University of Padova

Department of Mathematics

Master Thesis in Data Science

Computational Prediction of Clinical Phenotypes and Causal Variants in Neurodevelopmental Disorders: An Analysis of Genetic Variants Data

Supervisor
Professor Emanuela Leonardi
University of Padova

Master Candidate
Can Abdullah Camuz

Academic Year
2023-2024

To my mother and father, for their endless love.

Dum vita est spes est.

Abstract

Advancements in genomic sequencing technologies and bioinformatics tools have allowed the understanding of the genetic variations in organisms and their potential effects on phenotypes (traits). This study presents an approach, employing machine learning models for the automated prediction of patient phenotypes and the identification of genetic variants, with a specific focus on causative, likely causative and contributing variants in neurodevelopmental disorders. It also observes the connection between patient phenotypes and variants, categorizing individuals based on neurodevelopmental manifestations such as intellectual disability, autism, epilepsy, microcephaly, macrocephaly, hypotonia, and ataxia. The study utilized genetic variant data from 867 patients, building upon & combining the previous works in the field. Unlike manual variant filtering and classification, as commonly used in the field for similar purposes, the aim was to contribute to the development of an automated tool. This tool streamlines the variant classification process and enhances disease classification accuracy with a systematic and data-driven approach to variant interpretation. To validate the approach, the results were shared and compared with those of previous groups that participated in the CAGI Challenge (Critical Assessment of Genome Interpretation) in the years 2018 and 2021, addressing the same task. This analysis provides insights into the performance of the tool in comparison to manual approaches.

Contents

Abstract v

List of figures ix

List of tables xi

Listing of acronyms xii

1 Introduction 1
- 1.1 Introductory Overview 1
  - 1.1.1 BioComputing UP Group and CAGI Challenge Workflow 1
  - 1.1.2 Neurodevelopmental Disorders and Gene Panel 2
- 1.2 Thesis Outline 3

2 Literature Review and Related Work 5
- 2.1 CAGI5 Workflow and Findings 5
  - 2.1.1 Variant Filtering And Classification 6
  - 2.1.2 Phenotype Classification and Assignment 10
- 2.2 CAGI6 Workflow and Findings 13
  - 2.2.1 Variant Filtering And Classification 13
  - 2.2.2 Phenotype Classification and Assignment 15

3 Background Information 18
- 3.1 Neurodevelopmental Disorders (NDDs) 18
- 3.2 Genetic Mutations and Variations 20
- 3.3 Omics Data & Databases 23
- 3.4 Workflow and File Types in Variant Calling 24
- 3.5 Annotation 25

4 Data Acquisition & Retrieving 26
- 4.1 Raw Data Acquisition: Sequencing 26
- 4.2 Introduction to Data at Hand 27

5 Methodology And Workflow 35
- 5.1 Initial Data Analysis 35
- 5.2 Data Preparation & Cleaning 37

5.2.1 Handling Missing Values 37
5.2.2 Gene Filtering 39
5.2.3 Encoding 40
5.3 Feature Engineering & Extracting Features 41
5.3.1 Allele Frequencies In The Population 41
5.3.2 Information Related To Cohort And Quality 43
5.3.3 Genetic Associations With Diseases 43
5.3.4 Special Cohort Frequency of the Variants 44
5.3.5 Dropping Columns 44
5.4 Creating Data Structures 46
5.4.1 Variant Classification 46
5.4.2 Phenotype Classification 47

6 Experiments & Results 50
6.1 Variant Classification 50
6.1.1 Random Forests 51
6.1.2 Logistic Regression 60
6.1.3 Support Vector Machines 65
6.1.4 Neural Network 71
6.1.5 K-Means for Unsupervised Learning 76
6.1.6 Gaussian Mixture Models for Unsupervised Learning 85
6.2 Phenotype Classification 91
6.2.1 Predicted Variant Labels 93
6.2.2 True Variant Labels 96
6.2.3 Test Data Results 98

7 Conclusion 100

References 104

Acknowledgments 107

Listing of figures

1.1 CAGI Challenge Workflow 2

2.1 Variant Filtering Steps Used by BioComputing UP 7
2.2 Variant Labeling Workflow (Conducted Manually) 8
2.3 Predicted variants distribution among submissions of CAGI5 9
2.4 Proportional amount of variants, correctly classified by the groups of CAGI5 9
2.5 Proportional number of patients with the phenotype, correctly predicted by the groups of CAGI5 12
2.6 The number of patients with the phenotypes (True Labels of CAGI5). 12
2.7 Predicted variants distribution among groups of CAGI6 14
2.8 Proportional amount of variants, correctly classified by the groups of CAGI6 15
2.9 Proportional number of patients with the phenotype, correctly predicted by the groups of CAGI6 17

3.1 Chromosome representation of some structural variants 22
3.2 Different regions of DNA in the transcription site 23

4.1 Some of the columns of annotated file 30
4.2 Patients with labels and with diseases 33
4.3 Variant type and variant amount distribution 34

5.1 Variance values of the dataset columns 37
5.2 Amount of missing values within the features 38
5.3 Encoding procedures based on data types 40

6.1 ROC Curves and AUC Values of the Final Random Forest Model 60
6.2 ROC Curves and AUC Values of the Final Logistic Regression Model 66
6.3 ROC Curves and AUC Values of the Final SVM Model 71
6.4 Normalized Confusion Matrix For Improved Neural Network 75
6.5 ROC Curves and AUC Values of the Improved Neural Network Model . . . 75
6.6 Scree Plot for Cumulative Explained Variance During PCA Process 77
6.7 3D Scatter Plot of 4 Clusters With 3 Components 79
6.8 Distribution of Variant Types Among the Clusters 79
6.9 3D Scatter Plot of 2 Clusters With 3 Components 81
6.10 Distribution of Variant Types Among 2 Clusters 82
6.11 ARI Score Variation with Majority Class Sample Size in 2-Cluster Scenario . . 84

6.12 ARI Score Variation with Majority Class Sample Size in 4-Cluster Scenario . . 84
6.13 Distribution of Variants with Feature Selection (4 clusters) 87
6.14 Distribution of Variants with Feature Selection (2 clusters) 88
6.15 3D Visualization of 4 Clusters with t-SNE 90
6.16 Comparison of Test Accuracies of The Two Approaches 99

Listing of tables

6.1 Classification Report for Basic Random Forest 54
6.2 Classification Report for Random Forest After Grid Search 56
6.3 Classification Report for Random Forest with Selected Features and Undersampling 59
6.4 Classification Report for Basic Logistic Regression 63
6.5 Classification Report for Logistic Regression After Grid Search 64
6.6 Classification Report for Logistic Regression with Selected Features and Undersampling 66
6.7 Classification Report for Basic SVM 68
6.8 Classification Report for SVM After Grid Search 69
6.9 Classification Report for SVM with Selected Features and Undersampling . . 71
6.10 Classification Report for Initial Neural Network 73
6.11 Classification Report for Improved Neural Network 74
6.12 Clustering Results with 4 Clusters 80
6.13 Clustering Result With 2 Clusters 81
6.14 Initial GMM Clustering Results with 4 Clusters 86
6.15 Percentages of Cluster Content with Feature Selection (4 Clusters) 87
6.16 Percentages of Cluster Content with Feature Selection (4 Clusters) 88
6.17 Class Counts With Feature Selection 88
6.18 Cluster Percentages With Feature Selection 88
6.19 NN Structures and Performance Metrics with Predicted Labels 94
6.20 NN Structures and Performance Metrics with Predicted Labels and Resampling 95
6.21 NN Structures and Performance Metrics with True Labels 96
6.22 NN Structures and Performance Metrics with True Labels and Resampling . 97
6.23 Test Accuracies with Predicted and True Labels 99

Listing of acronyms

CAGI	Critical Assessment of Genome Interpretation
NDD	Neurodevelopmental Disorder
ID	Intellectual Disability
ASD	Autism Spectrum Disorder
SNV	Single Nucleotide Variant
SNP	Single Nucleotide Polymorphism
VCF	Variant Call Format
ROC	Receiver Operating Characteristic
AUC	Area Under the Curve
NGS	Next-Generation Sequencing
DNA	Deoxyribonucleic Acid
RNA	Ribonucleic Acid
PCA	Principal Component Analysis
DC	Disease Causing
LP	Likely Pathogenic
CF	Contributing Factor
MCC	Matthew Correlation Coefficient
SID	Submission Identity
UTR	Untranslated Region

1 Introduction

1.1 Introductory Overview

1.1.1 BioComputing UP Group and CAGI Challenge Workflow

Being the laboratory in which this thesis project and a curricular internship were conducted, BioComputing UP is a research laboratory and is part of the Department of Biomedical Sciences of the University of Padua. The focus of the research is using bioinformatics tools and computational methods to solve biological issues in the fields of structural and functional biology, large-scale and genome-wide analyses, genetic diseases, and cancer studies. The group regularly participates in international competitions like the Critical Assessment of Techniques for Protein Structure Prediction (CASP), the Critical Assessment of Genome Interpretation (CAGI), and the Critical Assessment of Function Annotation experiment (CAFA).

CAGI is an initiative and a consortium that organizes conferences, workshops and challenges with a main focus on evaluating and improving methods for predicting the phenotypic impacts of genetic variations and assessing the state-of-the-art computational methods for interpreting genomic information. Each year, the CAGI experiment involves a series of challenges where participants are provided with sets of genetic variants, and their task is to predict the phenotypic outcomes or functional consequences associated with these variants. The results provide valuable insights into the strengths and limitations of existing computational methods, contributing to the development of more accurate and reliable tools for genomic interpretation.

These challenges cover a range of topics, including the interpretation of single nucleotide variants (SNVs), insertions/deletions, and other types of genomic variations, which are explained in the following chapters. CAGI experiment participants are provided genetic variants and make predictions about resulting phenotypes. Independent assessors evaluate the predictions against experimentally characterized phenotypes. The challenges conclude with conferences and publications in a special journal issue. There have been six editions of CAGI experiments, held between 2010-2022, and the seventh round is planned to take place over the Summer of 2024. Participation is open to researchers, scientists, and teams from around the world who are working on developing computational methods for biological data. The evaluations provided by CAGI promote cooperation and improvement by comparing the effectiveness of different tools and methods.

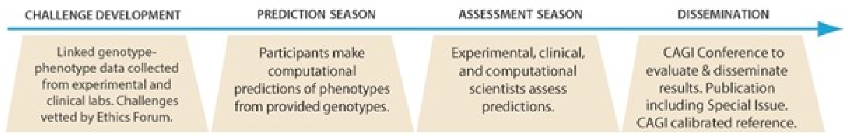

CHALLENGE DEVELOPMENT
PREDICTION SEASON
ASSESSMENT SEASON
DISSEMINATION
Linked genotype-phenotype data collected from experimental and clinical labs. Challenges vetted by Ethics Forum.
Participants make computational predictions of phenotypes from provided genotypes.
Experimental, clinical, and computational scientists assess predictions.
CAGI Conference to evaluate & disseminate results. Publication including Special Issue. CAGI calibrated reference.

Figure 1.1: CAGI Challenge Workflow

In this thesis, a machine learning approach for the CAGI Challenge was introduced by using the patient data that had been previously obtained by a clinical laboratory for the same purpose. The raw data of CAGI5 (2018) and CAGI6 (2021), from the challenge named "Intellectual Disability Panel" was employed to make useful interpretations and discover the relationships between genetic variants and Neurodevelopmental Disorders.

1.1.2 Neurodevelopmental Disorders and Gene Panel

As mentioned, NDDs exhibit clinical diversity, but it has been discovered that they share a significant genetic component, with numerous common disease genes identified, particularly in complex conditions such as ID and ASD [1]. In this study, a gene panel of 74 genes was used, each known to be associated with these NDDs of interest. This selection was based on the hypothesis that shared functional pathways contribute to the comorbidity observed among diverse NDDs [2]. The design of the 74-gene panel involved an innovative in silico approach, based on disease networks and mining data from public resources to establish a scoring system for disease-gene associations.

Despite significant advancements in genomics, it is still difficult to understand the effects of genetic variants associated with NDDs. One of the challenges is that NDDs are genetically heterogeneous diseases, meaning that they can be caused by variations in different genes. In other words, the disease does not have a single, specific genetic cause, but rather multiple genetic factors can contribute to its development. Also, the complex interactions between hereditary factors and the wide range of symptoms associated with these disorders make understanding and diagnosis very difficult, especially when depending only on clinical findings.

To bridge this gap, this study introduces an approach that benefits the power of machine learning models. Unlike traditional methods relying on manual interpretation and variant filtering, these models are designed to predict patient phenotypes and identify the causative (pathogenic), likely causative (likely pathogenic), and contributing genetic variants in NDDs. The relationship between genetic variations and the diverse phenotypic expressions was observed.

The primary objective of this study is to develop a systematic and data-driven methodology to provide a more accurate and efficient means of classifying variants, facilitating the traditionally manual process. Secondly, the study focuses on categorizing individuals based on traits such as ID, ASD, Epilepsy, Microcephaly, Macrocephaly, Hypotonia, and Ataxia specifically, which are the diseases that have proven to share some common variants and cellular pathways [3]. The approach utilizes genetic variant data from a cohort of 867 patients, building upon and analyzing insights from previous works of CAGI5 and CAGI6 in the field. The source and the means of obtaining the data are explained in the following chapters.

To validate the effectiveness of the methods, the results are compared with those of previous groups participating in the CAGI Challenge. This comparative analysis provides insight into the accuracy and efficiency of the automated tool by comparing its performance to traditional procedures.

1.2 Thesis Outline

This thesis report consists of 7 chapters and the structure is as follows:

- Chapter 1 gives a brief introduction to the project, the challenges of the study, the objectives, and the outline of the general structure of the thesis.
- Chapter 2 provides an insight into the previous work and research, reviews the relevant literature on CAGI Challenge and variant classification, and shares the results.
- Chapter 3 describes the concepts by giving some background information on biology and genetics, discusses the diseases of interest, data types used in this field, and ways of obtaining the data from biological databases.

- Chapter 4 explains the dataset and gives information about the sources and how the data was obtained.
- Chapter 5 includes the methodology, how data was transformed and prepared, what assumptions were made, and introduces the workflow.
- Chapter 6 reports the various experiments carried out, details the steps taken with specifications, and finally reports and discusses the results obtained from all the experiments and compares them.
- Chapter 7 summarizes the outcome, concludes the findings of the project, and provides suggestions for future research direction.

2
Literature Review and Related Work

In this chapter, a detailed description of the work and methodology for the previous CAGI challenges is given. The research outcomes and findings of the Biocomputing UP Laboratory have been shared, which was in the position of "Assessor" for the groups that participated in the challenge. At the end of each section, the evaluation metrics and comparisons are given.

2.1 CAGI5 Workflow and Findings

The CAGI-5 Intellectual Disability challenge asked to use computational methods to predict patient clinical phenotypes and the causal variant(s) based on an analysis of their gene panel sequence data in 2018. The Padua Genetics of Neurodevelopmental Disorders Lab at the Department of Woman and Child Health, the University of Padua (Padua NDD lab) has been using a gene panel to diagnose different NDD subtypes for the past couple of years [3]. Subsequently, they released a dataset of 150 pediatric patients, obtained from hospitals in Padua and other cities in Italy, for predictors to analyze. The study involved 4 groups submitting a total of 13 predictions for the challenge. Group 2 submitted 6 predictions, and groups 2 and 3 submitted three predictions each. After conducting its own experiments (ground truth), BioComputing UP Lab merged the results of variant analysis and the clinical traits to assess the findings of each predictor.

Firstly, phenotypes were derived from clinical notes, and candidate variants were validated through segregation analysis. A genetic approach called **segregation analysis** is used to examine how particular traits or variants are inherited within families. It helps researchers to distinguish whether a trait follows an autosomal dominant, autosomal recessive, X-linked, or other inheritance pattern. Once the segregation of candidate variants within families is verified, researchers can identify variants that are likely causative for a particular trait or disorder. This is essential for establishing a genetic basis for NDDs. For CAGI, that was a necessary step to verify the absence of variants in the parents according to the de novo paradigm, discovering whether the variants were inherited from affected parents or not.

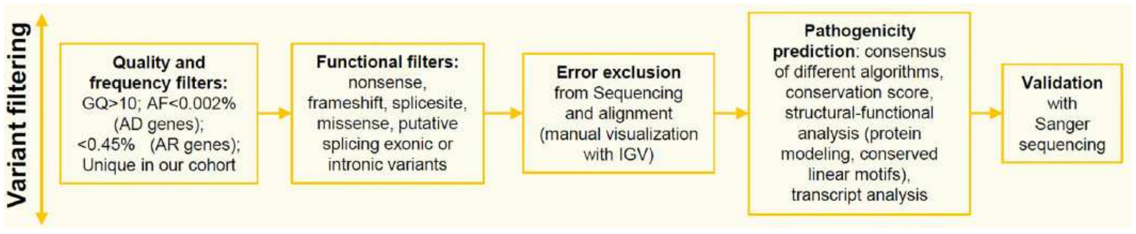
The **de novo paradigm** refers to the origin or occurrence of a genetic variant in an individual that is not present in their parents. De novo variants occur spontaneously in the germ cells (egg or sperm) or during early embryonic development. Investigating de novo variants is crucial since they can be associated with the development of genetic disorders. These variants can have a significant impact on an individual's health, especially in cases where the variant is pathogenic and contributes to the manifestation of a disorder.

The task reflects the complexity of the challenge, as clinical notes might be subjective and only a subset of genes had been screened instead of the whole human genome [2]. Furthermore, in contrast to other CAGI Challenges, patients may manifest more than one of these phenotypes, in different combinations. Overall, both studies highlight the realistic aspects of predicting clinical phenotypes and causative variants in NDDs using computational methods.

2.1.1 Variant Filtering And Classification

The methodology for variant filtering and classification involved several steps. Firstly, a pipeline was used to create a database of genetic variants identified in the cohort and annotate them with features provided by ANNOVAR, an annotation application that will be explained in the following chapters. The information retrieved with the annotation was allelic frequency in control cohorts, variant interpretation from InterVar automated, ClinVar report, pathogenicity predictions, and conservation scores. The detected variants were then ranked based on their frequency in public databases and the in-house database, and single nucleotide variants (SNVs) that were found more than twice in the cohort or with an allele frequency higher than expected for the disorder were excluded.

For the classification process, InterVar was used for the clinical assessment which is a bioinformatics tool interpretation of genetic variants. Following this, a manual review was conducted, and variants were categorized into five groups (pathogenic, likely pathogenic, uncertain significance, likely benign, benign) through a comprehensive evaluation of various evidence lines.



Variant filtering
Quality and frequency filters: GQ>10; AF<0.002% (AD genes); <0.45% (AR genes); Unique in our cohort
Functional filters: nonsense, frameshift, splicesite, missense, putative splicing exonic or intronic variants
Error exclusion from Sequencing and alignment (manual visualization with IGV)
Pathogenicity prediction: consensus of different algorithms, conservation score, structural-functional analysis (protein modeling, conserved linear motifs), transcript analysis
Validation with Sanger sequencing

Figure 2.1: Variant Filtering Steps Used by BioComputing UP

This assessment included considerations such as conservation, allele frequency in population databases, variant effect inferences, mode of inheritance, computational X-inactivation patterns, and disease segregation.

To evaluate the putative clinical impact of the variants, the following criteria were applied:

- allele frequency <0.002% in the Gnomad database, or <0.45% for variants in autosomal-recessive genes, as indicated by [4],
- absence of the variant in other samples (in-house database),
- stop gain, frameshift, and splicing variants were a priori considered to be most likely pathogenic,
- for missense mutations, amino acid conservation, and consensus of pathogenicity predictions were evaluated,
- inheritance mode,
- phenotypic consistency with the clinical signs associated with mutations in the same gene [2].

It is important to note, that for a diagnostic purpose, the thresholds used by the Padua NDD lab to filter candidate variants, have been calculated based on the assumption that the patient phenotype follows a Mendelian transmission [2]. The criteria used to classify the variants were reported for both causative and likely pathogenic variants, and all causative and likely pathogenic variants were submitted to the LOVD database.

Then, the predictors have been assessed for their ability to detect variants in patients. Among the submissions, Group 2 demonstrated the best performance, accurately predicting the highest number of variants associated with diverse patient phenotypes (37 out of 56). Also, it excelled in predicting causative (16 out of 25), putative causative (12 out of 18), and contributing factor (9 out of 13) variants, outperforming other groups in each category. The second-highest

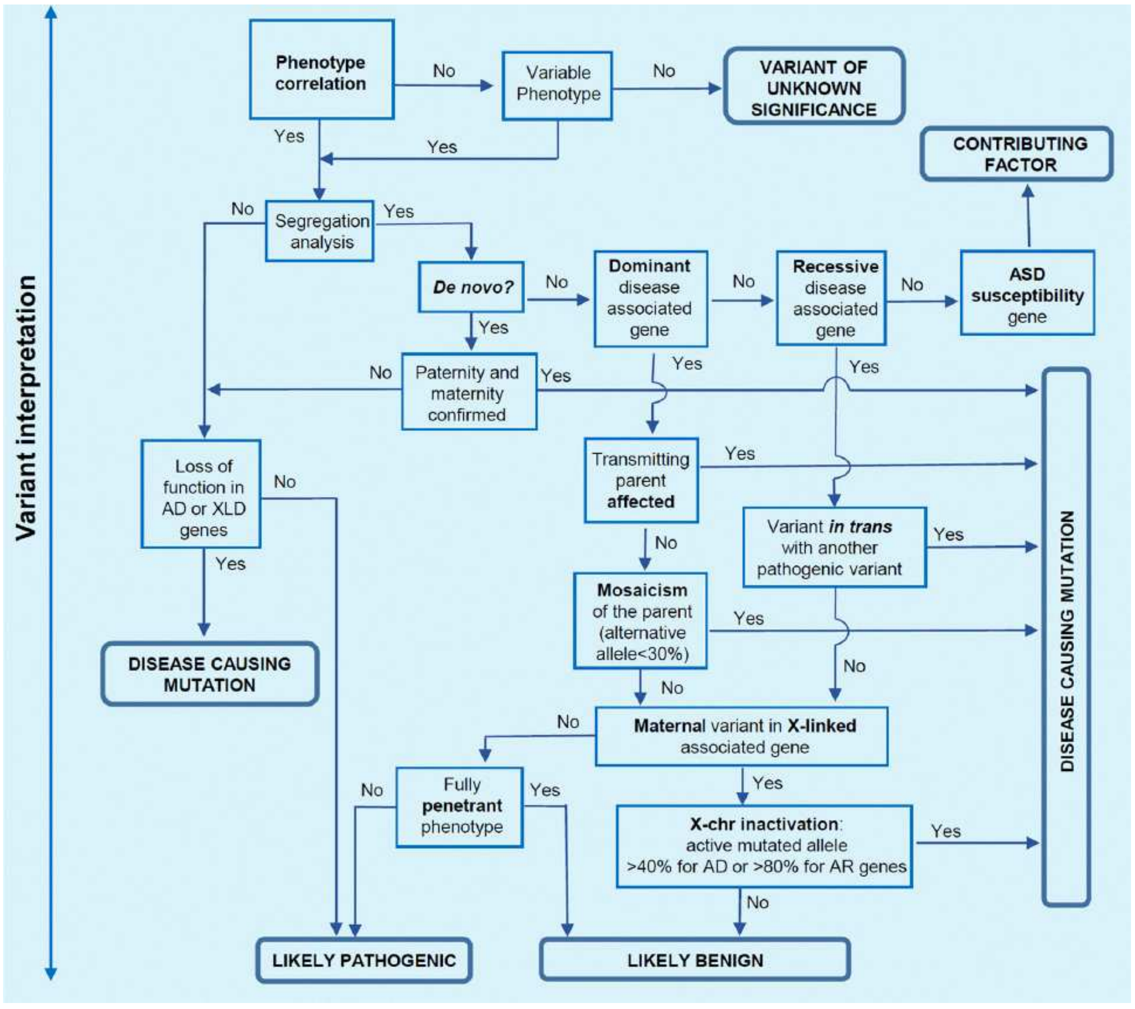

Variant interpretation
Phenotype correlation
No
Variable Phenotype
No
VARIANT OF UNKNOWN SIGNIFICANCE
Yes
Yes
CONTRIBUTING FACTOR
No
Segregation analysis
Yes
De novo?
No
Dominant disease associated gene
No
Recessive disease associated gene
No
ASD susceptibility gene
Yes
No
Paternity and maternity confirmed
Yes
Yes
Yes
Loss of function in AD or XLD genes
No
Transmitting parent affected
Yes
Yes
DISEASE CAUSING MUTATION
No
Variant in trans with another pathogenic variant
Yes
Mosaicism of the parent (alternative allele<30%)
Yes
No
No
No
Maternal variant in X-linked associated gene
No
Fully penetrant phenotype
Yes
Yes
X-chr inactivation: active mutated allele >40% for AD or >80% for AR genes
Yes
No
LIKELY PATHOGENIC
LIKELY BENIGN
DISEASE CAUSING MUTATION

Figure 2.2: Variant Labeling Workflow (Conducted Manually)

performance was observed in Submission 3 of Group 4, correctly predicting 29 variants (11 causative, 9 putative causative, and 9 contributing factor variants).

Only a small amount of variants were accurately predicted by all groups. Specifically, 28% of causative and 15% of contributing variants were correctly identified by at least three groups, while 17% of putative variants were well predicted by the same criteria. Notably, Group 2 not only predicted the most variants but also achieved the highest fraction of correctly predicted variants, calculated as the number of well-predicted variants divided by all predicted variants across all patients and phenotypes. At least one group accurately predicted 16 of 18 putative mutations, with 7 variants consistently identified by the majority of the groups. Among these, 3 inherited variants were suspected to contribute to the disease in conjunction with other genetic or environmental factors.

Additionally, at least one group successfully predicted all 13 variants classified as contributing factors, with 7 variants consistently identified by the majority of the groups. Notably, this variant class was found to be particularly relevant for ASD susceptibility. In Figure 3.2, "Experimental" refers to the variants that were identified by the BioComputing UP Group.

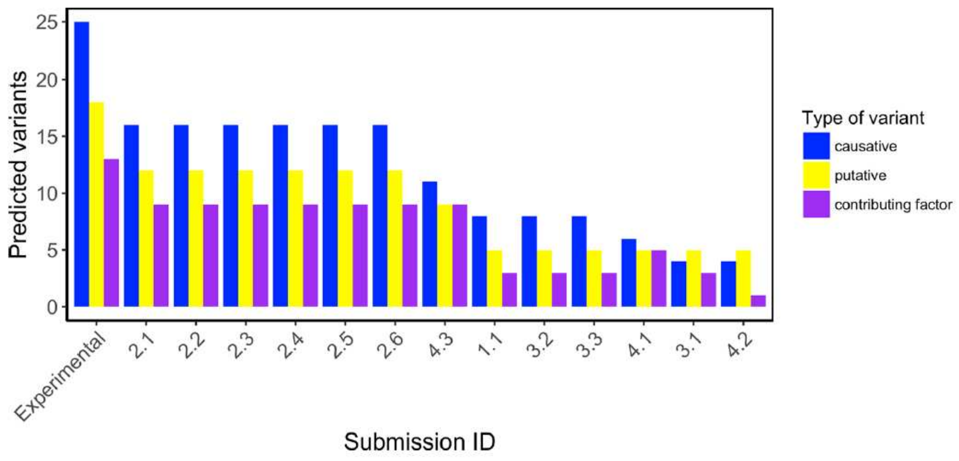


Figure 2.3: Predicted variants distribution among submissions of CAGI5

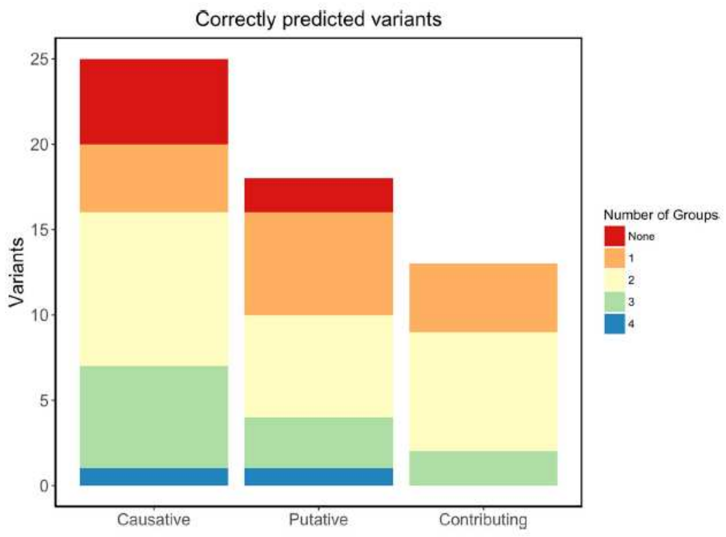


Figure 2.4: Proportional amount of variants, correctly classified by the groups of CAGI5

2.1.2 Phenotype Classification and Assignment

After collecting the submissions of different groups, the prediction assessment was focused on evaluating and considering their performances on each phenotype. This approach has been successfully used for the analysis of multilabel classifier performance since it focuses on a set of two-class prediction problems [5]. It has also simplified the assessment procedure, allowing to compare performances on each single phenotype, instead of evaluating the whole predicted class matrix (150×7, one prediction for each patient and phenotype). Predicted disease classes for each participant were evaluated against the clinical phenotype given in the BioComputing UP Lab answer key. The assessment was performed using various performance metrics such as sensitivity, specificity, AUC (Area Under the Curve), MCC (Matthew Correlation Coefficient), ACC (Accuracy), and F1 measures to evaluate predictions for each phenotype.

