Final Degree Project

Biomedical Engineernig Degree

“Exploring machine learning approaches for phenotype prediction of Huntington's disease’’

Barcelona, 5th of May, 2024

Author: Caterina Fuses i Kuzmina

Director: Jordi Abante Llenas

Tutor: Josep Maria Canals Coll

Abstract

Huntington's disease onset of symptoms is clinically predicted primarily using the length of the CAG trinucleotide expansion in the HTT gene. However, this prediction can only explain around 50% of the variability of the phenotype. It is estimated that 40% of the remaining variability is heritable, suggesting the presence of other genetic factors. Genome Wide Association Studies (GWAS) have identified potential genetic modifiers, although only through the revelation of linear effects and via computationally demanding processes.

This project benchmarks various machine learning algorithms trained with an Enroll-HD GWAS dataset to predict the age at HD onset. The dataset comprises the genotype of millions of SNPs from approximately 9,000 individuals. The models considered include regularized linear models (Lasso and Elastic Net) and tree-based models (Random Forest and XGBoost), and their predictive power is compared to an Ordinary Least Squares baseline model trained solely with sex and CAG as covariates. The results indicate that tree-based models achieve the best estimation of age of onset (AO), improving the prediction by 3% with respect to the baseline, possibly due to their implicit consideration of interactions between SNPs. For each model, we extract the most significant features contributing to the model, thereby identifying genetic modifiers. Some of these key SNPs are in well-known AO modifier candidates such as *FAN1* and *MYT1L*, while others are in genes like *CDYL2* proposed as new candidates.

Keywords: Machine Learning, High Dimensional Data, Huntington's Disease, Single Nucleotide Polymorphism.

Acknowledgements

The most important person to acknowledge in this project is my director, Jordi Abante. The project originated from his innovative idea for a novel research path to develop a better AO prediction algorithm. I am deeply honored to have contributed to the early stages of this endeavor with this initial exploration. My utmost gratitude goes to him for his constant guidance and indispensable insights at every step. I have enjoyed working with him enormously and am very grateful to have completed this project under his supervision.

List of Figures

List of Tables

List of Abbreviations

HD	Huntington's Disease
AO	Age of Onset
GeM	Genetic Modifier
ML	Machine Learning
SNP	Single Nucleotide Polymorphism
MSN	Medium-sized spiny neuron
GWAS	Genome-Wide Association Studies
GeM-HD	Genetic Modifiers of HD
LD	Linkage Disequilibrium
MRI	Magnetic Resonance Imaging
RSS	Residual Sum of Squares
GLM	Generalized Linear Model
CLT	Central Limit Theorem
OLS	Ordinary Least Squares
MAE	Mean Absolute Error
MSE	Mean Squared Error
Rsid	Reference SNP cluster ID
CSR	Compressed Sparse Row
WBS	Work Breakdown Structure
PERT	Program Evaluation and Review Technique
GDPR	General Data Protection Regulation
SaMD	Software as a Medical Device

Contents

1 Introduction

Huntington's disease (HD) is a hereditary neurodegenerative disease whose first symptoms can appear at different points of a lifetime. The age of disease onset correlates strongly with the length of the mutation related to the disease, a CAG trinucleotide expansion in the huntingtin gene (*HTT*) [1], [2]. Genetic testing and clinical prediction for age of onset (AO) both rely on the length of this expansion. Nevertheless, this is not a perfect predictor, as the standard deviation of AO at a specific CAG repeat length is quite large, specially for short expansions (see Figure 1). Expansion length accounts for 40-70% of the variability of AO. The remaining variance is considered to be due to environmental and other genetic factors, showing a high degree of heritability [3], [4]. Large genetic studies have been done during the last two decades and are still ongoing in the search for genetic modifiers (GeMs) of AO, genes or genetic elements involved in the process of disease onset either by accelerating or delaying the emergence of HD's first symptoms [5]. The main AO modifier candidates are related to DNA maintenance machinery, through their mechanisms are still being studied [6].

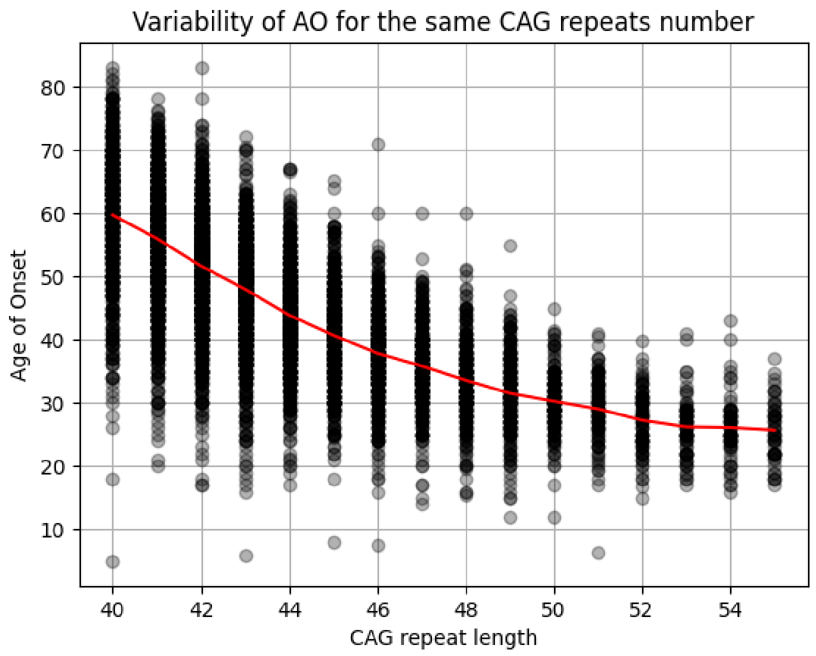


Figure 1: Inverse correlation of age of onset and CAG repeat length observed in the Enroll-HD dataset used in this project.

Phenotype prediction in HD is not only interesting from a clinical point of view for life planning addressed to mutation carriers. It can also provide valuable insight of the disease mechanisms that generate such phenotype, and apply this knowledge into the design of clinical trials, where the effect of putative modifiers of HD pathogenesis needs to be controlled for the effect of genetic background [7]. Knowing more about disease onset mechanisms can also reveal possible targets for treatment that could delay onset of symptoms [2].

1.1 Objective

The main research objectives of the thesis, by order of relevance, are:

1. Benchmark different machine learning (ML) regression models on how well can they predict AO in HD using SNP genotyping data.

2. Identify which genetic variants are most important in predicting AO and which processes are they involved in.

1.2 Methodology

The workflow of the project can be divided into four main parts, which are not strictly sequential. First, a comprehensive literature review is conducted to understand the background and context in which the project is situated. This is followed by data preprocessing, where the raw data is tailored and prepared for model training. Subsequently, ML models are selected and trained to analyze the preprocessed data. Finally, the results are discussed, offering insights into the performance of the models and their implications for the study.

Models are going to be trained with data provided by Enroll-HD, genetic data containing the genotype of millions of Single Nucleotide Polymorphisms (SNP) for thousands of HD patients. The dataset was assembled by Lee et al. [8] in June 2023. Data handling will be done with different programming languages, depending on the specific needs (e.g., Python, R, bash, C++). The models themselves will be coded and trained using Python and several open-source packages for easy implementation of ML algorithms.

1.3 Scope

This project is an exploratory approach to phenotype prediction. The use of (ML) is not intended to create a new prediction tool but to review how well existing models can predict the onset of the disease based solely on genetic data. The results obtained with the final trained models will be compared to published studies that propose candidates for HD GeMs. This comparison will serve as a check for positive control discoveries and provide an alternative method for GeM discovery.

1.4 Limitations

The limitations of the project stem from several challenges across its various stages. First, the vast amount of data requires significant computational resources for preprocessing and analysis, which might not be readily available. Secondly, the choice of feature engineering methods, such as treating each SNP individually or aggregating them at the gene level, presents a trade-off between specificity and dimensionality reduction, potentially masking important polymorphisms when reducing dimensions. Additionally, selecting the right subset of SNPs impacts the discovery potential and computational feasibility. The high dimensionality of data also impacts the training of ML models, which can led to models which suffer from high variance.

2 Background

2.1 Huntington's disease

HD is an autosomal dominant neurodegenerative disease. At disease onset an irreversible progression of motor, psychiatric and cognitive symptoms starts. As there is currently no cure, this progression inevitably spans over 15-20 years, ending with death. In most cases the first manifestations of HD appear in mid-life. Prior to this onset, called "manifest HD", a "prodromal" phase may happen during some years where more subtle symptoms start appearing [9]. During this time, the striatum suffers a progressive a neuronal loss, finally leading to the development of manifest HD.

Clinical symptoms of manifested HD are very heterogeneous, even within families. The most distinct motor symptom is chorea: involuntary, excessive movements whose amplitude increases with disease progression. At more advanced stages, bradykinesia and rigidity dominate the motor symptoms. Cognitive impairments that appear are related to visual attention, psychomotor speed, and visuomotor and spatial integration, executive capacity, and short-term memory. The most common psychiatric symptoms are depression and anxiety, which can occur in premanifest HD, followed by apathy, irritability, and obsessive, compulsive thoughts and behaviours [9]. Pneumonia and suicide are the most common cases of death in HD [10].

HD pathology is cytologically located mainly in the striatum. Medium-sized spiny neurons (MSNs) of the striatum which use γ-aminobutyric acid (GABA) die, and cortical pyramidal neurons that project to the striatum and striatal neurons projecting to the substantia nigra degenerate [11]. The normal distribution of glial cells is also affected [12]. The overall brain volume is reduced, different brain compartments showing different rates of loss [13].

In 1993, the mutation related to this disease was identified in the huntingtin gene (*HTT*) located in the short arm of chromosome 4: a CAG trinucleotide expansion in its exon 1, a trinucleotide which is transcribed into glutamine [14]. This finding indicated that genetic testing could be an accurate and specific diagnostic test for HD. The same study revealed that there was a wide range of AO for any specific repeat number, although larger expansions correlated with juvenile onset, extreme rigidity and more widespread neuropathology. Now it is estimated that only around 50-70% of variance of AO can be explained by CAG repeat length. The remainder is accounted for by environmental and individual genetic background [2], [3], [15].

The exon where the expansion mutation is located encodes a polyglutamine tract near the amino terminus, an alpha-helical solenoid-like scaffold. Repeats in this expansion up to 35 CAGs are found in the general population, while length polymorphisms exceeding this number can cause HD, affecting the structure, phosphorylation pattern and activities of the protein [16]. Depending on the length of the expansion, the allele is usually classified into normal, reduced penetrance and full penetrance allele (see Table 1).

This mutation is inherited following an autosomal dominant pattern, meaning that if

Allele classification	**nº CAG repetitions**	**Expression**
Normal Allele	<27	Not associated with a phenotype and are inherited in a stable manner.
High Normal/ Intermediate Allele	27-35	Do not typically develop signs of HD but may show some degree of germline instability of the CAG repeat.
Reduced Penetrance HD Allele	36-39	Associated with HD, but not all individuals with these alleles will develop clinical symptoms.
Full Penetrance HD Allele	>39	Almost always associated with development of clinical signs and symptoms of HD.

Table 1: Classification of CAG expansion alleles in *HTT* [16].

either parent carries the CAG repeat expansion, the child has a 50% chance of inheriting it. It is also an unstable mutation: the length of the expansion can increase when passed from parent to child, being unstable in 80% of intergenerational transmissions. The higher size changes usually occur in the male germline [17]. Due to this instability, parents with CAG allele in the intermediate range which is not causing symptoms might pass a disease-causing range allele to their offspring. Decreases in size seldom occur. Expansion of HTT alleles can happen somatically as well as gametically, being prevalent in brain regions that are most susceptible to neurodegeneration [18]. Somatic expansion is predicted to accelerate the disease process [19], [20].

Backing the dominant behaviour, heterozygotes which have two expanded alleles with different sizes do not seem to follow a different pattern of onset than heterozygous when taking the larger allele for comparison. This means that the effects this mutation has are caused by the longest expanded allele, the dominant one [21]. However, there are some studies indicating that there could be an interaction between the two alleles: among HD subjects with large fully penetrant alleles, the length of the unexpanded CAG repeat was positively associated with a delayed HD onset [4]. Whether having two mutant alleles means displaying a more severe phenotype is still under discussion [2].

Upon further research on what other factors could be involved in HD manifestation and development, HD modifiers are still being discovered and described. Up until this moment, the majority of proven modifiers are DNA maintenance genes. The correlation between HD modifiers and the disease's precise toxicity mechanisms are yet to be discovered [16].

The protein's activity is altered in other neurological diseases apart from HD [22]. Its function is not completely known, but it is involved in brain development, transcription regulation, balance of histone acetylation and deacetylation, glial activation, mitochondrial functions, axonal transport, signaling pathways regulatization, and autophagy[2].

The mutant *HTT* affects the regulation of transcription factors, impacts chromatin regulation, impairs mitochondrial functions, alters protein homeostasis, affects axonal trafficking, dysregulates glutamatergic signaling, induces synaptic plasticity failure and glial activation [2], [23]. The mutant version of huntingtin is misfolded and forms aggregates with toxic properties, whereas in unaffected individuals it is localized in a diffused fashion around the cell cytoplasm [24]. Polyglutamine tracts with 37 or more glutamines (almost coinciding with the fully penetrant allele) self-assemble into such aggregates in experiments in vitro. The formation of this nucleus is dependant on time and protein concentration, and the rate of aggregate formation directly correlates with repeat length [25]. Mutant *HTT* also can co-aggregate with other proteins reducing their availability. Both toxic gain-of-function of the mutant *HTT* and loss of the wildtype *HTT* functions contribute to the disease phenotype.

2.2 SNP genotyping

2.2.1 Single nucleotide polymorphisms

A SNP is a described variation at a single position in a DNA sequence among individuals, only named as such if the variation is found in more than 1% of a population [26]. For every SNP there is a reference allele, which is the nucleotide present in the majority of the population, and one or multiple alleles, which are the minoritary nucleotides found in that position.

2.2.2 Genome-wide association studies

The primary use of Genome-Wide Association Studies (GWAS) is to identify genetic variations associated with complex traits or diseases in populations. GWAS scan the entire genome of individuals to find variations such as SNPs that occur more frequently in people with a particular trait or disease (referred to as phenotype) compared to those without it. This way enhancers and suppressors of selected phenotypes are identified.

Given the lack of variance explained in the phenotype by the CAG repeat length, the HD research community started the search for HD modifiers. When GWAS became technologically possible, large human molecular genetic tests enabled an unbiased search for modifier loci (position on a genome). The HD MAPS study by Li et al. (2003) [3] was a large linkage analysis on a genome scan which found evident linkage at chromosomes 4 and 6, near the localization of potential modifiers *GRIK2*. A combination of three GWAS carried out by the Genetic Modifiers of HD (GeM-HD) Consortium in 2015 identified important loci on chromosomes 8 and 15 that accelerated or delayed onset with respect to the mean [5]. Genes in these loci have been confirmed as HD modifiers years later through biological models: *FAN1*, in chromosome 15, a gene involved in DNA repair; and *RRM2B*, in chromosome 8, a ribonucleotide reductase [16]. Further studies revealed new modifier loci in other chromosomes; the most recent one by Lee, MacDonald and Gusella (2022) [8] has identified modifier loci in chromosome 7 too, with a greater power analysis by including motor and cognitive measures as additional phenotypes. The potential of such studies is still being exploited.

2.2.3 Linkage disequilibrium

An important fact to consider when relating SNPs to a certain phenotype is the linkage disequilibrium (LD), the non-random association of alleles at two or more loci: the allele of one polymorphism in an LD block, also known as haplotype, can predict the allele of the other polymorphisms in the block [27]. The size of the LD blocks depends primarily on the recombination rate in the region where they are found [28].

2.3 State of the art: Phenotype prediction in HD.

HD prediction in presymptomatic patients started for research purposes, as studying the evolution of the disease in already manifested HD was an important research limitation. The discovery of the trinucleotide repeat expansion in 1993 gave the basis for a highly accurate and easy to perform test, without the need of family samples [14]. It then evolved into a clinical service accompanied by counselling and support protocols which were later on internationally standardized [29].

One of the most spread model for age at HD onset is the Langbehn formula, derived from a parametric survival model [30]. This survival model was created with CAG repeat length specific survival functions, that take into consideration not only the CAG length but the current age of the case at hand. The resulting model can be expressed through a formula with parameters age and CAG that computes the probability of surviving without neurological symptoms until at least the given age. For example, a 40 year old individual without symptoms and a CAG repeat size of 44 will have a probability of onset by age 50 years, 10 years into the future, of 0.7; in an individual of this same age but with a repeat size of 44, the probability of onset by age 50 is 0.95. These probabilities are given with a 95% confidence interval for CAG lengths between 41 and 56, as these are the lengths of the samples used to build the model. Predictions for CAG beyond these limits are extrapolations. Figure 2 shows the cumulative probabilities for different CAG lengths of the model.

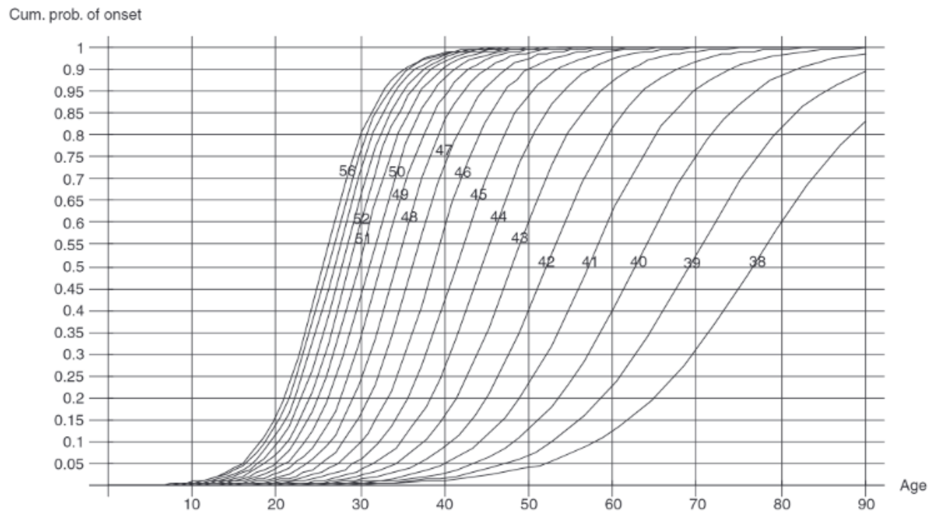


Figure 2: Cumulative probability of onset of HD for various CAG lengths based on the Langbehn model. Figure by Langbehn et al. [30].

There are other published statistical models that fit relationships between CAG length and clinical onset. There are two main approaches: using some sort of linear regression, or as the Langbehn model, using modeling techniques particular to survival analysis. Regressions tend to provide overly pessimistic estimates of AO specially for shorter CAG lengths, because they do not considerate cases which never reach HD diagnosis. Survival analysis accounts for samples that do not reach onset [31]. All models struggle predicting onset for the shortest expansions, due to either statistical biases or unrepresentative sample sets because subjects in this range are mostly non-symptomatic and are rare in clinical samples. Figure 3 overlaps the mean AO predicted for different CAG lengths by different models.

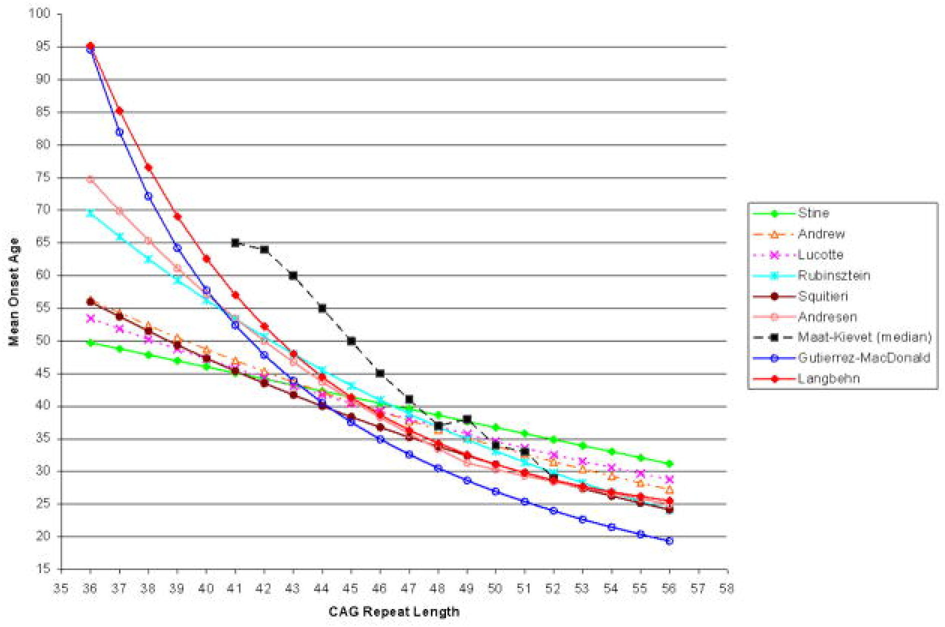


Figure 3: Mean as estimated by various published formulae. Figure by Langbehn, Hayden, Paulsen et al. [31].

The PREDICT-HD study was a multi-site observational study ongoing for more than a decade at the start of the century aimed at identifying biological and refined clinical markers of early HD in humans, to validate them for use in preventive clinical trials. Outcome measures include basal ganglia volumes on magnetic resonance imaging (MRI), estimated probability of diagnosis based on CAG length, performances on 21 standardized cognitive tasks, total scores on 3 scales of psychiatric distress, and motor diagnosis based on the Unified Huntington's Disease Rating Scale [32]. Studies done over the data generated by this study try to relate biological and clinical measures to the estimation of years to diagnosis, to better understand how the disease symptoms start manifesting and evolve. Estimated time to diagnosis is correlated to most clinical and neuroimaging markers [33]. Detectable changes were seen 10-20 years before the predicted age of clinical diagnosis. Structural MRI scans alone have been proven sufficient to separate presymptomatic disease gene mutation carriers and prodromal from controls [34], [35].

2.4 Machine Learning

ML is a branch of artificial intelligence that focuses on the development of algorithms and statistical models that enable computers to perform tasks without explicit instructions. It is considered a *knowledge discovery* technique, together with statistics, as it is an approach to characterize and predict complex phenomena described by a set of variables [36].

In the context of the study of biological systems, the two major goals pursued are inference and prediction. Inference refers to the capability of creating a mathematical model of the process creating the studied data in order to understand such process or test a hypothesis about its behaviour. Prediction is focused in forecasting unobserved outcomes, without necessarily understanding the underlying mechanisms of the involved process. Both statistics and ML are methods that can achieve both goals, but classical statistical methods are historically more focused on inference, and ML methods concentrate on prediction [37]. Statistics requires previous knowledge about the data-generating system, as different methods take different assumptions [38]. This is not the case of ML, which makes minimal assumptions about the systems. This proves particularly helpful when exploring unknown systems with interactions that can be complicated and nonlinear [39]. Another advantage with respect to classical statistics is the size of the data ML can deal with, allowing a larger number of input variables (features) than the number of samples.

