Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Mathematical Sciences

Didrik Lindløv Sand

Genomic Prediction in a Wild Population Using Gradient Boosting Algorithms

Master's thesis in Applied Physics and Mathematics
Supervisor: Stefanie Muff
June 2024

NTNU
Norwegian University of
Science and Technology

Didrik Lindløv Sand

Genomic Prediction in a Wild Population Using Gradient Boosting Algorithms

Master's thesis in Applied Physics and Mathematics
Supervisor: Stefanie Muff
June 2024

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Mathematical Sciences

NTNU
Norwegian University of
Science and Technology

Acknowledgements

This thesis concludes my five years of applied physics and mathematics studies at NTNU. First, I want to thank my supervisor, Stefanie Muff, at the Department of Mathematical Sciences, for insightful discussions and enthusiastic guidance on this thesis. Thank you for an introduction to an exciting field I knew very little about before starting, where I have been given the freedom to explore methodologies that I find engaging. I also want to thank Henrik Jensen at the Department of Biology for making the data used in this thesis available.

I thank my parents for their unconditional support throughout my entire educational journey, and my friends for making life outside school enjoyable. Finally, I want to thank my girlfriend Sara for her encouragement and support while writing this thesis.

Didrik Lindløv Sand
Trondheim, June 2024

Abstract

Genomic prediction is the cross-point between statistics and biology, where genetic data is used to predict the genetic contribution to an individual's phenotype. Genomic prediction is important in the field of genetic ecology as it allows ecologists to understand how genetics influence the phenotype, which increases their understanding of the processes related to evolution. In this thesis, we have examined how a class of models denoted gradient boosting can be used for genomic prediction of two traits, body mass and tarsus length, in a population of wild house sparrows. The wild house sparrow population is located off the Helgeland coast in northern Norway and has been under study since 1993. The genetic material is available through sequenced single nucleotide polymorphisms (SNPs). Gradient boosting models have been used for genomic prediction in animal and plant breeding, but never before in a wild population.

Gradient boosting is a flexible framework that builds an ensemble of any base learner. We have deployed different base learners and compared their performances. The base learners fitted were the binary regression tree, the elastic net, and the piecewise linear regression tree. The gradient boosting models were compared to a state-of-the-art Bayesian animal model. We say that the elsatic net and the Bayesian animal model are linear based models, while a binary regression tree and piecewise linear regression trees are tree-based. Finally, we used an explainable AI method denoted Shapley additive explanation (SHAP) values on a tree-based model to assess SNP importance in the model. The SHAP values were further compared to results from a univariate linear mixed genome-wide association study (GWAS).

Our results show that the gradient boosting models are competitive to the Bayesian animal model. However, the choice of trait to predict determines which base learner is the best choice. When body mass was used as the response variable, the tree-based models performed slightly better than the linear-based boosting models and the animal model. On the other hand, when tarsus length was used as the response variable, the linear-based boosting models and the animal model performed slightly better than the tree-based models. We discuss that regression trees mainly incorporate epistatic effects while the linear-based models focus on additive effects. An explanation of why regression trees are better for body mass is that body mass can rely more on epistatic effects than tarsus length. Results from the comparison between the SHAP values and a univariate GWAS show little correspondence between the SNPs with the highest SHAP value and those with the lowest GWAS p-value. As the SHAP values explain an ensemble of regression trees that primarily model epistatic effects, the low correspondence indicates that the SNPs associated with epistatic effects are not the same as those associated with additive effects.

We conclude that gradient-boosting models can be effective in genomic prediction and that the choice of the base learner is dependent on the complexity of the trait one wishes to model and the genetic effects one wants to model. If additive effects are in focus, a linear-based model should be deployed, and if epistatic effects are of interest, a tree-based model is a good choice. We also conclude that SHAP values can be an alternative to standard univariate GWAS to identify SNPs associated with epistatic effects.

Sammendrag

Genomisk prediksjon bruker genetisk data til å prediktere det genetiske bidraget til et individs fenotype. Genomisk prediksjon er mye brukt i fagfeltet genetisk økologi som et verktøy til å forstå samspillet mellom genetikk og fenotyper, noe som igjen øker forståelsen for prosesser relatert til evolusjon. I denne masteroppgaven har vi undersøkt hvordan en klasse av modeller kalt "gradient boosting" kan brukes til å genomisk prediktere to fenotyper, kroppsvekt og tarsuslengde, i en populasjon av ville gråspurv. Gråspurv-populasjonen er lokalisert langs Helgelandskysten i Nord-Norge og har vært studert siden 1993. Det genetiske materialet i denne populasjonen er tilgjengelig gjennom sekvenserte enkeltnukleotidpolymorfismer (SNP-er). "Gradient boosting"-modeller har tidligere blitt brukt for genomisk prediksjon innenfor plante- og dyreavl, men aldri før i en vill populasjon.

Gradient boosting er et fleksibelt rammeverk for å bygge en meta-modell bestående av en rekke base-modeller. Vi har undersøkt tre forskjellige valg av base-modeller og sammenlignet hvordan det påvirker presisjonen i den genomiske prediksjonen. De tre forskjellige base-modellene er binære regresjonstrær, "elastic net" og stykkevis lineære regresjonstrær. Disse "gradient boosting" ble sammenlignet med en avansert Bayesiansk dyremodell. Elastic net og den Bayesianske dyremodellen er lineære, mens binære regresjonstrær og stykkevis lineære regresjonstrær er tre-baserte. Vi undersøkte også hvordan en forklarbarhets metode kalt Shapley additive forklarings verdier (SHAP verdier) kan bli brukt på en tre-basert modell til å undersøke viktigheten av ulike SNP i modellen. SHAP-verdiene ble så sammenlignet med resultatene fra en univariat blandet lineær genomomfattende assosiasjonsstudie (GWAS).

Resultatene fra masteroppgaven viser at gradient boosting-modeller er konkurransedyktige med den Bayesianske dyremodellen. Valget av hvilken fenotype man ønsker å modellere påvirker hvilken basemodell som bør bli brukt. De tre-baserte modellene ga litt bedre ytelse enn de andre modellene når kroppsvekt ble modellert. Når tarsuslengde ble modellert, gjorde de lineære modellene det litt bedre enn de tre-baserte modellene. I oppgaven diskuterer vi at regresjonstrær i hovedsak modellerer epistatiske effekter, mens de lineære fokuserer på additive effekter. Derfor er en forklaring på hvorfor tre-baserte modeller er bedre for kroppsvekt at kroppsvekt er mer avhengig av epistatiske effekter enn det tarsuslengde er. Vi fant lite korrelasjon mellom SHAP-verdiene fra en tre-basert modell og resultatene fra en univariat GWAS, noe som kan indikere at SNP-er assosiert med epistatiske effekter ikke er de samme som de assosiert med additive effekter.

Vi konkluderer med at gradient boosting-modeller kan være effektive i genomisk prediksjon, og at det beste valget av base-modell er avhengig av kompleksiteten til fenotypen man ønsker å undersøke. Hvis man primært ønsker å se på additive effekter bør det benyttes en lineær modell, mens hvis epistatiske effekter er mest interessant så burde en tre-basert modell benyttes. I tilegg konkluderer vi med at SHAP-verdier kan være et alternativ til univariat GWAS for å identifisere SNP-er assosiert med epistatiske effekter.

Table of Contents

List of Figures

List of Tables

1 Introduction

The core idea in evolutionary genetics is that physical expressions of an individual, called phenotypes or traits, are a function of effects from its genetic material and effects from its surrounding environment (Jefferey K. Conner and Hartl, 2004a). An individual's genetic material is the unique set of instructions in the biological material that determines how said individual will be built. Furthermore, the genetic material is passed on from parent to offspring, implying that some effects on phenotypes caused by the genetic material are heritable. Thus, if an individual exhibits a particular phenotype value and comes from a family of similar phenotypic expressions, the phenotype value may be due to genetic effects and is therefore heritable. Animal and plant breeders have long used such observations to artificially select specific traits that improve their breeding program, which means that from the livestock, they choose particular individuals who are allowed to mate and pass their genes on to the next generation (Jeffrey K. Conner, 2003). The individuals selected usually have a positive expression of the trait of interest (e.g., milk yield or meat content) and have a family history of individuals who also had a positive expression of the trait. With the rise of statistical tools and a broader understanding of genetics, breeders have been able to perform quite efficient selection by using pedigrees to estimate the effect on the phenotype due to heritable and selective genetics, denoted the *breeding value* (Henderson, 1950, 1975).

The field of breeding was again revolutionized with the increase in accessibility of genetic sequencing techniques, as it became possible to access the genetic material of each individual (Meuwissen et al., 2016). Consequently, the field of *genomic prediction*, which is the study of how a phenotype of an individual can be modeled as a function of the genetic material, emerged to estimate the breeding value using genetic data directly (Meuwissen et al., 2001, 2016). While genomic prediction was initially developed in and for the field of animal and plant breeding, it has become a useful tool in ecological genetics (McGaugh et al., 2021; Hunter et al., 2022). Ecological geneticists study how a population is affected by the dynamics between the environment and the genetic variation in the population, and how these dynamics relates to evolutionary processes. Evolution is the change of allele frequencies across generations, meaning that evolution happens when the genetic distribution of a population is shifted across generations (Jefferey K. Conner and Hartl, 2004a). Evolution is often due to selection pressure on a particular trait, where the selection pressure can be artificial from breeding or natural from the environment. Thus, one must understand the population's genetics to understand evolution and the inner mechanics of the formation of a phenotype in a population.

Since 1993, the Center of Biodiversity at NTNU has studied a meta-population of house sparrows off the Helgeland coast in Norway (Ringsby et al., 2002; Jensen, Steinsland et al., 2008). The meta-population is made up of several islands with varying environmental conditions and possibilities for interactions with humans. Migration between the islands maintains genetic flow and prevents the populations from becoming genetically isolated. This setup provides a natural "laboratory" to study ecological genetics. Moreover, with the increase of genetic variation that comes as a product of migration and variation in

selection pressure from the different environments, there is a need to develop better tools to understand the genetic architecture.

One way to understand the genetics of a population is through statistical modeling. By recording observations of different individuals' phenotypes, environmental factors and genetic data, one can build a statistical model, such as the *animal model* (Lynch and Walsh, 1998; Kruuk, 2004; VanRaden, 2008), to make inference about the population and its dynamics. Most models used in genomic prediction are linear (or generalized linear) models which assume a linear additive relationship between the covariates (Meuwissen et al., 2016). The effects of the genetic material on the phenotype are then taken to be independent and linear. In the field of genetics, there are three main groups of effects on the phenotype, *additive* effects, *dominance* effects, and *epistatic* effects (Jefferey K. Conner and Hartl, 2004c). Additive effects are contributions to the phenotype from numerous independent alleles at different locations (loci) along the chromosome. Each allele has a small effect that can be summed into one additive effect, the breeding value. Dominance effects are contributions due to interactions between polymorphisms within an allele, while epistatic effects are contributions due to interactions between alleles at different loci. The contribution to the phenotype from additive effects is the only one, of the three genetic effects, that easily can be used to predict the response to a selection pressure on a quantitative trait. A linear model, therefore, models only the selective part of genetics and neglects the non-additive contributions. Such an approach may lead to a loss in the prediction accuracies, as it could be the case that an individual has a high breeding value but, due to epistatic effects, has a lower phenotype value than expected. The simplification done by a linear model could lead to an underestimation of the importance of genetics when accumulating errors in the modeling over the entire population.

Quantitative traits are often assumed to be polygenetic, meaning that numerous alleles at different loci determine the genetic contributions to the trait. Compared to qualitative traits, traditional (linear) methods of estimating the genetic component of a quantitative phenotype often perform worse (Ashraf et al., 2022). When increasing the number of players that contribute to a game, it is also reasonable to think that interactions among the players improve. Therefore, it may be essential to include non-additive effects when modeling quantitative traits, which could also explain the difference in the accuracies for quantitative and qualitative traits. Parallel with the rise of data availability in all fields of science and industries, various machine-learning models, outside traditional statistical models, have risen in popularity (Sarker, 2021; Sardanelli et al., 2023; Stanford Institute for Human-Centred Artifical Intelligence, 2024). Models such as gradient boosted decision trees (GBDT; Friedman, 2001; Chen and Guestrin, 2016; Ke et al., 2017; Prokhorenkova et al., 2017) and neural networks (LeCun et al., 2015) can find complex patterns in large datasets and impose no parametric assumption on the data-generating mechanism. GBDTs have performed well on tabular data in various fields and competitions (Natekin and Knoll, 2013; Abdurrahman et al., 2020; Tyralis and Papacharalampous, 2021; Asselman et al., 2023; Carlens, 2023). Gradient boosting is a general method used to build an ensemble of a chosen base learner. A popular base learner is the binary regression tree (Breiman et al., 1984), a method that identifies non-linear interactions between features. A GBDT is, thus, a model that can be used in genomic prediction to include epistatic effects.

Gradient boosting models and neural nets have been explored in the setting of genomic prediction in the fields of breeding and medicine (Li et al., 2018; Montesinos-López et al., 2021; Gill et al., 2022; Mathew et al., 2022; Nazzicari and Biscarini, 2022; Chafai et al., 2023; Lourenço et al., 2024), but there has been little consistency in whether they

outperform linear models. To our knowledge, gradient boosting models have never been tried for genomic prediction in a wild population. Individuals in a wild population are more exposed to varying environmental conditions and have experienced less artificial selection from humans, which can increase the genetic variation within the population. More considerable genetic variation may lead to more complexity in the dynamics of genetics. Therefore, we believe that complex models, such as gradient boosting, can have a greater potential in wild populations than in animal and plant breeding populations.

In this thesis, we investigate gradient boosting models' ability to perform genomic prediction in various settings using data from a wild population of house sparrows. We experiment with different, but popular, variations of gradient boosting in a *within-population* prediction context and compare them to results from a genomic Bayesian animal model. Within-population prediction means performing genomic prediction in the same population as the one used to make the genomic prediction model. For the within-population setting, we use two different datasets that differ in the trade-off between the number of genetic markers and the number of individuals recorded. We also investigate how the choice of base learner and loss function affect the performance in an *across-population* prediction setting. Across-population prediction is done when the genomic prediction model is used to predict a population different from the training population. It is important to note that genomic prediction is often used to describe procedures where the breeding value is predicted, that is, the genetic contribution due to additive effects. In this thesis, however, we do not restrict ourselves to modeling only additive effects. Instead, we will use models that include both additive and epistatic effects. We, therefore, try to model more of the genetic contribution to the phenotype than just the breeding value.

Lastly, we look at how one can use a method from explainable AI called *Shapley values* (Shapley, 1953; Lundberg, Erion et al., 2020) to assess the feature importance of the genetic data in the gradient-boosted models and compare it to genome-wide efficient mixed-model association (GEMMA; Zhou and Stephens, 2012), a standard single marker method in genome-wide association studies (GWAS; Uffelmann et al., 2021). GWAS are studies where one searches across the genome to find SNPs most associated with the phenotype. This is typically done by regressing each SNP on the phenotype separately and then using the p-value of the regression coefficient as a measure of association. GWAS often fails to find strong causal loci's associated with the phenotype in wild populations, and the loci identified usually only explain a small proportion of the phenotypic variance (Husby et al., 2015; Bosse et al., 2017; Silva et al., 2017; Lundregan et al., 2018). This is a problem as it illustrates that the models used in GWAS do not capture the underlying genetic structure, making it harder to understand the genetic architecture of quantitative traits. As the features in the genetic data are the SNPs, assessing feature importance in a genomic prediction model is comparable to searching for the SNPs most associated with the phenotype. We, therefore, propose the Shapley values of a tree-based model as an alternative to univariate GWAS.

This thesis aims to improve genomic prediction methodology in a wild population setting. Better methods for genomic prediction can potentially increase our understanding of the dynamics between phenotypes and genetics. By gaining more knowledge about the genetic architecture in a wild population, we make ourselves more prone to understand the consequences of microevolution and environmental changes on that population. Suppose we more accurately understand which loci are prone to be affected by environmental change. In that case, we can use the genomic prediction models to more accurately predict the consequences of such change. One can further use the insight gained from genomic prediction to improve conservation missions by more accurately predicting the response in

a captive breeding program, which helps maintain biodiversity (McGaugh et al., 2021). Therefore, developing more accurate genomic prediction models for wild populations is essential. Hence, this thesis contributes to the 13th and 15th goals of the United Nations sustainable development goals (climate action and life on land; United Nations, 2015).

Note to reader: This thesis is a continuation of Sand (2023) and, therefore, a lot of the material presented in chapter 2 will contain identical formulations to the works of Sand (2023). Some results for the within-population prediction setting are from Sand (2023), this is marked in the text where applicable.

2 Background

In this chapter, we review the background and theory needed to understand the methods used in the thesis. This chapter contains formulations from Sand (2023), which applies to sections 2.1, 2.2, 2.4 and 2.5.

2.1 Methods in Ecological Genetics

Before delving into gradient boosting methods, it is essential to understand the data one works with and what it represents. It is also important to understand which traditional methods have been used and how they work. This section reviews some conventional methods and principles that ecologists have used.

2.1.1 Basic Genetics

The genetics of an organism is the biologically inherited material that specifies the construction and development of an individual (Jefferey K. Conner and Hartl, 2004a). The genetic material is made up of a long molecule called the deoxyribonucleic acid (DNA), spread across several packages called chromosomes. The DNA can be seen as a long ladder where each step of the ladder is a base pair. Four different bases make up the base pairs, Adenine (A), Guanine (G), Cytosine (C) and Thymine (T), where A and T pair together, and C and G pair together. This gives four distinct base pair permutations, AG, GA, CT and TC. A gene is a specific set of base pairs that specify how a protein or a set of proteins is to be constructed, while alleles are different versions of the same gene. Each allele is located on a specific location along the chromosome which is called a locus. Sexually reproducing organisms inherit two sets of chromosomes (diploid population), one from their father and one from their mother (Jefferey K. Conner and Hartl, 2004b). This means that sexually reproducing organisms inherit a set of two alleles placed on the same locus, and together they make up a single genotype.

A way to quantify the genetic material is through marker-based gene sequencing, where an individual's DNA is recorded. As much of the genetic material is identical across species (and thus uninteresting), it is common to sequence only the base pairs that separate one allele from another, called SNPs (single nucleotide polymorphisms). SNPs are often quantified by choosing a reference genome and then encoding the SNP of an individual for each base pair locus as either $0, 1, 2$ (Mittag et al., 2015). The encoding refers to whether both bases are equal to the reference genome (0), only one of the bases is equal to the reference genome (1), or both bases are different from the reference (2). We say that the SNP is homozygous if it has encoding 0 or 2 and heterozygous if it has encoding 1.

Even though only a tiny proportion of the DNA is recorded with SNP-sequencing, it can still effectively capture the genetic architecture and its relation to heritability due to

linkage disequilibrium (Uffelmann et al., 2021). Linkage disequilibrium is the phenomenon where alleles at different loci close to each other are inherited together (Slatkin, 2008). If a recorded SNP is close to the causal SNP of a particular effect, it will, due to linkage disequilibrium, be strongly correlated with the causal SNP. Such dynamics allow us to capture the causal SNP's effect without recording its precise position.

2.1.2 Quantitative Genetics and Phenotypic Variation

A phenotype P of an *individual* is thought to be made up of effects from the environment E and effects from the inherited genes G, that is $P = E + G$ (Jefferey K. Conner and Hartl, 2004c). One can further decompose the genetic effect G into the effects from additive genetic effects, dominance effects, and epistatic effects. Additive genetic effects refer to effects from independent contributions of alleles. Dominance is the effect of interactions between alleles at the same locus, while epistatic effects are the effect of interactions between alleles at different loci. In quantitative genetics, one studies quantitative traits, which are believed to be polygenetic, meaning that the phenotype is made up of contributions from many genes. As genotypes in sexually reproducing organisms are not clones of their parents' genotypes, but rather a new random combination of the parent's alleles (Jefferey K. Conner and Hartl, 2004c), the alleles explaining a phenotype are thought to contribute with independent effects. Therefore, one is particularly interested in quantifying the effect from the additive part of the genotype, which is called the breeding value. The breeding value describes the individual effect a parent's genes have on the offspring's phenotype.

From $P = E + G$, it follows that the variation of a phenotype between individuals in a population is $V_P = V_E + V_G + V_R$, where V_E is the phenotypic variation due to the environment, V_G is the variation due to variation in the genome and V_R are variation due to random noise but also variation due to gene-environment effects. As genes are passed on from parent to offspring V_G gives a measure of heritability, and thus also a quantification of how fast potential selection can drive forth an evolution of the phenotype. More quantitatively, we define the *broad sense heritability* $h_B^2 = V_G/V_P$ which measures the proportion of variation in a phenotype that is due to inherited effects. One frequently breaks down the genetic variance further into the sum of the additive genetic variance V_A, the variance explained by dominant effects V_D, and the variance explained by epistatic effects V_I. The variance of the breeding values of a phenotype in a population is then V_A. To quantify the proportion of variation in a trait due to additive genetic effects, we define the *narrow sense heritability* $h^2 = V_A/V_P$ (Jefferey K. Conner and Hartl, 2004c). In the field of genomic prediction, the broad and narrow sense heritabilities give an upper limit to the prediction accuracy, and narrow heritability is especially used when assessing the accuracy of predicted breeding values.

2.1.3 The Animal Model

One of the pioneering and widely used methods to model the phenotypic trait $\mathbf{y}$ as a function of genetic and environmental variables is the animal model (Henderson, 1984; Kruuk, 2004; Wilson et al., 2010). Animal models were first based on pedigrees of the individuals and could therefore quantify genetic effects without the need for gene sequencing, which can be costly and time-consuming. In its simplest form, it can be stated as a linear mixed

model,

$$y_i = \mu + g_i + \varepsilon_i \quad i = 1, \ldots, n, \tag{2.1}$$

where μ denotes the population mean of a trait, g_i is the breeding value of individual i, and ε_i accounts for the environmental and non-additive effects. Here g_i is a random effect, $\mathbf{g} \sim N(\mathbf{0}, V_A\mathbf{G})$, where V_A is the additive genetic variance and $\mathbf{G} \in \mathbb{R}^{n\times n}$ is the relatedness matrix constructed from a pedigree. The relatedness matrix quantifies how much the different individuals $i = 1, \ldots, n$ are related to each other, for instance, if individual i and j are parent and offspring then $\mathbf{G}_{ij} = 0.5$ as they share half of their genetic material. The animal model is highly flexible, as one can include multiple fixed and random effects to account for specific environmental effects. This allows for investigating the variation between different subgroups in the populations, such as gender, geographic location, or variation between offspring from different mothers.

The animal model can also be extended to use sequenced genetic data by letting the elements of the relatedness matrix $\mathbf{G}_{ij}$ contain the proportion of alleles shared between individual i and j. The genomic relatedness matrix can be constructed by using the first method proposed by VanRaden (2008). Let $\mathbf{M}$ be the matrix of dimension $n \times p$ consisting of p SNP-markers for n individuals, and let p_i be the second allele frequency at locus i (meaning the minor allele frequency). The rows of $\mathbf{M}$ then contain each individual's SNP vector, while the columns contain all individuals' alleles at a specific locus. First we define the zero-centered matrix $\widetilde{\mathbf{M}} = \mathbf{M} - \mathbf{1}$, which means that the SNPs now are encoded as $-1, 0$ or 1 (instead of $0, 1, 2$). This gives an interpretation of the matrix $\widetilde{\mathbf{M}}\widetilde{\mathbf{M}}^T$, where the diagonal elements, $\text{diag}(\widetilde{\mathbf{M}}\widetilde{\mathbf{M}}^T)$, will be the l_2 norm of an individual's SNP vector. Since each genotype is encoded as $-1, 0$ or 1, the l_2 norm will be the sum of number of -1s and 1s present, *i.e.*, the number of homozygous loci an individual has. The matrix products that make up the off-diagonal (*i.e.*, between individuals) will be 1 when they share a homozygous locus, 0 when one or both of the individuals has a heterozygous locus and -1 if they have an opposite homozygous locus. The off-diagonal elements are, therefore, the differences between the number of equal homozygous loci and the number of opposite homozygous loci, *i.e.*, the off-diagonal works as a measure of the number of alleles shared between individuals. Second, VanRaden (2008) suggest defining a new matrix $\mathbf{Z}$ by subtracting $2(p_i - 0.5)$ from the columns/loci of $\widetilde{\mathbf{M}}$ to give rare alleles more credit than common ones. The genomic relatedness matrix $\mathbf{G}$ is then defined as

$$\mathbf{G} = \frac{\mathbf{Z}\mathbf{Z}^T}{2\sum_{i=1}^{p} p_i(1-p_i)} \ ,$$

where the divisor $2\sum_{i=1}^{p} p_i(1-p_i)$ is equal to the sum of heterozygous frequencies under a Hardy-Weinberg equilibrium (see Jefferey K. Conner and Hartl, 2004b). The divisor scales $\mathbf{G}$ to take the heterozygous frequencies into account and make the behavior of the genomic relatedness matrix more similar to the pedigree relatedness matrix. The estimate of the proportion of genomic material shared between individual k and j is given by $\mathbf{G}_{kj}/\sqrt{\mathbf{G}_{\kappa}\mathbf{G}_{jj}}$ (VanRaden, 2008). In that sense, one can think of the genomic relatedness matrix as a covariance matrix, and to get "genomic correlations" one must scale it relative to the two individuals' own "genomic variance".

2.1.4 BLUP and GBLUP

Animal models provide a way to decompose the phenotypic variation into different components, such as additive genetic and environmental variance, and quantify them (Wilson

et al., 2010). However, it is often desirable to obtain estimates of the breeding values $\hat{g}_i$, denoted estimated breeding values (EBV; Meuwissen et al., 2016). For that purpose, we use the best linear unbiased predictor (BLUP) approach, which originated from the animal model with a pedigree (Henderson, 1975). Assume that we have an animal model where the phenotype $\mathbf{y}$ is modeled as

$$\mathbf{y} = \mathbf{1}\mu + \mathbf{X}\boldsymbol{\beta} + \mathbf{g} + \mathbf{W}\mathbf{d} + \boldsymbol{\varepsilon} \ , \tag{2.2}$$

where μ is the population mean, $\boldsymbol{\beta}$ is the fixed effects, $\mathbf{g} \sim N(\mathbf{0}, V_A\mathbf{G})$ is the breeding values (random effect), $\mathbf{d} \sim N(\mathbf{0}, \mathbf{R})$ is the vector containing other random effects, where $\mathbf{R}$ is a diagonal covariance matrix such that the random effects are independent of each other. $\mathbf{X}$ and $\mathbf{W}$ are design matrices of appropriate dimensions. Finally, $\boldsymbol{\varepsilon} \sim N(\mathbf{0}, \sigma^2 I)$ is the residual term. The model can be rewritten to a more compact form by bundling the breeding values and the random effects together into a single vector $\boldsymbol{\gamma}$ with a block-diagonal indices matrix $\mathbf{U}$, and the mean is bundled together with the fixed effects into $\boldsymbol{\beta}^*$ with indices matrix $\widetilde{\mathbf{X}}$. The phenotype is then modeled as

$$\begin{aligned}\mathbf{y} = \widetilde{\mathbf{X}}\boldsymbol{\beta}^* + \mathbf{U}\boldsymbol{\gamma} + \boldsymbol{\varepsilon}, \quad \boldsymbol{\varepsilon} \sim N(\mathbf{0}, I\sigma^2), \quad \boldsymbol{\gamma} \sim N\left(\mathbf{0}, \begin{pmatrix} V_A\mathbf{G} & \mathbf{0} \\ \mathbf{0} & \mathbf{R} \end{pmatrix}\right) = N(\mathbf{0}, \mathbf{C}) \\ \implies \mathbf{y} \sim N(\widetilde{\mathbf{X}}\boldsymbol{\beta}^*, \mathbf{U}\mathbf{C}\mathbf{U}^T + \sigma^2 I) = N(\widetilde{\mathbf{X}}\boldsymbol{\beta}^*, \mathbf{V}) \ ,\end{aligned}$$

which reduces to the pseudo-linear regression problem if we write

$$\mathbf{y} = \widetilde{\mathbf{X}}\boldsymbol{\beta}^* + \boldsymbol{\epsilon}, \quad \epsilon = \mathbf{U}\boldsymbol{\gamma} + \boldsymbol{\varepsilon} \sim N(\mathbf{0}, \mathbf{V}) \ ,$$

which can be standardized by multiplying with $\mathbf{V}^{-1/2}$,

$$\mathbf{V}^{-1/2}\mathbf{y} = \mathbf{V}^{-1/2}\widetilde{\mathbf{X}}\boldsymbol{\beta}^* + \mathbf{V}^{-1/2}\boldsymbol{\epsilon} \iff \mathbf{y}^* = \widetilde{\mathbf{X}}^*\boldsymbol{\beta}^* + \boldsymbol{\epsilon}^*, \quad \boldsymbol{\epsilon}^* \sim N(\mathbf{0}, I) \ ,$$

which is an ordinary linear regression problem. The unbiased MLE estimate of $\boldsymbol{\beta}^*$ is then

$$\hat{\boldsymbol{\beta}}^* = \left(\widetilde{\mathbf{X}}^{*T}\widetilde{\mathbf{X}}^*\right)^{-1}\widetilde{\mathbf{X}}^{*T}\mathbf{y}^* = \left(\widetilde{\mathbf{X}}^T\mathbf{V}^{-1}\widetilde{\mathbf{X}}\right)^{-1}\widetilde{\mathbf{X}}^T\mathbf{V}^{-1}\mathbf{y} \ . \tag{2.3}$$

To obtain estimates of the random effects $\boldsymbol{\gamma}$ we use that for two Gaussian variables $\mathbf{z}_1$ and $\mathbf{z}_2$ on the form

$$\begin{pmatrix} \mathbf{z}_1 \\ \mathbf{z}_2 \end{pmatrix} \sim N\left(\begin{pmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{pmatrix}, \begin{pmatrix} \Sigma_{1,1} & \Sigma_{1,2} \\ \Sigma_{2,1} & \Sigma_{2,2} \end{pmatrix}\right) \ ,$$

we have that

$$E(\mathbf{z}_1|\mathbf{z}_2) = \boldsymbol{\mu}_1 + \Sigma_{1,2}\Sigma_{2,2}^{-1}(\mathbf{z}_2 - \boldsymbol{\mu}_2) \ .$$

Thus, if we look at

$$\begin{pmatrix} \boldsymbol{\gamma} \\ \mathbf{y} \end{pmatrix} \sim N\left(\begin{pmatrix} \mathbf{0} \\ \widetilde{\mathbf{X}}\boldsymbol{\beta}^* \end{pmatrix}, \begin{pmatrix} \mathbf{C} & \mathbf{C}\mathbf{U}^T \\ \mathbf{U}\mathbf{C} & \boldsymbol{V} \end{pmatrix}\right) \ ,$$

we get that

$$E(\boldsymbol{\gamma}|\mathbf{y}) = \mathbf{C}\mathbf{U}^T\mathbf{V}^{-1}(\mathbf{y} - \widetilde{\mathbf{X}}\boldsymbol{\beta}^*) \ . \tag{2.4}$$

An unbiased prediction of $\boldsymbol{\gamma}$ can then be obtained by inserting (2.3) into (2.4), giving the EBV $\hat{\mathbf{g}}$ for each individual, which often are denoted BLUPs (Henderson, 1950; Kruuk, 2004).

