The Pennsylvania State University

The Graduate School

SUPERVISED VARIANT PRIORITIZATION AND ITS EVALUATION

A Thesis in

Bioinformatics and Genomics

By

Jui-Shan Lin

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

May 2021

The thesis of Jui-Shan Lin was reviewed and approved by the following:

Yifei Huang

Assistant Professor of Biology

Thesis Advisor

George PJ Perry

Associate Professor of Anthropology and Biology

Chair, Intercollege Graduate Degree Program in Bioinformatics and Genomics

Qunhua Li

Associate Professor of Statistics

Associate Chair, Intercollege Graduate Degree Program in Bioinformatics and Genomics

ABSTRACT

Nonsynonymous single nucleotide variants (nsSNVs) play a critical role in rare genetic disorders. These mutations result in single amino acid substitutions in protein sequences, which may further affect protein function. Various computational tools have been developed to predict the functional and clinical impact of nonsynonymous variants. Nevertheless, the pre-existing variant prioritization methods have limited accuracy, which hinders their applications in both basic and translational studies. To address this problem, here I develop and evaluate an inheritance-mode-aware machine learning model for prioritizing nonsynonymous pathogenic variants.

First, although dominant and recessive pathogenic variants may have distinct characteristics, most pre-existing tools for variant prioritization are agnostic to the inheritance mode of pathogenic variants. I hypothesize that an inheritance-mode-aware machine model will potentially improve the prediction of pathogenic variants when we know the modes of inheritance of the associated disease. To test this hypothesis, I split ClinVar variants into three groups: variants associated with dominant diseases, variants associated with recessive diseases, and those with unknown modes of inheritance. Then, I used logistic regression, support vector machine (SVM), and gradient boosting to train supervised models by the inheritance mode of the variants. Among them, the gradient boosting models trained on variants without labels of inheritance has the best performance.

Second, I compared the performance of my gradient boosting models with 14 well-used variant prioritization tools using known pathogenic variants from ClinVar. To prevent the potential ascertainment bias in ClinVar variants, I also evaluated model performance using *de novo* missense mutation associated with severe developmental disorders. My gradient boosting models showed satisfactory performance in both of the two tests.

Last, I examined the characteristics of the misclassified variants in our model. I found that the variants without assertion criteria in ClinVar account for 50% of false-negative variants, however, model performance could not be improved by simply removing those variants from training data. I also found that the distributions of predicted residue exposure level are different between correctly and wrongly classified variants: pathogenic variants are more difficult to predict if they are located on protein surfaces, while the benign variants are more likely to be incorrectly predicted when they are buried in the hydrophobic cores of proteins.

TABLE OF CONTENTS

List of Figures

List of Tables

ACKNOWLEDGEMENTS

First and foremost, I would like to share my gratitude to my supervisor, Dr. Yifei Huang, for guiding me to this fascinating world of machine learning. The completion of my dissertation could not have been done without his strongest support. Dr. Huang keeps pushing me through the frustration barrier by bringing new ideas and shared his own learning experience. Thank you for always being patient and considerate.

Secondly, I want to extend my sincere thanks to my two mentors, Dr. George Perry and Dr. Cooduvalli Shashikant. Both of them take good care of me, although they have such a tight schedule as program directors. The meetings with mentors ensure that I am on the right path and have the correct balance between research, school life, and personal life.

Thirdly, a big thank you to my lab members and friends. Graduate school has been a long and sometimes lonely journey, especially during this never-ended pandemic. Thanks for being my role models and warriors to fight along.

Finally, I want to give my warmest gratitude to my family. Regardless of how worried they were when I decided to pursue a graduate degree overseas, they have always given me unconditional love and faith. They have been my strongest support all along. This year is an unforgettable year. By overcoming these difficult times, I feel truly blessed and beloved.

Chapter 1

INTRODUCTION

1.1 Background

1.1.1 The study of variant prioritization

In biological research, we often use mutagenesis to study the function of genes and their involvement with diseases. Although many novel breakthroughs like CRISPR/Cas9 and random mutagenesis make this process less time consuming, it is still labor-intensive and expensive to examine the function of genes *in vivo*. In contrast, naturally occurring mutations in human act like a spontaneous mutagenesis experiment. By studying those mutations, we can investigate the function of the mutated genes and obtain new insights into their roles in human diseases.

Thanks to the rapid development of next-generation sequencing and machine learning techniques, the research of variant interpretation has been facilitated in the past decades. By pinpointing variants that are likely associated with human genetic diseases, we can identify causal genes of various diseases, which can be later used to elucidate the molecular mechanisms of genetic disorders for efficient diagnosis and treatment.

However, even in a patient with a monotonic genetic disorder, there are still thousands of genetic variants, whereas only a tiny proportion of them are disease-causing. Common variants that are found in genome-wide association study (GWAS) often have weak effect size and are usually benign. The procedure of finding causal variants is even harder than finding a needle in a haystack. Although a lot of previous

studies tried to tackle this issue, their high false positive rates and inevitable overfitting still leaves rooms of improvement.

Mendelian disorders, diseases that result from single genetic alteration, are the perfect subjects for studying variant prioritization. Typically, these genetic abnormalities are rare, but are highly penetrant. In general, Mendelian disorders are early onset. Also, due to its low allele frequency and their well-known inheritance from Mendel's Laws, the study of Mendelian disorders often involves family pedigree analysis.

1.1.2 Mutations

The classification of deleteriousness of missense mutation becomes an important factor in identifying disease-causal variants. We first narrow down the complication of mutation by only discussing point mutations. Because of the redundancy in genetic code, synonymous mutations usually have no effects on protein sequences. On the contrary, non-synonymous mutations, which lead to amino acid substitutions, are more likely to be pathogenic.

In many cases, when the substituted amino acid has similar chemical properties with the original one, the translated protein might still function properly. This kind of missense mutation is named as a "neutral" mutation. On the other hand, when the substituted amino acid has very different properties, such as different polarity, it may result in protein misfolding and degeneration. This kind of missense mutation is named as a "deleterious" mutation.

Since deleterious mutations often reduce the fitness level of an individual and will be eliminated by natural selection, many missense mutations identified in a patient's genome are likely to be neutral, making it extremely difficult to distinguish pathogenic from neutral variants. Also, the mutation location matters in the prediction of missense variant effects. Even if a missense variant is one base away from another nucleotide, they might have completely different effects on protein function and disease risk. Thus, it is challenging to predict pathogenic variants.

1.2 Previous methods

Several previous studies have pinpointed pathogenic variants using various statistical models and predictive genomic features. In this session, I will first introduce the features that are commonly used in variant prioritization. Then, I will discuss the statistical methods used in previous studies.

1.2.1 Genomic features

The features used for variant effect prediction can be classified into three categories—structure-based scores, sequence conservation scores, and functional genomic scores.

1.2.1.1 Structure-based score

The different chemical and physical properties between the wild-type and the altered amino acid will potentially affect protein folding (*e.g.,* destabilizing the hydrophobic core of the protein), and further impacts its stability and interaction with other particles. The unfolding or misfolding of proteins are reported to cause several diseases, commonly regarded as loss-of-function **(**LOF). Thus, structure-based computational scores can be used to predict pathogenic variants.

There are several ways to translate protein structure information into a variant effect score. The raw data often come from protein structure data base (PDB), or dictionary of secondary structure in proteins (DSSP). To predict the potential effect of missense variants, the location of the altered amino acid matters a lot. However, with our limited understanding of 3D protein structures, we need other information to support. This is where sequence conservation scores come in handy.

1.2.1.2 Sequence conservation score

Due to negative selection, mutations that severely impact an individual's fitness will be eliminated throughout the evolutionary process. Hence, missense mutations in conserved protein sites are more likely to be pathogenic, while mutations located outside of these regions are more likely to be tolerated. By inferring sequence conservation across species, computational methods leverage the signatures of selection pressure as an to quantify the deleteriousness of genetic mutated variants.

Sequence conservational score is the first feature used in variant prioritization. It can be used to classify variants in both coding and noncoding regions. Also, it is powerful enough that some methods can already achieve satisfying results solely using this piece of information (*e.g.*, SIFT (Ng and Henikoff, 2003) and PROVEAN (Choi *et al.*, 2012)). However, we need to be aware of some limitations of sequence conservation scores:

1. Sequence conservation between species reflects long-term selection. Thus, it is challenging to distinguish weak and strong selection from sequence conservation scores.
2. Mutations that affect fitness level in other species may not be equally influential in humans (epistasis).
3. Although many human proteins have a large number of orthologs in other vertebrates, there are still proteins without many orthologous sequences. Due to a lack of data, sequence conservation may not be an accurate predictor of variant effects in these genes.
4. Finally, even if we know that a mutation will cause the LOF of a protein, it is still challenging to establish a causal relationship between the protein and the disease of interest.

Therefore, the goal of predicting the deleteriousness of variant could not be achieved solely based on conservation scores.

1.2.1.3 Other features

Predicting how the mutated variants impact protein function is a powerful feature in prioritizing deleterious variants. However, this does not work for noncoding variants. As it is becoming clear that many disease-causing variants are in non-coding regions, we will need some features that can also work in non-coding regions (Hindorff *et al.*, 2009). The features often used in non-coding regions are related to epigenomic, like chromatin accessibility, RNA expression, DNA-protein interactions, and histone modifications (Zhou and Troyanskaya, 2015; Kelley, Snoek and Rinn, 2016). Most of these data are from the Encyclopedia of DNA Elements (ENCODE) and the Roadmap Epigenomics project (Dunham *et al.*, 2012; Roadmap Epigenomics Consortium *et al.*, 2015).

1.2.2 Ensemble scores

Each method has its own strengths and limitations. Owing to the different features and training methods, tools that build for variant prioritization often disagree with each other. Many studies have proven that combining different multiple genomic features can improve the accuracy of variant prediction (*e.g.,* REVEL (Ioannidis *et al.*, 2016), CADD (Rentzsch *et al.*, 2019)(Kircher *et al.*, 2014), DANN (Quang, Chen and Xie, 2015), CONDEL (González-Pérez and López-Bigas, 2011), Eigen (Ionita-Laza *et al.*, 2016), among others). Individual variant scores used in these ensemble methods include PolyPhen-2 (Adzhubei *et al.*, 2010) , SIFT (Ng and Henikoff, 2003), and PROVEAN (Choi *et al.*, 2012). Although building a new method by integrating multiple scores looks straightforward and convenient, one should keep in mind that adding supervised scores in training data might cause data contamination. Also, if there is no new feature added or no real innovation in feature extracting process, it is difficult to achieve significantly better performance by merely combining other scores together.

1.2.3 Statistical methods

To build up an efficient machine learning model for variant prediction, there are two kinds of training methods. (I) Supervised methods and (II) Unsupervised/Semi-Supervised methods.

Supervised methods are directly trained on variant data with labeled pathogenicity. It can be easily treated as a classification problem. Supervised methods aim to fit a model that can best capture the relationship between the input features and the label. Another category of methods is unsupervised/semi-supervised, and they did not directly use labeled pathogenic variants. Usually, unsupervised learning uses clustering methods to learn the inherent data structure without pre-labeling the data. However, in variant prediction, unsupervised methods are trained on simulated data or with surrogate objective functions. The tools using this unsupervised/semi-supervised method include CADD (Kircher *et al.*, 2014), PrimateAI (Sundaram *et al.*, 2018), and UNEECON (Huang, 2020). By creating simulated data, they can either use the same machine learning algorithms as in supervised methods or customized models.

The most wildly used supervised algorithms in variant prioritization are SVM (Cortes and Vapnik, 1995; Kircher *et al.*, 2014), Random Forests (Liaw and Wiener, 2002)(Ioannidis *et al.*, 2016), Gradient Boosting(Friedman, 2001), and logistic regression. Although in other domains of machine learning, such as computer vision and audio processing, models that can handle non-linear relationships between features often have better performance, linear models often show good performance in biological research.

1.3 Limitations from previous methods

While many existing methods can predict the clinical impact of missense mutations, they often suffer from the following four limitations. First, there are biases in their training or testing data. In many cases, supervised methods overfit the training data, making their methods overly optimistic and hard to

generalize (Grimm *et al.*, 2015). Also, it is reported that many tools are trained with common benign variants and rare pathogenic variants (Ioannidis *et al.*, 2016)(Grimm *et al.*, 2015). However, common variants and rare variants have distinct characteristics, resulting in even larger biases. Second, existing methods used different features, data, and algorithms, and their predictions seldom stay consistent with each other (Ioannidis *et al.*, 2016). It is hard to find relative merits and compare their performance objectively. Third, in many novel ensemble prediction tools, they only considered variant level constraints ignored gene-level information. (*e.g.*, Condel (González-Pérez and López-Bigas, 2011)). Last but not the least, current methods neglect the mode of inheritance, which might be informative in variant prediction.

Hence, I introduced an ensemble tool that utilized UNEECON, a feature that considers both variant-level and gene-level constraints. Also, I separated known pathogenic variants into three categories based on their inheritance mode and trained an inheritance-mode-aware model on each of the dataset. Last, I compared the performance of our models with existing methods using ClinVar variants and *de novo* missense mutations.

Chapter 2

METHODOLOGY

2.1 Building main dataset

2.1.1 Extracting pathogenicity and inheritance mode from ClinVar

I used genomic features from the UNEECON model (Huang, 2020) as the primary variant features. In this data, there are 29 features, containing both sequence conservation scores and structural information. Since UNEECON is not a supervised method, the pathogenicity of variants is not included in this model. Hence, the pathogenicity of genetic variants needs to be combined with genomic features from UNEECON to perform supervised learning.

I used ClinVar data to obtain the pathogenicity and inheritance mode of missense variants. The ClinVar data I used is based on the hg19/GRCH37 assembly (https://ftp.ncbi.nlm.nih.gov/pub/clinvar/vcf_GRCh37/archive_2.0/2020/; released on January 13, 2020). In this dataset, the pathogenicity of variants is labeled as 'CLINSIG' (clinical significance on ClinVar submitted record) under the "info" column. I only extracted four categories of variants from the total of twelve: benign, likely benign, likely pathogenic, and pathogenic. I did not find any contradicted label in these four categories. After binarizing the benign and likely benign variants as 0, and pathogenetic and likely pathogenic variants as 1, I added this information to the primary variant feature data and named this column as "clinvar_result". This dichotomous column was used as the response (dependent) variable in later model training. For clarity, the pathogenic and benign here refers to the functional consequences of Mendelian disorders that contributed by missense variants throughout this thesis. It should not be confused with damaging that undergo purifying selection.

