NTNU
Kunnskap for en bedre verden

Department of Mathematical Sciences

TMA4500 - Project Thesis

Genomic prediction in wild populations with interpretable machine learning

Author:
Gard Westrum Gravdal

Date: 20.12.2024

Contents

List of Figures

Abstract

Advancement in genetic sequencing has enabled an expansion of genomic datasets from wild populations. At the same time, the development of flexible machine learning models has opened new possibilities for improving prediction tasks in genomic studies. For genomic prediction, the animal model is the traditional and state-of-the-art linear method, which is ideal in capturing the linear additive genetic effects of the phenotype. The development of new machine learning models in prediction, with the ability to capture both linear effects and non-linear interaction effects, motivates the search for alternative methods in genomic prediction that can outperform the traditional methods. In this thesis, we evaluate XGBoost, a flexible machine learning model with the ability to capture non-linear interaction effects, as an alternative to the traditional linear animal model for genomic prediction. Using a dataset of more than 180 000 single nucleotide polymorphisms (SNPs) from a meta-population of house sparrow in the Helgeland island system in northern Norway, we explore predictions of genetic contributions to the phenotypes of the traits body mass and wing length. In addition to prediction, this thesis investigates the use of Shapley additive explanation (SHAP) values as an alternative to genome-wide association studies (GWAS) for assessing SNP-trait associations. SHAP values originate from the Shapley values in coalition game theory and were adopted by interpretable machine learning as a way to give interpretability to "black box" machine learning models such as XGBoost. We will in this thesis investigate the method of using SHAP values to provide insight into the importance of SNPs in XGBoost predictions while addressing the limitations of traditional GWAS. We have in this thesis found that the Bayesian animal model predicts the genetic component of the phenotype more accurately than XGBoost. However, alternative configurations of the XGBoost may still have the potential to achieve better results. The results demonstrate that SHAP values represent a viable alternative to GWAS for assessing SNP-trait association.

1 Introduction

Quantitative traits are complex continuous traits, such as for example body mass or the length of an animal's leg, that usually show considerable variation within and among populations, and a key question in evolution is the nature of the underlying forces generating this variation (Walsh and Lynch, 2018). A better understanding of quantitative traits can drive progress in several scientific fields, and advancements in the mapping of quantitative traits has already helped us better understand evolution, human health and diseases (Huang, 2015; Marioni et al., 2018), conservation (Arenas et al., 2021) and improve breeding in animals and plants (Hickey et al., 2017). The variation in the quantitative traits is complex because the traits are influenced by both genetic effects, which result from the combined contributions of many loci, each with a small effect size, and non-genetic effects, such as environmental effects (Wood et al., 2014; McGaugh et al., 2021). One of the main goals in quantitative genetics is to partition the variation in the expression of a trait, otherwise known as the phenotypic variation, into its various components, where the component due to the linear additive genetic effects (V_A) are of particular interest to measure. The task of determining the phenotypic variation components is further complicated by the fact that the genetic component of this variation stem from several factors, such as pleiotropy or gene-gene interaction effects such as epistatic effects and dominance effects.

One of the traditional methods used to partition the variation in the phenotype is a linear model called the *animal model*, where the phenotype serves as the response variable (Henderson, 1988; Kruuk, 2004; Wilson et al., 2010; Bérénos et al., 2014). Many of the classical linear models, such as the linear regression model, assume independence between observations. Animal populations have structures that cause this assumption to be broken, where the most important structure to consider is the correlation between individuals due to relatedness. To account for relatedness, along with other hierarchical structures in the population, the animal model is a linear mixed model that estimates V_A. In the original animal model, pedigree data was used to quantify the relatedness between individuals, which lead to the pedigree-based animal model (Henderson, 1988). However, recent times have seen a decline in the cost of sequencing genetic markers, in particular the single nucleotide polymorphism (SNP), with growing availability of genetic material as a result (Misztal et al., 2020). The SNPs serve effectively as genetic markers for genomic regions due to linkage disequilibrium (LD) and since they are abundant in the genome, making the SNPs an alternative way to account for relatedness between individuals (Bush and Moore, 2012). Using SNPs instead of pedigrees to account for relatedness leads to the genomic animal model, which has become the preferred approach today (VanRaden, 2008; Ashraf et al., 2021).

Prediction of the genetic component of the phenotype, known as *genomic prediction*, is another main goal in quantitative genetics. The field of genomic prediction has seen a similar evolution from the use of traditional pedigree-based animal models, which led to the *best linear unbiased prediction* (BLUP, Henderson, 1975), to the genomic animal model, which introduced the *genomic best linear unbiased prediction* (GBLUP VanRaden, 2008), and even to methods that directly use genetic markers in marker-based regression (Meuwissen et al., 2001). Although genomic prediction is a widely used tool in animal and plant breeding (Meuwissen et al., 2001; Boubacar et al., 2013), it is not in widespread use for quantitative genetic studies of wild populations, though some studies have been made (Bérénos et al., 2014; Ashraf et al., 2021; Aspheim et al., 2024). Genomic data has become increasingly available also for wild populations (Aspheim et al., 2024). At the same time, there is a development of flexible prediction models from machine learning and neural networks that potentially could challenge the traditional methods in genomic prediction (Lourenço et al., 2024). XGBoost has become a popular model in prediction tasks because of its flexibility and accuracy (Chen and Guestrin, 2016). XGBoost uses regression trees as weak learners, which are ideal in capturing interactions between features in the model (Molnar, 2022). From a genomic perspective, this makes XGBoost ideal at capturing interactions between SNPs, such as epistatic effects and dominance effects (Johnsen et al., 2021). The ability to capture interaction effects in addition to the additive genetic effect gives the XGBoost potential to challenge the traditional methods in genomic prediction such as the animal model, which from the linearity assumption is ideal in capturing the linear additive genetic effects.

Though genomic prediction can be useful in many applications, it does not explicitly give inform-

ation about which parts of the genome are important to the expression of a trait. Predictions from machine learning models such as XGBoost are famously hard to interpret, in the sense that it is hard to understand how the model arrived at its prediction (Boehmke and Greenwell, 2019; Molnar, 2022). Assessing which parts of the genome that contribute to the phenotype is also of interest in many applications, and another main task in the studies of quantitative genetics is to assess which SNPs are associated with the phenotype. The conventional method of assessing SNP association is through methods in GWAS (Uffelmann et al., 2021). GWAS have become widely used and is a powerful tool for studying the genetic architecture of a trait (Murcray et al., 2009; Bush and Moore, 2012; Visscher et al., 2012; McKinney and Pajewski, 2012; Brodie et al., 2016; Yengo et al., 2022). However, a concern with GWAS is that much more of the genome than previously thought is required to be able to explain the heritability of complex polygenic traits (McKinney and Pajewski, 2012). Complex polygenic traits comprise a big number of alleles with very small effect sizes, making it hard for the SNPs to pass the strict significance tests of GWAS (Yang et al., 2010). This is a dilemma that has become known as "missing heritability", and several studies have shown that huge sample sizes are needed to overcome this issue (Eichler et al., 2010; Yang et al., 2010; Yengo et al., 2022). In GWAS, it is customary to use null-hypothesis significance testing with an arbitrary p-value cutoff, which as a general practice has become a highly controversial and debated topic in many scientific communities (Goodman, 2008; Head et al., 2015; Ioannidis, 2018; Muff, Nilsen et al., 2022). The issues with GWAS motivate the search for a new alternative method of assessing which SNPs are associated with the phenotype of interest.

Use of *Shapley additive explanation* (SHAP) values is a method of explaining and providing interpretability to predictions from machine learning models, and were developed from the concept of Shapley values in coalition game theory (Shapley, 1952; Lundberg et al., 2018). The SHAP values provide a measure of importance for the covariates used in the computation of a prediction (Molnar, 2022), and in a genomic context the covariates of a machine learning model are the SNPs. Using SHAP values in conjunction with genomic predictions from a machine learning model such as XGBoost thus represents an alternative to the traditional methods in GWAS in assessing the SNP association with the phenotype. The SHAP values have several desirable properties that are not found in GWAS' methods, in particular their ability to measure the importance of SNPs to predictions, as opposed to using p-values as in GWAS.

In this thesis, we investigate the XGBoost model as an alternative to the traditional genomic animal model for genomic prediction using a dataset of more than 180 000 SNP markers. Both models will predict the genetic contribution to the phenotypes of the two traits body mass and wing length in a dataset from the meta-population of wild house sparrows on the Helgeland island system in northern Norway. The phenotype undergoes pre-processing in order to remove the environmental effects on the phenotype before fitting the XGBoost model. The XGBoost model undergoes hyperparameter tuning using a Bayesian optimization framework and the *tree-structured Parzen estimator* (TPE) algorithm (Lundberg et al., 2018). The animal model is implemented in the Bayesian framework of *integrated nested Laplace approximation* (INLA Rue et al., 2009). As a measure of prediction accuracy, we use the Pearson correlation between the predictions and the mean phenotype. We also investigate using SHAP values to explain the predictions of the XGBoost model, offering an alternative to GWAS for assessing SNP association with the phenotypes. The mean |SHAP| values are computed as an approximation for global feature importance for the SNPs. In this thesis we use 10-fold CV for the predictions of the XGBoost model and the Bayesian animal model, and the resulting correlations are displayed and compared in a box-plot for both phenotypes. The results from GWAS and SHAP are displayed and compared in Manhattan-style plots. In addition, this thesis proposes a new method of visualizing mean |SHAP| values in a cumulative feature importance plot. All code used in this thesis is publicly available in the GitHub repository https://github.com/GardGravdal/Project-thesis.

2 Background

2.1 The foundations of quantitative genetics

2.1.1 Basic concepts and terminology in quantitative genetics

In most multicellular organisms, each individual inherits two sets of chromosomes, one from each parent, giving them two copies of every gene. A gene is located at a specific position on a specific chromosome, called a locus, and different versions of a gene are known as alleles (Hartl and Conner, 2004). When an individual has two different alleles, one may have a stronger influence on the outward appearance of a trait, dominating the other recessive allele. For example, an allele that codes for the brown eye (iris) color is typically dominant over the alleles that code for green or blue eye color, and green takes precedence over blue (White and Rabago-Smith, 2011). The dominant alleles are typically represented by an uppercase letter (*e.g.*, "A"), while recessive alleles are represented by lowercase letters (*e.g.*, "a") as introduced by Sutton (1903). Since an individual has two sets of chromosomes, for a given gene with two alleles, an individual can have one of three possible combinations of alleles: "AA", "Aa", or "aa", corresponding to dominant homozygous, heterozygous, and recessive homozygous combinations respectively. The specific combination of alleles that an individual carries is referred to as their genotype (Hartl and Conner, 2004).

2.1.2 Partition of phenotypic variance

An individual's outward appearance, known as the phenotype (P), results from the combined effects of the genetic components (G) and environmental factors (E), which mathematically can be expressed as $P = G + E$. Most phenotypes are not determined by alleles at a single locus. Instead, many are continuously distributed and are referred to as quantitative traits, which are influenced by multiple loci and environmental factors. Measuring variation in quantitative traits can be complex because it involves several factors, such as the polygenic basis, pleiotropy, epistatic effects, dominance effects and environmental factors:

- *The polygenic basis:* Refers to a trait having a number of loci and alleles affecting it.
- *Pleiotropy:* The phenomenon of a single gene influencing multiple traits.
- *Dominance effects:* Result from interactions between alleles at the same locus.
- *Epistatic effects:* Result from interactions between genes at different loci.

Additionally, there are often interactions between genes and the environment, known as gene-environment (G×E) interactions, which further complicate the expression of these traits (Hartl and Conner, 2004).

The total phenotypic variation of a quantitative trait, denoted V_P, is often partitioned into two parts, namely the variation due to the genetic component (V_G) and the variation due to the environmental factors (V_E). The variation due to the genetic component can be further decomposed into the variation due to additive genetic effects (V_A), the variation due to epistatic effects (V_I) and the variation due to dominance effects (V_D), which we can express as $V_G = V_A + V_I + V_D$ (Hartl and Conner, 2004). Additive genetic effects are the effects on a phenotype that are accumulated by summing over the individual contributions from alleles for a polygenic trait. The *breeding value* is an individual-specific quantity that is defined as the additive effect of an individual's genotype on a trait expressed relative to the population mean phenotype (Wilson et al., 2010). The name stems from the fact that the breeding value is the effect of an individual's genes on the value of a given trait in its offspring (Hartl and Conner, 2004). From these definitions, we can further define the *heritability* as the proportion of the total phenotypic variance that is due to genetic causes, or in other words, the relative importance of genetic variance in determining phenotypic variance.

We define the broad sense heritability as

$$h_B^2 = \frac{V_G}{V_P} \ ,$$

while the narrow sense heritability (or simply heritability) is defined as (Hartl and Conner, 2004)

$$h^2 = \frac{V_A}{V_P} \ .$$

The narrow sense heritability is often preferred to the broad sense heritability since the additive genetic effect provides a measure of the genetic component that can be directly passed on additively from parents to offspring.

2.1.3 Single nucleotide polymorphism

The basic building-blocks in DNA are nucleotides. The bases in DNA are the nucleotides adenine (A), cytosine (C), guanine (G) and thymine (T). The strands of DNA are held together by pairs of these bases, where adenine pairs with thymine and guanine pairs with cytosine (Pu et al., 2018). A SNP (often pronounced "snip") is, as the name suggests, a difference in a single nucleotide (Brookes, 1999). Such mutations are normal and occur approximately every 1000 nucleotide (Sachidanandam et al., 2001). We classify the mutation as a SNP if a variant is found in at least 1% of the population (Karki et al., 2015). The terminology 'allele' is used also for SNPs, meaning a variant of a SNP. Most SNPs are biallelic, which means they come in two variants (Sachidanandam et al., 2001). From the biallelic nature of the SNPs, we also inherit the terms heterozygous and homozygous. For example, if a SNP comes in the form of the two nucleotides A and C, we can define the "genotypes" AA, CC and AC. The latter genotype would then be heterozygous, while the other two are homozygous. This terminology simplifies the encoding of the SNPs. The SNP data is typically encoded $\{0, 1, 2\}$, where 0, 1, 2 denote the number of minor alleles in the genotype. The minor allele of a SNP is the second most frequent allele of that SNP. Note that this encoding is consistent with the assumption of additive genetic effects, which is important for the statistical models which will be discussed later in the thesis. We typically describe the position of a SNP at two levels: which chromosome it is located in, and the SNPs base pair (bp) number describing where in the chromosome it is located.

Most SNPs do not have an effect on the phenotype as they fall in the non-coding regions of the DNA, but some SNPs can change gene function or the level of its expression (Brodie et al., 2016). Although most SNPs do not directly have an effect on the phenotype, they are widely used as genome-wide markers because they are highly abundant and they serve effectively as markers for genomic regions due to linkage disequilibrium (LD Bush and Moore, 2012). LD is the genetic correlation between two or more alleles from the fact that they are inherited together more often than would be expected by random chance (Hartl and Conner, 2004). For example, alleles in loci that are close to each other on the chromosome will tend to be inherited together, since they are less likely to be broken apart by recombination during meiosis. SNPs can thus be genetic markers for regions of DNA associated with the specific trait of interest. Such a DNA region is called a quantitative trait locus (QTL) (Hartl and Conner, 2004). SNPs are thus excellent markers for studying complex traits (Syvänen, 2001; Oraguzie et al., 2008).

2.1.4 Animal model

As outlined in Section 2.1.2, the variation in phenotype for a quantitative trait is complex, and is influenced by several factors such as environmental effects and genetic effects. One of the main goals in quantitative genetics is to understand how the variation in phenotype is distributed, and in particular to estimate V_A, the component of phenotypic variance that is due to the additive genetic effect. The *animal model* is a LMM which attempts to estimate the distribution of the variability in phenotype for the different components. A LMM attempts to handle different sources of variability in the data stemming from *fixed effects* and *random effects* in situations when the data is grouped in hierarchical structures (Chen and Chen, 2017). Mixed effects models enable us

to partition the variance. Some of the typical hierarchical structures that can arise in an animal model is through permanent environmental effects, relatedness or individual-specific effects such as age or sex (Wilson et al., 2010; Aspheim et al., 2024). The animal model is a LMM where the breeding value is treated as a random effect (Wilson et al., 2010). In the simplest case, when we model a single phenotype y_i on a single individual i, the animal model can be expressed as

$$y_i = \mu + a_i + \epsilon_i \ ,$$

where μ is the population mean for the trait (fixed effect), a_i is the breeding value (random effect), and ϵ_i is an (independent) residual term (random effect). In a LMM, we assume that each random effect originates from some distribution (often Gaussian) with zero mean and unknown variance that is of interest to measure. The variance of the additive genetic effect is defined as V_A. The variance of the residual term, V_e, covers the remaining variance in the phenotype. Assuming both the random effects follow a normal distribution and a sample of n individuals, the vectors $\boldsymbol{a}$ and $\boldsymbol{\epsilon}$ are each of dimension $n \times 1$, with $\boldsymbol{a} \sim N(0, \mathbf{G}V_A)$ and $\boldsymbol{\epsilon} \sim N(0, \mathbf{I}_n V_e)$. Note that $\mathbf{I}_n$ denotes the $n \times n$ identity matrix, such that $\boldsymbol{\epsilon}$ is consistent with the assumption of independence between residual errors. The matrix $\mathbf{G}V_A$ denotes the variance-covariance matrix of breeding values.

Traditionally, pedigrees were used to model the covariance between breeding values of different individuals. In this method, pedigree data is used to calculate the *relatedness matrix* $\mathbf{G}$ for individuals i and j, $G_{ij} = 2\kappa_{ij}$. Here κ_{ij} is the coefficient of coancestry, which is defined as the probability that an allele that is drawn at random from individual i is *identical by descent* (IBD) to one drawn at random from individual j (Wilson et al., 2010). IBD refers to alleles that are descended by DNA replication from a single allele present in an ancestor (Hartl and Conner, 2004). Then the relatedness G_{ij} can intuitively be thought of as the "expected proportion of relatedness", for example 0.5 for a parent and its offspring.