When replacing the pedigree matrix $\mathbf{G}$ with the genomic relatedness matrix, derived from the SNPs of each individual, we get what is denoted as the genetic estimated breeding values (GEBV). The predictions of the breeding values are named genetic best linear unbiased predictors (GBLUPs; VanRaden, 2008; Meuwissen et al., 2016).

2.1.5 Genome Wide Association Studies

The breeding value aggregates the additive effect from many alleles on a phenotype for a specific individual and is a valuable tool for both polygenetic, oligogenic, and Mendelian traits. However, with the rise of more accurate and available gene sequencing, researchers in human medicine moved on to look after specific allele variants or regions in the genome associated with diseases (The Wellcome Trust Case Control Consortium et al., 2007). Studies that search for areas related to a trait across different loci and chromosomes became known as genome-wide association studies (GWAS; Uffelmann et al., 2021).

A GWAS using SNP data is typically performed by building a univariate single-marker regression model for each recorded SNP,

$$y_i = \beta_j x_i^j + \varepsilon_i \quad i = 1, \ldots, n \ ,$$

where y_i is the recorded trait value of individual i, x_i^j is the recorded value of SNP j for individual i, β_j is the regression coefficient of SNP j and ε_i is the residual term of individual i. Further, to assess the "significance" of SNP j, one can use a Wald test or a likelihood test on the estimated coefficient, $\hat{\beta}_j$. A reasonable response to such a procedure is why one does not build a multivariate regression model where all $\{\beta_j\}_{j=1}^p$ are fitted simultaneously. Genomic data is often high-dimensional, resulting in the number of observations being smaller than the number of features, $p >> n$, which linear models do not handle well. For regularized variants that perform model selection, such as the Lasso, which handles $p >> n$, there is not clear how to correctly make inferences about the coefficients (see Kuchibhotla et al. (2022) for a review of the problem and some solutions).

If one is to draw correct inferences from a regression model, the model's underlying assumptions must be met. The core assumption in a linear regression model is the assumption of independence between the features and independence between the observations. The first assumption is met as a univariate regression model only uses one feature. However, the second one, about independence between observations, is likelier to be broken when working with genetic data. In data from wild populations, there will be observations of individuals who are related and thereby have correlated trait values due to more shared genetic material and environmental conditions. To account for population structures in the data, a univariate linear mixed model is instead fitted, where a random intercept on the identity is included. The random intercept is drawn from a normal distribution where the covariance matrix is the genomic relatedness matrix, similar to the genomic animal model (Zhou and Stephens, 2012).

2.2 Bayesian Methods for Animal Models

To grasp how well the gradient boosting models perform, we will compare them to a state-of-the-art Bayesian animal model. The Bayesian animal model can be fitted using the integrated nested Laplace approximation (INLA) approach. To understand what INLA is and how to use it to fit a Bayesian animal model, a basic understanding of Bayesian statistics is needed, which we review in this section.

2.2.1 Bayesian Modelling

In traditional frequentist statistics, one believes that the parameter θ is a fixed unknown number or vector. The parameter is estimated through the data $\mathbf{y}$, giving $\hat{\theta}$ which is a random variable as the data is modeled to be random. This is the frequentist point of view. Another paradigm is the Bayesian point of view, which is that the parameter θ is random and follows a distribution $p(\theta)$ denoted the prior distribution (Casella and Berger, 2002a). Bayesians seek a distribution of the parameter given the data we have observed, this distribution is denoted the posterior distribution $p(\theta|\mathbf{y})$. By using the Bayes rule one can relate the posterior to the data and prior,

$$p(\theta|\mathbf{y}) = \frac{f(\mathbf{y}|\theta) \cdot p(\theta)}{m(\mathbf{y})} \ , \tag{2.5}$$

where $f(\mathbf{y}|\theta)$ is the likelihood (often denoted $L(\mathbf{y}, \theta)$ in frequentist statistics), $p(\theta)$ is the prior distribution and $m(\mathbf{y})$ is the marginal distribution of the data given by

$$m(\mathbf{y}) = \int f(\mathbf{y}|\theta)p(\theta)d\theta \ .$$

Bayesian analysis aims to find and use the posterior distribution to make inferences about the data. Bayesians often use Markov Chain Monte Carlo (MCMC) techniques to generate samples from the posterior distribution (Geof H. Givens and Jennifer A. Hoeting, 2013). MCMC is non-deterministic, usually time-consuming, and has no guarantee of reaching convergence. In recent years, newer deterministic methods such as the integrated nested Laplace approximation approach (INLA; Rue et al., 2009) have arisen. Before we review INLA, we review a Bayesian modelling paradigm denoted hierarchical models.

2.2.2 Hierarchical Models

A popular modeling approach within Bayesian modeling is the hierarchical (latent) model which generally consists of three parts (Gamerman and Lopes, 2006). The first part is the *observational* model of the data $\mathbf{y}$ which, when conditioned on a latent field $\mathbf{x}$ and some hyperparameters $\boldsymbol{\theta}_1$, is independent and identically distributed,

$$f(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta}_1) = \prod_{i=1}^{N} f(y_i|\mathbf{x}, \boldsymbol{\theta}_1) \ ,$$

the observational model thus contains information about the generation of the observed data. The second part of the hierarchical model is the *latent* model $\mathbf{x}|\boldsymbol{\theta}_2$, which contains information about the unobserved process, where $\boldsymbol{\theta}_2$ contains hyperparameters for the latent model. The third part of the model is the *hyperpriors* $\pi()$ that are placed on the hyperparameters $\boldsymbol{\theta} = \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2\}$.

As an example of a well-known model that fits the framework of Bayesian hierarchical models, consider a simple linear regression model

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i, \quad i = 1, \dots N \quad \varepsilon_i \sim N(0, \sigma^2)$$

where (y_i, x_i) is observation i. The Bayesian way to specify this model would be to assume that,

$$y_i|\beta_0, \beta_1, \sigma^2 \sim N(\mu_i, \sigma^2), \quad \mu_i = \beta_0 + \beta_1 x_i \ ,$$

which make up the observational part of the model. β_0 is the intercept and β_1 is the regression slope. The latent part of the model would be to assume that β_0 and β_1 follow some prior distribution, where a popular prior is

$$\beta_0 \sim N(a_0, \nu_0^2), \quad \beta_1 \sim N(a_1, \nu_1^2) \ ,$$

where a_0 and a_1 are user-defined numbers. We thus have $\theta_1 = \sigma^2$ as the hyperparameter for the observational model and $\boldsymbol{\theta}_2 = \{\nu_0^2, \nu_1^2\}$ as hyperparameters for the latent model. Thus, to complete the hierarchical model we only need to place some hyperpriors on the hyperparameters $\boldsymbol{\theta} = \{\sigma^2, \nu_0^2, \nu_1^2\}$, where a popular choice is the inverse-gamma distribution. While the hierarchical model may seem too complex for a simple linear regression, it provides a powerful framework for situations where the data is nested with several layers of clusters. The hierarchical model is convenient when building intertwined mixed models where the inference about model parameters can be difficult to interpret.

2.2.3 Bayesian Animal Model by INLA

Integrated nested Laplace approximation (INLA) is an efficient method to obtain approximate posterior distributions for latent Gaussian models (Rue et al., 2009). The details of the numerical scheme of INLA are beyond the scope of this project, but the reader is referred to Rue et al. (2009) and Martino and Riebler (2019) for an in-depth explanation.

A latent Gaussian model is a hierarchical model where the latent field $\mathbf{x}|\boldsymbol{\theta}_2$ is assumed to be Gaussian,

$$\mathbf{x}|\boldsymbol{\theta}_2 \sim N(0, \mathbf{Q}^{-1}(\boldsymbol{\theta}_2)) \ ,$$

where $\mathbf{Q}$ is the precision matrix, *i.e.*, the inverse of the covariance matrix which INLA requires to be sparse. It is easy to see that the simple regression model considered in Section 2.2.2 is a latent Gaussian model if one puts $a_0 = a_1 = 0$ in the priors of β_0 and β_1.

Animal models on the form of (2.2) can easily be stated as Gaussian latent model, and thus fitted using the INLA approach by letting $\mathbf{x} = (\boldsymbol{\beta}, \mathbf{g}, \mathbf{d})$ be the latent field and place a Gaussian prior on $\boldsymbol{\beta}$ with zero mean, $\boldsymbol{\beta} \sim N(0, I\sigma_\beta^2)$. The hyperparameters are then $\boldsymbol{\theta} = \{V_A, \sigma^2, \mathbf{R}\}$, and the observational model is

$$\mathbf{y}|\boldsymbol{\beta}, \mathbf{g}, \mathbf{d} \sim N(\boldsymbol{\eta}, \sigma^2 I), \quad \boldsymbol{\eta} = \mathbf{1}\mu + \mathbf{X}\boldsymbol{\beta} + \mathbf{g} + \mathbf{W}\mathbf{d} \ .$$

Note that σ_β^2 was not included in $\boldsymbol{\theta}$, which is because we want $\boldsymbol{\beta}$ to act like fixed effects where we don't want to model its variance as a hyperparameter. One typically chooses a prior with a large variance for fixed effects, which means that the posterior distribution of the fixed effects will be dominated by the data. INLA also requires a sparse inverse precision matrix $\mathbf{Q}$, which is fulfilled for the animal model as the inverse of the relatedness matrix $\mathbf{G}$ is sparse (Steinsland and Jensen, 2010).

Using the INLA framework one ends up with approximate marginal posterior sample distributions for each of the $l = 1, \ldots, k$ variables in the latent field, $\tilde{\pi}(x_l|\mathbf{y})$, and for each of the $j = 1, \ldots, r$ hyperparameters, $\tilde{\pi}(\theta_j|\mathbf{y})$. It is thus easy to obtain GEBVs by taking the mode of the sample distribution of the breeding values $\tilde{\pi}(g_i|\mathbf{y})$ for each individuals $i = 1 \ldots, n$. For more explanations and examples of how INLA can be used with animal models, we refer the reader to (Holand et al., 2013).

2.3 Statistical Learning

The empirical scientific approach usually involves formulating a hypothesis and then gathering empirical data to determine its likelihood. A subset of such approaches involves hypotheses where one wants to explore how a variable is affected by other factors. For example, "How does my cake taste depend on the different ingredients I use?" or it could be more complex, such as "How is the biodiversity of the world affected by current climate changes". In both cases, the empirical scientific approach involves collecting data to create a model that can predict the most likely response to a given input. With the cake example, we seek a model such that if I say the ingredients used are A, B, and C, I will receive a predicted taste score from the model.

Two things are needed to build such a model. First, we need a dataset of observations showcasing the relationship between the response (e.g., cake taste score) and the other variables that impact the response, denoted features, or covariates. Second, we need a way of evaluating our model, both to improve the model and to compare it with other models. As the model should be able to predict a likely response given some input features, a way to evaluate it is to look at the distance between the predicted response and the actual response value. The distance between the truth and the prediction is measured through a loss function.

Statistical learning is the field that formalizes how such models can be built. We say that statistical learning is the field of determining how the response y is related to the covariates $\mathbf{x}$, and the goal is to determine a function (or rule) $f(\mathbf{x})$ such that $y = f(\mathbf{x})$. A covariate is another observed variable which we believe influences y, sometimes also denoted as a feature. We gather multiple (paired) observations of the response $\mathbf{y}$ and the covariates $\mathbf{X}$, giving the dataset $D = (\mathbf{y}, \mathbf{X})$. The function $f(\mathbf{x})$ is then determined by minimizing a loss $l(y, f(\mathbf{x})$ over all data $D = (\mathbf{y}, \mathbf{X})$. Statistical learning separates itself from ordinary statistics by being more prediction-focused than inference-focused. In statistical learning, having a model that predicts well is more important than an interpretable model. This section reviews some theoretical ideas and results from statistical learning. We focus on how the choice of loss function affects any model, and in later sections, we will present different model choices.

To avoid confusion, we first define some notations regarding the data.

- $y \in \mathbb{R}$: Ground truth observation of response variable, the trait we want to model. Often we add a suffix, y_i and y_j, to distinguish two observations i and j from each other.
- $\mathbf{y} \in \mathbb{R}^n$: a vector (or a set) of n ground truth observations, often we write $\mathbf{y} = \{y_i\}_{i=1}^n$.
- $\mathbf{x}_i \in \mathbb{R}^p$: a vector of p covariates (explanatory variables) corresponding to observation y_i. If the suffix i is dropped, $\mathbf{x}$ corresponds to a general observation y.
- $\mathbf{x}^m \in \mathbb{R}^n$: a vector containing all n observations of covariate $m \in \{1, \ldots, p\}$, we refer to this as a *covariate vector*.
- $\mathbf{X} \in \mathbb{R}^{n \times p}$: a matrix consisting of all the covariate vectors as columns. Such that $\mathbf{X}_{i,m} = \mathbf{x}_i^m$ is the observation of covariate m corresponding to trait observation y_i.

2.3.1 The Decision-Theoretic Framework of Statistical Learning

By using ideas from decision theory (Casella and Berger, 2002b), we develop a useful tool to investigate the different loss functions' influence on the model behavior. The tool in question is the *expected prediction error* (Hastie et al., 2009), often denoted as the expected risk in decision theory. The function $f(\mathbf{x})$ represents a decision on what we think that y is, given the new feature vector $\mathbf{x}$ we observe and the previous data $\{y_i, \mathbf{x}_i\}_{i=1}^n$. We can treat the response variable as a random vector Y that can take on values $Y = y \in \mathbb{R}$ following some distribution $p(y)$. In the same way, the covariate vector $\mathbf{x} \in \mathbb{R}^p$ can be treated as random vector $\boldsymbol{\chi}$. In the decision-theoretic framework, we seek to minimize the expected loss,

$$\text{EPE}(f) = E_{Y,\boldsymbol{\chi}}(l(Y, f(\boldsymbol{\chi}))) = \int_{\boldsymbol{\chi}} \int_Y l(y, f(\mathbf{x}))p(y, \mathbf{x})dyd\mathbf{x} \ , \tag{2.6}$$

which is denoted the *expected prediction error* (EPE). The optimal decision function is then

$$\hat{f}(\mathbf{x}) = \arg\min_f \text{EPE}(f) = \arg\min_f E_{Y,\boldsymbol{\chi}}(l(Y, f(\boldsymbol{\chi}))) \ . \tag{2.7}$$

The EPE can be written in a slightly more convenient form by exploiting the fact that $p(y, \mathbf{x}) = p(y|\mathbf{x})p(\mathbf{x})$, (2.6) can then be written as

$$\text{EPE}(f) = \int_{\boldsymbol{\chi}} \int_Y l(y, f(\mathbf{x}))p(y|\mathbf{x})p(\mathbf{x})dyd\mathbf{x}, = \int_{\boldsymbol{\chi}} p(\mathbf{x}) \int_Y l(y, f(\mathbf{x}))p(y|\mathbf{x})dyd\mathbf{x} \ . \tag{2.8}$$

The loss function l plays an important role in the shape of f, as it determines what part of the data the model should pay attention to, and more importantly, it determines how much attention the model gives to the different parts of the model. Different loss functions lead to different objectives giving different models. In practice when evaluating a model $\hat{f}$, we estimate the EPE by taking the mean of $l(y_i, \hat{f}(\mathbf{x}_i))$,

$$\widehat{\text{EPE}}(\hat{f}) = L(\mathbf{y}, \hat{f}(\mathbf{X})) = \frac{1}{n}\sum_{i=1}^n l(y_i, \hat{f}(\mathbf{x}_i)) \ . \tag{2.9}$$

2.3.2 Mean Squared Error

The most popular loss is the mean-squared error, a variant of the l_2 norm, which results in estimators that estimate the expected value (Hastie et al., 2009). The squared error loss between a (scalar) prediction $\hat{y} = \hat{f}(\mathbf{x})$ and a ground truth y is given by

$$l(y, \hat{y}) = ||\hat{y} - y||_2^2 = (\hat{y} - y)^2 \ , \tag{2.10}$$

so that for a set of predictions $\hat{\mathbf{y}} = \hat{f}(\mathbf{X}) \in \mathbb{R}^n$ and a set of true $\mathbf{y} \in \mathbb{R}^n$, the estimated EPE in this case called the mean-squared error is

$$L(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{n}\sum_{i=1}^n l(\hat{y}_i, y_i) = \frac{1}{n}\sum_{i=1}^n (\hat{y}_i - y_i)^2 \ .$$

To explain the popularity of the squared loss, we must look at the true EPE from decision theory. By inserting (2.10) into (2.8) we can deduce that the EPE for squared error can be written as

$$\text{EPE}(\hat{f}) = \int_{\boldsymbol{\chi}} p(\mathbf{x}) \int_Y (\hat{f}(\mathbf{x}) - y)^2 p(y|\mathbf{x})dyd\mathbf{x}$$

To find the prediction which minimizes the EPE we find where the derivative of the EPE is zero,

$$\begin{aligned}
\frac{\partial \text{EPE}}{\partial \hat{f}} &= \int_{\boldsymbol{\chi}} p(\mathbf{x}) \int_Y 2(\hat{f}(\mathbf{x}) - y) p(y|\mathbf{x}) dy d\mathbf{x} = 0 \\
\implies & \int_{\boldsymbol{\chi}} p(\mathbf{x}) \left(\int_Y \hat{f}(\mathbf{x}) p(y|x) dy - \int_Y y \cdot p(y|\mathbf{x}) \right) d\mathbf{x} = 0 \\
\implies & \int_{\boldsymbol{\chi}} p(\mathbf{x}) \left(\hat{f}(\mathbf{x}) \int_Y p(y|x) dy - \int_Y y \cdot p(y|\mathbf{x}) \right) d\mathbf{x} = 0 \ .
\end{aligned}$$

For $p(y|\mathbf{x})$ to be a valid probability distribution, the total probability must equal to one, that is,

$$\int_Y p(y|\mathbf{x}) dy = 1 \ .$$

Also, note that

$$\int_Y y \cdot p(y|\mathbf{x}) dy = E(Y|\mathbf{x}) \ ,$$

which gives

$$\begin{aligned}
\frac{\partial \text{EPE}}{\partial \hat{f}} = 0 \implies & \int_{\boldsymbol{\chi}} p(\mathbf{x}) \left(\hat{f}(\mathbf{x}) \cdot 1 - E(Y|\mathbf{x}) \right) d\mathbf{x} = 0 \\
\implies & \hat{f}(\mathbf{x}) = E(Y|\mathbf{x}) \quad \forall \ \{\mathbf{x} \in \boldsymbol{\chi} | p(\mathbf{x}) > 0\} \ ,
\end{aligned}$$

meaning that the EPE for squared loss is minimized when the decision function estimates the conditional expected value of the response variable. The squared error loss's ability to yield estimators that estimate the expected value is a property that has made it the most popular loss function. The squared error loss is also easy to optimize as it is a smooth function with a simple derivative.

2.3.3 Mean Absolute Error

While the squared error yields the expected value, which is desirable when all the data is equally representative, it's important to note that there are situations where outliers are present. These extreme values, when present, are not ideal for model learning as they are often considered errors or highly improbable. Consider the case where the true conditional expected response is $E(Y|\mathbf{x}) = 2$ and the observed data is $\mathbf{y} = (1, 2, 3)$. If we wanted to build a constant model, $f(\mathbf{x}) = a \in \mathbb{R}$, based on $\mathbf{y}$, the squared error loss would result in the mean as an estimator (Casella and Berger, 2002b). The mean of $\mathbf{y}$ is now 2, but if we add an outlier of 50 to the data, the mean is now 14. Thus, only one data point is enough to move the mean away from its original and correct value. Therefore, finding other loss functions that yield more robust estimates is desirable. Another popular loss function is absolute loss, which, as we will show, can yield a more robust estimate. The mean absolute error uses the l_1 norm instead of the l_2 norm, such that

$$l(y, \hat{y}) = ||\hat{y} - y||_1 = |\hat{y} - y|, \quad L(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{n} \sum_{i=1}^{n} l(y_i, \hat{y}_i) = \frac{1}{n} \sum_{i=1}^{n} |\hat{y}_i - y_i|.$$

To find properties of the absolute loss we again look at the EPE (2.8) which now takes the form

$$\begin{aligned}\text{EPE}(\hat{f}) &= \int_{\boldsymbol{\chi}} p(\mathbf{x}) \int_{Y} |\hat{f}(\mathbf{x}) - y| p(y|\mathbf{x}) dy d\mathbf{x} \\ &= \int_{\boldsymbol{\chi}} p(\mathbf{x}) \left(\int_{Y<\hat{f}(\mathbf{x})} (\hat{f}(\mathbf{x}) - y) p(y|\mathbf{x}) dy - \int_{Y>\hat{f}(\mathbf{x})} (\hat{f}(\mathbf{x}) - y) p(y|\mathbf{x}) dy \right) d\mathbf{x} \ ,\end{aligned} \tag{2.11}$$

meaning that the derivative of the EPE to $\hat{f}$ is

$$\frac{\partial \text{EPE}}{\partial \hat{f}} = \int_{\boldsymbol{\chi}} p(\mathbf{x}) \left(\int_{Y<\hat{f}(\mathbf{x})} p(y|\mathbf{x}) dy - \int_{Y>\hat{f}(\mathbf{x})} p(y|\mathbf{x}) dy \right) d\mathbf{x}. \tag{2.12}$$

To find the optimal $\hat{f}$ we can exploit the fact that

$$\int_{Y<\hat{f}(\mathbf{x})} p(y|\mathbf{x}) dy = \mathbb{P}(Y < \hat{f}(\mathbf{x})|\mathbf{x}) \quad \text{and}$$
$$\int_{Y>\hat{f}(\mathbf{x})} p(y|\mathbf{x}) dy = \mathbb{P}(Y > \hat{f}(\mathbf{x})|\mathbf{x}) = 1 - \mathbb{P}(Y < \hat{f}(\mathbf{x})|\mathbf{x}) \ ,$$

and insert it into the equation $\frac{\partial \text{EPE}}{\partial \hat{f}} = 0$,

$$\iff \int_{\boldsymbol{\chi}} p(\mathbf{x}) \left(2\mathbb{P}(Y < \hat{f}(\mathbf{x})|\mathbf{x}) - 1 \right) d\mathbf{x} = 0 \implies \mathbb{P}(Y < \hat{f}(\mathbf{x})|\mathbf{x}) = \frac{1}{2} \quad \forall \ \{\mathbf{x} \in \boldsymbol{\chi} | p(\mathbf{x}) > 0\} \ .$$

Implying that the optimal $\hat{f}(\mathbf{x})$ when using absolute loss is the conditional median,

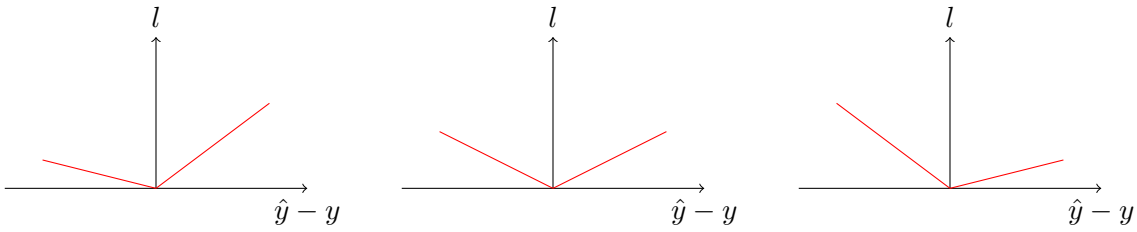
$$\hat{f}(\mathbf{x}) = \text{median}(Y|\mathbf{x}) \ .$$

The absolute loss ability to estimate the median is a property that results in more robust estimators compared to the squared error, which is helpful in cases where outliers are present (Hastie et al., 2009). To illustrate the robustness of the median, consider again the data $\mathbf{y} = (1, 2, 3)$, where the median is 2 and the mean is 2. If we now again receive an outlier value of 50, the median is changed to 2.5, whereas the mean was 14. Such behavior of the estimates makes sense, as the squared loss will pay quadratic attention to an outlier compared to the absolute loss, which applies as linear weighting of the error, meaning that an outlier will receive far more attention than ordinary samples with the squared error than with the absolute error.

2.3.4 Quantile Regression by Pinball Loss

The absolute loss results in estimators that estimate the conditional median of the response variable, where the median is the 50% quantile of the conditional distribution. A more flexible extension of the absolute loss is the so-called *pinball loss* which aims to give estimators that estimate an arbitrary quantile α of the conditional distribution (Koenker, 2005; Yu et al., 2018). That is, we want the loss to yield an estimator $\hat{f}$ such that

$$\mathbb{P}(Y < \hat{f}(\mathbf{x})|\mathbf{x}) = \alpha \ .$$



(a) $\alpha = 0.25$ (b) $\alpha = 0.5$ (c) $\alpha = 0.75$

Figure 2.1: Pinball loss for different alpha values. When $\alpha = 0.25$ the loss applies more focus to the cases where $\hat{y} > y$, whereas when $\alpha = 0.75$ the cases where $\hat{y} < y$ receive the most focus. For $\alpha = 0.5$, both cases receive equal focus.

Consider now the pinball loss function of the form

$$l(y, \hat{y}) = ((1-\alpha)I(\hat{y} > y) + \alpha I(\hat{y} \leq y))\,|\hat{y} - y| = \begin{cases} (1-\alpha)(\hat{y} - y), & \hat{y} > y \\ -\alpha(\hat{y} - y), & \hat{y} \leq y \end{cases}$$

whose derivative is

$$\frac{\partial l(y, \hat{y})}{\partial \hat{y}} = \begin{cases} (1-\alpha), & \hat{y} > y \\ -\alpha, & \hat{y} \leq y \end{cases} .$$

It is then easy to see that the derivative of the EPE with the pinball loss is a modified version of (2.12),

$$\begin{aligned} \frac{\partial \text{EPE}}{\partial \hat{f}} &= \int_{\boldsymbol{\chi}} p(\mathbf{x}) \left((1-\alpha) \int_{Y < \hat{f}(\mathbf{x})} p(y|\mathbf{x}) dy - \alpha \int_{Y > \hat{f}(\mathbf{x})} p(y|\mathbf{x}) dy \right) d\mathbf{x} \\ &= \int_{\boldsymbol{\chi}} p(\mathbf{x}) \left((1-\alpha)\mathbb{P}(Y < \hat{f}(\mathbf{x})|\mathbf{x}) - \alpha \mathbb{P}(Y > \hat{f}(\mathbf{x})|\mathbf{x}) \right) d\mathbf{x} \\ &= \int_{\boldsymbol{\chi}} p(\mathbf{x}) \left((1-\alpha)\mathbb{P}(Y < \hat{f}(\mathbf{x})|\mathbf{x}) - \alpha \left(1 - \mathbb{P}(Y < \hat{f}(\mathbf{x})|\mathbf{x}) \right) \right) d\mathbf{x} \\ &= \int_{\boldsymbol{\chi}} p(\mathbf{x}) \left(\mathbb{P}(Y < \hat{f}(\mathbf{x})|\mathbf{x}) - \alpha \right) d\mathbf{x} \, . \end{aligned}$$

The optimal $\hat{f}$ is then given by

$$\begin{aligned} \frac{\partial \text{EPE}}{\partial \hat{f}} = 0 \iff & \int_{\boldsymbol{\chi}} p(\mathbf{x}) \left(\mathbb{P}(Y < \hat{f}(\mathbf{x})|\mathbf{x}) - \alpha \right) d\mathbf{x} = 0 \\ \implies & \mathbb{P}(Y < \hat{f}(\mathbf{x})|\mathbf{x}) = \alpha \quad \forall \, \{\mathbf{x} \in \boldsymbol{\chi} | p(\mathbf{x}) > 0\} \, , \end{aligned}$$

which is fulfilled when $\hat{f}(\mathbf{x})$ is equal to the α quantile of $Y|\mathbf{x}$. In other words, an estimator that is made by minimizing the pinball loss estimates the α quantile of the conditional distribution of $Y|\mathbf{x}$. The ability to estimate the quantiles is a useful property as it allows us to control how we deal with overestimation and underestimation. A low α results in punishing overestimation (the cases where $\hat{y} > y$) harder than an underestimation, whereas with a high α we punish underestimation harder (see Figure 2.1).

