Master's thesis

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

Hennie Schau-Hansen

Improving Direct Response Modelling through Hyperparameter Optimization of Extreme Gradient Boosting and Random Forests

Master's thesis in Applied Physics and Mathematics
Supervisor: John Sølve Tyssedal
Co-supervisor: Marianne Frisvold Røe
June 2022

Hennie Schau-Hansen

Improving Direct Response Modelling through Hyperparameter Optimization of Extreme Gradient Boosting and Random Forests

Master's thesis in Applied Physics and Mathematics
Supervisor: John Sølve Tyssedal
Co-supervisor: Marianne Frisvold Røe
June 2022

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

Preface

This master thesis completes my Master of Science (M.Sc.) in Applied Physics and Mathematics with specialization in Industrial Mathematics at Norwegian University of Science and Technology (NTNU). The work was carried out in the spring of 2022 with John Sølve Tyssedal as supervisor and submitted to the Department of Mathematical Sciences. The task and data are provided by SpareBank1 Kreditt AS with Marianne Frisvold Røe as a company supervisor. The reader is assumed to have a basic understanding of statistical modelling and be familiar with some banking and credit card terminology.

I would like to thank SpareBank1 for giving me the opportunity to write this thesis and a special thank you to my supervisor John Sølve Tyssedal for all the help, guidance, and support. Our weekly meetings have always been informative and pleasant. Lastly, I would like to thank all my friends and family for the continuous support. The last 5 years have been wonderful, and I will always be grateful for all memories and friendships.

Trondheim, 8th June 2022

....

Hennie Schau-Hansen

Abstract

Response modelling can be applied in direct marketing to rank customers by the likelihood of response. This is done to increase the response rate of a campaign, and thus increase revenue. This thesis will focus on a call campaign with an offer to refinance conducted by SpareBank1, directed at customers eligible for refinancing of consumer loans and credit cards. The main objective of this thesis is to build and optimize models that can predict which customers will respond to the campaign. This is a binary classification task on an imbalanced dataset, where the response is either "yes" or "no". The dataset is provided by SpareBank1 and contains historical data from previous call campaigns collected from March 2020 to July 2021. Furthermore, it is essential to understand what type of customer accepts the offer to refinance.

Extreme Gradient Boosting (XGBoost) and Random Forests were the two machine learning algorithms used to build the predicative models in this thesis. XGBoost was chosen because it is effective and often outperforms other methods, while Random Forests was chosen because it is a well-established method that has been proven to be robust. The models were evaluated and optimized with emphasis on the balanced accuracy and the sensitivity, which is the model's ability to classify the positive responders. To improve the classification, the hyperparameters of the two methods were tuned. First, the tuning was performed with a screening experiment using Design of Experiments (DoE) and then further optimization using Response Surface Methodology (RSM). DoE can identify which hyperparameters are most significant and in what configuration. Second, the hyperparameters were optimized using Bayesian optimization. Combinations of Bayesian optimization, DoE, and RSM were also tested to check the effects of screening and applying a central composite design as an initial grid. Lastly, feature importance before and after tuning was investigated.

For both methods, the screening experiment identified the most influential hyperparameters as those directly affecting the class weights. These hyperparameters were chosen for further optimization using RSM. RSM successfully optimized the hyperparameter values and improved the balanced accuracy. More importantly, the classification of the important positive class improved, leading to an increase in the sensitivity. Bayesian optimization was also applied, which also improved the classification by increasing the balanced accuracy and the sensitivity. A more stable optimization was achieved with Bayesian optimization in combination with RSM. The highest balanced accuracy scores were obtained from models tuned with Bayesian optimization. Compared to benchmark results, this led to an improvement of 19% and 16% in the balanced accuracy for XGBoost and Random Forests. However, the difference between the optimized models was not evident. Tuning did not significantly affect the calculated feature importance, and the variable INTEREST_EARNING_LENDING_AMT proved to be important in the prediction.

Sammendrag

Responsmodellering brukes i direkte markedsføring til å rangere kunder etter sannsynligheten for respons. Dette gjøres for å øke responsraten til en kampanje, og dermed øke inntektene. Denne oppgaven vil fokusere på en ringekampanje med et tilbud om å refinansiere utført av SpareBank1, rettet mot kunder som er kvalifisert for refinansiering av forbrukslån og kredittkort. Hovedmålet med denne oppgaven er å bygge og optimalisere modeller som kan forutsi hvilke kunder som vil respondere på kampanjen. Dette er en binær klassifiseringsoppgave på et ubalansert datasett, der svaret enten er "ja" eller "nei". Datasettet er levert av SpareBank1 og inneholder historiske data fra tidligere ringekampanjer samlet inn fra mars 2020 til juli 2021. I tillegg er det viktig å få en forståelse av hva slags type kunde som aksepterer et slikt tilbud om refinansiering.

Extreme Gradient Boosting (XGBoost) og Random Forests var de to maskinlæringsalgoritmene brukt til å bygge de predikative modellene i denne oppgaven. XGBoost ble valgt fordi den er effektiv og ofte utkonkurrerer andre metoder, mens Random Forests ble valgt fordi det er en robust og veletablert metode. Modellene ble evaluert og optimalisert med vekt på den balanserte nøyaktigheten og sensitiviteten, altså modellenes evne til å klassifisere kundene som takker ja. For å forbedre klassifiseringen, ble hyperparametrene til de to metodene optimalisert. Optimaliseringen ble først gjort med et screeningseksperiment ved bruk av forsøksplanlegging (DoE) og deretter videre optimalisering gjennom responsflatemetodikk (RSM). DoE kan identifisere hvilke hyperparametere som er mest betydningsfulle og i hvilken konfigurasjon. I tillegg ble hyperparametrene optimalisert med Bayesiansk optimering. Kombinasjoner av Bayesiansk optimering, DoE og RSM ble også testet for å sjekke effekten av screening og bruk av et sentralt sammensatt forsøk (CCD) som innledende verdier. Til slutt ble variabel viktighet før og etter optimalisering undersøkt.

For begge metodene identifiserte screeningseksperimentet de mest innflytelsesrike hyperparametrene som de som direkte påvirket vektingen av klassene. Disse hyperparametrene ble valgt for ytterligere optimalisering ved bruk av RSM. RSM optimaliserte verdiene til hyperparametrene og forbedret den balanserte nøyaktigheten. Klassifiseringen av den viktige positive klassen ble også forbedret, noe som førte til en økning i sensitiviteten. Bayesiansk optimering forbedret også klassifiseringen ved å øke sensitiviteten og den balanserte nøyaktigheten. En mer stabil optimalisering ble oppnådd med Bayesiansk optimering i kombinasjon med RSM. Modeller optimalisert med Bayesiansk optimering oppnådde de høyeste verdiene for balansert nøyaktighet. Sammenlignet med referanseresultater, førte dette til en forbedring på 19% og 16% i den balanserte nøyaktigheten for XGBoost og Random Forests. Forskjellene mellom de optimaliserte modellene var imidlertid ikke store. Variabel viktigheten før og etter optimalisering viste ikke store forskjeller, og variabelen INTEREST_EARNING_LENDING_AMT var viktig i prediksjonen.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Consumers are constantly surrounded by marketing through advertisements, campaigns, and companies that try to promote themselves. The main objective of marketing is to build profitable customer relationships, [1]. Companies must understand the customer's needs and wants, decide which markets to target, and develop a compelling value proposition where the company is able to attract and keep the targeted customers. A good marketing strategy is usually based on information and can lead to an increase in the return on investment and effectiveness of promotional efforts.

Mass marketing is the cheapest market strategy and aims to reach as many people as possible. Individual consumers in a product market are believed to have the same needs and preferences, [4]. Most consumers are also believed to be satisfied by a single market offering. Marketing is usually done with little or no variation, such as a marketing campaign that targets everyone. In mass marketing, market segmentation is ignored. The main drawback of mass marketing is its inability to meet different consumer needs. The opposite of mass marketing is market segmentation, where the market is divided into segments of customers with similar needs. In this way, the marketing can better satisfy the needs of customers across segments and therefore increase the customer relationship. However, market segmentation can be expensive and time-consuming.

Direct marketing takes market segmentation one step further by generating a direct response from the customer. Direct marketing is done through companies targeting a pre-selected customer and requiring a response, [51]. This can be done, for example, through personal sales, phone calls, direct mail, catalogues, and coupons. The most fundamental question of direct marketing is to decide who should be targeted, [39]. A good relationship with the customers is crucial, meaning that a company should not make irrelevant offers to customers. This can weaken the company's ability to build trust, earn customer loyalty, and strengthen the customer relationship. Additionally, there can be a significant cost related to each communication effort, and sending offers to customers who are unlikely to respond is unprofitable.

The process of identifying patterns and rules in large volumes of data is called data mining and is an important part of analytic customer relationship management, [7]. Many companies collect large volumes of data through the company's interaction with the customers, for instance, through transaction processing systems such as telephone switch records and supermarket scanner files. Different data mining tools can be applied to the data to learn more about customers, their user patterns, and their interactions with the company. Data mining through response modelling is often applied in direct marketing. Response modelling estimates the likelihood of the response of the targeted customers. Customers can then be ranked by this score so that the company knows which customers to focus on first. Consequently, the response rate of campaigns can be increased. The estimates are usually built on customer responses from past customer behaviour and marketing activity.

Gjeldsregisteret, launched in Norway in 2019, gave banks and other financial institutions a better overview of unsecured debts and loans in the Norwegian population. This allows them to give correct credit evaluations of customers and thereby impede the growth of unsecured debt. Norway, being one of the richest countries in the world, is at the top of the world in the amount of private debt, [24]. As of January 2022, almost 3.2 million Norwegians were registered with unsecured debt, [22]. This is equal to 59% of the total population. The total amount of unsecured debt has decreased from almost 170 billion NOK in September 2019 to 141 billion NOK in January 2022. The reduction is in part because of better credit evaluations due to the launch of *Gjeldsregisteret*, as well as the Corona pandemic. However, now that society has reopened, unsecured consumer debt is again increasing. In February, the total amount of unsecured consumer debt increased with 0.3%, reflecting the increase in consumption due to the opening. In a report by Statistics Norway, [26], it is shown that the distribution of unsecured debt among the population is unevenly distributed. The report stated that people of the age 40 to 60 have the highest amount of unsecured debt, in addition to men having a significant higher debt than women. Furthermore, people with low formal education and immigrant backgrounds, single fathers, and disability benefit recipients are more vulnerable to unsecured debt. These groups have on average markedly higher debts than the average of the entire population, in addition to lower income and wealth.

Refinancing consumer loans and credit cards can be a helpful solution when a person has accumulated a large amount of debt. SpareBank1 offers refinancing, where a customer receives a new loan to pay off existing consumer loans or credit cards, often with better terms and interest rate. Consumer loans and credit cards are in this way collected into one, single loan. This can lead to reduced expenses, fewer bills, and more control over own finances. A customer can save money by refinancing expensive consumer loans and credit cards. By choosing to refinance with SpareBank1, a young man saved 180000 NOK, as well as receiving lower interest rates and shorter repayments.

SpareBank1 conducts a call campaign targeting customers eligible for refinancing of consumer loans and credit cards. Customers often need a ”push” to refinance, and this is easier achieved through a phone call than, for instance, through mail. The phone activity started in the middle of March 2020, initially aimed at customers with a good payment history who might want to refinance. These customers were selected primarily because they had high interest-bearing debt in SpareBank1, in addition to payments to other competing banks. This is an indication that the customer would want to collect all accumulated debt into one refinancing loan. Thus, the extracted customers were based on business logic. In order to increase volume and success rate, SpareBank1 then decided to perform response modelling on the call campaign. During the summer of 2020, customers were selected based on a score provided by a machine learning model. The customers with the highest score, in addition to some business rules, were called. The model takes several measures into account, such as the age of the customer, the age of the account, and whether the customer has been in debt collection or had other defaults. The model is continuously updated to increase volume as the call activity is extended.

Therefore, the objective of this thesis is to build and optimize models that are able to classify which customers would accept an offer to refinance. The models are trained based on past customer history, and customer data need to be available. The modelling consists of several steps, such as data pre-processing, feature construction, hyperparameter optimization, classification, and evaluation. Two machine learning algorithms are explored and the predictive powers of the models are investigated. Then, the models are tuned and optimized to see if better classification can be obtained. This is done by optimizing the hyperparameters of the algorithms that control the learning process. In this way, SpareBank1 can gain valuable knowledge of what type of customer accepts the offer to refinance and thus which customers to target in the future. Lastly, it is desirable to see which features are influential and can be considered significant in the classification.

1.1 Methodological Background and Previous Research

Direct response modelling has been investigated using many different methods. Knowing which customers are more likely to respond to a certain campaign is of utmost importance for marketers. In [16], customer response modelling was performed to improve direct mail targeting. This paper investigated well-known statistical and data mining classification techniques, which were then evaluated on four real-life direct marketing datasets. The methods investigated were logistic regression, linear and quadratic discriminant analysis, naïve Bayes, neural networks, the k-nearest neighbour algorithm (k-NN), and decision trees including Chi-squared Automatic Interaction Detection (CHAID), Classification And Regression Trees (CART), and C4.5. C4.5 is a decision tree algorithm that make splits based on information gain, [50]. The evaluation metric used was the area under the receiver operating characteristics curve (AUC) calculated using 10-fold cross-validation. The findings showed that the data mining algorithms CHAID, CART, and neural networks performed best, followed by the statistical classifiers logistic regression and linear discriminant analysis. The average rank of the algorithms showed that CHAID was ranked highest, while k-NN with 10 nearest neighbours was ranked lowest. In [47], customer response models were evaluated on a set of data related to the sale of beef products. Logistic regression, neural networks, and decision trees were compared to a variety of recency, frequency, and monetary methods, again showing that the data mining techniques logistic regression, decision trees, and neural networks achieved best accuracy on test data.

In the context of response modelling, most of the real-world marketing datasets, such as customer churn predictions and customer response predictions, are imbalanced. This can lead to data analysis techniques being biased towards the majority cases, leading to deficient classification performances. In [31], this problem was investigated by developing customer response models with support vector machines (SVM) to imbalanced datasets. Random undersampling was applied to accommodate the class imbalance problem. The performance of the models was evaluated based on accuracy and gain score. The findings suggested that the predictive models were affected by the class imbalance. In highly imbalanced datasets decision trees, logistic regression, and neural networks classified all instances as belonging to the majority class, while SVM achieved a positive sensitivity. In the case of moderate class imbalance, SVM performed worse than the other techniques. When the data were balanced, there were no significant differences in the performance of the different response models.

In many situations, customer response models rely heavily on feature engineering and the analyst's domain knowledge and expertise in constructing relevant predictors. Traditional methods can be complicated when the complexity of the data increases. In [52], a long-short-term memory (LSTM) neural network was proposed to avoid feature engineering, as LSTM neural networks require only raw data as input. The authors of this article gave the task of maximizing the net financial performance of a marketing campaign to 299 graduate students. The students competed against each other and the authors and could freely choose which methods to use. Among the contestants, the top 20 results used logistic regression with lasso and/or ridge penalties. Moreover, LSTM neural networks showed excellent promise of being used in response modelling, as it outperformed both Random Forests and Extreme Gradient Boosting (XGBoost).

In [10], credit risk assessment models were built for financial institutions to reduce the risk associated with defaults and improve loan business efficiency. This was also a class imbalance problem, considering ther were more non-defaults than the number of defaults. Class imbalance was dealt with using cluster-based undersampling. The models were evaluated using the AUC. The paper concluded that Extreme Gradient Boosting (XGBoost) was the superior model according to the AUC, compared to logistic regression, SVM, and neural networks based on group method of handling data (GMDH). The research found that XGBoost can be particularly useful in the development of credit risk models for financial institutions.

Most machine learning algorithms contain several hyperparameters that can be tuned. This can optimize the performance of the model and increase the predictive capability. In [37], the hyperparameters were tuned using Design of Experiments as a screening experiment and Response Surface Methodology for optimization, with a Random Forests case study. The results showed an outstanding increase in the cross-validated balanced accuracy, from a default of 0.64 to 0.81. The proposed methodology correctly tuned the hyperparameters but, more importantly, provided information about which factors had the largest effect on the response.

Efficient and automatic tuning of the hyperparameters are also desirable characteristics. Bayesian optimization is an automatic tuning process and can be applied to tune hyperparameters. In [61], Bayesian optimization was applied to tune the hyperparameters of Random Forests, neural networks, and multi-grained cascade forest. The proposed method was able to find the best configuration of hyperparameters by increasing the predictive accuracy compared to the default hyperparameters. Bayesian optimization is also shown to be less time-consuming compared to the exhaustive grid search. In [57], the hyperparameters of logistic regression and Random Forests were tuned using Bayesian optimization and evaluated using a landslide susceptibility mapping. The AUC values improved by 4% and 10%, respectively. The findings suggest that Bayesian optimization had a significant impact on the accuracy of the models.

This master thesis is a continuation of the project thesis, [53], which covered the same problem statement. Two methods for classification were assessed, Extreme Gradient Boosting (XGBoost) and logistic regression. Several methods for feature selection and class balancing were tested, improving the model performances compared to the benchmark results. The best model according to the balanced accuracy was logistic regression with feature selection through Lasso regression and optimal cut-off value. Additionally, several master theses have been written on the same theme. In [46], logistic regression was compared to Random Forests with tuned hyperparameters through Design of Experiments and Response Surface Methodology. Methods for balancing the data were also tested. According to the balanced accuracy, Random Forests with tuned hyperparameters performed best. In [60], the hyperparameters of Random Forests were optimized through Design of Experiments and Response Surface Methodology in combination with resampling methods, with a case study on credit scoring. The result showed that after optimizing the hyperparameters, the balanced accuracy was increased by 39%, resulting in improved classification performance. With the optimized hyperparameters, Random Forests performed approximately equal when trained on unsampled, undersampled, and oversampled data.

1.2 Outline

Two methods are chosen to investigate the problem statement of this thesis. The common practise in SpareBank1 is to use Random Forests, a tree ensemble method based on bagging. Therefore, this method is chosen. In addition to being a well-established method, it has been proven to produce reliable results. The second method chosen is Extreme Gradient Boosting (XGBoost), another tree ensemble method based on gradient boosting. This method is chosen because it has been shown to be efficient and often outperforms other methods. One of the authors of XGBoost, Tianqi Chen claims: "The name XGBoost, though, actually refers to the engineering goal to push the limit of computation resources for boosted tree algorithms. Which is the reason why many people use XGBoost".

The outline of this thesis is as follows: this chapter has introduced the problem statement, motivation, and results from previous studies. Chapter 2 provides relevant background theory for the two methods, in addition to methods for model evaluation and hyperparameter optimization, including Design of Experiments, Response Surface Methodology, and Bayesian optimization. Chapter 3 describes the investigation of the dataset and provides information and visualizations of the explanatory variables and the response. Lastly, feature construction and pre-processing of the data are explained. Chapter 4 provides descriptions of the modelling and experiments performed to investigate the two methods. Chapter 5 analyses the main results. Finally, the results are discussed in Chapter 6 and concluding remarks are made in Chapter 7.

Chapter 2

Background Theory

2.1 Statistical Learning

Statistical learning is learning from data and refers to different tools used to understand data, [28]. Statistical learning can be divided into two groups: supervised and unsupervised learning. In supervised learning, the goal is to build statistical models to predict an output based on inputs. In contrast, unsupervised learning includes only inputs and no measurement of the outcome. The goal is to describe how data are structured and related. This thesis will focus on supervised learning.

In supervised learning, a response y is related to p different predictors, $\mathbf{x} = [x_1, x_2, ..., x_p]^T$. In general, the relationship can be described as

$$y = f(\mathbf{x}) + \epsilon, \tag{2.1}$$

where f represents the systematic information that $\mathbf{x}$ provides about y and ϵ is a random error term. The error term is assumed to be independent of $\mathbf{x}$. Estimating f is important in statistical learning and can be used for both prediction and inference. The emphasis of this thesis is to make as accurate predictions as possible. The function can generally be estimated as $\hat{y} = \hat{f}$, where $\hat{f}$ is an estimate of f. It introduces a reducible error. Therefore, the objective of statistical learning is to estimate f by minimizing the reducible error and thus improving the predicative capability of the model. Some machine learning methods are considered black boxes since it can be hard to understand and explain the behaviour of the model.

To estimate the error associated with the fit of a statistical learner method on a set of observations, the validation set approach can be applied, [28]. This involves randomly splitting the data into a training set and a test set. The training set is used to fit the model, while the test set is used to predict the responses for the test observations based on the fitted model. In this way, a reliable test set error rate can be estimated. It can often be useful to split the data into three parts in data-rich situations; a training set, a validation set, and a test set. In this case, the validation set is used in model selection to estimate the prediction error, while the test set is used to evaluate the final chosen model. The test set should therefore only be used at the end of the data analysis. However, deciding how much data are necessary for division into three parts is hard to estimate. Therefore, this thesis will focus on division into a training and test set. There are two potential drawbacks to this approach; the first being that the estimate of test error can depend on which observations are included in the training set and test set. Thus, it can be highly variable. Secondly, the division leads to fewer observations being used for training, which might lead to an overestimate of the error rate for the fit of the model across the entire dataset.

2.1.1 K-Fold Cross-Validation

Cross-validation is a refinement of the validation set approach and aims to address the two potential drawbacks. There are two types: Leave-One-Out cross-validation and K-fold cross-validation. Often, K-fold cross-validation gives more accurate estimates of the test error than Leave-One-Out cross-validation, [28], and will be the focus in this thesis. This procedure involves splitting the data into K equal sized parts, [21]. Then, $K-1$ parts of the data are used to fit a model, while the Kth part is used to calculate the prediction error. This is done for each Kth set, with a different test set in each run. The K estimates of the prediction error are then averaged, resulting in the cross-validation estimate of the prediction error. In practice, $K = 5$ or $K = 10$ are two reasonable choices. When $K = N$, where N is the number of observations in the dataset, K-fold cross-validation coincides with Leave-One-Out cross-validation. Compared to $K = 5$ and $K = 10$, $K = N$ has one obvious disadvantage, namely a potential large computational cost. An illustration of 5-fold cross-validation is provided in Figure 2.1.

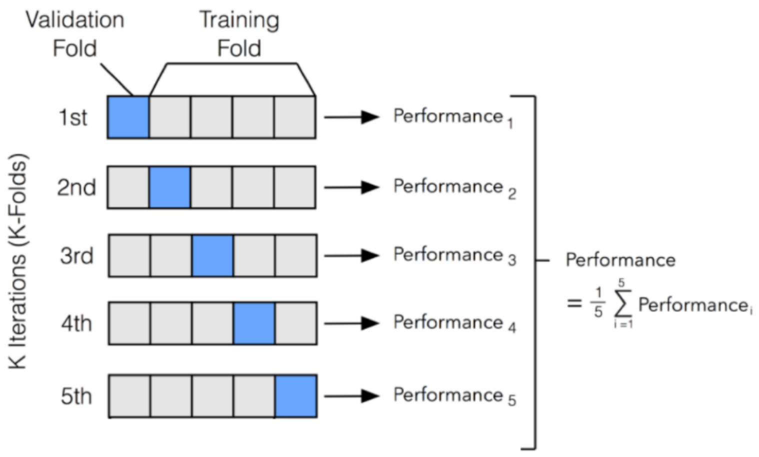


Figure 2.1: Illustration of 5-fold cross-validation. For each of the 5 folds, the prediction error is measured. The cross-validation estimate is then the average of the 5 measured predictions errors. Taken from [23].

2.1.2 The Bootstrap

The bootstrap can be used to quantify the uncertainty associated with a statistical learning method. It is an extremely powerful statistical tool and is widely applicable, [28]. The bootstrap is a resampling method, where n observations from the data are randomly selected in a bootstrap dataset. The sampling is done with replacement, meaning that an observation can be chosen several times. The resampling of n random observations is repeated B times, yielding B bootstrap datasets. For each bootstrap dataset, a model can be refitted and the behaviour of the fit can be examined over the B replications. In each sample, an estimate of θ can be calculated, giving B estimates $\hat{\theta}^{*1}, \hat{\theta}^{*2}, ..., \hat{\theta}^{*B}$. The bootstrap estimate is then given by

$$\frac{1}{B}\sum_{r'=1}^{B}\hat{\theta}^{*r'}, \tag{2.2}$$

with standard error

$$\mathrm{SE}_B(\hat{\theta}) = \sqrt{\frac{1}{B-1}\sum_{r=1}^{B}\left(\hat{\theta}^{*r} - \frac{1}{B}\sum_{r'=1}^{B}\hat{\theta}^{*r'}\right)^2}\,. \tag{2.3}$$

The bootstrap procedure can be used to create bootstrap confidence intervals based on bootstrap percentiles from the distribution of a statistics, [18]. In many practical cases, a confidence interval of a statistics is more useful than a point estimate. The percentile interval is based on first generating B bootstrap datasets, denoted $\mathbf{x}^{*1}, \mathbf{x}^{*2}, ..., \mathbf{x}^{*B}$. For each bootstrap dataset, a bootstrap replication $\hat{\theta}^*(b) = s(\mathbf{x}^{*b}), b = 1, 2, ..., B$ is calculated, and the B replications of $\hat{\theta}^*$ are ordered. Then, the $B \cdot \alpha$th value of the ordered list equals the $100 \cdot \alpha$ empirical percentile of the $\hat{\theta}^*(b)$ values, denoted $\hat{\theta}_B^{*(\alpha)}$. Thus, an approximate $1 - 2\alpha$ percentile interval for $\hat{\theta}$ is given by

$$[\hat{\theta}_B^{*(\alpha)}, \hat{\theta}_B^{*(1-\alpha)}].$$

Good confidence intervals should give accurate cover probabilities in all situations. The percentile interval has been shown to be unpredictable and struggle with satisfactory coverage properties. There exist several methods to improve the percentile interval, such as the bias-corrected and accelerated method, called BC$_a$, or the approximate bootstrap confidence interval, denoted ABC, [18].

2.1.3 Evaluation Metrics for Classification

The generalization performance of a model is related to its prediction performance on independent test data, [21]. It is very important to assess this performance, since it can provide reasons for choosing one specific learning method over another. In this way, it can give a measure of the quality of the selected model. The Bias-Variance Trade-Off is an important aspect of the expected test error, since it involves three quantities: the variance of $\hat{f}$, the squared bias of $\hat{f}$, and the variance of the error terms ϵ. The goal is to select a statistical learning method that achieves low variance and low bias at the same time.

In classification problems, the response is quantitative, meaning it takes values in one of K different classes or categories. In the case of two classes, it is a binary classification problem. There exist several evaluation metrics for binary classification, many of which are based on the results from the confusion matrix. The confusion matrix is used to evaluate binary classifiers, [19]. It displays the count of correct and incorrect predictions in the two classes, [20]. There are four outcomes given a classifier and an instance. A true positive (TP) is a positive instance correctly classified as positive, and contrarily a false negative (FN) if it is classified as negative. A false negative is also known as a "Type-II error". A true negative (TN) is a negative instance correctly classified as negative, and contrarily a false positive (FP) if it is classified as positive. A false positive is also known as a "Type-I error". Then, a confusion matrix, which is a 2×2 matrix, can be constructed based on the dispositions of the set of instances. Figure 2.2 shows the confusion matrix with the four possible outcomes.

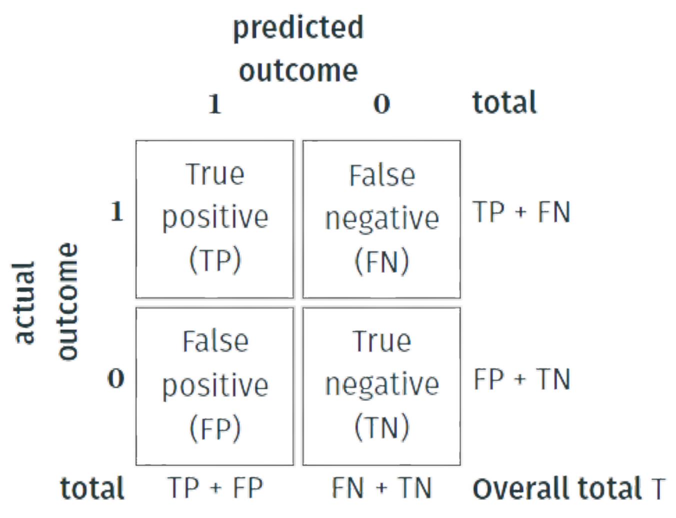


Figure 2.2: Confusion matrix in binary classification. The figure is taken from [25].

From the confusion matrix, several evaluation metrics can be calculated. The accuracy measures the total fraction of correct predictions,

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{T}}.$$

This is a symmetric measure and is useful if the costs of false negatives and false positives are equally high. The sensitivity or the true positive rate (TPR) is defined as the fraction of correct positive predictions to all actual positive outcomes,

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}.$$

Sensitivity is an important measure when the cost of false negatives is high and can tend to overpredict positive instances. Specificity or the true negative rate (TNR) is defined as the fraction of correct negative predictions to all actual negative outcomes,

$$\text{TNR} = \frac{\text{TN}}{\text{TN} + \text{FP}}.$$

In contrast to sensitivity, specificity is useful for recognizing the negative cases. When having imbalanced data, the minority class is often of interest. This is the case in many real-world problems, such as response modelling. In these situations, it is desirable to classify the positive cases correctly, that is obtaining a high score for sensitivity, while still maintaining an appropriate score for specificity. The false positive rate is given by $\text{FPR} = 1 - \text{TNR}$. The balanced accuracy (BACC) is a combined measure that takes both classes into account. It is defined as the average of sensitivity and specificity,

$$\text{BACC} = \frac{1}{2}(\text{Sensitivity} + \text{Specificity}) = \frac{1}{2}\left(\frac{\text{TP}}{\text{TP} + \text{FN}} + \frac{\text{TN}}{\text{TN} + \text{FP}}\right) \in [0, 1].$$

The balanced accuracy is useful for imbalanced data. A high value reflects a better classifier.

Matthews correlation coefficient (MCC), another combined measure, is defined as

$$\text{MCC} = \frac{\text{TP} \cdot \text{TN} - \text{FP} \cdot \text{FN}}{\sqrt{(\text{TP}+\text{FP})(\text{TP}+\text{FN})(\text{TN}+\text{FP})(\text{TN}+\text{FN})}} \in [-1, 1].$$

In the situation where the classifier is perfect, i.e. FP = FN = 0, then MCC = 1. This indicates perfect positive correlation. On the contrary, when the classifier always misclassifies, that is, TP = TN = 0, then MCC$= -1$. A score of MCC around 0 indicates a random classifier. Moreover, MCC is symmetric and takes both classes into account. A high MCC score reflects good scores in all outcomes of the confusion matrix. Another combined measure is the receiver operating characteristic (ROC) curve, which plots the trade-off between the false positive rate (FPR) and the true positive rate (TPR)/sensitivity, [49]. An illustration of four ROC curves is shown in Figure 2.3. A random classifier will correspond to a diagonal line from $(0, 0)$ to $(1, 1)$ or line D in the figure. In the situation where TPR $<$ FPR the classifier is worse than random. This corresponds to a curve below the diagonal line. Conversely, if TPR $>$ FPR the classifier is better than random. A better classifier is one that hugs the top left corner. Therefore, in the figure, curve B represents a better classifier than curve C. The point $(1, 0)$ corresponds to perfect classification. Accordingly, curve A corresponds to a perfect classifier. From the ROC curve, the Area under the ROC curve (AUC) can be calculated. For AUC, a random classifier will take the score 0.5, which corresponds to a classifier with line D, Figure 2.3. A better classifier will have an area under the curve larger than 0.5, corresponding to curves B and C, where B has a higher score than C. A perfect classifier will have a score of 1 corresponding to curve A. This means that AUC takes values in the interval $[0, 1]$. In this way, the AUC measures the overall performance of a classifier, but should not be used blindly. Two classifiers can achieve the same AUC but have different ROC curves, [49].

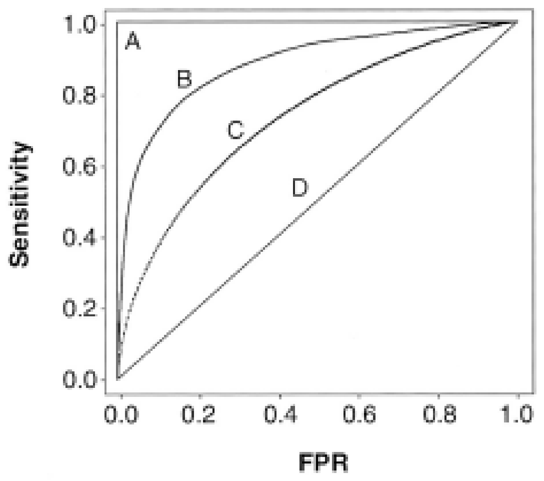


Figure 2.3: Illustration of 5 ROC curves. Curve A represents perfect classification, while curve D represents a random classifier. Curve B is a better classifier than curve C. Taken from [49]

2.1.4 Imbalanced Data

A binary classification task is imbalanced when one class is significantly larger than the other. This happens when the majority class (negative class) is larger than the minority class (positive class), [63]. Typical situations are medical diagnoses, spam filtering, or, as in this thesis, response modelling. There are several challenges to address when dealing with imbalanced data. There may be too little information available for the minority class to build a model. Several classification models can struggle to classify the minority class, due to the sparsity of information provided by this class, [27]. Standard classification algorithms will tend to be biased towards the majority class. Since more general rules are preferred, a classifier can often ignore rules that predict examples from the minority class. In this way, the minority class can be treated as noise, which leads to instances of the minority class being misclassified more often than those of the majority class, [20]. Thus, it is preferable with classifiers that are biased towards the minority class, but not at the expense of the accuracy over the majority class.

Furthermore, determining the most suitable performance metric is an important issue in the classification of imbalanced data, [38]. Accuracy can give misleading classification results for imbalanced data. This is known as the accuracy paradox, where a high score of accuracy may not be an indicator of a good classifier [59]. A classifier that classifies all instances as the majority class will achieve a high score, but will not reflect the classifiers failure to classify the minority class. Different metrics should therefore be considered, such as balanced accuracy or Matthews correlation coefficient, which both are combined measures that take both classes into account. In particular, Matthews correlation coefficient has been shown to be the best choice when classification errors are also considered, [38]. Another widely used metric is the AUC. However, this is also a disputed metric, since it has been shown to give misleading results, [36]. Often, the AUC summarizes the test performance over the whole region of the ROC space, and not only in the regions of interest.

2.2 Tree-Based Methods

Tree-based methods can be used for both regression and classification. The predictor space is segmented into smaller regions based on a set of splitting rules, which can be summarized in a tree [28]. This gives rise to decision tree methods. In concept, they are simple but still powerful. One of the advantages of tree methods is their interpretability. The tree can describe the partitioning of the feature space in full. Both Extreme Gradient Boosting and Random Forests discussed in this thesis are tree-based methods that originates from decision trees.

2.2.1 Decision Trees

Decision trees are the basis of tree-based methods, used in both regression and classification, [28]. In a classification tree, the response is qualitative, while in a regression tree, the response is quantitative. The process of growing classification and regression trees is quite similar, but due to the different type of response, there are some differences. First, when building a decision tree, the predictor space is divided into J distinct, non-overlapping regions. If the data consist of p predictors, the predictor space $X_1, X_2, ..., X_p$ is divided into regions $R_1, R_2, ..., R_J$ based on minimizing a criterion. In a regression tree, it is natural to define the response as the mean of the responses of the training observations in the same terminal node. Therefore, for regression trees, the regions are constructed by dividing the predictor space into boxes by minimizing the residual sum of squares (RSS),

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2, \tag{2.4}$$

where y_i is the response of instance i in region R_j and $\hat{y}_{R_j}$ is the mean response of training instances in the jth region. In a classification tree, each observation is classified as the most commonly occurring class of the training observations in the same terminal node. The residual sum of squares is not a suitable measure for classification; instead there are other alternative measures that can be used to divide the predictor space. An alternative to the RSS is the classification error rate, given by

$$E = 1 - \max_k(\hat{p}_{mk}), \tag{2.5}$$

where $\hat{p}_{mk}$ represents the proportion of training observations belonging to the kth class in the mth region. However, this measure is not sufficiently sensitive to build trees. Therefore, two other measures are preferable. The Gini index is defined as

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}), \tag{2.6}$$

where K is the number of classes and $\hat{p}_{mk}$ represents the class proportions. The Gini index can be seen as a measure of total variance across the K classes. Additionally, a small value indicates that a node contains significantly more observations from one single class, which leads to the Gini index being referred to as a measure of node purity. For binary classification, the Gini index is equal to $2p(1-p)$, where p is the proportion of the second class. The Gini index is suitable for numerical optimization since it is differentiable. Alternatively, the entropy can be used instead of the Gini index,

$$D = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}. \tag{2.7}$$

The Gini index and entropy are more sensitive to changes in the node probabilities than the misclassification rate and should be used in the tree growing.

To avoid considering every possible partition of the feature space, a greedy top-down approach called recursive binary splitting can be applied. From the top of the tree, the predictor space is successively split by two new branches down the tree. The best split is chosen in each tree-building step, making it a greedy approach. The best split is found by considering all predictors $X_1, X_2, ..., X_p$ and all possible cutpoints s for each of the predictors, and choosing the predictor and cutpoint that results in a tree with the lowest score for some criterion. More formally, the pair of half-planes for any j and s are defined as

$$R_1(j,s) = \{X|X_j < s\} \quad \text{and} \quad R_2(j,s) = \{X|X_j \geq s\}, \tag{2.8}$$

where the splitting variable j and the split point s are chosen to minimize a criterion. For regression trees, the RSS is used, i.e.

$$\sum_{x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2, \tag{2.9}$$

where $\hat{y}_{R_1}$ and $\hat{y}_{R_2}$ are the mean responses for the training instances in regions $R_1(j,s)$ and $R_2(j,s)$, respectively. For a classification tree, the Gini index or the entropy, Eq.(2.6) or Eq.(2.7), can be used to evaluate the splits. Next, the data are split further by minimizing the same criterion (RSS or Gini/entropy). This time only the two previously defined regions are considered. This continues until a stopping criterion is reached. After the J regions, $R_1, R_2, ..., R_J$, have been created, the response of a test observation is predicted. This is done either as the mean of training observations inside the region of the test observation for regression or as the most common class of training observations inside the region of the test observation for classification.

Performing the process above might lead to too complex trees, which are likely to overfit and have poor prediction performance. Therefore, tree pruning can be applied. A smaller tree can achieve lower variance and better interpretation at the cost of a little bias. A subtree can be constructed by growing a large tree T_0 and then pruning it. Cost complexity pruning or the weakest link pruning considers a sequence of trees given by α, a tuning parameter. There exists a subtree $T \subset T_0$, for each value of α, such that a cost-complexity criterion is minimized. For regression trees, the RSS is used, i.e.

$$\sum_{m=1}^{|T|} \sum_{x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha|T| \tag{2.10}$$

is minimized. The number of terminal nodes of the tree T is given by $|T|$, R_m is the region corresponding to the mth terminal node, and $\hat{y}_{R_m}$ is the predicted response in this region. For classification trees, it is typically the misclassification error, Eq.(2.5), that is used as a cost-complexity criterion for tree pruning. The Gini index or the entropy can also be used, but the classification error is known to lead to higher prediction accuracy. In the tree pruning, the trade-off between the fit to the training data and the complexity of the subtree is controlled by α. In the case where $\alpha = 0$, the subtree T is equal to the full tree T_0. The larger α gets, the larger the pruning gets. The value of α is usually set using cross-validation by minimizing the average prediction error over the folds.

There are some issues related to tree-based methods, [21]. Trees can exhibit high variance, and the final prediction can vary widely if there are small changes in the training data. This is due to the hierarchical nature of the process. If there is an error in the top split, this can propagate down to all the splits below. The trees can in this way be very non-robust. The variance can be reduced by using a more stable split criterion, but the instability is not removed. Bagging (Section 2.2.2) reduces variance by averaging many trees. In general, trees tend to not have the same level of predictive accuracy as other regression and classification approaches.

2.2.2 Ensemble Methods

Ensemble methods are approaches that combine simpler methods to obtain one single, potentially powerful, method, [28]. The predictions of a set of individual trained classifiers are combined in an ensemble, leading to more accurate predictions. The simpler classifiers are known as weak learners, since they alone may lead to worse predictions. Boosting and bagging are two methods for producing ensembles and thus improving the predictions.

Bagging

To counteract the possible high variance of decision trees, bootstrap aggregation or bagging can be applied to a tree-based model, [28]. It is frequently used in decision trees and can be very useful. The idea behind bagging is that averaging a set of observations reduces variance. This can be applied to prediction models. Since in general only one training set is available when building a statistical model, the bootstrap approach can be used (Section 2.1.2). Bagging involves creating B bootstrap samples and fitting a separate prediction tree in each of the samples, generating prediction $\hat{f}^{*b}$ for the bth bootstrap sample. This results in $\hat{f}^{*1}(x), \hat{f}^{*2}(x), ..., \hat{f}^{*B}(x)$ predictions fitted on the B separate bootstrap training sets. To obtain a low-variance statistical learning model, the predictions are averaged

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{*b}(x),$$

yielding the final bagging prediction. Normally, the number of trees, B, is not a critical parameter. A large value will not lead to overfitting. Bagging is a powerful procedure that has shown impressive improvements in accuracy by combining hundreds of trees.

Boosting

Boosting is an ensemble tree method that learns slowly and was initially designed for classification problems. In boosting, trees are grown sequentially using information from previous trees, [28]. The trees are fit, not on the response, but on the residuals given a current model. The new tree is then added to the fitted function, and the residuals are updated. Then, a new decision tree is fitted to the newly updated residuals. This leads to improvements in the model where it does not perform well. To further slow down the process, a shrinkage parameter λ can be introduced. This allows different shaped trees to attack the residuals. In this way, the building of new trees is highly dependent on the trees that have previously been grown, unlike bagging.

2.3 Random Forests

Random Forests, [9], is a popular machine learning algorithm that is easy to train and tune, [21]. Random Forests uses a modification of bagging (Section 2.2.2) to build a large collection of de-correlated trees which are averaged. Following the same procedure as bagging, multiple decision trees are built on bootstrap training samples, [28]. This reduces variance and leads to improved predictions as well as greater robustness to outliers. When trees are built, splits are made based on split candidates chosen from a random sample of m predictors from the set of all p predictors. A new random sample of m predictors is taken at each split. The splits are made until the minimum node size is reached. A conceptual illustration of Random Forests is shown in Figure 2.4. For classification, the node size is usually set to 1 and m is usually set to the square root of the total number of predictors, i.e. $m \approx \sqrt{p}$. In this way, Random Forests is not allowed to consider the majority of the predictors. This is beneficial, since in the presence of a very strong predictor, most of the bagged trees will use this predictor at the top of the split. Then, most of the trees will look quite similar, leading to highly correlated predictions from the bagged trees. Again, this will not substantially reduce the variance. When only a subset of the predictors are considered in the split, an average of $(p-m)/p$ splits will not consider the strong predictor. Then other predictors will be used in the splits, and as a result, the trees get de-correlated. Thus, the average of the trees will be less variable, and the results will be more reliable. In the case where $m = p$, Random Forests equals bagging. When dealing with a large number of correlated predictors, choosing a small value for m is helpful. Random Forests will not overfit as number of bagged trees, B, increases, because of the Law of Large Numbers. Instead, it will produce a limiting value of the generalization error, [9]. One of the advantages of Random Forests is its robustness to outliers and noise. Additionally, internal estimates can monitor error and correlations, and the algorithm can also be run in parallel because the trees are grown independently.

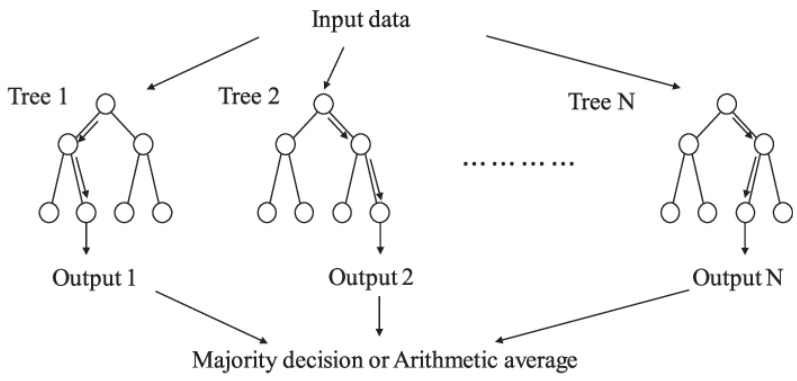


Figure 2.4: Illustration of the Random Forests algorithm. Taken from [54].

2.4 Extreme Gradient Boosting

2.4.1 Gradient Tree Boosting

Gradient tree boosting has been shown to give state-of-the-art results in many real-world problems, and is used for both classification and regression. Assume data with n observations and m features, that is $\{\mathbf{x}_i, y_i\}_{i=1}^n$, where $\mathbf{x}_i \in R^m, y_i \in R$, [13]. A tree ensemble model can predict an output based on K additive functions,

$$\hat{y}_i = \phi(\mathbf{x}_i) = \sum_{k=1}^{K} f_k(\mathbf{x}_k). \tag{2.11}$$

Each tree, f_k, has its own tree structure given by q and leaf weights $\mathbf{w}$. The tree structure is a mapping from the feature space to the corresponding leaf index, ($q : R^m \to T$), where T is the number of leaves in the tree. Each tree contains leaf weights, $\mathbf{w} \in R^T$. The tree structure represents the decision rules in the tree, which are used to classify a given observation into the leaves. The final prediction is calculated by summing the weights in the corresponding leaves, given by $\mathbf{w}$. The tree ensemble method is shown in Figure 2.5. In tree 1 the leaf weights are given by $\mathbf{w} = [2, 0.1, -1]^T$ and there are three leaves, that is, $T = 3$. For the boy, the mapping from the tree structure in tree 1 to the leaves is given by $q(\mathbf{x}) = 1$, that is the first of the three leaves. The predicted response of the boy will be the sum of the leaf weights of which the boy is part of in both trees, that is, $\hat{y}_{\text{boy}} = 2 + 0.9 = 2.9$.

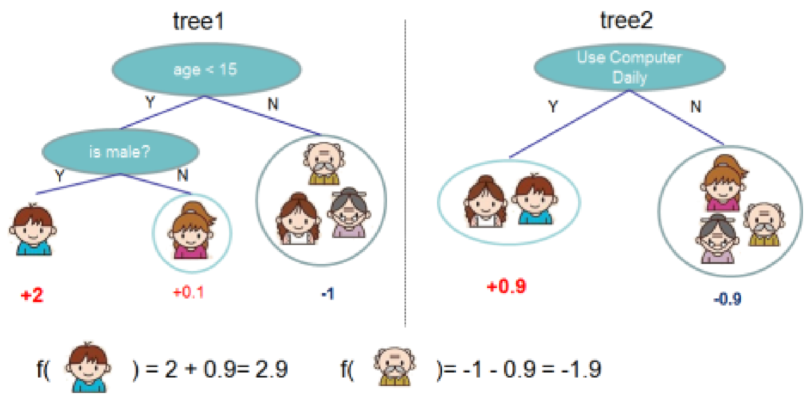


Figure 2.5: Illustration of the tree ensemble method. The sum of predictions from each tree yields the final prediction. Taken from [13].