In the same ClinVar data, I obtained the information of whether a variant is likely associated with a dominant or a recessive disease, which can be used to train an inheritance-mode-aware model for variant interpretations. It is worth noting that the mode of inheritance was not used as a training features but what we use to separate our dataset by their dominancy was used to split the training data. However, some variants are labeled as both dominant and recessive because a single variant can be either dominant or recessive in different diseases. Hence, I regarded the variants labeled as both dominant and recessive to be dominant and treated the variants that are only labeled as recessive to be recessive. I used this definition of inheritance mode dominancy consistently throughout this thesis. I combined it with the dataset that already contains pathogenicity information from the previous step. However, the information of inheritance mode was unknown in most of pathogenic missense variants. , I regarded these variants to have an unknown mode of inheritance.

After extracting the pathogenicity and inheritance mode from the ClinVar data and combining them with the genomic features from the UNEECON paper, I obtained a new dataset with 30 features (Supplement A.2.1). These variables were later used in constructing predictive models.

2.1.2 Separating dataset by the mode of inheritance

Based on the annotated mode of inheritance from ClinVar, I separated ClinVar variants into three subgroups: (1) variants with dominant mode of inheritance, (2) variants with recessive mode of inheritance, and (3) variants with unknown mode of inheritance.

To be mentioned, the aim of this model is to predict the pathogenicity of genetic variants. However, two of the three datasets were highly imbalanced. To prevent the skewness in classification, I took 677 and 1036 benign missense variants with unknown mode of inheritance to the dominant and recessive groups respectively, making both of them balanced. After moving a subset of benign variants to the dominant and recessive datasets, I made the dataset with unknown mode of inheritance balanced by randomly removing

the surplus variants. Finally, I obtained a dominant dataset with 1950 variants, a recessive dataset with 2764 variants, and a third dataset consisting of 62634 variants with unknown mode of inheritance (Table 2-1).

	pathogenic	benign	total
dominant	975	975 (677 variants from unlabeled group)	1950
recessive	1382	1382 (1036 variants from the unlabeled group)	2764
unlabeled group	31317	31317 (randomly removed 15046 variants)	62634

Table 2-1: **The numbers of variants in three inheritance-mode-aware datasets after balancing their pathogenicity.** The dataset with type "unlabeled" contains the variants labeled with pathogenicity but have unknown model of inheritance. We took some portion of variants from the unlabeled group to dominant and recessive groups to make their data balanced, preventing possible skewness in our prediction. Then, we also made the unlabeled group balanced.

2.1.3 Adding UNEECON score to the data

UNEECON score was not included in the previous training dataset, so I added it to my dataset independently. It is worth noting that, although the UNEECON score was based on the GRch37 (hg19) assembly, it was 1-based rather than 0-based (the three dataset we created previous were all 0-based). Because the UNEECON file was too large to directly be processed by the pandas library in python, I used tabix to efficiently combine the UNEECON score with other data. The code I used to perform the integration is available in appendix B.1.

2.1.4 Feature standardization

After getting all the features, I performed feature scaling by standardizing continuous features. Because continuous features are at different scales before scaling, this ensures machine learning algorithms

that I used in the later training process work properly. In this standardization process, I subtracted each feature with its mean, and then divided it by its standard deviation.

2.2 Accessing data for model evaluation

2.2.1 Comparison of variant scores using ClinVar data

In order to compare the performance of my variant scores with that of pre-existed methods, I obtained various variant scores from dbNSFP.((Liu, Jian and Boerwinkle, 2013),(Liu *et al.*, 2020)) dbNSFP is a handy database with numerous pre-existed variant scores for each potential nonsynonymous single-nucleotide variants. It saved me from the tedious procedures of independently extracting and processing scores from each variant prioritization method. I used the fourth version of dbNSFP and took 14 well-used variant prioritization tools in this comparison (Table 2-2). Some of the tools have more than one score for each variant (*e.g.*, MPC, FATHMM, VEST4, and LIST-S2). I retained the maximum score in every variant to ensure that only a single score was used in downstream analysis. Also, among all the methods, FATHMM score is the only one that is negatively correlated to pathogenicity: the smaller the score is, the more disease-prone the variant is. Hence, I used of the negative value of FATHMM score in model evaluation SIFT, LRT, PROVEAN, and Polyphen-2 were not included in this comparison test because we used them as training features in our model.

Method	Category	Prediction model	Reference
Meta SVM	Ensemble score	Radial kernel SVM	(Dong *et al.*, 2015)
Meta LR	Ensemble score	Logistic Regression	(Dong *et al.*, 2015)
M-CAP (v1.3)	Ensemble score	Gradient Boosting	
REVEL	Ensemble score	Random forest	(Ioannidis *et al.*, 2016)

CADD (v1.6)	Ensemble score	Linear kernel SVM	(Kircher *et al.*, 2014)
DANN	Ensemble score	Deep Neural Network	(Quang, Chen and Xie, 2015)
Integrated fitCons (INSIGHT)	Ensemble score	Clustering Method	(Gulko *et al.*, 2015)
ClinPred	Ensemble score	Random forest and Gradient boosting	(Alirezaie *et al.*, 2018)
MPC	Sequence conservation	Logistic regression	(Samocha *et al.*, 2017)
FATHMM	Sequence homology	Hidden Markov Model	(Shihab *et al.*, 2013)
PrimateAI (v0.2)	Orthologous sequence and functional prediction	Deep Neural Network	(Sundaram *et al.*, 2018)
VEST4	Protein activity	Random Forest	(Carter *et al.*, 2013)
LIST-S2	Sequence conservation	Bayes Rules	(Malhis *et al.*, 2020)
H1-hESC_fitCons (FitConsH)	Epigenomic signal of H1-hESC	Clustering Method	(Gulko *et al.*, 2015)

Table 2-2: The methods included in model comparison. The scores of each method are extracted from dbNSFP. CADD: Combined Annotation-Dependent Depletion; DANN; deleterious annotation of genetic variants using neural networks; INSIGHT: Inference of Natural Selection from Interspersed Genomically Coherent Elements; MPC: Missense badness; FATHMM: Functional Analysis Through Hidden Markov Models; REVEL: rare exome variant ensemble learner; VEST: Variant Effect Scoring Tool; LIST: Local Identity and Shared Taxa.

2.2.2 Comparison of variant scores using *de novo* missense mutations

I also used *de novo* missense mutations from the Deciphering Developmental Disorders (DDD) project to evaluate the performance of variant scores (Turner *et al.*, 2017). I downloaded both Simons Simplex Collection (SSC) and non-SSC samples from denovo-db. The data version is 1.6.1 (released on August 19, 2018) and is available at https://denovo-db.gs.washington.edu/denovo-db/Download.jsp.

First, I ruled out structural mutations and only kept de novo single nucleotide variants. Also, I only kept variants with the functional classification of 'missense.' Based on whether the variants are from probands or controls, I gave them binarize labels. Then, I removed all variants that are from targeted gene studies, retaining only those reported in exon sequencing or whole genome sequence studies.

After getting the desired *de novo* variants, I integrated them with variant scores reported by other tools and my machine learning models. Again, I used the locations and the mutated variants as a common column and then integrated variant scores from the dbNSFP together. Likewise, I added genomic features and scores from the UNEECON study using tabix. This time, I built my own index file with tabix.

Here my primary goal is to predict de novo missense variants associated with developmental disorders. However, there were not enough control variants in non-SSC samples. Hence, I added the controls from SSC and combined them with the non-SSC ones. After the data processing process, I obtained a *de* novo mutation dataset with 4990 single nucleotide variants. Among these variants, ~73% are from probands, while the rest are from controls. Details of the processing steps are documented in appendix B.3.

2.2 Creating training, testing, and validation datasets

For all the processes separating dataset into training, testing, and validation sets, I used the train_test_split function from the sklearn package. The random_state was always set to 42, ensuring that the results can be reproduced. Unless noted specifically, I always randomly separated validation data, testing data, and training data in a ~1:1:8 ratio throughout this thesis.

To overcome the possible ascertainment bias in model evaluation because of randomly separated data (Grimm *et al.*, 2015), I also generated chromosome-separated data. Variants on chromosome 1 were

used as testing data, while those on chromosome 2 were used as validation data. I treated the remaining variants as the training dataset. The numbers and the proportions of variants in each dataset are listed in Supplement A Table 2.

2.3 Regularization in machine learning algorithms

I used logistic regression, support vector machine, and gradient boosting to predict pathogenic variants from benign variants. Since there were 30 features in the training set, to reach better performance and avoid overfitting, I performed regularization during model training.

2.3.1 Ridge regularization

Using the implementation of generalized linear models in scikit-learn, we can directly perform ridge regularization without explicitly specifying cross-validation independently. I chose Ridge regularization (L2) over LASSO (L1) because it does not shrink parameters directly to zero. Here, I set the penalty term to 'l2' in sklearn.linear_model.LogisticRegression class.

2.3.2 Cross validation

To get the best hyperparameters in the support vector machine and gradient boosting, I performed cross-validation tunning. Here, I used the class of sklearn.model_selection.GridSearchCV. By default, it runs 5-fold cross-validation. For the support vector machine, I tested it with both the radial basis function (RBF) kernel and the linear kernels. In both the kernels, I evaluated model performance with a gamma parameter of 0.01, 0.1, 1, and10, and a regularization parameter "C" of 1, 10, and 100. For gradient boosting, I evaluated its performance with a learning rate of 0.05, 0.1, and 0.15, the number of boosting stages of

100, 150, and 200, and the maximum depth of 3,5, and 8. The best-performing hyperparameters are available in supplement A.2.2.

Chapter 3

Results

3.1 Training inheritance-mode-aware models

Dominant and recessive pathogenic mutations may have distinct characteristics. However, there are no previous method designed to separately predict dominant and recessive variants. I hypothesize that accommodating the mode of inheritance may improve the prediction of deleterious variants.

To test this hypothesis, I separated missense variants in training data into a set with dominant mode of inheritance and another set with a recessive mode of inheritance. I used logistic regression, SVM, and gradient boosting to predict pathogenic variants from benign variants in each variant set separately. To prevent overfitting, I performed Ridge regularization in logistic regression, while in SVM and gradient boosting, I performed 5-fold cross-validation to select the best-performing hyperparameters for the data (Table S1).

I used features that contain sequence conservation scores, protein structure prediction, and regulatory information to build a training dataset with 30 features. (Supplement A.2.1). Logistic regression is a linear model and here served as a baseline for SVM and gradient boosting. Although in many other machine learning fields, non-linear models often out-perform linear ones. In bioinformatics, this is not always the case.

Figure 3-1 shows the performance of machine learning models in the datasets with different inheritance modes. Gradient boosting models have the best performance in both dominant dataset and the one with unknown modes of inheritance. However, no model has statistically superior performance in the recessive dataset (Table 3-1).

To check if those auROC values are significantly different from each other, I used the DeLong test to quantify their statistical significance (Table 3-1). The differences in area under ROC curve (auROC) between machine learning models were significant in the dataset with dominant and unknown mode of inheritance, whereas these differences became neglectable in the recessive datasets.

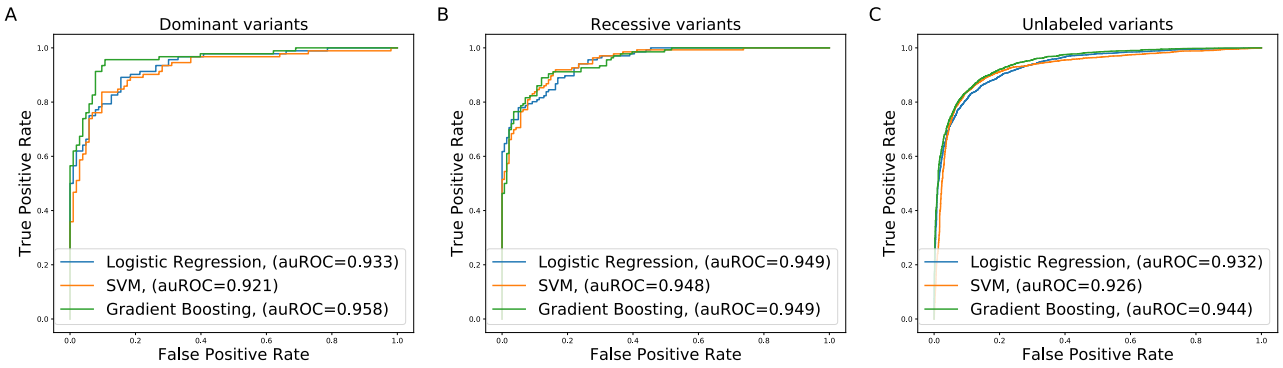


Figure 3-1: Receiver operating characteristic (ROC) curves of different machine learning models in variants with different modes of inheritance. The legends specify the machine learning methods used to train the model. The group "unlabeled" are the variants with unknown modes of inheritance. All models under the same mode of inheritance have the same training, testing, and validation data.

	Dominant	Recessive	Unknown modes of inheritance
Logistic regression vs. SVM	0.0910	0.9229	7.6×10^{-28}
Logistic regression vs. Gradient Boosting	0.0085	0.6654	1.9×10^{-3}
SVM vs. Gradient Boosting	0.0002	0.7289	1.18×10^{-21}

Table 3-1: Results of DeLong test p-values for the performance comparison between three different statistical models. The column shows the results from datasets with different modes of inheritance, whereas the rows indicate the comparison of two algorithms.

In this comparison, logistic regression model had a higher auROC than SVM in variants with unknown modes of inheritance. This result is consistent with what reported in a previous study (Quang, Chen and Xie, 2015). Both trained by gradient boosting, dominant variants model has a higher auROC

value compared to the model trained and tested on variants with unknown mode of inheritance (Figure 3-1). However, since the two auROC were measured by two different testing data, it is hard to tell if this difference is significant. So we further tested their performance using the same testing data.