2.3.5 Model Assesment and Model Complexity

In general, the true distribution of the data $p_{Y,\mathbf{X}}(y, \mathbf{x})$ is not known and must be estimated from the data $D = (\mathbf{y}, \mathbf{X})$. This implies that we cannot calculate the EPE directly, and as a consequence, we do not find $\hat{f}(\mathbf{x})$ by (2.7). Instead, we use the estimate of the EPE given in (2.9), giving the objective

$$\hat{f}(\mathbf{x}) = \arg\min_f \widehat{\text{EPE}}(f) = \arg\min_f L\Big(\mathbf{y}, f(\mathbf{X})\Big) \ .$$

This means that we build our model $\hat{f}$ based on minimization of the *training error*,

$$L\Big(\mathbf{y}, \hat{f}(\mathbf{X})\Big) \ .$$

In this setup, the model is built and evaluated using the same data points. However, the model is usually not built to only predict within the dataset. Usually, we want to deploy the model in a setting where it must predict on data it has not seen before. We say that we desire a model that is generalizable to data that is different from its training data. A generalizable model leads to the notion of the *generalizable error*, which is what the EPE measures. Unfortunately, it turns out that the training error is a bad estimate of the EPE (Hastie et al., 2009). The fallacy of the training error is that complex and flexible models adapt too well to the training data. We say that the models start overfitting or interpolating the training data, resulting in poor capabilities to capture the underlying data-generating mechanisms. An overfitted model yields an artificially low training error that does not reflect the true EPE well. These issues raise the need for better strategies to assess the performance of a model both during and after training.

A common strategy to obtain more accurate estimates of the generalizable error is to split the data in two. One part denotes the training set, while another denotes the test set. As the name implies, the model is trained on the training set, and after training, the EPE is estimated using the test set. This way, the EPE of the model is estimated without using data to which the model has already been adapted, which results in a fairer evaluation of a potentially overfitted model. It is still worth noting that the data is a random variable, making both the model fitting process and the model evaluation random. One way to account for the randomness in the model assessment is by performing a *cross-validation*. In cross-validation, or a k-fold cross-validation, the entire data is first split into k distinct parts or folds, and the model is trained and evaluated at k times. This is done by iterating through the folds, and in each iteration, the current fold is used as a test set, while the remaining $k - 1$ folds are used as training sets. After iterating all the loops, we end up with k models, but more importantly, k model evaluations that give a more representative picture of the model's generalization error. However, we still need methods to prevent overfitting from occurring.

One way to monitor overfitting is by splitting the entire data into three parts: the training, test, and validation sets. The test and training set is as before, but the validation set is used to evaluate the model during training. The validation set is not used to build the model but only for evaluation. The validation set is typically used for models iteratively built during training, resulting in an iterative increase in complexity. For each iteration, the loss on the training set and the loss on the validation set are recorded. Once the validation error deviates from the training error, signs of overfitting appear, and the model training should end. Many more methods exist to prevent overfitting, such as regularization, which will be reviewed in the coming sections.

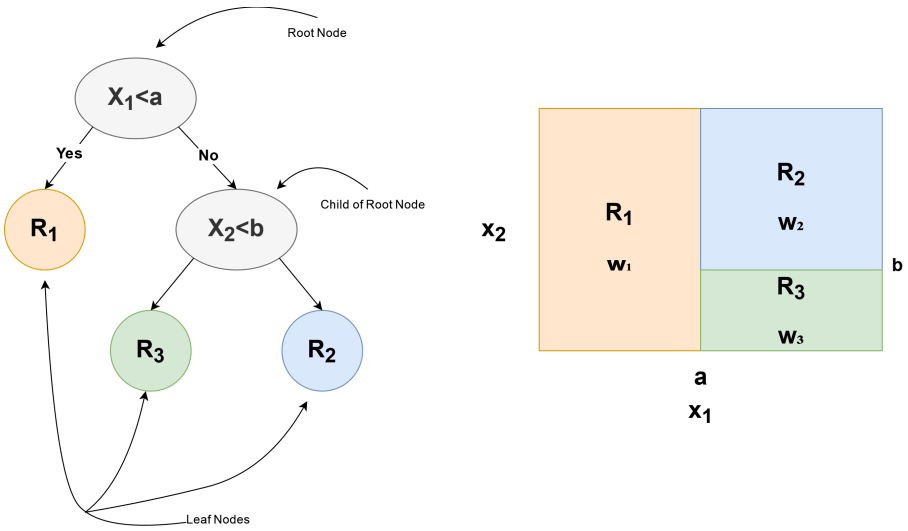


Figure 2.2: Simple sketch of a regression tree from Sand (2023) showing how the splitting of the feature space corresponds to a binary tree.

2.4 Regression Trees

The previous section on statistical learning only considered a general model $f(\mathbf{x})$ and how it was affected by the loss function. This section reviews a potential model, namely the binary regression/classification tree (Breiman et al., 1984; Hastie et al., 2009), often denoted CART (classification and regression trees). This thesis only considers continuous traits, so we only review binary regression trees. Binary regression trees are popular models due to their efficiency and ability to capture non-linear interactions.

2.4.1 Fitting a Regression Tree

The idea behind CART is to divide the feature space into separate regions so that most similar response observations are grouped by their covariates. The regions a regression tree makes are represented by several recursive binary splits of one feature at a time. That is, the regions are determined by answering a series of yes-no (e.g. binary) questions on the form "Is $\mathbf{x}^c > s_c$?", where each question splits the previous region into two. Here c is the feature dimension to split and s_c is a specific feature value to split on. Starting with the entire feature space as a region, in the end, we obtain J regions $R_1, R_2, ..., R_J$ each equipped with a weight w_j, $j = 1, \ldots, J$. A tree makes a new prediction $\hat{y}^*$ for a new feature vector $\mathbf{x}^*$ by placing $\mathbf{x}^*$ in its corresponding region R_j and letting $\hat{y}^* = w_j$. The binary splitting performed by trees can be represented as a binary tree where the *root node* contains the first question and based on the answer (yes or no) we are pointed to one of the two child nodes (Figure 2.2). Each leaf node represents a region in the feature space. Therefore, we use the phrases "region R_j" and "leaf node R_j" interchangeably. More formally, we express the regression tree as

$$f(\mathbf{x}) = \sum_{j=1}^{J} I(\mathbf{x} \in R_j) \cdot w_j \ .$$

Assume now that we have obtained several regions, and wish to split region R_j into two new ones R_{j+1} and R_{j+1}. To do so, we have to determine which feature c_j to split, and

where to split it, s_{c_j}, and what weights to assign the new regions, w_{j+1} and w_{j+2}. The current loss for the n_j training samples belonging to R_j is

$$L_j = \frac{1}{n_j} \sum_{\mathbf{x}_i \in R_j} l(y_i, w_j),$$

and the reduction in loss by splitting the region using feature c and split point s is

$$\begin{aligned}
&\Delta_j(s, c) = L_j - L_{j+1} - L_{j+2} \\
&= \frac{1}{n_j} \sum_{\mathbf{x}_i \in R_j} l(y_i, w_j) - \frac{1}{n_{j+1}} \sum_{\mathbf{x}_i \in R_{j+1}} l(y_i, w_{j+1}) - \frac{1}{n_{j+2}} \sum_{\mathbf{x}_i \in R_{j+2}} l(y_i, w_{j+2}) \qquad (2.13) \\
&= \sum_{\mathbf{x}_i \in R_j} \left(\sum_{\mathbf{x}_i^c > s} \left[\frac{1}{n_j} l(y_i, w_j) - \frac{1}{n_{j+1}} l(y_i, w_{j+1}) \right] + \sum_{\mathbf{x}_i^c \leq s} \left[\frac{1}{n_j} l(y_i, w_j) - \frac{1}{n_{j+2}} l(y_i, w_{j+2}) \right] \right) .
\end{aligned}$$

To find c_j and s_{c_j} we consider all covariates $m = 1 \dots p$, and for each covariate m we consider all observations $\{x_i^m\}_{i=1}^{n_j}$ belonging to region R_j as a candidate split point. To evaluate (2.13) for all candidate split points we need to find the weights w_{j+1} and w_{j+2} for all candidate split points. The weights are the constants that minimize reduce the loss for its region the most. That is,

$$w_{j+1} = \arg\min_w L_{j+1} = \arg\min_w \frac{1}{n_{j+1}} \sum_{\mathbf{x}_i \in R_{j+1}} l(y_i, w) \ . \qquad (2.14)$$

And equivalent for R_{j+2}'s weight. Such constants are easy to find when the loss is known. For instance, if we use a squared loss we can find w_{j+1} by derivation to zero,

$$\begin{aligned}
\frac{\partial L_{j+1}}{\partial w} = \frac{\partial}{\partial w} \frac{1}{n_{j+1}} \sum_{\mathbf{x}_i \in R_{j+1}} (y_i - w)^2 = \frac{2}{n_{j+1}} \sum_{\mathbf{x}_i \in R_{j+1}} (w - y_i) = 0 \\
\implies \sum_{\mathbf{x}_i \in R_{j+1}} w = \sum_{\mathbf{x}_i \in R_{j+1}} y_i \implies w_{j+1} = \frac{1}{n_{j+1}} \sum_{\mathbf{x}_i \in R_{j+1}} y_i \ .
\end{aligned}$$

The best split-point for feature m, s_{m_j}, is the observation that reduces training loss the most,

$$s_{m_j} = \arg\max_{\mathbf{x}_i \in R_j} \Delta_j(\mathbf{x}_i, m) \ .$$

All the split-point's reduction in the loss $\{\Delta_j(s_{m_j}, m)\}_{m=1}^p$ is then compared, and we choose the feature c that gives the greatest reduction in loss,

$$c_j = \arg\max_m \Delta_j(s_{m_j}, m) \ .$$

The tree fitting procedure is summarized in Algorithm 2 which uses Algorithm 1 to do the binary splitting of features. The binary splitting presented in Algorithm 1 is a greedy algorithm, as the algorithm only considers the best split for the current leaf node without considering how such a split will affect future splittings.

Algorithm 1 Binary splitting in trees

Input: Data $(\mathbf{X}, \mathbf{y}) \in (\mathbb{R}^{n \times p}, \mathbb{R}^n)$ and current leaf node R_j

Initialize c_j
for m = 1, . . . , p **do**
    Initialize s_{m_j}
    for each observation $\mathbf{x}_i \in R_j$ **do**
        Calculate new weights w_{j+1} and w_{j+2} using (2.14)
        if $\Delta_j(\mathbf{x}_i, m) > \Delta_j(s_{m_j}, m)$ **then**
            $s_{m_j} = \mathbf{x}_i^m$
        end if
    end for
    if $\Delta_j(s_{m_j}, m) > \Delta_j(s_{c_j}, c_j)$ **then**
        $c_j = m$
    end if
end for

Output: Feature to split c_j, corresponding split point s_{c_j} and weights for new region, w_{j+1} and w_{j+2}.

Algorithm 2 Regression tree fitting algorithm

Input: Data $(\mathbf{X}, \mathbf{y}) \in (\mathbb{R}^{n \times p}, \mathbb{R}^n)$

Initialize tree $\hat{f}(\mathbf{x})$ with only a root node having entire feature space as region R_0.
while tree complexity is less than some threshold **do**
    Choose a leaf node R_j
    Split R_j in two using Algorithm 1
    Update tree $\hat{f}(\mathbf{x})$
end while

Output: Regression tree $\hat{f}(\mathbf{x})$.

2.4.2 Drawbacks of Regression Trees

Regression trees are efficient in building and capturing non-linear interactions through feature-dependent sequences of binary split points, but they are not without flaws. First, without a stopping strategy, regression trees are prone to overfitting the data by assigning each training observation its own region. Second, regression trees are not deterministic. When multiple split points result in the best gain, it is random which one is chosen, and as the fitting strategy is greedy, such randomness can considerably impact the model's performance. Lastly, regression trees do not capture linear relationships well, and as each region is assigned a constant weight only, they cannot extrapolate outside the training data. However, it is still possible to build powerful models with regression trees; the clue is to make not only one tree but many. In the next section, we will review a powerful technique to build an ensemble of several models, which allows us to remove some of the flaws of regression trees.

2.5 Gradient Boosting

The core methods of this thesis rely on a general ensemble method called gradient boosting (Mason et al., 1999; Friedman, 2001, 2002; Hastie et al., 2009). An ensemble method is a method, or model, which uses several base-learners (could *i.e.* be 10 linear regression models) to build a single "meta-learner" called the ensemble model. Before explaining gradient boosting, we review the boosting principle and the idea behind an optimization technique called gradient descent.

2.5.1 Boosting Models

Boosting is a family of ensemble methods that is an additive sum of base-learners $f(\mathbf{x})$, where the base-learners are sequentially added to the meta-learner,

$$f^{b+1}(\mathbf{x}) = f^b(\mathbf{x}) + f(\mathbf{x}) \ .$$

The performance of the current boosting model f^b determines a reweighting of the data on which the next base learner is trained. We will let $f_b(\mathbf{x})$ denote the base learner that is added to the current boosting model $f^b(\mathbf{x})$. Assuming that we use B boosting iterations and a *learning rate* $h \in \mathbb{R}$, the full model reads

$$f^{b+1}(\mathbf{x}) = f^b(\mathbf{x}) + hf_{b+1}(\mathbf{x}), \quad b = 0, \ldots, B, \quad f^0 = f_0 \ .$$

After every boosting iteration b the data is reweighed in some way based on the performance of the current boosting model f^b and the next base-learner f_{b+1} is then trained on the reweighed data. The subfamilies of boosting models differ in how they choose to reweigh the data, a common choice is for instance to use the residuals of the current boosting model $\boldsymbol{\varepsilon}^b = \mathbf{y} - f^b(\mathbf{X})$ as target for the next base-learner. However, a more general and popular family of boosting methods is *gradient boosting*, a method that allows us to build boosting models that minimize any differentiable loss using any base-learner. To explain gradient boosting we first need to review an optimization technique called gradient descent.

2.5.2 Gradient Descent

Imagine that we seek to minimize a function $F : \mathbb{R}^n \to \mathbb{R}$, that is we try to find the function argument $\mathbf{x}^*$ such that

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} F(\mathbf{x}) \ . \tag{2.15}$$

The gradient $\nabla F(\mathbf{x})$ gives the direction in which F increases the most, when located at $\mathbf{x}$. Similarly, the negative gradient $-\nabla F(\mathbf{x})$ gives the direction in which F decreases the most when located at $\mathbf{x}$. Starting from a random point $\mathbf{x}^0$ one can sequentially find a new better minimizer of F by going a step of size h in the direction of the negative gradient. That is

$$\mathbf{x}^1 = \mathbf{x}^0 - h\nabla F(\mathbf{x}^0) \ ,$$

giving the more general update rule

$$\mathbf{x}^{t+1} = \mathbf{x}^t - h\nabla F(\mathbf{x}^t), \quad t = 0, \ldots, T \ . \tag{2.16}$$

When using (2.16) we are using the method called *gradient descent* (Curry, 1944), which for small enough step-lengths h and a large enough number of iterations M are ensured to find a local minimum of F.

The idea of gradient descent can be extended into a Newton-Raphson optimization method (Ypma, 1995), which makes use of the Hessian (*i.e.* $\nabla^2 F$) as well as the gradient. By using the second-order Taylor approximation on F around $\mathbf{x}^t$,

$$F(\mathbf{x}) \approx F(\mathbf{x}^t) + (\mathbf{x} - \mathbf{x}^t)\nabla F(\mathbf{x}^t) + \frac{1}{2}\nabla^2 F(\mathbf{x}^t)(\mathbf{x} - \mathbf{x}^t)^2 \ ,$$

as an approximation of F, we can find that the gradient (derivative with respect to $\mathbf{x}$) is approximately

$$\nabla F(\mathbf{x}) \approx \nabla F(\mathbf{x}^t) + \nabla^2 F(\mathbf{x}^t)(\mathbf{x} - \mathbf{x}^t) \ .$$

Putting the gradient to zero, we obtain the optimal $\mathbf{x}$,

$$\mathbf{x} = \nabla^2 F(\mathbf{x}^t)^{-1} \left(\nabla^2 F(\mathbf{x}^t)\mathbf{x}^t - \nabla F(\mathbf{x}^t)\right) = \mathbf{x}^t - \nabla^2 F(\mathbf{x}^t)^{-1}\nabla F(\mathbf{x}^t),$$

giving the more general update rule for a Newton-Raphson procedure with a learning rate h

$$\mathbf{x}^{t+1} = \mathbf{x}^t - h\nabla^2 F(\mathbf{x}^t)^{-1}\nabla F(\mathbf{x}^t), \quad t = 0 \dots T \ . \tag{2.17}$$

2.5.3 Back to Gradient Boosting

Our objective in supervised learning is to find a function $\hat{f}(\mathbf{x})$ that predicts y by minimizing a loss $L(,)$,

$$\hat{f}(\mathbf{x}) = \arg\min_f L(y, f(\mathbf{x})) \ , \tag{2.18}$$

that is a minimization problem. We can compare this to the minimization problem in (2.15), and see that the loss L takes the role of F, while the learner f is the function argument ($\mathbf{x}$ in 2.15). If we apply the idea of gradient descent (2.16) to (2.18) with an initial base-learner f^0, we get the updated rule

$$\hat{f}^{b+1}(\mathbf{x}) = \hat{f}^b(\mathbf{x}) - h\nabla L(y, \hat{f}^b(\mathbf{x})) \ , \tag{2.19}$$

meaning that to ensure that the loss is minimized the next learner f_{b+1} should approximate the negative gradient of the loss of the previous boosting iteration (Mason et al., 1999). In each boosting iteration, the target variable of the base learner is the negative gradient of the previous learner. Therefore, for each training sample $\{y_i\}_{i=1}^n$ and after each boosting iteration $b = 1, \dots, B$ we find the negative gradient

$$g_{ib} = -\left[\frac{\partial L(y_i, f(\mathbf{x}_i)}{\partial f(\mathbf{x}_i)}\right]_{f=\hat{f}^b} \ ,$$

and train a new base-learner on the data set $\{g_{ib}, \mathbf{x}_i\}_{i=1}^n$. The gradient-boosting procedure is described in Algorithm 3.

To obtain a second order gradient boosting procedure (Saberian et al., 2011; Sigrist, 2021) we apply (2.17) on (2.18),

$$\hat{f}^{b+1}(\mathbf{x}) = \hat{f}^b(\mathbf{x}) - h\nabla^2 L(y, \hat{f}^b(\mathbf{x}))^{-1}\nabla L(y, \hat{f}^b(\mathbf{x})) \tag{2.20}$$

Algorithm 3 Gradient Boosting

Input: Base-learner f, Loss L, data $\{y_i, \mathbf{x}_i\}_{i=1}^n$ and learning rate h

1: Initialize learner $\hat{f}^0(\mathbf{x})$ by training the base-learner on the input data.
2: **for** b = 1, ..., B **do**
3: Calculate gradients $g_{ib} = -\left[\frac{\partial L(y_i, f(\mathbf{x}_i))}{\partial f(\mathbf{x}_i)}\right]_{f=\hat{f}^b}$
4: Train new base-learner $\hat{f}_{b+1}(\mathbf{x})$ using the data $\{g_{ib}, \mathbf{x}_i\}_{i=1}^n$
5: Update the meta-learner: $\hat{f}^{b+1} = \hat{f}^b(\mathbf{x}) + h\hat{f}_{b+1}(\mathbf{x})$
6: **end for**

Output: $\hat{f}^B$

and as we assume that y is a scalar, meaning that $L(y, f(\mathbf{x}))$ is a scalar loss, we know that both $\nabla L(y, \hat{f}^b(\mathbf{x}))$ and $\nabla^2 L(y, \hat{f}^b(\mathbf{x}))$ is scalars. We therefore know that $\nabla^2 L(y, \hat{f}^b(\mathbf{x}))^{-1} = \frac{1}{\nabla^2 L(y, f^b(\mathbf{x}))}$. The update rule is thus

$$\hat{f}^{b+1}(\mathbf{x}) = \hat{f}^b(\mathbf{x}) - h\frac{\nabla L(y, \hat{f}^b(\mathbf{x}))}{\nabla^2 L(y, \hat{f}^b(\mathbf{x}))} \ . \tag{2.21}$$

Using the same logic as before, we want the next weak learner to approximate the fraction between the gradient and the Hessian of the loss of the previous learner in (2.21). This way, the loss is minimized in Newton-Raphson fashion, which we denote as second-order boosting. In second-order boosting, we find the Hessian for each training sample,

$$h_{ib} = \left[\frac{\partial^2 L(y_i, f(\mathbf{x}_i))}{\partial^2 f(\mathbf{x}_i)}\right]_{f=\hat{f}^b},$$

in addition to the gradients g_{ib}. The next base-learner $\hat{f}_{b+1}$ is then trained using the set $\{\frac{g_{ib}}{h_{ib}}, \mathbf{x}_i\}_{i=1}^n$. Second-order boosting is summarized in Algorithm 4. Note that if the Hessian is constant, such as for the squared loss, we end up with a scaled version of ordinary gradient boosting, and if the Hessian is exactly equal to one we end up with a procedure exactly equal to gradient boosting.

Algorithm 4 Second Order Gradient Boosting

Input: Base-learner f, Loss L, data $\{y_i, \mathbf{x}_i\}_{i=1}^n$ and learning rate h

1: Initialize learner $\hat{f}^0(\mathbf{x})$ by training the base-learner on the input data.
2: **for** b = 1, ..., B **do**
3: Calculate gradients $g_{ib} = -\left[\frac{\partial L(y_i, f(\mathbf{x}_i))}{\partial f(\mathbf{x}_i)}\right]_{f=\hat{f}^b}$
4: Calculate Hessians $h_{ib} = \left[\frac{\partial^2 L(y_i, f(\mathbf{x}_i))}{\partial^2 f(\mathbf{x}_i)}\right]_{f=\hat{f}^b}$
5: Train new base-learner $\hat{f}_{b+1}(\mathbf{x})$ using the data $\{\frac{g_{ib}}{h_{ib}}, \mathbf{x}_i\}_{i=1}^n$
6: Update the meta-learner: $\hat{f}^{b+1} = \hat{f}^b(\mathbf{x}) + h\hat{f}_{b+1}(\mathbf{x})$
7: **end for**

Output: $\hat{f}^B$

2.5.4 Adding the Tricks of the Trade

As gradient boosting has increased in popularity, more and more "tricks" have been added to the initially simple algorithm to improve accuracy and efficiency. These tricks have become part of the standard package in a gradient boosting model, and we, therefore, review

the most important ones here before looking at different versions of gradient boosting. The most common base learner in gradient boosting is binary regression trees, denoted gradient boosted regression trees (GBDT). Therefore, many of these "tricks" are developed for GBDTs. However, some ideas in the optimizations presented are also transferable to other base learners.

Regularization

One of the most common ways to combat overfitting in trees is through *regularization*, which means that one either explicitly determines how complex each tree should be or adds a penalty term on the loss function that increases when the tree complexity increases. Regularization aims to reduce the model's variance and, consequently, reduce the chance of overfitting (Hastie et al., 2009). Tree complexity refers here to sizes such as the total depth of the tree, the number of leaf nodes, and the total size of the weights in a tree. If J is the number of leaf nodes, and we have the corresponding weights $\mathbf{w} = \{w_j\}_{j=1}^{J}$ in a tree $f(\mathbf{x})$, one can regulate those sizes during training through an extended loss function

$$\mathcal{L}(\mathbf{y}, f(\mathbf{X})) = L(\mathbf{y}, f(\mathbf{X})) + \Omega(f) = L(\mathbf{y}, f(\mathbf{X})) + \gamma J + \lambda ||\mathbf{w}||_2^2 + \alpha ||\mathbf{w}||_1, \tag{2.22}$$

where γ, λ, and α are regularization parameters that determine the degrees of the penalties. Here, we have included an l_2 and l_1 penalty on the weights $\mathbf{w}$, which are the most common ones to use. Recall the reduction in the loss by splitting a leaf node R_j in two, $\Delta_j(s, c)$ given in (2.13). It is natural only to split a node if the reduction is positive, $\Delta_j(s, c) > 0$. Assume now that we only penalize the number of leaf nodes J. The reduction in loss by splitting a region R_j ((2.13)) is then

$$\begin{aligned}\Delta_j(s, c) &= \mathcal{L}_j - \mathcal{L}_{j+1} - \mathcal{L}_{j+1} \\ &= \underbrace{L_j - L_{j+1} - L_{j+2}}_{=\Delta_j(s,c)^{\text{orig}}} + \gamma J - \gamma(J+1) = \Delta_j(s, c)^{\text{orig}} - \gamma \ ,\end{aligned}$$

meaning that the criterion for splitting a node is now

$$\Delta_j(s, c) = \Delta_j(s, c)^{\text{orig}} - \gamma > 0 \implies \Delta_j(s, c)^{\text{orig}} > \gamma \ .$$

Thus, the γ parameter controls the minimum reduction in the loss needed to split a node, which is a stricter criterion than only requiring a positive gain in the loss. We achieve even more stringent splitting criteria by adding the regularization of the weights. The regularization is, therefore, a way to perform automatic *pruning* of the tree. Automatic pruning ensures that we do not keep splitting regions only to achieve minimal reductions in training loss. Instead, we get trees that only split the regions when it considerably reduces the loss.

Stochastic Gradient Boosting

One of the pioneering ways to build a tree ensemble is through bagging (Breiman, 1996) or random forest (Breiman, 2001), both of which decorrelate the trees in an ensemble by letting them train on different parts of the data. Stochastic gradient boosting (Friedman, 2002) uses the same idea by including subsampling of the data between the boosting iterations. Subsampling means to create subsets of the data by random sampling and is used to prevent overfitting, decorrelate the trees and expose different parts of the data.

The subsampling can be done observation-wise or feature-wise, where observation-wise subsampling omits entire observations $(y_i, \mathbf{x}_i)$ in the training data. In contrast, feature-wise subsampling drops a random set of feature vectors $\mathbf{x}^m$.

Histogram Bundling

The binary splitting, Algorithm 1, has a time complexity of $O(n_u \times p)$, where n_u is the number of unique observations and p is the number of features. The time complexity becomes large when having many continuous features, as then n_u becomes large . A method to reduce the time complexity is the histogram approach (Alsabti et al., 1998) which places continuous features into discrete bins and then uses the bins of the histogram as candidates for split points. The histogram is built for each feature $m = 1, \ldots, p$ (after line 2 in Algorithm 1) such that the number of unique observations is reduced to the number of bins. This gives far fewer candidates to consider compared to using all unique samples and the time complexity reduces to $O(\#\text{bins} \times p)$.

2.5.5 Extensions of Gradient Boosted Trees

Gradient boosting, and gradient boosted trees specifically, is a method that has shown great results in tabular data competitions (Carlens, 2023; Google Cloud, 2024). However, gradient boosting suffers from high time complexity, as each base learner is trained sequentially, making it hard to train the model in parallel. As a consequence, various new implementations have arisen that aim at both improvement of efficiency and improvement in accuracy. We review here the most notable versions of gradient boosting, namely XGBoost (Chen and Guestrin, 2016), CatBoost (Prokhorenkova et al., 2017) and LightGBM (Ke et al., 2017). Each of these add new techniques to the core ideas presented in Section 2.5.3 and Section 2.5.4, improving the efficiency and accuracy, which makes the gradient boosting framework more accessible to anyone.

XGBoost

XGBoost (Chen and Guestrin, 2016) is perhaps the most popular version of gradient boosting and combines various techniques to improve accuracy and efficiency compared to the original gradient boosting machine (Friedman, 2001). XGBoost uses a second-order gradient boosting procedure, meaning that the hessian and gradient are used as targets during training. XGBoost is strongly regularized through various hyperparameters that impose penalties on the complexity of the tree such as l_1 and l_2 penalties on the weights of the tree, or l_1 penalty on the depth of the tree. Regularization ensures that the trees in the XGBoost ensemble are automatically pruned. XGBoost also allows the user to explicitly define certain aspects of the tree's properties such as how many observations a leaf must contain or the total depth of the tree.