Besides having the highest number of predictions, Group 2 was the most accurate in predicting the correct combination of phenotypic traits in patients for whom pathogenic variants were identified. Group 1 was the other best-performing group, accurately predicting the overall phenotype of 11 patients that other groups did not predict correctly. However, there were discrepancies between the accuracy of predicting causal variants and phenotypic traits. For instance, Group 1's method was less accurate in predicting causal or putative variants indicated by the NDDs Genetic Laboratory of the University of Padua.

Moreover, the assessment of variant predictions revealed that Group 2 outperformed other groups in predicting well-associated variants to patient phenotypes. The study also emphasized certain variants that were reconsidered by the NDDs Genetic Laboratory for Sanger validation and segregation analysis, given that they were identified as potentially pathogenic by the majority of the groups.

For the ID phenotype, Submission 4 from Group 2 attained the highest AUC value (0.78), closely followed by submissions 2, 6, and 3 from the same group, as well as submission 3 from Group 3. Submission 3.3 demonstrated the highest overall performance considering all metrics, accurately predicting 146 out of 150 patients for ID. Among patients with ASD, the second most prevalent phenotype, all Group 4 submissions and submission 2.3 achieved higher AUC values than other groups, although these values (average 0.56) and ROC curves remained close to random. Notably, submissions 4.3 and 1.1 demonstrated the best performance based on other metrics, with submission 1.1 equaling accuracy (ACC), Matthew Correlation Coefficient (MCC), and F1 score. Both submissions accurately identified the patient phenotype in nearly 100% of cases. Despite the rather good AUC, ACC, and F1 values reached by some groups for the ID phenotype, and also for ASD, the MCC values remain quite low. Since MCC is not influenced by unbalanced categories, it shows a more realistic picture of prediction performance. As most of the patients have ID and ASD phenotypes, the confusion matrix is completely biased towards true positive values due to the highly imbalanced classes. This causes the ROC curve and consequent AUC not to reflect correctly the real predictor performance.

The presence or absence of the Epilepsy phenotype was poorly predicted by most groups, with an average MCC value of 0.05. This phenotype was particularly difficult, as roughly half of the patients had the disease. The best performances were achieved by Group 4 and submission 1.1, predicting adequately more than 60% of patients.

Information about the presence or absence of Microcephaly and Macrocephaly was available for about half (81) of the patients. Microcephaly was reported in 18 patients and Macrocephaly in 12 patients. Predictions for Microcephaly performed modestly, the best AUC being reached by submission 4.3, which correctly predicted 42 patients. Group 1 also predicted most of the patients with the phenotype (15 correct). In addition, most group 2 submissions obtained the best MCC and ACC values compared to other groups, predicting correctly 66 patients. However, the best MCC values are again poor compared to other measures, denoting the effect of unbalanced categories in the predictions. Group 2 predictions were biased to identify patients without the phenotype (63 of 63 patients) and just three patients with the disease. On the contrary, submission 4.3 was biased to predict patients with the disease (17 of 18 patients) and 25 patients without the phenotype.

The hypotonia phenotype was positively or negatively noted in 68 patients by the Padua NDD lab. AUC values reached by different groups are poor, averaging around 0.5. Indeed, performance measures such as MCC and ACC are lower than in other phenotypes. Submission 4.3 obtained the best AUC, MCC, and ACC values compared with other groups, correctly predicting 44 patients (6 of 28 with the phenotype and 38 of 40 without the phenotype). Submissions 2.1 and 2.3 predicted most of the patients with the phenotype (17 and 16, respectively).

The ataxia phenotype was noted positively and negatively in 54 patients and only 11 patients had the disease. Submissions 4.1, 4.3, 2.1, 2.2, and 2.5 predicted well most of the patients but were biased to detect patients without the phenotype. Submissions 2.3 and 2.4 correctly predicted the presence of the disease in 7 and 8 patients, respectively. Consequently, the best AUC and MCC values were obtained by submission 2.4.

The overall submission ranking of this challenge was made considering the average AUC rankings for each phenotype. The best average ranked was submission 4.3, followed by other submissions of the same group.

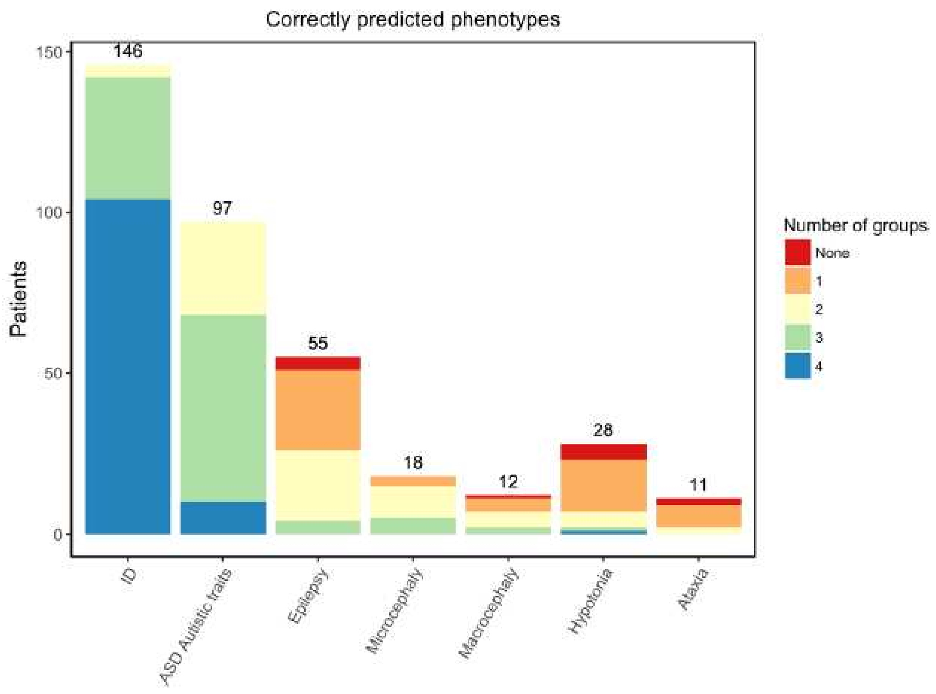


Figure 2.5: Proportional number of patients with the phenotype, correctly predicted by the groups of CAGI5

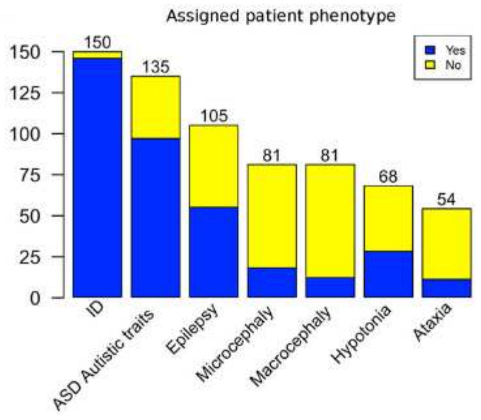


Figure 2.6: The number of patients with the phenotypes (True Labels of CAGI5).

2.2 CAGI6 Workflow and Findings

Being the same challenge as CAGI5 and BioComputing UP Lab the assessor once again, the purpose of the CAGI6 ID challenge was a critical assessment of computational methods in predicting patient clinical phenotypes and causal variants based on gene panel sequence data in NDDs [6]. While CAGI5 involved 150 patients and 13 submissions from 4 groups, CAGI6 expanded its scope to 415 patients with NDDs, involving 8 research teams and 30 models. Before starting CAGI6, the predictors were able to access the 74-gene panel data and clinical descriptions of the CAGI5 cohort (N = 150), to train their prediction methods and find the best strategies. The gene panel consisted of 74 genes again and genetic data was based on the sequence data of patients in VCF format. Once again, after conducting its own experiments (ground truth), BioComputing UP Lab merged the results of variant analysis and the clinical traits to assess the findings of each predictor. The evaluation process used for submissions included performance metrics such as sensitivity, specificity, MCC, accuracy, and F1 score. The study compared the results among different predictors using z-scores at the phenotypic level on the ROC curves' AUC values. Using a similar approach for variant analysis and filtering, the results of the CAGI6 made significant improvements, compared to CAGI5. The assessment showed that some submissions were more successful than CAGI5 for the majority of patients, with varying performance across different clinical traits. Here, the experimental methods will not be explained again, as it has been described in detail for CAGI5, but the outcomes and new findings will be discussed.

2.2.1 Variant Filtering And Classification

For the category of causal variants, which is the most interesting clinical class, the best result was achieved with 54 out of 60 by Group SID8.1 (SID: Submission ID) and 8.6, followed by four additional groups (6, 4, 3, 7). The SID8, SID3, and SID7 correctly predicted the highest number of putative and contributing factors. Contributing Factor variations have been found in genes linked to ASD, even though they did not fully meet the primary classification criteria for pathogenic variations. A significant improvement is evident when comparing the outcomes to the CAGI5 challenge, with coverage of causal variant predictions increasing from 64 to 90% when examining the corresponding best model. There has also been a significant improvement for putative and contributory factors, which went from 66–79% and 69–76%, respectively.

It was found that a tiny percentage of the causative variants (3 out of 60) were predicted by all groups, but the majority of variants were predicted by at least two groups. Regarding the putative variants, one was missed by all the groups. Furthermore, whereas 3 groups have predicted the putative variants, the majority of the causative variants have been predicted by at least 4 groups. Overall, it can be observed that contributing factor variants were sparsely predicted, where 36% of them were predicted only by one group (29 variants) or not predicted at all (8 variants). In this case, no putative or contributory mutation was anticipated by all groups with complete agreement.

Group SID8.6, which had an 82% recall rate, was the most effective model for capturing a variety of mutations, according to the data provided in the research. However, its accuracy of 58% was significantly lower, pointing out a high percentage of false positives in the findings. However, submission 6.2 outperformed all other models in terms of accurate prediction, with an accuracy of 72.4%. On the other hand, it showed a reduced recall of 35%. This result showed how well SID6 performed in predicting causal variants, but it had limitations in detecting putative and contributing variants. In Figure 3.2, “Experimental” refers to the variants that were identified by the BioComputing UP Group.

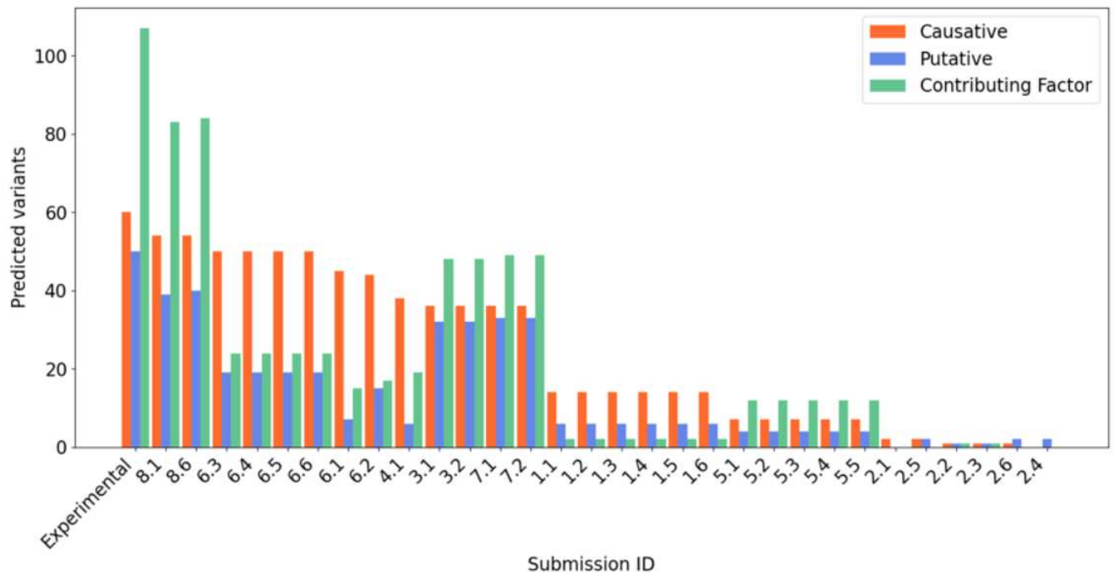


Figure 2.7: Predicted variants distribution among groups of CAGI6

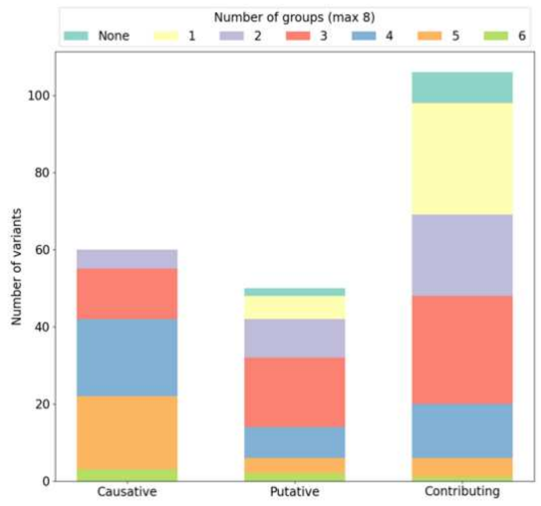


Figure 2.8: Proportional amount of variants, correctly classified by the groups of CAGI6

2.2.2 Phenotype Classification and Assignment

For the evaluation of phenotype predictions, the focus was on following an approach of a two-class prediction problem and this assessment approach has been proven successful among different domains [5]. It offered the advantage of simplifying the assessment process by enabling a comparison of the performance of different methods for each phenotype. This was in contrast to evaluating them based on the entire predicted class matrix (415 x 7), which consisted of one prediction for each patient and phenotype. The assessment was done separately for each phenotype column, with the submitted probability values in binary classes. A threshold probability value that maximized MCC was determined for that particular phenotype. Subsequently, all probability values for each phenotype were compared with their corresponding threshold and assigned a binary value considering the threshold. Regarding the results, all groups correctly predicted the ID in 338 (49.4%) out of 352 patients; while 7 groups accurately predicted the 90.8% of patients with ID. A small number of patients (N=45) showed microcephaly as a phenotype. Eight groups correctly identified the presence of this trait in 55% of individuals and at least 7 groups correctly identified the microcephaly in 93.3% of the patients. There was a noticeable general discrepancy between the MCC value and the AUC results when evaluating the performance of the phenotypic predictions. In particular, some relatively high AUC values led to lower MCC, and vice versa. Generally, the MCC offers a more accurate representation of prediction performance and is immune to the impacts of imbalanced categories. The confusion matrix was entirely skewed towards true positive values because of the severely imbalanced classes, which made sense given that the majority of patients exhibit the ID and ASD characteristics. Thus, a bootstrapping technique was used to lessen this effect, given that ROC and AUC do not accurately reflect the true predictor performance [6].

Among phenotypes, ID was the easiest to match for Groups SID8.6 and SID8.1, followed by SID2.4 and SID5.3. We notice that SID8.6 accurately predicted the phenotype for 255 out of 352 patients, displaying a positive correlation with the available clinical data and achieving an F1-score of 0.85.

The second most prevalent trait in the cohort was ASD, reported by clinicians in 202 out of 415 pediatric patients. The highest AUC values for this phenotype were attained by Groups SID1.5 (AUC 0.59) and SID1.2 (AUC 0.58). Additionally, SID5.4 ranked third with an AUC of 0.55. However, it is worth noting that the AUC values for ASD remain relatively low, approaching random performance.

Contrary to the previous CAGI5 [3], some differences were noticed in the phenotype predictions. The prediction of the Epilepsy phenotype in CAGI6 exhibited superior performance, as evidenced by a mean MCC value of 0.09, nearly twice the previous value. Group SID1 attained the highest results, achieving an AUC of 0.58, an MCC of 0.12, and an F1 score of 0.38 for SID4. These performances, when considering all submissions and groups, stood out as the best performances.

For microcephaly and macrocephaly, any improvement was not observed, but it is important to specify that the CAGI5 dataset included a small number of patients with microcephaly (18) and with macrocephaly (12). It was difficult to train with a small number of examples. However, certain submissions demonstrated accurate prediction of the patient phenotypes, such as Group SID6.5 for microcephaly achieving an AUC of 0.64 and a recall of 0.67, and SID1.5 for macrocephaly achieving an AUC of 0.65 and a recall of 0.3.

71 patients out of 415 were reported with hypotonia, while 254 out of 415 patients were negative. Compared to CAGI5, a significant improvement for this phenotype was not observed. The maximum AUC across all submissions was 0.54, achieved by Group SID1.1, achieving a recall of 0.5 and an F1 score of 0.35.

The presence of the ataxia phenotype was observed in a group of 30 patients, while 285 patients resulted negative. Group SID5.3, has the highest-performing model with an AUC of 0.66, and an F1 score of 0.23. This result was consistent with the previous assessment.

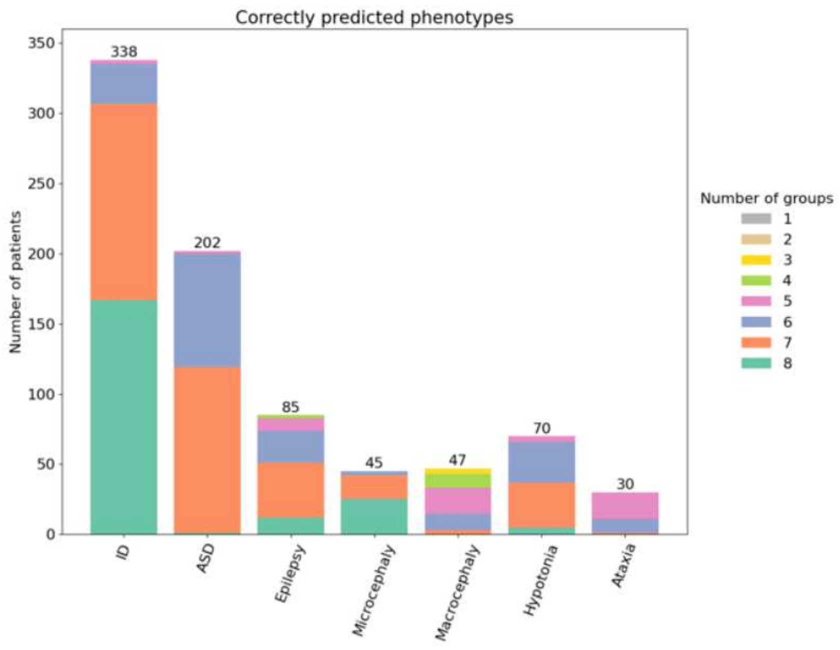


Figure 2.9: Proportional number of patients with the phenotype, correctly predicted by the groups of CAGI6

3

Background Information

In this chapter, the concepts that constitute the basis for the project and that are frequently used in the rest of the study are defined and some information related to genetics is given.

3.1 Neurodevelopmental Disorders (NDDs)

As mentioned, this study aims to predict the disease phenotypes of the patients from genetic variants. For this sake, the target neurodevelopmental diseases and some symptoms are explained. NDDs include a diverse group of conditions characterized by impairments in the growth and development of the nervous system. These disorders typically manifest early in development, influencing cognitive, motor, social, and emotional functions. They tend to be genetically diverse and heterogeneous, meaning that there are various genetic factors influencing their manifestation, making relevant studies difficult.

ID is a condition that affects a person's day-to-day functioning and is characterized by limits in intellectual functioning and adaptive behaviors. There is a large range in the severity of intellectual disability, from minor to profound. Individuals diagnosed with ID may face difficulties with learning, solving problems, communicating, and interacting with others. Children with ID, for instance, may struggle to remember things, speak incoherently, or comprehend social norms. With an estimated prevalence of 1-3% globally, ID is the most prevalent NDD in the population.

ASD is defined by anomalies in a range of characteristics, including communication, social interaction, and repetitive activities. Individuals diagnosed with ASD may experience issues with eye contact, nonverbal cues, and communication. Sensitivities to light or sound stimuli can be present in many people. On the other hand, some people could not react at all to specific sensory stimuli. Symptoms usually appear in early childhood, frequently before the age of three. ADHD and rage disorders are examples of co-occurring disorders in people with ASD.

In the case of **Epilepsy**, patients have recurrent, unpredictable seizures resulting from abnormal electrical activity in the brain, leading to temporary disruptions in normal brain function. The manifestation of seizures can vary greatly, from brief awareness lapses to complete unconsciousness. Although the exact origin of many cases is still unknown, potential causes include infections, brain traumas, and genetic factors. A thorough evaluation is required to diagnose epilepsy, consisting of a neurological examination, a full medical history, brain imaging, and EEG.

Macrocephaly refers to an enlarged head size or circumference beyond what is considered typical for a given age, sex, and ethnic background (using a common 2 standard deviations threshold). The reasons may be genetic conditions, metabolic disorders or certain syndromes, but generally, are not always apparent. Patients with macrocephaly may have developmental problems, ID or other neurological symptoms.

Microcephaly, on the other hand, is defined by a reduced head size (2 standard deviations or the 3rd percentile) compared to the norm, indicating inadequate brain development. It may result from genetic factors, prenatal exposure to certain substances, or infections during pregnancy. Similar to Macrocephaly, patients may exhibit some neurological symptoms such as poor motor function, poor speech, abnormal facial features, seizures, and dwarfism.

Hypotonia is a medical condition characterized by reduced muscle tone or tension in the muscles, leading to decreased resistance to passive movement. This lack of muscle tone can affect various muscle groups in the body. Individuals with hypotonia may have symptoms such as speech difficulties, poor reflexes, and challenges with balance and posture. The reasons may depend on neurological disorders, and genetic and metabolic conditions. Physical therapy is often employed to improve muscle strength and coordination.

Ataxia is characterized by a lack of coordination and voluntary muscle control, leading to unsteady movements and difficulty in maintaining balance. The cerebellum, a region of the brain responsible for coordinating movements, is often affected. Individuals with ataxia may struggle with tasks that require fine motor skills, such as writing, buttoning a shirt, or handling objects along with speech and swallowing difficulty.

3.2 Genetic Mutations and Variations

As the purpose of this study is to classify the genetic variants that may cause NDDs, it is important to know about the concept of genetic variations and mutations. Here, these two notions are discussed and different types of genetic variations that are encountered in this study are explained.

Genetic mutations and variations are fundamental concepts in genetics that refer to changes in the DNA sequence. They are often confused; while both terms involve alterations in the genetic code, they have distinct characteristics. In spite of their similarities, they differ in terms of frequency, effect, and clinical significance.

Mutations are abnormal changes that occur in the DNA sequence, either in a single nucleotide or involving larger segments of the genetic material. They can happen spontaneously during DNA replication or in response to external factors such as exposure to radiation or chemicals. They are usually rare occurrences, and their frequency is low in a population. They may have varying effects on the organism. Some mutations are neutral, while others can be harmful or beneficial. The majority is harmful, leading to genetic disorders or diseases. Rarely, mutations may provide a survival advantage, contributing to evolution.

Genetic variations refer to the naturally occurring differences in DNA sequences between individuals of the same species. Similar to mutations, they may manifest in various forms, such as single nucleotide polymorphisms (SNPs), insertions, deletions, and other structural changes that contribute to the genetic diversity within a population. Many genetic variations are neutral and do not have a significant impact on an individual's health, instead, they cause normal genetic variation within a population. Some variations, however, may have advantages or disadvantages to certain traits.

Variants are key factors in personalized medicine and understanding individual responses to drugs and treatments. In most of the studies for understanding the effects of variants, de novo (not inherited from parents) variants are focused on since they are believed to be clinically significant or effective [7], [8]. De novo mutations are shown to be a major cause of severe early-onset genetic disorders such as intellectual disability, autism spectrum disorder, and other developmental diseases [9], [10]. Some of the variant types are described below.

- **Single Nucleotide Variations (SNVs)**
  Being the most common variations, SNVs or SNPs refer to a single point (nucleotide) change in the DNA or RNA sequence. This change can arise from a substitution of a nucleotide with a different one by the insertion or deletion of nucleotides in the DNA

sequence of a gene. Depending on their location along the gene, these variants can have various effects.

- **Exonic and Intronic Variants**
  Exons are coding regions of genes that contain the information for protein synthesis, while introns are non-coding regions that intervene between exons in the gene sequence. Variations in exons can lead to changes in the protein-coding sequence, potentially affecting the function of the protein. Exonic variants are often implicated in genetic disorders and diseases. While introns do not code for proteins, intronic variants can impact gene expression, splicing, and other regulatory elements and should not be ignored. Some intronic variants are associated with diseases through their influence on gene regulation. Variants occurring in exonic regions can be classified as follows:

  * **Missense** variants result in the substitution of one amino acid in a protein for another. This can alter the structure and function of the protein, potentially leading to changes in its normal activity or function.
  * **Nonsense** variants cause a premature stop codon, resulting in a shortened and often nonfunctional protein during synthesis.
  * **Synonymous (Silent)** variants are observed when a nucleotide is altered but it does not change the corresponding amino acid, thus the protein. This is because some different codes of DNA can be translated to the same amino acid.
  * **Frameshift Variants** can be defined as any alteration in the sequence that causes a shift in the transcription process. They can result from insertions or deletions that disrupt the reading frame, altering the entire amino acid sequence downstream. They also involve **repeat expansions**, an increase in the number of repetitive DNA sequences leading to frameshifts.

  Variants occurring in intronic regions can be classified as follows:

  * **Splicing Variants**
    Splicing is a process where introns are removed and the remaining exons (coding regions) are joined together to form the mature mRNA. Variations in splicing allow the synthesis of different protein variants and contribute to proteomic diversity by inclusion/exclusion of specific exons, influencing the structure and function of the resulting proteins. Incorrect splicing is associated with various genetic disorders.
  * **UTR3 and UTR5 (Untranslated Region) Variants**
    Untranslated regions 3 and 5 are situated at both ends of the non-coding region of the mRNA molecule. Since they have regulatory effects on the transcription process, variants that occur here can influence mRNA stability, and

translation and may impact protein production and contribute to disease susceptibility.

* **Upstream Variants**
  Upstream in molecular biology describes the orientation that is toward the 5' end of a DNA or RNA molecule. The phosphate group of the nucleotide is connected at its 5' end. On the other hand, "downstream" describes the path that leads toward the nucleotide's 3' end. The relative locations of genetic elements are denoted by these phrases. Variations that occur in the regulatory regions located upstream of a gene play a crucial role in transcription, may impact the binding of regulatory elements and affect gene function to cause a disease.

- **Structural Variations**
  They are observed when the structure of the chromosomes or DNA segments is changed. Unlike SNVs and small indels, structural variations impact larger portions of the genome. They can include insertions and deletions in the macro scale as well.

  – **Duplication**: involves the duplication of a segment of DNA, altering the number of copies of a particular gene or genomic region.

  – **Inversion**: reverses the orientation of a segment of DNA within a chromosome.

  – **Translocation**: involves the transfer of a segment of DNA from one chromosome to another.

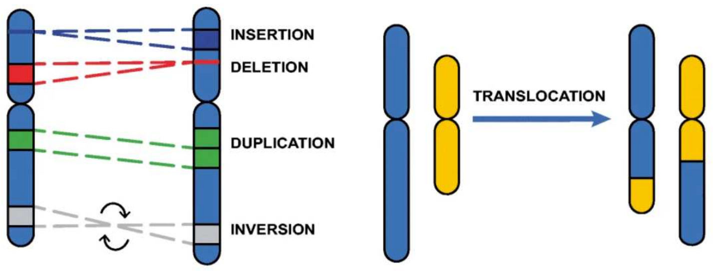


Figure 3.1: Chromosome representation of some structural variants

This thesis study included miscellaneous kinds of variants that are intronic and exonic, in addition, to variants in the non-coding but regulatory regions such as UTR3 and UTR5, splicing, and upstream variants. Even though the vast majority of the variants were SNVs, the other types were kept to analyze, especially the ones causing frameshifts, considering the possibility of being associated with a disease. The detailed description and visualization of the data and variant types are given in the following sections.

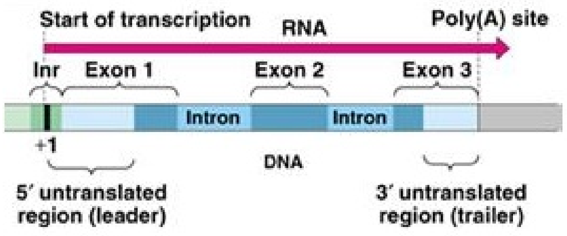


Figure 3.2: Different regions of DNA in the transcription site

3.3 Omics Data & Databases

The concept of **Omics** emerged to describe the comprehensive, systematic study of various components or processes within a biological system; it is a collective term used to describe high-throughput approaches and disciplines such as genomics, transcriptomics, proteomics, metabolomics, and others. The aim is to analyze and understand the relationships between these components.

Omics data refers to the large-scale datasets. For instance, genomic data encompasses information about an organism's complete set of genes, while transcriptomic data involves the study of all RNA transcripts. Proteomic and metabolomic data cover the entire complement of proteins and metabolites, respectively. These datasets provide a comprehensive view of biological systems, aiding in the identification of patterns, relationships, and key elements. Omics databases play an important role in organizing, storing, and providing accessibility to large-scale biological data generated through various omics technologies such as Next-Generation Sequencing and Mass Spectrometry. These databases facilitate data sharing, analysis, and interpretation, contributing significantly to advancements in biomedical research. Here are the examples of some omics databases that are frequently used:

- Genomic Databases: NCBI GenBank, Ensembl, and Genome Browser.
- Proteomic Databases: UniProt, MobiDB, and PeptideAtlas.
- Transcriptomic Databases: GEO (Gene Expression Omnibus) and ArrayExpress.