To learn the functions given in Eq.(2.11), the following regularized objective is minimized

$$\mathrm{L}(\phi) = \sum_i l(y_i, \hat{y}_i,) + \sum_k \Omega(f_k), \tag{2.12}$$

where l is a convex loss function that measures the difference between the prediction $\hat{y}_i$ and the true response y_i. The second term Ω penalizes model complexity and is given by

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \|\mathbf{w}\|^2,$$

where γ is a user-defined penalty for pruning and λ is a regularization term, or shrinkage parameter, used to reduce the intensity of the model to individual observations. The model in Eq.(2.12) needs to be trained in an additive manner. Let the prediction of the ith observation in the tth iteration be given by $\hat{y}_i^{(t)}$. Then, a new prediction is equal to the previous prediction plus the output of the ith tree,

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(\mathbf{x}_i).$$

The regularized objective at step t is then equal to

$$\mathrm{L}^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(\mathbf{x}_i)) + \Omega(f_t). \tag{2.13}$$

Using a second-order Taylor approximation of the loss function, the regularized objective can be approximated by

$$\mathrm{L}^{(t)} \simeq \sum_{i=1}^{n} \left[l(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(\mathbf{x}_i) + \frac{1}{2} h_i f_t^2(\mathbf{x_i}) \right] + \Omega(f_t), \tag{2.14}$$

where

$$g_i = \frac{\partial l\left(y_i, \hat{y}_i^{(t-1)}\right)}{\partial \hat{y}_i^{(t-1)}} \quad \text{and} \quad h_i = \frac{\partial^2 l\left(y_i, \hat{y}_i^{(t-1)}\right)}{\partial \hat{y}_i^{(t-1)} \partial \hat{y}_i^{(t-1)}}.$$

Observe that the term $l(y_i, \hat{y}_i^{(t-1)})$ is independent of $f_t(\mathbf{x}_i)$ and does not influence the final output. Thus, by omitting this term, the objective is simplified to

$$\widetilde{\mathrm{L}}^{(t)} = \sum_{i=1}^{n} \left[g_i f_t(\mathbf{x}_i) + \frac{1}{2} h_i f_t^2(\mathbf{x_i}) \right] + \Omega(f_t). \tag{2.15}$$

The set of observations in leaf j can be defined as $I_j = \{i | q(\mathbf{x}_i) = j\}$, where q is the mapping from the tree structure to leaf j. The predicted value of all observations in I_j is given by $f_t(\mathbf{x}_i) = w_j$. Then, the objective can be written as

$$\begin{aligned} \widetilde{\mathrm{L}}^{(t)} &= \sum_{i=1}^{n} \left[g_i f_t(\mathbf{x}_i) + \frac{1}{2} h_i f_t^2(\mathbf{x_i}) \right] + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 \\ &= \sum_{j=1}^{T} \left[\left(\sum_{i \in I_j} g_i \right) w_j + \frac{1}{2} \left(\sum_{i \in I_j} h_i + \lambda \right) w_j^2 \right] + \gamma T. \end{aligned} \tag{2.16}$$

Given a fixed structure $q(\mathbf{x})$, the optimal weight w_j^* of leaf j is defined as

$$w_j^* = -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda}, \tag{2.17}$$

obtained by differentiating Eq.(2.16) with respect to w_j and setting it to zero. The optimal value of $\widetilde{\mathrm{L}}$ is then

$$\widetilde{\mathrm{L}}^{(t)}(q) = -\frac{1}{2}\sum_{j=1}^{T}\frac{\left(\sum_{i\in I_j} g_i\right)^2}{\sum_{i\in I_j} h_i + \lambda} + \gamma T. \tag{2.18}$$

This can be used as a scoring function for the tree structure q, similar to the node impurity measures for evaluating decision trees. This is because the optimal weight of each leaf is used in the scoring function. In this way, the quality of the tree can be measured. Note that this score is measured based on the optimal leaf weight calculated by Eq.(2.17). Figure 2.6 illustrates the calculation of the score, which requires only the scores of the gradient G and the second-order gradient H. Since it can be time consuming and often impossible to evaluate all possible tree structures q, a greedy algorithm can be applied instead. It starts from a single leaf and add branches iteratively. The split candidates are then evaluated by the loss reduction obtained from Eq.(2.18), multiplied by -1

$$\mathrm{L}_{\mathrm{split}} = \frac{1}{2}\left[\frac{\left(\sum_{i\in I_L} g_i\right)^2}{\sum_{i\in I_L} h_i + \lambda} + \frac{\left(\sum_{i\in I_R} g_i\right)^2}{\sum_{i\in I_R} h_i + \lambda} - \frac{\left(\sum_{i\in I} g_i\right)^2}{\sum_{i\in I} h_i + \lambda}\right] - \gamma, \tag{2.19}$$

where I_L and I_R represent the set of instances of the left and right nodes after the split, respectively, and $I = I_L \cup I_R$. This measure can be used to find the largest gain that yields the best split.

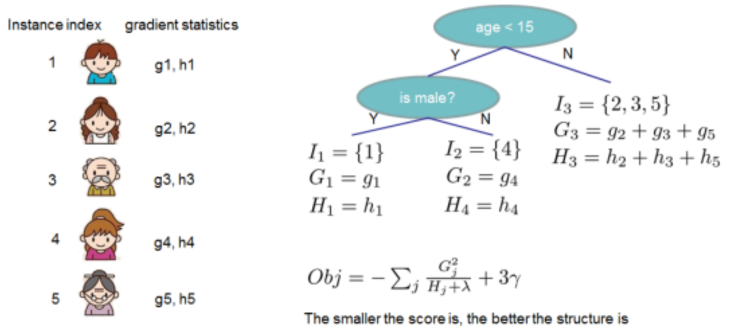


Figure 2.6: Scoring function for the ensemble tree method. Only the sum of the gradient and second-order gradient statistics on each leaf is needed. Taken from [13].

2.4.2 Extreme Gradient Boosting

Extreme gradient boosting (XGBoost), [13], is a boosting learning method that uses the gradient tree boosting method described in Section 2.4.1. XGBoost constructs an additive stage-wise model that creates new trees based on the residual errors of the previous ones. New trees are built until a maximum number of iterations is reached or the residuals are smaller than a given threshold. The tree structure is evaluated by minimizing Eq.(2.18) and the best splits are found by calculating the biggest gain, Eq.(2.19). The gain gives a measure of how much better the leaves fit the residuals than the root of the tree. A high gain reflects a better split. The tree can be pruned by choosing a threshold γ and calculating the difference between the threshold and the gain associated with the lowest branch of the tree. If the difference is negative, the branch is pruned. Conversely, if the branch is positive, it is kept. When the root is pruned, only the initial prediction is left, yielding extreme pruning. This is called the exact greedy algorithm, which can be computationally expensive.

Detailed information about XGBoost can be found in [13], which involves a split finding algorithm, a novel sparsity algorithm, and a weighted quantile sketch for approximate learning. In addition, the system design is described, such as column blocking for parallel learning and cache-aware access. In this way, XGBoost can solve real-world scale problems with a minimal amount of resources. XGBoost is known to be versatile and handle structured data well, but can have problems when handling sparse and dispersed data, [42]. Similarly to Random Forests, XGBoost reduces variance, but often also reduces bias due to the sequential approach by building decision trees based on the residuals of previous ones. Compared to Random Forests, XGBoost contains more adjustable hyperparameters, which requires more computational effort.

2.5 Hyperparameter Tuning

Most machine learning algorithms contain several hyperparameters that control the learning process, [37]. They decide the learning rate and control the behaviour of the model. In order to optimize the model performance, the hyperparameters should be tuned. Examples of hyperparameters are number of trees to be grown in a decision tree, the learning rate for training a neural network, or the fraction of columns to be used for training, among others. The process of optimizing the hyperparameter configuration and values is called hyperparameter tuning, [62]. Tuning the hyperparameters is very important, but actually understanding how the hyperparameters interact with the model performance can be hard. Hyperparameter optimization is an example of a black-box optimization problem, where it is often impossible to access the object of optimization analytically, [30]. There are several methods that can be applied in the tuning process. Grid search is the most widely used strategy for hyperparameter optimization, [6]. However, grid search can be highly inefficient and infeasible. Choosing the set of trials is a critical step in hyperparameter optimization. In grid search, a set of values needs to be decided for each variable. Then a set of trials is formed by assembling every possible combination of values. The number of joint values grows exponentially with the number of hyperparameters, leading to grid search suffering from the curse of dimensionality. However, there are some advantages to grid search. It is simple and easy to implement and is often reliable in low dimensional spaces. An alternative to grid search is random search, which possesses all the practical advantages of grid search. Random search draws values from a uniform density from the same configuration space that would be spanned by a regular grid. In high dimensional spaces, random search may be more efficient than grid search, in particular if the function to optimize has low effective dimensionality. A function of two variables that can be approximated by another function of one variable, $f(x, y) \approx g(x)$, is said to have a low effective dimensionality. The difference in how point grids and uniformly random point grids handle low effective dimensionality is demonstrated in Figure 2.7. The function $f(x, y) = g(x) + h(y) \approx g(x)$, where $g(x)$ is shown in green and $h(y)$ in yellow, has low effective dimensionality. As can be seen in the grid layout, the points are evenly spread out in the original space but inefficiently spread out in the subspace. Conversely, in the random grid the points are slightly less evenly distributed in the original space but more evenly distributed in the subspace. In this case, the optimal point may be found only in the random layout, probably by chance. In grid search, only three points of $g(x)$ are examined, while in random search all nine trials examine distinct points of $g(x)$. Usually, grid search fails in high dimensional hyperparameter optimization.

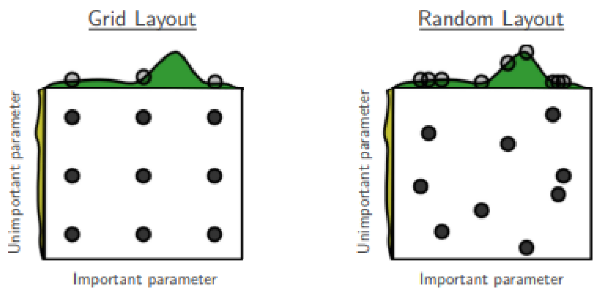


Figure 2.7: Illustration to show the difference between grid search and random search of nine trials for optimizing a function with low effective dimensionality. The function is $f(x, y) = g(x) + h(y) \approx g(x)$ where $g(x)$ is shown in green and $h(y)$ in yellow. Taken from [6].

However, random search has been shown to be unreliable, [37]. Instead, this thesis will focus on hyperparameter optimization through Design of Experiments and Response Surface Methodology. Design of Experiments is a systematic approach that can be used as a screening experiment to identify the important hyperparameters and in which configuration, [45]. Moreover, it can be used to perform the method of steepest ascent to move the experimental region closer to an optimum. Two-level designs are of great use in the process of tuning hyperparameters, but not for optimization. To further optimize hyperparameters, Response Surface Methodology can be applied. Another desirable aspect of hyperparameter optimization is that it should be automatic, such that the human effort required is reduced, [62]. A flexible and automatic approach for hyperparameter optimization is Bayesian optimization, which will also be discussed in this thesis.

2.5.1 Design of Experiments

The idea behind Design of Experiments (DoE) originates from the book *Design of Experiments* written by Ronald Aylmer Fisher in 1935, [58]. Fisher worked for a science institute focused on agriculture and soon discovered the many benefits of collecting data in a planned and controlled manner. In DoE, the main objective is to identify the most influential parameters on the response and in what configuration. This can be applied in a variety of situations and experiments. Moreover, DoE can be used in the context of tuning hyperparameters and is a systematic approach. It can be used as an initial screening to identify the most influential hyperparameters, and thus reduce the optimization problem to fewer hyperparameters, [37]. DoE involves selecting the variables, their levels, and the number of experiments to run. This is done in order to identify the relationship between the response and the factors.

2.5.2 Two-Level Factorial Design

In a two-level factorial design, each factor has two levels. These are often referred to as 2^k factorials where k represents the number of factors. Each factor has a low level, usually denoted -1, and a high level, usually denoted 1. In the context of tuning hyperparameters, the factors in the experiment correspond to the hyperparameters, and the levels are the values to be examined. The response is usually an evaluation metric evaluated on the model with the hyperparameter settings given in the design. Often, the response is calculated using cross-validation. The responses are denoted Y_i, $i = 1, 2, ..., n$, where n is the number of experiments, and y_i is the observed value. Since the purpose of two-level factorial design is to find the most influential factors and in which configuration, a regression model is fit to the response with the factors as covariates.

Consider an experiment with $k = 3$ factors, that is a 2^3 experiment. The full factorial design consists of 8 experiments, one for each of the configurations. The experiments should ideally be done in a random order to guarantee independent observations and to minimize the chance of non-influential factors affecting the response. When tuning hyperparameters, this is normally not an issue. The standard form of a general design for a 2^3 full factorial design is shown in Table 2.1. The 8 experiments are visualized in a sign matrix, with -1 corresponding to the low level and 1 to the high level. The design consists of three factors A, B, and C. The level codes represent which factor is on high level, with a meaning A is on high level, ab meaning A and B are on high level, and 1 meaning all factors are on low level. The observed responses $y_1, y_2, ..., y_8$ can be used to estimate the main effects and interaction effects of A, B, and C.

Table 2.1: Standard form of a general design for a 2^3 full factorial design, with factors A, B, and C.

Experiment number	A	B	C	AB	AC	BC	ABC	Level code	y
1	-1	-1	-1	1	1	1	-1	1	y_1
2	1	-1	-1	-1	-1	1	1	a	y_2
3	-1	1	-1	-1	1	-1	1	b	y_3
4	1	1	-1	1	-1	-1	-1	ab	y_4
5	-1	-1	1	1	-1	-1	1	c	y_5
6	1	-1	1	-1	1	-1	-1	ac	y_6
7	-1	1	1	-1	-1	1	-1	bc	y_7
8	1	1	1	1	1	1	1	abc	y_8

The main effect of a factor is defined as the expected average response when the factor is on the high level subtracted by the expected average response when the factor is on the low level, [58]. The main effect of A in the 2^3 experiment can be estimated from

$$\hat{A} = \bar{y}_{A_H} - \bar{y}_{A_L} = \frac{1}{4}\left(y_2 + y_4 + y_6 + y_8 - (y_1 + y_3 + y_5 + y_7)\right).$$

The interaction effect between two factors is defined as half of the main effect of the first factor when the second is on high level subtracted by half of the main effect of the first factor when the second is on low level. The interaction effect between A and B can be estimated as

$$\widehat{AB} = \frac{1}{4}\left(y_1 + y_4 + y_5 + y_8 - (y_2 + y_3 + y_6 + y_7)\right).$$

The effect of the three-factor interaction ABC, is defined as the average difference between the interaction AB for the two levels of C, i.e.

$$\widehat{ABC} = \frac{1}{4}\left(y_5 + y_3 + y_2 + y_8 - (y_1 + y_6 + y_4 + y_7)\right).$$

The above calculations assume only one replication of the experiment. When the experiment is done with replicates, the estimated main effects and interaction effects are simply the average of the estimates in each replicate. When a factor is not involved in any interactions, the estimated main effect is interpreted as the estimated change in expected response when the factor is moved from low to high level. If a factor is involved in an interaction, the effect of the factor is interpreted from the interaction effect and not the main effect. To further analyse the results of the 2^3 experiment, a linear regression model can be fit to the response,

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_{12} x_1 x_2 + \beta_{13} x_1 x_3 + \beta_{23} x_2 x_3 + \beta_{123} x_1 x_2 x_3 + \epsilon,$$

where ϵ is the error term assumed independent and approximately normal, $\epsilon_i \sim N(0, \sigma^2)$ for $i = 1, 2, ..., 8$. The regression coefficients, β's, are half of the estimated effects of the three factors. To ensure orthogonal factor columns, the covariates x_1, x_2, and x_3 correspond to the factors A, B, and C decoded in -1 and 1 (low and high level). To estimate a model independent variance, the experiment can be replicated. Assuming that two replicates of the experiment are performed, there are two responses for each level combination. Let y_{11} and y_{12} be the two observed responses for the first level combination. For each level combination, an estimator of the variance is

$$\begin{aligned}\sum_{j=1}^{2} (y_{1j} - \bar{y}_1)^2 &= \left(y_{11} - \frac{y_{11} + y_{12}}{2}\right)^2 + \left(y_{12} - \frac{y_{11} + y_{12}}{2}\right)^2 \\ &= \left(\frac{y_{11} - y_{12}}{2}\right)^2 + \left(\frac{-y_{11} + y_{12}}{2}\right)^2 = \frac{(y_{11} - y_{12})^2}{2}.\end{aligned}$$

The final estimator for the variance can in a 2^3 experiment be found by averaging the estimates for the 8 level combinations. More formally, an estimator for σ^2 for each i when doing $(m - 1)$ replicates is

$$\hat{\sigma}_i^2 = \sum_{j=1}^{m} \frac{(Y_{ij} - \bar{Y}_i)^2}{m - 1}.$$

Two-Level Fractional Factorial Design

In many situations, the higher order interaction effects of an experiment are not significant, [5]. Often, only the main effects and two-factor interactions are significant. In this case, all the relevant information can be obtained by running only a fraction of the number of treatments required in a complete factorial experiment. In this way, it is possible to include more than k factors in a 2^k experiment. In general, a $2^{(k-p)}$ fractional factorial design is a two-level design where k is the number of factors and $2^{(k-p)}$ denotes the number of treatment combinations to explore. A half fraction of a 2^k design is a $2^{(k-1)}$ design. This design can save computational cost and effort, since it is now only necessary to perform $2^{(k-1)}$ experiments and not 2^k. It is constructed by k factors, where the design column for the kth factor is constructed from the interaction column of the $k - 1$ other factors. This thesis will focus on a $2^{(5-1)}$ fractional factorial design, where the factors are denoted A, B, C, D, and E. The factor E can be expressed by the four-factor interaction $ABCD$, that is $E = ABCD$. This is called a design generator, where E and $ABCD$ have the same sign. The defining relation of the experiment can be expressed as

$$I = ABCDE,$$

where I is the identity element. Table 2.2 shows the design of a $2^{(5-1)}$ fractional design. Again, the level codes describe which factors are on high level.

Table 2.2: Standard form of the general design for a $2^{(5-1)}$ fractional factorial design, with factors A, B, C, D, and E.

Experiment number	A	B	C	D	E = ABCD	Level code
1	-1	-1	-1	-1	1	1
2	1	-1	-1	-1	-1	a
3	-1	1	-1	-1	-1	b
4	1	1	-1	-1	1	abe
5	-1	-1	1	-1	-1	c
6	1	-1	1	-1	1	ace
7	-1	1	1	-1	1	bce
8	1	1	1	-1	-1	abc
9	-1	-1	-1	1	-1	d
10	1	-1	-1	1	1	ade
11	-1	1	-1	1	1	bde
12	1	1	-1	1	-1	abd
13	-1	-1	1	1	1	cde
14	1	-1	1	1	-1	acd
15	-1	1	1	1	-1	bcd
16	1	1	1	1	1	abcde

In fractional factorial designs, there will always be effects that are not separable, meaning they are aliased, [58]. In some cases, they can both be significant but cancel each other. The resolution of a fractional design is defined as the length of the smallest word in the defining relation. For a fractional factorial design with resolution greater than or equal to five, all main and two-factor interaction effects can be estimated. For a $2^{(5-1)}$ design, the resolution is equal to V, which means that the main effects are aliased with four-factor interactions and the two-factor interactions are aliased with three-factor interactions. In general, it can be beneficial to select a design with as high resolution as possible, [17].

2.5.3 Response Surface Methodology

Response surface methodology (RSM) is a collection of techniques used to develop, improve, and optimize processes, [45]. It is particularly useful for optimizing hyperparameters and is most effective in situations where several input variables potentially influence the response. RSM is typically represented graphically, both through plots of the response surface and contour plots showing the relationship between the response and variables. An example of a response surface fitted to a chemical reaction is shown in Figure 2.8. The surface, Figure 2.8 a), shows the relationship between the yield of the response variable y and the two process variables, the reaction time ζ_1 and the reaction temperature ζ_2. For each value of ζ_1 and ζ_2 there is a corresponding response yield y. The response surface can be viewed in a two-dimensional time-temperature plane, Figure 2.8 b), where points with the same response value produce contour lines.

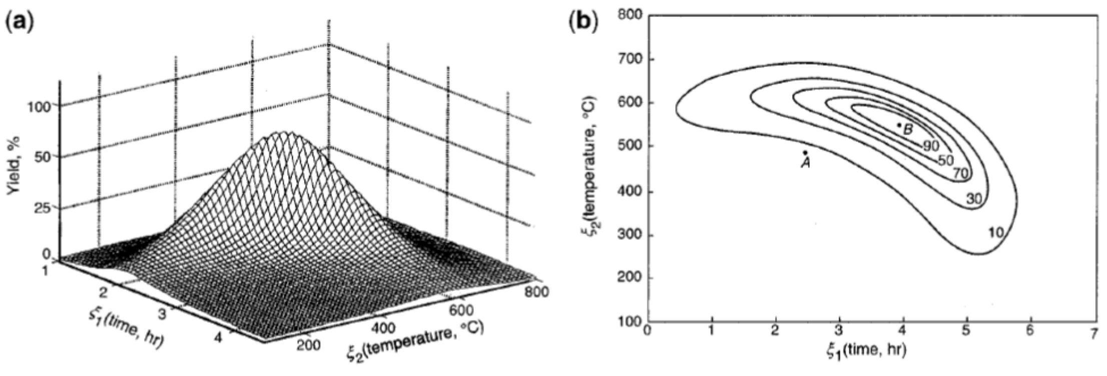


Figure 2.8: Graphic representation of a true response surface. a) The relationship between the yield of the response variable y and the two process variables, reaction time ζ_1 and reaction temperature ζ_2. b) Contour plot of the two-dimensional time-temperature plane. Taken from [45].

In general, the relationship between the response and the independent variables is given by

$$y = f(x_1, x_2, ..., x_k) + \epsilon, \tag{2.20}$$

where y represents the response, $x_1, x_2, ..., x_k$ the k independent variables, and ϵ the error not accounted for in f. The response function is unknown and must be approximated, usually through a first-order or a second-order model. If a first-order model is suitable, the relationship takes the form

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_k x_k + \epsilon. \tag{2.21}$$

This is often referred to as a main effects model, since it only includes the main effects of the variables. The interactions can be easily added. Often, a first-order model is inadequate if the true response surface contains a strong curvature. Then a second-order model can be applied

$$y = \beta_0 + \sum_{i=1}^{k} \beta_i x_i + \sum_{i=1}^{k} \beta_{ii} x_i^2 + \sum_{i<j}\sum \beta_{ij} x_i x_j + \epsilon. \tag{2.22}$$

The second-order model is known to be flexible and often works well in solving real response surface problems, [45]. Both models are expected to yield good approximations in a small region around the independent variables. The parameters (the β's) can be estimated using the least squares method. The data used in RSM should be collected using a good experimental design. There are several types of response surface designs. The central composite design (CCD) and the Box-Behnken design (BBD) are widely used. This thesis will focus on the CCD.

RSM usually follows a sequential procedure. Initially, a screening experiment is performed to investigate which variables are important, often assuming a first-order model (DoE). The objective of factor screening is to reduce the number of variables, which leads to more efficient experiments that require fewer runs. The next phase of RSM is to determine if the current levels of the variables result in a response value near an optimum. If the levels are not compatible with an optimum, the process must be moved towards the optimum. The method of steepest ascent, can be applied to move the experimental region towards an optimum. Once the process is near the optimum, a more sophisticated model can be applied. Near the optimum, the true response surface usually exhibits curvature, and a second-order model should be applied. This sequential process is performed within the operability region, which is some region of the variable space.

Method of Steepest Ascent

Once a model is fitted to the data from the experiment, the next step is to search for a new region in which the response is improved. When performing Design of Experiment, the design is often based on an educated guess or preliminary experiments. The method of steepest ascent can be used as a next step to find a new region, which can potentially improve the response, [45]. This is a gradient-based first-order optimization technique. The method of steepest ascent moves sequentially in the direction of the maximum increase in the response. Conversely, the path of steepest descent is followed if the minimum response is required. A first-order model is fitted using an orthogonal design, given by

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + ... + \hat{\beta}_k x_k,$$

where the contours are series of parallel lines. Then, the path of steepest ascent is the path where the expected increase in the response $\hat{y}$ is at its maximum, given a distance r from the center of the design. The center point of the design is coded as $(0, 0, ..., 0)$. This can be formulated as a constrained optimization problem

$$\max_{x_1, x_2, ..., x_x} \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + ... + \hat{\beta}_k x_k \quad \text{s.t.} \quad \sum_{i=1}^{k} x_i^2 = r^2, \tag{2.23}$$

where the constraint $\sum_{i=1}^{k} x_i^2 = r^2$ corresponds to a sphere with radius r. The solution to this problem uses Lagrange multipliers. The Lagrange function is equal to

$$L = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + ... + \hat{\beta}_k x_k - \lambda \left(\sum_{i=1}^{k} x_i^2 - r^2 \right), \tag{2.24}$$

where λ is the Lagrange multiplier. To find the maximum, the partial derivatives with respect to $x_j, j = 1, ..., k$ of the Lagrange function must be calculated,

$$\frac{\partial L}{\partial x_j} = \hat{\beta}_j - 2\lambda x_j \quad \text{for} \quad j = 1, 2, ..., k. \tag{2.25}$$

Setting the derivatives to zero yields the coordinate of x_j of the path of steepest ascent

$$x_j = \frac{\hat{\beta}_j}{2\lambda} \quad \text{for} \quad j = 1, 2, ..., k. \tag{2.26}$$

The quantity $1/(2\lambda) = \rho$ is called a constant of proportionality and is up to the experimenter to decide. The coordinates of the steepest ascent are thus given by

$$x_1 = \rho\hat{\beta}_1, \quad x_2 = \rho\hat{\beta}_2, \quad ... \quad , x_k = \rho\hat{\beta}_k. \tag{2.27}$$

An illustration of the path of steepest ascent is shown in Figure 2.9. Along the path, experimental runs are conducted, hopefully showing improvements. Experimentation along the path is continued until the improvements decline. Along the path, a point can be used as an approximation for a maximum, and this creates a base for a second region for experimenting. Again, a first-order design should be fitted. At this point it is important to test the curvature and check the lack of fit. Assume there are n_i observations of the response at level combination i of the factors $\mathbf{x}_i$, $i = 1, 2, ..., m$, [45]. The jth response at $\mathbf{x}_i$ is denoted y_{ij}, $i = 1, 2, ..., m$ and $j = 1, 2, ..., n_i$. The test statistic for the lack of fit constists of two parts, the pure error sum of squares and the sum of squares for lack of fit. The pure error sum of squares is given by

$$\mathrm{SS}_{\mathrm{PE}} = \sum_{i=1}^{m} \sum_{j=1}^{n_i} \left(y_{ij} - \bar{y}_i\right)^2 ,$$

where the average of the n_i observations at $\mathbf{x}_i$ is given by $\bar{y}_i$. This measure gives a model independent measure of the pure error. The sum of squares for the lack of fit equals

$$\mathrm{SS}_{\mathrm{LoF}} = \sum_{i=1}^{m} n_i \left(\bar{y}_i - \hat{y}_i\right)^2 ,$$

where $\hat{y}_i$ is the fitted response at level combination i. When the fitted values $\hat{y}_i$ are close to the average responses $\bar{y}_i$, the regression function is close to linear. The test statistic of the lack of fit is

$$F_0 = \frac{\mathrm{SS}_{\mathrm{LoF}}/(m-p)}{\mathrm{SS}_{\mathrm{PE}}/(n-m)},$$

where $n = \sum_{i=1}^{m} n_i$ and $p = k+1$ is the number of the model parameters. When the true regression function is linear, the test statics is assumed to follow the $F_{m-p,n-m}$ distribution. Hence, in the case where $F_0 > F_{m-p,n-m}$, the regression function is not linear and the lack of fit is significant, [45]. A second path based on the new model is calculated if the lack of fit is not significant. Often, this is referred to as a mid-course correction. Desirably, the response is near its optimum after some rounds of experiments. Here, more sophisticated experiments can be conducted. This means typically fitting a second-order model in the vicinity of the optimum.

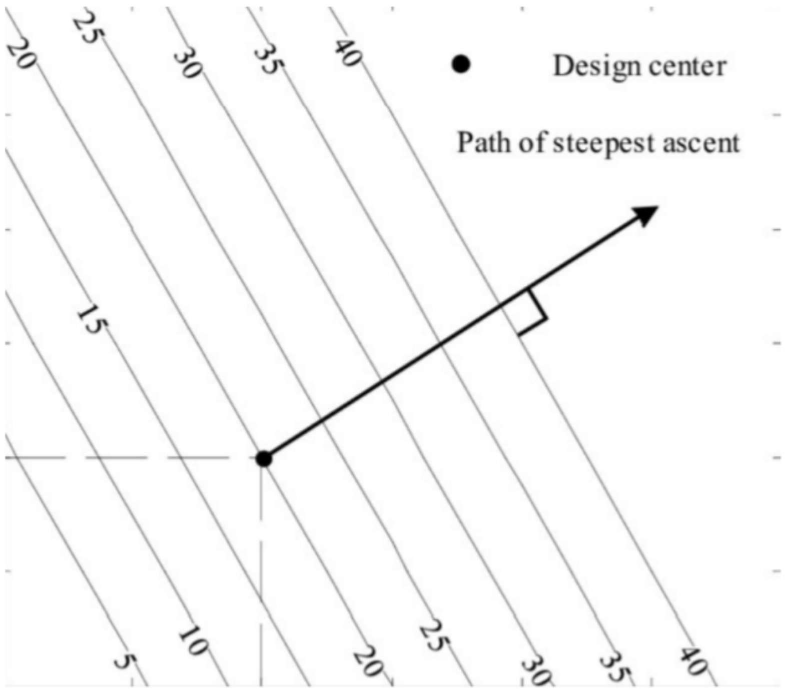


Figure 2.9: Illustration of the path of steepest ascent. The contours refer to the expected response. Taken from [44].

Central Composite Design for fitting Second-Order Response Surfaces

Recall that a second-order response surface model is given by Eq.(2.22). This model contains $1+2k+k(k-1)/2$ parameters, [45]. Consequently, the experimental design for second-order models must contain at least this number of runs. Each design point must have at least three levels. For first-order designs, the main property is orthogonality, while this ceases to be important in second-order designs. The central composite designs (CCD), [8], are the most popular class of second-order designs, [45]. The motivation behind CCD comes from its use in sequential experimentation. The CCD combines a two-level factorial design (2^k) or a fractional factorial resolution V design with $2k$ axial or star points. In this way, the design involves F factorial points, $2k$ axial points, and n_c center runs, and allows the estimation of main effects, two-factor interaction effects, and quadratic effects. The center runs provide information about the curvature in the system. They also provide an internal error estimate and can be used to estimate the lack of fit. The total number of runs is then $n_c + 2k + n_F$, where n_F is the number of factorial runs. The axial distance from the center to the axial points is usually given by α. A CCD design with three variables, axial distance α, and one center run is shown in Table 2.3. The choices of n_c and α are important for the design. Usually, CCDs are rotatable or spherical. If the desired region of the design is spherical, it is encouraged to use $\alpha = \sqrt{k}$ and three to five center runs. This will give an effective CCD from a variance point of view. An example of a CCD design with three factors x_1, x_2, and x_3 and $\alpha = \sqrt{3}$ is shown in Figure 2.10.

Table 2.3: A central composite design for three variables x_1, x_2, and x_3 with one center run and axial distance α.

x_1	x_2	x_3
1	1	1
1	1	-1
1	-1	1
1	-1	-1
-1	1	1
-1	1	-1
-1	-1	1
-1	-1	-1
α	0	0
-α	0	0
0	α	0
0	-α	0
0	0	α
0	0	-α
0	0	0

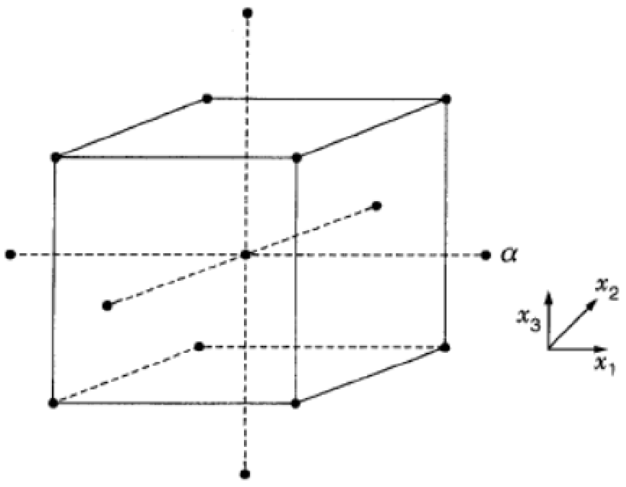


Figure 2.10: Central composite design for three factors x_1, x_2, and x_3 with $\alpha = \sqrt{3}$. Taken from [45].

Canonical analysis of Second-Order Response Surfaces

The second-order response surface model, Eq.(2.22), is a flexible approximation to describe data with a curvature. A canonical analysis of the response surface can be performed to learn more about the properties of the second-order surface. Additionally, it is used to locate the stationary point of the surface and identify the type. This is where the response $\hat{y}$ is optimized and the partial derivatives $\partial\hat{y}/\partial x_1, \partial\hat{y}/\partial x_2, ..., \partial\hat{y}/\partial x_k$ are all equal to zero. The stationary point can be either a maximum, minimum, or saddle point. By looking at the contour plots of the response surface, the stationary point can be identified. However, in some situations, it can be difficult to interpret the stationary point, for instance in the case where there are more than two factors. Furthermore, the contours plots represent the contours of the estimated response and not the true response. The system itself is part of the estimation process, and the stationary point does not reflect the true structure but rather the fitted model. Therefore, a more formal analytical approach can be applied to identify the stationary point, [45]. Eq.(2.22) can be written in matrix notation

$$\hat{y} = \hat{\beta}_0 + \mathbf{x}^T\mathbf{b} + \mathbf{x}^T\mathbf{B}\mathbf{x}, \tag{2.28}$$

where $\hat{\beta}_0$, $\mathbf{b}$, and $\mathbf{B}$ are the estimates of the intercept, linear, and second-order coefficients, respectively. That is, $\mathbf{x} = (x_1, x_2, ..., x_k)^T$, $\mathbf{b} = (\hat{\beta}_1, \hat{\beta}_2, ..., \hat{\beta}_k)^T$, and $\mathbf{B}$ is the $k \times k$ symmetric matrix

$$\mathbf{B} = \begin{bmatrix} \hat{\beta}_{11} & \hat{\beta}_{12}/2 & \cdot & \cdot & \cdot & \hat{\beta}_{1k}/2 \\ \hat{\beta}_{12}/2 & \hat{\beta}_{22} & \cdot & \cdot & \cdot & \hat{\beta}_{2k}/2 \\ \cdot & \cdot & \cdot & & & \cdot \\ \cdot & \cdot & & \cdot & & \cdot \\ \cdot & \cdot & & & \cdot & \cdot \\ \hat{\beta}_{1k}/2 & \hat{\beta}_{2k}/2 & \cdot & \cdot & \cdot & \hat{\beta}_{kk} \end{bmatrix}. \tag{2.29}$$

The stationary point can then be found by differentiating $\hat{y}$ in Eq.(2.28) with respect to $\mathbf{x}$,

$$\frac{\partial\hat{y}}{\partial\mathbf{x}} = \mathbf{b} + 2\mathbf{B}\mathbf{x}.$$

Setting the derivative equal to zero yields the stationary point

$$\mathbf{x}_s = -\frac{1}{2}\mathbf{B}^{-1}\mathbf{b}, \tag{2.30}$$

where the corresponding predicted response equals

$$\hat{y}_s = \hat{\beta}_0 + \frac{1}{2}\mathbf{x}_s^T\mathbf{b}. \tag{2.31}$$

The signs of the eigenvalues of the matrix **B** determine the nature of the stationary point. However, the second-order model first has to be transformed to the canonical form, since the relative magnitudes of the eigenvalues are helpful in the interpretation. The model, Eq.(2.28), is translated to a new coordinate system, where the center is at the stationary point. The axes are rotated corresponding to the principal axes of the contour system or the eigenvectors of **B**. This yields

$$\hat{y} = \hat{y}_s + \sum_{i=1}^{k} \lambda_i w_i^2, \tag{2.32}$$

where $\hat{y}_s$ is the estimated response at the stationary point and $\lambda_1, \lambda_2, ..., \lambda_k$ are the eigenvalues of **B**. The variables $w_1, w_2, ..., w_k$ are the canonical variables, given by

$$\mathbf{w} = \mathbf{P}^T(\mathbf{x} - \mathbf{x}_s),$$

where **P** is the $k \times k$ orthogonal matrix corresponding to the normalized eigenvectors associated with the eigenvalues of **B**. That is

$$\mathbf{P}^T\mathbf{B}\mathbf{P} = \mathbf{\Lambda},$$

where $\mathbf{\Lambda}$ is a diagonal matrix that contains the eigenvalues of **B**. The canonical form of a second-order model is shown in Figure 2.11.

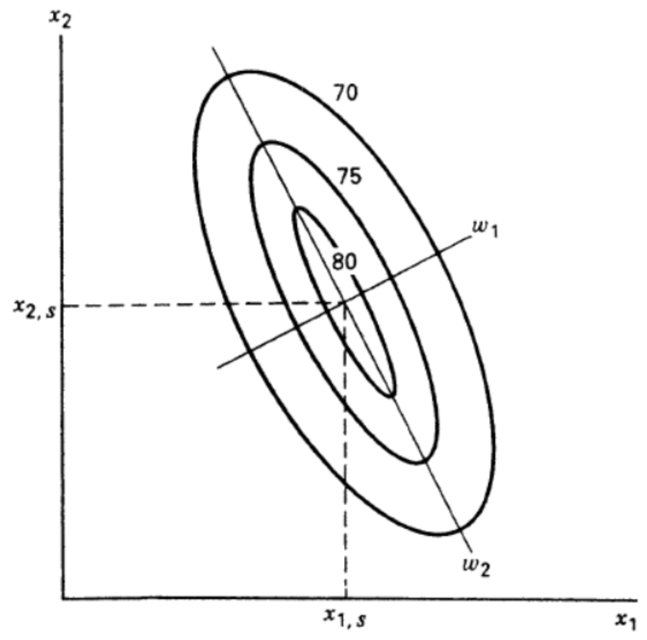


Figure 2.11: The canonical form of a second-order model of two variables x_1 and x_2. Taken from [45].

The signs of the eigenvalues, λ's, determine the type of the stationary point $\mathbf{x}_s$. If all the eigenvalues are negative, the stationary point is a maximum. If all the eigenvalues are positive, the stationary point is a minimum. Lastly, if the eigenvalues are mixed in signs, the stationary point is a saddle point. The magnitude of the eigenvalues can provide information about the sensitivity of the response with respect to the design factors. The eigenvectors to the corresponding eigenvalues can help deciding which direction to explore further if the stationary point turns out to be a saddle point or is far away from the center of the design. A linear path should be followed in the direction of increasing response or decreasing response when looking for a minimum. The function `canonical.path()` in R finds a linear path that originates at the stationary point and moves along the direction of the eigenvector with largest positive λ if a maximum is sought, and contrarily, the largest negative λ if a minimum is sought.

2.5.4 Direct Variance Modeling

Direct variance modeling arises from a classical paper, [2], claiming that variance can be modelled directly when some design points are replicated, [45]. Without violating assumptions, a log-linear model can be fit to the sample variance s_i^2 from n_i observations at each of the d design points,

$$\log(s_i^2) = \mathbf{x}_i^T \boldsymbol{\gamma} + \epsilon_i^*, \quad \text{for} \quad i = 1, 2, ..., d, \tag{2.33}$$

where ϵ_i^* is the error term. Here, $\mathbf{x}_i^T \boldsymbol{\gamma}$ describes a linear model in a set of the design variables. The model given by Eq.(2.33) has approximately normal errors with constant variance, if the errors of the response variables are normal around the mean model. That is if

$$y_{ij} = \mathbf{x}_i^T \boldsymbol{\beta} + \epsilon_{ij}\sigma_i, \quad \text{for} \quad i = 1, 2, ..., d, \quad j = 1, 2, ..., n,$$

where $\epsilon_{ij} \sim N(0, 1)$ and $\log(\sigma_i^2) = \mathbf{x}_i^T \boldsymbol{\gamma}$, [2]. The variance reflects within-run variance, which is of interest to the experimenter when the replicates of the design show variability. In addition, a log-linear model often reduces the effects of curvature and interactions, making the results easier to interpret.

2.5.5 Bayesian optimization

An appealing idea is to develop automatic methods for optimizing machine learning hyperparameters, [55]. Bayesian optimization can be used for automatic tuning and has been shown to be highly effective when dealing with computationally expensive functions, such as many machine learning algorithms, [61]. Moreover, it can be applied in situations where the derivatives are hard to evaluate, or the function is non-convex. The maximum value at a sampling point for an unknown function f is given by

$$\mathbf{x}^* = \underset{\mathbf{x} \in A}{\operatorname{argmax}} f(\mathbf{x}),$$

where A denotes the search space for $\mathbf{x}$. Bayesian optimization is based on Bayes' theorem, that is

$$P(M|E) \propto P(E|M)P(M),$$

where $P(M|E)$ is the posterior probability of a model M given some evidence data E. $P(E|M)$ is the conditional likelihood of observing E given a model M and $P(M)$ is the prior probability of the model. Bayesian optimization works in a sequential manner that combines the prior distribution of the function $f(\mathbf{x})$ with sample information to obtain the posterior. This information is used to find the maximum of the function $f(\mathbf{x})$ according to a utility function a, also called the acquisition function. The prior distribution is updated continuously as new data are gathered and the maximum of the acquisition function from the resulting posterior distribution determines the next point to evaluate. The entire process is repeated until the maximum number of iterations is reached or the difference between the current value and the optimal value is less than a given threshold. The sampling area is searched, and it is important to consider both exploration and exploitation, which is sampling from areas with high uncertainty and sampling from high values, respectively. High exploration can lead to an inefficient search, while high exploitation can lead to local optimization and missing the global optimum, [30].

There are two important choices to be made in Bayesian optimization: the prior and the acquisition function. The prior must be decided, specifying assumptions made about the function being optimized, [55]. In general, a Gaussian process is assumed to be well suited for the prior distribution of Bayesian optimization, because of its flexibility and tractability. The Gaussian process is stochastic and assumes that any finite sub-collection of random variables has a multivariate Gaussian distribution. Assuming that there are D hyperparameters to optimize, the data are given by $\mathbf{x} \in R^D$. The process is defined by its mean function $m(\mathbf{x})$ and positive definite covariance function k, that is

$$f(\mathbf{x}) \sim \mathrm{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')).$$

Initially the expectation is set to zero, that is, $m(\mathbf{x}) = 0$, [61]. Suppose that t observations are sampled from the training set, that is $\{\mathbf{x}_i, f(\mathbf{x}_i)\}_{i=1}^t$, such that

$$f(\mathbf{x}_{1:t}) \sim \mathrm{N}(0, \mathbf{K}),$$

where $\mathbf{x}_{1:t} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_t]^T$ and

$$\mathbf{K} = \begin{bmatrix} k(\mathbf{x}_1,\mathbf{x}_1) & k(\mathbf{x}_1,\mathbf{x}_2) & \cdot\ \cdot\ \cdot & k(\mathbf{x}_1,\mathbf{x}_t) \\ k(\mathbf{x}_2,\mathbf{x}_1) & k(\mathbf{x}_2,\mathbf{x}_2) & \cdot\ \cdot\ \cdot & k(\mathbf{x}_2,\mathbf{x}_t) \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ k(\mathbf{x}_t,\mathbf{x}_1) & k(\mathbf{x}_t,\mathbf{x}_2) & \cdot\ \cdot\ \cdot & k(\mathbf{x}_t,\mathbf{x}_t) \end{bmatrix}. \tag{2.34}$$

The function value at the next sample point $\mathbf{x}_{t+1}$ and the previous observations $\mathbf{f}_{1:t} = [f(\mathbf{x}_1), f(\mathbf{x}_2), ..., f(\mathbf{x}_t)]^T$ follow the $t+1$ dimensional joint normal distribution, that is

$$\begin{bmatrix} \mathbf{f}_{1:t} \\ f(\mathbf{x}_{t+1}) \end{bmatrix} = \mathrm{N}\left(0, \begin{bmatrix} \mathbf{K} & \mathbf{k} \\ \mathbf{k}^T & k(\mathbf{x}_{t+1}, \mathbf{x}_{t+1}) \end{bmatrix}\right), \tag{2.35}$$

where

$$\mathbf{k} = [k(\mathbf{x}_{t+1}, \mathbf{x}_1), k(\mathbf{x}_{t+1}, \mathbf{x}_2), ..., k(\mathbf{x}_{t+1}, \mathbf{x}_t)]^T.$$

By the properties of the joint normal distribution, the expectation and variance at point $t+1$ given the previous observations $\mathbf{f}_{1:t}$ are

$$\mu_{t+1}(\mathbf{x}_{t+1}) = \mathbf{k}^T \mathbf{K}^{-1} \mathbf{f}_{1:t} \tag{2.36}$$

and

$$\sigma^2_{t+1}(\mathbf{x}_{t+1}) = -\mathbf{k}^T \mathbf{K}^{-1} \mathbf{k} + k(\mathbf{x}_{t+1}, \mathbf{x}_{t+1}). \tag{2.37}$$

Thus, the posterior distribution is Gaussian with

$$f(\mathbf{x}_{t+1})|\mathbf{f}_{1:t} \sim N\left(\mu_{t+1}(\mathbf{x}_{t+1}), \sigma^2_{t+1}(\mathbf{x}_{t+1})\right).$$

The appropriate choice of covariance function k for the Gaussian process can often be unclear in many practical problems, [55]. Often, the squared exponential kernel is a default choice for Gaussian process regression,

$$k_{\mathrm{SE}}(\mathbf{x}, \mathbf{x}') = \theta_0 \exp\left\{-\frac{1}{2}r^2(\mathbf{x}, \mathbf{x}')\right\}, \quad r^2(\mathbf{x}, \mathbf{x}') = \sum_{d=1}^{D}(x_d - x'_d)^2/\theta_d^2, \tag{2.38}$$

where $\mathbf{x} \in R^D$, D is the number of hyperparameters to optimize, θ_0 is the covariance amplitude, and θ_1, θ_2, ..., θ_D are length scales. This gives a smooth covariance function. However, for practical optimization problems, this covariance function can be unrealistically smooth for the sample function. Instead, the Matérn 5/2 kernel can be used

$$k_{\mathrm{M52}}(\mathbf{x}, \mathbf{x}') = \theta_0\left(1 + \sqrt{5r^2(\mathbf{x}, \mathbf{x}')} + \frac{5}{3}r^2(\mathbf{x}, \mathbf{x}')\right)\exp\left\{-\sqrt{5r^2(\mathbf{x}, \mathbf{x}')}\right\}, \tag{2.39}$$

which is twice differentiable. After the choice of covariance, the hyperparameters that govern the Gaussian process must be tuned, referred to as hyper-hyperparameters. Note that these are not the same hyperparameters that are subject to the overall optimization. This involves the D length scales $\theta_1, \theta_2, ..., \theta_D$ and the covariance amplitude θ_0. The most common approach is to use maximum likelihood estimation. A sample of points from f is initially used to find the hyper-hyperparameter values that maximize the probability of observing those points in the Gaussian process model. Let $\boldsymbol{\theta} = [\theta_0, \theta_1, ..., \theta_D]$ and consider the observed points $\mathbf{f}_{1:t}$. The hyper-hyperparameter values are then selected based on maximising the probability calculated within the Gaussian process model,

$$\boldsymbol{\theta} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\ P(\mathbf{f}_{1:t}).$$

However, it is desirable to marginalize over the hyper-hyperparameters and instead compute the integrated acquisition function. Then, it is said to be a fully-Bayesian treatment of hyper-hyperparameters. That is

$$\begin{aligned}
\hat{a}(\mathbf{x}) &= \int a(\mathbf{x}; \boldsymbol{\theta})P(\boldsymbol{\theta}|\mathbf{f}_{1:t})d\boldsymbol{\theta} \\
&= \int a(\mathbf{x}; \boldsymbol{\theta})\frac{P(\boldsymbol{\theta}) \cdot P(\mathbf{f}_{1:t}|\boldsymbol{\theta})}{P(\mathbf{f}_{1:t})}d\boldsymbol{\theta} \\
&\propto \int a(\mathbf{x}; \boldsymbol{\theta})P(\boldsymbol{\theta})P(\mathbf{f}_{1:t}|\boldsymbol{\theta})d\boldsymbol{\theta},
\end{aligned}$$

where $a(\mathbf{x}; \boldsymbol{\theta})$ is the value of the acquisition function when the hyper-hyperparameters are set to $\boldsymbol{\theta}$. To compute the integrated acquisition function, a prior over $\boldsymbol{\theta}$ needs to be set.

