Statistical Learning in High-Dimensional Data

The performance of random forests compared to penalized linear regression in scenarios with sparse informative features

by

E.C. Brouwer

to obtain the degree of Bachelor of Science
at the Delft University of Technology,
to be defended publicly on Friday June 20, 2025 at 13:00 PM.

Student number:	5172705	
Project duration:	February 24, 2025 – June 20, 2025	
Thesis committee:	Dr. H.N. Kekkonen,	TU Delft, supervisor
	Ir. J.C. van der Voort,	TU Delft, supervisor
	Dr.ir. M. Keijzer,	TU Delft

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Layman's Abstract

In today's world, we collect huge amounts of data. This results in very large datasets, making it hard to figure out what the relationship between the data is. For example, we can collect data of the human genetics where we have data on many genes but only from a relatively small number of people. Furthermore, only a few genes might be important for predicting the chance of someone developing a disease, while the other genes can be considered as noisy variables. This thesis compares three popular methods used to analyse such data: random forests, Ridge regression, and Lasso regression. Random forests are powerful, flexible models that can capture complex relationships, especially when the data patterns are not straightforward. Ridge and Lasso are simpler, linear methods that reduce the focus on the noisy variables and help identify the most important ones.

To study these methods fairly, synthetic datasets were generated: one where relationships are simple and linear, and one where they are complex and non-linear. Next, it was tested how well each method predicts outcomes and how accurately they find the few important features, while also considering how long it takes to build the model.

The results show that no single method always works best, and that the performance depends on the structure of the data. Lasso was often the preferred choice when quick predictions were needed or when the number of important variables was very small. It performed well even in some complex situations, challenging the idea that only advanced models work for non-linear problems. On the other hand, random forests were better at identifying important features in non-linear data, making them useful for exploring which variables matter most. Ridge regression was the fastest method but showed weaker performance overall. This study highlights that choosing the right model depends not just on prediction accuracy, but also on how complex the data is, how many important features are present, and how quickly results are needed.

E.C. Brouwer
Delft, June 2025

Abstract

In modern data analysis, high-dimensional datasets where the number of features far exceeds the number of observations are increasingly common. In such settings, identifying sparse informative features is a critical challenge. This thesis investigates the comparative performance of random forests and two penalized linear regression techniques, Ridge and Lasso regression, in scenarios with few informative features. Synthetic datasets were generated to simulate both linear and non-linear relationships between features and the target variable. The models were evaluated using mean squared error (MSE) to measure prediction accuracy and two custom metrics assessing the ability to identify informative features among the top-ranked variables. Additionally, computational efficiency was assessed.

In linear settings, Lasso consistently outperformed the other models in both prediction and feature selection, particularly at the lower informative ratios. Ridge regression demonstrated reasonable feature selection accuracy in linear settings but underperformed in prediction, likely due to a limited hyperparameter grid. In non-linear settings, random forests were most effective at identifying informative features, though the prediction performance of Lasso remained competitive and slightly outperformed random forests under high sparsity. Training time analysis further showed that the penalized linear models were substantially faster than random forests. These findings underscore that no single model is universally optimal, as performance varies depending on data structure, sparsity, and the modelling objective. They also demonstrate that improved feature selection does not necessarily guarantee better prediction in complex, high-dimensional settings. These results have practical implications for domains like genomics, where balancing accuracy, interpretability, and computational efficiency is critical.

E.C. Brouwer
Delft, June 2025

Contents

1 Introduction

> "We're drowning in information and starving for knowledge"
> — Rutherford D. Rogers

This quote highlights a critical challenge of our time: while the availability of data has exploded, extracting meaningful insights remains a difficult task [5]. The last decades have witnessed rapid technological advancements that have enabled the collection and storage of enormous amounts of data. As a result, high-dimensional datasets have become increasingly common. These datasets arise in diverse fields such as genomics, neuroscience, healthcare analytics, astronomy, and image or text analysis, business and finance [6, 11]. An example is the UK Biobank, which collects extensive health-related data from approximately half a million UK participants. A recent study completed exome sequencing for these participants, which makes up about 2% of the genome, includes the protein-coding regions of DNA that are especially important for identifying disease-related or rare genetic variants [3, 12]. Statistical learning, which refers to a collection of methods designed to understand and model relationships in data, could be applied to such datasets to predict if someone is likely to have a disease based on their genetic information.

A problem that arises in high-dimensional data is that often only a few features are informative. The non-informative features in the data can be considered as noise. This brings us to the following quote:

> "Signals always come with noise: It is trying to separate out the two that makes the subject interesting."
> — David Spiegelhalter[1]

This quote emphasizes the need to recognize which features are informative and which are not. For example, when looking at human DNA, only a few genes may be relevant for the development of a certain disease. Another goal of statistical learning could be to find out which features, or in this case, genes, are important or strongly related to a disease. One way to deal with sparsity of informative features is through feature selection. In high-dimensional settings, feature selection methods aim to identify the informative features while discarding the irrelevant ones, thus improving model interpretability and potentially reducing prediction error. Several statistical methods have been developed to address high-dimensional sparse data. For example, Ridge regression and Lasso regression [10]. These penalized regression techniques add a regularization term into the model to prevent overfitting and perform feature selection. Ridge regression adds an ℓ_2 penalty, which shrinks coefficients but does not set them exactly to zero, while Lasso regression uses an ℓ_1 penalty that encourages sparsity by setting some coefficients exactly to zero [4, 6].

Another popular method in high-dimensional analysis is random forests, which is a flexible, non-linear ensemble method that often yields high prediction accuracy, particularly in complex biological datasets [8]. Random forests are commonly applied in gene expression classification, protein expression analysis, and statistical genetics. However, due to their flexibility, random forests can overfit, especially in sparse high-dimensional settings. In such cases, simpler models such as Ridge and Lasso may be more robust [4, 6]. Although random forests are expected to perform well with sufficient data, in practice, limited data and many irrelevant features in high-dimensional datasets often hinder their performance. In such situations, simpler models like Ridge and Lasso regression, which apply regularization to reduce variance and control model complexity, can offer more robust performance. Although these linear models are less flexible, this constraint could be an advantage when the number of informative features is sparse or when the number of features exceeds the number

[1] Quoted in 'The Art of Statistics: How to Learn from Data' by David Spiegelhalter [13].

of observations.

This thesis aims to investigate the performance of random forests compared to penalized regression in high dimensional settings, where the number of informative features is sparse. Specifically, we compare the performance of random forests with two forms of penalized linear regression: Ridge regression and Lasso regression. The performance will be assessed in terms of prediction error and by the ability of the models to identify the informative features. In addition, the computational efficiency will be assessed. The effect on the performance will be researched by changing the proportion of informative features, and the noise amplitude.

To conduct this investigation, synthetic data will be generated for two scenarios: one where the underlying relationship between inputs and the target is linear, and one where it is non-linear. Based on this data, a simulation study will be carried out. The performance of the models will be evaluated using three main criteria: mean squared error (MSE) for assessing prediction accuracy, and two new metrics will be introduced for evaluating a model's ability to identify the informative features. The first new metric looks only at the correct selected informative features in the five most relevant features according to the model. The second metric is more lenient and looks at the correct selected features in the ten most relevant features according to the model. The computational efficiency will also be taken into account.

Throughout this thesis, two primary textbooks serve as foundational references for the literature background: 'The Elements of Statistical Learning: Data Mining, Inference, and Prediction' by T. Hastie et al. [4] and 'An Introduction to Statistical Learning: with Applications in Python' by G. James et al. [6].

The thesis is structured as follows. Chapter 2 provides an introduction to several important concepts in statistical learning. Chapter 3 introduces linear regression. First, the least squares method is explained, followed by Ridge and Lasso regression, including their respective shrinkage penalties. Chapter 4 presents the workings of random forests for regression after introducing regression trees and bagging. Chapter 5 describes the procedure used to generate synthetic linear and non-linear data. Chapter 6 outlines the setup and design of the simulation study. Finally, the results for the linear and non-linear cases are presented in Chapter 7 and Chapter 8, respectively. Lastly, the conclusion and discussion points can be found in Chapter 9.

2

Fundamental Concepts in Statistical Learning

Supervised statistical learning involves building a statistical model for predicting a target variable, based on one or more inputs. In this thesis, we will discuss regression problems, where the output of the statistical model is continuous. This chapter will introduce the relevant and fundamental concepts of statistical learning for regression, on which the next chapters of this thesis will build.

2.1. A Statistical Model

The input variables of the statistical model are denoted as $X_1, X_2, ..., X_p$, where each $X_j \in \mathbb{R}$. These input variables are random variables and referred to as features. The total number of features is represented by p. Features are measurable properties or characteristics that serve as inputs to the model. For example, in a genomics context, a feature could represent the expression level of a specific gene in a sample. The target variable $Y \in \mathbb{R}$ is the quantity the model aims to predict. In the same setting, Y could represent a continuous score, such as a patient's likelihood of developing a certain disease. We assume that the features $X = (X_1, X_2, ..., X_p) \in \mathbb{R}^p$ and target variable Y have a relationship of the following form:

$$Y = f(X) + \sigma\varepsilon, \tag{2.1}$$

where $f : \mathbb{R}^p \to \mathbb{R}$ is a fixed, unknown function and $\varepsilon \sim \mathcal{N}(0,1)$ represents standard Gaussian noise. The term $\sigma > 0$ controls the noise amplitude. The ε in the error term $\sigma\varepsilon$ is independent of X. The quantity $\sigma\varepsilon$ is added because there is always some noise in the measurements. The noise term may contain unmeasured features that are relevant for predicting Y, or contain unmeasurable variation. As an example, in gene expression data, unmeasurable variation can come from measurement errors, and unmeasured features from environmental influences. Statistical learning refers to the set of approaches for estimating the function f.

It is often relatively easy to gather a set of inputs X, but difficult to obtain the corresponding target variable Y. For example, in medical diagnosis, it is straightforward to collect patient data such as age, blood pressure, and cholesterol levels, the features, but it is much harder to obtain the true disease status, the target variable, because it requires costly and invasive tests. Therefore, a prediction $\hat{Y} \in \mathbb{R}$ for the value of Y is often desirable. The prediction $\hat{Y}$ is created as follows:

$$\hat{Y} = \hat{f}(X), \tag{2.2}$$

where $\hat{f} : \mathbb{R}^p \to \mathbb{R}$ represents the estimate for f.

The total prediction error depends on two terms, the reducible error and the irreducible error. The expected prediction error can be decomposed into two types of errors as follows:

$$\begin{aligned}\mathbb{E}(Y - \hat{Y})^2 &= \mathbb{E}\left[f(X) + \sigma\varepsilon - \hat{f}(X)\right]^2 \\ &= \underbrace{\left[f(X) - \hat{f}(X)\right]^2}_{\text{Reducible}} + \underbrace{\text{Var}(\sigma\varepsilon)}_{\text{Irreducible}},\end{aligned}$$

where $\text{Var}(\sigma\varepsilon)$ is the variance of the error term. The first term is called the reducible error because it depends on how well we estimate the function f. By applying the most fitting statistical learning technique, the value of the reducible error term could be decreased. The irreducible error term however, comes from the noise term

when generating our true value Y in Equation (2.1) and cannot be reduced by our statistical model.

When prediction performance is the primary goal, we aim to choose a function $\hat{f}$ that best approximates the true underlying function f. A model that closely estimates f will yield the most accurate predictions $\hat{Y}$ for the true target variable Y and will decrease the reducible error. However, in some cases, the objective is not just to predict accurately, but to understand which features are most influential. For instance, in fraud detection, capturing complex patterns is crucial for accuracy, whereas in medical research, identifying key biomarkers demands interpretability.

The ability to identify the important features is particularly important in high-dimensional settings, where the number of features exceeds the number of observations, or only a small subset of features is truly informative. An informative feature contributes significantly to predicting Y, while a non-informative feature has little or no association with the target variable. In such scenarios, interpretability and the identifying of important variables become more important than prediction performance alone. Linear models are relatively simple and interpretable, but do not always lead to a model with the highest performance. In contrast, non-linear approaches like random forests often lack the interpretability, but could lead to a higher performance due to their complexity. This is known as the trade-off between model complexity and interpretability.

2.2. The Training and Evaluation Process

To estimate the function f in Equation (2.1), we use available data known as the training set. This training set consists of n observations, where the values of each feature and the corresponding value of the target variable are known. Each observation in the training set is represented as a pair (x_i, y_i), where $x_i = (x_{i1}, x_{i2}, \ldots, x_{ip}) \in \mathbb{R}^p$ and $y_i \in \mathbb{R}$. Each $x_{ij} \in \mathbb{R}$ corresponds to a realization of the random variable X_j for observation i, and each entry $y_i \in \mathbb{R}$ denotes the target value corresponding to observation i. Thus, x_i is a row vector of p features, and y_i is the associated target value. The training set can be organized into a feature matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$ by stacking the feature vectors x_i, together with a target column vector $\mathbf{Y} \in \mathbb{R}^n$ by stacking the values of y_i. The size of the training set is determined by the number of observations n and the number of features p. When the amount of available data is limited, the available data is split into a training set and a test set. The training set is used to train the model, while the test set is used to estimate the performance of the prediction for new, unseen data. Usually, 80% of the available data is part of the training set, while the other 20% is part of the test set. The exact split ratio can vary depending on the application. Generally, the majority of the data is allocated to the training set to ensure the model has enough information to learn from.

A statistical learning method, such as linear regression, is then applied to the training data. The goal during training is to find a function $\hat{f}$ such that $Y \approx \hat{f}(X)$ for any observation (X, Y). After selecting a statistical model, the training starts. In the training process, the model will see the observations of the training set and adjust certain model parameters, such as coefficients in linear regression, to improve the model. This process is called fitting the model and is performed using various optimization techniques. Some models also have hyperparameters, such as the regularization strength or the number of trees in random forests, which are determined independently of the training process. The values of the hyperparameters are not known beforehand, but are tuned with the use of k-fold cross-validation or other validation methods. For example, a separate validation set can be used for tuning hyperparameters. An advantage of cross-validation is that it avoids splitting the available data into three separate sets: training, validation, and testing.

A representation of 5-fold cross-validation can be found in Figure 2.1. During k-fold cross-validation, the original training set is split into k equally sized, non-overlapping, folds. Each fold takes a turn as the validation set, while the remaining folds are used for training. This implies that the process is repeated k times. During one iteration, the model is trained with a combination of a set of possible values for each hyperparameter setting, and the performance of that setting is measured using the validation set. The optimal hyperparameters are the ones that achieve the best average performance across all validation sets. Once the optimal hyperparameters are found, the final model is retrained on the entire training set using these optimal hyperparameter values. It is important to note that the performance on the validation set should not be used to estimate the performance on new, unseen data. During cross-validation, each validation fold contributes to the selection of the hyperparameters, which can lead to an overestimation of the model's performance. For estimating the performance on new, unseen data, only the test set should be used.

Finally, once a model has been trained and validated, we evaluate its performance using appropriate metrics. In regression problems, a common measure is the mean squared error (MSE), which captures the average squared difference between the predicted and true values of the target variables of the test set. The MSE is defined as:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{f}(x_i))^2. \tag{2.3}$$

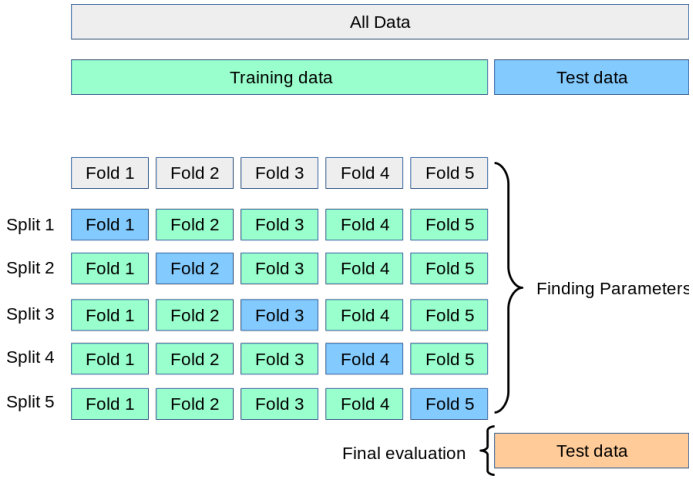


Figure 2.1: An example of 5-fold cross validation [7].

A low MSE indicates that the model's predictions are close to the actual values, whereas a high MSE suggests poor prediction performance and large deviations between predicted and true values.

Precision and recall are commonly used to evaluate classification models. In such models, the prediction is not continuous value but a class to which the measured variables belong. These metrics can also be used for evaluating feature selection methods, where the classes are defined as 'informative' (the positive class) and 'noninformative' (the negative class). This is particularly useful in simulated settings, where the truly relevant features are known. The precision measures the proportion of predicted positive cases, the selected features, that were actually informative and is defined as:

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}, \tag{2.4}$$

where the true positives are informative features that were correctly selected and false positives are noninformative features that were incorrectly selected. A precision of 1 means that all selected features are informative, and a precision of 0 means that none of the selected features are informative. The recall measures the proportion of actual positive cases, the number of truly informative features, that the model was able to identify as informative:

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}, \tag{2.5}$$

where the false negatives are informative features that were not selected. A recall of 1 indicates that all informative features have been selected, while a recall of 0 means that none of the informative features were correctly identified. Feature selection helps identify and keep only the most relevant features, which reduces noise and improves model performance. Whether a feature is considered selected by the model, and therefore classified as informative, depends on the model itself. This will be discussed in Chapter 5.

A graphical representation that summarizes the training, validation and evaluation process can be found in Figure 2.2. The training and evaluation process is often applied to various statistical learning methods. Depending on the application of the statistical model, different evaluation metrics are used to determine which

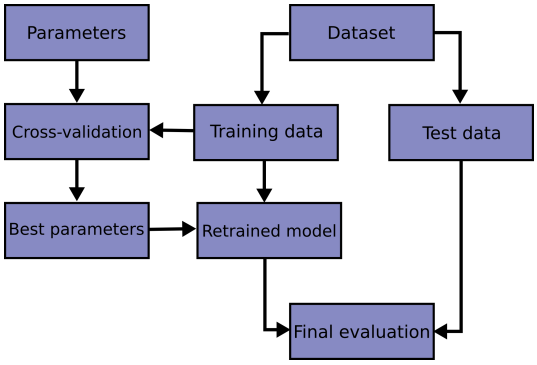


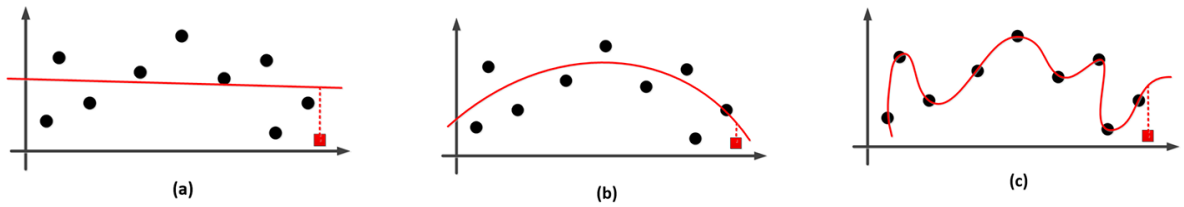
Figure 2.2: A graphical representation of the training, validation and evaluation process [7].

model performs best. When the prediction performance is most important in the application, the MSE is used to identify the best model. When feature selection is most important, either for selecting as many informative features as possible or for having a high accuracy in selecting informative features, the recall and precision can be used as metrics, respectively.

2.3. Bias-Variance Trade-Off

No single statistical model consistently performs best across all scenarios. The optimal model depends on the underlying structure of the data, and each model offers distinct strengths. Flexible models, like random forests, are able to capture complex patterns in data. They are useful when the true relationship between features and the target variable is non-linear or complicated. However, with this flexibility comes a risk: the model might become too closely tailored to the training data, capturing not just the true patterns but also the random noise. This issue is known as overfitting. An overfit model performs well on the training data but poorly on new, unseen data. In contrast, less flexible models such as linear regression assume a linear relationship between input features and the target variable. These models are simpler and easier to interpret. They are less likely to overfit, but may miss important patterns, a problem known as underfitting.

Figure 2.3 illustrates how model complexity affects the training and test error, showing examples of underfitting, good fitting and overfitting. The left graph shows a simple linear model that underfits. Both the error of the training set and the error of the test set are large. The middle graph seems like a model with a good fit as it captures the true pattern quite well. The error on the training set and on the test set is small. The right graph shows a model that overfits, where the model performs very well on the training data, but performs poor on the test set.



(a) Underfitting (b) Good Fit (c) Overfitting

Figure 2.3: Model fitting comparison: three different models trained on the same observations that are represented as black dots. The test set consists of one red square observation. The error for the test set is shown by the red dotted line [2].

Restricting the shape of the model function $\hat{f}$, as defined in Equation (2.2), introduces error. This type of error is known as bias. Approximating a complex underlying relationship with a simpler model causes high bias, which leads to underfitting. Variance, on the other hand, is the error introduced by the model's sensitivity to small fluctuations in the training data. High variance leads to overfitting. This leads to an important concept in statistical learning: the bias-variance trade-off. A highly flexible model typically has low bias, so it can closely follow the training data. However, they generally have a high variance, causing its predictions to change a lot with small changes in the data. A simpler, less flexible model often has high bias, so it misses some patterns. But they generally have low variance, resulting in more stable predictions. The goal is to find a good balance between bias and variance so that the model generalizes well to new data. An example of a simple model with a low variance and an example of a complex model with a high variance are shown in Figure 2.4. On the left, a simple linear model remains relatively unchanged when a single data point is slightly moved, which indicates low variance. On the right, a complex model changes its shape a lot after the same adjustement, demonstrating high variance. Although the complex model may fit the original training data more accurately, this instability makes it unreliable for making predictions on new, unseen data. The goal in statistical learning is to find a model that finds the right balance between bias and variance. This balance ensures that the model captures meaningful patterns in the training data while maintaining the ability to generalize well to unseen data.