I observed that gradient boosting models have the best performance among the three algorithms. Hence, I further tested how the three inheritance-mode-aware models perform when training data and testing data do not have the same inheritance mode (Figure 3-2). Similar tests done by logistic regression and SVM are shown in Supplement A.1.2. and Supplement A.1.3. In the testing data composed of recessive variants, I found that models trained and tested on variants with the same inheritance mode do not guarantee a higher auROC. I observed dominant-trained model was not inferior to the recessive-trained model in recessive testing data but had a superior performance in dominant testing data. Interestingly, in the testing data made out of variants with unknown inheritance modes, the two models had similar performance. Figure 3-2 also pointed out that the model trained on variants with unknown inheritance modes significantly outperformed the other two models. This result does not support my previous hypothesis that training the inheritance-mode-aware model will improve model prediction.

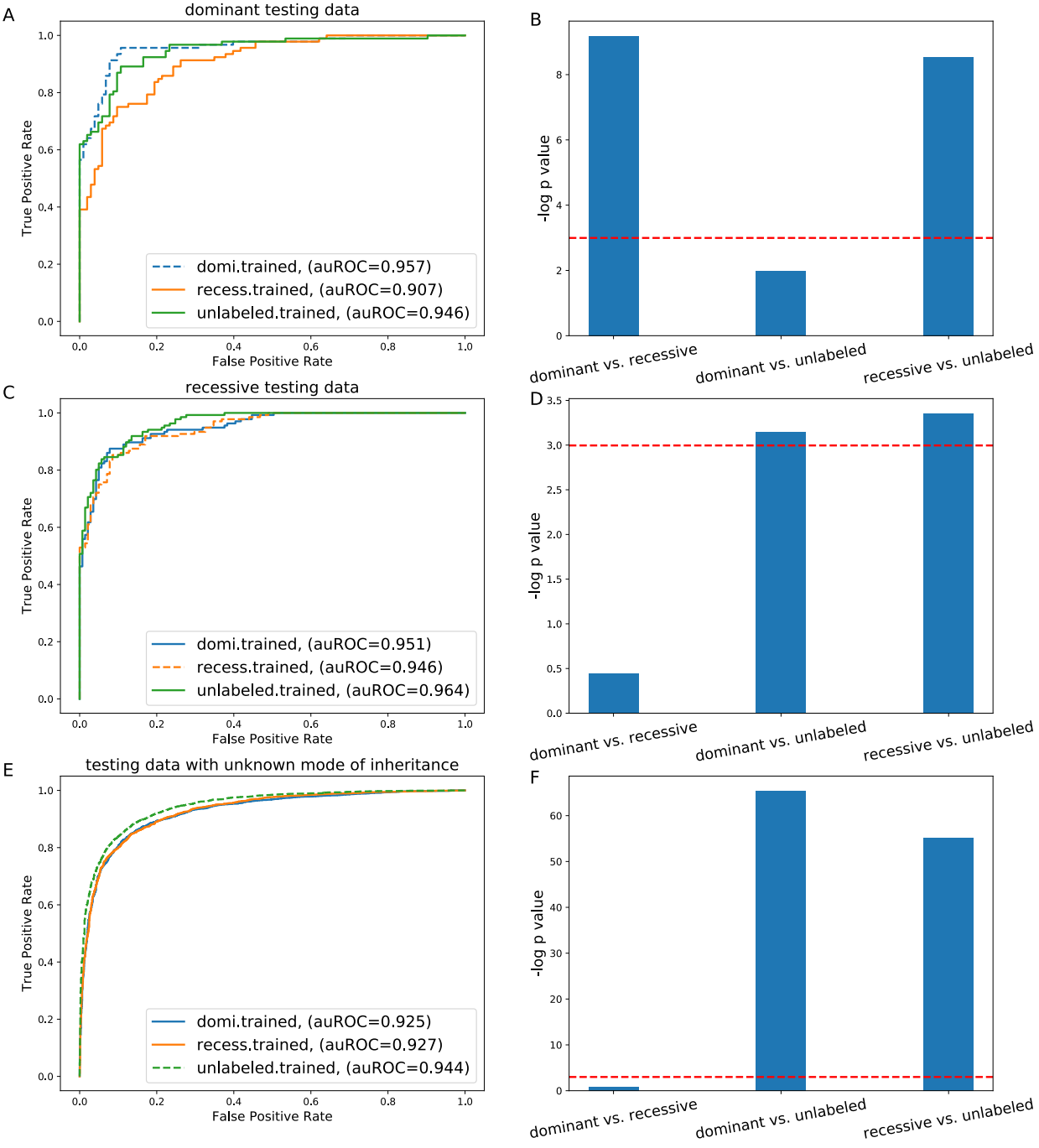


Figure 3-2: The performance of models trained and tested on variants with different modes of inheritance using gradient boosting. (A)(C)(E) shows the results of ROC curves in testing data with different modes of inheritance, all trained with gradient boosting. Models trained and tested on matched datasets are labeled with dashed lines. (B)(D)(E) are the p-values of (A)(C)(E) using DeLong test, showing

how significant the model is different from other models in the same testing data. The red dashed line indicates the position where the p-value equals 0.05. Noted that the log here are log with the base of *e.*

To see if the features have different predictive power over dominant or recessive data, I examined regression coefficients from dominant and recessive models trained with logistic regression (Figure 3-3). Many features used in the training process were highly correlated, so this figure can only explain if certain features have higher weights over dominant or recessive data than their contribution to the logistic regression model. As shown in this figure, UNEECON showed a higher coefficient in the dominant model than in the recessive model, probably because it is a metric of negative selection on heterozygous mutations. Also, PROVEAN prediction, the binary PROVEAN prediction of functional variants, and PROVEAN score had a higher coefficient in the dominant model than in the recessive model. Whereas mammalian phyloP , a feature that measures sequence conservation across mammals, and PredSSC, the secondary protein structure prediction have a higher coefficient in the recessive model, however, these two features did not show strong predictive power in the individual gradient boosting model (Figure 3-4).

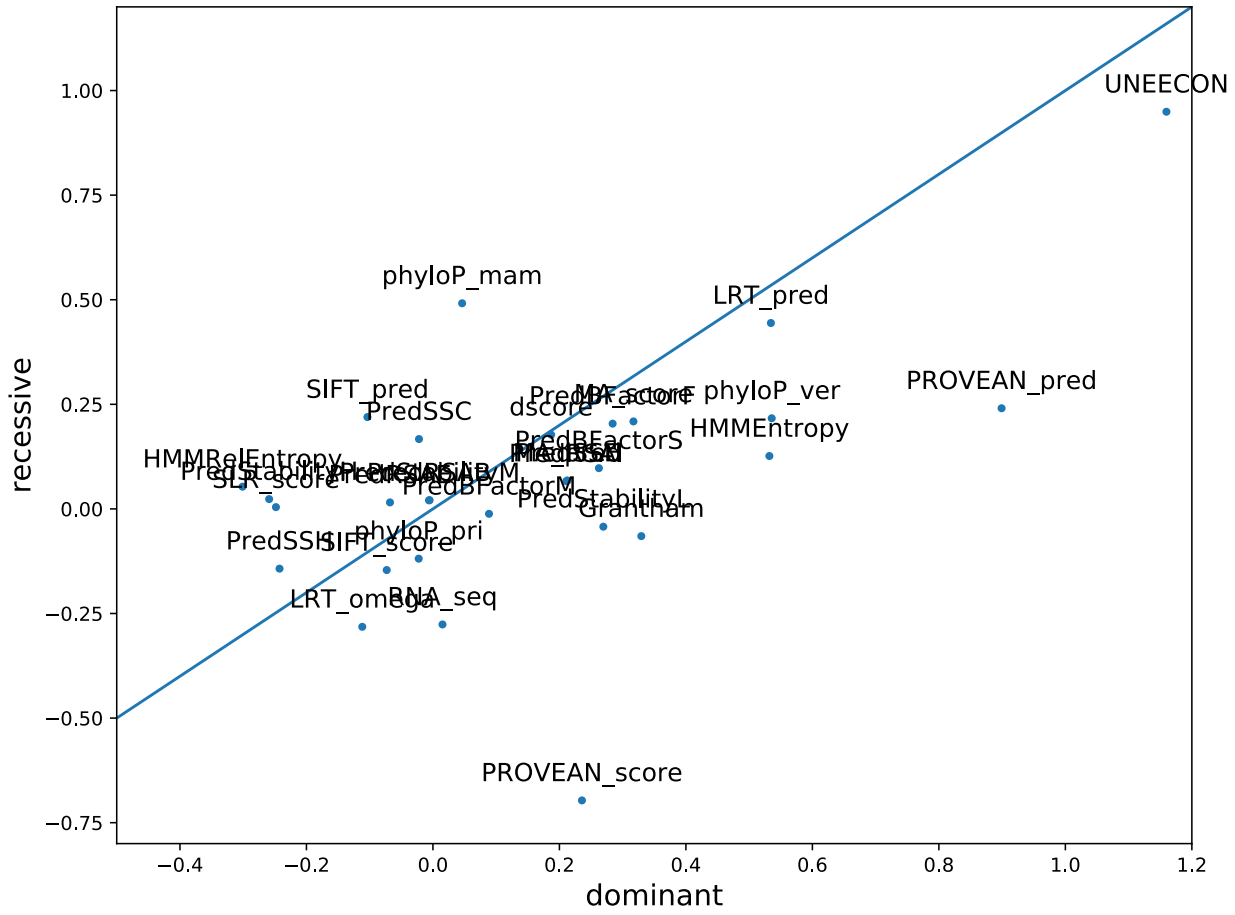


Figure 3-3: The coefficients of the genomic features in logistic regression. The figure shows coefficients trained and tested in dominant variants or recessive variants. The blue line represents that a feature has the same coefficient in the dominant and the recessive models.

Since the gradient boosting model trained on variants with unknown modes of inheritance had the best performance overall, I used this model for downstream comparison and named it "our model." To examine the predictive power of each feature, I trained multiple gradient boosting models using one feature at a time and compared the result with our model (Figure 3-4). In this test, I removed all binary features (SIFT prediction, LRT prediction, MA prediction, PROVEAN prediction), only retained the continuous ones. In figure 3-4, I found UNEECON, SIFT score, PROVEAN score, phyloP vertebrate score, and LRT

omega score were the top five features in this prediction comparison. Interestingly, except for PROVEAN score, the other four features have a slightly higher weight in the dominant model when trained with logistic regression (Figure 3-3, Supplement A.1.4), suggesting that they have more information on dominant variants over recessive variants. This can possibly explain why the dominant variants-trained model can perform better than the model trained by recessive variants.

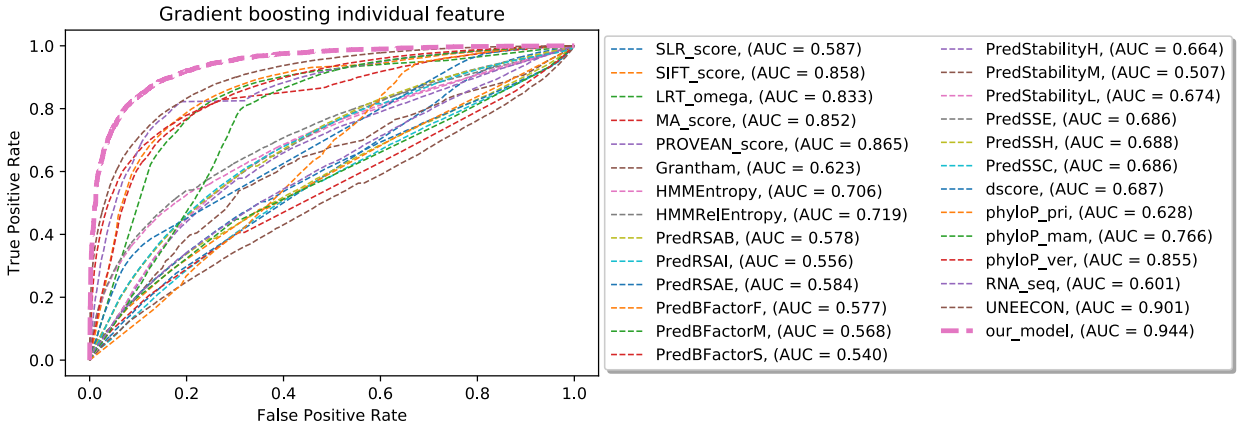


Figure 3-4: Testing the predictive power of each features used in our model. Each ROC curve is an independent gradient boosting model trained by one of the features in our model. "Our model" that showed in the thickest line is the model trained by variants with unknown mode of inheritance, using all 30 features in the training set.

However, it is common that variants from the same gene are all labeled as benign or pathogenic in the ClinVar database (Grimm *et al.*, 2015). In this case, the performance of a supervised machine learning will be overoptimistic if variants from the gene are present in both the training and testing datasets. To prevent this kind of data leakage, I separated training, testing, and validation sets by their chromosomes, which ensures that genes in the testing set are not present in the training data. With this chromosome-separated validation test, I obtained a similar result (Supplement A.1.4). This ensures the high performance of our model is not due to the data leakage.

3.2 Model comparison with pre-existed tools

There are many methods developed for variant prioritization. Thus, it is interesting to examine which tools have higher power in predicting the pathogenicity of missense variants. Hence, I compared the performance of our model with other 14 well-used methods extracted from dbNSFP (Figure 3-5, Table 2-2). In this comparison, our model showed state-of-the-art performance and was only outperformed by a few supervised methods, such as ClinPred and REVEL. It is worth noting that the performance of previously developed supervised methods may be overoptimistic in this comparison, because a subset of variants used in my testing data may have also been used as training data in previous studies. Since I did not have access to the training data of previous methods, it is difficult to address this problem in this thesis. In particular, the high performance of ClinPred score might reflex data leakage because it also used ClinVar data for training. Nonetheless, by integrating a large number of genomic features, including UNEECON score that utilized gene-level constraints, our model outperformed most previous methods in predicting ClinVar pathogenic variants. .

