synthetic sampling approach.
AF allele frequency.
AGD Arbeitsgemeinschaft für Gen-Diagnostik e.V..
AI artificial intelligence.
ANN artificial neural network.
API application programming interface.
AR autosomal recessive.
AUC area under the curve.
AUPRC area under the precision recall curve.
AUROC area under the receiver operating characteristic curve.

BMA best-matched average.
bp base pair.

C cytosine.
CA common ancestor.
CADD Combined Annotation–Dependent Depletion.
CART classification and regression tree.
ChIP-seq Chromatin immunoprecipitation sequencing.
CNN convolutional neural network.
CNV copy-number variation.
COSMIC Catalogue of Somatic Mutations in Cancer.
CRAN the Comprehensive R Archive Network.
CV cross-validation.
cytoband-aware 10-**fold CV** cytogenetic band-aware 10-fold cross-validation.

DAG directed acyclic graph.
DECIPHER DatabasE of Chromosomal Imbalance and Phenotype in Humans using Ensembl Resources.
DeepSEA deep learning-based sequence analyzer.
DGV Database of Genomic Variants.
DNA deoxyribonucleic acid.
DNase-seq DNase I hypersensitive sites sequencing.

ENCODE Encyclopedia of DNA Elements.
EPO Enredo-Pecan-Ortheus.
eQTL expression quantitative trait locus.
ESE exonic splicing enhancer.
ESP Exome Server Project.

FASTA FAST-All.
FP false positives.
FP rate false positive rate.

G guanine.
GCB German Conference on Bioinformatics.
GNU GPLv3 GNU General Public License Version 3.

GO Gene Ontology.
GPU Graphics Processing Unit.
GWAS Genome Wide Association Studies.
GWAVA genome-wide annotation of variants.

HGMD Human Gene Mutation Database.
hiPHIVE human/interactome-PHIVE.
HPO Human Phenotype Ontology.
hyperSMURF Hyper SMOTE Undersampling with Random Forests.

IC information content.
ICR imprinting control region.
InDel small insertion or deletion.
INSIGHT Inference of Natural Selection from Interspersed Genomically coHerent elemenTs.

Kb kilo base.
k-fold CV k-fold cross-validation.

LCA lowest common ancestor.
LINSIGHT linear INSIGHT.
LOOCV leave-one-out cross-validation.

MAF minor allele frequency.
MAVE multiplex assay of variant effect.
MHC major histocompatibility complex.
miRNA microRNA.
miRNA-relation CV microRNA-relation cross-validation.
MKL multiple kernel learning.
ML machine learning.
MPRA massive parallel reporter assay.
mRNA messenger ribonucleic acid.

NCBI National Center for Biotechnology Information.
NCV non-coding variant.
NGS next-generation sequencing.

OMIM Online Mendelian Inheritance in Man.
ORF open reading frame.

PCA principal-component analysis.
PMID PubMed identifier.
PPV positive predictive value.
PR precision-recall.

RD regulatory domain.
RefSeq National Center for Biotechnology Information Reference Sequence Database.
ReMM score Regulatory Mendelian Mutation score.
REST representational state transfer.
RF Random Forest.
RNA ribonucleic acid.
ROC receiver operating characteristic.

SIFT Sorting Intolerant From Tolerant.
SMOTE synthetic minority oversampling technique.
SNP single nucleotide polymorphism.
SNV single nucleotide variant.
SO Sequence Ontology.
SV structural variation.
SVM Support Vector Machine.

T thymine.
TAD topologically-associated domain.
TFBS transcription factor binding site.
TN true negatives.
topologically-aware CV topologically-aware cross-validation.
TP true positives.
TP rate true positive rate.
TSS transcription start site.

U uracil.
UCSC University of California, Santa Cruz.
uORF upstream open reading frame.
UTR untranslated region.

VCF Variant Call Format.
VEP Variant Effect Predictor.

WGS whole-genome sequencing.

XD X-linked dominant.
XR X-linked recessive.

Glossary

A

Allele Frequency (AF) Relative frequency of an allele at a particular locus in a population.
Allosome Sex related chromosome.
AlphaGo An AI computer program that plays the board game Go.
Artificial Intelligence (AI) A branch of computer science dealing with the simulation of intelligent behavior in computers.
Artificial Neural Network (ANN) A ML system which is inspired by a network of neurons in the brain. Raw information is processed through the different neurons of the network. Thereby different artificial neurons (edges) get activated or deactivated and information is combined at synapses (vertices) until a final interpretable output is generated.
  Deep Learning An ANN with at least three hidden layers.
  Hidden Layer Layers of vertices in between the input and the output layer of an ANN.
Autosome A chromosome other than a sex related chromosome (allosome).