An alternative to the use of pedigree data is to model the covariance of individuals through a *genomic relationship matrix* $\mathbf{G}$ derived from genetic markers such as SNPs. These methods became more popular as genome sequencing became relatively inexpensive and SNP markers could cover the genome with high density (VanRaden, 2008). Several methods to derive a genomic relationship matrix were proposed by vanRaden, and here we will present what is denoted as method 1 (VanRaden, 2008). Let $\mathbf{M}^*$ be the $n \times p$ matrix of markers (*e.g.*, the SNP matrix) for individuals $i = 1, ...n$ and SNPs $j = 1, ..., p$ encoded as $\{-1, 0, 1\}$ instead of $\{0, 1, 2\}$. This encoding ensures that the diagonal of the matrix $\mathbf{M}^*(\mathbf{M}^*)^T$ count the number of heterozygotes for each individual, while the off-diagonal elements measures (#equal alleles - #different alleles) for each individual (*i.e.* a measure of similarity). Now we let the frequency of the minor allele be p_i, and we define $\mathbf{P} = \{P_{ij}\}_{ij} \in \mathbb{R}^{n \times p}$ for $P_{ij} = 2(p_i - 0.5)$. We can then define $\mathbf{C} := \mathbf{M}^* - \mathbf{P}$. This transformation by subtracting $\mathbf{P}$ has the effect of giving credit to more rare alleles compared to more common ones in calculating the genomic relationship matrix. Finally, we define the genomic relationship matrix as

$$\mathbf{G} = \frac{\mathbf{C}\mathbf{C}^T}{2\sum_{i=1}^{n} p_i(1 - p_i)} \ . \tag{1}$$

Scaling by the divisor $2\sum_{i=1}^{n} p_i(1 - p_i)$ makes it so that the behavior of $\mathbf{G}$ is analogous to $\mathbf{G}$ calculated by pedigrees. Using the genomic relationship matrix instead of the relatedness matrix leads to the so-called *genomic animal model*.

2.1.5 Genomic prediction

Another main goal of quantitative genetics is to estimate the breeding value, a field of study often called *genomic prediction* (Meuwissen et al., 2001). Estimating breeding values entails extracting the additive genetic effects from the phenotype. For example, an animal breeder is only interested in knowing the breeding value of a trait, and the environment or other components of the phenotype is "noise" that they want to remove to understand the genetic merit of a trait. Animal models can be used to estimate breeding values by employing a method called BLUP (Henderson, 1975).

In short, BLUP is a method that predicts an unknown random variable by a linear function on the observed data (response). It is an unbiased estimator of the random variable, meaning that the

expected value of the predicted value is equal to the expected value of the variable it is estimating. It is the best estimator in the sense that there are no other unbiased estimators that has smaller variance than the BLUP. In a typical pedigree based animal model, we can express the response $\boldsymbol{y}$ as

$$\boldsymbol{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{D}\boldsymbol{r} + \boldsymbol{\epsilon} \ ,$$

where the response $\boldsymbol{y} \in \mathbb{R}^n$ is the phenotype. Further, $\boldsymbol{\beta}$ denotes the vector of fixed effects, such as the population mean $\boldsymbol{\mu}$, with corresponding design matrix $\mathbf{X}$ of appropriate dimension. $\boldsymbol{r} \sim N(\mathbf{0}, \mathbf{V}_r)$ is the vector containing the random effects, such as the breeding value $\boldsymbol{a} \sim N(\mathbf{0}, \mathbf{G}V_A)$, with the corresponding design matrix $\mathbf{D}$ of appropriate dimension. Finally, $\boldsymbol{\epsilon} \sim N(0, \mathbf{I}_n V_e) \in \mathbb{R}^n$ is the vector of residual errors. Assuming we have observed the response $\boldsymbol{y}$, the BLUP of $\boldsymbol{r}$, denoted $\hat{\boldsymbol{r}}$, is given as

$$\hat{\boldsymbol{r}} = E\left[\boldsymbol{r}|\boldsymbol{y}\right] = \left(\mathbf{D}^T\mathbf{V}_r^{-1}\mathbf{D}\right)^{-1}\mathbf{D}^T\mathbf{V}_r^{-1}\boldsymbol{y} \ . \tag{2}$$

The derivation of eqution (2) is out of the scope of this thesis, but we refer curious readers to Henderson (1975). If instead a genomic animal model is used, we simply replace the relatedness matrix $\mathbf{G}$ with a genomic relationship matrix such as that of method 1 given by vanRaden in equation (1). If a genomic relationship matrix is used, we call the estimator $\hat{\boldsymbol{r}}$ the GBLUP.

An alternative to the (G)BLUP methods of estimating the breeding value is by regressing the phenotype $\boldsymbol{y}$ directly on all the markers, which leads to the method of marker-based regression (Aspheim et al., 2024). Now the breeding value a_i is replaced by a sum over all the additive effects of the genome-wide markers, which as we recall is our definition of the breeding value. This model is consistent with the observation that many quantitative traits are polygenic, with small contributions from a large number of loci (Wood et al., 2014). Assuming we have a dataset of p SNPs and n individuals, we can model the phenotype $\boldsymbol{y}$ in marker-based regression as

$$\boldsymbol{y} = \mu\mathbf{1}_n + \boldsymbol{\Gamma}\boldsymbol{b} + \mathbf{Z}\boldsymbol{u} + \mathbf{W}\boldsymbol{d} + \boldsymbol{\epsilon} \ , \tag{3}$$

where μ is the population mean, $\mathbf{1}_n \in \mathbb{R}^n$ denotes the vector of ones, and $\boldsymbol{\Gamma}$ is matrix containing fixed effects with the corresponding regression parameter vector $\boldsymbol{b}$. Moreover, $\mathbf{Z} \in \mathbb{R}^{n \times p}$ is the matrix of marker codes (SNPs), and where the matrix has been mean-centered such that each column (SNP) sum to one. The vector $\boldsymbol{u} \sim N(\mathbf{0}, \mathbf{I}V_u)$ is a random effect for the allele substitution effects of each SNP. Further, $\boldsymbol{d}$ is the vector of random environmental effects with the corresponding design matrix $\mathbf{W}$. Finally, $\boldsymbol{\epsilon} \sim N(\mathbf{0}, \mathbf{I}V_e)$ is the vector of residuals. The vector of breeding values is in this case given by $\boldsymbol{a} = \mathbf{Z}\boldsymbol{u}$ (Aspheim et al., 2024).

It can be shown that the estimated breeding values by equation (3) is identical to the breeding values estimated by GBLUP as long as the same SNPs are used and the effect sizes $\boldsymbol{u}$ all stem from the same distribution (VanRaden, 2008). One issue of the marker-based regression method is that since $p > n$, the linear system is underdetermined, and estimating by standard least squares regression is not possible.

2.2 Statistical learning

2.2.1 Machine learning

Machine learning can at a low level be described as a set of methods that computers use to make and improve predictions or behaviors based on data (Molnar, 2022). At its core, machine learning is about developing algorithms that can infer mappings from inputs to outputs by optimizing a function based on a dataset. The naming of *learning* was coined by Arthur Samuel, and describes the process of using machine learning algorithms to build models from data (Zhou, 2021). We may also use the term *training* interchangeably. The set of data used in this phase is often called the training data, and the set of all the training samples is called the training set. It is customary to set aside an independent set for testing the model, called the testing set. It is worth noting that the performance of a machine learning model is generally speaking very dependent on the sample size: the bigger the sample size, the more data we have for training the model, leading to a better model fit.

In this thesis we will use methods based on *supervised learning*, which is a subcategory of machine learning. In supervised learning, the goal is to learn a predictive model that maps variables, often called covariates or features, to an output, often called the response. The response works as a label to guide the learning of a predictive model, thus we say that the learning is supervised (Nasteski, 2017). Mathematically, the task of supervised learning can be phrased as follows: Given dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, where $\boldsymbol{x}_i \in \mathcal{X}$ represents the input features and $y_i \in \mathcal{Y}$ represents the corresponding response, the goal is to find a function $f : \mathcal{X} \mapsto \mathcal{Y}$ that minimizes some predefined loss function $L\left(y, f(\mathbf{x}; \boldsymbol{\theta})\right)$, $L : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}$. The function f aims to model the connection between $\mathbf{x}$ and y and contains a set of flexible parameters $\boldsymbol{\theta}$ that are found by minimizing the loss function. Supervised learning can be formulated as the following optimization problem

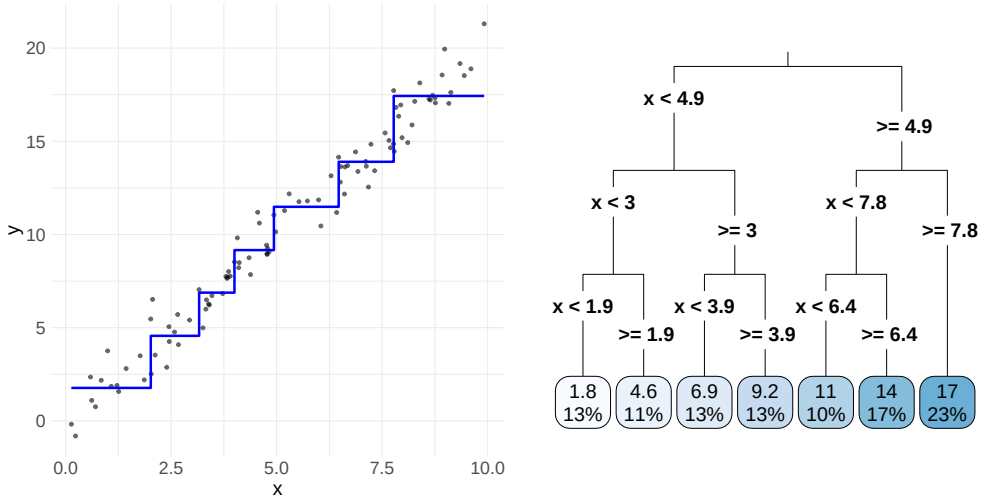
$$\hat{y} = f(\mathbf{x}; \hat{\boldsymbol{\theta}}), \quad \hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} L\left(y, f(\mathbf{x}, \boldsymbol{\theta})\right) \ . \tag{4}$$

The field of machine learning encompasses many different models which differ in many ways, such as the choice of loss function L and choice of function f. The aim of the loss function is to provide a measure of how well or poorly a model $y^* = f(\mathbf{x}, \boldsymbol{\theta}^*)$ is performing by assigning a penalty (or "loss"). This will allow the model to adjust its parameters in order to minimize the loss, which is designed to minimize the test error and thus improve the performance on unseen data, *i.e.* the data that is not in the training set. Some loss functions contain *regularization* parameters that aim to prevent overfitting by "punishing" more complex or flexible models in order to generalize the model. There exists many different choices of f, all with their own strengths and weaknesses. Some functions may be better at capturing linear relationships, such as the regular linear regression model, while others may be better at capturing non-linear effects or interactions between features, such as regression trees. The best choice of function f and loss function L is dependent on the dataset $\mathcal{D}$ and the task at hand. A model that performs well on one task might perform poorly on another. The fact that there exists no optimal machine learning model that can perform well in any given task or situation is reflected in the "No free lunch" theorem (Zhou, 2021). The conclusion to take from the theorem is that we need to understand the data we are working with and the task at hand before we can decide which machine learning model is the best choice.

2.2.2 Regression trees

Tree-based methods were developed from a simple and intuitive concept. The tree-based methods partition the feature space into rectangles called regions R, and then fit a simple constant model c, which in regression tasks is typically the mean of each sample in the region, in that region. An illustration of a regression tree is seen in Figure 1a. To model a tree-based method, we need a rule on how to split the feature space and create the different regions. Most tree-based methods use a process called recursive binary splitting (Hastie et al., 2009). This is a greedy approach that chooses which feature and split-point that "best fits" the response y in each step, where the best fit is decided by the chosen loss-function. In regression, the loss-function aims to minimize the variance in the response y (Molnar, 2022). The splitting into regions continues until some stopping criterion is met. The greedy approach does not guarantee globally optimal solution, but it can provide a good approximate model.

The terminology of "tree" stems from the fact that each tree can be visualized by a figure called the *binary decision tree*. These figures provide an intuition for the tree model. A binary decision tree should be read from the top down, such that we envision the "root" of the tree at the top. Each node in the tree represent a split in the feature space, and the leaves at the end of the tree are called leaf nodes or terminal nodes. Each leaf node represent a region in the feature space, and each sample falls into exactly one leaf node (Molnar, 2022). Figure 1b illustrates the decision tree corresponding to the regression tree in Figure 1a. In this thesis we will be working with a continuous response and the task is regression. We can model the response of a regression tree model through the p variables $x_{i1}, x_{i2}, \ldots, x_{ip}$ and the responses y_i, $i = 1, ...n$, where n is the sample size. Suppose we have partitioned into regions $R_1, R_2, ..., R_J$ each with constants c_j,



(a) Regression tree

(b) Decision tree

Figure 1: A simple illustration of a regression tree with corresponding decision tree. Data is 100 samples drawn from $\boldsymbol{y} \sim 2\boldsymbol{x} + \boldsymbol{\epsilon}$, where $x_i \sim \text{unif}(0, 10),\ \epsilon \sim N(0, 1),\ \forall i = 1, ..., 100$.

$j = 1, 2, ..., J$. We can then model the response as

$$f(\boldsymbol{x}_i) = \sum_{j=1}^{J} c_j I(\boldsymbol{x}_i \in R_j) \ ,$$

where $I(\cdot)$ denotes the indicator function (Hastie et al., 2009).

As with any supervised learning method, the bias-variance trade off needs to be considered when fitting a regression tree model. Regression trees are very prone to overfitting when the tree grows too large. So far we have not discussed any stopping criteria for the regression trees. In the next section we will discuss boosted regression trees which uses regression trees as weak learners. There we will discuss some of the methods used to avoid overfitting. One advantage of the regression tree model is that they are ideal in capturing the interactions between features in the model. On the other hand, trees fail to deal with linear relationships (Molnar, 2022). In the regression tree model, a linear relationship between an input feature and the response is approximated by splits, leading to an inefficient "step function" in the feature space, something that is clearly visible in Figure 1a, where a linear regression model likely would be a better choice. Regression trees also have inherent lack of smoothness, such that slight changes in input feature can lead to big impact on the predicted outcome.

2.2.3 Boosted regression trees

Boosting is a supervised learning technique that involves sequentially building simple models, often referred to as weak learners, with each new weak learner "learning" from the previous one. The final model is a weighted sum of all the weak learners, which together fits a powerful ensemble model (Hastie et al., 2009). We begin by introducing a simple illustrative boosting scheme. Assume we have the dataset $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, where y is the response and $\boldsymbol{x}$ the covariates. We start with a simple model, often simply the mean of the response $F_0(\boldsymbol{x}) = \frac{1}{n}\sum_{i=1}^n y_i = \bar{y}$. We fit the next model $f_1(\boldsymbol{x})$ on the residuals of the previous model $y - F_0(\boldsymbol{x})$. However, in doing so, we run the risk of overfitting the data, especially as we keep sequentially adding more weak learners in the same way. The solution is to constrain the contribution of each weak learner by scaling it with a constant $\eta \in [0, 1]$ called the *learning rate*. For each step k, we get the composite model $F_k(\boldsymbol{x}) = F_{k-1}(\boldsymbol{x}) + \eta f_k(\boldsymbol{x})$. The final model is then given as

$$F_K(\boldsymbol{x}) = F_0(\boldsymbol{x}) + \sum_{k=1}^{K} \eta f_k(\boldsymbol{x}) \ . \tag{5}$$

Most boosting algorithms follow the fundamental principles of this illustrative scheme, but they differ in several key aspects, particularly in the choice of weak learners $f(\boldsymbol{x})$. Modern boosting algorithms have evolved to include several sophisticated strategies for managing the bias-variance trade-off, and are even better at balancing flexibility with generalization to avoid overfitting.

Boosted regression trees is a boosting technique which uses simple regression trees as weak learners to fit the ensemble model. A tree-ensemble model extends the general framework in equation (5) by utilizing regression trees as weak learners. Given the dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}$ ($|\mathcal{D}| = n, \boldsymbol{x}_i \in \mathbb{R}^m, y_i \in \mathbb{R}$), with sample size n and m covariates, and assuming the model incorporates K regression trees, the resulting predictive model can be expressed as

$$\hat{y}_i = f(\boldsymbol{x}_i) = \sum_{k=1}^{K} f_k(\boldsymbol{x}_i), \;\; f_k \in \mathcal{F} \;, \tag{6}$$

where we define the space of regression trees as $\mathcal{F} = \{f(\boldsymbol{x}) = \omega_{q(\boldsymbol{x})} \mid q : \mathbb{R}^m \mapsto \{1, 2, ...J\}, \; \omega : \{1, 2, ..., J\} \mapsto \mathbb{R}\}$ (Chen and Guestrin, 2016). Here, $q(\boldsymbol{x}_i)$ represents the tree structure itself, mapping the covariates $\boldsymbol{x}_i$ to the corresponding leaf node. Number of leaves in the tree is denoted J. We can then interpret $f_k(\boldsymbol{x}_i)$, $\boldsymbol{x}_i \in R_j$, as the independent tree structure $q(\boldsymbol{x}_i)$ with corresponding leaf score (weight) ω_j (Chen and Guestrin, 2016).

2.2.4 XGBoost model

XGBoost extends the principles of boosted regression trees described in Section 2.2.3 by incorporating several improvements and layers of complexity. XGBoost uses a regularized loss function that makes it less prone to overfitting. The regularized loss function is also twice differentiable, such that XGBoost can utilize second-order gradients to refine the optimization process. XGBoost also includes techniques like tree pruning and parallelized computation for faster model training. These innovations make XGBoost a powerful and scalable boosting framework. In this section we will go through some of the specific implementation details of XGBoost derived by Chen and Guestrin (2016). Assuming XGBoost follows the predictive model given in equation (6), the regularized loss function used by XGBoost, hereby the *objective function*, is given in general form as

$$\mathcal{L}(y, f(\boldsymbol{x})) = \sum_{i=1}^{n} L(y_i, f(\boldsymbol{x}_i)) + \sum_{k=1}^{K} \Omega(f_k) \;, \tag{7}$$

where Ω denotes the regularization parameter defined as

$$\Omega(f) = \gamma J + \frac{1}{2}\lambda \|\omega\|_2^2 \;.$$

Here $\|\cdot\|_2$ denotes the L^2-norm. The hyperparameter γ controls the complexity of the model by penalizing each additional leaf node in a tree, thus effectively acting as a threshold for the minimum improvement in the loss reduction required to add a new leaf. The hyperparameter λ is the L^2 regularization parameter, while ω is the leaf weight assigned to a specific leaf in the tree. The loss function $L(\cdot)$ given in equation (7) is some twice differentiable loss function.