Once the posterior distribution of the objective function is obtained, the next point is found by maximizing the acquisition function, i.e.

$$\mathbf{x}_{\mathrm{next}} = \underset{\mathbf{x}}{\operatorname{argmax}}\ a\left(\mathbf{x}|\{\mathbf{x}_{1:t}, \mathbf{f}_{1:t}\}\right).$$

The acquisition function is given by $a : \mathbf{x} \to \mathbb{R}^+$ and depends on previous observations, as well as the Gaussian process hyper-hyperparameters. It should be chosen such that a high value of the acquisition function corresponds to a high value of the objective function. The choice of the appropriate acquisition function is often not clear a priori. In general, there are two types of strategies, improvement based strategies and information based strategies. The acquisition function must satisfy the trade-off between exploitation and exploration. High exploitation means that points with higher posterior mean are preferred, while high exploration means that points with higher posterior variance are preferred [40]. The function $f(\mathbf{x})$ is assumed to be drawn from a Gaussian process with mean $\mu(\mathbf{x})$ defined as in Eq.(2.36) and variance $\sigma^2(\mathbf{x})$ defined as in Eq.(2.37). The maximum value of the true objective function so far is denoted $f(\mathbf{x}^+)$, where

$$\mathbf{x}^+ = \underset{\mathbf{x} \in \mathbf{x}_{1:t}}{\operatorname{argmax}} \, f(\mathbf{x}).$$

There are two types of acquisition functions that are improvement based, Probability of Improvement (PI) and Expected Improvement (EI). Probability of Improvement was historically the first acquisition function proposed, [32], and is based on the intuitive idea of maximizing the probability of improvement over the best current value, [61]. In this way, it measures the probability that the next point is better than the previous best point, that is

$$a_{\mathrm{PI}}(\mathbf{x}) = P\left(f(\mathbf{x}) > f(\mathbf{x}^+)\right).$$

Since $f(\mathbf{x})$ is Gaussian, a closed-form expression can be obtained

$$\begin{aligned} a_{\mathrm{PI}}(\mathbf{x}) &= 1 - P\left(f(\mathbf{x}) \leq f(\mathbf{x}^+)\right) = 1 - P\left(\frac{f(\mathbf{x}) - \mu(\mathbf{x})}{\sigma(\mathbf{x})} \leq \frac{f(\mathbf{x}^+) - \mu(\mathbf{x})}{\sigma(\mathbf{x})}\right) \\ &= 1 - \Phi\left(\frac{f(\mathbf{x}^+) - \mu(\mathbf{x})}{\sigma(\mathbf{x})}\right) = \Phi\left(\frac{\mu(\mathbf{x}) - f(\mathbf{x}^+)}{\sigma(\mathbf{x})}\right). \end{aligned}$$

An obvious disadvantage of PI is that it is a greedy algorithm that only considers exploration, and the sampling point will be limited to a small range. Thus, the optimization can easily be stuck at a local optimum. Hence, a parameter ϵ is added to the PI function. This is a tuneable exploration parameter which adjusts exploitation or exploration. Increasing ϵ will incentivise exploration, while decreasing ϵ will incentivise exploitation. There is one significant drawback to adding ϵ, that is adding a new parameter that needs to be tuned. The choice of ϵ can affect the performance of the search, [30]. The extended PI function is expressed as

$$a_{\mathrm{PI}}(\mathbf{x}) = \Phi\left(\frac{\mu(\mathbf{x}) - f(\mathbf{x}^+) - \epsilon}{\sigma(\mathbf{x})}\right). \tag{2.40}$$

An alternative to PI is to maximize the expected improvement that a point can achieve in the vicinity of the current optimum, [61]. This gives rise to the function Expected Improvement (EI), [43], which is supposed to remedy the exploitation problem with PI. In contrast to PI, EI considers the amount of improvement and not only the probability of improvement. The improvement function can be defined as

$$I(\mathbf{x}) = \max\{f(\mathbf{x}) - f(\mathbf{x}^+), 0\},$$

and the EI strategy involves maximizing $E[I(\mathbf{x})]$. To calculate the expectation of improvement, a reparameterization trick is introduced. Recall that $f(\mathbf{x})$ is normally distributed with mean $\mu(\mathbf{x})$ and variance $\sigma^2(\mathbf{x})$. If $z \sim N(0,1)$, then $f(\mathbf{x}) = \mu(\mathbf{x}) + \sigma(\mathbf{x})z$ is normal with mean $\mu(\mathbf{x})$ and variance $\sigma^2(\mathbf{x})$. Then, the improvement can be written as

$$I(\mathbf{x}) = \max(f(\mathbf{x}) - f(\mathbf{x}^+), 0) = \max(\mu(\mathbf{x}) + \sigma(\mathbf{x})z - f(\mathbf{x}^+), 0).$$

The expectation of the improvement is given by

$$E[I(\mathbf{x})] = \int_{-\infty}^{\infty} I(\mathbf{x})\phi(z)dz,$$

where $\phi(z)$ is the probability density function of the normal distribution. The integral can be broken down to two parts, one where $f(\mathbf{x}) - f(\mathbf{x}^+)$ is positive and one where it is negative. The switch occurs at $f(\mathbf{x}) = \mu(\mathbf{x}) + \sigma(\mathbf{x})z = f(\mathbf{x}^+)$ and the point is denoted

$$z_0 = \frac{f(\mathbf{x}^+) - \mu(\mathbf{x})}{\sigma(\mathbf{x})}.$$

Then, the acquisition function EI can be calculated

$$\begin{aligned}
a_{\text{EI}}(\mathbf{x}) = E[I(\mathbf{x})] &= \underbrace{\int_{-\infty}^{z_0} I(\mathbf{x})\phi(z)dz}_{=0} + \int_{z_0}^{\infty} I(\mathbf{x})\phi(z)dz = \int_{z_0}^{\infty} I(\mathbf{x})\phi(z)dz \\
&= \int_{z_0}^{\infty} \max(f(\mathbf{x}) - f(\mathbf{x}^+), 0)\phi(z)dz = \int_{z_0}^{\infty} (\mu(\mathbf{x}) + \sigma(\mathbf{x})z - f(\mathbf{x}^+))\phi(z)dz \\
&= \int_{z_0}^{\infty} (\mu(\mathbf{x}) - f(\mathbf{x}^+))\phi(z)dz + \int_{z_0}^{\infty} \sigma(\mathbf{x})z\frac{1}{\sqrt{2\pi}}e^{-z^2/2}dz \\
&= (\mu(\mathbf{x}) - f(\mathbf{x}^+))\underbrace{\int_{z_0}^{\infty} \phi(z)dz}_{=1-\Phi(z_0)} + \sigma(\mathbf{x})\frac{1}{\sqrt{2\pi}}\int_{z_0}^{\infty} ze^{-z^2/2}dz \\
&= (\mu(\mathbf{x}) - f(\mathbf{x}^+))(1 - \Phi(z_0)) + \sigma(\mathbf{x})\underbrace{\frac{1}{\sqrt{2\pi}}e^{-z_0^2/2}}_{=\phi(z_0)} \\
&= (\mu(\mathbf{x}) - f(\mathbf{x}^+))\Phi(-z_0) + \sigma(\mathbf{x})\phi(z_0) \\
&= (\mu(\mathbf{x}) - f(\mathbf{x}^+))\Phi\left(\frac{\mu(\mathbf{x}) - f(\mathbf{x}^+)}{\sigma(\mathbf{x})}\right) + \sigma(\mathbf{x})\phi\left(\frac{\mu(\mathbf{x}) - f(\mathbf{x}^+)}{\sigma(\mathbf{x})}\right),
\end{aligned}$$

where the last equality is obtained by taking advantage of the symmetric properties of the normal probability density function, that is $\phi(z_0) = \phi(-z_0)$. The expected improvement contains two terms, which can be interpreted as the exploration and exploitation properties of the function. This is because the first term increases when the variance increases, while the second term increases when the mean increases. Similarly to PI, the parameter ϵ can be introduced to adjust exploitation versus exploration. The EI acquisition function then becomes

$$a_{\text{EI}}(\mathbf{x}) = (\mu(\mathbf{x}) - f(\mathbf{x}^+))\Phi\left(\frac{\mu(\mathbf{x}) - f(\mathbf{x}^+) - \epsilon}{\sigma(\mathbf{x})}\right) + \sigma(\mathbf{x})\phi\left(\frac{\mu(\mathbf{x}) - f(\mathbf{x}^+) - \epsilon}{\sigma(\mathbf{x})}\right). \tag{2.41}$$

An acquisition function based on information is the GP Upper Confidence Bound (UCB), [56]. The idea is to exploit upper confidence bounds (lower when considering minimization). The next sampling point can use either the current optimum value or explore the confidence bounds. In this way, the function balances the trade-off between exploitation and exploration. The function takes the form

$$a_{\text{UCB}}(\mathbf{x}) = \mu(\mathbf{x}) + \kappa\sigma(\mathbf{x}), \tag{2.42}$$

where κ is a tuneable parameter that balances exploitation against exploration. A high value of κ favours exploration, while a lower value favours exploitation. It can be used to set the trade-off between the use of the current optimum or the confidence bounds. Therefore, UCB is a weighted sum of the expected performance and the uncertainty, captured by $\mu(\mathbf{x})$ and $\sigma(\mathbf{x})$, respectively.

Chapter 3

The Dataset

The dataset is provided by SpareBank1 Kreditt AS, a part of the SpareBank1 alliance. The alliance consists of 14 private savings banks across Norway that collaborate on a common platform and brand. The data were retrieved from March 2020 to July 2021 and consist of 32722 observations and 60 variables, including the response. Each observation corresponds to a customer who has been contacted by SpareBank1 with an offer to refinance his or her consumer loans and credit cards. The customer can either accept (success) or decline (not success) the offer. The response, AppliedInd, reflects the outcome of the call activity, encoded in 0 (not success) and 1 (success). The variables in the dataset can be divided into three groups. The first group consists of bank data, the second group consists of credit card data, and the third group consists of variables constructed from machine learning algorithms by SpareBank1. This chapter presents the explanatory variables and the response, as well as visualizations of some selected variables. This is done to gain a better understanding of how the variables affect the response. Lastly, a description of the data pre-processing is presented. Much of this chapter will coincide with the corresponding chapter in the previous project thesis, [53].

3.1 Explanatory Variables and Response

There are 59 explanatory variables in the dataset. Among others, the variables describe which type of bank the customer belongs to, credit card transactions, and user patterns. The variables are all presented with explanations in Appendix A. The division of the data into the three groups is shown in Table 3.1. It can be useful to divide the data into groups to better understand what type of information is included in the dataset. However, the models will not be built with this division, since most information can be extracted from the full dataset. The division was previously tested in the project thesis, [53], without showing significant improvements.

Table 3.1: The division of the dataset into the three groups; bank, credit card, and machine learning (ML) data. The response, AppliedInd, is not included.

Dataset	Description	Number of variables
Bank data	Information about the customers and their payments	18
Credit card data	Information about the customers credit cards, spendings, and transactions	39
Machine learning data	Scores calculated by SpareBank1 using machine learning algorithms	2
Full data	Consists of all the above	59

The bank data contains information about the customers and their payments from bank accounts. Information about the customers includes gender, age, start time of the call activity, and which bank they belong to. The payments can be made to known external credit card accounts, repayment loan accounts, or collection accounts. The credit card data contains information about the customers credit cards, spendings, and transactions. It also includes the interest earning balance, revolving balance, cash balance, and credit card limit. In addition, the customers are segmented by the variable "Segment23Name". The segmentation is done based on several factors, such as user patterns and to what extent the customer is a revolving customer. Revolving credit is a type of credit that allows a customer to borrow money repeatedly up to a certain limit. Then, a portion of the current balance needs to be repaid on time. The machine learning data consist of two variables, P_REFIN and BehaviourScore_P_DCA2, describing the probability of a customer refinancing on own initiative and the probability of defaulting in the next 12 months, respectively.

The response, AppliedInd, reflects whether the customer accepts the offer to refinance or not. It is encoded in 0 and 1 and is thus a binary response. The frequency and percentage of the two classes are shown in Table 3.2. Sine there are more customers who decline the offer, 77.7% versus 22.3%, the data are imbalanced.

Table 3.2: The frequency and percentage of the two classes of the response, AppliedInd, in the dataset.

	Frequency	Percentage
AppliedInd = 1	7289	22.3 %
AppliedInd = 0	25433	77.7%

3.2 Pre-Processing

Prior to fitting the models, the data needs to be pre-processed. The dataset contains several observations with missing values. The reason why they are missing is unknown. The observations with missing variables that count and sum payments to different accounts are set to zero. It is reasonable to assume that they are missing since there are no payments. The remaining observations with missing values are removed. This leads to omitting 377 observations, leading to a total of 32345 observations in the dataset.

The next step in the data pre-processing is feature construction. The variable PeriodId, giving the date of the start of the call activity, is transformed to months since the activity started. It is renamed MonthsAgo and is now an integer. Next, the variable BK_ACCOUNT_ID is removed, since it is only a label and does not provide any information. Lastly, the variables that sum transactions in given classes over the last three months are summed to one common variable. The same is done to the variables that sum transactions in given classes over the last 12 months.

Therefore, these 30 variables are reduced to two; SumL3 and SumL12. Thus, the dataset consists of 32345 observations and 31 variables, including the response.

The XGBoost algorithm does not accept categorical variables. When modelling XGBoost the categorical variables are dummy encoded, meaning each category gets its own column. The majority category is removed and used as a reference for the other categories. This leads to an addition of 37 variables, resulting in a total of 68 variables, including the response. Random Forests accepts categorical variables, and dummy encoding is not necessary.

The data are then split into a training set and a test set. The split is done randomly, with 75% of the data used for training and 25% used for testing. The split is done using stratification, meaning both the training and test sets contain the same class imbalance as the original dataset. The training and test sets contain 24259 and 8086 observations, respectively. The training set is used to build the models, whereas the test set is used only to evaluate the model predictions. This is done to avoid overfitting and obtain reliable model evaluations. Normally, the data should be scaled when applying classification methods, but since XGBoost and Random Forest are tree-based methods this is not necessary.

3.3 Visualization

To get a better understanding of how the explanatory variables and the response are related, it is crucial with some visualizations. First, it is important to investigate how the variables correlate. The categorical variables are therefore dummy encoded such that correlations can be calculated. The data are divided into three groups in the dataset. The correlations between the bank variables are shown in Figure 3.1. Naturally, the variables that count the number of payments to the same type of known external accounts the last 12 months correlate. For instance, the variable CountPaidToCCL12 correlates with CountDistinctPaidToCCL12 and CountRoundPaidToCCL12. All three variables count payments to known credit card accounts. The variable MonthsAgo, constructed from PeriodId, describes the number of months since the call activity started. This variable correlates with several other variables such as weeknr, SumPaidToCCL12, CountPaidToCCL12, and CountRoundPaidToCCL12. This means that there is a connection between the time from the start of the call activity and payments to credit card accounts. The longer the time that has passed, the more payments will be made.

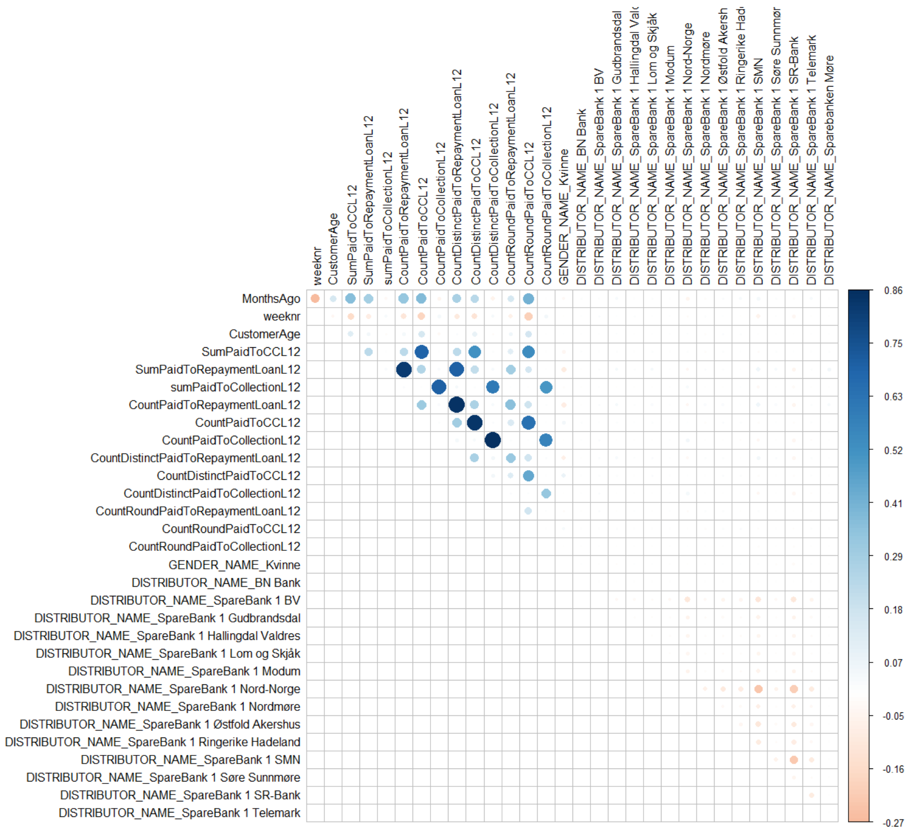


Figure 3.1: Correlation plot of the bank variables in the dataset.

For the credit card dataset, only the correlations higher than 0.5 in absolute value are shown in Figure 3.2, for simplifying purposes. The full correlation plot is displayed in Appendix B, Figure B.1. The variable INTEREST_EARNING_LENDING_AMT, describing the interest earning balance or the amount not paid in full last statement, correlates with both CASH_BALANCE_AMT (cash balance) and CreditLimitAmt (credit limit). Its not surprising that there is a connection between the interest earning balance and the balance originating from cash withdrawals and transactions. Moreover, the credit card limit also affects the earning balance. In addition, the variables that sum the transactions of different classes in the last 3 and 12 months correlate. If a customer has many transactions in one class during 3 months, the customer is likely to show a similar pattern during the next 12 months. This can be seen in for instance the variables sumOTHER_RETAILL12 and sumOTHER_RETAILL3, which have a strong positive correlation.

The correlations in the ML dataset are displayed in Appendix B, Figure B.2.

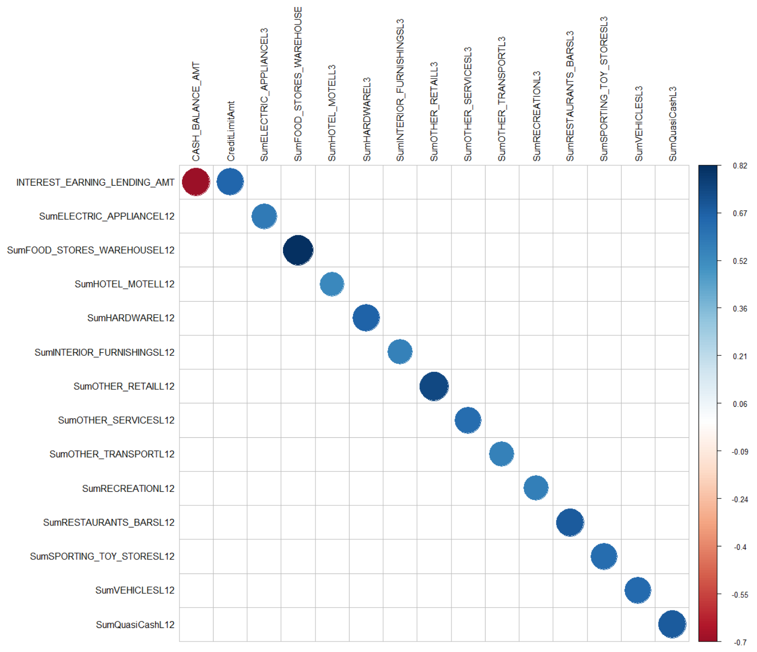


Figure 3.2: Correlation plot of the credit card variables with correlations higher than 0.5 in absolute value.

Density plots of some selected variables are shown in Figure 3.3, and can help understand where the different values of the variables are concentrated over the interval. The densities are separated by the response, AppliedInd, with pink displaying 0 (not accept) and blue displaying 1 (accept). The densities vary, indicating differences between customers who accept and not accept the offer to refinance. Looking at the density for the variable INTEREST_EARNING_LENDING_AMT (interest earning balance), top left, customers who accept the offer have higher values of interest earning balance, while customers who do not accept the offer tend to have lower values. Both densities are left skewed, meaning most customers have lower values. The values of the probability of refinancing on own initiative, P_REFIN (top right), is also differentiated over the interval. The density for those who did not accept the offer has a large peak at a probability around 5% and a lower density for higher probabilities. For customers who accept the offer, the density is more flat and takes higher probabilities. This indicates that a customer with a high score for P_REFIN is more likely to accept the offer, reflecting the purpose of the variable. A difference in densities can also be seen for the variable revUtilL12. This variable is the average revolving balance last 12 months divided by the average credit limit last 12 months. A customer with a lower value is more likely to decline the offer, while higher values indicate customers who accepts. The peak at around 0.9 is a lot higher for customers that accepts the offer, than for those who decline. There is also a difference in the credit limit amount, CreditLimitAmt (bottom right). Customers who refinance are likely to have higher credit limits. A high credit limit can potentially lead to higher unsecured debt and hence more reason to refinance. Most customers who decline the offer has lower credit limits.

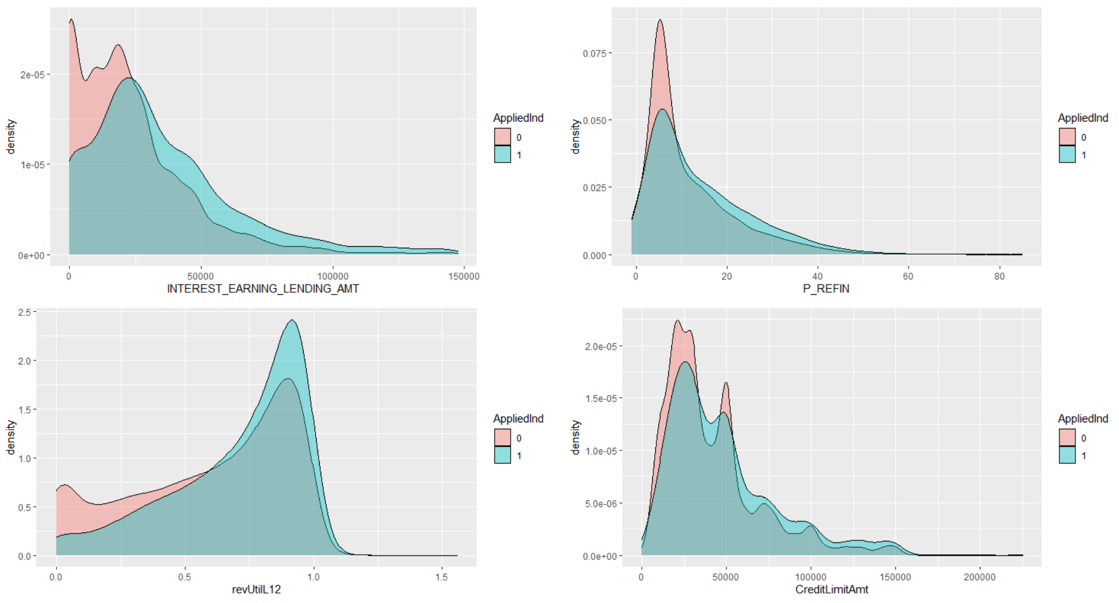


Figure 3.3: Density plots of selected variables. **Top left:** Interest earning lending amount or amount not paid in full last statement (INTEREST_EARNING_LENDING_AMT). **Top right:** Probability of refinancing on own initiative (P_REFIN). **Bottom left:** Average revolving balance last 12 months divided by average credit limit last 12 months (revUtilL12). **Bottom right:** Credit limit amount (CreditLimitAmt).

To further investigate the variables, boxplots of some selected variables are shown in Figure 3.4. Boxplots are used to get a visual summary of the explanatory variables and their median values. Additionally, dispersion of the data and skewness can be easily identified. Looking at the boxplot for cash balance amount (CASH_BALANCE_AMT, top left), the median is below zero, around -10000. The distribution appears slightly negatively skewed, with the median being closer to the top of the box and the whisker being shorter on the upper end of the box. The interquartile range is quite small, indicating the data is not very dispersed. However, there are several outliers below the lower whisker, indicating that some customers have a negative cash balance greater than the minimum. These customers have a large negative cash balance, which coincides with the call campaign targeting customers in debt. It can also be interesting to look at the distribution of the age of the customers (CustomerAge, top right). The median is around age 45, with a minimum around 22 and maximum around 75. The distribution looks almost normally distributed, but the top whisker is slightly larger than the lower one. The distribution of the number of months since call activity started is also displayed in the bottom left (MonthsAgo). The median of months ago since the call activity started is around 7 months. The distribution appears positively skewed, with the median being closer to the bottom of the box and the whisker being shorter on the lower end of the box. The maximum is around 20 months. It is natural to think that the longer the call activity lasts, the more likely the customer is to accept the offer to refinance. The distribution for the account (cards) age in months almost looks normally distributed, (MonthsSinceAccountCreated, bottom right). There is a slight positive skew with a shorter whisker on the lower end of the box and the median being closer to the bottom of the box. The account age has a median around 90 months or around 7.5 years, and a maximum over 200 months (almost 17 years). This indicates that many of the customers eligible for refinancing have had credit cards for many years, which means that some may have accumulated a large amount of debt.

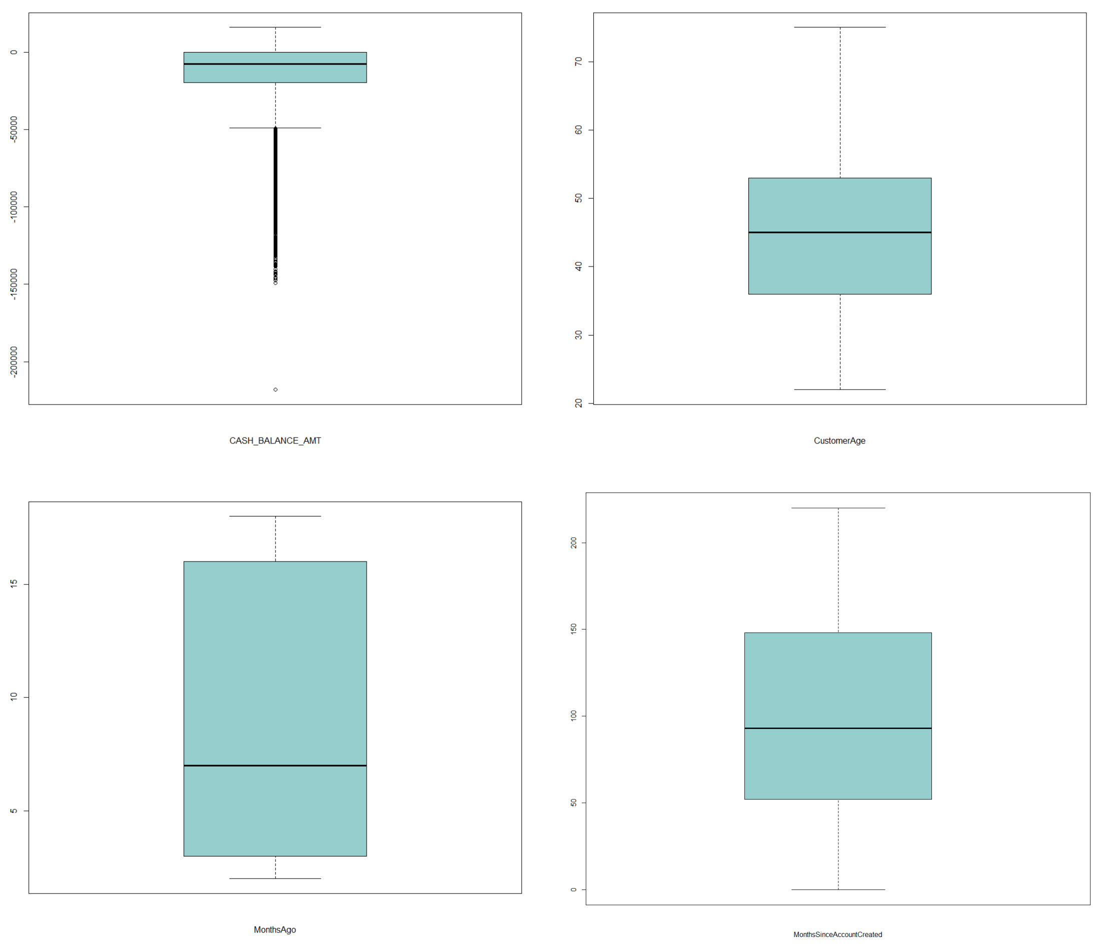


Figure 3.4: Boxplots of selected variables. **Top left:** Cash balance originating from cash withdrawals and transactions (CASH_BALANCE_AMT). **Top right:** Customer age (CustomerAge). **Bottom left:** Months since call activity started (MonthsAgo). **Bottom right:** Account (cards) age in months (MonthsSinceAccountCreated).

Further, there might be a connection between the gender of the customer and whether they choose to refinance or not. In the dataset, there are more men than women, with 55.9% men and 44.1% women. Looking at the count of customers applying for refinance based on their gender, Figure 3.5, there are more men accepting the offer than women. 23.4% of all men choose to refinance, while 19.2% of all females choose to refinance. This might suggest that is is easier for men to accept than for women. The figure gives the percentage of the relative count of women and men in the dataset.

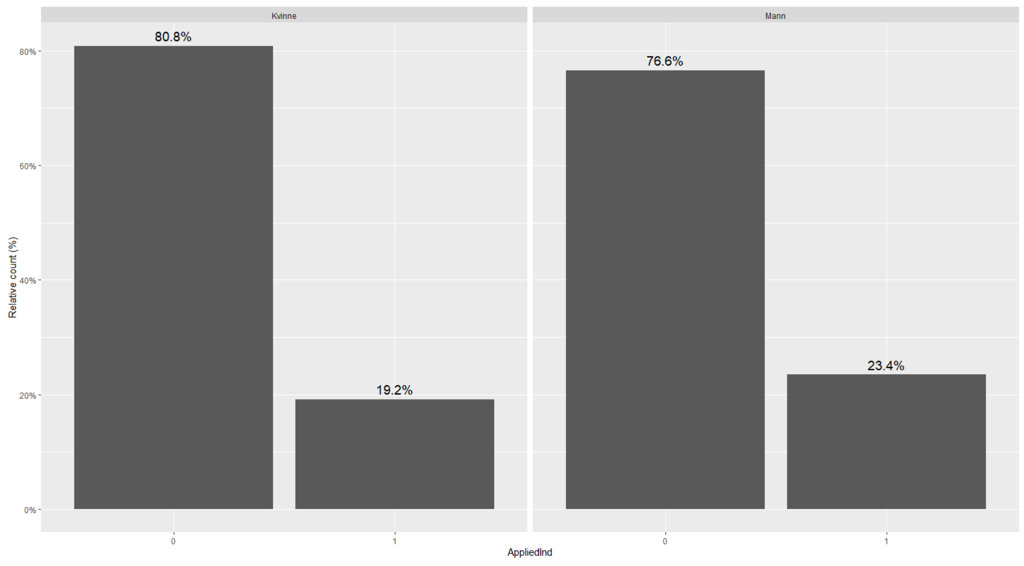


Figure 3.5: Relative count of customers applying for refinance (1) or not applying for refinance (0), based on their gender. Women are to the left and men to the right.

The same type of plot is shown for the variable INTEREST_EARNING_LENDING_AMT, Figure 3.6, but divided into three groups; a balance lower than 50000, between 50000 and 100000, and above 100000. In the dataset, only 1.6% of the customers have a balance over 1000000, while 10.5% have a balance between 50000 and 100000 and 88% have a balance below 50000. Looking at the figure, almost 50% of all customers who have a balance greater than 100000 accept the offer. For customers with a balance between 50000 and 10000, as much as 35.3% accepts the offer. It should be noted that most customers are in the third group with a balance less than 50000. In this group, only 19.4% choose to refinance their loans. This again suggests that customers with a high interest earning balance are more likely to accept an offer to refinance.

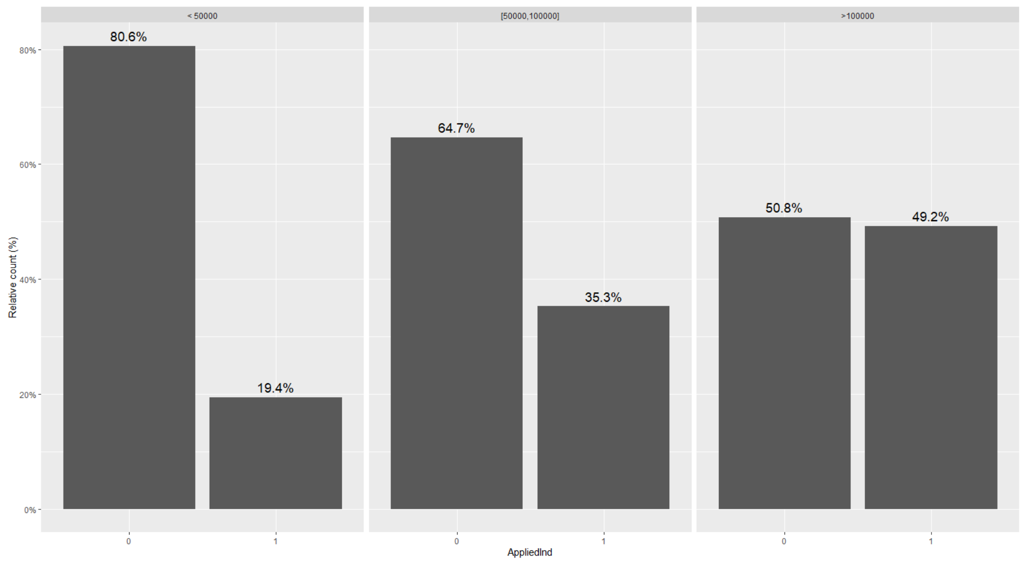


Figure 3.6: Relative count of customers applying for refinance (1) or not applying for refinance (0), based on their interest earning balance (INTEREST_EARNING_LENDING_AMT). The balance is divided into three groups; lower than 50000, between 50000 and 100000, and above 100000.

Chapter 4

Numerical Experiments and Methods, an Overview

This chapter will explain the numerical experiments and methods tested to build and improve the models. All models were implemented using RStudio. The algorithms implemented are presented in Appendix B, and the R output is presented in Appendix C and Appendix D for XGBoost and Random Forests, respectively. The methods applied are described in Chapter 2.

4.1 Modelling XGBoost

XGBoost was modelled using the function `xgb.train()` from package `xgboost`. The full training set was used to fit the models. XGBoost was first run with default values for the hyperparameters and evaluated on the test set. The XGBoost-function is not yet implemented to return the actual class labels, but returns a probability instead. Thus, a cut-off must be decided to classify the predictions. Initially, the cut-off was set to 0.5 for all XGBoost models. This means that if an instance was predicted to be less than 0.5 it was classified as 0 and higher than 0.5 it was classified as 1. To optimize the classification performance, the optimal cut-off was calculated for all XGBoost models. To calculate the optimal cut-off value, an optimal cost algorithm was implemented. The algorithm uses the ROC curve, which describes the trade-off between the false positive rate and the true positive rate. To calculate the optimal cut-off value, the cost of false positives and false negatives must be decided. The cost of false negatives should be higher than the cost of false positives. For SpareBank1 it is more important to detect customers who would have accepted the offer to refinance than those who would not. The cost of classifying a positive customer as negative is therefore high. Consequently, the cost of false positives was kept constant at 100, while the cost of false negatives was varied in the range $[200, 300, ..., 1000]$. For each combination of the costs, the optimal cut-off value was calculated. The optimal cut-off was calculated by minimizing the following cost function obtained by summing the positive and negative instances multiplied by their respective weights,

$$\text{Cost} = \text{FPR} \cdot \text{C}_{FP} \cdot \sum_i I\{y_i = 0\} + \text{FNR} \cdot \text{C}_{FN} \cdot \sum_i I\{y_i = 1\},$$

where FPR is the false positive rate, FNR is the false negative rate, and C_{FP} and C_{FN} are the costs of false positives and negatives, respectively. With all the found cut-offs, the corresponding balanced accuracy was calculated on the test set. The cut-off and corresponding cost of false negatives that resulted in the highest balanced accuracy were kept as the optimal ones. The same procedure was followed in the project thesis from 2021, [53]. Appendix B contains the algorithms implemented, that is `OptFN()` to calculate the optimal cost of false negatives and `ROCInfo()` to calculate the optimal cut-off. The function `ROCInfo()` is taken from [35].

4.2 Modelling Random Forests

Random Forests was modelled using the function `randomForest()` from package `randomForest`. The algorithm was run on the full training set and with default values for the hyperparameters. The model was evaluated on the test set.

4.3 Tuning Hyperparameters

The hyperparameters of both XGBoost and Random Forests were first tuned using Design of Experiments. This was initially done to identify the most significant hyperparameters and in what configuration. Only the training set was used in the tuning, and the hyperparameters were tuned by maximizing the balanced accuracy (BACC). Furthermore, a $2^{(5-1)}$ fractional factorial design was applied according to Table 2.2. The design was replicated once to ensure reliable results and to estimate variance. The low and high levels of the hyperparameters were decided to vary around their default values. Since both XGBoost and Random Forests are stochastic methods, different trees are built in each iteration. Thus, for each hyperparameter configuration (level combination of the design), 5 runs of 10-fold cross-validation were performed. The mean BACC over the 5 runs was used as the response. The results of the experiment were analysed using a linear model by means of the function `lm()`. The hyperparameter values were then decided based on the conclusion of the analysis, by checking the main effects, interactions, and model assumptions. The most significant hyperparameters were then identified and chosen for further optimization using Response Surface Methodology. First, the path of steepest ascent was followed for these identified significant hyperparameters. The path was followed until there was no improvements in the mean BACC after 5 runs of 10-fold cross-validation. This was done to move the experiment to a new region closer to an optimum. In the new experimental region, a response surface model was fitted with a central composite design using the function `ccd()` from package `rsm`. The axial distance, α, was set to $\sqrt{k}$, where k was the number of hyperparameters to optimize, and the number of center runs was set to 3. A second-order model, Eq.(2.22), was applied to the design using the function `rsm()`. A canonical analysis of the response surface was then performed, and the stationary point was analysed. If the found stationary point was classified as a saddle point, a linear path was followed to move the experiment closer to an optimum. The path was calculated using the function `canonical.path()` from package `rsm`, and actual experiments were carried out along the path. Again, the response was calculated as the mean BACC after 5 runs of 10-fold cross-validation. At the newly found optimum, a new second-order response surface model was fitted.

Since XGBoost has been shown to give varying results based on what is used as the training set, direct variance modelling was applied. Especially in the case where only a subset of the training set is used for training, quite different models arise. To find a set of hyperparameters that minimizes the in-run variance, an analysis of the variance from the two replicates of the $2^{(5-1)}$ fractional factorial design was performed. The variance for all 16 experiments was estimated as

$$\text{Variance}_i = \frac{1}{2}\left(\text{BACC}_{i1} - \text{BACC}_{i2}\right)^2, \quad i = 1, ..., 16 \tag{4.1}$$

where BACC_{i1} and BACC_{i2}, $i = 1, ..., 16$ are the BACC scores from the first and second run of the $2^{(5-1)}$ fractional factorial design, respectively. Recall that the BACC score was calculated as the mean BACC of 5 runs of 10-fold cross-validation on the training set with the given hyperparameter configuration. First, a linear model was fitted to the estimated in-run variance. The logarithm of the estimated variance was used as a response. This initial screening experiment was performed to reduce the number of hyperparameters. From the analysis of the linear model, significant hyperparameters were chosen for further optimization. The path of steepest descent was then followed to minimize the response, in order to move the experiment closer to a minimum. At the new minimum, a second-order response surface model was fitted using a central composite design with 3 center points. The response was again the logarithm of the estimated in-run variance. Here, a hyperparameter configuration that minimized the in-run variance was found.

Second, the hyperparameters for both XGBoost and Random Forests were tuned using Bayesian optimization. This was performed using the function `BayesianOptimization()` from package `rBayesianOptimization`. The optimization was performed on the training set with 5-fold cross-validation. The resulting BACC score was the mean BACC over the 5 folds. The function contains two choices for the covariance matrix, either the exponential kernel, Eq.(2.38), or the Matérn 5/2 kernel, Eq.(2.39). The Matérn 5/2 kernel was chosen since the exponential kernel can be unrealistically smooth. In addition, the function contains three choices for the acquisition function, either Probability of Improvement (Eq.(2.40)), Expected Improvement (Eq.(2.41)) or the GP Upper Confidence Bound (Eq.(2.42)). The three functions were tested and the values for ϵ and κ were tuned. This was done by testing three values for ϵ and κ and choosing the one that obtained the highest BACC score on the training set. This was applied to both XGBoost and Random Forests. Initially, all hyperparameters were tuned simultaneously. The initial spacings of the bounds of the hyperparameters were chosen to be large, to cover as much of the parameter space as possible. XGBoost was run with 20 initial points and 50 additional runs, while Random Forests was run with 10 initial points and 40 additional runs. The initial points are randomly chosen points to sample from the target function before fitting the Gaussian process. The reason for the difference in the number of iterations is the long computational time required for Random Forests. This gave a total of 70 and 50 iterations for XGBoost and Random Forests, respectively, to ensure that an optimum was found.

Bayesian optimization was then combined with the initial screening experiment, Design of Experiments. The hyperparameters chosen for further optimization in the initial fractional factorial $2^{(5-1)}$ experiment, were chosen for optimization using Bayesian optimization. This was done to investigate the effect of screening on Bayesian optimization, and to check if better exploration can be obtained with fewer hyperparameters. The ranges of the chosen hyperparameters were again set to be large to cover as much of the parameter space as possible. The other hyperparameters were set to the levels decided in the screening experiment. This was applied to both XGBoost and Random Forests, again with 70 and 50 iterations, respectively.

In addition, Bayesian optimization was combined with Response Surface Methodology. Bayesian optimization was run with the central composite designs (CCD) from the response surfaces as initial grids. Compared to a smooth second-order polynomial, applying the CCD as an initial grid can possibly improve the exploitation. This was done for both XGBoost and Random Forests and the respective hyperparameters chosen for optimization in the response surfaces. The other hyperparameters were then kept at the levels decided from the results of the initial screening experiment. In addition to the initial grid, 50 and 40 iterations were added for XGBoost and Random Forests, respectively.

As previously mentioned, both XGBoost and Random Forests are stochastic algorithms, generating different models each time they are run. This can lead to different evaluation results each time a model is evaluated on the test set. How much the evaluation metrics vary was therefore investigated by creating empirical bootstrap confidence intervals. This was done for the evaluation metrics sensitivity, specificity, and balanced accuracy for models trained with tuned hyperparameters from the full Bayesian optimization and evaluated on the test set. Both XGBoost and Random Forests were trained on the training set and evaluated on the test set 30 times, generating 30 measures of sensitivity, specificity, and balanced accuracy. For each of the metrics, a modified approach to the percentile interval, inspired by [48], was followed. From the 30 measures, the mean was calculated, denoted $\bar{x}$. Then, 1000 bootstrap samples with length 30 were created from the 30 measures. For each bootstrap sample, the mean was calculated, denoted $\bar{x}_i^*$. The distribution of the variation of $\bar{x}_i^*$ around its center is given by

$$\delta_i^* = \bar{x}_i^* - \bar{x} \quad \text{for} \quad i = 1, ..., 1000,$$

and was approximated with high precision, due to the law of large numbers. To calculate the bounds for the interval, the 2.5% and 97.5% quantiles of $\delta_i^*, i = 1, ..., 10000$ were estimated using the function `quantile()`. This function calculates the quantiles from the underlying distribution of the data. The 95% bootstrap confidence interval was thus given by

$$[\bar{x} - \delta_{0.025}, \bar{x} - \delta_{0.975}],$$

where $\delta_{0.025}$ and $\delta_{0.975}$ are the 2.5% and 97.5% quantiles, respectively. The algorithms implemented are shown in Appendix B.

Lastly, feature importance was calculated for both XGBoost and Random Forests, before and after tuning. Feature importance calculated for the models trained with default hyperparameters was compared to the models trained with tuned hyperparameters from RSM and Bayesian optimization. For Bayesian optimization, the model chosen was the one with the highest BACC score on the test set out of all models. For XGBoost, the function `xgb.importance()` from package `xgboost` was used to calculate the importance. The importance is measured by calculating the fractional contribution each feature has on the prediction, based on the total gain of the feature splits. For Random Forests, the function `importance()` from package `randomForest` was used to calculate the importance. The feature importance is measured as the total decrease in node impurities from splitting on the variable, averaged over all trees. The node impurity is measured by the Gini index, Eq.(2.6).

Chapter 5

Analysis and Results

This chapter will present and analyse the obtained results. The outputs and results from R are shown in Appendix C and Appendix D for XGBoost and Random Forests, respectively.

5.1 Extreme Gradient Boosting Modelling

The Extreme Gradient Boosting (XGBoost) algorithm contains several hyperparameters divided into three types: general parameters, booster parameters, and task parameters. A full overview can be found in [12]. In this thesis, five hyperparameters are chosen for tuning, displayed with default values in Table 5.1. These variables are chosen since they control overfitting and are suggested for tuning by [12]. Eta is used to reduce the step size by shrinking the weights of the features, giving a more conservative model. Decreasing eta will prevent overfitting. Subsample gives the fraction of the training instances used in the model. If subsample is set to 0.5, it means that half of the training data are used in the training. This can be used to add randomness and make the training more robust to noise, and thus reduce overfitting. It should be noted that when subsample= 1, i.e. all the training data are used in the training, XGBoost will build the same model when the algorithm is run successively. The variable max_depth describes the maximum depth of a tree and directly controls the model complexity. A higher value for max_depth will give a more complex model and is more likely to overfit. The minimum loss reduction required to make a partition on a leaf node of the tree is controlled by gamma. A larger gamma gives a more conservative model. When dealing with binary classification, the parameter scale_pos_weight controls the balance between the positive and negative weights in the tree. This parameter is particularly useful for imbalanced datasets and can contribute to improve the classification of the minority class.

Table 5.1: The hyperparameters of XGBoost chosen for tuning with default values.

Hyperparameter	Default value
eta	0.3
subsample	1
max_depth	6
gamma	0
scale_pos_weight	1

XGBoost was first run with default hyperparameters and cut-off equal to 0.5, and evaluated on the test set. This is done to have benchmark results to compare optimized models to. The default results, Table 5.2, show poor scores in all metrics, except for specificity. Especially, the sensitivity is low, meaning the model struggles to classify the positive class. The specificity is almost 1, indicating that almost all instances are classified as the negative class. This is expected due to the class imbalance in the data. The BACC score of 0.5536 reflects a classifier that is not much better than random. The corresponding confusion matrix, Table 5.3, shows that the number of correctly classified positive customers is small, equal to 215, reflecting the low sensitivity score. This can also be seen in the number of wrongly predicted positive customers, equal to 1519. The model only predicts 107 negative customers as positive, reflecting the high score for specificity.

Table 5.2: Benchmark results from classification metrics with XGBoost trained on the training set with default hyperparameters and cut-off value (0.5). The model is evaluated on the test set.

Hyperparameter values	Sensitivity	Specificity	BACC	AUC	MCC
Default	0.1240	0.9832	0.5536	0.554	0.2249

Table 5.3: Confusion matrix from XGBoost model trained with default hyperparameters and cut-off value (0.5). The model is evaluated on the test set.

Pred. / True	1	0
1	215	1519
0	107	6245

For all XGBoost models, a cut-off value must be decided. The default at 0.5 does not necessarily produce the best classifier. An optimal cut-off is found by the algorithm described in Appendix B.2. For the benchmark model, the cost of false negatives (FN) is found to be 400, and the optimal cut-off is found at 0.26. The cost of false positives (FP) is kept constant at 100. The corresponding ROC curve and cost curve are shown in Figure 5.1. The cost plot (right) shows the optimal cut-off as a blue dotted line. This is where the cost function is minimized. The intersection between the two dotted blue lines in the ROC plot (left) corresponds to the optimal cut-off point. The colours of the curves represent the cost at the given position. A dark red colour reflects a higher cost, while a light green colour reflects a lower cost.