B

Bias-Variance Decomposition The total expected classification error of a learning method made up of the sum of bias and variance.
  Bias Error from erroneous assumptions in the learning algorithm.
  Variance An error from sensitivity to small fluctuations or incompleteness in the training set.
Big Data An accumulation of data that is too large and complex for processing by traditional data processing software. Big data can be characterized by the five Vs [153]: (1) Volume – the quantity of generated and stored data. (2) Variety – the type of the data. (3) Velocity – the speed of data being generated and processed. (4) Variability – inconsistency of the data set. (5) Veracity – the data quality of the data.

C

Chromatin Complex of macromolecules, consisting of DNA, proteins, and RNAs. Its function is to package DNA into a compact, dense shape, to reinforce the DNA macromolecule

to allow mitosis, to prevent DNA damage, and to control gene expression and DNA replication.

Chromosomal Aberration Missing, extra, or irregular portion of chromosomal DNA compared to a reference sequence. Complete chromosomal aberrations are the loss or the gain of a complete chromosome.

Classifier Also known as learner. An algorithm that implements classification, especially in a concrete implementation.

Codon Nucleotide triplets that encode an AA.

Open Reading Frame (ORF) A continuous stretch of codons that contain a start-codon and a stop-codon.

Upstream Open Reading Frame (uORF) A specific open reading frame (ORF) within the 5'UTR of the mRNA.

Start-Codon First codon of a mRNA transcript (usually AUG). It always codes for AA methionine in eukaryotes.

Stop-Codon Terminates the translation of a mRNA transcript (usually UAA, UAG or UGA).

Contig A set of overlapping DNA segments that represents a consensus region of DNA.

Cross-Validation (CV) A model validation technique for assessing how the results of a statistical analysis will generalize to an independent data set. The validation data is split into training and test sets.

k-Fold Cross-Validation (k-fold CV) Partitioning of the validation data into k $in\mathbb{N}$ test sets. For the ith test set, $i = \{1, 2, \ldots, k\}$, all other test sets will be used for training ($\{1, 2, \ldots, k\} \setminus i$).

Cytogenetic band-aware 10-fold Cross-Validation (cytoband-aware 10-fold CV) The human genome will be partitioned into its cytogenetic bands. All variants within a band will be assigned to the same fold. For Mendelian and GWAS data bands with positive variants will be added to the folds such that every fold has a similar amount of positives. All remaining bands will be added randomly to the 10 folds.

MicroRNA-Relation Cross-Validation (miRNA-relation CV) Folds are the number of available miRNAs. Every variant will be in the fold of their corresponding miRNA Strategy is used with the eQTL data.

Topologically-Aware Cross-Validation (topologically-aware CV) The human genome will be partitioned into TADs. All variants within a TAD will be assigned to one fold. The number of folds are the number of available TADs that contain at least one variant.

D

Data Mining The process of discovering patterns in large datasets.

Divide-And-Conquer Recursively breaking down a problem into sub-problems of the same or related type, until these problems become simple enough to be solved directly. Then the sub-problem solutions are combined together to a general solution of the original problem.

E

Ensemble Learning An ML method that combines multiple ML algorithms to raise the predictive performance.

Bagging An ensemble learning method with multiple classifiers which all have the same weight for a final prediction, mostly done by majority or average vote. Every classifier is trained from a bootstrap sample taken out of the initial training instances.

Out-Of-The-Bag Error Method to measure an unbiased performance during the training phase of bagging. Because every classifier is trained from a bootstrap of the initial training set, every instance can be used as a test set in all classifiers not using this instance for training.

Boosting An ensemble learning method that adapts weak learners to generate a strong learner. The boosting approach learns the weight of the weak learners to combine them optimally.

Randomization Method to increase the variance by using random input, different seeds or random subspaces (features).

Stacking Training a learning algorithm with all features as well as the predictions of several other learning algorithms as additional input.

Exome All protein coding regions (exons) in the genome.

Exomiser An application/framework that filters and prioritizes variants together with genes in NGS projects for novel disease-gene discovery or differential diagnostics of Mendelian diseases.

Genomiser Extension of Exomiser to filter and rank NCVs in addition to coding variants.

F

Feature Selection Process of selecting a subset of relevant features for model construction. In general redundant or irrelevant features are removed.

Functional Class The functional context of a sequence variation, defined by inspecting the flanking sequence for gene features.