ML can be broadly categorized into two types: **supervised learning** which uses prelabeled samples to allow the algorithm to see how accurate its performance is; and **unsupervised learning**, where the algorithm can identify patterns amongst unlabeled data.

2.5 Market analysis

HD is a disease present in all populations but occurs at much higher frequencies among individuals of European ancestry. Prevalence studies show that approximately 1 in 7,300 individuals are affected in Western populations [15]. The demand of HD testing and phenotype prediction is overall considered to be between 5 and 20% amongst at-risk patients. Data on subjects tested from different centres show a higher percentage of female subjects being tested than men [40]. Reasons to undergo genetic testing generally include future planning in career and family decisions. Because there does not exist an effective treatment of the disease, genetic testing is still low. The uncertainty of predictions for short CAG expansions also contributes in a low testing percentage. With a better model, genetic counselling would be more reliable.

A better AO regression model could also be a potential research tool in HD research, to understand the mechanisms of disease onset and reveal possible targets for treatment [2].

2.5.1 ML in disease phenotype prediction

The most common type of studies to reveal genotype-phenotype associations are the already introduced GWAS, centered in identifying statistically significant associations.

Using ML algorithms to build models that correlate genotype with phenotype shifts the goal from deeply understanding the phenotype (main GWAS goal) towards accurately predicting it [41]. Still, in a disease diagnostics setting, the explainability of a prediction is crucial, and thus algorithms that produce interpretable models are preferred.

There are two main challenges when applying ML onto the study of the effects of multiple genetic variables: the size of the datasets that are usually used (with more variables than samples due to the extension of the genome) and the effect of LD between SNPs, which is most likely to be present when a large number of SNPs are genotyped on a genome-wide scale, and which makes the dataset have correlated variables. This is why either penalized regression methods applied onto standard linear regressions or non-parametric approaches like the ensemble methods are needed to model the complex relationships contained in the data even though it probably contains correlated variables [42]. With this consideration, many researchers have shown the potential of SNP information for phenotype prediction [42], [43].

2.5.2 Enroll-HD studies

The study of rare diseases with ML has an extra challenge involved due to the ML requirement for large datasets. Enroll-HD is an outstanding observational study with almost 25,000 participants, being the world largest study for HD [44]. This dimensions make studying HD with ML possible. The dataset provides a very complete dataset with very diverse information about HD patients: general information such as sociodemographic data; medical information like medication and nutrition; motor, functional, behavioural and cognitive assessments, ratings and tests; and genotyping information from blood samples.

There are several published studies which use diverse ML techniques trained with Enroll-HD datasets. Ouwerkerk et al. (2023) [45] explores an Enroll-HD dataset of over 300 categorical and numerical features applied into improving AO prognosis, achieving an improvement of 9.2% with respect to the Langbehn formula estimation by using a light gradient boosting machine; and being able to predict future time points of disease progression. Ko et al. (2023) [46] recently applied ML into modeling disease progression with a similar heterogeneous dataset, using an unsupervised method to cluster samples according to progression velocity, followed by a supervised method (XGBoost) to identify features which could predict disease trajectory. The feature corresponding to the product of age and CAG repeat length was the top predicting feature, followed by years since symptom onset and medical history of apathy.

No published study has been found using ML with the GWAS dataset from Enroll-HD. The specific dataset used in this project was assembled by the GeM-HD Consortium [47] and later on expanded by Lee et al. [8], and used in GWAS analysis which revealed possible AO modifying SNPs.

3 Concept engineering

The main objective that summarizes the driving purpose of this thesis is to predict the AO in HD from SNP information. There are several approaches regarding different stages of the project, mainly: i) how to treat the available data (approximately 40GB of compressed raw data), and ii) what algorithms should be tested and how to compare them. The project's workflow (Figure 4) highlights the steps where we could uptake different methodologies, which will be discussed in this section.

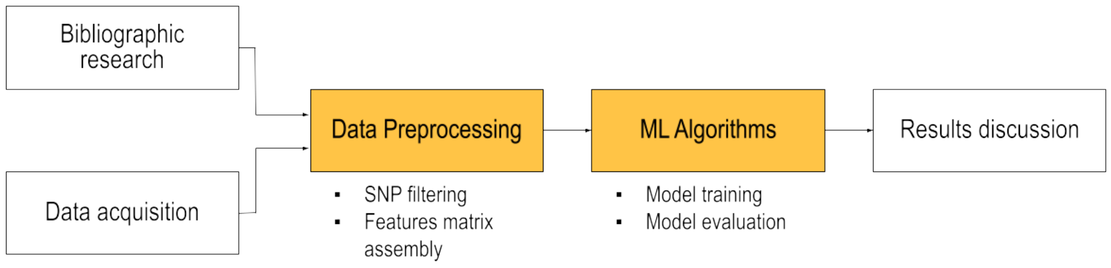


Figure 4: Project's general workflow.

3.1 Data Preprocessing

3.1.1 Feature engineering

The most usual approach to encode SNPs as numerical features consists in taking each SNP as an individual feature, with values 0, 1 or 2 depending on the allele of each sample (heterozygous or homozygous with the reference or alternative allele). This is a more suitable data format for further ML steps, as the raw data is encoded by *0/0*, *0/1* and *1/1*, which is not an appropriate input format.

We could also define as feature the sum of the SNP encodings in each gene, resulting in one feature per gene and, consequently, reducing drastically the dimensionality. We could even add only the encoded value of SNPs that have an effect on the gene's corresponding protein's structure, those found in coding regions. Considering that taking all SNPs in our data pertaining to a single gene we can find up to 2000 SNPs/gene and in average being 204 SNPs/gene, reducing these numbers to a single dimension would extremely reduce the number of features given to the models, making them easier to train. Nevertheless, this places the focus of the conclusions on a gene level rather than the SNP level, which would reduce the specificity of the results. Moreover, masking of important events in large genes could make important polymorphisms invisible. Still, the main interest of the thesis is to work at SNP level, which will allow us to identify relevant variants and model epistatic interactions (interactions between genes to express a certain phenotype).

Using all available SNPs directly would be a first exploratory analysis similar to a large Genome-Wide Association Study (GWAS), but it would require an unfeasible amount of computational power to train models with millions of features. Instead, we need to establish which is the best sub-setting criterion.

a. **Literature-Based SNP Selection.** One option could be taking the SNPs that have been previously described in literature to explain AO variance. This way we would be working with a low number of features, but the potential of revealing new AO modifiers would be zero. The next logical step would be to take the SNPs pertaining to genes described as either HD modifiers or related to AO explanation. Now we could find what SNPs inside each gene are the ones that could be responsible for the already observed gene effect. The discovery potential of this option is also limited.

b. **Ontology-Based Selection.** A broader option is to take the genes of ontology terms related to HD processes, after a thorough literature review. This would include genes which have been previously studied in this context and genes related to them, which have not been studied, but yet they are not randomly selected. The set of genes of each term can be extracted as files from the *AmiGO* web application [48].

c. **Protein Interaction Network-Based Selection.** Another parallel approach to broaden the subset of selected SNPs is to include the genes related to a core set of genes on the protein level, taken from the *STRING* database [49]. In this case another problem arises: how many levels of interaction, which we named *interaction order*, should we include? Should we just take those genes directly related to the core set of genes (first order interactions), or should we expand it further to those related to this first order interactions, related to the core genes through a two link path (second order interaction), and so on? We would have to test how such a network would grow including interactions of increasing order. This interactions are extracted from databases where a confidence interval for each interaction is given, representing how confident is the community in that interaction's existence. Different thresholds of this confidence should also be considered when exploring this option.

d. **Alternative Allele Prevalence Selection.** An additional option for dimensionality reduction whilst treating single SNPs as features is to filter them by prevalence of the alternative alleles, by removing those SNPs that do not reach a minimum threshold of the alternative allele in our sample set. This would follow the reasoning behind treating a single nucleotide variation as a SNP.

3.1.2 Feature scaling

Some ML models are more easily interpretable if the data given as input has a specific distribution, or they require a specific feature value range. As a general rule, models perform better if all the used features have values in comparable ranges. This is why feature scaling is an important step to consider.

In out data we include the sex and CAG repeat length of each sample as two additional features apart from the selected SNPs, as we have to control for the features that most highly explain AO variance and this is most probably the CAG repeat lenght. Sex is encoded with 1 (male) or 2 (female), which are values similar to SNP encoding (0, 1 or 2). CAG length, on the other hand, is much larger, containing values between 40 and

55. This feature must be scaled down.

The most common scaling methods for ML are:

a. **Min Max Scaler**: preserves the original distribution of data but scales it linearly to be in the range $[0, 1]$.

$$y' = \frac{y - y_{min}}{y_{max} - y_{min}} \tag{1}$$

b. **Standard Scaler**: transforms data to have mean (μ) equal to 0 and unit standard deviation (σ).

$$y' = \frac{y - \mu}{\sigma} \tag{2}$$

3.1.3 Outcome scaling

AO is also spanning between 5 and 83. Depending on what algorithm we will be using, we might want to scale it too in order to make the residuals centered at 0. The scaling that make this transforms data to be Gaussian, so we will use the standard scaler on the outcome vector.

3.2 ML models

Being a continuous random variable, AO prediction can be thought as a regression problem using a supervised learning process: we use the known outcome (AO) to build a regressor. As in any ML application, having more samples than features is an important limitation, as any model build with such high-dimensional data are prone to overfitting, giving a model that could not predict new unseen observations successfully. Penalized regression models are a recorded necessity in settings similar to ours [42], [50] to constrain the model to represent the overall relations in the dataset rather than each sample's particularities.

3.2.1 Regression models

Regression models that are used for similar applications to ours (regression over SNPs data for phenotype) comprise linear least squares methods and tree-based methods [51].

1. **Linear regression methods**

   These methods include all models that can ultimately be expressed as a fit of the response Y to the data X through a vector of coefficients β and a disturbance term ϵ:

$$Y \approx \beta X + \epsilon \tag{3}$$

A multiple linear regression model estimates as many coefficients as features used as predictors from the data. After optimizing the unknown parameters of the model (β vector), these can be used to predict the outcome of new observations, being the new observations x and their prediction $\hat{y}$. The most common way of estimating the coefficients is the **least squares method**: it looks for coefficient values that minimize the error between the true outcome and the predicted outcome, which is expressed by the residual sum of squares (RSS), being a residual the difference between the true outcome and the predicted outcome. In Eq. 4, n is the number of samples used for training the model, and p the number of predictors.

$$\text{RSS} = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{4}$$

This method is very straightforward, but fitting a linear regression onto any type of data is not guaranteed to yield good results. Problems as multicollinearity, overfitting and non-linearity of the response-predictor relationships pose the need for complementary methods to model a broader set of cases.

Non-linearity can be addressed by using polynomial regression, interaction terms and transformations, amongst other options. They all involve fitting the data to functions which are no longer linear. Generalized Linear Models (GLMs) extend linear regression to handle non-normal response variables like counts of discrete while allowing for different types of response distributions and link functions. This would be an interesting approach to apply in our case, as AO can be thought as counts coming from a Poisson distribution, and we could use a **Poisson regression**, a specific type of GLM [52].

But our AO vector contains decimal onset ages, and therefore we should round its values to integers, which would involve loosing information. The possible non-normality coming from the theoretic distribution of our response variable can be ignored considering the approximation given by the Central Limit Theorem (CLT) which implies that for a sufficiently large sample size, the distribution of the sample mean of random variables will be approximately normal [53]. This normality approximation should be tested empirically (with statistical tests such as Shapiro-Wilks or Kolmogorov–Smirnov) as part of the preprocessing steps, as SNP counts (considering the counts as the genotype) are not entirely independent in the case of LD blocks. Such tests have to be computed over subsets of the outcome vector, as any test will find enough evidence of a non-normal distribution with such large statistical power provided with our data (over 9k samples). We can also test for normality graphically, by fitting the histogram of AO to a normal distribution and assessing the fit, or through a Q-Q plot. If we can prove that the outcome variable can be approximated to a Gaussian distribution, we can use the least squares approach.

We can build a simple model using Ordinary Least Squares (OLS) to work as the baseline, using as predictors only CAG repeat length and sex. Then we can build more complex models including the SNP genotypes. Our main interest resides in shrinking methods which focus on regularizing the coefficient estimates, making them close or equal to 0. Such methods

- Act as **feature selectors**, which is important to highlight feature importance in a model towards finding AO modifiers.
- Control **overfitting** by introducing a controlled amount of bias which reduces the variance of the model, thereby encouraging the model to generalize well to unseen data.
- Deal with **multicollinearity**, the situation where features are correlated with each other and thus either provide the same information or their effect can be canceled if the correlated features have opposite effects (a large positive coefficient can be canceled by its related large negative coefficient) [54]. This could happen in our context when having SNPs of the same LD block.

The three main shrinkage methods are the following:

a. **Ridge**

This technique uses the l_2 regularization: it introduces a penalty term in the function to minimize (Eq. 5), which is the product of the tuning parameter l_2 to the norm two of the coefficients' vector.

$$F(\beta) = \text{RSS} + \alpha \sum_{j=1}^{p} \beta_j^2 \tag{5}$$

The effect of this added penalty is that it shrinks closer to 0 the largest coefficients, and acts less in those which are already close to 0.

A higher α value decreases variance but increases bias. An optimal parameter value has to be found to reach a nice equilibrium between bias and variance and have a good associated model performance.

b. **Lasso**

This approach also includes a penalty term to the minimized function, but in this case it uses the norm 1 of the coefficients' vector,

$$F(\beta) = \text{RSS} + \alpha \sum_{j=1}^{p} |\beta_j| \tag{6}$$

Such penalty acts on the model by performing variable selection when α is sufficiently large, shrinking to exactly 0 those features which contribute less in the model. It is indifferent to very correlated predictors, it will tend to pick one and ignore the rest.

c. **Elastic Net**

Elastic Net can be considered a fusion of both Ridge and Lasso, as it uses l_2 and l_1 regularization at the same time. The minimized function (Eq. 7), when implemented, considers $\alpha_2 = 1 - \alpha_1$.

$$F(\beta) = \text{RSS} + \alpha_1 \sum_{j=1}^{p} |\beta_j| + \alpha_2 \frac{1}{2} \sum_{j=1}^{p} \beta_j^2 \tag{7}$$

This penalty is particularly useful in situations where $p >> N$ or where there are many correlated predictor variables. For a penalty close to the Lasso, it can perform just like a Lasso while removing strange behaviours caused by extreme correlations [55].

As we will most probably be working with a very big dataset where the number of features is going to be very large, the most useful regularization to implement is an l_1 regularization for feature selection, and thus Lasso is the principal model to try out. Elastic Net is also a good option.

2. **Tree-based methods**

These methods involve segmenting the predictor space into a number of simple regions: the prediction of a new observation is set to the mean or mode response value for the training observations similar to the new input. The segmentation of the predictor space follows a set of splitting rules, which can be thought of as a tree, hence the method's name. Each split, called node, generates two leafs, creating its own unique simple regression model [56]. The used approach to segment the predictor space (the space made up of all features used in the model) is known as *recursive binary splitting*, which is: *top-down* in the sense that it begins splitting by just one feature, at the top of the tree where all observations are still in the same region; and *greedy* because each split step is chosen considering the immediate goodness of the split, without thinking of possible better combinations with other splits further down. The goodness of a split is assessed by minimizing the RSS over all regions (Eq. 8) where R_j are the predictor regions (having a total of J regions), and $\hat{y}_{R_j}$ the mean response for the training observations within the jth region.

$$RSS = \sum_{j=1}^{J} \sum_{i \in R_j} \left(y_i - \hat{y}_{R_j}\right)^2 \tag{8}$$

The main benefit of tree-based methods is their interpretability. But a single decision tree may not always have a good prediction accuracy due to its simplicity. It has a very high variance, and it tends to overfit, as feature space splitting usually ends when the split space contains a small number of train observations, which can make the tree too case specific. To solve this, a threshold on RSS could be applied, but this could leave out low minimizations of RSS that later on would reveal important splits. This overfit is solved with *cost complexity pruning* or *weakest link pruning*: a tuning parameter α controls a trade-off between the subtree's complexity (amount of nodes) and the fit to the training data. In the minimized equation optimized in this approach (Eq. 9), $|T|$ is the number of end nodes of the tree T, R_m is the predictor subset of each terminal or end node, and $\hat{y}_{R_m}$ is the predicted response associated with R_m. A more complex tree will have a higher number of terminal nodes, which will make the function's independent term value, the penalization, larger.

$$F(|T|) = \sum_{m=1}^{|T|} \sum_{i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T| \tag{9}$$

Tree ensemble methods are more complex approaches which produce multiple weak trees or learners and later combine them to produce a single output prediction. These are:

a. **Bagging**: trees are grown independently using a random set of samples and all features being available for every tree growth.

b. **Random Forests**: trees are also grown using random sets of samples, but for each split on each tree only a random subset of features is available. A similar approach is followed by **Extra Trees**, with the difference that it chooses the split points at each node randomly, whereas Random Forests searches for the best splitting points for each random subset of features at each node. This makes Extra Trees faster to train, with the price of having higher bias.

c. **Boosting**: trees are grown successively, starting from a single tree which uses all samples and all features, and continuing on fitting the following tree to the residuals of the previous one. It is a numerical optimization technique for minimizing the loss function by adding a new tree at each step that best reduces the loss function (stepping down its gradient) [57].

Random forests are preferable over bagging in our case as we restrain the used features as predictors to a random subset of features. This way if there is a very strong predictor as CAG is, we can create trees that will explore relations between other features apart from CAG. We can try Extra Trees if the implementation of Random Forest is computationally unfeasible.

Both bagging and random forests make predictions taking the average of all their regression trees which are independent. This is why increasing the number of trees can prevent overfitting. Boosting grows trees by capturing signals which have not been accounted for by the previous set of trees, so a large number of trees can overfit data. The size of trees in boosting methods is also reduced compared to bagging and random forests [58].

Inside boosting methods, there are different tree-methods: the exact solution and the approximated solutions. The exact solution involves the booster considering all candidates from data for tree splitting, which makes this method very slow computationally [59]. Approximated solutions build a gradient histogram for each node and iterate through the histogram instead of seeing all observations of the real dataset [60].

3.2.2 ML packages

The most commonly used resource for easy ML algorithm training is *Scikit-learn* for Python. Apart from containing functions for several algorithms, it also contains modules on metrics to evaluate the fit of a model, and model selection to perform cross validation and grid search to find the best parameters. Similar packages are *statsmodels* and *skglm*. For boosting algorithms, the most popular package is *XGBoost*.

As a starting point, we will implement the chosen models with the two most simple sources: *Scikit-learn* [61] and *XGBoost* [59].

3.2.3 Data sparsity

Another important aspect to keep in mind when choosing algorithms is the large dimension of our data. Training any model with a feature matrix taking several gigabytes of memory can turn out to be impossible with the available resources. Considering the matrix is mostly formed by zeros, as the reference homozygous is the average genotype we will find in most cases, we can use **sparse matrices**. This is a way of representing a matrix by listing the non-zero elements of the matrix and their position. This way we store less values than by recording all the positions that simply contain zeros.

Most ML training packages in python, including Scikit-learn and XGBoost allow giving as input a sparse matrix.

3.2.4 Model metrics

Because we are training models with high-dimensional data where the number of features is larger than the number of samples, reporting measures of model fit using statistical metrics on the training data will not be very informative, as the models will most probably be overfitted. Instead, these metrics should be computed over a test set to compare the true predictive ability of the models. Calculating these metrics over the train set is also done, but to check for overfitting rather than for model comparison.

There are different statistics to evaluate the goodness of fit of a model [62]. In the following expressions, y represents the outcome vector true values, and $\hat{y}$ their corresponding predictions.

a. **Max error**: single worst error amongst all predictions.

b. **Mean absolute error (MAE)**: average of the absolute difference between the true and predicted values of y.

$$\text{MAE}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} |y_i - \hat{y}_i| \tag{10}$$

c. **Mean squared error (MSE)**: expected value of the squared error.

$$\text{MSE}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} (y_i - \hat{y}_i)^2 \tag{11}$$

d. **Coefficient of determination** R^2: proportion of variance of y explained by the independent variables in the model.

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2} \tag{12}$$

$\bar{y}$ is the true average $\frac{1}{n}\sum_{i=1}^{n} y_i$ and $\sum_{i=1}^{n} (y_i - \hat{y}_i)^2$, which is the RSS.

The closer the score is to 1, the better the fit to the data. A 0 value of R^2 means the model is constant, always predicting the expected value of y disregarding the input features ($\hat{y}_i = \bar{y}$ for $i = 1, ..., n$). It can also be negative, if the model is arbitrarily worse than the constant model.

e. **Explained variance score**: fraction of deviance explained. A perfect fit explains the totality of the data variance, hence the score would be 1. Lower values correspond to worse fits.

$$\text{explained variance}(y, \hat{y}) = 1 - \frac{\text{Var}\{y - \hat{y}\}}{\text{Var}\{y\}} \tag{13}$$

It is similar to R^2, but the explained variance score does not account for systematic offset in the prediction.

There are many other metrics we could use, but these are the simpler ones which should be enough to compare between models. These methods tend to be very dataset dependent, thus the test set to evaluate all models will be the same. The preferred metric in many different settings is the coefficient of determination.

The model's performance can also be assessed visually by plotting the predicted values over the actual values of all samples in the data set. Another useful plot is the representation of the residuals (predictions subtracted from the true values) over the actual values, to see whether a range of outcomes is better or worse predicted than the rest.

4 Detailed engineering

All generated code can be consulted in the ML-HD GitHub repository (URL: `https://github.com/cfuses/ML-HD`). The repository organization follows the same directory structure as the code directory of the computer where the scripts were executed. Throughout this section some code snippets are included between the text to show the parameters used in relevant functions or to see in a more direct way how certain non-standard processes were executed.

Scripts were build and executed in a remote machine provided by Creatio. To manage the required packages, a Conda environment was created with the necessary packages: *Plink2*, *Numpy*, *Pandas*, *Scipy*, *Sklearn*, *Matplotlib*, *os*, *XGBoost*, and *StatsModels*.

The "GWAS12345" Enroll-HD data was acquired from CHDI in *.tar* format. The uncompressed folder contained *.bed*, *.bim*, and *.fam* files for each autosomal chromosome plus a metadata file containing sex, CAG repeat length and different residual calculations of each subject, as well as a readme file with information about the compression format. The total number of subjects is 9064, with 4,647 females and 4,417 males. CAG values range from 44 to 55, with a mean of 44 and a standard deviation of 3.06, which means we will only be working with full penetrance HD alleles. The mean AO is 45.54 years, with a standard deviation of 11.58 years.