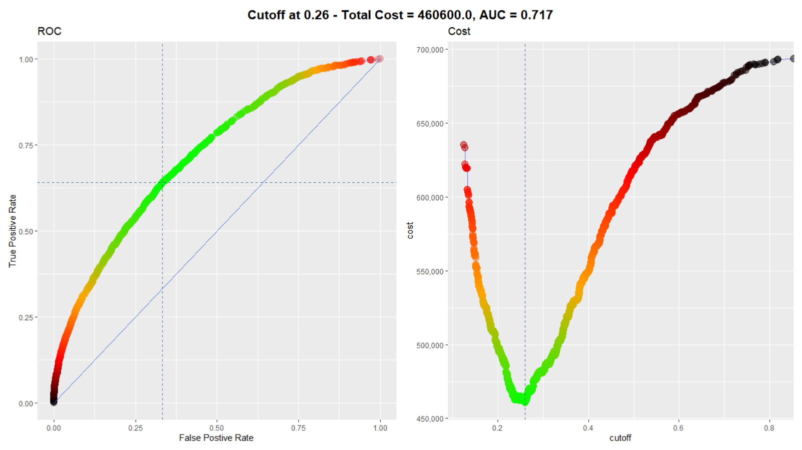


Figure 5.1: The ROC curve and cost-plot for different cut-off values for XGBoost trained on the training set with default hyperparameters. The cost of FP is set to 100 and the cost of FN is found to be 400. The optimal cut-off value is displayed in the title, together with the AUC.

The benchmark model with default hyperparameters and optimal cut-off value is evaluated on the test set, Table 5.4. In comparison to the model with default cut-off 0.5, all metrics have improved, except specificity. Particularly, the sensitivity has increased from 0.1240 to 0.6442. This is desirable considering that the positive class is of interest. Additionally, the BACC is equal to 0.6528, which is a reasonably good score. The corresponding confusion matrix for the benchmark model with optimal cut-off, Table 5.5, shows that the number of wrongly predicted positive instances is lower with optimal cut-off. It is reduced from 1519 to 617, reflecting the large increase in sensitivity. The number of wrongly classified negative instances has increased from 107 to 2150, reflecting the decrease in specificity. The benchmark model with optimal cut-off value is thus a better classifier than the model with default cut-off.

Table 5.4: Benchmark results from classification metrics with XGBoost trained on the training set with default hyperparameters. The models are evaluated on the test set.

Cut-off	Sensitivity	Specificity	BACC	AUC	MCC
0.5	0.1240	0.9832	0.5536	0.554	0.2249
0.26	0.6442	0.6615	0.6528	0.653	0.2557

Table 5.5: Confusion matrix from XGBoost model trained with default hyperparameters and optimal cut-off value, 0.26. The model is evaluated on the test set.

True \ Pred.	1	0
1	1117	617
0	2150	4202

5.2 Hyperparameter tuning XGBoost

The hyperparameters of XGBoost are first tuned using Response Surface Methodology. First, a screening experiment is performed to identify the most important hyperparameters and in what configuration. This is done using Design of Experiments. To move the experiment to a new experimental region, the method of steepest ascent is applied. In the new region, a response surface model is fitted. Next, the hyperparameters are tuned using Bayesian optimization. All the tuning is done on the training set with default cut-off value 0.5. After the tuning, the optimal cost of false negatives and corresponding optimal cut-off value are calculated for all tuned models.

5.2.1 Optimization through Response Surface Methodology

To perform Design of Experiments, the low and high levels of the hyperparameters need to be decided. Eta has default value 0.3, and the levels are decided to be 0.3 ± 0.2. Subsample has default value 1 with range $(0, 1]$, and therefore, the levels are decided to be 0.5 and 1. The default value of max_depth is 6 and the levels are decided to be 6 ± 3. Gamma has default value 0 and range $[0, \infty]$, and the levels are decided to be 0 and 1. The variable scale_pos_weight controls the balance of positive and negative weights and can accommodate the class imbalance. The default value is 1. When dealing with imbalanced data, a recommended value is the number of negative instances divided by the number of positive instances in the dataset. This factor is equal to 3.631 in the training set. Therefore, the low value is decided to be the default, 1, and the high value is decided to be 3.631. The factor names and low and high levels of the hyperparameters are presented in Table 5.6.

Table 5.6: The factor names and low and high levels of the hyperparameters of XGBoost.

Factor	Hyperparameter	Low level (-1)	High level (+1)
A	eta	0.1	0.5
B	subsample	0.5	1
C	max_depth	3	9
D	gamma	0	1
E	scale_pos_weight	1	3.631

The $2^{(5-1)}$ fractional factorial experiment is performed according to Table 2.2 with two replicates. The resulting BACC score is the mean BACC after 5 runs of 10-fold cross-validation on the training set. The results after both replicates of the experiment are shown in Appendix C.1, Table C.1. The mean BACC scores after the two replicates with the given hyperparameter configurations are shown in Table 5.7. Looking at the scores, the maximum BACC score is 0.6471 at level code *ade*.

A linear model is fitted to the two replicates of the $2^{(5-1)}$ fractional factorial experiment, with the full model summary displayed in Appendix C.1. The coefficient estimates are shown in Table 5.8, with 13 significant coefficient estimates at level 0.05. Among the interactions, the coefficient estimates of CE, AE, and AC are the most significant when looking at the p-values. Factors E, C, and A also have significant estimates. The linear model assumes that the residuals are independent and approximately normal distributed. The normal Q-Q plot of the residuals, displayed in Appendix C.1, Figure C.2, shows that the residuals fall on the line, with only a few deviations. This confirms that the assumption holds. Additionally, the residuals look randomly spread out, Figure C.3, suggesting that the assumption of normal independent errors holds for the model.

Table 5.7: The mean BACC score after the two replicates of XGBoost trained on the training set using the $2^{(5-1)}$ fractional factorial design. The values of factors A, B, C, D, and E are set to the values in Table 5.6

Run	A	B	C	D	E = ABCD	Level code	Mean BACC
1	-1	-1	-1	-1	1	1	0.6387
2	1	-1	-1	-1	-1	a	0.5340
3	-1	1	-1	-1	-1	b	0.5108
4	1	1	-1	-1	1	abe	0.6467
5	-1	-1	1	-1	-1	c	0.5624
6	1	-1	1	-1	1	ace	0.6255
7	-1	1	1	-1	1	bce	0.6450
8	1	1	1	-1	-1	abc	0.5720
9	-1	-1	-1	1	-1	d	0.5145
10	1	-1	-1	1	1	ade	0.6471
11	-1	1	-1	1	1	bde	0.6407
12	1	1	-1	1	-1	abd	0.5314
13	-1	-1	1	1	1	cde	0.6400
14	1	-1	1	1	-1	acd	0.5735
15	-1	1	1	1	-1	bcd	0.5614
16	1	1	1	1	1	abcde	0.6375

Table 5.8: The model summary of the coefficients of the linear model fitted to the results of the $2^{(5-1)}$ fractional factorial design of XGBoost.

	Estimate	Std. Error	t value	Pr(> $\|t\|$)	
(Intercept)	0.5925823	0.0001671	3547.278	< 2e-16	***
A	0.0033953	0.0001671	20.325	< 7.46e-13	***
B	0.0006130	0.0001671	3.669	0.002072	**
C	0.0095886	0.0001671	57.399	< 2e-16	***
D	0.0006896	0.0001671	4.128	0.000789	***
E	0.0475777	0.0001671	284.807	< 2e-16	***
A:B	0.0003148	0.0001671	1.885	0.077795	.
A:C	-0.0034256	0.0001671	-20.506	6.51e-13	***
A:D	0.0007241	0.0001671	4.335	0.000512	***
A:E	-0.0043392	0.0001671	-25.975	1.65e-14	***
B:C	0.0011835	0.0001671	7.085	2.58e-06	***
B:D	-0.0011196	0.0001671	-6.702	5.08e-06	***
B:E	0.0017142	0.0001671	10.261	1.92e-08	***
C:D	0.0002615	0.0001671	1.565	0.137036	
C:E	-0.0127201	0.0001671	-76.144	< 2e-16	***
D:E	0.0004880	0.0001671	2.921	0.009986	**

The main effects of factors A, B, C, D, and E are displayed in Figure 5.2. The main effects contain a line connecting two points corresponding to the response mean of the two levels of the factors. The resulting effect on the response when the factor is moved from low to high level is described by the slope of the line. The greater the effect the factor has on the response, the steeper the slope. If the slope is approximately 0, that is a horizontal line, the factor does not significantly affect the response. Looking at the main effects of the factors, Figure 5.2, factor E stands out, while factors A and C have a positive slope, and factors B and D have almost a horizontal line. Factor E has the greatest main effect with a steep slope. This can also be seen in the coefficient estimates in the model summary, Table 5.8, showing that factor E has the largest estimate, equal to 0.04758. However, since there are interactions present in the design, the levels of the factors need to be decided based on the two-factor interaction effects. The interaction effects are shown in Figure 5.3. The interactions describe how the relationship between one factor and the response depends on a second factor. A separate line for each of the levels of the second factor shows the response means for the two levels of the first factor. The low and high levels are represented by the colours red and black, respectively. Parallel lines indicate no interaction effect, while the more nonparallel the lines are, the greater the strength of the interaction. There are interactions between factors A and C, A and E, and C and E. This can also be seen in the coefficient estimates and p-values in the model summary. In particular, the interaction CE looks strong with a large coefficient estimate, equal to -0.01272. Additionally there are small interactions between A and D, B and C, B and D, B and E, and D and E. All of these have significant p-values at level 0.05 when looking at the model summary.

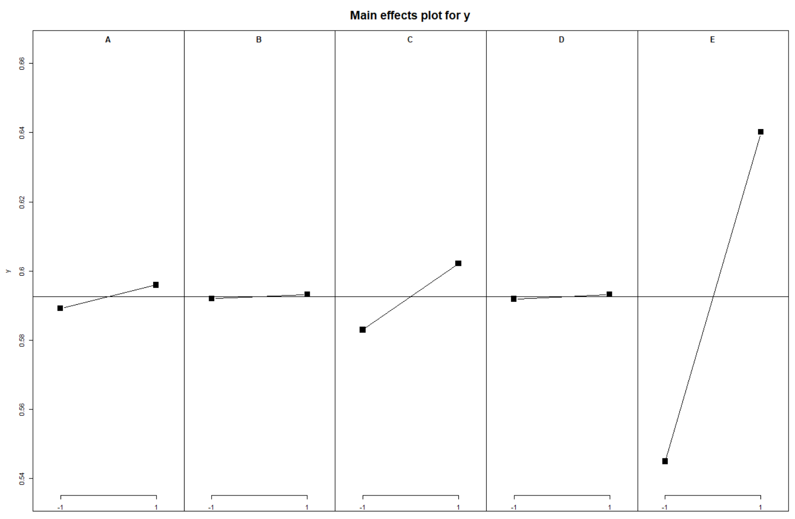


Figure 5.2: The main effects of factors A, B, C, D, and E after conducting the experiments in the $2^{(5-1)}$ fractional factorial design on the training set with XGBoost.

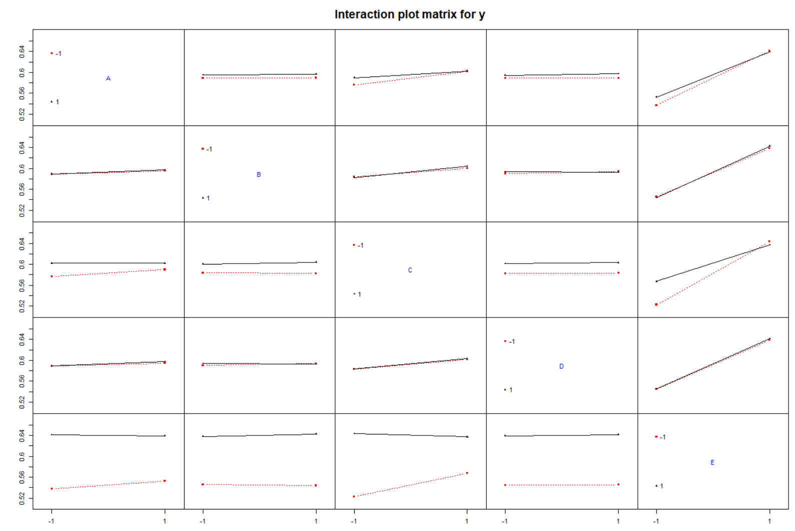


Figure 5.3: The interaction effects of factors A, B, C, D, and E after conducting the experiments in the $2^{(5-1)}$ fractional factorial design on the training set with XGBoost.

The normal plot of the effects is shown in Figure 5.4. The effects that are furthest away from 0 and fall off the line are considered significant. Assuming the response is normally distributed, these effects are also normally distributed with nonzero mean. The negligible effects usually fall on a straight line and are normally distributed with zero mean. With significance level $\alpha = 0.05$, the factors CE, AE, AC, A, C, and E are considered significant. Factors E and C and interaction effect CE fall furthest of the line. This suggests that these are the most significant factors, supporting the conclusion from the analysis of main effects and interaction effects. The pareto plot of the effects, displayed in Appendix C.1, Figure C.1, confirms that the factors E, CE, C, AE, AC, and A are most significant according to Lenth's method at significance level 0.05. Additionally, the factors BE, BC, and BD are considered significant.

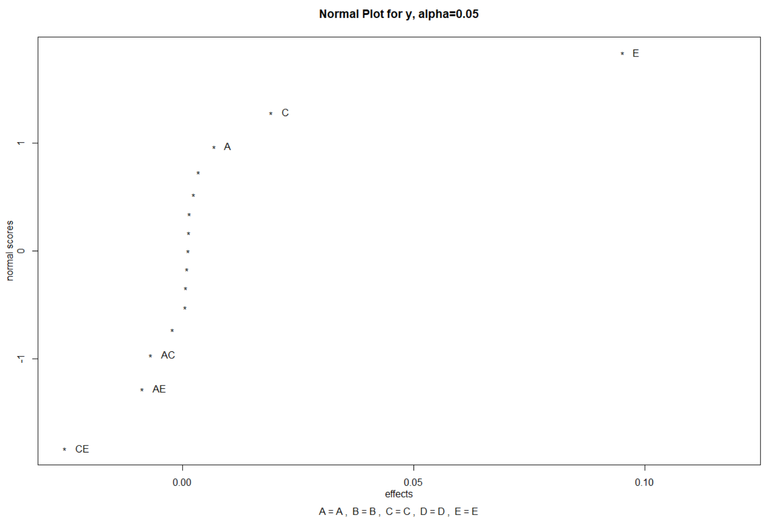


Figure 5.4: Normal plot of the estimates of the effects after conducting the experiments in the $2^{(5-1)}$ fractional factorial design on the training set with XGBoost.

Since there are interaction effects present in the design, the levels of the factors are decided based on these, Figure 5.3. Looking at the most significant interaction effect CE, the highest response is obtained when E is on high level and C on low. This can also be seen in the estimate for the coefficient, Table 5.8, which is negative and equal to -0.01272. When looking at the interaction plot, factor A should be on low level when E is on high level according to the interaction AE. This interaction effect also has a negative coefficient estimate equal to -0.004339. The interaction effect AC also has a negative coefficient estimate. Thus, when C is on low level, A should be on high level. Due to the contradictory conclusions, it is difficult to decide the appropriate level of factor A. Hence, a cube plot between factors A, C, and E is used to see which factor levels result in the highest response. Looking at the cube plot, Figure 5.5, the highest response is obtained when factors A and E are on high level and factor C on low. Factor B is decided based on the interaction BE. When E is on high level, factor B should also be on high level, according to the plot. Lastly, factor D contributes in the interaction BD. When B is on high level, factor D should be on low level, according to the negative coefficient estimate, equal to -0.00112. From the conclusion of the screening experiment, interaction effect CE and factors E and C are most significant. The concluded hyperparameter values are eta= 0.5 (A), subsample= 1 (B), max_depth= 3 (C), gamma= 0 (D), and scale_pos_weight= 3.631 (E).

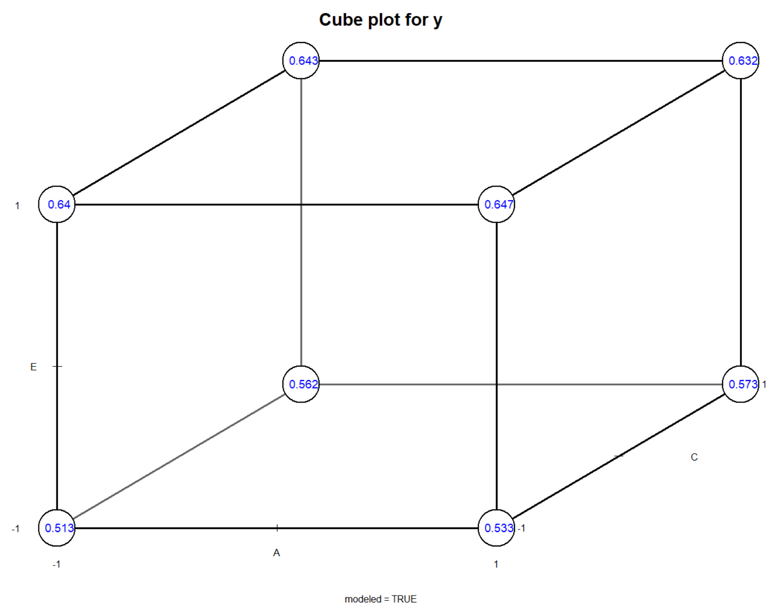


Figure 5.5: Cube plot of factors A, C, and E based on the results of the $2^{(5-1)}$ fractional factorial design.

Next, the method of steepest ascent is performed to move the experimental region closer to an optimum. Here, a response surface can be fitted. The interaction effect CE is investigated further since this was the most influential factor in the screening experiment. Therefore, the factors E, scale_pos_weight, and C, max_depth, are chosen for further optimization. The conclusion of the screening experiment was to set E to the high level, and C to the low. Additionally, the interaction effect CE has a negative coefficient estimate. Thus, the movement will be along the vector where the slope of factor E increases and the vector where the slope of factor C decreases. The estimated slope of factor E, scale_pos_weight, is 0.04758 and the slope of factor C, max_depth, is 0.009589, Table 5.8. The magnitude of the slope of scale_pos_weight is almost 5 times larger than max_depth. Translating this into appropriate units of the hyperparameters can be troublesome, especially since max_depth only accepts integers. The step size for scale_pos_weight is chosen to be 0.02 with 11 steps, starting at scale_pos_weight$= 3.571$. For each value of scale_pos_weight, three values around the low level of max_depth are tested, that is 4, 3, and 2. The other hyperparameters are set to their respective levels from the conclusion of the initial screening experiment. The response is the mean BACC of 5 runs of 10-fold cross-validation performed on the training set. The first run of the path of steepest ascent is displayed in Table 5.9, with max_depth$= 4$. The path of steepest ascent for max_depth$= [3, 2]$ is displayed in Appendix C.1, Table C.2 and Table C.3, respectively.

Table 5.9: Path of steepest ascent followed for scale_pos_weight and max_depth= 4 in XGBoost, while the other hyperparameters are held at obtained levels. The resulting BACC is the mean of 5 runs of 10-fold cross-validation performed on the training set.

A (eta)	B (subsample)	C (max_depth)	D (gamma)	E (scale_pos_weight)	BACC
0.5	1	4	0	3.571	0.6491
0.5	1	4	0	3.591	0.6476
0.5	1	4	0	3.611	0.6494
0.5	1	4	0	3.631	0.6481
0.5	1	4	0	3.651	0.6492
0.5	1	4	0	3.671	**0.6521**
0.5	1	4	0	3.691	0.6472
0.5	1	4	0	3.711	0.6493
0.5	1	4	0	3.731	0.6500
0.5	1	4	0	3.751	0.6520
0.5	1	4	0	3.771	0.6497

The highest value for BACC is 0.6521, obtained with scale_pos_weight= 3.671 and max_depth= 4. With this hyperparameter configuration, 5 runs of 10-fold cross-validation is performed on the training set, resulting in the BACC values 0.6517, 0.6495, 0.6531, 0.6506, and 0.6512, with a mean of 0.6512.

To see if the BACC can be further improved, a second-order response surface model is fitted in the new experimental region. Again, the variables scale_pos_weight and max_depth are further investigated. Additionally, the variable eta, factor A, is included in the response surface model, due to the strong interaction effects AE and AC from the screening experiment. The variables subsample and gamma are kept constant at their obtained values, that is subsample= 1 and gamma= 0. A central composite design is applied, with $\alpha = \sqrt{3}$ and 3 center runs. The levels of scale_pos_weight and max_depth are varied around the optimal values from the results of the steepest ascent, Table 5.9. Eta is varied around the high level, 0.5. The center point for scale_pos_weight is set to 3.671 with low and high levels equal to 3.671 ± 0.1, respectively. The axial points are equal to 3.4978 and 3.8442. The center point for max_depth is 4, with low and high levels equal to 4 ± 1 and axial points equal to 2 and 6, to ensure 5 different values. For eta, the center is chosen to be 0.5, the low and high levels are 0.5 ± 0.05 and the axial points equal 0.4134 and 0.5866. The result of the design is shown in Table 5.10, where the BACC score is the mean BACC after 5 runs of 10-fold cross-validation performed on the training set.

Table 5.10: The central composite design obtained on the training set for the variables eta, max_depth, and scale_pos_weight of XGBoost, with subsample = 1 and gamma = 0.

Run	A (eta)	C (max_depth)	E (scale_pos_weight)	BACC
1	-1	-1	-1	0.6481
2	1	-1	-1	0.6459
3	-1	1	-1	0.6497
4	1	1	-1	0.6504
5	-1	-1	1	0.6494
6	1	-1	1	0.6449
7	-1	1	1	0.6506
8	1	1	1	0.6519
9	$-\sqrt{3}$	0	0	0.6508
10	$\sqrt{3}$	0	0	0.6512
11	0	$-\sqrt{3}$	0	0.6490
12	0	$\sqrt{3}$	0	0.6498
13	0	0	$-\sqrt{3}$	0.6503
14	0	0	$\sqrt{3}$	0.6505
15	0	0	0	0.6497
16	0	0	0	0.6518
17	0	0	0	0.6507

The summary of the response model is presented in Appendix C.2, indicated first RSM. Only the first-order coefficient estimate of max_depth is significant at level 0.05. None of the other coefficient estimates are significant at level 0.05 or 0.1. The lack of fit has p-value 0.2564 meaning at level 0.05 it is not significant. This suggests that the fitted second-order model is a reasonable fit for the true response surface. The proposed stationary point is at eta= 0.4389, max_depth= 3.9686 ≈ 4, and scale_pos_weight= 3.7329. The stationary point is a saddle point because the eigenvalues have opposite signs, 0.0001553, -0.0003411, and -0.001093. This can also be seen in the contour plots of the response surface, Figure 5.6. The plot between max_depth and eta taken at slice scale_pos_weight= 3.73, shows a clear saddle. This is also verified by the perspective plots of the response surface, presented in Appendix C.2 first RSM, Figure C.4. At the proposed stationary point, the BACC after 5 runs of 10-fold cross-validation are 0.6465, 0.6515, 0.6492, 0.6538, and 0.6509, with a mean of 0.6504. This is not an improvement compared to the BACC scores from the results after path of steepest ascent.

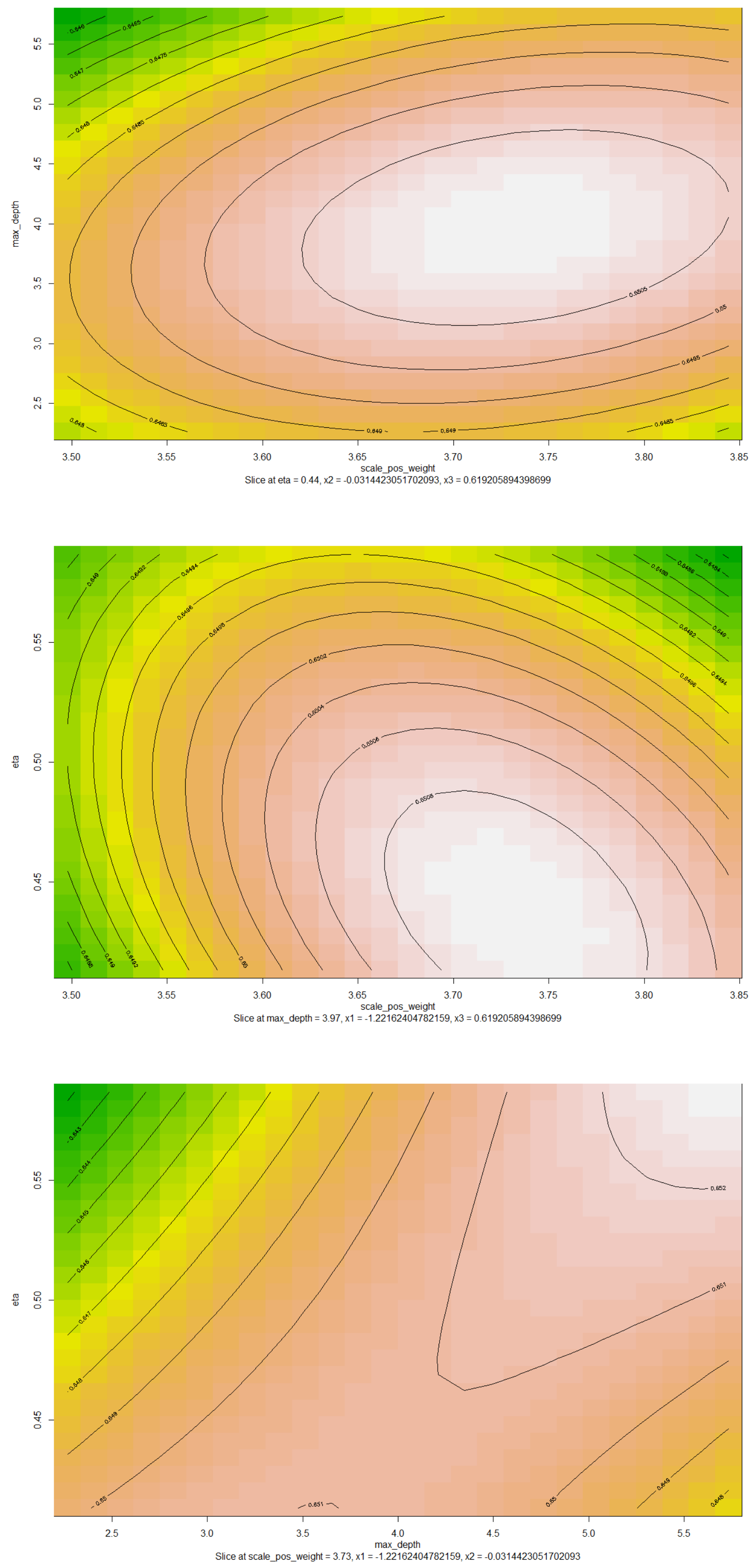


Figure 5.6: Contour plots of the fitted second-order response surface model with XGBoost. The design has center in eta= 0.5, max_depth= 4, and scale_pos_weight= 3.671.

Since the stationary point is a saddle point, the experimental region should again be moved towards an optimum. According to the response surface, the BACC can be improved by moving in the direction of the eigenvector of the positive eigenvalue. This corresponds to the vector $[0.8506, 0.5238, -0.04655]^T$, meaning eta and max_depth should be increased, while scale_pos_weight should be decreased. Therefore, a linear path is calculated using the function `canonical.path()`, originating at the stationary point. The estimated canonical path for distances $[-2, 5]$ from the stationary point is presented in Appendix C.2 first RSM, but actual experiments are performed along the path. The resulting mean BACC after 5 runs of 10-fold cross-validation along the path is shown in Table 5.11. The maximum BACC is found at eta= 0.48145, max_depth= 4, and scale_pos_weight= 3.7283, with a value of 0.6517. With this hyperparameter configuration, 5 runs of 10-fold cross-validation on the training set give the BACC scores of 0.6533, 0.6489, 0.6536, 0.6529, and 0.6544, with a mean of 0.6524. These scores show an improvement compared to the BACC scores from the first RSM.

Table 5.11: Results of the experiments performed along the linear path originating at the stationary point of the first response surface for XGBoost with subsample= 1 and gamma= 0.

Distance	A (eta)	C (max_depth)	E (scale_pos_weight)	BACC
-2	0.35385	$2.921 \approx 3$	3.7422	0.6456
-1	0.39640	$3.445 \approx 3$	3.7376	0.6454
0	0.43890	$3.969 \approx 4$	3.7329	0.6503
1	0.48145	$4.492 \approx 4$	3.7283	**0.6517**
2	0.52400	$5.016 \approx 5$	3.7236	0.6514
3	0.56650	$5.540 \approx 6$	3.7190	0.6509
4	0.60905	$6.064 \approx 6$	3.7143	0.66507
5	0.65155	$6.588 \approx 7$	3.7096	0.6421

At this new maximum, a new response surface is fitted to the interactions between eta (A), max_depth (C), and scale_pos_weight (E) using the CCD with $\alpha = \sqrt{3}$ and 3 center runs. Thus, the new center points are at eta= 0.48145, max_depth= 4, and scale_pos_weight= 3.7283. For eta, the corresponding low and high levels are 0.43145 and 0.53145, respectively, with axial points 0.3948 and 0.5681. For max_depth, the levels are set as before. The low and high levels of scale_pos_weight are set to 3.6783 and 3.7783, respectively, with axial points 3.6417 and 3.8149. The CCD and the model summary of the response surface are presented in Appendix C.2, indicated second RSM. There are no significant estimated coefficients at level 0.05 or 0.1. The p-value of the lack of fit is equal to 0.1641, meaning it is non-significant at level 0.05. The stationary point is estimated to be at eta= 0.4727, max_depth= $5.05587 \approx 5$, and scale_pos_weight= 3.6905. All three eigenvalues are negative, -0.0001623, -0.0003911, and -0.0004693, meaning the stationary point is a maximum. This is verified by looking at the contour plots of the response surface, Figure 5.7. A maximum can be seen in all plots with decreasing response around the optimum. The perspective plots of the response surface, presented in Appendix C.2 second RSM, Figure C.5, also show a maximum. At the stationary point, 5 runs of 10-fold cross-validation are performed on the training set, resulting in the mean BACC scores 0.6517, 0.6523, 0.6541, 0.6518, and 0.6526, with a mean of 0.6525. This is a small improvement compared to the previous results.

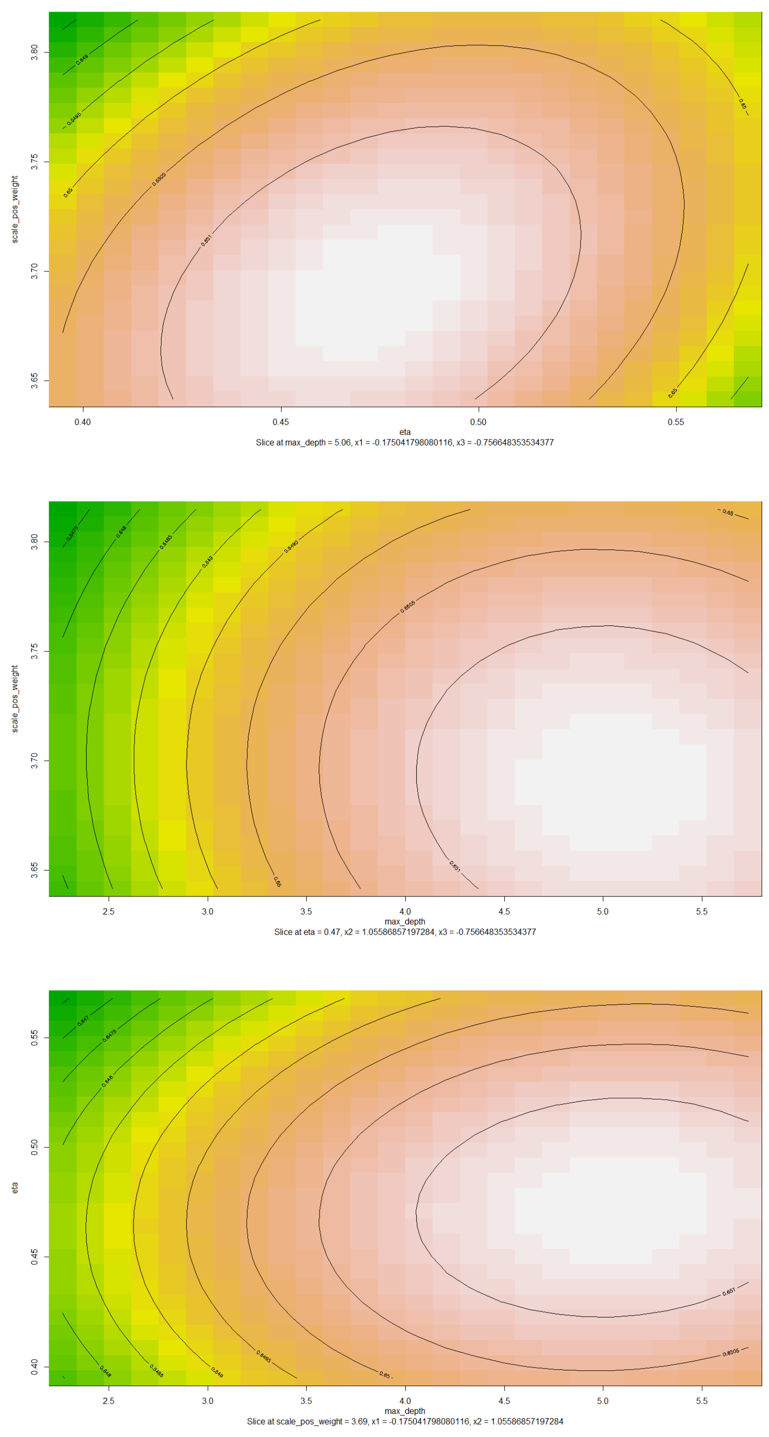


Figure 5.7: Contour plots of the fitted second-order response surface model with XGBoost. The design has center in eta= 0.48145, max_depth= 4, and scale_pos_weight= 3.7283.

The found optimal hyperparameter values are shown in Table 5.12. Subsample and gamma end up with default values, while the hyperparameters chosen for further tuning deviate from the default. First, XGBoost is trained with optimal hyperparameters on the training set and evaluated on the test set with default cut-off value 0.5. Then the optimal cut-off is calculated based on the same algorithm as before, presented in Appendix B.2. The optimal cost of false negatives is found to be 400 with an optimal cut-off equal to 0.47. The ROC plot and corresponding cost plot are displayed in Appendix C.2 second RSM, Figure C.6. The model is then evaluated on the test set using the optimal cut-off value 0.47. The results are displayed in Table 5.13. Applying the optimal cut-off leads to an increase in sensitivity from 0.6130 to 0.6915 which is favourable, considering the positive class is of interest. The specificity has decreased from 0.6979 to 0.6186, leading to a decrease in the BACC from 0.6555 to 0.6551. The optimal cut-off is found based on maximizing the BACC while adjusting the cost of false negatives and false positives. This can potentially lead to a small decrease in the BACC, compared to applying the default cut-off where the costs are not adjusted. However, this decrease is not considered significant. The corresponding confusion matrices, Table 5.14, reflect these changes. The number of wrongly predicted positive instances has decreased from 671 to 535, with optimal cut-off. The decrease in specificity can be seen in the increase in wrongly predicted negative instances from 1919 to 2422. Compared to the default model with optimal cut-off, Table 5.4, the sensitivity is higher for the tuned model using RSM with optimal cut-off. In particular, with optimal cut-off value, the number of wrongly predicted positive customers has decreased from 617 to 535. Additionally, with optimal cut-off, the tuned model using RSM obtains a higher BACC, 0.6551 versus 0.6528, due to the increase in sensitivity. In this way, the tuned model has improved the classification of the important positive class. The tuned model using RSM with optimal cut-off value will be referred to as the RSM model.

Table 5.12: Optimal values of the hyperparameters of XGBoost after optimization using RSM.

eta	subsample	max_depth	gamma	scale_pos_weight
0.4727	1	5	0	3.6905

Table 5.13: Results from classification metrics with XGBoost trained on the training set with tuned hyperparameters using RSM, Table 5.12. The models are evaluated on the test set.

Cut-off	Sensitivity	Specificity	BACC	AUC	MCC
0.5	0.6130	0.6979	0.6555	0.655	0.2645
0.47	0.6915	0.6187	0.6551	0.655	0.2560

Table 5.14: Confusion matrix from the XGBoost model trained with tuned hyperparameters using RSM, Table 5.12. The models are evaluated on the test set with default cut-off 0.5 and optimal cut-off 0.47.

5.14(a) Cut-off 0.5

True \ Pred.	1	0
1	1063	671
0	1919	4433

5.14(b) Cut-off 0.47

True \ Pred.	1	0
1	1199	535
0	2422	3930

5.2.2 Direct Variance Modelling

XGBoost tends to be unstable and can give quite different results depending on what is used as the training set. This can make it troublesome to decide which hyperparameter values actually yield the best model. In particular, this is an issue when subsample is not set to 1. In addition, the cross-validation estimates often differ because a different training set is used in each fold. Therefore, direct variance modelling is performed to find the hyperparameter configuration that gives the minimum in-run variance. This can result in a more stable model and provide more reliable results. The results from the two replicates of the $2^{(5-1)}$ fractional factorial design, Table C.1, are used to estimate the in-run variance for each configuration of A, B, C, D, and E. The levels of the factors are the same as before, Table 5.6, and the logarithm of estimated variance is used as response. The variance is estimated as the mean squared difference in the BACC scores from the two replicates of the $2^{(5-1)}$ fractional factorial design, Eq.(4.1). The results are shown in Table 5.15. It is desirable with as high negative value as possible. The minimum is found to be -22.6365 at level code *abe*.

Table 5.15: The results of direct variance modelling of the results of the $2^{(5-1)}$ fractional factorial design with XGBoost. The resulting variance is estimated as the mean squared difference in the BACC scores from the results of the $2^{(5-1)}$ fractional factorial design, Eq.(4.1).

Run	A	B	C	D	E = ABCD	Level code	Log of estimated variance
1	-1	-1	-1	-1	1	1	-13.1634
2	1	-1	-1	-1	-1	a	-16.3465
3	-1	1	-1	-1	-1	b	-15.2424
4	1	1	-1	-1	1	abe	-22.6365
5	-1	-1	1	-1	-1	c	-13.8028
6	1	-1	1	-1	1	ace	-12.4406
7	-1	1	1	-1	1	bce	-14.8141
8	1	1	1	-1	-1	abc	-13.9299
9	-1	-1	-1	1	-1	d	-15.5613
10	1	-1	-1	1	1	ade	-16.6863
11	-1	1	-1	1	1	bde	-17.8551
12	1	1	-1	1	-1	abd	-14.1587
13	-1	-1	1	1	1	cde	-19.9955
14	1	-1	1	1	-1	acd	-13.2796
15	-1	1	1	1	-1	bcd	-14.1114
16	1	1	1	1	1	abcde	-12.9367

Initially, a linear model with main effects and interactions is fit to the results of the direct variance modelling. The summary of the model is displayed in Appendix C.3. Since no replicates are performed, it is not possible to measure the standard errors or the p-values of the estimated coefficients. The estimated main effects of the factors, Figure 5.8, show that all factors have a slope greater than 0 in absolute value. The main effect of the factor C seems strong. However, there are interaction effects present in the design, Figure 5.9. There are interactions between all factors, except A and E. Looking at the plots, especially the interactions AC, AD, BC, and BD looks strong. When considering the coefficient estimates from the model summary, the interactions AC, AD, and BD stand out, with coefficient estimates above 1. In particular, the interaction effects AC and AD have coefficient estimates equal to 1.1339 and 1.1746, respectively. The normal and half-normal plots of the effects are displayed in Appendix C.3, Figure C.7 and Figure C.8. None of the estimates of the effects are significant at level $\alpha = 0.05$, however, at level $\alpha = 0.12$, interaction effects AC and AD are significant, Figure 5.10. These two factors fall off the line in the plot, strengthening the conclusion that these two interaction effects have a significant impact on the response.

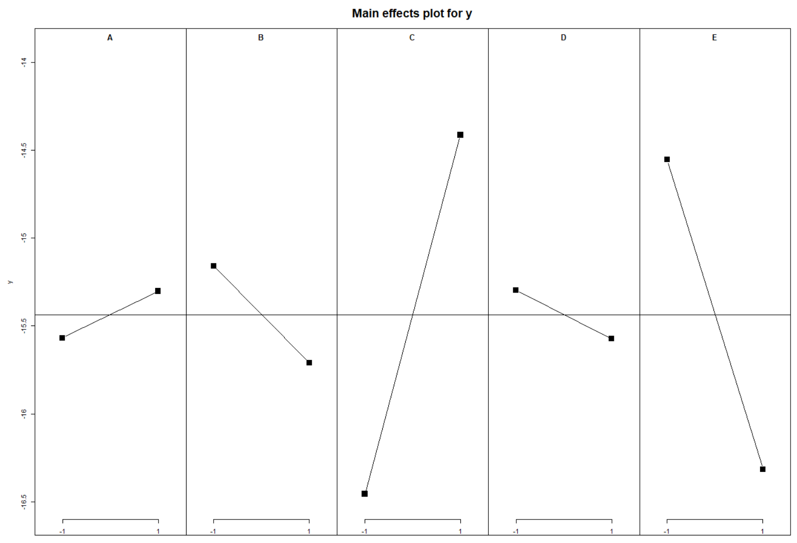


Figure 5.8: The estimated main effects of factors A, B, C, D, and E after conducting direct variance modelling of the results of the $2^{(5-1)}$ fractional factorial design with XGBoost.

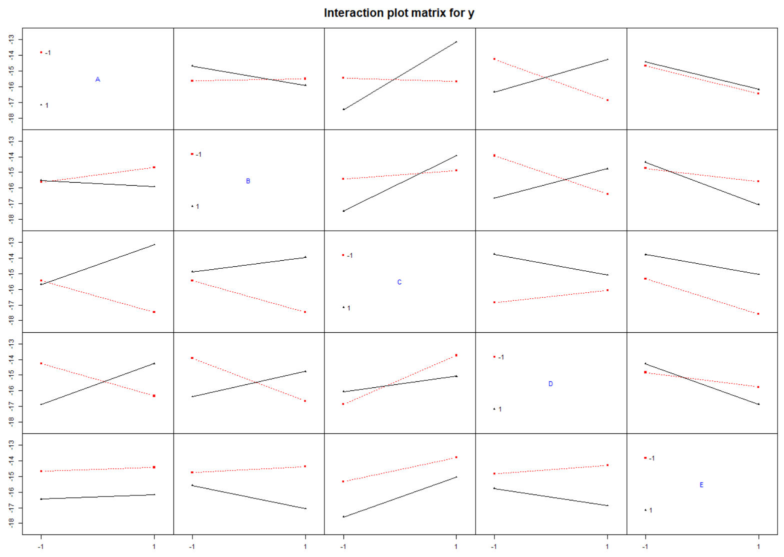


Figure 5.9: The interaction effects of factors A, B, C, D, and E after conducting direct variance modelling of the results of the $2^{(5-1)}$ fractional factorial design with XGBoost.

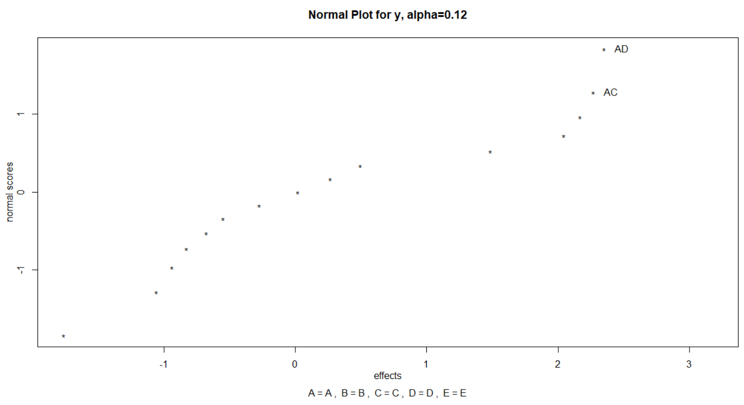


Figure 5.10: Normal plot of the estimates of the effects after conducting direct variance modelling of the results of the $2^{(5-1)}$ fractional factorial design on the training set with XGBoost. Interaction effects AC and AD are significant at $\alpha = 0.12$.

To achieve a minimum response, factor C should be on low level. According to the interaction AC, Figure 5.9, factor A should be on high level when C is on low level. When factor A is on high level, factor D should be on low level based on the interaction AD. Furthermore, the interaction BD is strong, and B is set to the high level, when D is on low level. This also coincides with the effect of interaction BC, which is also strong. Factor E has a strong main effect and should be set to the high level when looking at the interaction BE. This interaction effect is stronger than the interaction DE, when looking at the coefficient estimates, equal to -0.4690 and -0.4143, respectively. Thus, the resulting hyperparameter values are eta= 0.5, subsample= 1, max_depth= 3, gamma= 0, and scale_pos_weight= 3.631. Interestingly, these are the same levels as the concluded levels from the original $2^{(5-1)}$ fractional factorial design. This means that the initial levels from the screening experiment seem to reduce variance and produce a stable model.

Factors eta (A), max_depth (C), and gamma (D) are chosen for further optimization, as they contributed to the strongest interactions (AC and AD). Thus, the path of steepest descent is applied to eta, max_depth, and gamma, to move the experiment closer to an optimum, in this case a minimum. The movement will be along the vectors where the slopes of the interaction effects AC and AD are decreased. The step sizes are based on the magnitude of the coefficient estimate of factor C being almost 8 times larger than the coefficient estimates of factors A and D. However, it is again troublesome to translate this into appropriate units. The coefficient estimate of interaction AC is 1.1339, and eta and max_depth were set to high and low levels, respectively. Therefore, eta is increased from below the high level 0.5, from 0.4 to 0.675, with step size 0.025. Factor C, max_depth, is decreased from above the low level, from 4 down to 1. The coefficient estimate of interaction AD is 1.1746, and gamma was set to the low level. Since A eta, is increased, gamma is decreased from above the low level, from 0.275 to 0 with step size -0.025. The result of the steepest descent is shown in Table 5.16. The resulting variance is calculated as the mean squared difference in the BACC scores, given the corresponding hyperparameter values, Eq.(4.1). Recall that each BACC score is calculated as the mean BACC of 5 runs of 10-fold cross-validation on the training set. The logarithm of the estimated variance is used as a response. Along the path, a minimum is found at eta= 0.525, max_depth=3, and gamma= 0.15, with value -25.8027.

Table 5.16: Path of steepest descent followed for eta, max_depth, and gamma in XGBoost estimating the in-run variance, while the other hyperparameters are held constant, that is subsample= 1 and scale_pos_weight= 3.631. The resulting variance is computed as the mean squared difference in the BACC scores, given the corresponding hyperparameter values, Eq.(4.1).