Frameshift InDel within the coding part of an exon. The variant destroys the reading frame of the exon, resulting in a complete different sequence of AAs after the mutation compared to a reference.

In-Frame InDel within the coding part of an exon. The variant inserts or deletes triplets, but the reading frame of the exon will stay intact. The translated AA sequence after the variant will be the same compared to a reference.

Non-Synonymous Combined functional class of missense and nonsense.

Missense Variant within the coding part of an exon, resulting in a new AA compared to a reference.

Nonsense Variant creating a new stop- or start-codon or destroys it.

Synonymous Variant within the coding part of an exon not changing the AA at the position compared to a reference due to the degenerative code.

G

Go An abstract strategy board game for two players, invented in ancient china around 2500 years ago. The aim of the game is to surround more territory with stones than the opponent.

Graph A structured set of objects (vertices V), where pairs of objects are in a related sense (edges E). A graph is denoted as $G = (V, E)$ An edge $z \in E$ is a subset of two vertices, e.g. $z = (x, y), x, y \in V$, which the edge connects.

Directed Graph A graph $G = (V, E)$ in which edges have orientations. An arrow (x, y) is considered to be directed from x to y, $x, y \in V$.

Undirected Graph A graph $G = (V, E)$ in which edges have no orientation. The edge (x, y) is identical to the edge (y, x), $x, y \in V$.

H

Hyper SMOTE Undersampling with Random Forests (hyperSMURF) ML method specifically conceived to handle extremely imbalanced data. To deal with such problems and to achieve high coverage of the available input data as well as a high accuracy of the predictions hyperSMURF is based on three complementary strategies: (1) sampling techniques; (2) ensemble methods; (3) hyper-ensemble approach.

I

Instance An example/observation of a classification problem. Normally described by its feature vector.

Attribute An individual measurable property or characteristic of an instance.

Nominal Attribute Classification of entities into particular categories. Also known as categorical attribute.

Feature An individual measurable property or characteristic of an instance.

J

Jannovar Java framework for transcript-based annotation and pedigree analysis.

K

k-fold CV k-fold cross-validation.

Leave-One-Out Cross-Validation (LOOCV) Special type of k-fold CV. Uses every single instance in the validation set as test set. So k is the number of all instances.

M

Machine Learning (ML) Application of AI that automates analytical model building by using algorithms that automatically and iteratively learn from data.

Matrix A matrix $\mathbf{M}$ is a rectangular array of elements, mostly numbers, with m rows and n columns, $m, n \in \mathbb{N}$.

Column Vector An $m \times 1$ vector $\mathbf{v}$ of a $m \times n$ matrix $\mathbf{M}$, denoted by a bold lower case letter.

Covariance Matrix A matrix that stores the covariance between the ith and jth elements of a random vector at its element $a_{i,j}$. A covariance matrix is denoted as Σ.

Diagonal Matrix A matrix in which the entries outside the main diagonal are all zero.

Eigendecomposition Factorization of a diagonal matrix into a canonical form, whereby the matrix is represented in terms of its eigenvalues and eigenvectors.

Eigenvalue A number λ and together with the eigenvector $\mathbf{v}$ satisfying $\mathbf{Mv} = \lambda\mathbf{v}$.

Eigenvector A non-zero vector $\mathbf{v}$ together with a number λ satisfying $\mathbf{Mv} = \lambda\mathbf{v}$.

Row Vector An $1 \times n$ vector $\mathbf{v}$ of a $m \times n$ matrix $\mathbf{M}$, denoted by a bold lower case letter.

Square Matrix A $n \times n$ matrix $\mathbf{M}$ with same number of rows and columns.

Identity Matrix A square matrix where $a_{i,j} = 1$ for all $i = j$ and $a_{i,j} = 0$ for all $i \neq j$, denoted as $\mathbf{I}$.

Inverse Matrix The inverse of a square matrix $\mathbf{M}$ such that $\mathbf{M}\mathbf{M}^{-1} = \mathbf{I}$, denoted by $\mathbf{M}^{-1}$.

Mutation Causative (mostly rare or unique) variant resulting in a genetic disease.

O

Ontology A standardized way for knowledge representation.

Axiom Defined statements to specify the representation, for example relationships of terms.

Domain Ontology An ontology that provide a controlled vocabulary for objects in a domain.

Attribute Ontology An attribute ontology that links terms to other objects.

Most informative common Ancestor The most specific common ancestor (CA) in an ontology.

Term Relevant concepts of an ontology which are named and defined entities of the domain of interest. Also sometimes called object.