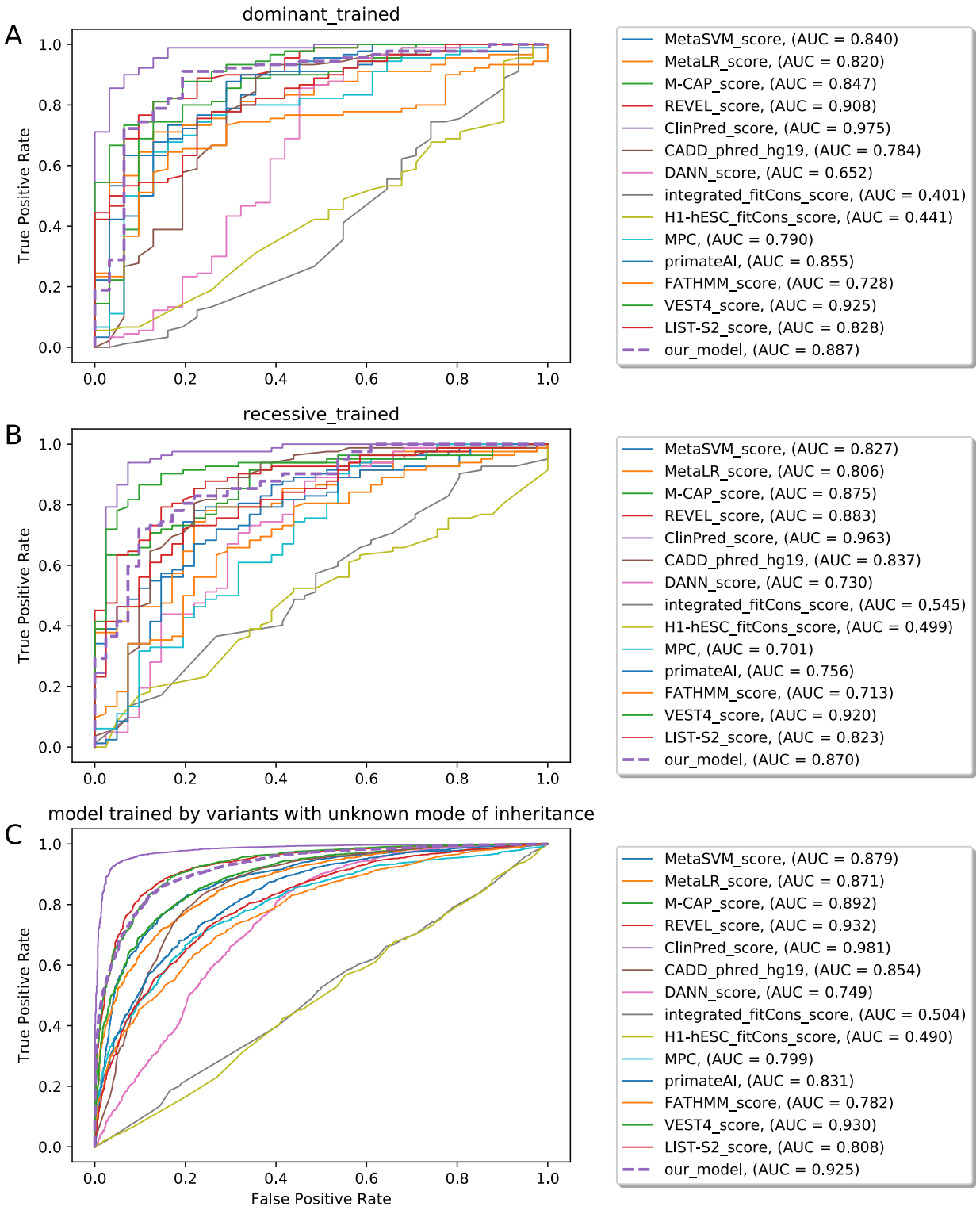

A
dominant_trained
True Positive Rate
MetaSVM_score, (AUC = 0.840)
MetaLR_score, (AUC = 0.820)
M-CAP_score, (AUC = 0.847)
REVEL_score, (AUC = 0.908)
ClinPred_score, (AUC = 0.975)
CADD_phred_hg19, (AUC = 0.784)
DANN_score, (AUC = 0.652)
integrated_fitCons_score, (AUC = 0.401)
H1-hESC_fitCons_score, (AUC = 0.441)
MPC, (AUC = 0.790)
primateAI, (AUC = 0.855)
FATHMM_score, (AUC = 0.728)
VEST4_score, (AUC = 0.925)
LIST-S2_score, (AUC = 0.828)
our_model, (AUC = 0.887)
B
recessive_trained
True Positive Rate
MetaSVM_score, (AUC = 0.827)
MetaLR_score, (AUC = 0.806)
M-CAP_score, (AUC = 0.875)
REVEL_score, (AUC = 0.883)
ClinPred_score, (AUC = 0.963)
CADD_phred_hg19, (AUC = 0.837)
DANN_score, (AUC = 0.730)
integrated_fitCons_score, (AUC = 0.545)
H1-hESC_fitCons_score, (AUC = 0.499)
MPC, (AUC = 0.701)
primateAI, (AUC = 0.756)
FATHMM_score, (AUC = 0.713)
VEST4_score, (AUC = 0.920)
LIST-S2_score, (AUC = 0.823)
our_model, (AUC = 0.870)
C
model trained by variants with unknown mode of inheritance
True Positive Rate
False Positive Rate
MetaSVM_score, (AUC = 0.879)
MetaLR_score, (AUC = 0.871)
M-CAP_score, (AUC = 0.892)
REVEL_score, (AUC = 0.932)
ClinPred_score, (AUC = 0.981)
CADD_phred_hg19, (AUC = 0.854)
DANN_score, (AUC = 0.749)
integrated_fitCons_score, (AUC = 0.504)
H1-hESC_fitCons_score, (AUC = 0.490)
MPC, (AUC = 0.799)
primateAI, (AUC = 0.831)
FATHMM_score, (AUC = 0.782)
VEST4_score, (AUC = 0.930)
LIST-S2_score, (AUC = 0.808)
our_model, (AUC = 0.925)

Figure 3-5: The comparison of predictive power between our model and other pre-existed models. The models I trained with gradient boosting are labeled "domi_model" in (A), "recess_mode" in (B), and "our_model" in (C), all drawn in dashed lines. The three panels here are trained and tested with different dataset of ClinVar variants (A) The comparison with all the variants that are labeled as dominant. (B) The comparison with all the recessive variants. (C) The comparison with all the variants with unknown modes of inheritance.

3.3 Model comparison with *de novo* missense variants from DDD

As mentioned above, using ClinVar missense variants to evaluate model performance may lead to inflated results. To perform a more objective model comparison by preventing possible data leakage, I also used *de novo* missense variants from the DDD project to evaluate model performance(Turner *et al.*, 2017). Unlike variants from the ClinVar database, the variants from the DDD project only contain information of whether they were from probands affected by severe developmental disorders or from unaffected controls. While it is reasonable to assume that all variants from controls are benign, only a subset of variants from probands may be pathogenic. Thus, I used a different approach to evaluate model performance. In this test, I removed integrated fitCons score and H1-hESC fitCons score because their performance in ClinVar testing data were not better than random guessing (Figure 3-5).

Specifically, I evaluated the enrichment of top 10%, 15%, and 20% deleterious variants predicted by each computational method in probands compared with controls (Figure 3-6, Supplement A.1.5). The performance of ClinPred score, the tool that showed the best performance in the previous comparison with ClinVar testing data, decreased significantly in the dataset of *de novo* mutations. Thus, the high performance of this method in the ClinVar data might be due to model overfitting or data leakage. In contrast, UNEECON, an unsupervised method that integrated both variant-level features and gene-level constraints, showed the highest performance in predicting *de novo* mutations associated with developmental disorders.

Besides, other tools with higher performance than our model (*e.g.,* M-CAP, MPC, and primateAI) all showed lower auROCs in ClinVar testing data. This indicates that they may be better in predicting *de novo* pathogenic variants or variants in unknown disease genes. On the other hand, DANN showed the lowest auROC in this test. The training data and features used in DANN are derived CADD, so it is also reasonable that CADD did not have a significantly better performance than DANN.

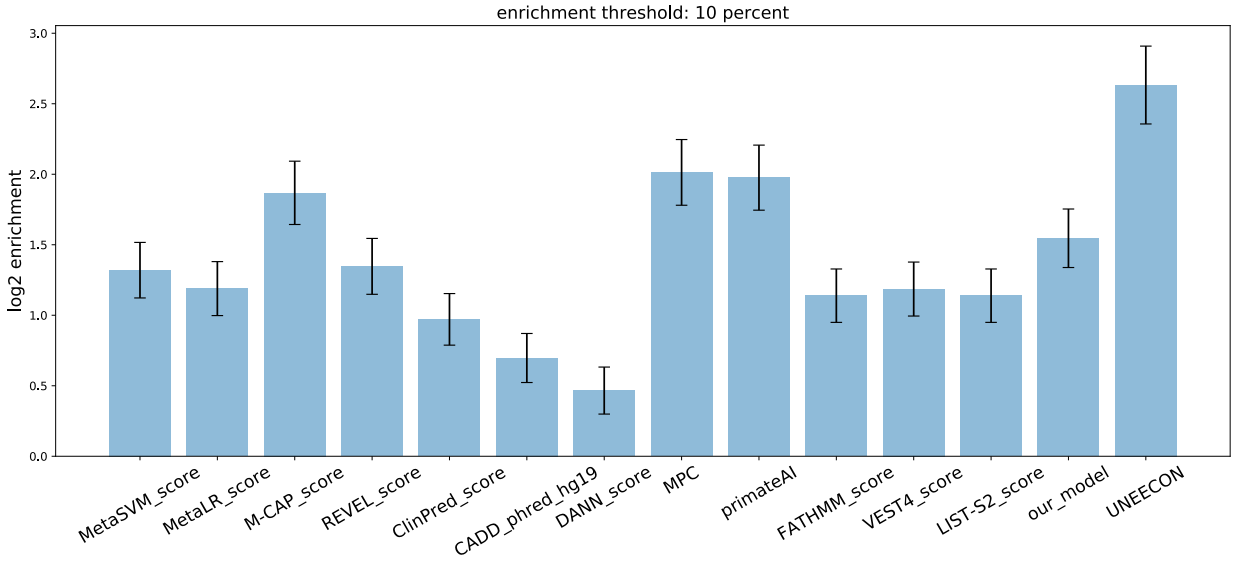


Figure 3-6: Enrichment of top 10% deleterious *de novo* mutations predicted by each method in probands in the DDD data. The error bar shows the standard error of $\log_2 Odds\ Ratio$.

3.4 Characteristics of misclassified ClinVar variants

After evaluating the performance of our model in comparison with other methods, I further investigated if there are any different characteristics between the correctly classified variants and the misclassified ones. By knowing why variants are incorrectly classified, we may prevent these problems and further improve variant prioritization.

3.4.1 Review status

The number of submitters with supporting assertion is a metric of the quality of pathogenic variants in the ClinVar database. It is represented as a gold stars level from none to four in ClinVar (Table 3-3). A higher gold stars level indicates a more reliable annotation of pathogenicity. However, the majority of variants used in our model only contained zero to two gold stars levels. Thus, in the following analysis, I only included variants with multiple submitters (two gold stars), single submitter (one gold star), or no assertion criteria provided (zero gold star).

Review Stats	Gold Stars Level
Multiple submitters	Two
Single submitter	One
No assertion criteria provided	None
Conflicting interpretations, No interpretations, Review by expert Practice guideline	Not included in our model

Table 3-2: The review status and the gold stars level reported in ClinVar. The gold stars level is positively related to the validation a variant has. However, based on our data's majority categories, I only kept variants in the upper half of this table in downstream analysis.

In the testing set of our model that consisted of ~6000 variants, the proportions of review status in correctly and misclassified variants are shown in Figure 3-7. The number of variants in each category of the contingency table is shown in Supplement A.2.4. I found that true positive and false negative have a similar distribution of review status; likewise, true negative and false positive have a similar distribution of review status, indicating that variants with no assertion criteria are more likely to be pathogenic.

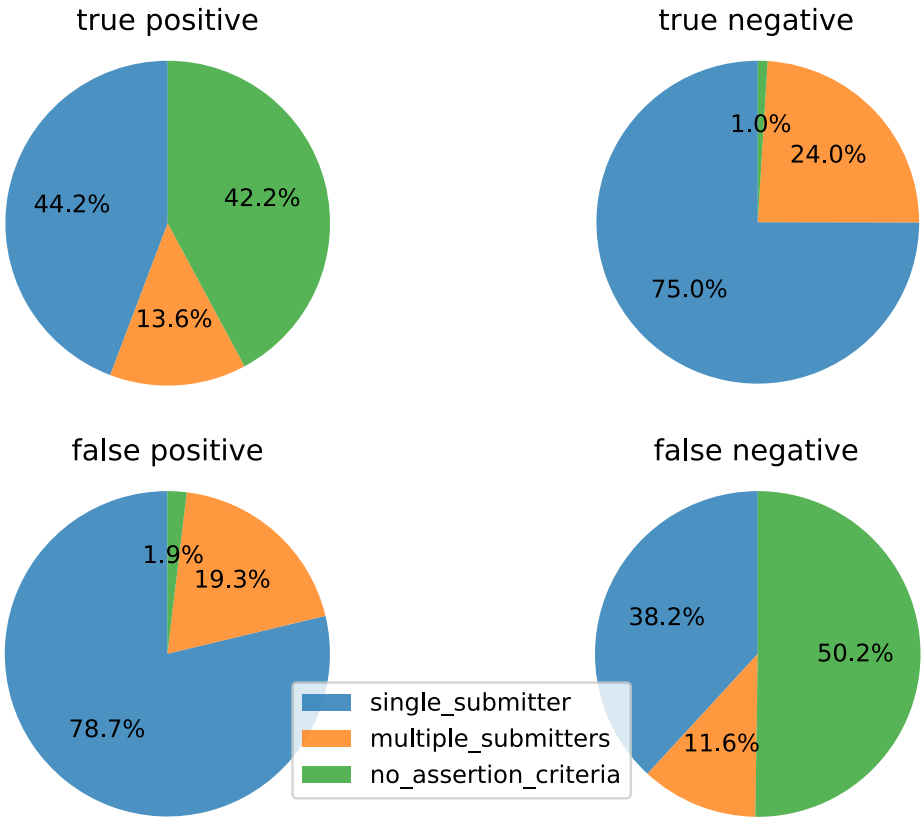


Figure 3-7: The reported ClinVar review status of variants in the testing data. True positives report the pathogenetic variants that are correctly predicted as pathogenic; while false positives are the benign variants that are wrongfully predicted as pathogenic. Likewise, true negatives are the benign variants that are also predicted as benign, and the false negative ones are the pathogenetic variants predicted as benign.