A (eta)	C (max_depth)	D (gamma)	Log of estimated variance
0.4	4	0.275	-17.8930
0.425	4	0.25	-12.1383
0.45	4	0.225	-14.4120
0.475	3	0.2	-12.5174
0.5	3	0.175	-11.3806
0.525	3	0.15	**-25.8027**
0.55	3	0.125	-12.3809
0.575	2	0.1	-14.9089
0.6	2	0.075	-15.8076
0.625	2	0.05	-14.8320
0.65	2	0.025	-14.9769
0.675	1	0	-14.2513

At this new proposed minimum, a second-order response surface model is fitted. A central composite design is applied with 3 center runs and $\alpha = \sqrt{3}$. For eta, the center is set to 0.525, with low and high levels 0.425 and 0.625, respectively. The axial points equal 0.3518 and 0.6982. For max_depth, the center is set to 3, with low and high levels 2 and 4, respectively. The axial points are set to 1 and 5 to ensure 5 different values. The center for gamma is set to 0.15, with low and high levels 0.05 and 0.25 with axial points equal to 0 and 0.3232. The CCD is presented in Appendix C.3, where again the response is the logarithm of the estimated in-run variance of the given hyperparameter configurations. The summary of the second-order model is presented in Appendix C.3. The p-value of the lack of fit equals 0.9286, meaning it is non-significant. None of the estimated coefficients are significant at level 0.05 or 0.1. The stationary point is found at eta= 0.5471, max_depth= $2.9498 \approx 3$, and gamma= 0.1732. Looking at the eigenvalues, they are all positive, meaning the stationary point is a minimum. This is verified by looking at contour plots of the response surface, Figure 5.11. A minimum is seen in all three plots. Additionally, the perspective plots of the response surface, presented in Appendix C.3 Figure C.9, display the same minimum. Thus, at the proposed minimum, that is eta= 0.5471, subsample= 1, max_depth= 3, gamma= 0.1732, and scale_pos_weight= 3.631, 5 runs of 10-fold cross-validation is performed on the training set. The resulting BACC scores are 0.6488, 0.6485, 0.6473, 0.6448, and 0.6488, with a mean of 0.6477. These scores do not vary much and are quite similar. This suggests that the found hyperparameter values reduce variance. Compared to the optimal hyperparameter values from RSM, Table 5.12, eta and gamma achieve a higher value and max_depth and scale_pos_weight a lower value. Increasing eta makes the model less conservative and decreasing max_depth reduces the complexity of the model, possibly resulting in a more stable model.

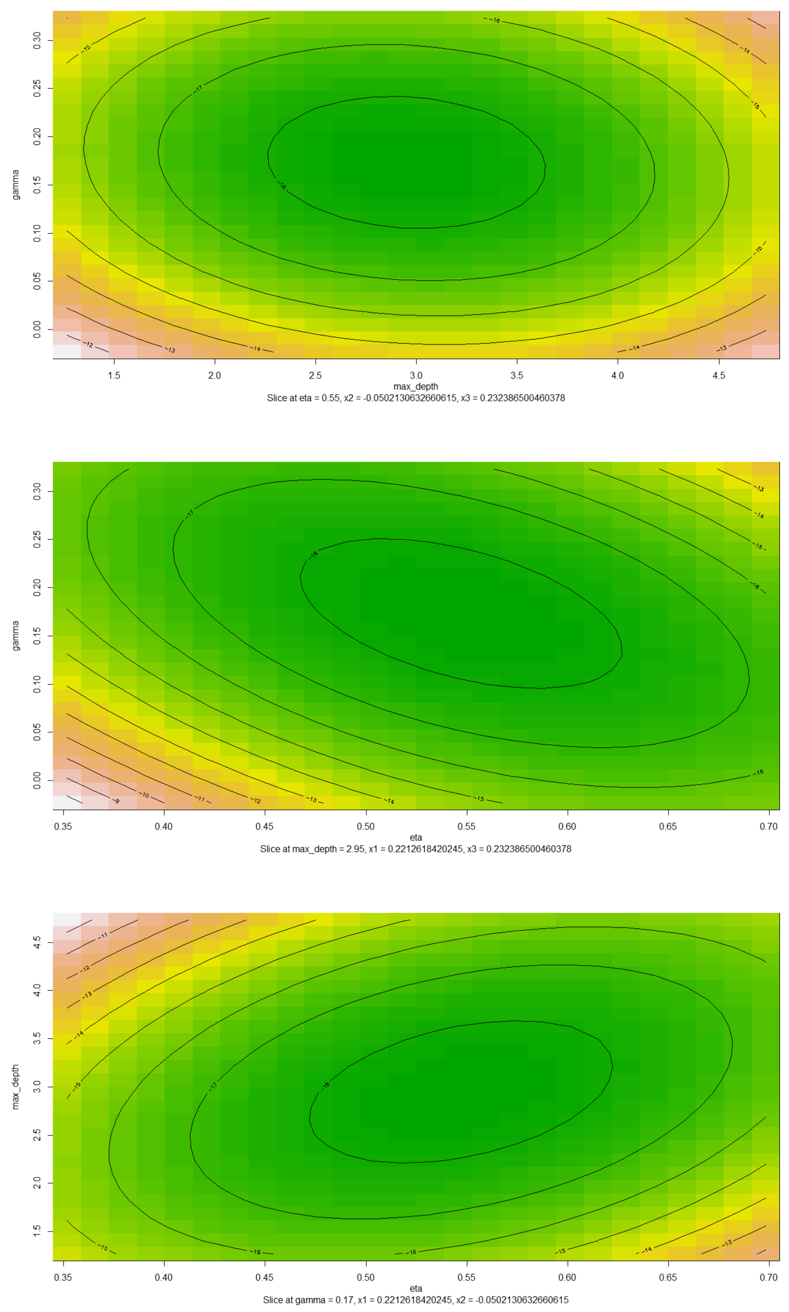


Figure 5.11: Contour plots of the fitted second-order response surface model with XGBoost minimizing the logarithm of the estimated in-run variance. The design has center in eta= 0.525, max_depth= 3, and gamma= 0.15.

The model is then trained on the training set and evaluated on the test set with default cut-off 0.5. The optimal cut-off is then calculated using the same procedure as before and is found to be 0.51 with a cost of false negatives equal to 300. The ROC and corresponding cost plot are displayed in Appendix C.3, Figure C.10. The model is then evaluated on the test set with this optimal cut-off. The results are displayed in Table 5.17 with corresponding confusion matrices, Table 5.18. With optimal cut-off, the sensitivity decreases from 0.5986 to 0.5629, which is not beneficial. This can also be seen in the number of wrongly predicted positive customers, which increases from 696 to 758. The specificity is higher with optimal cut-off, leading to a decrease in the number of wrongly predicted negative customers, from 1989 to 1789. However, the BACC has decreased from 0.6427 to 0.6406. Considering the positive class is of interest, the model with default cut-off would be the preferred one, now referred to as the variance model. This is an example of a situation where the algorithm that finds the optimal cut-off fails and does not lead to improvements in the results. Compared to the RSM model, Table 5.13, the sensitivity has decreased from 0.6915 to 0.5986, resulting in a lower BACC. This can also be seen in the confusion matrix, where the number of wrongly classified positive customers has increased from 535 to 696 for the variance model. The specificity is higher for the variance model. These findings suggest that minimizing the in-run variance does not lead to higher classification performance in this case, but possibly a more stable model. Minimizing the variance of BACC with some restriction on its mean or on sensitivity could be a path to investigate further. This involves robust design experimentation, but will not be pursued in this thesis.

Table 5.17: Results from classification metrics with XGBoost trained on the training set with tuned hyperparameters from direct variance modelling. The model is evaluated on the test set.

Cut-off	Sensitivity	Specificity	BACC	AUC	MCC
0.5	0.5986	0.6869	0.6427	0.643	0.2421
0.51	0.5629	0.7184	0.6406	0.641	0.2433

Table 5.18: Confusion matrix from the XGBoost model trained with tuned hyperparameters from direct variance modelling. The models are evaluated on the test set with default cut-off 0.5 and optimal cut-off 0.51.

5.18(a) Cut-off 0.5

Pred. / True	1	0
1	1038	696
0	1989	4363

5.18(b) Cut-off 0.51

Pred. / True	1	0
1	976	758
0	1789	4563

5.2.3 Bayesian optimization

Then, Bayesian optimization is applied to tune the hyperparameters of XGBoost. Three outputs from the Bayesian optimization procedure are displayed in Appendix C.4. Initially, all hyperparameters are tuned simultaneously. The ranges are chosen to be large, such that as much as possible of the parameter space is covered, Table 5.19. All three possible acquisition functions are used in the tuning: the Probability of Improvement (PI), the Expected Improvement (EI), and the GP Upper Confidence Bound (UCB). For PI and EI, the parameter ϵ must be tuned. For UCB, the parameter κ must be tuned. To tune both parameters, three values are chosen and, for each value, Bayesian optimization is run on all hyperparameters with 20 initial points and 50 iterations. The response is the mean BACC after 5-fold cross-validation on the training set. The parameter values that result in the highest BACC are kept as the optimal ones. Starting with UCB, the values of κ are varied around the default value, 2.576, the upper 0.5% quantile of the normal distribution. The low value is decided to be the 2.5% quantile of the normal distribution, that is 1.96. The high value is decided to be the 0.01% quantile of the normal distribution, that is 3.719. The results are shown in Table 5.20. The highest BACC score was found for $\kappa = 1.96$ equal to 0.6549, and κ will now be set to this throughout this thesis.

Table 5.19: The hyperparameters of XGBoost with corresponding range of values used in the Bayesian optimization. Note that max_depth is an integer.

Hyperparameter	Range
eta	[0.1,0.7]
subsample	[0.25,1]
max_depth	[3,9]
gamma	[0,1.5]
scale_pos_weight	[1,4]

Table 5.20: Bayesian optimization performed with XGBoost on the training set with acquisition function GP Upper Confidence Bound and hyperparameter values according to Table 5.19. Three values for κ are tested.

κ	eta	subsample	max_depth	gamma	scale_pos_weight	Best BACC
1.96	0.4729	0.8447	4	0.6297	3.4990	**0.6549**
2.576	0.3113	0.8619	6	1.4989	3.615	0.6538
3.719	0.2368	0.5318	5	1.3011	3.7906	0.6539

For EI and PI, the three values of ϵ are set to -0.001, 0, and 0.001. The chosen values are varied around the default value, 0. Since ϵ is added to the current found optimum in the process, the magnitude is chosen to be a thousandth. The best BACC scores with corresponding hyperparameter values for EI and PI are shown in Table 5.21 and Table 5.22, respectively. For both EI and PI, the optimal value for ϵ is 0, with BACC scores of 0.65443 and 0.6545, respectively. This means that exploration and exploitation is not adjusted. Throughout this section, this value for ϵ will be used in EI and PI.

Table 5.21: Bayesian optimization performed with XGBoost on the training set with acquisition function Expected Improvement and hyperparameter values according to Table 5.19. Three values for ϵ are tested.

ϵ	eta	subsample	max_depth	gamma	scale_pos_weight	Best BACC
-0.001	0.2762	0.9891	6	0.2594	3.3153	0.65441
0	0.2160	0.9543	6	0.2138	3.7112	**0.65443**
0.001	0.2881	0.8512	5	1.3348	3.9366	0.6531

Table 5.22: Bayesian optimization performed with XGBoost on the training set with acquisition function Probability of Improvement and hyperparameter ranges according to Table 5.19. Three values for ϵ are tested.

ϵ	eta	subsample	max_depth	gamma	scale_pos_weight	Best BACC
-0.001	0.1667	0.7222	6	0.1612	3.8294	0.6538
0	0.3953	0.9276	5	0.8320	3.4608	**0.6545**
0.001	0.4141	0.9333	5	0.3426	3.4197	0.6526

All three acquisition functions find quite different optimal hyperparameter values. XGBoost is then trained with the found optimal hyperparameter values corresponding to the best BACC scores from UCB, EI, and PI, and evaluated on the test set first default cut-off. For each model, the optimal cost of false negatives and cut-off are calculated, presented in Appendix C.4, Figure C.11 (UCB), Figure C.12 (EI), and Figure C.13 (PI). All models end up with an optimal cost of false negatives equal to 400 and cut-offs 0.46, 0.48, and 0.46 for UCB, EI, and PI, respectively. The models are then evaluated on the test set with the optimal cut-offs, Table 5.23. The confusion matrices with optimal cut-off are displayed in Table 5.24, while the confusion matrices with default cut-off are displayed in Appendix C.4, Table C.4. For UCB, applying the optimal cut-off leads to an increase in the sensitivity, from 0.5952 to 0.7019, which is beneficial. However, due to a decrease in the specificity, the BACC is lower with optimal cut-off. This is because the costs of false negatives and false positives are adjusted, so the cost of making a false negative is higher than a false positive. Thus, at the cost of the majority class, the classification of the minority class improves. For EI, the optimal cut-off leads to an increase in the sensitivity, from 0.6188 to 0.6920, and an increase in the BACC. For PI, the optimal cut-off also increases the sensitivity from 0.5888 to 0.6915 in addition to an increase in the BACC. The corresponding confusion matrices, Table 5.24 and Table C.4, reflect these changes. The best BACC score equal to 0.6555 is obtained for the PI model with optimal cut-off. The highest sensitivity score equal to 0.7018 is obtained for UCB with optimal cut-off. This can be seen in the corresponding confusion matrix, Table 5.24, where UCB obtains the lowest number of wrongly predicted positive customers, equal to 517. The highest number of wrongly predicted negative customers is also obtained by UCB, equal to 2580, reflecting the lowest score for specificity, 0.5938. Compared to the RSM model, Table 5.13, the Bayesian models obtain equal or higher sensitivity, which is beneficial. Furthermore, the PI model obtains a slightly higher score for BACC equal to 0.6555, while the UCB and EI models obtain a lower score. The specificity scores are quite similar. Compared to the benchmark model with optimal cut-off, Table 5.4, the BACC scores are quite similar, but again the Bayesian models obtain higher sensitivity, which is preferable.

Table 5.23: Results from classification metrics with XGBoost trained on the training set with optimal hyperparameters found with Bayesian optimization. The models are evaluated on the test set.

Acquisition function	Cut-off	Sensitivity	Specificity	BACC	AUC	MCC
UCB ($\kappa = 1.96$)	0.5	0.5952	0.7037	0.6494	0.649	0.2555
	0.46	0.7018	0.5938	0.6478	0.648	0.2432
EI ($\epsilon = 0$)	0.5	0.6188	0.6832	0.6510	0.651	0.2552
	0.48	0.6920	0.6138	0.6529	0.653	0.2523
PI ($\epsilon = 0$)	0.5	0.5888	0.7184	0.6536	0.654	0.2648
	0.46	0.6915	0.6195	0.6555	0.655	0.2567

Table 5.24: The confusion matrices corresponding to XGBoost trained on the training set with optimal hyperparameters found with Bayesian optimization with optimal cut-off. The models are evaluated on the test set.

5.24(a) UCB (cut-off 0.46)

True \ Pred.	1	0
1	1217	517
0	2580	3772

5.24(b) EI (cut-off 0.48)

True \ Pred.	1	0
1	1200	534
0	2453	3899

5.24(c) PI (cut-off 0.46)

True \ Pred.	1	0
1	1199	535
0	2417	3935

Empirical bootstrap confidence intervals for sensitivity, specificity, and balanced accuracy are calculated for XGBoost with tuned hyperparameters from Bayesian optimization with optimal cut-off values, Table 5.25. The intervals are created based on 1000 bootstrap samples of 30 measures of sensitivity, specificity, and balanced accuracy evaluated on the test set. The models chosen are those that obtained the highest score for BACC in the full Bayesian optimization for the three acquisition functions, UCB, EI, and PI. Looking at the intervals and the obtained results, Table 5.23, many of the scores lie outside the confidence intervals. The BACC scores for EI and PI lie above the upper confidence bound for BACC, suggesting that the obtained results on the test set may be too optimistic. For UCB, the BACC score is below the lower confidence bound for BACC. Additionally, the score for sensitivity for UCB is higher than the upper confidence bound, whereas for EI the score is lower than the lower bound. This suggests that the scores from the evaluation metrics should not be trusted blindly, since the same hyperparameter values can yield quite different models each time they are built. Therefore, one model can be "luckier" than another. This applies especially when subsample is not equal to 1, which is the case for all the trained models from the full Bayesian optimization. The highest range between the intervals of BACC is found for EI and PI, equal to 0.0026.

Table 5.25: Empirical bootstrap confidence intervals for sensitivity, specificity, and BACC calculated for XGBoost trained on the training set with optimal hyperparameters found with Bayesian optimization. The models are evaluated on the test set with optimal cut-off.

Acquisition function	Cut-off	Sensitivity	Specificity	BACC
UCB ($\kappa = 1.96$)	0.46	[0.6883, 0.7001]	[0.6008, 0.6138]	[0.6494, 0.6518]
EI ($\epsilon = 0$)	0.48	[0.6933, 0.7005]	[0.5988, 0.6059]	[0.6485, 0.6511]
PI ($\epsilon = 0$)	0.46	[0.6903, 0.6983]	[0.6028, 0.6114]	[0.6494, 0.6520]

Bayesian optimization combined with Design of Experiments

Bayesian optimization is then combined with the results from the initial screening experiment, that is the $2^{(5-1)}$ fractional factorial design for XGBoost. This is done to check the effect of screening on Bayesian optimization, especially to see if reduction of the dimension of the hyperparameter space can lead to better exploration. The factors eta (A), max_depth (C), and scale_pos_weight (E) were chosen for further optimization, while the other factors were set to specific values. Therefore, Bayesian optimization is performed on eta, max_depth, and scale_pos_weight, with ranges displayed in Table 5.26. Due to convergence issues, EI only manages to perform 31 iterations, as opposed to the original 70. One potential reason for this is that EI gets stuck at similar values, leading to a singular covariance matrix, possibly because ϵ is set to 0. The found optimal hyperparameter values are shown in Table 5.27 for the three acquisition functions. The best BACC score on the training set is found for PI, equal to 0.6557. The acquisition functions find different values for eta and scale_pos_weight, while max_depth ends up at the default equal to 6 for both UCB and PI.

Table 5.26: The hyperparameter values used in Bayesian optimization with XGBoost in combination with the results of the $2^{(5-1)}$ fractional factorial design. Only the hyperparameters eta, max_depth, and scale_pos_weight are optimized.

eta	subsample	max_depth	gamma	scale_pos_weight
[0.1,0.7]	1	[3,9]	0	[1,4]

Table 5.27: Bayesian optimization with XGBoost in combination with the results of the $2^{(5-1)}$ fractional factorial design. Only the hyperparameters eta, max_depth, and scale_pos_weight are optimized.

Acquisition function	eta	subsample	max_depth	gamma	scale_pos_weight	Best BACC
UCB ($\kappa = 1.96$)	0.2796	1	6	0	3.5152	0.6534
EI ($\epsilon = 0$)	0.6703	1	5	0	3.7736	0.6533
PI ($\epsilon = 0$)	0.3574	1	6	0	3.8474	0.6557

XGBoost is then trained with the found optimal values and evaluated on the test set with default cut-off. Again, the optimal cost of false negatives and cut-off are calculated for the three models, presented in Appendix C.4, Figure C.14 (UCB), Figure C.15 (EI), and Figure C.16 (PI). All three models end up with an optimal cost of false negatives equal to 400. For UCB the optimal cut-off is 0.47, while the optimal cut-off for EI and PI is equal to the default 0.5. This suggests that these two models correctly balance the class weights through the hyperparameter values. The UCB model is evaluated on the test set with optimal cut-off. The results are shown in Table 5.28, with corresponding confusion matrices displayed in Table 5.29. Applying the optimal cut-off for UCB increases the sensitivity from 0.5928 to 0.6995, leading to an increase in the BACC from 0.6456 to 0.6567, which is beneficial. The confusion matrix corresponding to the default cut-off for UCB is presented in Appendix C.4, Table C.5, showing that the number of wrongly classified positive customers has decreased from 706 to 521.

The highest score for BACC is obtained by EI, equal to 0.6576. The EI model also achieves the highest score for AUC and MCC. The UCB model with optimal cut-off obtains the highest score for sensitivity, 0.6995. This can be seen in the confusion matrices, where UCB wrongly predicts 521 positive customers as negative. The PI model obtains the highest number of wrongly predicted positive customers equal to 608, which reflects the low sensitivity score. However, the PI model obtains the highest score for specificity and wrongly predicts 2191 negative customers as positive. Compared to the full optimization, Table 5.23, the sensitivity scores are now lower. However, the BACC scores are slightly higher, and the EI model obtains the highest score for BACC for all XGBoost models, equal to 0.6576. Considering that the positive class is of interest, the UCB model with optimal cut-off would be the preferred one. This model obtains a high score for sensitivity while maintaining a reasonable score for specificity, leading to a high BACC.

Table 5.28: Results from classification metrics with XGBoost trained on the training set with optimal hyperparameters (Table 5.27) found with Bayesian optimization in combination with the results of the $2^{(5-1)}$ fractional factorial design. The models are evaluated on the test set.

Acquisition function	Cut-off	Sensitivity	Specificity	BACC	AUC	MCC
UCB ($\kappa = 1.96$)	0.5	0.5928	0.6984	0.6456	0.646	0.2484
	0.47	0.6995	0.6138	0.6567	0.657	0.2584
EI ($\epsilon = 0$)	0.5	0.6747	0.6404	0.6576	0.658	0.2615
PI ($\epsilon = 0$)	0.5	0.6494	0.6551	0.6522	0.652	0.2540

Table 5.29: The confusion matrices corresponding to XGBoost trained on the training set with optimal hyperparameters (Table 5.27) found with Bayesian optimization in combination with the results of the $2^{(5-1)}$ fractional factorial design. The models are evaluated on the test set with optimal cut-off.

5.29(a) UCB (cut-off 0.47)

True \ Pred.	1	0
1	1213	521
0	2453	3899

5.29(b) EI (cut-off 0.5)

True \ Pred.	1	0
1	1170	564
0	2284	4068

5.29(c) PI (cut-off 0.5)

True \ Pred.	1	0
1	1126	608
0	2191	4161

Bayesian optimization combined with Response Surface Methodology

Bayesian optimization is combined with Response Surface Methodology for XGBoost by setting the central composite design as an initial grid. The motivation behind this is that applying the CCD as an initial grid can possibly improve the exploitation, compared to a smooth second-order polynomial. The optimization is again performed to the variables eta, max_depth, and scale_pos_weight with the CCD as in Table 5.10. From the results of the response surface methodology, a maximum was found for a second-order response surface model with a CCD with center in eta= 0.48145, max_depth= 5, and scale_pos_weight= 3.7283. This design is applied as an initial grid in the optimization. The ranges are set similarly to Table 5.26. The found optimal hyperparameter values for the three acquisition functions are shown in Table 5.30. The three acquisition functions end up with quite similar values, suggesting that using the CCD as an initial grid leads to a more stable optimization. EI and UCB both find a value for eta around 0.3 and max_depth equal to the default 6. The found values for scale_pos_weight varies more. Compared to the found optimal hyperparameter values from RSM, Table 5.12, none of the values found are an exact match. UCB obtains the closest value for scale_pos_weight equal to 3.66. This indicates that RSM finds a different optimum than the Bayesian procedure. It is possible that the optimum is almost flat. The hyperparameter values for UCB are almost similar to the values obtained with Bayesian optimization combined with DoE, Table 5.28, only with a higher value for scale_pos_weight. The best BACC score on the training set is obtained by EI, equal to 0.6547.

Table 5.30: Bayesian optimization with the central composite design, Table 5.10, of XGBoost as initial grid. Only the hyperparameters eta, max_depth, and scale_pos_weight are optimized.

Acquisition function	eta	subsample	max_depth	gamma	scale_pos_weight	Best BACC
UCB ($\kappa = 1.96$)	0.2904	1	6	0	3.6626	0.6544
EI ($\epsilon = 0$)	0.3220	1	6	0	3.8572	0.6539
PI ($\epsilon = 0$)	0.2360	1	7	0	3.5273	0.6547

XGBoost is then trained with the found optimal values and evaluated on the test set first with default cut-off value. The optimal cost of false negatives and corresponding cut-off are calculated as before, presented in Appendix C.4, Figure C.17 (UCB), Figure C.18 (EI), and Figure C.19 (PI). The cost of false negatives is found to be 400 for EI and PI with optimal cut-off equal to 0.49, while for UCB the cost of false negatives is found to be 300 with optimal cut-off equal to 0.51. The models are then evaluated on the test set with the optimal cut-off value, Table 5.31 with corresponding confusion matrices, Table 5.32. The confusion matrices corresponding to the default cut-off are presented in Appendix C.4, Table C.6. Applying the optimal cut-off increases the sensitivity for both PI and EI. This can also be seen in the change in the number of wrongly predicted positive customers, which decreases for the two models. For UCB, the sensitivity decreases from 0.6292 to 0.6044, which is not favorable. This is a consequence of the cut-off being decided based on maximizing the BACC and not the sensitivity. In this situation, the adjustment of the cost of FN and FP has not served its purpose; increasing the sensitivity over the specificity. Nonetheless, for all models, the BACC increases, with the highest score equal to 0.6548 obtained for PI. The highest sensitivity score is obtained for EI with optimal cut-off equal to 0.6701. This is verified by the confusion matrices, where EI obtains the lowest number of wrongly predicted positive instances, equal to 572. The UCB model with optimal cut-off obtains the highest score for specificity, 0.7015, reflected by the lowest number of wrongly predicted negative instances, equal to 1986. Compared to the results from the full Bayesian optimization, Table 5.23, and Bayesian optimization combined with DoE, Table 5.27, the scores do not stand out. The sensitivity scores are a bit lower, and the BACC scores are all quite similar. Using the CCD as an initial grid did not improve the optimization, in this case. It is possible this is a coincidence, since the hyperparameter ranges are set similarly in the Bayesian optimization combined with DoE.

Table 5.31: Results from classification metrics with XGBoost trained on the training set with optimal hyperparameters (Table 5.30) found with Bayesian optimization with the central composite design as initial grid. The models are evaluated on the test set.

Acquisition function	Cut-off	Sensitivity	Specificity	BACC	AUC	MCC
UCB ($\kappa = 1.96$)	0.5	0.6292	0.6749	0.6520	0.652	0.2558
	0.51	0.6044	0.7015	0.6529	0.653	0.2609
EI ($\epsilon = 0$)	0.5	0.6459	0.6492	0.6476	0.648	0.2459
	0.49	0.6701	0.6269	0.6485	0.649	0.2458
PI ($\epsilon = 0$)	0.5	0.5969	0.7108	0.6538	0.654	0.2638
	0.49	0.6344	0.6752	0.6548	0.655	0.2604

Table 5.32: The confusion matrices corresponding to XGBoost trained on the training set with optimal hyperparameters (Table 5.30) found with Bayesian optimization with the central composite design as initial grid. The models are evaluated on the test set with optimal cut-off.

5.32(a) UCB (cut-off 0.51)

True \ Pred.	1	0
1	1048	686
0	1896	4456

5.32(b) EI (cut-off 0.49)

True \ Pred.	1	0
1	1162	572
0	2370	3982

5.32(c) PI (cut-off 0.49)

True \ Pred.	1	0
1	1100	634
0	2063	4289

5.3 Random Forests Modelling

The function `randomForest()` is implemented based on Breiman and Cutler's original Fortran code, [9]. The algorithm contains several hyperparameters that can be tuned. Only 5 hyperparameters are chosen for tuning, displayed with default values in Table 5.33. The hyperparameter mtry is the number of variables randomly sampled as candidates in each split. For classification, the default is $\sqrt{p}$, where p is the number of variables. For this dataset, $p = 30$. A small value for mtry saves computational time, but can make the model prone to overfitting. If mtry is large, Random Forests can resemble original decision trees, where mtry equals the number of variables. Cutoff is a vector of length equal to the number of classes that decides the predicted class for an observation. An observation is predicted as the class with the maximum ratio of proportion of votes to cutoff. The default value is $1/k$ where k is the number of classes, and in this way the majority vote wins. Ntree gives the number of trees to grow in the model. Since Random Forests does not overfit, it is not critical to set a high value, but it will lead to a high computational time. However, it should not be set to a too small value, to ensure that every input row is predicted at least a few times. The nodesize controls the maximum size of terminal nodes. A large nodesize causes smaller trees to be grown leading to less computational time. Replace is a Boolean variable indicating whether sampling of observations should be done with or without replacement.

Table 5.33: The hyperparameters of Random Forests chosen for tuning with default values.

Hyperparameter	Default value
mtry	$\sqrt{30} \approx 5$
cutoff	(0.5,0.5)
ntree	500
nodesize	1
replace	TRUE

Initially, Random Forests is run on the training set with default values for the hyperparameters and evaluated on the test set. The results are displayed in Table 5.34 with corresponding confusion matrix displayed in Table 5.35. The model struggles to identify the positive class, which can be seen in both the low sensitivity score equal to 0.1684 and the number of wrongly predicted positive instances equal to 1442. The low sensitivity score also reflects the BACC equal to 0.5737, which is not much better than random. The specificity is close to 1, meaning the model classifies almost all instances as the negative class. The score for MCC is surprisingly high, considering the poor classification of the positive class.

Table 5.34: Benchmark results from classification metrics with Random Forests trained on the training set with default hyperparameters. The model is evaluated on the test set.

Hyperparameter values	Sensitivity	Specificity	BACC	AUC	MCC
Default	0.1684	0.9789	0.5737	0.574	0.2706

Table 5.35: Confusion matrix from the Random Forests model trained with default hyperparameters. The model is evaluated on the test set.

Pred. / True	1	0
1	292	1442
0	134	6218

5.4 Hyperparameter tuning Random Forests

As before, the hyperparameters of Random Forests are first tuned using Response Surface Methodology. A screening experiment is performed using Design of Experiment, followed by the method of steepest ascent to move the experiment region closer to an optimum. Here, a response surface is fitted. Lastly, the hyperparameters are tuned using Bayesian optimization. All the tuning is done on the training set.

5.4.1 Optimization through Response Surface Methodology

First, an initial screening experiment is performed using Design of Experiments. The low and high levels of the hyperparameters need to be decided. Mtry has default value 5, and the levels are decided to be 5 ± 3. For cutoff, the default is $(0.5, 0.5)$, and the low level is decided to be $(0.2, 0.8)$ and the high level is decided to be $(0.8, 0.2)$. The default value for ntree is 500, and the levels are decided to be 500 ± 250. Nodesize has default 1, and the low and high levels are decided to be 1 and 5, respectively. The low level of replace is set to FALSE and high level to TRUE. The factor names and low and high levels of the hyperparameters are shown in Table 5.36.

Table 5.36: The factor names and low and high levels of the hyperparameters of Random Forests.

Factor	Hyperparameter	Low level (-1)	High level (+1)
A	mtry	2	8
B	cutoff	(0.2,0.8)	(0.8,0.2)
C	ntree	250	750
D	nodesize	1	5
E	replace	FALSE	TRUE

The $2^{(5-1)}$ fractional factorial design is then performed according to Table 2.2 with two replicates. The resulting BACC is the mean BACC after 5 runs of 10-fold cross-validation on the training set. The results after both replicates are presented in Appendix D.1, Table D.1. The mean BACC scores after the two replicates with the given hyperparameter configurations are shown in Table 5.37. The results vary around two values, 0.5 and 0.65, with the highest score equal to 0.6612 obtained at level code *bce*. The values change when factor B is moved from low to high level, already suggesting that this is the most influential factor. A linear model is fitted to the results of the two replicates of the experiment, with the summary of the model coefficients displayed in Table 5.38. The full model summary is displayed in Appendix D.1. There are 10 coefficient estimates that are significant at level 0.05. From the summary, factor B and interaction effect AB have the most significant estimates, according to the p-values. Looking at the coefficient estimates, factor B and interaction effect AB have larger estimates than the others, equal to 0.07573 and -0.003653. The Q-Q plot of the residuals, displayed in Appendix D.1, Figure D.2, shows some deviations from the line. The residuals, Figure D.3, are clearly separated into two groups which is normal in the presence of one strong, influential factor. This coincides with the BACC scores varying between two values which changes when factor B is moved from low to high level. The residuals in the left group with a fitted score around 0.5 exhibit low variance, while the residuals in the right group with a score around 0.65 exhibit higher variance. Therefore, both plots show that the assumption of independent normal errors holds. This suggests that a linear model is a good fit to the data.

Table 5.37: The mean BACC score after the two replicates of Random Forests trained on the training set using the $2^{(5-1)}$ fractional factorial design. The values of the factors A, B, C, D, and E are set to the values in Table 5.36.

Run	A	B	C	D	E = ABCD	Level code	Mean BACC
1	-1	-1	-1	-1	1	1	0.5005
2	1	-1	-1	-1	-1	a	0.5087
3	-1	1	-1	-1	-1	b	0.6561
4	1	1	-1	-1	1	abe	0.6484
5	-1	-1	1	-1	-1	c	0.5004
6	1	-1	1	-1	1	ace	0.5066
7	-1	1	1	-1	1	bce	0.6612
8	1	1	1	-1	-1	abc	0.6505
9	-1	-1	-1	1	-1	d	0.5002
10	1	-1	-1	1	1	ade	0.5062
11	-1	1	-1	1	1	bde	0.6594
12	1	1	-1	1	-1	abd	0.6522
13	-1	-1	1	1	1	cde	0.5000
14	1	-1	1	1	-1	acd	0.5078
15	-1	1	1	1	-1	bcd	0.6595
16	1	1	1	1	1	abcde	0.6550

Table 5.38: The model summary of the coefficients of the linear model fitted to the results of the $2^{(5-1)}$ fractional factorial design of Random Forests.

	Estimate	Std. Error	t value	Pr(> $\|t\|$)	
(Intercept)	5.795e-01	1.516e-04	3822.420	< 2e-16	***
A	-1.265e-04	1.516e-04	-0.834	0.416498	
B	7.573e-02	1.516e-04	499.504	< 2e-16	***
C	5.805e-04	1.516e-04	3.829	0.001479	**
D	4.963e-04	1.516e-04	3.273	0.004783	**
E	1.150e-04	1.516e-04	0.758	0.459303	
A:B	-3.653e-03	1.516e-04	-24.094	5.33e-14	***
A:C	-5.065e-05	1.516e-04	-0.334	0.742695	
A:D	3.686e-04	1.516e-04	2.431	0.027175	*
A:E	-4.838e-04	1.516e-04	-3.191	0.005683	**
B:C	6.832e-04	1.516e-04	4.506	0.000359	***
B:D	7.358e-04	1.516e-04	4.853	0.000176	***
B:E	6.001e-04	1.516e-04	3.958	0.001127	**
C:D	-4.112e-05	1.516e-04	-0.271	0.789682	
C:E	4.593e-04	1.516e-04	3.030	0.007972	**
D:E	-7.747e-06	1.516e-04	-0.051	0.959881	

The main effects of the factors are plotted in Figure 5.12, where only factor B has a slope not approximately equal to 0. Looking at the interactions, Figure 5.13, there is evidence of a small interaction between factors A and B. All other lines look very close to parallel, indicating there are no other interactions in the design. In the normal plot, Figure 5.14, factor B and the interaction effect AB are significant at level $\alpha = 0.05$. The interaction effect AB lies almost along the line with only a small deviation. Again, factor B is the most significant factor and falls off the line. The pareto plot, Appendix D.1, Figure D.1, shows that factors B and interaction AB are significant at level 0.05 according to Lenth's method.

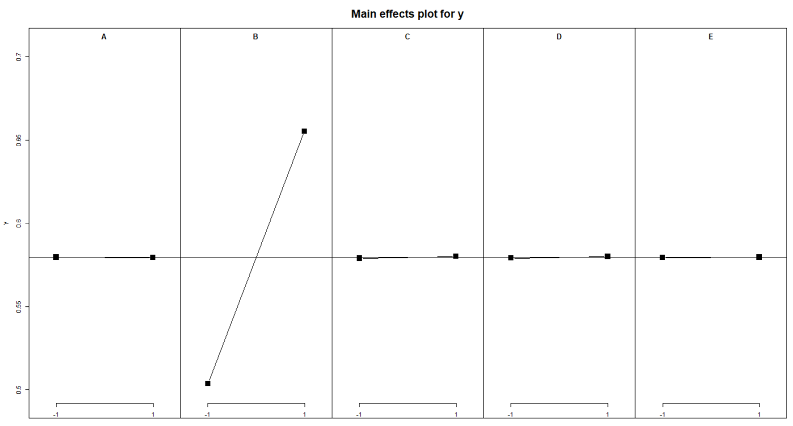


Figure 5.12: The main effects of factors A, B, C, D, and E after conducting the experiments in the $2^{(5-1)}$ fractional factorial design on the training set with Random Forests.

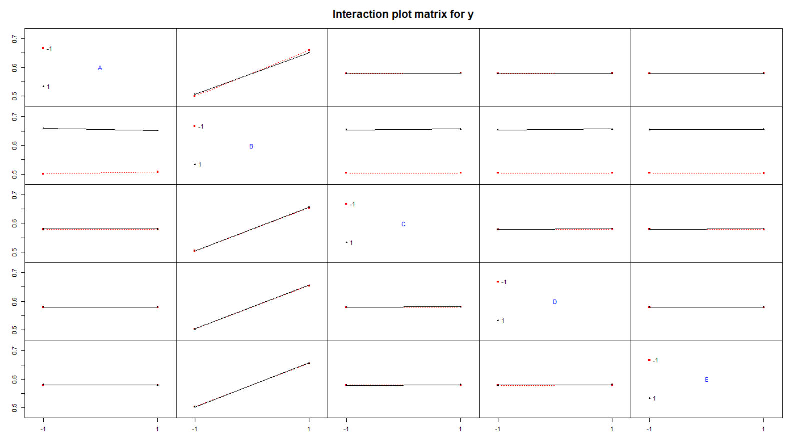


Figure 5.13: The interaction effects of factors A, B, C, D, and E after conducting the experiments in the $2^{(5-1)}$ fractional factorial design on the training set with Random Forests.

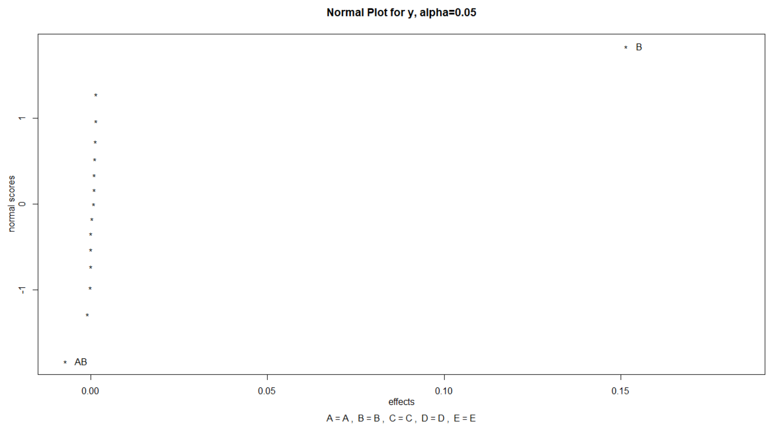


Figure 5.14: Normal plot of the estimates of the effects after conducting the experiments in the $2^{(5-1)}$ fractional factorial design on the training set with Random Forests.

Factor B is the most influential factor and is set on high level based on the main effects plot, Figure 5.13. The other hyperparameters are set to their default values since there are almost no contributions to main effects or interaction effects. That is mtry= 5 (A), ntree= 500 (C), nodesize= 1 (D), and replace=TRUE (E). The value of factor A, mtry, could be decided based on the interaction effect AB. However, since the interaction is so small, it is for now set to its default value.

Next, the path of steepest ascent is followed for factor B, cutoff, to move the experiment closer to an optimum. Factor B is chosen because it was the most influential in the screening experiment. The coefficient estimate for factor B is 0.07573, meaning the values for cutoff are chosen to gradually increase in the direction of the high level. Again, the response is the mean BACC after 5 runs of 10-fold cross-validation performed on the training set with the given hyperparameter values. The results from the path of steepest ascent, Table 5.39, show that a new optimum is found at cutoff= $(0.76, 0.24)$ with a BACC score equal to 0.6581. Here, 5 runs of 10-fold cross-validation is performed on the training set, resulting in the BACC scores of 0.6574, 0.6594, 0.6601, 0.6590, and 0.6594, with a mean of 0.6591.

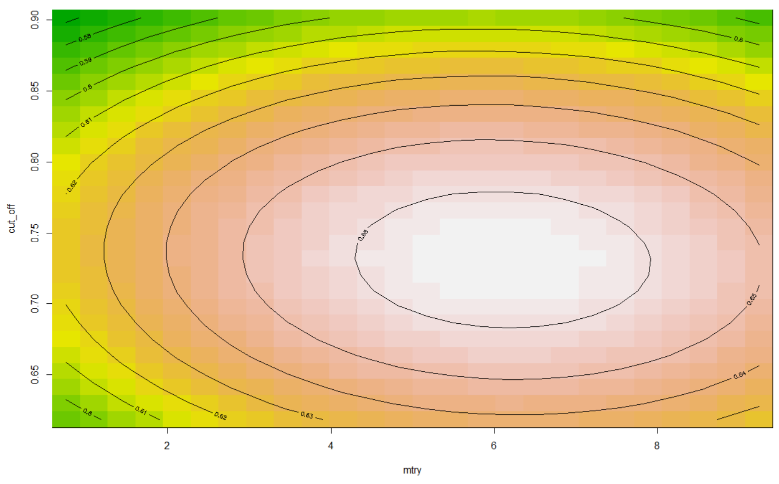
Table 5.39: Path of steepest ascent followed for cutoff, while the other hyperparameters are held constant. The resulting BACC is the mean of 5 runs of 10-fold cross-validation performed on the training set.

A (mtry)	B (cutoff)	C (ntree)	D (nodesize)	E (replace)	BACC
5	(0.75,0.25)	500	1	TRUE	0.6568
5	(0.76,0.24)	500	1	TRUE	**0.6581**
5	(0.77,0.23)	500	1	TRUE	0.6558
5	(0.78,0.22)	500	1	TRUE	0.6561
5	(0.79,0.21)	500	1	TRUE	0.6553
5	(0.80,0.20)	500	1	TRUE	0.6527
5	(0.81,0.19)	500	1	TRUE	0.6489
5	(0.82,0.18)	500	1	TRUE	0.6431
5	(0.83,0.17)	500	1	TRUE	0.6396
5	(0.84,0.16)	500	1	TRUE	0.6359
5	(0.85,0.15)	500	1	TRUE	0.6268

Around this new optimum for factor B, a second-order response surface model is fitted. Due to the small interaction effect AB, factor A (mtry) is also included in the second-order model. A central composite design is used with 3 center runs and $\alpha = \sqrt{2}$. For cutoff, the center is set to the found optimum, $(0.76, 0.24)$, with low and high levels $(0.66, 0.34)$ and $(0.86, 0.14)$, respectively. The axial points are equal to $(0.6186, 0.3214)$ and $(0.9014, 0.09856)$. Mtry is varied around the default value, with center in 5 and low and high levels 2 and 8, respectively. The axial points are set to 1 and 9. The CCD is displayed in Table 5.40 and the model summary of the response surface is displayed in Appendix D.2. The lack of fit has p-value 0.002478, meaning the lack of fit is significant at level 0.05. This suggests that the second-order model is not a good fit to the data. Additionally, none of the coefficient estimates are significant at level 0.05 or 0.1.

Table 5.40: The central composite designed obtained on the training set for the variables mtry and cutoff of Random Forest, with ntree= 500, nodesize= 1, and replace=TRUE.

Run	A (mtry)	B (cutoff)	BACC
1	-1	-1	0.6583
2	1	-1	0.6430
3	-1	1	0.6422
4	1	1	0.6203
5	$-\sqrt{2}$	0	0.5827
6	$\sqrt{2}$	0	0.6610
7	0	$-\sqrt{2}$	0.6251
8	0	$\sqrt{2}$	0.5881
9	0	0	0.6629
10	0	0	0.6594
11	0	0	0.6622



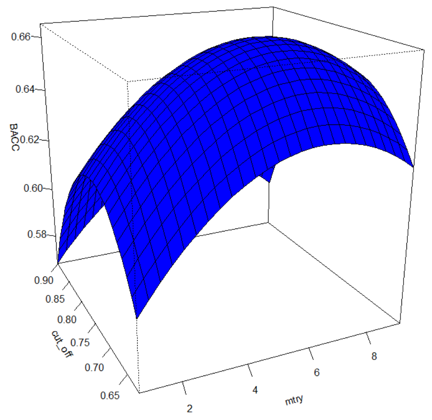


Figure 5.15: Contour plot (**top**) and perspective plot (**bottom**) of the fitted second-order response surface model with Random Forests. The design has center in cutoff= $(0.76, 0.24)$ and mtry= 5.

The stationary point is located at mtry= $6.1030 \approx 6$ and cutoff= $(0.7312, 0.2688)$. The corresponding eigenvalues are both negative, -0.01305 and -0.02086, meaning the stationary point is classified as a maximum. This is verified by looking at the contour and perspective plot of the response surface, Figure 5.15. The plots show a maximum around cutoff= $(0.75, 0.25)$ and mtry= 6, with decreasing values around the optimum. At the stationary point, 5 runs of 10-fold cross-validation is performed yielding the BACC scores of 0.6627, 0.6626, 0.6625, 0.6619, and 0.6638, with a mean of 0.6627. This is an improvement compared to previous results.

Table 5.41: Optimal values of the hyperparameters of Random Forests after optimization using RSM.

mtry	cutoff	ntree	nodesize	replace
6	$(0.7312, 0.2688)$	500	1	TRUE

The found optimal hyperparameter values are shown in Table 5.41. Random Forests is then trained on the training set and evaluated on the test set using the optimal hyperparameter values. The classification scores, Table 5.42, are improved compared to the benchmark results, Table 5.34. In particular, the sensitivity has increased from 0.1684 to 0.6133, resulting in an increase in the BACC from 0.5737 to 0.6661. This is beneficial and the model now classifies the data better.

The scores for AUC and MCC are also higher, reflecting better classification of the positive class. Especially, the MCC achieves a high score equal to 0.2859. The specificity has decreased from 0.9789 to 0.7189. The improvements can also be seen in the corresponding confusion matrix, Table 5.43, where the number of wrongly predicted positive customers has decreased from 1442 to 674. The number of wrongly predicted negative customers has increased from 134 to 1783, reflecting the decrease in specificity.

Table 5.42: Results from classification metrics with Random Forests trained on the training set with tuned hyperparameters using RSM, Table 5.41. The model is evaluated on the test set.

Hyperparameter values	Sensitivity	Specificity	BACC	AUC	MCC
Tuned (RSM)	0.6133	0.7189	0.6661	0.666	0.2859

Table 5.43: Confusion matrix from the Random Forests model trained with tuned hyperparameters using RSM, Table 5.41. The model is evaluated on the test set.

True \ Pred.	1	0
1	1069	674
0	1783	4560

5.4.2 Bayesian optimization

Next, the hyperparameters are tuned using Bayesian optimization. Initially, all hyperparameters are optimized simultaneously. The ranges of the hyperparameter values are again decided to vary around the default values, Table 5.44. As before, each of the three acquisition functions are tested: the GP Upper confidence bound (UCB), the Probability of Improvement (PI), and the Expected Improvement (EI), but first κ and ϵ are tuned. To tune both parameters, three values are chosen, and for each value Bayesian optimization is run on all hyperparameters with 10 initial points and 40 iterations. The values for ϵ and κ that result in the highest response on the training set are kept as the optimal one. The response is again the mean BACC after 5-fold cross-validation on the training set. The same three values of κ are tested, $\kappa = 1.96, 2.576, 3.719$. The results are shown in Table 5.45 where $\kappa = 2.576$ yields the highest BACC equal to 0.6654. This value for κ is used in UCB throughout this section.

Table 5.44: The hyperparameters of Random Forests with corresponding range of values used in the Bayesian optimization. Note that mtry, ntree, and nodesize are integers.

Hyperparameter	Range
mtry	[2,8]
cutoff	[(0.5,0.5), (0.9,0.1)]
ntree	[250,750]
nodesize	[1,5]
replace	[TRUE,FALSE]

Table 5.45: Bayesian optimization performed with Random Forests on the training set with acquisition function GP Upper Confidence Bound and hyperparameter values according to Table 5.44. Three values for κ are tested.

κ	mtry	cutoff	ntree	nodesize	replace	Best BACC
1.96	2	(0.7809,0.2191)	605	1	FALSE	0.6639
2.576	2	(0.7933,0.2067)	589	4	FALSE	**0.6654**
3.719	3	(0.7635,0.2365)	750	1	TRUE	0.6646

For EI and PI, the parameter ϵ must be tuned. Due to convergence issues, the value for ϵ is decreased compared to XGBoost. One probable reason for this is that Random Forests gives more stable results, while the results of XGBoost seem to vary more. Therefore, the optimization might get stuck at similar values. Thus, the three values for ϵ are now chosen to be -0.0001, 0, and 0.0001. The results for EI and PI are shown in Table 5.46 and Table 5.47, respectively, where again $\epsilon = 0$ gives the best BACC for EI and PI. This value for ϵ will be used for EI and PI throughout this section. Looking at the found optimal values for mtry and cutoff, they are almost identical for EI and PI. From previous results, these variables are the most influential, meaning both EI and PI result in quite similar models.