XGBoost trains their tree slightly differently from the ones explained in Section 2.4. Being at boosting iteration $b \in \{1, \ldots, B\}$, XGBoost consider the next base learner to be expressed by

$$\hat{f}_b(\mathbf{x}) = \arg\min_f L\left(\mathbf{y}, f^b(\mathbf{X}) + f(\mathbf{X})\right) = \arg\min_f \sum_{i=1}^{n} l(y_i, f^b(\mathbf{x}_i) + f(\mathbf{x}_i)) \ ,$$

where we for simplicity have omitted the regularization term $\Omega(f)$ from (2.22). XGBoost now approximates the loss $l(\cdot,\cdot)$ through a second order Taylor expansion,

$$\hat{f}_b(\mathbf{x}) \approx \arg\min_f \sum_{i=1}^{n} l(y_i, f^b(\mathbf{x}_i)) - g_{ib} f(\mathbf{x}_i) + \frac{1}{2} h_{ib} f(\mathbf{x}_i)^2 \ ,$$

and by removing the constant term, we get the objective

$$\hat{f}_b(\mathbf{x}) \approx \arg\min_f \sum_{i=1}^{n} -g_{ib} f(\mathbf{x}_i) + \frac{1}{2} h_{ib} f(\mathbf{x}_i)^2 \ . \tag{2.23}$$

If we now let I_j be the indices of the observations $\mathbf{x}_i$ belonging to leaf node R_j, we can expand (2.23) to leaf level,

$$\hat{f}_b(\mathbf{x}) \approx \arg\min_f \sum_{j=1}^{J} \sum_{i \in I_j} -g_{ib} w_j + \frac{1}{2} h_{ib} w_j^2 \ . \tag{2.24}$$

Assuming that the leaf regions are fixed, the optimal weight value w_j is

$$w_j = \frac{\sum_{i \in I_j} g_{ib}}{\sum_{i \in I_j} h_{ib}} \ ,$$

which we can insert into (2.24),

$$\hat{f}_b(\mathbf{x}) \approx \arg\min_f -\frac{1}{2} \sum_{j=1}^{J} \frac{(\sum_{i \in I_j} g_{ib})^2}{\sum_{i \in I_j} h_{ib}} \ ,$$

meaning that the optimal loss value for a given tree structure is

$$\tilde{L}(f) = -\frac{1}{2} \sum_{j=1}^{J} \frac{(\sum_{i \in I_j} g_{ib})^2}{\sum_{i \in I_j} h_{ib}} = -\frac{1}{2} \sum_{j=1}^{J} \tilde{L}_j \ . \tag{2.25}$$

We can use $\tilde{L}(f)$ as a way to determine the gain in splitting a leaf node by looking at the reduction in the optimal loss. If we split leaf node R_j into a left and right node R_L and R_R with index sets I_L and I_R ($I_j = I_L \cup I_R$), the gain is

$$\begin{aligned} \text{gain} &= -\frac{1}{2}\tilde{L}_j - (-\frac{1}{2}\tilde{L}_L - \frac{1}{2}L_R) = \frac{1}{2}\Big(\tilde{L}_L + \tilde{L}_R - L_j\Big) \\ &= \frac{1}{2}\Big(\frac{(\sum_{i \in I_L} g_{ib})^2}{\sum_{i \in I_L} h_{ib}} + \frac{(\sum_{i \in I_R} g_{ib})^2}{\sum_{i \in I_R} h_{ib}} - \frac{(\sum_{i \in I_L} g_{ib})^2}{\sum_{i \in I_L} h_{ib}}\Big) \ . \end{aligned} \tag{2.26}$$

The gain gives a way of measuring the quality of each candidate split point without calculating the weights. The weights are instead calculated after one has determined the choice of split points. This way of growing trees is almost identical to the second-order boosting procedure (Algorithm 4), as we can rewrite (2.23) into

$$\hat{f}_b \approx \arg\min_f \sum_{i=1}^{n} \frac{1}{2} h_{ib} \Big(f(\mathbf{x}_i) - \frac{g_{ib}}{h_{ib}}\Big)^2 - \frac{1}{2}\frac{g_{ib}^2}{h_{ib}^2} = \arg\min_f \sum_{i=1}^{n} \frac{1}{2} h_{ib} \Big(f(\mathbf{x}_i) - \frac{g_{ib}}{h_{ib}}\Big)^2 \ ,$$

which is equal to the objective in a weighted squared loss regression with $\frac{g_{ib}}{h_{ib}}$ as the response variable, where the weight is half the Hessian of each sample, $\frac{1}{2}h_{ib}$. A weighted least squares loss is often used to account for unequal variance in the samples or to give certain samples more attention than others. As the Hessian describes the curvature of the loss function, it gives more weight to samples that are curving away from the minimum.

XGBoost implements the histogram approach to improve efficiency but extends it further into a weighted histogram approach. Usually, bins in a histogram are constructed such that each bin contains an equal number of observations. XGBoost instead constructs the bins such that all bins have an equal sum of weights, where the sum of weights, in this case, is the sum of Hessians of the loss. However, in the regression case, most Hessians of the loss are constant, such that there is no difference between the weighted and unweighted histograms. XGBoost also implements various algorithms and ideas from computer science, such as support for parallel learning and cache-aware access, to improve efficiency, which is out of the scope of this thesis. See Chen and Guestrin (2016) for more details.

CatBoost

Another popular version of gradient boosting is CatBoost (Prokhorenkova et al., 2017), originally developed as an alternative to XGBoost for situations with categorical data. CatBoost uses first-order gradient boosting by default but also supports second-order gradient boosting. As XGBoost, CatBoost is heavily regularized to minimize the degree of overfitting. CatBoost's main motivation and difference from other versions of gradient boosting is to solve *the prediction shift problem*. The prediction shift problem occurs during the training of a gradient boosting model when we use the same observations in both the estimation of the gradients and the estimation of the new tree. Base learners in gradient boosting are trained to estimate the negative gradients of each of the observations in the training set. In boosting iteration b we use the data $\{(g_{ib}, \mathbf{x}_i)\}_{i=1}^n$ to train a base-learner $\hat{f}_b(\mathbf{x})$, but g_{ib} is calculated by using $\mathbf{x}_i$ as

$$g_{ib} = -\Big[\frac{\partial L(y_i, f(\mathbf{x}_i))}{\partial f(\mathbf{x}_i)}\Big]_{f=\hat{f}^b} ,$$

creating a relationship between g_{ib} and $\mathbf{x}_i$ that shifts the distribution of the gradients in the training set compared to the gradients of the test set. The prediction shift problem is essentially an issue of overfitting, where the same data $\mathbf{x}_i$ is used to train the base learner and find the gradient g_{ib}. To address the prediction shift problem, CatBoost uses a special gradient boosting procedure called *ordered boosting*. This procedure ensures that the estimation of the base learner and the gradients does not involve using the same data twice.

Ordered boosting ensures that the training of the next tree is done by using a subset of the training data, and then the gradient is calculated using a different non-overlapping subset. The tree-fitting algorithm may be perceived as complicated and hard to understand. Still, the important takeaway is that CatBoost builds the tree as the data were coming in *sequentially*, one observation at a time, see Figure 2.3.

At the start, before any training is performed, CatBoost makes s independent random permutations $\{\sigma_r\}_{r=1}^s$ of the training data. $\sigma_r(i)$ is the mapping between an observation $\mathbf{x}_i$'s original placement in the data and its position in permutation r. If the original data consisted of indexes $\{1, 2, 3\}$ and the rth permutation is $\{2, 3, 1\}$, then $\sigma_r(1) = 3$. In each

boosting iteration b, a random choice σ_r of the permutations is used to estimate the leaf weights. As mentioned, each tree in CatBoost treats the data as if it were sequentially receiving it, one observation at a time. Therefore, during the construction of a model in boosting iteration b, CatBoost saves the temporary models $f^b_{r,k}$, where $f^b_{r,k}(\mathbf{x}_i)$ is the *current* prediction of the i-th observation using the k first observations in permutation σ_r. Similarly, $g_{b,r,k,i}$ is the gradient based on $f^b_{r,k}(\mathbf{x}_i)$. CatBoost constructs the tree's architecture using all available training data, but the weights of the leaves are calculated sequentially for each training observation. Let $w_{b,r}(\mathbf{x}_i)$ be the current leaf weight for observation $\mathbf{x}_i$ based on the $\sigma_r(i) - 1$ previous observations in the permuted dataset, meaning that it is constructed using the gradients $\{g_{b,r,t-1,t}\}_{t=1}^{\sigma_r(i)-1}$, again see Figure 2.3. The weight can be related to the notation we established in Section 2.4 as $w_{b,r}(\mathbf{x}_i) = w_j \iff \mathbf{x}_i \in R_j$. The weight $w_{b,r}(\mathbf{x}_i)$, which is the prediction of training observation $\mathbf{x}_i$ in boosting iteration b is then calculated by gradients that are not based on sample $\mathbf{x}_i$, which combats the prediction shift problem (Prokhorenkova et al., 2017). If one had used the same permutation of the data in each boosting iteration, the weight of the first observation $w_{b,r}(\mathbf{x}_0)$ would always have no-previous gradients to be based, and thus be constantly equal to its initial value, which is the reason why CatBoost changes between the s random permutations.

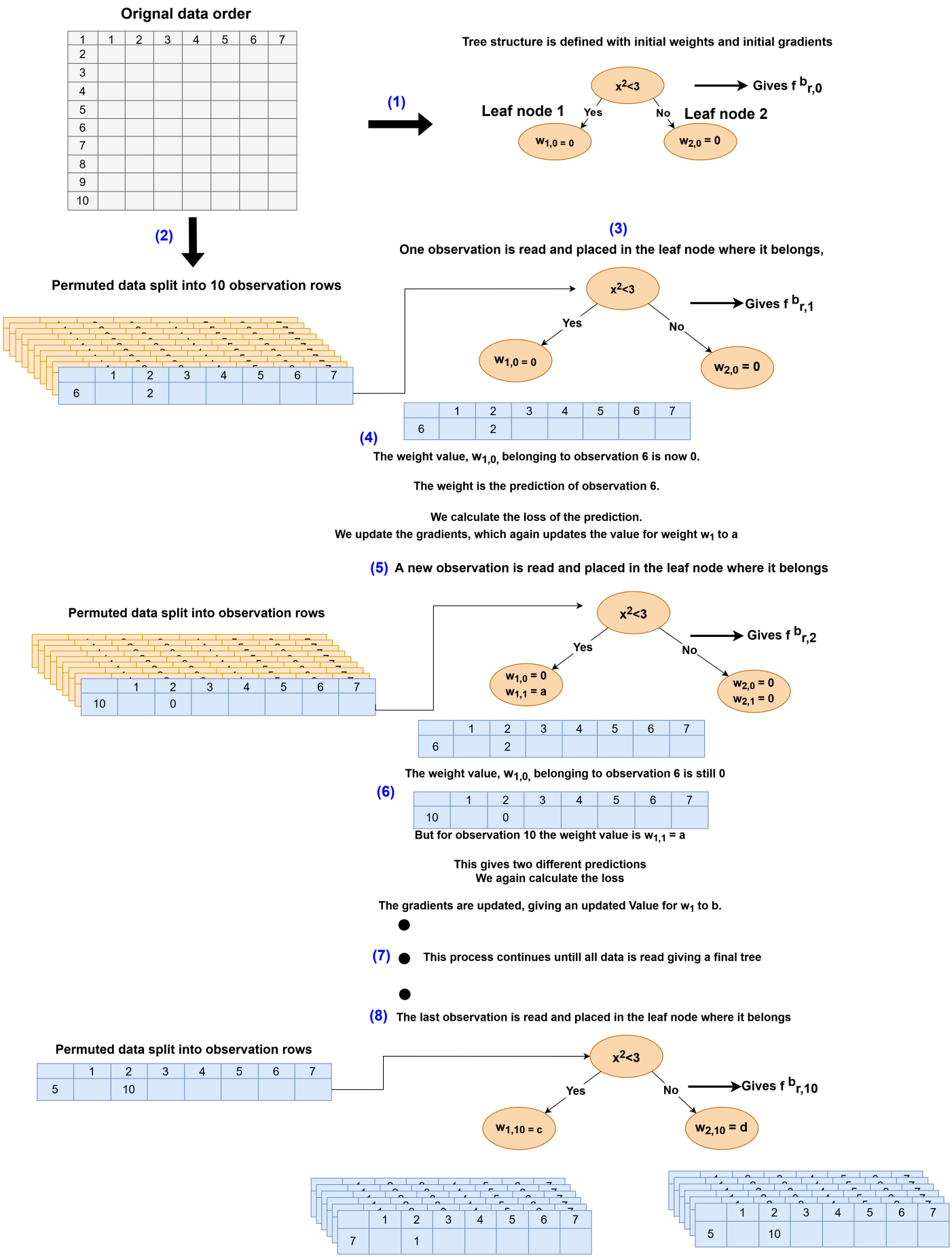

Orignal data order
Tree structure is defined with initial weights and initial gradients
(1)
$x^2<3$
Gives f $^b_{r,0}$
Leaf node 1
Yes
No
Leaf node 2
$w_{1,0} = 0$
$w_{2,0} = 0$
(2)
Permuted data split into 10 observation rows
(3)
One observation is read and placed in the leaf node where it belongs,
$x^2<3$
Gives f $^b_{r,1}$
Yes
No
$w_{1,0} = 0$
$w_{2,0} = 0$
(4)
The weight value, $w_{1,0}$, belonging to observation 6 is now 0.
The weight is the prediction of observation 6.
We calculate the loss of the prediction.
We update the gradients, which again updates the value for weight w_1 to a
(5) A new observation is read and placed in the leaf node where it belongs
Permuted data split into observation rows
$x^2<3$
Gives f $^b_{r,2}$
Yes
No
$w_{1,0} = 0$
$w_{1,1} = a$
$w_{2,0} = 0$
$w_{2,1} = 0$
The weight value, $w_{1,0}$, belonging to observation 6 is still 0
(6)
But for observation 10 the weight value is $w_{1,1} = a$
This gives two different predictions
We again calculate the loss
The gradients are updated, giving an updated Value for w_1 to b.
(7) This process continues untill all data is read giving a final tree
(8) The last observation is read and placed in the leaf node where it belongs
Permuted data split into observation rows
$x^2<3$
Gives f $^b_{r,10}$
Yes
No
$w_{1,10} = c$
$w_{2,10} = d$

Figure 2.3 *(previous page)*: Sketch of how CatBoost builds a tree using ordered boosting, taken from Sand (2023). The data is read sequentially, and each observation is assigned its weight by sequentially updating the gradients. The numbers in blue indicate the order of the sequential process. (1) The data, consisting of 10 observations, defines a tree structure with one split on feature 2. Weights are only initialized, no observations have been predicted, and no loss has been calculated. As a result, we currently have no gradients. (2) A permutation r of the data is chosen such that the rows of observations are mixed in different orders. (3) The first observation in the permuted order, observation 6, is selected and placed in a leaf node. (4) The only weight in leaf 1 is $w_{1,0} = 0$ meaning that the prediction for observation 6 is 0. We calculate the loss and the gradient of the loss of the prediction, which we use to calculate a new weight for leaf 1,$w_{1,1} = a$. (5) We read the next observation in the permuted order, observation 10, and place it in a leaf node. (6) We now use weight $w_{1,1} = a$ as a prediction for observation 10 and calculate the loss and the gradient of the loss for the prediction. We now have two gradients, one for observation 6 and one for observation 10. We use the two gradients to calculate a new weight for leaf 1, $w_{1,2} = b$. (7) We repeat the steps (5)-(6) for all observations. (8) After reading the last observation, a prediction is made, resulting in a new loss and a new gradient. Finally, a new weight for the leaf node is calculated using the gradient of the latest observation and all previous gradients belonging to the same leaf node. These final weights are used to predict new samples that are not in the training set.

LightGBM

Another popular boosting method is LightGBM developed by Microsoft (Ke et al., 2017). LightGBM was initially created as a faster, more computationally efficient alternative to ordinary gradient boosting. Like CatBoost, LightGBM is by default a first-order method that uses gradients but also supports second-order boosting. XGBoost and CatBoost grow their trees level-wise, meaning that leaf nodes closest to the root node are prioritized for splitting (Figure 2.4, left). This forces a symmetric tree structure, which acts as an extra regularization of the model. LightGBM grows trees leaf-wise, which means that it splits leaf nodes based on the greatest reduction in loss, without considering their distance from the root node. This creates non-symmetric trees (see Figure 2.4, right). Leaf-wise growth has the potential to improve accuracy but may also increase the risk of overfitting. LightGBM uses row-wise subsampling as XGBoost, but with a slightly different tweak. Samples with larger gradients (in absolute value) contribute more to the information gain. Therefore, when performing row-wise subsampling, keeping the samples with large gradients present is wise. This ensures that the samples that contribute the most to the information gain are always present when building the tree. This method is referred to as *gradient-based one-side sampling.*

Row-wise subsampling reduces the sample space, and column-wise subsampling reduces

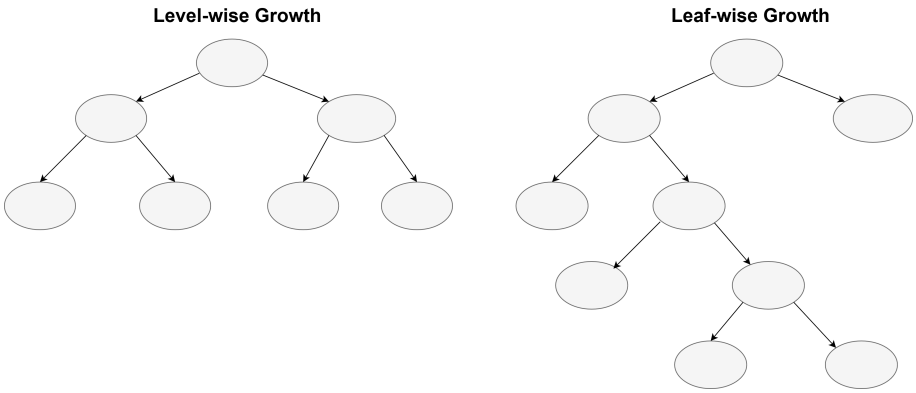


Figure 2.4: Taken from Sand (2023). Different growth strategies for trees. (left) Level-wise growth where nodes closest to the root are prioritized. (right) Leaf-wise growth where splits creating the greatest reduction in loss are prioritized.

the feature space. LightGBM implements a new method to reduce the feature space further called *exclusive feature bundling* (EFB). EFB is a method that bundles features together with near-zero loss in information. By creating bundles of the features, the time complexity of finding split points in the histogram-based approach reduces to $O(\text{Number of bundles} \times \text{Number of bins})$. This increases the time efficiency during training, and by grouping features clearer differentiation between features is also created. The bundling is done by exploiting sparsity in the data and the fact that many features rarely are non-zero at the same observation, resulting in one being able to bundle and merge them safely. We say that two features are mutually exclusive if they never take nonzero values simultaneously.

The bundles are found by formulating the problem as a graph-coloring problem, see Figure 2.5 for some graph-related terminology. The graph-coloring problem is a problem within graph theory where, given a set of colors, one seeks a coloring of the nodes such that no node sharing the same edge has the same color (see Figure 2.6, and Brooks (1941)). In the EFB a graph is constructed where each node is a feature and an edge is added between two nodes if the two features are not mutually exclusive. The features of the same bundle correspond to the edges of the same color.

In practice, there will only be a few features that are entirely mutually exclusive, and a slightly different graph is constructed. The features are still the nodes, but we now use weighted edges where the weight corresponds to the number of mutual conflicts between two features, that is, the number of observations where both of the features are non-zero. One finds the features that can be bundled together by first sorting the graph by the weight of the edges in descending order and then traversing through the features. Starting with a set of bundles consisting of just one bundle, we iterate through the features, and for each feature, we iterate through the set of bundles. For each bundle, one finds the sum of the number of conflicts the feature has with the features in the bundle. The feature is then added to the bundle with the lowest sum of the number of conflicts. However, if the sum of the number of conflicts is larger than some threshold, a new bundle is created, and the feature is added to this empty bundle.

Now that we have found the bundle to which each feature belongs, the next step of EFB is to merge the bundles' features. EFB extends the histogram algorithm further. Recall that features are placed in specific bins, and that the bins are used to evaluate the split points. Each of the features in a bundle has its own set of bins. EFB first looks at the first bin for all the features and then adds different offsets to all the first bins so that the bins

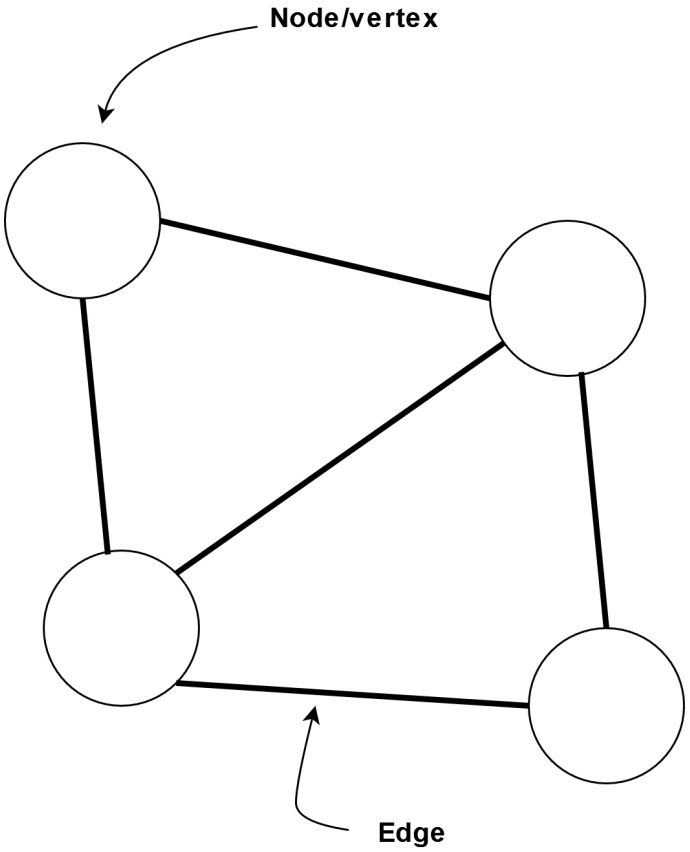


Figure 2.5: A simple graph consisting of four nodes (circles) and five edges (black lines). Taken from Sand (2023)

are disjointed. The first bin of the bundle then ranges from the smallest offset-ed first bin to the largest offset-ed first bin. The procedure is then repeated for all the following bins of the features, creating a single set of bins for the bundle. This may seem a bit cryptic, but consider the example presented by Ke et al. (2017), where we have a bundle consisting of feature A whose first bin is $[0, 15)$ and feature B whose first bin is $[10, 20]$. If we now add an offset of 10 to the bin of feature B, the two bins are disjoint, and the merged bin is $[0, 30]$, which becomes the bundle's first bin.

2.5.6 Linear Regression as Base Learner

While binary regression trees are by far the most used base learners in gradient boosting, other choices of base learners are possible. The most common model used in all statistics is the linear model,

$$\begin{aligned} &\mathbf{y} = \mathbf{X}\boldsymbol{\beta}, \quad \mathbf{y} \in \mathbb{R}^n, \mathbf{X} \in \mathbb{R}^{n \times p}, \boldsymbol{\beta} \in \mathbb{R}^p, \\ &\hat{\boldsymbol{\beta}} = \arg\min_{\beta} L(\mathbf{y}, \mathbf{X}\boldsymbol{\beta}) \ , \end{aligned}$$

which yields great performance in situations where there is a linear relationship between the variables and little multicollinearity. However, in situations where there is a high degree of multicollinearity or more features than observations, the linear model performs poorly. A way to combat such issues is by imposing regularization on the linear model,

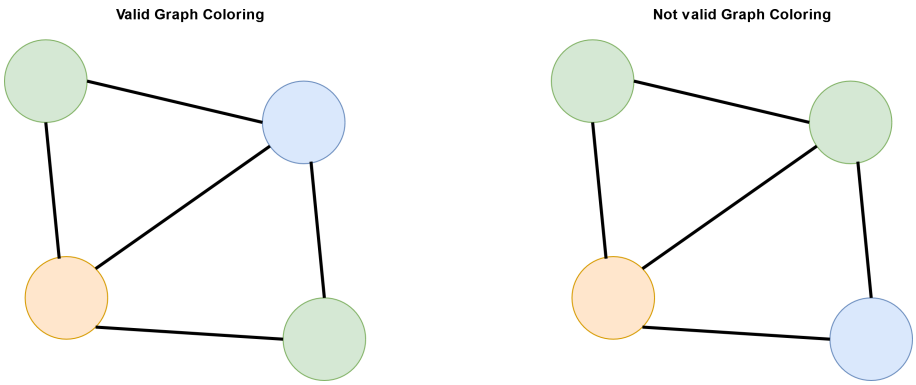


Figure 2.6: Two simple graphs illustrating the graph coloring problem, taken from Sand (2023). (left) A valid colored graph where no nodes that share an edge have the same color. (right) Not a valid colored graph as the two green nodes share an edge.

regularization reduces the variance in the linear model by increasing the bias by adding a penalty term on the objective. By imposing both l_1 and l_2 regularization on the linear model we end up with the naive elastic net model (Zou and Hastie, 2005),

$$\hat{\boldsymbol{\beta}}_{\text{ElasticNet}} = \arg\min_{\beta} L(\mathbf{y}, \boldsymbol{\beta}\mathbf{X}) + \lambda||\boldsymbol{\beta}||_2^2 + \alpha||\boldsymbol{\beta}||_1 \ . \tag{2.27}$$

The naive elastic net model handles both multicollinearity and performs model selection during training, which means it can handle high-dimensional data. The Elastic net is therefore a more versatile base learner to use in gradient boosting than the ordinary linear model. Using the gradient boosting procedures (Algorithm 3 or Algorithm 4), one can fit multiple elastic net models sequentially and thus keep minimizing the loss across multiple iterations. The elastic net can be fitted through a coordinate descent optimization, which we do not explain here, but we refer the reader to Friedman et al. (2010) for an explanation.

2.5.7 Piece-wise Linear Regression Trees

A variant of regression trees that has increased in popularity as a base learner is the piece-wise linear regression tree (Shi et al., 2019). The idea is to replace the constant weights in a leaf node with a linear model. Meaning that each region R_j has a linear model of the form

$$t_j(\mathbf{x}) = b_j + \sum_{m \in M_j} \alpha_{m,j} x^m \ ,$$

where b_j is the intercept term, $\alpha_{m,j}$ is the coefficient belonging to feature m and x^m is feature value m of $\mathbf{x}$. Each leaf node performs model selection such that we end up using a subset $M_j \subseteq \{1, \ldots, p\}$ of the features. By fitting a linear model for each of the regions of the tree, we make the trees more capable of interpolating within the training data and extrapolating beyond the training data scope. The time complexity increases as a linear model is more computationally complex, but the interpolating property decreases the number of trees needed to reach convergence (Shi et al., 2019). The tree can now be expressed as

$$f(\mathbf{x}) = \sum_{j=1}^{J} I(\mathbf{x} \in R_j) t_j(\mathbf{x}).$$

Using the same tree-fitting strategy as XGBoost, the objective ((2.23)) can now be expressed on the leaf level as

$$\hat{f}_b(\mathbf{x}) = \arg\min_f \sum_{j=1}^{J} \sum_{i \in I_j} -g_{ib} t_j(\mathbf{x}_i) + \frac{1}{2} h_{ib} t_j(\mathbf{x}_i)^2 + \Omega(t_j),$$

where $\Omega(t_j)$ is the regularization term on t_j. Let now $\mathbf{X}_j$ be the subset of features belonging to leaf node j including a column of ones, $\mathbf{g}_{bj}$ be the vector of gradients belonging to leaf node j, $\mathbf{H}_{bj} = \text{diag}(h_{ib})_{i \in I_j}$ and $\boldsymbol{\alpha}_j = \{b_j\} \cup \{\alpha_{m,j}\}_{m \in M_j}$. By imposing an l_2 regularization on the coefficients, the objective is now

$$\hat{f}_b(\mathbf{x}) = \arg\min_f \sum_{j=1}^{J} \frac{1}{2} \boldsymbol{\alpha}_j^T \left(\mathbf{X}_j^T H_{bj} \mathbf{X}_j + \lambda \mathbf{I} \right) \boldsymbol{\alpha}_j - \mathbf{g}_{bj}^T \mathbf{X}_j \boldsymbol{\alpha}_j, \tag{2.28}$$

where $\mathbf{I}$ is the identity matrix. Thus, for a fixed leaf node j, by derivation to zero the optimal value of $\boldsymbol{\alpha}_j$ is

$$\boldsymbol{\alpha}_j = \left(\mathbf{X}_j^T H_{bj} \mathbf{X}_j + \lambda \mathbf{I} \right)^{-1} \mathbf{X}_j^T \mathbf{g}_j. \tag{2.29}$$

Shi et al. (2019) now suggest inserting (2.29) into (2.28) giving an optimal loss similar to what was done for XGBoost, and then use this optimal loss to evaluate the splits by defining a new gain (such as in (2.26)). However, this would involve calculating the inverse of potentially large matrices multiple times during the search for the best split. Therefore, implementations such as LightGBM find the split points the usual way, as if one were using ordinary regression trees, and after the tree structure is established (2.29) is used to fit a linear model for each of the leaf nodes.

To avoid using all existing features $m = 1, \ldots, p$ as covariates in the model, a subset M_j of the features is used. M_j is the features used as split points on the path to leaf node j, meaning that M_j is the features used to define R_j. The rationale behind this approach is that the features used for the split points are the ones that contribute the most to the reduction of the loss.

2.6 Explainable AI

The improved performance of complex models used in machine learning, such as gradient boosted models, comes at the cost of interpretability, leading to a trade-off between model performance and model interpretability. As a consequence, a new field called *explainable AI* has emerged in an attempt to develop methods that can explain specific models, called *model-specific* explanations, and methods that explain any models, called *model-agnostic* explanations. Further explainable AI separates between explaining a specific prediction $\hat{y} = \hat{f}(\mathbf{x})$, called *local explanations*, and explaining the model in its entirety, called *global explanations*. An explanation here refers to quantification and understanding of how much the features $\{\mathbf{x}^m\}_{m=1}^p$ impact the model prediction. A widely used *global model-agnostic* method are the Shapely values (Shapley, 1953), which fulfill a set of desirable properties but often are impossible to calculate due to computational costs. This thesis uses a *local model-specific* and efficient version of Shapley values called Tree Shapley additive explanations (Tree SHAP; Lundberg, Erion et al., 2020) to gain global explanations by aggregating the local explanations. Therefore, we first review the Shapley values before outlining the Tree SHAP method.