To test whether the model will have better performance if there are only variants with higher assertion criteria, I further removed all the variants with no assertion criteria in both the training and validation set of our model, then re-tunned and re-trained the model. I compared its performance with the model before removing these variants. Our model is here served as a baseline (Figure 3-8 (A)). Note that we lost ~20% of variants in both training data and validation data when removing these variants. Although

we removed the variants that accounted for half of the false-negative cases, we also removed the variants that contributed to forty percent of the true positive cases at the same time (Figure 3-7). Hence, we did not see an improvement in the model after removing these variants, with a p-value of the DeLong test equals to 0.004. Likewise, when we used the identical models to predict the testing data that do not contain variants with no assertion criteria, we did not see a superiority of the variant-removed model, with a p-value of 0.003. Undoubtedly, both models before and after removing variants without assertion criteria have extraordinary performance. However, our model still performed better than the variants-removed model in both tests. To conclude, removing the variants with no assertion criteria is not an effective way to improve our model.

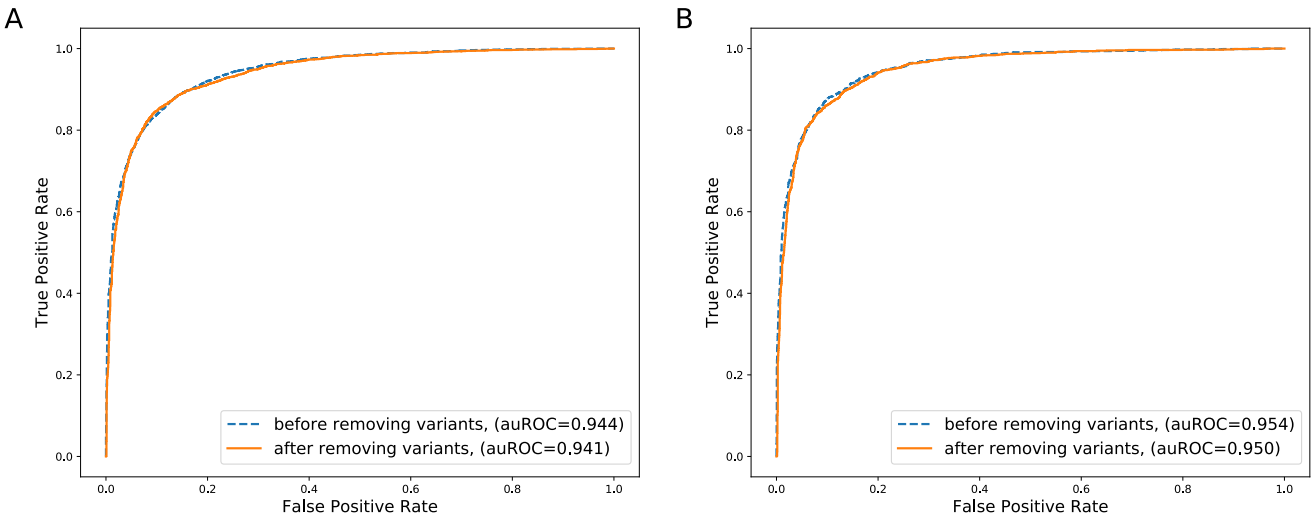


Figure 3-8: The performance of our model before and after removing variants with no assertion criteria. The removed variants are those without assertion criteria in ClinVar. The models are all trained by gradient boosting. (A) Our model without any modification is shown in a blue dashed line as a baseline. In the variants-removed model, I removed the variants without assertion criteria in its training set and validation set but kept the testing data the same as our model. The variants-removed model was later re-tunned. (B) The performance of the same models in (A) when testing data without assertion criteria were also removed.

3.4.2 Protein structure characteristics of misclassified variants

It has been previously reported that missense variants located in the hydrophobic core are more likely to be pathogenic than those that occurred on the protein surface (David *et al.*, 2012; David and Sternberg, 2015). I further examined whether the residue of the misclassified variants tend to be located on the surface or in the hydrophobic core of a protein tertiary structure. In this test, I utilized the RASE, which is the predicted probability of the residue being exposed in proteins (Figure 3-9).

In this analysis, we used Mann Whitney U test to test the difference in the median of RASE (Mann and Whitney, 1947). The difference in median between true positive and false negative, and that between true negative and false positives, are statistically significant under the significance level of 0.01 (Figure 3-9). According to this result, our model could not well classify the pathogenic variants that are exposed on the surface. On the other hand, the benign variants located in the hydrophobic core are less likely to be predicted correctly. Notice that the correctly predicted true positive and true negative cases have a significantly different median, indicating that the feature RSAE works great in our model. This result is further verified by performing the same test on RSAB, a feature opposite to RSAE that can predict the possibility of residue being buried in the hydrophobic core (Supplement A.1.6).

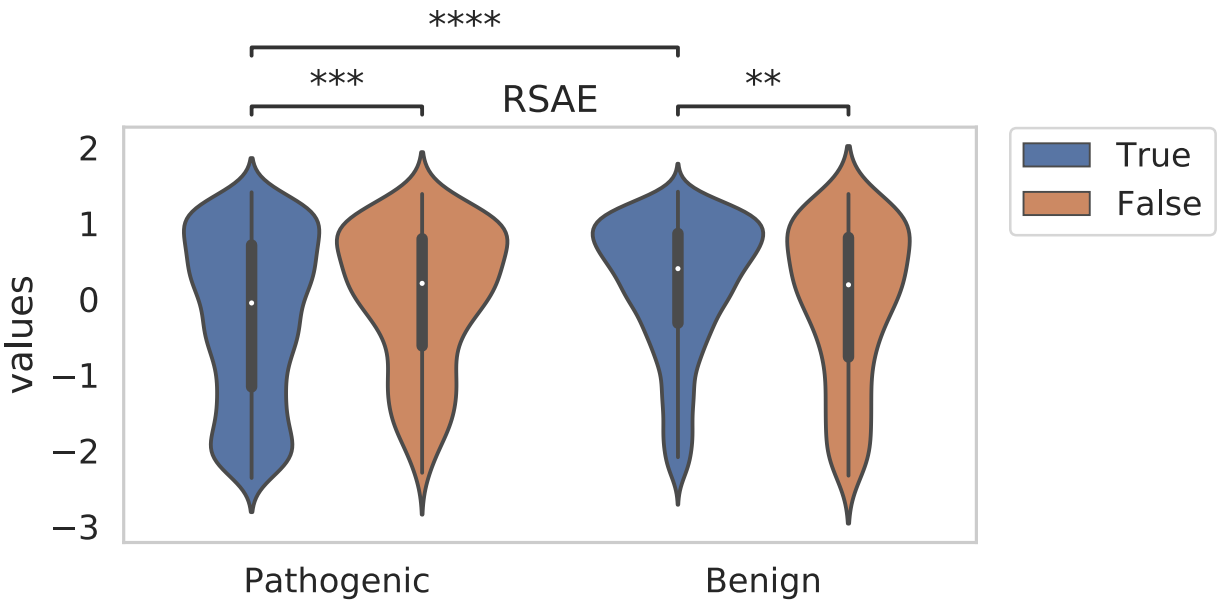


Figure 3-9: The distribution of RASE in different categories in the contingency table. The violin plot shows the distribution of four categories of data: true positive, false negative, true negative, and false positive; The first two categories are pathogenic, and the latter two are benign. The p-values are calculated with Mann Whitney U test. The significance code are as followed:
*: 10^{-2}< p <= 5×10^{-2}; **: 10^{-3}< p <= 10^{-2}; ***: 10^{-4} < p <= 10^{-3}; ****: p <= 10^{-4}.

Chapter 4

Discussion and Conclusion

Here, I hypothesized that pathogenic variants with different inheritance modes may have distinct characteristics and training separate models for each mode of inheritance may improve the prioritization of pathogenic variants. However, I did not observe a remarkable improvement of the model separately trained on dominant variants or recessive variants, compared to the model trained on variants without labels of inheritance mode. There are two possible explanations for this result. First, the features I used in this study may not well capture the distinct characteristics of dominant and recessive variants. In the comparison between the logistic regression model trained on dominant variants and the one trained on recessive variants, I observed that most of the features did not show different regression coefficients between the dominant and recessive models (Figure 3-3). Second, the number of variants with known mode of inheritance is much smaller than the ones without labels of inheritance mode, which may limit the accuracy of the inheritance-mode-aware model. Typically, having more data is beneficial for training machine learning models. In this thesis, the model's training set with unknown mode of inheritance is thirty times larger than those with modes of inheritance labeled (Table 2-1). However, it is possible that adding more variants with known mode of inheritance to the training set may further improve model performance to reach a similar or greater performance than the model trained on variants with unknown inheritance modes.

My dominant variants-trained models often have a relatively better performance compared to recessive variants-trained models (Figure 3-1, 3-2). This discrepancy may be due to the following two reasons. First, the features used in this thesis have better power in predicting dominant pathogenic variants. For instance, UNEECON score has better performance in predicting dominant variants than predicting recessive variants (Figure 3-4, (Huang, 2020)). Second, the contradicted labeling of modes of inheritance

in ClinVar variants and how we define inheritance modes. Since we treated all the variants with contradicted labels as dominant, this might cause the dominant variants to have some recessive traits. In contrast, recessive variants have none of the dominant information.

In comparing model performance, I observed a significant discrepancy in model performance between the testing data based on ClinVar and the ones based on *de novo* variants from the DDD project (Figure 3-5, Figure 3-6). Because *de novo* mutations are less likely to suffer from ascertainment bias compared with ClinVar variants, it may be more objective to evaluate the performance of variant prioritization methods using *de novo* mutation as testing data. Our model did not show superior performance in the testing data of *de novo* variants (Figure 3-6), which may be due to that this model was trained on variants with unknown mode of inheritance.

I also evaluated how UNEECON, an evolution-based method for integrating gene-level constraints and variant-level features, can predict de novo mutations associated with severe developmental disorders because it was the one with the highest predictive power when tested independently in the gradient boosting model (Figure 3-4). In the testing data of *de novo* mutations, UNEECON score itself had better predictive power than our gradient boosting model (Figure 3-6). Interestingly, it shows opposite trend when tested with ClinVar variants(Figure 3-4). This again shows the boarder range of application in variants with different inheritance modes of our model and the powerfulness of UNEECON is to predict *de novo* mutations associated with severe developmental disorders.

When evaluating the relationship between the number of submitters and how the variants are correctly or wrongfully classified, we found that variants with no assertion criteria account for ~ 21% of our training data. Removing these large amounts of variants will not improve model's performance, although these variants also occupied half of the false positive cases (Figure 3-7, Figure 3-8).

In this thesis, I also found that the predicted residue exposure level is a powerful predictor of misclassified pathogenic variants even though it is difficult to predict protein tertiary structures directly

from sequences (Figure 3-8, Supplement A.1.6). However, I still believe that adding a feature that can more precisely predict protein structure features, such as AlphaFold 2 (Senior *et al.*, 2020), may bring a substantial rise in variant prioritization performance.

In conclusion, I evaluated multiple novel strategies for predicting pathogenic variants and examined the characteristics of misclassified variants. The results of this thesis will form a foundation for the future development of disease-specific methods for variant interpretation.

Reference

Adzhubei, I. *et al.* (2010) 'A method and server for predicting damaging missense mutations', *Nat Methods*, 7.

Alirezaie, N. *et al.* (2018) 'ClinPred: Prediction Tool to Identify Disease-Relevant Nonsynonymous Single-Nucleotide Variants', *American Journal of Human Genetics*. doi: 10.1016/j.ajhg.2018.08.005.

Carter, H. *et al.* (2013) 'Identifying Mendelian disease genes with the variant effect scoring tool.', *BMC genomics*. doi: 10.1186/1471-2164-14-s3-s3.

Choi, Y. *et al.* (2012) 'Predicting the Functional Effect of Amino Acid Substitutions and Indels', *PLoS ONE*. doi: 10.1371/journal.pone.0046688.

Cortes, C. and Vapnik, V. (1995) 'Support-Vector Networks', *Machine Learning*. doi: 10.1023/A:1022627411411.

David, A. *et al.* (2012) 'Protein-protein interaction sites are hot spots for disease-associated nonsynonymous SNPs', *Human Mutation*. doi: 10.1002/humu.21656.

David, A. and Sternberg, M. J. E. (2015) 'The Contribution of Missense Mutations in Core and Rim Residues of Protein-Protein Interfaces to Human Disease', *Journal of Molecular Biology*. doi: 10.1016/j.jmb.2015.07.004.

Dong, C. *et al.* (2015) 'Comparison and integration of deleteriousness prediction methods for nonsynonymous SNVs in whole exome sequencing studies', *Human Molecular Genetics*. doi: 10.1093/hmg/ddu733.

Dunham, I. *et al.* (2012) 'An integrated encyclopedia of DNA elements in the human genome', *Nature*. doi: 10.1038/nature11247.

Friedman, J. H. (2001) 'Greedy function approximation: A gradient boosting machine', *Annals of Statistics*. doi: 10.1214/aos/1013203451.

González-Pérez, A. and López-Bigas, N. (2011) 'Improving the assessment of the outcome of nonsynonymous SNVs with a consensus deleteriousness score, Condel', *American Journal of Human Genetics*. doi: 10.1016/j.ajhg.2011.03.004.

Grantham, R. (1974) 'Amino acid difference formula to help explain protein evolution', *Science*. doi: 10.1126/science.185.4154.862.

Grimm, D. G. *et al.* (2015) 'The evaluation of tools used to predict the impact of missense variants is hindered by two types of circularity', *Human Mutation*. John Wiley and Sons Inc., 36(5), pp. 513–523. doi: 10.1002/humu.22768.

Gulko, B. *et al.* (2015) 'A method for calculating probabilities of fitness consequences for point mutations across the human genome', *Nature Genetics*. Nature Publishing Group, 47(3), pp. 276–283. doi: 10.1038/ng.3196.

Hindorff, L. A. *et al.* (2009) 'Potential etiologic and functional implications of genome-wide association loci for human diseases and traits', *Proceedings of the National Academy of Sciences of the United States of America*. doi: 10.1073/pnas.0903103106.

Huang, Y. F. (2020) 'Unified inference of missense variant effects and gene constraints in the human genome', *PLoS Genetics*. Public Library of Science, 16(7), p. e1008922. doi: 10.1371/journal.pgen.1008922.