- Metabolomic Databases: HMDB (Human Metabolome Database) and METLIN.

These databases serve as valuable resources for researchers to retrieve, analyze, and interpret omics data, fostering advancements in various biological and medical fields. For this study and the previous CAGI challenges, some omics databases were used to retrieve information to make inferences on variants and phenotypes and to manipulate the data. UCSC Genomic's Liftover Tool was used to change chromosome positions from an older version of the human genome database into the new one. ANNOVAR databases were used to annotate the genetic variants based on their chromosomal locations. The information was retrieved from CLINVAR, Gnomad and GATK Broad Institute. HPO, OMIM and Orphanet were employed to discover gene–disease associations for the genes selected for the study.

3.4 Workflow and File Types in Variant Calling

Since the field of omics encompasses various disciplines each focusing on different molecular aspects, there are various processes and each process yields and/or requires a different data type. Clinicians and bioinformaticians benefit from standardized processes consisting of certain methods and specific file types to make analyses and inferences on diseases, genetic conditions or biological pathways.

Generally, the workflow starts with **sequencing**, a high-throughput laboratory technique used to determine the genetic code of the organism, which yields **FASTQ** files. A FASTQ file contains a list of short DNA sequences (reads) generated by the sequencing machine, along with corresponding quality scores that indicate confidence in each base call. After obtaining this raw sequence data **SAM** or **BAM** files are used for preprocessing. This involves the alignment process, which is mapping the reads of sequencing to a reference genome in order to determine where each read belongs in the genome. SAM file is a text-based file format that stores sequence alignment information. They can be large and are often compressed into a more efficient format called BAM. BAM files are the compressed and binary-encoded versions of SAM files, making them faster to read and write. After this step the variants of the organism must be retrieved, this process is called **Variant Calling** and specialized algorithms are used for this purpose. The results of variant calling are often stored in a **VCF (Variant Call Format)** file. This file contains information about various types of variants, including single nucleotide polymorphisms (SNPs), insertions, deletions, and structural variants. VCF files are essential for annotating and interpreting genetic variants. This process involves accessing the available databases containing the variant lists of that organism and bringing the information of found variants. Annotations can include information about the location of variants in genes, their potential impact on protein function, and known associations with diseases. For this study, ANNOVAR was used for annotation, a description of the algorithm and more information about this process can be found in the following section.

3.5 Annotation

In the case of genetic variants, annotation involves adding information or metadata to the variants, providing context and interpretation for researchers and clinicians. It has a crucial role in prioritizing variants for further investigation in functional studies or disease association studies, facilitating the identification of disease-causing mutations. With annotation, necessary information is retrieved such as the position of the variant, region attributes, functional impacts, population frequencies, conservation scores, disease-causing prediction scores, etc. After obtaining the variants in VCF file format, specialized software or tools are used to perform annotation. The most comprehensive and popular annotation software are **ANNOVAR (ANNOtate VARiation)** and **VEP (Variant Effect Predictor)**.

In this thesis, ANNOVAR was employed due to its ease of use and extensive genomic databases. It is an efficient software tool to utilize updated information from diverse genomes (including human genomes hg18, hg19, and hg38, as well as mouse, worm, fly, yeast, etc.). To perform the annotation, the program is required to have a list of variants with chromosome, start position, end position, reference nucleotide and observed nucleotides, which are naturally present in VCF files. ANNOVAR can perform the following annotations:

- Gene-based annotation: identify whether variants cause protein-coding changes and the amino acids that are affected.
- Region-based annotation: identify variants in specific genomic regions, for instance, conserved regions, binding sites, duplication regions, etc.
- Filter-based annotation: identify variants that are documented in specific databases, for instance, checking the presence of a variant in dbSNP, filtering with gnomAD allele frequency, GERP or CADD scores.

The following sections on data and methodology discuss the databases used to acquire the information as well as other parameters of the ANNOVAR annotation performed in this study.

4
Data Acquisition & Retrieving

This section explains the process of generating raw sequencing data in the previous CAGI studies by Padua NDD Lab, which serves as the foundation for this study. Secondly, It gives an introduction to the data used in this study and the methodology employed to obtain it.

4.1 Raw Data Acquisition: Sequencing

Next-Generation Sequencing (NGS) technologies have revolutionized the field of genomics, allowing rapid and cost-effective analysis of DNA and RNA. Sequencing, at its core, is the process of determining the exact order of nucleotides within a given DNA or RNA molecule. NGS techniques employ high-throughput methods that allow for the simultaneous sequencing of numerous fragments, generating massive amounts of data in a single run. Some key NGS platforms include Illumina, Ion Torrent, and PacBio, each with its unique strengths. For instance, Ion Torrent, which was used during this research, relies on semiconductor sequencing, measuring changes in pH as nucleotides are added. These diverse technologies offer researchers insights into genomic structures, variations, and gene expressions, contributing significantly to clinical diagnostics.

First, clinical data of patients from Italian public hospitals with a diagnosis of NDD were collected. To construct an efficient and cost-effective gene panel, the study selected candidate genes associated with ID or ASD by integrating data from various public databases, including AutismKB, SFARI, OMIM, and PubMed. Then a meta-analysis study was made including annotation for clinical phenotype, gene function, subcellular localization, and interactions. Utilizing a dedicated SQL database and STRING 9.0, a disease protein–protein interaction (PPI) network was established. This network is derived from 66 high-confidence genes shared by ASD and ID lists. After an enrichment analysis and gene prioritization, the result consisted of a gene panel of 74 genes including known causative genes, top-ranked genes by prioritization, and genes meeting PPI network parameters, providing a comprehensive set for subsequent analyses.

Blood samples were treated with the Wizard genomic DNA Promega Kit (Promega Corporation) to extract nucleic acids. Using Thermo Fisher Scientific's Ion AmpliSeqTM Designer, multiplex PCR-based (polymerase chain reaction) primer panels were created to amplify all 74 of the targeted genes' exons and surrounding areas (10 bp). Thermo Fisher Scientific's Ion One Touch 2 and Ion One Touch ES Systems were used for template preparation and enrichment, respectively. Thermo Fisher Scientific's Ion Torrent Suite Software v5.02 was used to read and align to the human genome reference (hg19/GRCh37) and to call variants. Variant calling process yielded VCF files, having information on sequence data in the headers. Finally, called variants were annotated with the ANNOVAR software, and these annotated VCF files constituted the basis for this study.

4.2 Introduction to Data at Hand

At the beginning of the project, VCF files used in the course of the CAGI5 and CAGI6 challenges, as well as their annotated files, were gathered as training datasets. The dataset included 867 VCF-formatted patient files with accompanying annotation files. Out of these, 415 patients were added by CAGI6 after 150 files were used in CAGI5. For a variety of reasons, including the lack of useful variation types, hardness to analyze, belonging to patients of the same family (e.g., siblings), or because of additional technical and clinical factors related to the patients, the remaining 302 files were decided not to be used in the predictions by the group. Also, the patient clinical files, which contained anonymous information about patient records were collected. They contained gender, age, physical symptoms, and cohort frequency information as well as many other data, and clinical diagnosis of 7 phenotypes that are the interests of this study. Those constituted the labels (target feature) for phenotype prediction purposes. As for the second purpose, which was the examination and filtration of these variants in order to build a powerful tool for the classification of variants, the selected variants by CAGI5 and CAGI6 merged and the different classes of variants were used as labels in supervised learning.

At first, up-to-dateness and compatibility of the data were investigated since the latest data that was collected belonged to 4 years ago. The annotations of 867 patients were based on "hg37" (human genome build 37 [GRCh37]), which is a specific version of the human genome reference assembly.

Genome assemblies serve as a comprehensive sequence of the human DNA, capturing the typical genetic representations of the complete sequence of an organism. Different human genome assemblies are created by making additional builds or updates over time as technology improves and more data becomes available. This has contributed to their refinement and improved their accuracy. Up to now, 4 versions of the new human reference genome hg38 (GRCh38) have been published in gnomAD, v4.0.0 being the latest one, published in November 2023.

In the context of ANNOVAR annotation, the genome build is crucial for accurately annotating genetic variants. ANNOVAR uses a reference genome to compare and annotate the variations found in individual genomes or sequencing data. When annotating variants with ANNOVAR, it is essential to specify the reference genome build as a parameter to ensure accurate mapping and interpretation of the genomic coordinates of these variants. This parameter was the hg37 genome in the case of previous works related to CAGI. Therefore it was necessary to perform annotation with new versions of the databases. Here, a mapping problem arose since the chromosome positions of the variants in the VCF files belonged to the former version of the human reference genome. Since the chromosome positions of the variants in different databases vary due to the addition or discovery of new variants, it was necessary to convert the old chromosome positions into new ones to be able to use new databases in the annotation.

A liftover, in the context of genomic data, refers to the process of converting coordinates or positions of genetic elements (such as variants or features) from one reference genome assembly to another. This is necessary when working with data from different genome builds or versions. For this study, The Picard Liftover tool was used, published by Broad Institute GATK (Genome Analysis Toolkit), which is a collection of command-line tools for working with high-throughput sequencing data. The Liftover tool is specifically designed to convert the coordinates of genomic intervals between different genome assemblies. It utilizes chain files that define the correspondence between the two genome builds. Picard converted most of the variants' (97%) chromosome positions, but it failed the ones that did not exist in the chain files. Those variants were new variants that were not listed in the database. The reason was the genetic diversity among individuals, having unique variants that are not published. Picard discarded those variants and separated them by making new files. After this liftover process, The

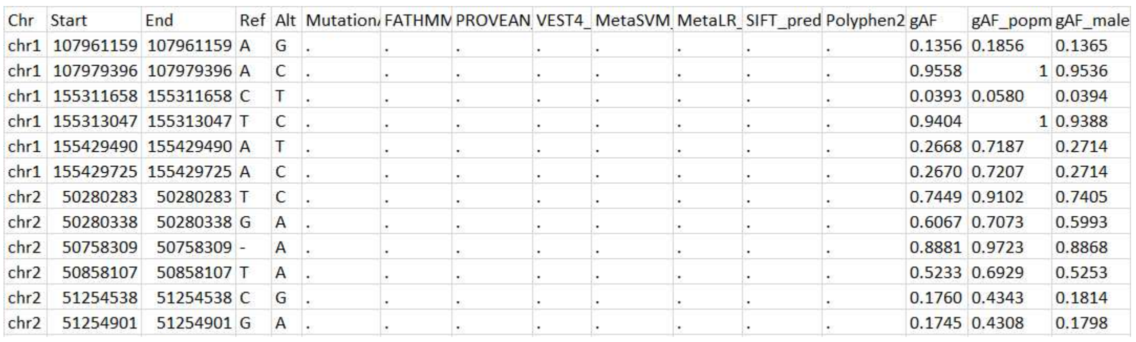
UCSC (University of California, Santa Cruz) LiftOver tool was used to verify the results of Picard. It offered a web-based tool allowing users to select the source and target genome builds, and the tool performed the conversion. In the end, the VCF files with updated chromosome positions were obtained, having the same data for the remaining columns. The variant files (VCFs) were then made input for the annotation process. ANNOVAR was used in a Linux environment, having the following parameters as databases:

protocols="gnomad312_genome, clinvar_20221231, dbnsfp42a, avsnp150, refGene"

In ANNOVAR, **a protocol** refers to a set of rules or guidelines for the annotation and interpretation of genetic variants. ANNOVAR provides various annotation protocols that help users analyze and interpret genetic variants in the context of their potential impact on genes and biological functions. These protocols define how ANNOVAR extracts information from genomic databases, functional prediction algorithms, and other relevant sources to provide comprehensive annotations for each variant. With the protocols indicated above, the following data sources and operations were employed:

- **GnomAD v3.1.2 Genome (gnomad312_genome):** This protocol refers to data from the Genome Aggregation Database (gnomAD version 3.1.2.), providing information on genetic variants across diverse populations and bringing information about allele frequencies among different populations.
- **ClinVar (clinvar_20221231):** ClinVar is a widely used clinical genetics database. This protocol annotates variants using ClinVar data available until December 31, 2022, providing information on the clinical significance of variants based on curated interpretations by experts.
- **dbNSFP (dbnsfp42a):** The Database for Nonsynonymous SNPs' Functional Predictions (dbNSFP) provides functional predictions for nonsynonymous SNVs.
- **avsnp150:** This protocol is related to the Annotated dbSNP database (avsnp), version 150. It provides information about variations listed in the dbSNP database.
- **refGene:** This protocol is related to the RefGene annotation, which provides information about the genomic context of variants, such as their location within different gene regions (exonic, intronic, etc.).

This annotation process generated 867 text-based files each corresponding to a patient VCF file, with 170 columns of information from different databases mentioned above. Since the variants rejected to be converted by the Picard tool did not exist in the VCF files, the total number of the variants was less than the actual number of variants in the original VCF files. Also, when the annotation output files were analyzed, it was noticed that there were a lot of unknown values represented with dots (.) in the columns, especially for the prediction scores of pathogenicity, as seen on the visual in Figure 4.1



Chr	Start	End	Ref	Alt	Mutation/	FATHMM	PROVEAN	VEST4_	MetaSVM_	MetaLR_	SIFT_pred	Polyphen2	gAF	gAF_popm	gAF_male
chr1	107961159	107961159	A	G	.	.	.	.	.	.	.	.	0.1356	0.1856	0.1365
chr1	107979396	107979396	A	C	.	.	.	.	.	.	.	.	0.9558	1	0.9536
chr1	155311658	155311658	C	T	.	.	.	.	.	.	.	.	0.0393	0.0580	0.0394
chr1	155313047	155313047	T	C	.	.	.	.	.	.	.	.	0.9404	1	0.9388
chr1	155429490	155429490	A	T	.	.	.	.	.	.	.	.	0.2668	0.7187	0.2714
chr1	155429725	155429725	A	C	.	.	.	.	.	.	.	.	0.2670	0.7207	0.2714
chr2	50280283	50280283	T	C	.	.	.	.	.	.	.	.	0.7449	0.9102	0.7405
chr2	50280338	50280338	G	A	.	.	.	.	.	.	.	.	0.6067	0.7073	0.5993
chr2	50758309	50758309	-	A	.	.	.	.	.	.	.	.	0.8881	0.9723	0.8868
chr2	50858107	50858107	T	A	.	.	.	.	.	.	.	.	0.5233	0.6929	0.5253
chr2	51254538	51254538	C	G	.	.	.	.	.	.	.	.	0.1760	0.4343	0.1814
chr2	51254901	51254901	G	A	.	.	.	.	.	.	.	.	0.1745	0.4308	0.1798

Figure 4.1: Some of the columns of annotated file

The reason for the unknown values where the annotation information for a particular variant is not available or cannot be determined. When ANNOVAR processes VCF files and annotates variants, it may encounter instances where specific annotations cannot be retrieved or calculated for various reasons. These reasons could include:

- **Lack of Information:** The variant possibly may lack needed information for some annotations. For instance, ANNOVAR may assign an ”unknown” value, if the database used for annotation does not contain information for a particular variant. Most of the available computational tools for variant pathogenicity predictions work only for genetic variants in the coding part of the genes that cause an amino acid substitution or introduce a stop codon. For this reason, intronic variants can not be annotated with a pathogenicity score by these methods.

- **Uncertain Predictions:** Some annotations depend on predictive algorithms or external databases, and if the confidence or evidence for a particular annotation is low or uncertain, ANNOVAR may use ”unknown” to indicate this case.

- **Database Limitations:** If a specific annotation relies on external databases or resources that are not available or updated, ANNOVAR may not be able to provide a value.

It’s important to carefully interpret “unknown” values and consider them in the broader context of variant annotation. In such cases, additional validation or exploration may be necessary to understand the significance of these variants.

In 867 annotation files, there were 15.813.245 unknown values in total. Due to this high uncertainty and also the portion of the variants that were rejected by the Pickard liftover tool, a decision was made to change the parameters and repeat the annotation using the old chromosome positions, in alignment with the previous version of the databases. This way, by eliminating data of rejected variants, the high number of unknown values resulting from discrepancies among versions and position mismatches would have been avoided.
To perform the annotation again, the following parameters were used:

protocols="gnomad211_genome, gnomad211_exome, clinvar_20221231, dbnsfp42a, avsnp150, refGene"

Here, gnomAD version 2.1.1 datasets for annotating variants found in the genome and exome were used. The reason for using both genome and exome protocols was to ensure validation and gain insights into variant impact across diverse genomic contexts by comparing annotations. This approach helped to identify similarities and discrepancies in annotations, that might reveal insights. However, it also produced some extra columns providing the variant frequency in different populations, which will constitute features for later tasks such as supervised learning. Therefore it was needed to compare these two annotations with genome and exome which produced 18 columns each (36 columns in total), and only keep the significant ones, which are exampled in section 5.3.5, for the later tasks. At the end of the annotation of 867 files, there were 17.853.670 unknown values in total. Notably, that value was decreased compared to the previous annotation because of the reasons explained above, the possible position mismatches and version discrepancies. Also, the rejected variants by the Picard tool were able to be used this way, meaning more data available to conduct the study. There were 170 columns containing different types of information about the variants, which is 18 columns more than the output of the previous annotation. These columns were the annotations of the exonic regions, since "gnomad211_exome" was added to the protocols. Obtained annotated files included the following information about the variants, the total number is given in parentheses:

- Information related to chromosome position and nucleotide change: Chr, Start, End, Ref, Alt, (5).
- gnomAD allele frequencies of genome and exome: gAF, gAF_popmax, gAF_male, gAF_female, gAF_raw, gAF_afr, gAF_sas, gAF_amr, gAF_eas, gAF_nfe, gAF_fin, gAF_asj, gAF_oth, gnon_topmed_AF_popmax, gnon_neuro_AF_popmax,

gnon_cancer_AF_popmax, gcontrols_AF_popmax, eAF, eAF_popmax, eAF_male, eAF_female, eAF_raw, eAF_afr, eAF_sas, eAF_amr, eAF_eas, eAF_nfe, eAF_fin, eAF_asj, eAF_oth, enon_topmed_AF_popmax, enon_neuro_AF_popmax, enon_cancer_AF_popmax, econtrols_AF_popmax, (36).

- Clinical implications and significance: CLNALLELEID, CLNDN, CLNDISDB, CLNREVSTAT, CLNSIG, (5).

- Gene type, genetic function variant id and type: avsnp150, Func.refGene, Gene.refGene, GeneDetail.refGene, ExonicFunc.refGene, AAChange.refGene, (6).

- Tools related to conversation, pathogenicity, protein function impact and variant impact: SIFT, SIFT4G, Polyphen2_HDIV, Polyphen2_HVAR, LRT, MutationTaster, MutationAssessor, FATHMM, PROVEAN, VEST4, MetaSVM, MetaLR, MetaRNN, M-CAP, REVEL, MutPred, MVP, MPC, PrimateAI, DEOGEN2, BayesDel, ClinPred, LIST-S2, Aloft_Confidence, CADD_raw, DANN, fathmm-MKL, Eigen, Eigen-PC, GenoCanyon, integrated_fitCons, LINSIGHT, GERP_NR, GERP_RS, phyloP30way, pphastCons30way, SiPhy_29way, (103). (Each predictor tool produced more than one column, all of them are not given here.)

- Reading quality, protein domains, expression: Interpro_domain, GTEx, and other 13 columns related to identity, (15).

To improve traceability, the files were arranged according to patient codes supplied by BioComputing UP. Then they were categorized into training and test datasets. Recently annotated data from CAGI5 and CAGI6 VCF files made up the training dataset which added up to 565 files. The remaining 302 files, on the other hand, were identified as the test dataset. Files in the test set with variants that did not fit certain requirements, such as those having a read (DP) value of 0 that indicates a sequencing error, went through the appropriate modifications to guarantee data integrity and made ready for the next processes.

In the end, there were annotated VCF files for 867 patients, each containing 170 columns of information about variants. Each row in the files represented a variant and each file had a different number of variants, which is rational due to genetic diversity among individuals. Of course, a significant part of these variants were shared, since many of these genetic variants were neutral and widespread in a wide population. This phenomenon is explained by neutral drift, where certain variants have no important impact on individuals and, as a result, can become widespread within a population. Neutral drift represents a random fluctuation of allele frequencies across generations [11]. While not all patients within the cohort received a definitive diagnosis for every disease, the outline of disease distribution among individuals is provided below. This data is illustrated through a bar graph representation in Figure 4.2. It's important to note that lacking labels for the patient phenotypes contributes to the complexity of the task, especially when performing supervised learning.

- ID: 497 patients, diagnosed out of 511 with available data.
- ASD: 301 patients, diagnosed out of 487 with available data.
- Epilepsy: 139 patients, diagnosed out of 487 with available data.
- Microcephaly: 63 patients, diagnosed out of 421 with available data.
- Macrocephaly: 59 patients, diagnosed out of 421 with available data.
- Hypotonia: 99 patients, diagnosed out of 393 with available data.
- Ataxia: 41 patients, diagnosed out of 369 with available data.

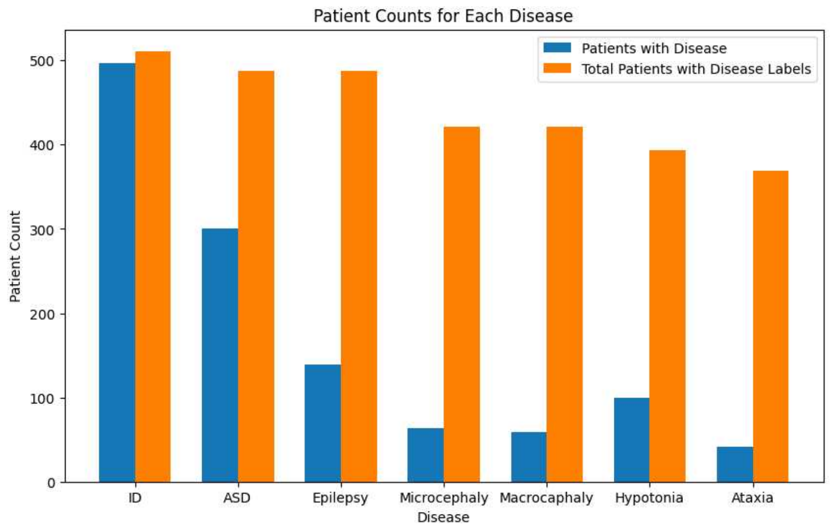
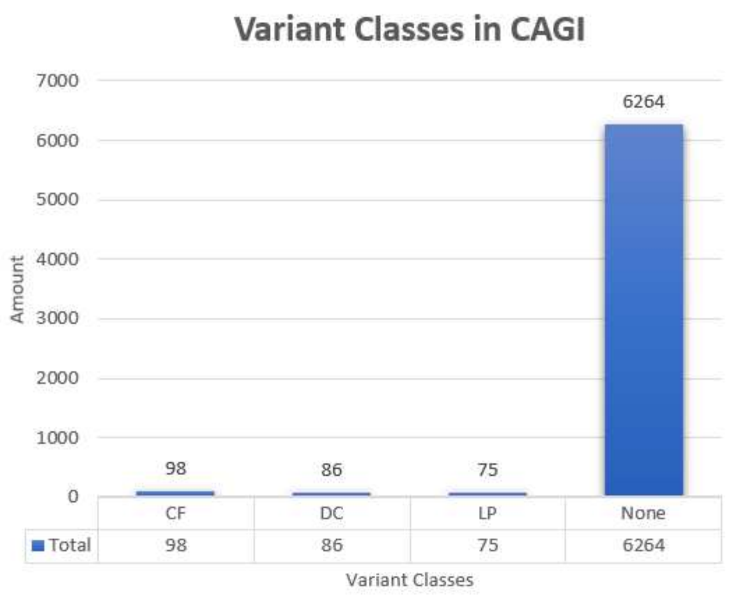


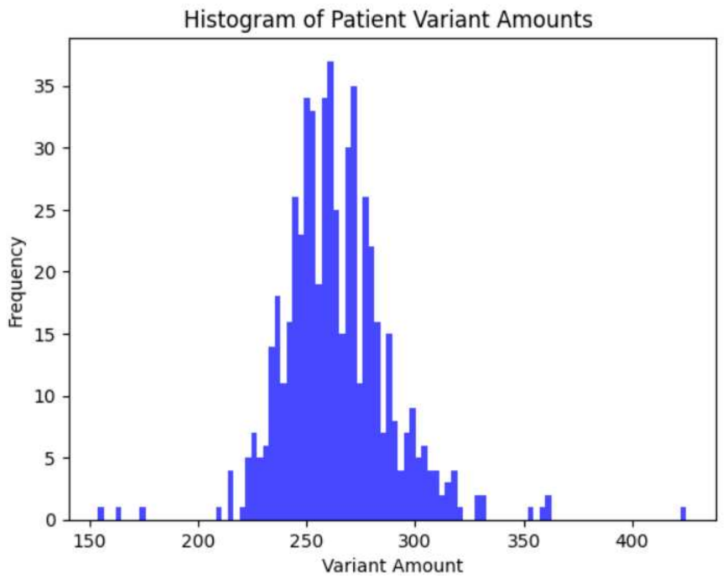
Figure 4.2: Patients with labels and with diseases

As for the variants, BioComputing UP provided the selected variants, including DC, LP, and CF labels. In the patient files, there were 6.523 unique variants among 240.474, since the variants tend to be common in the population, as explained before. Below, the distribution of the number of variants per patient and the classes are illustrated. As observed, there is a significant imbalance in both the distribution of phenotype classes and variant classes. This imbalance made the categorization more complex, which led to the implementation of several



	CF	DC	LP	None
■ Total	98	86	75	6264

(a) Variant types of unique variants



(b) Variant amounts of patients

Figure 4.3: Variant type and variant amount distribution

experiments using various techniques. Supervised machine learning was given priority due to the presence of labeled data, but unsupervised techniques were also employed. The experiments will be thoroughly discussed in the following chapter, which will also share an analysis of the approaches taken to deal with the complexities caused by the unbalanced data.

5 Methodology And Workflow

This chapter covers the initial approaches used to prepare the data, exploratory analyses to understand the dataset, and the experiments that were carried out for a successful classification of variants and phenotypes.

5.1 Initial Data Analysis

In the early stages of this study, an examination of the dataset was conducted, such as variance analysis and exploration of missing values, to have insights into the completeness and quality of the data, guiding subsequent steps in data preprocessing. **Python 3** was used in the Google Colab environment, because of its ease of accessing Google Drive files and GPU support, which is important for high throughput processes and parallel calculations, especially on tensor operations and machine learning models.

Initial data analysis (IDA) is a crucial step in the data exploration process, including preliminary tasks to understand the characteristics and structure of the dataset before applying other modeling techniques. The main goals are identifying potential issues and to be able to create a proper data pipeline [12]. Some common components of initial data analysis include: calculating descriptive statistics, data visualization, data cleaning, correlation analysis, variance analysis, identifying data types, etc. Details of the work regarding these points are explained in the following sections.

- Identifying Data Types: Most variables (columns) consisted of numeric values represented as float data types, while a minority (44) were string values containing information about the variants. Proper handling (either elimination or encoding) of the string values was crucial to facilitate the use of classification models, as these models typically cannot process string values directly.

- Calculating Descriptive Statistics: The data coming from the annotation files comprised 170 columns, each containing a different kind of value as explained previously. Some statistical values are calculated for each column, such as minimum, maximum, variance, and mean to understand tendencies and variability (for the numeric variables). For most of the numeric variables (119), the minimum value was 0 and the maximum was 1. This made sense since in an annotation, most of the columns are related to different predictor scores, coverage or frequency. These values are naturally between the interval of [0, 1]. Remaining numeric values (9), had a minimum value of 0 and theoretically could have all positive numbers. These variables belonged to chromosome positions, number of reads, quality score, and other countable values.

- Variance Analysis: The variance of numerical columns was calculated to understand the distribution and variability putting light on the significance of individual attributes. The variance values were low, the maximum being 0.27. This indicated that the feature values are relatively close to each other, and there is not much diversity. Also, the presence of redundant information can explain the low variance of features, which led to an importance analysis and feature selection. In addition, considering that the values were in the interval of [0, 1], it is expected to have low variance values. Of course, the biggest variance value belonged to the features that were not in the constrained range of [0, 1], but they were scaled to have a consistent machine learning model. It's important to highlight that the presence of repeated unknown values throughout the dataset can reduce the significance of the variance within a feature.

- Correlation Analysis: A heatmap was generated to see the correlations between each feature, aiming to understand the underlying relationships and determine the minimum number of features necessary to explain the target variable. Although the visualization was poor due to the high number of features, it was noticed that 19 features, originating from various prediction tools, exhibit a substantial correlation level (above 76%). This observation was reasonable, considering that all predictors in the annotation process utilize similar parameters and rely on publicly available databases to predict the pathogenicity or impact of the variants.