2.6.1 Shapely Values

Shapely values originate from game theory and determine how one should divide a payoff between a set of players based on how much they contribute (Shapley, 1953). Assuming that we have p players, and let F be the set of all possible combinations of the p players. Let $v : S \to \mathbb{R}, S \subseteq F$ be a *contribution function* that tells us the *expected payoff* for the players in $S \subseteq F$. The Shapley value for each player $m = 1, \ldots, p$ is defined as

$$\phi_m(v) = \sum_{S \subseteq F \setminus \{m\}} \frac{|S|!(p - |S| - 1)!}{p!} \Big(v(S \cup \{m\}) - v(S) \Big) \ ,$$

where S is any subset of players in F that do not contain player m. We sum over all sets to account for the fact that the effect of removing or adding player m depends on the other players that are present. The Shapley value for player m is a sum of the difference between the contribution with player m, $v(S \cup \{m\}))$ and the contribution without player m, $v(S)$, over all possible combinations of players. Some factorial terms weight each term in the sum. To understand the factorial terms, we use the definition of the binomial coefficient to rewrite the Shapely value on form

$$\phi_m(v) = \frac{1}{p} \sum_{S \subseteq F \setminus \{m\}} \frac{v(S \cup \{m\}) - v(S)}{\binom{p-1}{|S|}} \ . \tag{2.30}$$

The Shapley value is thus a weighted mean for all possible differences between the contribution with player m and without player m, with the weight being the number of ways to create a set of cardinality $|S|$ out of the $p - 1$ players. In other words, the Shapley value tells us the average expected marginal contribution of player m in a coalition of p players after considering all possible combinations of players. It can be shown that the Shapely values are the unique solution to a set of four properties (Shapley, 1953; Young, 1985). The first property is the property of *efficiency* which states that the total payoff is distributed over all Shapley values,

$$v(F) = \sum_{m=1}^{p} \phi_m(v) \ .$$

The second property is *symmetry*, which states that two players with equal contributions have equal Shapley values,

$$v(S \cup \{m\}) = v(S \cup \{k\}) \quad \forall \ S \subseteq F \setminus \{m, k\} \implies \phi_m(v) = \phi_k(v) \ .$$

The third property is the *dummy player* property, namely that a player that does not contribute to any coalitions has zero Shapley value,

$$v(S \cup \{m\}) = v(S) \quad \forall \ S \subseteq F \setminus \{m\} \implies \phi_m(v) = 0 \ .$$

The last property is the *linear* property, which states that if a game has a payoff determined by the sum of two different contribution functions v and w, then the Shapely value for all players $m = 1, \ldots, p$ is

$$\phi_m(v + w) = \phi_m(v) + \phi_m(w) \ .$$

The first property ensures that the total payoff from the game is divided among all players. The second and third property ensures fairness of the division where similar players are treated equally, and players with no contribution are given no payoff. The last property

makes it possible to deploy Shapely values on more complex games with multiple contribution functions in play. These properties make the Shapely values a sensible way to divide the payoff.

In explainable AI, the players $m = 1, \ldots, p$ are the features in the model, and the contribution function is a measure of model quality. Lipovetsky and Conklin (2001) showed how one can use the proportion of variance explained (R^2) as a contribution function for linear models to gain global explanations. One can also use the Shapley values to explain a single prediction $\hat{f}(\mathbf{x})$ by letting the contribution function be the model itself $\hat{f}$. Let $\hat{f}_S(\mathbf{x})$ be the model trained with features $S \subseteq \{1, \ldots, p\}$, then the Shapley regression formula for an observation $\mathbf{x}$ and a feature $m = 1, \ldots, p$ is

$$\phi_m(f, \mathbf{x}) = \sum_{S \subseteq F \setminus \{m\}} \frac{|S|!(p - |S| - 1)!}{p!} \Big(\hat{f}_{S \cup \{m\}}(\mathbf{x}) - \hat{f}_S(\mathbf{x}) \Big). \tag{2.31}$$

S is a subset of the set of all possible combinations of features, F, excluding feature m. $\hat{f}(\mathbf{x})_{S \cup \{m\}}$ is the model trained with the features in S and feature m, while $\hat{f}(\mathbf{x})_S$ is the model trained with the features in S, which do not include m. In the Shapley regression setting, the Shapley value of a feature m estimates the expected increase of a prediction $\hat{f}(\mathbf{x})$ when including feature m in the model. The problem with both the global R^2 method and using $\hat{f}(\mathbf{x})$ as contribution function is that the number of models $\hat{f}_S(\mathbf{x})$ that have to be fitted snowballs out of hand as the number of features increase. As we in each subset S choose m features out of p, the total number of combinations we have to iterate through to calculate (2.31) is

$$\sum_{m=1}^{p} \binom{p}{m} = 2^p - 1 \ ,$$

by the binomial theorem. As we have to repeat this process for all features $m = 1, \ldots, p$, we get an exponential time complexity of $O(p2^p)$, on top of the time complexity of fitting the models. SHAP (Lundberg and Lee, 2017) and TreeSHAP (Lundberg, Erion et al., 2020) introduce a methodology to reduce the time complexity, which we will cover in the next subsection.

2.6.2 Tree SHAP

Shapely Additive Explanations (SHAP; Lundberg and Lee, 2017) gives a local explanation, meaning it explains the prediction of a specified observation $\mathbf{x}_*$ and considers the following setup of Shapely values. The total payoff is the prediction of the observation to be explained, $\hat{f}(\mathbf{x}_*)$. Let $\mathbf{x}^S$ denote the subvector of $\mathbf{x}$ that only contains the features in S. The contribution function $v(S)$ used by SHAP is the expected prediction conditioned on having the features in S set to be those in $\mathbf{x}_*^S$, that is

$$v(S) = f_{\mathbf{x}_*}(S) = E\Big[\hat{f}(\mathbf{x}) | \mathbf{x}^S = \mathbf{x}_*^S \Big] \ . \tag{2.32}$$

The SHAP values for $\mathbf{x}_*$ are then the Shapley values such that the decomposition of the prediction $\hat{f}(\mathbf{x}_*)$ is

$$f(\mathbf{x}_*) = \phi_0(\hat{f}) + \sum_{m=1}^{p} \phi_m(\hat{f}, \mathbf{x}_*) \ , \quad \phi_0(\hat{f}) = E\Big[\hat{f}(\mathbf{x}) \Big] \ , \tag{2.33}$$

where $\phi_m(\hat{f}, \mathbf{x}_*)$ is now denoted the SHAP value for feature m of the prediction $\hat{f}(\mathbf{x}_*)$. There are two things worth stressing with this formulation. First, we have turned the Shapely values into a linear local explanation model. One can view (2.33) as a surrogate linear model for $\hat{f}(\mathbf{x})$ near the point $\mathbf{x} = \mathbf{x}_*$. However, there is no explicit dependency on $\mathbf{x}$ after the SHAP values $\phi_m(f, \mathbf{x}_*)$ are obtained. Therefore, one cannot use SHAP values to anything other than assessing feature importance. Second, the SHAP values are calculated relative to the globally expected prediction $\phi_0(\hat{f}) = E[\hat{f}(\mathbf{x})]$, which gives the interpretation that SHAP values are the explanations of the difference between the average prediction and the prediction of $\hat{f}(\mathbf{x}_*)$.

The reason for these alternative Shapely values (compared to the Shapley regression values) is to avoid fitting an exponential number of models, and Lundberg and Lee (2017) show that SHAP values satisfy three important properties similar to the original Shapely values. The first one is the property of *local additivity*, which says that the sum of the Shapley values of a prediction $\mathbf{x}$ is equal to the original prediction, that is

$$\hat{f}(\mathbf{x}) = \phi_0(\hat{f}) + \sum_{m=1}^{p} \phi_m(\hat{f}, \mathbf{x}).$$

Additivity is essentially the same as the efficiency property. The second property is the property of *missingness*, which states that features with no effect are assigned a Shapley value of zero. That is,

$$\begin{aligned} f_{\mathbf{x}}(S \cup \{m\}) = f_{\mathbf{x}}(S) \quad \forall\ S \subseteq F \\ \implies \phi_m(\hat{f}, \mathbf{x}) = 0\ . \end{aligned}$$

The third property is the property of *consistency*, which states that, if a different model $\hat{f}^*$ is such that the feature m increases the prediction more than for $\hat{f}$, then feature m's Shapley value for $\hat{f}^*$ is greater than for $\hat{f}$. More precisely, the property of consistency states that

$$\begin{aligned} f^*_{\mathbf{x}}(S) - f^*_{\mathbf{x}}(S \backslash \{i\}) \geq f_{\mathbf{x}}(S) - f_{\mathbf{x}}(S \backslash \{i\}) \\ \implies \phi_m(f^*, \mathbf{x}) \geq \phi_m(f, \mathbf{x}) \quad \forall\ S \subseteq F\ . \end{aligned}$$

The first and second properties give an interpretation of the SHAP values, as they represent a decomposition of the prediction into each feature's contribution, where contributions from unimportant features are set to zero. The property of consistency ensures that the SHAP values of different models are comparable. We now must find a way to calculate the contribution function $v(S) = f_{\mathbf{x}}(S)$ in (2.32). In the original formulation of SHAP values, a model-agnostic algorithm is proposed, denoted Kernel SHAP (Lundberg and Lee, 2017). However, there exists a model-specific method for trees denoted Tree SHAP (Lundberg, Erion et al., 2020) that allows us to calculate the contribution function and the SHAP values efficiently for regression trees. We can exploit this algorithm to calculate the SHAP values of each tree in a boosting ensemble. Then, using the linearity property of Shapely values, we can sum up the SHAP values of each tree to gain the SHAP values for the entire ensemble.

The Tree SHAP (Lundberg, Erion et al., 2020) algorithm approximates the conditional expectation $f_{\mathbf{x}}(S)$ without the need to refit the model for each subset of features S. Instead, Tree SHAP uses the defined structure in the trees, and traverses one tree at a time until we either reach a leaf node R_j or a split point that uses a feature $r \notin S$. If we reach a leaf node, we use the weight w_j as the tree's prediction of $\mathbf{x}|\mathbf{x}^S = \mathbf{x}^S_*$. If we reach a split point that uses a feature not in S, we traverse both the left and right child until we

reach all leaf nodes. Then, we take a weighted average of all the leaf nodes' weights, where we weigh by how many training samples belong to each leaf node. This approach ensures that every tree in the boosting ensemble contributes with a prediction for $\mathbf{x}|\mathbf{x}^S = \mathbf{x}_*^S$, either with one weight directly or with a weighted average of weights. We then obtain an estimate of $f_{\mathbf{x}}(S)$, denoted $\hat{f}_{\mathbf{x}}(S)$, by summing up the predictions in the usual boosting fashion. To traverse one tree is proportional to the number of leaves in the tree J, and, as we have to traverse all B trees the time complexity of estimating $f_{\mathbf{x}}(S)$ is $O(JB)$. Thus, if we had plugged $\hat{f}_{\mathbf{x}}(S)$ in (2.31) we would get a time complexity of $O(JBp2^p)$, still an exponential time complexity.

Tree SHAP further reduces the time complexity by exploiting the trees' structure to calculate (2.31), allowing them to compute the Shapley value for all possible subset S simultaneously. The Tree SHAP path dependent algorithm which is used in this thesis has a time complexity of $O(JBD^2)$, where D is the maximum depth of a tree. However, the implementation details of the polynomial time algorithm are beyond the scope of this thesis, see Lundberg, Erion et al. (2020) for details.

Although SHAP values is a local explanation method, it is possible to use SHAP to gain global explanations. The SHAP value of a feature tells us the mean of the average contribution a feature gives when included in a fixed-size set of features over all features for a specific observation ((2.30)). If we for all features take the mean absolute value of each observation's Shapley value, we obtain the mean magnitude of the feature's contribution to all predictions. Therefore, the mean absolute SHAP value,

$$\Phi_m(f) = \frac{1}{n}\sum_{i=1}^{n} |\phi_m(f, \mathbf{x}_i)|, \quad m = 1, \ldots, p \ ,$$

is a reasonable global feature importance measure.

3 Methods

In this chapter, we describe what methods have been deployed, as well as the data used. All the methods used aim at performing genomic prediction in wild populations for quantitative traits, that is, to build models that predict the genomic contribution to an individual's phenotype as a function of the genomic data. We investigate the ability of boosting models to perform genomic prediction within populations and across populations. We also make use of an explainable AI method called Shapely values to investigate the importance of the different SNPs in the models and compare it to genome-wide efficient mixed model association (GEMMA; Zhou and Stephens, 2012), a standard genome-wide association study (GWAS) method.

3.1 The House Sparrow Data

The genomic and morphological data used in this thesis comes from a meta-population of house sparrows consisting of several island populations located off the Helgeland coast of Norway (Ringsby et al., 2002; Jensen, Steinsland et al., 2008; Jensen, Moe et al., 2013; Baalsrud et al., 2014). Each island is a population in itself, with varying environmental conditions, but due to migration between the islands, there is relatedness across the islands making the whole island system a meta-population. The meta-population has been studied since 1993, such that the monitored individuals are studied from birth to death through seasonal environmental conditions. Genomic data about adult individuals are available through sequenced single-nucleotide polymorphism (SNP) data, which we use to investigate how the two traits of body mass and tarsus length are modeled as a function of the SNP data.

We have used two different datasets from the Helgeland system, which differ in how many SNPs are sequenced and in how many individuals are recorded. The first dataset, which we will refer to as the 180K dataset, contains 4371 body mass recordings of 1918 unique individuals and 4495 tarsus length recordings of 1915 unique individuals, where each individual has been sequenced for 182 854 SNPs. The second data which we call the 70K dataset contains 12 987 body mass recordings of 3442 unique individuals and 12 870 tarsus length recordings of 3442 unique individuals, where each individual has been sequenced for 65 245 SNPs. The 180K dataset thus gives higher exposure to variation in the genetic material, but at the cost of having less exposure to differences between individuals compared to the 70K dataset. The genomic data is quality controlled and preprocessed by using the PLINK software (Chang et al., 2015; Purcell and Chang, 2023). Non-genetic and non-phenotypic data related to the house-sparrows used in this thesis are listed in Table 3.1, all the variables was available for both the 180K and the 70K data, except for "month" which only were available in the 180K data.

Table 3.1: Names of non-genetic variables and a short description.

Variable name	Description
ringnr	A unique identifier of each individual
hatchisland	The island where the first recording was made
Age	Age of the individual
Sex	The sex of the individual
month	The month of the recording
hatchyear	The year of birth

3.2 Prediction of Genetic Values

This thesis aims to improve methods on how to perform genomic prediction, that is, how to predict the genetic component of each individual's phenotype using genomic data. The true underlying genetic contribution is unknown because the data is from a real wild population. As the environment influences both the phenotype and the genetics, there is a chance that the models will pick up on the environmental effects that are exposed in the genetic data. In this thesis we are only interested in modeling the true genetic effect, meaning that the environmental effect must be removed prior to training. We, therefore, perform a preprocessing step to obtain a pseudo-response where we adjust the phenotype for environmental factors by first fitting a linear mixed model with a random identity effect. Then, using the identity effect as a new pseudo-response, the models are trained with the SNP data as covariates. The linear mixed model is of the form

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{W}\mathbf{u} + \boldsymbol{\gamma} + \boldsymbol{\varepsilon} \ , \tag{3.1}$$

where $\boldsymbol{\beta}$ contains the fixed environmental effects, $\mathbf{u}$ contains the random environmental effects, $\boldsymbol{\gamma}$ contains the random identity intercepts, and $\boldsymbol{\varepsilon}$ is the residual term. $\mathbf{X}$ and $\mathbf{W}$ are design matrices of appropriate dimensions. The term $X\boldsymbol{\beta}+W\mathbf{u}$ thus explains the variance due to differences in observed environmental factors and $\boldsymbol{\varepsilon}$ explains the variance due to differences in unobserved factors. As $\boldsymbol{\gamma}$ is the identity effect it explains the variance due to differences between individuals that are not due to environmental factors. This means that $\boldsymbol{\gamma}$ contains the contributions to the phenotype due to individual genetic factors. Therefore, we can reasonably use the estimated value of the identity, $\hat{\boldsymbol{\gamma}}$, to create a new pseudo-phenotype

$$\mathbf{y}^* = \hat{\boldsymbol{\gamma}} \ , \tag{3.2}$$

which we wish to predict and explain using the SNP-marker data $\mathbf{Z}$ by building a model $\hat{f}$ such that $\hat{f}(\mathbf{Z})$ estimates $\mathbf{y}^*$, we say that $\hat{\mathbf{y}}^* = \hat{f}(\mathbf{Z})$. The model's performance is measured by the Pearson correlation between the predicted genetic component $\hat{\mathbf{y}}^*$ of each individual and the mean phenotype value of each individual. If we have a total of n unique individuals and $\mathcal{I}_j$ is the set of indices of observations belonging to individual $j \in \{1, \ldots, n\}$, the mean phenotype value is thus calculated as $\bar{y}_j = \frac{1}{|\mathcal{I}_j|} \sum_{i \in \mathcal{I}_j} \mathbf{y}_i$. The model's performance is evaluated by

$$\text{Corr}(\hat{\mathbf{y}}^*, \bar{\mathbf{y}}), \quad \bar{\mathbf{y}} = \{\bar{y}_j\}_{j=1}^n \ ,$$

where

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

is the sample Pearson correlation between two vectors $\mathbf{x} = \{x_i\}_{i=1}^n$ and $\mathbf{y} = \{y_i\}_{i=1}^n$ with mean $\bar{x}$ and $\bar{y}$. This procedure is the same as the two-step procedure described by Sand (2023) and similar to the one used for real plant data by Lourenço et al. (2024).

3.3 Within- and Across-Population Predictions

The purpose of genomic prediction in wild populations can be divided into two main branches: genomic prediction *within* populations and genomic prediction *across* populations. Within-population predictions refer to training and testing on data from the same population, meaning that the model's ability to identify patterns in the genomic data that explain differences between individuals from the same population is tested. In across-population predictions, the model is trained on data from one population and then tested on data from a different population, which tests the model's ability to extrapolate the patterns it has learned in one population to another. Within-population predictions are more suitable when there is one particular population's dynamics we are interested in, but also to get a better understanding of microevolution (McGaugh et al., 2021; Hunter et al., 2022). Breeding values from genomic prediction are also much used in plant and animal breeding to predict the quality of an individual (Meuwissen et al., 2016).

Across-population predictions, on the other hand, are more about the model's general ability to identify effects that are relevant outside the training data. Achieving good results in across-population predictions is often more challenging, as the actual SNP effect may differ between populations. More generally, the local population one tests on may have a different distribution in the genetic effect due to different environmental conditions that can cause local adaptations. Another reason for the difference in SNP effects may be due to changes in the linkage disequilibrium between the recorded SNP and the causal SNP.

The Helgeland system is a meta-population consisting of several islands where we either can treat the entire system as one population, or view each island as a separate population from the other islands. When performing within-population predictions, all islands are present in the data and there is no restriction on island location when making training and test sets, such that the model may be tested on individuals from an island it has seen before (the island, not the individuals). By using a 10-fold cross-validation procedure, we obtain multiple sample points of the model's performance, yielding an estimate of the model's true performance together with an uncertainty estimate.

For the across-population predictions we use a procedure we will refer to as *island-for-island* predictions, where we treat the eight largest islands, namely Nesøy, Myken, Træna, Selvær, Gjerøy, Hestmannøy, Indre Kvarøy, and Aldra, as separate populations. Next, in a cross-validation-like manner, we iterate through each island and let the data from the current island be the test set while the data from all other islands goes into the training set. The island-for-island procedure thus results in eight data points of the model's performance in an across-population prediction setting, giving an uncertainty estimate of the model performance as well.

For the within-population procedure we fit models both on the 180K dataset and the 70K dataset, whereas for the across-population procedure, only the 70K dataset is used in order to have reasonably large sample sizes.

3.4 Description of Employed Models

In this section, we describe the models that were tested in this thesis. The core contribution of this thesis is the investigation of gradient boosting models' ability to perform genomic prediction in a wild population. Although tested in animal and plant breeding (Li et al., 2018; Chafai et al., 2023; Lourenço et al., 2024), gradient boosting has never been employed in a wild population setting. Wild populations differ from breeding populations by having a more varying environment, which leads to wild populations having more heterogeneous populations. As it is easier to monitor and record data from a breeding population than a wild population, the breeding field datasets contain more observations of unique individuals than datasets from wild populations. This raises a challenge in modeling wild populations, as we need flexible models that capture genetic diversity in heterogeneous populations with less data than the breeding fields. We investigate whether gradient boosting can yield the necessary framework for genomic prediction in wild populations. As stated in Section 2.5, gradient boosting is a general procedure that builds an ensemble of base learners, where the base learners can be any model (Friedman, 2001). We test three different choices of base learners, binary regression trees (Breiman et al., 1984), elastic net (Zou and Hastie, 2005) and piecewise linear regression trees (Lefakis et al., 2019), and compare them to a Bayesian animal model.

The most popular versions of gradient boosting models are tree-based boosting models (Friedman, 2001; Chen and Guestrin, 2016; Ke et al., 2017; Prokhorenkova et al., 2017), meaning that they use a binary regression tree (Breiman et al., 1984) as base learners. As trees split the feature space by iteratively look for the best feature to split, trees perform by default feature selection. Consequently, trees work well with high-dimensional data. Binary regression trees divide the feature space into separate regions by repeatedly splitting a feature axis into two, making them suitable for modeling non-linear interactions between features. Trees can, therefore, model epistatic effects. We test out some of the most popular versions of tree boosting, namely XGBoost (Chen and Guestrin, 2016), LightGBM (Ke et al., 2017), CatBoost (Prokhorenkova et al., 2017) and an ordinary gradient boosting machine (GBM; Friedman, 2002). All models are implemented in Python using their respective libraries (for GBM we use the implementation provided by the Scikit-learn package (Pedregosa et al., 2012)). See Section 2.5.5 for detailed explanations of differences between the different versions of gradient-boosted trees.

Linear boosting is a version of gradient boosting where the base learner is a linear model (see Section 2.5.6). While binary regression trees are the most popular base learners, there are situations where trees may be a bad choice for base learners. As trees naturally assume strong interactions between features, they are a poor choice when the features are independent. Another weakness of trees is that they assign each region a constant weight calculated from the training observations belonging to that region. This means that if a tree is exposed to a new data point far away from those used in training, it will place the new data point in its nearest belonging region and use the region's weight as the predicted value. In other words, trees cannot extrapolate beyond their training range. A linear model may be more suitable in both situations as it assumes independent features and can extrapolate beyond its training range. A linear model that handles both correlated features and high-dimensional data is the elastic net, an already popular method in genomic prediction (Ogutu et al., 2012; Wang et al., 2019; Lourenço et al., 2024). The elastic net may thus be a sensible base learner to use in the hope that the boosting approach will improve its already stable performance. We use the XGBoost framework to fit a gradient boosting model with the elastic net as a base learner on the

genomic data. We will refer to such a model as a linear booster.

There may be situations where we desire the non-linear interaction modeling of trees but also want the model to extrapolate (or interpolate) beyond its training range. In such a case, the piecewise linear regression tree (Lefakis et al., 2019; Shi et al., 2019) may be a good base learner. As explained in Section 2.5.7, a piecewise linear regression tree is an extension of the binary regression tree, where instead of having a constant weight in each leaf node, one fits a linear model for each leaf node. This way, based on a non-linear interaction, the model chooses which linear model is most appropriate when exposed to a new data point. We use the LightGBM framework to fit a gradient-boosting model with piecewise linear regression trees. We sometimes denote such a model as a linear tree booster.

The genomic animal model (Henderson, 1984; Kruuk, 2004) is one of the most commonly used models in genomic prediction. The genomic animal model is a statistical parametric linear model where the breeding value $\mathbf{g}$ is explicitly included as a random intercept placed on the identity of the individuals, see Section 2.1.3. To get a better hold on the quality of the boosting models' performance, we use the genetic estimated breeding values (GEBV) from a fitted state-of-the-art Bayesian animal model (O'Hara et al., 2008; Ovaskainen et al., 2008) to act as a "benchmark model". The Bayesian animal model is fitted using the integrated nested Laplace approximation (INLA), see Section 2.2.

3.5 Loss functions

From Section 3.5, we know that the loss function that the model is trained to minimize determines what part of the data the model views as important. The choice of the loss function is thus a dimension that can considerably impact the model performance. A squared error loss, which is the most commonly used, will, as the name implies, pay quadratically more attention to the errors than an absolute loss. A sample with high error will then receive more attention from the model, such that the model adapts to said sample to decrease the corresponding error. This is positive if all samples and their corresponding predictions are equally representative of the general population. However, if the sample associated with the high error, in reality, is an extreme value, it ends up receiving too much attention. In the setting of across-population predictions, we fear that by using squared error loss, the models get too well-adjusted to their training islands by focusing too much on the extreme cases of the different islands, instead of the average samples across the islands.

To investigate this potential behavior we fit models for the across-population setting and see how the choice of loss function affects the model performance. Specifically, we train a linear tree booster using mean squared error (MSE),

$$L(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2,$$

mean absolute error (MAE),

$$L(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i|,$$

25% quantile loss,

$$L(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{n} \sum_{i=1}^{n} \begin{cases} 0.75 \cdot (\hat{y}_i - y_i), & \hat{y}_i > y_i \\ -0.25 \cdot (\hat{y}_i - y_i), & \hat{y}_i \leq y_i \end{cases},$$

and 75% quantile loss,

$$L(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{n} \sum_{i=1}^{n} \begin{cases} 0.25 \cdot (\hat{y}_i - y_i), & \hat{y}_i > y_i \\ -0.75 \cdot (\hat{y}_i - y_i), & \hat{y}_i \leq y_i \end{cases},$$

for both body mass and tarsus length. MAE estimates the conditional median, which is more robust to outliers than the expected value (which is what the MSE estimates). The 25% quantile loss estimates the 25% quantile and yields more attention to cases where the model overestimates the true value. Similarly, the 75% quantile loss estimates the 75% quantile and applies more focus to where the model underestimates. The performance of the losses therefore gives insight into what part of the data is more important when performing across-population predictions. See Section 2.3.1 for more details about the differences between the losses and their properties.

3.6 Interpretability: GWAS and SHAP

Another interesting aspect of genomic prediction is identifying the SNPs that are most associated with the phenotype of interest. Identification of candidate SNPs can help us better understand the relationship between the phenotype and the genomic material, making it easier to forecast the consequences of a potential evolution (McGaugh et al., 2021). Genome-wide association studies (GWAS) are used to determine the association between a SNP and a phenotype by regressing the SNP on the phenotype and then calculating the p-value of the SNP's regression coefficient from a likelihood ratio test (Uffelmann et al., 2021). The univariate method of GWAS has been proven helpful in identifying causal loci associated with diseases such as diabetes and Crohn's syndrome (The Wellcome Trust Case Control Consortium et al., 2007). However, when researchers from other fields applied the same GWAS method to the quantitative traits, the loci identified only explained a small proportion of the total phenotypic variation. Such phenomena even arose in traits that, from pedigree methods, we knew were strongly heritable (Manolio et al., 2009). Typically, studies with the problem of "missing heritability" have few identified significant SNPs. Researchers have concluded that high-density genome sampling is needed to find the, far more numerous than anticipated, associated SNPs (Yengo et al., 2022).

Most regression models used in GWAS are linear or generalized linear models, which assume independent effects from all SNPs and do not include epistatic effects. Moreover, a GWAS relies on linkage disequilibrium to identify regions containing causal alleles. Even though the effect from a causal allele may be additive and independent from the other effects, it is not certain that the processes regarding the linkage disequilibrium are linear. It could be that some of the recorded alleles around the causal allele are inherited in a non-linear fashion, due to being involved in other processes as well. A consequence of such behavior can be that the associated effect looks non-linear, which makes it undetectable for a univariate linear model. Therefore, a plausible explanation of the missing effects and the need for dense recordings of SNPs, is that the imposed linearity results in insufficient modeling of the SNP effects and the process that can occur around linkage disequilibrium. Therefore, we think that an alternative to the univariate GWAS is needed.

A possible alternative is to use explainable AI on the complex gradient-boosted tree models from the genomic prediction to identify candidate SNPs. As trees are excellent methods to model non-linearities, they have the potential to solve the problems discussed in univariate GWAS. We, therefore, use SHAP (see Section 2.6.2 and Lundberg and Lee, 2017; Lundberg, Erion et al., 2020) to decompose each SNP's contribution to each prediction in the test set of each cross-validation fold. Then, we take each SNP's mean absolute SHAP value across all observations as a global feature importance measure. We further compare the mean absolute SHAP value with the results from a univariate GWAS. A standard univariate GWAS usually requires a dataset of unrelated individuals to avoid clusters in the data that break the assumption of independent observations. There is a high degree of relatedness among the individuals in the Helgeland data. Therefore, we use a mixed-model implementation of GWAS denoted GEMMA (Zhou and Stephens, 2012) to account for the relatedness.

Even though we compare the results from SHAP and GEMMA, we expect little overlap between the two methods in which SNPs are considered most important. The SHAP values are explanations of a gradient boosted tree, which emphasizes interaction effects and non-linearity in the phenotype. GEMMA, on the other hand, is a single-marker linear model, which, therefore, emphasizes the SNPs that have a linear relationship with the phenotype. SNPs with a linear relationship with the phenotype cannot be the same as those with a non-linear relationship with the phenotype. Further, it is also reasonable to think that the SNPs involved in the most considerable epistatic effect are not those with the strongest additive effect. Therefore, the SHAP values of a tree-based model and GEMMA are clearly looking for two different groups of SNP effects.