In summary, building a good statistical model involves more than just fitting the data. It requires balancing flexibility and simplicity, selecting the right features, tuning hyperparameters, and carefully evaluating performance.

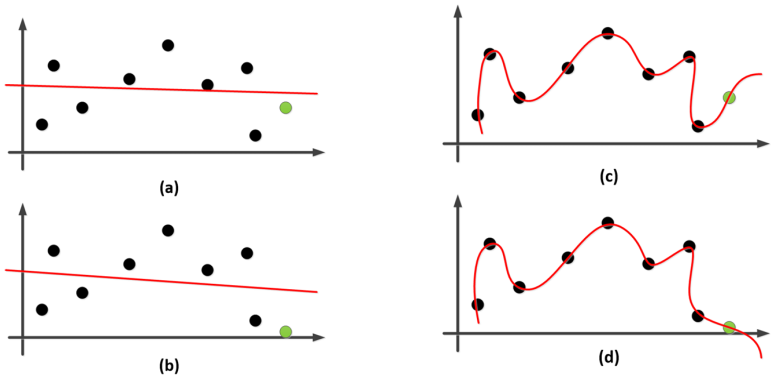


Figure 2.4: The red lines represent the model after training on the observations which are represented as the dots. The left side shows a simple linear model and the right side a complex model. When the green dot in the training set is moved and the model retrained, we get the figures below [2].

3

Linear Regression and Regularization

This chapter will focus on linear regression and regularization techniques. Two of the statistical learning models that will be discussed are Ridge regression and Lasso regression. This chapter will first introduce the linear regression model and explain how it is fitted using the least squares method, and discuss its limitations. Then, Ridge regression and Lasso regression will be described in detail, emphasizing on how they improve compared to fitting using the least squares method through regularization and feature selection.

3.1. Linear Regression with Ordinary Least Squares

Linear regression assumes that the relationship between the features and target variable of the observations (X, Y) is linear. Therefore, the function $f(X)$ of Equation (2.1) is assumed to have the following form:

$$f(X) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j. \tag{3.1}$$

Here, β_j influences feature X_j. The true values of the parameters $\beta = (\beta_0, \beta_1, \ldots, \beta_p)^T$ are unknown. During training, these coefficients are optimized by a selected method of choice. The resulting estimation of $f(X)$, and thus the prediction $\hat{Y}$ according to Equation (2.2), is as follows:

$$\begin{aligned} \hat{Y} &= \hat{f}(X) \\ &= \hat{\beta}_0 + \sum_{j=1}^{p} X_j \hat{\beta}_j, \end{aligned}$$

where $\hat{\beta} = (\hat{\beta}_0, \hat{\beta}_1, ..., \hat{\beta}_p)^T$ are the estimates of β.

A method that is often used to fit the coefficients in linear regression is ordinary least squares (OLS). This method optimizes the estimates of β in Equation (3.1) such that the residual sum of squares (RSS) is minimized. The expression of the RSS is:

$$\begin{aligned} \text{RSS}(\beta) &= \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \\ &= \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_{i1} - \beta_2 x_{i2} - \ldots - \beta_p x_{ip})^2, \end{aligned} \tag{3.2}$$

where $\hat{y}_i = \hat{f}(x_i)$ is the prediction of the target value based on x_i from observation (x_i, y_i) of the training set. Minimizing the RSS ensures the model fits the training data as closely as possible by reducing the prediction error. A solution using the least squares method always exists. However, it is not always unique. The solution is unique when the matrix $\mathbf{X}^T\mathbf{X}$ is of full rank, meaning that the columns of random feature matrix $\mathbf{X}$ must be linearly independent. As a result, $\mathbf{X}^T\mathbf{X}$ is invertible. When the solution is unique, the closed-form solution is given by:

$$\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y},$$

where **Y** represents the target vector. If the solution is not unique, there are infinitely many solutions that minimize the RSS. More details on the existence and uniqueness of the least squares solutions are provided in Chapter 3 of 'The Elements of Statistical Learning' by T. Hastie et al. [4]. Figure 3.1 presents an example of a linear regression model constructed using the ordinary least squares method.

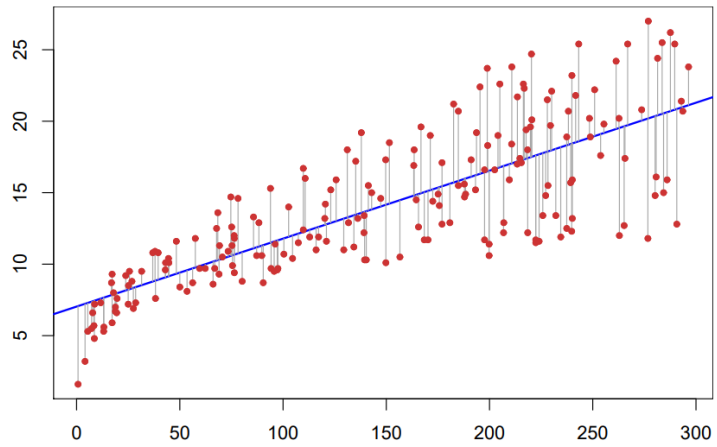


Figure 3.1: The least squares fit for the training observations that are represented as red dots. The grey lines indicate the residual and the blue line represents the fitted model [6].

Linear regression with OLS typically has lower bias than regularized models such as Ridge or Lasso regression. However, it can suffer from high variance, especially in high dimensional data where the number of features is large relative to the number of observations. In that case, the model is too flexible, which can cause a low prediction accuracy. Specifically, when the number of features p is close to, equal to or greater than the number of observations n, least squares regression can perfectly fit the training data, resulting in an RSS equal to zero. While this may seem ideal, it almost always leads to overfitting, where the model captures noise instead of the true underlying relationship. As a consequence, the model performs poorly on new, unseen data. An example of this phenomenon is provided in Figure 3.2, where $p \approx n$. The model on the right fits the training data perfectly, including any noise present in those few observations. In contrast, the model on the left trained on more observations, shows that the model balances between fitting the data, including the noise, and capturing the true underlying pattern. If there was only one observation in this example, the least squares method would yield infinitely many solutions, none of which would be meaningful. This demonstrates why linear regression with OLS is not preferred in such settings.

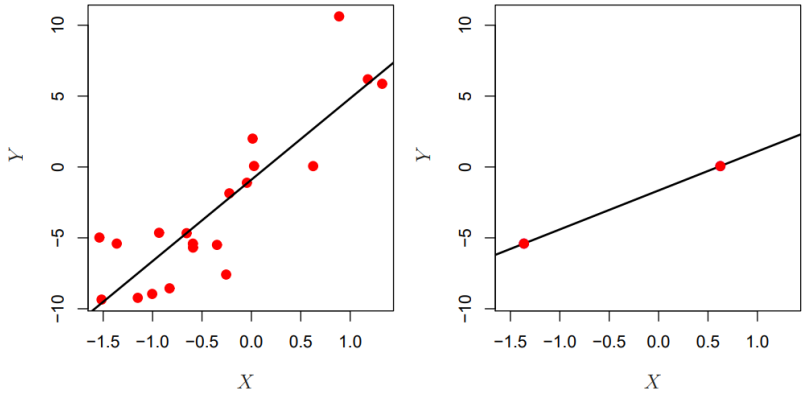


Figure 3.2: Linear Regression with OLS in a one dimensional feature space. On the left, the model is trained with 20 simulated observations. On the right, the model was trained with two simulated observations [6].

3.2. Ridge Regression

Ridge regression is similar to linear regression with ordinary least squares, as described in the previous section. The difference lies in how the optimal parameters for $\beta = (\beta_0, \beta_1, \ldots, \beta_p)^T$ in Equation (3.1) are determined.

While OLS minimizes the residual sum of squares (RSS) as shown in Equation (3.2) to determine the optimal coefficients $\hat{\beta}$, Ridge regression instead finds the optimal coefficients by minimizing the following expression:

$$\text{RSS} + \lambda \sum_{j=1}^{p} \beta_j^2, \tag{3.3}$$

where $\lambda > 0$ is hyperparameter that is tuned using cross-validation or a separate validation set. The term that is added to the RSS is called a shrinkage penalty. One of the advantages of Ridge regression is that this optimization problem has a closed-form solution:

$$\hat{\beta} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{Y},$$

where $\mathbf{I}$ is the identity matrix, $\mathbf{X}$ the random feature matrix and $\mathbf{Y}$ the target vector. This solution always exists and is unique for any $\lambda > 0$, in contrast with the least squares method. By adding the term $\lambda\mathbf{I}$, a positive value λ is added to each of the diagonal elements of the matrix $\mathbf{X}^T\mathbf{X}$. Since $\mathbf{X}^T\mathbf{X}$ is symmetric and positive semi-definite, this addition shifts all its eigenvalues by λ, making the resulting matrix $\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}$ strictly positive definite. Consequently, this matrix is always invertible, ensuring the existence and uniqueness of the Ridge regression solution for any $\lambda > 0$.

Furthermore, the closed-form solution makes Ridge regression computationally efficient for moderate-sized datasets. However, when the number of features is very large, iterative methods are often used instead of computing the matrix inverse directly. For the details of obtaining the closed-form solution, the reader is referred to Chapter 3 of 'The Elements of Statistical Learning' by T. Hastie et al. [4].

The goal of the RSS in the OLS method was to reduce the prediction error, by fitting the training data as well as possible. Keeping the RSS low is still important in Equation (3.3). However, it is also important to minimize the second term, the shrinkage penalty. The tuning parameter λ influences the impact of the shrinkage penalty. The larger λ becomes, the more the values of estimated coefficients $\hat{\beta}_1, \hat{\beta}_2, \ldots, \hat{\beta}_p$ are pushed close to zero to reduce the value of the final expression (3.2). Thus the optimal values for β are dependent on hyperparameter λ. This hyperparameter controls the trade-off between fitting the training data well and keeping the model coefficients small and centred around zero. This restriction reduces the variance. The shrinkage penalty excludes the intercept term β_0. This is because β_0 represents the baseline prediction when all feature values are zero and is not associated with any particular feature. Including it in the penalty would undesirably shift the entire prediction up or down regardless of the input features. By leaving β_0 unpenalized, Ridge regression maintains the model's ability to accurately capture the mean of the target value.

In contrast to linear regression with OLS, which is scale equivariant, Ridge regression is sensitive to the scale of the input features. The penalty term $\sum_{j=1}^{p} \beta_j^2$ depends on the size of the coefficients, and the scale of each feature X_j affects the magnitude of its corresponding coefficient β_j. As a result, features measured on different scales are penalized unevenly. Specifically, if a feature has a larger scale, its coefficient must become smaller to produce a similar effect on the predicted outcome. To ensure that all features are treated equally by the penalty term, it is important to standardize them before fitting the model. Standardization, which transforms each feature to have mean zero and standard deviation one, removes the dependency between the feature scales and the coefficient magnitudes. As a result, the regularization accurately reflects their relative importance in predicting the outcome rather than differences in their units of measurement.

The strength of Ridge regression lies in the bias-variance trade-off due to the shrinkage penalty. For some values of λ in the shrinkage penalty, the bias increases slightly and the variance decreases significantly. However, if λ is too large, the bias will have increased too much resulting in a larger prediction error. Therefore, λ is tuned. Compared to linear regression with OLS, Ridge regression can perform quite well using the bias-variance trade-off in high-dimensional settings.

Ridge regression may shrink the coefficients β closer to zero, but the coefficients are not set exactly to zero. As a result, every feature will be selected and contributes to the final Ridge regression model. This will not lead to a sparse model, which may be a limitation in situations where interpretability or feature selection is desired. To determine which features are considered informative and play an important role in the prediction, the magnitude of the coefficients can be used as measure. A larger magnitude of a coefficient β_j, indicates a greater impact of the corresponding feature X_j on the model's prediction. However, when the number of features is large, this can make the model more difficult to interpret, as all features remain part of the final model, even those with only a minor influence or that are uninformative. Lastly, Ridge regression generally performs well when the underlying relationship is linear as it assumes linearity.

3.3. Lasso Regression

Similar to Ridge regression, a shrinkage penalty is added to the RSS from Equation (3.2) to find the optimal coefficients in linear regression. For Lasso, to find the optimal estimates $\hat{\beta} = (\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_p)^T$ for the parameters in Equation (3.1), the following expression is minimized:

$$\text{RSS} + \lambda \sum_{j=1}^{p} |\beta_j|, \tag{3.4}$$

where $\lambda > 0$ is a hyperparameter that is tuned using cross-validation or a separate validation set. In contrast to Equation (3.3), there is no closed form solution due to the nondifferentiable penalty term. Therefore, the expression is minimized using various iterative algorithms, making Lasso generally more computationally expensive than Ridge regression. Examples of the algorithms that minimize the expression, are provided at the end of Chapter 3 in 'The Elements of Statistical Learning' by T. Hastie et al. [4]. There always exists at least one solution that minimizes Equation (3.4), although the solution is not necessarily unique. The uniqueness depends on the structure of random feature matrix $\mathbf{X}$. In general, when $\mathbf{X}$ has full column rank and $p < n$, the solution is unique. In high dimensional settings where $p > n$, or when features are highly correlated, multiple solutions optimize the equation. More details on the solution of Lasso and its uniqueness are provided in a paper by R. Tibshirani [14].

The key difference in the shrinkage penalty between Ridge and Lasso regression is that Lasso takes the absolute value of the coefficients β_j, rather than squaring them. Mathematically, this means that Lasso applies the ℓ_1 norm to β, and Ridge applies the ℓ_2 norm to β, both excluding β_0. As a result, with the same reasoning from the Ridge penalty, each coefficient β_j is shrunk toward zero. Similar to Ridge regression, the intercept β_0 is not included in the penalty term to avoid shifting the baseline prediction. Lasso is also sensitive to the scale of the features, so it is important to standardize the input variables before fitting the model. This ensures that the regularization treats all features equally regardless of their units or scales.

An important distinction between the solutions of Ridge and Lasso is that when λ becomes sufficiently large in expression (3.4), some of the estimated coefficients are set exactly to zero. To explain this phenomenon, we look at an equivalent expression of the minimization expressions of Ridge and Lasso in Equation (3.3) and Equation (3.4):

$$\min_{\beta} \quad \left\{ \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 \right\} \qquad \text{subject to} \quad \sum_{j=1}^{p} \beta_j^2 \leq s \tag{3.5}$$

and

$$\min_{\beta} \quad \left\{ \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 \right\} \qquad \text{subject to} \quad \sum_{j=1}^{p} |\beta_j| \leq s, \tag{3.6}$$

respectively. For each λ in Equation (3.3) or Equation (3.4), there exists a corresponding scalar s such that Equation (3.5) or Equation (3.6) will obtain the same coefficient estimates. Note that the objective functions for Ridge and Lasso are equivalent, only the constraint regions differ. To visualize the results, we inspect Figure 3.3 that shows a two dimensional feature space. The least squares solution for β is represented by $\hat{\beta}$ and the blue diamond and circle represent the constraint regions of Lasso and Ridge, respectively. For sufficiently large s, the constraint regions will contain $\hat{\beta}$. In that case, Ridge and Lasso both obtain the least squares estimate. When s is small, the least squares estimates are outside of the constraint regions and we will find a different estimation. Each ellipse represents a contour of constant RSS, meaning all points on it share the same RSS value. The further away the countours are from $\hat{\beta}$, the more the RSS increases. The solutions for Ridge and Lasso correspond to the first point where the RSS ellipse intersects the constraint region. This point represents the minimum RSS within the constrained region. Because the constraint region of Ridge is a circle, the intersection generally does not lie on the axis. The diamond-shaped constraint region of Lasso has sharp corners aligned with the coordinate axes. As a result, the optimal solution is more likely to occur at a corner, where one or more coefficients are exactly zero. In higher dimensions, Ridge's constraint region remains smooth, while Lasso's retains its sharp corners. This increases the likelihood that multiple coefficients are set to zero simultaneously.

As a result of the ability to set coefficients equal to zero, Lasso regression produces a sparse model in which only a subset of the features is used. Therefore, feature selection takes place automatically, because when a coefficient β_j is set to zero, feature X_j will not influence the model's prediction in Equation (2.2). This sparsity

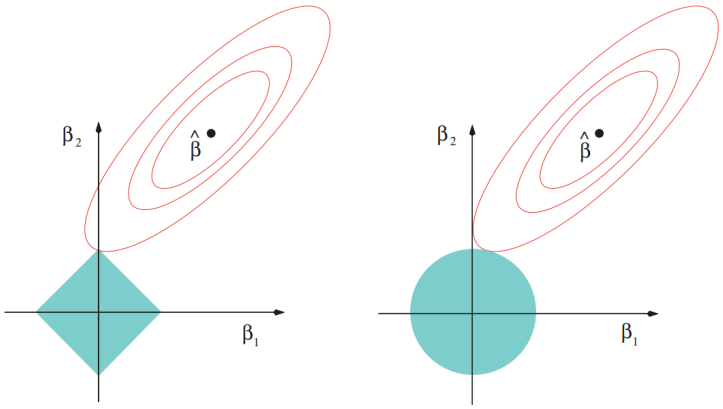


Figure 3.3: Visualisation of Ridge and Lasso regression in a two dimensional feature space. Contours of the RSS are shown as red ellipses and the constraint regions are shown as a blue diamond for Lasso and a blue circle for Ridge. These correspond to $|\beta_1| + |\beta_2| \leq s$ and $\beta_1^2 + \beta_2^2 \leq s$ respectively [4].

generally makes Lasso regression models more interpretable than Ridge regression models. The final values of the coefficients, and which ones are reduced to zero, depend on the chosen value of λ.

Similar to Ridge regression, when λ increases, the variance decreases and the bias increases. When many features are relevant in predicting the target variable, Ridge regression generally has the advantage because all features are retained in the model. When only a few features are relevant, Lasso regression has the advantage since some coefficients can be shrunk exactly to zero. A limitation of Lasso is that strong feature selection can lead to instability of the feature selection. This instability arises because Lasso makes discrete decisions, as it either includes or excludes a feature based on the value of λ. As a result, small changes in the data can lead to different subsets of features being selected. Lastly, as Lasso regression is a linear methods it often performs well when the underlying relationship is linear.

4

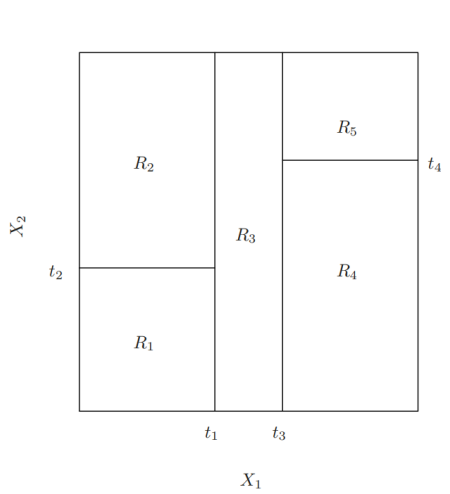
The Random Forests Algorithm

This chapter focuses on random forests, which is an ensemble method that builds on the concept of decision trees. Although a single decision tree is a simple and interpretable model, it often suffers from high variance and a tendency to overfit the training data. Random forests aim to solve this by combining the predictions of many individual decision trees to form a more robust and accurate model.

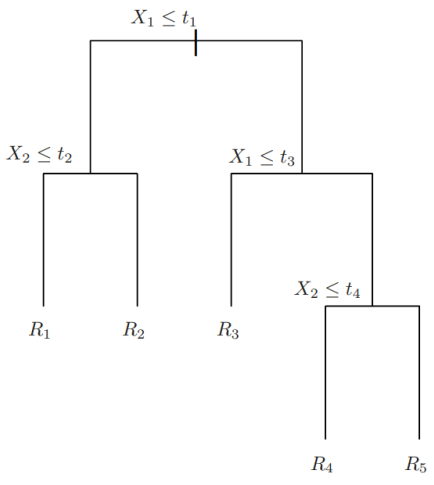
Before diving into the details of random forests, it is important to first understand decision trees for regression and the concept of bagging. Therefore, this chapter will begin by explaining how regression trees are constructed, followed by an introduction to bootstrap aggregation (bagging). Finally, it will present the technique of feature sampling, which is essential to random forests, and conclude with a discussion on feature importance.

4.1. Decision Trees for Regression

Decision trees are statistical learning models that can be used for both classification and regression. In this thesis, only regression trees will be discussed. In regression trees, the feature space, referring to the set of possible values for features $X_1, X_2, \cdots, X_p$, is divided into J distinct and non-overlapping regions, $R_1, R_2, \cdots, R_J$. For any observation (X, Y), with Y unknown, that falls into region R_j, the regression tree will predict $\hat{Y}$ as the average of the values of the target variable of all the training observations of that region R_j. To maintain simplicity and create an interpretable model, the feature space is split into regions that are high-dimensional rectangles, sometimes called boxes [6]. An example of a decision tree with only two features, where the feature space is therefore two dimensional, is shown in Figure 4.1.



(a) A two dimensional feature space that is split in different non-overlapping regions using recursive binary splitting.



(b) A decision tree corresponding to the non-overlapping regions shown in Figure 4.1a.

Figure 4.1: Example of recursive binary splitting in a two dimensional feature space and its corresponding decision tree representation [6].

The criterion that determines the measure of the quality of a split is the residual sum of squares (RSS), which for regression trees will have the following form:

$$\text{RSS} = \sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2,$$

where $\hat{y}_{R_j}$ is the average target variable of the training observations (x_i, y_i) within region R_j. However, it is computationally infeasible to consider every possible way to split the feature space to find the most optimal one that minimizes the RSS. Therefore, the method recursive binary splitting is used.