To minimize the objective function, XGBoost utilizes its twice differentiable loss function by implementing a version of Newton's method. Assuming we are at the k'th boosting step, the composite objective function is given as

$$\mathcal{L}^{(k)} = \sum_{i=1}^{n} L\left(y_i, f^{(k-1)}(\boldsymbol{x}_i) + f_k(\boldsymbol{x_i})\right) + \Omega(f_k) \;.$$

In order to decrease the loss, we utilize Newton's method to find the next suitable f_k. The loss function is approximated by a Taylor series expansion

$$L\left(y_i, f^{(k-1)}(\boldsymbol{x}_i) + f_k(\boldsymbol{x}_i)\right) \approx L\left(y_i, f^{(k-1)}(\boldsymbol{x}_i)\right) + g_i f_k(\boldsymbol{x}_i) + \frac{1}{2} h_i f_k(\boldsymbol{x}_i)^2 \;,$$

where $g_i = \partial_{f^{(k-1)}(\boldsymbol{x}_i)} L\left(y_i, f^{(k-1)}(\boldsymbol{x}_i)\right)$ and $h_i = \partial^2_{f^{(k-1)}(\boldsymbol{x}_i)} L\left(y_i, f^{(k-1)}(\boldsymbol{x}_i)\right)$ denote the gradient and Hessian respectively, with respect to the current prediction. Substituting the loss function by its quadratic Taylor expansion in the neighborhood of $f^{(k-1)}(\boldsymbol{x}_i)$ and removing the constant term $L\left(y_i, f^{(k-1)}(\boldsymbol{x}_i)\right)$ gives the following simplified objective function at boosting step k

$$\tilde{\mathcal{L}}^{(k)} = \sum_{i=1}^{n} \left[g_i f_k(\boldsymbol{x}_i) + \frac{1}{2} h_i f_k(\boldsymbol{x}_i)^2 \right] + \Omega(f_k) \ . \tag{8}$$

Having derived the simplified objective, we are closer to our two main goals, namely to find the optimal weight for each leaf ω_j^* and to find a good splitting criterion. To find the optimal weights, we start by defining the index set I_j as the set of indices i that are in the j'th leaf node of f_k, or more formally, $I_j = \{i \mid q(\boldsymbol{x}_i) = j\}$. By expanding $\Omega(f_k)$ and writing equation (8) at leaf level, we get the following expression

$$\begin{aligned}
\tilde{\mathcal{L}}^{(k)} &= \sum_{i=1}^{n} \left[g_i f_k(\boldsymbol{x}_i) + \frac{1}{2} h_i f_k(\boldsymbol{x}_i)^2 \right] + \gamma J + \frac{1}{2} \sum_{j=1}^{J} \omega_j^2 \\
&= \sum_{j=1}^{J} \left[\omega_j \left(\sum_{i \in I_j} g_i \right) + \frac{1}{2} \omega_j^2 \left(\sum_{i \in I_j} h_i + \lambda \right) \right] + \gamma J \ .
\end{aligned}$$

Assuming a fixed tree structure $q(\boldsymbol{x})$, we can find the optimal weight ω_j^* for a general region R_j by differentiating and equating to zero

$$\begin{aligned}
0 &= \partial_{\omega_j} \sum_{j=1}^{J} \left[\omega_j \left(\sum_{i \in I_j} g_i \right) + \frac{1}{2} \omega_j^2 \left(\sum_{i \in I_j} h_i + \lambda \right) \right] + \partial_{\omega_j} \gamma J \\
&= \sum_{i \in I_j} g_i + \omega_j \left(\sum_{i \in I_j} h_i + \lambda \right) \\
&\Rightarrow \omega_j^* = -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda} \ .
\end{aligned}$$

We now move on to the goal of finding a good splitting criterion. Having found the optimal weight ω_j^*, we can find the optimal loss by inserting ω_j^* into equation (8), giving us the expression

$$\tilde{\mathcal{L}}_*^{(k)} = -\frac{1}{2} \sum_{j=1}^{J} \frac{\left(\sum_{i \in I_j} g_i \right)^2}{\sum_{i \in I_j} h_i + \lambda} + \gamma J \ . \tag{9}$$

Finding the optimal split is a very demanding optimization task. Instead, a greedy approach of recursive binary splitting is used. Here, the objective $\tilde{\mathcal{L}}_*^{(k)}$ from equation (9) becomes useful as it provides a way to measure the "quality" of a tree structure $q(\boldsymbol{x})$ (Chen and Guestrin, 2016). Splits can then be evaluated by comparing the objective $\tilde{\mathcal{L}}_*^{(k)}$ before and after a split. We take the greedy approach of choosing the split with the biggest reduction in the objective $\tilde{\mathcal{L}}_{(k)}$. Formally, for a given node j, let I_j denote the indices of the node. We propose a split for the node into left and right child nodes with I_L and I_R as their respective index sets, such that $I_j = I_L \cup I_R$. The total loss before the split is given by

$$\mathcal{L}_{before} = -\frac{1}{2} \frac{\left(\sum_{i \in I_j} g_i \right)^2}{\sum_{i \in I_j} h_i + \lambda} + \gamma \ .$$

The total loss after the split is

$$\begin{aligned}
\mathcal{L}_{after} &= \mathcal{L}_{(L)} + \mathcal{L}_{(R)} \\
&= -\frac{1}{2} \frac{\left(\sum_{i \in I_L} g_i \right)^2}{\sum_{i \in I_L} h_i + \lambda} - \frac{1}{2} \frac{\left(\sum_{i \in I_R} g_i \right)^2}{\sum_{i \in I_R} h_i + \lambda} + 2\gamma \ .
\end{aligned}$$

Thus, the total change in loss (*i.e.* the gain) from making this split is

$$\mathcal{L}_{split} = \frac{1}{2}\left[\frac{\left(\sum_{i\in I_L} g_i\right)^2}{\sum_{i\in I_L} h_i + \lambda} + \frac{\left(\sum_{i\in I_R} g_i\right)^2}{\sum_{i\in I_R} h_i + \lambda} - \frac{\left(\sum_{i\in I_j} g_i\right)^2}{\sum_{i\in I_j} h_i + \lambda}\right] - \gamma \ ,$$

which is greedily maximized in the top-down greedy binary splitting approach. For more specific implementation details of XGBoost, the reader is referred to Chen and Guestrin (2016).

2.2.5 Hyperparameters of XGBoost

In this section we will look at the some of the most important hyperparameters ($\boldsymbol{\theta}$) of XGBoost. As XGBoost is a tree ensemble, the number of trees to include in the ensemble is an important factor in the model. The number of trees is set by the hyperparameter *n_estimators* in XGBoost. As a general rule, the more trees we include, the more reliable the predictions of the model are. However, the number of trees necessary for a good result is highly dependent on the hyperparameter learning_rate (η). The concept of learning rate and its function to constrain the contribution of each tree to the final model in order to reduce overfitting was already introduced in Section 2.2.2. We can metaphorically think about the relationship of these variables as if walking a fixed distance, where learning_rate is the step size and n_estimators is the number of steps. When increasing the number of estimators (trees), the learning rate should decrease. Increasing the number of trees will also increase the time complexity, which is the constraining factor when choosing a good value for n_estimator.

The hyperparameter *subsample* is a concept that is borrowed from the random forest model, among other models. It decides the proportion of data used for building each tree in the model. The subset of data points is chosen at random. By fitting each tree on only a subset of the data drawn at random, we ensure a diverse tree sample with less correlation between trees. Without subsample, we risk that each tree will be similar since they, after all, use the same greedy approach of choosing a split. Subsample thus helps combat overfitting. The hyperparameter *colsample_bytree* is similar, except it is the proportion of randomly sampled covariates to train each tree on. This method, sometimes also called feature bagging, has similar effects as subsample, and it allows us to explore more of the covariate space. Feature bagging may also help detect outliers (Lazarevic and Kumar, 2005).

The hyperparameter *max_depth* decides the maximum depth that a tree can grow. The deeper a tree grows, the higher the complexity and probability of overfitting. However, a deeper tree with more decision paths will allow the model to capture more patterns in the data. Somewhat correlated to the max_depth is the hyperparameter *min_child_weight*, which in a regression setting decides the minimum number of data points allowed in a leaf node. As a tree keeps making splits and growing in depth, there will be fewer and fewer data points in each leaf node. Sometimes it may be better to allow a tree to grow deep and keep exploring specific patterns, and instead fix the minimum number of data points allowed in a leaf node to prevent overfitting.

Hyperparameters are the parameters that are set before training of a model begins and they define a lot of the internal structure of the model. They are crucial in machine learning algorithms like XGBoost as they control the learning process of the model. Tuning of hyperparameters is a process that is done prior to fitting a model and is a topic that is covered in Section 2.2.6.

2.2.6 Bayesian optimization

The analysis of hyperparameters for XGBoost in Section 2.2.5 underlines that choosing the right set of hyperparameters is important for the performance of a model. In addition, when it comes to optimizing the objective function, the hyperparameters are often highly correlated. Finding the optimal set of hyperparameters, commonly referred to as *hyperparameter tuning*, therefore leads to a complicated optimization problem. Figure 2 shows a simple simulated example of what such an optimization problem might look like in a two-dimensional hyperparameter space. One naive

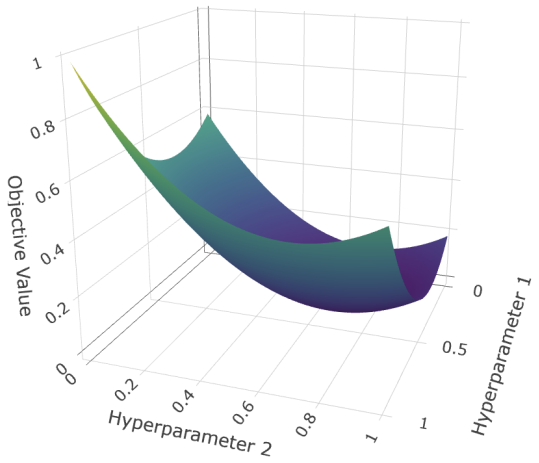


Figure 2: Simulated example of hyperparameter space $(\theta_1, \theta_2) \in [0, 1]^2$ with the objective function $\mathcal{L}(\theta_1, \theta_2) = (\theta_1 - 0.7)^2 + (\theta_2 - 0.3)^2$.

way of searching for a good set of hyperparameters is to simply evaluate the objective function on a uniform grid in some finite subspace of the hyperparameter space. This approach suffers from the curse of dimensionality (Kuo and Sloan, 2005), as the number of function evaluations needed grows exponentially with increasing number of dimensions d. Another approach is to choose a fixed number of function evaluations, then uniformly randomly choose some points in a subspace of the hyperparameter space to evaluate, and finally choose the set of hyperparameters with the smallest objective value. However, this approach is also inefficient, as it will treat all the grid points the same way regardless of their relevance to the objective function. One can thus wonder whether it would not be better to have a method that learns from previous evaluations to guide the future sampling. In fact, this is the idea behind *Bayesian optimization*.

To describe Bayesian optimization, we first have to cover some of the basics in Bayesian statistics. The Bayesian approach to statistics is an alternative to the "classical" frequentist approach (Bolastad, 2009). In the Bayesian approach, the laws of probability are applied directly to the problem. For any continuous random variables X and Y, the conditional probability density of X given $Y = y$ is given as

$$f(x|y) = \frac{f(y|x)f(x)}{f(y)} \ .$$

In the Bayesian approach, the probabilities are extended to include the prior belief in different values of unknown parameters θ, and then updated via Bayes theorem such that

$$f(\theta|x) \propto f(x|\theta)f(\theta) \ , \tag{10}$$

where the updated probability distribution $f(\theta|x)$ is called the *posterior probability distribution*, or simply the posterior distribution, $f(x|\theta)$ is the *likelihood* and $f(\theta)$ is the *prior probability distribution*, or simply the prior.

Having derived some of the basics of Bayesian statistics, we can now turn our attention to Bayesian optimization. Bayesian optimization is a method that can be used in optimizing the objective function in machine learning algorithms in order to tune the hyperparameters. The methods in Bayesian optimization leverage the previously evaluated objective points to strategically select the new point for evaluation, and thereby improving efficiency (Frazier, 2018). The problem that Bayesian optimization tries to solve can be expressed through the following optimization problem

$$\arg\max_{\boldsymbol{\theta} \in H} \ \mathcal{L}(\boldsymbol{\theta}) \ , \tag{11}$$

where $\boldsymbol{\theta} \in \mathbb{R}^d$ are the hyperparameters we want to tune. Moreover, $H \subset \mathbb{R}^d$ is the subset in which we will search for the optimal hyperparameters, and is often called the *search space*. Bayesian optimization is typically a good choice when the objective $\mathcal{L}$ is "expensive" to evaluate in terms

of time complexity and when the dimension d is not very small. If d is small and the objective "cheap" to evaluate, the naive grid method might be a good alternative since we can evaluate $\mathcal{L}$ on a refined grid which when minimized should give a reasonably good approximation for the best hyperparameters. An advantage of Bayesian optimization is that it only needs evaluations of the objective, and it does not require the objective to be differentiable. This makes the Bayesian optimization technique flexible, as many objective functions in machine learning are not differentiable. If Bayesian optimization is used to tune the hyperparameters of a machine learning model, $\mathcal{L}$ in equation (11) can be interpreted as the negative of the objective function, as the aim is to minimize the objective function in this case.

We can describe the Bayesian optimization process in two steps. The first step is to create a computationally cheap approximation of the objective $\mathcal{L}$, often called a surrogate model (Frazier, 2018). We build the surrogate model from function evaluations of the objective function. The surrogate model provides predictions with mean and variation in the whole domain of interest. The second step is to evaluate the *acquisition function*. The aim of the acquisition function is to provide a "score" for each point in the domain by using the predictions of the surrogate model, and then the next point to evaluate is decided by maximizing the score. Since the goal in Bayesian optimization is to find the global optimum, the score needs to strike a balance between exploring the search space and exploiting regions where the objective function gives promising results. This dilemma is a famous problem in *reinforcement learning* in machine learning, and is often referred to as the exploration vs. exploitation trade-off (Pack Kaelbling et al., 1996). There are a number of choices for the surrogate model and the acquisition function. Here we will cover two of the most common ones, namely the surrogate model by *Gaussian process regression* and *expected improvement* (EI) acquisition function.

In Bayesian optimization, the surrogate is developed using methods from Bayesian statistics, and the most common choice is to use Gaussian process regression (Frazier, 2018). In short, the goal of Gaussian process regression is to provide a posterior distribution of $\mathcal{L}$, which provides a measure for the mean and uncertainty of $\mathcal{L}$ in the whole hyperparameter space, given the objective evaluations calculated at that point in time. When a new objective evaluation is made, the posterior distribution can be updated with this evaluation, increasing the "knowledge" of the posterior distribution, similar to the process described in equation (10). Suppose we have evaluated $\mathcal{L}$ at the points $\theta_1, ..., \theta_k \in \mathbb{R}^d$ giving us the function evaluations $\boldsymbol{\mathcal{L}}^{(k)} := (\mathcal{L}_1, ..., \mathcal{L}_k)^T$, where $\mathcal{L}_i := \mathcal{L}(\boldsymbol{\theta}_i)$ and $\boldsymbol{\mathcal{L}}^{(k)}$ is the vector of k objective evaluations. We then have the prior distribution

$$\boldsymbol{\mathcal{L}}^{(k)} \sim N_k(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \ , \tag{12}$$

where $\boldsymbol{\mu}_k \in \mathbb{R}^k$ is the mean and $\boldsymbol{\Sigma}_k \in \mathbb{R}^{k \times k}$ is the covariance matrix, often called the *kernel* in Bayesian optimization. The mean vector and kernel are calculated by some chosen mean and kernel functions, denoted $\mu(\boldsymbol{\theta})$ and $\Sigma(\boldsymbol{\theta})$ respectively. Popular choices of mean and kernel functions are outside the scope of this thesis, for a more detailed explanation we refer to Frazier (2018). However, it is worth mentioning that the mean function approximates the expected value of $\mathcal{L}(\boldsymbol{\theta})$ and that the kernel function has the property that two points, $\boldsymbol{\theta}_i$, $\boldsymbol{\theta}_j$, that are close to each other have a high positive correlation in order to model the belief that points close by tend to have similar function evaluations (Frazier, 2018). The idea behind the posterior distribution is that we can infer the value of some new point $\boldsymbol{\theta}_{k+1}$ using the information from previous points. Assume we have the function evaluations $\boldsymbol{\mathcal{L}}^{(k)} = (\mathcal{L}_1, ..., \mathcal{L}_k)^T$, which we give a prior distribution by equation (12). We can then calculate the posterior distribution $\mathcal{L}(\boldsymbol{\theta}_{k+1})|\mathcal{L}(\boldsymbol{\theta}^{(k)})$ by computing the conditional distribution, which is given as (Bousquet et al., 2003)

$$\begin{aligned} \mathcal{L}(\boldsymbol{\theta}_{k+1})|\mathcal{L}(\boldsymbol{\theta}^{(k)}) &\sim N(\mu_{cond}, \sigma^2_{cond}), \\ \mu_{cond} &= \Sigma(\boldsymbol{\theta}_{k+1}, \boldsymbol{\theta}^{(k)})\Sigma(\boldsymbol{\theta}^{(k)}, \boldsymbol{\theta}^{(k)})^{-1}\left(\mathcal{L}^{(k)} - \mu(\boldsymbol{\theta}^{(k)})\right) + \mu(\boldsymbol{\theta}_{k+1}), \\ \sigma^2_{cond} &= \Sigma(\boldsymbol{\theta}_{k+1}, \boldsymbol{\theta}_{k+1}) - \Sigma(\boldsymbol{\theta}_{k+1}, \boldsymbol{\theta}^{(k)})\Sigma(\boldsymbol{\theta}^{(k)}, \boldsymbol{\theta}^{(k)})^{-1}\Sigma(\boldsymbol{\theta}^{(k)}, \boldsymbol{\theta}_{k+1}) \ . \end{aligned} \tag{13}$$

Now that we have a way to approximate the mean and uncertainty of $\mathcal{L}$ in the search space for a given number of function evaluations using the posterior distribution, we need a method of choosing *which* new point $\boldsymbol{\theta}_{k+1}$ to evaluate. In other words, we need an acquisition function.