3.7 Software and Technical Considerations

All code used in this thesis is available at the GitHub repository didrik1812/GenomicPrediction. Details about how to replicate the results are described in the repository's README.md file. Python (version 3.8) has been the main programming language, but R (version 4.4) was also used. The boosting models XGBoost, Catboost and LigthGBM come with existing efficient implementations available at GitHub which we exploited to fit models efficiently. Specifically, we used XGBoost v2.0.3, CatBoost v1.2.2 and LightGBM v4.2.0. To fit the GBM the Scikit-learn framework Sklearn v1.3.0 was used, and the Bayesian Animal model was fitted in R using the R-INLA v4.4.0 package. The Boosting models were fitted by first doing the preprocessing step described in Section 3.2, while the Bayesian animal model does not need this preprocessing step.

The Bayesian Animal model used the non-genetic variables listed in Table 3.1, where "age", "month" where fixed effects, while "hatchyear" and "hatchisland" were set as random effects with Gaussian priors. Two different random intercepts were placed on the identity, the first one being the breeding value with a precision matrix defined from the inverse of the genomic relatedness matrix. The second one has a standard Gaussian prior and exist to account for other permanent environmental conditions that might affect each individual. Lastly, three extra fixed variables were included to ensure the animal model performs at its best. We included the genomic inbreeding coefficient, F_{GRM} as a fixed effect. We included two fixed variables, "other" and "outer" that denotes how much of the genetic material comes from the *outer* island group and the *other* island group. The outer island is the group of the three islands (Myken, Selvær, and Træna) furthest away from the mainland. The islands closest to the mainland are denoted *inner* islands, and

the islands that are neither in the inner or the outer group are denoted other islands. We include the "outer" and "inner" variables to account for how migration might have affected the phenotype and for genetic groups within the population (see Muff et al., 2019). The Bayesian Animal Model was only fitted on the 180K dataset due to computational costs related to calculating and using the inverse of the genomic relatedness matrix for the 70K dataset.

The preprocessing step for the 180K dataset includes the same variables as the Bayesian animal model, except for the breeding value, to give a fair comparsion between the models fitted on the same dataset. For the 70K dataset, we dropped the genetic variables ("FGRM", "outer" and "inner") so that the genetic data is only contained in the SNPs. The linear mixed model used in the preprocessing step to obtain the adjusted phenotype was fitted using the lme4 package available at CRAN. For the boosting models, the adjusted phenotype was standardized prior to training.

All boosting models have a vast number of hyperparameters that impact the model's performance, therefore it is essential to perform hyperparameter tuning to maximize the capabilities of the boosting models. In Sand (2023) Bayesian optimization was used to find the best set of hyperparameters. While Bayesian optimization has a strong theoretical foundation (Frazier, 2018), it suffers from high time complexity and cannot be run in parallel, making cross-validation of the hyperparameters too time-consuming. In this thesis, we therefore instead use randomized search. Randomized hyperparameter search is done by establishing a grid for each hyperparameter, making up a discrete hyper-dimensional search space of all hyperparameters, and then randomly sampling combinations of hyperparameters from this space. A model is trained with the sampled hyperparameters and then tested on the validation set. The set of hyperparameters that results in the best model performance is chosen in the end. As the different samples are independent, one can fit the models in parallel, decreasing the total time spent. As the time consumption is lower, cross-validation on the sampled hyperparameters is possible, giving a more generalizable evaluation of the model's performance. We used a 5-fold cross-validation on the test set, which splits the training set into a validation set and a new training set. To implement the randomized search we used the `RandomizedSearchCV` module provided by the Scikit-learn package (Pedregosa et al., 2012, Sklearn v1.3.0). The hyperparameters tuned for each model and their corresponding search space are shown in Appendix A. All boosting models were trained with mean squared error as loss function, if not else stated.

The SHAP values are obtained using the shap v0.44.1 package, which has implemented the path-dependent Tree SHAP algorithm. We obtain SHAP values on unseen data by explaining the test sets in the 10-fold cross-validation loop. This gives one SHAP value for each SNP in each individual (*i.e.*, the number of SHAP values is #individuals $\times$ #SNPs). The global SHAP importance of each SNP can then be calculated by taking the average absolute SHAP value over all individuals.

4 Results

4.1 Descriptive Data Analysis

Before moving on the model performances, we provide a short data analysis of the two traits and the island system. Figure 4.1 shows the geographical position of the island system and the names of eight of the main islands. The outermost islands Myken, Selvær and Træna are islands without farms, resulting in a less stable supply of food for the sparrows than those on the inner islands with farms. Therefore, it is unsurprising that the islands with the highest populations are farm-islands (Table 4.1). This "farm effect" does not seem to have an impact on the lifetime of the sparrows (Figure 4.2) as all islands have similar lifetime distribution, except for Lurøy and Sleneset where the densities are shifted more towards the lower ages. There are also no major differences in the distributions of body mass and tarsus length across the islands and the sexes (Figure 4.3 and Figure 4.4). However, we see some small, but significant differences between the islands in the distributions of body mass and tarsus length.

We also observe that there is slightly more variation in the body mass recordings, both before and after adjusting the phenotype, for the 180K dataset than the 70K dataset, whereas for tarsus length the variations in the datasets are very similar (Table 4.2). The adjusting of the phenotype (see Section 3.2) does not lose the information contained in the phenotype for tarsus length in both the 70K and the 180K dataset, nor for body mass in the 180K dataset (Table 4.3). However, some information for body mass is lost in the 70K dataset as the correlation is significantly lower than 1 (Table 4.3). We use the term "loss of information" here when we refer to a decrease in correlation between the adjusted and actual mean phenotype, as a decrease implies that the linear relationship between the variables is weakened.

Table 4.1: Number of unique recorded individuals per island in the 70k dataset.

Island	**#Individuals**	**Type of Island**
Hestmannøy	1033	Farm island
Gjerøy	543	Farm island
Indre Kvarøy	374	Farm island
Træna	271	Non-farm island
Selvær	223	Non-farm island
Sleneset	189	Non-farm island
Onøy	183	Farm island
Aldra	173	Farm island
Lovund	147	Non-farm island
Nesøy	133	Farm island
Myken	89	Non-farm island
Lurøy	84	Farm island

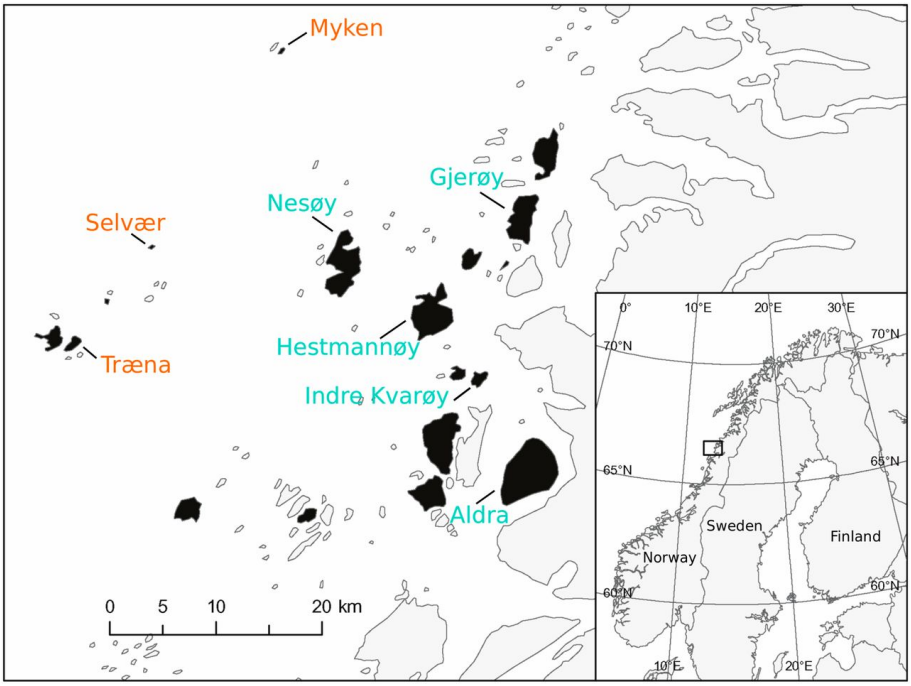


Figure 4.1: Map from Niskanen et al. (2020) showing the islands in the Helgeland system and naming some islands. Islands with black fill color are populated with house sparrows (18 in total), while gray ones are not. Orange font color refers to islands without farms, and turquoise font color refers to islands with farms.

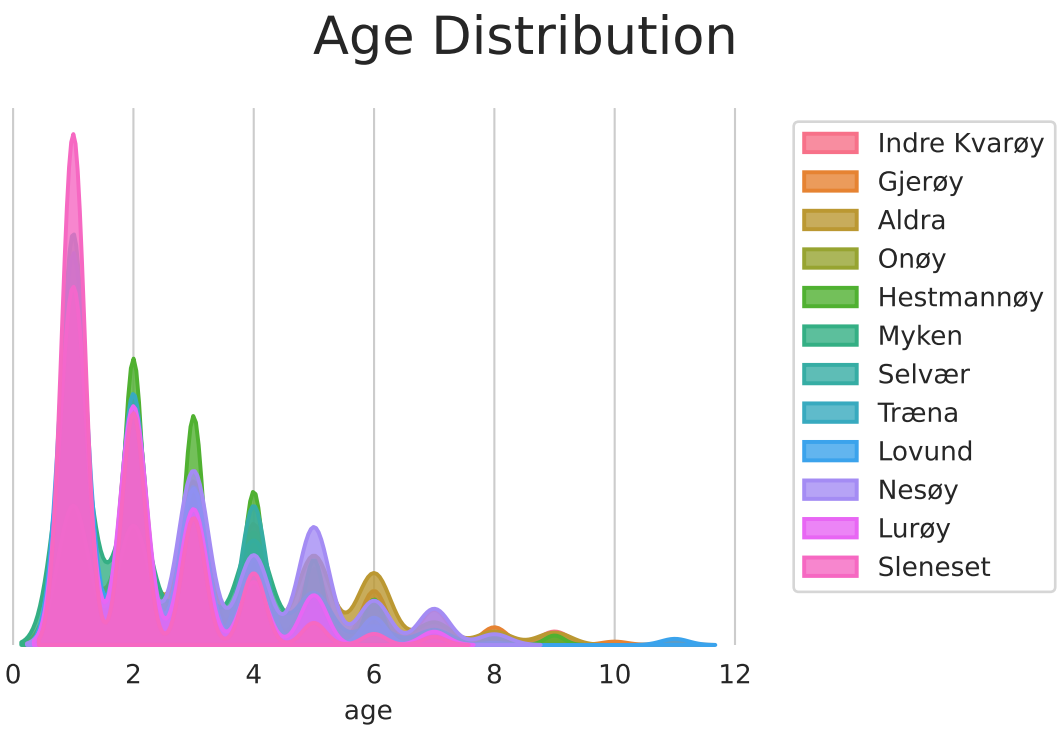


Figure 4.2: Densities of the individuals' age across 12 of the islands from the 70K dataset.

Figure 4.3: Densities of the individuals' body mass across 12 of the islands for males and females from the 70K dataset.

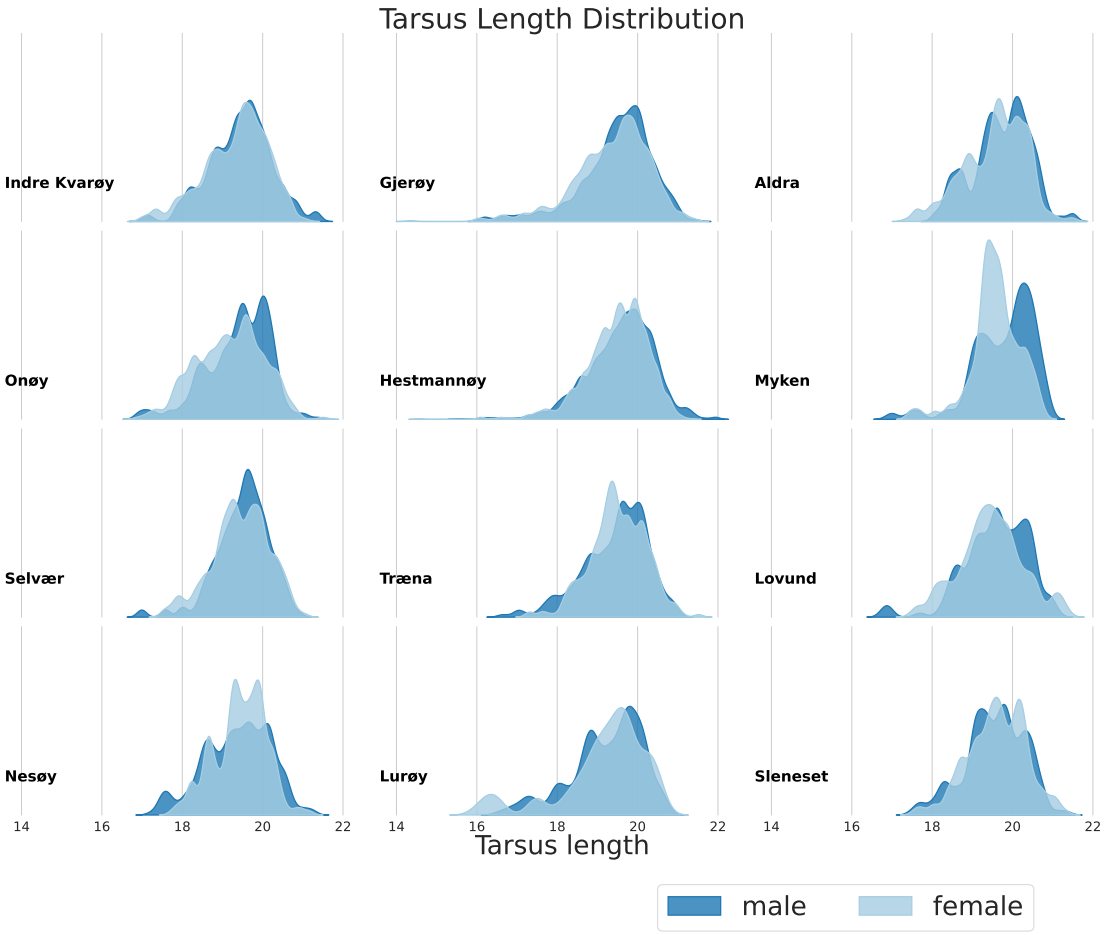


Figure 4.4: Densities of the individuals' tarsus length across 12 of the islands for males and females from the 70K dataset.

Table 4.2: Variation in body mass and tarsus length for 180K dataset and 70K dataset in both the average phenotype for each individual and the adjusted phenotype for each individual.

	Body mass variation	**Tarsus length variation**
180K mean phenotype value, $\bar{\mathbf{y}}$	4.649	0.642
70K mean phenotype value, $\bar{\mathbf{y}}$	4.336	0.648
180K adjusted phenotype, $\mathbf{y}^*$	1.435	0.592
70K adjusted phenotype, $\mathbf{y}^*$	1.308	0.585

Table 4.3: Correlations between the adjusted phenotype, $\mathbf{y}^*$, and the mean phenotype, $\bar{\mathbf{y}}$, in body mass and tarsus length for 180K dataset and 70K

$\text{Corr}(\bar{\mathbf{y}}, \mathbf{y}^*)$	**Body mass**	**Tarsus length**
180K dataset	0.951	0.988
70K dataset	0.816	0.989

4.1.1 Model Performance

The 10-fold cross-validation for the 180K dataset shows no clear differences between the performance of the animal model (GEBV) and the boosting models (Figure 4.5). However, when comparing the median values of the model performance there is an indication that especially XGBoost and CatBoost, but also LightGBM, can outperform the animal model when body mass is the response variable (Figure 4.5, left). The gradient boosting model with piecewise linear regression trees as a base learner, fitted with the LightGBM package (lgbmLinearTree), and the gradient boosting model with the elastic net as a base learner (xgboostLinear), fitted with the XGBoost package, and the GBM perform all worse than the animal model for body mass when comparing the medians. For the tarsus length, the animal model is the best-performing model in terms of the median, but the xgboostLinear, XGBoost and CatBoost models give comparable performances (Figure 4.5, right). GBM, LightGBM and lgbmLinearTree are the models with the poorest performance for tarsus length. It is worth noting that many of the models have a large variability in their distributions of the model accuracies from the 10-fold cross-validation.

For the 70K dataset, the 10-fold cross-validation again shows no clear-cut difference between the model performances (Figure 4.6). Median-wise, XGBoost and xgboostLinear both yield the highest performance for both body mass and tarsus length, although xgboostLinear seems to have a slight advantage over XGBoost for tarsus length. lgbmLinearTree performs median-wise worse than xgboostLinear and XGBoost for both traits but has more consistent accuracies for the body mass trait. CatBoost performs by far the worst of all models for both traits. No animal model was trained for the 70K dataset, but if we compare them to the animal model from the 180K dataset, we observe that the 70K tree-based models lie closer to the animal model than their 180K tree-based counterparts. This difference indicates that the tree-based models are not performing as optimally as they could have.

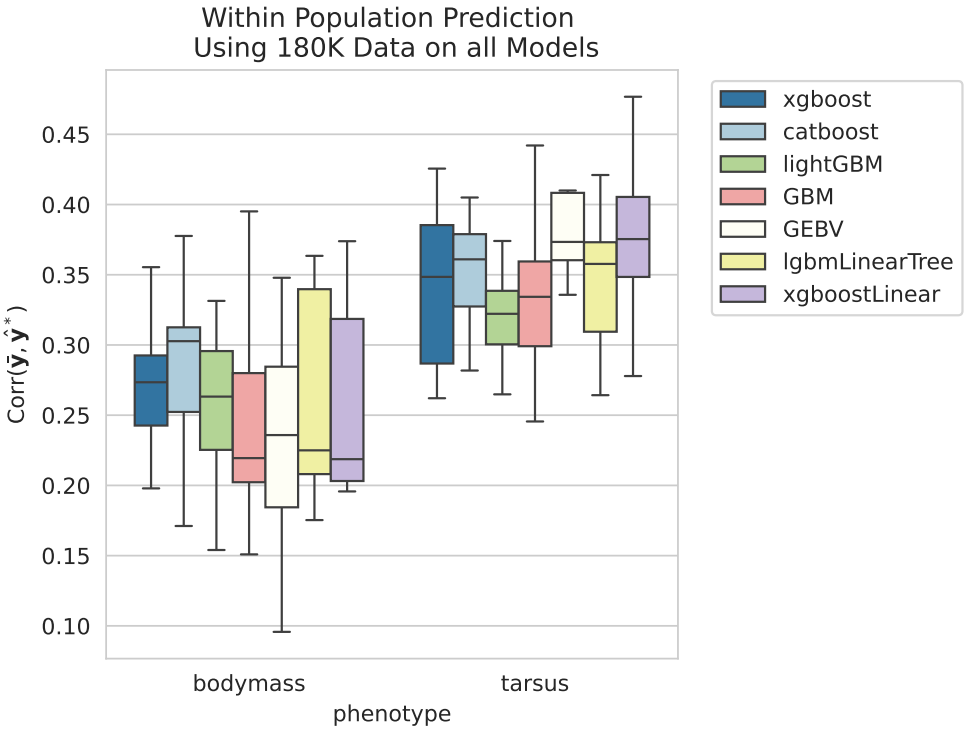


Figure 4.5: Boxplots of the correlation between the predicted genetic contribution for each individual and the individual's mean phenotype for the 180K dataset. Results of XGBoost, CatBoost, LightGBM, GBM are from Sand (2023) and are all tree-based methods. The GEBVs are from a Bayesian animal model and are also from Sand (2023). "xgboostLinear" refers to a gradient boosting model with a linear model as the base learner, fitted with the XGBoost package. "lgbmLinearTree" refers to a piece-wise linear regression tree fitted with the LightGBM package. Each box shows the distribution of a model's performance obtained from a 10-fold cross-validation loop.

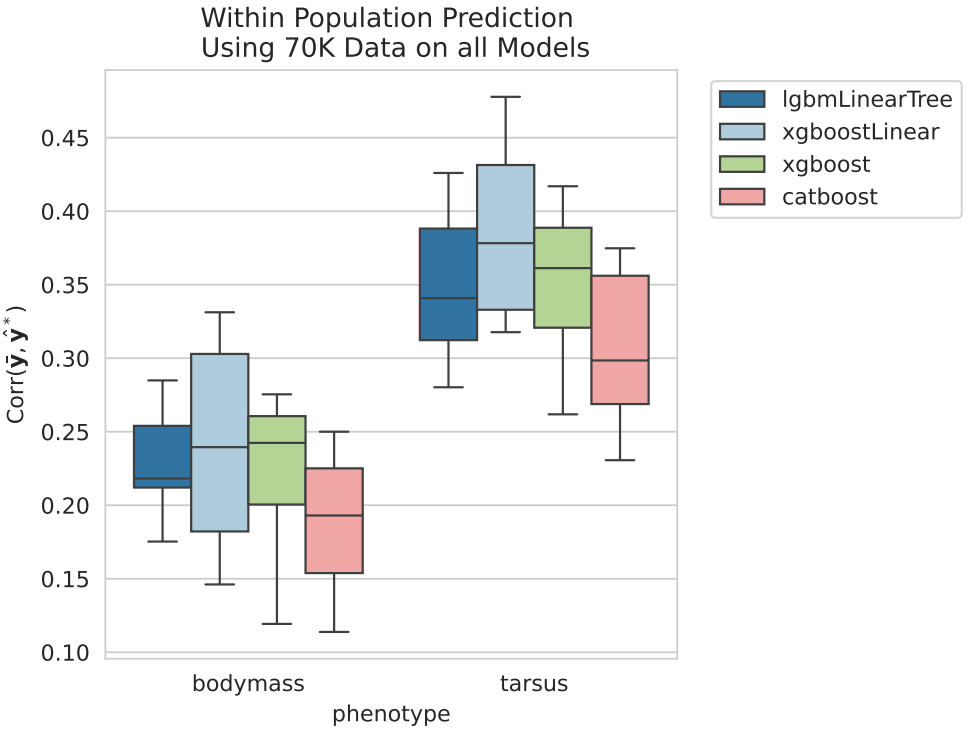


Figure 4.6: Boxplots of the correlation between the predicted genetic contribution for each individual and the individual's mean phenotype for the 70K dataset. "xgboost" and "catboost" refer to tree-based models fitted with the XGBoost package and the CatBoost package, respectively. "xgboostLinear" refers to a gradient boosting model with a linear model as base learner, fitted with the XGBoost package. "lgbmLinearTree" refers to a piece-wise linear regression tree fitted with the LightGBM package. Each box shows the distribution of a model's performance obtained from a 10-fold cross-validation loop.

4.1.2 GWAS and SHAP

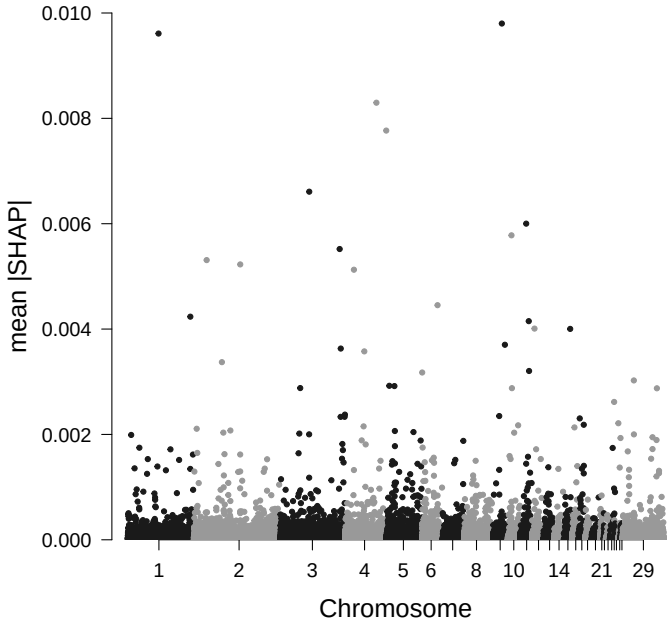
The mean absolute SHAP values obtained from the tree-based XGBoost model trained on the 70K dataset show that, for both traits, SNPs from all chromosomes affect the phenotypic traits (Figure 4.7). However, it is clear that the SNPs with the highest impact on tarsus, are not the same SNPs that affects body mass the most. For instance, at chromosome 13 for the tarsus model there is a "string" of SNPs close to each other that have an increasing SHAP value, which could indicate that an important allele for tarsus is close to these base pairs (Figure 4.7b). Similar patterns can be observed for the body mass model at both chromosome 11, chromosome 9 and chromosome 3 (Figure 4.7a). Among the SHAP values for the tarsus length model and the ones for body mass, some single SNPs seem relatively isolated, with much higher SHAP values than the rest. The SHAP values for tarsus length also distinguish themselves from the ones for body mass, by having generally more SNPs in the SHAP value range of $(0.002 - 0.004)$ (Figure 4.7b). Similarly, we observe that the number of SNPs with higher SHAP values is larger for the tarsus length model than for the body mass model (Table 4.4). However, the highest SHAP values for the body mass model are larger than the highest SHAP values for the tarsus length model.

Results from GWAS with single marker models fitted with GEMMA on the 70K dataset find no SNPs for the body mass trait and one significant SNP (at chromosome 19) for the tarsus length trait that has a p-value below the Bonferoni threshold (Figure 4.8). The tarsus GWAS also finds one SNP at chromosome 9 that is very close to the significance threshold.

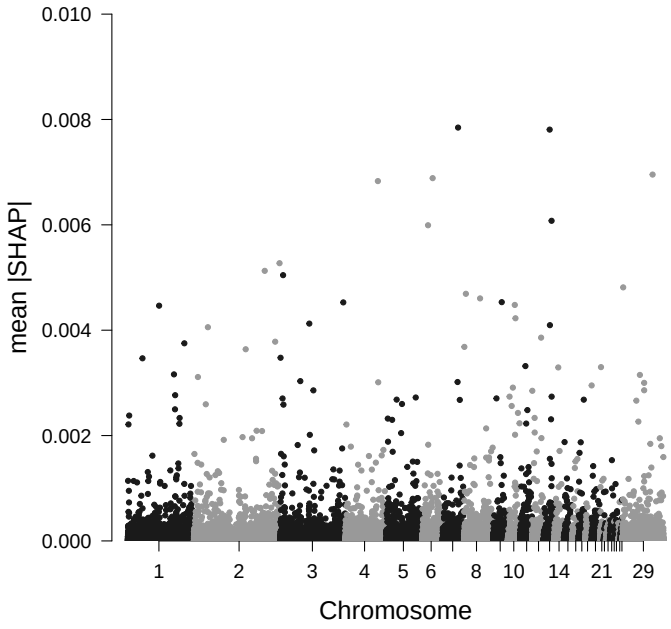
As expected, there is only a very weak linear relationship between the negative log p-values from GEMMA and the mean absolute SHAP values for both tarsus length and body mass (Figure 4.9b and Figure 4.10b). The weak relationship persists when removing SNPs that receive near zero SHAP values (Figure 4.9a and Figure 4.10a). Even though the relationship between the SHAP values and the negative log p-values has a low correlation coefficient for body mass, a group of four SNPs has high absolute SHAP values and relatively low p-values.

Table 4.4: Number of SHAP values that meet different thresholds, displayed for both traits.

	Body mass	**Tarsus length**
Total Number of SNPs	65 245	65 245
Number of non-zero SHAP SNPs	47 226	47 640
Number of SNPs with SHAP greater than 10^{-4}	5 401	7 980
Number of SNPs with SHAP greater than 10^{-3}	153	237

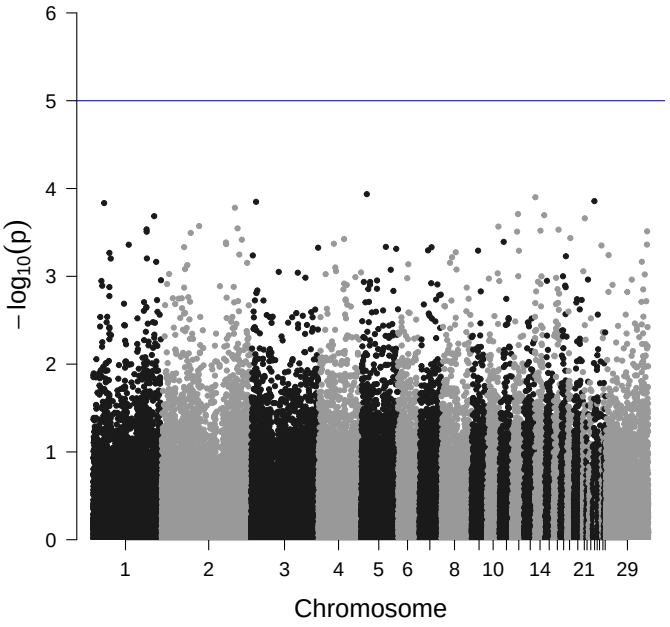


(a) Body mass

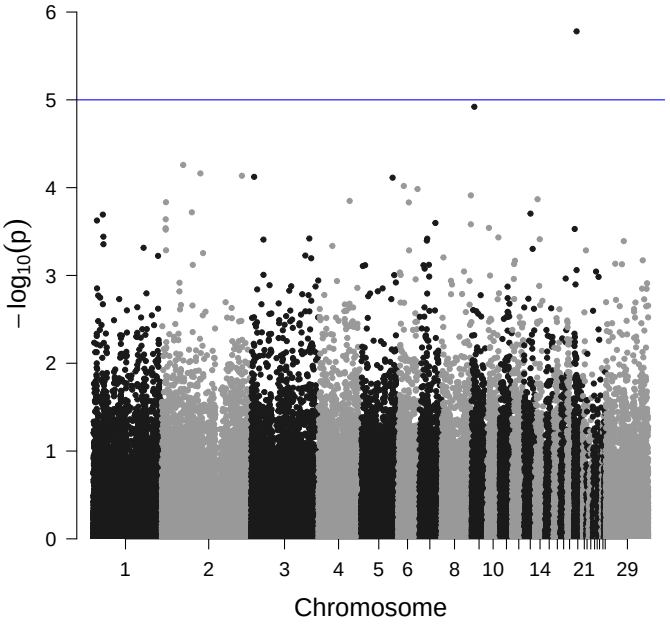


(b) Tarsus

Figure 4.7: Mean absolute SHAP values for tree-based XGBoost model fitted on 70K dataset with (a) body mass and (b) tarsus length as the response. Each dot is a SNP marker, with the x-axis being the position of each SNP along the different chromosomes.