Recursive binary splitting is a method to split the feature space into different regions. This method is top-down, because it starts at the top of the tree when there is only one region, namely, the full feature space. Then the region is split in two, so two new branches are created downwards in the tree. Only binary splits are performed at each step, as they are computationally efficient and ensure that sufficient data remains in each resulting region. Although multi-way splits could be considered, repeated binary splits can achieve the same resulting regions. The method is also called greedy, because the best split is made at a particular step in the tree at that moment, without taking into account possibilities that could later on result in a better future split and thus final result. When deciding to split the region in a certain way, all features $X_1, X_2, \cdots, X_\text{p}$ and all possible cutpoints s are considered to base the split on. For example, the first two regions R_1 and R_2 will have the following form:

$$R_1(j,s) = (x_i \mid x_{ij} < s), \quad R_2(j,s) = (x_i \mid x_{ij} \geq s),$$

where each element x_{ij} represents the value of feature X_j for observation (x_i, y_i) of the training set. So cutpoint s represents the value at which the region is split, the feature X_j indicates the feature at which the split will take place and (x_i, y_i) are observations used during training that satisfy the condition. Minimizing the RSS for regression trees for the first split is equivalent to minimizing the following expression:

$$\sum_{i:x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2.$$

This expression is computationally feasible as it considers only two regions.The same process is repeated at the next split. However, the difference is that from this point on, only one of the newly created regions is split at a time. This process is repeated until a certain stopping criterion is satisfied, such as having a maximum amount of regions or by having no region containing more than a certain number of training observations. The maximum depth of a decision tree, related to the maximum amount of regions, is a hyperparameter that can be tuned. The minimum number of observations to split a node and the minimum number of observations required in the leaf of a node are also hyperparameters. The regression tree, built with recursive binary splitting, will result in the following estimate of $f(X)$ of Equation (2.1):

$$\begin{aligned} \hat{Y} &= \hat{f}(X) \\ &= \sum_{j=1}^{J} c_j \cdot \mathbf{1}_{(X \in R_j)}, \end{aligned} \tag{4.1}$$

where $c_j = \hat{y}_{R_j}$ and the indicator function is defined as

$$1_{(X \in R_j)} = \begin{cases} 1 & \text{if } X \in R_j, \\ 0 & \text{otherwise.} \end{cases}$$

Thus Equation (4.1) states that when X lies in region R_j, the prediction $\hat{Y}$ is the average of the target variables of the training observations in region R_j.

A decision tree built using the described method typically performs well on the training data. However, it often tends to overfit, resulting in poor generalization to unseen data. This is because the tree structure may become overly complex, capturing noise rather than underlying patterns in the training set. As a result, decision trees have a high variance. Even a small change in the training data can result in a very different final decision tree. As a result, an individual tree generally has a lower prediction accuracy than other methods.

An advantage of decision trees is that it has a relatively low bias. As a result, when the features and true target variable Y have a highly non-linear and complex relationship, decision trees may outperform classical approaches such as linear regression, which was discussed in Chapter 3. A comparison of the performance of

a decision tree versus a linear model can be found in Figure 4.2. For simplicity, the visualization takes only two features into account. The two graphs on top, where the true decision boundary is linear, the linear model performs well, while the decision tree requires many splits to approximate the boundary. On the bottom, where the true decision boundary is non-linear, the decision tree clearly outperforms the linear model. In addition to the flexibility of trees, they are often easily interpretable. Furthermore, they mirror human decision-making, can be easily visualized and can handle both categorical (non-numerical) features and numerical features without needing to take extra steps.

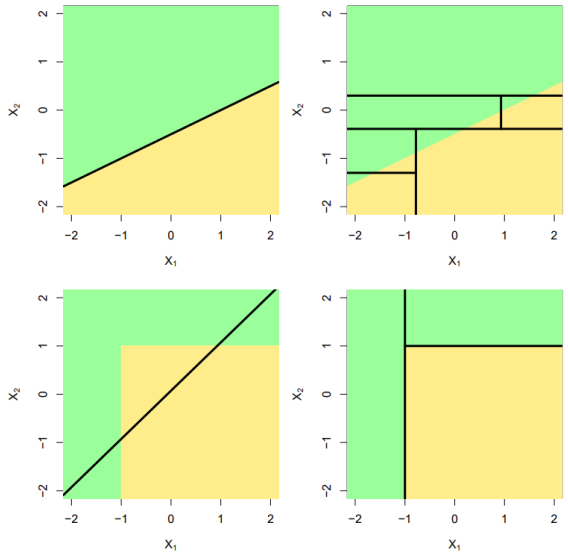


Figure 4.2: The two graphs above have a true linear decision boundary, indicated by the boundary of the two coloured regions. The two graphs below have a non-linear boundary, also indicated by the boundary of the two coloured regions. The left graphs show the result of a linear model, whereas the right graphs show the result of the non-linear model decision tree [6].

4.2. Bagging

An ensemble method combines multiple simple models, which often do not perform well on their own. Each individual model in the ensemble, referred to as a weak learner, makes its own prediction. These predictions are then aggregated to form the final prediction. In the case of regression, the final prediction is obtained by averaging the outputs of all individual models. By taking the average, the variance of the final model is reduced while maintaining a low bias. An example of a simple model that benefits from ensembling methods, is a decision tree, which suffers from high variance.

Bagging, short for bootstrap aggregation, is an ensemble method designed to reduce the variance of a predictive model. A diagram of bagging is provided in Figure 4.3. Instead of training a single model on the

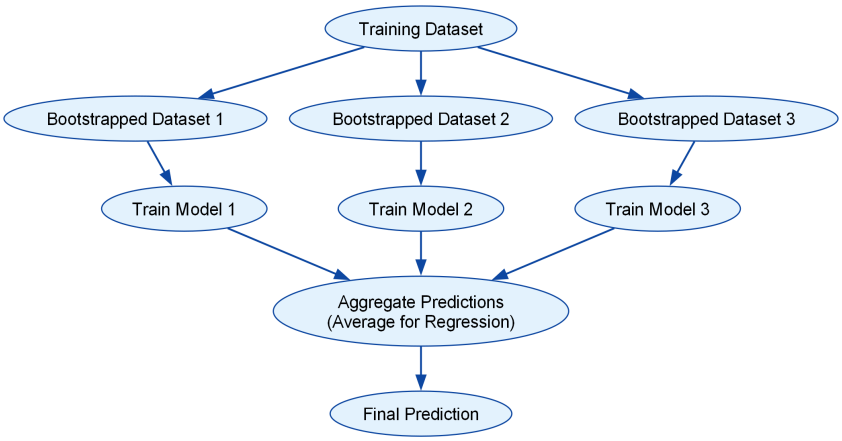


Figure 4.3: Schematic diagram of the bagging algorithm with three models showing bootrapping and aggregation of predictions (created by the author).

entire training dataset, bagging involves training multiple models. As bagging is typically applied to decision trees, the process will be described using this simple model. Each decision tree in the ensemble is trained on a different dataset, obtained by sampling with replacement from the original training set. This means that some rows may appear multiple times in the new training set for a certain decision tree, while others may not appear at all. Each bootstrapped dataset is the same size as the original training set. Because the sampling of the bootstrapped datasets is random and independent, the decision trees are trained on generally different version of the data. Therefore, the decision trees tend to end up being different from each other. The decision trees in the ensemble are intentionally grown deep, since overfitting by individual trees is counteracted by averaging their predictions. Unfortunately, the improvement of the performance of ensemble methods such as bagging, causes the interpretability to go down. In this case, a collection of trees is more difficult to interpret than an individual tree. Furthermore, a key limitation of bagging is that the decision trees in the ensemble can still be highly correlated if they often select the same features early in the tree to split on. For instance, if a particular feature is highly predictive, it might be selected in the top splits of every tree in bagging, making the trees too similar. This makes the variance reduction less effective in bagging.

4.3. Random Forests

Random forests build on the bagging method by also training an ensemble of decision trees on bootstrapped datasets. However, they further improve the effectiveness of the variance reduction by introducing feature sampling at each split. Each decision tree within the random forests makes its own prediction. The results of these individual predictions are used to form the final output of the random forests model. A diagram of the random forests algorithm is provided in Figure 4.4. The final estimation of $f(X)$ from Equation (2.1) using random forests for regression with B decision trees, will have the following form:

$$\begin{aligned}\hat{Y} &= \hat{f}^B(X) \\ &= \frac{1}{B}\sum_{b=1}^{B} T_b(X),\end{aligned}$$

where T_b represents the bth regression tree in the random forest. The number of trees, B, is a hyperparameter that is tuned using cross-validation or a separate validation set. Random forests generally do not overfit when the number of trees is increased. Usually a large enough number of trees is chosen, such that the error rate does not decrease significantly any more or it is based on computational limitations.

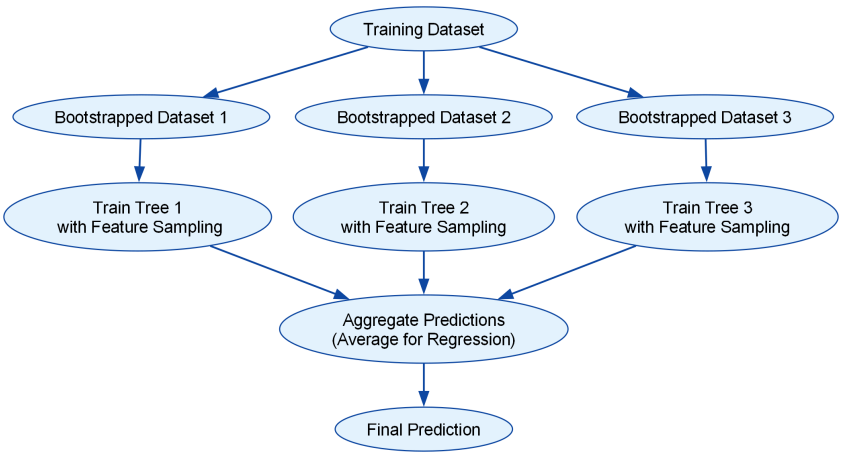


Figure 4.4: Schematic diagram of the random forests algorithm with three trees, showing bootstrapping, feature sampling during training, and aggregation of predictions (created by the author).

At each potential split in the decision tree, a random subset of $m \leq p$ features is selected, and only this subset is considered when deciding the best split. An important result of feature sampling is that the trees are decorrelated. By forcing each split to consider only a random subset of features, random forests encourage diversity among trees. When the trees are decorrelated, their individual errors are less likely to reinforce each other when averaged, making the ensemble prediction more stable. Generally, the lower m, the lower the correlation between the trees. The most optimal value of m is dependent on the data, therefore the value of m is a hyperparameter and tuned with cross-validation or a separate validation set. When the trees are decorrelated, the average of the resulting tree is less variable and therefore more reliable. In other words, the

variance is reduced. The bias of a random forest is the same as the bias of the individual decision trees in that forest. Thus when the variance is reduced using bootstrapped datasets and feature sampling, the performance of the prediction is improved.

Although random forests are effective in reducing variance and improving prediction performance, they do not consider the global relevance of individual features when building the trees. At each split, a random subset of features is selected with equal probability, and the feature importance is only measured using features within this subset. Therefore, the split is based on local relevance of the features. In contrast, standard decision trees always select the optimal feature out of all features at each split based on a splitting criterion such as the reduction in RSS. In random forests, the chance of selecting many irrelevant features increases as the number of such features grows. As a result, the irrelevant features may be chosen for splits. They do not meaningfully separate the data, which can lead to weaker splits, larger and more complex trees, and ultimately a higher generalisation error. Additionally, using irrelevant features may increase the computational cost of training the model, as more splits are needed to achieve a reasonable level of prediction accuracy [9]. Furthermore, random forests generally are computationally more intensive than simple models like Ridge and Lasso, as they require training and evaluating multiple decision trees. This can increase training time, especially on large datasets. Moreover, the input data must be passed through all trees in the forests, which can slow down the prediction time. This depends on the number of trees and the depth of the trees.

Random forests do provide an estimate of feature importance. This is typically measured by summing the total reduction in RSS for all splits that use a given feature, across all trees in the forest, and then normalizing it. The larger this normalized total reduction of the RSS for a feature, the more important the feature is considered by the model. An example of a graph presenting feature importance is shown in Figure 4.5. The length of the bars indicate the relative importance of that feature for predicting the target variable.Note that two features stand out as the most important, indicating they are likely the most informative for predicting the target variable.

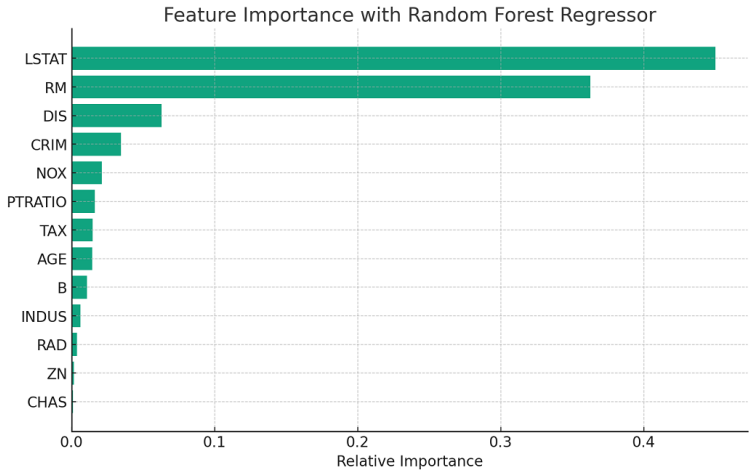


Figure 4.5: An example of a graph with the feature importance from a random forests regressor after training. The features are stated on the left and the corresponding bar represents the relative importance [1].

5
Setting

This chapter describes how the synthetic high-dimensional datasets used in the simulation study, described in Chapter 6, are generated. The objective is to create data settings that allow for a controlled, reproducible and systematic evaluation of model performance under varying conditions.

Two types of datasets are considered: linear and non-linear. In both cases, a random feature matrix is constructed by sampling values from a standard normal distribution. The number of observations and the number of informative features are fixed across all simulations. The proportion of informative features, and the noise amplitude are varied according to the experimental setup. The true target variable is then computed as either a linear or non-linear function of the informative features, depending on the type of dataset.

5.1. Data Generating Parameters

The high-dimensional datasets are generated using `Python` version 3.10.12. First, we fix the number of observations n in the training set to 100 observations. The number of observations n_{test} in the test set also contains 100 observations. As the data is synthetically generated, there is no limitation in the amount of available data for the test set. Choosing a sufficiently large test set will give a more robust and reliable estimation of model performance. We are interested in the high-dimensional settings where only a small number of features are informative and fix the number of informative features $p_{\text{informative}}$ at five informative features for every generated dataset. Since the number of informative features is fixed, the total number of features varies according to the chosen informative feature ratio. The total number of features p is determined by the informative feature ratio $r \in (0, 1]$ through the relation

$$p = \left\lfloor \frac{p_{\text{informative}}}{r} \right\rfloor,$$

where r controls the proportion of informative features among all features and $\lfloor \cdot \rfloor$ denotes the floor operation. We generate datasets with three different low informative feature ratios: 0.1, 0.01, and 0.001. Lastly, two values for the noise amplitude will be varied. Table 5.1 summarizes the relevant parameter values for generating the data. An overview of the chosen ratios r and the corresponding numbers of informative features $p_{\text{informative}}$ and noninformative features $p_{\text{noninformative}}$ is provided in Table 5.2. From these values, it is clear that the datasets are considered high-dimensional.

Parameter	**Definition**	**Value**
n	Total number of observations in the training set	100
n_{test}	Total number of observations in the test set	100
$p_{\text{informative}}$	Number of informative features	5
r	Proportion of informative features relative to the total number of features	0.001, 0.01, 0.1
σ	Noise amplitude	0.01, 0.1, 1

Table 5.1: Overview of relevant values for generating the high-dimensional datasets.

We consider a regression setting where the true target variable Y is generated according to the statistical model introduced in Equation (2.1):

$$Y = f(X) + \sigma\varepsilon, \tag{5.1}$$

r	$p_{\text{informative}}$	$p_{\text{noninformative}}$	p
0.1	5	45	50
0.01	5	495	500
0.001	5	4995	5000

Table 5.2: Overview of the number of informative, noninformative and total features based on the informative ratios.

where $X = (X_1, X_2, \ldots, X_p)$ denotes the vector of p features, σ represents the noise amplitude, and $\varepsilon \sim \mathcal{N}(0,1)$ is a standard Gaussian noise term independent of X.

In Chapter 2 the random feature matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$ was described, where each $x_{ij} \in \mathbb{R}$ corresponds to a realization of the random variable X_j for observation i. The values of these features are set as follows, they are sampled independently from a standard normal distribution:

$$x_{ij} \sim \mathcal{N}(0,1), \quad \text{for } i = 1, \ldots, n, \quad j = 1, \ldots, p.$$

As a result, all features are independently and identically distributed. Therefore, the features are uncorrelated with each other and with noise term ε.

Since we vary the parameters r and σ, a total of $3 \times 3 = 9$ distinct parameter combinations are considered. For each of these combinations, 50 datasets are generated using different random seeds from NumPy to ensure reproducibility. The number of simulations is fixed at 50 to ensure robustness. The random seed of the first simulation is set to 42, the second one to 43, and so on until 91 for the fiftieth simulation. This set up results in a total of $9 \times 50 = 450$ datasets that are generated for each of the linear and non-linear data settings. More details on the simulations design are provided in Chapter 6.

It is important to note that only the noise amplitude σ influences the generation of the target vector $\mathbf{Y} \in \mathbb{R}^n$, where each entry $y_i \in \mathbb{R}$ denotes the target value corresponding to observation i. In contrast, the informative ratio r does not affect the generation of the target values. Instead, it determines the total number of features p by controlling the number of noninformative features added to the dataset used for training the models. Consequently, it affects the size of the random feature matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$.

5.2. Data Generating Functions

We define two different functions for function f of Equation (5.1) to generate the values of a target variable. For the linear datasets, we want to generate data with an underlying linear pattern. Thus we set the function f as linear:

$$f(X) = \sum_{j=1}^{p} \theta_j X_j = X\theta, \tag{5.2}$$

where $\theta = (\theta_1, \theta_2, \ldots, \theta_p)^\top \in \mathbb{R}^p$ is the true coefficient vector. The coefficients of θ determine which features are set as truly informative. For a clear distinction between the coefficients for the informative and noninformative features, we set the values of θ as follows. Let $p_{\text{informative}} \leq p$ denote the number of informative features. Then

$$\theta_j = \begin{cases} 1, & \text{if } 1 \leq j \leq p_{\text{informative}}, \\ 0, & \text{if } p_{\text{informative}} < j \leq p. \end{cases}$$

That is, the first $p_{\text{informative}} = 5$ features have nonzero coefficients, and the remaining features are noninformative with coefficients zero. In the linear case, we can summarize the process for the full dataset using matrix notation. The generated target vector $\mathbf{Y} \in \mathbb{R}^n$ is given by

$$\mathbf{Y} = \mathbf{X}\theta + \sigma\boldsymbol{\varepsilon},$$

where each entry $y_i \in \mathbb{R}$ denotes the target value corresponding to observation i and $\boldsymbol{\varepsilon} = (\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n)^\top \in \mathbb{R}^n$ with $\varepsilon_i \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0,1)$.

For the non-linear datasets, we set the function f to apply non-linear transformations to the measurable variables. Therefore, we arbitrarily combine various non-linear transformations and add an interaction term as follows:

$$f(X) = \sqrt{|X_1|} + e^{X_2} + \sin(X_3) + (X_4)^2 + \log_2(|X_5| + 1) + (X_1 X_2). \tag{5.3}$$

These transformations represent a diverse range of non-linear behaviours, including square root, exponential, periodic, quadratic, and logarithmic transformations, as well as an interaction term, mimicking complex relationships. The remaining $p_{\text{noninformative}}$ features, though present in the dataset, do not influence the true target variable. This ensures that the noninformative features serve purely as noise.

To ensure that both training and test sets are generated from the same distribution, we apply the described process above to first generate datasets for each combination of variations with a total of 200 observations. This full dataset is then split into a training set and a test set, each containing $n = n_{\text{test}} = 100$ observations. This approach guarantees that the training and test data follow the same distribution and generation mechanism, while also maintaining independence between the sets.

6

Simulation Design

This chapter presents a simulation study aimed at evaluating the performance of three regression models in the context of high-dimensional data. Specifically, random forests, Ridge regression, and Lasso regression are compared under varying conditions of informative feature ratios and noise amplitudes. Synthetic datasets are generated systematically by altering the proportion of informative features and the noise amplitude, while keeping the number of observations fixed. The noise amplitude influences how the informative features generate the target values. The informative feature ratio determines the total number of features by adding non-informative features to the dataset used for model training.

The goal is to assess how well these models predict continuous target variables and how effectively they identify informative features in challenging high-dimensional scenarios where the number of features exceeds the number of observations, or where the number of informative features is small. This chapter will discuss the experimental setup. Results from this study will provide insights into the strengths and limitations of these models in high-dimensional settings.

6.1. General Simulation Setup

The simulations investigate the performance of random forests for regression, Ridge regression, and Lasso regression in high-dimensional data. The simulations are run in `Python` version 3.10.12. The code used for running the simulations is made available on GitHub[1]. An overview of the simulation setup is provided in Figure 6.1. The synthetic linear and non-linear datasets are generated as described in Chapter 5. Table 5.1 and Table 5.2, provide an overview of the different settings used for the datasets. Since for each x_i in the training set, the corresponding y_i is known, this is a supervised learning task.