Top-Level Ontology Ontology that consists of very general terms that are common across all domains.

P

Polymorphism Frequent variants, where the allele, which is different to the reference, occurs in more than 1 % of a population.

Single Nucleotide Polymorphism (SNP) Variant polymorphisms, where the variant has only a single nucleotide change.

R

Random Tree A decision tree using a random, individual sampled subset Θ_i of all available attributes Θ.

Reference Sequence A consensus sequence (DNA sequence), obtained by a collection of sequenced individuals.

Regulatory Code All regulatory elements/domains/regions in the DNA. Literature estimates its proportion to around 5 % to 15 % of the whole genome.

Enhancer An enhancer is a region in the DNA that can recruit transcription factors (activators). When it comes in close proximity (looping) to a promoter it activates or increases the transcription of a gene.

Promoter A promoter is a region in the DNA, located near the transcription start site of a gene. It contains TFBSs that recruit the RNA polymerase to initiate the transcription of the gene.

Silencer A silencer is a region in the DNA that can recruit transcription factors (repressors). When it comes in close proximity (looping) to a promoter it prevents or decreases the transcription of a gene.

Regulatory Mendelian Mutation Score (ReMM score) Score per position in the human genome to assess the pathogenicity in terms of a Mendelian disease from a mutation at the position. Because of its creation it is restricted to the non-coding regulatory genome. The score ranges from zero (likely benign) to one (pathogenic).

S

Seed A number (or vector) used to initialize a pseudorandom number generator.

Sequence Assembly Aligning and merging short DNA fragments in order to reconstruct the original DNA sequence.

- **De-Novo Assembly** Assembling DNA fragments to create the full (novel) original sequence.
- **Mapping Assembly** Assembling reads against an existing backbone sequence (reference sequence), building a sequence that is similar but not necessarily identical to the reference sequence.

Sequencing Determine the primary structure of an unbranched biopolymer. In terms of DNA the result is a linear ordering of the nucleotides.

- **Chromatin immunoprecipitation Sequencing (ChIP-seq)** Identifies genomic locations that are either bound by specific transcription factor or harbor epigenetic modifications to histones or DNA. The method uses a specific antibody against the protein or epigenetic modification of interest. Afterwards the bound DNA is investigated by NGS.
- **Classical Sequencing** First sequencing method.
- **DNase I hypersensitive Sites Sequencing (DNase-seq)** Chromatin is treated with DNase I, and DNA fragments that arise from regions sensitive to DNase attack are enriched, amplified and sequenced.
- **Next-Generation Sequencing (NGS)** Is a generic term for sequencing technologies developed after Sanger sequencing. These methods have a higher throughput than Sanger sequencing, enabling sequencing of complete genomes within reasonable costs. The are subdivided into second-generation (e.g. Illumnina, Roche/454) with high throughput but short reads and third-generation with a lower throughput but reads of several Kb in length.
- **Sanger Sequencing** First sequencing method developed by Sanger *et al.* [4] and Sanger & Coulson [5] in 1977. It is based on the selective incorporation of chain-terminating dideoxynucleotides by DNA polymerase.

Set A set S is a well-defined collection of distinct objects or elements. In this work a set is denoted by curly brackets. $\mathbb{N}$ denoting the set of all natural numbers $\mathbb{N} = \{1, 2, 3, \ldots\}$, $\mathbb{N}_0$ are all non-negative numbers $\mathbb{N}_0 = \{0, 1, 2, 3, \ldots\}$, $\mathbb{Z}$ is the set of all integers $\mathbb{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$ (positive, negative, and zero) and $\mathbb{R}$ is the set of all real numbers (rational and irrational).

Single Nucleotide Variant (SNV) Variant (compared to a reference sequence) affecting only a single nucleotide change.

Small Insertion or Deletion (InDel) Variant class of short deletions or insertions compared to a reference sequence. Can be detected by a single read from a NGS technology of the second-generation. Sizes of InDels are defined as equal or smaller than 25 bp.

Splicing Editing of the nascent pre-mRNA transcript into a mature-mRNA. During splicing introns are removed and exons are ligated together.

Exon Any part of a gene that will encode a part of the final mature-mRNA.

Intron Any part of a gene that will be removed during splicing.

Structural Variation (SV) Large structural differences compared to a reference sequence. Contains larger insertions, deletions or translocations including inversions.

T

Testing Performance evaluation of trained ML classifier.