Ioannidis, N. M. *et al.* (2016) 'REVEL: An Ensemble Method for Predicting the Pathogenicity of Rare Missense Variants', *American Journal of Human Genetics*. doi: 10.1016/j.ajhg.2016.08.016.

Ionita-Laza, I. *et al.* (2016) 'A spectral approach integrating functional genomic annotations for coding and noncoding variants', *Nature Genetics*. doi: 10.1038/ng.3477.

Kelley, D. R., Snoek, J. and Rinn, J. L. (2016) 'Basset: Learning the regulatory code of the accessible genome with deep convolutional neural networks', *Genome Research*. doi: 10.1101/gr.200535.115.

Kircher, M. *et al.* (2014) 'A general framework for estimating the relative pathogenicity of human genetic

variants', *Nature Genetics*. Nature Publishing Group, 46(3), pp. 310–315. doi: 10.1038/ng.2892.

Liaw, A. and Wiener, M. (2002) 'Classification and Regression by randomForest', *R News*.

Liu, X. *et al.* (2020) 'dbNSFP v4: a comprehensive database of transcript-specific functional predictions and annotations for human nonsynonymous and splice-site SNVs', *Genome Medicine*. doi: 10.1186/s13073-020-00803-9.

Liu, X., Jian, X. and Boerwinkle, E. (2013) 'dbNSFP v2.0: A database of human non-synonymous SNVs and their functional predictions and annotations', *Human Mutation*, 34(9). doi: 10.1002/humu.22376.

Malhis, N. *et al.* (2020) 'LIST-S2: taxonomy based sorting of deleterious missense mutations across species', *Nucleic acids research*. doi: 10.1093/nar/gkaa288.

Mann, H. B. and Whitney, D. R. (1947) 'On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other', *The Annals of Mathematical Statistics*. doi: 10.1214/aoms/1177730491.

Ng, P. C. and Henikoff, S. (2003) 'SIFT: Predicting amino acid changes that affect protein function', *Nucleic Acids Research*. doi: 10.1093/nar/gkg509.

Pollard, K. S. *et al.* (2010) 'Detection of nonneutral substitution rates on mammalian phylogenies', *Genome Research*. doi: 10.1101/gr.097857.109.

Quang, D., Chen, Y. and Xie, X. (2015) 'DANN: A deep learning approach for annotating the pathogenicity of genetic variants', *Bioinformatics*. doi: 10.1093/bioinformatics/btu703.

Rentzsch, P. *et al.* (2019) 'CADD: Predicting the deleteriousness of variants throughout the human genome', *Nucleic Acids Research*. doi: 10.1093/nar/gky1016.

Roadmap Epigenomics Consortium *et al.* (2015) 'Integrative analysis of 111 reference human epigenomes', *Nature*. doi: 10.1038/nature14248.

Samocha, K. E. *et al.* (2017) 'Regional missense constraint improves variant deleteriousness prediction', *bioRxiv*. doi: 10.1101/148353.

Senior, A. W. *et al.* (2020) 'Improved protein structure prediction using potentials from deep learning', *Nature*. doi: 10.1038/s41586-019-1923-7.

Shihab, H. A. *et al.* (2013) 'Predicting the Functional, Molecular, and Phenotypic Consequences of Amino Acid Substitutions using Hidden Markov Models', *Human Mutation*. doi: 10.1002/humu.22225.

Sundaram, L. *et al.* (2018) 'Predicting the clinical impact of human mutation with deep neural networks', *Nature Genetics*. Nature Publishing Group, 50(8), pp. 1161–1170. doi: 10.1038/s41588-018-0167-z.

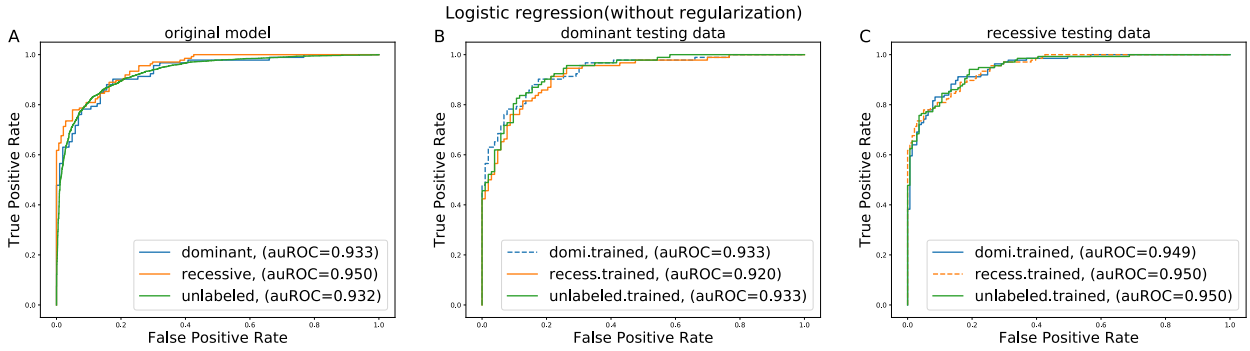
Turner, T. N. *et al.* (2017) ' denovo-db: a compendium of human de novo variants ', *Nucleic Acids Research*. doi: 10.1093/nar/gkw865.

Wong, W. C. *et al.* (2011) 'CHASM and SNVBox: Toolkit for detecting biologically important single nucleotide mutations in cancer', *Bioinformatics*. doi: 10.1093/bioinformatics/btr357.

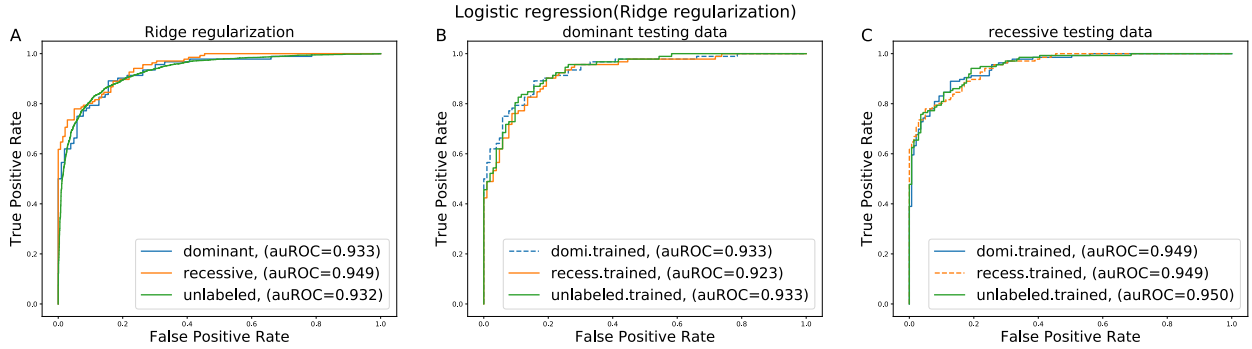
Zhou, J. and Troyanskaya, O. G. (2015) 'Predicting effects of noncoding variants with deep learning-based sequence model', *Nature Methods*. doi: 10.1038/nmeth.3547.

Appendix A

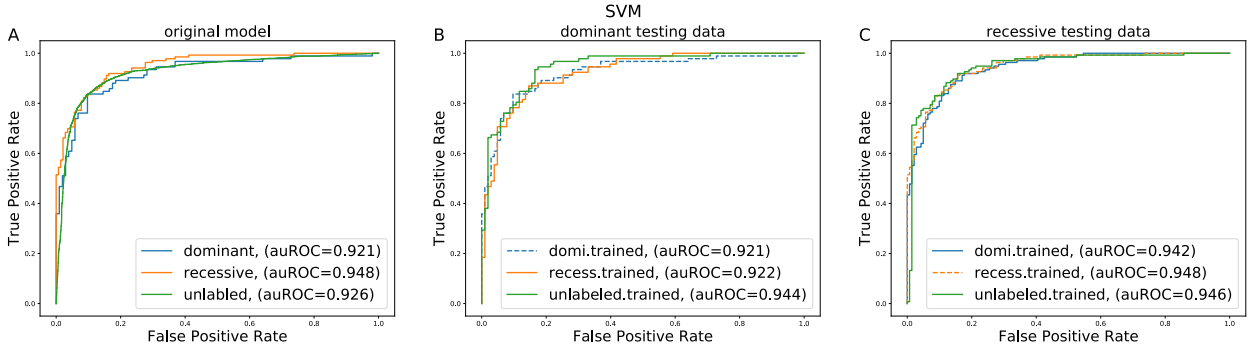
A.1 Supplementary figures



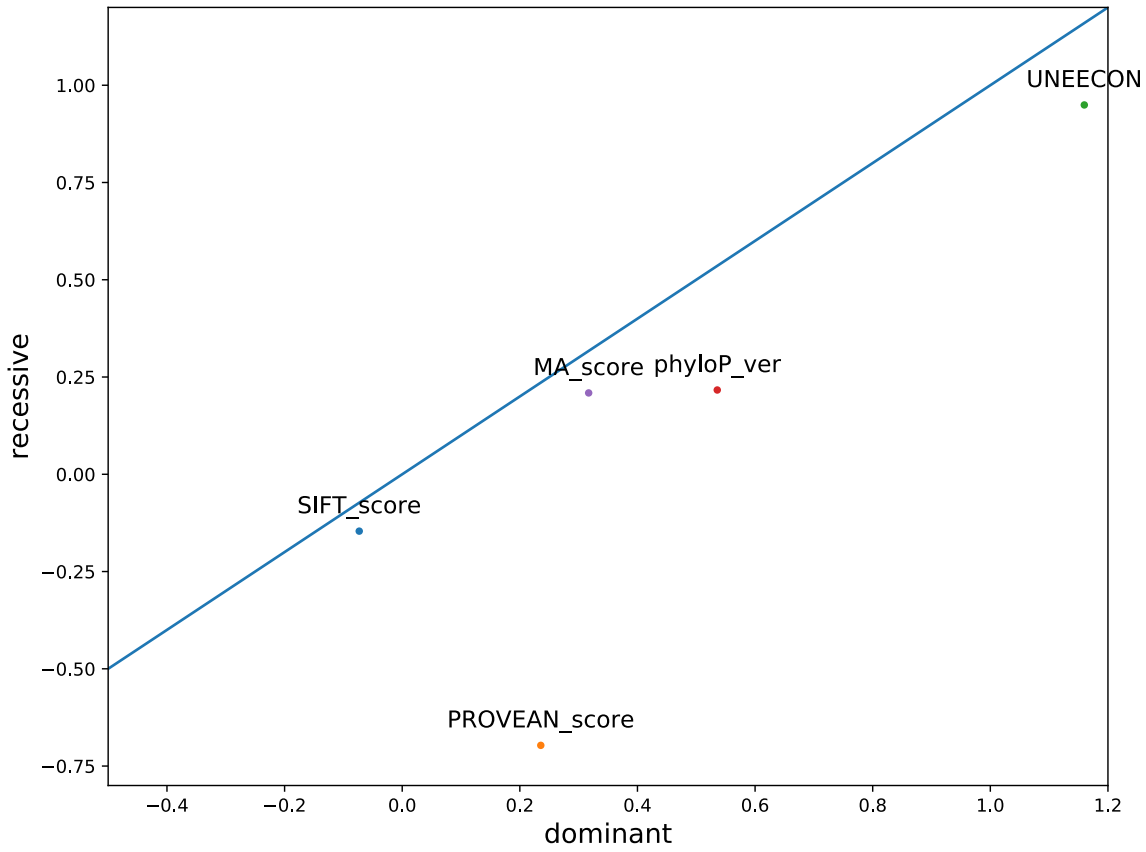
A.1.1: The ROC curve of logistic regression before performing ridge regularization.



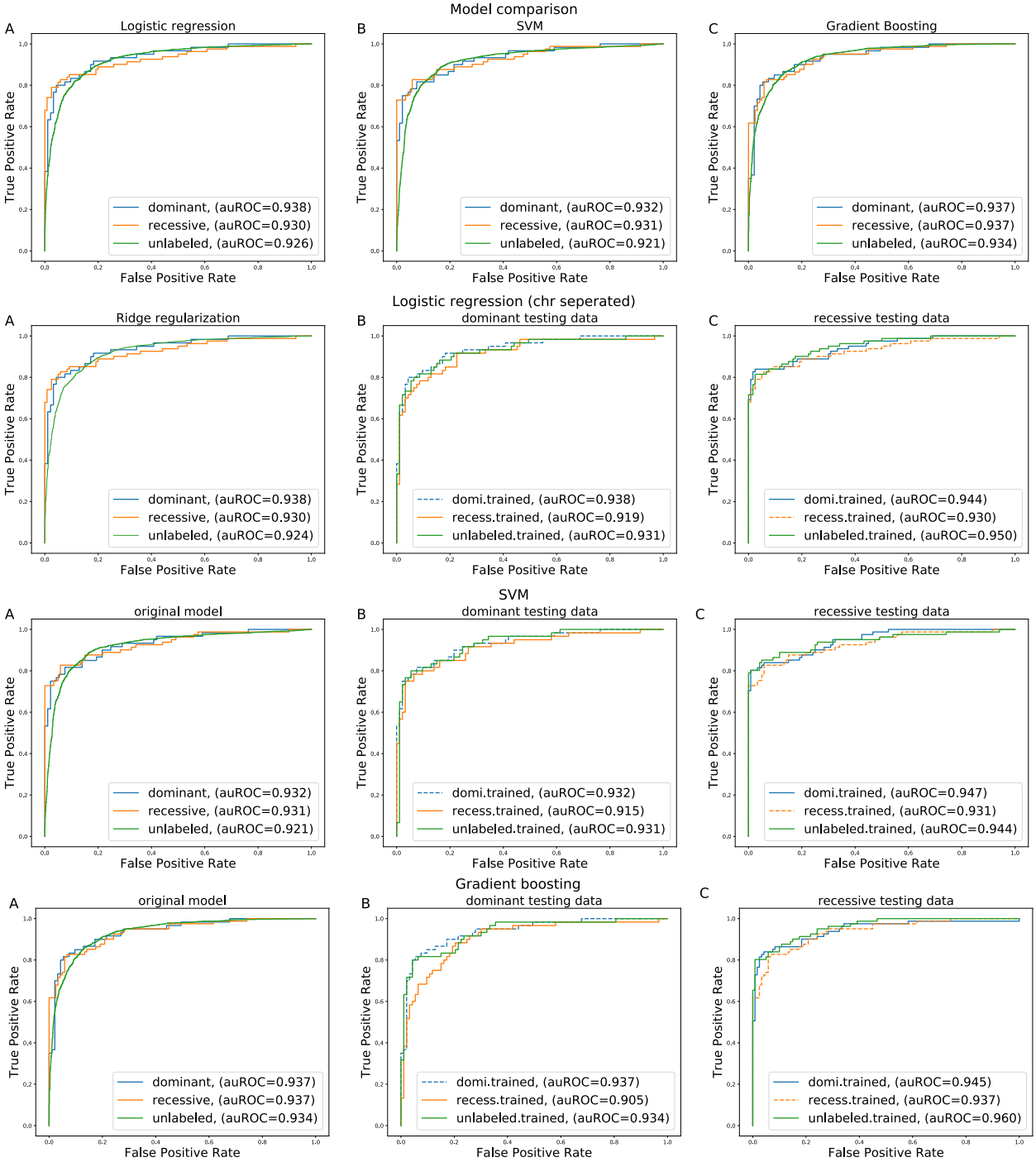
A.1.2: The ROC curve of logistic regression (after ridge regularization) and its comparison with model trained with different groups of inheritance.