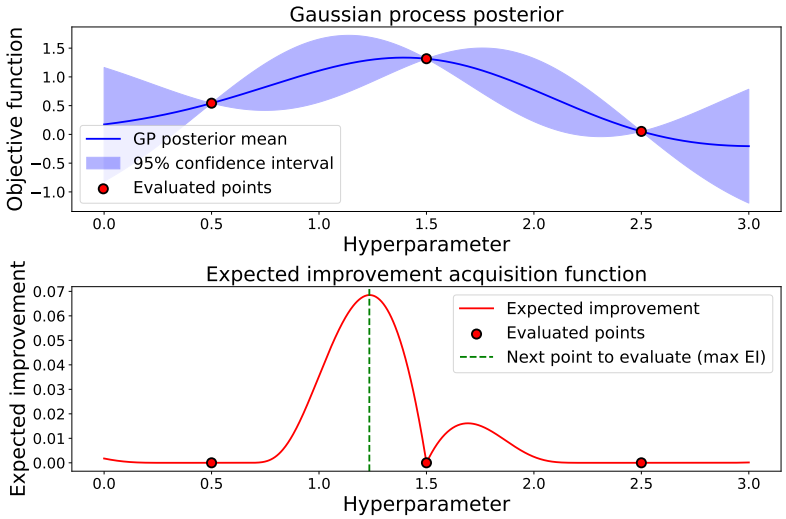


Figure 3: Objective function $\mathcal{L}(\theta) = \exp(-(\theta-1.5)^2) - 0.5\exp(-(\theta-2.5)^2)$ with Gaussian process posterior mean and 95% confidence interval (above) and expected improvement acquisition function (below). Evaluated points shown in red. Illustration in 1-dimensional hyperparameter space.

One of the most popular acquisition functions is the expected improvement acquisition function (Frazier, 2018). The expected improvement can be formulated in the following way. Assume the best objective evaluation so far is given by

$$\mathcal{L}^* := \max_i \mathcal{L}_i, \ i = 1, ..., n \ .$$

If the new objective evaluation $\mathcal{L}_{k+1}$ is higher than $\mathcal{L}^*$, we have an *improvement* of $\mathcal{L}_{k+1} - \mathcal{L}^*$, otherwise we have no improvement, and we set the improvement to be 0. We can express the improvement as

$$[\mathcal{L}_{k+1} - \mathcal{L}^*]^+ := \begin{cases} \mathcal{L}_{k+1} - \mathcal{L}^*, & \mathcal{L}_{k+1} > \mathcal{L}^* \\ 0, & \text{otherwise} \end{cases}$$

The tricky part is that we do not know the improvement until after we have made the evaluation $\mathcal{L}_{k+1}$. Expected improvement is a method that tries to maximize the expectation of this improvement, and it can be expressed as

$$E[I_k(\boldsymbol{\theta})] := E_k\left[[\mathcal{L}_{k+1} - \mathcal{L}^*]^+\right] \ ,$$

where the expectation $E_k[\cdot] = E[\cdot|\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_k, \mathcal{L}_1, ..., \mathcal{L}_k]$ is calculated under the posterior distribution of $\mathcal{L}$ at $\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_k$ (Frazier, 2018). For a more detailed explanation we again refer the reader to Frazier (2018). Note that maximizing the expected improvement under the posterior distribution will tend to choose a new point $\boldsymbol{\theta}_{new}$ at points where the posterior mean and posterior uncertainty (*i.e.* variation) is large, which is consistent with respecting the exploration vs. exploitation trade-off. A simple example of the Gaussian process posterior with expected improvement acquisition function is illustrated in Figure 3 for a one-dimensional hyperparameter space.

The algorithm for the whole Bayesian optimization process is given in Algorithm 1. The algorithm assumes only one initial randomly uniformly distributed point $\boldsymbol{\theta}_1 \in H$ is chosen, but it is possible to initialize $\mathcal{L}$ with more than one such point. Note that in the algorithm we define $(\mathcal{L}_1, \mathcal{L}^{(0)})^T$ to be equal to $\mathcal{L}_1$ such that the first iteration of the for-loop is well-defined.

2.2.7 Interpretable machine learning

Statistical learning methods can at a very low level be divided into the task of *prediction* and the task of *inference*. Assuming we have the dataset $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, where $\boldsymbol{x} \in \mathbb{R}^p$ are the covariates

Algorithm 1 Bayesian optimization procedure

Choose the amount of trials (evaluations) n to run
Place a Gaussian process prior on $\mathcal{L}$
Observe $\mathcal{L}$ at some uniformly randomly chosen point $\boldsymbol{\theta}_1 \in H$
Initialize $\mathcal{L}^* = \mathcal{L}(\boldsymbol{\theta}_1)$ and $\boldsymbol{\theta}_1 = \boldsymbol{\theta}^*$
for $i = 2, ..., n-1$:
  Update posterior $\mathcal{L}(\boldsymbol{\theta})|(\mathcal{L}_i, \mathcal{L}^{(i-1)})^T$ using all evaluations
  Decide $\boldsymbol{\theta}_{i+1}$ by maximizing the acquisition function over the posterior $\mathcal{L}(\boldsymbol{\theta})|(\mathcal{L}_i, \mathcal{L}^{(i-1)})^T$
  if $\mathcal{L}(\boldsymbol{\theta}_{i+1}) > \mathcal{L}^*$:
    $\mathcal{L}^* = \mathcal{L}(\boldsymbol{\theta}_{i+1})$
    $\boldsymbol{\theta}^* = \boldsymbol{\theta}_{i+1}$
    end if
  end for
Return the best set of hyperparameters $\boldsymbol{\theta}^*$

and $y \in \mathbb{R}$ is the response, the goal of prediction can in short be described as finding a model that predicts the response y while using the covariates $\boldsymbol{x}$ as input. Moreover, the model should aim to minimize the test error by minimizing the loss function, or in other words it should balance bias and variance so as to best predict the response on unseen data by capturing the underlying relationship between the covariates $\boldsymbol{x}$ and response y. The XGBoost model described in Section 2.2.4 is an example of a powerful prediction model. A prediction model is often described as a "black box" model, in the sense that we are not interested in what is going on "under the hood" of the model, as long as it yields accurate predictions for y (James et al., 2013). However, in many situations, we want more information than just the prediction itself. When the goal is inference, we are more interested in how an algorithm used the data to arrive at the prediction. For example, we might want to know which covariates are (the most) associated with the response. In the context of genomic studies, genomic prediction aims to estimate the breeding values based on genetic markers (SNPs) and other covariates. In the inference setting, one of the tasks is to find which SNPs are the most associated with the breeding value (trait).

In machine learning, we often discuss the *interpretability* of a model. We can tentatively describe the interpretability of a model as the degree to which a human can comprehend the decisions of the model, however there is no real consensus about what interpretability is in machine learning, nor is it clear how to measure it (Molnar, 2022). The terms interpretability and explainability are often used interchangeably. Complex and flexible models like XGBoost are typically more accurate for predicting nonlinear, faint, or rare phenomena, but unfortunately more accuracy often comes at the expense of interpretability (Boehmke and Greenwell, 2019). Interpreting machine learning models is an emerging field that has become known as *interpretable machine learning* (IML Boehmke and Greenwell, 2019).

IML encompasses methods that are interpretable as standalone models, such as regression trees, where the decision tree can mimic the way a human makes binary decisions. On the other hand, it also encompasses methods that can explain complex models like XGBoost after they have been fitted, which is a process often called post-hoc explanation (Molnar, 2022). These model explanations can be categorized as either global or local. Global methods are methods that provide insights into the average behavior of the model across the entire dataset. Local methods on the other hand focus on individual predictions (Molnar, 2022). We can categorize the different methods in IML as either model-agnostic or model specific. Model-agnostic methods are methods that can be applied to any machine learning model (Boehmke and Greenwell, 2019). Model specific methods work only for a specific type of model, usually exploiting the specific structure of the model. Model-agnostic methods allow us to utilize the predictive power of the machine learning model and interpretability component of the model-agnostic method, while also making it easier to compare feature importance across models. Section 2.3.2 covers the method of SHAP, which is a model-agnostic method of explaining XGBoost post-hoc that will be implemented on a genomic dataset later in this thesis.

2.3 Methods of inference in genomic studies

2.3.1 Genome-wide association studies

GWAS encompass approaches of finding which SNPs are associated with the trait of interest, and thus encompass methods of inference in genomic studies (Uffelmann et al., 2021). It is the genes that encode for polypeptides that make up proteins, and which eventually can affect a phenotype. As discussed in section 2.1.3, SNPs are nevertheless effective as genetic markers as they can be in LD with genes that code for the trait of interest, and as such can be used to map QTL. One of the methods GWAS use to find out if a SNP is significantly associated with a trait is to regress each SNP M_{ij} on the phenotype y_i. Assuming we have $j = 1, ..., m$ different SNPs for $i = 1, ..., n$ individuals, we can express this as

$$y_i = M_{ij}\beta_j + \epsilon_i, \quad i = 1, ..., n \ , \tag{14}$$

where ϵ_i is the residual term for individual i and β_j the coefficient for SNP j, M_{ij}, in the matrix of SNPs $\mathbf{M} \in \mathbb{R}^{n \times m}$. We thus fit a univariate regression model for each SNP. A univariate model is fit instead of the full model since the standard genomic dataset typically has a higher number of covariates m than sample size n, $m > n$, leading to an underdetermined system. We can now test the significance of each SNP by a standard hypothesis test $H_0 : \ \beta_j = 0$ against alternative hypothesis $H_1 : \beta_j \neq 0$. This test can be done by known methods such as the Wald test or the likelihood test. Doing this for each SNP, we get m different p-values. These p-values are typically visualized by a so-called Manhattan-plot, where the SNPs are ordered in their chromosome and bp position along the x-axis while the y-axis shows the $-\log_{10}(p)$ values. Regions of interest can then be spotted from outliers in p-values.

When doing GWAS-analysis on wild populations, chances are that some of the individuals in the sample are related. Relatedness naturally implies that those individuals will share more genetic material than on average otherwise, and they may also share environmental effects. This creates a correlation between related individuals in genetic structure or otherwise, such that the assumption of independence of observations in linear regression is broken. An alternative to the standard GWAS method given in equation (14) is to instead fit a univariate LMM where we include a random intercept on the identity of individuals. The point is then that the random intercept should take care of the correlation between individuals due to relatedness. The random intercept is typically assumed to follow a normal distribution with zero mean, and where the covariance is modeled by a relatedness matrix $\mathbf{G}$ similarly to the pedigree-based or genomic animal model.

One potential issue with GWAS is that the results can be context-dependent. As already discussed, relatedness can be one source of false positive/negative results in GWAS. The methods described also do not factor in epistatic effects, that is, gene-by-gene interaction effects, which also can influence the results of GWAS (McKinney and Pajewski, 2012). Similarly, effects from gene-by-sex interactions (Kyrgiafini et al., 2023) and gene-by-environment interactions (Murcray et al., 2009) are not taken into account. Another problem in GWAS, that has become known as the problem of "missing heritability", is the observation that GWAS-hits often explain only a small proportion of the heritability of most quantitative traits (Eichler et al., 2010). The missing heritability problem is seen in for example the highly heritable trait human height, and believed to arise because such traits are complex and polygenic and thus affected by many genes with alleles of small effect sizes (Yang et al., 2010; Yengo et al., 2022). Most of the associations observed in genetic studies are due to a confounding effects, that is, a factor which is associated with both the SNP (or other genetic marker) and the trait of interest (Aulchenko, 2011). If we would have controlled for this factor, for example in a LMM, the association between the SNP and the trait would be removed. If the factor in question has a causative relation to the trait of interest, this is an association we would want to pick up on in GWAS, and as such it is a "good" association. However, another type of confounding is confounding by population structure (Aulchenko, 2011), which can be a concern in GWAS when dealing with several slightly genetically different populations, as it can lead to false positives (Berg et al., 2019). This effect is relevant to this thesis as we will be working with data from the meta-population of house sparrow in the Helgeland island system, where differentiation due to dispersal and genetic drift has been demonstrated (Jensen, Moe et al., 2013; Saatoglu et al., 2021).

Another potential issue with GWAS is its use of p-values. Let us first formally define what a (one-sided) p-value is

Definition 2.1 (p-value)**.** The p-value is the probability of obtaining the observed statistic, or more extreme results, assuming the null-hypothesis is true.

$$p = P(X \geq x_{obs}|H_0) \ .$$

The general use of p-values has become a highly debated topic in the many scientific communities, particularly the standard method of using null-hypothesis significance testing with an arbitrary p-value cutoff (Goodman, 2008; Head et al., 2015; Ioannidis, 2018; Muff, Nilsen et al., 2022). One of the issues discussed is that the interpretation of p-values is confusing for many, and they can often be misinterpreted, as pointed out by Goodman (2008). Some are also worried about the potential to (mis)use p-values to get misleading results in form of "p-hacking", which is a type of bias that occurs when researchers collect or select data or statistical analyses until nonsignificant results become significant (Head et al., 2015). An issue that is directly related to genomic studies is the fact that the value of a p-value stem from both the magnitude of the effect size and the uncertainty of the effect size. Figure 4 demonstrates two different scenarios where the p-value is approximately equal to 0.01, but have completely different effect sizes. This brings the attention back to the issue of missing heritability. Complex polygenic traits require huge sample sizes to reduce the uncertainty to a level were significance is achieved, as was shown for example by the analysis of human height (Yang et al., 2010; Yengo et al., 2022). Since p-values do not provide an estimate for the effect size, it also makes it more difficult to compare results across different experiments using p-values.

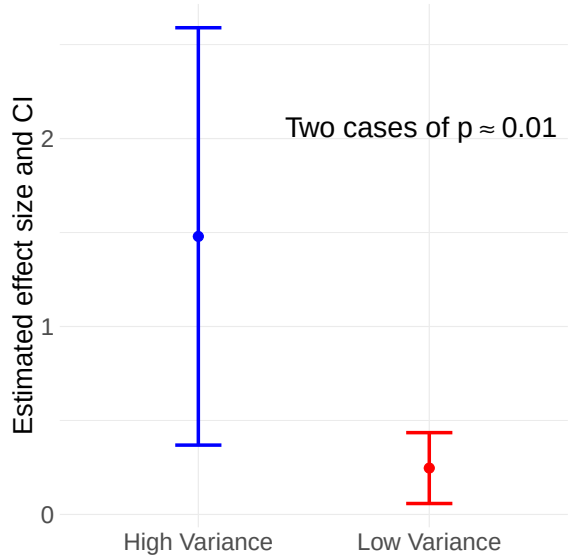


Figure 4: Simulated example of two hypothesis tests with $H_0 : \beta_i = 0$ for two linear models $\boldsymbol{y}_i \sim \boldsymbol{x}\boldsymbol{\beta}_i + \boldsymbol{\epsilon}_i$, $i = 1, 2$. Both significance tests result in p-value ≈ 0.01. Linear models simulated with sample sizes $n_{1,2} = 24, 130$, parameters $\beta_{1,2} = 1.5, 0.2$ and variables $\epsilon_1,\ \epsilon_2,\ x \sim N(0, 3^2),\ N(0, 1^2),\ N(0, 1^2)$.

2.3.2 Shapley additive explanations

Another method of finding which SNPs are associated with the trait of interest is by calculating *Shapley values*. The method of using Shapley values is a concept developed from coalition game theory by L. S. Shapley (Shapley, 1952). The Shapley value is a method for assigning payouts to players depending on their contribution to the total value of a game (Molnar, 2022). To explain Shapley values, we first have to introduce some terminology and define what a *game* is. For a

set of players S in the set of all combinations of players U, $S \subset U$, we define the value function $v : S \mapsto \mathbb{R}$ as the value function of a game if it has the following properties

$$\begin{aligned} &v(\emptyset) = 0 \ , \\ &v(S) \geq v(S \cap T) + v(S - T) \quad \forall S, T \subset U \ . \end{aligned}$$

Further, we define the carrier of v as any set $N \subset U$ such that $v(S) = v(N \cup S)\ \forall S \subset U$. Carriers are useful as the players outside any carrier do not make a contribution to the payout through the value function. Now we define $\Pi(U)$ as the set of permutations of U. Then, for $\pi \in \Pi(U)$, πS is the image of S under π. Further, we define the function πv by $\pi v(\pi S) = v(S)\ \forall S \subseteq U$. With the groundwork done, we can define the Shapley value $\phi_i(v)$ of a game v as

$$\phi_i(v) = \sum_{S \subseteq N, i \in S} \frac{(|S|-1)!(|N|-|S|)!}{|N|!} v(S) - \sum_{S \subseteq N, i \notin S} \frac{|S|!(|N|-|S|-1)!}{|N|!} v(S), \quad \forall i \in N \ .$$

Another formulation for the Shapley values is

$$\phi_i(v) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(|N|-|S|-1)!}{|N|!} \left(v(S \cup \{i\}) - v(S)\right), \quad \forall i \in N \ . \tag{15}$$

Here, $|S|!$ represents the number of permutations of the players in S. The number of permutations for the remaining players not in S is given by $(|N|-|S|-1)!$. The denominator $|N|!$ represents the total number of permutations for all players, and scaling by this constant normalizes the weights so that they sum to 1 over all subsets in the sum. The weights scale the marginal contribution of player i, $v(S \cup \{i\}) - v(S)$, by the fraction of permutations where S appears in a random ordering of players. Thus, the Shapley value averages the marginal contribution of player i across all possible subsets of players S.