In the simulations, the three models are trained on the training set using 5-fold cross-validation combined with a grid search, implemented via `GridSearchCV`. The hyperparameters used in the grid search will be discussed in the following two sections. After training, the final model for each simulation is used to make predictions on the test set. The test performance is evaluated using the mean squared error (MSE) described in Chapter 2. In addition to MSE, we also assess feature selection performance. However, standard precision and recall are not ideal in this context due to differences in how models indicate feature relevance.

For Lasso, feature selection is straightforward as features with non-zero coefficients can be considered as selected. For Ridge, a hard threshold can be used, where coefficients below this threshold are treated as unimportant as they are close to zero. For Random Forests, feature importance values can be used, and a similar hard threshold can be applied. However, this approach showed some inconsistencies. In certain simulations, Ridge assigned very small coefficients to all features, which led to the incorrect conclusion that no features were important, even though the five informative ones still had the highest relative influence. In other cases, many features had similar magnitudes, leading to overestimation of selected features when using the hard threshold. Different methods for setting the hard threshold in Ridge were tried, but none yielded reliable improvements for determining how many informative features were identified by the model. Similar issues were observed with Random Forests due to its different mechanism for estimating feature relevance.

Given these model-specific behaviours, it was concluded that directly comparing the precision and recall, as described in Equation (2.4) and Equation (2.5), respectively, across models was not entirely fair. It is an unfair comparison as the metrics directly depend on the set threshold which is chosen manually. Therefore,

[1] `https://github.com/ifels266/statistical-learning`

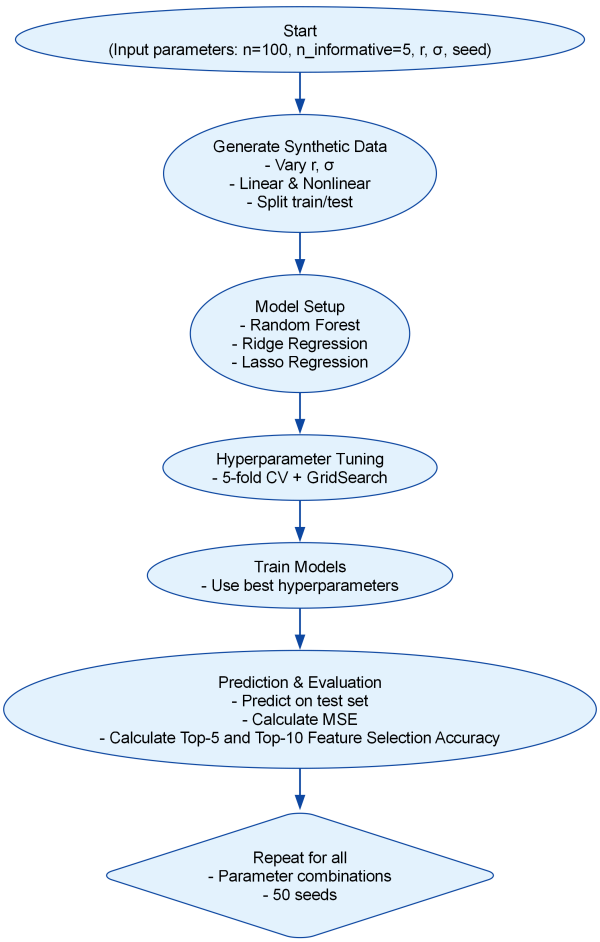


Figure 6.1: Diagram of the Simulation Design

we introduce two metrics that evaluate how well the model ranks the truly informative features among the most relevant ones:

- **Top-5 Feature Selection Accuracy** (A_{Top5}): The number of truly informative features among the top 5 features selected by the model, divided by 5.
- **Top-10 Feature Selection Accuracy** (A_{Top10}): The number of truly informative features among the top 10 features selected by the model, divided by 5.

Formally, let S_k denote the set of top-k most relevant features selected by the model, based on the absolute value of coefficients (for Lasso and Ridge) or feature importance (for Random Forests). Let I denote the set of truly informative features. Then:

$$A_{\text{Top}k} = \frac{|S_k \cap I|}{|I|}, \quad \text{with } |I| = 5 \text{ and } k \in \{5, 10\}. \tag{6.1}$$

A perfect score is obtained when $A_{\text{Top}k} = 1$ and the lowest score is obtained when $A_{\text{Top}k} = 0$. These two metrics mirror the intention behind traditional precision and recall. The Top-5 metric serves as a stricter version (similar to precision), while the Top-10 metric offers a more lenient evaluation (similar to recall). The lenient metric uses the top 10 features to account for the varying feature dimensions introduced by different informative feature ratios ($r \in \{0.1, 0.01, 0.001\}$), corresponding to total feature counts of 50, 500, and 5000. Both metrics provide a fair and reliable model comparison, but they are not suitable for comparing performance across different informative feature ratios. This is because the likelihood of correctly identifying the informative features purely by chance varies drastically depending on the total number of features. For example, when $r = 0.1$ and therefore $p = 50$, the five informative features represent 10% of the total features. In contrast, when $r = 0.001$ and therefore $p = 5000$, they account for only 0.1%. As a result, these metrics should not be used to compare feature selection performance across different values of r. They are best interpreted with a fixed informative feature ratio to assess relative feature selection performance between models.

For each simulation of each parameter setting of the models, the results are saved to a separate CSV file. In addition, a summary file is created for each parameter setting, containing results such as the mean MSE over the 50 simulations, standard deviation, and other relevant statistics.

6.2. Model Setup

Random forests are trained using the hyperparameters specified in Table 6.1. These hyperparameter values were chosen to cover a wide range of settings commonly used in practice, ensuring that each model could be properly tuned. The decision trees within the random forest are built using the method described in Chapter 4, as this is the default setting in `scikit-learn`.

In random forests, feature importances are extracted from the trained model and normalized so that they sum to one. The features corresponding to the five largest values of the feature importances are set as the informative features of the model.

Hyperparameter	**Definition**	**Values**
`n_estimators`	The number of trees in the forest	100, 200, 300, 400, 500
`max_depth`	The maximum depth of each decision tree. If `None`, nodes are expanded until all leaves are pure	None, 10, 20
`min_samples_split`	The minimum number of observations required to split a node	2, 5
`min_samples_leaf`	The minimum number of observations required to be at a leaf node	1, 2, 4
`max_features`	The number of features to consider when looking for the best split	$\sqrt{p}$, $\log_2(p)$, 1, $\lfloor p/3 \rfloor$

Table 6.1: Random Forest hyperparameter grid and corresponding definitions. For other hyperparameters, the default values of `scikit-learn` version 1.6.1 are used.

Ridge and Lasso regression models use the hyperparameters specified in Table 6.2. The range of `alpha` $= \lambda$ values was chosen to cover very weak to very strong regularization, allowing the models to adapt well across different settings. Although the features were generated from a standard normal distribution, we applied `StandardScaler` from `scikit-learn` to ensure exact centering and scaling of each feature before training Ridge and Lasso models. Lastly, the features corresponding to the coefficients of the five largest magnitudes are set as the informative features of the model.

The `random_state` parameter, which controls the randomness in the algorithms to ensure reproducibility of results, was set to 42 for all three models prior to training. For Lasso regression, the maximum number of iterations (`max_iter`), which determines how many times the algorithm updates its estimates during optimization, was increased to 10,000 to ensure convergence in high-dimensional settings where the optimization problem can be more challenging. This higher limit was chosen because lower values resulted in convergence warnings from `scikit-learn`. All other parameters that were not explicitly tuned were left at their default values in `scikit-learn` version 1.6.1, including settings such as convergence tolerance and solver algorithms.

Hyperparameter	**Definition**	**Values**
`alpha`	Regularization tuning parameter controlling the strength of the shrinkage penalty. In statistical notation, this corresponds to λ; `scikit-learn` uses the name `alpha`.	The thirty values are logarithmically spaced from 0.0001 to 10,000

Table 6.2: Hyperparameter grid for Ridge and Lasso regression. For other hyperparameters, the default values of `scikit-learn` version 1.6.1 are used.

6.3. Computational Efficiency Measurement

In addition to prediction accuracy and feature selection, the computational efficiency of each model is a critical factor. During the simulations, the duration of the grid search with cross-validation was measured. Each model was tuned using a hyperparameter grid of different size, which directly influenced training time. For clarity, we define training time as the total duration of the training and validation process during the grid search. Specifically, for Ridge and Lasso, the hyperparameter grid consists of 30 values of the regularization parameter α, and 5-fold cross-validation is applied, resulting in $30 \times 5 = 150$ model fits. For Random Forest, the grid is significantly larger, with $5 \times 3 \times 2 \times 3 \times 4 = 360$ parameter combinations and 5 folds, yielding $360 \times 5 = 1800$

fits. Table 6.3 summarizes the number of hyperparameter settings and total model fits per cross-validation for each model. Note that even if the training time per fit would be equivalent across the three models, random forests would take 12 times longer due to the size of the hyperparameter grid. The time it takes to make a prediction after the models have been trained, was not measured in this study.

During the simulations, computations within the grid search were parallelized using 124 CPU cores. While hyperparameter tuning was parallelized across different parameter combinations, additional parallelization is possible within the Random Forest model itself, specifically in the construction of individual trees. Using fewer CPU cores would increase the relative training time of Random Forest compared to Ridge and Lasso, making the difference in computational efficiency more pronounced.

Model	**Number of Hyperparameter Settings**	**Total Fits (Settings × 5 folds)**
Ridge Regression	30	150
Lasso Regression	30	150
Random Forest Regression	360	1800

Table 6.3: Hyperparameter grid sizes and total number of model fits during grid search with 5-fold cross-validation.

7

Results: High-Dimensional Linear Data

In this chapter, we present the results of the linear case of the simulation study from Chapter 6. The performance measures investigated are MSE, stated in Equation (2.3), and two metrics for feature selection accuracy, discussed in Chapter 6. Remember that these metrics are only fairly comparable within each informative ratio setting. The computational efficiency of the models will also be discussed. Since the underlying pattern of the data is linear, it is expected that the linear models Lasso and Ridge perform better than the non-linear model random forests. As the results for noise amplitude $\sigma = 0.01$ and $\sigma = 0.1$ are extremely similar, the results for $\sigma = 0.01$ are omitted in this chapter, but are included in Appendix B.1.

7.1. Prediction Error

First, we look at the prediction error measured with the MSE. Figure 7.1 shows the boxplots of the MSE results from simulations with noise amplitude $\sigma = 0.01$. It is immediately clear that Lasso outperforms random forests in all three scenarios, which is in line with the expectations. For the lowest two informative ratios, Ridge performs worse than random forests which is unexpected. As both Ridge and random forests struggle with feature selection, it aligns with expectations that their MSE decreases as the informative ratio increases. However, the MSE of Ridge increases significantly for the lower informative ratios.

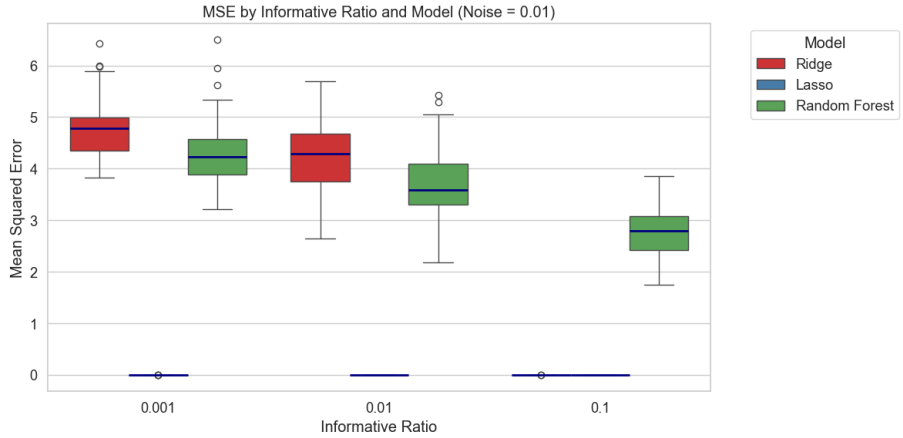


Figure 7.1: Mean squared error (MSE) boxplots of the 50 simulations for the three models in the linear case with noise amplitude set to 0.01. For each informative ratio, the left boxplot corresponds to Ridge, the middle to Lasso, and the right to random forests.

Using Table 5.2, we know that for informative feature ratios 0.001, 0.01 and 0.1, there are 4995, 495, 45 non-informative features. Even though Ridge applies a shrinkage penalty, the coefficients of the noninformative features are not equal to zero and influence the predicted outcome $\hat{Y}$. Especially when the informative ratio is low, this results in a lot of noise, increasing the MSE. This could explain why Ridge performs significantly better for highest informative feature ratio compared to the lower informative ratios. Another reason why the performance of Ridge could be bad, is that the hyperparameter grid is not large enough. Inspection showed that for the lowest informative ratio, the largest λ of 10,000 was often chosen. As the thirty values are logarithmically spaced, the second largest value was around 5,000. This is probably the cause for Ridges bad performance and

should be taken into account when drawing conclusions. To investigate if this plays a significant role in the prediction error of Ridge, further researched should be done.

Although Random Forest is not a linear model, the absolute values of the MSE are not drastically large values. This is due to the hyperparameter grid for Random Forest, shown in Table 6.1. As the grid is extensive, it offers high model flexibility, resulting in reasonable MSE values.

To better visualize small differences in Lasso's near-zero MSE values, we use a logarithmic scale in Figure 7.2. The advantage of feature selection in Lasso can explain why even though Ridge performs very well when the informative ratio is largest, Lasso still performs better. The performance of Lasso seems to improve when the informative ratio increases. To investigate this phenomenon further, we will take a look at the feature selection accuracies in the next section, which might be related to the MSE results.

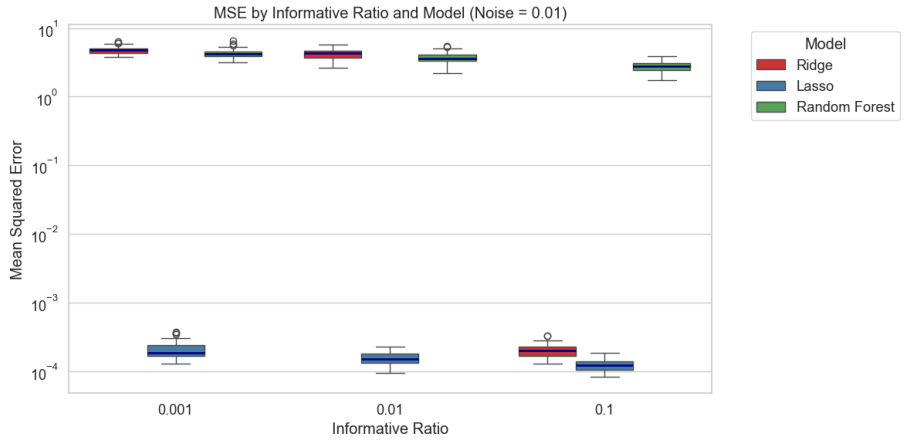


Figure 7.2: Mean squared error (MSE) boxplots of the 50 simulations for the three models in the linear case with noise amplitude set to 0.01.

When the noise amplitude σ is increased to 1, we obtain Figure 7.3. The drop in prediction performance shown by the increase of the MSE is expected, because added noise makes it harder to predict the true target values. Appendix B.2 provides detailed visualisations of how the MSE changes with the noise amplitudes for each model and setting. Relatively, we see similar results when comparing the three models across the different informative ratios. This can indicate that relative performance of the models is primarily driven by the sparsity of the informative features, rather than the noise level.

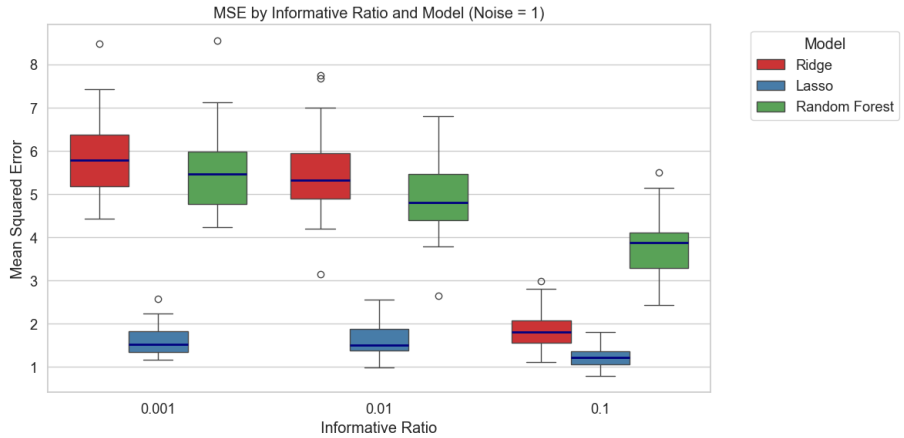


Figure 7.3: Mean squared error (MSE) boxplots of the 50 simulations for the three models in the linear case with noise amplitude set to 1.

7.2. Feature Selection Accuracy

The feature selection accuracy is measured with the metrics A_{Top5} and A_{Top10} as defined in Equation (6.1).

Accuracy Top 5

The boxplots of simulation results for A_{Top5} with noise amplitude $\sigma = 0.01$ can be found in Figure 7.4. Lasso

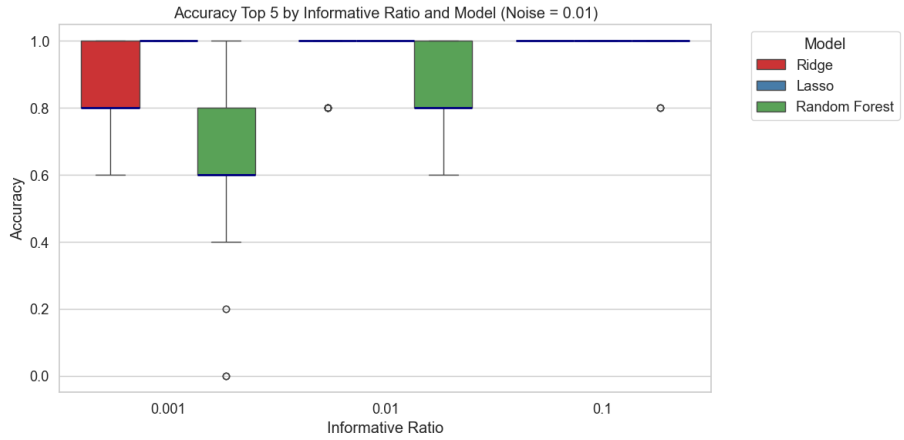


Figure 7.4: Accuracy Top 5 of the 50 simulations per model in the linear case with noise amplitude set to 0.01. For each informative ratio, the left boxplot corresponds to Ridge, the middle to Lasso, and the right to random forests.

achieves a perfect $A_{\text{Top5}} = 1$ score in every simulation and informative ratio, consistently identifying all five truly informative features. This highlights the strength of its built-in feature selection. Ridge also performs well, particularly at the higher informative ratios, but occasionally misses one feature. Random forests show more variability in the feature selection accuracy, especially at low informative ratios, confirming their difficulty in distinguishing truly informative features.

Interestingly, although Ridge often ranks the informative features more accurately than random forests, this advantage does not consistently translate into better MSE performance. This might be another sign that indeed the hyperparameter grid was too small for Ridge. Since λ, after tuning, was often set very large in the settings with informative ratio 0.001, all coefficients are pushed very close to zero. Therefore, while the relative ranking remains correct, the actual contribution of the informative features in the prediction becomes minimal, reducing overall accuracy.

Lasso may have obtained a perfect top 5 ranking of the informative features, but did not have a perfect MSE score, although it came very close. This can be caused by the irreducible error, or by the fact that Lasso assigns a small value close to zero to the coefficients of some noninformative features. In that case, some noninformative features still affect the prediction of the target value. To investigate the second matter further, the relative importance of the informative features will be discussed in Section 7.4.

Figure 7.5 shows the results for the increased noise amplitude of 1. Lasso's feature selection accuracy remains unaffected by the increased noise, consistently identifying all five informative features across all settings. This shows that Lasso is robust regarding feature selection in this case. In contrast, both Ridge and random forests show a drop in accuracy, typically missing one additional feature compared to the low-noise setting. For Ridge, this decline is likely due to the strong regularization shrinking all coefficients, including those of informative features. As a result, the coefficients of the informative features are so small that some no

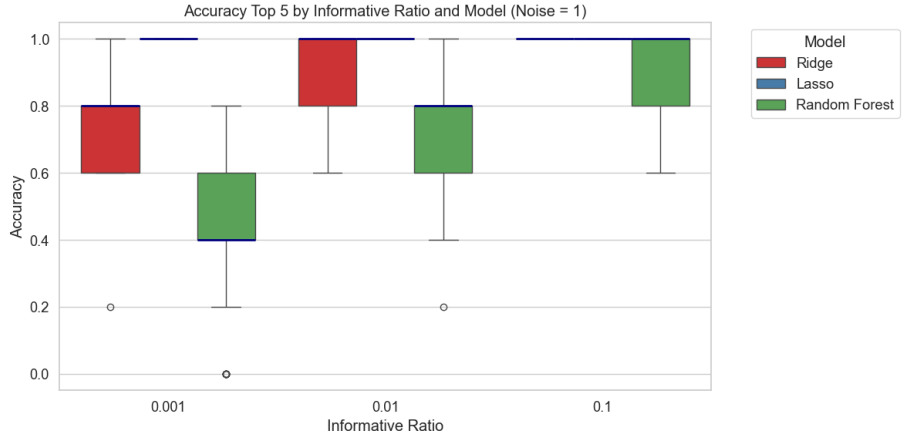


Figure 7.5: Accuracy Top 5 of the 50 simulations per model in the linear case with noise amplitude set to 1. For each informative ratio, the left boxplot corresponds to Ridge, the middle to Lasso, and the right to random forests.

longer rank among the top five due to the increased noise, despite having slightly lower absolute values than the five largest coefficients. This will be further researched in Section 7.4.