- Missing values analysis: It is crucial to explore and handle properly the missing values in the data commonly denoted as NaN (Not a Number) before proceeding to experiments. An analysis was made throughout the data files and it was noticed that the missing data was represented with a dot (".") or set of dots (".;.;.;.;."). These missing values indicated

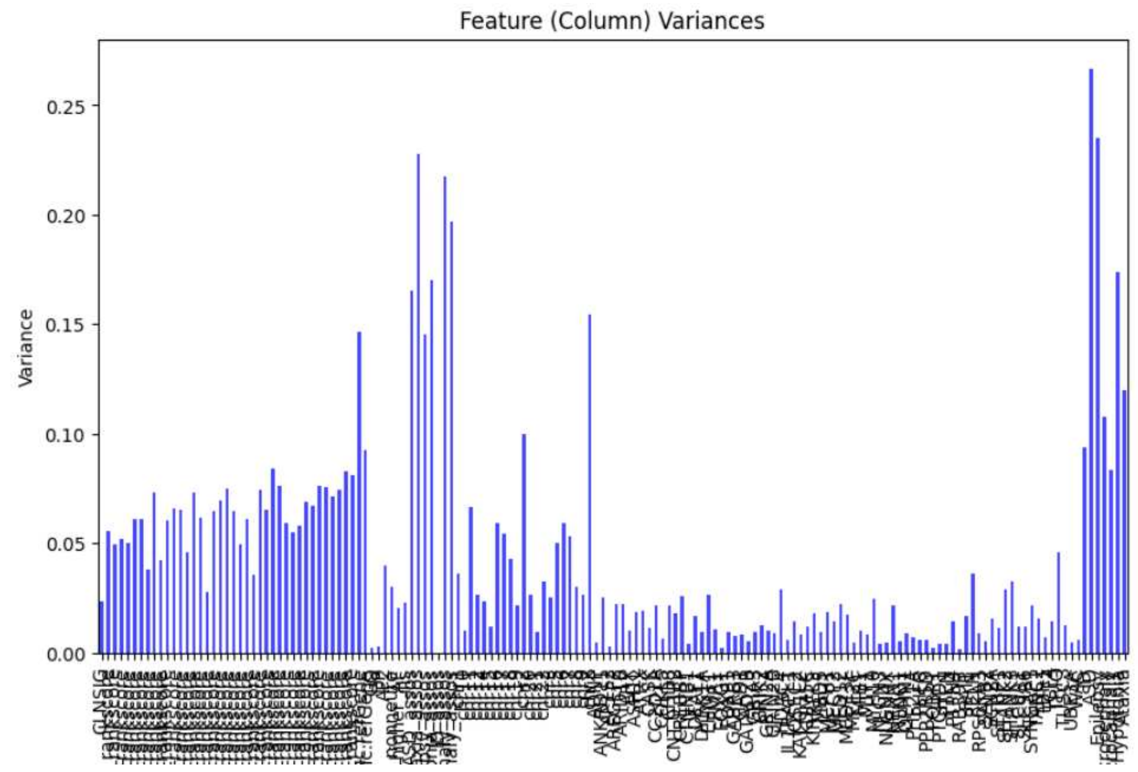


Figure 5.1: Variance values of the dataset columns

that the annotation process could not retrieve the values from public databases related to those variants, this occurred especially for de novo variants, intronic variants whose effects were difficult to predict, and variants that did not exist in the databases. There were 232.327 missing values in the dataframe of unique variants (6.523). An illustration can be seen in Figure 5.2. For ease of visualization, the columns without missing values were not shown.

5.2 Data Preparation & Cleaning

5.2.1 Handling Missing Values

There are several methods to handle missing values such as removing, imputation, forward fill, estimation, etc. For this case, removing the samples with missing values would not work since almost each row had at least one missing value. imputation techniques were chosen for this

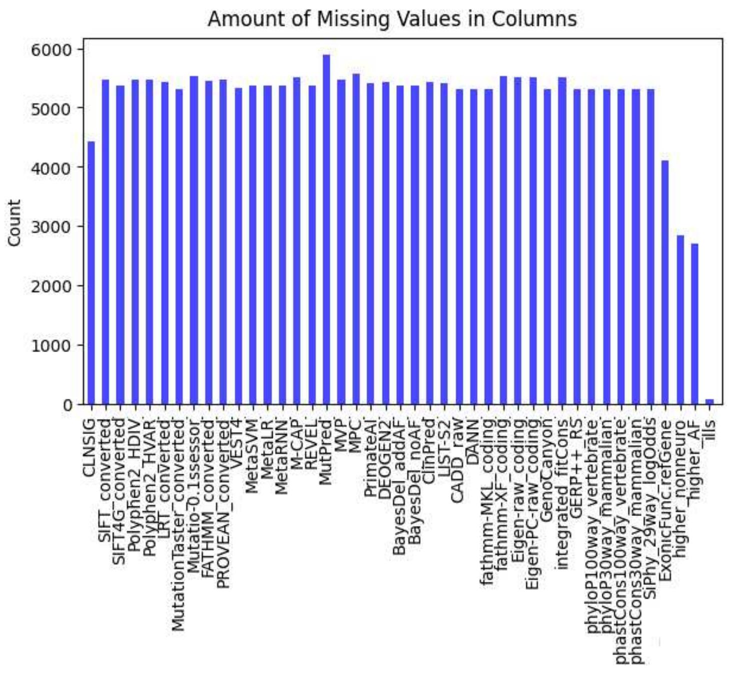


Figure 5.2: Amount of missing values within the features

task to preserve the Sample Size, improve Statistical Power, and enhance robustness. Imputation techniques include filling in missing values with a suitable replacement. Common methods are using mean, median, or mode for numerical data, using the most frequent category for categorical data, or using another value to represent missing values. For the data at hand, the experiments with the mean and median of the columns were made. A random forest model was built to monitor the prediction accuracy and other scores in order to determine the best approach. Mean caused a lot of bias towards the class of "None" which are the variants not belonging to any category (disease-causing, likely pathogenic, contributing). This is because of the abundance of these categories, which has been shown before in Figure 4.3a. The model failed to predict likely pathogenic cases. Also, undersampling methods were employed by randomly selecting some examples from the "None" class to reduce the amount. The model was rerun to compare the results. With the median, there was less bias towards the "None" class, yet, the prediction for likely pathogenic variants was not satisfactory when accuracy, precision, and F1 score were checked. Next, a constant value of "-0.1" was used for the replacement which was outside the range that the missing values normally had. This approach was chosen based on several beneficial factors, as outlined below:

- Identifiability: As it stands out when a constant value outside the range is assigned to missing data, it allows to account for missing values explicitly.

- Preserving information: While imputing with a constant may not accurately represent the true values of the missing data, it helps in preserving the fact that data is missing. This is important for the models to understand the patterns when the nature of the biological data is considered.

- Minimization of bias: The presence of missing data could give some clues about the variants, especially if the reason is related to a specific condition or attribute. Assigning a constant value outside the normal range can help prevent biases in analyses. The constant serves as a marker, indicating that these values are different from observed values.

- Compatibility: Having a constant value might work better for some models in terms of preventing the imputed values from influencing the model in the wrong direction. Also, it allows testing the impact of the imputation choice on study results by comparing analyses with different imputation strategies.

When the initial Random Forest Model was run again with imputed missing values with a constant value of "-0.1", the accuracy compared to mean and median imputation did not change significantly. However, the F1 Score average was increased by 6%, even higher when the undersampled data was used (8%). This was a desired development since the F1 Score is a harmonic mean of precision and recall. The reason for having limited improvement in accuracy was that the classes comprised a small number of examples, which led to a higher increase in precision and recall with small changes but less increase in accuracy. Therefore this method was selected for handling missing values.

5.2.2 Gene Filtering

During the analysis, the gene filtering step involved filtering out variants that did not correspond to the genes listed in our selected gene panel. This process ensured that only variants directly associated with the genes of interest were retained for further investigation. The variants that were obtained after variant calling naturally included examples of genes that were found to be located outside of the predefined gene list. By systematically eliminating these non-conforming variants, the dataset was refined, so that the focus was on the genetic variations of the scope. The number of unique variants, as mentioned, was dropped to 6.253 after this elimination, allowing further investigations and pre-processing of the data.

5.2.3 Encoding

As mentioned, some of the important columns had string data types storing nucleotide changes, genetic function and location information. These columns needed to be converted into numeric values to be able to perform a proper classification. This process of converting categorical data into a numerical format is called **Encoding** and it is used for machine learning algorithms. Since machine learning models typically work with mathematical operations, converting categorical variables into numerical representations was essential for these models to learn from the data effectively.

There are two main encoding types which include Label Encoding and One-Hot Encoding. Label encoding assigns a unique numerical label to each category. It is suitable for ordinal data where the order matters but is not suitable for nominal data as it may introduce unintended ordinal relationships. One-Hot Encoding, on the other hand, creates binary columns (binary vectors) for each category, representing the presence or absence of that category. Thus, it adds new columns to the data as much as the number of categories. It is suitable for nominal data and avoids introducing ordinal relationships.

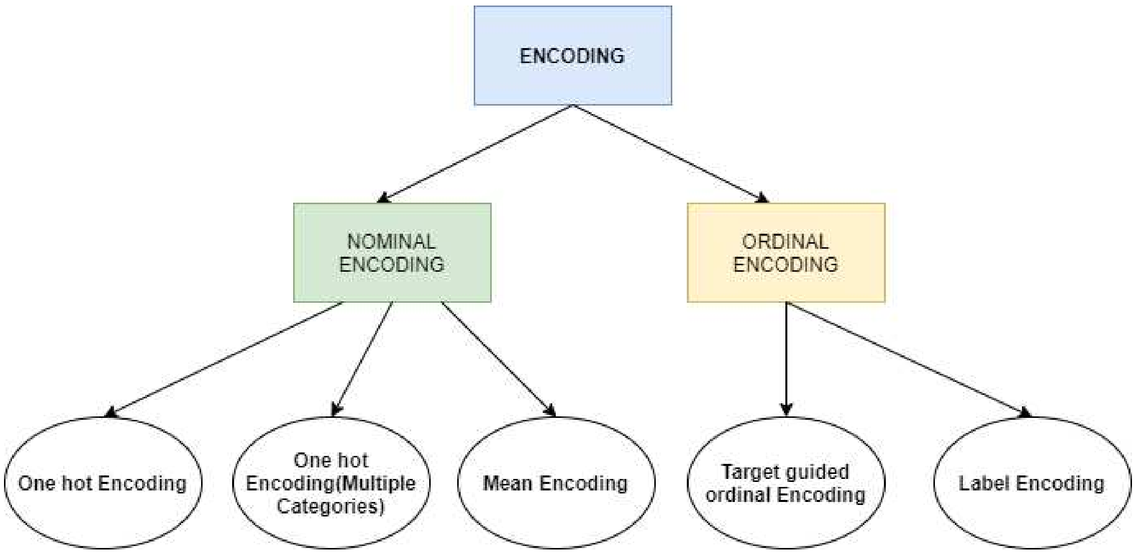


Figure 5.3: Encoding procedures based on data types

For a successful encoding that represents the nature of the data, the columns were treated differently. The columns with values that did not have an ordinal relationship were encoded with One-Hot Encoding, while the others were applied Label Encoding. The columns that represented gene names and chromosomes were one-hot encoded. This caused the dataframe to have additional columns (74 for genes, 21 for chromosomes) increasing dimensionality. Nevertheless, this is a common and appropriate approach when dealing with categorical variables without order, as it allows the model to treat each category independently. The other categorical columns, containing information about clinical annotation, gene region and variant type were label encoded due to the importance of the order in the nature of these attributes. For instance, the clinical annotation could have the following values: 'Benign', 'Benign/Likely benign', 'Likely benign', 'Uncertain significance', 'Conflicting interpretations of pathogenicity', 'Likely pathogenic', 'Pathogenic/Likely pathogenic', 'Pathogenic'. In this case, the variants annotated as benign in ClinVar had a lower possibility of being predicted as pathogenic, and the variants that were reported as Pathogenic were more likely to be related to the diseases of interest. Thus, an encoding was made considering this hierarchy, where benign cases were labeled with "0", uncertain cases were labeled with "0.5", and pathogenic cases were labeled with "1".

As for gene region, the variants in exonic regions were assigned "1", since they have the most impact on the resulting protein. The variants in the splicing region were assigned "0.75", because of their regulatory effects. UTR5, UTR3 and Upstream region categories were assigned "0.5" since they also may have some effects on proteins. Finally, variants in intronic regions were labeled with "0.25" as they rarely have impacts on pathogenicity.

The treatment of the variant type column followed a similar approach, assigning higher values for more critical variants and lower values to the ones with less potential impact. Nonframeshift variants were assigned a value of "0.5", frameshift, stopgain and startloss variants were labeled with "1". And synonymous SNVs were assigned a small value of "0.1" since they normally do not have any impact on the resulting protein but the possibility of being combined with other variations throughout the DNA molecule.
Of course, as it can be noticed, it was made sure that all the encoded values fall within the [0, 1] interval, aiming for consistency and maintaining a standardized scale across the dataset.

5.3 Feature Engineering & Extracting Features

5.3.1 Allele Frequencies In The Population

To have more explanatory data at hand for the task, some features (columns) were added to dataframe since additional relevant features can enhance the predictive power of machine learning models. They provide more information to the algorithm, helping to identify complex patterns and relationships within the data, and facilitating the decision-making process. Also, feature engineering can help balance the dataset, especially when dealing with imbalanced classes. Creating new features that highlight important aspects of minority classes can lead to better model performance.

As mentioned before in Section 4.2, there were 36 columns related to genome and exome allele frequencies obtained from the gnomAD database. This amount was more than needed to explain the relationships, therefore it was necessary to decide the most useful frequencies to discard others. With the suggestion of the supervisor, two allele frequency values were decided to be used: "AF" and "non_neuro_AF_popmax". **"AF"** stands for allele frequency in the entire population, obtained from the gnomAD database. It is a measure of how common a specific version of a gene is within a group of individuals. Allele frequency is often expressed as a proportion or percentage, indicating the fraction of alleles in a population that carry a particular genetic variant. For example, if a certain variant of a gene is present in 10 out of 100 alleles in a population, the allele frequency would be 0.1 or 10%. This value is used to understand the genetic diversity within populations, identify common or rare variants, and investigate the potential association of specific alleles with traits or diseases.

"non_neuro_AF_popmax" on the contrary, refers to the allele frequency of variants in the non-neuro (non-neuropsychiatric) subset of the population. This subset excludes individuals with conditions such as ASD, ID, and other NDDs. GnomAD provides allele frequency information for various subsets, including non-neuro populations. Analyzing allele frequencies in different subpopulations helps researchers and clinicians understand the distribution of genetic variants in specific contexts. In the case of working on NDDs, understanding the allele frequencies in the non-neuro subset provides valuable insights into the genetic outline. It allows for more targeted investigations into variants associated with neurodevelopmental conditions by distinguishing them from variations present in the entire population.

To utilize these two frequency values, a decision had to be made between the frequency in the genome and the frequency in the exome. Upon consultation with the supervisor, it was determined to compare both values and retain the higher one, discarding the other. Prioritizing a higher frequency was considered more significant in this context, given that the focus of the investigation is on smaller frequencies. Therefore the higher value should have been compared with the threshold. In the end, two new columns were added to the dataframe to store these values: "higher_AF" and "higher_nonneuro".

5.3.2 Information Related To Cohort And Quality

In the last 13 columns of the annotated files, there was a comma-separated special set of information regarding variant quality and cohort statistics. The three of them were decided to be useful: “GQ”, “DP” and “AF”.

In the context of variant data, these statistics represent the following properties:

- **GQ** stands for genotype quality, it is a measure of confidence in the assignment of a genotype to a particular individual. Higher GQ values generally indicate higher confidence in the accuracy of the called genotype.

- **DP** represents sequencing depth, which is the number of times a particular nucleotide has been sequenced at a specific position in the genome. It provides information about the coverage and how well a region has been sampled. More DP value indicates a more reliable read.

- **AF**, as in the previous case, represents allele frequency, however, it belongs to the cohort rather than the entire population. In this context, it reflects the presence of the allele within the cohort. To avoid any potential confusion, the extracted column has been designated as ”AF2.”

These parameters were used to assess the quality of sequencing, and the prevalence of specific alleles in the cohort. Of course, rare variants were important for the research since they might have a greater possibility of being disease-causing. Also, a GQ threshold of 10 and a DP threshold of 10 were used to filter the unreliable variants in order to proceed with the meaningful ones. Then, these values of GQ and DP were scaled using “MinMaxScaler” of scikit-learn library, to have a consistent dataframe, since all the other columns contain scaled data.

5.3.3 Genetic Associations With Diseases

In order to provide insights and increase the accuracy of machine learning models, a gene association matrix was constructed. This matrix involved mapping the gene panel with the diseases of interest. To accomplish this task, three public databases were utilized: HPO, SysNDD, and MONDO. Each of these databases contains annotations about diseases, providing standardized and integrated information on clinical features, symptoms, gene associations, and inheritance patterns. The decision to incorporate data from multiple databases was made to ensure the integrity of different databases since they employ different curation methods, information sources, and inclusion criteria. This approach of cross-checking across multiple databases strengthened the evidence of relationships between genes and diseases and also ensured the reliability of gene-disease mappings.

Following data collection from the databases, a gene was considered associated if any evidence was present in at least one database. Then, this information was utilized to construct a matrix comprising 74 genes and 7 diseases. The matrix consisted of binary values, indicating the presence or absence of association. By using this matrix, 7 columns were added to dataframe depending on the gene information of that variant.

5.3.4 Special Cohort Frequency of the Variants

At this step, variant frequency in the cohort was calculated manually by counting the number of recurrences of that variant in the entire patient files. Then this number was divided by 867, which was the total number of patients sequenced. This value has been added to the dataframe as a new column named "ills", representing the number of patients having the variant (infected). The main difference between this value from other frequencies is that this value was unaffected by the genders and inheritance patterns. When considering X chromosome variants, there can be gender-specific implications due to the differences in the number of X chromosomes between males and females. For females, since there are two X chromosomes, variants on the X chromosome follow the same principles as autosomal variants (not belonging to gender chromosomes). The allele frequency is calculated based on the number of occurrences of a specific allele divided by the total number of X chromosomes in the female population. For males, the only X chromosome is inherited from the mother. Therefore, allele frequencies for X chromosome variants in males are influenced by the maternal population. For this reason, in studies involving X chromosome variants, genders and inheritance patterns must be taken into account, since specific calculations may be applied to obtain allele frequencies.

This way, "ills" values satisfied the need to have an independent frequency value that belongs to the cohort. This frequency value has become the fourth column related to frequencies alongside "AF", "AF2", and "non_neuro".

5.3.5 Dropping Columns

The steps and processes explained above created some columns that either were unnecessary or made the other columns redundant/unnecessary. For this reason, each of them had to be assessed and the ones that would not be useful for the purposes of the study had to be eliminated before proceeding to further applications. First of all, the annotation process retrieved some columns that were not able to be used since they did not have informative aspects. As mentioned previously in section 4.2, there were 103 columns generated by the tools related to the conversation, pathogenicity, protein function impact, and variant impact. These tools did not generate only one score but 3 – 6 columns containing different predictions, their confidence, and how they compare to other variants, enhancing versatility, interpretability, and comparison capabilities of the tool's outputs, responding to various needs. Different columns represent different aspects or details of the predicted impact. For example, a tool might provide a continuous score indicating the likelihood of a variant being deleterious. Still, researchers might be interested in a more straightforward prediction or a converted rank score for easier comparison across tools. Sometimes, there are cases when the same tool uses different models. PolyPhen-2 for instance, has separate models for HumDiv (HDIV) and HumVar (HVAR), each with its set of scores and predictions. This allows for a more nuanced assessment of the impacts. To have an idea of different columns and their contents generated in this study, an example is given on the SIFT algorithm and its extended version, SIFT4G:

- SIFT_score: Original SIFT score, which is a normalized probability that a substitution is tolerated.
- SIFT_converted_rankscore: A phred-scaled converted rank score for the SIFT score, often used for ranking.
- SIFT_pred: Classification based on the SIFT score ("D" for damaging, "T" for tolerated).
- SIFT4G_score: Similar to the SIFT score using the SIFT4G algorithm.
- SIFT4G_converted_rankscore: Phred-scaled converted rank score for the SIFT4G score.
- SIFT4G_pred: Similar to SIFT, classification is based on the SIFT4G score.

Depending on the purpose of the study, different levels of detail can be selected for analysis or applications. For this study, rank scores (or converted rank scores when provided) provided by the tools were chosen to be kept since they have the probability values of being deleterious. This approach also satisfied that the columns had the same scale, which was [0, 1]. Among 103 columns of 27 predictor tools, 38 columns were kept in the dataframe. Different models for the same tool were considered to a different scores and were kept. The others were removed due to their redundancy. In the end, 65 columns were eliminated at this step. After the selection of the gnomAD allele frequencies to be used among 36 columns, the other frequency information has become redundant. As explained in Section 5.3.1, "AF" (entire population allele frequency) and "non_neuro_AF_popmax" (allele frequency of variants in the non-neuro subset of the population) were calculated by comparing their genome and exome values, and contained in other columns. Therefore the original 36 columns needed to be removed to ensure the minimum number of features explaining the target variable. Following the extraction of GQ, DP, and AF2 columns, a parallel process was applied to address the 13 columns associated with identity, as mentioned in Section 4.2. These columns included a range of statistics and quality-related data for the variants. Given that only these three values compressed in these columns were considered essential for filtration and analysis, new columns were created to store this information. Then, the original 13 columns were removed to retain only the relevant features. Another column elimination was carried out for the encoded features. As explained in Section 5.2.3, the columns associated with gene names and chromosomes were one-hot encoded. While this process added new columns to the dataframe, it also allowed for the removal of the redundant original columns containing string values. This operation resulted in the elimination of 2 columns. In the final step of feature extraction and data preparation, unnecessary columns such as patient ID, variant number, start and end positions, and nucleotide changes were eliminated. As a result of this process, 115 new columns were added to the initial features, while 123 columns were dropped, resulting in a net reduction of 8 columns. Therefore the number of columns was reduced from 170 to 162.

5.4 Creating Data Structures

After obtaining the necessary features and data preprocessing, different data structures were created containing data in suitable forms for different purposes. Since there were two classification tasks, the data at hand must be in the proper form, aligned with labels, and contained in the most efficient way that machine learning models could use. Different structures are explained in this section, with the reasons of selection.

5.4.1 Variant Classification

For variant classification purposes, the labels provided for CAGI5 and CAGI6 were used. These labels were in the shared clinical files of patients and needed to be extracted and formatted as mentioned in 4.2. They were collected and cleaned of unnecessary information, categorized, and brought together to be aligned with the training data. Then, a Python dataframe was created, by following the previously explained procedures. The steps are not described here as they have been already detailed in the previous sections (i.e. Sections 4 and 5). There were 170 columns and 6.523 unique variants from 867 patient files. All columns had the same scale of [0, 1] and the columns containing string values were encoded. The labels data was merged with this dataframe by using "merge" function from the pandas library.

5.4.2 Phenotype Classification

For phenotype classification, a different data structure was needed since the dataframe for variants had the variant labels and degraded to unique variants. This classification required phenotype labels, therefore patient IDs, and patient-wise representation of the information. Since each patient had a different number of variants and each variant had numerous columns of information, the structure had three dimensions. To construct a comprehensive data structure, a set of steps were followed. The initial step involved establishing correspondences between patient names and file names, important for linking variant information to the correct individuals. The patient correspondences were systematically read from a file containing VCF file names and patient codes provided. Then, a loop iterated through each patient, accessing their files containing variant information. A nested dictionary, named 'patient_data_structure', was used to contain this information. A new nested dictionary was created for each patient, allowing the representation of patients with different numbers of variants. Each line in the files was parsed, identifying column names from the header and associating them with their respective values. This information was encapsulated in a second-level dictionary named 'variant_info_dict'. Each patient's data structure was constructed in such a way that the patient's name served as the primary key, while each variant had a unique secondary key ("variant1," "variant2," etc.). The attributes of each variant, such as predictive scores, genetic information and location were systematically stored in the innermost dictionary. This three-dimensional structure with patients, variants, and variant information provided a coherent representation of the dataset and constituted a basis for phenotype analysis.

For the same purpose, the same feature engineering and data cleaning procedures were applied. Each patient and their associated variants stored in the patient_data_structure were looped and for each variant, GQ, DP and AF were extracted after parsing relative columns. GQ and DP values were scaled between 0 and 1 using minimum and maximum values. Then, the dictionary was updated with the extracted values by adding them to the innermost dictionary, representing the variant information. Patients and variants in the patient_data_structure were looped again to retrieve the values of genome and exome allele frequencies from the variant information for each variant. Genome and exome values of Population allele frequency and Non-neurological allele frequency were compared and the maximum values were stored in two new spaces of the variant_info_dict: ”higher_AF” and ”higher_nonneuro”. A filtering process was performed, specifically focusing on variants associated with certain genes. 74 desired genes of the gene panel were defined and each variant was checked if they belonged to these genes, by comparing with the value of ”Gene.refGene” key. After this step, only the variants with the desired genes remained in the dictionary.

5 of the variant information features as mentioned in section 5.2.3 were encoded with proper technique. Gene names and chromosomes were one-hot encoded, while clinical annotation, gene region, and variant type were label encoded due to the importance of the order in the nature of these attributes. Missing values were treated in the same way as in 5.2.1, a constant value of ”-0.1” was chosen for replacement. This decision was motivated by the need for explicit identifiability, as it helps in preserving the acknowledgment that data is missing. This was a crucial consideration for understanding patterns in the biological data although imputing with a constant may not precisely reflect true values. Moreover, selecting a constant outside the normal range helped minimize biases in the direction of the dominant class. To enhance the interpretability and accuracy of machine learning models, a gene association matrix was created by mapping a gene panel to diseases of interest. HPO, SysNDD, and MONDO databases provided standardized details about diseases, including gene association. This matrix was merged with the variant_info_dict to have the features in the corresponding positions. The decision to integrate data from multiple sources aimed at ensuring integrity and guaranteed the reliability of gene-disease mappings.

The ”ills” values calculated in 5.3.4, were added to the variant_info_dict dictionary, representing the count of patients with a specific variant in the cohort. This frequency value remained independent of gender and inheritance patterns. It became the fourth feature about frequencies, together with ”AF,” ”AF2,” and ”non_neuro”.

After the feature extraction and data manipulation process, redundant features were removed to have the minimum required features. The remaining features of predictor tools after choosing the ones to be used were discarded as exemplified in section 5.3.5. The frequency values belonging to the genome and exome were eliminated after preserving 4 features related to frequency. Additionally, unnecessary features generated during the annotation process, along with those considered irrelevant to the training process, were removed from the dictionary to construct better models.

The conversion of patient data from a dictionary to a 3-dimensional tensor was needed and it was a crucial step in the preparation of data for machine learning models, particularly those based on neural networks. In the machine learning area, tensors are the preferred format for data input due to their compatibility with neural network architectures and efficient handling of multidimensional data. Since the data structure at hand was also three-dimensional, this transformation ensured a consistent input shape across all samples, addressing information on the variants among patients.

In Python, both PyTorch and TensorFlow are prominent deep learning frameworks that provide tensor functionalities for efficient computations, forming the basis of neural network implementations. PyTorch tensors and TensorFlow tensors share similarities, with their data structures, model parameters, and outputs in neural networks. Both frameworks support operations like element-wise addition, multiplication, and linear algebra operations. Tensors in both frameworks represent multidimensional arrays, however, there are notable differences between PyTorch and TensorFlow tensors, each with its advantages.

PyTorch and TensorFlow, have key differences in their approach to graph computation, ecosystem, and syntax. TensorFlow uses a static graph, predetermined before execution, enabling efficient implementation but offering less flexibility during development. In contrast, PyTorch utilizes a dynamic graph, allowing real-time adjustments to the model architecture. While TensorFlow has a broader ecosystem, PyTorch has gained popularity in research due to its intuitive design. PyTorch tensors offer a more ”Pythonic” interface, resembling standard Python syntax. This simplicity helps in readability and ease of use, contributing to faster development and experimentation cycles. PyTorch's dynamic computational graph and more intuitive design have gained popularity among researchers and practitioners, especially in fields requiring frequent model adjustments and experimentation.

In this study, Pytorch Tensors were employed due to the previously mentioned advantages. By adopting tensors for phenotype classification, the data became more efficient in batch processing, leveraging parallel computation. This conversion of the data optimized memory usage, especially in Neural Network experiments. Also, it supported fundamental tensor operations crucial for machine learning. As a result, conversion to a 3D tensor was a foundational data manipulation step, aligning the data with the requirements of machine learning models and enabling the application of advanced techniques to have insights from patient and variant information. With this procedure, two data structures were obtained for two purposes of this study: variant classification and phenotype assignment. Different experiments with these structures and their assessments are explained in the following chapter.

6
Experiments & Results

This chapter introduces the complexities of the models and techniques used for variant classification and phenotypic assignment. In addition to explaining the experimental design, it offers a detailed analysis of the outcomes obtained from these experiments.

6.1 Variant Classification

The constructed dataframe, as explained in section 5.4.1, was used for variant classification purposes, as the input for machine learning models and tools. This task aimed the classification of 6523 variants into 4 classes: Disease-causing (DC), Likely pathogenic (LP), Contributing factor (CF), and Neutral (None). Both supervised learning and unsupervised learning models were applied for classification and detection. This approach was followed to find the models simulating the real-world scenarios in the most realistic way. In clinical cases of the real world, harmful variants are very rare to encounter and they are not numerous in a single individual. Most of the time, the variants of an individual are neutral, not causing any significant life-changing effects or disturbances. Hence, the task was detecting a few potentially damaging variants in a pile of neutral variants. Because of this natural aspect of the variants, the database at hand was highly unbalanced, having 96% of the variants neutral, not disease-causing or contributing. Some methods such as undersampling and feature selection were applied to overcome the drawbacks of this imbalance. Also, different machine learning models, each with distinct strengths and weaknesses, were employed to identify the most effective model and methods for accurate classification. In this section, different models are introduced with information on their working principles and applications.