It can be shown that the Shapley value is the unique solution to three properties (Shapley, 1952). The first property is the symmetry property, which states that for each π in $\Pi(U)$, $\phi_{\pi i}(\pi v) = \phi_i(v)\ \forall i \in U$. The second property is that of efficiency, which states that for each carrier N of v, $\sum_N \phi_i(v) = v(N)$. The final property is the property of linearity, such that for any two games v and w, we have that $\phi(v + w) = \phi(v) + \phi(w)$. The intuitive understanding of the three properties and the Shapley value will become more clear as we relate it to predictions in machine learning.

We connect these concepts from game theory to machine learning by treating the prediction $f(\boldsymbol{x}_i)$ as the "game" and the covariates $\boldsymbol{x}_i \in \mathbb{R}^p$ as the "players" in the game. Shapley values then allow us to fairly attribute the contributions of each feature to the prediction. Assume we have the training set $\mathcal{D}_{train} = \{y_i, \boldsymbol{x}_i\}_{i=1}^{n_{train}}$ and let $f(\boldsymbol{x}_i^*)$, $\boldsymbol{x}_i^* \notin \mathcal{D}_{train}$ be the model trying to predict $y_i^* \notin \mathcal{D}_{train}$. We can then decompose the prediction as

$$f(\boldsymbol{x}_i^*) = \phi_0 + \sum_{j=1}^{p} \phi_{ij} \ ,$$

where $\phi_0 = E[f(\boldsymbol{x}) \mid \boldsymbol{x} \in \mathcal{D}_{train}]$ is the global mean prediction and ϕ_{ij} is the SHAP value for covariate j and observation $\boldsymbol{x}_i^*$ (Aas et al., 2021). The SHAP values thus distribute the difference between prediction $f(\boldsymbol{x}_i^*)$ and the global mean prediction. The SHAP values still satisfy the three properties of Shapley values, and through the lens of SHAP values, the three properties can be stated as

- Efficiency: The sum of SHAP values equals the total game value:
  $\sum_{j=1}^{p} \phi_{ij} = f(\boldsymbol{x}_i^*) - \phi_0 = v(U)$, where U is the set of all covariates.
- Symmetry: If two covariates contribute equally to all subsets S, they receive the same SHAP value:
  $v(S \cup \{m\}) = v(S \cup \{t\}) \quad \forall S \subseteq U \setminus \{m, t\} \ \Rightarrow \ \phi_m(v) = \phi_t(v)$.
- Linearity: For two combined models v and w, SHAP values add linearly:
  $\phi_j(v + w) = \phi_j(v) + \phi_j(w)$.

We can also state a fourth property from the property of v having carriers. We define the dummy property as the property that covariates that do not influence the prediction, receive a SHAP value of zero:

$$v(S \cup \{m\}) = v(S) \quad \forall S \subseteq U \backslash \{m\} \;\Rightarrow\; \phi_m(v) = 0 \; .$$

It is worth stressing that SHAP values are unique in satisfying these properties, and it is because of these properties that we can call the SHAP values "fair".

To compute the SHAP values, we need to define the contribution of a subset of covariates $v(S)$, $S \subseteq \{1, ..., p\}$. In SHAP, $v(S)$ often represents the expected model output conditional on the values of S (Aas et al., 2021). This allows us to connect the game theory function to feature importance in machine learning. The value function is then given as

$$v(S) = E[f(\boldsymbol{x}|\boldsymbol{x}_S = \boldsymbol{x}_S^{(i)})] \; .$$

Here, $\boldsymbol{x}_S^{(i)}$ denotes the covariates in S for observation i. In summary, SHAP values explain individual predictions of machine learning models such as XGBoost. Thus, SHAP values are local explanation methods. The SHAP value of a specific covariate for a specific prediction gives a measure for the importance of this feature to the prediction relative to the other covariates. The SHAP values also need to be interpreted relative to the global mean prediction, since all the SHAP values for a specific prediction sum to the deviation of this prediction from the global mean prediction.

Calculation of a single Shapley value given by equation (15) requires us to evaluate all possible combinations of subsets of covariates from the set $\{1, ..., p\}$, meaning a total of 2^p possible subsets. This quickly becomes infeasible for large p, and in particular for large genomic datasets. In this thesis we will use SHAP values to explain predictions from the XGBoost model. For tree-based methods, the *Tree Shapley additive explanations* (TreeSHAP Lundberg et al., 2018) is a good alternative way to compute the Shapley values. TreeSHAP is a high-speed model specific method of estimating Shapley values of tree ensembles that reduces the time complexity from $O(TL2^p)$ to (TLD^2). Here, T denotes the number of trees, L denotes the maximum number of leaves in a tree, D is the maximum depth of a tree and p is the amount of covariates in the tree model. The TreeSHAP method provides the same exact Shapley values as SHAP does. It has a faster computation time since it exploits the structure of the trees to calculate the SHAP values without evaluating every subset explicitly. In short, the method calculates the marginal contributions of each covariate based on how the covariate values affect the model's output when the covariate is included in different paths or branches of the tree. This ensures that the Shapley values remain consistent with the original SHAP method and thus still maintains the same fairness properties. The exact algorithm is beyond the scope of this thesis, we refer to Lundberg et al. (2018).

3 Methods

3.1 Data description

In this thesis we will be working with morphological and genotypic data from a long-term study of an insular house sparrow off the island system off Helgeland coast in northern Norway (Jensen, Steinsland et al., 2008; Pärn et al., 2011; Muff, Niskanen et al., 2019). The study on the house sparrows has been running continuously since 1993. The island system consists of 18 different islands, and the different islands differ in environmental conditions, habitat type and population sizes, such that each island is a population in itself. However, since there is migration between the islands, we can view the whole island system as a meta-population. For the data collected on the house sparrows, we use the naming convention of *inner* for the main islands closest to the mainland, *outer* for the main islands further away from the mainland and *other* for other smaller islands (Muff, Niskanen et al., 2019). The genomic data we will be working with consists of two datasets, which will be referred to as the 180k dataset and the 70k dataset. The 180k dataset contains 182 848 sequenced SNPs and 4625 recordings from 1984 unique individuals. The 70k dataset contains 65 247 sequenced SNPs and 24 951 recordings from 12 560 unique individuals.

Variable	Description
Sex	Sex of individual (1 male, 2 female)
FGRM	Inbreeding coefficient
Month	Month of capture
Age	Age of individual at capture
Outer	Proportion of genetic material from *outer* islands
Other	Proportion of genetic material from *outer* islands
Hatch island	Island the individual was born
Hatch year	Year the individual was born
Island current	Island individual was captured at
Ringnr	Unique individual ID

Table 1: Description of house sparrow variables used in the thesis.

The 180k dataset is thus characterized by having a lot of genomic material, but at the cost of fewer individuals. The 70k dataset has a higher sample size, but contains less genomic material. The morphological data available contains recorded phenotypes for many different traits. In this thesis we will be focusing on the traits *body mass* (referred to as mass) and *wing length* (referred to as wing). Several other house sparrow variables are recorded and used in this thesis. These variables are listen in Table 1. The non-genetic house sparrow data listed in the table will allow us to control for some of the variation in various conditions on the different individuals. This is useful for the statistical models used in this thesis. For a detailed explanation of the recording and sequencing process, we refer to Jensen, Steinsland et al. (2008).

3.2 Prediction of genetic contribution

3.2.1 Genomic prediction with XGBoost model

Machine learning, together with deep learning, are the front-runners in prediction based problems in data analysis, and various methods have been tested in animal and plant breeding (Gill et al., 2022). Genomic prediction in general has hardly been utilized in studying quantitative genetics in wild populations (Ashraf et al., 2021). In this thesis, we will utilize the XGBoost model on the genomic data from house sparrows dataset introduced in Section 3.1 to perform genomic prediction. The goal is to predict the genetic contribution on the phenotype for the traits mass and wing. XGBoost, which is a boosted regression tree ensemble, utilizes regression trees as weak learners. In Section 2.2.2 we discussed some of the strengths and weaknesses of the regression trees. Regression trees are ideal at capturing non-linear interaction effects. The traits mass and wing are assumed to be complex polygenic traits with epistatic effects between SNPs. In this regard, the XGBoost model is ideal at capturing these epistatic effects that the traditional animal model might not pick up on.

As discussed in Section 2.1.1, the variance of quantitative traits is complex to predict since it involves several effects, some of them non-genetic such as the environmental effects. If we train the XGBoost model on the phenotypic data y, the noise from these non-genetic effects will reduce the precision of the predictions. Thus, we first perform a pre-processing of the phenotypic data in order to remove as much of the noise as possible. To do this, we make use of some of the variables listed in Table 1 by including them in a LMM with the phenotype as response. The following variables are included in the LMM for both the 180k and the 70k datasets: Sex, Month, Age, Island current, Hatch year and Ringnr. For the 70k dataset we also include the variables Outer and Other. We can describe the LMM with the equation

$$\boldsymbol{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{W}\boldsymbol{d} + \boldsymbol{\rho} + \boldsymbol{\epsilon} \ .$$

Here, $\boldsymbol{\beta}$ is the vector of fixed effects, such as the fixed environmental effects, with corresponding design matrix $\mathbf{X}$ of appropriate dimension. The vector $\boldsymbol{d}$ contains the random environmental effects, with corresponding design matrix $\mathbf{W}$. The vector $\boldsymbol{\epsilon}$ is the residual error containing errors from factors not accounted for in the model. The vector $\boldsymbol{\rho}$ contains the identity effects of the

individuals, and is assumed to explain the variance between individuals that is not from factors accounted for in the model. In other words, $\boldsymbol{\rho}$ contains the variance due to genetic component in the phenotype, V_G. The variable $\boldsymbol{\rho}$ thus contains the genetic component we want to predict in the XGBoost model. The processed response we will use in training of the XGBoost model is therefore the estimate of the identity effect, given as

$$\boldsymbol{y}^* = \hat{\boldsymbol{\rho}} \ .$$

We thus use the processed phenotype y^* as response and the SNP data from either the 180k or the 70k dataset as covariates in the XGBoost model.

For the implementation of the XGBoost model, we need to choose which loss function L to use in the objective function given in equation (7). We have several options in the choice of loss functions, all with different strengths and weaknesses. In this thesis we will be using the *mean absolute error* (MAE) loss function, defined as

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i| \ .$$

The mean absolute error measures the magnitude of errors between predicted values $\hat{\boldsymbol{y}}$ and response $\boldsymbol{y}$ with a sample size n. Using the MAE as a loss function, the sum of absolute deviations are minimized, thus MAE measures the conditional median. The main strength of the MAE as opposed to for example the popular mean squared error (MSE) is that MAE is less sensitive to outliers since it estimates the conditional median, as opposed to the conditional mean in MSE. The observant reader will notice that the MAE is not a differentiable function, and that from the analysis of XGBoost in Section 2.2.4 we stated that the objective function in equation (7) requires a differentiable loss function. In this thesis we will use the Python package *xgboost* v2.1.1, which contains the option of using the MAE as a loss function (*XGBoost Parameters* 2022). Using MAE will be slower than using differentiable loss functions since non-derivative line search methods are used instead (*Configure XGBoost* 2024).

For the hyperparameter tuning of the XGBoost model, we will apply the Bayesian optimization method using Python package *Optuna* (Akiba et al., 2019). The Optuna package utilizes the *tree-structured Parzen estimator* (TPE Bergstra et al., 2011). The TPE algorithm is a parameter-sampling algorithm. In short, the TPE algorithm does not model the posterior distribution of the objective function. Instead, it treats the hyperparameters $\boldsymbol{\theta}$ as random variables, and splits the hyperparameter space into two regions based on the probability density of the hyperparameters. One region contains hyperparameters for which objective function is below a certain threshold $l(\boldsymbol{\theta})$, while the other contains the hyperparameters for which the objective function is above the threshold $g(\boldsymbol{\theta})$. Thus $l(\boldsymbol{\theta})$ contains the "good" hyperparameters and $g(\boldsymbol{\theta})$ contains the "bad" hyperparameters, since the goal is to minimize the objective function. $l(\boldsymbol{\theta})$ and $g(\boldsymbol{\theta})$ are thus probability densities of hyperparameters $\boldsymbol{\theta}$ conditioned on good and bad evaluations respectively. The acquisition function in the TPE algorithm is proportional to the ratio (Watanabe, 2023)

$$\frac{l(\boldsymbol{\theta})}{g(\boldsymbol{\theta})} \ ,$$

which effectively guides the search toward regions of the hyperparameter space where good results are more likely while balancing exploration and exploitation. The specific implementation details are out of the scope of this thesis, we refer to Bergstra et al. (2011). TPE requires a predefined search space in which it will search for the optimal hyperparameters. The hyperparameter space for each hyperparameter is given in Table 2. Note the choice of fixing n_estimators at 600. The choice of fixing n_estimators is made since this hyperparameter is very correlated with the hyperparameter learning_rate, as per the analysis in Section 2.2.5. Fixing n_estimators will allow the model to exploit and explore the learning_rate more. Each XGBoost model fitted in this thesis uses the Optuna package with 20 trials (total evaluations of the objective function) for hyperparameter tuning.

Hyperparameter	Search space
n_estimators	600
learning_rate	$[\ln 10^{-3},\ \ln 0.1]$
max_depth	$[4,\ 14]$
Subsample	$[0.05,\ 1.0]$
Colsample_bytree	$[0.05,\ 1.0]$
min_child_weight	$[2,\ 25]$

Table 2: Search spaces used in Optuna hyperparameter tuning for the XGBoost model.

3.2.2 Genomic prediction with Bayesian animal model

The animal model is one of the traditional ways to perform genomic prediction. In order to assess the performance of the XGBoost model, we will in this thesis fit an animal model to act as a reference. For the XGBoost model, we performed a pre-processing of the phenotype in order to remove noise from factors such as environmental effects. The animal model, which is a LMM model, has the breeding value $\boldsymbol{a}$ explicitly included as a random effect, such that any pre-processing on the phenotype is not necessary. We will in this thesis fit an animal model on the form

$$\boldsymbol{y} = \mu\mathbf{1}_n + \mathbf{X}\boldsymbol{\beta} + \boldsymbol{a} + \mathbf{D}\boldsymbol{r} + \boldsymbol{\epsilon}\ , \tag{16}$$

where $\boldsymbol{y}$ denotes the phenotype of either mass or wing, and where we assume the breeding values $\boldsymbol{a}$ follows the Gaussian distribution, $\boldsymbol{a} \sim N(\mathbf{0}, \mathbf{G}V_A)$, with $\mathbf{G}$ being the genomic relationship matrix derived by method 1 as described in Section 2.1.4. μ denotes the population mean and $\boldsymbol{\beta}$ and $\boldsymbol{r} \sim N(\mathbf{0}, \mathbf{V}_r)$ denote the fixed and random effects respectively, with their respective design matrices $\mathbf{X}$ and $\mathbf{D}$. Finally, $\boldsymbol{\epsilon} \sim N(\mathbf{0}, \boldsymbol{I}_n V_e)$ represents the independent residual errors. The genomic animal model will be fitted using variables Sex, FGRM, Month, Age, Outer, Other, Hatch island, Hatch year and Ringnr from the house sparrow variables in Table 1.

A Bayesian approach can be used to estimate the posterior probability distribution $p(\boldsymbol{\theta}|\boldsymbol{y})$ from a LMM, where $\boldsymbol{\theta}$ denotes the hyperparameters of the LMM and $\boldsymbol{y}$ denotes the response. In this approach, the posterior probability distribution obtained can provide estimates for statistics such as the median of the posterior distribution. The LMM describing the animal model in equation (16) can be expressed as a *latent Gaussian model* (LGM Martino and Riebler, 2019). For LGMs, there exists a specific Bayesian method called INLA which provides approximations for the posterior distribution (Rue et al., 2009; Martino and Riebler, 2019). In this thesis, we will use INLA to fit a *Bayesian animal model*, giving us approximations for the posterior distribution $p(\boldsymbol{a}|\boldsymbol{y})$. Taking the median of this distribution gives us an estimate for the breeding values called *genomic estimated breeding values* (GEBV). We want to show that the genomic animal model can be expressed as a LGM, which in general from is written as (Martino and Riebler, 2019)

$$\begin{aligned} \boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta} &\sim \prod_i p(y_i|\delta_i, \boldsymbol{\theta})\ , \\ \boldsymbol{x}|\boldsymbol{\theta} &\sim N(\mathbf{0}, \mathbf{Q}^{-1}(\boldsymbol{\theta}))\ , \\ \boldsymbol{\theta} &\sim p(\boldsymbol{\theta})\ . \end{aligned}$$

Here, $\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta}$ represents the likelihood, where $\boldsymbol{x}$ is the joint distribution of all parameters in the linear predictor $\boldsymbol{\delta}$. $\boldsymbol{y}$ is the (observed) response and $\boldsymbol{\theta}$ are the hyperparameters. Further, $\boldsymbol{x}|\boldsymbol{\theta}$ represents the latent field with precision matrix (*i.e.* inverse covariance matrix) $\mathbf{Q}(\boldsymbol{\theta})$, and $p(\boldsymbol{\theta})$ represents the hyperprior probability distribution. By putting a Gaussian prior on the fixed effects $\boldsymbol{\beta}$, the genomic animal model in equation (16) can be expressed as a LGM with variables $\boldsymbol{x} = (\boldsymbol{\beta}, \boldsymbol{a}, \boldsymbol{r})^T$, $\boldsymbol{\theta} = (V_A, V_e, \mathbf{V}_r)$ and linear predictor $\boldsymbol{\delta} = \mu\mathbf{1}_n + \mathbf{X}\boldsymbol{\beta} + \boldsymbol{a} + \mathbf{D}\boldsymbol{r}$. Thus, we can compute the GEBV using INLA by deriving the approximate sample distribution $\tilde{p}(\boldsymbol{a}|\boldsymbol{y})$ and taking the median. The exact INLA procedure is out of the scope of this project, we refer to Rue et al. (2009). In this thesis we fitted the Bayesian animal model using R-package *R-INLA*.