The outline of the preprocessing steps taken is represented in Figure 5. Each process is explained in detail in the following subsections.

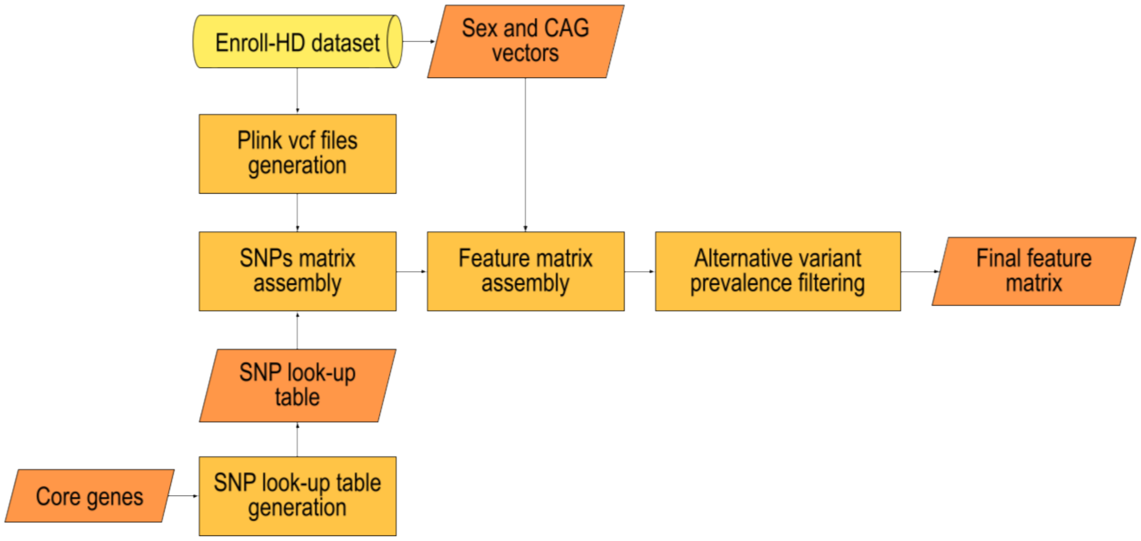


Figure 5: Data preprocessing flow chart.

4.1 Core gene selection

We considered that for an application with simple ML models, the ontology-based selection of genes would be enough. After a thorough literature review on genes and processes related to HD mechanisms [2], [5], [7], [8], [15], [19], [63]–[66], a list of GO terms to be included in the analysis was assembled, presented in Table 2.

Process	**GO term**	**Related Genes**
Mismatch repair	GO:0006298	*FAN1, MLH1, MSH3, MLH3*
Synaptic transmission, Glutamatergic	GO:0035249	*GRIK2, GRIN2A, GRIN2B*
Omega peptidase activity	GO:0008242	
Cysteine-type endopeptidase activity	GO:0004197	
Proteasome-mediated ubiquitin-dependent protein catabolic process	GO:0043161	
Ubiquitin binding	GO:0043130	
Ubiquitin protein ligase binding	GO:0031625	
Protein deubiquitination	GO:0016579	*UCHL1*
Transcription regulator activity	GO:0140110	*TCERG1, TP53*
Neuron apoptotic process	GO:0051402	*DFFB, MAP3K5, MAP2K6*
Lipoprotein metabolic process	GO:0042157	*APOE*
Axonal transport	GO:0098930	*HAP1*
Folic acid metabolic process	GO:0046655	*MTHFR*
Energy reserve metabolic process	GO:0006112	*PPARGC1A*

Table 2: Included GO terms as core genes.

These set of genes associated to each GO term was downloaded from the *AmiGO* web application [48], selecting three fields to download: the *Gene/product*, which contains the UniProt identifier; the *Gene/product name* which is the name of the gene (a string with words separated by spaces); and the *direct GO class*, as GO terms are hierarchical and a single gene can be found under different terms (which makes a single gene appear in several rows if it has more than one direct GO class). The downloaded files are tab-separated text files. In a local Jupyter Notebook, all these files were concatenated, and an additional column containing the GO class from which each set was taken (*broad_GOclass* was included. Using *ExPASy* and *SwissProt* from *Biomanager*'s Python package the UniProt ID of each gene was translated into the gene symbol, saving it into a new column called *Gene*. Later on the process of this thesis, further literature review revealed potentially important genes in HD processes which were not included in the core genes set. These are *HTT*, *MAP2K6*, *PRMT6*, *CCDC82*, *SOSTDC1*, and *FAN1*. They were added manually to the list. We could have expanded the list further on by including the entire set of genes of the GO terms of these new genes, but at that moment we were facing important memory availability issues and decided not to.

Once this table was assembled, we uploaded it onto the remote machine used for the rest of the analysis. The list of unique *Genes* are from now on referred to as the core gene set of our study.

4.1.1 Core genes SNPs

The Enroll dataset identifies SNPs through their chromosome position, but we need to relate them to their corresponding gene, which is easier when they are identified with their Reference SNP cluster ID (Rsid). This was achieved by creating a look-up table containing all SNPs from each core gene. The script performing this task is `biomart_retrieval/biomart_snpid_gene_retrieval_v2.R`.

For such task, the *Ensembl Biomart* database was used, and to easily access it we used its *R* API. We first extract the starting and ending positions of each gene, giving as input a vector containing the list of core genes (`gene.list`), and then we save the output in a data frame `gene.symb`.

```r
# Connect to the Ensembl BioMart database
ensembl <- useMart("ensembl", dataset = "hsapiens_gene_ensembl",
                   "https://feb2014.archive.ensembl.org")

# Retrieve gene coordinates
gene.symb <- getBM(
      attributes = c("ensembl_gene_id", "external_gene_id",
      "chromosome_name", "start_position", "end_position", "strand"),
      filters = "hgnc_symbol", values = gene.list,
      mart = ensembl)
```

It is important to access a dataset using the same genome assembly reference as our data, which is GRCh37/hg19, so the positions of our SNPs are comparable to the chromosome coordinates obtained with this script.

Next, we access another Ensembl dataset, now containing information on SNPs. We read the `gene.symb` row by row, which makes the iteration consider one chromosome at a time, and ask for the position, reference identification and reference variant of all SNPs inside each chromosome by passing a query with the format `<chromosome number>:<start position>:<end position>`. This petition is tried at least 5 times per chromosome, as the connection with the dataset is often unstable. When the output data frame is obtained, we bind it to a general data frame where we concatenate the outputs of all chromosomes, finally obtaining the look-up table. The following code snip is a reduced version of the used code, leaving out the error handling:

```r
# Empty data frame to store ensembl outputs
snp.tab <- data.frame()

# Iterate over chromosomes
for (i in 1:nrow(gene.symb)) {

      # Create coords vector as required for getBM
      query <- paste(gene.symb$chr[i], gene.symb$st[i],
                     gene.symb$end[i], sep = ":")

      ... (while loop to handle connection errors)
        # Get table from ensembl
        sub.snp.tab <- getBM(
              attributes = c("chr_name", "chrom_start",
              "refsnp_id", "allele"),
              filters = c("chromosomal_region"),
              values = query, mart = mart
        )
      }

      # assign gene name
      sub.snp.tab$gene <- gene.symb$gene_name[i]

      # Append to general snp tab
      snp.tab <- rbind(snp.tab, sub.snp.tab)
}
```

The table generated through this script was then revised manually using bash commands. Initially we had a table with 4,153,337 rows. We then applied two filters:

- Only those SNPs which had Rsid (reference identifiers starting with *rs*) were kept, reducing the rows to 4,106,699.
- We just want to include SNPs, which are single nucleotide variations. We leave out all those variations that are insertions, deletions, and other types of mutations which do not follow the format N_r/N_a (reference and alternative bases). The final revised and filtered look-up SNP table has 3,590,278 SNPs.

For easier result visualization, a final table relating SNPs to the GO terms from which we extracted their corresponding genes was generated with bash in the script `data_preprocessing/mapping_snps_to_GO_v2.sh`. It iterates over each SNP in the look-up table and searches their gene in the table of core genes which relates genes to their GO terms.

4.2 Data preprocessing

4.2.1 Feature matrix assembly

The Enroll-HD dataset contains three files per chromosome which encode each individual's alleles for all the sequenced SNPs: *.bed*, *.bim*, *.fam*. To work with a simpler format, *Plink2* software combines this kind of data into a single *vcf* file (Variant Coding File). Having all three files for each chromosome in the same directory (each group of three is named equally, only changing their suffix so we can point to the files through a single file name), we generate the *vcf* files and build the final SNP matrix with the bash script `vcf_generation/run_plink.sh`. The first step is to iterate over each chromosome with a for loop and generate the vcf files with the following command:

```
    plink2 --bfile "$filename" --recode vcf --out "$out"
```

This command generates two files: a *.log* file which will not be used and hence it is deleted, and the *.vcf*, which is a text file where each row is a SNP position, the first 10 columns describe the SNP and the following ones are the samples' genotype in that chromosome position. The genotype is encoded with 0/0 (homozygous for the reference variant), 0/1 (heterozygous) and 1/1 (homozygous for the alternative variant).

To avoid saving the totality of the encoded SNPs contained in the original dataset, just after creating each vcf and still inside the for loop we filter the recently generated *vcf* file using the SNP look-up table: we take the positions of all those SNPs in the table whose chromosome matches with the for loop iteration, and we keep the rows in the *vcf* which match these positions using the command `grep`. With this filtering we are not including the file's headers. The row containing the samples' ID is saved in a separate file before filtering, and we externally checked from the command line that the column order is the same in all generated files for each chromosome.

Right after this filtering, we also discard the columns that give additional information about the SNP, such as what are the reference and alternative bases on that position, the quality score that indicates the confidence that the variant is correctly called, filter status, the format of the genotype fields. By doing so, we delete the information that could differentiate between multiallelic SNPs, but they are still being included in the model. This is a common simplification done in the genomics field, as we are mainly interested in representing if a sample has an alternative variant or not, rather than what precise nucleotide is in that alternative variant, disregarding the possible different effects these variants could have down the transcription processes. This is why Rsid nomenclature is unique for the SNP position, and not for the alternative alleles.

Once the *vcf* is filtered, the genotype is re-encoded with bash command `sed` to be easier to work as ML features, changing 0/0, 0/1 and 1/1 for 0, 1 and 2, respectively.

Once this has been done for all 22 chromosomes, all filtered files are concatenated, generating a single table with all the SNPs of the core genes and their genotype in each sample. Since we are treating SNPs as features, we need them as columns, not rows. Due to the large size of the table, we transpose the numeric part of the table (transposing the chromosome and position columns separately afterwards) with the C++ script `vcf_generation_/transpose_matrix.cpp`. Regarding the chromosome and position columns, we need to merge them into a single identifier. We do so by using their Rsid, once again using the look-up SNP table extracted with *BioMart*. Now we can concatenate back together SNP identification and their genotypes, now having each SNP as a column.

Finally, we add three columns containing the sample identification, their sex and the number of CAG repetitions (extracted from the sample information file) as the first three columns, and we have a first version of the feature matrix, occupying approximately 11GB.

4.2.2 Alternative variant prevalence filtering

To reduce the dimensionality of our data in the most unbiased fashion possible, in the concept engineering we presented the option at hand: filtering out those SNPs which are homozygous for the reference variant in more than 99% of the subjects, or equivalently only taking the SNPs for which at least 1% of the samples have an alternative variant. This is done with Python in the script `data_preprocessing/alternative_prevalence_colsuming.py`.

The filtering is done by applying a boolean mask to the feature matrix. This mask is created from an initialized vector containing as many indices as SNPs, and each value, which starts as zero, will ultimately be the number of subjects which have at least one copy of the alternative variant. This way if all subjects had at least one copy of the alternative variant in a certain SNP, the corresponding index of that SNP in the vector would contain the total number of samples of the dataset.

As the table to filter is so large and loading it entirely onto processing memory can make everything slower and more expensive in terms of computation resources, the script reads it line by line (one subject at a time). It then looks at all SNPs genotypes one by one, adding 1 to the SNP index of the already mentioned vector if the genotype is different from 0 (either 1 or 2). Once the iterator has gone through the entire matrix, the boolean mask is created by comparing each vector value with the minimum value chosen as threshold, which is the 1% of the total number of samples.

This mask is then applied to the feature matrix by opening an output file in write mode and opening again our feature matrix in read mode, and for each line read, it applies the mask and writes it onto the new output matrix.

The final matrix contains 339886 features, which means we are including **339884 SNPs** in the models.

4.2.3 Toy example matrix

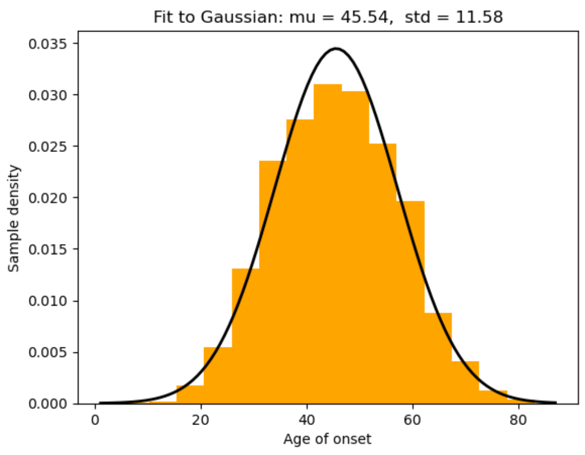
The filtered version of the table achieved with the previous step takes 5GB of storage, having reduced 6GB of memory. Still, to make further developments faster to build and test, a toy example of 50MB is generated with 900 samples and 70k SNPs, forcing to include the SNPs of the genes which are some of the known HD modifiers candidates (*MLH1*, *MLH3*, *GRIK2*, *GRIN2A*, *GRIN2B*, *UCHL1*, *APOE*, *ASK1*, *MAP3K5*, *PPARGC1A*), and the SNPs of *HTT*, and picking randomly the rest to sum up to 70k. This is done in `subsampling/random_sample_snps_pick.ipynb`.

4.2.4 AO normality check

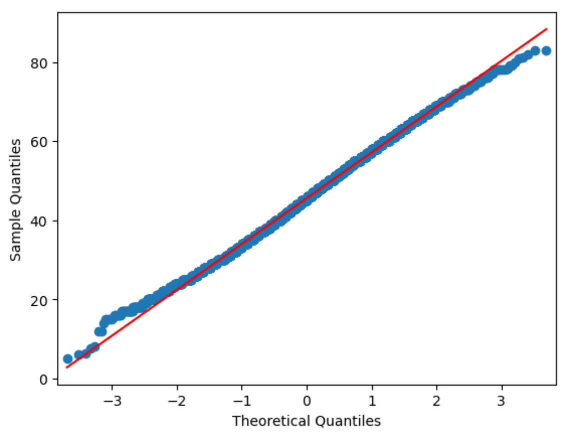
In the concept engineering section on ML models we have stated that AO can be considered a normally distributed variable through the approximation of the Central Limit Theorem. This approximation can be tested both graphically and statistically.

We can do two different graphical tests: fitting the AO vector to a normal distribution and plotting the fitted curve on the histogram of the experimental data to see how accurate is the fit (Figure 6a); and a Q-Q plot (Figure 6b). Both tests show a normal distribution.

Statistically, several tests were run: *Shapiro-Wilk*, *Kolmogorov-Smirnov* and *normaltest*, all from *Scipy*'s module *stats*. These tests take as the null hypothesis that the data comes from a normal distribution, so a p-value lower than 0.05 would indicate that there is enough evidence to reject the null hypothesis, and thus would be contrary to the normal approximation. When run over the raw AO all three tests reject the null hypothesis, and again when transforming the entire raw AO vector to try to make them more normal (logarithmic, reciprocal, exponential, Box-Cox, Yeo-Johnson, standardization).



(a) Normal fit over actual AO histogram.



(b) Q-Q plot of AO.

Figure 6: Graphical normality tests.

The problem lies in overpowering the tests with a too large number of samples. When the vector was split into 30 subsets and each subset was tested individually, more than 80% of the subsets did not reject the null hypothesis. We can consequently treat AO as a normally distributed variable and continue implementing a wider set of model options.

4.3 Regression models

All Python code to train and test the ML models was first tested using the toy example in the same Jupyter Notebook, and once the code was optimized, a separate Python script for each model was created to execute them in parallel. All these scripts can be found in the directory `code/regression_models/individual_models`.

Each individual script follows the same structure:

1. Import dependencies, including the custom functions to load data and evaluate the model performance contained in separate scripts inside the same directory.
2. Declare the working directory and the different directories which will be accessed throughout the script, as well as the paths to the data files.
3. Load the feature matrix as a sparse array with the custom function `read_sparse` and the outcome vector as a Numpy array.
4. Scale the CAG column of the feature matrix with the custom function `scale_CAG`, and the outcome vector with the pickled scaler `aooStandardScaler` generated in the script `regression_models/aoo_scaler.py`.
5. Split the data in training and testing sets with the function `train_test_split` of the *Sklearn*'s module *model_selection*. We ensure that the samples set aside for testing across different scripts are the same samples by setting the same *random_state* number. The testing set size is set to the 30% of the total number of samples.

Model	**Parameter**	**Parameter ranges**
Lasso	alpha	[0.005, 0.01, 0.05]
	max_iter	[1000, 2000]
Elastic Net	alpha	[0.005, 0.01, 0.05]
	l1_ratio	[0.2, 0.5, 0.9]
	max_iter	[5000, 10000]
Random Forest	ccp_alpha	[0.001, 0.01, 0.04, 0.1]
	max_depth	[2, 4, 6, 7]
	n_estimators	[10,20,30,40]
XGBoost (approx)	reg_alpha	[0, 0.1, 0.5]
	max_depth	[3, 5]
	n_estimators	[10, 20, 40, 80]
XGBoost (hist)	reg_alpha	[0, 0.5, 0.8]
	max_depth	[2, 3, 5]
	n_estimators	[5, 10, 15, 20]

Table 3: Grid search parameter values for trained models. All parameters not stated here were left with the default value.

6. Model definition, different in each script, followed by a dictionary of hyperparameters to test in a grid search.

7. Grid search with 5 fold cross validation using the *Sklearn*'s function `GridSearchCV`, from the *model_selection* module, over the training sets, using the default R^2 scoring as the metric to evaluate the trained model on the validating set in each fold.

8. Extract the best estimator fitted with the previous grid search, and pickle it in a sub-directory inside the results folder to examine it further on a separate script.

9. Evaluate the training of the model using the custom function `model_description`, saving the generated plot as a *png* image in the results directory.

10. Evaluate the model performance with the custom function `model_metrics`, saving the generated outputs in the results directory.

The custom functions are explained in greater detail in the following subsections. The parameter values included in the grid search were chosen on the preliminary testing done with the toy example in the notebook, starting with standard values and tuning them to ranges closer to the best parameters chosen in each execution. When running the scripts which use the entire dataset, we also executed the scripts again (when possible) changing the parameter ranges if the chosen hyper parameters in the first execution were one of the limit values to ensure we were not limiting the best possible model performance in this aspect.

The models that were chosen as most suitable for our objective and the parameters tested in their grid search are presented in Table 3.

The OLS script differs slightly from the models above, as it is a simple linear regression with no hyperparameters to tune, and the feature matrix given to the model only contains the variables sex and CAG repeat length. The cross validation is performed with the function `cross_validate` of the same *model_selection* module as the grid search function, with the most similar configuration as the rest of cross validations performed (returning the best trained estimator, doing 5 folds and scoring with R^2).

With such a large dataset it is very difficult to obtain an unbalanced split if done randomly. Nevertheless, we also checked in the testing notebook that the split does not affect the balance of the dataset with regards to sex, CAG and AO distributions. This was done by plotting the amount of samples from each sex at each set in a bar plot, and the distribution of CAG lengths in each set with boxplots, separating also by sex (see the results in appendix A).

Each Python script was executed through a Slurm job, creating batch scripts for each model. The maximum resources given are 200GB for execution, with a time limit of 24 hours. Any model configuration which took more than a day to execute was discarded.

4.3.1 Feature matrix loading

The feature matrix is loaded in the model training scripts using `read_sparse_X`, a function created in the script `regression_models/individual_models/data_loading.py`. Trying to load the matrix directly using *Pandas* or *Numpy* was too demanding memory-wise. Our dataset is made of mostly zeros, as the reference genotype is the predominant one. Hence the usage of a sparse matrix, which only saves the non-zero elements and its indices, reducing drastically the memory needed for script execution.

As the matrix is not saved as a sparse matrix in its directory, we need to convert it. We could load once the matrix with a computationally demanding tool such as the ones provided by *Pandas* or *Numpy*, convert it to a sparse matrix and save it as such to directly access the sparse version from the model scripts, or create a function which allows reading the original matrix directly. Using this last option we avoid having multiple copies of the same information in our memory. This is done by reading the dense matrix by chunks, converting these chunks into *csr* (Compressed Sparse Row) matrices, and concatenating the sparse chunks:

```
# Initialize an empty list to store the chunks
chunks = []

# Open the file
with open(X_path, 'r') as file:
    # Read header line
    header = file.readline().strip().split("\t")

    # Initialize a list to store chunk data
    chunk_data = []

    # Read the file in chunks
    while True:
        # Read chunk_size lines
        for _ in range(chunk_size):

            line = file.readline()

            if not line:
                break  # Reached end of line

            data = line.strip().split("\t")

            # Add feature data to row vector as float32
            rowdata = [np.float32(val) for val in data[1:]]

            # Add row vector to chunk_data list
            chunk_data.append(rowdata)

        if not chunk_data:
            break  # No more data to read

        # Convert the chunk data to a CSR matrix
        chunk_sparse = csr_matrix(chunk_data)

        # Append the chunk to the list
        chunks.append(chunk_sparse)

        # Clear chunk data for the next iteration
        chunk_data = []

# Concatenate the list of CSR matrices into a single CSR matrix
X = vstack(chunks)
```

It is important to note that the sparse matrix does not include the sample names to be able to fix the data type of the matrix to 32 bit floats, and thus reduce even further the memory usage. Using 16 bit integers would be even better, but as we scale CAG between 0 and 1, we need a float data type.

The final `X` is the sparse matrix which is returned when calling this function. The function itself takes two parameters: `X_path` where the matrix to read is saved, and `chunk_size`, with default 100, which is the amount of rows read once at a time from the dense matrix. The bigger the chunk size, the larger the working memory requirement.