6.1.1 Random Forests

Random Forest is an ensemble machine learning technique that benefits the strength of multiple decision trees to enhance predictive performance. It is considered as an ensemble method, since it brings the information from different sources together to make a decision, in this case, different decision trees. Each decision tree in the forest is trained independently on a random subset of the original training data, a process known as bootstrapping. This helps introduce diversity among the trees, as they are exposed to different subsets of examples, which, in turn, reduces the risk of overfitting to the peculiarities of any single subset. Random Forest's key feature is the use of feature randomization, at each decision tree node, instead of considering all features for potential splits, a random subset is chosen. This introduces diversity among trees, making each tree a unique learner. The ensemble model aggregates individual tree outcomes during the prediction phase. For classification, it employs majority voting, selecting the class with the most votes across all trees. In regression, it averages tree predictions. This approach enhances the model's robustness and accuracy by mitigating the impact of outliers or noise in the data, contributing to improved generalization and overall performance.

Random Forests are known for their ability to handle high-dimensional datasets, capturing complex relationships between features and the target variable. Additionally, they provide a natural mechanism for assessing feature importance by analyzing the impact of each feature's contribution to the overall predictive performance. These advantages made Random Forest a suitable model for variant classification since the data is unbalanced and high-dimensional. Random Forests were the models that were used the most for CAGI5 and CAGI6 challenges, among the predictor groups [2], [6].

Basic Random Forest

For starters, a basic random forest model was run with default parameters by using RandomForestClassifier, which resides in the ensemble module of scikit-learn library in Python 3. Scikit-learn library is a popular machine learning library for performing various models easily and efficiently. RandomForestClassifier is a class provided by the scikit-learn and enables the creation, training, and evaluation of Random Forest models. This class has default parameters initially defined to train models to facilitate the training process, which can be adjusted to handle different scenarios. Firstly, the model was run with the following default hyperparameters:

RandomForestClassifier(n_estimators=100, max_depth=None, min_samples_split=2, min_samples_leaf=1, random_state=None)[source]

The explanations of these values and their effects on the model are given below:

- n_estimators: The number of decision trees in the forest.
- max_depth: The maximum depth of a tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.
- min_samples_split: The minimum number of samples required to split an internal node.
- min_samples_leaf: The minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least min_samples_leaf training samples in each of the left and right branches. This may have the effect of smoothing the model, especially in regression.
- max_features: The number of features to consider when looking for the best split.
- random_state: Controls both the randomness of the bootstrapping of the samples used when building trees and the sampling of the features to consider when looking for the best split at each node.

The dataframe was given as input to the model with the default hyperparameter values. First, feature columns and labels were separated, then the data was split into training and testing sets. According to this, 20% of the data (1.305 examples) was separated to be the test set. The process of splitting a dataset into training and test sets is a fundamental practice in machine learning, crucial for model evaluation and generalization assessment. The training set is utilized to teach the model by exposing it to patterns and relationships within the data, allowing it to adjust its parameters accordingly. On the other hand, the test set remains unseen during the model training phase, serving as an independent benchmark for evaluating the model's performance in unfamiliar instances. This practice aims to ensure that the model can generalize well to new, real-world data, preventing overfitting where the model may memorize peculiarities of the training data rather than learning underlying patterns. By systematically dividing the data in this manner, researchers can confidently assess the model's predictive capabilities and make informed decisions about its implementation in diverse scenarios.

Next, the random forest classifier was initialized and trained on the training set. Predictions were generated for both the training and testing sets, and accuracy metrics were computed to evaluate the model's performance. Additionally, a detailed classification report, providing precision, recall, and F1-score for each class, was printed for the test set. The feature importances determined by the Random Forest model were revealed to have an idea of the feature importances individually. Also, ROC curves were drawn and the areas under these curves were calculated (AUC). The following paragraphs explain the performance metrics used in this study, as they are repeatedly mentioned, offering crucial insights into how the model was assessed:

- Accuracy: Accuracy is a measure of how many predictions the model gets correct out of the total predictions. It is calculated as the ratio of correctly predicted instances to the total instances. While it provides an overall sense of model performance, it might not be suitable for imbalanced datasets where one class dominates.

- Precision: Precision is the proportion of true positive predictions out of the instances predicted as positive. It focuses on the accuracy of positive predictions and is calculated as the ratio of true positives to the sum of true positives and false positives. High precision indicates a low false positive rate.

- Recall (Sensitivity or True Positive Rate): Recall measures the ability of the model to capture all positive instances. It is calculated as the ratio of true positives to the sum of true positives and false negatives. A high recall value indicates that the model effectively identifies most of the actual positive instances.

- F1 Score: The F1 score is the harmonic mean of precision and recall. It provides a balance between precision and recall, particularly useful when there is an uneven class distribution.

- ROC AUC (Receiver Operating Characteristic Area Under the Curve): ROC AUC is a metric used to evaluate the performance of binary classification models. The ROC curve is a graphical representation of the trade-off between the true positive rate (sensitivity) and the false positive rate. The AUC represents the area under this curve. A higher AUC value indicates a better model for distinguishing between positive and negative instances.

The model reached an accuracy value of 97.32%, however, despite this high accuracy value, precision, recall, and F1-scores needed attention, particularly for the minority classes, since the dataset was highly unbalanced. For CF, the model achieved a precision of 88%, indicating that when it predicts CF, it is correct 88% of the time. Similarly, for DC class, the precision was 85%, indicating a high precision. However, for LP, the precision was way lower at 33%, suggesting that the model is less accurate when predicting LP class.

The recall for the None class was unrealistically high at 100%, indicating that the model effectively identified instances of this class. However, for CF, DC, and LP, the recall values were way lower, 7% for LP, suggesting that the model was weak for predicting instances from these classes. Also, F1-scores of the minority classes were notably low, compared to exceptional accuracy value, with an instance of 11% for LP, indicating challenges in achieving a balance between precision and recall for these classes. This overfitting was expected since the nature of the data was known. Having a highly unbalanced dataset, supported by the outcome of the initial model with low F1 scores pointed out the need for improvement of the results by refining the predictions. In addition to improving scores, since a very high accuracy value indicated overfitting, it was necessary to find a balance among the scores, to better generalize the model for different scenarios. For this reason, for future experiments, the F1-score and accuracy values were focused on for improvement.

Classification Report	Precision	Recall	F1-Score	Support
CF	0.88	0.41	0.56	17
DC	0.85	0.58	0.69	19
LP	0.33	0.07	0.11	15
None	0.98	1.00	0.99	1254
Macro Avg	0.76	0.51	0.59	1305
Weighted Avg	0.97	0.97	0.97	1305
Overall Accuracy	0.97			

Table 6.1: Classification Report for Basic Random Forest

Grid Search Hyperparameter Tuning

After trying a basic model, a Grid Search approach was followed to adjust the hyperparameters of the Random Forest model to find the best parameters for optimizing the model. **Grid Search** is a systematic hyperparameter optimization technique widely employed in machine learning to tune the performance of models. It involves creating a predefined grid of hyperparameter values, where each combination represents a unique configuration for the model. The grid search algorithm then systematically evaluates the model's performance using cross-validation for each hyperparameter combination, enabling the identification of the set of values that yield the optimal performance. By exhaustively searching the hyperparameter space, grid search helps discover the optimal configuration for a given machine learning algorithm, ensuring that the model is tuned to achieve the best possible results. Despite its computational cost due to the exploration of multiple combinations, grid search provides a transparent and methodical approach for selecting hyperparameters, contributing to the reproducibility and robustness of the model. For the parameters n_estimators, max_depth, min_samples_split, and min_samples_leaf, a grid of values was defined and the model was run with all possible combinations. random_state parameter remained the same for comparability since it influences the randomness and reproducibility of the model. By keeping the random_state constant, it was ensured that the random processes within the model were consistent across multiple runs. This was important for comparability, as it allowed to make a fair comparison between different models or experimental conditions. Without fixing the random_state, variations in the random processes could lead to differences in results, making it challenging to interpret the results or adjust the parameters.

The parameter grid used to configure the model is given below:

- n_estimators: [10, 25, 50, 100],
- max_depth: [10, 15, 20, 25, 30, 35, 40, 45, 50],
- min_samples_split: [2, 5, 10],
- min_samples_leaf: [1, 2, 4],
- max_features: ['auto', 'sqrt', 'log2']

After running the model with all combinations, the grid search, the best combination of the parameters was obtained as follows:

```
Best Parameters: {'max_depth': 30, 'max_features': 'auto',
'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 10}
```

The resulting classification report when these parameters were used can be seen in Table 6.2. Building upon the results of the Grid Search, the optimized Random Forest model, incorporating the identified optimal hyperparameters, was used for further experiments regarding Random Forests.

After the Grid Search, It was observed that the improvement in the precision and recall scores caused an increase of 4% in the F1 score, making the macro average 63%. The metrics of the LP class were significantly increased while maintaining similar scores for other classes with a precision of 40%, recall of 13%, and F1 score of 20%. The overall accuracy climbed up to 98%

which pointed out overfitting once again, meaning that there was a huge impact of the 'None' class, which was highly unbalanced. In the following section, it was attempted to overcome this issue with feature selection methods. As a result, the Grid Search approach did not only help to increase the performance of the model but also provided the optimal hyperparameters for the later experiments.

Classification Report	Precision	Recall	F1-Score	Support
CF	0.62	0.47	0.53	17
DC	0.88	0.74	0.80	19
LP	0.40	0.13	0.20	15
None	0.98	1.00	0.99	1254
Macro Avg	0.72	0.58	0.63	1305
Weighted Avg	0.97	0.98	0.97	1305
Overall Accuracy	0.98			

Table 6.2: Classification Report for Random Forest After Grid Search

Feature Selection Methods

Feature selection is a crucial step in the machine learning pipeline that involves choosing a subset of relevant features or variables from the original set of features. The goal is to increase model performance, reduce computational complexity, and improve interpretability by excluding irrelevant or redundant features. Feature selection is particularly beneficial when dealing with high-dimensional data where not all features contribute equally to the predictive power of the model, such as in the case of variant classification. High-dimensional data indicates that overfitting and increased processing demands might result from the exponential expansion in large feature spaces. Furthermore, performance considerations highlight the influence of irrelevant features, which can create noise or prevent the model from identifying significant patterns. The predicted accuracy of the model can be improved by carefully selecting only the most pertinent features.

Since there were 170 columns of data in the dataframe, a selection process was needed to achieve higher performance. Some statistical and built-in methods were integrated with Random Forest and later machine learning models to identify and retain the most informative attributes from the dataset. The methods are described as follows:

1. **SelectKBest with ANOVA F-statistic:**

   - The **ANOVA F-statistic**, accessible through the **f_classif** scoring function, is an essential tool when dealing with classification tasks. It evaluates whether the means of different groups in our data are significantly different. In the case of feature selection, f_classif computes an F-statistic for each feature, assessing how much its mean varies across different classes. A higher F-statistic indicates that the feature is more discriminative and is thus prioritized in the feature selection process.
   - The **SelectKBest** method focuses on retaining the top k features based on their scores. In conjunction with the ANOVA F-statistic ('f_classif'), this approach evaluates the significance of differences in means among different classes for each feature. By selecting the most discriminative features through their F-statistic scores, the most relevant attributes are aimed to be retrieved. The choice of k enables to adjustment of the dataset dimension manually.

2. **SelectKBest with Mutual Information:**

   - **Mutual information** is a powerful metric for capturing both linear and non-linear relationships between variables. When coupled with the SelectKBest method, it identifies the top k features based on their mutual information scores with the target variable. This approach is particularly beneficial when dealing with complex and non-linear dependencies, providing a perspective on feature relevance. The selection of the optimal k allows us to strike a balance between informativeness and model complexity.

3. **Random Forest Feature Importances:**

   - **rf_classifier.feature_importances_** is an attribute provided by scikit-learn's RandomForestClassifier class. It represents the feature importances assigned to each feature in the dataset after the model during the training phase. The Random Forest algorithm, composed of an ensemble of decision trees, assesses the contribution of each feature by evaluating metrics such as Gini impurity or information gain in each decision tree. The resulting feature importances are then aggregated and normalized for meaningful comparison, revealing the varying degrees of influence each feature holds. This information enables prioritizing the most influential features of the Random Forest model.

These methods were employed and the Random Forest model was run with the same optimal parameters for each method. For SelectKBest method with f_classif, the threshold was set to 65 features due to its performance, after many trials. 110 features also yielded similar results, however, the focus was reducing the number of features with a tolerable sacrifice on accuracy. With the most important 65 features selected by this method, the same accuracy and F1 score average were obtained despite the different precision and recall values. The predictions were more accurate on the LP class but less accuracy was obtained for the other classes. Although this method did not improve the results, it gave an idea of the feature importances.

The mutual information method, on the other hand, produced a similar result to the f_classif method, but with increasing accuracy and F1 score. 60 features were chosen after many trials and the model was trained. An accuracy level of 98% and an F1 score of 66% was reached. This indicated that the mutual information method was better for selecting important features, however, it was more tended to cause overfitting by being affected by the dominant class.

The random forest feature importance method gave the best result in terms of F1 score but had the lowest accuracy. Only the top 80 important features were selected and when the model was run, more balanced predictions were made. This method caused a decrease in the scores of the 'None' class while increasing the other class metrics. Overall accuracy was 76%, which was a sign of avoiding overfitting, and 69% of the F1 score was obtained, indicating a more balanced prediction among the classes.

Undersampling

Imbalanced data refers to a situation in a dataset where the distribution of classes is not approximately equal when some classes significantly outnumber the others. In this case, the dataset with genetic variants had a very limited number of causative variants or contributing factors, allowing the dominance of the 'None' class.

Imbalanced datasets pose significant challenges to machine learning models, such as biased model training, poor generalization, and potentially misleading evaluation metrics. When models are trained on such datasets, they may exhibit bias towards the majority class, diminishing their ability to effectively predict instances from the minority class. Additionally, the consequences extend to poor generalization, where models struggle to perform well on new and unseen data, particularly for the underrepresented minority class. Traditional evaluation metrics like accuracy become misleading in such scenarios, as a model could achieve a high accuracy score by simply favoring the majority class while failing to adequately identify instances from the minority class. As a solution, employing alternative metrics and data balancing techniques becomes crucial for ensuring fair and effective model assessment on imbalanced datasets.

Undersampling is a technique used to address the issue of imbalanced datasets. It involves reducing the number of instances in the over-represented class to make the class distribution more balanced. This is typically done by randomly removing instances from the majority class until the class distribution is more equal. While undersampling can help balance the dataset, it comes with the risk of losing valuable information from the majority class.

At this step, a random undersampling method was integrated into the model with the selected features by different methods. SelectKBest, mutual information, and random forest feature importance methods were employed for the experiment. Different random state parameters were used for choosing different samples to perform cross-validation to find the most representative random sample for the majority class: 'None'. Also, the number of samples drawn from the majority class was experimented by gradually increasing to see the effect of the imbalance. The experiment with the best result was the random forest feature importance method, selecting the top 100 features, a random forest with 13 estimators, and 50 max_depth value. There were 200 samples drawn from the majority class, and the rest of the random forest parameters were the same. The accuracy was increased to 84% and the F1 score had an average value of 80%, the best of all experiments with the random forest. This improvement was a result of creating a more balanced prediction, diminishing the dominance of the majority class. The predictions of the other classes had higher precision and recall scores, notably the 'LP' class, with a precision of 73%. The classification report is seen in Table 6.3.

Classification Report	Precision	Recall	F1-Score	Support
CF	0.70	0.95	0.81	20
DC	0.93	0.82	0.87	17
LP	0.73	0.50	0.59	16
None	0.92	0.92	0.92	39
Macro Avg	0.82	0.80	0.80	92
Weighted Avg	0.84	0.84	0.83	92
Overall Accuracy	0.84			

Table 6.3: Classification Report for Random Forest with Selected Features and Undersampling

This approach yielded better results and overcame overfitting. The ROC graph can be seen in Figure 6.1. However, it must be noted that while preventing model bias towards the majority class, undersampling comes with drawbacks. The major concern is the potential loss of valuable information from the majority class, impacting the model's ability to generalize effectively. In real-world scenarios of genetic variants, the causative diseases are very rare, both among the

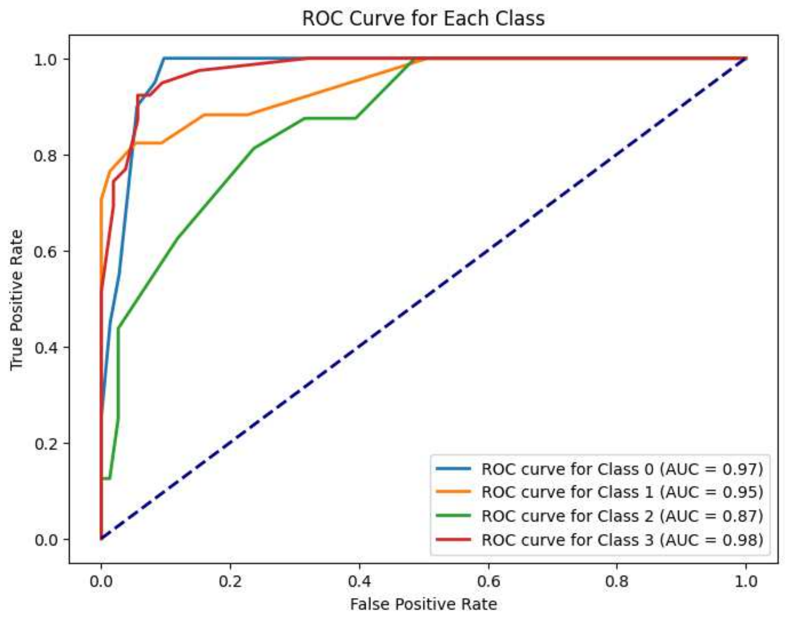


Figure 6.1: ROC Curves and AUC Values of the Final Random Forest Model

variants of an individual and among the population. Therefore the undersampled data does not point out a realistic representation. Additionally, the reduction in dataset size may lead to decreased diversity and increased susceptibility to noise and outliers, compromising the model's robustness.

6.1.2 Logistic Regression

Logistic Regression is a widely employed statistical method and a fundamental tool in machine learning for binary and multiclass classification tasks. Despite its name, logistic regression is used for classification, not regression. It is designed to predict the probability that an instance belongs to a particular category. The core idea is modeling the relationship between the input features and the outcome using the logistic function, also known as the sigmoid function. The logistic function transforms a linear combination of input features into a range between 0 and 1, producing a probability value. Mathematically, it is expressed as:

$$P(Y = 1) = \frac{1}{1 + e^{-(b_0 + b_1 X_1 + \ldots + b_n X_n)}}$$

where $P(Y = 1)$ is the probability of the positive class, e is the base of the natural logarithm, and $b_0, b_1, \ldots, b_n$ are the coefficients learned during the training phase.

Training a Logistic Regression model involves finding the optimal values for these coefficients. This is typically done using optimization algorithms like gradient descent, which iteratively adjusts the coefficients to maximize the likelihood of observing the actual class labels in the training data. Regularization techniques may also be applied to prevent overfitting. Once trained, the model can make predictions on new data by estimating the probability of belonging to the positive class.

The interpretability of Logistic Regression is one of its strengths. The coefficients associated with each feature provide insights into the impact of those features on the classification outcome. In the context of variant classification for genetic data, Logistic Regression offers a transparent and interpretable approach, making it an essential tool for understanding the influence of different genetic features on the likelihood of specific variant classes.

Basic Logistic Regression

In Python, the logistic regression classifier is available in various machine learning libraries, with scikit-learn being one of the most popular ones. In scikit-learn, the logistic regression classifier is implemented as a class called LogisticRegression within the linear_model module. Similar to the Random Forest classifier, this class has default parameters initially defined to train models to facilitate the training process, which can be adjusted to handle different scenarios. Firstly, the model was run with the following default hyperparameters:

LogisticRegression(penalty='l2', C=1.0, solver='lbfgs', max_iter=100)[source]

Here are the explanations of these values and their effect on the model:

- penalty: Specifies the norm of the penalty (regularization). L1 and L2 stand for regression penalty terms, if this parameter is None, no term will be added.
- C: Inverse of regularization strength. Like in support vector machines, smaller values specify stronger regularization.
- solver: Algorithm to use in the optimization problem.
- max_iter: Maximum number of iterations taken for the solvers to converge.

Here, there is a need to explain regularization, to understand the effects on logistic regression and further utilized methods in this study. **Regularization** is a technique used in machine learning to prevent overfitting and improve the generalization performance of models. It involves adding a penalty term to the cost function, which discourages the model from learning overly complex patterns that may not generalize well to new, unseen data.

In logistic regression, two common types of regularization are L1 regularization (Lasso) and L2 regularization (Ridge).

1. **L1 Regularization (Lasso):**
   L1 regularization adds the sum of the absolute values of the coefficients to the cost function. It is expressed as the following penalty term: $\lambda \sum_{i=1}^{n} |w_i|$, where w_i are the coefficients, and λ is the regularization strength. L1 regularization tends to yield sparse weight vectors. It encourages some of the coefficients to become exactly zero, effectively performing feature selection. This property makes L1 regularization useful when dealing with datasets with a large number of features, as it can automatically select a subset of the most important features.

2. **L2 Regularization (Ridge):**
   L2 regularization adds the sum of the squared values of the coefficients to the cost function. It is expressed as the following penalty term: $\lambda \sum_{i=1}^{n} w_i^2$, where w_i are the coefficients, and λ is the regularization strength. L2 regularization penalizes large weights but doesn't force them to be exactly zero. It tends to distribute the penalty across all features, leading to a more even reduction of the impact of less influential features. L2 regularization is effective when all features are potentially relevant to the prediction task.

The regularization strength (λ) is a hyperparameter that controls the amount of regularization applied. A higher λ increases the regularization strength, which, in turn, tends to result in simpler models with smaller coefficients.

In the context of logistic regression, applying L1 or L2 regularization helps prevent overfitting, control the model complexity, and improve its generalization to new data. Scikit-learn's logistic regression implementation allows you to set the regularization type and strength using the 'penalty' and 'C' parameters, as explained above.

The Logistic Regression model was given the dataframe and was run with the default hyperparameter values. Similarly, the dataset was separated into training and validation sets, allocating 20% for testing. Predictions were generated for both the training and testing sets, and accuracy metrics were computed to evaluate the model's performance. Additionally, a detailed classification report was printed for the test set. Throughout the experiments, ROC curves and AUC values were used for evaluation as well.

The model reached an accuracy value of 97.01%, and an F1 score of 59%. The result was very similar to the initial random forest model, with slightly better results on the LP class precision and recall. Yet, the model needed to be properly reconstructed to prevent overfitting and the impact of the majority class. The majority class had 99% recall and F1 score values. The classification report of the first model is given in Table 6.4.

Classification Report	**Precision**	**Recall**	**F1-Score**	**Support**
CF	0.64	0.53	0.58	17
DC	0.62	0.53	0.57	19
LP	0.67	0.13	0.22	15
None	0.98	0.99	0.99	1254
Macro Avg	0.73	0.55	0.59	1305
Weighted Avg	0.97	0.97	0.97	1305
Overall Accuracy	0.97			

Table 6.4: Classification Report for Basic Logistic Regression

Grid Search Hyperparameter Tuning

A Grid Search approach was followed to find the optimum hyperparameters of the basic Logistic Regression model. Among the parameters described for the model, the following grid of values was introduced:

- penalty: ['l1', 'l2'],
- C: [0.1, 0.7, 1, 5, 10],
- solver: ['liblinear', 'newton-cg', 'lbfgs', 'sag', 'saga'],
- max_iter: [100, 200, 300]

The model was run with all possible combinations, similarly, the random_state parameter remained the same for comparability since it influences the randomness and reproducibility.

After running the model with all combinations, the best combination of the parameters was obtained as follows:

Best Parameters: `{'C': 0.7, 'penalty': 'l1', 'max_iter': 100, 'solver': 'saga'}`

The resulting classification report when these parameters were used can be seen in Table 6.5. Building upon the results of the Grid Search, identified optimal hyperparameters were used for further experiments with Logistic Regression.

The optimum input values resulted in a 2% increase in the average F1 score and improved the precision and recall scores of the minority classes. However, this result was still an example of overfitting, considering the unrealistic accuracy value of 97% and the 'None' class scores. This Grid Search pointed out that the hyperparameter adjustment did not cause adequate improvement and the model needed further refinements.

Classification Report	Precision	Recall	F1-Score	Support
CF	0.80	0.47	0.59	17
DC	0.82	0.47	0.60	19
LP	0.67	0.13	0.22	15
None	0.98	1.00	0.99	1254
Macro Avg	0.82	0.52	0.60	1305
Weighted Avg	0.97	0.97	0.97	1305
Overall Accuracy	0.97			

Table 6.5: Classification Report for Logistic Regression After Grid Search

Feature Selection Methods

For the Logistic Regression model, with the optimum hyperparameters obtained with the Grid Search, some feature selection methods were employed to eliminate the redundant columns for a better analysis of the variants. SelectKBest, SelectPercentile and Mutual information methods were used to evaluate the feature importance.

SelectPercentile is a feature selection method, a class within the sklearn.feature_selection module of scikit-learn library. This method is used to select the top features based on univariate statistical tests. SelectPercentile is a class that takes a scoring function as an argument and selects a specified percentage of features based on their scores. The scoring function assesses the statistical significance of each feature independently. In this study, the F-statistic was used to assess the significance, as in SelectKBest method. The top percentage of features was determined by the percentile parameter, that was selected. Some experiments were made with the percentile value in order to perform an effective selection process.

The results were not satisfactory in terms of accuracy, F1 score, and preventing overfitting when these methods were used for the model. SelectPercentile method yielded the same F1 score and accuracy score by using 30% of the features (51). This, of course, gave an idea of the feature significance. The results were not notably different when the other selection methods were used, SelectKBest method produced an F1 score of 61%, and an accuracy of 98%, by selecting 125 features. And mutual information method had the lower F1 score macro average of 56% by choosing 100 features. One common aspect of the three methods was that the scores of the 'LP' class dropped as more features were eliminated. Therefore it could be concluded that the 'LP' class was the most sensitive and harder to predict because of the nature of the problem and the smaller amount of examples.

Undersampling

Experiments with undersampling were conducted, together with the selected features by different methods to achieve a better generalization and a high F1 score. Different amounts of random samples were selected gradually at every trial with the same and different random states, performing a cross-validation, to choose the most representative subset of the 'None' class (majority class). During the experiments, the amount of the selected features was also adjusted by monitoring outcomes at every trial. This approach helped to obtain more tailored and target-oriented results and in fact, allowed the discovery of the best Logistic Regression model for the problem.

The integration of feature selection methods and undersampling outperformed the previous Logistic Regression models and yielded the best results in terms of generalization, better scores, and preventing overfitting. With a random sample of 400 examples of the 'None' class, SelectKBest method provided an F1 score of 81% and 81% of overall accuracy. SelectPercentile and Mutual information methods followed this with 88% and 86% of accuracy, respectively. SelectKBest method was the best with the other performance metrics as well, besides F1 score and accuracy. A detailed classification report can be seen in Table 6.6. In addition, ROC Curves and AUC values of this model were shared in Figure 6.2.

6.1.3 Support Vector Machines

SVM constitutes a sophisticated machine learning algorithm with versatile applications, also in bioinformatics and research studies. The fundamental principle of SVM is the identification of an optimal hyperplane within the feature space that effectively separates different classes. This hyperplane is strategically positioned to maximize the distance between instances of different classes. The data points contributing to the definition of this hyperplane are known as support

Classification Report	Precision	Recall	F1-Score	Support
CF	0.72	0.90	0.80	20
DC	0.91	0.71	0.80	14
LP	0.73	0.67	0.70	12
None	0.96	0.95	0.96	86
Macro Avg	0.83	0.81	0.81	132
Weighted Avg	0.90	0.89	0.89	132
Overall Accuracy	0.89			

Table 6.6: Classification Report for Logistic Regression with Selected Features and Undersampling

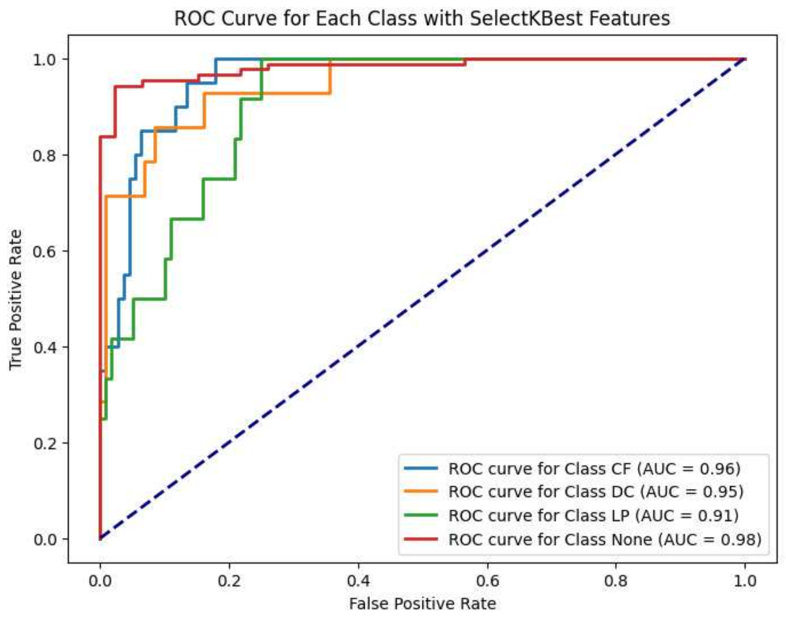


Figure 6.2: ROC Curves and AUC Values of the Final Logistic Regression Model

vectors. SVM aims to find the hyperplane that not only classifies the data accurately but also maximizes the margin, providing robust generalization to unseen instances. To achieve this, SVM employs a mathematical optimization approach that considers the support vectors and their associated weights. The algorithm seeks to find the weights that minimize the classification error while simultaneously maximizing the margin.