3.2.3 Measuring accuracy of prediction models

So far we have described the method of predicting the processed phenotype y^* using the XGBoost method and computing GEBV using the Bayesian animal model. In order to compare the prediction accuracy of the methods, we need a fair accuracy measure. This section will describe the accuracy measure of computing the *Pearson correlation* between the mean phenotype $\bar{y}_i$ and prediction of genetic component $\hat{y}^*$. The mean phenotype is used to account for repeat measurements of individuals. We describe the mean phenotype for individual i as $\bar{y}_i = \frac{1}{n_i}\sum_{j=1}^{n_i} y_{ij}$, where n_i is the number of repeats for individual i and y_{ij} is the observation j for individual i. For the Bayesian animal model, the prediction $\hat{y}^*$ is the GEBV, while for the XGBoost model, the prediction $\hat{y}^*$ is the prediction of the processed phenotype. We thus have the accuracy measure

$$\text{Corr}(\hat{\boldsymbol{y}}^*, \bar{\boldsymbol{y}}) \ ,$$

where the Pearson correlation is defined as

$$\text{Corr}(\boldsymbol{x}, \boldsymbol{y}) := \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \ .$$

In order to assess the performance of the algorithms based on the accuracy measured by the Pearson correlations, we use 10-fold cross-validation (CV) for both the Bayesian animal model and the XGBoost model. In 10-fold CV, we split the whole dataset into training and testing sets 10 times, where each instance of splitting is referred to as a fold. In each fold, 10% of the data is assigned to the testing set, and the remaining 90% is assigned to the training set. Each individual in the dataset is included in the testing set of exactly one fold, ensuring that the entire dataset is used for testing. In addition, the partitioning of the data into the 10 test sets is done at random when creating the folds, ensuring that we do not pick up a bias from the ordering of the data. For each fold, a model is trained on the training set and tested using the testing set, giving us 10 different test results. The CV procedure thus allows us to evaluate the accuracy and consistency of the methods by applying them on diverse test sets while utilizing the whole dataset. An illustration of the 10-fold CV procedure is shown in Figure 5. For the Bayesian animal model, the 10-fold CV procedure produces 10 Pearson correlation measures, one for each test set. We visualize the result in a box-plot, which provides a picture of the accuracy and consistency of the method. We perform this procedure separately for each trait mass and wing.

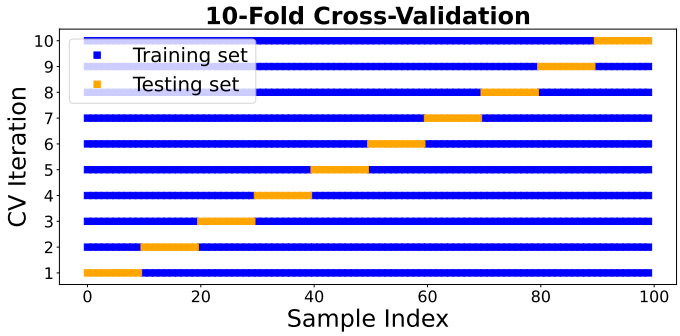


Figure 5: Partitioning of dataset in 10-fold cross-validation technique used on Bayesian animal model for calculating GEBV.

The XGBoost model requires hyperparameter tuning to optimize performance, as explained in Section 2.2.5. When using CV on a model that requires hyperparameter tuning, a procedure called *nested CV* is typically used. As we recall, the goal in hyperparameter tuning is to find the hyperparameters that minimize the objective function. The objective function, such as that of XGBoost given in equation (7), contains a loss function $L(\cdot)$ that is the chosen measure of error between the predicted value $\hat{y}$ and true value y. In other words, to tune the hyperparameters, we need to set aside yet another part of the dataset to act as a second "testing set", where this testing set, referred to as the *validation set*, acts as an independent testing set in the validation of hyperparameters. The whole testing procedure can be described as follows. Prior to fitting the model, we divide the

training set into a training set and a validation set. Then, we find the best hyperparameters by fitting a model on the training set and choosing the hyperparameters that minimize the objective value in the independent validation set. Having obtained the hyperparameters, we fit a new model using both the training set and validation set in the training. Finally, we can test the accuracy of the model by using the independent test set. In this whole process, we fit the model twice, first when validating the hyperparameters, then when testing the model accuracy. This is where the idea of nested CV comes in. In nested CV, we generate 10 folds, where each fold divides the dataset into training and testing sets, just like in regular 10-fold CV. In each fold, before fitting the model on the training data, we run a new 10-fold (or a general k-fold) CV on the training data, where each fold in the new CV divides the training data into training and validation sets. We obtain 10 sets of hyperparameters, one from each validation set, from which we choose the one giving the smallest objective value. We can now fit a model on the training data (including the validation set), with the chosen hyperparameters, before testing on the testing set.

The nested CV procedure requires us to do hyperparameter tuning $10 \times 10 = 100$ times. The Bayesian hyperparameter tuning is computationally expensive. To address this, we use a simplified "custom CV" procedure. Here, we do not perform CV for the hyperparameter tuning. Instead, for each fold, we divide the dataset into independent training, validation and testing sets in proportions of 80% 10% and 10% respectively. Tuning is only done once in each fold. For each fold, we do:

1. Hyperparameters are tuned using the training and validation subsets.
2. The model is trained on both the training and validation subsets using the obtained hyperparameters.
3. Predictions $\hat{y}^*$ are made for the individuals in the testing set, and Pearson correlation is computed.

This approach provides 10 different accuracy measures, one for each testing set. An illustration of the custom CV procedure can be seen in Figure 6. We ensure that we use the same testing sets for the Bayesian animal model and XGBoost, such that we minimize the variance between the two models due to differences in testing data. The resulting 10 Pearson correlation values for each model and for each trait can be visualized using a box-plot, where the median accuracy and variation in accuracy score can be compared.

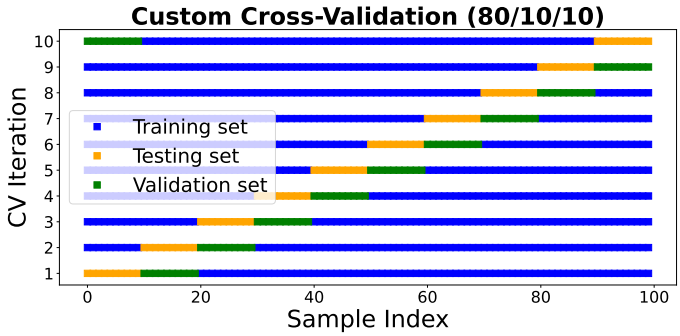


Figure 6: Partitioning of dataset in custom 10-fold cross-validation technique used in the fitting of XGBoost models used in prediction of genetic contribution and calculation of SHAP values.

3.3 Methods of inference with GWAS and SHAP

In addition to performing genomic prediction, this thesis explores methods of inference in genomic studies. A method within GWAS called *Genome-wide Efficient Mixed Model Association algorithm* (GEMMA Zhou and Stephens, 2012) will be implemented in order to assess which SNPs are associated with the traits mass and wing. In addition, we will implement a method of computing

SHAP values that explain the predictions of the XGBoost model for both traits. Both methods will be applied on the 70k dataset.

We apply GEMMA by fitting a univariate LMM, where we include the random intercept on the identity of individuals in order to factor in the correlation between individuals due to relatedness. Assuming we have $i = 1, ..., n$ individuals and $j = 1, ..., m$ SNPs, the univariate LMM for SNP j is given in form

$$\boldsymbol{y} = \mathbf{1}_n \mu + \boldsymbol{M}_j \beta_j + \boldsymbol{\rho} + \boldsymbol{\epsilon} \ ,$$

where $\boldsymbol{y}$ is the $(n \times 1)$ vector of phenotypes, μ is the population mean phenotype, $\boldsymbol{\rho}$ is the random intercept on the identity and $\boldsymbol{\epsilon}$ is the independent random vector of residual errors. Moreover, $\boldsymbol{M}_j$ is the $(n \times 1)$ vector of SNP genotypes for SNP j and β_j is the effect size of SNP j. All the random effects are assumed to be multivariate normally distributed with zero mean, and $\boldsymbol{\rho}$ is assumed to have covariance matrix given by the genomic relatedness matrix. Using GEMMA, we then test the hypothesis $H_0 : \beta_j = 0$ against $H_1 : \beta_j \neq 0$ for each SNP in turn, using one of the three test statistics Wald, likelihood ratio or score (Zhou and Stephens, 2012). The corresponding m p-values for each SNP can then be visualized in a Manhattan plot. When doing multiple hypothesis testing, it is common to use a stricter threshold for significance in order to decrease the number of false positives. We will in this thesis use the Bonferroni threshold (α_b), which scales the significance threshold used in a single hypothesis test, typically $\alpha = 0.05$, by the number of tests, in this case m, such that

$$\alpha_b = \frac{\alpha}{m} \ .$$

In order to draw inference from the XGBoost model and explain its predictions, we utilize the Python package *shap* to generate SHAP values for each prediction of the XGBoost model in the 70k dataset. To generate the predictions for every individual in the 70k dataset, we utilize the custom CV procedure. The process can be described the following way. Let $\mathcal{D}^{(k)} = \{\mathcal{D}^{(k)}_{train}, \mathcal{D}^{(k)}_{test}\},\ k = 1, ..., 10$ be the dataset with the partition in fold k. The validation set is in this case included in the training set. We further define the training set of fold k as

$$\mathcal{D}^{(k)}_{train} = \{y^{(k)}_i, \boldsymbol{x}^{(k)}_i\}_{i=1}^{n^{(k)}_{train}} \ ,$$

and the testing set of fold k as

$$\mathcal{D}^{(k)}_{test} = \{y^{(k),*}_i, \boldsymbol{x}^{(k),*}_i\}_{i=1}^{n^{(k)}_{test}} \ .$$

Here, $n^{(k)}_{train}$ and $n^{(k)}_{test}$ denote the training and testing sample sizes of fold k, respectively. Further, let $p^{(k)}$ denote the number of covariates used in fold k, such that $\boldsymbol{x}^{(k)}_i, \boldsymbol{x}^{(k),*}_i \in \mathbb{R}^{p^{(k)}}\ \forall i$. From these definitions, the process of computing the SHAP values for each XGBoost prediction in the 70k dataset is given in Algorithm 2.

Algorithm 2 SHAP value computation process

For $k = 1, ..., 10$:
  Train XGBoost model f_k on $\mathcal{D}^{(k)}_{train}$
  For $i = 1, ..., n^{(k)}_{test}$:
    $\hat{y}^{(k)}_i = f_k(\boldsymbol{x}^{(k),*}_i) = E[f_k(\mathbf{x}^{(k)}) \mid \mathbf{x}^{(k)} \in \mathcal{D}^{(k)}_{train}] + \sum_{j=1}^{p^{(k)}} \phi^{(k)}_{ij}$
  end for
end for
Return SHAP values $\phi^{(k)}_{ij},\ k = 1, ..., 10,\ i = 1, ..., n^{(k)}_{test},\ j = 1, ..., p^{(k)}$.

We recall from Section 2.3.2 that the SHAP values generate local explanations, meaning they explain a single prediction. In this thesis we are interested in global explanations, meaning we want insight into the average behavior of the model, or more precisely, the average contribution of the SNPs on the predictions. One way to generate an approximate global model from SHAP values is to take the absolute value of every SHAP value, then for each SNP_j, $j = 1, ..., p$, sum

the j'th SHAP value from every prediction, and divide them by the total number of predictions $|\mathcal{D}| = n$. Formally, the mean |SHAP| value is defined as

$$\text{mean } |\text{SHAP}|_j := \frac{1}{n} \sum_{k=1}^{10} \sum_{i=1}^{n_{test}^{(k)}} |\phi_{ij}^{(k)}|, \ j = 1, ..., p \ .$$

Each mean |SHAP| value now gives an approximation for the average contribution of a SNP to the prediction. In other words, it is a measure of feature importance. However, we have to be careful in interpreting the mean |SHAP| values. From Algorithm 2, we see that each SHAP value $\phi_{ij}^{(k)}$ depends on the fold-specific global prediction mean $E[f_k(\mathbf{x}^{(k)}) \mid \mathbf{x}^{(k)} \in \mathcal{D}_{train}^{(k)}]$ and prediction from fold-specific XGBoost model f_k. Thus, there is a bias introduced on the mean |SHAP| values from the fold effects. Because of this, the mean |SHAP| values lose some of the inherent "fairness" from the properties of Shapley values. Nevertheless, the mean |SHAP| value gives a reasonable measure of feature importance.

The mean |SHAP| values can be visualized similarly to the p-values in a Manhattan-style plot, where the SNPs are ordered in their chromosome and bp position along the x-axis like in GWAS, and where the y-axis shows the mean |SHAP| values. A method of visually comparing the two methods is to plot the mean |SHAP| values against the p-values from the GWAS and then fit a regression line between them. This can give us an indication on whether the features that are important for predictions in the XGBoost model are the same ones that generate small p-values for the GWAS. We do not necessarily expect the outliers in p-values and mean |SHAP| to overlap given the different characteristics of the methods. We assume that the mean |SHAP| values, generated from the predictions of the XGBoost model, will favor the SNPs that have interaction effects, such as epistatic effects or dominance effects. On the other hand, the p-values of GWAS are generated by a LMM, thus we expect the linear additive genetic effects to be the driving factor behind (true) outliers in p-values.

SHAP values give us insight into the effect size of a SNP, since they represent the average contribution of the SNP to the deviation of a prediction from the global prediction mean. This property allows us to use the mean |SHAP| values as a way to visualize how the importances of the SNPs are distributed in the predictions of the XGBoost model. This will give us insight into the genetic architecture of the trait, for example whether the trait is polygenic or not. This can be done by normalizing and ordering the mean |SHAP| values in descending order, then cumulatively adding them from the first to the last. If we plot this cumulation against the number of SNPs, we can inspect how many SNPs are needed for the XGBoost model to reach a certain proportion of the total predicting power. We refer to the resulting figure as a *cumulative feature importance plot*.

To summarize the inference methods used in this thesis, we provide Table 3, which compares some of the attributes of SHAP values, mean |SHAP| values and GWAS in assessing which SNPs are associated with a trait.

	Easy to interpret	**Estimates feature importance**	**Can be used to mislead**	**Easily compared across experiments**
SHAP values	Yes	Yes	No	Yes
mean |SHAP|	Yes	Yes	Yes	Yes
GWAS	No	No	Yes	No

Table 3: Comparison of some attributes for SHAP values, mean |SHAP| and GWAS.

4 Results

4.1 Prediction of genetic component with XGBoost and Bayesian animal model

We observe that the GEBV computed by Bayesian animal model generally outperforms the XGBoost counterpart in terms of the correlation measure we have used (Figure 7). For the mass trait, we see that XGBoost has a median accuracy of around 0.27, while the median for the GEBV accuracy is around 0.28. Moreover, the variation is much smaller in the GEBV prediction accuracy than for the XGBoost model when looking at the trait mass. Interestingly, we see a big outlier for both the GEBV and XGBoost accuracy at around 0.16 and 0.09 for the respective model accuracies. Since we have used the same test sets for both models, this result is likely from some test set that is relatively genetically differentiated from the training set that was used, causing both models to perform poorly. We see that the accuracy of GEBV is better than the accuracy of the XGBoost model for this test set also.

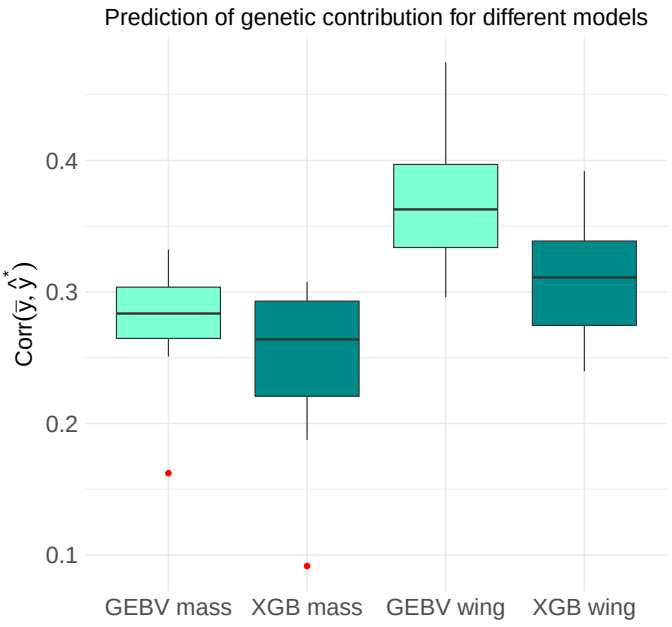
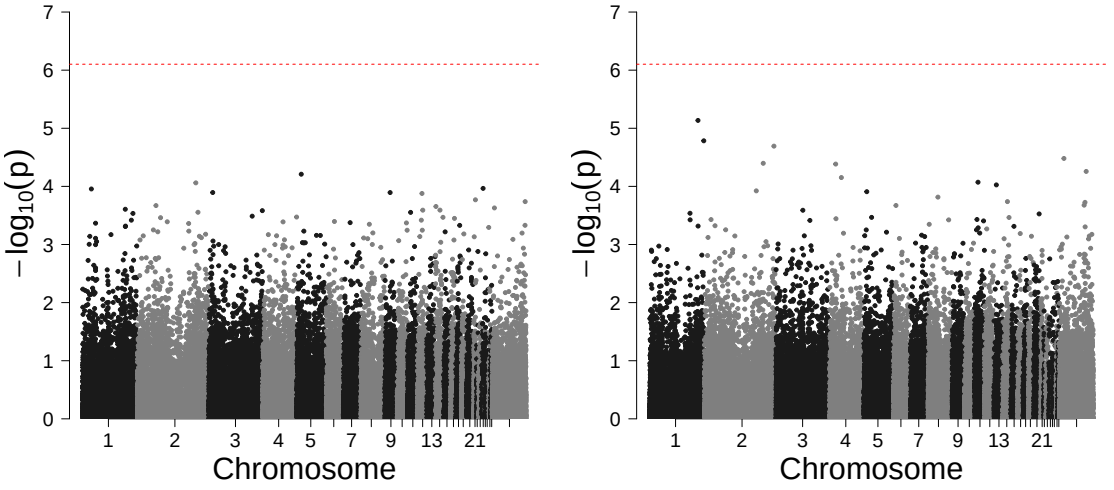


Figure 7: Box plot of Pearson correlation between mean phenotype and predicted genetic component for XGBoost (XGB) and Bayesian genomic animal model (GEBV) for phenotypes of mass and wing. All models are trained and tested on the same train/test set using the 180k dataset.