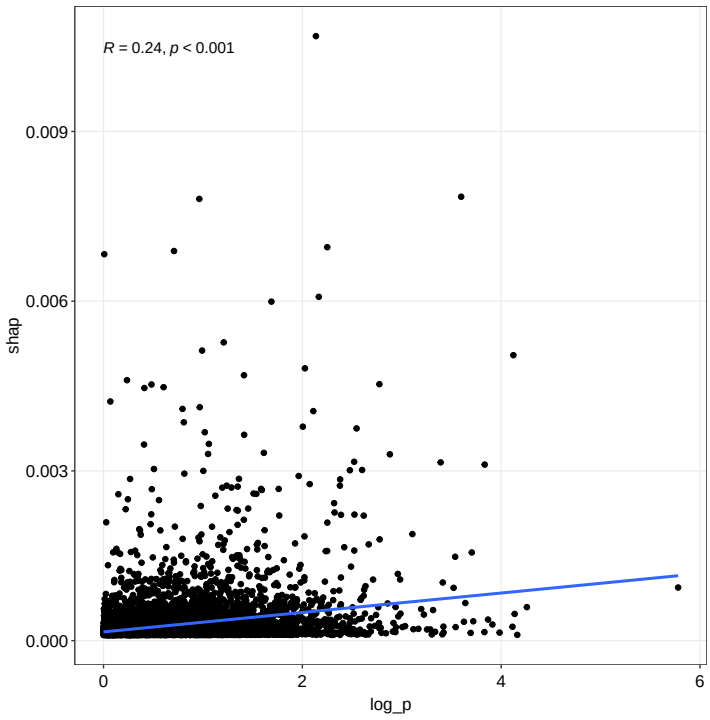


(a) Body mass

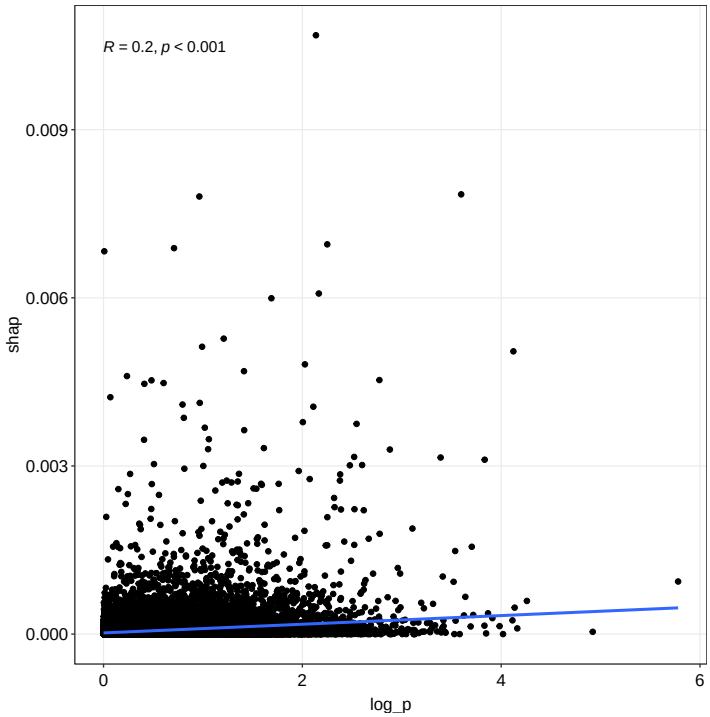


(b) Tarsus

Figure 4.8: GWAS results from GEMMA model fitted on 70K dataset using (a) body mass and (b) tarsus length as the response. The y-axis shows $-\log_{10}(p\text{-value})$ of each SNP. The blue lines show the Bonferoni threshold, for body mass all SNPs are below the Bonferoni threshold. Each dot is a SNP marker, with the x-axis being the position of each SNP along the different chromosomes.

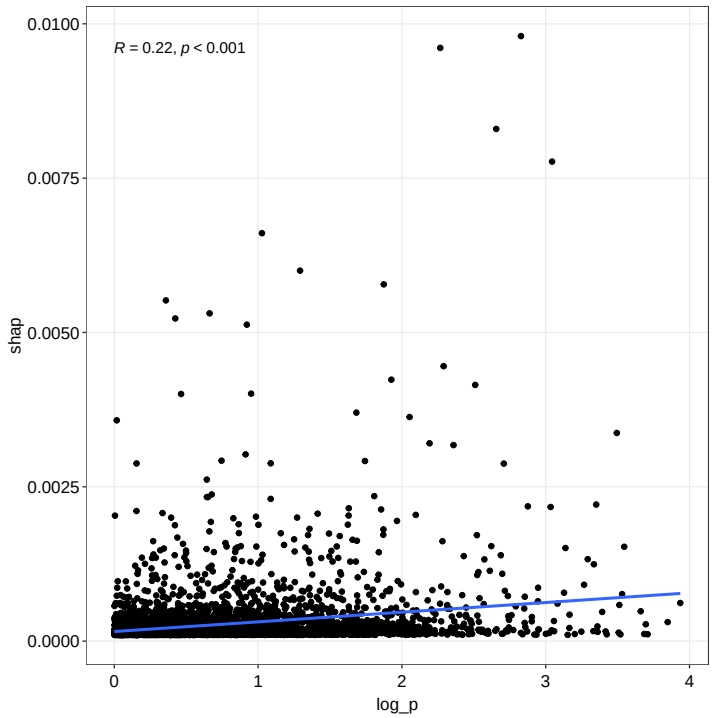


(a) SNPs with SHAP higher than $1 \cdot 10^{-4}$

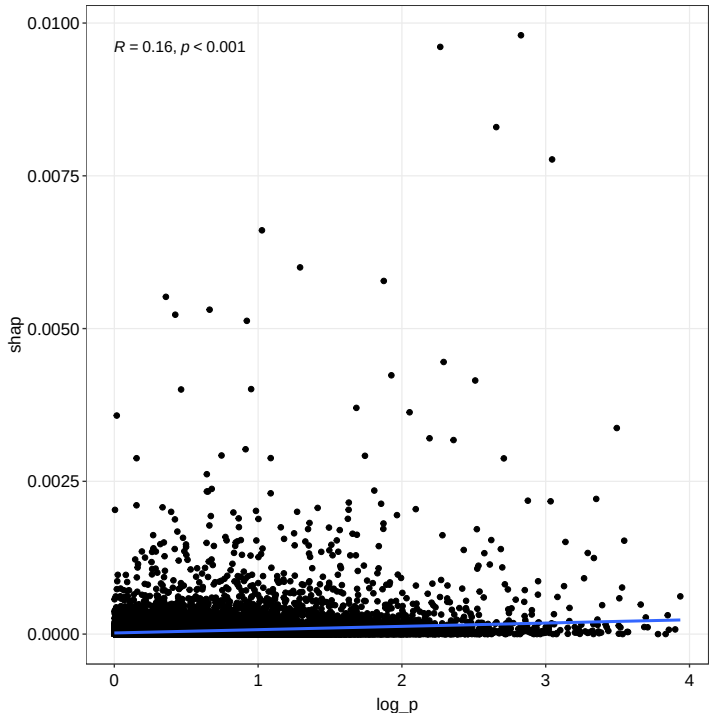


(b) All SNPs

Figure 4.9: Scatterplots showing the relationship between GWAS results ($-\log_{10}$ of p-value) on the x-axis and mean absolute SHAP results on the y-axis. Results are for tarsus length. R denotes the Pearson correlation coefficient, and the blue line is the least squares regression line. (a) Only SNPs with SHAP value higher than $1 \cdot 10^{-4}$ are present. (b) All SNPs, regardless of their SHAP value, are present.



(a) SNPs with SHAP higher than $1 \cdot 10^{-4}$



(b) All SNPs

Figure 4.10: Scatterplots showing the relationship between GWAS results ($-\log_{10}$ of p-value) on the x-axis and mean absolute SHAP results on the y-axis. Results are for body mass. R denotes the Pearson correlation coefficient, and the blue line is the least squares regression line. (a) Only SNPs with SHAP value higher than $1 \cdot 10^{-4}$ are present. (b) All SNPs, regardless of their SHAP value, are present.

4.2 Across-Population Predictions

Results from the across-population models fitted in the island-for-island procedure on the 70K data, show that the choice of base learner affects the model performance for both body mass and tarsus length (Figure 4.11). The gradient boosting model with the elastic net as base learner fitted with the XGBoost package (xgboostLinear) is by far the best-performing model for the tarsus trait, and median-wise the best choice for body mass as well. The tree-based XGBoost model and the piecewise linear regression tree-based LightGBM model (lgbmLinearTree) yield similar results, but the lgbmLinearTree model gives slightly more consistent results. Common for all three models is that they yield worse accuracies in the across-population prediction setting than their within-population prediction counterparts (Figure 4.6). The tree-based XGBoost for body mass also has an extreme value where the accuracy is negative, showcasing its more random behavior compared to its within-population prediction counterpart.

Where the choice of base learner affected the model performance median-wise, the choice of loss function for the lgbmLinearTree model does not (Figure 4.12). However, the volatility of the model performance increases greatly when opting for the 25% or the 75% quantile loss function reducing the quality of models fitted with these losses. The MAE and the MSE loss give similar model accuracies, but the MSE loss is more consistent by having less variation around the median.

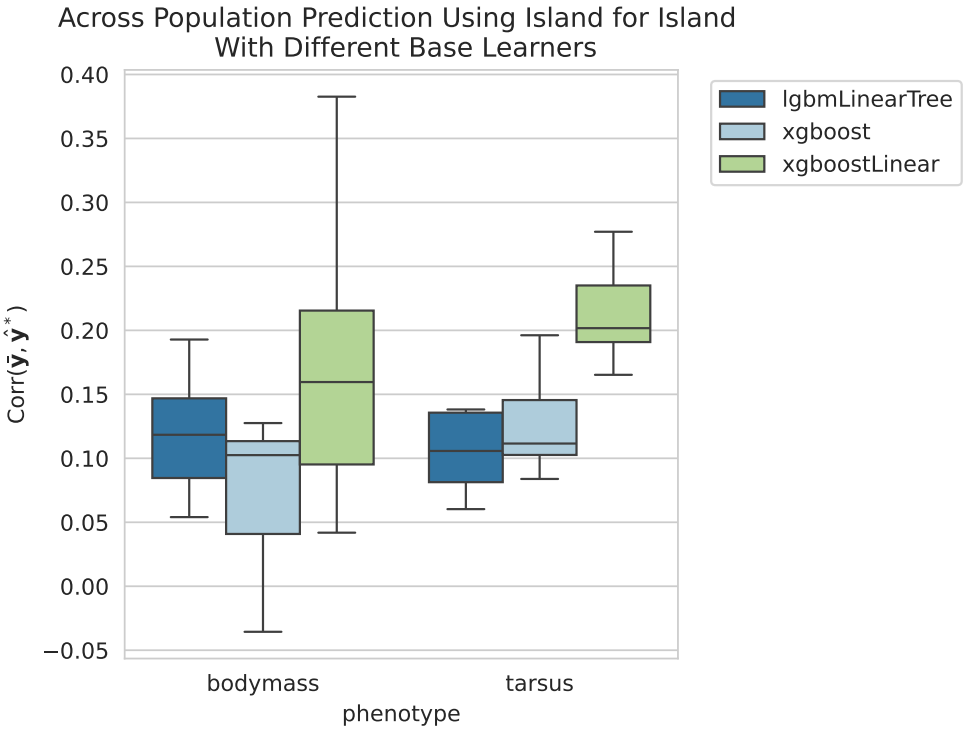


Figure 4.11: Box plots of the correlation between the predicted genetic contribution for each individual and the individual's mean phenotype for the 70K dataset using different boosting models. "lgbmLinearTree" refers to a piece-wise linear regression tree fitted with the LightGBM package. "xgboostLinear" refers to a gradient boosting model with a linear model as base learners, fitted with the XGBoost package. "xgboost" is a tree-based gradient boosting model fitted with the XGBoost package. Each box shows the distribution of a model's performance obtained from the island-for-island procedure described in Section 3.3. Each data point is the model's performance when tested on a specific island after being trained with data from all other islands.

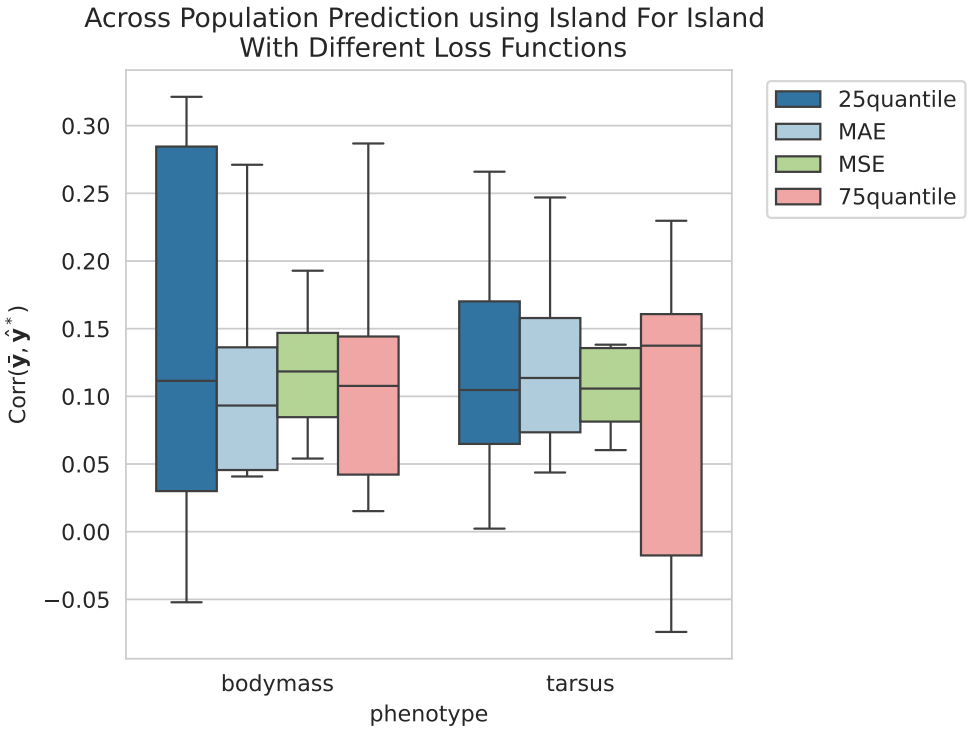


Figure 4.12: Box plots of the correlation between the predicted genetic contribution for each individual and the individual's mean phenotype for the 70K dataset using different loss functions. The boosting model is fitted with the LightGBM package with piecewise linear regression trees as base learners. Each box shows the distribution of a model's performance obtained from the island-for-island procedure described in Section 3.3. Each data point is the model's performance when tested on a specific island after being trained with data from all other islands.

5 Discussion

In this thesis, we have continued the work of Sand (2023) and looked at how gradient-boosting algorithms can be used for genomic prediction in a wild population of house sparrows. We performed genomic prediction for two quantitative traits, body mass and tarsus length. We have investigated the most popular versions of gradient-boosting models' performance in a within-population prediction setting and compared them to a Bayesian animal model. The models were fitted on two datasets, the 180K one with fewer individuals but more sequenced SNPs and the 70K one with more individuals but fewer sequenced SNPs. For the within-population predictions, we used four different tree-based gradient-boosting models: XGBoost, CatBoost, LightGBM, and GBM, but we also included a linear boosting model and a boosting model with piecewise linear regression trees as base learners. We also investigated how the choice of base-learner and loss function in a gradient-boosting model affect the model performance in an across-population prediction setting. The different base learners were binary regression trees, the elastic net, and piecewise linear regression trees. We tested four different loss functions for the piecewise linear regression tree: mean squared error (MSE), mean absolute error (MAE), the 25% quantile loss, and the 75% quantile loss. Lastly, we used SHAP values for regression trees to assess the SNP importance in a tree-based XGBoost model fitted on the 70K dataset in a within-population prediction setting. The SHAP values were compared to the results of GEMMA, a widely used single marker model in GWAS.

The results for the within-population predictions for the 180K dataset show that gradient-boosting models can give similar performance as the more traditional animal model. When comparing the medians of the model performances for body mass, we found that the tree-based models (except GBM) have a slight advantage over both the animal model and the linear-based gradient-boosting ones. The tree-based models of XGBoost and CatBoost performed well for body mass, where CatBoost had the highest median, and XGBoost had a high median and less variation around than the rest of the models. Adding a gradient-boosting model with the elastic net as a base learner did not give a higher performance for body mass. Still, it is worth noting that the linear booster's entire distribution lies within the 25% quantile of the Bayesian animal model's model performance distribution. For the tarsus length models, we found that the Bayesian animal model and the linear boosting model performed best regarding medians. Still, the Bayesian animal model had less variation around the median, making it a preferable model choice for tarsus length.

As to why the tree-based models seem to be a better choice for the body mass trait while the linear models are a better choice for the tarsus length trait, we hypothesize that it has to do with the genetic complexities of the two traits. In our setting, a regression tree's primary modeling capability is to capture non-linear interactions between features, meaning that regression trees model non-linear epistatic effects from the SNPs. Meanwhile, a linear model tries to capture additive effects from the SNPs. Further, if model A performs better on a dataset than model B, it is natural to think that model A captures the underlying data structure better than model B. By a similar argument, it seems reasonable to believe that body mass is a trait more reliant on epistatic effects than tarsus length.

There was a slight change in the models' behavior when moving to the 70K dataset for the within-population predictions. The most noticeable change was the drop in CatBoost's performance relative to the other boosting models. When using the 180K dataset, CatBoost was, median-wise, one of the top-performing models for both body mass and tarsus length. Whereas with the 70K dataset, it became the worst-performing model for both traits. Similarly, we observed that XGBoost's performance for body mass drops in performance for the 70K dataset compared to the 180K dataset. The tree-based models trained on the 180K dataset are from Sand (2023), which used Bayesian hyperparameter tuning, whereas this thesis used randomized hyperparameter search, which makes the results not entirely comparable. It might be that the Bayesian optimization worked better in finding the best combination of hyperparameters for the body mass XGBoost model and both the body mass and tarsus length CatBoost model. Another explanation is that the 70K dataset contains more individuals and includes more islands, which could make the 70K dataset more heterogeneous compared to the 180K dataset, which again makes the variation harder to capture. However, this explanation is contradicted by Table 4.2, where we found that the body mass variation in the 180K dataset is greater than in the 70K dataset. A third explanation is that the decrease in the genetic density when moving from the 180K data to the 70K data leads to fewer SNPs associated with epistatic effects, yielding poorer performance for the tree-based models. This third explanation is plausible as the drop in quality only happened for the tree-based XGBoost and CatBoost models, which rely on epistatic effects to predict the genetic component of the phenotype. However, a last most plausible explanation is the loss of information experienced when creating the adjusted phenotype for body mass in the 70K dataset (Table 4.3). This loss in information might be because the inbreeding coefficient FGRM was not included in adjusting the phenotype for the 70K dataset. Niskanen et al. (2020) found that inbreeding weakly influences the body mass in the Helgeland system, which might explain the loss of information we are experiencing, but more investigation is needed. The differences in the preprocessing step, therefore, make the two datasets less comparable.

In light of having less information on the body mass trait available in the adjusted phenotype, the results of the linear booster (xgboostLinear) and the piecewise linear regression tree booster (lgbmLinearTree) for the 70K dataset are more impressive as they give comparable performances to their 180K dataset counterparts but with less information available. The maximum correlation a model can achieve is the correlation between the adjusted phenotype and the mean phenotype, shown in Table 4.3. If we now divide the median correlation by the maximum correlation xgboostLinear can achieve for body mass for both the 70K and the 180K dataset, we obtain a median correlation of about $0.25/0.81 \approx 0.308$ for the 70K dataset and $0.22/0.95 \approx 0.23$ for the 180K dataset. Showing that xgboostLinear was doing a much better job with the 70K dataset than the 180K when accounting for the loss of information. This again raises the question of why the tree-based models performed comparably to xgboostLinear and lgbmLinearTree for body mass with the 70K dataset when they outperformed them (in terms of the median) with the 180K dataset. Here, the explanation regarding the decrease of SNPs associated with epistatic effects when decreasing the genetic density is more suitable. The 70K dataset has SNPs chosen such that they are more independent regarding linkage disequilibrium than in the 180K dataset, making it likely that the additive effects are preserved. In contrast, the epistatic effects might get watered out with less genetic density.

Overall, regarding the model performances in the within-population predictions, we want to highlight that the tree-based models are not good at capturing linear effects and relationships. In terms of genetic effects, regression trees, therefore, mainly capture effects due to interactions between alleles. Further, as the gradient-boosted regression trees in

this thesis give comparable performances with the models based on linear models, it challenges the viewpoint that additive effects are the most crucial effect to include in a model. We therefore think that a combination model that consists of both epistatic effects and additive effects has the potential to explain even more of the genetic contributions and give more insight into the genetic architecture. Therefore, it is a surprise that the gradient boosting model with a piecewise regression tree as base learner (lgbmLinearTree) does not distinguish itself more positively from the rest. A piecewise linear regression tree combines the capabilities to model non-linear interactions from regression with the additive capabilities of a linear model, meaning it combines additive and epistatic effects into one model. The linear regression model fitted in a leaf node only uses the features used to split the nodes on the path to said leaf node. Therefore, the piecewise linear regression tree implicitly assumes that the SNPs associated with epistatic effects are also associated with the additive effects. This assumption is perhaps too optimistic and might explain why the lgbmLinearTree model does not outmatch the other models.

The models used in the across-population predictions perform as expected worse than their within-population prediction counterparts. The decrease in accuracy is expected because the individuals on the different islands experience different environmental conditions, which can change the distribution of the genetic effects from one island to another. It is, therefore, of little use to discuss the reasons for the differences between the accuracies in the within-population predictions and the across-population predictions. What is worth discussing is how the choice of base learner affected the model performance. Overall, the elastic net is the preferable base learner (xgboostLinear) when looking at both tarsus length and body mass. That a linear model dominates in a across-population prediction is unsurprising as the models must extrapolate beyond their training range. The values a regression tree can predict are unalterable after training is finished. In other words, regression trees cannot extrapolate outside its training values. We also think it is more likely that the changes in SNPs associated with epistatic effects are greater than in those associated with the additive effects when changing the population. Such an explanation might explain why the piecewise linear tree (lgbmLinearTree) is not as effective base learner as the linear model, as it relies on epistatic effects to choose the best linear model.

The choice of loss function affects the variability in the lgbmLinearTree's model performance, where the mean squared error (MSE) gave the most consistent results. This was not expected as we believed that the MSE would pay too much attention to the extreme cases of each island, such that the model would neglect the similar contributions for all islands. Based on the results, letting the model adapt to more extreme cases might be beneficial. The mean absolute error (MAE) is only a little more volatile than the MSE, and its upper tail of the model performance distribution lies much higher than the upper tail of the MSE. Therefore, one cannot conclude which of the MSE and the MAE is the best choice for a loss function. The two quantile loss functions 25quantile and 75quantile, are clearly both worse choices of loss functions than the MSE and the MAE. However, notice that for body mass in Figure 4.12 the 75quantile's distribution is very similar to the MAE's distribution. The 75quantile loss estimates the 75% quantile of the response and therefore applies more attention to cases where the model underpredicts (*i.e.*, where $\hat{y} < y$), which could indicate that for body mass there is a tendency of underestimation with the MAE. Similarly, we observe the opposite for tarsus length, where the 25quantile's distribution is almost identical to the one of MAE, which could indicate that there is more overestimation happening in the tarsus length case. In the end, the safest and best choice of loss function are the well-known and widely used mean squared error and mean absolute error.

Another part of this thesis has been exploring the use of SHAP values to gain feature importance of the SNPs from a tree-based model. We found that the tree-based XGBoost model from the within-population predictions uses SNPs from every chromosome. Further work is needed to determine if this implies that the epistatic effects are happening between SNPs at different chromosomes or if they are mainly happening within chromosomes. There was little correspondence between the SNPs with high absolute SHAP values and those with low p-values from a univariate GWAS using GEMMA. The low correspondence is not a bad sign, as the SHAP values explain a tree-based model emphasizing epistatic effect. Therefore, the low correlation is more a sign that SNPs associated with epistatic effects are not the same as those associated with additive effects. This interpretation also supports the explanation of why the piecewise regression tree (lgbmLinearTree) did not perform as well as expected. However, for the body mass model (Figure 4.10) we observed that the top four SHAP value SNPs also had a high p-value which could indicate these four SNPs are both associated with epistatic effects and additive effects. In the end, we believe that the SHAP values can reveal more of the dynamics between the genetic effects and the phenotype. The SHAP values could potentially be used as an alternative to the breeding value if one wishes to select on epistatic effects instead on additive ones.

It is important to stress that the interpretation of a SHAP value of a feature is the deviation from the mean prediction when said feature is included, averaged over all possible coalitions of features. In Figure 4.7, we displayed each SNP's mean absolute SHAP value, meaning that we have aggregated away the direction in which the SNP value contributes. Therefore, the mean absolute SHAP value is only a measure of feature importance and does not say anything if the SNP contributes negatively or positively to the phenotype. The absolute SHAP value also does not indicate which SNPs it interacts with. Further work on the pure SHAP values is needed to understand more about the mechanisms of each SNP and their dependencies on other SHAP. Another type of SHAP value, denoted the SHAP interaction value (Lundberg, Erion et al., 2020), can be calculated to investigate which values two SNPs dependent on each other must have to obtain maximum interaction effect. We also want to highlight that although TreeSHAP claims to account for dependence among features by exploiting the tree structure (Lundberg, Erion et al., 2020), there have been some cases where it provides less intuitive explanations when the dependence among features is high (Aas et al., 2021). Therefore, in further works, we would use the model-agnostic Kernel SHAP method proposed by Aas et al. (2021), which handles dependent features better. We do not think this has impacted our analysis on the 70K dataset, as the SNPs present were chosen to be as independent as possible. However, if the study had been done on the 180K dataset, which is more likely to have a higher correlation among the SNPs, we could have ended up with less factual explanations. Another potential method to gain certain factual explanations is using the explainable boosting machine (Lou et al., 2013) designed to be an interpretable version of gradient boosting.

We observed differences in the SHAP values for body mass and tarsus length. The SNPs with the highest SHAP values had higher SHAP values for body mass than tarsus length (Figure 4.7). On the other hand, there were more SNPs with medium to high SHAP value for tarsus length than body mass (Table 4.4). One can interpret this difference in the distribution of the SHAP values as tarsus being more dependent on many SNPs that each contribute with a small effect, while body mass is more dependent on a smaller set of SNPs that each contribute with a larger effect. This interpretation of tarsus explains why the linear and animal models work well for tarsus length, as those models are built around the idea that many alleles across the chromosomes contribute with a tiny effect each. The body mass interpretation contrasts with the hypothesis we presented about body mass being a more epistatic trait, and one would assume that if a trait is more

reliant on epistasis, more SNPs are involved. However, as we found that the SNPs with the highest SHAP values also had relatively low p-value in the GWAS, it might be that these SNPs are associated with both additive and epistatic effects, which leads to them gaining high importance compared to the other SNPs that are primarily associated with either additive or epistatic effects.

We have in this thesis focused on modeling epistatic and additive effects using different boosting models. However, we also think using gradient boosted regression trees to model dominance effects is possible. First, encoding the SNP values to be categorical instead of numerical opens up for dominance effects, as then the tree splits have to say which SNP value they split on explicitly. Second, to restrict the dominance effect within the loci, one could limit each tree in the ensemble to only use one SNP. This approach then leads to a model that solely focuses on dominance effects, and then one could use the SHAP values to gain insight into which SNPs are most associated with the dominance effect. In contrast to Sand (2023), we have only focused on genetic effects. Therefore, a natural extension of this thesis is to incorporate environmental effects to investigate the interaction between the genetic and environmental effects.

In this thesis, we found that gradient boosting models can be an alternative to the well-known animal model for genomic prediction in a wild population of house sparrows. This thesis focused on modeling epistatic and additive effects separately by employing gradient-boosting models with different base learners. We conclude that the choice of base learners depends on the complexity of the trait one wishes to model. We also find indications of tree-based models needing higher genetic density than linear-based models. We found that SHAP values from a tree-based model can be an alternative to univariate GWAS in identifying SNPs associated with epistatic effects on the phenotype. Further works based on this thesis would be to create models that can capture dominance effects. Once a dominance model is established, one should incorporate dominance, epistatic, and additive effects into one model. Different algorithms to calculate SHAP values should be tried out, and the SHAP values need more investigation to identify processes between the different SNPs and how they affect the phenotype.

Bibliography

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, p. 103502. DOI: 10.1016/j.artint.2021.103502.

Abdurrahman, Muhammad Hafizh, Budhi Irawan and Casi Setianingsih (2020). 'A Review of Light Gradient Boosting Machine Method for Hate Speech Classification on Twitter'. In: *2020 2nd International Conference on Electrical, Control and Instrumentation Engineering (ICECIE)*, pp. 1–6. DOI: 10.1109/ICECIE50279.2020.9309565.