Table 5.46: Bayesian optimization performed with Random Forests on the training set with acquisition function Expected Improvement and hyperparameter values according to Table 5.44. Three values for ϵ are tested.

ϵ	mtry	cutoff	ntree	nodesize	replace	Best BACC
-0.0001	3	(0.7830,0.2170)	578	4	TRUE	0.6616
0	3	(0.7583,0.2417)	515	3	TRUE	**0.6642**
0.0001	4	(0.7422,0.2578)	459	1	TRUE	0.6640

Table 5.47: Bayesian optimization performed with Random Forests on the training set with acquisition function Probability of Improvement and hyperparameter values according to Table 5.44. Three values for ϵ are tested.

ϵ	mtry	cutoff	ntree	nodesize	replace	Best BACC
-0.0001	2	(0.8135, 0.1865)	637	5	TRUE	0.6621
0	3	(0.7509, 0.2491)	258	2	TRUE	**0.6639**
0.0001	4	(0.7494,0.2506)	426	5	FALSE	0.6637

Random Forests is then trained with the found optimal hyperparameter values corresponding to the best BACC scores from UCB, EI, and PI, and evaluated on the test set, Table 5.48. The scores are all quite similar because the found optimal values for mtry and cutoff are almost equal, in particular for EI and PI. The best scores in all metrics are obtained for PI, except for specificity, meaning the PI model is the best classifier of these three models. The PI model obtains a score of 0.6656 for BACC and a score of 0.6298 for sensitivity. The same conclusions can also be drawn from the corresponding confusion matrices, Table 5.49, where the numbers are again quite similar. The lowest number of wrongly predicted positive instances is obtained for PI equal to 642, reflecting the highest sensitivity score. The lowest number of wrongly predicted negative instances is obtained for UCB equal to 1885, reflecting the highest specificity score. Compared to the model with tuned hyperparameters using RSM, Table 5.42, the Bayesian models obtain a higher sensitivity, which is beneficial. The cutoff is higher in the Bayesian models than in the RSM model, leading to better classification of the positive class. The RSM model has the highest specificity score, leading to a higher BACC score. However, the differences are not large.

Table 5.48: Results from classification metrics with Random Forests trained on the training set with optimal hyperparameters found with Bayesian optimization. The models are evaluated on the test set.

Acquisition function	Sensitivity	Specificity	BACC	AUC	MCC
UCB ($\kappa = 2.576$)	0.6228	0.7032	0.6630	0.663	0.2777
EI ($\epsilon = 0$)	0.6275	0.6982	0.6628	0.663	0.2766
PI ($\epsilon = 0$)	0.6298	0.7015	0.6656	0.666	0.2817

Table 5.49: The confusion matrices corresponding to Random Forests trained on the training set with optimal hyperparameters found with Bayesian optimization. The models are evaluated on the test set.

5.49(a) UCB

True \ Pred.	1	0
1	1080	654
0	1885	4467

5.49(b) EI

True \ Pred.	1	0
1	1088	646
0	1917	4435

5.49(c) PI

True \ Pred.	1	0
1	1092	642
0	1896	4456

Empirical bootstrap confidence intervals for sensitivity, specificity, and balanced accuracy are calculated for Random Forests, Table 5.50. As before, the models chosen are those that obtained the highest score for BACC in the full Bayesian optimization for the three acquisition functions, UCB, EI, and PI. Looking at the confidence intervals and the obtained results, Table 5.48, many of the scores again lie outside the confidence bounds. For all models, the obtained scores for sensitivity and BACC are higher than the upper bound. Again, this means that the fitted models are too optimistic with respect to their performance given the hyperparameter values. This supports the conclusion of not trusting the evaluation metrics blindly. Compared to the confidence intervals for XGBoost, Table 5.25, the widths of all the intervals for Random Forests are smaller. This verifies that Random Forests yields more stable models than XGBoost. One reason for this is that Random Forests averages many trees when building a model, while XGBoost improves the residuals of the trees that are built.

Table 5.50: Empirical bootstrap confidence intervals for sensitivity, specificity, and BACC calculated for Random Forests trained on the training set with optimal hyperparameters found with Bayesian optimization. The models are evaluated on the test set.

Acquisition function	Sensitivity	Specificity	BACC
UCB ($\kappa = 2.576$)	[0.6198, 0.6225]	[0.6997, 0.7014]	[0.6602, 0.6615]
EI ($\epsilon = 0$)	[0.6222, 0.6248]	[0.7002, 0.7013]	[0.6614, 0.6627]
PI ($\epsilon = 0$)	[0.6177, 0.6210]	[0.7046,0.7065]	[0.6616, 0.6633]

Bayesian optimization combined with Design of Experiments

Bayesian optimization is then combined with the results from the initial screening experiment, the $2^{(5-1)}$ fractional factorial design for Random Forests. This is done to check the effect of screening on Bayesian optimization. The screening experiment found that factor B was the most influential, while the other factors were set to their default values. However, there was evidence of a small interaction effect, AB, which was significant in the normal plot, Figure 5.14. Thus, Bayesian optimization is performed on factor B, cutoff, and A, mtry, with ranges as in Table 5.51. The other hyperparameters are set to their default values, according to Table 5.33. This is done for the three acquisition functions, with tuned values for ϵ and κ. Due to convergence issues with EI, only a total of 19 iterations are performed, as opposed to the original 50. When increasing the number of iterations, the covariance matrix becomes singular, probably due to the algorithm generating too similar points. A potential reason can be that ϵ is set to 0. The optimized hyperparameter values are shown in Table 5.52. The best BACC on the training set, equal to 0.6633, is obtained by EI, showing that only a small number of iterations is needed to obtain good results. The three acquisition functions all find different values for mtry, but quite similar values for cutoff. In this case, UCB and PI result in almost the same model.

Table 5.51: The hyperparameter values used in Bayesian optimization with Random Forests in combination with the results of the $2^{(5-1)}$ fractional factorial design. Only the hyperparameters mtry and cutoff are optimized.

mtry	cutoff	ntree	nodesize	replace
[2,8]	[(0.5,0.5), (0.9,0.1))]	500	1	TRUE

Table 5.52: Bayesian optimization with Random Forests in combination with the results of the $2^{(5-1)}$ fractional factorial design. Only the hyperparameters mtry and cutoff are optimized.

Aqcuisition function	mtry	cutoff	ntree	nodesize	replace	Best BACC
UCB ($\kappa = 2.576$)	5	(0.7403,0.2597)	500	1	TRUE	0.6627
EI ($\epsilon = 0$)	2	(0.7740,0.2260)	500	1	TRUE	0.6633
PI ($\epsilon = 0$)	7	(0.7491,0.2509)	500	1	TRUE	0.6615

Random Forests is then trained with the found optimal hyperparameter values and evaluated on the test set, Table 5.53. The highest score for BACC, sensitivity, and AUC are again obtained for the PI model with a BACC equal to 0.6649. Compared to the scores from the full optimization, Table 5.48, PI now obtains a lower BACC but a higher sensitivity equal to 0.6592. This is beneficial considering the positive class is of interest. In general, the BACC scores are quite similar. The highest specificity is obtained for EI, equal to 0.7099, while PI obtains the lowest score equal to 0.6707. The corresponding confusion matrices, Table 5.54, confirm the same conclusion. PI has the lowest number of wrongly predicted instances, 591, while EI has the lowest number of wrongly predicted negative instances, 1843.

Table 5.53: Results from classification metrics with Random Forests trained on the training set with optimal hyperparameters (Table 5.52) found with Bayesian optimization in combination with the results of the $2^{(5-1)}$ fractional factorial design. The models are evaluated on the test set.

Acquisition function	Sensitivity	Specificity	BACC	AUC	MCC
UCB ($\kappa = 2.576$)	0.6424	0.6837	0.6631	0.663	0.2750
EI ($\epsilon = 0$)	0.6142	0.7099	0.6620	0.662	0.2771
PI ($\epsilon = 0$)	0.6592	0.6707	0.6649	0.665	0.2763

Table 5.54: The confusion matrices corresponding to Random Forests trained on the training set with optimal hyperparameters (Table 5.52) found with Bayesian optimization in combination with the results of the $2^{(5-1)}$ fractional factorial design. The models are evaluated on the test set.

5.54(a) UCB

True \ Pred.	1	0
1	1114	620
0	2009	4343

5.54(b) EI

True \ Pred.	1	0
1	1065	669
0	1843	4509

5.54(c) PI

True \ Pred.	1	0
1	1143	591
0	2092	4260

Bayesian optimization combined with Response Surface Methodology

Bayesian optimization is combined with Response Surface Methodology for Random Forests by setting the central composite design as an initial grid. The optimization is done on mtry and cutoff, with the CCD as in Table 5.40. From the results of the response surface methodology, a maximum was found for a second-order response surface model with CCD with center in mtry= 5 and cutoff= $(0.76, 0.24)$. This design is applied as an initial grid in the optimization. The value ranges are the same as before, Table 5.51. The found optimal hyperparameter values are shown in Table 5.55, once more with quite similar values for mtry and cutoff. Again, this supports that using the CCD as an initial grid might result in a more stable optimization for all three acquisition functions. Interestingly, UCB finds the center point as the optimal value. For both UCB and EI, mtry is equal to the default value, 5. The cutoff values do not differ much, and EI and PI find almost equal values. Compared to the obtained optimal values from RSM, Table 5.41, the cutoffs are quite similar and vary around $(0.73, 0.27)$. The found optimal values for mtry are lower than the one obtained using RSM. The highest BACC score on the training set is obtained by UCB equal to 0.6626.

Table 5.55: Bayesian optimization with the central composite design of Random Forests as initial grid. Only the hyperparameters mtry and cutoff are optimized.

Aqcuisition function	mtry	cutoff	ntree	nodesize	replace	Best BACC
UCB ($\kappa = 2.576$)	5	(0.76, 0.24)	500	1	TRUE	0.6626
EI ($\epsilon = 0$)	5	(0.7308, 0.2692)	500	1	TRUE	0.6612
PI ($\epsilon = 0$)	4	(0.7324, 0.2676)	500	1	TRUE	0.6616

Random Forests is then trained with the found optimal hyperparameter values, and evaluated on the test set, Table 5.56. Now, UCB obtains the highest score for sensitivity, BACC, and AUC, with a BACC equal to 0.6671 and a sensitivity equal to 0.6840. Compared to the results from the full Bayesian optimization, Table 5.48, and the Bayesian optimization combined with DoE, Table 5.52, these are the highest scores for sensitivity and BACC. PI obtains the highest score for specificity, while EI obtains the highest score for MCC. EI and PI obtain almost the same scores in all metrics, reflecting that the models have very similar hyperparameter values. The corresponding confusion matrices, Table 5.43, show that UCB has the lowest number of wrongly predicted positive instances, equal to 548. PI has the lowest number of wrongly predicted negative instances, equal to 1793. Compared to the scores from the RSM model, Table 5.42, the sensitivity and BACC scores are higher for the UCB model. This suggests that the UCB model is the preferred model.

Table 5.56: Results from classification metrics with Random Forests trained on the training set with optimal hyperparameters (Table 5.55) found with Bayesian optimization with the central composite design as initial grid. The models are evaluated on the test set.

Aquisition function	Sensitivity	Specificity	BACC	AUC	MCC
UCB ($\kappa = 2.576$)	0.6840	0.6502	0.6671	0.667	0.2777
EI ($\epsilon = 0$)	0.6101	0.7152	0.6627	0.663	0.2791
PI ($\epsilon = 0$)	0.6050	0.7177	0.6613	0.661	0.2774

Table 5.57: The confusion matrices corresponding to Random Forests trained on the training set with optimal hyperparameters (Table 5.55) found with Bayesian optimization with the central composite design as initial grid. The models are evaluated on the test set.

5.57(a) UCB

True \ Pred.	1	0
1	1186	548
0	2222	4130

5.57(b) EI

True \ Pred.	1	0
1	1058	676
0	1809	4543

5.57(c) PI

True \ Pred.	1	0
1	1049	685
0	1793	4559

5.5 Feature Importance

Feature importance is investigated for both XGBoost and Random Forests, before and after tuning. Only the 15 most important features are displayed. For XGBoost, feature importance is calculated with default values for the hyperparameters (Table 5.1), with tuned hyperparameters from RSM (Table 5.12), and with tuned hyperparameters from Bayesian optimization in combination with the results of the $2^{(5-1)}$ fractional factorial design with acquisition function EI (Table 5.27). The Bayesian model chosen was based on the best score for BACC on the test set out of all the Bayesian models. Feature importance before and after tuning with RSM are displayed in Figure 5.16. Feature importance calculated for the Bayesian model is displayed in Appendix C.5, Figure C.20. The top two features are the same, that is INTEREST_EARNING_LENDING_AMT and MonthsAgo, meaning these two are considered to have a significant impact on the response. Recall that these variables describe the interest earning balance (or the amount not paid in full last statement) and the number of months since the call activity started, respectively. The variables revUtil, P_REFIN, and weeknr are all among the next 5 important variables for all models, meaning they can also be said to be influential. There are some differences in the order, indicating differences between the three models. Additionally, the default model and the RSM model consider the variable Segment23Name_Closed as important, while the RSM model and the Bayesian model consider the variable revUtilL12 important. However, given that many of the top 7 variables are similar, tuning does not seem to impact the calculated feature importances too much.

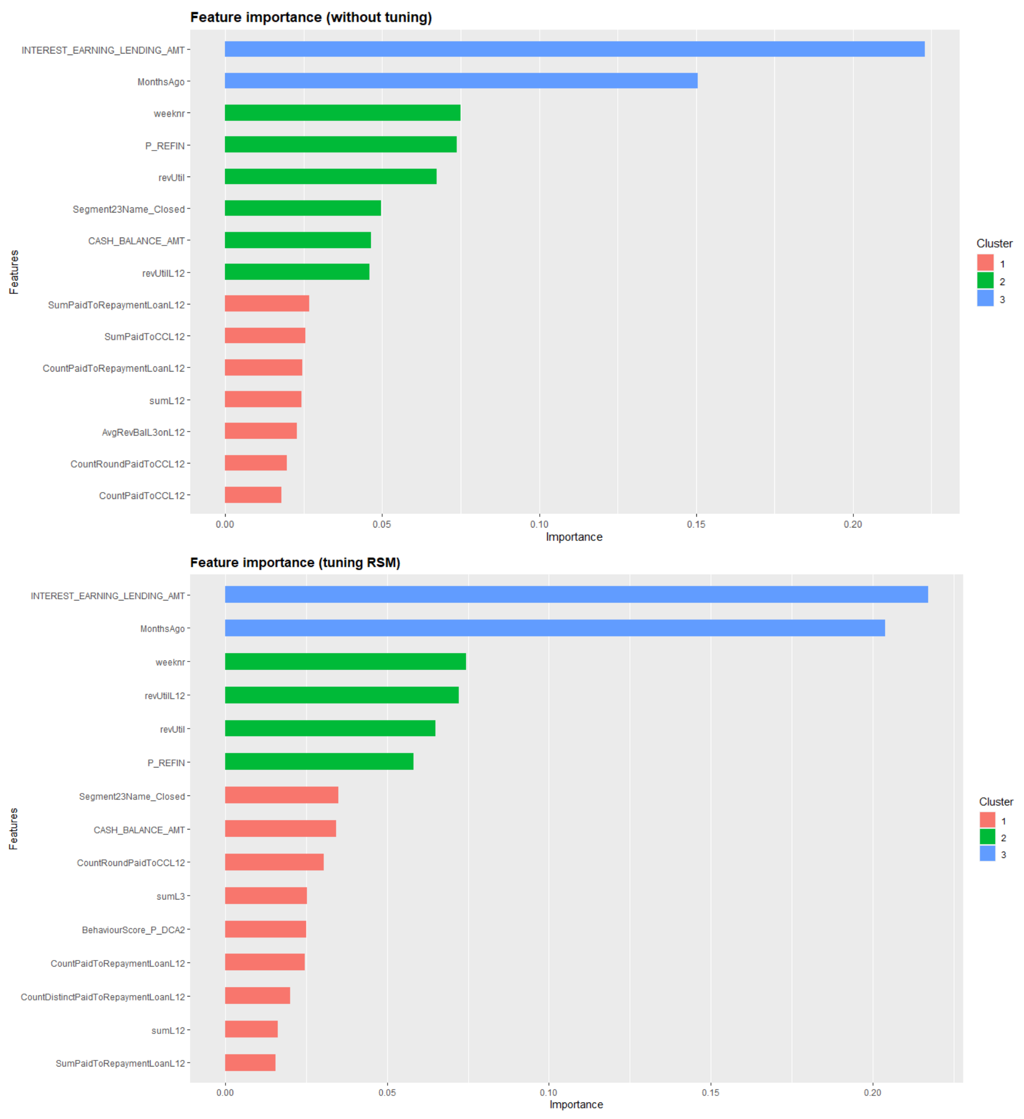
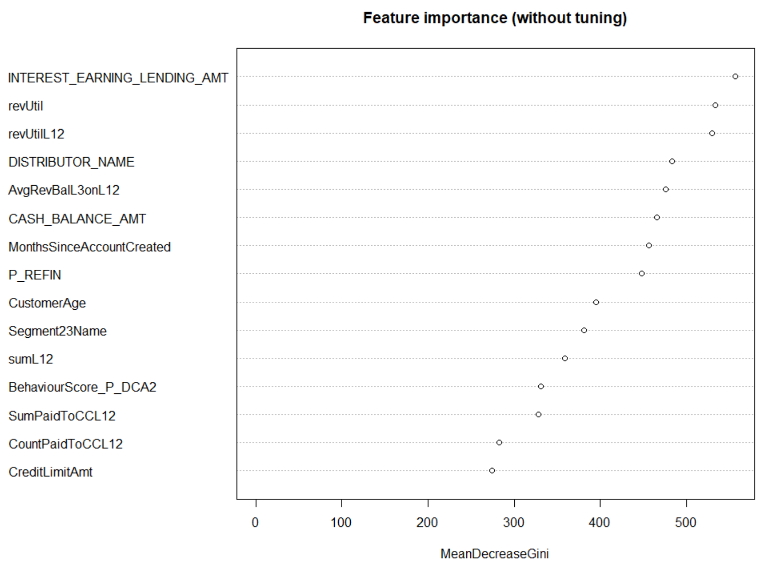


Figure 5.16: Feature importance calculated for XGBoost trained on the training set with **Top:** default hyperparameters, Table 5.1. **Bottom:** tuned hyperparameters from RSM, Table 5.12.

Feature importance is similarly calculated for Random Forests with default hyperparameters, with tuned hyperparameters from RSM (Table 5.41), and with tuned hyperparameters from Bayesian optimization combined with RSM with acquisition function UCB (Table 5.55). Again, the Bayesian model was chosen based on the best score for BACC on the test set. Feature importance calculated for the default model and the model from RSM are displayed in Figure 5.17. Feature importance calculated for the Bayesian model is displayed in Appendix D.3, Figure D.4, showing equivalent results. The most important features are similar for all models, meaning tuning does not influence the feature importance. Again, the most important variable according to the mean decrease in the Gini index, for all models, is INTEREST_EARNING_LENDING_AMT. Hence, this is the most influential variable. The next features are revUtil and revUtilL12, which were also important for XGBoost. However, the next important features are different compared to XGBoost, that is DISTRIBUTOR_NAME and AvgRevBalL3onL12. This indicates that Random Forests and XGBoost build models differently. One probable reason may be that Random Forests accepts categorical variables, while XGBoost only accepts numerical variables.



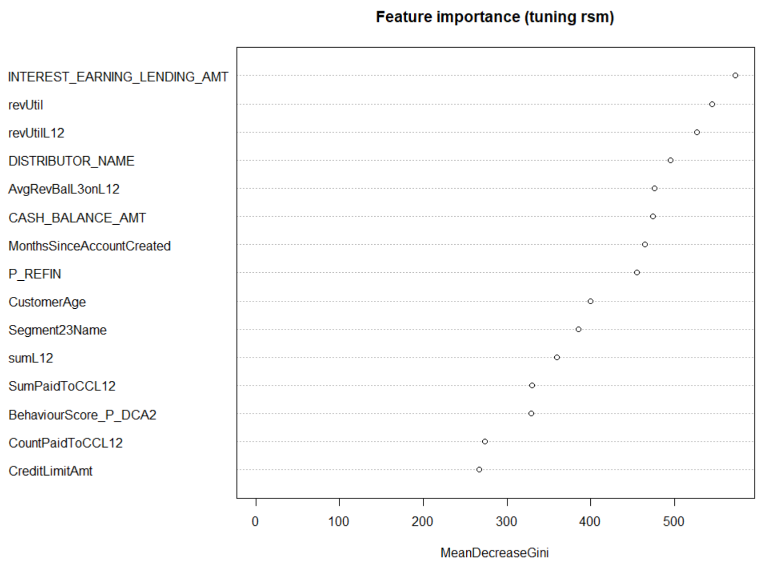


Figure 5.17: Feature importance calculated for Random Forests trained on the training set with **Top:** default hyperparameters, Table 5.33. **Bottom:** tuned hyperparameters from RSM, Table 5.41.

Chapter 6

Discussion

SpareBank1 conducts a call campaign to customers eligible for refinancing of credit cards and consumer loans. The objective of this thesis has been to build and optimize models that are capable of classifying which customers accept such an offer, based on historical data. A binary classification study has been performed on an imbalanced dataset, and two methods, XGBoost and Random Forests, have been investigated. The hyperparameters of the classification algorithms have been optimized by maximizing the balanced accuracy of the classification. This has successfully resulted in improved classification performance for both methods, with emphasis on the important positive class. This is the class of interest in response modelling. It is more important for SpareBank1 to reach the customers who would have accepted the offer of to refinance, than those who would not.

First, XGBoost was trained with default values for the hyperparameters, and evaluated both with default cut-off, 0.5, and optimal cut-off 0.26, Table 5.4. The benchmark results with default cut-off were poor, showing the model's inability to classify the important positive class. When the optimal cut-off was applied, the classification improved, resulting in an improvement in the sensitivity and BACC. This is desirable considering the positive class is of interest. Applying the optimal cut-off actually led to very well benchmark results for the default values of the hyperparameters, with a sensitivity equal to 0.6442 and a BACC equal to and 0.6528.

The initial screening experiment, the $2^{(5-1)}$ fractional factorial design, identified the most significant hyperparameters. In particular, the hyperparameters scale_pos_weight and max_depth stood out as important. The parameter scale_pos_weight adjusts the weights of the positive and negative instances, and thus accommodates the class imbalance found in the data. Therefore, it is not a surprise that this parameter was important. These hyperparameters, in addition to eta, were further optimized using Response Surface Methodology, where a maximum was identified. Again, applying the optimal cut-off in the evaluation led to an improvement in the sensitivity, Table 5.13. The BACC decreased slightly as a result of the adjustment of the cost of false positives and negatives. However, this was not considered substantial. Compared to the benchmark model with optimal cut-off, Table 5.4, the sensitivity increased to 0.6915 and the BACC increased to 0.6551. Thus, tuning the hyperparameters using RSM successfully increased the classification performance of the model, especially the classification of the positive class.

Direct variance modelling was investigated for XGBoost to see if more stable models could be obtained. Lower in-run variance was obtained for higher values of the hyperparameter eta and gamma and lower values for max_depth ans scale_pos_weight, compared to the optimal values from RSM, Table 5.12. Increasing eta makes the model less conservative and decreasing max_depth reduces the model complexity, possibly resulting in a more stable model. Not surprisingly, subsample was set to 1, meaning the whole training set was applied. This reduces variance. Looking at the resulting BACC scores on the training set, there were quite similar, indicating variance was reduced. Minimizing the in-run variance led to a model which was not improved when applying the optimal cut-off value, Table 5.15. The sensitivity actually decreased which was surprising, meaning the implemented optimal cut-off algorithm failed. One potential reason for this could be that with the given hyperparameter values, it was not possible to achieve better sensitivity, not even when the cost of false negatives was adjusted. It is therefore reasonable to assume that a lot of the instability found in the models was due to the models struggling to correctly classify the positive class.

Bayesian optimization was then applied to tune the hyperparameters of XGBoost simultaneously. The parameters κ and ϵ were tuned but looking at the differences in the obtained BACC scores on the training set, the differences were not large, Table 5.20, Table 5.21, and Table 5.22. Tuning of these parameters could affect the results but not greatly. The differences between the three acquisition functions were not large when looking at the evaluation metrics, Table 5.23, suggesting that the choice of acquisition function is only of little importance. This can also be seen in the confidence bounds, Table 5.25, which were quite similar for all acquisition functions. It should be noted that acquisition function EI sometimes ran into convergence issues. This happened when the obtained values were too similar and the covariance matrix turned singular. This was not an issue for UCB or PI. Applying the optimal cut-off value to the models again improved the sensitivity. The highest sensitivity score for all XGBoost models was obtained for the UCB model, equal to 0.7018. The BACC scores were all around 0.65, with the best score obtained for the PI model, equal to 0.6555. Bayesian optimization was combined with DoE, showing slight improvements in the BACC, Table 5.28, compared to the results of the full optimization. It was seen that applying the CCD as an initial grid resulted in similar optimal hyperparameter values for the three acquisition functions, Table 5.30, suggesting this provided a more stable optimization. However, the results were not a substantial improvement compared to previous Bayesian models, Table 5.31. The best XGBoost model, according to a BACC equal to 0.6576, was the EI model from Bayesian optimization in combination with DoE. This was an improvement of almost 19% compared to the benchmark model (with default cut-off value).

Random Forests was the second method applied in this thesis. With default hyperparameters, the model struggled to classify the positive instances. This was seen in the low sensitivity score, equal to 0.1684, Table 5.34. The benchmark model classified almost all instances as the negative class, resulting in a specificity score close to 1 and a BACC equal to 0.5737. Again, the initial screening experiment, the $2^{(5-1)}$ fractional factorial design, successfully identified the most significant hyperparameter. This was the hyperparameter cutoff, which decides the predicted class of an observation. Due to the class imbalance, it was again not surprising that this hyperparameter was important. Cutoff and mtry were chosen for further optimization using Response Surface Methodology. Cutoff ended up with a value that reflected the class imbalance in the dataset. Tuning using RSM successfully improved the classification performance of the model, in particular the classification of the positive class. The sensitivity ended up at 0.6133 and the BACC improved from 0.5737 to 0.6661, Table 5.42.

Bayesian optimization successfully tuned the hyperparameters of Random Forests resulting in better classification performance than the default model. The full optimization resulted in models that performed approximately equal, Table 5.48. The models all obtained BACC scores around 0.663, which was lower than for the tuned model through RSM, Table 5.42. However, the sensitivity scores were around 0.63, which was an improvement. Combining Bayesian optimization with DoE and RSM, also improved the classification performance, Table 5.53 and Table 5.56, respectively. In combination with DoE, the scores were not a substantial improvement compared to the previous Bayesian models. Again, it was seen that using the CCD as an initial grid led to similar optimized hyperparameter values for the three acquisition functions, Table 5.55. This strengthens the conclusion that this led to a more stable optimization. The best scores for BACC and sensitivity for Random Forests were obtained for the UCB model in combination with RSM, Table 5.56, equal to 0.6840 and 0.6671, respectively. Compared to the benchmark result, the BACC improved with 16%. Interestingly, the hyperparameter values of this model, Table 5.55, were the center of the CCD applied, but were not found to be the optimal values in the RSM procedure.

An important aspect to discuss is whether tuning improves the classification performance of the algorithms enough. Is the effort of hyperparameter tuning worth it? When looking at the default performance of the two algorithms, Table 5.2 (XGBoost) and Table 5.34 (Random Forests), they were poor. Due to the class imbalance, none of the algorithms managed to identify the positive class. The short answer would be therefore be yes; tuning is critical to obtain decent classification results. However, applying the optimal cut-off for XGBoost with default values improved the results and the BACC score was satisfactory, Table 5.4. One could argue that finding the optimal cut-off is one way of tuning the algorithm and corrects the class imbalance directly. Further optimization for XGBoost improved the classification; especially the sensitivity increased, but the BACC score only improved with some decimals. It can be discussed whether further tuning is worth it, since the BACC scores were quite similar. However, considering the positive class is of interest, the increase in sensitivity is important, making the tuning worth it. In the tuning process of both algorithms, the hyperparameters that corrected the class imbalance were most important. Correct tuning of these is therefore crucial, but the values of the other hyperparameters seemed to vary more and did not influence the classification performance greatly. Once the class imbalance was corrected, the models performed approximately equal.

In general, Random Forests achieved higher scores for BACC than XGBoost, and could therefore be the preferred method. On the other hand, XGBoost achieved the highest score for sensitivity, which is also desirable. The best XGBoost model achieved a BACC of 0.6576 and the best Random Forests model achieved a BACC of 0.6671. These scores are better than random but are still not particularly good. Due to the class imbalance in the data, the objective of classifying most customers correctly might be hard to achieve. One of the biggest drawbacks of Random Forests is its long computational time. Compared to XGBoost, Random Forests is slow, due to the algorithm averaging many trees. However, this results in more stable models compared to XGBoost. Looking at the bootstrap confidence intervals for Random Forests and XGBoost, Table 5.50 and Table 5.25, the intervals for Random Forests were shorter than XGBoost. This applied for all metrics, but especially for sensitivity. The largest interval for sensitivity for XGBoost was equal to 0.0118, in contrast to 0.0033 for Random Forests. In this way, Random Forests is more reliable than XGBoost. Both algorithms are stochastic, resulting in different cross-validation estimates each time they are run. This can make the results differ, not a lot, but often with a few hundreds. Moreover, the response surfaces constructed varied if they were run repeatedly, leading to different results. This may not seem like a big problem, but when fighting for a few decimals these differences can be substantial.

When evaluating the models, there are several metrics to consider, and deciding which metrics are most important is not straightforward. There is no clear consensus as to which metric is the most suitable. It is desirable with a metric that takes both classes into account, but considering the positive class is more important, the emphasis should be on this class. In particular, it has been discussed that AUC can be a misleading measure, [36]. The AUC often summarizes the test performance over the whole region of the ROC space, and not only in the regions of interest. Additionally, as pointed out by [49], different ROC curves can result in the same AUC score. In [14], MCC is favoured over the F1 score and accuracy for binary classification evaluation. This is because a good score for MCC reflects satisfactory results in all four categories of the confusion matrix. However, given that the sensitivity is more important than specificity in this thesis, a high MCC score does not necessarily reflect a good classifier. The emphasis of this thesis has been to achieve a high score for BACC while maintaining a good score for sensitivity. It can be discussed whether other metrics could have been more suitable. Additionally, the constructed bootstrap confidence intervals, Table 5.25 and Table 5.50, showed that the evaluation metrics should not be trusted blindly. One model can be luckier than another, making it hard to conclude which hyperparameter values actually yield the best model. Ideally, confidence intervals of the evaluation metrics should be made for all models, making the evaluation more reliable and robust.

Both RSM and Bayesian optimization successfully improved the classification performances of the methods. However, the two methods are very different. Optimization through DoE and RSM requires some prior knowledge or assumptions about the hyperparameter values since the levels of the hyperparameters need to be set. However, the path of steepest ascent/descent can be applied to move the experimental region closer to an optimum and adjust the values. Nonetheless, there is a risk that RSM can get stuck in a local optimum. Bayesian optimization has more freedom to explore the hyperparameter space and can in this way avoid getting stuck in a local optimum. Adjusting ϵ and κ can help balance exploitation and exploration. Bayesian optimization is automatic and therefore lacks interpretability. It can be difficult to understand why the algorithm chooses the next values to consider when looking at the output (Appendix C.4). This is not a problem for DoE and RSM, where the tuning process is transparent and controlled by the analyst. More importantly, DoE has the ability to identify the most important factors and in what configuration. This is valuable knowledge in the context of tuning. This is a property Bayesian optimization does not possess. Therefore, combining DoE and RSM with Bayesian optimization is an interesting thought. DoE and RSM require less computational time, and satisfactory results can be achieved with a few iterations. Although Bayesian optimization was performed with many iterations, fewer iterations can be applied while stile achieving reasonable values. This was seen in two occasions with the acquisition function EI (due to convergence issues).

The trade-off between interpretability and model performance is important in response modelling. It is important to gain knowledge about what makes a customer more likely to accept the offer to refinance. Then SpareBank1 can direct the call campaign, and possibly other marketing campaigns, towards them. When looking at interpretability and simplicity of the two models, they are quite similar. Both XGBoost and Random Forests are machine learning algorithms that lack model interpretability, since they are based on boosting and bagging, respectively. Looking at the simplicity, Random Forests follows a simpler procedure than XGBoost. Since XGBoost utilizes gradient boosting, it is more complex. Additionally, there are more tuneable hyperparameters for XGBoost than for Random Forests, making it harder to optimize the algorithm. Applying more traditional statistical methods, such as logistic regression, can help improve model interpretability. However, the calculated feature importances contribute to understanding the models. It is clear from the calculations, Figure 5.16 and Figure 5.17, that the variable INTEREST_EARNING_LENDING_AMT was the most important. This variable describes the interest earning balance or the amount not paid in full last statement. The same can be said for the variables revUtil, revUtilL12, and MonthsAgo. In this way, SpareBank1 can gain valuable knowledge about which customers to target in the future. Interestingly, tuning the hyperparameters did not change the calculated feature importances substantially. The greatest differences were seen for XGBoost, but for Random Forests, the tuning did not affect the results at all.

Chapter 7

Conclusion

This thesis has shown that optimizing the hyperparameters of XGBoost and Random Forests have greatly improved the classification performance of the models, compared to the benchmark results. Especially the classification of the important positive class improved. More importantly, Design of Experiments (DoE) has the ability to identify the most significant hyperparameters, which for both algorithms were the parameters that affect the class weights. It was seen that accommodating the class imbalance was the biggest challenge in this thesis. Once the class imbalance was corrected, the models performed approximately equal. The best XGBoost model according to a balanced accuracy of 0.6576 was the model tuned with Bayesian optimization combined with DoE with the acquisition function Expected Improvement. The best Random Forests model according to a balanced accuracy of 0.6671 was the model tuned with Bayesian optimization combined with RSM with the acquisition function GP Upper Confidence Bound. Compared to benchmark results, the balanced accuracy improved with 19% and 16% for XGBoost and Random Forests, respectively. The calculated feature importances can increase the model interpretability and help understanding what type of customer accepts the offer. The tuning did not significantly affect the feature importances, and the most significant feature was INTEREST_EARNING_LENDING_AMT for both algorithms. These findings can provide valuable insights for SpareBank1.

7.1 Recommendations for Further Work

There are several recommendations for further work. First of all, it is possible that the dataset does not contain enough information about the customers to make correct predictions. Adding new personal variables or macroeconomic variables could possibly improve the classification. Suggested macroeconomic variables could be unemployment rates, gross domestic product (GDP), or inflation. Due to GDPR, personal variables can be difficult to use, but some examples could be marital status, whether the customer has children, or different health issues.

Neither XGBoost nor Random Forests worked optimally on this binary classification task. Tuning improved the classification, but it is possible that other methods could work just as well. For example, accommodating the class imbalance through synthetic generation of data using random undersampling or oversampling can improve the classification, [29]. Moreover, RandomlyOverSamplingExamles (ROSE), [41], or Synthetic Minority Over-sampling TEchnique (SMOTE), [11], are methods that have shown potential when dealing with imbalanced data. Different methods for optimizing the hyperparameters could also be tested, one example is Sequential Model-Based Ensemble optimization (SMBO), [33]. SMBO works in the same manner as Bayesian optimization and involves running several trials with different hyperparameter values and updating a probability model. Another way to optimize the hyperparameter values is using Gradient-Based optimization, for instance based on the gradient of a model selection criterion with respect to the hyperparameters, [3]. Additionally, only 5 hyperparameters for XGBoost and Random Forests were chosen for tuning. Both algorithms contain several other hyperparameters that control the learning process. It is possible that optimizing more hyperparameters or other hyperparameters could lead to better classification performance.

Different classification methods can also be applied, which possibly fits the data better. A suggestion is support vector machines (SVM), which separates data points into classes using higher dimensional hyperplanes, [15]. Looking at previous research papers, neural networks have shown potential in response modelling, [52]. In particular, a long-short-term memory (LSTM) neural network showed great promise of being used in response modelling. Although they lack interpretability and have many tuneable parameters, they can possibly improve the classification performance.

Bibliography

[1] G. Armstrong. *Marketing: an introduction*. Pearson Education, 2009.

[2] M. S. Bartlett and D. G. Kendall. 'The Statistical Analysis of Variance-Heterogeneity and the Logarithmic Transformation'. In: *Supplement to the Journal of the Royal Statistical Society* 8.1 (1946), pp. 128–138. URL: http://www.jstor.org/stable/2983618.

[3] Y. Bengio. 'Gradient-based optimization of hyperparameters'. In: *Neural computation* 12.8 (2000), pp. 1889–1900.

[4] J. A. Bennett and J. W. Strydom. *Introduction to travel and tourism marketing*. Juta and Company Ltd, 2001.

[5] P. D. Berger, R. E. Maurer and G. B. Celli. 'Two-Level Fractional-Factorial Designs'. In: *Experimental Design: With Application in Management, Engineering, and the Sciences*. Cham: Springer International Publishing, 2018, pp. 371–421. ISBN: 978-3-319-64583-4. DOI: 10.1007/978-3-319-64583-4_11. URL: https://doi.org/10.1007/978-3-319-64583-4_11.

[6] J. Bergstra and Y. Bengio. 'Random Search for Hyper-Parameter Optimization'. In: *Journal of Machine Learning Research* 13 (2012), pp. 281–305.

[7] M. J. Berry and G. S. Linoff. *Data mining techniques: for marketing, sales, and customer relationship management*. John Wiley & Sons, 2004.

[8] G. E. P. Box and K. B. Wilson. 'On the Experimental Attainment of Optimum Conditions'. In: *Journal of the Royal Statistical Society. Series B (Methodological)* 13.1 (1951), pp. 1–45. URL: http://www.jstor.org/stable/2983966.

[9] L. Breiman. 'Random Forests'. In: *Machine Learning* 45 (2001), pp. 5–32. DOI: https://doi.org/10.1023/A:1010933404324.

[10] Y. C. Chang, K. H. Chang and G. J. Wu. 'Application of eXtreme gradient boosting trees in the construction of credit risk assessment models for financial institutions'. In: *Applied Soft Computing* 73 (2018), pp. 914–920.

[11] N. V. Chawla et al. 'SMOTE: synthetic minority over-sampling technique'. In: *Journal of artificial intelligence research* 16 (2002), pp. 321–357.

[12] T. Chen and C. Guestrin. *Hyperparameters of XGBoost.* URL: https://xgboost.readthedocs.io/en/latest/parameter.html.

[13] T. Chen and C. Guestrin. 'Xgboost: A scalable tree boosting system'. In: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining.* 2016, pp. 785–794.

[14] D. Chicco and G. Jurman. 'The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation'. In: *BMC genomics* 21.1 (2020), pp. 1–13.

[15] C. Cortes and V. Vapnik. 'Support-vector networks'. In: *Machine learning* 20.3 (1995), pp. 273–297.

[16] K. Coussement, P. Harrigan and D. F. Benoit. 'Improving direct mail targeting through customer response modeling'. In: *Expert Systems With Applications* 42.22 (2015), pp. 8403–8412.

[17] A. Dean, D. Voss and D. Draguljić. *Design and Analysis of Experiments.* 2nd ed. Springer, 2017. DOI: https://doi.org/10.1007/978-3-319-52250-0.

[18] B. Efron and R. J. Tibshirani. *An introduction to the bootstrap.* CRC press, 1994.

[19] T. Fawcett. 'An introduction to ROC analysis'. In: *Pattern Recognition Letters* 27.8 (2006). ROC Analysis in Pattern Recognition, pp. 861–874. ISSN: 0167-8655. DOI: https://doi.org/10.1016/j.patrec.2005.10.010. URL: https://www.sciencedirect.com/science/article/pii/S016786550500303X.

[20] A. Fernández et al. *Learning from Imbalanced Data Sets.* Springer, 2018. DOI: https://doi.org/10.1007/978-3-319-98074-4.

[21] J. Friedman, T. Hastie and R. Tibshirani. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* 2nd ed. Springer, 2009. DOI: https://doi.org/10.1007/978-0-387-84858-7.

[22] *Gjeldsregisteret.* URL: https://www.gjeldsregisteret.com/pages/nokkeltall.

[23] S. Harikant. *5-Fold Cross-Validation.* URL: https://inblog.in/K-Fold-Cross-Validation-Technique-NCaSQ8Kmfh (visited on 7th Feb. 2022).

[24] I. Haugan. *Norske husholdninger ligger i verdenstoppen i privat gjeld.* URL: https://forskning.no/ntnu-partner-penger/norske-husholdninger-ligger-i-verdenstoppen-i-privat-gjeld/1770716.

[25] F. Held. *MVE441/MSA220 Statistical learning for big data.* Lecture 3, slide 22. Chalmers University of Technology. 2021.

[26] H. Høie. 'Usikret gjeld - omfang og kjennetegn ved lånetakerne'. In: *Statistics Norway* (2021). URL: https://www.ssb.no/inntekt-og-forbruk/artikler-og-publikasjoner/_attachment/449421?_ts=1786315b3f0.

[27] G. G. Hsu, J. H. Tomal and W. J. Welch. 'EPX: An R package for the ensemble of subsets of variables for highly unbalanced binary classification'. In: *Computers in Biology and Medicine* 136 (2021), p. 104760. ISSN: 0010-4825. DOI: https://doi.org/10.1016/j.compbiomed.2021.104760. URL: https://www.sciencedirect.com/science/article/pii/S0010482521005540.

[28] G. James et al. *An Introduction to Statistical Learning with Applications in R.* 2nd ed. Springer, 2021. DOI: https://doi.org/10.1007/978-1-0716-1418-1.

[29] N. Japkowicz and S. Stephen. 'The class imbalance problem: A systematic study'. In: *Intelligent data analysis* 6.5 (2002), pp. 429–449.

[30] D. Jasrasaria and E. O. Pyzer-Knapp. 'Dynamic control of explore/exploit trade-off in bayesian optimization'. In: *Science and Information Conference.* Springer. 2018, pp. 1–15.

[31] G. Kim, B. Kevin Chae and D. L. Olson. 'A support vector machine (SVM) approach to imbalanced datasets of customer responses: comparison with other customer response models'. In: *Service Business* 7.1 (2013), pp. 167–182.

[32] H. J. Kushner. 'A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise'. In: (1964).

[33] H. Lacoste A.and Larochelle, F. Laviolette and M. Marchand. 'Sequential model-based ensemble optimization'. In: *arXiv preprint arXiv:1402.0796* (2014).

[34] R. V. Lenth. 'Quick and easy analysis of unreplicated factorials'. In: *Technometrics* 31.4 (1989), pp. 469–473.

[35] E. Liu. *Function ROCInfo().* URL: https://github.com/ethen8181/machine-learning/blob/master/unbalanced/unbalanced_code/unbalanced_functions.R (visited on 1st Feb. 2022).

[36] J. M. Lobo, A. Jiménez-Valverde and R. Real. 'AUC: a misleading measure of the performance of predictive distribution models'. In: *Global ecology and Biogeography* 17.2 (2008), pp. 145–151.

[37] G. A. Lujan-Moreno et al. 'Design of experiments and response surface methodology to tune machine learning hyperparameters, with a random forest case-study'. In: *Expert Systems with Applications* 109 (2018), pp. 195–205. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2018.05.024. URL: https://www.sciencedirect.com/science/article/pii/S0957417418303178.

[38] A. Luque et al. 'The impact of class imbalance in classification performance metrics based on the binary confusion matrix'. In: *Pattern Recognition* 91 (2019), pp. 216–231. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2019.02.023. URL: https://www.sciencedirect.com/science/article/pii/S0031320319300950.

[39] E. C. Malthouse. 'Ridge regression and direct marketing scoring models'. In: *Journal of Interactive Marketing* 13.4 (1999), pp. 10–23. ISSN: 1094-9968. DOI: https://doi.org/10.1002/(SICI)1520-6653(199923)13:4⟨10::AID-DIR2⟩3.0.CO;2-3. URL: https://www.sciencedirect.com/science/article/pii/S1094996899702446.

[40] A. Matosevic. 'On Bayesian optimization and its application to hyperparameter tuning'. Linnaeus University, 2018.

[41] G. Menardi and N. Torelli. 'Training and assessing classification rules with imbalanced data'. In: *Data mining and knowledge discovery* 28.1 (2014), pp. 92–122.

[42] M. Mittendorf, U. D. Nielsen and H. B. Bingham. 'Data-driven prediction of added-wave resistance on ships in oblique waves—A comparison between tree-based ensemble methods and artificial neural networks'. In: *Applied Ocean Research* 118 (2022), p. 102964.

[43] J. Mockus, V. Tiesis and A. Zilinskas. 'The application of Bayesian methods for seeking the extremum'. In: *Towards global optimization* 2.117-129 (1978), p. 2.

[44] D. Montgomery. *Design and Analysis of Experiments.* Student solutions manual, John Wiley & Sons, 2008.

[45] R. H. Myers, D. C. Montgomery and C. M. Anderson-Cook. *Response surface methodology: process and product optimization using designed experiments.* John Wiley & Sons, 2016.

[46] E. H. Olsen. 'Improving Direct Phone Targeting through Customer Response Modeling'. NTNU, 2021.

[47] D. L. Olson et al. 'Comparison of customer response models'. In: *Service Business* 3.2 (2009), pp. 117–130.

[48] J. Orloff and J. Bloom. *Bootstrap confidence intervals.* Lecture notes. Class 24 18.05 Introduction to Probability and Statistics. MIT. 2014.

[49] S. H. Park, J. M. Goo and C-H. Jo. 'Receiver Operating Characteristic (ROC) Curve: Practical Review for Radiologists]'. In: *Korean J Radiol* 5.1 (2004), pp. 11–18.

[50] J Ross Quinlan. *C4. 5: programs for machine learning.* Elsevier, 2014.

[51] M. Sarkar and A. De Bruyn. 'LSTM Response Models for Direct Marketing Analytics: Replacing Feature Engineering with Deep Learning'. In: *Journal of Interactive Marketing* 53 (2021), pp. 80–95. ISSN: 1094-9968. DOI: https://doi.org/10.1016/j.intmar.2020.07.002. URL: https://www.sciencedirect.com/science/article/pii/S1094996820301080.

[52] M. Sarkar and A. De Bruyn. 'LSTM response models for direct marketing analytics: replacing feature engineering with deep learning'. In: *Journal of Interactive Marketing* 53 (2021), pp. 80–95.

[53] H. Schau−Hansen. 'Using statistical modelling to predict outcome of response campaigns'. NTNU, project thesis. 2021.

[54] R. Shiomi et al. 'A study on operating lifetime estimation for electrical components in power grids on the basis of analysis of maintenance records'. In: *Journal of International Council on Electrical Engineering* 9.1 (2019), pp. 45–52.

[55] J. Snoek, H. Larochelle and R. P. Adams. ‘Practical bayesian optimization of machine learning algorithms’. In: *Advances in neural information processing systems* 25 (2012).

[56] N. Srinivas et al. ‘Gaussian process optimization in the bandit setting: No regret and experimental design’. In: *arXiv preprint arXiv:0912.3995* (2009).

[57] D. Sun et al. ‘Assessment of landslide susceptibility mapping based on Bayesian hyperparameter optimization: A comparison between logistic regression and random forest’. In: *Engineering Geology* 281 (2021), p. 105972.

[58] J. Tyssedal. *Two-level factorial design.* NTNU, Lecture notes. 1996.

[59] F. J. Valverde-Albacete, J. Carrillo-de-Albornoz and C. Peláez-Moreno. ‘A Proposal for New Evaluation Metrics and Result Visualization Technique for Sentiment Analysis Tasks.’ In: *Forner P., Müller H., Paredes R., Rosso P., Stein B. (eds) Information Access Evaluation. Multilinguality, Multimodality, and Visualization. CLEF 2013. Lecture Notes in Computer Science* 8138 (2013). DOI: https://doi.org/10.1007/978-3-642-40802-1_5.