4.3.2 Scaling functions

After loading the data we scale the CAG feature and the outcome vector AO. CAG scaling is performed by giving the sparse feature matrix to the function `scale_CAG`, created in the script `regression_models/individual_models/data_loading.py`. This function takes the second column of the sparse matrix, corresponding to CAG values, and performs a Min Max scaling over it. To ease computations, we extract the column as a *Numpy* array, scale it, and reallocate the array as a sparse matrix (using the function `csr_matrix`) from the *sparse* module of the *Scipy* package. The function returns the entire sparse matrix, now with CAG scaled between 0 and 1.

The AO scaling is done through a standard scaler, generated once executing the script `regression_models/aoo_scaler.py`. The script takes the file which contains the AO vector, loads it as a *Numpy* array, converts it to a 1 dimension array and uses it to fit the `StandardScaler` from the *preprocessing* module of *Sklearn* with the method `fit`.

The generated scaler is then saved with a pickle format, as later on we need to convert the scaled values of AO back to their original range in a controlled manner. When the saved scaler object is loaded in the model training scripts, it transforms AO values with the `StandardScaler` method `transform`, without needing to fit it again.

4.3.3 Evaluating functions

Two custom functions were created to evaluate the model after training:

a. `model_description`, intended for seeing whether the model is overfitted by plotting the predictions of the training samples versus their actual values and by computing the **percentage of deviance explained by the model**. The input parameters are the regressor object, the feature matrix, the outcome vector, and optionally the outcome scaler if we want the plot to represent the values scaled back to the original AO values. The predictions are computed inside the function with the regressor object's method `predict`. Then the percentage of deviance explained by the model is calculated with *Sklearn metrics*' function `explained_variance_score`. If the scaler is passed as a parameter, both predictions and actual values are transformed to the original range with the scaler method `inverse_transform`. Finally the plot is done with *matplotlib*, using the function `hexbin` to plot a heatmap containing the density of points inside each hexagon that splits the plotting area. We use a heatmap approach rather than plotting each sample as a point in a scatter plot because a scatter is not as informative when we have thousands of samples overlapping. The subtitle of the plot contains the value of the explained variance.

b. `model_metrics`, which returns the metrics of the model (calculated using the predictions of the test set), the model parameters and a plot to visually see how close are the predictions to their real values. The main structure is similar to the previous function described, but this time around the input data must be the feature matrix and outcome vector of the testing samples. The metrics computed here are the **coefficient of determination $\mathbf{R^2}$**, **mean squared error MSE** and **mean absolute error MAE**. Their values are written in a text file, alongside the parameters of the model, obtained with the regressor object method `get_params`. The plot generated with this function is made up of two subplots, one plotting the predictions versus their actual values as in the `model_description` plot, and another representing the residuals of the predictions versus the predictions, to see whether the error in predictions is bigger along a specific range or it is homogeneous over all predictions. Both plots have a dashed line which shows where should all points fall in a perfectly fitted model (the function $x = y$ in the case of predictions over actual values, and the line at $y = 0$ in the residuals plot). In this case we also use `hexbin` plots.

4.4 Results presentation

Apart from seeing how well each model explains the AO effect with the given feature matrix through the outputs of the evaluating functions, we also want to see which are the features that contribute the most to each trained model. In the least squares methods, this can be done by looking at the values of the optimized coefficients. In the tree based methods, feature importance changes across packages. In XGBoost, we can see feature importance through two amounts: weight and gain. Weight is the number of times a feature appears in a tree across all trees in the model, so a higher weight means that many trees have considered that feature important in their tree building process. Gain represents the average improvement in accuracy of the decision tree when used across all trees in the model. This is a more direct way of calculating the contribution of a feature to the model's performance. *Sklearn*'s Random Forest already has a feature importance property, which is the Gini importance.

These feature importance amounts are extracted from the regressor objects pickled just after training them, and loaded into `regression_models/model_results.ipynb`. Then the most important features, selected using the feature importance measurement suited for each model type, are related to the SNP they represent, and a sorted table presenting the results is assembled, containing the columns SNP, importance amount (coefficient, gain or Gini importance), gene and GO term. These tables are saved as text files.

To further present our findings in a simple way, we can do plots following the Manhattan plot format used in GWAS: the horizontal axis represents the genome positions, starting from chromosome 1 up, and on their corresponding relative positions over a chromosome, each SNP is represented as a bar, with its height being the feature importance measurement of the represented model. In least square methods, the vertical axis will be the coefficient value of a SNP, which means that it can take both positive and negative values (whether a SNP coefficient is positive or negative is valuable information and we must make it visible). With tree-based methods, the vertical axis will be the gain or Gini importance of each SNP, and thus will only have positive values. These plots are done in `regression_models/results_visualization/model_results_plots.ipynb`.

From the results tables we can also see from which GO term are the genes of the SNPs contributing to each model. This information is best seen using a segmented bar chart that visualizes the distribution of GO categories across the different models. Each bar represents a different model, and the segments within each bar represent the proportion of various GO categories present in that model. The height of each segment corresponds to the percentage of that particular GO category out of the total GO categories for that model. We also include the background proportions, the GO proportions in the feature matrix. This plot is generated in the notebook `regression_models/results_visualization/go_contributions.ipynb`. In the same notebook we also compute a two-sided Fisher test to see which GO terms are significantly enriched in the model's representation compared to the background.

5 Results

5.1 Baseline model

The current models of HD AO estimation predominantly rely on CAG repeat length, with the explained variability ranging from 40 to 70%, depending on the source. With the Enroll-HD dataset, a simple OLS model using as predictors sex and CAG achieves an R^2 of 0.5572. This indicates that the model can explain approximately 56% of the variance in AO, based on the performance on a test set (observations not used during training).

5.2 SNP models

The performance of the models that use genotype information will be compared to this baseline to assess any improvement in prediction accuracy and to identify GeMs.

5.2.1 Metrics comparison

All models were subjected to extensive regularization, with hyperparameters tuned through a grid search (see Table 3 in section 4). The selected hyperparameters for each technique are detailed in Table 4. The final models were trained with these optimized hyperparameters. The predictions computed on the training set predictions to check for overfitting demonstrate satisfactory generalization capability (see appendix B).

Model	**Parameter**	**Value**
Lasso	alpha	0.01
	max_iter	1000
Elastic Net	alpha	0.01
	l1_ratio	0.9
	max_iter	10000
Random Forest	ccp_alpha	0.001
	max_depth	6
	n_estimators	40
XGBoost (approx)	reg_alpha	0.5
	max_depth	3
	n_estimators	10
XGBoost (hist)	reg_alpha	0.1
	max_depth	2
	n_estimators	20

Table 4: Hyperparameter values of the final models found using cross-validation.

The metrics from each resulting model are shown in Table 5, which includes the baseline for easier comparison. The best performance, in terms of metric values, was achieved by the XGBoost model using the histogram tree method, which improved the baseline R^2 by 0.03. The approximate tree method performed almost as well while requiring less training time. Random Forest also achieved a similar R^2, but it took more than

Model	R^2	**MAE**	**MSE**	**Training time**	**Nº features**
OLS	0.5572	0.5124	0.4462	11.3 ms	2
Lasso	0.5478	0.5188	0.4556	10.2 min	867
Elastic Net	0.5395	0.5247	0.464	13.4 min	1164
Random Forest	0.5866	0.4883	0.4165	180.5 min	1159
XGBoost (hist)	**0.5908**	**0.4844**	**0.4123**	**13.7 min**	**33**
XGBoost (approx)	0.5847	0.4904	0.4184	8.3 min	34

Table 5: Model metrics, execution time and number of features used, including the baseline.

10 times the time required to train the XGBoost models. In contrast, the least squares methods did not enhance the prediction compared to the baseline.

Thus, adding more features does not necessarily lead to better results. In addition, the relationship between genetic information and AO seems to be non-linear since models capable of capturing non-linear relationships, such as the XGBoost variants, outperform regularized linear models, which did not uncover additional predictive information from the provided features. These models can take interactions between SNPs into account implicitly. This is in contrast with linear models, which need to explicitly incorporate interaction terms in turn increasing the size of the models.

Comparing the plots produced to graphically study the model's predictions reveals interesting differences between the two methods. In Figure 7 we compare the best-performing models of each type (similar plots for the other models can be found in appendix B). Predictions of the least squares methods exhibit non-symmetric residuals, underestimating AO at older ages and overestimating AO at younger ages. Conversely, tree based methods show a more homogeneous distribution of residuals, although they fail to predict the youngest AOs.

5.2.2 Feature importance

To further inspect the trained models, we examined the features they use. All models identified CAG as the most important predictor. In tree-based methods, CAG is almost always the splitting feature at the root node. However, the subsequent predictors vary between models.

Using Manhattan-like plots, we can easily visualize the SNPs that contribute the most to each model. Figure 8 shows two types of Manhattan plots, again showing just one model of each method. Plots for the remaining models can be found in appendix C, followed by the tables with the top SNPs of each model. The labeling of SNPs in these plots does not indicate significance, it simply highlights some of the most important genes in each model. Notably, certain genes are used across multiple models.

There is no single SNP which is used by all models, though there are intersections within model groups. Between the two linear models, there is an overlap of 839 features. Given that the Lasso model, which has the fewest features among the two, includes 866 SNPs,

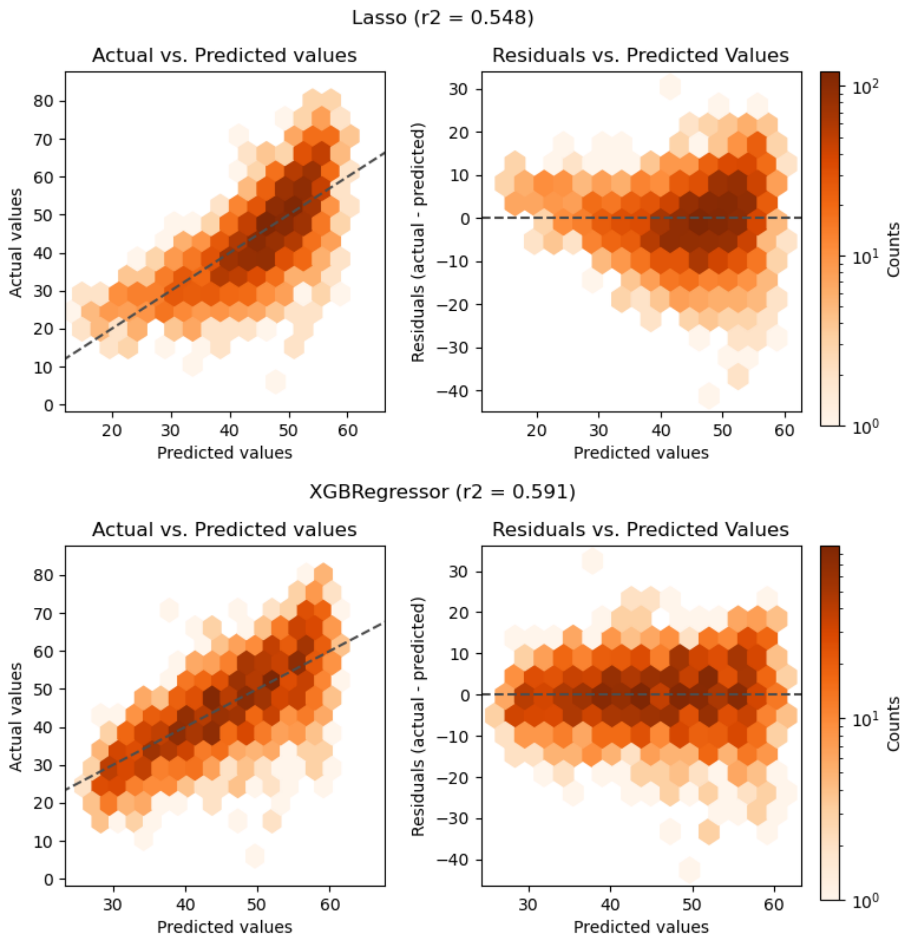
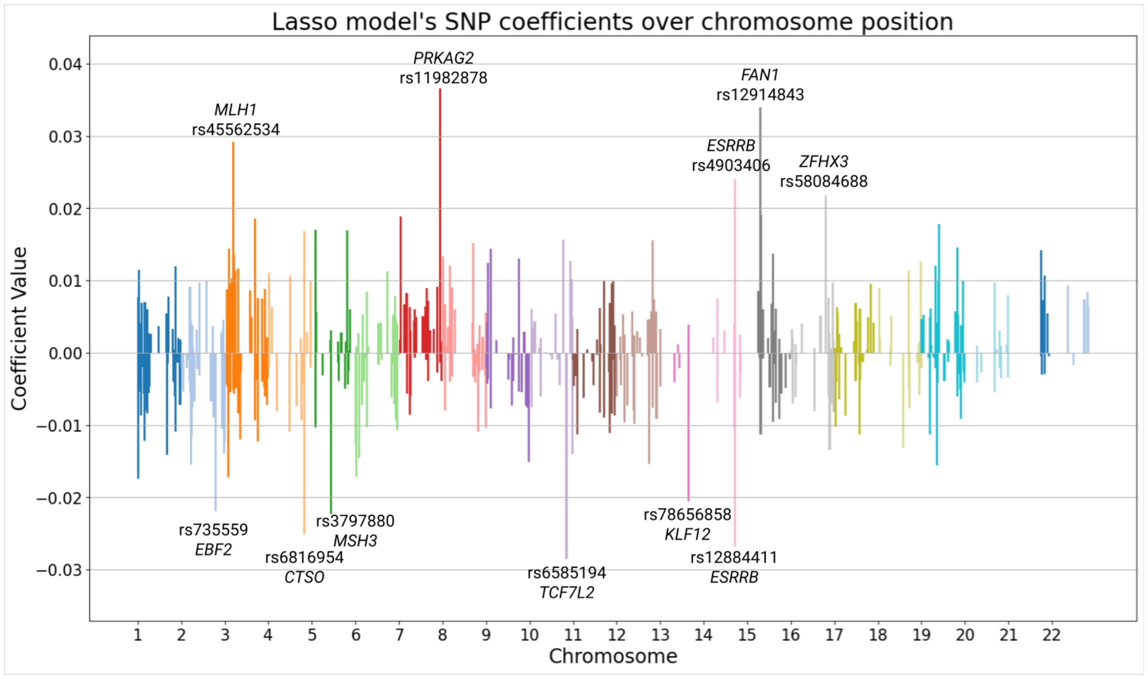


Figure 7: Comparison of least squares method prediction (Lasso), top image, and tree-based method prediction (Histogram XGBoost), bottom image.

we can consider that both models are practically the same. This is further justified upon realizing that the chosen *l1_ratio* for Elastic Net is 0.9, shifting its regularization more towards a Lass-like regularization rather than an *l2* regularization. The five most important SNPs in these models come from the genes *SMARCD3, FAN1, MLH1, TCF7L2* and *ESRRB*. Interestingly, *ESRRB* has two SNPs with high coefficients but opposite effects, rs8017707 hastens onset while rs2278687 delays it.

FAN1 and *MLH1* are commonly reported AO GeMs [16]. *TCF7L2* is not considered to be an AO GeM, but a repressed expression of *TCF7L2* in HD mouse models has been proven to affect the myelin maintenance [67]. *ESRRB* is not a known AO GeM either, although it is used as a pluripotency marker in HD induced pluripotent stem cells from mice models in [68].

The effect of such coefficients, however, cannot be considered to be the cause of the large variability of AO introduced at the start of this project. Since the coefficients were estimated with standardized AO values, we can calculate the shift from the mean that a unit change in the feature's value causes by multiplying the coefficient's value by the AO's original standard deviation, 11.58 years. A SNP with a related coefficient of 0.03 would shift the AO prediction from the mean by 0.35 years if it only had one



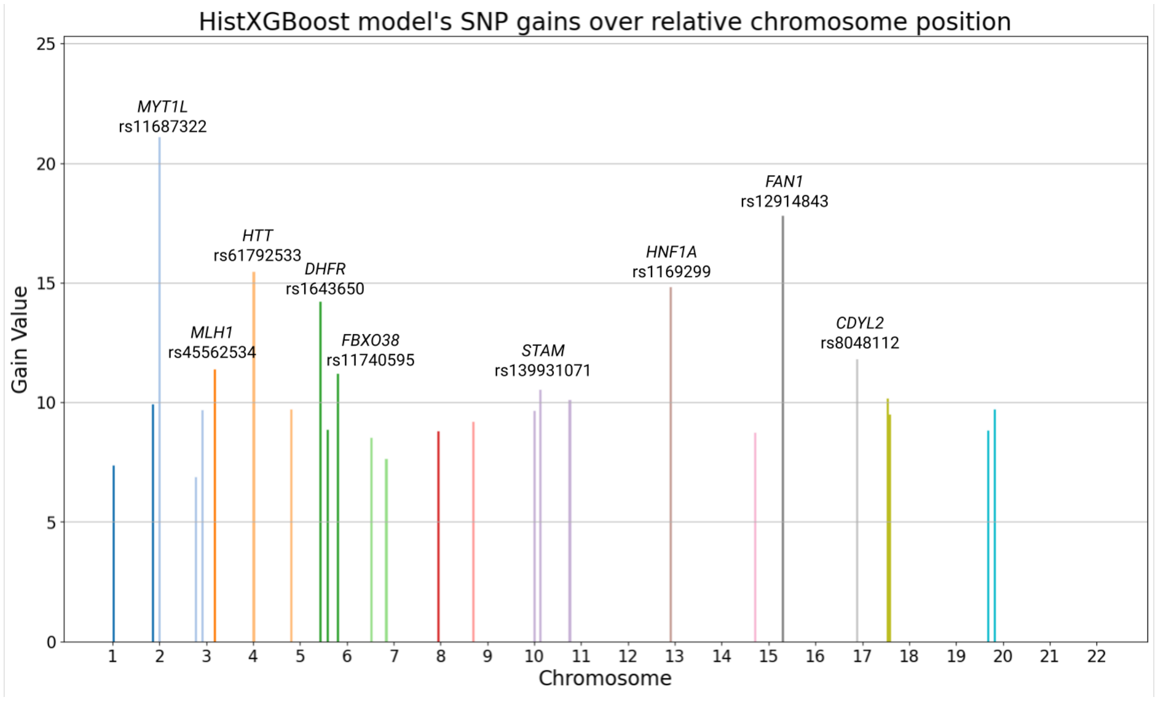


Figure 8: Manhattan plots of a least squares method (Lasso), top image, and a tree-based method (Histogram XGBoost), bottom image.

alternative variant, and by 0.69 years if both chromosomes had the alternative variant. This demonstrates that AO variability cannot be explained by a single SNP, as AO can shift up to 30 years from the mean AO in the most extreme cases.

This is also graphically illustrated in Figure 9, which shows the AO variability over CAG values for a random subset of 500 samples. Cases with an alternative allele for a specific SNP are highlighted in black. If the effect of the alternative variant were significant, we would see these samples shifted consistently to one side or another of the line representing the mean AO for each CAG repeat length. However, this is not the case. The mean curve is calculated using the entire dataset to make it constant between plots.

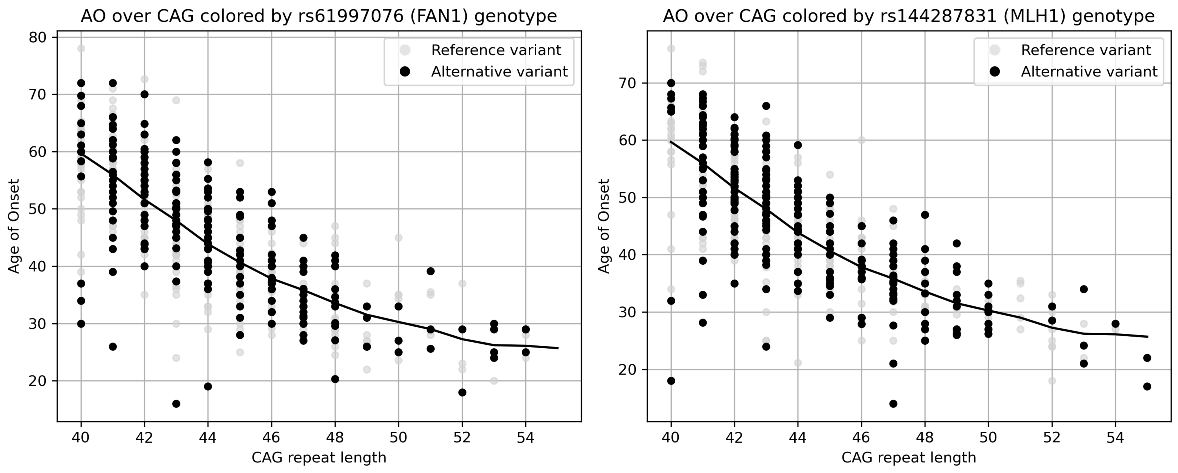


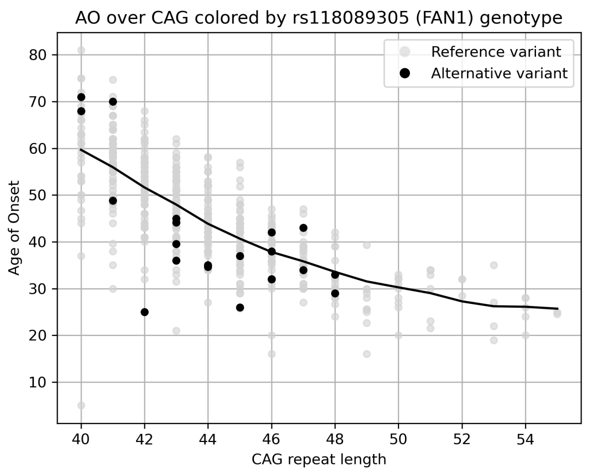
Figure 9: AO over CAG repeat length of 500 random samples, colored by the presence of an alternative allele in SNPs rs61997076 (left) and rs144287831 (right).

Between the tree-based methods we only have two intersecting SNPs: rs10169129 (*MYT1L*) and rs118089305 (*FAN1*). The plots of samples colored by their genotype over the CAG-AO plane in this two cases are different from the linear examples in the density of samples with an alternative allele, being much smaller than in the cases of the SNPs chosen by the linear models. This characteristic is seen in many more SNPs chosen by the two XGBoost methods. Nevertheless, their effect on AO is not clearer than the last examples.

The two XGBoost tree methods have 7 SNPs in common. The first two are the already seen rs10169129 (*MYT1L*) and rs61997076 (*FAN1*), followed by rs141338757, an *HTT* SNP whose alternative variant is very scarcely found but seems to delay AO in the few samples which have it (Figure 11). The rest of SNPs are rs57013064 (*CDYL2*), rs72841819 (*BTRC*), rs118089305 (*FAN1*), and rs5743061 (*PMS1*).

Amongst the reviewed literature, we found SNPs which coincide with some of the top SNPs amongst the models:

- **rs144287831** from the mismatch repair gene ***MLH1***, which is suspected to act on CAG repeat instability [5]. This SNP is used by the linear models and XGBoost (hist).