For the wing trait, both methods generate a higher correlation on average than for the mass trait, which is consistent with previous findings (Aspheim et al., 2024). The median accuracy of XGBoost is around 0.31 while the median accuracy of GEBV is at around 0.36. Interestingly, the uncertainty in accuracy increased considerably in the GEBV while the uncertainty in accuracy for the XGBoost model is relatively similar comparing across the traits. For the wing trait, the variation in accuracy ends up being approximately the same for the two models. However, the GEBV clearly tends to have a slightly better accuracy than the XGBoost model. Overall, the relative accuracy seems to be consistent between the models, which likely is the result of using the same test sets for the models. The GEBV seems to have a slightly better accuracy in all test sets, however this would need to be verified by checking the accuracy "set-by-set" for the different test sets.

4.2 Inference with GWAS and SHAP

From the Manhattan plot of trait mass (Figure 8a), we see the general trend that all chromosomes tend to contribute to the phenotype to some degree. There is also no single chromosome that stands out in terms of outliers in the Manhattan plot. There are no SNPs that meet the significance level given by the Bonferroni threshold for trait mass. Even the smallest p-value for mass in the magnitude of a little less than 10^{-4} on chromosome 5 is quite far away from the Bonferroni threshold at magnitude of around 10^{-6}. The outliers in the Manhattan plot for mass tend to be isolated, which could indicate that those outliers do not lie close to some important allele, but are instead false "positive" results. However, there seems to be a cluster of outliers in chromosome 2, 12, and 29 that could indicate that these areas of the genome are important for the trait mass.



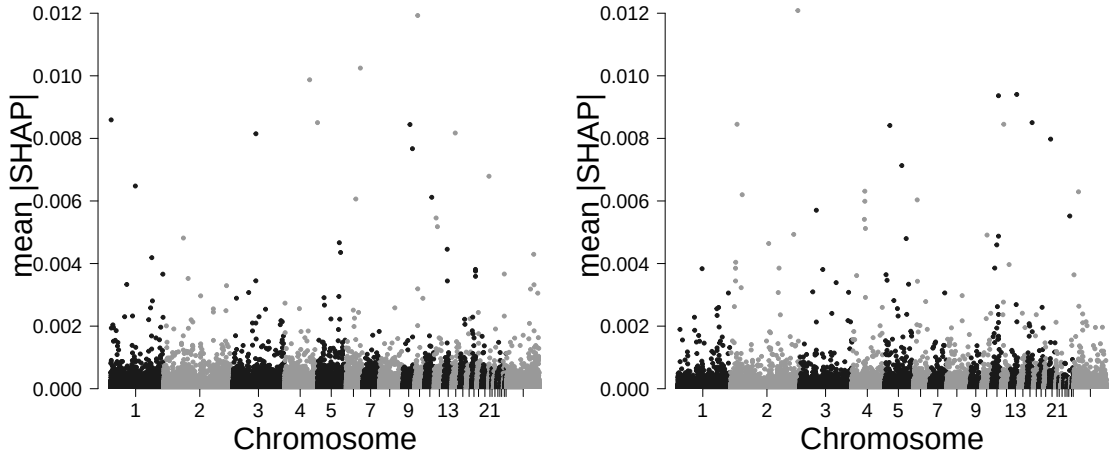
(a) Mass GWAS

(b) Wing GWAS

Figure 8: Manhattan plot of GWAS results from GEMMA model for traits mass and wing. Both models are fitted on the 70k data set. Each dot represents a SNP, and x-axis shows position in chromosome and bp.

For the Manhattan plot of trait wing (Figure 8b), we see a similar trend in that many chromosomes tend to contribute to the phenotype. There are also no p-values that meet the Bonferroni threshold of significance for the wing trait. However, the smallest p-values for the wing trait are to be closer to the Bonferroni threshold than compared to the smallest p-values for the mass trait. The smallest p-value for the wing trait at chromosome 1 is at a magnitude of somewhere between 10^{-5} and 10^{-6}, which is not far from reaching significance. Similarly to the mass trait, the outliers of the wing trait tend to be isolated and not part of a cluster of outlier SNPs. However, the two smallest p-values are both on chromosome 1 with a high bp number. This could indicate that there is an allele in this part of the chromosome that has an important affect on the wing trait. Interestingly, this part of the genome has a cluster of relatively small p-values for the mass trait also. The same pattern can be seen for high bp numbers in chromosome 2. This could indicate that there is an allele that affects both traits in these parts of the genome. Chromosome 29 also seems to have a cluster of SNPs that reach a small p-value for the wing trait, which makes this part of the genome interesting to investigate more.

Figure 9 shows the results from the mean |SHAP| values, which we for the sake of convenience will simply call the SHAP values for the rest of this section, in a Manhattan-style plot. For the trait mass, we see that some chromosomes clearly have a smaller contribution than others (Figure 9a). For example, chromosome 2 is the largest chromosome in terms of number of bp, however the largest SHAP value is only at around 0.005. In general, most SNPs have a SHAP value of less than 0.002. The largest SHAP value for trait mass is found at chromosome 10 with a SHAP value of around 0.012, which indicates that this SNP has a large contribution magnitude and is important



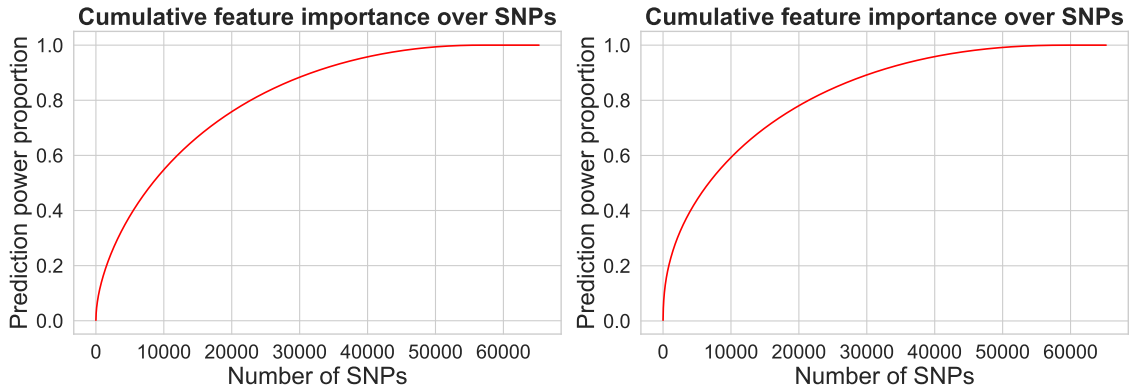
(a) Mass SHAP

(b) Wing SHAP

Figure 9: Manhattan-style plot of mean |SHAP| values fitted using the TreeSHAP algorithm on XGBoost model using 70k dataset for traits mass and wing. Each dot represents a SNP, and x-axis shows position in chromosome and bp.

for the predictions in the XGBoost model. The second and third largest SHAP values are found at chromosome 8 and 4 respectively. All of the three largest SHAP values are relatively isolated and not part of a cluster of outliers. For the wing trait, the highest SHAP value of approximately 0.012 is found in chromosome 2 (Figure 9b). This SNP is also relatively isolated, and not part of any cluster of SNPs. However, one such cluster of SNPs can be found on chromosome 4 for the wing trait.

We can see that the cumulative feature importance plot for wing trait has a more steep curve for the first 10 000 SNPs compared to that of mass trait (Figure 10). This trend is even stronger for the first 1000 or so most important SNPs. The 10 000 most important SNPs for the wing trait account for approximately 60% of the XGBoost models prediction power for this trait, while the mass trait needs approximately 13 000 SNPs to reach the same predictive power. To explain 80% of the predictive power of the XGBoost model, both traits need somewhere between 20 000 and 25 000 of the most important SNPs. We generally see that the SNPs that are important in SHAP



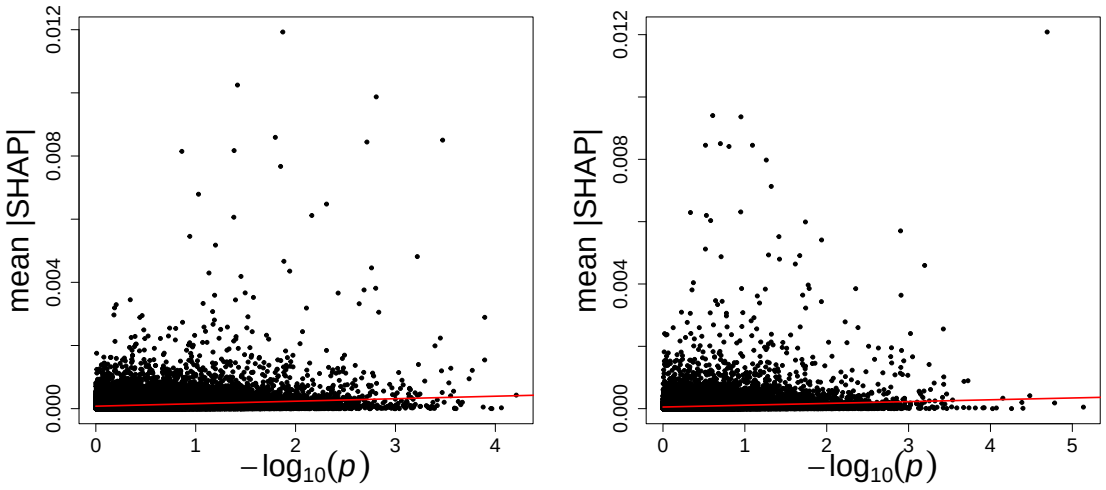
(a) Cumulative feature importance mass

(b) Cumulative feature importance wing

Figure 10: Cumulative feature importance derived from mean |SHAP| values from XGBoost model on the 70k data set for traits mass and wing.

values are not the same as the outlier p-values in GWAS. This is not surprising given the different characteristics of the Bayesian animal model and the XGBoost model. The XGBoost model, using

regression trees as weak learners, should be able to pick up interaction (epistatic) effects, while the Bayesian animal model should pick up the linear (additive) effect. The weak correlation between SNPs deemed important in the different models can also be seen (Figure 11). From the figure, we see that the regression line shows weak correlation in both traits. One interesting observation is the observation with the highest SHAP value for wing trait. This observation has a relatively high p-value in GWAS also, indicating that this SNP might have an additive effect and also an interaction effect, or alternatively that the SHAP value was able to pick up the additive effect.



(a) Scatterplot mass

(b) Scatterplot wing

Figure 11: Scatterplot of mean |SHAP| values and p-values from GEMMA model with regression line using all SNPs from the 70k data set for both traits. SHAP values are explained predictions from the XGBoost model fitted on the 70k data set. Each dot represents a SNP, and x-axis shows position in chromosome and bp.

5 Discussion and conclusions

We have in this thesis introduced the XGBoost model as an alternative to the traditional animal model and GEBV in genomic prediction. We have also introduced SHAP values as an alternative to GWAS in finding which SNPs are associated with a phenotype. The methods have been tested on data from a wild population of house sparrows on the Helgeland island system in northern Norway. The results indicate that the traditional animal model outperforms XGBoost when it comes to accuracy of predicting the genetic component of the phenotype and that SHAP values represent a good alternative method of finding which SNPs are associated with a phenotype. The genomic prediction methods was performed on two traits, body mass and wing lengths. The assessment of SNP association with the phenotypes body mass and wing length was also performed. The dataset used for prediction was characterized with being rich in genetic material but smaller sample size compared to the dataset used in the SNP association analysis.

The results indicate that the GEBVs perform better than predictions from XGBoost in terms of accuracy of prediction. It seems like this observation holds true also for test sets that are relatively genetically differentiated from the training set, leading to outliers in predictions for both models. This is an interesting result given the choice of MAE as loss function in XGBoost in order to handle outliers better. Even though XGBoost uses the MAE to be more robust in terms of outliers, the accuracy is significantly worse compared to the Bayesian animal model for the differentiated test sets.

Both models generate a higher correlation on average for the wing trait than for the mass trait,

which is consistent with previous findings (Aspheim et al., 2024). For the trait wing, the Bayesian animal model has a higher variation in prediction accuracy when compared to body mass, however the same is not true for the XGBoost model, which seems to have a stable variation in prediction accuracy between the traits. One explanation for this observation could be the different characteristics of the XGBoost model and the Bayesian animal model. Whereas the XGBoost model is good at capturing interactions between SNPs in form of epistatic and dominance effects, the Bayesian animal model captures the linear additive genetic effects. Differences in variation of the additive genetic effects between traits would then affect the accuracy of the Bayesian animal model more than for the XGBoost model. Overall, the Bayesian animal model seems to consistently outperform the XGBoost model in terms of the correlation measure we have used for both traits and in any test set, however this would have to be confirmed by comparing the results fold-by-fold.

One of the reasons the XGBoost model performed poorly relative to expectations could be that we did not make the correct choices in the model configurations. For example, the XGBoost model has a number of other hyperparameters that we could have chosen to optimize instead of (or in addition to) the ones chosen, but that we instead decided to keep at default values. Another potential source of poor performance could be the choice of number of objective evaluations to use in the Bayesian optimization framework. In this thesis, we decided on 20 evaluations for each model, where the choice was made considering the trade off in accuracy and time complexity. It might be the case that 20 evaluations is not enough in order to provide a good set of hyperparameters of the XGBoost model. In relation to the hyperparameters, the choice of using the custom CV approach instead of the standard nested CV approach may also have affect the result. Using the standard nested CV approach likely leads to better set of hyperparameters, but this comes at the cost of time complexity. Not finding the best set of hyperparameters could affect the result since the XGBoost model is dependent on tuning the hyperparameters in order to give predictions with high accuracy.

Another source of inaccuracy for the XGBoost model could be the choice of loss function. The choice of MAE as loss function was made in order to be more robust in terms of outliers, however it could be the case that a different loss function would perform better. The observation that the XGBoost model performed considerably worse than the Bayesian animal model even for (negative) outliers in accuracy strengthens the belief that MAE was the wrong choice of loss function. One hypothesis is that the XGBoost model is limited by using regression trees as weak learners. The regression trees are not good at capturing linear relationships, and thus the XGBoost model will not capture the additive genetic effects as well as the Bayesian animal model. In addition, another potential issue with using regression trees as weak learners is that they cannot extrapolate beyond their training data, since they fit constant weights in the leaf nodes. A further potential source of inaccuracy is added due to the two-step procedure of pre-processing the phenotype before fitting the XGBoost model. Other studies have shown that V_A may be heavily underestimated using this procedure (Aspheim et al., 2024). A potential expansion of the work in this thesis could be to try an XGBoost model with a weak learner that better captures the additive genetic effects. New tests will have to be made in order to see if the XGBoost model can provide even better prediction accuracy than the Bayesian animal model using different configurations.

We found that there were no SNPs significantly associated with either of the traits mass and wing from the analysis in GWAS. This was the case even though we used a dataset with higher sample size and fewer SNPs than the dataset used for the predictions. This result is consistent with the "missing heritability" observation in previous studies using GWAS, where it was found that a relatively big sample size was needed in order to reduce the uncertainty to a level where the p-values reach the significance threshold (Yang et al., 2010; Yengo et al., 2022). On two of the chromosomes we found that there were clusters of p-values that were relatively high for both traits. This result could indicate that there is an allele that affects both traits in these parts of the genome. Intuitively, it could be the case that individuals that have longer wings tend to have larger body mass, which would make the traits correlated. This is something to keep in mind when interpreting these results.

In general, the results from the GWAS analysis show that many chromosomes tend to contribute to the phenotype for both traits. The results from the analysis of the mean |SHAP| values indicate that some chromosomes seem to have a smaller contribution than others. We found that fewer SNPs were needed to reach a predictive power of 60% in the XGBoost model for the trait wing

than for the trait mass, however both traits needed approximately 30 000 SNPs in order to reach a prediction power of 90%. The result indicates that both traits are likely to be polygenic, and that mass is more polygenic than wing. A similar study was conducted on Soay sheep, where they found that the estimate of heritability of several traits known to be polygenic, such as height, failed to improve when including more than approximately 20 000 SNPs (Bérénos et al., 2014). The results we have found are similar to the study on Soay sheep, as 20 000 SNPs have a predictive power of approximately 80% for both traits. The consistency between the results is interesting, as it could indicate that there is some universality in that many complex polygenic traits can be explained by a number of SNPs in the magnitude of 20 000. Relevant to this is also the study study of human height by Yengo et al. (2022), which showed by GWAS that approximately 12 000 significant SNPs account for approximately 40% of phenotypic variance in populations of European ancestry, when common SNPs are predicted to collectively explain 40-50% of phenotypic variation in human height. We also found that the SNPs with a small p-value from the analysis in GWAS are generally not the same ones that are deemed important by the mean |SHAP| values for both the traits. However, the SNP generating the highest mean |SHAP| value for the wing trait also generated a relatively small p-value also. The low correlation between the mean |SHAP| values and p-values is consistent with our prior expectation, given the different characteristics of the XGBoost model and the p-values generated with an assumption of linearity. The outlier in chromosome 2 in both p-value and mean |SHAP| value could indicate that there is SNP associated with the wing trait with an additive effect as well as an interaction effect.

We have found that the mean |SHAP| values provide an alternative to the p-values generated in GWAS when the aim is to find which SNPs are associated with the phenotype of interest. The SHAP values have a lot of desirable attributes that the p-values do not have. The SHAP values provide a fair way to do local interpretations, and even though some of the fairness is lost, we can also use them to provide global explanations by generating mean |SHAP| values. The mean |SHAP| values gives more insight into the genetic architecture of the trait of interest, and also provide more information about the effect size of a SNP than the p-values through the interpretation of mean |SHAP| values as the average contribution of a SNP in the deviation of a prediction from the prediction mean. This property also makes the SHAP values easier to compare across different experiments compared to p-values, as the p-values do not provide a measure of the effect size. The SHAP values also have an inherent fairness through its properties, which makes it difficult to use the SHAP values to mislead.