Accuracy Top 10

The Accuracy Top 10 metric is a more lenient evaluation of feature selection accuracy, as a perfect score is achieved when all five truly informative features are present in the ten most important features selected by the model. Figure 7.6 shows the boxplots of simulation results for A_{Top10} with noise amplitude $\sigma = 0.01$. Lasso already had the perfect score in the top 5, so naturally obtains the perfect score in the top 10 as well. Ridge shows a significant improvement compared to the strict metric. This implies that even in settings with sparse informative features, it assigns the informative features a larger coefficients than most noninformative features. Interestingly, the accuracy of random forests have not increased with the more lenient metric. This again highlights random forests' difficulty in identifying the true informative features.

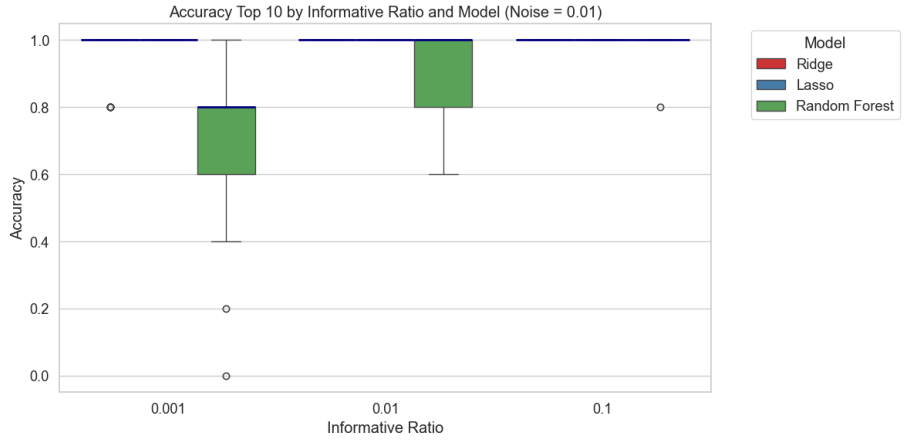


Figure 7.6: Accuracy Top 10 of the 50 simulations per model in the linear case with noise amplitude set to 0.01. For each informative ratio, the left boxplot corresponds to Ridge, the middle to Lasso, and the right to random forests.

Figure 7.7 includes the results for the higher noise amplitude $\sigma = 1$. Similarly, the results of Ridge have improved compared to the strict metric. A notable difference compared to the results with lower noise amplitude, is that accuracy of random forests have also increased in the lenient metric for the two higher informative ratios.

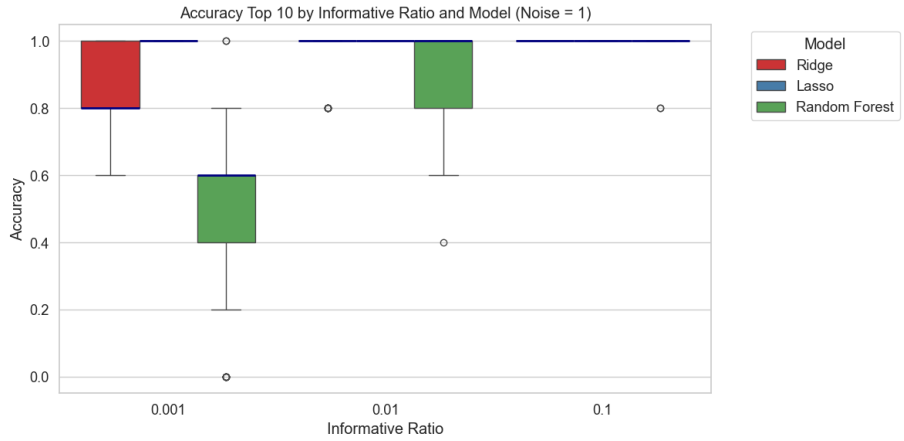


Figure 7.7: Accuracy Top 10 of the 50 simulations per model in the linear case with noise amplitude set to 1. For each informative ratio, the left boxplot corresponds to Ridge, the middle to Lasso, and the right to random forests.

7.3. Computational Efficiency

Figure 7.8 shows the training times for noise amplitude $\sigma = 0.01$. Since training times are not directly affected by the noise level, results for the other noise amplitudes are included in Appendix B.3. It is clear that random forests have by far the highest training times, especially when the dataset size is large. This is expected, as

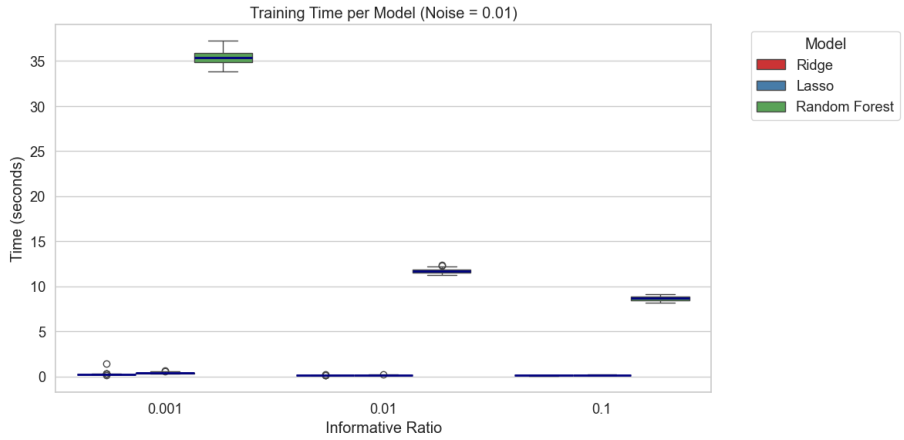


Figure 7.8: Training time for each model under low noise amplitude ($\sigma = 0.01$), across different informative feature ratios. For each informative ratio, the left boxplot corresponds to Ridge, the middle to Lasso, and the right to random forests.

random forests must build a large number of decision trees and has a larger hyperparameter grid in this study. The higher training times for lower informative feature ratios can be explained by the increased number of features. When there are more features, the model evaluates more potential split candidates at each set in the tree, increasing computational cost. Figure 7.9 provides a clearer view of the differences in training time between Lasso and Ridge by using a logarithmic scale. Ridge is generally faster than Lasso, particularly when the dataset size is large, which can be seen in the results. This observation aligns with findings in the literature as Lasso uses a more computationally intensive optimization due to the non-differentiable ℓ_1 penalty.

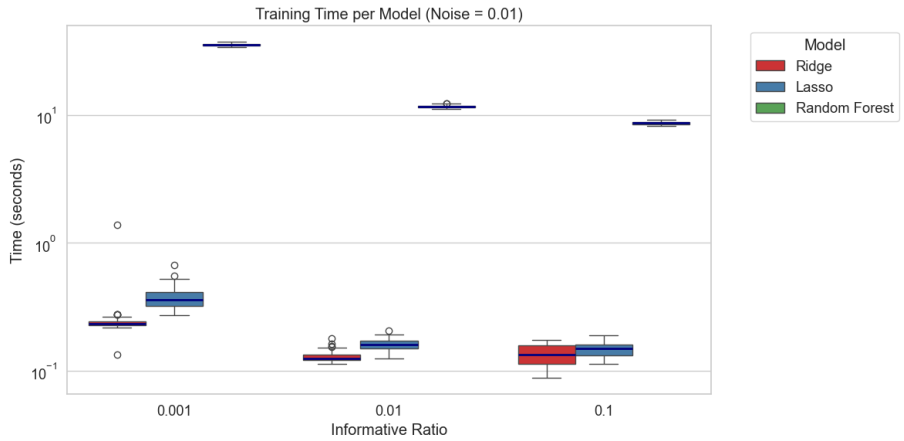


Figure 7.9: Training time for each model under low noise amplitude ($\sigma = 0.01$), across different informative feature ratios. For each informative ratio, the left boxplot corresponds to Ridge, the middle to Lasso, and the right to random forests.

7.4. Relative Importance of the Informative Features

With the current set up for the linear case, each informative feature contributes equally to the target value, and the noninformative features do not contribute. To obtain the relative importance of the features, we can normalize the coefficients of Ridge and Lasso. Random forests have a built-in relative feature importance that is already normalized. A relative importance equal to 1 indicates that the prediction is determined solely by that feature. When the relative importance is zero, that feature has no contribution to the prediction. In the optimal results, as there are five informative features with equal importance, the relative importance should be 0.2 for the informative features. The noninformative features should not contribute to the prediction, thus the relative importance of these features should be equal to zero. This method enables us to compare the relative importance between the informative features and provides information about the relative importance that is assigned to noninformative features.

The results of the relative importance, presented in Appendix D.1, reveal several patterns. In all settings, the models correctly assign similar importance levels to the informative features. This indicates that the models

correctly capture that the informative features contribute equally. Lasso generally assigns the five informative features a correct relative importance. Only when the noise amplitude is very high, the relative importance becomes notably lower than 0.2. Further inspection of the dataset showed that in all but one simulation, Lasso assigned at least one coefficient of a noninformative feature to a nonzero value. Although these coefficients were close to zero, they still contributed to the prediction, which explains the nonzero MSE scores.

As the informative ratio decreases, and therefore the number of noninformative features increases, Ridge and random forests assign a significantly lower relative importance to the informative features than Lasso. Since the relative importances sum to 1, this means that more noninformative features were mistakenly considered informative. For Ridge, the poor prediction performance at the two highest informative ratios can be explained because the informative features have an extremely low relative importance. Even though Ridge often recognized the informative features in the top 5 or top 10 rankings, their actual contribution to the prediction is limited. Both Ridge and random forests perform slightly worse when the noise amplitude increases.

To investigate if the models can also recover the correct ordering among the five informative features, a small study was conducted. The noise amplitude was set to 0.01, and the true coefficients of the informative features were changed to 1, -2, 3, -4, and 5. As a result, the first informative feature is the least relevant to the prediction, and the fifth the most. As before, the noninformative features did not contribute. In the optimal results, an equivalent ordering of the relative importance of the informative features should be obtained, and the noninformative features should have a relative importance of zero. The results are presented in Appendix D.2. Lasso accurately recovers the order of importance and performs consistently well across all informative ratios. Ridge's performance is sensitive to the informative ratio, and only performs well at the lowest informative ratio. At the higher informative ratios, it assigns an extremely low relative importance to the informative features. This is likely due to the poor regularization, as previously discussed. Random forests assign a lower relative importance to the informative features than Lasso, but seem to capture the order correctly. However, random forests have more difficulty with identifying the correct order as the informative ratios decreases, where the relative importance of some informative features is extremely small.

8

Results: High-Dimensional Non-Linear Data

This chapter presents the results for the non-linear case of the simulation study described in Chapter 6. The performance measures investigated are MSE, stated in Equation (2.3), and two metrics for feature selection accuracy, discussed in Chapter 6. Note that the second two metrics are only fairly comparable within each informative ratio setting. The computational efficiency of the models will also be discussed. Since the underlying pattern of the data is non-linear, it is expected that the the non-linear model random forests will perform better than the penalized linear models. However, as the number of informative features is sparse, this might not be the case. The results for noise amplitude $\sigma = 0.01$ and $\sigma = 0.1$ are found to be extremely similar, therefore, the results for $\sigma = 0.01$ are omitted in this chapter, but are included in Appendix C.1.

8.1. Prediction Error

An overview of the simulation results with metric MSE and noise amplitude 0.01 is provided in Figure 8.1. While there was a clear difference in terms of performance measured with the MSE of the models in the linear setting, the performances do not differ that much to each other in the non-linear case. In fact, even though the underlying pattern is linear, Lasso slightly outperforms random forests for informative ratio 0.001 and is extremely competitive for informative ratio 0.01. As stated in the literacy research, this is probably due to the fact that random forests uses feature sampling: when there are many noninformative features, these features are more often split on, resulting in meaningless splits. Similar to the linear case, is that there is often strong regularization in Ridges models after tuning λ. In that case, the largest values for λ are selected and this happens when the informative ratio is 0.001 or 0.01, which influences the prediction error strongly. Further research should indicate how much the prediction error could be decreased by extending the hyperparameter grid for Ridge.

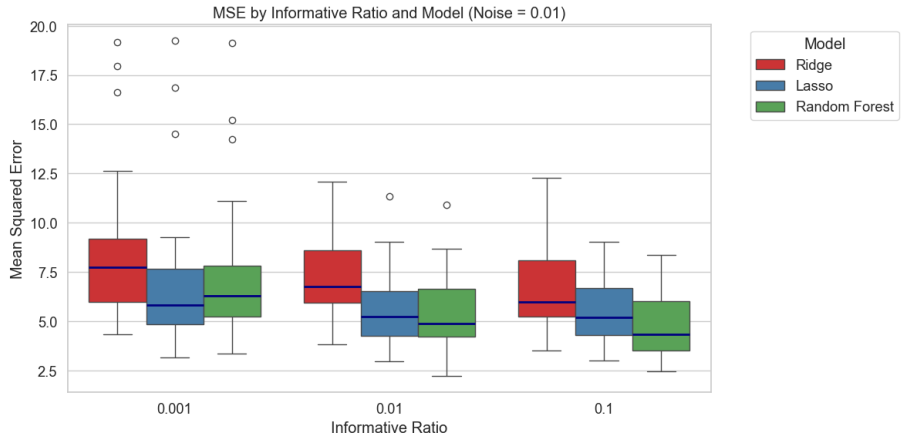


Figure 8.1: Mean squared error (MSE) boxplots of the 50 simulations for the three models in the nonlinear case with noise amplitude set to 0.01.

When noise amplitude is increased to 1 we obtain the results from Figure 8.2. The increase in noise amplitude to 1 results in minimal changes in performance across models and the relative performance ranking remains the same. Appendix C.2 provides clear visualizations of how the MSE changes when varying the noise amplitude for each model over all settings.

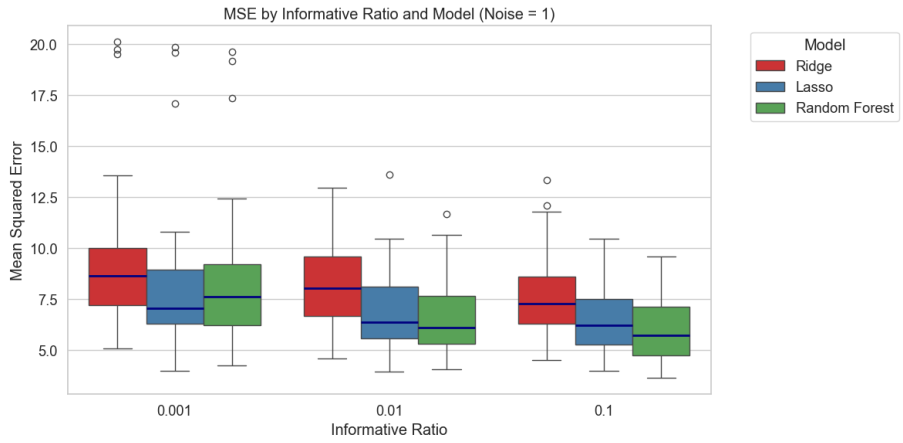


Figure 8.2: Mean squared error (MSE) boxplots of the 50 simulations for the three models in the nonlinear case with noise amplitude set to 1.

8.2. Feature Selection Accuracy

The feature selection accuracy is measured with the metrics A_{Top5} and A_{Top10} as defined in Equation (6.1).

Accuracy Top 5

The simulation results for the top 5 feature selection accuracy is shown in Figure 8.3. While in the linear case Ridge and Lasso outperformed random forests in terms of feature selection accuracy, the roles are now reversed. Random forests shows a noticeable higher accuracy than the linear models for the two highest informative ratios. This suggests that random forests is better at recognizing the most relevant features in these settings for the non-linear case. One possible reason why Ridge and Lasso perform worse in terms of feature selection is that feature importance in these models is measured through coefficient magnitudes. However, in the case of non-linear interactions, such as the interaction term $(X_1 X_2)$ in function f from Equation (5.3), no single coefficient directly captures the effect, making it harder for linear models to identify the contributing features. In fact, Lasso may even assign a non-zero coefficient to a noninformative feature in an attempt to approximate such interaction effects, potentially leading to misleading feature selection.

A possible explanation for the lower accuracy of random forests in terms of prediction error, is that while random forests is better at assigning a higher importance to the correct informative features, it tends to use many features when making predictions due to feature selection. Especially when the number of informative

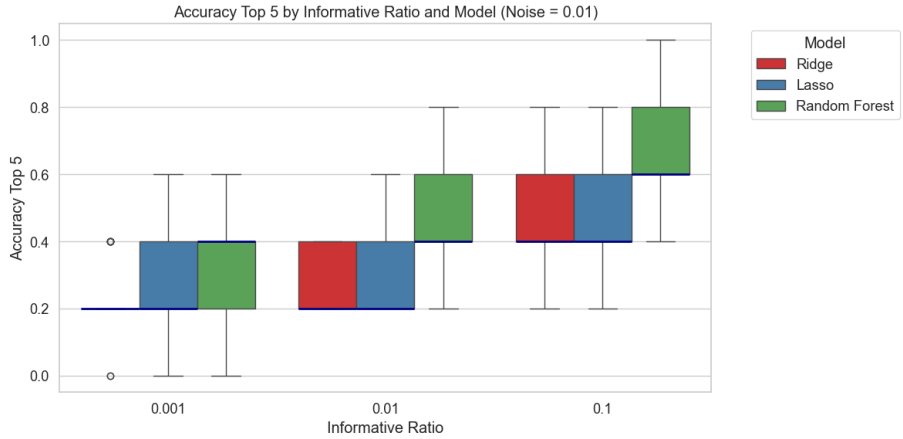


Figure 8.3: Accuracy Top 5 boxplots of the 50 simulations per model in the nonlinear case with noise amplitude set to 0.01.

features is sparse, this results in a large prediction error. In contrast, Lasso creates a sparse, linear model where each selected feature has a more direct effect on the prediction, determined by the corresponding coefficients values. This targeted use of features allows Lasso to achieve prediction performance on a similar level to that of random forests, even when its feature selection accuracy is slightly lower.

Increasing the noise amplitude to 1, results in Figure 8.4. The accuracy of Ridge, Lasso and random forests remained similar or slightly decreased with the higher noise amplitude. This indicates that all three models maintain a relatively robust ability to identify the most informative features even under higher noise conditions, though random forests continues to show superior selection accuracy in more informative settings.

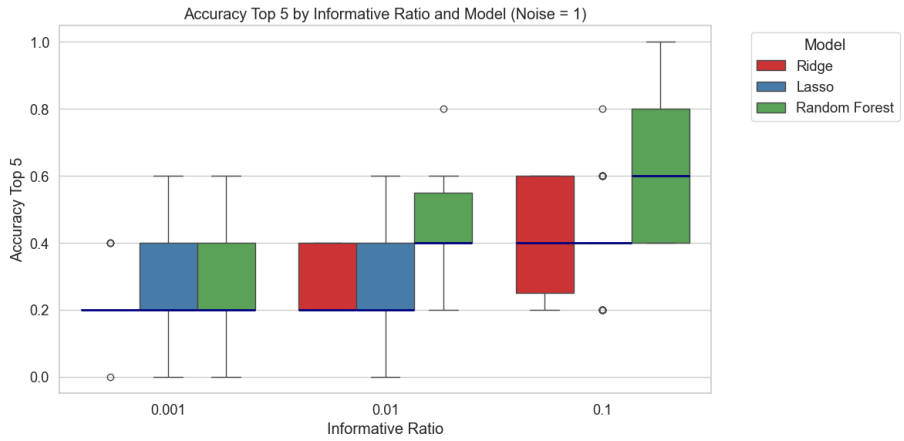


Figure 8.4: Accuracy Top 5 boxplots of the 50 simulations per model in the nonlinear case with noise amplitude set to 1.

Accuracy Top 10

Figure 8.5 shows the simulation results for the top 10 feature selection accuracy. The accuracy scores of Ridge and Lasso with the lenient metric remained similar or are slightly higher than the top 5 results. Random forests maintains a clear advantage in recognizing the informative features in general and shows the most improvement with the lenient metric. The limited improvement for Lasso and Ridge indicates that these models are still less capable of recognizing the informative features. Note that it was the other way around in the linear case. Meanwhile, the growing advantage of random forests in feature selection, is also represented in terms of the MSE results, where in the setting with the highest informative ratio random forests started to improve compared to Lasso.

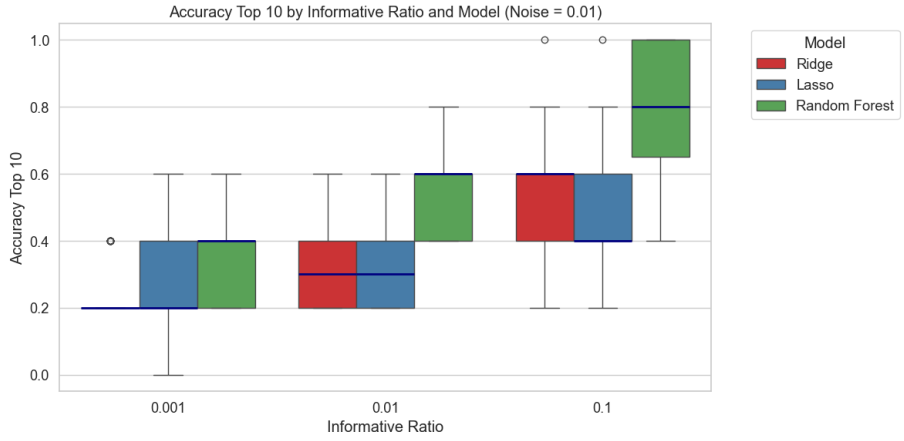


Figure 8.5: Accuracy Top 10 boxplots of the 50 simulations per model in the non-linear case with noise amplitude set to 0.01.