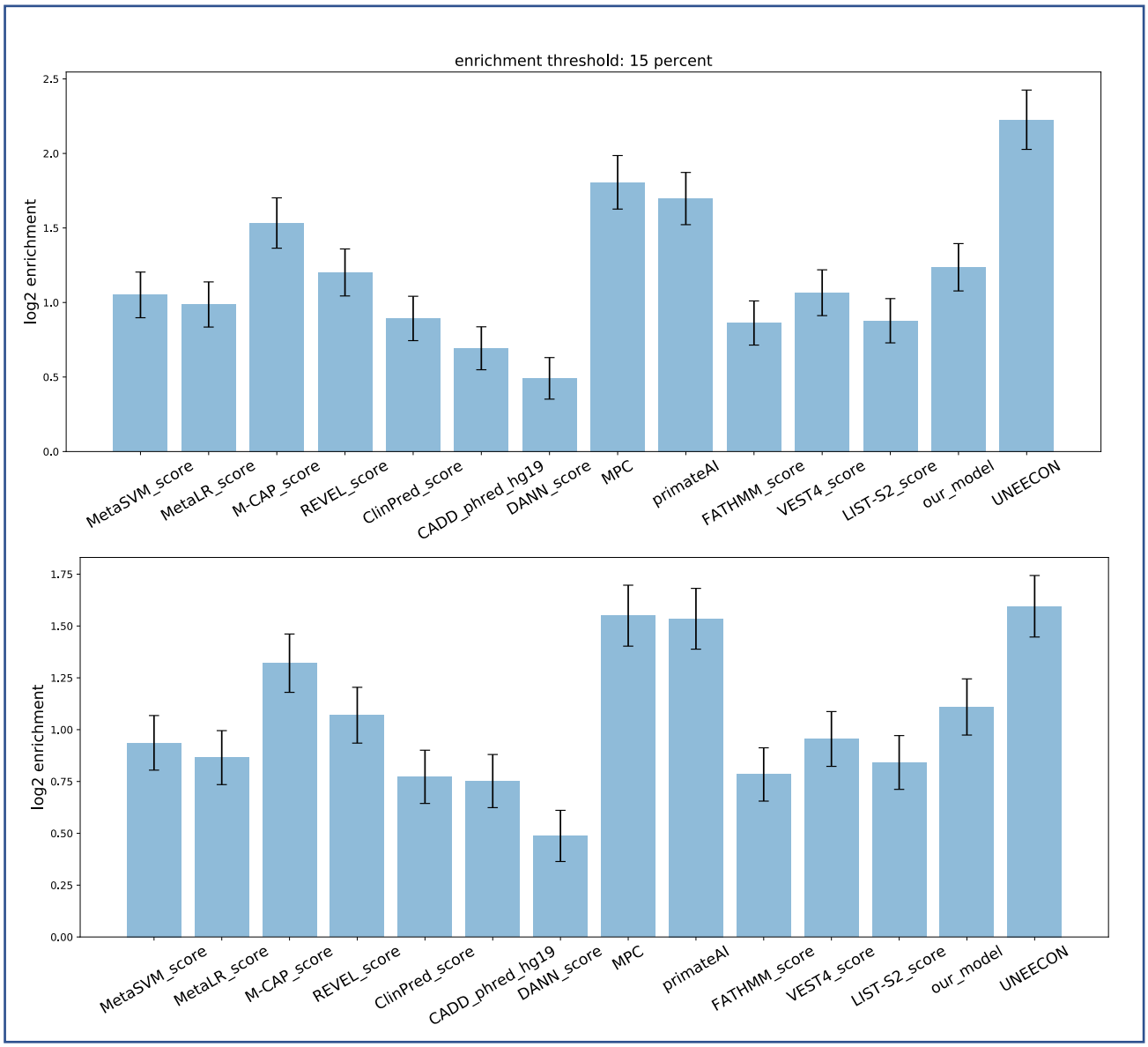
A.1.3: The ROC curve of SVM in testing model performance from model trained in other dominance group.



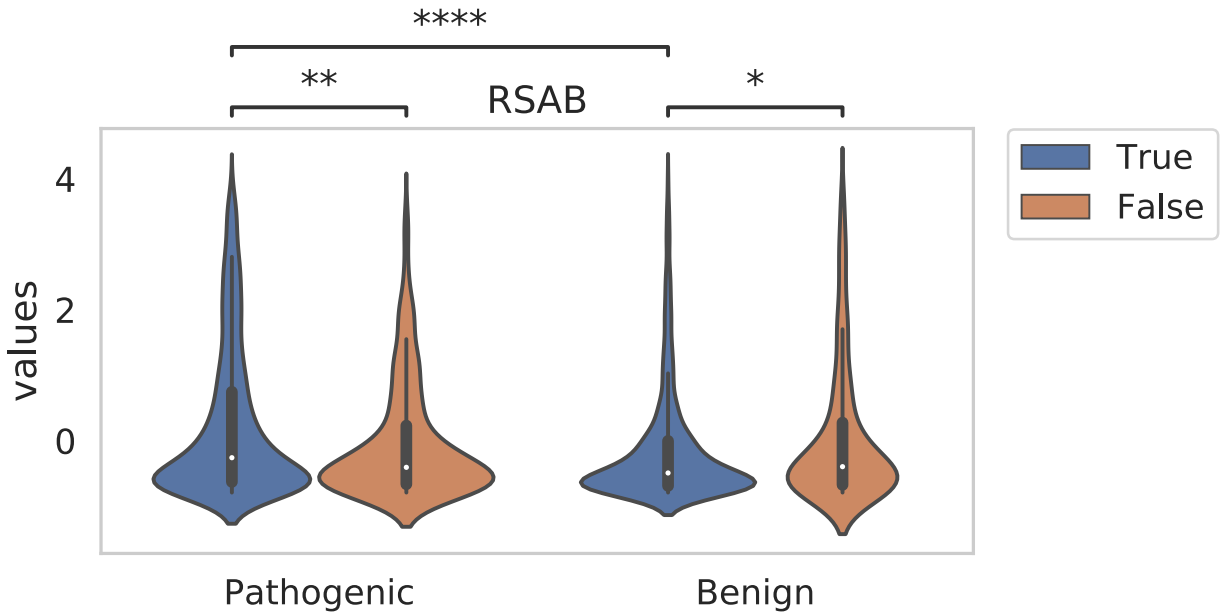
A.1.4: The coefficients of dominant and recessive-variants trained models in logistic regression. This figure only consists the features that have the top 5 individual predictive power in gradient boosting model.



A.1.5: The test using chromosome-separated models. This validates previous result from randomly separated data in 3.2 .



A.1.6: The enrichment test under the threshold of 15% and 20%. The error bar shows the standard error of odds ratio.



A.1.7 The distribution RASB of different categories in the contingency table.

RSAB is a feature that can predict the possibility of residue being buried, which is opposite to RSAE. The violin plot shows the distribution of four categories of data: true positive, false negative, true negative, and false positive; The first two categories are pathogenic, and the latter two are benign. The p-values are calculated with Mann Whitney U test. The significance code are as followed:

*: $10^{-2} < p <= 5 \times 10^{-2}$; **: $10^{-3} < p <= 10^{-2}$; ***: $10^{-4} < p <= 10^{-3}$; ****: $p <= 10^{-4}$.

A.2 Supplementary tables

Feature Name	Feature category	Reference
SIFT prediction	Sequence conservation	(Liu, Jian and Boerwinkle, 2013)
LRT prediction		(Liu, Jian and Boerwinkle, 2013)
MA prediction		(Liu, Jian and Boerwinkle, 2013)
PROVEAN prediction		(Liu, Jian and Boerwinkle, 2013)
SLR score		(Liu, Jian and Boerwinkle, 2013)
SIFT score		(Liu, Jian and Boerwinkle, 2013)
LRT omega		(Liu, Jian and Boerwinkle, 2013)
MA score		(Liu, Jian and Boerwinkle, 2013)
PROVEAN score		(Liu, Jian and Boerwinkle, 2013)
Grantham score		(Grantham, 1974)
HMM entropy		(Wong *et al.*, 2011)
HMM relative entropy		(Wong *et al.*, 2011)
dscore		(Liu, Jian and Boerwinkle, 2013)
Primate phyloP score		(Pollard *et al.*, 2010)
Mammalian phyloP score		(Pollard *et al.*, 2010)
Vetebrate phyloP score		(Pollard *et al.*, 2010)
PredRSAB	Protein Structure Prediction	(Wong *et al.*, 2011)
PredRSAI		(Wong *et al.*, 2011)
PredRSAE		(Wong *et al.*, 2011)
PredBFactorF		(Wong *et al.*, 2011)
PredBFactorM		(Wong *et al.*, 2011)
PredBFactorS		(Wong *et al.*, 2011)
PredStabilityH		(Wong *et al.*, 2011)
PredStabilityM		(Wong *et al.*, 2011)
PredStabilityL		(Wong *et al.*, 2011)
PredSSE		(Wong *et al.*, 2011)
PredSSH		(Wong *et al.*, 2011)
PredSSC		(Wong *et al.*, 2011)
Maximum RNA-seq signal	Regulatory Information	(Roadmap Epigenomics Consortium *et al.*, 2015)
UNEECON	Functional Information	(Huang, 2020)

A.2.1 Genomics features used in this model. The score of each features are extract from (Huang, 2020). Except for four predictive scores that are binary type (*e.g.,* SIFT prediction, LRT prediction, MA prediction, PROVEAN prediction), other features are all float type.

	Total variant number	Training data		Testing data (Chr1)		Validation data (Chr2)	
Dominant	1950	1539	78.9%	153	7.8%	258	13.2%
Recessive	2764	2302	83.2%	201	7.2%	261	9.4%
Unlabeled	62634	51464	82.3%	5391	8.6%	5779	9.2%

A.2.2: The number and ratio of training, testing and validation data in each inheritance-mode-aware that are separated by chromosome.

The mode of inheritance	Logistic Regression	SVM	Gradient Boosting
Dominant	Ridge regularization	Rbf kernel, C=10, Gamma = 0.01	Learning rate = 0.15, max_depth = 5, n_estimators = 150
Recessive		Rbf kernel, C = 1, Gamma = 0.1	Learning rate = 0.1, max_depth = 3, n_estimators = 200
Unlabeled		Rbf kernel, C = 1, gamma = 0.1	Learning rate = 0.05, max_depth = 8, n_estimators = 200

A.2.3: Tunned parameters of each inheritance-mode-aware model after 5-fold cross-validation.

True Positive	True Negative	False Positive	False Negative
2651	2805	374	434

A.2.4: The number of variants in the four different categories of our model's contingency table. .

Appendix B

Code for Feature Extraction and Statistical Analysis

All codes are written in python 3.6.3 unless otherwise specified.

B.1 Using Tabix to add UNEECON score in our dataset.

```
!cat unannotated_omit.tsv |cut -f 1| grep -|tr -d 'chr'|cut -d '-' -f 1,2,3|tr '-
' '\t' > location_omit.bed
!sortBed -i location_omit.bed |mergeBed > location_omit_sorted.bed
!tabix ~/work/project/original_data/UNEECON.tsv.gz -
R location_omit_sorted.bed > tabix_omit.tsv
```

B.2 Processed features that have multiple scores in single variants.