[60] R. P. Vatnedal. ‘Optimizing Predictive Performance of Random Forests by means of Design of Experiments and Resampling, with a Case-study in Credit Scoring’. NTNU, 2020.

[61] J. Wu et al. ‘Hyperparameter Optimization for Machine Learning Models Based on Bayesian Optimization’. In: *Journal of Electronic Science and Technology* 17.1 (2019), pp. 26–40. ISSN: 1674-862X. DOI: https://doi.org/10.11989/JEST.1674-862X.80904120. URL: https://www.sciencedirect.com/science/article/pii/S1674862X19300047.

[62] L. Yang and A. Shami. ‘On hyperparameter optimization of machine learning algorithms: Theory and practice’. In: *Neurocomputing* 415 (2020), pp. 295–316. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2020.07.061. URL: https://www.sciencedirect.com/science/article/pii/S0925231220311693.

[63] J. Zhai, J. Qi and C. Shen. ‘Binary imbalanced data classification based on diversity oversampling by generative models’. In: *Information Sciences* 585 (2022), pp. 313–343. ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2021.11.058. URL: https://www.sciencedirect.com/science/article/pii/S0020025521011804.

Appendix A

Explanation of the Variables in the Dataset

Table A.1: Variables in the dataset with explanation.

Variable name	Explanation
BK _ACCOUNT _ID	Internal account ID
PeriodId	Date of start of call activity
weeknr	Week number for start of call activity
CustomerAge	Customer's age in years
GENDER _NAME	Gender
DISTRIBUTOR _NAME	Bank Name
SumPaidToCCL12	Sum paid from bank account to known external credit card accounts last 12 months
SumPaidToRepaymentLoanL12	Sum paid from bank account to known external repayment loan accounts last 12 months
sumPaidToCollectionL12	Sum paid from bank account to known external collection accounts last 12 months
CountPaidToRepaymentLoanL12	Number of payments from bank account to known external repayment loan accounts last 12 months
CountPaidToCCL12	Number of payments from bank account to known external credit card accounts last 12 months
CountPaidToCollectionL12	Number of payments from bank account to known external collection accounts last 12 months
CountDistinctPaidToRepaymentLoanL12	Number of payments from bank account to known distinct external repayment loan accounts last 12 months
CountDistinctPaidToCCL12	Number of payments from bank account to known distinct external credit card accounts last 12 months
CountDistinctPaidToCollectionL12	Number of payments from bank account to known distinct external collection accounts last 12 months
CountRoundPaidToRepaymentLoanL12	Number of round (whole 100 nok) payments from bank account to known distinct external repayment loan accounts last 12 months
CountRoundPaidToCCL12	Number of round (whole 100 nok) payments from bank account to known external credit card accounts last 12 months

CountRoundPaidToCollectionL12	Number of round (whole 100 nok) payments from bank account to known external collection accounts last 12 months
MonthsSinceAccountCreated	Account (cards) age in months
INTEREST _EARNING _LENDING _AMT	Interest earning balance (amount not payed in full last statement)
CASH _BALANCE _AMT	Balance originating from cash withdrawals and transfers
HAS _ESTATEMENT _AGREEMENT _IND	Indicator, e-statement selected ("e-faktura")
CreditLimitAmt	Credit limit on card
revUtil	Average revolving balance last month divided by average credit limit last 12 months
SumAirlineL12	Sum of transactions in given class last 12 months
SumELECTRIC _APPLIANCEL12	Sum of transactions in given class last 12 months
SumFOOD _STORES _WAREHOUSEL12	Sum of transactions in given class last 12 months
SumHOTEL _MOTELL12	Sum of transactions in given class last 12 months
SumHARDWAREL12	Sum of transactions in given class last 12 months
SumINTERIOR _FURNISHINGSL12	Sum of transactions in given class last 12 months
SumOTHER _RETAILL12	Sum of transactions in given class last 12 months
SumOTHER _SERVICESL12	Sum of transactions in given class last 12 months
SumOTHER _TRANSPORTL12	Sum of transactions in given class last 12 months
SumRECREATIONL12	Sum of transactions in given class last 12 months
SumRESTAURANTS _BARSL12	Sum of transactions in given class last 12 months
SumSPORTING _TOY _STORESL12	Sum of transactions in given class last 12 months
SumTRAVEL _AGENCIESL12	Sum of transactions in given class last 12 months
SumVEHICLESL12	Sum of transactions in given class last 12 months
SumQuasiCashL12	Sum of transactions in given class last 12 months
SumAirlineL3	Sum of transactions in given class last 3 months
SumELECTRIC _APPLIANCEL3	Sum of transactions in given class last 3 months
SumFOOD _STORES _WAREHOUSEL3	Sum of transactions in given class last 3 months
SumHOTEL _MOTELL3	Sum of transactions in given class last 3 months
SumHARDWAREL3	Sum of transactions in given class last 3 months
SumINTERIOR _FURNISHINGSL3	Sum of transactions in given class last 3 months
SumOTHER _RETAILL3	Sum of transactions in given class last 3 months
SumOTHER _SERVICESL3	Sum of transactions in given class last 3 months
SumOTHER _TRANSPORTL3	Sum of transactions in given class last 3 months
SumRECREATIONL3	Sum of transactions in given class last 3 months
SumRESTAURANTS _BARSL3	Sum of transactions in given class last 3 months
SumSPORTING _TOY _STORESL3	Sum of transactions in given class last 3 months
SumTRAVEL _AGENCIESL3	Sum of transactions in given class last 3 months
SumVEHICLESL3	Sum of transactions in given class last 3 months
SumQuasiCashL3	Sum of transactions in given class last 3 months
revUtilL12	Average revolving balance last 12 months divided by average credit limit last 12 months
AvgRevBalL3onL12	Average revolving balance last 3 months divided by average revolving balance last 12 months
Segment23Name	Customer segmentation
BehaviourScore _P _DCA2	Behaviour Score, probability of defaulting next 12 months (0-100, separate model)

P _REFIN	Probability of refinancing on own initiative (0-100, separate model)
AppliedInd	Response: Indicator, applied for refinancing after phone call

Appendix B

Correlations and Implemented Algorithms

B.1 Correlations

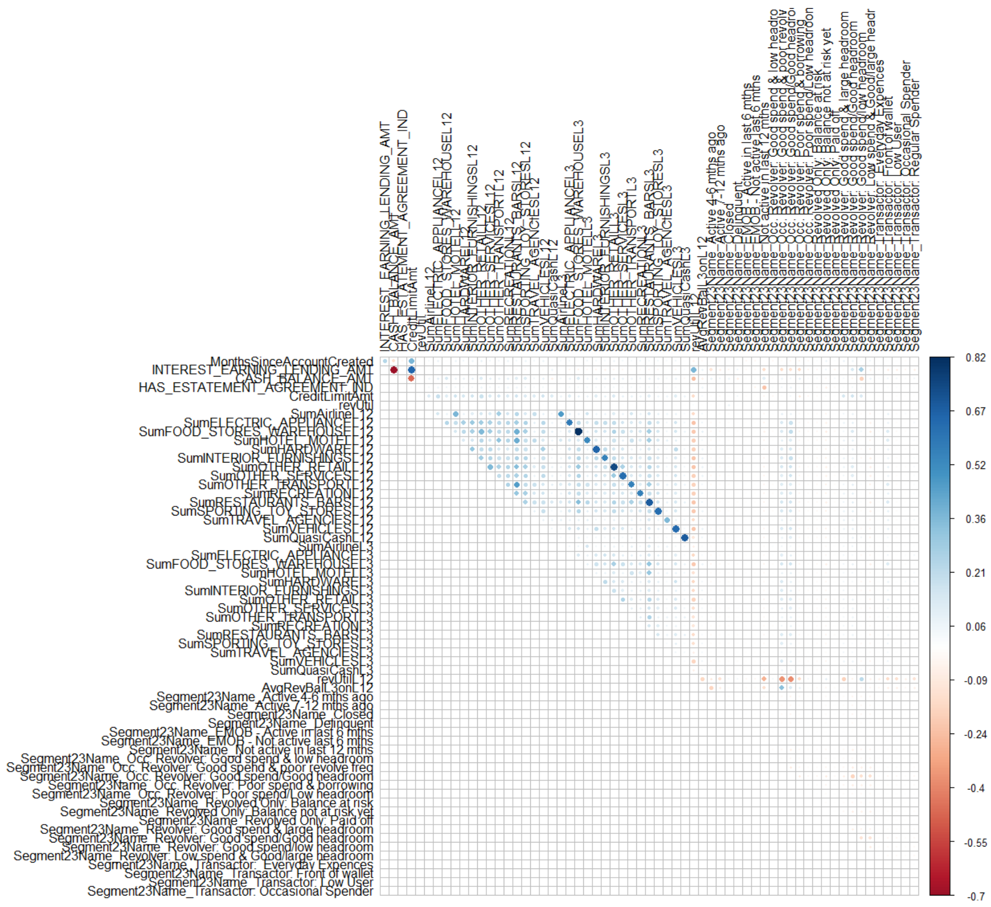


Figure B.1: Correlations in the credit card dataset.

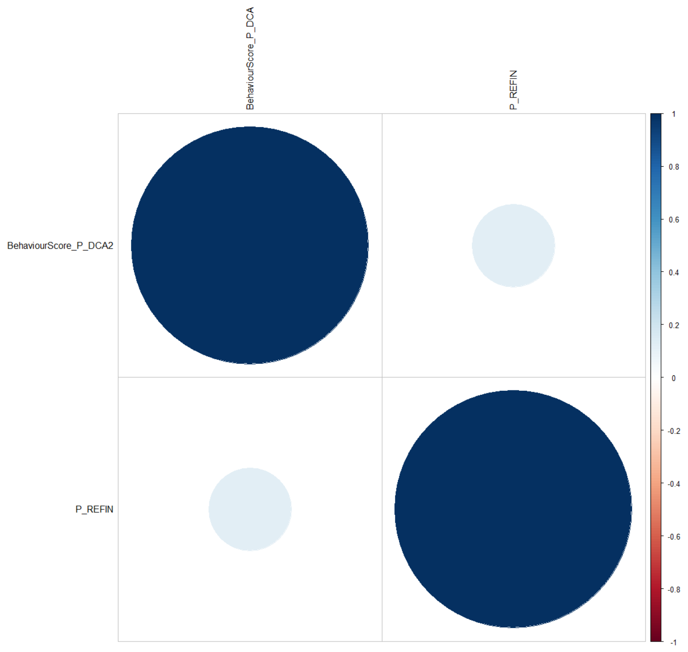


Figure B.2: Correlations in the ML dataset.

B.2 Implemented Algorithms

Optimal cost function:

```
OptFN <- function(cost_fp, test, response, mod){

  bacc_prev <- 0
  cost_best <- 0

  for (i in 2:10){
    pred <- predict(mod, newdata = test, type="response")

    cost_fn <- cost_fp * i
    roc_info <- ROCInfo(data = test, predict = pred,
                        actual = response, cost.fp = cost_fp, cost.fn = cost_fn)

    cut <- roc_info$cutoff
    pred <- ifelse(pred < cut, 0, 1)

    bacc_next <- confusionMatrix(as.factor(pred), as.factor(response), positive = "1")$byClass["Balanced Accuracy"]
    if (bacc_next > bacc_prev){
      cost_best <- cost_fn
      bacc_prev <- bacc_next
    }

  }
  return(cost_best)
}
```

Optimal cut-off function (taken from [35]):

```
ROCInfo <- function(data,predict, actual, cost.fp, cost.fn )
{
  # calculate the values using the ROCR library
  # true positive, false postive
  pred <- prediction(predict, actual )
  perf <- performance( pred, "tpr", "fpr" )
  roc_dt <- data.frame( fpr = perf@x.values[[1]], tpr = perf@y.values[[1]] )

  # cost with the specified false positive and false negative cost
  # false postive rate * number of negative instances * false positive cost +
  # false negative rate * number of positive instances * false negative cost
  cost <- perf@x.values[[1]] * cost.fp * sum( actual== 0 ) +
    ( 1 - perf@y.values[[1]] ) * cost.fn * sum( actual == 1 )
  cost_dt <- data.frame( cutoff = pred@cutoffs[[1]], cost = cost )

  # optimal cutoff value, and the corresponding true positive and false positive rate
  best_index  <- which.min(cost)
  best_cost   <- cost_dt[ best_index, "cost" ]
  best_tpr    <- roc_dt[ best_index, "tpr" ]
  best_fpr    <- roc_dt[ best_index, "fpr" ]
  best_cutoff <- pred@cutoffs[[1]][ best_index ]

  # area under the curve
  auc <- performance( pred, "auc" )@y.values[[1]]

  # normalize the cost to assign colors to 1
  normalize <- function(v) ( v - min(v) ) / diff( range(v) )

  # create color from a palette to assign to the 100 generated threshold between 0 ~ 1
  # then normalize each cost and assign colors to it, the higher the blacker
  # don't times it by 100, there will be 0 in the vector
  col_ramp <- colorRampPalette( c( "green", "orange", "red", "black" ) )(100)
  col_by_cost <- col_ramp[ ceiling( normalize(cost) * 99 ) + 1 ]

  roc_plot <- ggplot( roc_dt, aes( fpr, tpr ) ) +
    geom_line( color = rgb( 0, 0, 1, alpha = 0.3 ) ) +
    geom_point( color = col_by_cost, size = 4, alpha = 0.2 ) +
    geom_segment( aes( x = 0, y = 0, xend = 1, yend = 1 ), alpha = 0.8, color = "royalblue" ) +
    labs( title = "ROC", x = "False Postive Rate", y = "True Positive Rate" ) +
    geom_hline( yintercept = best_tpr, alpha = 0.8, linetype = "dashed", color = "steelblue4" ) +
    geom_vline( xintercept = best_fpr, alpha = 0.8, linetype = "dashed", color = "steelblue4" )

  cost_plot <- ggplot( cost_dt, aes( cutoff, cost ) ) +
    geom_line( color = "blue", alpha = 0.5 ) +
    geom_point( color = col_by_cost, size = 4, alpha = 0.5 ) +
    ggtitle( "Cost" ) +
    scale_y_continuous( labels = comma ) +
    geom_vline( xintercept = best_cutoff, alpha = 0.8, linetype = "dashed", color = "steelblue4" )

  # the main title for the two arranged plot
  sub_title <- sprintf( "Cutoff at %.2f - Total Cost = %.1f, AUC = %.3f",
                        best_cutoff, best_cost, auc )

  # arranged into a side by side plot
  plot <- arrangeGrob( roc_plot, cost_plot, ncol = 2,
                       top = textGrob( sub_title, gp = gpar( fontsize = 16, fontface = "bold" ) ) )

  return( list( plot    = plot,
                cutoff    = best_cutoff,
                totalcost   = best_cost,
                auc         = auc,
                sensitivity = best_tpr,
                specificity = 1 - best_fpr ) )
}
```

Functions to calculate empirical bootstrap confidence intervals, inspired by [48], here implemented for XGBoost:

```
bootstrap<- function(params, cutoff){
  bacc_list <- c()
  sens_list <- c()
  spes_list <- c()

  for (i in 1:30){
    xgb.mod <- xgb.train(params = params, data = dtrain, nrounds = 5, eval_metric = list("rmse","auc"), objective = "binary:logistic")

    resp <- predict(xgb.mod, newdata = dtest, type="response")
    resp <- ifelse(resp < cutoff,0,1)

    bacc <- confusionMatrix(as.factor(resp), as.factor(test$AppliedInd), positive = "1")$byClass["Balanced Accuracy"]
    sens <- confusionMatrix(as.factor(resp), as.factor(test$AppliedInd), positive = "1")$byClass["Sensitivity"]
    spes <- confusionMatrix(as.factor(resp), as.factor(test$AppliedInd), positive = "1")$byClass["Specificity"]

    bacc_list <- cbind(bacc_list, bacc)
    sens_list <- cbind(sens_list, sens)
    spes_list <- cbind(spes_list, spes)
  }
  list(bacc = bacc_list, sens = sens_list, spes = spes_list)
}

conf_int <- function(x, nboot){
  n <- length(x)
  mean <- mean(x)
  tmp <- sample(x,n*nboot, replace=TRUE)
  boot_sample <- matrix(tmp, nrow=n, ncol=nboot)
  bs_means <- colMeans(boot_sample)
  delta <- bs_means - mean
  d <- quantile(delta, c(0.1, 0.9))
  ci <- mean - c(d[2], d[1])
  list(ci)
}
```

Appendix C

Extreme Gradient Boosting Modelling

C.1 Design of Experiments

The output from the results of the two replicates of the $2^{(5-1)}$ fractional factorial design using XGBoost. The calculated responses after both replicates of the design are displayed in Table C.1.

Table C.1: The results of 16 runs of two replicates of XGBoost trained on the training set using the $2^{(5-1)}$ fractional factorial design. The values of the factors A, B, C, D, and E are set to the values in Table 5.6.

Run	A	B	C	D	E = ABCD	Level code	BACC
1	-1	-1	-1	-1	1	1	0.6377
2	1	-1	-1	-1	-1	a	0.5342
3	-1	1	-1	-1	-1	b	0.5112
4	1	1	-1	-1	1	abe	0.6467
5	-1	-1	1	-1	-1	c	0.5631
6	1	-1	1	-1	1	ace	0.6270
7	-1	1	1	-1	1	bce	0.6454
8	1	1	1	-1	-1	abc	0.5713
9	-1	-1	-1	1	-1	d	0.5148
10	1	-1	-1	1	1	ade	0.6472
11	-1	1	-1	1	1	bde	0.6408
12	1	1	-1	1	-1	abd	0.5308
13	-1	-1	1	1	1	cde	0.6401
14	1	-1	1	1	-1	acd	0.5726
15	-1	1	1	1	-1	bcd	0.5608
16	1	1	1	1	1	abcde	0.6386
17	-1	-1	-1	-1	1	1	0.6396
18	1	-1	-1	-1	-1	a	0.5338
19	-1	1	-1	-1	-1	b	0.5105
20	1	1	-1	-1	1	abe	0.6467
21	-1	-1	1	-1	-1	c	0.5617
22	1	-1	1	-1	1	ace	0.6241
23	-1	1	1	-1	1	bce	0.6446
24	1	1	1	-1	-1	abc	0.5726
25	-1	-1	-1	1	-1	d	0.5142
26	1	-1	-1	1	1	ade	0.6469
27	-1	1	-1	1	1	bde	0.6406
28	1	1	-1	1	-1	abd	0.5320
29	-1	-1	1	1	1	cde	0.6400
30	1	-1	1	1	-1	acd	0.5745
31	-1	1	1	1	-1	bcd	0.5620
32	1	1	1	1	1	abcde	0.6364

The model summary of the fitted linear model to the results of the $2^{(5-1)}$ fractional factorial design:

```
Call:
lm.default(formula = y ~ .^2, data = plan)

Residuals:
       Min         1Q     Median         3Q        Max
-0.0014062 -0.0004708  0.0000000  0.0004708  0.0014062

Coefficients:
              Estimate Std. Error  t value Pr(>|t|)
(Intercept)  0.5925823  0.0001671 3547.278  < 2e-16 ***
A            0.0033953  0.0001671   20.325 7.46e-13 ***
B            0.0006130  0.0001671    3.669 0.002072 **
C            0.0095886  0.0001671   57.399  < 2e-16 ***
D            0.0006896  0.0001671    4.128 0.000789 ***
E            0.0475777  0.0001671  284.807  < 2e-16 ***
A:B          0.0003148  0.0001671    1.885 0.077795 .
A:C         -0.0034256  0.0001671  -20.506 6.51e-13 ***
A:D          0.0007241  0.0001671    4.335 0.000512 ***
A:E         -0.0043392  0.0001671  -25.975 1.65e-14 ***
B:C          0.0011835  0.0001671    7.085 2.58e-06 ***
B:D         -0.0011196  0.0001671   -6.702 5.08e-06 ***
B:E          0.0017142  0.0001671   10.261 1.92e-08 ***
```

```
C:D          0.0002615  0.0001671    1.565 0.137036
C:E         -0.0127201  0.0001671  -76.144  < 2e-16 ***
D:E          0.0004880  0.0001671    2.921 0.009986 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.000945 on 16 degrees of freedom
Multiple R-squared:  0.9998,Adjusted R-squared:  0.9997
F-statistic:  6132 on 15 and 16 DF,  p-value: < 2.2e-16
```

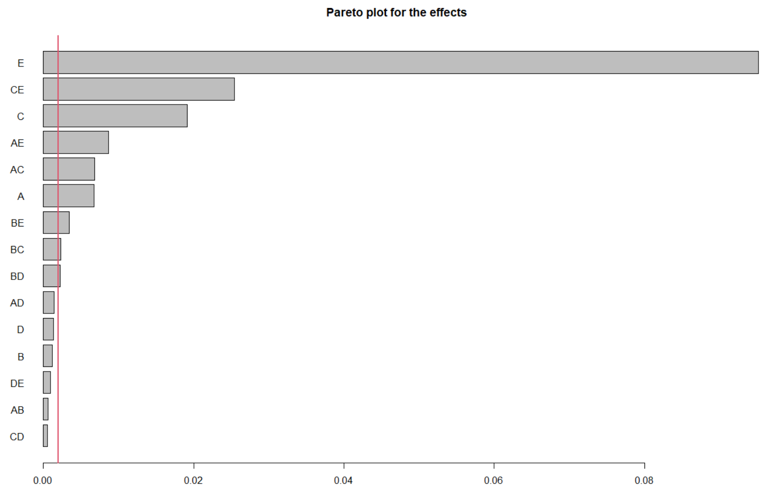


Figure C.1: Pareto plot of the estimated effects of the $2^{(5-1)}$ fractional factorial design with XGBoost. The significance level is calculated using Lenth's method, [34], with significance level $\alpha = 0.05$

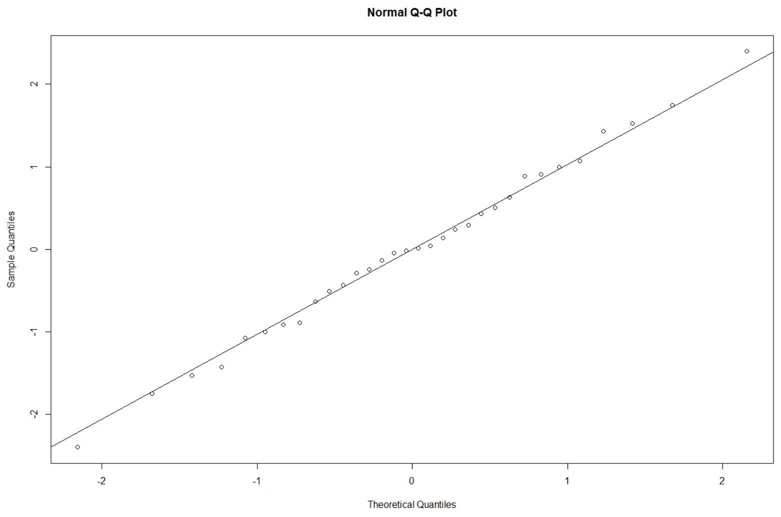


Figure C.2: The Q-Q plot of the residuals from the fitted linear model of the results from the $2^{(5-1)}$ fractional factorial design with XGBoost.

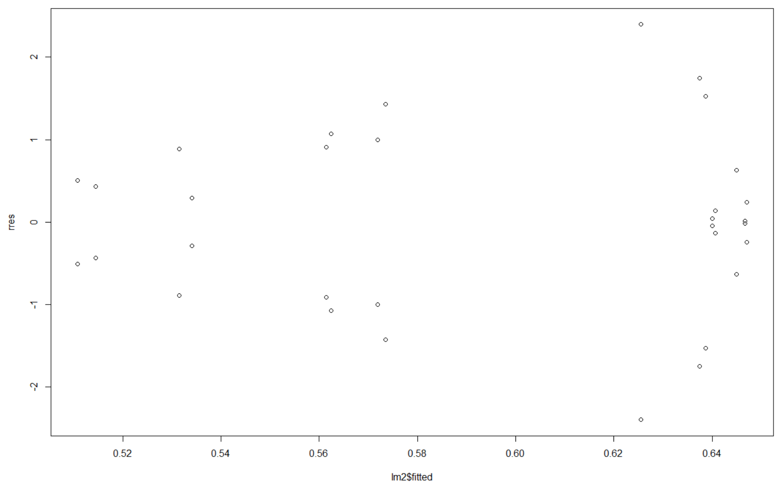


Figure C.3: Plot of the residuals from the fitted linear model of the results from the $2^{(5-1)}$ fractional factorial design with XGBoost.

The path of steepest ascent followed for eta and max_depth= 3, while the other hyperparameters are held at obtained levels:

Table C.2: Path of steepest ascent followed for scale_pos_weight and max_depth= 3 in XGBoost, while the other hyperparameters are at obtained levels. The resulting BACC is the mean of 5 runs of 10-fold cross-validation performed on the training set.

A (eta)	B (subsample)	C (max_depth)	D (gamma)	E (scale_pos_weight)	BACC
0.5	1	3	0	3.571	0.6443
0.5	1	3	0	3.591	0.6445
0.5	1	3	0	3.611	0.6433
0.5	1	3	0	3.631	0.6438
0.5	1	3	0	3.651	0.6437
0.5	1	3	0	3.671	0.6429
0.5	1	3	0	3.691	0.6424
0.5	1	3	0	3.711	0.6432
0.5	1	3	0	3.731	0.6467
0.5	1	3	0	3.751	0.6420
0.5	1	3	0	3.771	0.6431

The path of steepest ascent followed for eta and max_depth= 2, while the other hyperparameters are held at obtained levels:

Table C.3: Path of steepest ascent followed for scale_pos_weight and max_depth= 2 in XGBoost, while the other hyperparameters are at obtained levels. The resulting BACC is the mean of 5 runs of 10-fold cross-validation performed on the training set.

A (eta)	B (subsample)	C (max_depth)	D (gamma)	E (scale_pos_weight)	BACC
0.5	1	2	0	3.571	0.6408
0.5	1	2	0	3.591	0.6409
0.5	1	2	0	3.611	0.6422
0.5	1	2	0	3.631	0.6419
0.5	1	2	0	3.651	0.6408
0.5	1	2	0	3.671	0.6426
0.5	1	2	0	3.691	0.6427
0.5	1	2	0	3.711	0.6411
0.5	1	2	0	3.731	0.6410
0.5	1	2	0	3.751	0.6415
0.5	1	2	0	3.771	0.6420

C.2 Response Surface Methodology

C.2.1 First RSM

The central composite design of the first response surface model fitted to the interactions between eta, max_depth, and scale_pos_weight:

```
   run.order std.order       eta max_depth scale_pos_weight         y
1          1         1 0.4500000  3.000000         3.571000 0.6480842
2          2         2 0.5500000  3.000000         3.571000 0.6458637
3          3         3 0.4500000  5.000000         3.571000 0.6496819
4          4         4 0.5500000  5.000000         3.571000 0.6504052
5          5         5 0.4500000  3.000000         3.771000 0.6493995
6          6         6 0.5500000  3.000000         3.771000 0.6449155
7          7         7 0.4500000  5.000000         3.771000 0.6506113
8          8         8 0.5500000  5.000000         3.771000 0.6518779
9          1         1 0.4133975  4.000000         3.671000 0.6507574
10         2         2 0.5866025  4.000000         3.671000 0.6511733
11         3         3 0.5000000  2.267949         3.671000 0.6489584
12         4         4 0.5000000  5.732051         3.671000 0.6497888
13         5         5 0.5000000  4.000000         3.497795 0.6503241
14         6         6 0.5000000  4.000000         3.844205 0.6504503
15         7         7 0.5000000  4.000000         3.671000 0.6496693
16         8         8 0.5000000  4.000000         3.671000 0.6517895
17         9         9 0.5000000  4.000000         3.671000 0.6507266

Data are stored in coded form using these coding formulas ...
x1 ~ (eta - 0.5)/0.05
x2 ~ (max_depth - 4)/1
x3 ~ (scale_pos_weight - 3.671)/0.1
```

Model summary of the first second-order response surface model fitted to the central composite design:

```
Call:
rsm(formula = y ~ SO(x1, x2, x3), data = ccd)

               Estimate  Std. Error  t value Pr(>|t|)
(Intercept)  0.65072847  0.00097871 664.8871   <2e-16 ***
x1          -0.00028530  0.00045305  -0.6297   0.5489
x2           0.00112512  0.00045305   2.4834   0.0420 *
x3           0.00021343  0.00045305   0.4711   0.6519
x1:x2        0.00108679  0.00059933   1.8133   0.1127
x1:x3       -0.00021501  0.00059933  -0.3588   0.7304
x2:x3        0.00025439  0.00059933   0.4244   0.6840
x1^2        -0.00018525  0.00048545  -0.3816   0.7141
x2^2        -0.00071584  0.00048545  -1.4746   0.1838
x3^2        -0.00037798  0.00048545  -0.7786   0.4617
```

```
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Multiple R-squared:  0.6445,Adjusted R-squared:  0.1875
F-statistic:  1.41 on 9 and 7 DF,  p-value: 0.3325

Analysis of Variance Table

Response: y
                Df     Sum Sq    Mean Sq F value Pr(>F)
FO(x1, x2, x3)   3 1.9500e-05 6.4999e-06  2.2620 0.1685
TWI(x1, x2, x3)  3 1.0337e-05 3.4455e-06  1.1990 0.3777
PQ(x1, x2, x3)   3 6.6388e-06 2.2129e-06  0.7701 0.5463
Residuals        7 2.0115e-05 2.8736e-06
Lack of fit      5 1.7868e-05 3.5735e-06  3.1797 0.2564
Pure error       2 2.2477e-06 1.1238e-06

Stationary point of response surface:
         x1          x2          x3
-1.22162405 -0.03144231  0.61920589

Stationary point in original units:
            eta       max_depth scale_pos_weight
       0.4389188        3.9685577        3.7329206

Eigenanalysis:
eigen() decomposition
$values
[1]  0.0001552633 -0.0003410815 -0.0010932517

$vectors
         [,1]        [,2]       [,3]
x1  0.8505725 -0.07933114  0.5198395
x2  0.5237939  0.21531432 -0.8241843
x3 -0.0465454  0.97331717  0.2246936
```

The corresponding estimated canonical path:

```
  dist     x1     x2    x3 |     eta max_depth scale_pos_weight |  yhat
1   -2 -2.923 -1.079 0.712 | 0.35385     2.921           3.7422 | 0.652
2   -1 -2.072 -0.555 0.666 | 0.39640     3.445           3.7376 | 0.651
3    0 -1.222 -0.031 0.619 | 0.43890     3.969           3.7329 | 0.651
4    1 -0.371  0.492 0.573 | 0.48145     4.492           3.7283 | 0.651
5    2  0.480  1.016 0.526 | 0.52400     5.016           3.7236 | 0.652
6    3  1.330  1.540 0.480 | 0.56650     5.540           3.7190 | 0.652
7    4  2.181  2.064 0.433 | 0.60905     6.064           3.7143 | 0.653
8    5  3.031  2.588 0.386 | 0.65155     6.588           3.7096 | 0.655
```

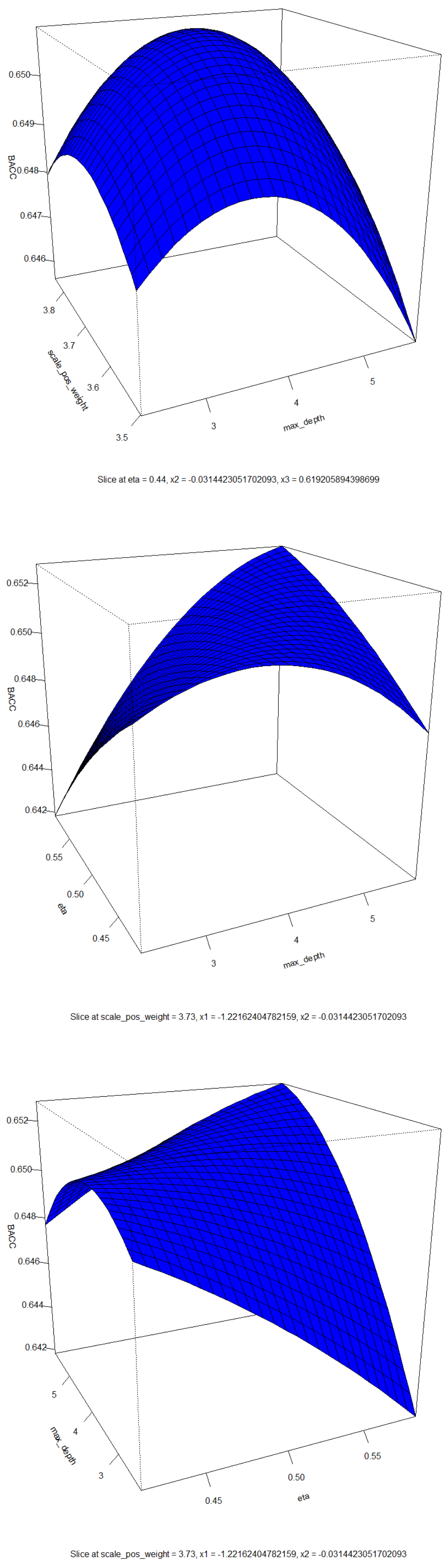


Figure C.4: Perspective plots of the fitted second-order response surface model with XGBoost. The design has center in eta= 0.5, max_depth= 4, and scale_pos_weight= 3.671.

C.2.2 Second RSM

The central composite design of the second response surface model fitted to the interactions between eta, max_depth, and scale_pos_weight:

```
   run.order std.order       eta max_depth scale_pos_weight         y
1          1         1 0.4314500  3.000000         3.678300 0.6482491
2          2         2 0.5314500  3.000000         3.678300 0.6467139
3          3         3 0.4314500  5.000000         3.678300 0.6509119
4          4         4 0.5314500  5.000000         3.678300 0.6512770
5          5         5 0.4314500  3.000000         3.778300 0.6465913
6          6         6 0.5314500  3.000000         3.778300 0.6475216
7          7         7 0.4314500  5.000000         3.778300 0.6507871
8          8         8 0.5314500  5.000000         3.778300 0.6502804
9          1         1 0.3948475  4.000000         3.728300 0.6507111
10         2         2 0.5680525  4.000000         3.728300 0.6507034
11         3         3 0.4814500  2.267949         3.728300 0.6513745
12         4         4 0.4814500  5.732051         3.728300 0.6500494
13         5         5 0.4814500  4.000000         3.641697 0.6517156
14         6         6 0.4814500  4.000000         3.814903 0.6509506
15         7         7 0.4814500  4.000000         3.728300 0.6504285
16         8         8 0.4814500  4.000000         3.728300 0.6501143
17         9         9 0.4814500  4.000000         3.728300 0.6520616

Data are stored in coded form using these coding formulas ...
x1 ~ (eta - 0.48145)/0.05
x2 ~ (max_depth - 4)/1
x3 ~ (scale_pos_weight - 3.7283)/0.05
```

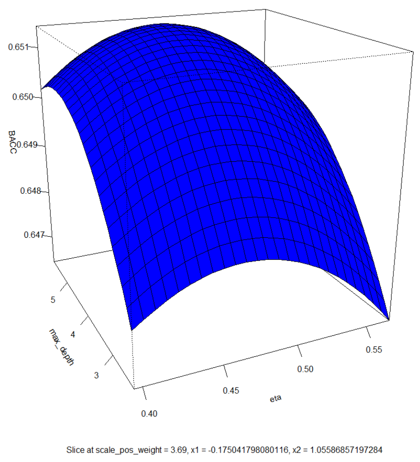
Model summary of the second second-order response surface model fitted to the central composite design:

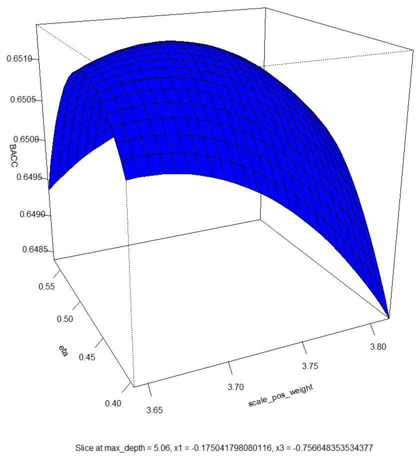
```
Call:
rsm(formula = y ~ SO(x1, x2, x3), data = ccd)

               Estimate  Std. Error  t value Pr(>|t|)
(Intercept)  6.5087e-01  1.2266e-03 530.6085   <2e-16 ***
x1          -5.4267e-05  5.6783e-04  -0.0956   0.9265
x2           8.4894e-04  5.6783e-04   1.4951   0.1785
x3          -2.3547e-04  5.6783e-04  -0.4147   0.6908
x1:x2        5.7898e-05  7.5116e-04   0.0771   0.9407
x1:x3        1.9920e-04  7.5116e-04   0.2652   0.7985
x2:x3       -3.3915e-05  7.5116e-04  -0.0452   0.9652
x1^2        -4.1093e-04  6.0844e-04  -0.6754   0.5211
x2^2        -4.0936e-04  6.0844e-04  -0.6728   0.5226
x3^2        -2.0231e-04  6.0844e-04  -0.3325   0.7492
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Multiple R-squared:  0.3125,Adjusted R-squared:  -0.5714
F-statistic: 0.3535 on 9 and 7 DF,  p-value: 0.9256

Analysis of Variance Table

Response: y
               Df     Sum Sq    Mean Sq F value Pr(>F)
FO(x1, x2, x3)   3 1.0907e-05 3.6358e-06  0.8054 0.5296
TWI(x1, x2, x3)  3 3.5350e-07 1.1780e-07  0.0261 0.9938
PQ(x1, x2, x3)   3 3.1019e-06 1.0340e-06  0.2291 0.8734
Residuals        7 3.1598e-05 4.5140e-06
Lack of fit      5 2.9412e-05 5.8824e-06  5.3822 0.1641
Pure error       2 2.1859e-06 1.0929e-06

Stationary point of response surface:
        x1         x2         x3
-0.1750418  1.0558686 -0.7566484

Stationary point in original units:
             eta        max_depth scale_pos_weight
       0.4726979        5.0558686        3.6904676
```

```
Eigenanalysis:
eigen() decomposition
$values
[1] -0.0001622910 -0.0003910583 -0.0004692523

$vectors
          [,1]       [,2]       [,3]
x1 -0.36972954 -0.4574174  0.8087456
x2  0.02043452 -0.8742170 -0.4851053
x3 -0.92891469  0.1628314 -0.3325709
```

The corresponding estimated canonical path:

```
  dist     x1    x2     x3 |     eta max_depth scale_pos_weight |  yhat
1   -2  0.564 1.015  1.101 | 0.50965     5.015          3.78335 | 0.651
2   -1  0.195 1.035  0.172 | 0.49120     5.035          3.73690 | 0.651
3    0 -0.175 1.056 -0.757 | 0.47270     5.056          3.69045 | 0.651
4    1 -0.545 1.076 -1.686 | 0.45420     5.076          3.64400 | 0.651
5    2 -0.915 1.097 -2.614 | 0.43570     5.097          3.59760 | 0.651
6    3 -1.284 1.117 -3.543 | 0.41725     5.117          3.55115 | 0.650
7    4 -1.654 1.138 -4.472 | 0.39875     5.138          3.50470 | 0.649
8    5 -2.024 1.158 -5.401 | 0.38025     5.158          3.45825 | 0.647
9    6 -2.393 1.178 -6.330 | 0.36180     5.178          3.41180 | 0.646
```





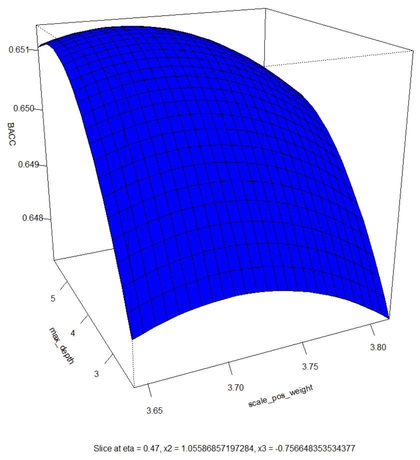


Figure C.5: Perspective plots of the fitted second-order response surface model with XGBoost. The design has center in eta= 0.48145, max_depth= 4, and scale_pos_weight= 3.7283.

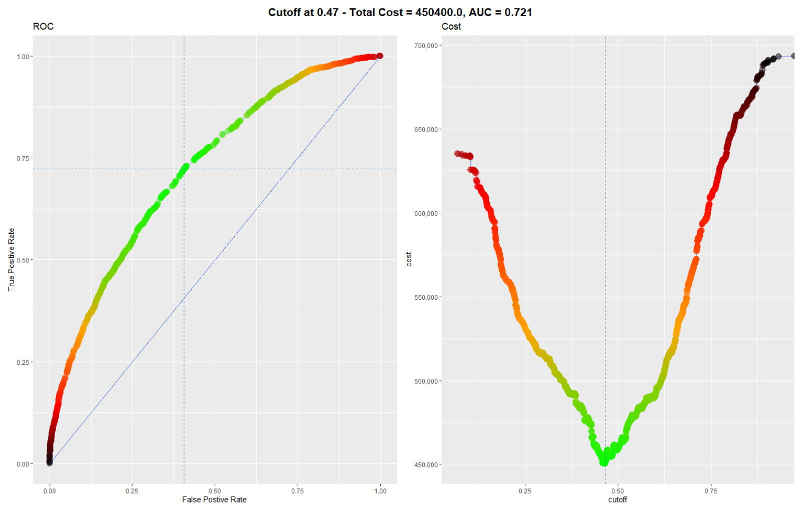


Figure C.6: The ROC curve and cost-plot for different cut-off values for XGBoost trained on the training set with tuned hyperparameters using RSM. The cost of FP is set to 100 and the cost of FN is found to be 400. The optimal cut-off value is displayed in the title, together with the AUC.

C.3 Direct Variance Modelling

Summary of the fitted linear model to the logarithm of the estimated variance. Since no replicates of the experiment is performed it is not possible to estimate the p-values nor the standard errors.

```
Call:
lm.default(formula = y ~ .^2, data = plan)

Residuals:
ALL 16 residuals are 0: no residual degrees of freedom!

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -15.435056         NA      NA       NA
A             0.133199         NA      NA       NA
B            -0.275535         NA      NA       NA
C             1.021216         NA      NA       NA
D            -0.138025         NA      NA       NA
E            -0.880986         NA      NA       NA
A:B          -0.338053         NA      NA       NA
A:C           1.133929         NA      NA       NA
A:D           1.174548         NA      NA       NA
A:E           0.007803         NA      NA       NA
B:C           0.741348         NA      NA       NA
B:D           1.083141         NA      NA       NA
B:E          -0.469036         NA      NA       NA
C:D          -0.528951         NA      NA       NA
C:E           0.248057         NA      NA       NA
D:E          -0.414349         NA      NA       NA

Residual standard error: NaN on 0 degrees of freedom
Multiple R-squared:      1,Adjusted R-squared:    NaN
F-statistic:   NaN on 15 and 0 DF,  p-value: NA
```

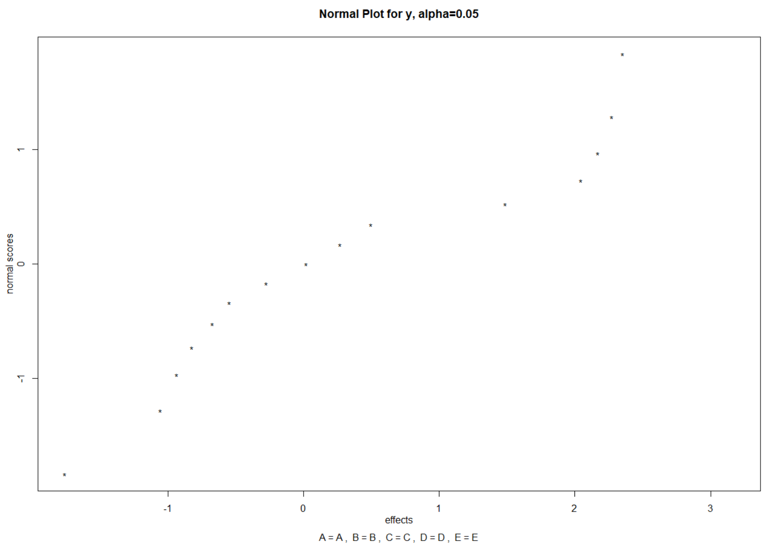


Figure C.7: Normal plot of the estimates of the effects with significance level $\alpha = 0.05$, after conducting direct variance modelling of the results of the $2^{(5-1)}$ fractional factorial design on the training set with XGBoost. None of the estimates of the coefficients are significant at level $\alpha = 0.05$.