Figure 8.6 visualises the results with the increased noise amplitude. The accuracy scores of the top 10 rankings remain similar or have slightly decreased compared to the setting with the lower noise amplitude. In the analyses with the lower noise amplitude, we found that the difference between the strict and lenient metrics was small. With the high noise amplitude, we find that the difference between the lenient and strict metrics increased somewhat. This is expected, as it is more difficult for the models to directly determine the

informative features when there is more noise. Therefore, the models might place some informative features just outside the top 5 ranking.

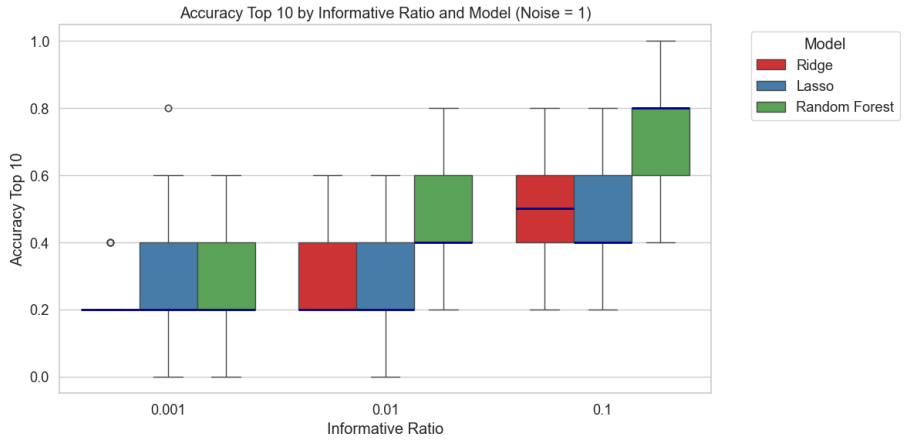


Figure 8.6: Accuracy Top 10 boxplots of the 50 simulations per model in the non-linear case with noise amplitude set to 1.

8.3. Computational Efficiency

To evaluate the computational efficiency of the models in the non-linear setting, we analyse the training time per model. As the noise amplitude does not directly influence the training time, this chapter only discusses the results for noise amplitude 0.01. The results for the training times when the noise amplitudes are 0.1 and 1 are provided in Appendix C.3. Figure 8.7 presents the corresponding times for noise amplitude $\sigma = 0.01$. The times have not changed significantly compared to the training times in the linear case. However, for the lowest informative ratio, there seems to be a decrease of five seconds in the training time for random forests.

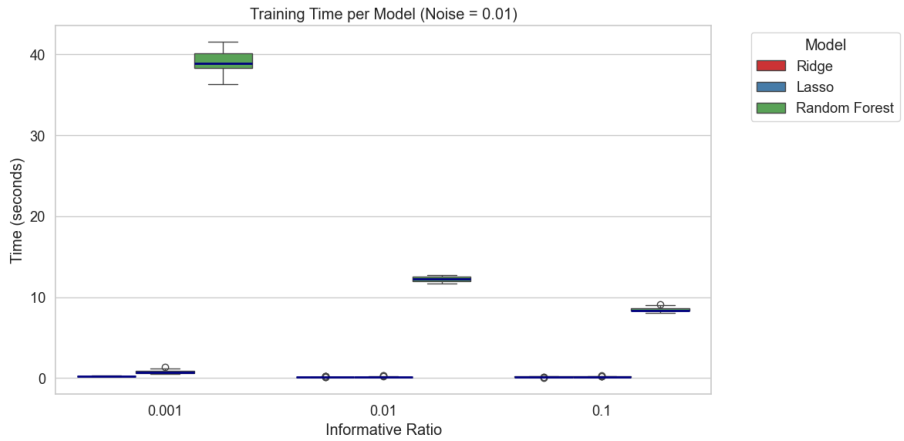


Figure 8.7: Training time for each model under low noise amplitude ($\sigma = 0.01$), across different informative feature ratios. For each informative ratio, the left boxplot corresponds to Ridge, the middle to Lasso, and the right to random forests.

To compare the training times of Ridge and Lasso, we use Figure 8.8. The behaviour of Ridge and Lasso is similar as in the linear case. For the lowest informative feature, as with random forests, there seems to be a slight decrease relatively of the training time. The increase in training times could be explained by the complexity of the data and the difficulty of the models to capture the true patterns. That the difficulty has increased is reflected by the increased MSE values for all models compared to the linear case.

8.4. Other Non-Linear Functions

In this chapter, all the results were based on one specific non-linear function that created the underlying non-linear pattern in the data. To show that the choice of the non-linear function directly influences the results, a small study was conducted. The same hyperparameter grids were used as in the main study. However, due to time limitations, we only investigate the informative ratio $r = 0.01$ and noise amplitude $\sigma = 0.01$. The

non-linear functions of this study that are used instead of function $f(X)$ from Equation (5.3) are stated in Appendix A.1. The mean MSE results, Top-5 Accuracy and Top-10 Accuracy of each function are included in Appendix A.2.

The results reveal notable variation depending on the chosen non-linear function. For some functions, there are large differences in prediction error between the models, while for others, the differences are minimal. Furthermore, no single model consistently performs best in terms of the prediction error. Additionally, the absolute values of the prediction error vary widely: some functions have a high MSE and some have a low MSE. Regarding feature selection accuracy, random forests consistently outperform the other models under both lenient and strict metrics. However, the magnitude of this advantage varies considerably. Similar to the prediction error, the accuracy scores themselves can fluctuate considerably depending on the function. Further research is needed to draw definitive conclusions, but it is clear that the choice of non-linear function, which defines the underlying relationship in the data, directly influences the results.

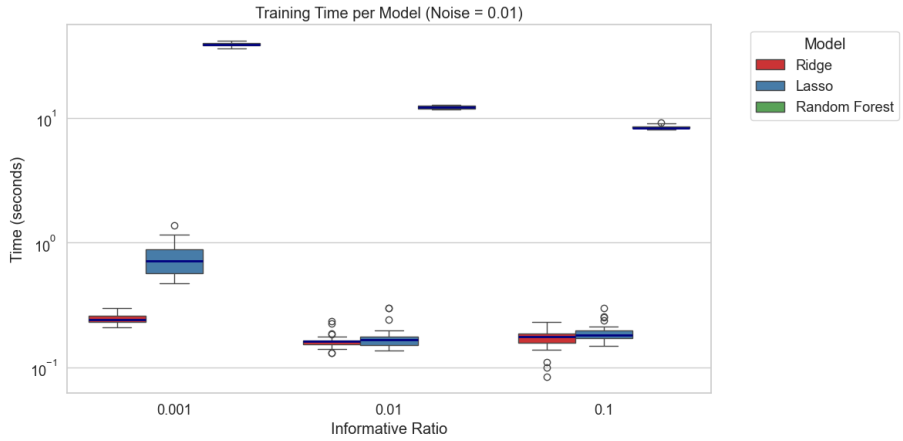


Figure 8.8: Training time for each model under noise amplitude ($\sigma = 0.01$), across different informative feature ratios. For each informative ratio, the left boxplot corresponds to Ridge, the middle to Lasso, and the right to random forests.

9

Conclusion and Discussion

This thesis investigated the comparative performance of random forests and the penalized linear regression methods, Ridge and Lasso regression, in high-dimensional settings characterized by sparsity in the number of informative features. The analysis focused on how varying the proportion of informative features and the noise amplitude influenced prediction accuracy and feature selection capability under both linear and non-linear data-generating scenarios. A comparison of the computational efficiency of the models was also included. It was expected that Ridge and Lasso regression would perform better in the linear settings due to their assumption of linearity. Conversely, in the non-linear scenarios, random forests were anticipated to have an advantage because of their ability to capture complex relationships. However, it was also considered that the strong feature selection capability of the penalized linear models might lead to competitive or even superior performance in some non-linear cases, especially when the number of informative features is sparse.

A Linear Case

In the linear case, where the true underlying pattern of the data is linear, the results showed that the linear model Lasso outperformed the non-linear model random forests in terms of both prediction accuracy and feature selection accuracy. Lasso consistently achieved near-zero MSE values and had perfect Top-5 and Top-10 accuracy score across all informative ratios and noise levels, confirming its strength in sparse high-dimensional settings. Ridge performed well in terms of feature selection accuracy, but its prediction performance deteriorated at the lowest informative ratios, likely due to the limited hyperparameter grid and the large number of noninformative features influencing the outcome. Random forests, while not drastically underperforming in terms of MSE, struggled to identify the informative features correctly, especially at low informative ratios. In addition, the relative importance analysis confirmed that Lasso is most capable of assigning the correct relative importance to informative features. In contrast to Ridge and random forests struggled with the correct relative importance of the informative features, especially when sparsity increased. The increase of the noise amplitude only had a minor impact, suggesting that the relative model performance is primarily determined by the sparsity of the data, rather than the noise level. Lastly, the linear models showed a lower training time.

A Non-Linear Case

In the non-linear case, the differences between the models were less pronounced than in the linear case. While random forests were expected to outperform the penalized linear models due to their ability to model complex relationships, this did not hold in settings with low informative ratios. When the proportion of informative features was low, Lasso was able to achieve similar or even slightly better prediction performance. This is likely caused by the sparsity of the data, which increases the likelihood of selecting noninformative features during splits in the random forests mechanism. The feature selection accuracy of random forests and Lasso was similar at the lowest informative ratio. However, at the other informative ratios, random forests consistently scored better than the linear models in identifying the informative features. This indicates that random forests are generally more effective at recognizing the informative features in non-linear settings. In contrast, the linear models rely on coefficient magnitude, which struggles to reflect non-linear dependencies like interaction effects. Consequently, their feature selection accuracy scores were lower than random forests. Nevertheless, Lasso maintained strong prediction performance by constructing sparse models that avoid overfitting to noninformative features. In addition, the linear models showed a lower training time. While Ridge had the ad-

vantage of having the lowest training time, it performed worst overall, which was likely caused by the limited hyperparameter grid. An additional study showed that the specific shape of the non-linear relationship has a substantial impact on how well each model performed.

Conclusion

From a practical perspective, when computational time is limited or fast decision-making is required, Lasso was often the preferred choice in the investigated high-dimensional settings. In a genomics context, for example, where each feature represents the expression level of a specific gene and the target is the likelihood of developing a certain disease, Lasso could quickly produce accurate predictions. This is particularly valuable when timely risk assessments are needed, such as in screening scenarios. However, when the primary goal is to identify which genes play a key role in disease mechanisms, for instance, distinguishing between genes that do and do not directly trigger the onset of a disease, random forests obtained slightly better results. Its ability to uncover complex, non-linear relationships made it better suited for capturing subtle but meaningful patterns in the data and assigning higher importance to the truly informative genes.

The results from both the linear and non-linear cases demonstrated that model performance depends strongly on the underlying data structure. Since the true data-generating process is usually unknown in practice, this highlights the importance of carefully considering the characteristics of the data when choosing a model. No single model consistently outperformed others across all settings. We found that penalized regression methods such as Lasso could outperform random forests even when the true pattern is non-linear, especially in situations where the number of informative features is very sparse. This challenges the common belief that more complex, non-linear models will always provide better results in non-linear problems. Overall, the choice of model should take into account not only prediction accuracy but also informative feature identification, computational efficiency, and the expected sparsity and complexity of the data.

Recommendations for Future Research

While synthetic data allows for controlled experimentation, the results may not fully capture the complexities of real-world datasets. It would therefore be valuable to apply the statistical learning methods discussed in this thesis to real-life data and examine the models' performances in practice. A limitation to note is that the custom metrics introduced for feature selection accuracy rely on prior knowledge of the truly informative features, which is typically unavailable in real-world scenarios.

Drawing conclusions in the non-linear setting should be approached with caution. There is no single, universally accepted function that generates non-linear data. To more thoroughly evaluate model performance in non-linear settings, future work could explore a variety of synthetic non-linear functions with different characteristics. Additionally, functions that combine linear and non-linear components could be considered.

In this thesis, the features were sampled from a standard normal distribution. Future research could investigate how alternative feature distributions affect model performance. To investigate the performance of the models in high-dimensional settings with sparse informative features, but sufficient data, the number of training observations could be increased to 1000 or more. Furthermore, smaller informative ratios can be considered. For example, the human genome contains between 20,000 and 25,000 protein-coding genes, and it would be insightful to assess how the models perform when working with datasets of that scale. However, increasing the number of observations or features will significantly raise computational demands.

Further improvement could be made by expanding the hyperparameter search grids. As Ridge and Lasso are relatively fast to train, a larger hyperparameter grid could be implemented to improve their performance. This may be particularly beneficial for Ridge, where the limited grid used in this thesis probably reduced its performance. Another possible approach is to begin tuning logarithmically to identify the optimal range and then refine the search linearly. For random forests, more extensive hyperparameter grids could also be explored, though this comes with increased training time. Alternatively, future research could investigate how reducing the size of the parameter grid impacts performance.

The analysis of the relative importance of informative features could also be extended. It would be interesting to evaluate whether the models can correctly identify the relative ranking of informative features across different noise amplitudes. Furthermore, comparing the top five most important features to the next five, or even subsequent sets, may offer deeper insights into how well the models differentiate between informative and non-informative features.

Other evaluation metrics could be considered to provide a more comprehensive comparison of model performance. Examples include the R^2 statistic or the Mean Absolute Error (MAE). A modified version of the Top-k metric used in this thesis could also be developed, where k is defined as a proportion of the total number of features rather than a fixed number. Moreover, extending the analysis to additional models such as

Support Vector Machines (SVM), which are known for their dimensionality reduction, ElasticNet, which combines Ridge and Lasso penalties, or other feature selection techniques, would be a worthwhile direction for future work.

Bibliography

[1] A. Chauhan. Feature importance using random forest classifier in python. https://vitalflux.com/feature-importance-random-forest-classifier-python/, 2021. Accessed: 2025-06-06.

[2] B. Ghojogh and M. Crowley. The theory behind overfitting, cross validation, regularization, bagging, and boosting: Tutorial. 2019. doi: 10.48550/arXiv.1905.12787. URL https://arxiv.org/abs/1905.12787.

[3] Bjarni V Halldorsson, Hannes P Eggertsson, Kristjan H S Moore, and et al. The sequences of 150,119 genomes in the uk biobank. Nature, 607(7920):732–740, 2022. doi: 10.1038/s41586-022-04965-x. URL https://www.nature.com/articles/s41586-022-04965-x.

[4] T. Hastie, R. Tibshirani, and J. Friedman. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer, New York, NY, 2 edition, 2009. ISBN 978-0-387-84858-7. doi: 10.1007/978-0-387-84858-7. URL https://doi.org/10.1007/978-0-387-84858-7.

[5] T. Hastie, R. Tibshirani, and M. Wainwright. Statistical Learning with Sparsity: The Lasso and Generalizations. Chapman & Hall/CRC, 2015. ISBN 1498712169.

[6] G. James, D. Witten, T. Hastie, R. Tibshirani, and J. Taylor. An Introduction to Statistical Learning: with Applications in Python. Springer International Publishing, Cham, 2023. ISBN 978-3-031-38747-0. doi: 10.1007/978-3-031-38747-0. URL https://doi.org/10.1007/978-3-031-38747-0.

[7] Scikit learn Developers. Cross-validation: evaluating estimator performance. https://scikit-learn.org/stable/modules/cross_validation.html, 2024. Accessed: 2025-06-04.

[8] Y. Qi. Random forest for bioinformatics. In Ensemble Machine Learning: Methods and Applications, pages 307–323. Springer, New York, NY, 2012. doi: 10.1007/978-1-4419-9326-7_11.

[9] J. Rogers and S. Gunn. Identifying feature relevance using a random forest. In Subspace, Latent Structure and Feature Selection. SLSFS 2005, volume 3940 of Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, 2006. doi: 10.1007/11752790_12. URL https://doi.org/10.1007/11752790_12.

[10] B. Sen Puliparambil, J. H. Tomal, and Y. Yan. A novel algorithm for feature selection using penalized regression with applications to single-cell rna sequencing data. Biology, 11(10):1495, 2022. doi: 10.3390/biology11101495. URL https://doi.org/10.3390/biology11101495. Received: 11 July 2022; Accepted: 30 September 2022; Published: 12 October 2022.

[11] T. Sirimongkolkasem and R. Drikvandi. On regularisation methods for analysis of high dimensional data. Annals of Data Science, 6(4):737–763, December 2019. ISSN 2198-5812. doi: 10.1007/s40745-019-00209-4. URL https://doi.org/10.1007/s40745-019-00209-4. Received: 14 January 2019; Revised: 23 March 2019; Accepted: 07 April 2019; Published online: 13 April 2019.

[12] Daniel Sorensen. Statistical Learning in Genetics. Statistics for Biology and Health. Springer Cham, 1 edition, 2023. ISBN 978-3-031-35850-0. doi: 10.1007/978-3-031-35851-7.

[13] David J. Spiegelhalter. The Art of Statistics: How to Learn from Data. Pelican Books, London, UK, 2020. ISBN 9780241359183.

[14] Ryan J. Tibshirani. The lasso problem and uniqueness. Electronic Journal of Statistics, 7:1456–1490, 2013.

A

Non-Linear Function Study

A.1. The Non-Linear Functions

ID	Name	Function Definition
f_1	Additive Sine and Square	$\sin(x_1) + x_2^2 + \sin(x_3) + x_4^2 + \sin(x_5)$
f_2	Pairwise Interaction (Arctangent)	$\arctan(x_1 \cdot x_2) + \arctan(x_3 \cdot x_4) + \arctan(x_5)$
f_3	Compositional Log-Sine Interaction	$\log(\lvert\sin(x_1) \cdot x_2\rvert + 1) + \log(\lvert\sin(x_3) \cdot x_4\rvert + 1) + x_5$
f_4	Thresholded Quadratic (Sparse)	$\sum_{i=1}^{5} \begin{cases} x_i^2 & \text{if } x_i > 1 \\ 0 & \text{otherwise} \end{cases}$
f_5	Mixed Univariate Nonlinearities	$\sin(x_1) + \log(\lvert x_2\rvert + 1) + x_3^2 + \arctan(x_4) + e^{x_5}$
f_6	Interaction and Composition	$(x_1 \cdot x_2) + \sqrt{\lvert x_3\rvert} + \log_2(\lvert x_4\rvert + 1) + \sin(x_5)$
f_7	Deep Composition	$\sin(\log(\lvert x_1 \cdot x_2\rvert + 1)) + (x_3 \cdot x_4) + e^{\sqrt{\lvert x_5\rvert}}$

Table A.1: Fixed non-linear functions applied to the first five informative features for generating target responses in the simulation study.

A.2. Performance Results

MSE

Function	Random Forest MSE	Ridge MSE	Lasso MSE
f_1	4.452820	5.241822	5.076747
f_2	0.842715	1.108070	0.774032
f_3	0.468622	1.020078	0.175697
f_4	4.667571	6.317491	5.496146
f_5	4.772339	7.165089	4.936369
f_6	1.604780	1.827565	1.609382
f_7	1.762679	1.916041	1.913722

Table A.2: Results of the small study with noise amplitude 0.01 and informative ratio 0.01. Values represent mean MSE across simulations.

Top-5 Feature Selection Accuracy

Function	Random Forest	Ridge	Lasso
f_1	0.604	0.340	0.288
f_2	0.224	0.212	0.212
f_3	0.228	0.208	0.208
f_4	0.760	0.556	0.544
f_5	0.508	0.352	0.424
f_6	0.348	0.232	0.212
f_7	0.232	0.044	0.008

Table A.3: Top-5 accuracy for each model across non-linear functions with noise amplitude 0.01 and informative ratio 0.01.

Top-10 Feature Selection Accuracy

Function	Random Forest	Ridge	Lasso
f_1	0.760	0.456	0.348
f_2	0.256	0.228	0.224
f_3	0.260	0.208	0.216
f_4	0.872	0.676	0.616
f_5	0.588	0.396	0.472
f_6	0.436	0.236	0.224
f_7	0.300	0.060	0.008

Table A.4: Top-10 accuracy for each model across non-linear functions with noise amplitude 0.01 and informative ratio 0.01.

B

Additional Figures: Linear Results

B.1. Overview of Results for $\sigma = 0.1$

Prediction Error

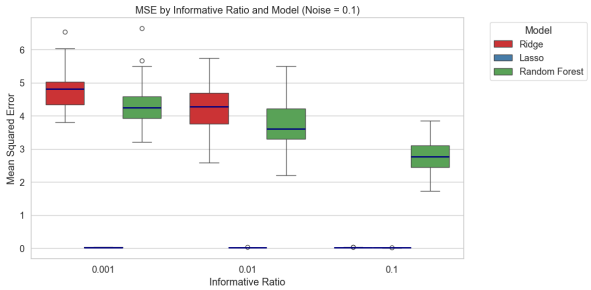


Figure B.1: Mean squared error (MSE) boxplots of the 50 simulations for the three models in the linear case with noise amplitude set to 0.1.

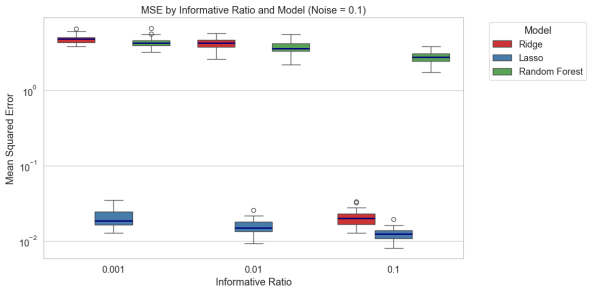


Figure B.2: Mean squared error (MSE) boxplots of the 50 simulations for the three models in the linear case with noise amplitude set to 0.1 (log scale).