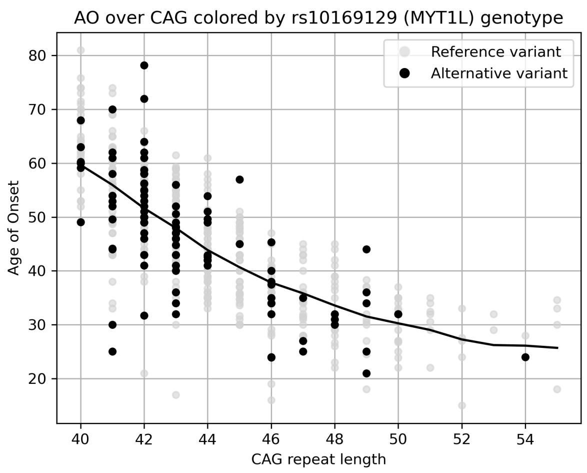


Figure 10: AO over CAG repeat length of 500 random samples, colored by the presence of an alternative allele in SNPs rs118089305 (left) and rs10169129 (right).

- **rs79727797** from the transcription regulator gene ***TCERG1***, which is associated with a quasi-tandem repeat in the gene that hastens HD onset [69]. This SNP is only used by XGBoost (hist).

These findings should not be interpreted as being genomic positions directly related to the phenotype. We could be viewing a SNP in the same LD block as the effect causing polymorphism. What is more informative, and indicative that the trained models identified true signals, is the comparison at the gene level. Overall, the genes where the SNPs with the highest importance as features in all models are located are consistent with the results of the GWAS analysis conducted in a recent study by Lee et al. [8], where the authors used the same dataset. Figure 12 summarizes their findings regarding AO, labeling with solid arrows loci that are related significantly to the phenotype, and including candidate modifier genes on top, in red those which are DNA maintenance genes. Comparing these external results obtained with a different methodology than ours with our Manhattan plots, we see that there are many similarities.

There are 3 genes with at least 1 SNP which is a feature in all 5 models: the already mentioned *FAN1* and *MYT1L*, both important AO modifier candidates, and *CDYL2*, another gene from GO:0140110 as *MYT1L*, related to the transcription regulation. While *CDYL2* does not appear among potential AO modifiers, it is mentioned in a profiling thesis of MSNs from HD mouse models by Fenster (2011) [70].

In summary, the majority of SNPs and genes highlighted by the feature importance study summarized in this section are related to DNA maintenance machinery, aligning with the initial hypothesis [6], and the regulation of transcription. This reinforces the notion that the models have likely captured meaningful associations between genetic factors and AO.

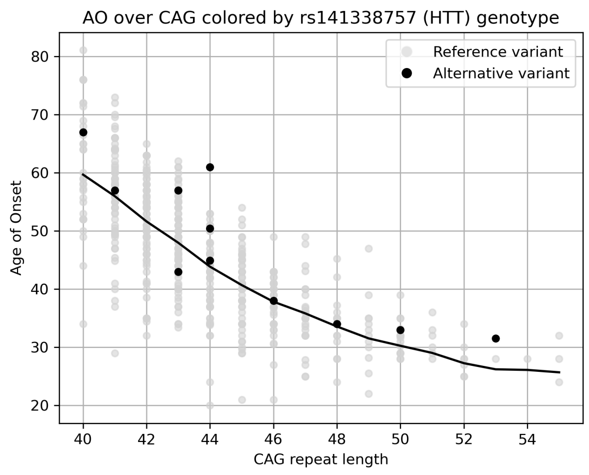


Figure 11: AO over CAG repeat length of 500 random samples, colored by the presence of an alternative allele in rs141338757.

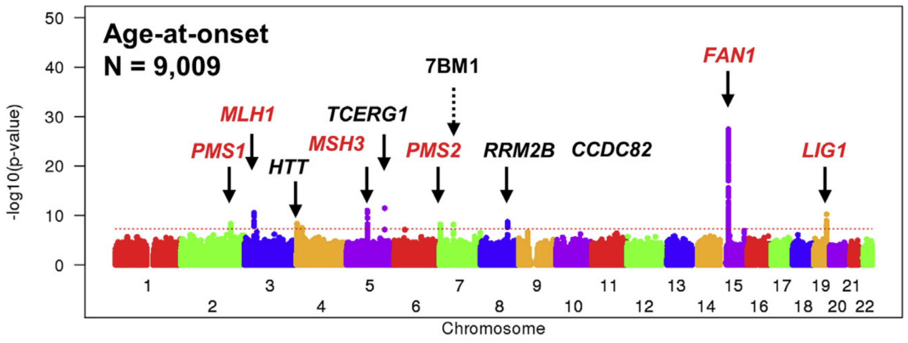


Figure 12: GWAS by Lee et al. [8], showing significant associations between AO and different chromosome loci.

5.2.3 GO enrichment analysis

Figure 13 illustrates the GO term proportions of the genes associated with all the SNPs used in each model, juxtaposed with the initial proportions of features in the data matrix used for training the models (background proportions).

Lasso, Elastic Net, and Random Forest models, with approximately a thousand features each, closely resemble the largest background proportions. In contrast, the XGBoost methods exhibit GO term proportions that deviate more from the background and vary between them. Across all models, over 40% of the SNPs used as regressors belong to GO:0140110 (transcription regulator activity), followed by terms related to protein degradation and ubiquitin. The enrichment analysis shows the best performing XGBoost model has three enriched terms: GO:0006298 (mismatch repair), GO:0046655 (folic acid metabolism) and the *Extra genes* group. This last group is enriched by all models, primarily due to multiple *FAN1* SNPs. The term of transcription regulation is only significantly enriched in Lasso, Elastic Net and Approx XGBoost. See appendix D for the complete results of the enrichment test.

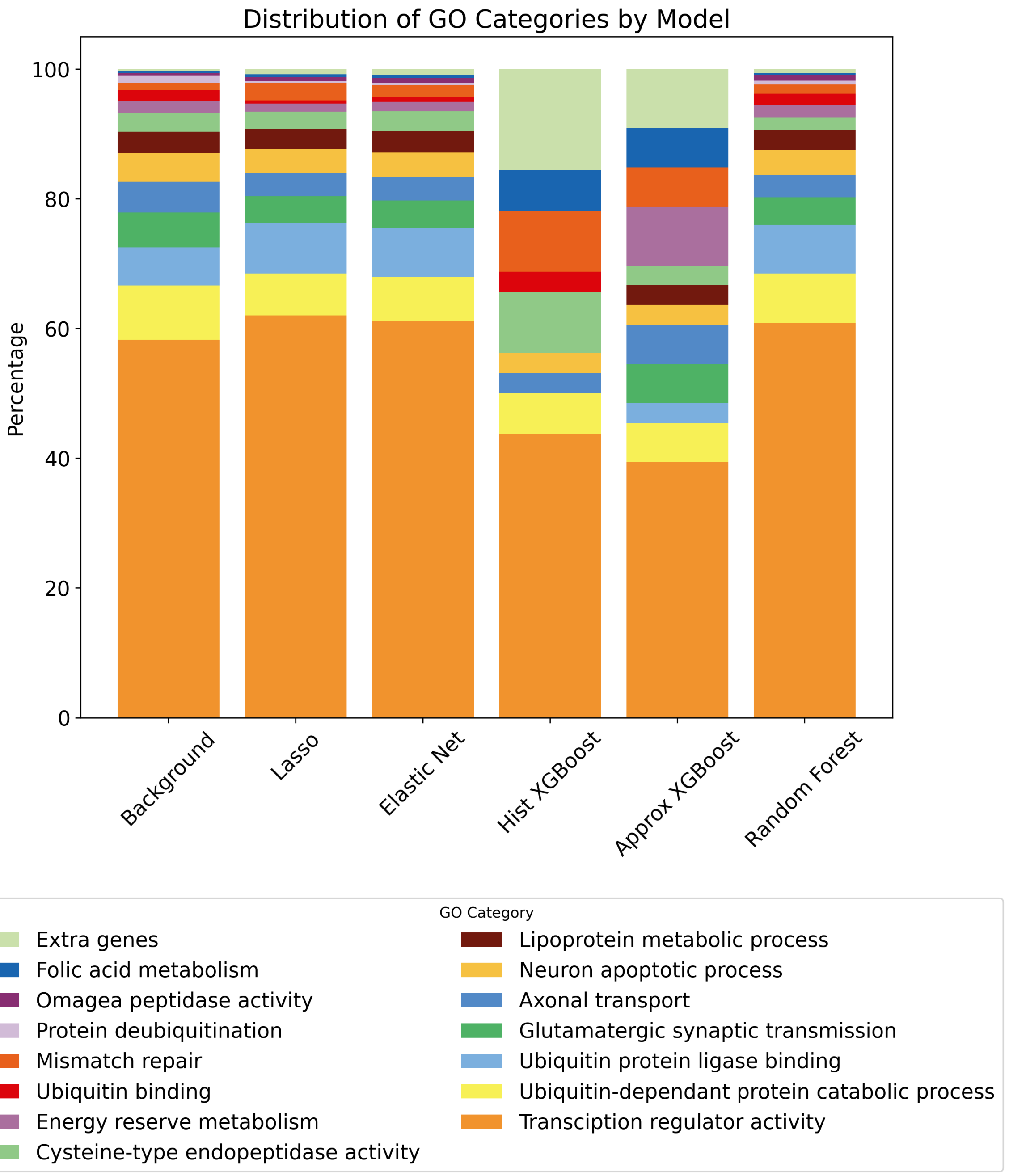


Figure 13: GO terms proportions of the features used in each model plus the background proportions.

6 Execution schedule

In this section, we present the execution schedule for the project, which is meticulously structured to ensure timely and efficient completion. The schedule encompasses a set of project management tools: a comprehensive Work Breakdown Structure (WBS) to delineate the tasks, a Program Evaluation and Review Technique (PERT) chart for critical path analysis, and a GANTT chart to visually track activity progress over time.

6.1 Work Breakdown Structure

The WBS delineates the project's key components, providing a detailed roadmap for successful execution. Our WBS is organized into four main parts: project preparation, data preprocessing, machine learning regression models, and project writing. Each segment is broken down into specific tasks to facilitate a clear understanding of the project workflow and ensure systematic progress toward our goals. Figure 14 shows what activities are included in each of these main parts. These individual activites are described below, including their estimated duration.

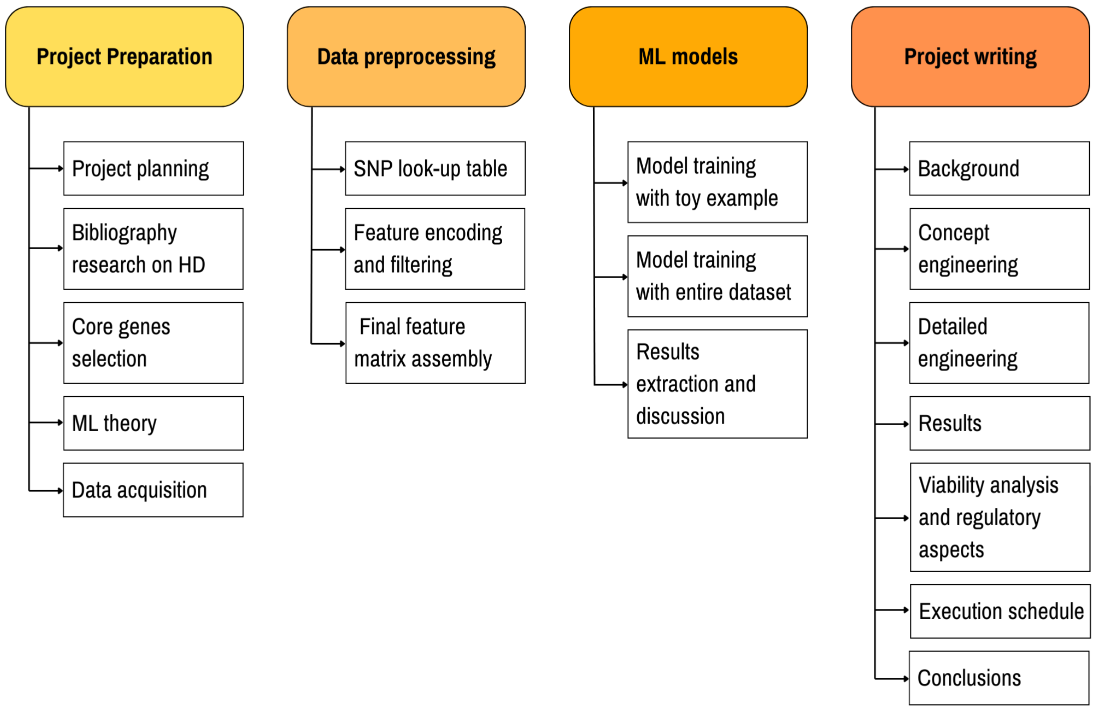

Project Preparation
Project planning
Bibliography research on HD
Core genes selection
ML theory
Data acquisition
Data preprocessing
SNP look-up table
Feature encoding and filtering
Final feature matrix assembly
ML models
Model training with toy example
Model training with entire dataset
Results extraction and discussion
Project writing
Background
Concept engineering
Detailed engineering
Results
Viability analysis and regulatory aspects
Execution schedule
Conclusions

Figure 14: WBS diagram of the project.

1	**Project preparation**

1.1	**Project planning**	Duration: 1 day
Set goals and time landmarks from the beginning of the project. Define the key activities of the project, where to start and how. Talk over the work methodology with the project tutor.		

1.2	**Bibliography research on HD**	Duration: 7 days
Literature review on the HD investigation stage, find out what has already been done on the AO prediction challenge. It is essential to look for studies which explore different pathways that could be related to disease mechanisms, most importantly to disease onset. Review basic biology concepts related to SNPs. Keep track of the references consulted through the refenerce management software Mendeley.		

1.3	**Core genes selection**	Duration: 2 days
From the literature review on HD mechanisms, assemble a list of genes to be included in the models, either by individual genes (specially HD genetic modifiers) or by groups of genes of a relevant pathway.		

1.4	**ML theory and packages research**	Duration: 5 days
Having read about the bibologic setting of the project, review the knowledge on ML algorithms that can be useful towards achieving the project's objective.		

1.5	**Data acquisition**	Duration: 1 day
Ask the project's director for the Enroll-HD data, and understand how to handle it through bash.		

2	**Data preprocessing**

2.1	**SNP look-up table generation**	Duration: 10 days
Assemble a table containing all SNPs registered for each defined core gene. Search for possible tools which can return SNPs based on an input list of genes, and implement the best option.		

2.2	**Feature encoding and filtering**	Duration: 20 days
Assemble a first version of a feature matrix, filtering the original data using the look-up table generated in the activity 2.1 to reduce the size of the dataset in a way which makes the dataset more HD focused. Make the matrix have a ML compatible format: observations in rows (each subject being represented by a row), the genotype being encoded as an integer, clear headers.		

2.3	**Final feature matrix assembly**	Duration: 10 days
Filter the feature matrix obtained in activity 2.2 by minimum alternative variant prevalence. Make a subset of this feature matrix for easy model testing, the toy example.		

3	**ML regression models**

3.1	**Model training with toy example**	Duration: 30 days
Test the ML models proposed after defining the concept engineering in a notebook with the toy example. Make prototypes of the custom functions to read large matrices, to scale features and AO values, and to evaluate model training and performance.		

3.2	**Model training with entire dataset**	Duration: 15 days
Make separate scripts for each model to execute separately with the entire dataset. Check that the resulting models are not overfitted, adjust the parameters of the grid search to obtain the best model possible for each technique.		

3.3	**Results extraction and discussion**	Duration: 5 days
From the saved regressors obtained in activity 3.2, study which are the most important features contributing to the predictions, and present the results in the most appropriate format for each model. Discuss the obtained results with the project's directors, and check for positive controls between the most relevant features found across models.		

4	**Project writing**

4.1	**Background**	Duration: 7 days
Commit to paper the literature review done at the beginning of the project. Complete throughout the project with new references found or with concepts which have become relevant at some point during the process.		

4.2	**Concept engineering**	Duration: 7 days
After knowing what are the common practices in the environment where the project is set, and what are the available resources, state what are the realistic options for all the steps necessary to attain our objective.		

4.3	**Detailed engineering**	Duration: 4 days
Describe what has been done and how regarding data handling, transformation, feature selection, model training and results.		

4.4	**Results**	Duration: 4 days
Write the results and discussions produced during the activity 3.3.		

4.5	**Viability analysis and regulatory aspects**	Duration: 2 days
Think about the strengths and weaknesses of the project by creating a SWOT diagram. Complement the knowledge about the project external limits by searching for regulatory factors that may influence the applications of this project.		

4.6	**Execution schedule creation**	Duration: 1 days
Create an execution schedule made up of a PERT diagram to control which activities should not be delayed, and a GANTT diagram to keep track of the activities that need to be done throughout the project.		

4.7	**Conclusions**	Duration: 2 days
Write the final conclusions of the project once all activities are completed.		

6.2 PERT diagram

PERT is a project management tool utilized to analyze and depict the tasks involved in project completion. Table 6 outlines all activities along with their precedence relationships and estimated duration. The PERT diagram takes this relationships and represents them graphically, having each activity on an arrow. Each node is a project's landmark, having two times: *t early* (left number), which is the time when the activities coming out of that node can be started, and *t last* (right number), the time when the activities pointing to that node should be finished to avoid delays in the project. The top number is simply a node identification.

The critical path is the combination of activities along which the time that its activities can be started and the time when they should be finished is the same, meaning that any delay in these activities affects directly the total duration of the project. Our critical path is mainly formed by the initial literature review to select the core genes, the feature matrix assembly and the model training. Any activity outside this path can be postponed until a limit (before reaching the *t last* of the node to which they are directed) without altering the project's total duration.

ID	Activity name	Dependencies	Duration
A	Project planning	-	1
B	Bibliography research on HD	-	7
C	Core genes selection	B	2
D	ML theory research	B	5
E	Data acquisition	-	1
F	SNP look-up table generation	C	10
G	Feature encoding and filtering	E,F	20
H	Final feature matrix assembly	G,M	10
I	Model training with toy example	H	30
J	Model training with entire dataset	I	15
K	Results extraction and discussion	J	5
L	Background writing	B	7
M	Concept engineering	B,D	7
N	Detailed engineering writing	J	4
O	Results writing	K	4
P	Viability analysis and regulatory aspects	M	2
Q	Execution schedule creation	A	1
R	Conclusions	L,N,O,P,Q	2

Table 6: Activity table for the PERT diagram.

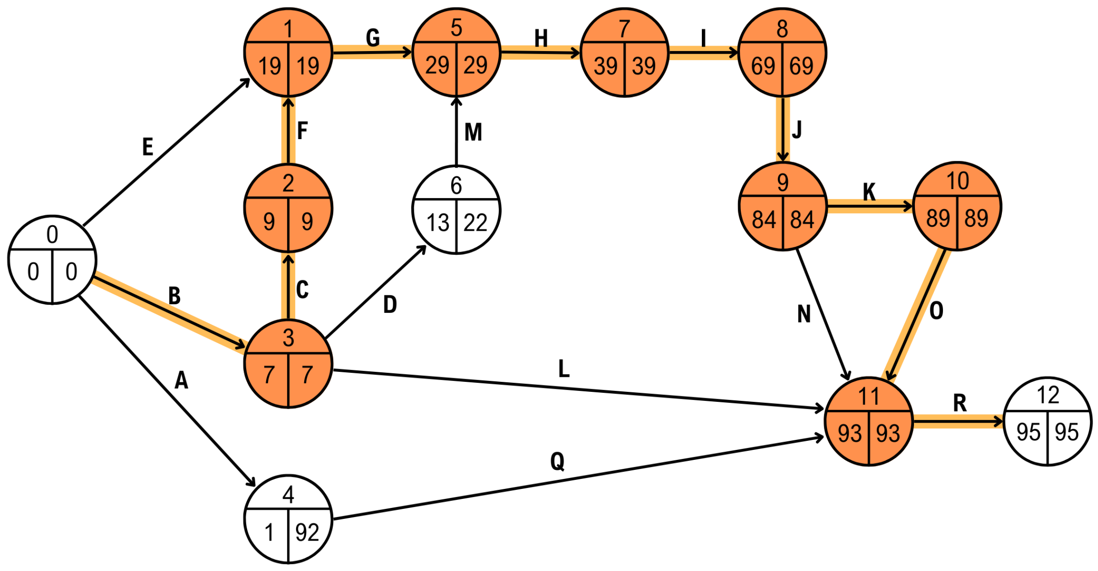


Figure 15: PERT diagram of the project.