One of the reasons for employing SVM is its ability to handle non-linear relationships within the data. This is accomplished through the use of kernel functions, which transform the original feature space into a higher-dimensional space, allowing SVM to find linear decision boundaries in the transformed space. SVM is particularly adept at handling high-dimensional data, suited for variant classification scenarios where genetic features may be numerous and diverse.

In variant classification tasks, SVM's emphasis on maximizing the margin ensures a generalized model, even in the presence of noisy or overlapping data. Given that the dataset had many unknown values and noise, SVM was expected to have a reasonable performance.

Basic SVM

An SVM model was run with default parameters by using the SVC (Support Vector Classification) class, which resides in the SVM module of the scikit-learn library in Python 3. This class has default parameters initially defined to train models to facilitate the training process, which can be adjusted to handle different scenarios. The default hyperparameters on which the model was run are as follows:

svm.SVC(C=1.0, kernel='rbf', degree=3, gamma='scale')[source]

The explanations of these values and their effects on the model are given below:

- C: Regularization parameter. The strength of the regularization is inversely proportional to C. The penalty is a squared l2 penalty.
- kernel: Specifies the kernel function to transform the input data. Some functions can handle linearly separable data, while others handle non-linear data.
- degree: Degree of the polynomial kernel function. Higher degrees can capture more complex relationships but may lead to overfitting.
- gamma: Kernel function coefficient, influences the shape of the decision boundary. A smaller gamma value results in a more generalized decision boundary, while a larger gamma can lead to a more complex decision boundary.

The dataframe was given as input to the SVM model with the default hyperparameter values, after being separated into training and validation sets in the same way as the other models. Predictions were generated for both the training and testing sets, and performance metrics such as accuracy, precision, recall, and F1 score were computed to evaluate the performance. Detailed classification reports were visualized at the end of each experiment. ROC curves and AUC values were calculated for evaluation as well.

The basic model reached an accuracy value of 96%, and an F1 score of 37%. The result was not satisfactory since the model exhibited deficiencies in accurately distinguishing between classes. The accuracy metrics showed a notable discrepancy, with a high overall accuracy masking the model's struggle to correctly identify instances from the minority class. This contradiction pointed out the model's inability to generalize to new, unseen data and also issues with overfitting or underfitting. The misclassification of instances, especially in the 'LP' class, pointed to a lack of sensitivity to important patterns. The majority class, on the other hand, had an unrealistic recall value of 100%. The resulting classification report is given in Table 6.7.

Classification Report	**Precision**	**Recall**	**F1-Score**	**Support**
CF	1.00	0.18	0.30	17
DC	0.67	0.11	0.18	19
LP	0.00	0.00	0.00	15
None	0.96	1.00	0.98	1254
Macro Avg	0.66	0.32	0.37	1305
Weighted Avg	0.95	0.96	0.95	1305
Overall Accuracy	0.96			

Table 6.7: Classification Report for Basic SVM

Grid Search Hyperparameter Tuning

A Grid Search approach was followed to find the optimum hyperparameters of the SVM model. Among the parameters described for the model, the following grid of values was introduced:

- C: [0.1, 0.25, 0.5, 0.75, 1, 2, 3],
- kernel: ['linear', 'rbf', 'poly'],
- degree: [2, 3, 4, 5],
- gamma: ['scale', 'auto']

The model was run with all possible combinations by using the same random_state parameter for comparability and reproducibility purposes. After running the model with all combinations, the optimal combination of the parameters was obtained as follows:

Best Parameters: `{'C': 2, 'kernel': 'poly', 'degree': 4,'gamma': 'scale'}`

The resulting classification report when these parameters were used can be seen in Table 6.8.

Identified optimal hyperparameters with this Grid Search were used for further experiments regarding SVMs.

The optimal hyperparameters resulted in a significant improvement in the model's metrics, compared to the initial state of the default setting. The accuracy was 97% and the F1 score had an average of 63%, including other metric improvements of minority classes. Despite the obvious overfitting issue and the strong impact of the majority class, this experiment helped to show that SVM is significantly influenced by hyperparameter adjustments, emphasizing the impacts of these parameter choices on the performance and behavior of the model.

Classification Report	Precision	Recall	F1-Score	Support
CF	0.50	0.59	0.54	17
DC	0.80	0.42	0.55	19
LP	0.62	0.33	0.43	15
None	0.98	0.99	0.99	1254
Macro Avg	0.73	0.58	0.63	1305
Weighted Avg	0.97	0.97	0.97	1305
Overall Accuracy	0.97			

Table 6.8: Classification Report for SVM After Grid Search

Feature Selection Methods

For the SVM model with the optimum hyperparameters, SelectKBest, SelectPercentile, and Mutual information methods were used to select the most relative features, also, PCA method was used for dimensionality reduction to investigate the effects of the dimensionality on SVM.

Principal Component Analysis (PCA) is a powerful dimensionality reduction technique employed to extract essential information from high-dimensional datasets while minimizing information loss. The process begins with the standardization of features to ensure a uniform distribution in scale. PCA then computes the covariance matrix, revealing how features co-vary. By finding the eigenvectors and eigenvalues of this matrix, PCA identifies the principal components—the directions capturing the maximum variance in the data. These components are ordered based on their associated eigenvalues, with higher eigenvalues signifying greater variance. Through projection, PCA transforms the original data into a lower-dimensional subspace defined by the selected principal components. This reduction in dimensionality facilitates a more precise representation of the dataset while retaining the most significant sources of variation. Practical applications of PCA include noise reduction, visualization, feature selec-

tion, identifying dominant patterns, and improving the efficiency of machine learning models in various fields, including genetics and machine learning.
At the end of the experiments with different numbers of selected features using different methods, the feature selection methods were not effective in terms of improving the results of classification. SelectKBest method was used for selecting 85 features, which led to the same accuracy and F1 score level without the feature selection. It was observed that with the SVM model, selected features differed from those selected with Logistic Regression and Random Forest models, meaning that the model considers prediction score values more than the other two models. SelectPercentile method produced its best result with a percentile value of 95, having an average F1 score of 63%, and 96% of accuracy. Mutual information, on the other hand, achieved the best results with 100 features, having a 67% F1 score and 97% accuracy. PCA method, however, did not exhibit an efficient integration with the SVM model, since the performance metrics were affected badly. Different number of components were used, and in the best scenario, the F1 score and precision values were dropped and overfitting was still present. This suggested that the original features in their raw form provide more discriminative information for the task at hand compared to the reduced set of principal components. The original features contain relevant and informative patterns that directly contribute to the SVM's ability to distinguish between different classes. Moreover, Training the SVM without PCA was computationally more efficient, as the PCA process took significantly more time. Thus, PCA gave an idea about dimensionality and was later used for the unsupervised learning algorithms.

Undersampling

A similar random undersampling method was used with cross-validation to find the most representative subset of the majority class. Different amounts of samples were drawn from the dataset and the SVM models with the best hyperparameters and feature selection methods were integrated.

The SVM model with a subset of the 'None' class which had 400 examples and with the same minority classes had the best performance, over the other amounts that were tried. The 85 features chosen with the SelectKBest method achieved the highest F1 score, 72%, and avoided overfitting by having a more balanced prediction among the classes. The overall accuracy was 84%, and the best-predicted class was 'None' as it was for the other methods. The classification report is given in Table 6.9 and the ROC curves can be seen in Figure 6.3.

Classification Report	Precision	Recall	F1-Score	Support
CF	0.75	0.90	0.82	20
DC	1.00	0.43	0.60	14
LP	0.60	0.50	0.55	12
None	0.87	0.93	0.90	86
Macro Avg	0.80	0.69	0.72	132
Weighted Avg	0.84	0.83	0.82	132
Overall Accuracy	0.84			

Table 6.9: Classification Report for SVM with Selected Features and Undersampling

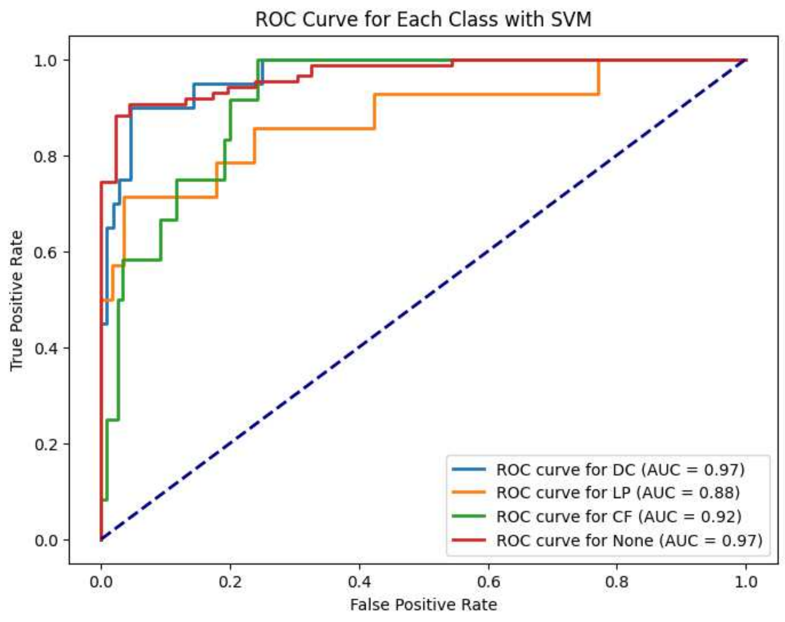


Figure 6.3: ROC Curves and AUC Values of the Final SVM Model

6.1.4 Neural Network

Neural Network (NN) is a computational model inspired by the structure and function of the human brain. It is a powerful and versatile machine learning technique used for various tasks, including classification, regression, and pattern recognition. The basic building block of a neural network is the artificial neuron, perceptron. Neurons are organized into layers, including an input layer, one or more hidden layers, and an output layer. Each connection between neurons has an associated weight, representing the strength of the connection. The weighted sum of its inputs determines the output of a neuron passed through an activation function. The training process of a neural network involves adjusting the weights to minimize the difference between the predicted output and the actual target values. This is typically done using optimization algorithms like gradient descent. During training, the network learns to recognize and capture the underlying patterns in the input data, allowing it to make accurate predictions on new, unseen data.

Deep neural networks, or deep learning models, are characterized by the presence of multiple hidden layers. These deep architectures enable the network to automatically learn hierarchical representations of features, making them highly effective. Neural networks have shown success in various domains of biology, including genomic sequence analysis, protein structure prediction, disease diagnosis and prediction, and biological image analysis. Their ability to automatically learn complex features from data has contributed to their widespread adoption and continues to drive advancements in artificial intelligence and machine learning.

Model Building and Customization

In this phase, different neural networks with different architectures were constructed to find the most suitable attributes and hyperparameters. Experiments were conducted to see the effects of layers, number of nodes, activation functions, epoch numbers and batch sizes to model accuracy and other performance metrics. The target variable was initially encoded using LabelEncoder, followed by one-hot encoding to prepare it for multiclass classification. Then the data was split into training and testing sets, as in the previous procedures. The network naturally included an input layer of 170 neurons as the number of features and an output layer of 4 neurons, given that the number of output neurons is the number of classes. Of course, the number of neurons in the input layer changed as different feature selection methods were used to drop some features for later experiments. A single hidden layer was employed with varying neuron counts to capture the relationships within the dataset. The evaluation of results involved considering previously used performance metrics, in addition to test loss and test accuracy. Additionally, graphical representations illustrating the evolution of loss and accuracy over epochs were generated to understand the model's behavior.

In a neural network model, the **test loss** is a metric that measures how well the model performs on the test set. It quantifies the difference between the predicted values and the actual target values. Lower test loss values indicate better performance, as they suggest that the model's predictions are closer to the true values. On the other hand, **test accuracy** represents the proportion of correctly classified instances in the test set, as used before.

In the conducted experiments, the neural network with one hidden layer encountered limitations in effectively capturing the underlying patterns. The model's performance was observed to need improvements. An accuracy value of 95% was reached, however F1 score and other metrics were below 50%, pointing out the inconsistent prediction and the dominance of the majority class. To address this issue, the number of hidden layers was increased by introducing additional hidden layers. The model's capacity to comprehend and represent complex relationships within the dataset was augmented.

After different experiments, the final neural network architecture comprised an input layer, two hidden layers with 64 and 12 neurons, and an output layer with 4 neurons representing different classes. The ReLU activation function was applied to the hidden layers, while the output layer used softmax. The model was compiled with the Adam optimizer and categorical cross-entropy loss, and training was performed on the training set. Evaluation of the test set yielded metrics such as loss and accuracy. After the generation of predictions, a classification report and a confusion matrix provided detailed insights into the model's performance for each class, together with visualization. The resulting classification report can be seen in Table 6.10. The model had an accuracy score of 97.08% and a test loss of 10.58%. Overfitting seemed a present issue due to unrealistic predictions of the 'None' class, but an average F1 score of 64% and better precision and recall metrics outperformed the former initial methods.

Classification Report	Precision	Recall	F1-Score	Support
CF	0.65	0.69	0.63	17
DC	0.58	0.58	0.58	19
LP	0.44	0.27	0.33	15
None	0.98	0.99	0.99	1254
Macro Avg	0.66	0.62	0.64	1305
Weighted Avg	0.97	0.97	0.97	1305
Overall Accuracy	0.97			

Table 6.10: Classification Report for Initial Neural Network

Feature Selection Methods and Undersampling

Feature selection methods were employed to drop redundant features and increase the efficiency of the model, together with random undersampling. In addressing class imbalance within the dataset, a strategic undersampling approach was implemented. Specifically, instances from the "None" class were resampled, with 400 instances selected based on experimentation. The same amount of instances from other classes remained for a more balanced dataset. After selecting the most representative subset, SelectKBest and mutual information methods were integrated into NN after encoding the target variable and splitting the data into training and testing sets with a ratio of 80% training and 20% testing.

Feature selection was applied using the SelectKBest method with the f_classif criterion, retaining the top 65 features. The neural network architecture was constructed with two hidden layers. Since the number of selected features was 65, the input layer consisted of 65 nodes. Then, hidden layers had 64 and 12 neurons respectively, using the ReLU activation function. For the output layer, 4 nodes were used representing variant classes, and a softmax activation function was introduced. Then, the neural network was trained using the selected features from the training set. The model was compiled with the Adam optimizer and sparse categorical cross-entropy loss function. The training process involved 25 epochs with a batch size of 32, and the model's performance was assessed on the test set. Predictions were generated for the test set, and the results were summarized using a classification report and a confusion matrix. The SelectKBest method achieved 86.36% overall accuracy and test loss was 39.77%. The average F1 score was 78%, which was better than the results obtained by the mutual information method.

The mutual information method produced similar results to the SelectKBest method, with better performance metrics than the initial model, however, it was significantly outperformed by the SelectKBest method. The integrated model, when mutual information was used, had 82% accuracy. The F1 score was 64% and all the other performance scores were lower than the ones with SelectKBest, minimum 4%.

In the end, the model with a subset of 400 examples from the majority class and employing 65 features selected through the SelectKBest method, demonstrated better performance compared to other experimental setups. The classification report is given in Table 6.11 and the confusion matrix can be seen in Figure 6.4. Also, ROC curves and AUC values for each class are given in Figure 6.5.

Classification Report	Precision	Recall	F1-Score	Support
CF	0.72	0.90	0.80	20
DC	0.85	0.79	0.81	14
LP	0.67	0.50	0.57	12
None	0.93	0.92	0.92	86
Macro Avg	0.79	0.78	0.78	132
Weighted Avg	0.86	0.86	0.86	132
Overall Accuracy	0.86			

Table 6.11: Classification Report for Improved Neural Network

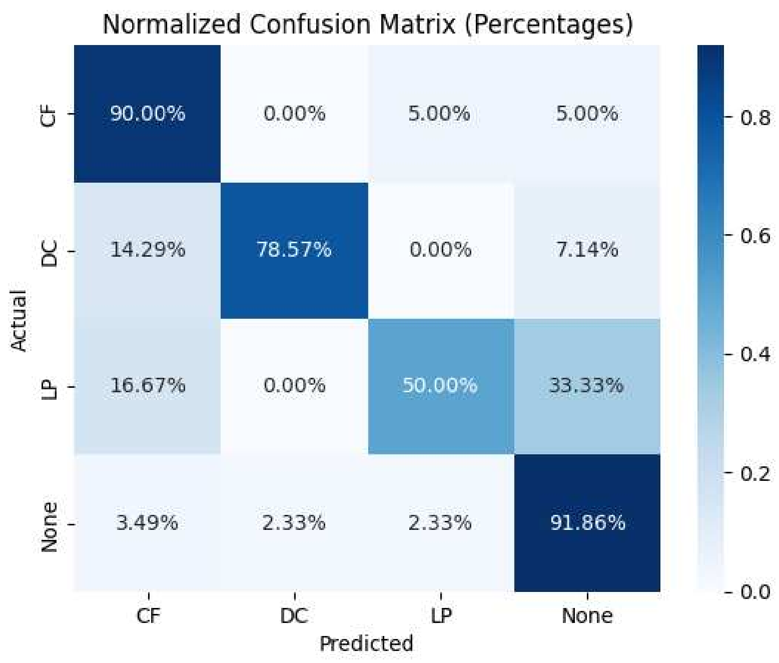


Figure 6.4: Normalized Confusion Matrix For Improved Neural Network

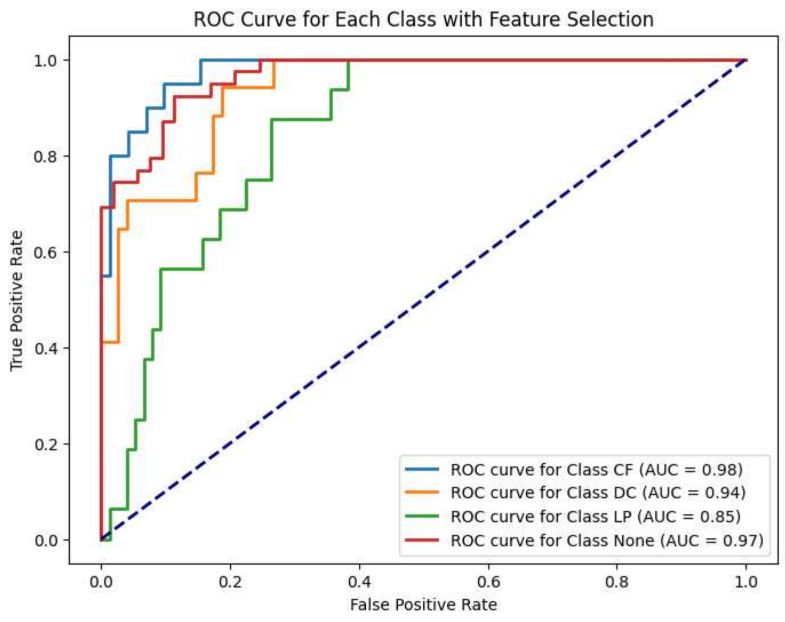


Figure 6.5: ROC Curves and AUC Values of the Improved Neural Network Model

6.1.5 K-Means for Unsupervised Learning

In the course of the thesis research, after the implementation and evaluation of various supervised learning algorithms, the scope was extended to unsupervised learning methods. This decision was motivated by the need to uncover underlying patterns, structures, or relationships within the data that might not be evident through the labeled outcomes alone. **Unsupervised learning** techniques, do not rely on predefined target labels during training. This allowed for a more comprehensive analysis of the dataset, facilitating the identification of data characteristics or clusters that may contribute valuable insights to the overall understanding of the problem.

Furthermore, the incorporation of unsupervised learning methods aimed to address the practical aspects of variant classification in real-life scenarios. In genetic variant data obtained from patient examinations, where labels for harmful variants might not be readily available, the ability to detect and characterize potential variants autonomously becomes crucial. By exploring unsupervised learning methodologies, the research sought to simulate the real-life scenario of variant analysis, where the absence of labeled data requires the algorithm to discern patterns and make informed decisions based only on the genetic variant data at hand. This approach aimed to simulate the challenges and complexities associated with variant classification in clinical genetics, contributing to a more robust and applicable framework for real-world scenarios.

K-means is an unsupervised clustering algorithm designed to separate a dataset into distinct groups, or clusters, based on patterns or similarities among data points. The method begins by randomly selecting cluster centroids, representing the center of each cluster. Subsequently, it assigns each data point to the cluster whose centroid is closest, usually determined by the Euclidean distance. After this assignment, the algorithm recalculates the centroids by computing the mean values of all points within each cluster. This process iterates until a convergence criterion is met, commonly defined by minimal changes in centroid positions or the stabilization of assignments. K-means aims to minimize the sum of squared distances between data points and cluster centroids to form strict clusters. The chosen value 'K' determines the number of clusters. In the context of variant classification, K-means can be applied to genetic data, allowing researchers to uncover patterns or groupings within the vast genomic landscape. It provides insights into potential functional implications or shared characteristics among genetic variants. In Python, the scikit-learn library's 'KMeans' class, residing in the 'sklearn.cluster' module, is used to implement the K-means algorithm.

Integrated with the K-means algorithm, PCA was applied to the feature set to address the challenge of high dimensionality in genetic variant classification. The initial step involved the extraction of features and the target column from the dataframe. To ensure uniformity in the dataset, the features were scaled using the StandardScaler, thereby standardizing their values. PCA transformed the scaled features into a new set of orthogonal variables (principal components). The objective was to reduce computational complexity, and potentially improve the performance of the K-means model.

A critical consideration in this process was determining the appropriate number of principal components to retain. To find a balance between retaining meaningful information and reducing dimensionality, a criterion of retaining 95% of the explained variance was integrated. This percentage was chosen after conducting experiments with different values, aiming to preserve a substantial portion of the original information while achieving a significant dimensionality reduction.

The computed explained variance ratio for each principal component and the cumulative explained variance were crucial outputs of the PCA. These metrics provided valuable insights into the contribution of each principal component to the overall variance in the dataset. A scree plot was used to visualize the cumulative explained variance, which is given in Figure 6.6.

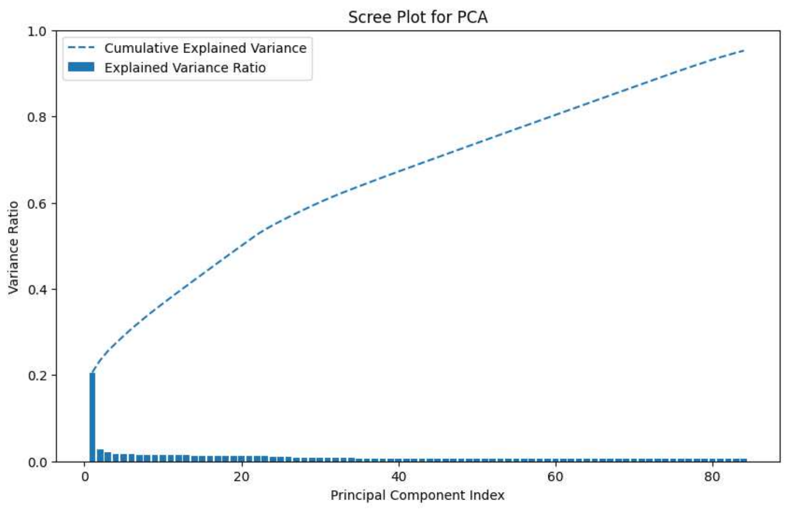


Figure 6.6: Scree Plot for Cumulative Explained Variance During PCA Process

By examining the cumulative explained variance, it was possible to understand the cumulative amount of information retained as the number of principal components increased. According to the graph, fewer components should have been used to have a more effective result. Especially, the elbow point of the graph was considered where 25 components were present.

Finally, the transformed features resulting from PCA, represented as principal components, were stored in a new dataframe. This enriched dataset, combining the reduced-dimensional features and the target column, served as a foundation for the subsequent K-means algorithm implementation. Initially, 4 clusters were introduced to the algorithm since there were 4 variant classes. The number of clusters (num_clusters) was a critical parameter in the KMeans algorithm and different numbers of clusters were tried through experimentation later. The algorithm assigned each variant to a specific cluster, creating a clustering structure that revealed relationships among the variants.

Integration of the cluster labels into the PCA-transformed features DataFrame facilitated the association of each genetic variant with its respective cluster. This linkage provided valuable information for downstream analysis, enabling the examination of distinct characteristics associated with each identified cluster.

To depict the clustering results visually, two types of plots were generated. The first was a 2D scatter plot, depicting the samples in the plane defined by the first two principal components. This plot allowed for a visual inspection of the separation between clusters, aiding in the assessment of the algorithm's efficacy. The second was a 3D scatter plot, providing a more comprehensive visualization by incorporating the third principal component. The colorized clusters in both plots enhanced interpretability, allowing for the identification of potential patterns and relationships within the genetic variant dataset. The 3D plot is given in Figure 6.7

In conclusion, this unsupervised learning approach, employing PCA and KMeans clustering, provided insights into the genetic variant dataset. In the absence of traditional supervised learning evaluation metrics such as accuracy or loss, the assessment of clustering results relies on alternative metrics used for unsupervised learning. Upon initial evaluation of the clustering outcomes, it became evident that the desired objective of separating genetic variants into distinct clusters based on similarities in their features was not fully achieved. Instead, the clustering exhibited suboptimal performance, as evidenced by the presence of multiple variant types within individual clusters. Despite this initial setback, the clustering results provide valuable insights and serve as a foundational framework for future experiments to refine the clustering approach. The table of the counts of variant classes within each cluster offers a summary of the distribution of variant types across the identified clusters. While the clustering outcomes may not meet the desired objectives in their current state, they represent a starting point for the exploration of the relationships. This exploration enriches the overall understanding of the genetic variant landscape, complementing the insights from supervised learning methods in variant classification. A graph representing the results, showing the distribution of variant types among the clusters can be seen in Figure 6.8.

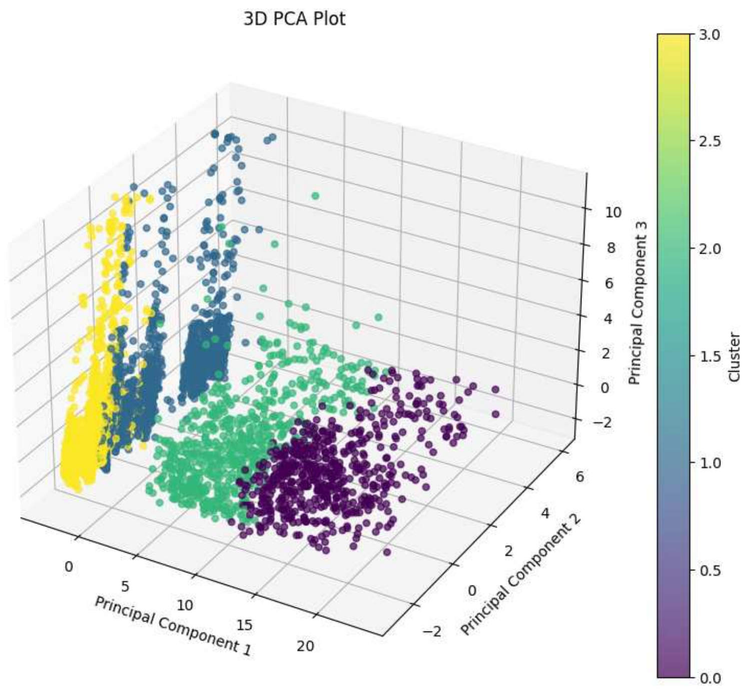


Figure 6.7: 3D Scatter Plot of 4 Clusters With 3 Components

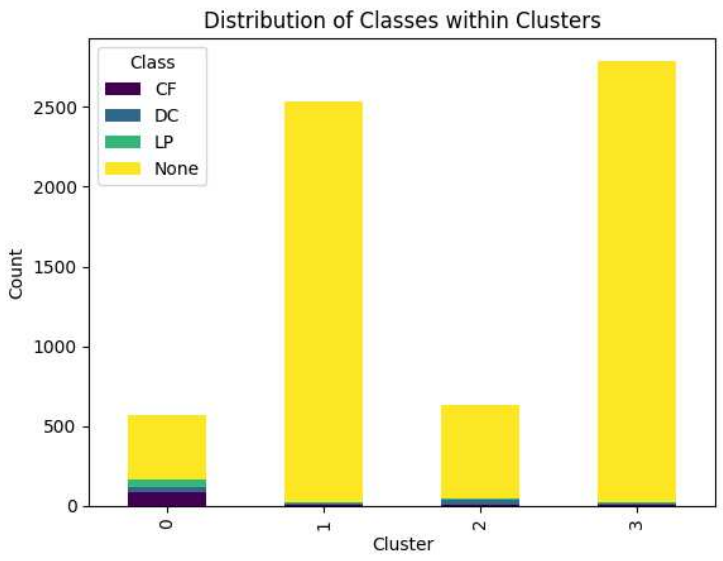


Figure 6.8: Distribution of Variant Types Among the Clusters

Table 6.12 has the counts providing the resulting distribution of genetic variant types across the identified clusters. Cluster 0 exhibits a diverse composition of variant classes, with a relatively higher representation of variants classified as "None" compared to the other clusters, suggesting a lack of clear separation between variant types within this cluster. On the contrary, Cluster 1 demonstrates an imbalance in class representation, with a predominant presence of variants classified as "None," indicating a potential classification within the cluster. Cluster 2 displays a more balanced distribution of variant classes, with lower overall counts compared to Clusters 0 and 1. Cluster 3 exhibits a similar pattern to Cluster 1, with a slightly more balanced distribution of variant classes. Overall, the results suggest the need for further refinement and optimization of the clustering approach to achieve better separation of variant types within distinct clusters.