Feature Selection Accuracy

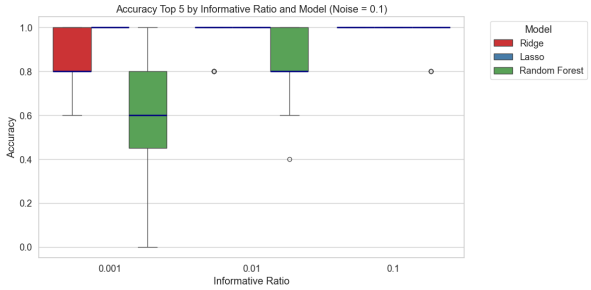


Figure B.3: Accuracy Top 5 of the 50 simulations per model in the linear case with noise amplitude set to 0.1.

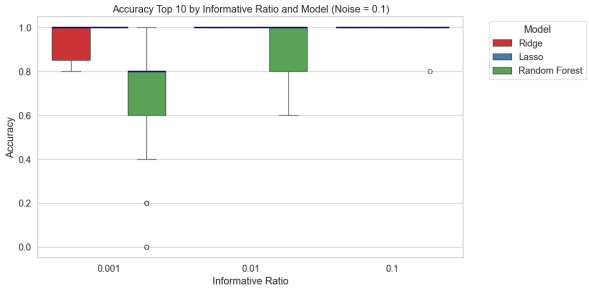


Figure B.4: Accuracy Top 10 of the 50 simulations per model in the linear case with noise amplitude set to 0.1.

B.2. Effect of Noise per Model

Ridge Regression

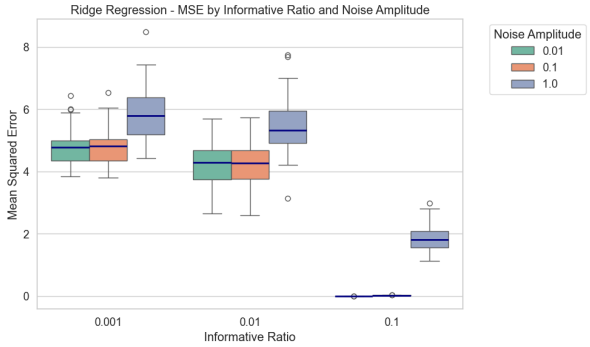


Figure B.5: Mean squared error (MSE) of Ridge regression in the linear case of 50 simulations.

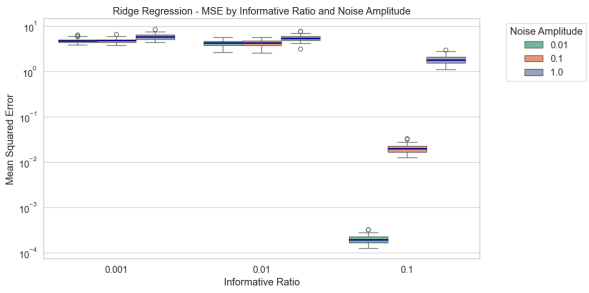
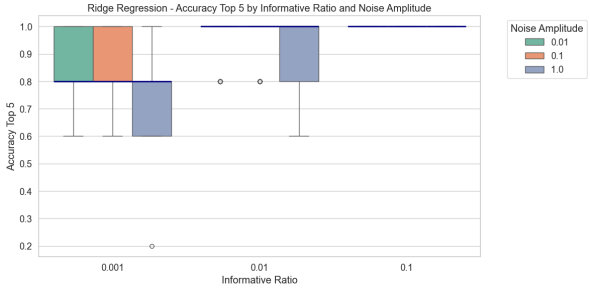
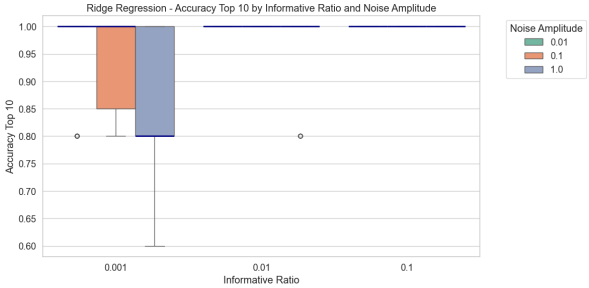


Figure B.6: Mean squared error (MSE) of Ridge regression in the linear case on a logarithmic scale.



(a) Top-5 Feature Accuracy (A_{Top5})



(b) Top-10 Feature Accuracy (A_{Top10})

Figure B.7: Feature selection accuracy of Ridge regression for the top-5 and top-10 most important features of 50 simulations in the linear setting.

Lasso Regression

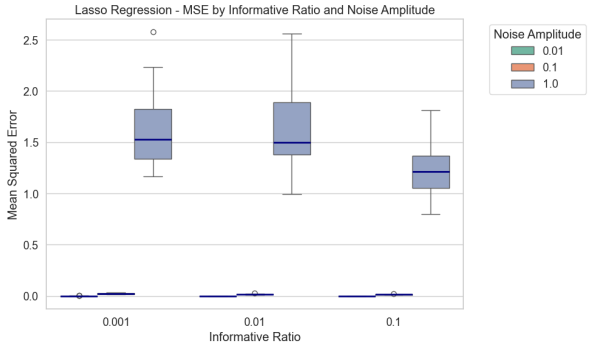


Figure B.8: Mean squared error (MSE) of Lasso regression in the linear case of 50 simulations.

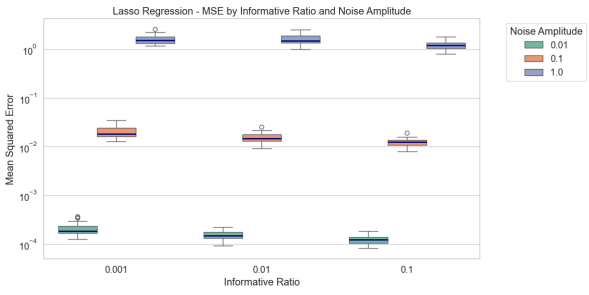
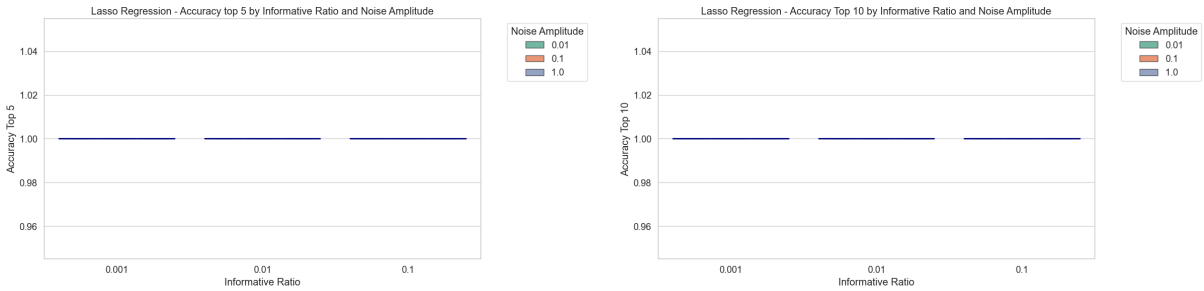


Figure B.9: Mean squared error (MSE) of Lasso regression in the linear case on a logarithmic scale.



(a) Top-5 Feature Accuracy (A_{Top5})

(b) Top-10 Feature Accuracy (A_{Top10})

Figure B.10: Feature selection accuracy of Lasso regression for the top-5 and top-10 most important features of 50 simulations in the linear setting.

Random Forests

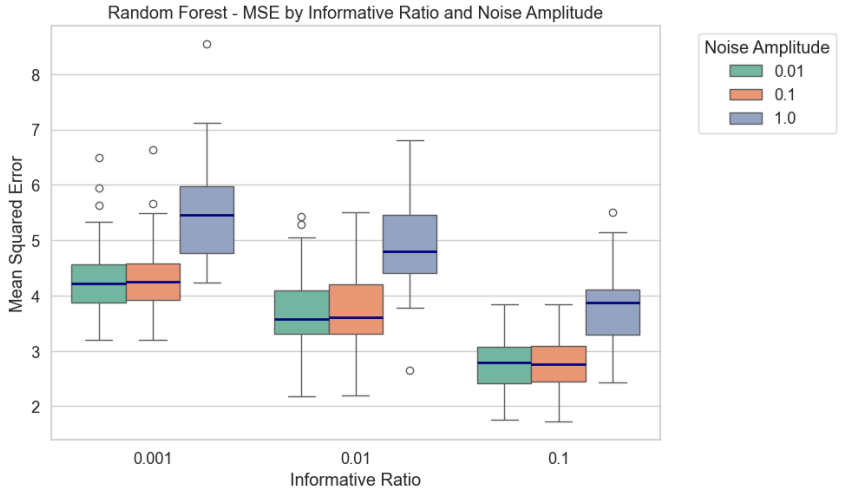
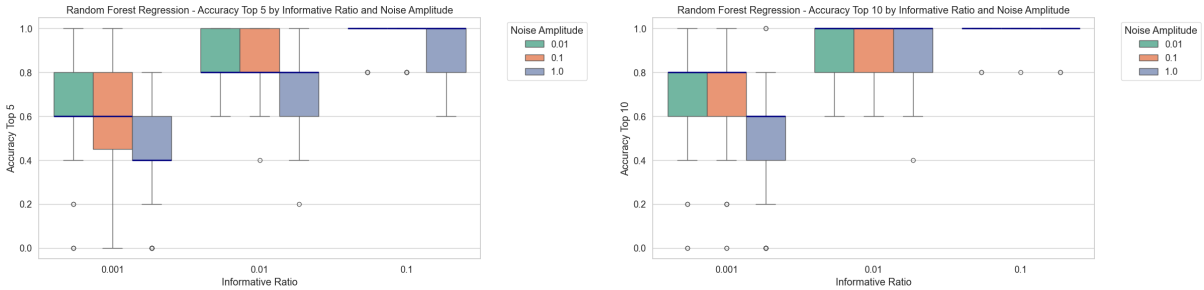


Figure B.11: Mean squared error (MSE) of random forests in the linear case of 50 simulations.



(a) Top-5 Feature Accuracy (A_{Top5})

(b) Top-10 Feature Accuracy (A_{Top10})

Figure B.12: Feature selection accuracy of random forests for the top-5 and top-10 most important features of 50 simulations in the linear setting.

B.3. Computational Efficiency

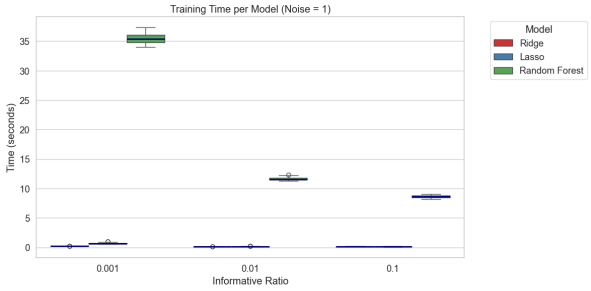


Figure B.13: Training time per fit for each model under higher noise amplitude ($\sigma = 1$), across different informative feature ratios.

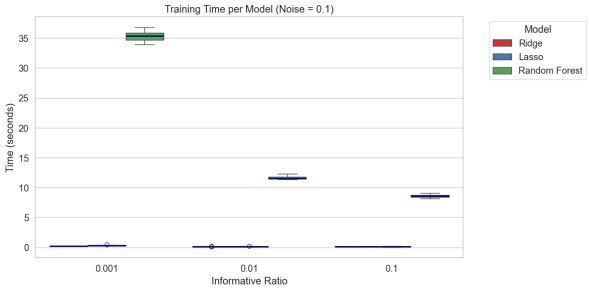


Figure B.14: Training time per fit for each model under higher noise amplitude ($\sigma = 0.1$), across different informative feature ratios.

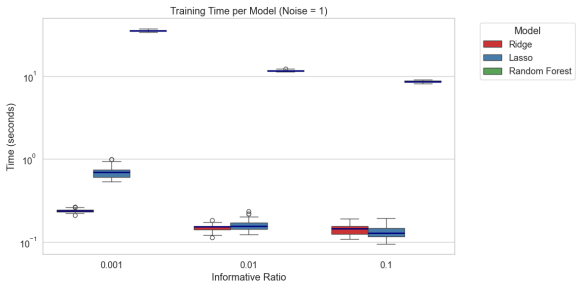


Figure B.15: Training time per fit for each model under higher noise amplitude ($\sigma = 1$), across different informative feature ratios.

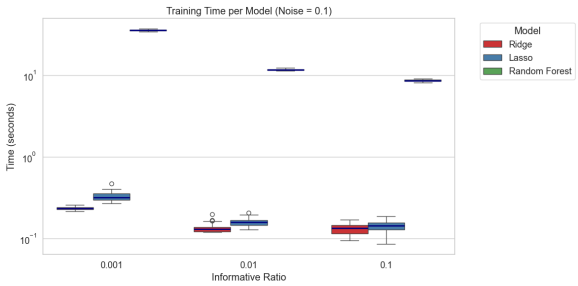


Figure B.16: Training time per fit for each model under moderate noise amplitude ($\sigma = 0.1$), across different informative feature ratios.

C

Additional Figures: Nonlinear Results

C.1. Overview of Results for $\sigma = 0.1$

Prediction Error

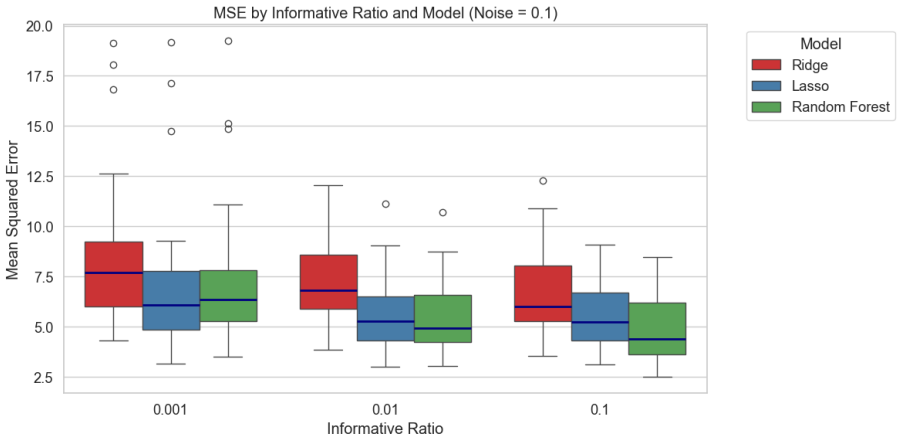


Figure C.1: Mean squared error (MSE) boxplots of the 50 simulations for the three models in the nonlinear case with noise amplitude set to 0.1.

Feature Selection Accuracy

Figure C.2: Accuracy Top 5 of the 50 simulations per model in the linear case with noise amplitude set to 0.1.

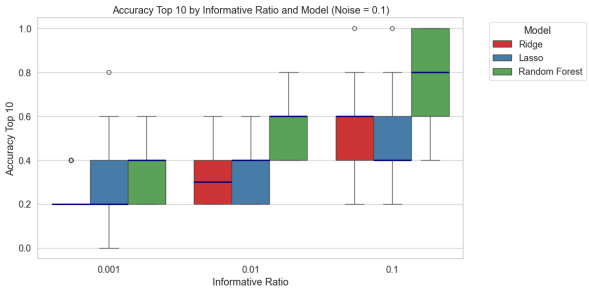


Figure C.3: Accuracy Top 10 of the 50 simulations per model in the linear case with noise amplitude set to 0.1.

C.2. Effect of Noise per Model

Ridge Regression

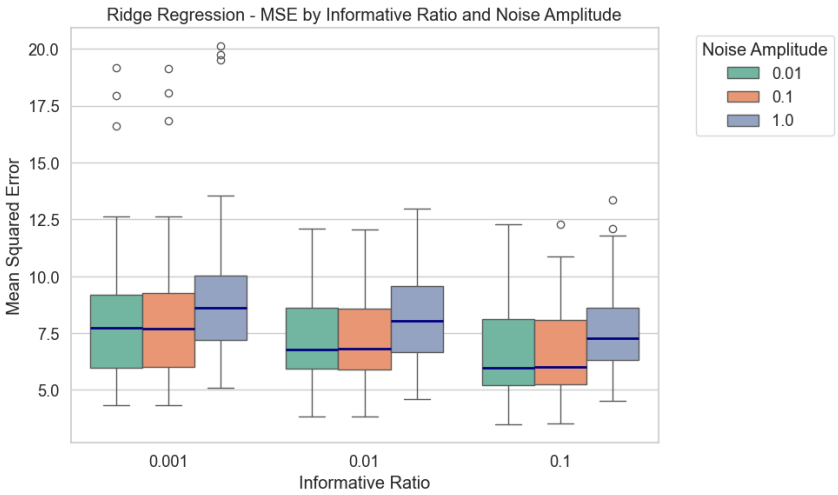
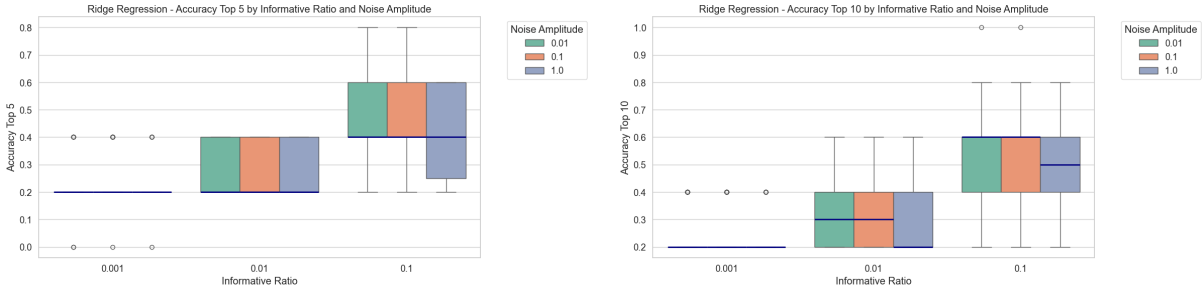


Figure C.4: Mean squared error (MSE) of Ridge regression in the nonlinear case of 50 simulations.



(a) Top-5 Feature Accuracy (A_{Top5})

(b) Top-10 Feature Accuracy (A_{Top10})

Figure C.5: Feature selection accuracy of Ridge regression for the top-5 and top-10 most important features of 50 simulations in the nonlinear setting.

Lasso Regression

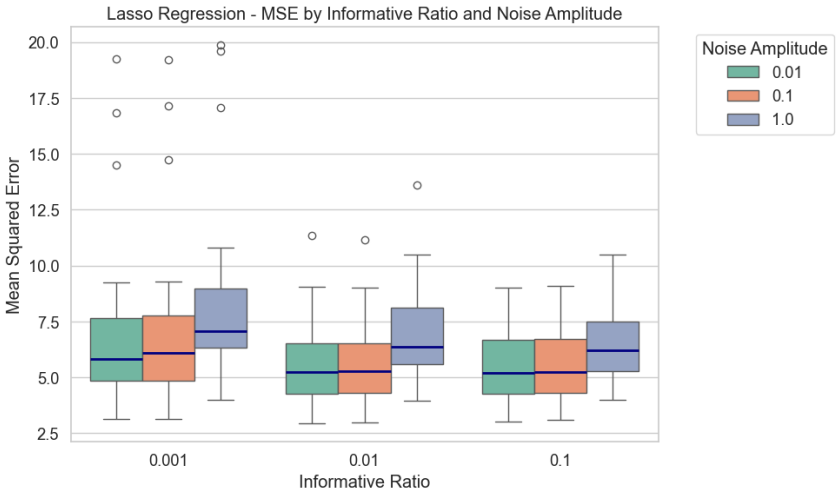
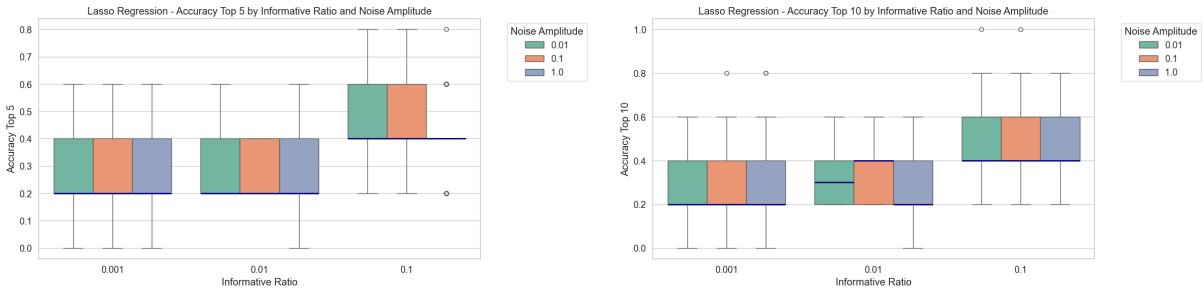


Figure C.6: Mean squared error (MSE) of Lasso regression in the nonlinear case of 50 simulations.



(a) Top-5 Feature Accuracy (A_{Top5})

(b) Top-10 Feature Accuracy (A_{Top10})

Figure C.7: Feature selection accuracy of Lasso regression for the top-5 and top-10 most important features of 50 simulations in the nonlinear setting.