Alsabti, Khaled, Sanjay Ranka and Vineet Singh (1998). 'CLOUDS: A Decision Tree Classifier for Large Datasets'. In: *Electrical Engineering and Computer Science - All Scholarship*.

Ashraf, Bilal et al. (2022). 'Genomic Prediction in the Wild: A Case Study in Soay Sheep'. In: *Molecular Ecology* 31.24, pp. 6541–6555. DOI: 10.1111/mec.16262.

Asselman, Amal, Mohamed Khaldi and Souhaib Aammou (2023). 'Enhancing the Prediction of Student Performance Based on the Machine Learning XGBoost Algorithm'. In: *Interactive Learning Environments* 31.6, pp. 3360–3379. DOI: 10.1080/10494820.2021.1928235.

Baalsrud, Helle Tessand et al. (2014). 'Effects of Population Characteristics and Structure on Estimates of Effective Population Size in a House Sparrow Metapopulation'. In: *Molecular Ecology* 23.11, pp. 2653–2668. DOI: 10.1111/mec.12770.

Bosse, Mirte et al. (2017). 'Recent Natural Selection Causes Adaptive Evolution of an Avian Polygenic Trait'. In: *Science* 358.6361, pp. 365–368. DOI: 10.1126/science.aal3298.

Breiman, Leo (1996). 'Bagging Predictors'. In: *Machine Learning* 24.2, pp. 123–140. DOI: 10.1007/BF00058655.

– (2001). 'Random Forests'. In: *Machine Learning* 45.1, pp. 5–32. DOI: 10.1023/A:1010933404324.

Breiman, Leo et al. (1984). *Classification and Regression Trees*. Belmont, CA: Wadsworth.

Brooks, R. L. (1941). 'On Colouring the Nodes of a Network'. In: *Mathematical Proceedings of the Cambridge Philosophical Society* 37.2, pp. 194–197. DOI: 10.1017/S030500410002168X.

Carlens, Harald (2023). 'State of Competitive Machine Learning in 2022'. In: *ML Contests Research*.

Casella, George and Roger L. Berger (2002a). 'Bayes Estimators'. In: *Statistical Inference*. 2nd ed. Pacific Grove, CA: Duxbury. Chap. 7, pp. 324–326.

– (2002b). 'Loss Function Optimality'. In: *Statistical Inference*. 2nd ed. Pacific Grove, CA: Duxbury. Chap. 7.3.4, pp. 348–354.

Chafai, Narjice et al. (2023). 'A Review of Machine Learning Models Applied to Genomic Prediction in Animal Breeding'. In: *Frontiers in Genetics* 14. DOI: 10.3389/fgene.2023.1150596.

Chang, Christopher C et al. (2015). 'Second-Generation PLINK: Rising to the Challenge of Larger and Richer Datasets'. In: *GigaScience* 4.1, s13742-015-0047–8. DOI: 10.1186/s13742-015-0047-8.

Chen, Tianqi and Carlos Guestrin (2016). 'XGBoost: A Scalable Tree Boosting System'. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge*

Discovery and Data Mining, pp. 785–794. DOI: 10.1145/2939672.2939785. arXiv: 1603.02754 [cs].

Conner, Jefferey K. and Daniel L. Hartl (2004a). 'Introduction'. In: *A Primer of Ecological Genetics*. 1st ed. Sunderland, Massachusetts: Sinauer Associates. Chap. 1, pp. 1–2.

– (2004b). 'Population Genetics I: Genetic Variation, Random and Nonrandom Mating'. In: *A Primer of Ecological Genetics*. 1st ed. Sunderland: Sinauer Associates, pp. 9–42.

– (2004c). 'Quantitative Genetics I'. In: *A Primer of Ecological Genetics*. 1st ed. Sunderland: Sinauer Associates, pp. 99–112.

Conner, Jeffrey K. (2003). 'Artificial Selection: A Powerful Tool for Ecologists'. In: *Ecology* 84.7, pp. 1650–1660. DOI: 10.1890/0012-9658(2003)084[1650:ASAPTF]2.0.CO;2.

Curry, Haskell B. (1944). 'The Method of Steepest Descent for Non-Linear Minimization Problems'. In: *Quarterly of Applied Mathematics* 2.258-261.

Frazier, Peter I. (2018). 'A Tutorial on Bayesian Optimization'. In.

Friedman, Jerome H. (2001). 'Greedy Function Approximation: A Gradient Boosting Machine.' In: *The Annals of Statistics* 29.5. DOI: 10.1214/aos/1013203451.

– (2002). 'Stochastic Gradient Boosting'. In: *Computational Statistics and Data Analysis* 38.4, pp. 367–378. DOI: 10.1016/S0167-9473(01)00065-2.

Friedman, Jerome H., Trevor Hastie and Rob Tibshirani (2010). 'Regularization Paths for Generalized Linear Models via Coordinate Descent'. In: *Journal of Statistical Software* 33, pp. 1–22. DOI: 10.18637/jss.v033.i01.

Gamerman, Danim and Hedibert F. Lopes (2006). 'Bayesian Inference'. In: *Markov Chain Monte Carlo: Stochastic Simulation for Bayesian Inference*. Chapman and Hall/CRC.

Geof H. Givens and Jennifer A. Hoeting (2013). 'Markov Chain Monte Carlo'. In: *Computational Statistics*. 2nd ed. John Wiley & Sons. Chap. 7.

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, p. 180. DOI: 10.1186/s12870-022-03559-z.

Google Cloud (2024). *State of Data Science and Machine Learning 2022*.

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

Henderson, Charles R. (1950). 'Estimation of Genetic Parameters'. In: *The Annals of Mathematical Statistics* 21.309-310.

– (1975). 'Best Linear Unbiased Estimation and Prediction under a Selection Model'. In: *Biometrics* 31.2, p. 423. DOI: 10.2307/2529430.

– (1984). *Applications of Linear Models in Animal Breeding*. University of Guelph.

Holand, Anna Marie et al. (2013). 'Animal Models and Integrated Nested Laplace Approximations'. In: *G3 Genes—Genomes—Genetics* 3.8, pp. 1241–1251. DOI: 10.1534/g3.113.006700.

Hunter, D. C. et al. (2022). 'Using Genomic Prediction to Detect Microevolutionary Change of a Quantitative Trait'. In: *Proceedings of the Royal Society B: Biological Sciences* 289.1974, p. 20220330. DOI: 10.1098/rspb.2022.0330.

Husby, Arild et al. (2015). 'Genome-Wide Association Mapping in a Wild Avian Population Identifies a Link between Genetic and Phenotypic Variation in a Life-History Trait'. In: *Proceedings of the Royal Society B: Biological Sciences* 282.1806, p. 20150156. DOI: 10.1098/rspb.2015.0156.

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. 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. DOI: 10.1111/j.1558-5646.2008.00395.x.

Ke, Guolin et al. (2017). 'LightGBM: A Highly Efficient Gradient Boosting Decision Tree'. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., pp. 3149–3157.

Koenker, Roger (2005). *Quantile Regression*. Econometric Society Monographs. Cambridge: Cambridge University Press. DOI: 10.1017/CBO9780511754098.

Kruuk, Loeske E. B. (2004). 'Estimating Genetic Parameters in Natural Populations Using the 'Animal Model''. In: *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences* 359.1446, pp. 873–890. DOI: 10.1098/rstb.2003.1437.

Kuchibhotla, Arun K., John E. Kolassa and Todd A. Kuffner (2022). 'Post-Selection Inference'. In: *Annual Review of Statistics and Its Application* 9.1, pp. 505–527. DOI: 10.1146/annurev-statistics-100421-044639.

LeCun, Yann, Yoshua Bengio and Geoffrey Hinton (2015). 'Deep Learning'. In: *Nature* 521.7553, pp. 436–444. DOI: 10.1038/nature14539.

Lefakis, Leonidas, Oleksandr Zadorozhnyi and Gilles Blanchard (2019). *Efficient Regularized Piecewise-Linear Regression Trees*. DOI: 10.48550/arXiv.1907.00275. arXiv: 1907.00275 [cs, stat].

Li, Bo et al. (2018). 'Genomic Prediction of Breeding Values Using a Subset of SNPs Identified by Three Machine Learning Methods'. In: *Frontiers in Genetics* 9. DOI: 10.3389/fgene.2018.00237.

Lipovetsky, Stan and Michael Conklin (2001). 'Analysis of Regression in Game Theory Approach'. In: *Applied Stochastic Models in Business and Industry* 17.4, pp. 319–330. DOI: 10.1002/asmb.446.

Lou, Yin et al. (2013). 'Accurate Intelligible Models with Pairwise Interactions'. In: *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Chicago Illinois USA: ACM, pp. 623–631. DOI: 10.1145/2487575.2487579.

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, p. 152. DOI: 10.1186/s12864-023-09933-x.

Lundberg, Scott M., Gabriel Erion et al. (2020). 'From Local Explanations to Global Understanding with Explainable AI for Trees'. In: *Nature Machine Intelligence* 2.1, pp. 56–67. DOI: 10.1038/s42256-019-0138-9.

Lundberg, Scott M. and Su-In Lee (2017). 'A Unified Approach to Interpreting Model Predictions'. In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc.

Lundregan, Sarah L. et al. (2018). 'Inferences of Genetic Architecture of Bill Morphology in House Sparrow Using a High-density SNP Array Point to a Polygenic Basis'. In: *Molecular Ecology* 27.17, pp. 3498–3514. DOI: 10.1111/mec.14811.

Lynch, Michael and Bruce Walsh (1998). *Genetics and Analysis of Quantitative Traits*. Vol. 1. Sinauer Sunderland, MA.

Manolio, Teri A et al. (2009). 'Finding the Missing Heritability of Complex Diseases.' In: *Nature* 461.7265, pp. 747–53. DOI: 10.1038/nature08494.

Martino, Sara and Andrea Riebler (2019). *Integrated Nested Laplace Approximations (INLA)*.

Mason, Llew et al. (1999). 'Boosting Algorithms as Gradient Descent'. In: *Advances in neural information processing systems* 12.

Mathew, Boby et al. (2022). 'NeuralLasso: Neural Networks Meet Lasso in Genomic Prediction'. In: *Frontiers in Plant Science* 13.

McGaugh, Suzanne E., Aaron J. Lorenz and Lex E. Flagel (2021). 'The Utility of Genomic Prediction Models in Evolutionary Genetics'. In: *Proceedings of the Royal Society B: Biological Sciences* 288.1956, p. 20210693. DOI: 10.1098/rspb.2021.0693.

Meuwissen, Theo, Ben Hayes and Mike Goddard (2001). 'Prediction of Total Genetic Value Using Genome-Wide Dense Marker Maps'. In: *Genetics* 157.4, pp. 1819–1829. DOI: 10.1093/genetics/157.4.1819.

– (2016). 'Genomic Selection: A Paradigm Shift in Animal Breeding'. In: *Animal Frontiers* 6.1, pp. 6–14. DOI: 10.2527/af.2016-0002.

Mittag, Florian, Michael Römer and Andreas Zell (2015). 'Influence of Feature Encoding and Choice of Classifier on Disease Risk Prediction in Genome-Wide Association Studies'. In: *PLOS ONE* 10.8, e0135832. DOI: 10.1371/journal.pone.0135832.

Montesinos-López, Osval Antonio et al. (2021). 'A Review of Deep Learning Applications for Genomic Selection'. In: *BMC Genomics* 22.1, p. 19. DOI: 10.1186/s12864-020-07319-x.

Muff, Stefanie et al. (2019). 'Animal Models with Group-Specific Additive Genetic Variances: Extending Genetic Group Models'. In: *Genetics Selection Evolution* 51.1, p. 7. DOI: 10.1186/s12711-019-0449-7.

Natekin, Alexey and Alois Knoll (2013). 'Gradient Boosting Machines, a Tutorial'. In: *Frontiers in Neurorobotics* 7. DOI: 10.3389/fnbot.2013.00021.

Nazzicari, Nelson and Filippo Biscarini (2022). 'Stacked Kinship CNN vs. GBLUP for Genomic Predictions of Additive and Complex Continuous Phenotypes'. In: *Scientific Reports* 12.1, p. 19889. DOI: 10.1038/s41598-022-24405-0.

Niskanen, Alina K. et al. (2020). 'Consistent Scaling of Inbreeding Depression in Space and Time in a House Sparrow Metapopulation'. In: *Proceedings of the National Academy of Sciences* 117.25, pp. 14584–14592. DOI: 10.1073/pnas.1909599117.

O'Hara, R. B. et al. (2008). 'Bayesian Approaches in Evolutionary Quantitative Genetics'. In: *Journal of Evolutionary Biology* 21.4, pp. 949–957. DOI: 10.1111/j.1420-9101.2008.01529.x.

Ogutu, Joseph O., Torben Schulz-Streeck and Hans-Peter Piepho (2012). 'Genomic Selection Using Regularized Linear Regression Models: Ridge Regression, Lasso, Elastic Net and Their Extensions'. In: *BMC Proceedings* 6.2, S10. DOI: 10.1186/1753-6561-6-S2-S10.

Ovaskainen, Otso, José Manuel Cano and Juha Merilä (2008). 'A Bayesian Framework for Comparative Quantitative Genetics'. In: *Proceedings of the Royal Society B: Biological Sciences* 275.1635, pp. 669–678. DOI: 10.1098/rspb.2007.0949.

Pedregosa, Fabian et al. (2012). 'Scikit-Learn: Machine Learning in Python'. In.

Prokhorenkova, Liudmila et al. (2017). 'CatBoost: Unbiased Boosting with Categorical Features'. In.

Purcell, Shaun M and Christopher C Chang (2023). *PLINK 1.9*.

Ringsby, Thor Harald et al. (2002). 'Asynchronous Spatiotemporal Demography of a House Sparrow Metapopulation in a Correlated Environment'. In: *Ecology* 83.2, pp. 561–569. DOI: 10.1890/0012-9658(2002)083[0561:ASDOAH]2.0.CO;2.

Rue, Håvard, Sara Martino and Nicolas Chopin (2009). 'Approximate Bayesian Inference for Latent Gaussian Models by Using Integrated Nested Laplace Approximations'. In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* 71.2, pp. 319–392. DOI: 10.1111/j.1467-9868.2008.00700.x.

Saberian, Mohammad J., Hamed Masnadi-Shirazi and Nuno Vasconcelos (2011). 'TaylorBoost: First and Second-Order Boosting Algorithms with Explicit Margin Control'. In: *CVPR 2011*, pp. 2929–2934. DOI: 10.1109/CVPR.2011.5995605.

Sand, Didrik Lindløv (2023). *Genotype to Phenotype Predictions Using Boosting Algorithms*. Tech. rep.

Sardanelli, Francesco et al. (2023). 'Artificial Intelligence (AI) in Biomedical Research: Discussion on Authors' Declaration of AI in Their Articles Title'. In: *European Radiology Experimental* 7.1, p. 2. DOI: 10.1186/s41747-022-00316-7.

Sarker, Iqbal H. (2021). 'Machine Learning: Algorithms, Real-World Applications and Research Directions'. In: *SN Computer Science* 2.3, p. 160. DOI: 10.1007/s42979-021-00592-x.

Shapley, L. S. (1953). 'A Value for N-Person Games'. In: *A Value for N-Person Games*. Princeton University Press, pp. 307–318. DOI: 10.1515/9781400881970-018.

Shi, Yu, Jian Li and Zhize Li (2019). *Gradient Boosting With Piece-Wise Linear Regression Trees*. arXiv: 1802.05640 [cs].

Sigrist, Fabio (2021). 'Gradient and Newton Boosting for Classification and Regression'. In: *Expert Systems with Applications* 167, p. 114080. DOI: 10.1016/j.eswa.2020.114080.

Silva, C. N. S. et al. (2017). 'Insights into the Genetic Architecture of Morphological Traits in Two Passerine Bird Species'. In: *Heredity* 119.3, pp. 197–205. DOI: 10.1038/hdy.2017.29.

Slatkin, Montgomery (2008). 'Linkage Disequilibrium — Understanding the Evolutionary Past and Mapping the Medical Future'. In: *Nature Reviews Genetics* 9.6, pp. 477–485. DOI: 10.1038/nrg2361.

Stanford Institute for Human-Centred Artifical Intelligence (2024). *Artificial Intelligence Index Report*. Tech. rep. Stanford Institute for Human-Centered Artificial Intelligenc.

Steinsland, Ingelin and Henrik Jensen (2010). 'Utilizing Gaussian Markov Random Field Properties of Bayesian Animal Models'. In: *Biometrics* 66.3, pp. 763–771. DOI: 10.1111/j.1541-0420.2009.01336.x.

The Wellcome Trust Case Control Consortium et al. (2007). 'Genome-Wide Association Study of 14,000 Cases of Seven Common Diseases and 3,000 Shared Controls'. In: *Nature* 447.7145, pp. 661–678. DOI: 10.1038/nature05911.

Tyralis, Hristos and Georgia Papacharalampous (2021). 'Boosting Algorithms in Energy Research: A Systematic Review'. In: *Neural Computing and Applications* 33.21, pp. 14101–14117. DOI: 10.1007/s00521-021-05995-8.

Uffelmann, Emil et al. (2021). 'Genome-Wide Association Studies'. In: *Nature Reviews Methods Primers* 1.1, pp. 1–21. DOI: 10.1038/s43586-021-00056-9.

United Nations (2015). *Transforming Our World: The 2030 Agenda for Sustainable Development*. Tech. rep. United Nations.

VanRaden, P.M. (2008). 'Efficient Methods to Compute Genomic Predictions'. In: *Journal of Dairy Science* 91.11, pp. 4414–4423. DOI: 10.3168/jds.2007-0980.

Wang, Xiaoqiao et al. (2019). 'Evaluation of GBLUP, BayesB and Elastic Net for Genomic Prediction in Chinese Simmental Beef Cattle'. In: *PLOS ONE* 14.2, e0210442. DOI: 10.1371/journal.pone.0210442.

Wilson, Alastair J. et al. (2010). 'An Ecologist's Guide to the Animal Model'. In: *Journal of Animal Ecology* 79.1, pp. 13–26. DOI: 10.1111/j.1365-2656.2009.01639.x.

Yengo, Loïc et al. (2022). 'A Saturated Map of Common Genetic Variants Associated with Human Height'. In: *Nature* 610.7933, pp. 704–712. DOI: 10.1038/s41586-022-05275-y.

Young, H. P. (1985). 'Monotonic Solutions of Cooperative Games'. In: *International Journal of Game Theory* 14.2, pp. 65–72. DOI: 10.1007/BF01769885.

Ypma, Tjalling J. (1995). 'Historical Development of the Newton–Raphson Method'. In: *SIAM Review* 37.4, pp. 531–551. DOI: 10.1137/1037125.

Yu, Lean, Zebin Yang and Ling Tang (2018). 'Quantile Estimators with Orthogonal Pinball Loss Function'. In: *Journal of Forecasting* 37.3, pp. 401–417. DOI: 10.1002/for.2510.

Zhou, Xiang and Matthew Stephens (2012). 'Genome-Wide Efficient Mixed-Model Analysis for Association Studies'. In: *Nature Genetics* 44.7, pp. 821–824. DOI: 10.1038/ng.2310.

Zou, Hui and Trevor Hastie (2005). 'Regularization and Variable Selection Via the Elastic Net'. In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* 67.2, pp. 301–320. DOI: 10.1111/j.1467-9868.2005.00503.x.

Appendix

A Hyperparameter Distributions

This section contains information about the hyperparameters tuned in this thesis and which distribution was used in the hyperparameter search.

Table 1: The tuned hyperparameters for XGBoost with binary regression tree as the base learner. The *name* column shows the name of the tuned hyperparameter, the *description* column contains a short description of each hyperparameter that is tuned, and the *Distribution* column displays what distribution/search range was used in the search.

Name	**Description**	**Distribution**
max_depth	Maximum depth of a tree	Random integers from $[0, 10]$
alpha	L1 regularization parameter	Random numbers from a loguniform distribution on $[10^{-5}, 10^{-1}]$
lambda	L2 regularization parameter	Random numbers from a loguniform distribution on $[10^{-5}, 10^{-1}]$
min_child_weight	Minimum sum of instance weight needed in a child	Random integers from $[1, 15]$
learing_rate	The learning rate	Random numbers from a loguniform distribution on $[10^{-3}, 10^{-1}]$
subsample	The fraction of training observations used for each tree	Random numbers from a uniform distribution on $[0.4, 0.9]$
n_estimators	The number of trees used	Random integers from $[100, 500]$
colsample_bytree	The fraction of features used for each tree	Random numbers from a uniform distribution on $[0.4, 0.8]$
colsample_bynode	The fraction of features used for each node to split	Random numbers from a uniform distribution on $[0.2, 0.9]$
gamma	The threshold in loss reduction (Equation 2.26) required to split a node	Random numbers from a loguniform distribution on $[10^{-3}, 10^{-1}]$

Table 2: The tuned hyperparameters for XGBoost with the elastic net as the base learner. The *name* column shows the name of the tuned hyperparameter, the *description* column contains a short description of each hyperparameter that is tuned, and the *Distribution* column displays what distribution/search range was used in the search.

Name	**Description**	**Distribution**
alpha	L1 regularization parameter	Random numbers from a loguniform distribution on $[10^{-5}, 10^{-1}]$
lambda	L2 regularization parameter	Random numbers from a loguniform distribution on $[10^{-5}, 10^{-1}]$
learing_rate	The learning rate	Random numbers from a loguniform distribution on $[10^{-3}, 10^{-1}]$
n_estimators	The number of base learners used	Random integers from $[20, 500]$

Table 3: The tuned hyperparameters for CatBoost with binary regression tree as the base learner. The *name* column shows the name of the tuned hyperparameter, the *description* column contains a short description of each hyperparameter that is tuned, and the *Distribution* column displays what distribution/search range was used in the search.

Name	**Description**	**Distribution**
depth	Maximum depth of a tree	Random integers from $[0, 10]$
l2_leaf_reg	L2 regularization parameter	Random numbers from a loguniform distribution on $[0.1, 40]$
leaf_estimation_iterations	Number of newton or gradient iterations to use for calculating the leaf weight	Random integers from $[0, 10]$
learing_rate	The learning rate	Random numbers from a loguniform distribution on $[10^{-3}, 10^{-1}]$
bagging_temperature	Controls the intensity of the bagging, that is, the subsampling of observations	Random numbers from a uniform distribution on $[0.1, 1]$
iterations	The number of trees used	Random integers from $[100, 800]$
random_strength	The amount of randomness to add when evaluating possible split points	Random integers from $[0, 20]$

Table 4: The tuned hyperparameters for LightGBM with piecewise linear regression tree as the base learner, using the 70K data in the within-prediction setting. The *name* column shows the name of the tuned Hyperparameter, the *description* column contains a short description of each hyperparameter that is tuned, and the *Distribution* column displays what distribution/search range was used in the search.

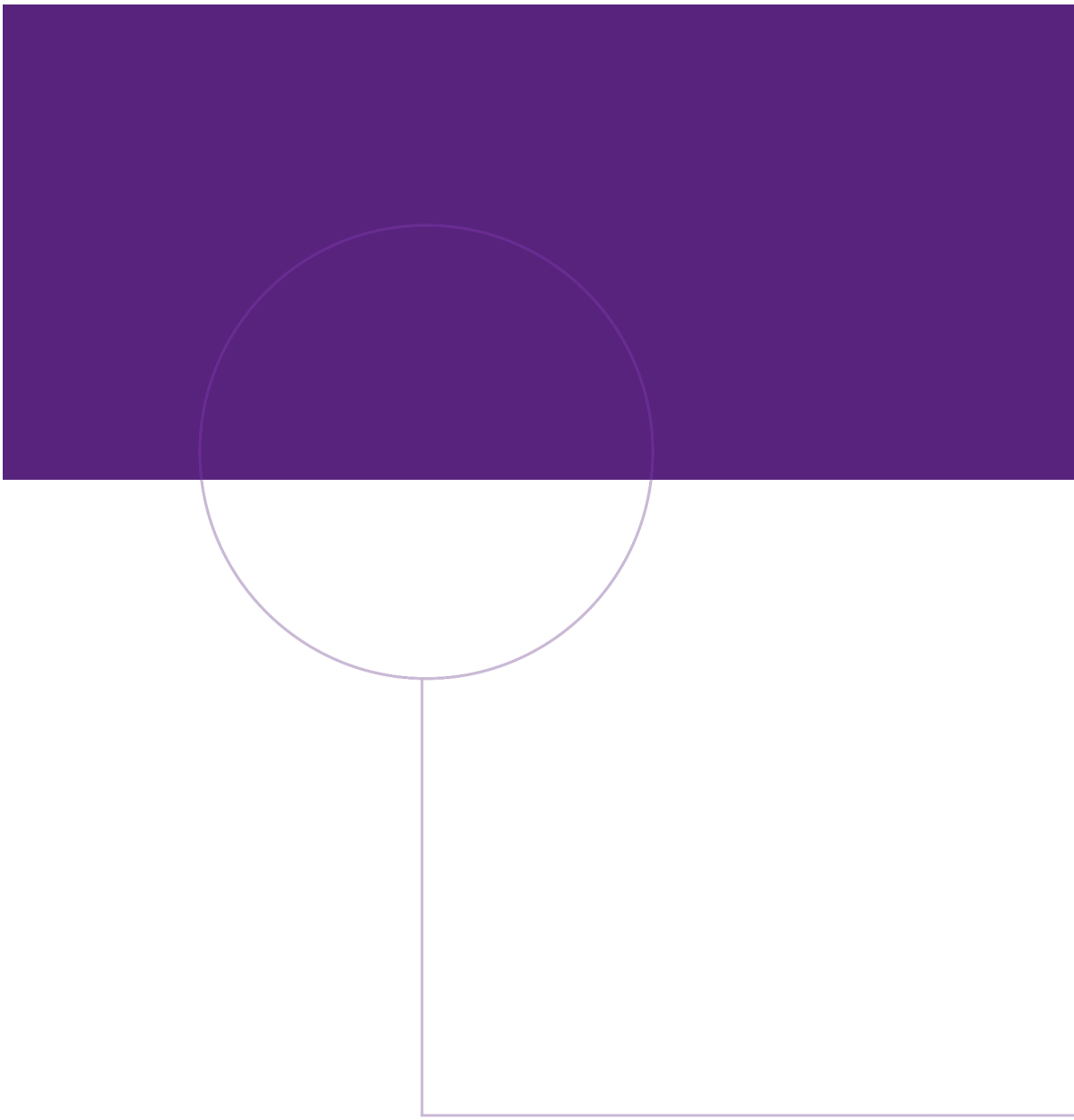
Name	**Description**	**Distribution**
max_depth	Maximum depth of a tree	Random integers from $[2, 25]$
reg_alpha	L1 regularization parameter	Random numbers from a loguniform distribution on $[0, 0.1]$
reg_lambda	L2 regularization parameter	Random numbers from a loguniform distribution on $[0, 0.1]$
min_child_weight	Minimum sum of instance weight needed in a child	Random numbers from a uniform distribution on $[0.02, 0.1]$
min_child_samples	Minimum number of samples needed in a child	Random integers from $[30, 700]$
learing_rate	The learning rate	Random numbers from a uniform distribution on $[0.01, 0.1]$
subsample	The fraction of training observations used for each tree	Random numbers from a uniform distribution on $[0.1, 0.9]$
n_estimators	The number of base learners used	Random integers from $[300, 600]$
colsample_bytree	The fraction of features used for each tree	Random numbers from a uniform distribution on $[0.2, 0.8]$
colsample_bynode	The fraction of features used for each node	Random numbers from a uniform distribution on $[0.2, 1]$
linear_lambda	L2 regularization on linear model	Random numbers from a loguniform distribution on $[10^{-3}, 10^{-1}]$

Table 5: Hyperparameter Distributions for LightGBM with binary regression tree as the base learner. The *name* column shows the name of the tuned hyperparameter, the *description* column contains a short description of each hyperparameter that is tuned, and the *Distribution* column displays what distribution/search range was used in the search.

Name	**Description**	**Distribution**
max_depth	Maximum depth of a tree	Random integers from $[15, 10000]$
reg_alpha	L1 regularization parameter on weights	Log-uniform at $[e^{-8}, e^{2}]$
reg_lambda	L2 regularization parameter on weights	Log-uniform at $[e^{-8}, e^{2}]$
num_leaves	Maximum number of leaves in a tree	Random integers from $[10, 10000]$
learning_rate	The learning rate	Log-uniform at $[e^{-7}, 1]$
subsample	The fraction of training data used for each tree	Uniform at $[0.5, 1]$
n_estimators	The number of trees used	Random integers from $[20, 205]$
colsample_bytree	The fraction of columns used for each tree	Uniform at $[0.5, 1]$
min_data_in_leaf	The minimum number of observations belonging to a leaf node during training	random integers from $[1, 5000]$
min_sum_hessian in_leaf	The minimum sum of the hessian required to split a leaf node	Log-uniform at $[-15, 5]$

Table 6: Hyperparameter Distributions for GBM with binary regression trees as base learners. The *name* column shows the name of the tuned hyperparameter, the *description* column contains a short description of each hyperparameter that is tuned, and the *Distribution* column displays what distribution/search range was used in the search.

Name	**Description**	**Distribution**
max_depth	Maximum depth of a tree	Random integers from $[5, 30]$
learning_rate	The learning rate	Log-uniform at $[e^{-7}, 1]$
subsample	The fraction of training data used for each tree	Uniform at $[0.5, 1]$
n_estimators	The number of trees used	Random integers from $[20, 205]$
min_weight fraction_leaf	The minimum fraction between the new weight and the sum of all weights required to be a leaf node.	Uniform at $[0, 0.45]$



NTNU

Norwegian University of
Science and Technology