FATHMM_score, VEST4_score, MPC, and LIST-S2_score are the four features that have more than one scores at one variant. While other than FATHMM_score, the scores of three other features are positive correlated to the pathogenicity, so we only leave the maximum score. For FATHMM_score, since it has a negative correlation to clinical impact, we first took the minimum then reverse its sign.

```
## FATHMM is a special case


FATHMM_score =[]
for i in omit_model['FATHMM_score'].str.split(';')[:]:
    i = list(filter(lambda a: a != '.', i))
    ## to prevent there are NAs in the data
    if len(i) == 0:
        FATHMM_score.append('.')

    ## reversing thier sign
    else:
        FATHMM_score.append(float(min(i))*(-1))
omit_model['FATHMM_score'] = np.array(FATHMM_score)

def savesemicol(df, col):
    col_max = []
    for i in df[col].str.split(';') :
        col_max.append(max(i))

    df[col] = np.array(col_max)
    return(df)

```

```
def save_allsemicol(df, columnname):
    for n in columnname:
        df = savesemicol(df, n)
    return(df)

omit_model_all = save_allsemicol(omit_model,['VEST4_score','MPC', 'LIST-S2_score'])
```

B.3 Creating *de novo* mutation data frame

```
## imported required packages
import pandas as pd
import numpy as np
from sklearn.metrics import roc_auc_score, auc, roc_curve
from sklearn.model_selection import cross_val_score, train_test_split, StratifiedKFold,
 GridSearchCV
from sklearn.metrics import classification_report
from sklearn.ensemble import GradientBoostingClassifier
import matplotlib.pyplot as plt
import math
import scipy.stats as stats
from test_function import *

f = open("/storage/home/jkl5991/group/dbnsfp/tabix_try/tabixindex.txt")
header = f.readline().strip().split('\t')
f.close()

omit_model = pd.read_csv("/storage/home/jkl5991/work/project/not_conflict/all_model_new
.tsv", sep = '\t')

omit = pd.read_csv('unannotated_omit_std.tsv', sep = '\t')
dominant = pd.read_csv('dominant_std.tsv', sep = '\t')
recessive = pd.read_csv("recessive_std.tsv",sep = '\t')

# the features we used in our original study
original_column = ['SIFT_pred','LRT_pred', 'MA_pred', 'PROVEN_pred', 'SLR_score', 'SIFT
_score','LRT_omega',
                   'MA_score', 'PROVEN_score', 'Grantham', 'HMMEntropy','HMMRelEntropy', '
PredRSAB', 'PredRSAI',
                   'PredRSAE','PredBFactorF', 'PredBFactorM', 'PredBFactorS', 'PredStabili
tyH','PredStabilityM',
                   'PredStabilityL', 'PredSSE', 'PredSSH','PredSSC', 'dscore', 'phyloP_pri
', 'phyloP_mam','phyloP_ver',
                      'RNA_seq','UNEECON']

# the features for comparison (to other supervised score)
features = [ 'MetaSVM_score', 'MetaLR_score', 'M-
CAP_score', 'REVEL_score','ClinPred_score','CADD_phred_hg19',
             'DANN_score','MPC','primateAI','FATHMM_score', 'VEST4_score', 'LIST-
S2_score']


```

```
allcolumn = original_column + features


# a function that create a "location" column for future combination
def createloc(df):
    df['location'] = df['chr'] + '-' + (df['pos']-1).map(str)+ '-
' + df['pos'].map(str)+ '-' + df['ref'].map(str)+ '-' + df['alt'].map(str)
    df = df.drop(columns = ['chr', 'pos', 'ref', 'alt'])
    return(df)


def merge_with_original_df(originaldf, newdf):
    newdf = createloc(newdf)
    result = pd.merge(originaldf, newdf, on=['location'], how = 'inner')
    return(result)

### build de novo dataframe

def find_denovo_pattern_new(readfile, find_pattern):
    denovo_column = ['SampleID', 'StudyName', 'PubmedID', 'NumProbands', 'NumControls',
 'SequenceType', 'PrimaryPhenotype',
                     'Validation', 'Chr', 'Position', 'Variant', 'rsID', 'DbsnpBuild' , 'An
cestralAllele', '1000GenomeCount',
                     'ExacFreq', 'EspAaFreq', 'EspEaFreq', 'Transcript', 'condingDnaSize',
'Gene', 'FunctionClass', 'cDnaVariant', 'ProteinVariant',
                     'Exon/Intron', 'PolyPhen(HDiv)', 'PolyPhen(Hvar)', 'SiftScore', 'CaddS
core', 'LofScore', 'LrtScore' ]
    denovo = pd.read_csv(readfile, sep = '\t', header = None, names = denovo_column, co
mment = '#')


    # filtering only 'PrimaryPhenotype' = control and find_pattern
    denovo = denovo[denovo['Variant'].str.len() == 3]
    denovo[np.logical_or(np.logical_and(denovo['PrimaryPhenotype'] == find_pattern,
                                            denovo['FunctionClass'] == 'missense'),
                                  denovo['PrimaryPhenotype'] == 'control')]
    denovo['origin'] = np.where(denovo['PrimaryPhenotype'] == find_pattern, 1,0)


    # only keep exon or whole genome sequencing data
    denovo = denovo[denovo['SequenceType'] != 'targeted']
    denovo = denovo[np.logical_or(denovo['PrimaryPhenotype'] == find_pattern, denovo['P
rimaryPhenotype'] == 'control')]


    denovo['Variant'] = denovo['Variant'].str.replace('>', '-')
    denovo = denovo.assign(location = 'chr' + denovo['Chr'].astype(str) + '-
' + (denovo['Position']-1).astype(str) +\
                       '-' + denovo['Position'].astype(str) +'-'+ denovo['Variant'])


    # remove all the unnecessary columns
    denovo = denovo.drop(['SampleID', 'StudyName', 'PubmedID', 'NumProbands', 'NumContr
ols',  'Chr', 'Position',
```

```
                            'Variant', 'rsID', 'DbsnpBuild' , 'AncestralAllele', '1000GenomeCount', 'Exon/Intron',   'ExacFreq', 'EspAaFreq', 'EspEaFreq', 'Transcript', 'condingDnaSize', 'Gene', 'cDnaVariant', 'ProteinVariant',
                        'PolyPhen(HDiv)', 'PolyPhen(Hvar)', 'SiftScore', 'CaddScore', 'LofScore', 'LrtScore'], axis=1)


    denovo = denovo.drop_duplicates()
    denovo_simple = denovo[['location','origin']].drop_duplicates()

    denovo_location = denovo['location'].drop_duplicates().to_frame()
    return(denovo, denovo_simple, denovo_location)

ssc = find_denovo_pattern_new('/storage/home/jkl5991/work/project/original_data/denovo-db.ssc-samples.variants.tsv.gz', 'autism')
#ssc[2].to_csv('/storage/home/jkl5991/group/dbnsfp/tabix_try/denovo_ssc_location.tsv', header = None, sep = '\t',index = False)

nonssc = find_denovo_pattern_new('/storage/home/jkl5991/work/project/original_data/denovo-db.non-ssc-samples.variants.tsv.gz', 'developmentalDisorder')
#nonssc[2].to_csv('/storage/home/jkl5991/group/dbnsfp/tabix_try/denovo_non_ssc_location.tsv', header = None, sep = '\t',index = False)
```

building tabix index

```
!cat dnovo_non-ssc_location.tsv |tr '-' '\t' > non_ssc_loc.bed
!sortBed -i non_ssc_loc.bed > non_ssc_sortedloc.bed

!cat denovo_ssc_location.tsv |tr '-' '\t' > ssc_loc.bed
!sortBed -i ssc_loc.bed > ssc_sortedloc.bed


!awk '{gsub("chr" , ""); print}' non_ssc_sortedloc.bed > non_ssc_loc4UNEECON.bed
!awk '{gsub("chr" , ""); print}' denovo_location_sorted.bed > denovo_location_sorted4UNEECON.bed


# in /storage/home/jkl5991/work/project/original_data
!tabix UNEECON_addchr.tsv.bgz -
R ~/group/dbnsfp/tabix_try/ssc_sortedloc.bed > tabix_ssc_uneecon.tsv
!tabix UNEECON_addchr.tsv.bgz -
R ~/group/dbnsfp/tabix_try/non_ssc_sortedloc.bed > tabix_non_ssc_uneecon.tsv

# in ~/group/dbnsfp/tabix_try
!tabix prediction_sorted.bed.bgz -R ssc_sortedloc.bed > tabix_ssc_original.tsv
!tabix prediction_sorted.bed.bgz -
R non_ssc_sortedloc.bed > tabix_non_ssc_original.tsv
!tabix output2.bed.bgz -R ssc_sortedloc.bed > tabix_ssc_othermodel.tsv
!tabix output2.bed.bgz -R non_ssc_sortedloc.bed > tabix_non_ssc_othermodel.tsv
```

further integrate all datasets together

```
# import uneecon & original model
```

```
2.
3.  column =['chr', 'pos-
    1','pos', 'ref','alt','accession_num', 'SIFT_pred', 'LRT_pred', 'MA_pred', 'PROVEAN_pre
    d',
4.                 'SLR_score', 'SIFT_score', 'LRT_omega', 'MA_score', 'PROVEN_score', 'Gra
    ntham', 'HMMEntropy',
5.                 'HMMRelEntropy', 'PredRSAB', 'PredRSAI', 'PredRSAE', 'PredBFactorF', 'Pr
    edBFactorM', 'PredBFactorS',
6.                 'PredStabilityH', 'PredStabilityM', 'PredStabilityL', 'PredSSE', 'PredSS
    H', 'PredSSC', 'dscore',
7.                 'phyloP_pri', 'phyloP_mam', 'phyloP_ver', 'RNA_seq']
8.
9.  nonssc_uneecon = pd.read_csv('/storage/home/jkl5991/group/dbnsfp/tabix_try/tabix_non_ss
    c_uneecon.tsv', sep = '\t', names = ['chr','pos','ref','alt','UNEECON'])
10. nonssc_original = pd.read_csv('/storage/home/jkl5991/group/dbnsfp/tabix_try/tabix_non_s
    sc_original.tsv', sep = '\t', names = column)
11. nonssc_othermodel = pd.read_csv('/storage/home/jkl5991/group/dbnsfp/tabix_try/tabix_non
    _ssc_othermodel.tsv', sep = '\t', names = header)
12.
13.
14. ssc_uneecon = pd.read_csv('/storage/home/jkl5991/group/dbnsfp/tabix_try/tabix_ssc_uneec
    on.tsv', sep = '\t', names = ['chr','pos','ref','alt','UNEECON'])
15. ssc_original = pd.read_csv('/storage/home/jkl5991/group/dbnsfp/tabix_try/tabix_ssc_orig
    inal.tsv', sep = '\t', names = column)
16. ssc_othermodel = pd.read_csv('/storage/home/jkl5991/group/dbnsfp/tabix_try/tabix_ssc_ot
    hermodel.tsv', sep = '\t', names = header)
17.
18. def denovo_process(original, uneecon, othermodel, df, semicol= ['LIST-
    S2_score', 'VEST4_score','MPC']):
19.
20.     uneecon = createloc(uneecon)
21.     original = original.drop_duplicates()
22.     original = createloc(original)
23.
24.     uneecon['location'] = 'chr' + uneecon['location']
25.
26.
27.     whole = pd.merge(original, uneecon , on = ['location'], how = 'inner')
28.
29.     othermodel = createloc(othermodel)
30.
31.     whole = pd.merge(whole, othermodel , on = ['location'], how = 'inner')
32.
33.     ## save FATHMM
34.     FATHMM_score =[]
35.     for i in whole['FATHMM_score'].str.split(';')[:]:
36.         i = list(filter(lambda a: a != '.', i))
37.         if len(i) == 0:
38.             FATHMM_score.append('.')
39.         else:
40.             FATHMM_score.append(float(min(i))*(-1))
41.     whole['FATHMM_score'] = np.array(FATHMM_score)
42.
43.     whole = save_allsemicol(whole, semicol)
44.
45.     # remove all the rows that contain '.' (empty)
```

```
    whole = whole.drop(columns = ['pos-1_x','pos-1_y','accession_num','integrated_fitCons_score','H1-hESC_fitCons_score'])
    for i in features:
        whole.drop(whole.loc[whole[i] == '.'].index, inplace= True)

    denovo = pd.merge(whole, df, on = ['location'])

    print(denovo.shape)
    return(denovo, whole)

nonssc_denovo = denovo_process(nonssc_original, nonssc_uneecon, nonssc_othermodel, nonssc[1])[0]
ssc_denovo = denovo_process(ssc_original, ssc_uneecon, ssc_othermodel, ssc[1])[0]
ssc_denovo_control = ssc_denovo[ssc_denovo['origin'] ==0]


denovo = nonssc_denovo.append(ssc_denovo_control) #4229,44
np.sum(denovo['origin'])  #3600 proband, 1300 (1183+117)control
```

B.4 Enrichment test

```
def enrichment(df, portion, features, log = True):
    num = int(df.shape[0]*portion*0.01)
    oddsratios = []
    errors = []
    for i in features:

        df[i] = pd.to_numeric(df[i])
        top = np.sum(df.nlargest(num,i)['result'])
        array1 = [top, num-top]

        tail = np.sum(df.nsmallest(df.shape[0]-num,i)['result'])
        array2 = [tail, df.shape[0]-num-tail]
        result = sm.stats.Table2x2(np.asarray([array1, array2])).oddsratio

        if(log == True):
            result = math.log2(result)

        oddsratios.append(result)


        # change of base
        error = math.sqrt(1/top + 1/(num-top) + 1/tail + 1/(df.shape[0]-num-tail))* math.log2(math.e)
        #error = math.sqrt(1/top + 1/(num-top) + 1/tail + 1/(df.shape[0]-num-tail))
        errors.append(error)

    # Build the plot
    fig, sub = plt.subplots(figsize=(15, 7))

    sub.bar(features, oddsratios,yerr=errors, align='center', alpha=0.5, ecolor='black', capsize=5)

    plt.ylabel("log2 enrichment of proband denovo data")

```

```
    plt.title("enrichment test, enrichment level = %s percent"%portion)

    plt.legend(features)
    plt.show()
```

B.5 Violin plots and Mann-Whitney test

```
## Mann-Whitney
from statannot import add_stat_annotation
def violin_box(df, predict , correct , col, col_name):
    TP = find_any_column(df, predict,  correct, 'true_positive',col)
    TN = find_any_column(df, predict,  correct, 'true_negative',col)
    FP = find_any_column(df, predict,  correct, 'false_positive',col)
    FN = find_any_column(df, predict,  correct, 'false_negative',col)


    # creating DataFrame for drawing violin plot
    df1 = pd.DataFrame(
    { 'values': TP,
     'types': 'True',
     'clinvar':'Pathogenic'
    })
    df2 = pd.DataFrame(
    { 'values': TN,
     'types': 'True',
     'clinvar':'Benign'
    })
    df3 = pd.DataFrame(
    { 'values': FP,
     'types': 'False',
     'clinvar':'Benign'
    })
    df4 = pd.DataFrame(
    { 'values': FN,
     'types': 'False',
     'clinvar':'Pathogenic'
    })

    df_violin = pd.concat([df1, df2, df3, df4])
    plt.rcParams['figure.figsize'] = [6,3]

    ax = sns.violinplot(x = 'clinvar', y = 'values',hue = 'types', data = df_violin,x_a
xis = False, inner = "box")
    ax.legend(loc='lower right',fontsize = 13,bbox_to_anchor = (1.3,0.7))


    sns.set_style("whitegrid")
    ax.grid(False)
    sns.set(font_scale=1.2)

    ## add stars for p-values
    test_results = add_stat_annotation(ax, data=df_violin, x='clinvar', y='values', hue
='types',
```

```
                                    box_pairs=[(('Pathogenic','True'),('Pathogenic','False')),
                                               (('Benign','True'),('Benign','False')),
                                               (('Pathogenic','True'),('Benign','True'))

                                              ],
                                    test='Mann-Whitney',text_format='star',
                                    loc='outside', verbose=2)
    ax.set(xlabel=None)
    ax.set_title(col_name)

    savename = 'figure/4_violin'+col_name+'.pdf'
    plt.savefig(savename,bbox_inches='tight')
```