Random Forests

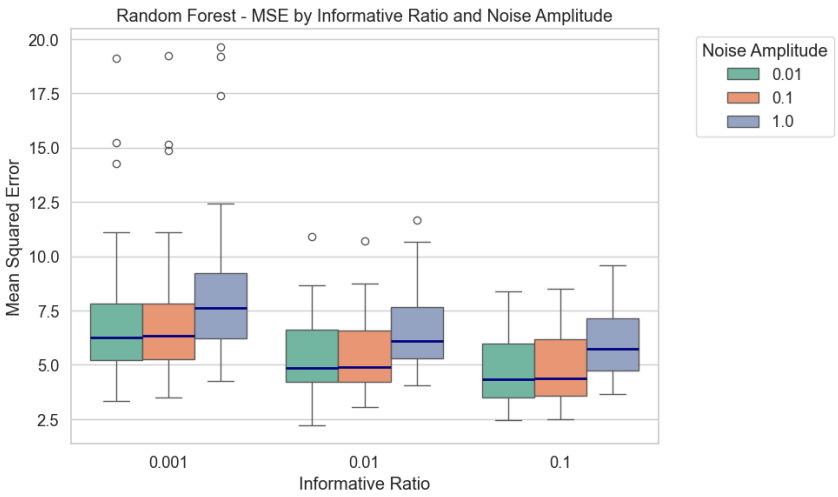
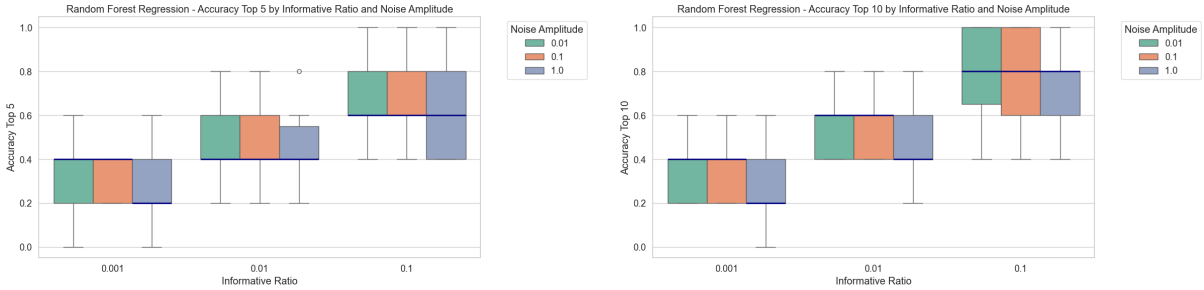


Figure C.8: Mean squared error (MSE) of random forests in the nonlinear case of 50 simulations.



(a) Top-5 Feature Accuracy (A_{Top5})

(b) Top-10 Feature Accuracy (A_{Top10})

Figure C.9: Feature selection accuracy of random forests for the top-5 and top-10 most important features of 50 simulations in the nonlinear setting.

C.3. Computational Efficiency

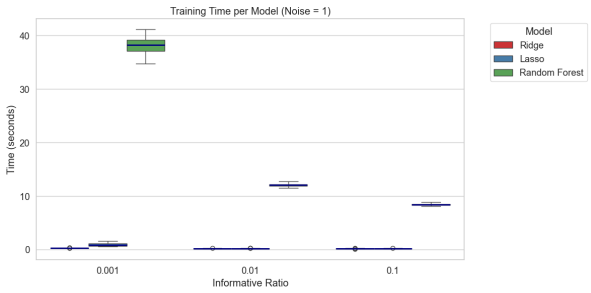


Figure C.10: Training time per fit for each model under higher noise amplitude ($\sigma = 1$), across different informative feature ratios.

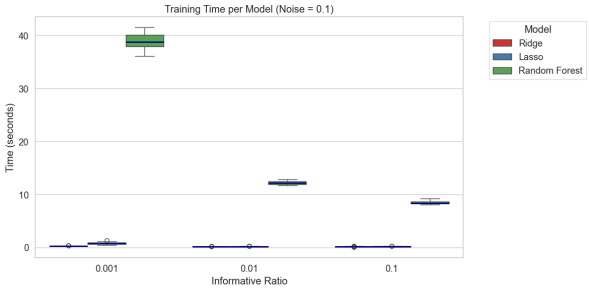


Figure C.11: Training time per fit for each model under moderate noise amplitude ($\sigma = 0.1$), across different informative feature ratios.

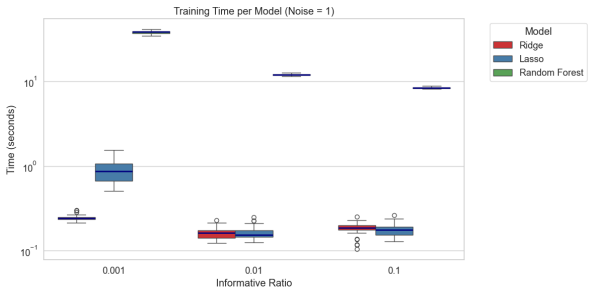


Figure C.12: Training time per fit for each model under higher noise amplitude ($\sigma = 1$), across different informative feature ratios.

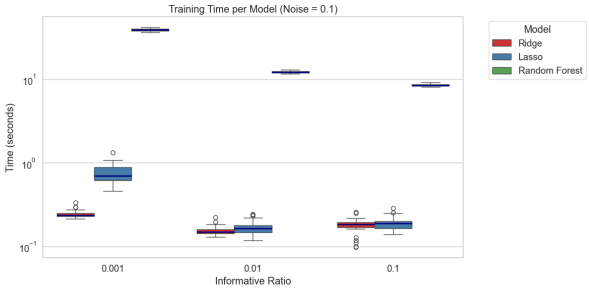


Figure C.13: Training time per fit for each model under moderate noise amplitude ($\sigma = 0.1$), across different informative feature ratios.

D

Relative Importance Study

D.1. Uniform Distribution of the Informative Features

Noise amplitude $\sigma = 0.01$

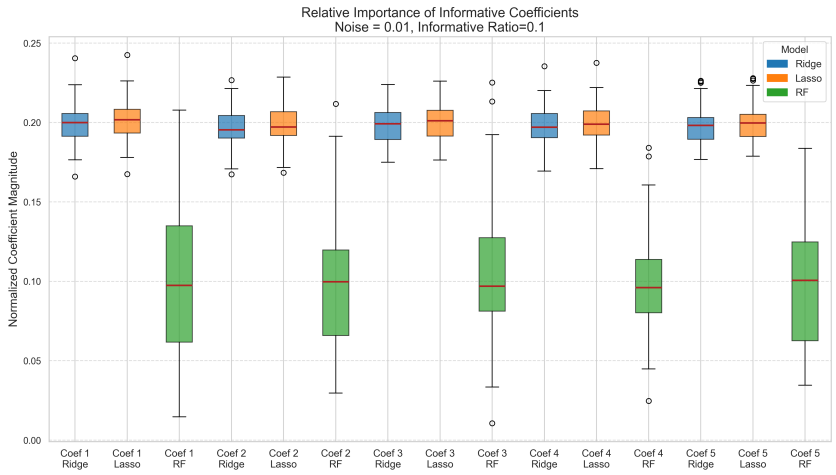


Figure D.1: Relative importance of the first five coefficients for noise amplitude 0.01 and informative ratio 0.1.

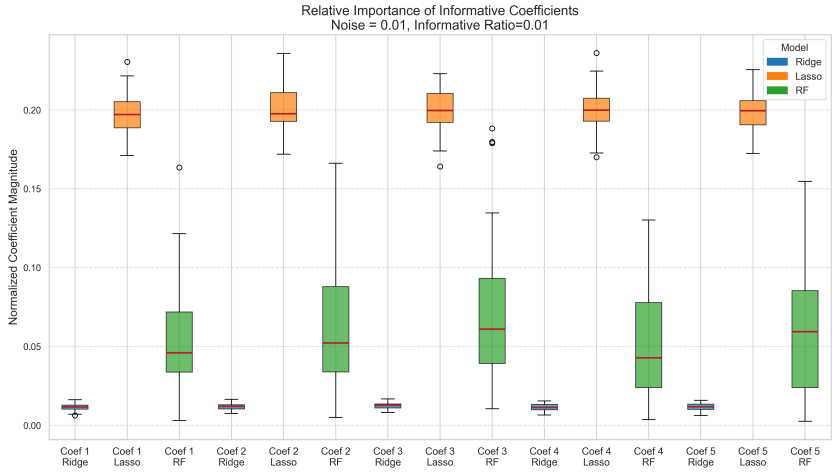


Figure D.2: Relative importance of the first five coefficients for noise amplitude 0.01 and informative ratio 0.01.

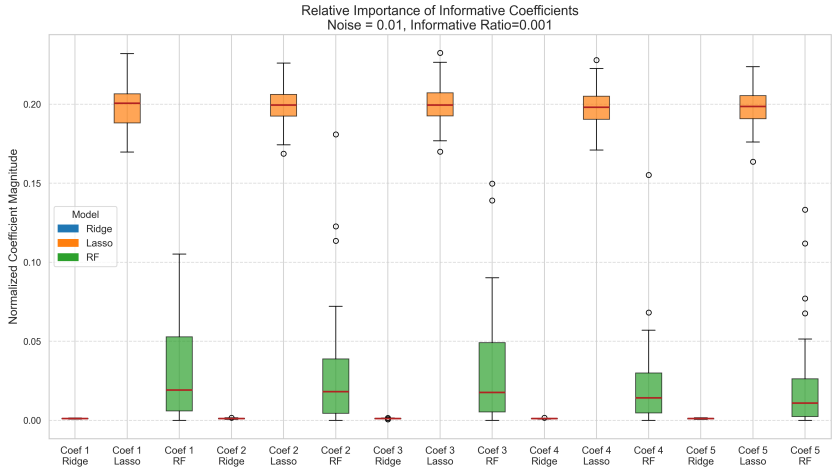


Figure D.3: Relative importance of the first five coefficients for noise amplitude 0.01 and informative ratio 0.001.

Noise amplitude $\sigma = 0.1$

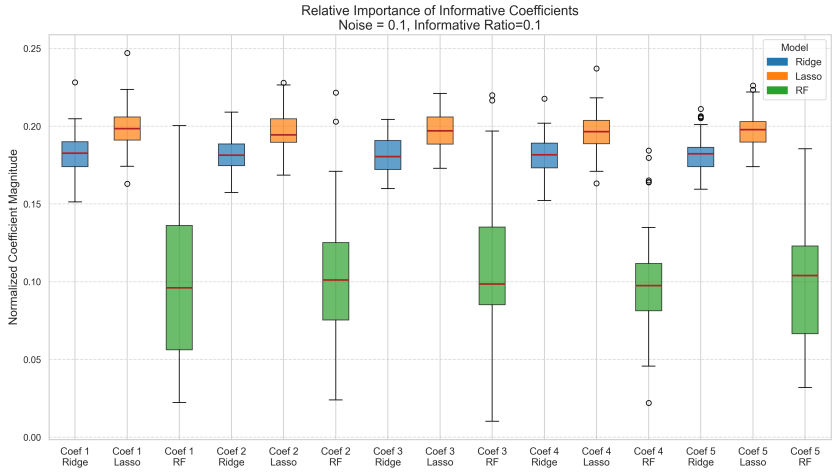


Figure D.4: Relative importance of the first five coefficients for noise amplitude 0.1 and informative ratio 0.1.

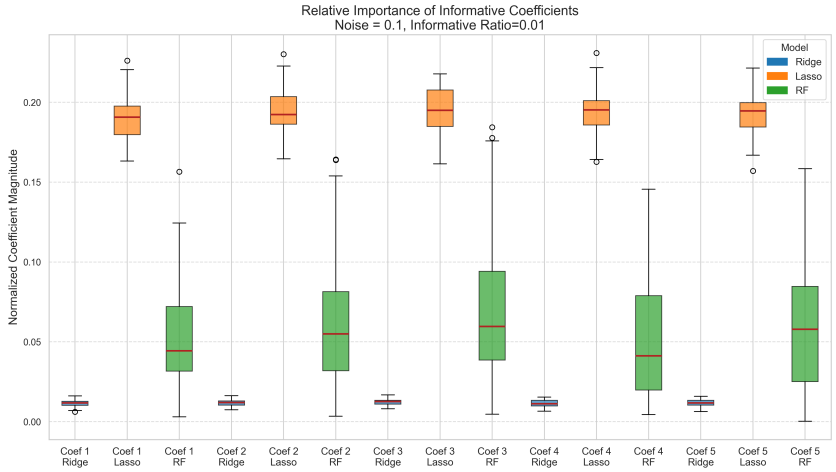


Figure D.5: Relative importance of the first five coefficients for noise amplitude 0.1 and informative ratio 0.01.

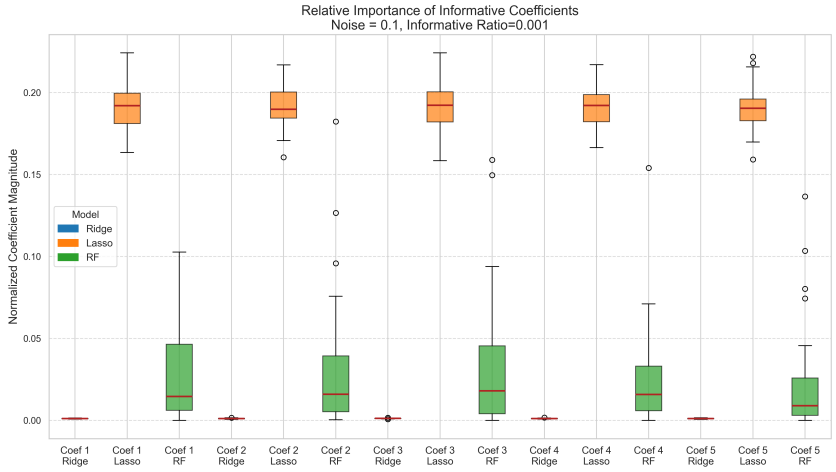


Figure D.6: Relative importance of the first five coefficients for noise amplitude 0.1 and informative ratio 0.001.

Noise amplitude $\sigma = 1$

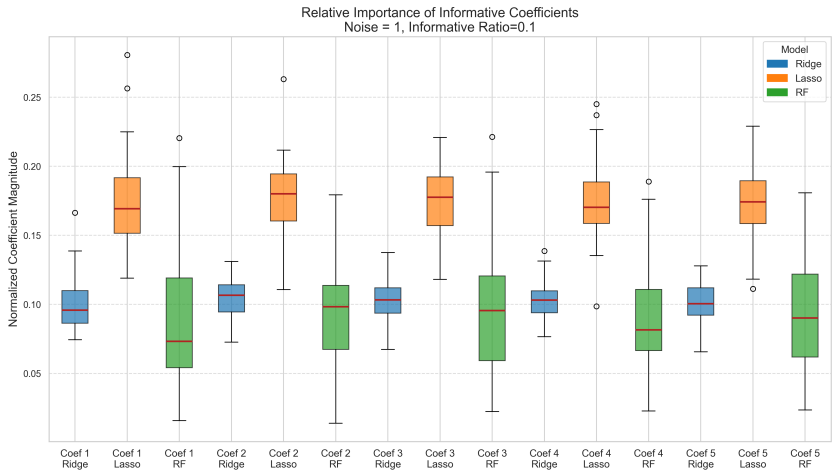


Figure D.7: Relative importance of the first five coefficients for noise amplitude 1 and informative ratio 0.1.

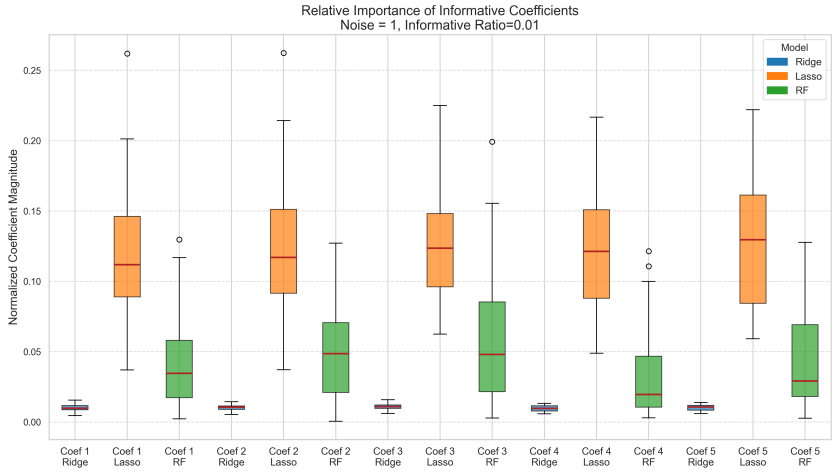


Figure D.8: Relative importance of the first five coefficients for noise amplitude 1 and informative ratio 0.01.

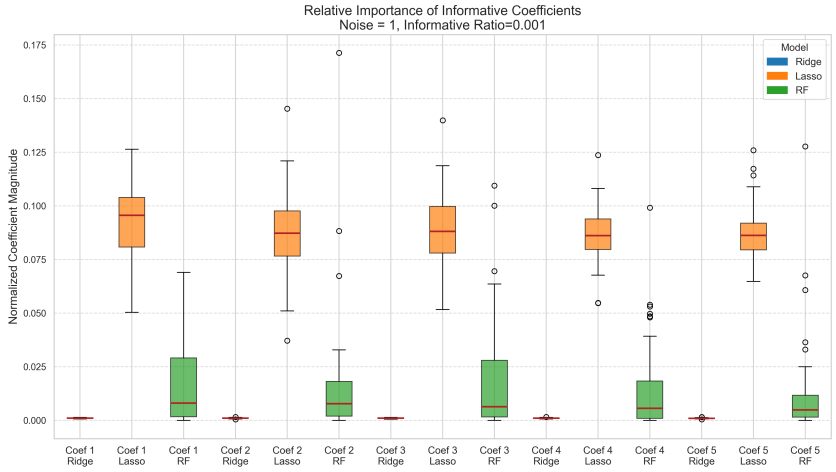


Figure D.9: Relative importance of the first five coefficients for noise amplitude 1 and informative ratio 0.001.

D.2. Non-Uniform Distribution of the Informative Features

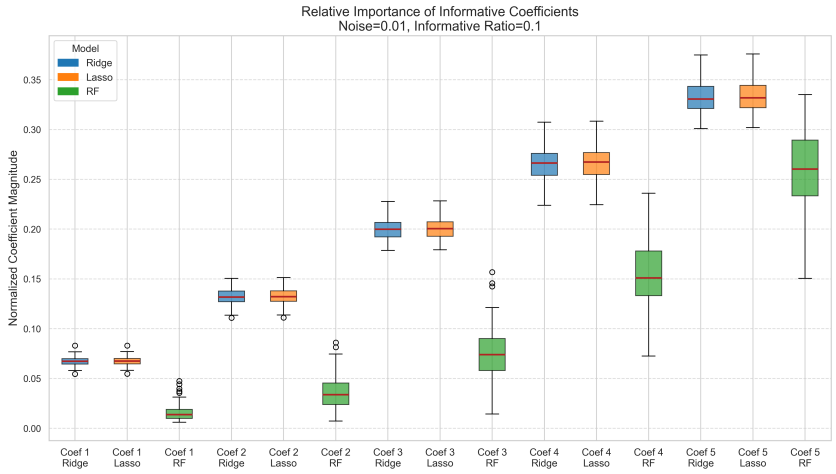


Figure D.10: Relative importance of the first five coefficients for noise amplitude 0.01 and informative ratio 0.1.

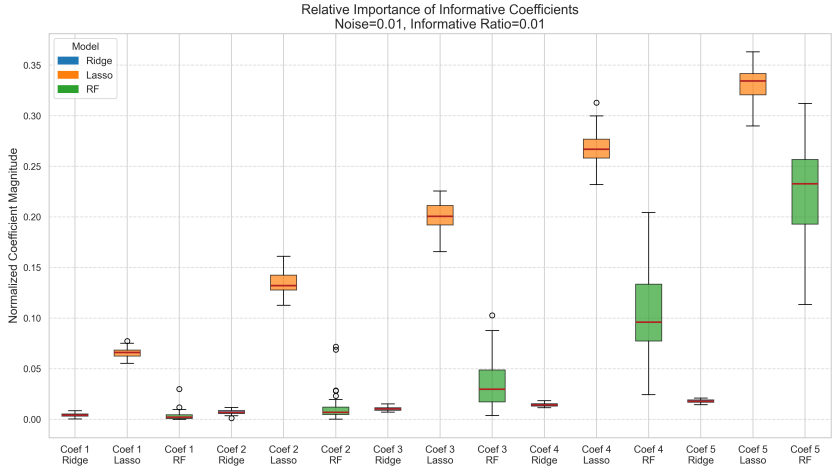


Figure D.11: Relative importance of the first five coefficients for noise amplitude 0.01 and informative ratio 0.01.

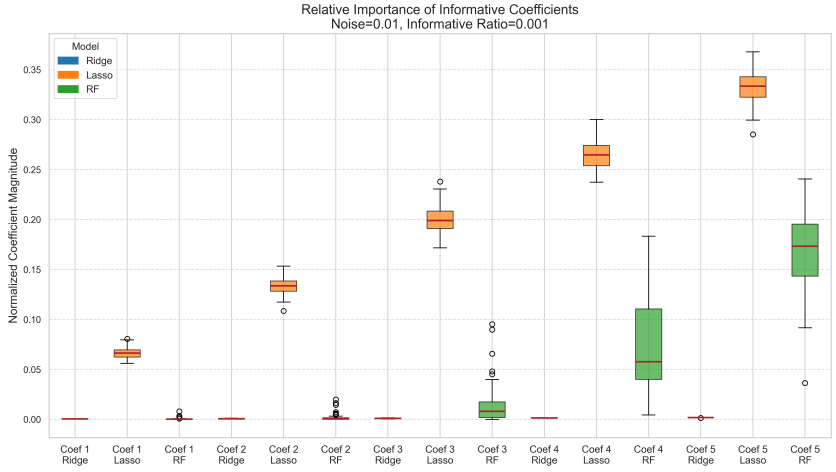


Figure D.12: Relative importance of the first five coefficients for noise amplitude 0.01 and informative ratio 0.001.