One caveat of SHAP values and by extension mean |SHAP| values is the procedure of using CV in order to get predictions for the XGBoost model. When different models are fitted with different training sets as in the CV approach, the SHAP values lose some of the inherent fairness since they are calculated relative to the prediction mean. Another potential problem of the SHAP values is that they are only as useful as the predictions they are trying to explain. The XGBoost model seems to not be able to capture the most relevant information from the genomic data, at least compared to the Bayesian animal model. In this sense, one could argue that GWAS would be a better alternative to the use of SHAP values in assessing the SNP association. Building on this, an interesting direction of further study would be to fit a machine learning model that better captures the linear effects. This way, we can provide SHAP values as explanations of a linear model, which can be interesting to compare with the results from GWAS.

We have in this thesis explored the method XGBoost as an alternative to the animal model in genomic prediction and SHAP values as an alternative to GWAS in assessing which SNPs are associated with a trait. In this thesis, we have not found a configuration of the XGBoost model that can outperform the traditional animal model, however the SHAP values provide an interesting alternative to GWAS. It remains to be seen whether XGBoost can be a viable alternative to the animal model in genomic prediction. SHAP represents an alternative method of finding which SNPs are associated with a phenotype with some advantageous properties that is not found in the traditional methods in GWAS.

References

Aas, Kjersti, Martin Jullum and Anders Løland (2021). 'Explaining individual predictions when features are dependent: More accurate approximations to Shapley values'. In: *Artificial Intelligence* 298. ISSN: 00043702. DOI: 10.1016/j.artint.2021.103502.

Akiba, Takuya et al. (2019). 'Optuna: A Next-generation Hyperparameter Optimization Framework'. In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Association for Computing Machinery, pp. 2623–2631. ISBN: 9781450362016. DOI: 10.1145/3292500.3330701.

Arenas, Sebastián et al. (2021). 'Evaluating the accuracy of genomic prediction for the management and conservation of relictual natural tree populations'. In: DOI: 10.1007/s11295-020-01489-1. URL: https://doi.org/10.1007/s11295-020-01489-1.

Ashraf, B et al. (2021). *Genomic prediction in the wild: A case study in Soay sheep*. English. Tech. rep. DOI: 10.1111/mec.16262.

Aspheim, Janne C. H. et al. (2024). *Bayesian marker-based principal component ridge regression – a flexible multipurpose framework for quantitative genetics in wild study systems*. DOI: 10.1101/2024.06.01.596874. URL: http://biorxiv.org/lookup/doi/10.1101/2024.06.01.596874.

Aulchenko, Yurii S (2011). *Analysis of Complex Disease Association Studies*. Ed. by Eleftheria Zeggini and Andrew Morris. Academic Press. ISBN: 978-0-12-375142-3. DOI: 10.1016/B978-0-12-375142-3.10009-4. URL: https://doi.org/10.1016/B978-0-12-375142-3.10009-4.

Bérénos, Camillo et al. (2014). 'Estimating quantitative genetic parameters in wild populations: A comparison of pedigree and genomic approaches'. In: *Molecular Ecology* 23.14, pp. 3434–3451. ISSN: 1365294X. DOI: 10.1111/mec.12827.

Berg, Jeremy J et al. (2019). 'Reduced signal for polygenic adaptation of height in UK Biobank'. In: 8. ISSN: 2050-084X. DOI: 10.7554/eLife.39725.001. URL: https://doi.org/10.7554/eLife.39725.

Bergstra, James et al. (2011). *Algorithms for Hyper-Parameter Optimization*. Tech. rep. URL: https://proceedings.neurips.cc/paper_files/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf.

Boehmke, Bradley and Brandon Greenwell (2019). *Hands-On Machine Learning with R*. 1st ed. Chapman and Hall/CRC. ISBN: 978-1138495685.

Bolastad, William (2009). *Understanding Computational Bayesian Statistics*. ISBN: 9780470046098. DOI: 10.1002/9780470567371.

Boubacar, Sidi et al. (2013). *Evaluation of approaches for estimating the accuracy of genomic prediction in plant breeding*. Tech. rep. URL: http://www.biomedcentral.com/1471-2164/14/860.

Bousquet, Olivier, Ulrike von Luxburg and Gunnar Rätsch (2003). *Advanced Lectures on Machine Learning*. English. Springer. ISBN: 978-3-540-28650-9. DOI: https://doi.org/10.1007/b100712. URL: https://link.springer.com/book/10.1007/b100712.

Brodie, Aharon, Johnathan Roy Azaria and Yanay Ofran (2016). 'How far from the SNP may the causative genes be?' In: *Nucleic Acids Research* 44.13, pp. 6046–6054. ISSN: 13624962. DOI: 10.1093/nar/gkw500.

Brookes, Anthony J. (1999). 'The essence of SNPs'. In: 234. DOI: https://doi.org/10.1016/S0378-1119(99)00219-X.

Bush, William S. and Jason H. Moore (2012). 'Chapter 11: Genome-Wide Association Studies'. In: *PLoS Computational Biology* 8.12. ISSN: 15537358. DOI: 10.1371/journal.pcbi.1002822.

Chen, Ding and John Chen (2017). *Monte-Carlo Simulation- Based Statistical Modeling*. Springer. ISBN: 978-981-10-3307-0. DOI: https://doi.org/10.1007/978-981-10-3307-0. URL: http://www.springer.com/series/13402.

Chen, Tianqi and Carlos Guestrin (2016). 'XGBoost: A scalable tree boosting system'. In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Vol. 13-17-August-2016. Association for Computing Machinery, pp. 785–794. ISBN: 9781450342322. DOI: 10.1145/2939672.2939785. URL: https://www.kdd.org/kdd2016/papers/files/rfp0697-chenAemb.pdf.

Configure XGBoost (2024). Date accessed: 09/12/2024. URL: https://xgboosting.com/configure-xgboost-regabsoluteerror-objective-mean-absolute-error/.

Eichler, Evan E. et al. (2010). *Missing heritability and strategies for finding the underlying causes of complex disease*. DOI: 10.1038/nrg2809.

Frazier, Peter I. (2018). 'A Tutorial on Bayesian Optimization'. In: URL: http://arxiv.org/abs/1807.02811.

Gill, Mitchell et al. (2022). 'Machine learning models outperform deep learning models, provide interpretation and facilitate feature selection for soybean trait prediction'. In: *BMC Plant Biology* 22.1. ISSN: 14712229. DOI: 10.1186/s12870-022-03559-z.

Goodman, Steven (2008). 'A Dirty Dozen: Twelve P-Value Misconceptions'. In: *Seminars in Hematology* 45.3, pp. 135–140. ISSN: 00371963. DOI: 10.1053/j.seminhematol.2008.04.003.

Hartl, Daniel L. and Jeffrey K. Conner (2004). *A Primer of Ecological Genetics*. English. Oxford University Press Inc. ISBN: 978-0878932023.

Hastie, Trevor, Robert Tibshirani and Jerome Friedman (2009). *The Elements of Statistical Learning*. Springer New York, NY. ISBN: 978-0-387-84857-0. DOI: https://doi.org/10.1007/978-0-387-84858-7.

Head, Megan L. et al. (2015). 'The Extent and Consequences of P-Hacking in Science'. In: *PLoS Biology* 13.3. ISSN: 15457885. DOI: 10.1371/journal.pbio.1002106.

Henderson, C R (1975). *Best Linear Unbiased Estimation and Prediction under a Selection Model*. Tech. rep. 2, pp. 423–447. URL: https://www.jstor.org/stable/2529430.

— (1988). 'Applications of Linear Models in Animal Breeding'. In: *Journal of Dairy Science*. URL: https://learnanimalbreeding.com/files/Henderson_1984.pdf.

Hickey, John M. et al. (2017). *Genomic prediction unifies animal and plant breeding programs to form platforms for biological discovery*. DOI: 10.1038/ng.3920.

Huang, Qingyang (2015). *Genetic Study of Complex Diseases in the Post-GWAS Era*. DOI: 10.1016/j.jgg.2015.02.001.

Ioannidis, John P.A. (2018). 'Why most published research findings are false'. In: *Getting to Good: Research Integrity in the Biomedical Sciences*. Springer International Publishing, pp. 2–8. ISBN: 9783319513584. DOI: 10.1371/journal.pmed.0020124.

James, Gareth et al. (2013). *An Introduction to Statistical Learning*. 1st ed. Springer. ISBN: 978-1461471370.

Jensen, Henrik, Rune Moe et al. (2013). 'Genetic variation and structure of house sparrow populations: Is there an island effect?' In: *Molecular Ecology* 22.7, pp. 1792–1805. ISSN: 09621083. DOI: 10.1111/mec.12226.

Jensen, Henrik, Ingelin Steinsland et al. (2008). 'Evolutionary dynamics of a sexual ornament in the house sparrow (Passer domesticus): The role of indirect selection within and between sexes'. In: *Evolution* 62.6, pp. 1275–1293. ISSN: 00143820. DOI: 10.1111/j.1558-5646.2008.00395.x.

Johnsen, Pål V. et al. (2021). 'A new method for exploring gene–gene and gene–environment interactions in GWAS with tree ensemble methods and SHAP values'. In: *BMC Bioinformatics* 22.1. ISSN: 14712105. DOI: 10.1186/s12859-021-04041-7.

Karki, Roshan et al. (2015). *Defining "mutation" and "polymorphism" in the era of personal genomics*. DOI: 10.1186/s12920-015-0115-z.

Kruuk, Loeske E.B. (2004). *Estimating genetic parameters in natural populations using the 'animal model'*. DOI: 10.1098/rstb.2003.1437.

Kuo, Frances Y and Ian H Sloan (2005). *Lifting the Curse of Dimensionality*. Tech. rep. URL: http://www.yaroslavvb.com/papers/kuo-lifting.pdf.

Kyrgiafini, Maria Anna et al. (2023). 'Gene-by-Sex Interactions: Genome-Wide Association Study Reveals Five SNPs Associated with Obesity and Overweight in a Male Population'. In: *Genes* 14.4. ISSN: 20734425. DOI: 10.3390/genes14040799.

Lazarevic, Aleksandar and Vipin Kumar (2005). *Feature Bagging for Outlier Detection*. Tech. rep. DOI: https://doi.org/10.1145/1081870.1081891. URL: https://dl.acm.org/doi/pdf/10.1145/1081870.1081891.

Lourenço, Vanda M et al. (2024). 'Genomic prediction using machine learning: a comparison of the performance of regularized regression, ensemble, instance-based and deep learning methods on synthetic and empirical data'. In: *BMC Genomics* 25.1. ISSN: 14712164. DOI: 10.1186/s12864-023-09933-x.

Lundberg, Scott M., Gabriel G. Erion and Su-In Lee (2018). 'Consistent Individualized Feature Attribution for Tree Ensembles'. In: DOI: https://doi.org/10.48550/arXiv.1802.03888. URL: http://arxiv.org/abs/1802.03888.

Marioni, Riccardo E. et al. (2018). 'GWAS on family history of Alzheimer's disease'. In: *Translational Psychiatry* 8.1. ISSN: 21583188. DOI: 10.1038/s41398-018-0150-6.

Martino, Sara and Andrea Riebler (2019). 'Integrated Nested Laplace Approximations (INLA)'. In: URL: http://arxiv.org/abs/1907.01248.

McGaugh, Suzanne E., Aaron J. Lorenz and Lex E. Flagel (2021). *The utility of genomic prediction models in evolutionary genetics*. DOI: 10.1098/rspb.2021.0693.

McKinney, B. A. and Nicholas M. Pajewski (2012). *Six degrees of epistasis: Statistical network models for GWAS*. DOI: 10.3389/fgene.2011.00109.

Meuwissen, T H E, B J Hayes and M E Goddard (2001). *Prediction of Total Genetic Value Using Genome-Wide Dense Marker Maps*. Tech. rep. DOI: 10.1093/genetics/157.4.1819. URL: https://doi.org/10.1093/genetics/157.4.1819.

Misztal, Ignacy, Daniela Lourenco and Andres Legarra (2020). 'Current status of genomic evaluation'. In: *Journal of Animal Science* 98.4. ISSN: 15253163. DOI: 10.1093/jas/skaa101.

Molnar, Christoph (2022). *A Guide for Making Black Box Models Explainable*. 2nd ed. URL: https://christophm.github.io/interpretable-ml-book.

Muff, Stefanie, Erlend B. Nilsen et al. (2022). *Rewriting results sections in the language of evidence*. DOI: 10.1016/j.tree.2021.10.009.

Muff, Stefanie, Alina K. Niskanen et al. (2019). 'Animal models with group-specific additive genetic variances: Extending genetic group models'. In: *Genetics Selection Evolution* 51.1. ISSN: 12979686. DOI: 10.1186/s12711-019-0449-7.

Murcray, Cassandra E., Juan Pablo Lewinger and W. James Gauderman (2009). 'Gene-environment interaction in genome-wide association studies'. In: *American Journal of Epidemiology* 169.2, pp. 219–226. ISSN: 00029262. DOI: 10.1093/aje/kwn353.

Nasteski, Vladimir (2017). 'An overview of the supervised machine learning methods'. In: *HORIZONS.B* 4, pp. 51–62. ISSN: 18578578. DOI: 10.20544/horizons.b.04.1.17.p05.

Oraguzie, Nnadozie C et al. (2008). *Association Mapping in Plants*. Springer. ISBN: 978-0387514925.

Pack Kaelbling, Leslie et al. (1996). *Reinforcement Learning: A Survey*. Tech. rep., pp. 237–285. DOI: https://doi.org/10.1613/jair.301. URL: https://www.jair.org/index.php/jair/article/view/10166/24110.

Pärn, Henrik et al. (2011). 'Spatial heterogeneity in the effects of climate and density-dependence on dispersal in a house sparrow metapopulation'. In: *Proceedings of the Royal Society B: Biological Sciences* 279.1726, pp. 144–152. ISSN: 14712970. DOI: 10.1098/rspb.2011.0673.

Pu, Fang, Jinsong Ren and Xiaogang Qu (2018). *Nucleobases, nucleosides, and nucleotides: Versatile biomolecules for generating functional nanomaterials*. DOI: 10.1039/c7cs00673j.

Rue, Håvard, Sara Martino and Nicolas Chopin (2009). *Approximate Bayesian inference for latent Gaussian models by using integrated nested Laplace approximations*. Tech. rep. 2, pp. 319–392. DOI: 10.1111/j.1467-9868.2008.00700.x.

Saatoglu, Dilan et al. (2021). 'Dispersal in a house sparrow metapopulation: An integrative case study of genetic assignment calibrated with ecological data and pedigree information'. In: *Molecular Ecology* 30.19, pp. 4740–4756. ISSN: 1365294X. DOI: 10.1111/mec.16083.

Sachidanandam, Ravi et al. (2001). *A map of human genome sequence variation containing 1.42 million single nucleotide polymorphisms*. Tech. rep. DOI: https://doi.org/10.1038/35057149.

Shapley, L. S. (1952). *A VALUE FOR n-PERSON GAMES*. Tech. rep. URL: https://www.rand.org/content/dam/rand/pubs/papers/2021/P295.pdf.

Sutton, Walter S (1903). *THE CHROMOSOMES IN HEREDITY*. Tech. rep. URL: https://www.journals.uchicago.edu/doi/pdf/10.2307/1535741.

Syvänen, Ann-Christine (2001). 'Accessing genetic variation: genotyping single nucleotide polymorphisms'. In: DOI: https://doi.org/10.1038/35103535.

Uffelmann, Emil et al. (2021). *Genome-wide association studies*. DOI: 10.1038/s43586-021-00056-9.

VanRaden, P. M. (2008). 'Efficient methods to compute genomic predictions'. In: *Journal of Dairy Science* 91.11, pp. 4414–4423. ISSN: 15253198. DOI: 10.3168/jds.2007-0980.

Visscher, Peter M. et al. (2012). *Five years of GWAS discovery*. DOI: 10.1016/j.ajhg.2011.11.029.

Walsh, Bruce and Michael Lynch (2018). *Evolution and Selection of Quantitative Traits*. Oxford University Press. ISBN: 978-0198830870.

Watanabe, Shuhei (2023). 'Tree-Structured Parzen Estimator: Understanding Its Algorithm Components and Their Roles for Better Empirical Performance'. In: URL: http://arxiv.org/abs/2304.11127.

White, Désirée and Montserrat Rabago-Smith (2011). *Genotype-phenotype associations and human eye color*. DOI: 10.1038/jhg.2010.126.

Wilson, Alastair J. et al. (2010). 'An ecologist's guide to the animal model'. In: *Journal of Animal Ecology* 79.1, pp. 13–26. ISSN: 00218790. DOI: 10.1111/j.1365-2656.2009.01639.x.

Wood, Andrew R. et al. (2014). 'Defining the role of common variation in the genomic and biological architecture of adult human height'. In: *Nature Genetics* 46.11, pp. 1173–1186. ISSN: 15461718. DOI: 10.1038/ng.3097.

XGBoost Parameters (2022). Date accessed: 09/12/2024. URL: https://xgboost.readthedocs.io/en/stable/parameter.html.

Yang, Jian et al. (2010). 'Common SNPs explain a large proportion of the heritability for human height'. In: *Nature Genetics* 42.7, pp. 565–569. ISSN: 10614036. DOI: 10.1038/ng.608.

Yengo, Loïc et al. (2022). 'A saturated map of common genetic variants associated with human height'. In: *Nature* 610.7933, pp. 704–712. ISSN: 0028-0836. DOI: 10.1038/s41586-022-05275-y.

Zhou, Xiang and Matthew Stephens (2012). *Software tool and univariate linear mixed models*. Tech. rep. URL: https://github.com/genetics-statistics/GEMMA/blob/master/doc/manual.pdf.

Zhou, Zhi Hua (2021). *Machine Learning*. Springer Nature, pp. 1–458. ISBN: 9789811519673. DOI: 10.1007/978-981-15-1967-3.