Table 6.12: Clustering Results with 4 Clusters

Cluster	CF	DC	LP	None
0	82	32	55	397
1	3	14	3	2515
2	10	26	11	586
3	3	14	6	2766

K-Means with 2 Clusters

The K-means algorithm was initially applied with four clusters to classify variants; however, suboptimal results were observed. As a response, a decision was made to explore an alternative approach by reducing the number of clusters to two. This adjustment aimed to classify variants into two broad categories: None and Causal. The transition from four to two clusters was motivated by the inadequacies observed in the four-cluster model, which inefficiently represented the underlying data distribution. By simplifying the clustering process into two distinct groups, a clearer separation among variants was sought to be achieved. This was an experimental approach employed to explore the effects of different clustering configurations.

The experiment started by transforming the variant classes into a binary classification scheme, determining neutral and causal variants. According to this, the 'None' class remained the same, and other classes were merged into one class named 'Causal'. This encoding strategy simplified the subsequent analysis by categorizing variants into two broad groups based on their potential impact. Following this, PCA was employed to reduce the dimensionality of the feature space while preserving the variance information within the dataset. This time, 27 principal components were used since it was discovered to be more efficient in terms of explained variance in the previous model. The next step involved applying K-means clustering to the transformed features. This clustering algorithm separated the variants into two distinct clusters based on their shared characteristics. Visualization of the clustering results in both two-dimensional and three-dimensional PCA spaces provided a comprehensive understanding of the spatial distribution of variant clusters and highlighted any patterns or separations. The 3D plot of the clusters is given in Figure 6.9. After computing the cluster and class counts, the results were used to

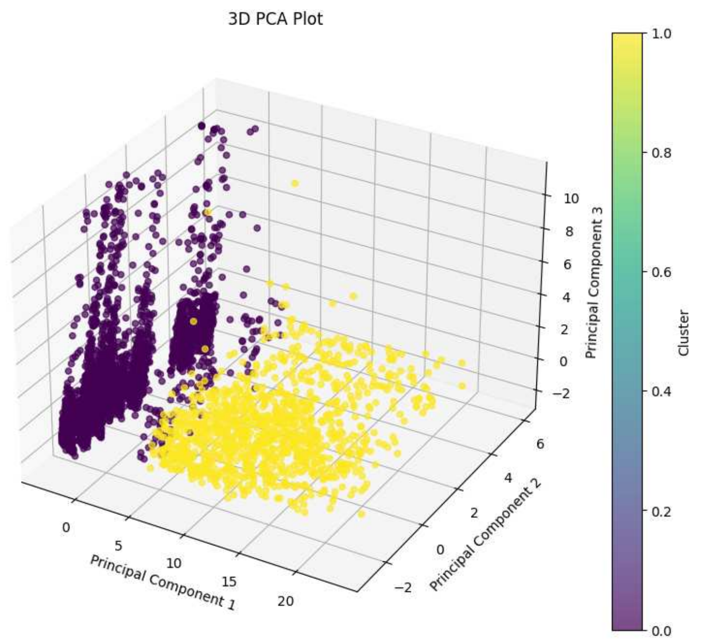


Figure 6.9: 3D Scatter Plot of 2 Clusters With 3 Components

create a table, which is given in Table 6.13. Following this, a stacked bar chart is generated to visually depict the distribution of classes across clusters, which can be seen in Figure 6.10. Overall, this visualization offers valuable insights into the clustering results and the distribution of variant classes within the identified clusters.

Table 6.13: Clustering Result With 2 Clusters

Cluster	Class 0	Class 1
0	882	213
1	5382	46

The comparison between the clustering results obtained with two clusters and four clusters revealed patterns in the distribution of variant classes within the identified clusters. In the case of the 2-cluster solution, the count of samples from each class was aggregated into two clus-

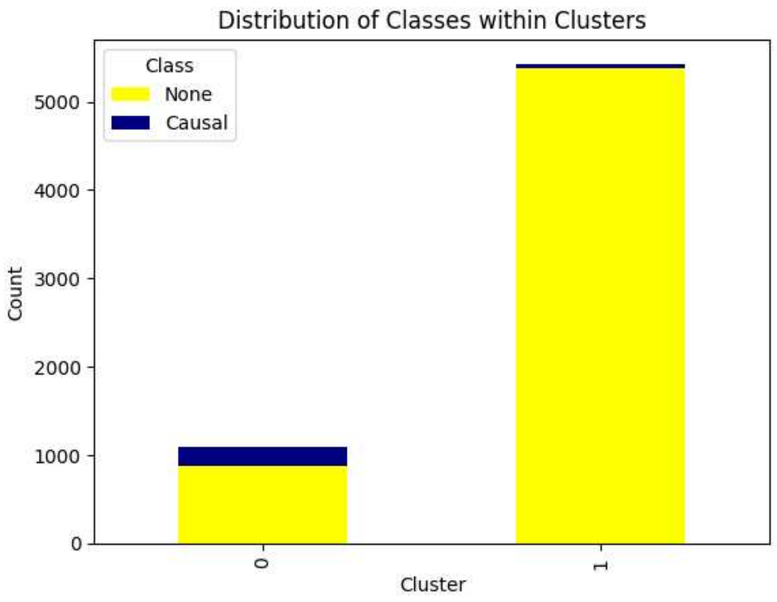


Figure 6.10: Distribution of Variant Types Among 2 Clusters

ters, denoted as '0' and '1.' Cluster '0' primarily comprised instances categorized as 'None,' with a substantial count of 882 and the vast majority of the 'Causal' variants, while Cluster '1' contained a dominant representation of the 'None' class, totaling 5382 samples. This simplified separation captured a broader differentiation among variants, despite some overlap within each cluster. Conversely, the 4-cluster solution outlined a more detailed separation of variant classes across distinct clusters. Cluster '0' had a heterogeneous distribution, including a mix of variants from different classes, including 'CF,' 'DC,' 'LP,' and 'None.' Clusters '1,' '2,' and '3,' on the other hand, demonstrated more focused allocations, each dominantly containing samples from specific classes. For instance, cluster '1' mostly comprised variants labeled as 'None,' while cluster '2' mostly contained variants classified as 'CF.' Cluster '3' had a balanced distribution across multiple classes, reflecting a diverse composition of variant types. Overall, this comparison emphasizes the trade-off between simplicity and detailedness in clustering solutions, with the choice between two clusters and four clusters offering different levels of resolution in capturing the underlying structure of variant classes within the dataset.

Other Experiments With K-Means

For the purpose of enhancing variant classification through K-means clustering, a series of experiments were designed and executed after initially trying clustering with 4 and 2 clusters.

Also, experiments with 3-cluster were performed, with two distinct combinations. In the first setup, Disease-causing and Likely Pathogenic variants were merged into a single group, while Contributing Factors were grouped separately, and the None class formed the third cluster. In the second setup, Disease-causing variants were isolated into one group, while Contributing Factors, alongside Likely Pathogenic variants, formed another group, and the None class constituted the third. These distinct configurations were designed to increase the effectiveness of clustering by segregating variant types in different ways.

Additionally, the experimentation process employed undersampling techniques, systematically varying the sample sizes from 200 to 1500 examples of the majority class, incrementing by 100 at each iteration. The reason behind this approach was to assess both the impact of various cluster numbers and the impact of a balanced dataset on the performance of the model.

Furthermore, the investigation extended to feature selection with SelectKBest, by systematically varying the number of selected features, the goal was to evaluate the influence of feature dimensionality on clustering performance. This approach aimed to identify an optimal feature subset that could capture the inherent variance in the data, for having an efficient cluster separation.

One of the metrics to evaluate the performance of the models was **The Adjusted Rand Index** (ARI), which is a measure used to evaluate the performance of clustering algorithms, such as K-means clustering. It assesses the similarity between two clusterings by considering how well they agree on which points belong together in the same cluster and which points belong to different clusters. It measures agreement between points that are grouped together in both the true and predicted clusters, as well as agreement between points that are separated into different clusters. The ARI score ranges from -1 to 1, where a score close to 1 indicates strong agreement between the true and predicted clusterings, a score around 0 indicates random clustering and a score close to -1 indicates disagreement between the clusterings. A higher ARI score suggests better clustering performance. In Figure 6.11 and 6.12, the graph showing the change in ARI score as the number of samples from the majority class increases is given in 2-cluster and 4-cluster scenarios, respectively.

Despite the thoroughness of the experimental approaches, the outcomes did not align with the initial objectives. In all configurations, the clusters were observed to blend different variant types, rather than forming distinct clusters corresponding to specific variant classes. Consequently, while the experimental results of the clusters were visualized with plots and tables for in-depth analysis, their inability to accurately differentiate variant types led to their exclusion from this thesis study. This emphasized the complexities in variant classification using

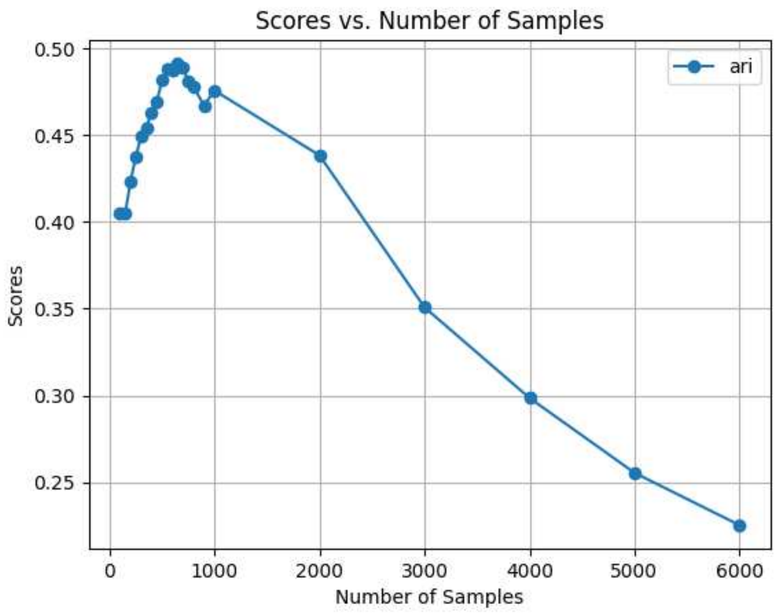


Figure 6.11: ARI Score Variation with Majority Class Sample Size in 2-Cluster Scenario

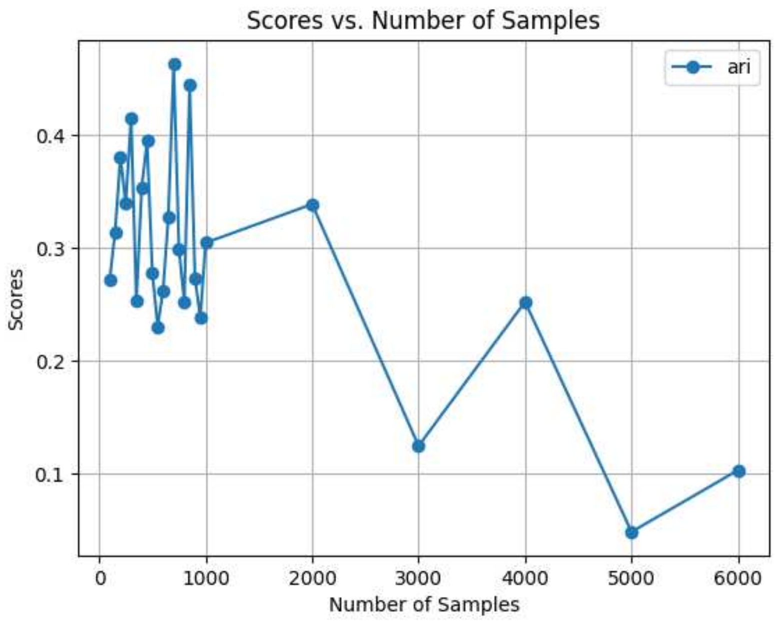


Figure 6.12: ARI Score Variation with Majority Class Sample Size in 4-Cluster Scenario

clustering methods, revealing the challenges encountered in achieving research objectives in a real-world setup.

6.1.6 Gaussian Mixture Models for Unsupervised Learning

In light of the unsatisfactory results obtained from various experiments with K-means clustering, the GMM algorithm was experimented with in this section, since GMMs offer a more sophisticated and probabilistic approach to clustering analysis. With the inherent limitations of K-means clustering, leveraging the probabilistic nature of GMMs was considered helpful. Through the experiments with GMMs, this section aims to achieve more accurate and informative clustering outcomes for variant classification.

Gaussian Mixture Model represents a powerful probabilistic framework widely employed for clustering and density estimation tasks in machine learning and statistical analysis. Unlike traditional clustering methods such as K-means, GMM employs a more probabilistic approach by assuming that the data points originate from a mixture of multiple Gaussian distributions, each characterized by its own mean and covariance matrix. This model essentially hypothesizes that the observed data points are generated from a combination of underlying subpopulations or clusters, with each cluster resembling a Gaussian distribution in the feature space. By iteratively estimating the parameters of these Gaussian distributions and optimizing the model likelihood, GMM effectively captures complex data patterns and provides a probabilistic assignment of data points to different clusters. Moreover, GMM allows for soft assignments, enabling data points to belong to multiple clusters simultaneously, with varying degrees of membership probabilities. This flexibility makes GMM particularly well-suited for uncovering structures in the data, accommodating clusters of varying shapes, sizes, and densities. In the context of variant classification, GMM emerges as a valuable tool for constructing clusters of genetic variants with similar attributes, facilitating the identification and characterization of distinct variant classes.

In the initial stage of the experimentation, a foundational GMM was constructed for variant classification. This basic GMM model was developed without any additional preprocessing steps such as feature selection or dimensionality reduction, serving as a starting point to establish a baseline understanding of the dataset's clustering behavior. The primary objective was to assess the clustering patterns present within the variant dataset. To accomplish this, the GMM algorithm was instantiated with 4 clusters (k=4) and fitted to the dataset. Then, the cluster labels assigned to each data point were obtained to count the variant types in each cluster. Visualization of this distribution using a heatmap provided valuable insights into the clustering behavior of the GMM model and laid the groundwork for subsequent experiments. These statistics can be seen in Table 6.14, which constitutes the first set of results.

Table 6.14: Initial GMM Clustering Results with 4 Clusters

Cluster	CF	DC	LP	None
0	92	57	65	880
1	2	18	3	2677
2	2	10	6	1727
3	2	1	1	980

GMM With Feature Selection for 4 Clusters

The first experiment included performing GMM with 4 clusters and in conjunction with feature selection. Initially, the dataset was prepared, separating the labels and the features. Various number of features were selected within the feasible range. The experiments iterated over the values, representing the number of features selected by the SelectKBest method.

For each iteration, SelectKBest was applied to select the top 'k' features based on their ANOVA F-value scores. Subsequently, the selected features were utilized to fit a GMM with 4 clusters (n_components=4). The resulting cluster predictions were then evaluated by generating a table of counts and a bar graph to see the distribution, similar to the K-means evaluation. Additionally, a table of the percentages of each variant class within the clusters was constructed.

Furthermore, the ARI score was computed as a quantitative measure of the agreement between the true class labels and the predicted cluster assignments. By systematically varying the number of selected features and observing the resulting cluster distributions and ARI scores, the experiments aimed to identify the optimal feature subset and evaluate GMM for variant classification.

The best result was achieved with 127 selected features, where the most important variant types were contained in the same cluster. This model provided more distinct clusters and was successful in terms of detecting the important variants, which was the ultimate purpose of the task. However, it did not separate all the variant types into different clusters as it was intended. Most of the variants from the minority classes DC, LP, and CF were contained in the same cluster, with some contamination of the 'None' class. This was one drawback of the model but it still constituted a great performance, considering the complexity of the real-world scenarios. Percentages were used for describing the contents of each cluster since they gave a better idea about the distribution. The bar graph and the percentage table are given in Figure 6.13 and Table 6.16 respectively. The first cluster, denoted as Cluster 0, had the vast majority of the important variants, and the rest of the clusters comprised mostly of 'None' variants (more than 99%). Cluster 0 had 97% of the important variants inside and those made up the 17.2% of the cluster.

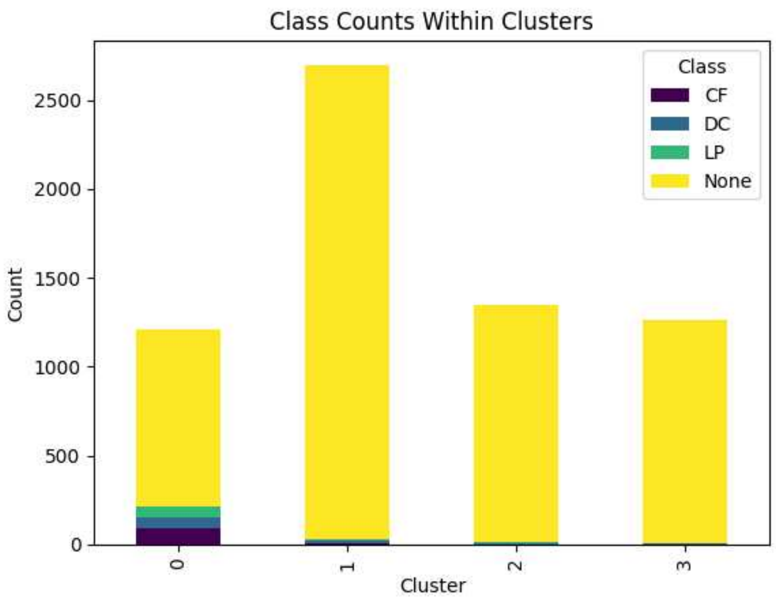


Figure 6.13: Distribution of Variants with Feature Selection (4 clusters)

Table 6.15: Percentages of Cluster Content with Feature Selection (4 Clusters)

Cluster	CF	DC	LP	None
0	7.6	4.8	5.4	82.2
1	0.1	0.8	0.1	99.00
2	0.0	0.4	0.4	99.2
3	0.2	0.2	0.0	99.7

GMM with Feature Selection for 2 Clusters

The same approach was used for 2 clusters, separating the variants into Neutral and Causal variants. The target variable was encoded categorizing DC, LP, and CF as 'Causal' while the rest of the variants were labeled as 'Neutral'.

SelectKBest method was applied to select the top 65 best features based on the F-value between the features and target variable, aiming to reduce the dimensionality of the data while retaining the most relevant features for clustering.

Next, a GMM with 2 components was constructed and fitted to the selected features. Following the clustering, a table was used to show the counts of each class within each cluster,

Table 6.16: Percentages of Cluster Content with Feature Selection (4 Clusters)

Cluster	CF	DC	LP	None
0	7.6%	4.8%	5.4%	82.2%
1	0.1%	0.8%	0.1%	99.00%
2	0.0%	0.4%	0.4%	99.2%
3	0.2%	0.2%	0.0%	99.7%

providing insight into the distribution of variant classes within the identified clusters. Additionally, percentages were calculated to represent the proportion of each class within each cluster, offering a more comprehensive understanding of the distribution. These tables are given by 6.17 and 6.18.

Also, a bar graph representing the class counts within each cluster can be seen in 6.14.

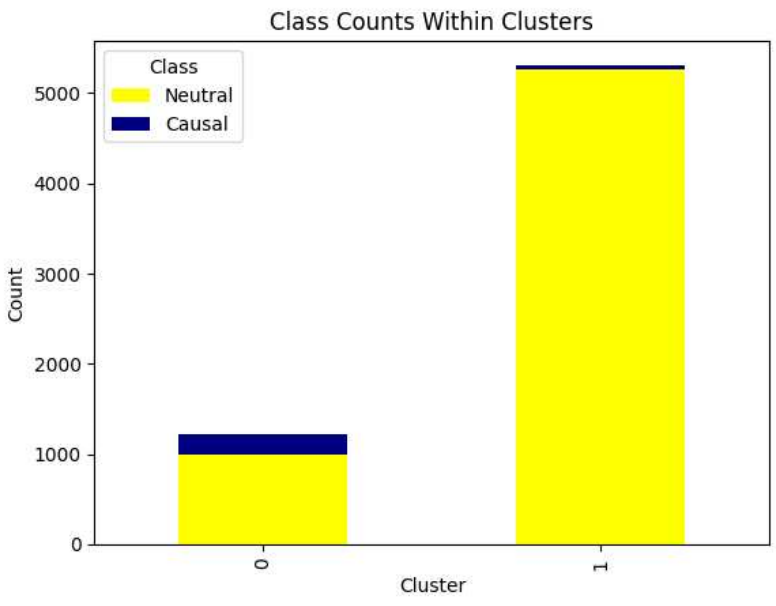


Figure 6.14: Distribution of Variants with Feature Selection (2 clusters)

Table 6.17: Class Counts With Feature Selection

Cluster	Neutral	Causal
0	997	216
1	5267	43

Table 6.18: Cluster Percentages With Feature Selection

Cluster	Neutral	Causal
0	82.2%	17.8%
1	99.2%	0.8%

In the results obtained with 2 clusters and 65 selected features, Cluster 0 comprised 82.2% of neutral variants and 17.8% potentially harmful variants. On the other hand, Cluster 1 predominantly contained neutral variants, constituting 99.2% of its members, with only 0.8% being potentially harmful. This suggested a clear separation between the two clusters, with Cluster 0 encapsulating potentially harmful variants, while Cluster 1 predominantly consists of neutral variants.

The performance of the 2-cluster approach appeared to offer a more distinct separation between variant types compared to the 4-cluster approach. In the 2-cluster scenario, one cluster predominantly contains variants labeled as potentially harmful, while the other cluster includes neutral variants. This clear separation suggested a simpler and more intuitive clustering pattern, which may facilitate the following analysis and interpretation. On the other hand, the 4-cluster approach results in a more heterogeneous distribution of variant types across clusters, with each cluster containing a mix of variant classes. This complexity may pose challenges in accurately interpreting the clusters, potentially leading to greater ambiguity in variant classification. Therefore, the 2-cluster approach may offer better performance in terms of cluster interpretability and utility for variant classification tasks.

T-SNE Integration

In the following experiments, **t-SNE** (t-distributed Stochastic Neighbor Embedding) was incorporated with GMMs after feature selection, aiming to enhance the efficiency of variant clustering. By employing t-SNE, a dimensionality reduction was performed to visualize high-dimensional data in lower dimensions while preserving local structures.

t-SNE is typically used for two or three dimensions for effective visualization. It works by first constructing a probability distribution over pairs of high-dimensional data points, representing similarities between points. This distribution is then optimized to be as similar as possible to a similar probability distribution over pairs of points in the lower-dimensional space. The optimization process minimizes the Kullback-Leibler divergence between the two distributions, effectively preserving local relationships and structure in the data. As a result, t-SNE can effectively reveal clusters, patterns, and relationships within complex datasets, making it particularly useful for further analysis and visualization. Subsequently, integrating t-SNE and GMM after feature selection offers better interpretability and robustness of the classification framework.

With t-SNE, 4 cluster and 2 cluster scenarios were performed with different amounts of features. At each trial, SelectKBest feature selection method was applied to the dataset to retain the most informative features. Subsequently, t-SNE was utilized for further dimensionality reduction, projecting the high-dimensional data onto a 3D space while preserving relationships. The reduced-dimensional data obtained from t-SNE was then used for GMM aiming to contain variant types in different clusters. The resulting clusters were visualized in a 3D plot, allowing for an intuitive interpretation of the cluster distribution. Additionally, a table displaying the counts of each class within each cluster was generated to assess the composition of the clusters. The 3D plot of the 4 cluster case is given in 6.15.

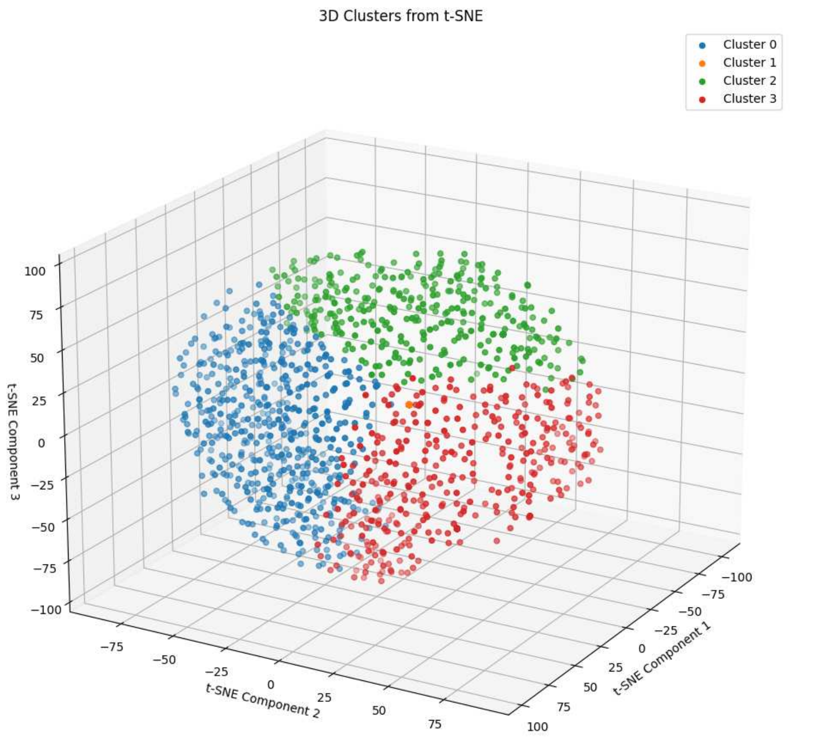


Figure 6.15: 3D Visualization of 4 Clusters with t-SNE

Comparing the results of 4-cluster scenarios with and without t-SNE, it was observed that the overall distributions of variant classes within clusters were similar between the two scenarios. However, there were notable differences. With 65 features, the t-SNE-enhanced clustering offered a slightly more balanced distribution of classes across clusters, with a more even spread of variants. Conversely, the previous case showed some clusters dominated by a single class (None), resulting in imbalanced distributions. This emphasized that the original scenario offered a better clustering in terms of the purity of the clusters, as a result of losing subtle patterns when compressing data into lower dimensions. Additionally, the stochastic nature of t-SNE and GMM introduced variability in clustering outcomes, contributing to inconsistent results across multiple runs.

The clustering results obtained using t-SNE followed by GMM with 2 clusters and 25 features selected by the SelectKBest, revealed a distribution of variants across clusters that closely resembled the pattern observed in the 2 cluster scenario without t-SNE. Despite the application of t-SNE for dimensionality reduction, the overall clustering pattern remained largely consistent, indicating that t-SNE did not introduce significant alterations to the clustering structure. Cluster 0 exhibited a composition similar to the initial two-cluster scenario, encompassing a mix of variants, including those categorized as DC, LP, and CF, with smaller proportions compared to the None class variants. Conversely, Cluster 1 predominantly consisted of variants classified as None, with a negligible presence of CF, DC, and LP variants (0.8%). The similarity in cluster composition suggests that while t-SNE effectively reduced the dimensionality of the feature space, it did not substantially change the clustering tendencies present in the dataset. This observation emphasized the robustness of the clustering approach and suggests that the underlying data structure driving the clustering pattern remained largely unchanged despite the application of t-SNE for dimensionality reduction.

In addition to the similarities observed in the clustering results between the t-SNE 2-cluster scenario and the original 2-cluster setup, it must be noted that the t-SNE approach required a reduced number of features to achieve comparable clustering outcomes. Specifically, the optimal number of features ranged from 10 to 30, suggesting that t-SNE effectively captured the essential variance in the data with a more compact feature set. However, despite the advantage of requiring fewer features, the utilization of t-SNE introduced a computational burden, significantly prolonging the runtime of the code execution. That was a result of an iterative optimization process for dimensionality reduction.

This analysis provided insights into the clustering patterns within the dataset and assessed the effectiveness of different clustering approaches.

6.2 Phenotype Classification

For phenotype assignment, different structures were used as explained in the section 5.4.2. The 3-dimensional nature of the data allowed the usage of 3D tensors and employing Neural Network based applications with PyTorch. Different experiments were conducted with different features and specific parameters. In this section, the significant results of these experiments are shared and supported with visuals.

In the phenotype prediction process, 7 NDDs were considered for each patient. Due to the inherent heterogeneity, diversity, and complexity of NDDs, seven distinct binary predictions were conducted for each disease and their outcomes were combined to assign diseases to individuals. This approach facilitated the determination of the disease set (combination) associated with each patient besides enhancing the interpretability and clinical relevance of the predictions. By employing individual predictors, model performance was optimized, class imbalance was addressed, and constant refinement and customization were facilitated to meet the unique needs of each disorder. Complexity caused by comorbidity was handled, simultaneously avoiding the bias of one disease to another.