Topologically-Associated Domain (TAD) A genomic region of increased contact probability compared to sequences outside of it. This chromatin folding at the sub-megabase scale is not random and the structures are present in animals as well as in some plants, fungi, and bacteria. TADs are stable units of replication-timing regulation. In general they are higher conserved contacts between cell-states, tissues and species. They range from several kilobases up to megabases.

Training The learning step of a classification problem.

Association Learning The predicted output of unseen examples is a numeric quantity and not a discrete class.

Clustering Finding logical groups of data through association of their features.

Classification Learning Learning from a set of classified examples to classify unseen examples. Output is a single nominal prediction.

Reinforcement Learning An algorithm (here mostly called agent) learns from the consequences of its actions, rather than from being taught explicitly. It selects its actions based on past experiences, maximizing a reward.

Semisupervised Learning A combination between supervised and unsupervised learning.

Supervised Learning ML training with given input and output examples.

Unsupervised Learning ML training with missing output examples. The learning algorithm has to structure the input by its own.

Transcription The process of translating the DNA into the mRNA.

Translation The process when the codons of a spliced mRNA are read by the ribosome complex and the corresponding AAs are concatenated.

Zusammenfassung

Bei der Genomsequenzierung stellt die Interpretation der nicht-kodierenden Bereiche des Genomes immer noch eine bedeutende Herausforderung dar. Im Vergleich zu den häufigen, meist neutralen, genetischen Veränderungen stellen Varianten, welche mit Krankheiten oder anderen Eigenschaften assoziiert sind, eine winzige Minderheit dar. In diesem Sinne stehen Methoden zur Vorhersage von nicht-kodierenden, krankheitsassozierten Varianten durch Maschinelles Lernen (ML) dem Henne-Ei-Problem gegenüber – solche Veränderungen sind ohne ML schwierig zu finden, aber ML ist meistens erst dann erfolgreich, wenn eine ausreichende Anzahl von Beispielen gefunden wurde. Die neuesten Methoden zur Vorhersage von Varianten durch ML integrieren keine speziellen Vorhersagetechniken um dieses Ungleichgewicht zu behandeln, was zu einer relativ schlechten Performanz mit reduzierter Sensitivität führt, da die zugrundeliegenden Anwendungen zur genomweiten Bewertung von Varianten nicht im Gleichgewicht sind.

In dieser Arbeit stelle ich hyperSMURF vor, einen Algorithmus, der Verfahren zum Lernen von Daten mit extremer Differenz zwischen Observationsmengen benutzt, basierend auf Techniken zur Stichprobewiederholung und einer Hyper-Vereinigung. Im Bereich von nicht-kodierenden Varianten, welche mit Mendel'schen oder komplexen Erkrankungen assoziiert sind, übertrifft er vorherige Methoden. Ich zeige, dass das ML durch explizit entwickelte Techniken für Daten mit hohem Ungleichgewicht ein Schlüsselkonzept für eine robuste und genaue Vorhersage in diesem Bereich ist. HyperSMURF ist open-source und in R und Java implementiert und kann somit mühelos in anderen Wissenschaftsprojekten genutzt werden um krankheits-assoziierte Varianten unter Millionen von neutralen Veränderngen bei Genomsequenzierung zu finden.

Des Weiteren wurde mit Hilfe des Algorithmus eine neue Bewertungsfunktion für Mendel'sche regulatorische Mutationen entwickelt (ReMM score). Sie ist signifikant besser als andere Bewertungen zum Erkennen von regulatorischen Varianten bei seltenen genetischen Funktionsstörungen. ReMM score ist in dem Analyseframework Genomiser integriert, welches nicht nur kodierende, sondern auch relevante nicht-kodierende genomische Varianten bewertet und diese dann einer Erkrankung zuordnen kann. Genomiser benutzt hierfür Bewertungsfunktionen und kombiniert diese mit Allelefrequenzen, der Raumstruktur von Chromosomen und der phänotypischen Relevanz von Varianten zu bekannten Syndromen. Dadurch wird Genomiser zu einem effizienten Tool zur Entdeckung von neuen regulatorischen Varianten bei Medel'schen Erkrankungen.

Curriculum Vitae

For reasons of privacy, the curriculum vitae is not part of the online version.

Eigenständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig und nur mit den angegebenen Hilfsmitteln angefertigt habe und dass alle Stellen, die dem Wortlaut oder dem Sinne nach anderen Werken entnommen sind, durch Angaben von Quellen als Entlehnung kenntlich gemacht worden sind. Diese Arbeit habe ich nicht schon einmal in einem früheren Promotionsverfahren eingereicht.

Ort, Datum

Unterschrift