6.3 GANTT diagram

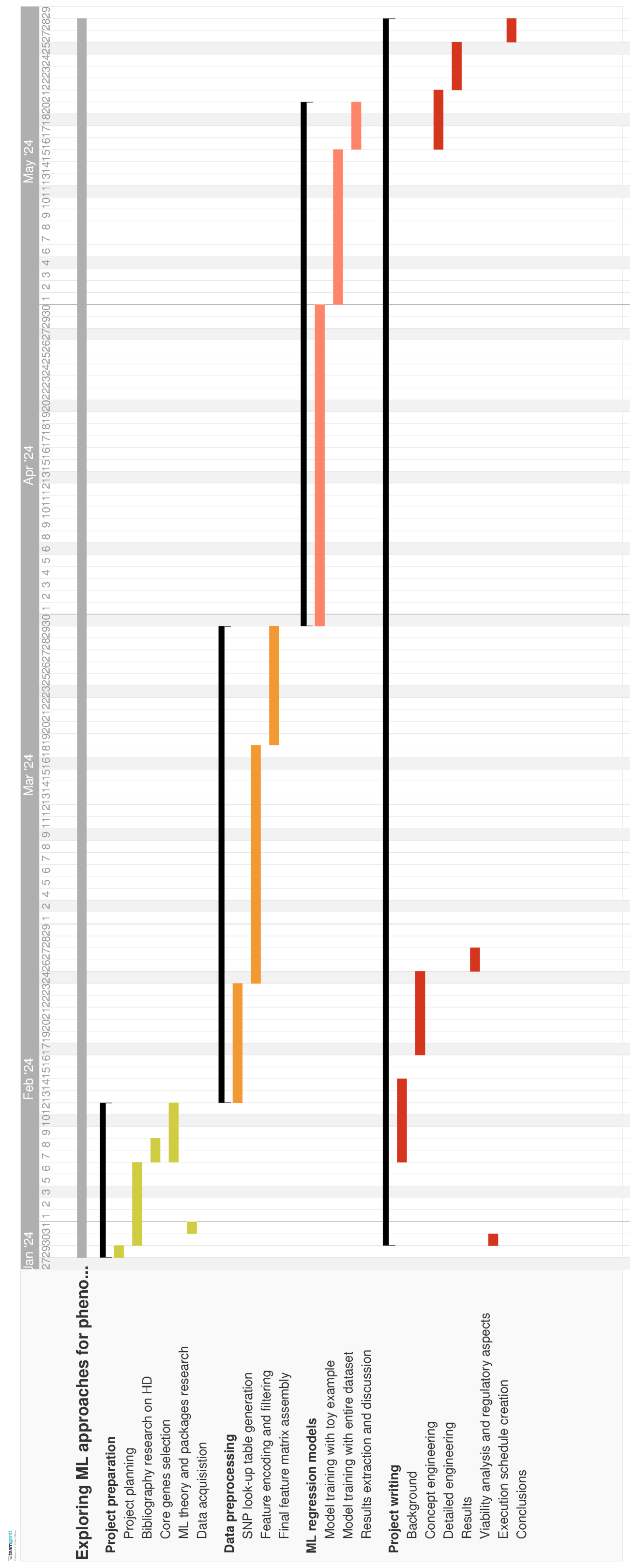
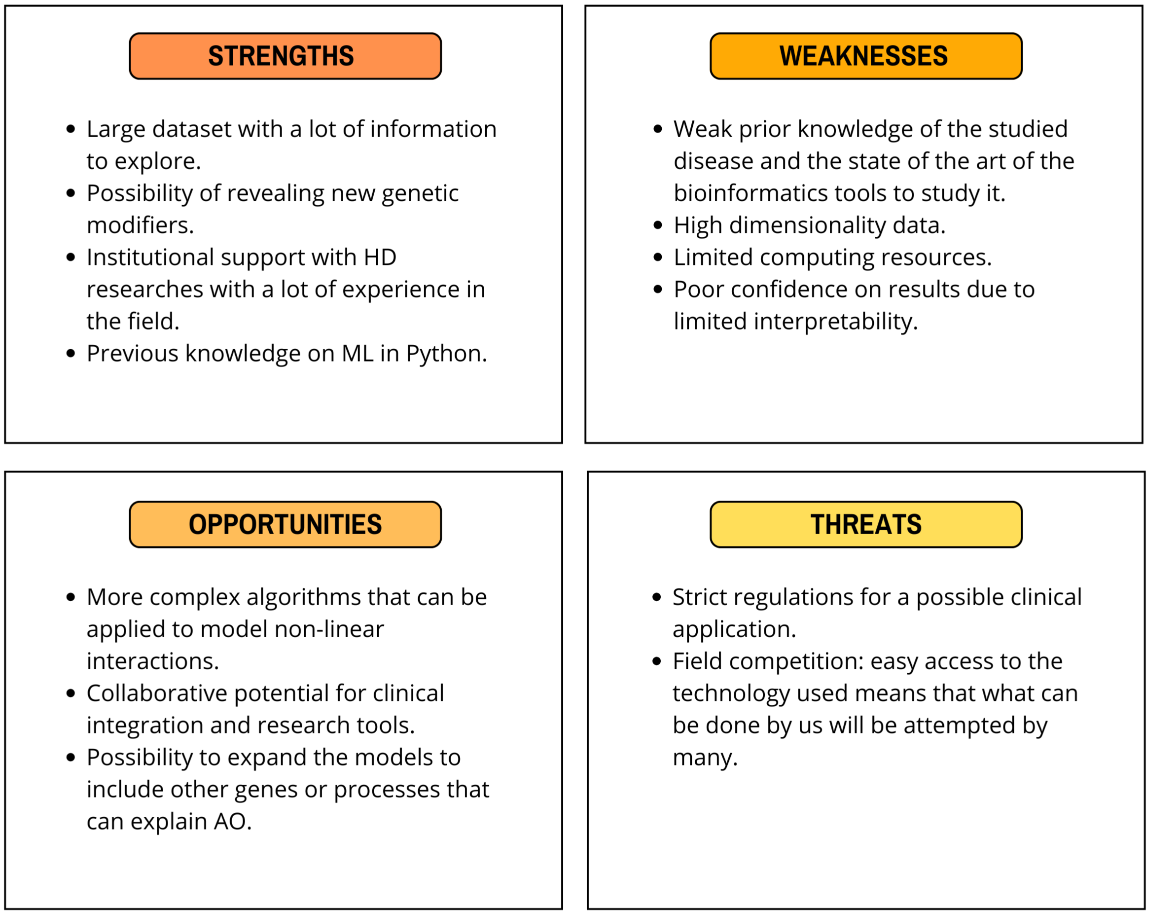


Figure 16: GANTT diagram of the project schedule, starting the 29th of January and ending the 28th of May.

7 Technical viability

To assess the technical viability of this research project a SWOT analysis is presented. With it, we identify the strengths, weaknesses, opportunities and threats associated with the technical aspects of the project in a structured framework to evaluate the internal and external factors that could influence the project's success from a technological standpoint. The diagram which summarizes the key points of the analysis is presented in Figure 17.



STRENGTHS

- Large dataset with a lot of information to explore.
- Possibility of revealing new genetic modifiers.
- Institutional support with HD researches with a lot of experience in the field.
- Previous knowledge on ML in Python.

WEAKNESSES

- Weak prior knowledge of the studied disease and the state of the art of the bioinformatics tools to study it.
- High dimensionality data.
- Limited computing resources.
- Poor confidence on results due to limited interpretability.

OPPORTUNITIES

- More complex algorithms that can be applied to model non-linear interactions.
- Collaborative potential for clinical integration and research tools.
- Possibility to expand the models to include other genes or processes that can explain AO.

THREATS

- Strict regulations for a possible clinical application.
- Field competition: easy access to the technology used means that what can be done by us will be attempted by many.

Figure 17: SWOT analysis of the project.

By examining the strengths, we aim to highlight the technical capabilities and resources that give the project a competitive edge. We can consider that the two major strengths of the project are the dataset to which we have been given access and the professional setting of HD researchers involved. The dataset contains information on all autosomal chromosomes and a large number of SNPs per gene, which provides us with the opportunity to expand our research without having information limits and reveal new genetic modifiers. Additionally, the project is situated in a professional environment with experienced HD researchers whom we can consult to check the validity of our literature findings, which is crucial for conducting a ML exploration with solid biological foundations that can yield significant results disease-wise. My previous knowledge of ML applications through Python also contributes to the strength of the project, making the model training phase quicker.

Conversely, identifying the weaknesses allows us to recognize the internal limitations and challenges that could impede progress. The most significant weakness of the project is related to one of our main strengths: the dimensionality of our data. Although it gives us the power to study the whole genome, the computational resources needed to include the entire genome are not available, necessitating a restriction on what information we keep, which could potentially leave out important modifier points in the process. Moreover, although we have institutional support for the biological aspect of the research, I personally lack initial knowledge of the disease. Understanding the biological aspect of the disease is key at the start of the project to choose the processes on which to limit our search. This step can be biased due to a short period to get acquainted with the disease, the literature on it, and the state-of-the-art procedures in similar cases. Lastly, the results we might obtain are not as conclusive as an experimental study, which can empirically prove that a potential GeM has a true effect on disease onset. We can only explain the results based on how each model gives importance to each feature and provide possible biological theories on how that effect could occur.

The opportunities section focuses on external factors and trends that the project can leverage to its advantage. The principal opportunity is that the algorithms tried in this project are very simple. If they can improve the baseline, it is likely that a more complex algorithm will be able to improve the AO prediction much more, especially algorithms that model non-linear interactions such as Neural Networks. We could also expand the model to include other processes and genes once the pipeline of the process toward model training is created. An AO prediction algorithm that provides more accurate predictions than the standard Langbehn's formula could potentially become a new clinical prediction tool in genetic counseling. It would also be a very useful tool in research, allowing us to control for the effect of the already known modifiers to test the effect of more polymorphisms and interactions between them.

Finally, the threats analysis considers external challenges and risks that could negatively impact the project's technical viability. The most direct threat to the research is losing our originality aspect, as the technology used is very simple and easily accessible, meaning that what we are doing can be done by many others simultaneously or even before us. If the results achieved are good enough to propose a new AO prediction tool for clinical application, the regulations surrounding technology testing and transfer are very strict and require long, demanding procedures.

8 Economic viability

Three main components are essential to this project: data, technical resources, and human resources.

Data for this project was graciously provided by CHDI Foundation, Inc. as part of the Enroll-HD study. CHDI Foundation is a nonprofit biomedical research organization dedicated to collaboratively developing therapeutics for Huntington's disease (HD). The economic costs associated with data collection are not within the scope of this project and therefore are not factored into our analysis.

Due to the substantial memory requirements of the data, analysis cannot be performed using standard laptops. Instead, a computationally capable computer is necessary. Most of the data manipulation and analysis were conducted on a remote computational cluster provided by Creatio, a service that had been paid for years prior to the initiation of this project. To estimate the cost of this service, we referenced the pricing of Amazon Elastic Compute Cloud (Amazon EC2) from Amazon Web Services. Based on our execution schedule (refer to section 6), we estimate that we utilized the cluster for approximately 93 days. Assuming an average of 3 hours of code execution per day, with a computational memory requirement of 10GB (or 10.74 GiB) and a Linux operating system, we found that the most cost-effective instance is the *t3.xlarge* [71]. Opting for the EC2 Instance Savings Plan for the minimum duration (1 year) would result in a total cost of 1,287.72 USD, or alternatively, selecting the on-demand option would amount to approximately 656.92 USD for four months, making it the more economical choice. Notably, the personal laptop used to access the remote computational resources is not considered a project expense.

Regarding human resources, the project team consists of myself and the supervising tutor. As we were not directly compensated for our involvement in the project, human resources are not considered as an economic limitation.

In conclusion, the primary economic consideration for this project is the acquisition of a sufficiently powerful computing machine. If Creatio did not possess such a resource prior to the project, the estimated total cost would approximate **656.92 USD**.

9 Regulatory and legal aspects

9.1 Enroll-HD data usage

The dataset used ("GWAS12345" data) involving human subjects is shared by the CHDI foundation with qualified investigators given their institutional assurance of subject confidentiality and compliance with General Data Protection Regulation (GDPR) 2016/679 requirements with respect to personal data. This regulation ensures the protection and privacy of individuals' data, particularly in sensitive research areas such as HD. GDPR mandates rigorous controls over data handling, requiring that all personal data be processed lawfully, transparently, and for a specific purpose [72].

The use of the data under GDPR involves several key considerations. Firstly, the data must be anonymized or pseudonymized to protect the identity of the subjects. We already received the data with anonymous identifications for each subject. We must ensure that data usage is restricted to the stated research objectives when we asked for the dataset to the foundation, and any deviation would require additional consent. Furthermore, data security measures are implemented to prevent unauthorized access or breaches, saving the data in password-protected machines. The principles of data minimization and purpose limitation under GDPR mean that only the data necessary for the ML models' development and validation should be utilized. This compliance not only protects the participants' privacy but also enhances the ethical integrity of the research. Therefore, throughout the thesis, stringent adherence to GDPR guidelines will be maintained to uphold the trust and confidentiality placed by the CHDI Foundation and the study participants.

9.2 Medical algorithms legislation

The models created throughout this thesis only have research purposes. If this was to be applied as clinician support, several additional steps should be taken to ensure compliance with the relevant regulatory frameworks. Firstly, the ML predictive software would need to be evaluated to determine if it qualifies as a "Software as a Medical Device" (SaMD) under the Regulation (EU) 2017/745 – Medical Device Regulation (MDR) [73]. This regulation stipulates that any software intended to process, analyze, create, or modify medical information must adhere to strict standards to ensure safety, performance, and compliance with intended medical purposes. A thorough regulatory assessment would be necessary to classify the software appropriately and to confirm it meets all applicable requirements. By addressing these regulatory aspects, the transition from research to clinical application can be achieved responsibly, ensuring that the benefits of the ML models are realized without compromising patient safety or legal compliance.

10 Conclusions and future outlook

This thesis explores the potential of ML models to predict the AO in HD using genotyping data, aiming to identify genetic variants significantly contributing to AO prediction. The study benchmarks various regression models, comparing their performance to a baseline model that relied on CAG repeat length and sex.

The baseline OLS model, using CAG repeat length and sex as predictors, explained approximately 56% of the variance in AO. Among the tested models, the XGBoost model using the histogram tree method achieved the best performance, with an R^2 of 0.5908, improving the baseline by 3%. Other tree-based methods like Random Forest and approximate XGBoost also performed well, while regularized linear models like Lasso and Elastic Net did not outperform the baseline although they were able to rediscover known GeMs and propose new candidates. The better performance of tree-based methods suggests that the relationship between genetic information and AO is non-linear and that these models are better suited for capturing complex interactions among genetic features. This points towards a future research direction of exploring algorithms which can better fit these non-linear interactions such as Neural Networks.

Given the limited number of observations compared to the number of features, a strategy to overcome this fundamental problem was required. To that end, we have implemented regularization, cross-validation, and ensemble techniques, successfully addressing the problem in all models.

CAG repeat length consistently emerged as the most significant predictor across all models. Several SNPs were identified as important across different models. Notably, SNPs in genes such as *FAN1, MLH1, TCF7L2, ESRRB*, and *MYT1L* were highlighted. The identified SNPs are consistent with known GeMs of HD and mechanisms related to DNA maintenance, transcription regulation, and protein degradation, reinforcing the validity of the models' feature selection. We found that the effect of individual SNPs on AO was relatively small, indicating that AO variability cannot be attributed to single genetic variants alone but rather a combination of multiple genetic factors.

Gene Ontology enrichment analysis revealed that the most important SNPs identified by the models were associated with genes involved in transcription regulation, protein degradation, and DNA mismatch repair. Notably, genes related to DNA maintenance machinery were consistently highlighted, aligning with existing literature on GeMs of HD. The best performing XGBoost model, in particular, significantly enriched the terms of mismatch repair and folic acid metabolism, plus the group containing other possible GeMs.

One important assumption worth noting is that we encoded the SNP genotype by considering the effect of a homozygous SNP for the alternative allele to be double the effect of a heterozygous SNP. This assumption could be biologically inaccurate. Another simplification is that the CAG repeat length used here is not representative of the actual CAG repeat length found in patients, as it is an unstable mutation that undergoes somatic expansions differently in various tissues. These expansions are thought to accelerate the disease process [19], so controlling for different CAG lengths in different tissues other than blood and considering the age at which the mutation was genotyped could improve the AO prediction models. Finally, the CAG length typically used in AO prediction, including in this project, is the longest mutation of the two chromosomes, which is the dominant one. However, studies suggest there may be an interaction between the two alleles [4]. Although the primary focus of this thesis was on the effect of SNPs rather than CAG length, uncontrolled variables could still influence the phenotype we are trying to explain.

In conclusion, the application of machine learning models to predict AO in HD using genotypic data demonstrated that incorporating additional genetic features beyond CAG repeat length can modestly improve prediction accuracy. The study confirmed the relevance of known GeMs and identified new potential candidates for further investigation. These findings enhance our understanding of the genetic factors influencing HD onset and highlight the complexity of the disease's genetic architecture. Future research could focus on refining these models and exploring the biological mechanisms underlying the identified genetic associations to develop targeted interventions for delaying HD onset.

11 Bibliography

[1] M. Duyao, A. Lazzarini, A. Falek, *et al.*, *Trinucleotide repeat length instability and age of onset in huntington's disease*, 1993. [Online]. Available: `http://www.nature.com/naturegenetics`.

[2] E. M. Gatto, N. G. Rojas, G. Persi, J. L. Etcheverry, M. E. Cesarini, and C. Perandones, "Huntington disease: Advances in the understanding of its mechanisms," *Clinical Parkinsonism & Related Disorders*, vol. 3, p. 100 056, 2020, ISSN: 25901125. DOI: `10.1016/j.prdoa.2020.100056`.

[3] J.-L. Li, M. R. Hayden, E. W. Almqvist, *et al.*, *A genome scan for modifiers of age at onset in huntington disease: The hd maps study*, 2003.

[4] L. Djoussé, B. Knowlton, M. Hayden, *et al.*, "Interaction of normal and expanded cag repeat sizes influences age at onset of huntington disease," *American Journal of Medical Genetics*, vol. 119 A, pp. 279–282, 3 Jun. 2003, ISSN: 15524825. DOI: `10.1002/ajmg.a.20190`.

[5] J. M. Lee, V. C. Wheeler, M. J. Chao, *et al.*, "Identification of genetic factors that modify clinical onset of huntington's disease," *Cell*, vol. 162, pp. 516–526, 3 Aug. 2015, ISSN: 10974172. DOI: `10.1016/j.cell.2015.07.003`.

[6] P. A. Holmans, T. H. Massey, and L. Jones, *Genetic modifiers of mendelian disease: Huntington's disease and the trinucleotide repeat disorders*, Oct. 2017. DOI: `10.1093/hmg/ddx261`.

[7] J. F. Gusella and M. E. MacDonald, "Huntington's disease: The case for genetic modifiers," *Genome Medicine*, Aug. 2009.

[8] J. M. Lee, Y. Huang, M. Orth, *et al.*, "Genetic modifiers of huntington disease differentially influence motor and cognitive domains," *American Journal of Human Genetics*, vol. 109, pp. 885–899, 5 May 2022, ISSN: 15376605. DOI: `10.1016/j.ajhg.2022.03.004`.

[9] R. Ghosh, S. J. Tabrizi, C. Nóbrega, and L. P. de Lameida, *Polyglutamine Disorders*. Springer, Jan. 2018, pp. 1–28. DOI: `https://doi.org/10.1007/978-3-319-71779-1`.

[10] A. A. Hubers, N. Reedeker, E. J. Giltay, R. A. Roos, E. V. Duijn, and R. C. V. D. Mast, "Suicidality in huntington's disease," *Journal of Affective Disorders*, vol. 136, pp. 550–557, 3 Feb. 2012, ISSN: 01650327. DOI: `10.1016/j.jad.2011.10.031`.

[11] A. Reiner, R. L. Albin, K. D. Anderson, C. J. D'amato, J. B. Penneyt, and A. B. Youngt, *Differential loss of striatal projection neurons in huntington disease*, 1988. [Online]. Available: `https://www.pnas.org`.

[12] J.-P. Vonsattel, R. H. Myers, T. J. Stevens, R. J. Ferrante, E. D. Bird, and E. P. Richardson, *Neuropathological classification of huntinqton's disease*, 1985. [Online]. Available: `https://academic.oup.com/jnen/article/44/6/559/2613653`.

[13] P. C. Nopoulos, E. H. Aylward, C. A. Ross, *et al.*, "Cerebral cortex structure in prodromal huntington disease," *Neurobiology of Disease*, vol. 40, pp. 544–554, 3 Dec. 2010, ISSN: 09699961. DOI: `10.1016/j.nbd.2010.07.014`.

[14] J. MacMillan, R. Snell, A. Tyler, *et al.*, "Molecular analysis and clinical correlations of the huntington's disease mutation," *The Lancet*, vol. 342, pp. 954–958, 8877 Oct. 1993, ISSN: 01406736. DOI: `10.1016/0140-6736(93)92002-B`.

[15] G. P. Bates, R. Dorsey, J. F. Gusella, *et al.*, *Huntington disease*, Apr. 2015. DOI: `10.1038/nrdp.2015.5`.

[16] J. F. Gusella, J. M. Lee, and M. E. Macdonald, *Huntington's disease: Nearly four decades of human molecular genetics*, Oct. 2021. DOI: `10.1093/hmg/ddab170`.

[17] V. C. Wheeler, W. Auerbach, J. K. White, *et al.*, *Length-dependent gametic cag repeat instability in the huntington's disease knock-in mouse*, 1999. [Online]. Available: `https://academic.oup.com/hmg/article/8/1/115/2356046`.

[18] L. Kennedy, E. Evans, C. M. Chen, *et al.*, "Dramatic tissue-specific mutation length increases are an early molecular event in huntington disease pathogenesis," *Human Molecular Genetics*, vol. 12, pp. 3359–3367, 24 Dec. 2003, ISSN: 09646906. DOI: `10.1093/hmg/ddg352`.

[19] R. M. Pinto, E. Dragileva, A. Kirby, *et al.*, "Mismatch repair genes mlh1 and mlh3 modify cag instability in huntington's disease mice: Genome-wide and candidate approaches," *PLoS Genetics*, vol. 9, 10 Oct. 2013, ISSN: 15537390. DOI: `10.1371/journal.pgen.1003930`.

[20] M. Ciosi, A. Maxwell, S. A. Cumming, *et al.*, "A genetic association study of glutamine-encoding dna sequence structures, somatic cag expansion, and dna repair gene variants, with huntington disease clinical outcomes," *EBioMedicine*, vol. 48, pp. 568–580, Oct. 2019, ISSN: 23523964. DOI: `10.1016/j.ebiom.2019.09.020`.

[21] J.-M. Lee, E. M. Ramos, J.-H. Lee, *et al.*, *Cag repeat expansion in huntington disease determines age at onset in a fully dominant fashion*, 2012. [Online]. Available: `www.neurology.org`.

[22] H. Vitet, V. Brandt, and F. Saudou, *Traffic signaling: New functions of huntingtin and axonal transport in neurological disease*, Aug. 2020. DOI: `10.1016/j.conb.2020.04.001`.

[23] A. S. Dickey and A. R. L. Spada, *Therapy development in huntington disease: From current strategies to emerging opportunities*, Apr. 2018. DOI: `10.1002/ajmg.a.38494`.

[24] J. Schulte and J. T. Littleton, "The biological function of the huntingtin protein and its relevance to huntington's disease pathology," *Curr Trends Neurol*, vol. 5, pp. 65–78, 2011.

[25] E. Scherzinger, A. Sittler, K. Schweiger, *et al.*, *Self-assembly of polyglutamine-containing huntingtin fragments into amyloid-like fibrils: Implications for huntington's disease pathology (huntingtinglutamine repeataggregation)*, 1999. [Online]. Available: `www.pnas.org.`.

[26] A. Vignal, D. Milan, M. SanCristobal, and A. Eggen, *A review on snp and other types of molecular markers and their use in animal genetics*, 2002. DOI: `10.1051/gse:2002009`.

[27] G. Hettiarachchi and A. A. Komar, "Gwas to identify snps associated with common diseases and individual risk: Genome wide association studies (gwas) to identify snps associated with common diseases and individual risk," in Springer International Publishing, Aug. 2022, pp. 51–76, ISBN: 9783031056161. DOI: 10.1007/978-3-031-05616-1_4.

[28] H. Oettgen and D. H. Broide, "Introduction to mechanisms of allergic disease," in Elsevier, Nov. 2011, pp. 1–32, ISBN: 9780702045707. DOI: 10.1016/B978-0-7234-3658-4.00005-6.

[29] J. Broholm, J.-J. Cassiman, D. Craufurd, *et al.*, "Guidelines for the molecular genetics predictive test in huntington's disease. international huntington association (iha) and the world federation of neurology (wfn) research group on huntington's chorea.," *J Med Genet*, vol. 31, pp. 555–559, 7 Jul. 1994.