A key objective of this phase was to remove features of pathogenicity prediction scores from the dataset. These scores, while initially informative, became redundant after the interpretation of the variant classification results. Notably, two main experiments were conducted for phenotype prediction purposes: one leveraging the actual variant labels provided by the BiocomputingUP Group, and another utilizing the predicted variant classes derived from the Logistic Regression model developed in this study, elaborated in section 6.1.2. The decision to employ the Logistic Regression model was influenced by the importance of incorporating variant labels into the phenotype prediction process. Using the model results, the aim was to exploit the rich information embedded within variant labels to inform the phenotype prediction model. By integrating variant labels into the phenotype prediction process the relationship between genetic variation and phenotype expression was investigated. Disease-causing variants, for instance, were more directly associated with NDDs and exhibited a stronger relationship with disease manifestation. Likely pathogenic variants demonstrated characteristics of potential harmfulness to contribute to disease development. Contributing factors encompassed variants that may modulate disease risk or phenotype severity, while neutral variants lacked associations with disease phenotypes. Notably, logistic regression emerged as the optimal choice for predicting variant classes with reliability, at same time avoiding overfitting. with an accuracy of 89% and an F1 score of 81%. These two experiments were performed to better simulate real-world scenarios and to obtain a more general model, both with real labels and predicted labels of variants. Subsequently, various methods were used, encompassing feature selection techniques and various neural network architectures, to enhance the accuracy and robustness of the phenotype prediction process.

6.2.1 Predicted Variant Labels

In the thesis section, **AutoKeras**, an automated machine learning framework, was used for structured data classification tasks. AutoKeras is an open-source Python library designed to automate the process of machine learning model selection, hyperparameter tuning, and architecture search, a field known as Automated Machine Learning (AutoML). It offers a user-friendly interface that reduces the complexities of traditional machine learning workflows, making it accessible to users with varying levels of expertise. It is built on top of TensorFlow and Keras and employs neural architecture search (NAS) techniques to systematically explore vast model architectures and hyperparameter spaces, automatically discovering optimal configurations tailored to specific datasets. By automating model training and optimization, AutoKeras accelerates the research process, efficient experimentation, and informed decision-making. It is particularly powerful for structured data classification tasks, where tabular datasets are organized into rows and columns.

In this task, AutoKeras was employed to build a structured data classifier. The Data was split into features and labels, and then further divided into training and testing sets. The training set was composed of 565 patients and there were 302 patient files used for the test set. The AutoKeras *'StructuredDataClassifier'* was initialized with key parameters such as *'max_trials'*, which specifies the number of different models to try during the optimization process. The training loop was iterated over a predefined number of epochs for each trial. After each epoch, the model's performance was evaluated, computing both the training and testing losses and accuracies.

The search approach was applied iteratively to develop optimized models for 7 distinct diseases. Each disease required a customized approach to the model due to variations in data characteristics, disease complexities, and diagnostic criteria. This approach reflected the recognition that disease classification tasks were inherently heterogeneous, and a multilabel model might not suffice to capture the relationships of each disease dataset. The results of this process are presented in a structured table format, indicating the specific model architectures and hyperparameters employed for each disease. Table 6.19 demonstrates this information and provides insights into the model structures. Additionally, it includes validation accuracies obtained for each disease model, offering a quantitative measure of model performance. This structured presentation enabled a comparative analysis of model performances across different diseases and made decision-making processes easier.

AutoKeras chose the reported structures after training 15 maximum networks for each dis-

Table 6.19: NN Structures and Performance Metrics with Predicted Labels

Disease	NN Architecture	Validation Accuracy	F1 Score
ID	2 layers; 32 neurons each	96.12%	51%
ASD	3 layers; 64, 32, 32 neurons	76.96%	69%
Epilepsy	2 layers; 64, 64 neurons	72.86%	58%
Microcephaly	2 layers; 64, 64 neurons	80.12%	69%
Macrocephaly	2 layers; 64, 64 neurons	66.85%	53%
Hypotonia	2 layers; 128, 32 neurons	77.23%	51%
Ataxia	2 layers; 64, 32 neurons	75.98%	59%

ease while running 10 epochs each time. The results from the selected networks showed varying performance across different diseases. For ID, the model achieved a high validation accuracy of 96.12% but a poor F1 score of 51%. This underlined the high imbalance for this disease dataset and motivated the use of resampling methods. Since almost all the patients had ID conditions in this study to detect the comorbidities with other NDDs, the predictions were highly biased towards positive class, i.e.'1'. For Microcephaly, Macrocephaly and Ataxia, the imbalance was towards the negative class, denoted as '0', due to the lack of patients with the disease in the dataset. However, the selected model for Microcephaly showed better performance compared to others, with a validation accuracy of 80%. Hypotonia prediction also yielded a good validation score, which can be explained by the lower level of imbalance it demonstrated. ASD and Epilepsy were relatively harder to predict, given the complexity of these diseases besides the imbalance. It was observed that the auto search provided relatively more complex NN structures to address these aspects for these diseases. Overall, while these selected models showed promising results for some diseases, It pointed out that further optimization was needed to improve the performance, especially for diseases with lower accuracy and F1 scores.

Resampling

To investigate more, after conducting AutoKeras trials for each of the seven diseases, an additional step involving undersampling was undertaken to address class imbalance within the dataset. Class imbalance posed a notable challenge in developing accurate predictive models for patients diagnosed with various diseases. Across disease categories such as ID, Microcephaly, Macrocephaly, Hypotonia, and Ataxia, the available data exhibited significant differences in the distribution of positive and negative class instances. For instance, while the ID category overwhelmingly consists of positive instances, comprising 97% of positive class, there were 11%

patients with positive class of Ataxia. To overcome this issue and ensure a balanced representation across classes during model training, undersampling and oversampling techniques were applied to adjust the number of examples from the majority classes. By addressing class imbalance through strategic resampling methods, the predictive models aimed to achieve greater accuracy and reliability in identifying patients with specific diseases, thereby increasing diagnostic capabilities in clinical practice.

The resampling was performed using the 'resample()' function from the scikit-learn library, both for undersampling and oversampling. Subsequently, balanced datasets were created, referred to as 'balanced_df'. Following this process, the models generated by AutoKeras were retrained on the newly balanced dataset to evaluate their performance under more equitable conditions. The results obtained from these datasets are given in Table 6.20, in the previously used format.

Table 6.20: NN Structures and Performance Metrics with Predicted Labels and Resampling

Disease	NN Architecture	Resampling	Val. Accuracy	F1 Score
ID	2 layers; 32 neurons each	50 from '1' class	73.96%	53%
ASD	2 layers; 128, 64 neurons	250 from '1' class	83.02%	71%
Epilepsy	2 layers; 64, 32 neurons	150 from '0' class	78.72%	60%
Microcephaly	2 layers; 128, 64 neurons	70 from '0' class	71.47%	56%
Macrocephaly	2 layers; 128, 64 neurons	65 from '0' class	70.36%	59%
Hypotonia	2 layers; 32 neurons each	110 from '0' class	79.52%	52%
Ataxia	2 layers; 32 neurons each	75 from '0' class	78.89%	61%

Generally, the implementation of resampling methods notably improved the performance metrics, leading to increased validation accuracy and F1 scores for classification. However, in the case of ID, the reduction in bias toward the majority class resulted in a significant decline in accuracy. This decline could be attributed to the previous occurrence of overfitting, where the model excessively tailored itself to the training data, compromising its ability to generalize to unseen examples. Additionally, another outcome that saw a decline in performance was associated with Microcephaly, exhibiting lower metrics. This decline stemmed from modifications in the network structure and hyperparameters. The model's capacity to accurately capture patterns and generalize was compromised, particularly in instances of undersampling. This emphasized the necessity for further refinement in the network architecture, despite AutoKeras's selection of the optimal hyperparameter combination. To address this need, true labels were introduced instead of predicted ones in the following experiments.

6.2.2 True Variant Labels

In continuation of the experiments, the same procedure was applied as explained before, this time integrating true variant labels into the structured data classification process. Using AutoKeras framework, the best model architecture was searched to handle the problem in the best way. Hyperparameter tuning was also a part of the search with AutoKeras, this way offering a user-friendly interface simplifying the machine learning workflow.

Integrating true variant labels and using them to boost the phenotype prediction process, as opposed to using predicted ones generated by logistic regression, granted several advantages in the context of NDDs. True labels provided direct insight into the functional consequences of genetic variants based on established knowledge and empirical evidence, representing ground truth information. This ensured greater accuracy and reliability in characterizing variant classes and their associations with disease phenotypes. While logistic regression was a powerful predictor of variant classes, it exhibited some bias and inaccuracies. Relying on true labels the risk of misclassification was reduced. Additionally, models trained on true variant labels were more likely to generalize well to unseen data and real-world clinical scenarios. By learning from genetic data, these models were able to find complex patterns and handle the heterogeneity of NDDs more accurately, and to adapt effectively to diverse patient populations and genetic landscapes. In summary, employing true labels instead of predicted ones with the logistic regression model, provided better accuracy, reduced uncertainty, increased interpretability, and improved generalization. The models with the best results of the experiments are given in Table 6.21.

Table 6.21: NN Structures and Performance Metrics with True Labels

Disease	NN Architecture	Validation Accuracy	F1 Score
ID	2 layers; 32 neurons each	96.42%	49%
ASD	3 layers; 64, 16, 16 neurons	70.27%	56%
Epilepsy	3 layers; 128, 64, 64 neurons	81.08%	48%
Microcephaly	2 layers; 64, 32 neurons	81.25%	57%
Macrocephaly	2 layers; 64, 64 neurons	89.58%	44%
Hypotonia	2 layers; 128, 128 neurons	74.13%	56%
Ataxia	2 layers; 64, 32 neurons	76.92%	59%

Introducing true labels caused a significant improvement in the performance metrics and in terms of model structures when they were compared with the models using predicted labels. In some cases, simpler models were chosen, with fewer layers and nodes, indicating the facilitation of making predictions with true labels. In the case of Epilepsy and Hypotonia, however, more complex model structures were selected for prediction, due to the complex natures of these diseases. This pointed out the biases introduced by the predicted variant labels to the model, affecting the behavior of the phenotype classification. Regarding ID, the recurring issue of overfitting persisted since the data was not handled to fix the class imbalance within the dataset.

Resampling

As in the previous experiments, resampling methods were used for each disease to address the class imbalance, which posed a challenge in correctly predicting the phenotypes. The same procedure of resampling was used, with the 'resample()' function from the scikit-learn library, both for undersampling and oversampling. The balanced data obtained in the previous resampling process was used in order to provide comparability and interpretability of the results. The results are shared in Table 6.22 in the previously used format, demonstrating NN structures, hyperparameters, and performance metrics.

Table 6.22: NN Structures and Performance Metrics with True Labels and Resampling

Disease	NN Architecture	Resampling	Val. Accuracy	F1 Score
ID	2 layers; 32 neurons each	50 from '1' class	78.07%	42%
ASD	2 layers; 128, 64 neurons	250 from '1' class	82.65%	64%
Epilepsy	2 layers; 128, 32 neurons	150 from '0' class	84.56%	66%
Microcephaly	2 layers; 64, 32 neurons	70 from '0' class	89.33%	70%
Macrocephaly	2 layers; 128, 64 neurons	65 from '0' class	90.01%	68%
Hypotonia	2 layers; 32 neurons each	110 from '0' class	86.12%	59%
Ataxia	2 layers; 32 neurons each	75 from '0' class	78.25%	76%

In this experiment, when the true labels were integrated into the models along with resampled datasets, the best model behaviors were observed, in terms of validation accuracy, F1 score, model complexity, and compilation time. As in the previous experiment with resampling, class '1' denoted the positive case, while class '0' denoted the absence of the disease, representing patients without the condition. The balancing quantities were determined by analyzing the distribution of positive and negative classes within the disease datasets for phenotype, as elaborated in Section 4.2.

ID class had a lower performance since undersampling methods truncated the dataset while providing balance and avoiding overfitting. Models lacked the data needed to learn the natural relationships with variant labels and phenotypes. In the end, more data was needed for a general diagnosis of this disease, but a reasonable performance was achieved. In terms of F1 scores; Hypotonia, ASD, and Epilepsy continued to have relatively poor performances, due to the lack of real-world clinical data and complex, heterogeneous attributes of these diseases. Ataxia produced more stable and consistent results throughout the experiments, having close accuracy value and F1 score.

6.2.3 Test Data Results

After conducting experiments with both predicted and true variant labels, the optimal network structures from each case were selected to construct neural networks for the test data. However, unlike previous iterations, the training sets were not resampled to integrate with the test data. This decision aimed to simulate a more realistic scenario that is closer to real-world conditions, allowing for an assessment of model performance without artificial adjustments.

Fundamentally, by using the full dataset without resampling, the study sought to evaluate the models' ability to generalize to unseen data and adapt to real-world variability. This approach is crucial as it provides insights into how well the models perform under more natural conditions, in a new cohort of patients, where data distributions may not be perfectly balanced or representative of the entire population.

Furthermore, adjustments were made to the chosen network structures, including fine-tuning epoch numbers and batch sizes, to optimize performance based on the non-resampled data. This iterative process ensured that the models were robust and effective in handling the inherent complexities present in clinical datasets. In summary, by employing the full dataset without resampling, making necessary adjustments to the network structures, and using the best model structures obtained in previous experiments, the study aimed to provide a more accurate evaluation of the models' performance in a realistic setting, ultimately improving their applicability and reliability in clinical scenarios. Test data included 302 patients and for each disease, the models were constructed with the decided architecture, then they were run iteratively to make final adjustments. A comparison of test accuracies obtained with this approach, for predicted labels and true labels can be seen in the following table 6.23, moreover, a graph is shared in Figure ?? for a better visualization of the results.

The results were consistent with the previous experiments; true labels facilitated the prediction of phenotypes and led to higher accuracies except in the cases of Macrocephaly and Ataxia. This could be explained by significant class imbalances and the lack of real-world data points

Table 6.23: Test Accuracies with Predicted and True Labels

Disease	Accuracy with Predicted Labels (%)	Accuracy with True Labels (%)
ID	97.46	98.28
ASD	59.85	64.86
Epilepsy	66.32	72.03
Microcephaly	72.12	75.19
Macrocephaly	70.07	64.91
Hypotonia	63.67	69.77
Ataxia	71.46	70.81

for these diseases. Predicted labels produced broader example sets for the variant classification. This approach attempted to compensate for missing patient diagnoses, constructing a more generalized model. However, this broader generalization came with a trade-off with the accuracy of phenotype prediction. Practically, while predicted variant labels expanded the scope of the model, they introduced a degree of uncertainty, impacting the precision of the predictions. For the case of ID, the results did not reflect a realistic scenario due to the overfitting issue, since resampling methods were not applied. Thus, the high accuracy was not reliable and another dataset was needed for further training and testing. This way, phenotype prediction was performed for both scenarios in a clinical setting. The conclusion of the study and important remarks, together with the suggestions for future work are shared in the following chapter.

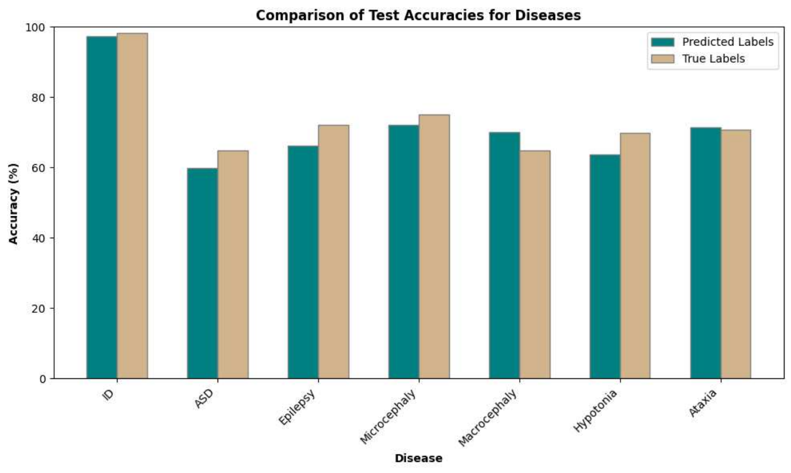


Figure 6.16: Comparison of Test Accuracies of The Two Approaches

7 Conclusion

This thesis aimed to employ machine learning models for predicting seven Neurodevelopmental Disorders and identify potentially causative variants associated with those. This was a challenging task, due to the heterogeneous and multi-faceted nature of these diseases. Comorbidity, which is described as the simultaneous presence of multiple medical conditions, also contributed to this complexity. The study also investigated the relationship between patient phenotypes and variants, by observing the effects of different variant types on phenotypes.

The set-up of this project was constructed by the BioComputing UP Group of Padua University, which was the assessor and data provider of the CAGI Challenge, to achieve the aforementioned task. This thesis project was a part of an internship, also conducted in the laboratory of the same group.

For the study, genetic data from 867 patients was used, combining the previous works (metadata) used for this purpose (CAGI). As opposed to manual variant filtering and phenotype assignment procedure as commonly followed in clinical methodologies, the aim was to contribute to the development of an automated tool that requires minimum human intervention and achieves the highest accuracy.

The metadata was used to retrieve important data from public databases about human genome references, with a process called Annotation. ANNOVAR tool was for annotation, to access and embed data into specific files that were useful for the study. Then, data cleaning and preparation were performed very carefully to avoid discrepancies, improve efficiency, and ensure compatibility with the machine learning models. This way, the data became ready for usage for all purposes and models.

The project had two phases: classifying the variants present in a specific gene panel of 74 genes related to NDDs and assigning phenotypes to patients depending on this classification. The variant classification phase included the application of different machine learning models, both supervised and unsupervised, trying to discover the most accountable and generalizable model. The labels obtained from the supervised learning were used for phenotype prediction while the experiments with unsupervised models were conducted to understand the complexity of the task and to demonstrate a real-world setting.

Among the supervised learning models; random forest, logistic regression, SVM, and Neural Networks were used, each with the same data but with a different set-up for finding the best model that can classify the variants into 4 categories: disease-causing, likely pathogenic, contributing factor, and neutral (none). Also, some statistical feature selection methods and data resampling techniques were employed to improve the performance. Throughout the experiments, feature selection methods such as SelectKBest, SelectKPercentile, Mutual Information, and Feature Importance (random forest) were employed. SelectKBest method (using ANOVA F-statistic) for feature selection seemed to be the best method to decide on the most relevant features for the variant classification. First, models were run with optimum hyperparameters obtained with grid search to constitute a baseline for further experiments. Then, features were reduced by the selection methods. Finally, resampled data was integrated to improve the performances.

The logistic regression model provided the most reliable results in variant classification, demonstrating a high accuracy and avoiding overfitting. For this study, F1 scores were primarily trusted in measuring performance since the issue of overfitting was present. The reason for overfitting was the class imbalance both in the variant and phenotype data, and the missing data points in the retrieved data. Logistic regression had an average F1 score of 81% and 89% of overall accuracy, using 125 features selected by SelectKBest method and 400 samples from the majority class, i.e. 'None'. This class was the best-predicted class as It was in all the experiments, due to its excessive presence. Logistic regression obtained an average AUC value of 95% among the variant classes. It also predicted the LP class with a precision of 73%, being the highest one among the supervised models. Random forest followed logistic regression by yielding a similar result, with an overall accuracy of 84% and an F1 score of 80%. However, logistic regression was preferred since it performed better with more data, proving to be a more generalizable model. Random Forest gave these results by using 200 samples from the majority class only, indicating a decrease in the performance when the number of samples increased. Also, the neural network was the model that required the least number of features, i.e. 65 selected by SelectKBest method, to explain the target variable, but it was not preferred because of an F1 score of 78%.

For unsupervised learning, K-Means and Gaussian Mixture Models were used to achieve a good separation of different variant types in different clusters. Potentially causal variants were targeted to identify, expecting to have an automatic process that facilitates manual handling in clinical studies. Along with 4-cluster and 3-cluster scenarios, the 2-cluster classification achieved the best performance, since the other scenarios required more distinction and human intervention. In the end, K-Means classified 82.2% of the harmful variants in one cluster, with contamination of the dominant 'None' class. GMM achieved almost the same result with only 3 more variants classified correctly in the 2-cluster scenario. This result was reasonable and highly satisfactory considering the manual processes and curation needed for variant interpretation in the field of molecular biology. Although this clustering did cause the loss of 43 variants in the 'None' cluster, it helped reduce the entire variants set 81.4%, by clustering 82.2% of important variants in the same cluster. Also, these unsupervised models demonstrated better generalization compared to supervised models in certain contexts, since they did not depend on labeled data and were able to discover unseen patterns. Because of this aspect, these models can be applied to even larger datasets, which are not limited by a gene panel or NDDs, and possibly facilitate the variant interpretation process by saving time and effort.

For the phenotype classification phase, two approaches were followed: using predicted labels with the logistic regression model and using the true labels already at hand. Using predicted labels showed the feasibility of a fully automated pipeline and a more realistic scenario, sacrificing accuracy, while benefiting the true labels improved the model performance. ASD was the hardest to predict, the ASD model could reach an accuracy of 64% with true labels. Macrocephaly and Hypotonia followed ASD with 64% and 69% of accuracy with true labels. Also, Epilepsy was hard to predict with labels obtained from logistic regression. Considering the genetic complexity of ASD and Epilepsy, and the difficulty of diagnosing these disorders even clinically, justifies the poor performance of the models. In the cases of Hypotonia and Macrocephaly, their poor performances can be explained by the insufficient number of examples available for training. This lack was particularly notable because of the hunger of neural networks for data, as they tend to exhibit better performance when trained on larger and more diverse datasets.

This project constituted a baseline for potential future research and continuous improvement for a very intriguing field focused on finding pathogenic variants and revealing the phenotype contributions. Considering the acquisition of more clinical data, expanding the sequencing to the entire exome or genome of the patients, and improving the annotations for intronic variants, splicing variants, and other exon-intron junction variants, a better decision-making process can be achieved. Also, the workflow could be even more facilitated with automatic pipeline creation and file-handling services such as Galaxy, which is an open-source platform for creating workflows for bioinformatics research. This integration is proposed to allow less human involvement and faster file processing.

In conclusion, this project showed some ways to facilitate variant interpretation and phenotype prediction in an NDD environment, automating and semi-automating the process by employing a combination of machine learning models. The lack of patients with diseases and natural imbalance observed in variant types, since most of the SNVs of an individual are neutral, made this challenge very difficult. Especially variant class 'LP' was the hardest to predict since it falls between disease-causing and contributing factors. This gray area still needs human interpretation, unless unsupervised methods are used for broader classification. The proposed models, logistic regression for variant classification and neural networks for phenotype assignment, performed significantly better than most of the predictor groups of the CAGI Challenge. For CAGI6, 8 groups submitted 29 predictions for variant and phenotype classification. The methodologies employed by these groups were not known in principle, since the results of CAGI6 have not been published yet, and groups did not share much about their model specifications in CAGI5. However, it is known that most of them followed manual approaches and some of them declared utilizing Random Forests for classification tasks. The proposed models have surpassed all the predictor group submissions in terms of AUC values except for ASD, Epilepsy, and Macrocephaly cases. And for those, competitive levels were reached. This showed the power of automated models and the trustworthiness of a systematic, data-driven approach over a manual approach.

References

[1] K. J. Mitchell, “The genetics of neurodevelopmental disease,” *Current Opinion in Neurobiology*, vol. 21, no. 1, pp. 197–203, 2011, developmental neuroscience. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0959438810001297

[2] M. Carraro, A. M. Monzon, L. Chiricosta, F. Reggiani, M. C. Aspromonte, M. Bellini, K. Pagel, Y. Jiang, P. Radivojac, K. Kundu, L. R. Pal, Y. Yin, I. Limongelli, G. Andreoletti, J. Moult, S. J. Wilson, P. Katsonis, O. Lichtarge, J. Chen, Y. Wang, Z. Hu, S. E. Brenner, C. Ferrari, A. Murgia, S. C. Tosatto, and E. Leonardi, “Assessment of patient clinical descriptions and pathogenic variants from gene panel sequences in the cagi-5 intellectual disability challenge,” *Human Mutation*, vol. 40, no. 9, pp. 1330–1345, 2019. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/humu.23823

[3] M. C. Aspromonte, M. Bellini, A. Gasparini, M. Carraro, E. Bettella, R. Polli, F. Cesca, S. Bigoni, S. Boni, O. Carlet, S. Negrin, I. Mammi, D. Milani, A. Peron, S. Sartori, I. Toldo, F. Soli, L. Turolla, F. Stanzial, F. Benedicenti, C. Marino-Buslje, S. C. Tosatto, A. Murgia, and E. Leonardi, “Characterization of intellectual disability and autism comorbidity through gene panel sequencing,” *Human Mutation*, vol. 40, no. 9, pp. 1346–1363, 2019. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/humu.23822

[4] N. Whiffin, A. M. Roberts, E. Minikel, Z. Zappala, R. Walsh, A. H. O’Donnell-Luria, K. J. Karczewski, S. M. Harrison, K. L. Thomson, H. Sage *et al.*, “Using high-resolution variant frequencies empowers clinical genome interpretation and enables investigation of genetic architecture,” *The American Journal of Human Genetics*, vol. 104, no. 1, pp. 187–190, 2019.

[5] T. Fawcett, “An introduction to roc analysis,” *Pattern recognition letters*, vol. 27, no. 8, pp. 861–874, 2006.

[6] M. C. Aspromonte, A. Del Conte, S. Zhu, W. Tan, Y. Shen, Y. Zhang, Q. Li, M. H. Wang, G. Babbi, S. Bovo, P. L. Martelli, R. Casadio, A. Althagafi, S. Toonsi, M. Kulmanov, R. Hoehndorf, P. Katsonis, A. Williams, O. Lichtarge, S. Xian, W. Surento, V. Pejaver, S. D. Mooney, U. Sunderam, R. Srinivasan, A. Murgia, D. Piovesan, S. C. E. Tosatto, and E. Leonardi, "Cagi6 ID-Challenge: Assessment of phenotype and variant predictions in 415 children with Neurodevelopmental Disorders (NDDs)," 2023. [Online]. Available: https://www.researchsquare.com/article/rs-3209168/v1

[7] I. Thiffault, M. Cadieux-Dion, E. Farrow, R. Caylor, N. Miller, S. Soden, and C. Saunders, "On the verge of diagnosis: Detection, reporting, and investigation of de novo variants in novel genes identified by clinical sequencing," *Human Mutation*, vol. 39, no. 11, pp. 1505–1516, 2018. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/humu.23646

[8] S. Zaidi, M. Choi, H. Wakimoto, L. Ma, J. Jiang, J. D. Overton, A. Romano-Adesman, R. D. Bjornson, R. E. Breitbart, K. K. Brown, N. J. Carriero, Y. H. Cheung, J. Deanfield, S. DePalma, K. A. Fakhro, J. Glessner, H. Hakonarson, M. J. Italia, J. R. Kaltman, J. Kaski, R. Kim, J. K. Kline, T. Lee, J. Leipzig, A. Lopez, S. M. Mane, L. E. Mitchell, J. W. Newburger, M. Parfenov, I. Pe'er, G. Porter, A. E. Roberts, R. Sachidanandam, S. J. Sanders, H. S. Seiden, M. W. State, S. Subramanian, I. R. Tikhonova, W. Wang, D. Warburton, P. S. White, I. A. Williams, H. Zhao, J. G. Seidman, M. Brueckner, W. K. Chung, B. D. Gelb, E. Goldmuntz, C. E. Seidman, and R. P. Lifton, "De novo mutations in histone-modifying genes in congenital heart disease - Nature," may 12 2013.

[9] P. T. van Doormaal, N. Ticozzi, J. H. Weishaupt, K. Kenna, F. P. Diekstra, F. Verde, P. M. Andersen, A. M. Dekker, C. Tiloca, N. Marroquin, D. J. Overste, V. Pensato, P. Nürnberg, S. L. Pulit, R. D. Schellevis, D. Calini, J. Altmüller, L. C. Francioli, B. Muller, B. Castellotti, S. Motameny, A. Ratti, J. Wolf, C. Gellera, A. C. Ludolph, L. H. van den Berg, C. Kubisch, J. E. Landers, J. H. Veldink, V. Silani, and A. E. Volk, "The role of de novo mutations in the development of amyotrophic lateral sclerosis," *Human Mutation*, vol. 38, no. 11, pp. 1534–1541, 2017. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/humu.23295

[10] J. A. Acuna-Hidalgo, Rocio Veltman and A. Hoischen, "New insights into the generation and role of de novo mutations in health and disease," *Genome Biology*,

vol. 17, no. 241, 2016. [Online]. Available: https://genomebiology.biomedcentral.com/articles/10.1186/s13059-016-1110-1

[11] M. Kimura and T. Ohta, "The average number of generations until fixation of a mutant gene in a finite population," *Genetics*, vol. 61, pp. 763–71, 04 1969.

[12] M. Baillie, S. le Cessie, C. O. Schmidt, L. Lusa, and M. Huebner, "Ten simple rules for initial data analysis," *PLOS Computational Biology*, vol. 18, no. 2, pp. 1–7, 02 2022. [Online]. Available: https://doi.org/10.1371/journal.pcbi.1009819

Acknowledgments

I would like to express my gratitude to my colleagues at BioComputing UP Group, for providing me with the opportunity to conduct this internship and project, and also for helping me to realize my dream of being included in the scientific community. I am deeply grateful to my supervisor, Emanuela Leonardi, for her mentorship, support, time, and encouraging feedback throughout the entire process. She has been more than a supervisor to me, a guide in my journey of discovering my potential. Also, I sincerely appreciate Silvio Tosatto, Damiano Piovesan, and Alexander Miguel Monzon for introducing the field of bioinformatics to me and for motivating me to pursue this path. My heartfelt thanks go to my parents, for all their love and care, for making me who I am right now, for always believing in me and supporting me during this journey. I cannot thank them enough for their unending endeavors to make me a better person, for exemplifying good moral values to me, and for teaching me to be independent and strong. At this moment, I extend my gratitude to everyone who has contributed to my growth. I am grateful for each good memory I shared with my colleagues and peers. Each of them holds a unique place in my heart, and I will remember them with happiness.