[30] D. R. Langbehn, R. R. Brinkman, D. Falush, J. S. Paulsen, and M. R. Hayden, "A new model for prediction of the age of onset and penetrance for huntington's disease based on cag length," *Clinical Genetics*, vol. 65, pp. 267–277, 4 Apr. 2004, ISSN: 00099163. DOI: 10.1111/j.1399-0004.2004.00241.x.

[31] D. R. Langbehn, M. R. Hayden, J. S. Paulsen, *et al.*, *Cag-repeat length and the age of onset in huntington disease (hd): A review and validation study of statistical approaches*, Mar. 2010. DOI: 10.1002/ajmg.b.30992.

[32] J. S. Paulsen, M. Hayden, J. C. Stout, *et al.*, "Preparing for preventive clinical trials the predict-hd study," *Arch Neurol*, vol. 63, pp. 883–890, 3 2006. DOI: https://doi.org/10.1001/archneur.63.6.883. [Online]. Available: http://archneur.jamanetwork.com/.

[33] J. S. Paulsen, D. R. Langbehn, J. C. Stout, *et al.*, "Detection of huntington's disease decades before diagnosis: The predict-hd study," *Journal of Neurology, Neurosurgery and Psychiatry*, vol. 79, pp. 874–880, 8 2008, ISSN: 1468330X. DOI: 10.1136/jnnp.2007.128728.

[34] S. Klöppel, C. Chu, G. C. Tan, *et al.*, *Automatic detection of preclinical neurodegeneration presymptomatic huntington disease*, 2009. [Online]. Available: http://www.cmmt.ubc.ca/clinical/hayden.

[35] J. A. Ciarochi, V. D. Calhoun, S. Lourens, *et al.*, "Patterns of co-occurring gray matter concentration loss across the huntington disease prodrome," *Frontiers in Neurology*, vol. 7, SEP Sep. 2016, ISSN: 16642295. DOI: 10.3389/fneur.2016.00147.

[36] D. Bzdok, M. Krzywinski, and N. Altman, *Points of significance: Machine learning: A primer*, Dec. 2017. DOI: 10.1038/nmeth.4526.

[37] Z. Ghahramani, *Probabilistic machine learning and artificial intelligence*, May 2015. DOI: 10.1038/nature14541.

[38] D. Bzdok, N. Altman, and M. Krzywinski, *Points of significance: Statistics versus machine learning*, Apr. 2018. DOI: 10.1038/nmeth.4642.

[39] P. P. Silva, J. D. Gaudillo, J. A. Vilela, *et al.*, "A machine learning-based snp-set analysis approach for identifying disease-associated susceptibility loci," *Scientific Reports*, vol. 12, 1 Dec. 2022, ISSN: 20452322. DOI: `10.1038/s41598-022-19708-1`.

[40] P. Harper, C. Lim, C. David, and P. Disease, "Ten years of presymptomatic testing for huntington's disease: The experience of the uk huntington's disease prediction consortium," *J Med Genet*, vol. 37, pp. 567–571, 8 Aug. 2000. DOI: `10.1136/jmg.37.8.567`.

[41] A. Drouin, G. Letarte, F. Raymond, M. Marchand, J. Corbeil, and F. Laviolette, "Interpretable genotype-to-phenotype classifiers with performance guarantees," *Scientific Reports*, vol. 9, 1 Dec. 2019, ISSN: 20452322. DOI: `10.1038/s41598-019-40561-2`.

[42] S. Szymczak, J. M. Biernacka, H. J. Cordell, *et al.*, "Machine learning in genome-wide association studies," vol. 33, 2009. DOI: `10.1002/gepi.20473`.

[43] A. Ziegler, A. L. DeStefano, and I. R. König, "Data mining, neural nets, trees - problems 2 and 3 of genetic analysis workshop 15," vol. 31, 2007. DOI: `10.1002/gepi.20280`.

[44] S. Sathe, J. Ware, J. Levey, *et al.*, *Enroll-hd: An integrated clinical research platform and worldwide observational study for huntington's disease*, Aug. 2021. DOI: `10.3389/fneur.2021.667420`.

[45] J. Ouwerkerk, S. Feleus, K. F. van der Zwaan, *et al.*, "Machine learning in huntington's disease: Exploring the enroll-hd dataset for prognosis and driving capability prediction," *Orphanet Journal of Rare Diseases*, vol. 18, 1 Dec. 2023, ISSN: 17501172. DOI: `10.1186/s13023-023-02785-4`.

[46] J. Ko, H. Furby, X. Ma, *et al.*, "Clustering and prediction of disease progression trajectories in huntington's disease: An analysis of enroll-hd data using a machine learning approach," *Frontiers in Neurology*, Jan. 2023. DOI: `10.3389/fneur.2022.1034269`.

[47] J. M. Lee, K. Correia, J. Loupe, *et al.*, "Cag repeat not polyglutamine length determines timing of huntington's disease onset," *Cell*, vol. 178, 887–900.e14, 4 Aug. 2019, ISSN: 10974172. DOI: `10.1016/j.cell.2019.06.036`.

[48] S. Carbon, A. Ireland, C. J. Mungall, S. Shu, B. Marshall, and S. Lewis, "Amigo: Online access to ontology and annotation data," *Bioinformatics*, vol. 25, pp. 288–289, 2 Jan. 2009, ISSN: 1367-4811. DOI: `10.1093/bioinformatics/btn615`.

[49] D. Szklarczyk, A. L. Gable, D. Lyon, *et al.*, "String v11: Protein-protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets.," *Nucleic acids research*, vol. 47, pp. D607–D613, D1 Jan. 2019, ISSN: 1362-4962. DOI: `10.1093/nar/gky1131`.

[50] S. Cherlin, R. A. Howey, and H. J. Cordell, "Using penalized regression to predict phenotype from snp data," vol. 12, BioMed Central Ltd., Sep. 2018. DOI: `10.1186/s12919-018-0149-2`.

[51] A. Meléndez, C. López, D. Bonet, *et al.*, "Assessing tree-based phenotype prediction on the uk biobank," 2023.

[52] P. Roback and J. Legler, *Beyond Multiple Linear Regression: Applied Generalized Linear Models and Multilevel Models in R*, 1st ed. Chapman and Hall/CRC, 2021.

[53] M. Rouaud, *Probability, Statistics and Estimation: Propagation of Uncertainties in Experimental Measurement.* 2013.

[54] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, 2nd ed. Springer, 2009.

[55] J. Friedman, T. Hastie, and R. Tibshirani, *Regularization paths for generalized linear models via coordinate descent*, 2010. [Online]. Available: `http://www.jstatsoft.org/`.

[56] M. Ghazwani and M. Y. Begum, "Computational intelligence modeling of hyoscine drug solubility and solvent density in supercritical processing: Gradient boosting, extra trees, and random forest models," *Scientific Reports*, vol. 13, 1 Dec. 2023, ISSN: 20452322. DOI: `10.1038/s41598-023-37232-8`.

[57] J. Elith, J. R. Leathwick, and T. Hastie, *A working guide to boosted regression trees*, Jul. 2008. DOI: `10.1111/j.1365-2656.2008.01390.x`.

[58] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning : with Applications in R*, 2nd ed. Springer, Jun. 2013.

[59] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," Mar. 2016. DOI: `10.1145/2939672.2939785`. [Online]. Available: `http://arxiv.org/abs/1603.02754%20http://dx.doi.org/10.1145/2939672.2939785`.

[60] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16, San Francisco, California, USA: ACM, 2016, pp. 785–794, ISBN: 978-1-4503-4232-2. DOI: `10.1145/2939672.2939785`. [Online]. Available: `http://doi.acm.org/10.1145/2939672.2939785`.

[61] F. Pedregosa, G. Varoquaux, A. Gramfort, *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[62] F. Pedregosa, G. Varoquaux, A. Gramfort, *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[63] K. A. Sap, K. W. Geijtenbeek, S. Schipper-Krom, A. T. Guler, and E. A. Reits, *Ubiquitin-modifying enzymes in huntington's disease*, Feb. 2023. DOI: `10.3389/fmolb.2023.1107323`.

[64] D. Moss, A. Pardiñas, D. Langbehn, *et al.*, "Identification of genetic variants associated with huntington's disease progression: A genome-wide association study," *The Lancet Neurology*, vol. 16, pp. 701–711, 9 2017. DOI: `10.1016/S1474-4422(17)30161-8`. [Online]. Available: `https://u-picardie.hal.science/hal-04073171`.

[65] B. McAllister, J. Donaldson, C. S. Binda, *et al.*, "Exome sequencing of individuals with huntington's disease implicates fan1 nuclease activity in slowing cag expansion and disease onset," *Nature Neuroscience*, vol. 25, pp. 446–457, 4 Apr. 2022, ISSN: 15461726. DOI: `10.1038/s41593-022-01033-5`.

[66] J. Cheng, H. P. Liu, W. Y. Lin, and F. J. Tsai, "Identification of contributing genes of huntington's disease by machine learning," *BMC Medical Genomics*, vol. 13, 1 Dec. 2020, ISSN: 17558794. DOI: 10.1186/s12920-020-00822-w.

[67] A. Benraiss, J. N. Mariani, A. Tate, *et al.*, "A tcf7l2-responsive suppression of both homeostatic and compensatory remyelination in huntington disease mice," *Cell Reports*, vol. 40, 9 Aug. 2022, ISSN: 22111247. DOI: 10.1016/j.celrep.2022.111291.

[68] W. J. Szlachcic, P. M. Switonski, W. J. Krzyzosiak, M. Figlerowicz, and M. Figiel, "Huntington disease ipscs show early molecular changes in intracellular signaling, the expression of oxidative stress proteins and the p53 pathway," *DMM Disease Models and Mechanisms*, vol. 8, pp. 1047–1057, 9 Sep. 2015, ISSN: 17548411. DOI: 10.1242/dmm.019406.

[69] S. V. Lobanov, B. McAllister, M. McDade-Kumar, *et al.*, "Huntington's disease age at motor onset is modified by the tandem hexamer repeat in tcerg1," *npj Genomic Medicine*, vol. 7, 1 Dec. 2022, ISSN: 20567944. DOI: 10.1038/s41525-022-00317-w.

[70] R. J. Fenster, "Cell-type specific translational profiling in huntington' s disease mouse models," 2011. [Online]. Available: http://digitalcommons.rockefeller.edu/.

[71] Amazon, *Aws pricing calculator*. [Online]. Available: https://calculator.aws/#/createCalculator/ec2-enhancement.

[72] Council of the European Union and European Parliament, *Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/ec (general data protection regulation).*

[73] Council of the European Union and European Parliament, *Regulation (eu) 2017/745 of the european parliament and of the council of 5 april 2017 on medical devices, amending directive 2001/83/ec, regulation (ec) no 178/2002 an.*

A Appendix A

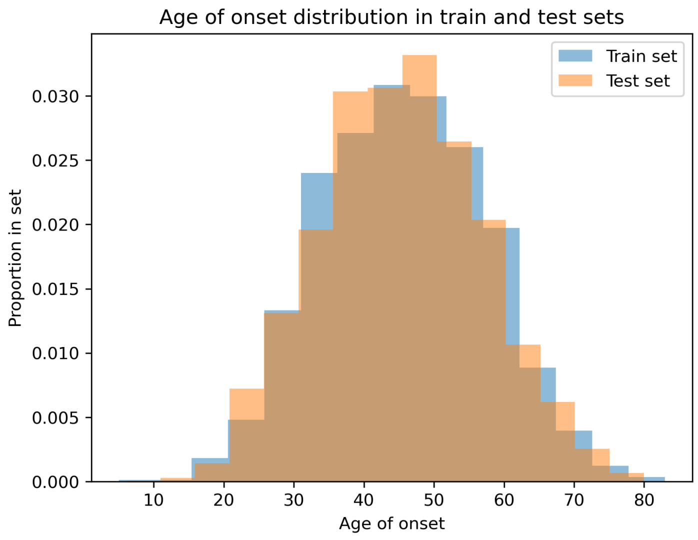


Figure A.1: Age distribution in training and testing sets.

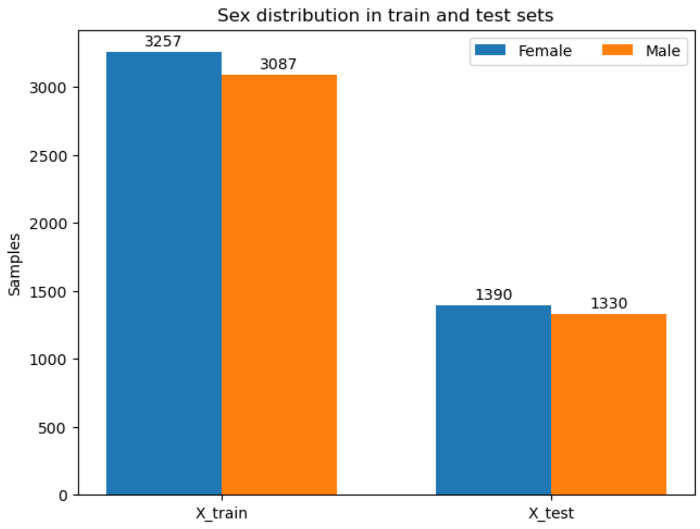


Figure A.2: Number of samples in training and testing sets separated by sex.

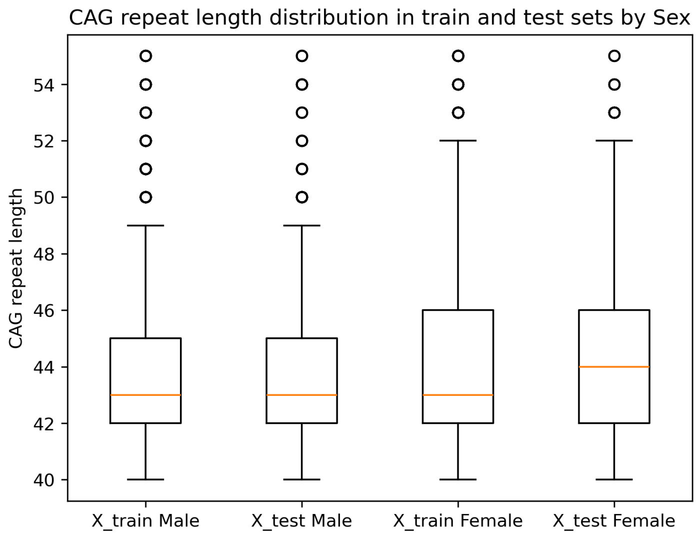
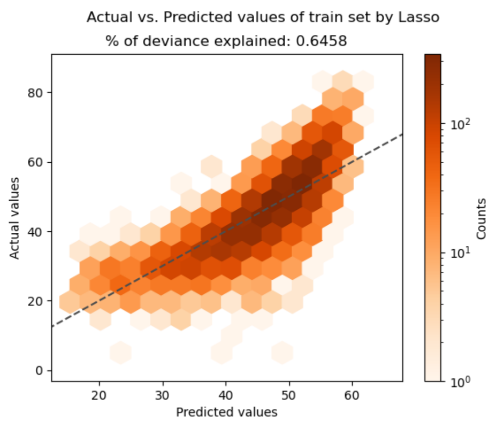


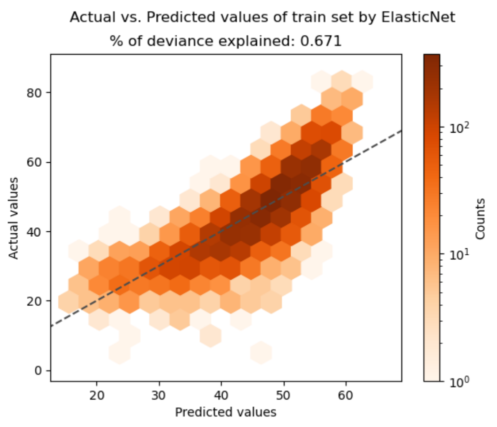
Figure A.3: CAG repeat length distribution in training and testing sets separated by sex.

B Appendix B

B.1 Training predictions

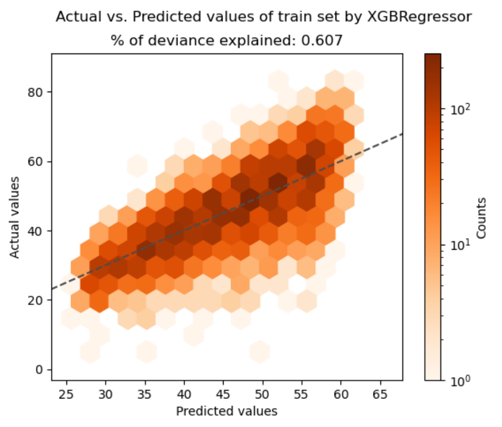


(a) Lasso model.

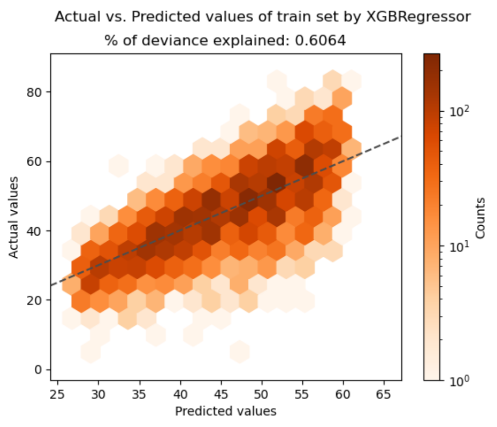


(b) Elastic Net model.

Figure B.1: Least squares methods predictions over training set.



(a) Hist XGBoost model.



(b) Approx XGBoost model.

Figure B.2: XGBoost predictions over training set.

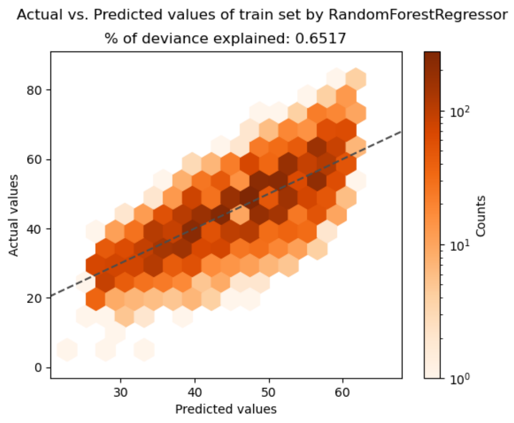


Figure B.3: Random Forest predictions over training set.

B.2 Testing predictions

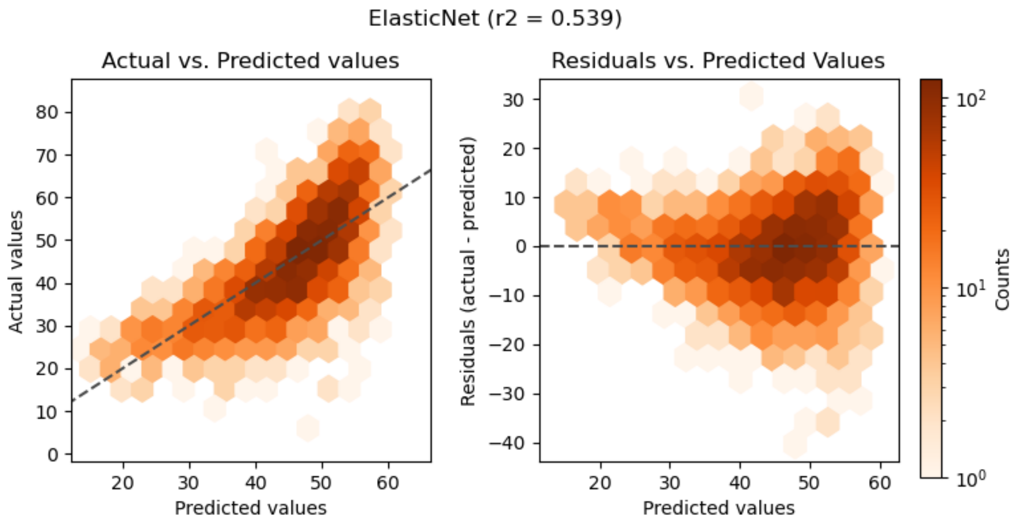


Figure B.4: Elastic Net predictions of testing set.

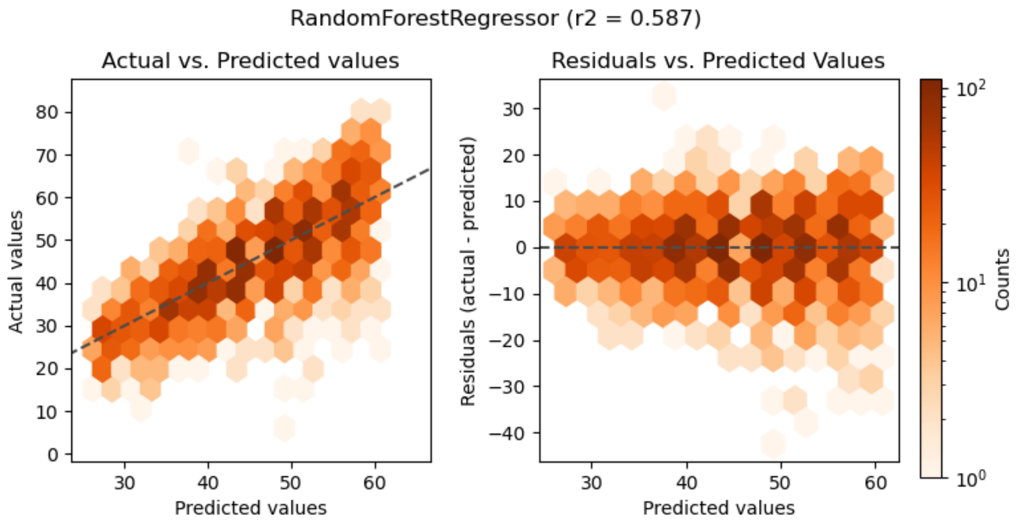


Figure B.5: Random Forest predictions of testing set.

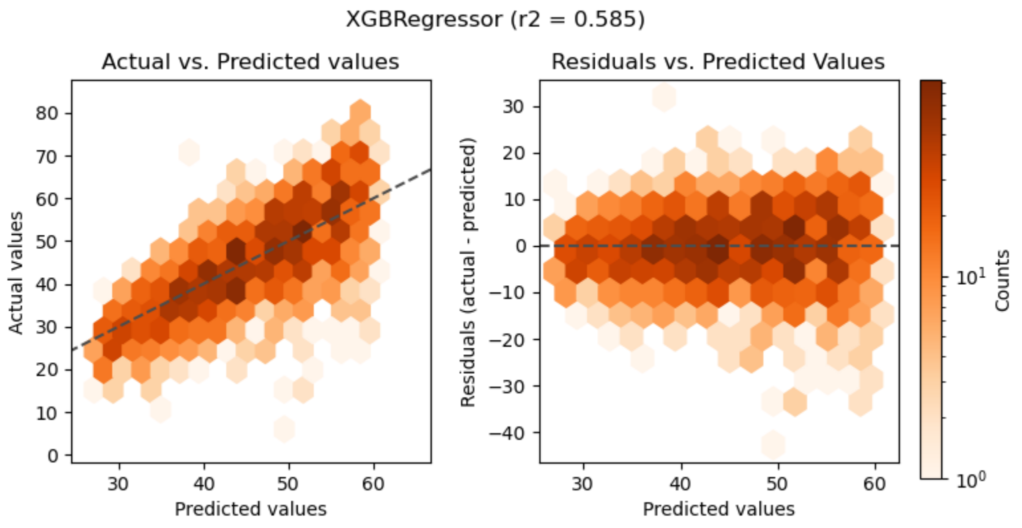


Figure B.6: Approx XGBoost predictions of testing set.

C Appendix C

C.1 Manhattan Plots

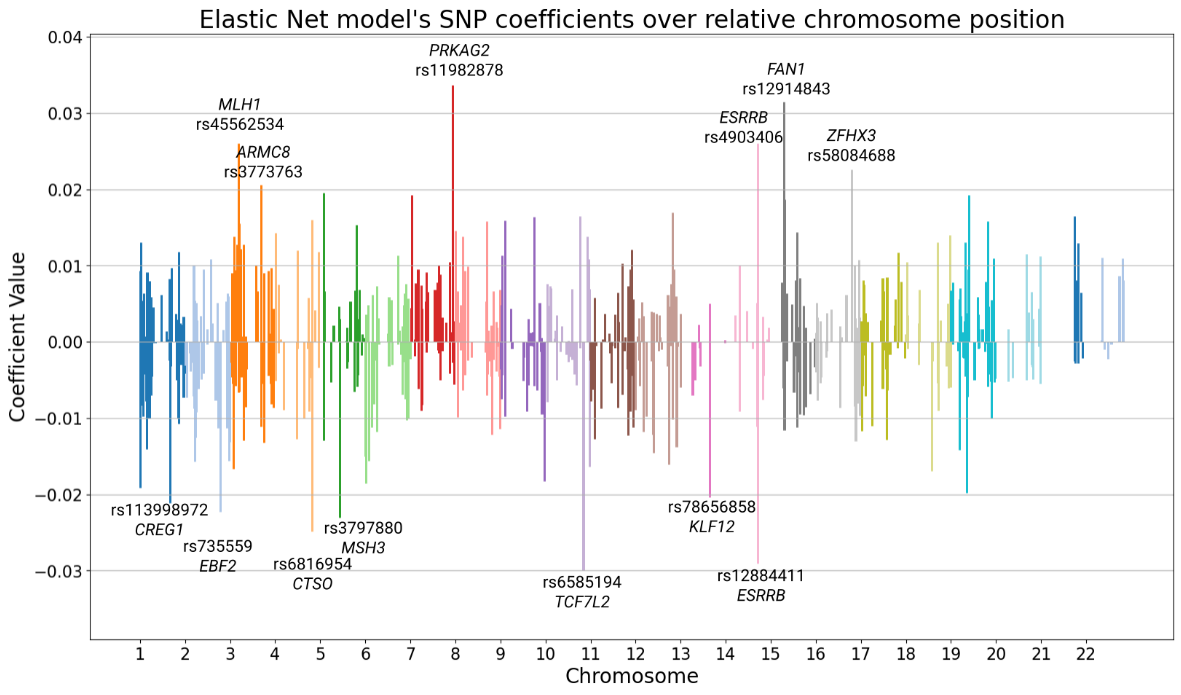


Figure C.1: Manhattan Plot of Elastic Net coefficients.

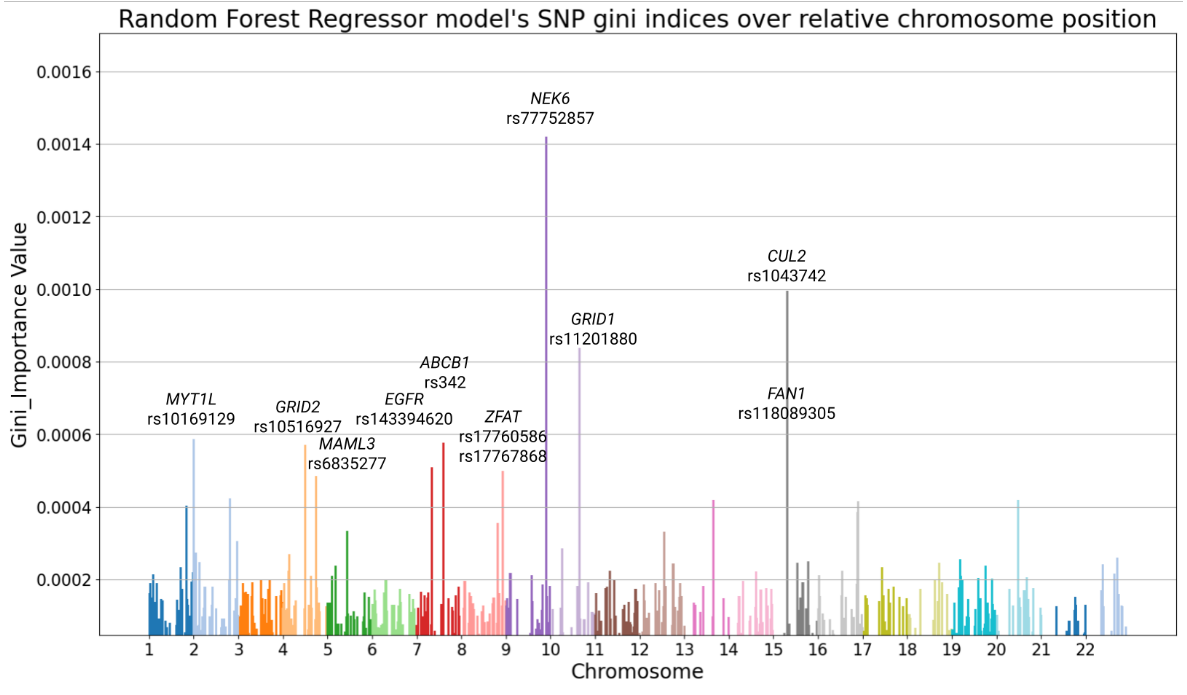


Figure C.2: Manhattan Plot of Random Forest.

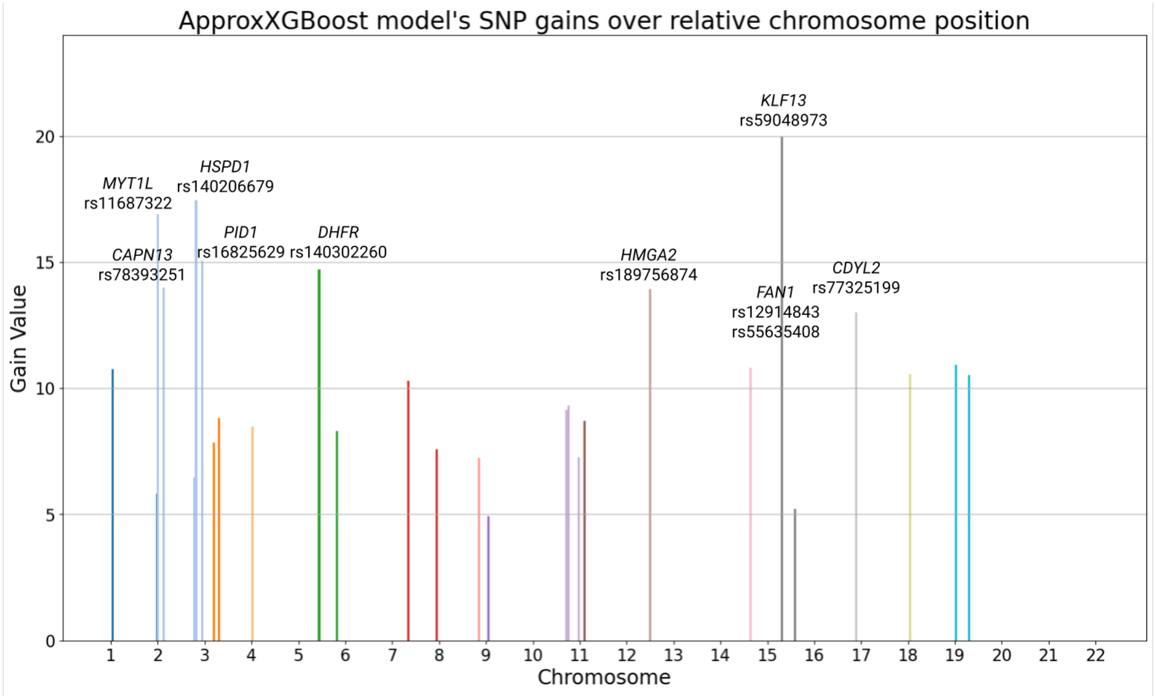


Figure C.3: Manhattan Plot of Approx XGBoost.

C.2 Feature importance tables

SNP	Coefficient	Gene	GO
rs17173770	0.03656065	SMARCD3	GO:0140110
rs61997076	0.033996664	FAN1	extra_genes
rs144287831	0.029233377	MLH1	GO:0006298
rs10885398	-0.028514741	TCF7L2	GO:0140110
rs8017707	-0.026800904	ESRRB	GO:0140110
rs1543108	-0.025090193	CTSO	GO:0004197
rs2278687	0.023999402	ESRRB	GO:0140110
rs1650649	-0.022265296	MSH3	GO:0006298
rs5743063	-0.021884425	PMS1	GO:0006298
rs7500197	0.021786358	GRID2	GO:0035249
rs2025533	-0.020597318	KLF12	GO:0140110
rs11293	0.01907387	ABCA1	GO:0042157
rs12531177	0.018855557	PMS2	GO:0006298
rs35782477	0.018614033	ANAPC13	GO:0043161
rs2262948	0.017807763	ZNF675	GO:0140110
rs2500286	-0.017432597	PRDM16	GO:0140110
rs9653966	-0.017269952	XPC	GO:0006298
rs2875957	-0.017144997	CDYL	GO:0140110
rs9312893	0.016999085	FBXL7	GO:0043161
rs17302697	0.016957078	NR3C1	GO:0035249
rs6856354	0.016785666	CTSO	GO:0004197
rs3740405	0.015683452	FHIT	GO:0031625
rs12422904	0.01560022	FOXN4	GO:0140110
rs201804677	-0.01556026	ZNF431	GO:0140110
rs2972071	-0.015510717	RTN4	GO:0031625
rs7316068	-0.015349336	SLC17A8	GO:0035249
rs3132292	-0.015137427	RXRA	GO:0140110
rs6987733	0.015124317	UBR5	GO:0043130
rs3512	0.014648994	AGBL4	GO:0098930
rs274883	0.014632466	LIG1	GO:0006298
rs13208655	-0.014536453	JARID2	GO:0043130

Table 7: Lasso top 30 SNPs by feature importance.

SNP	Coefficient	Gene	GO
rs17173770	0.03365854	SMARCD3	GO:0140110
rs61997076	0.03142683	FAN1	extra_genes
rs10885398	-0.029996023	TCF7L2	GO:0140110
rs8017707	-0.02911162	ESRRB	GO:0140110
rs144287831	0.026012165	MLH1	GO:0006298
rs2278687	0.025964953	ESRRB	GO:0140110
rs1543108	-0.024854636	CTSO	GO:0004197
rs1650649	-0.023066632	MSH3	GO:0006298
rs7500197	0.022596601	GRID2	GO:0035249
rs5743063	-0.022342289	PMS1	GO:0006298
rs1917533	-0.021176575	ACTN1	GO:0140110
rs35782477	0.020563638	ANAPC13	GO:0043161
rs2025533	-0.020403964	KLF12	GO:0140110
rs201804677	-0.019823277	ZNF431	GO:0140110
rs9312893	0.019517148	FBXL7	GO:0043161
rs2262948	0.019241005	ZNF675	GO:0140110
rs12531177	0.019214965	PMS2	GO:0006298
rs2500286	-0.019203568	PRDM16	GO:0140110
rs11293	0.018643003	ABCA1	GO:0042157
rs2875957	-0.018634377	CDYL	GO:0140110
rs3132292	-0.01829819	RXRA	GO:0140110
rs3512	0.017774522	AGBL4	GO:0098930
rs12422904	0.016952137	FOXN4	GO:0140110
rs199779315	-0.016934201	ZBTB7C	GO:0140110
rs9653966	-0.016681628	XPC	GO:0006298
rs17227161	0.016438538	RUNX1	GO:0140110
rs2118638	-0.016436614	TCERG1L	GO:0140110
rs3740405	0.01640511	FHIT	GO:0031625
rs2777803	0.016330175	ABCA1	GO:0042157
rs7316068	-0.01607368	SLC17A8	GO:0035249
rs6856354	0.016003186	CTSO	GO:0004197

Table 8: Elastic Net top 30 SNPs by feature importance.

SNP	**Gini Importance**	**Gene**	**GO**
rs77752857	0.001420583	NEK6	GO:0031625
rs1043742	0.000995081	CUL2	GO:0043161
rs11201880	0.000837296	GRID1	GO:0035249
rs10169129	0.000586326	MYT1L	GO:0140110
rs342	0.000577957	ABCB1	GO:0031625
rs10516927	0.000570458	GRID2	GO:0035249
rs143394620	0.000508776	EGFR	GO:0031625
rs118089305	0.000503646	FAN1	extra_genes
rs17767868	0.000498691	ZFAT	GO:0140110
rs17760586	0.000495167	ZFAT	GO:0140110
rs6835277	0.000484814	MAML3	GO:0140110
rs11293	0.000446546	ABCA1	GO:0042157
rs56141370	0.000431173	NEK6	GO:0031625
rs185978838	0.000424241	PGAP1	GO:0042157
rs7986341	0.000419989	KLF12	GO:0140110
rs62206880	0.000419746	ASXL1	GO:0140110
rs146175915	0.000415588	CDYL2	GO:0140110
rs138114240	0.000404046	HHAT	GO:0042157
rs3512	0.000400071	AGBL4	GO:0098930
rs4297916	0.00039359	MYT1L	GO:0140110
rs149682363	0.000386536	MAF	GO:0140110
rs191435283	0.000355128	EXT1	GO:0035249
rs111434006	0.000333013	MSH3	GO:0006298
rs6582088	0.000332103	TRHDE	GO:0008242
rs181825	0.000320682	ADARB1	GO:0051402
rs903372	0.000305975	LRRFIP1	GO:0140110
rs147241096	0.000285983	CUL2	GO:0043161
rs17703967	0.000284305	ASB4	GO:0031625
rs141939884	0.000273324	LPIN1	GO:0140110
rs111506732	0.000272638	TRHDE	GO:0008242
rs60026188	0.000270552	MYT1L	GO:0140110

Table 9: Random Forest top 30 SNPs by feature importance.

SNP	**Gain**	**Gene**	**GO**
rs3512	19.99822235107422	AGBL4	GO:0098930
rs115282897	17.450469970703125	PGAP1	GO:0042157
rs10169129	16.90975570678711	MYT1L	GO:0140110
rs2160644	15.064666748046875	PID1	GO:0006112
rs245100	14.723200798034668	DHFR	GO:0046655
rs62142184	13.985157012939453	CAPN13	GO:0004197
rs7487625	13.919387817382812	HMGA2	GO:0140110
rs61997076	13.325885772705078	FAN1	extra_genes
rs57013064	13.008210182189941	CDYL2	GO:0140110
rs118089305	12.423885345458984	FAN1	extra_genes
rs11678557	11.876806259155273	PID1	GO:0006112
rs116530043	10.932844161987305	GRIN3B	GO:0035249
rs113363660	10.814685821533203	ACTN1	GO:0140110
rs112574961	10.742321968078613	CASZ1	GO:0140110
rs607549	10.549150466918945	CIT	GO:0051402
rs8100660	10.517244338989258	UNC13A	GO:0035249
rs11487218	10.287309646606445	EGFR	GO:0031625
rs72841819	9.312403678894043	BTRC	GO:0043161
rs4083635	9.14719009399414	PDLIM1	GO:0140110
rs1908771	8.820863723754883	AEBP2	GO:0140110
rs151067506	8.694870948791504	TEAD1	GO:0140110
rs141338757	8.484679222106934	HTT	extra_genes
rs17720191	8.309086799621582	FBXO38	GO:0043161
rs1860968	7.850630760192871	DTNBP1	GO:0098930
rs111434006	7.695502281188965	MSH3	GO:0006298
rs2536065	7.573413848876953	PRKAG2	GO:0006112
rs893507	7.270939826965332	TCERG1L	GO:0140110
rs77047816	7.227169036865234	ZHX2	GO:0140110
rs5743061	6.460529327392578	PMS1	GO:0006298
rs146339947	5.831992149353027	ZNF670	GO:0140110
rs999449	5.206699848175049	RORA	GO:0140110
rs12380722	4.913415908813477	KDM4C	GO:0140110
rs149622839	4.911685943603516	DHFR	GO:0046655

Table 10: Approx XGBoost SNPs by feature importance.

SNP	Gain	Gene	GO
rs10169129	21.090667724609375	MYT1L	GO:0140110
rs61997076	17.790246963500977	FAN1	extra_genes
rs141338757	15.449226379394531	HTT	extra_genes
rs2393777	14.811421394348145	HNF1A	GO:0140110
rs245105	14.218707084655762	DHFR	GO:0046655
rs2316153	11.806947708129883	CDYL2	GO:0140110
rs144287831	11.36458683013916	MLH1	GO:0006298
rs79727797	11.205156326293945	TCERG1	GO:0140110
rs139931071	10.507678985595703	STAM	GO:0051402
rs2293317	10.322250366210938	FAN1	extra_genes
rs149561857	10.169393539428711	MAPT	GO:0098930
rs57013064	10.138040542602539	CDYL2	GO:0140110
rs72841819	10.081390380859375	BTRC	GO:0043161
rs10779614	9.906745910644531	PTPN14	GO:0140110
rs274883	9.713150024414062	LIG1	GO:0006298
rs1543108	9.711298942565918	CTSO	GO:0004197
rs45506797	9.655911445617676	PAX3	GO:0140110
rs2949568	9.652124404907227	FAN1	extra_genes
rs1476130	9.63764762878418	ESRRG	GO:0140110
rs6856354	9.573089599609375	CTSO	GO:0004197
rs118089305	9.558370590209961	FAN1	extra_genes
rs117606009	9.500611305236816	HOXB7	GO:0140110
rs34837214	9.182149887084961	UBR5	GO:0043130
rs6875184	8.844964027404785	DHFR	GO:0046655
rs6875489	8.843254089355469	RHOBTB3	GO:0043161
rs2074927	8.828694343566895	CLC	GO:0004197
rs3789821	8.79568099975586	SMARCD3	GO:0140110
rs34065247	8.734062194824219	ESRRB	GO:0140110
rs4053615	8.51319694519043	BACH2	GO:0140110
rs113189219	7.644144058227539	PLAGL1	GO:0140110
rs4908624	7.346027374267578	CAMTA1	GO:0140110
rs5743061	6.857511520385742	PMS1	GO:0006298

Table 11: Hist XGBoost SNPs by feature importance.

D Appendix D

GO	Lasso	Elastic Net	Hist XGBoost	Approx XGBoost	Random Forest
GO:0140110	1.1687	1.1263	0.5569	0.4654	1.1143
GO:0043161	0.7598	0.8009	0.7326	0.709	0.9038
GO:0031625	1.362	1.3084	0.0	0.4995	1.2984
GO:0035249	0.7434	0.7764	0.0	1.1388	0.7799
GO:0098930	0.7487	0.7556	0.6506	1.3011	0.7216
GO:0051402	0.832	0.8526	0.6995	0.6776	0.8767
GO:0042157	0.9395	1.0129	0.0	0.9123	0.9367
GO:0004197	0.8981	1.0214	3.4052	1.0287	0.6375
GO:0006112	0.68	0.7841	0.0	5.2856	0.9762
GO:0043130	0.2872	0.4826	1.9962	0.0	1.143
GO:0006298	2.2613	1.5241	8.5739	5.3471	1.1612
GO:0016579	0.3036	0.3014	0.0	0.0	0.5311
GO:0008242	1.6281	2.1865	0.0	0.0	2.6887
GO:0046655	1.3075	1.4612	18.7844	18.1784	0.7319
extra_genes	3.1109	3.3109	70.6949	38.1752	2.3217

Table 12: GO enrichment analysis results: odds ratio for each comparison against the background counts obtained with a Fisher test.

GO	Lasso	Elastic Net	Hist XGBoost	Approx XGBoost	Random Forest
GO:0140110	0.0272	0.0493	0.1077	0.0336	0.0733
GO:0043161	0.0486	0.0558	1.0	1.0	0.3942
GO:0031625	0.017	0.0177	0.2601	0.721	0.0242
GO:0035249	0.0958	0.0895	0.4162	0.6966	0.0893
GO:0098930	0.1268	0.0828	1.0	0.6687	0.0437
GO:0051402	0.3611	0.3515	1.0	1.0	0.43
GO:0042157	0.849	0.9345	0.6269	1.0	0.8048
GO:0004197	0.6876	0.8621	0.0675	1.0	0.036
GO:0006112	0.2543	0.3829	1.0	0.0231	1.0
GO:0043130	0.0039	0.0245	0.4013	1.0	0.5547
GO:0006298	0.0005	0.0582	0.0065	0.0589	0.4977
GO:0016579	0.0229	0.0074	1.0	1.0	0.0943
GO:0008242	0.2437	0.0412	1.0	1.0	0.0035
GO:0046655	0.5561	0.3158	0.0058	0.0062	0.8042
extra_genes	0.0087	0.0012	0.0	0.0001	0.035

Table 13: GO enrichment analysis results: p-values for each comparison against the background counts obtained with a Fisher test.

GO	Background	Lasso	Elastic Net	Hist XGBoost	Approx XGBoost	Random Forest
GO:0140110	198066	537	711	14	13	705
GO:0043161	28349	56	79	2	2	88
GO:0031625	20013	68	88	0	1	87
GO:0035249	18223	35	49	0	2	49
GO:0098930	16057	31	42	1	2	40
GO:0051402	14984	32	44	1	1	45
GO:0042157	11257	27	39	0	1	36
GO:0004197	10021	23	35	3	1	22
GO:0006112	6311	11	17	0	3	21
GO:0043130	5405	4	9	1	0	21
GO:0006298	4052	23	21	3	2	16
GO:0016579	3848	3	4	0	0	7
GO:0008242	1208	5	9	0	0	11
GO:0046655	1202	4	6	2	2	3
extra_genes	888	7	10	5	3	7

Table 14: Counts of GO terms amongst each model features, including the counts in the training feature matrix (background). Table used in the GO enrichment analysis.