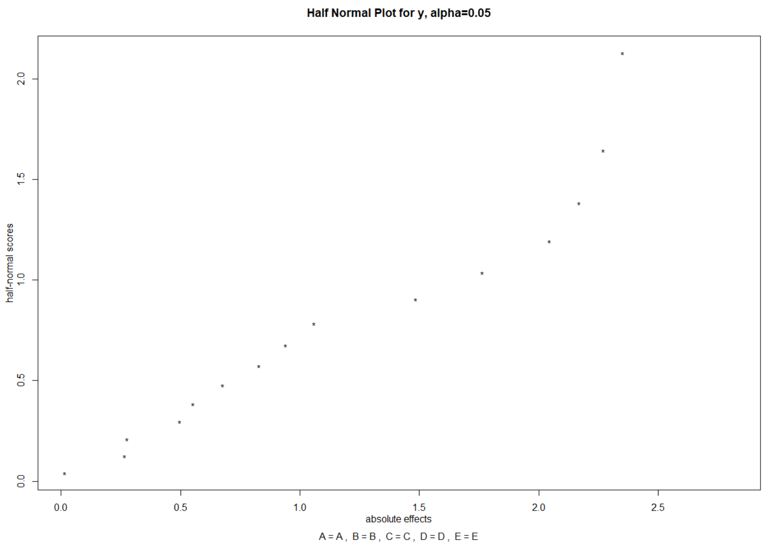


Figure C.8: Half-normal plot of the estimates of the effects after conducting direct variance modelling of the results of the $2^{(5-1)}$ fractional factorial design on the training set with XGBoost. None of the estimates of the coefficients are significant at level $\alpha = 0.05$

The central composite design of the second-order response surface model fitted to interaction between eta, max_depth, and gamma:

```
   run.order std.order       eta max_depth       gamma         y
1          1         1 0.4250000  2.000000  0.05000000 -13.80189
2          2         2 0.6250000  2.000000  0.05000000 -14.71836
3          3         3 0.4250000  4.000000  0.05000000 -13.81937
4          4         4 0.6250000  4.000000  0.05000000 -18.00417
5          5         5 0.4250000  2.000000  0.25000000 -16.06083
6          6         6 0.6250000  2.000000  0.25000000 -13.78424
7          7         7 0.4250000  4.000000  0.25000000 -15.79915
8          8         8 0.6250000  4.000000  0.25000000 -15.82076
9          1         1 0.3517949  3.000000  0.15000000 -13.94753
10         2         2 0.6982051  3.000000  0.15000000 -17.63304
11         3         3 0.5250000  1.267949  0.15000000 -17.97122
```

```
12         4         4 0.5250000  4.732051  0.15000000 -13.36788
13         5         5 0.5250000  3.000000  0.00000000 -13.33321
14         6         6 0.5250000  3.000000  0.32320508 -17.93471
15         7         7 0.5250000  3.000000  0.15000000 -17.12485
16         8         8 0.5250000  3.000000  0.15000000 -14.61414
17         9         9 0.5250000  3.000000  0.15000000 -23.17467

Data are stored in coded form using these coding formulas ...
x1 ~ (eta - 0.525)/0.1
x2 ~ (max_depth - 3)/1
x3 ~ (gamma - 0.15)/0.1
```

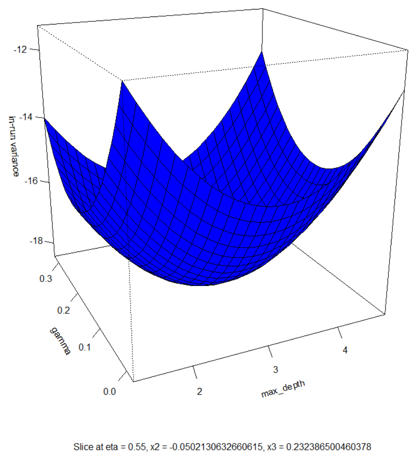
The model summary of the second-order response surface model fitted to the central composite design:

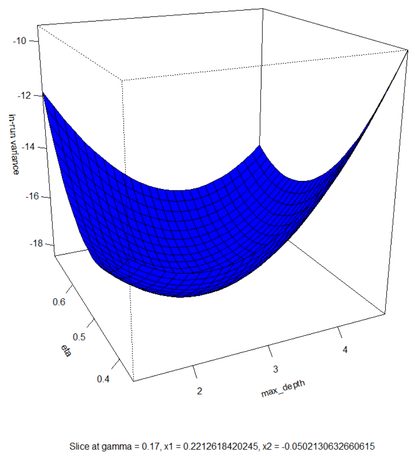
```
Call:
rsm(formula = y ~ SO(x1, x2, x3), data = ccd)

             Estimate Std. Error  t value Pr(>|t|)
(Intercept) -18.30455    1.68169 -10.8846 1.22e-05 ***
x1           -0.65927    0.77847  -0.8469   0.4251
x2            0.20679    0.77847   0.2656   0.7982
x3           -0.64937    0.77847  -0.8342   0.4317
x1:x2        -0.69582    1.02982  -0.6757   0.5209
x1:x3         0.91953    1.02982   0.8929   0.4016
x2:x3         0.19106    1.02982   0.1855   0.8581
x1^2          0.92796    0.83414   1.1125   0.3027
x2^2          0.96821    0.83414   1.1607   0.2838
x3^2          0.98007    0.83414   1.1749   0.2784
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Multiple R-squared:  0.4255,Adjusted R-squared:  -0.3132
F-statistic: 0.576 on 9 and 7 DF,  p-value: 0.7835

Analysis of Variance Table

Response: y
                Df Sum Sq Mean Sq F value Pr(>F)
FO(x1, x2, x3)   3 12.587  4.1957  0.4945 0.6974
TWI(x1, x2, x3)  3 10.930  3.6432  0.4294 0.7384
PQ(x1, x2, x3)   3 20.465  6.8218  0.8041 0.5302
Residuals        7 59.390  8.4842
Lack of fit      5 20.661  4.1321  0.2134 0.9286
Pure error       2 38.729 19.3644

Stationary point of response surface:
         x1          x2          x3
 0.22126184 -0.05021306  0.23238650

Stationary point in original units:
      eta max_depth     gamma
0.5471262 2.9497869 0.1732387

Eigenanalysis:
eigen() decomposition
$values
[1] 1.4840263 1.0640162 0.3281965

$vectors
         [,1]        [,2]       [,3]
x1  0.7187999 -0.05919371  0.6926924
x2 -0.3765878  0.80437787  0.4595191
x3  0.5843871  0.59116186 -0.5558951
```

The corresponding estimated canonical path:

```
  dist     x1     x2     x3 |    eta max_depth  gamma |    yhat
1   -2 -1.216  0.703 -0.936 | 0.4034     3.703 0.0564 | -12.525
2   -1 -0.498  0.326 -0.352 | 0.4752     3.326 0.1148 | -16.974
3    0  0.221 -0.050  0.232 | 0.5471     2.950 0.1732 | -18.458
```

```
4     1  0.940 -0.427  0.817 | 0.6190      2.573 0.2317 | -16.974
5     2  1.659 -0.803  1.401 | 0.6909      2.197 0.2901 | -12.523
6     3  2.378 -1.180  1.986 | 0.7628      1.820 0.3486 |  -5.097
7     4  3.096 -1.557  2.570 | 0.8346      1.443 0.4070 |   5.285
8     5  3.815 -1.933  3.154 | 0.9065      1.067 0.4654 |  18.636
```





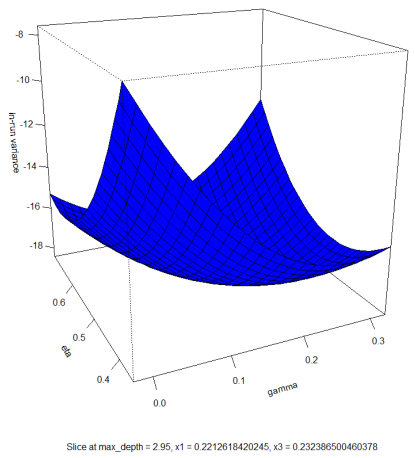


Figure C.9: Perspective plots of the fitted second-order response surface model with XGBoost minimizing the logarithm of the estimated in-run variance. The design has center in eta= 0.525, max_depth= 3, and gamma= 0.15.

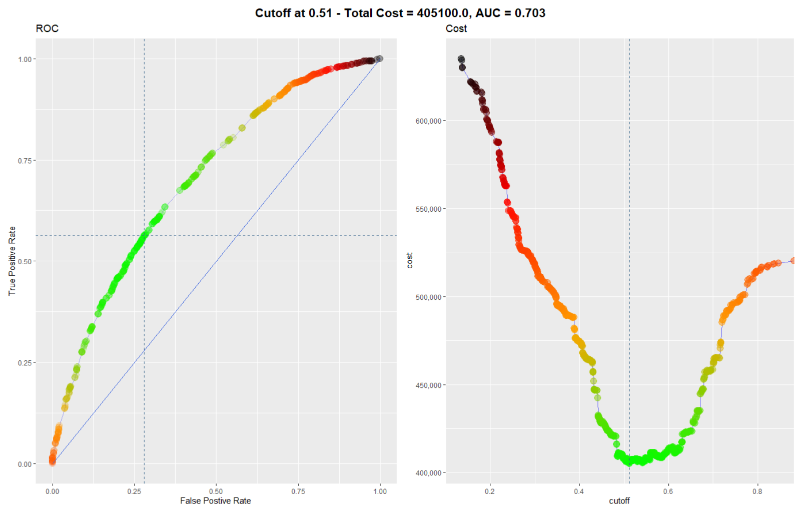


Figure C.10: The ROC curve and cost-plot for different cut-off values for XGBoost trained on the training set with tuned hyperparameters minimizing the logarithm of the estimated in-run variance. The cost of FP is set to 100 and the cost of FN is found to be 300. The optimal cut-off value is displayed in the title, together with the AUC.

C.4 Bayesian optimization

C.4.1 Bayesian optimization of all Hyperparameters

All iterations of Bayesian optimization on all hyperparameters with acquisition function UCB and $\kappa = 1.96$.

```
    Round       eta subsample max_depth        gamma scale_pos_weight     Value
 1:     1 0.5351844 0.9261821         8 1.091440e+00         1.631864 0.5997814
 2:     2 0.2067490 0.3949320         5 9.775905e-01         1.045800 0.5501347
 3:     3 0.6306343 0.9824657         5 2.319076e-01         3.769762 0.6470446
 4:     4 0.6741298 0.9235555         4 9.816296e-01         1.773249 0.6091220
 5:     5 0.3138231 0.6871425         9 1.301568e+00         1.415695 0.5925992
 6:     6 0.6167390 0.9125153         7 1.184564e+00         1.048791 0.5697728
 7:     7 0.5552561 0.4962021         5 5.960531e-01         3.149140 0.6424869
 8:     8 0.1987495 0.9074709         6 1.224354e-01         2.977914 0.6477091
 9:     9 0.4180184 0.2864579         6 1.275015e+00         3.768675 0.6362601
10:    10 0.4714295 0.3174471         6 1.460577e+00         2.989181 0.6314162
11:    11 0.1070972 0.7898709         4 1.032472e+00         2.027035 0.6111652
12:    12 0.2965547 0.4735345         8 6.519836e-02         3.469325 0.6366161
13:    13 0.6125320 0.4288584         7 8.702343e-01         2.558797 0.6256539
14:    14 0.2334372 0.6603097         6 6.862994e-01         2.034532 0.6154563
15:    15 0.2108636 0.9358751         8 1.299608e+00         1.453625 0.5861002
16:    16 0.6370487 0.3847805         6 3.105685e-01         3.901030 0.6307593
17:    17 0.6950661 0.6433003         3 5.396351e-01         3.545893 0.6464659
18:    18 0.4944986 0.6101284         7 9.383856e-01         3.672433 0.6404884
19:    19 0.6380448 0.5501484         6 1.436387e+00         3.361433 0.6397192
20:    20 0.5615533 0.9401621         4 1.265882e+00         2.921324 0.6466581
21:    21 0.4602041 0.3304304         7 1.178110e+00         3.736110 0.6345513
22:    22 0.6415953 0.4735650         5 3.432675e-01         3.169973 0.6412895
23:    23 0.5508142 0.6384155         7 1.147873e+00         3.661611 0.6351876
24:    24 0.2184162 0.3667865         4 6.483900e-01         3.478897 0.6434970
25:    25 0.5687787 0.5371696         7 4.884735e-01         3.828432 0.6318737
26:    26 0.2155268 0.4938153         7 2.821010e-01         3.304375 0.6425152
27:    27 0.2968529 0.7190589         7 8.387735e-01         3.639808 0.6524283
28:    28 0.1000000 1.0000000         5 2.500411e-03         3.721054 0.6457314
29:    29 0.3334083 0.8742898         6 5.211409e-01         2.916169 0.6458824
30:    30 0.1943026 0.7906576         7 8.539312e-01         3.738882 0.6498531
31:    31 0.3601686 1.0000000         3 1.972077e-01         3.696007 0.6464691
32:    32 0.5363136 0.5440309         5 1.480523e+00         3.500659 0.6450310
33:    33 0.5956435 0.5497040         6 1.412448e+00         3.049917 0.6353444
34:    34 0.2443999 1.0000000         5 1.037051e+00         3.384535 0.6517476
35:    35 0.7000000 1.0000000         3 2.220446e-16         2.601733 0.6377246
36:    36 0.3548609 1.0000000         6 2.172450e-01         3.523769 0.6522648
37:    37 0.7000000 1.0000000         3 1.500000e+00         3.275293 0.6486879
38:    38 0.1922055 0.8773061         7 3.271411e-01         2.889629 0.6451999
39:    39 0.5925558 0.6441024         5 4.862227e-01         3.870132 0.6446674
40:    40 0.6050456 0.8817630         7 5.527467e-01         3.415185 0.6420478
41:    41 0.1864647 0.7102174         8 8.182122e-01         2.193686 0.6221291
42:    42 0.6838267 0.7925174         9 3.115748e-01         1.979235 0.6051728
43:    43 0.5773312 0.9439566         5 1.490878e+00         3.073542 0.6480025
44:    44 0.1466633 0.7983474         4 1.500000e+00         1.757971 0.5921201
45:    45 0.7000000 1.0000000         4 8.332909e-01         3.422513 0.6467235
46:    46 0.5259264 0.7375864         6 8.395382e-01         3.244512 0.6455946
```

```
47:    47 0.2234413 0.7421706         7 6.289001e-01          3.117001 0.6469592
48:    48 0.1384970 1.0000000         4 1.347842e+00          3.304914 0.6430562
49:    49 0.3940161 0.7849640         7 1.217069e-01          3.676411 0.6437476
50:    50 0.6423706 0.9612511         3 1.353232e+00          1.724819 0.6044533
51:    51 0.2426930 0.5576068         3 1.500000e+00          2.450096 0.6262542
52:    52 0.2927153 0.7289104         6 1.437975e+00          3.636931 0.6467242
53:    53 0.1319265 0.7248704         6 5.560855e-01          2.083696 0.6190703
54:    54 0.6339010 0.9961568         6 6.465022e-01          3.369720 0.6507000
55:    55 0.3358889 1.0000000         8 1.053003e+00          2.410837 0.6317602
56:    56 0.5518624 0.7864634         7 1.433544e+00          2.998859 0.6375724
57:    57 0.4315427 0.7243250         9 1.183738e+00          1.586755 0.5974636
58:    58 0.2789593 0.8015799         5 5.029973e-01          3.320299 0.6528510
59:    59 0.5504379 0.7490422         7 1.018472e+00          1.698150 0.6065654
60:    60 0.4865279 0.7485887         4 7.199347e-02          2.981851 0.6406024
61:    61 0.1296976 0.8743269         7 3.112930e-01          3.247139 0.6489979
62:    62 0.3696095 0.9847200         6 3.078562e-01          3.535995 0.6501022
63:    63 0.3003067 1.0000000         4 1.026115e+00          3.766763 0.6471573
64:    64 0.2734712 0.8877327         6 9.500356e-01          3.586885 0.6510978
65:    65 0.2879443 1.0000000         4 8.227296e-01          3.526606 0.6463774
66:    66 0.4704788 0.9088392         5 2.479858e-01          3.555436 0.6520071
67:    67 0.1418718 0.6900475         6 9.205817e-01          3.579566 0.6479723
68:    68 0.4483164 0.9870596         5 8.156943e-02          3.523715 0.6485953
69:    69 0.3598200 0.9382778         5 5.191320e-01          3.780100 0.6518610
70:    70 0.4728556 0.8446750         4 6.296944e-01          3.499014 0.6549139

Optimal values:
             eta        subsample        max_depth            gamma scale_pos_weight
       0.4728556        0.8446750        4.0000000        0.6296944        3.4990144

Best BACC:  0.6549139
```

Table C.4: The confusion matrices corresponding to XGBoost trained on the training set with optimal hyperparameters found with Bayesian optimization with default cut-off. The models are evaluated on the test set.

C.4(a) UCB (cut-off 0.5)

True \ Pred.	1	0
1	1032	702
0	1882	4470

C.4(b) EI (cut-off 0.5)

True \ Pred.	1	0
1	1073	661
0	2012	4340

C.4(c) PI (cut-off 0.5)

True \ Pred.	1	0
1	1021	713
0	1789	4563

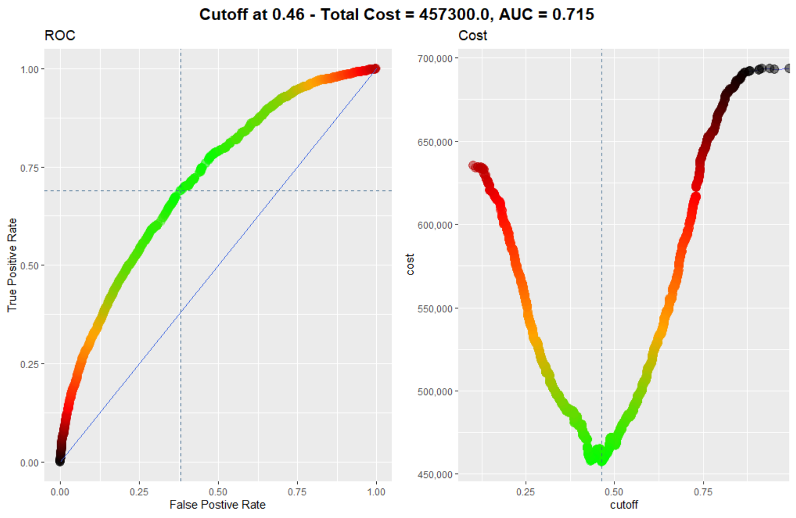


Figure C.11: The ROC curve and cost-plot for different cut-off values for XGBoost trained on the training set with tuned hyperparameters using Bayesian optimization with acquisition function UCB. The cost of FP is set to 100 and the cost of FN is found to be 400. The optimal cut-off value is displayed in the title, together with the AUC.

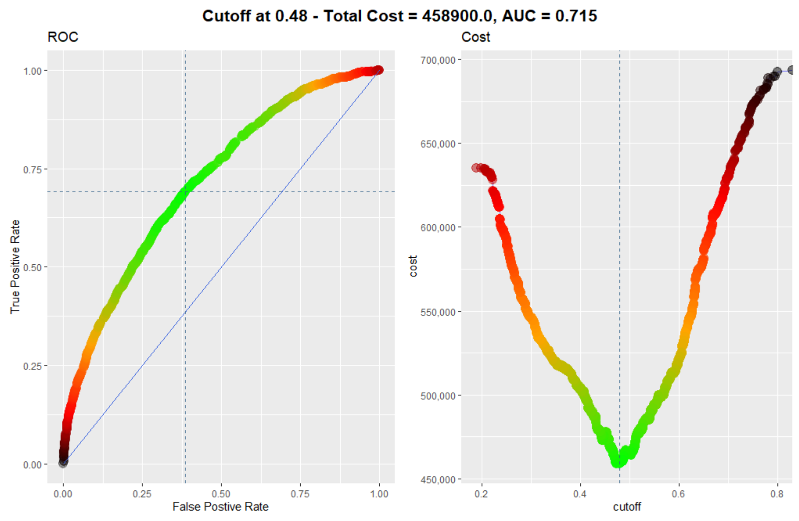


Figure C.12: The ROC curve and cost-plot for different cut-off values for XGBoost trained on the training set with tuned hyperparameters using Bayesian optimization with acquisition function EI. The cost of FP is set to 100 and the cost of FN is found to be 400. The optimal cut-off value is displayed in the title, together with the AUC.

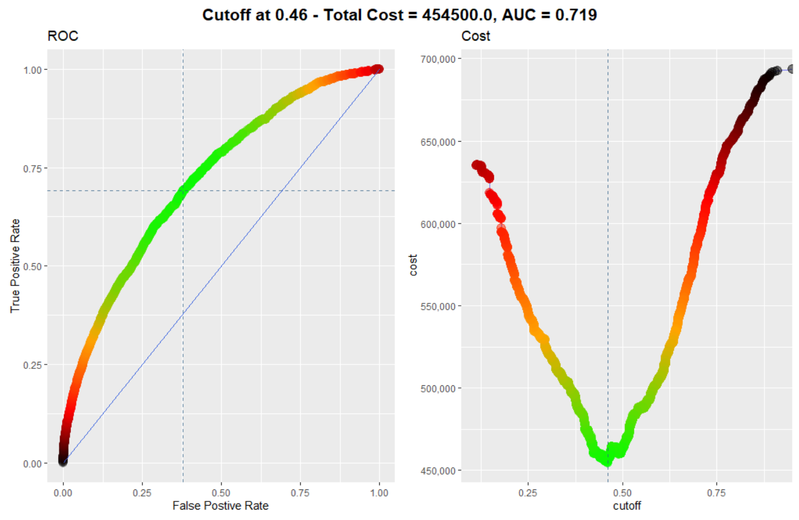


Figure C.13: The ROC curve and cost-plot for different cut-off values for XGBoost trained on the training set with tuned hyperparameters using Bayesian optimization with acquisition function PI. The cost of FP is set to 100 and the cost of FN is found to be 400. The optimal cut-off value is displayed in the title, together with the AUC.

C.4.2 Bayesian optimization combined with Design of Experiments

All iterations of Bayesian optimization in combination with the results of the $2^{(5-1)}$ fractional factorial design with acquisition function UCB and $\kappa = 1.96$.

```
   Round        eta max_depth scale_pos_weight     Value
 1:     1 0.4491402         8         3.637186 0.6421623
 2:     2 0.5921203         8         2.276899 0.6251207
 3:     3 0.3289006         7         1.330560 0.5794724
 4:     4 0.3050537         4         3.987109 0.6478267
 5:     5 0.2745000         6         3.905053 0.6520477
 6:     6 0.1489683         5         1.931084 0.6105997
 7:     7 0.5922910         5         1.463220 0.5936796
 8:     8 0.3816092         5         2.957466 0.6446656
 9:     9 0.6213691         5         1.626989 0.6051673
10:    10 0.5473566         8         3.636628 0.6398186
11:    11 0.5394707         9         2.480171 0.6265922
12:    12 0.2382245         6         2.906801 0.6486415
13:    13 0.6434993         4         1.972099 0.6173251
14:    14 0.5091328         8         2.047039 0.6203028
```

```
15:    15 0.6091114         4         1.944145 0.6162750
16:    16 0.3900268         5         1.760962 0.6030953
17:    17 0.4835125         7         2.553077 0.6314483
18:    18 0.3904014         8         1.361043 0.5859205
19:    19 0.1094583         8         3.134580 0.6474769
20:    20 0.2804745         9         1.987635 0.6186774
21:    21 0.1000000         3         3.449944 0.6401190
22:    22 0.1000000         9         4.000000 0.6463229
23:    23 0.1000000         6         4.000000 0.6507085
24:    24 0.1000000         8         3.536442 0.6461845
25:    25 0.4546565         7         4.000000 0.6477274
26:    26 0.1000000         5         2.935205 0.6411891
27:    27 0.2445655         6         3.274495 0.6530930
28:    28 0.2214957         7         4.000000 0.6504872
29:    29 0.7000000         5         4.000000 0.6430134
30:    30 0.2795892         6         3.515175 0.6533974
31:    31 0.2311313         5         3.636868 0.6515804
32:    32 0.3114030         6         3.416112 0.6483159
33:    33 0.1764628         6         3.830906 0.6502379
34:    34 0.1033362         6         3.721741 0.6509706
35:    35 0.1651823         6         3.793389 0.6493393
36:    36 0.1011427         7         4.000000 0.6483838
37:    37 0.2237802         6         3.693602 0.6494591
38:    38 0.3212641         5         3.799290 0.6482502
39:    39 0.1743965         7         3.491355 0.6507607
40:    40 0.2362646         6         3.672791 0.6498412
41:    41 0.1497512         6         3.750535 0.6497301
42:    42 0.3061736         6         3.850968 0.6532242
43:    43 0.2990772         5         3.699715 0.6499066
44:    44 0.1001262         6         3.734851 0.6522839
45:    45 0.3691539         9         4.000000 0.6366585
46:    46 0.2831781         5         3.643170 0.6515358
47:    47 0.1083346         6         4.000000 0.6506744
48:    48 0.1074982         7         3.647178 0.6506172
49:    49 0.1509843         6         3.718984 0.6497013
50:    50 0.1427504         6         3.764830 0.6505988
51:    51 0.1413750         7         3.651459 0.6493404
52:    52 0.2679421         6         3.670542 0.6525158
53:    53 0.1436086         6         4.000000 0.6492668
54:    54 0.3967746         5         3.728454 0.6483783
55:    55 0.2381692         5         3.698867 0.6498530
56:    56 0.2721443         6         3.666034 0.6533149
57:    57 0.1580903         6         3.718708 0.6494661
58:    58 0.2740354         5         3.834542 0.6496019
59:    59 0.1185241         6         3.872572 0.6523026
60:    60 0.1952997         6         3.522415 0.6507129
61:    61 0.1773719         6         3.610812 0.6528099
62:    62 0.1522158         6         3.771456 0.6489059
63:    63 0.1103067         7         3.675993 0.6484545
64:    64 0.1870598         6         3.659933 0.6502876
65:    65 0.2221276         6         3.680060 0.6484300
66:    66 0.1000035         6         3.935186 0.6509726
67:    67 0.2723804         6         3.556704 0.6527292
68:    68 0.1747034         6         3.610446 0.6514710
69:    69 0.2084980         6         3.721984 0.6508289
70:    70 0.1000000         7         3.635456 0.6495890

Optimal values:
             eta        max_depth scale_pos_weight
       0.2795892        6.0000000        3.5151750
Best BACC: 0.6533974
```

Table C.5: The confusion matrix corresponding to XGBoost trained on the training set with optimal hyperparameters (Table 5.27) found with Bayesian optimization with acquisition function UCB in combination with the results of the $2^{(5-1)}$ fractional factorial design. The model is evaluated on the test set with default cut-off.

Pred. / True	1	0
1	1028	706
0	1916	4436

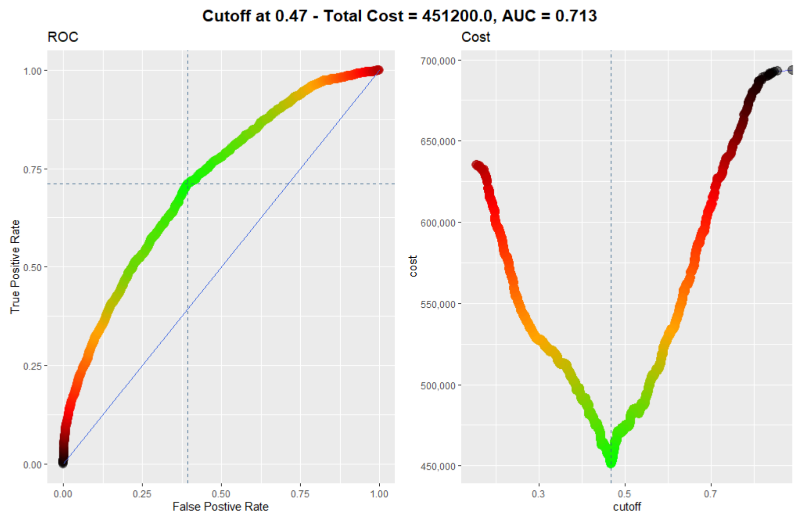


Figure C.14: The ROC curve and cost-plot for different cut-off values for XGBoost trained on the training set with tuned hyperparameters using Bayesian optimization in combination with the results of the $2^{(5-1)}$ fractional factorial design. The optimization is done with acquisition function UCB. The cost of FP is set to 100 and the cost of FN is found to be 400. The optimal cut-off value is displayed in the title, together with the AUC.

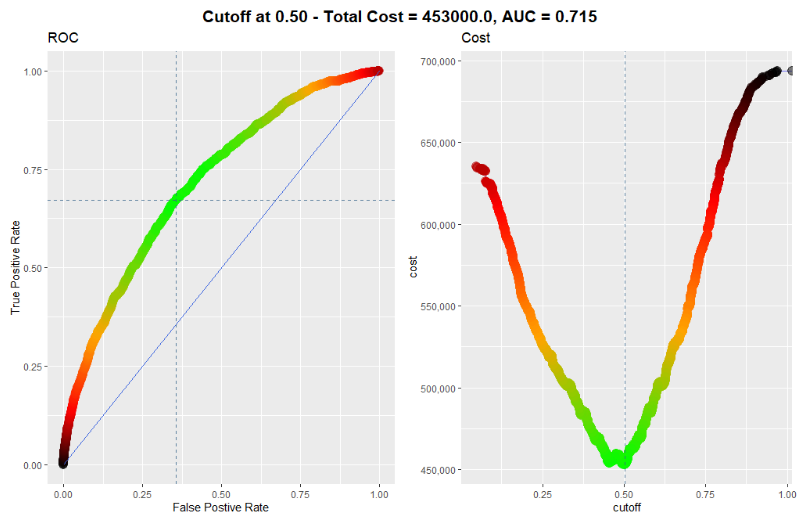


Figure C.15: The ROC curve and cost-plot for different cut-off values for XGBoost trained on the training set with tuned hyperparameters using Bayesian optimization in combination with the results of the $2^{(5-1)}$ fractional factorial design. The optimization is done with acquisition function EI. The cost of FP is set to 100 and the cost of FN is found to be 400. The optimal cut-off value is displayed in the title, together with the AUC.

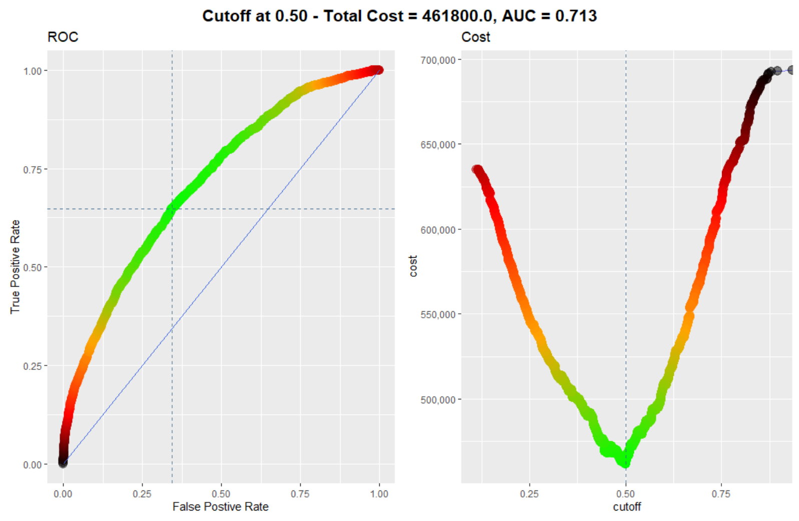


Figure C.16: The ROC curve and cost-plot for different cut-off values for XGBoost trained on the training set with tuned hyperparameters using Bayesian optimization in combination with the results of the $2^{(5-1)}$ fractional factorial design. The optimization is done with acquisition function PI. The cost of FP is set to 100 and the cost of FN is found to be 400. The optimal cut-off value is displayed in the title, together with the AUC.

C.4.3 Bayesian optimization combined with Response Surface Methodology

All iterations of Bayesian optimization with the CCD as initial grid with acquisition function UCB and $\kappa = 1.96$. The CCD has center in eta= 0.48145, max_depth= 4, and scale_pos_weight= 3.7283.

```
    Round        eta max_depth scale_pos_weight     Value
 1:     1 0.11950000         4         3.678400 0.6448606
 2:     2 0.21950000         4         3.678400 0.6455268
 3:     3 0.11950000         6         3.678400 0.6513193
 4:     4 0.21950000         6         3.678400 0.6488250
 5:     5 0.11950000         4         3.778400 0.6439457
 6:     6 0.21950000         4         3.778400 0.6461695
 7:     7 0.11950000         6         3.778400 0.6513541
 8:     8 0.21950000         6         3.778400 0.6491313
 9:     9 0.08289746         5         3.728400 0.6448474
10:    10 0.25610254         5         3.728400 0.6510108
11:    11 0.16950000         3         3.728400 0.6401650
12:    12 0.16950000         7         3.728400 0.6519722
13:    13 0.16950000         5         3.641797 0.6498186
14:    14 0.16950000         5         3.815003 0.6464415
15:    15 0.16950000         5         3.728400 0.6510334
16:    16 0.16950000         5         3.728400 0.6510334
17:    17 0.16950000         5         3.728400 0.6510334
18:    18 0.11953365         7         3.780385 0.6492457
19:    19 0.15847819         6         3.725199 0.6510749
20:    20 0.14144896         7         3.655156 0.6497033
21:    21 0.14088385         6         3.727541 0.6514946
22:    22 0.20620079         7         3.749546 0.6484951
23:    23 0.20610254         5         3.711997 0.6495495
24:    24 0.27877247         5         3.464582 0.6521518
25:    25 0.26832299         6         3.466624 0.6510827
26:    26 0.30023612         5         3.411516 0.6495663
27:    27 0.30649123         5         3.574209 0.6504574
28:    28 0.26819164         4         3.441790 0.6475747
29:    29 0.29038580         6         3.662587 0.6543608
30:    30 0.30362604         8         3.674509 0.6437095
31:    31 0.32196303         6         3.670189 0.6491460
32:    32 0.27246001         6         3.714330 0.6513072
33:    33 0.28757672         5         3.766943 0.6507238
34:    34 0.17058278         8         3.723572 0.6458608
35:    35 0.17526296         7         3.669562 0.6509063
36:    36 0.28235240         6         3.605512 0.6523434
37:    37 0.28859325         5         3.663076 0.6500279
38:    38 0.31133059         5         3.871676 0.6491087
39:    39 0.29203042         6         3.662334 0.6526838
40:    40 0.15816558         6         3.484730 0.6485452
41:    41 0.26079494         5         3.888879 0.6492235
42:    42 0.11673982         7         3.498153 0.6504708
43:    43 0.28315600         6         3.121361 0.6495109
44:    44 0.15142860         5         3.623549 0.6477146
45:    45 0.31552217         5         3.545384 0.6510352
46:    46 0.28415850         6         3.790268 0.6521820
```

```
47:    47 0.29207626         6          3.401113 0.6500192
48:    48 0.12923182         7          3.083409 0.6469584
49:    49 0.22420163         5          3.598956 0.6527979
50:    50 0.10745431         6          3.608417 0.6496004
51:    51 0.28595600         5          3.810014 0.6508668
52:    52 0.12736718         6          3.564444 0.6497548
53:    53 0.12492739         7          3.454451 0.6486004
54:    54 0.33283207         5          3.492735 0.6521752
55:    55 0.28461684         6          3.318335 0.6508412
56:    56 0.28080931         5          3.844808 0.6521230
57:    57 0.11649259         6          3.496623 0.6500896
58:    58 0.28017444         5          3.223380 0.6499064
59:    59 0.17539060         7          3.269598 0.6498650
60:    60 0.18472963         7          3.388756 0.6502689
61:    61 0.26962779         6          3.571927 0.6514782
62:    62 0.70000000         4          3.216179 0.6488110
63:    63 0.11244047         7          3.426398 0.6489993
64:    64 0.33688610         5          3.113000 0.6491731
65:    65 0.10385013         6          3.445556 0.6485118
66:    66 0.22893356         5          3.444621 0.6503941
67:    67 0.25446701         5          3.376234 0.6506822

Optimal values:
             eta        max_depth scale_pos_weight
       0.2903858        6.0000000        3.6625870

Best BACC: 0.6543608
```

Table C.6: The confusion matrices corresponding to XGBoost trained on the training set with optimal hyperparameters (Table 5.30) found with Bayesian optimization with the central composite design as initial grid. The models are evaluated on the test set with default cut-off.

C.6(a) UCB (cut-off 0.5)

True \ Pred.	1	0
1	1091	643
0	2065	4287

C.6(b) EI (cut-off 0.5)

True \ Pred.	1	0
1	1120	614
0	2228	4124

C.6(c) PI (cut-off 0.5)

True \ Pred.	1	0
1	1035	699
0	1837	4515

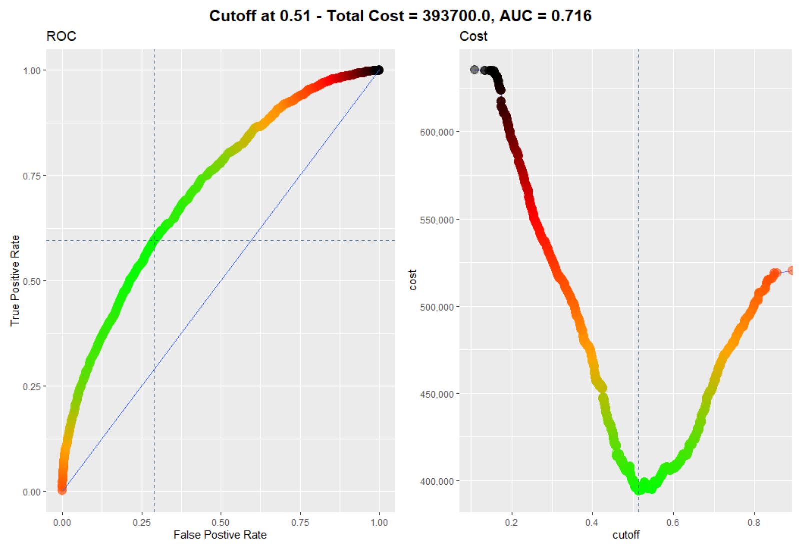


Figure C.17: The ROC curve and cost-plot for different cut-off values for XGBoost trained on the training set with tuned hyperparameters using Bayesian optimization with the central composite design as initial grid. The optimization is done with acquisition function UCB. The cost of FP is set to 100 and the cost of FN is found to be 300. The optimal cut-off value is displayed in the title, together with the AUC.

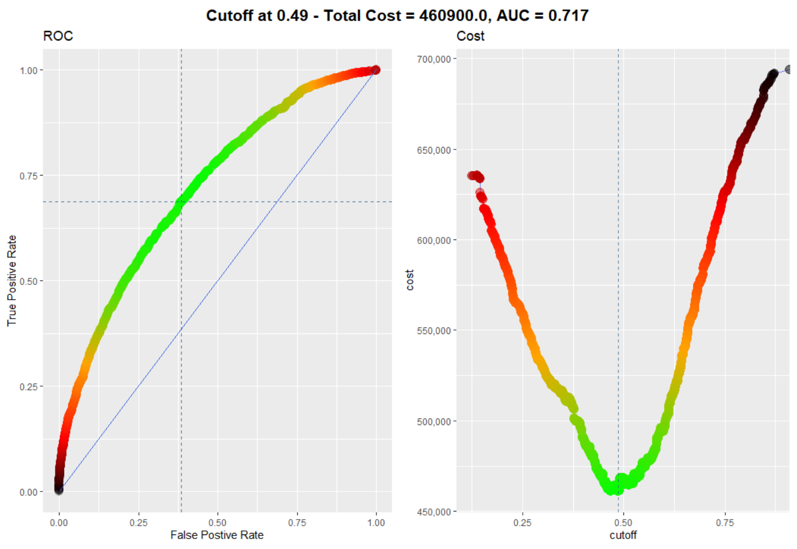


Figure C.18: The ROC curve and cost-plot for different cut-off values for XGBoost trained on the training set with tuned hyperparameters using Bayesian optimization with the central composite design as initial grid. The optimization is done with acquisition function EI. The cost of FP is set to 100 and the cost of FN is found to be 400. The optimal cut-off value is displayed in the title, together with the AUC.

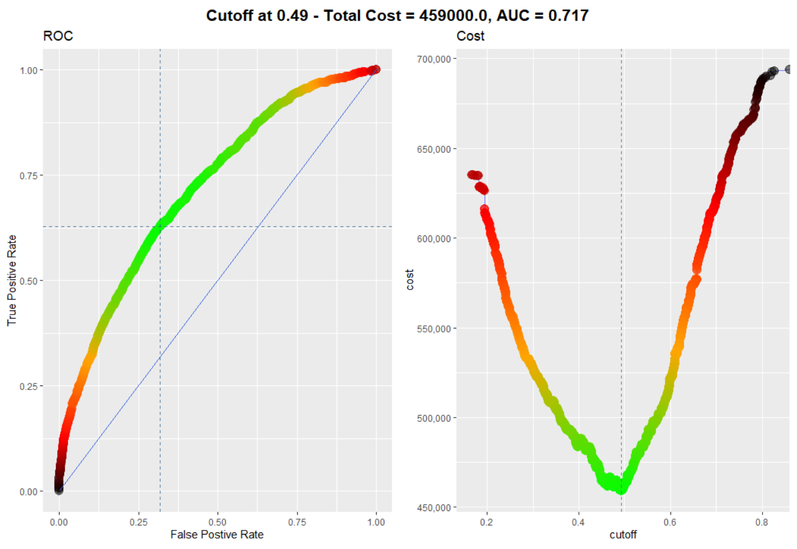


Figure C.19: The ROC curve and cost-plot for different cut-off values for XGBoost trained on the training set with tuned hyperparameters using Bayesian optimization with the central composite design as initial grid. The optimization is done with acquisition function PI. The cost of FP is set to 100 and the cost of FN is found to be 400. The optimal cut-off value is displayed in the title, together with the AUC.

C.5 Feature Importance

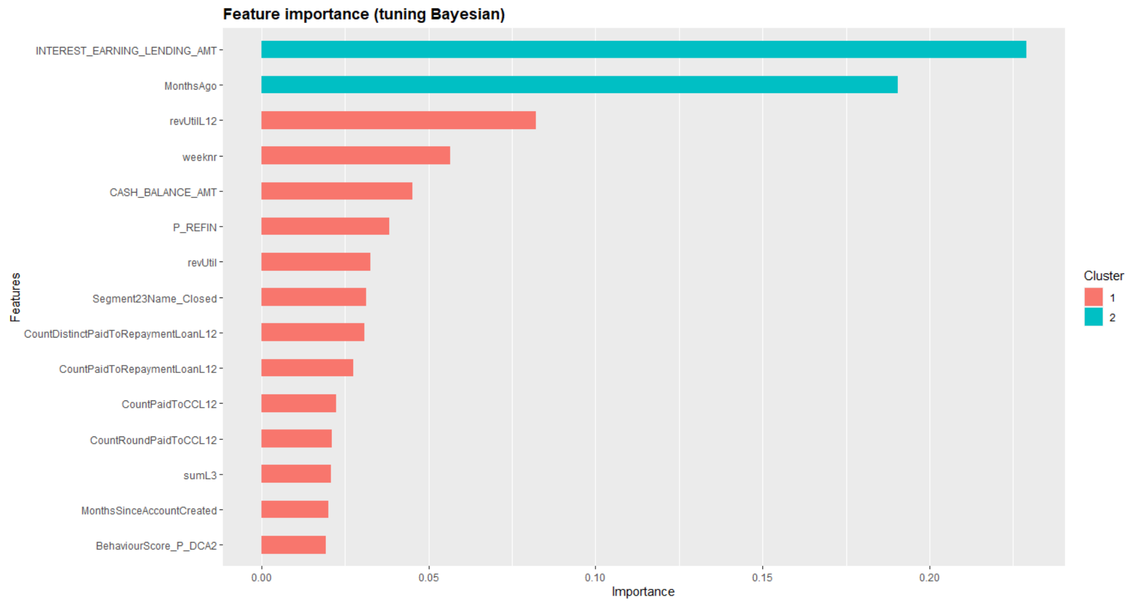


Figure C.20: Feature importance calculated for XGBoost trained on the training set with tuned hyperparameters from Bayesian optimization in combination with the results of the $2^{(5-1)}$ fractional factorial design with acquisition function EI (Table 5.27).

Appendix D

Random Forests Modelling

D.1 Design of Experiments

The output from the results of the two replicates of the $2^{(5-1)}$ fractional factorial design using Random Forests. The calculated responses after both replicates of the design are displayed in Table D.1.

Table D.1: The results of 16 runs of two replicates of Random Forests trained on the training set using the $2^{(5-1)}$ fractional factorial design. The values of the factors A, B, C, D, and E are set to the values in Table 5.36.

Run	A	B	C	D	E = ABCD	Level code	BACC
1	-1	-1	-1	-1	1	1	0.5005
2	1	-1	-1	-1	-1	a	0.5086
3	-1	1	-1	-1	-1	b	0.6552
4	1	1	-1	-1	1	abe	0.6476
5	-1	-1	1	-1	-1	c	0.5004
6	1	-1	1	-1	1	ace	0.5066
7	-1	1	1	-1	1	bce	0.6603
8	1	1	1	-1	-1	abc	0.6509
9	-1	-1	-1	1	-1	d	0.5002
10	1	-1	-1	1	1	ade	0.5059
11	-1	1	-1	1	1	bde	0.6601
12	1	1	-1	1	-1	abd	0.6517
13	-1	-1	1	1	1	cde	0.5000
14	1	-1	1	1	-1	acd	0.5076
15	-1	1	1	1	-1	bcd	0.6606
16	1	1	1	1	1	abcde	0.6539
17	-1	-1	-1	-1	1	1	0.5005
18	1	-1	-1	-1	-1	a	0.5089
19	-1	1	-1	-1	-1	b	0.6570
20	1	1	-1	-1	1	abe	0.6492
21	-1	-1	1	-1	-1	c	0.5005
22	1	-1	1	-1	1	ace	0.5066
23	-1	1	1	-1	1	bce	0.6621
24	1	1	1	-1	-1	abc	0.6500
25	-1	-1	-1	1	-1	d	0.5003
26	1	-1	-1	1	1	ade	0.5066
27	-1	1	-1	1	1	bde	0.6586
28	1	1	-1	1	-1	abd	0.6526
29	-1	-1	1	1	1	cde	0.5001
30	1	-1	1	1	-1	acd	0.5079
31	-1	1	1	1	-1	bcd	0.6585
32	1	1	1	1	1	abcde	0.6560

The model summary of the fitted linear model to the results of the $2^{(5-1)}$ fractional factorial design:

```
Call:
lm.default(formula = y ~ .^2, data = plan)

Residuals:
       Min         1Q     Median         3Q        Max
-0.0010949 -0.0003809  0.0000000  0.0003809  0.0010949

Coefficients:
              Estimate Std. Error  t value Pr(>|t|)
(Intercept)  5.795e-01  1.516e-04 3822.420  < 2e-16 ***
A           -1.265e-04  1.516e-04   -0.834 0.416498
B            7.573e-02  1.516e-04  499.504  < 2e-16 ***
C            5.805e-04  1.516e-04    3.829 0.001479 **
D            4.963e-04  1.516e-04    3.273 0.004783 **
E            1.150e-04  1.516e-04    0.758 0.459303
A:B         -3.653e-03  1.516e-04  -24.094 5.33e-14 ***
A:C         -5.065e-05  1.516e-04   -0.334 0.742695
A:D          3.686e-04  1.516e-04    2.431 0.027175 *
A:E         -4.838e-04  1.516e-04   -3.191 0.005683 **
B:C          6.832e-04  1.516e-04    4.506 0.000359 ***
B:D          7.358e-04  1.516e-04    4.853 0.000176 ***
B:E          6.001e-04  1.516e-04    3.958 0.001127 **
```

```
C:D         -4.112e-05  1.516e-04   -0.271 0.789682
C:E          4.593e-04  1.516e-04    3.030 0.007972 **
D:E         -7.747e-06  1.516e-04   -0.051 0.959881
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.0008577 on 16 degrees of freedom
Multiple R-squared:  0.9999,Adjusted R-squared:  0.9999
F-statistic: 1.668e+04 on 15 and 16 DF,  p-value: < 2.2e-16
```

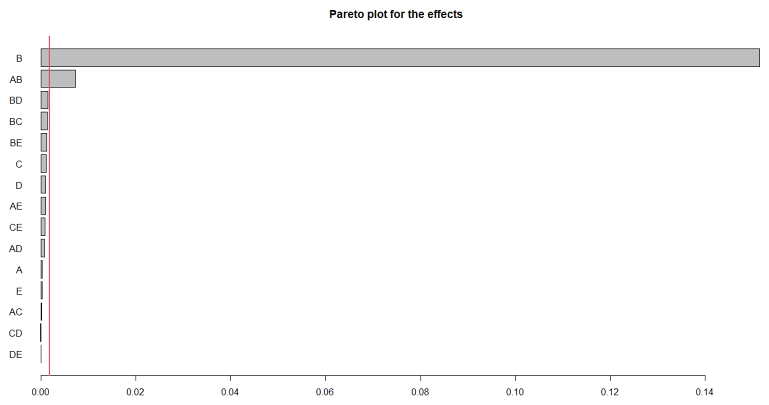


Figure D.1: Pareto plot of the estimated effects of the $2^{(5-1)}$ fractional factorial design. The significance level is calculated using Lenth's method, [34], with significance level $\alpha = 0.05$.

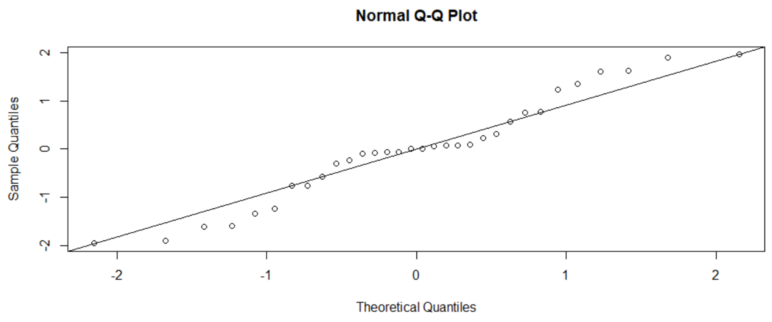


Figure D.2: The Q-Q plot of the residuals from the fitted linear model of the results from the $2^{(5-1)}$ fractional factorial design with Random Forests.

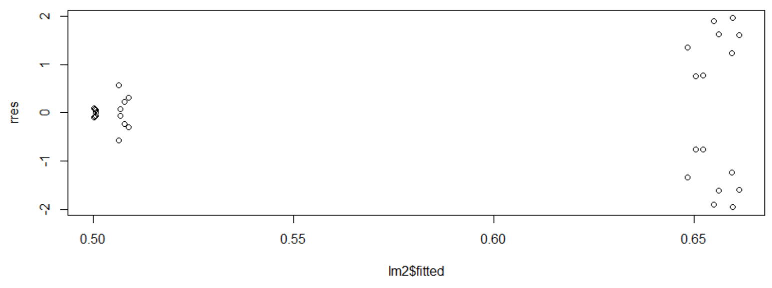


Figure D.3: Plot of the residuals from the fitted linear model of the results from the $2^{(5-1)}$ fractional factorial design with Random Forests.

D.2 Response Surface Methodology

The central composite design of the second-order response surface model fitted to the interaction between mtry and cutoff:

```
   run.order std.order      mtry   cut_off         y
1          1         1 2.0000000 0.6600000 0.6582536
2          2         2 8.0000000 0.6600000 0.6429666
3          3         3 2.0000000 0.8600000 0.6422321
4          4         4 8.0000000 0.8600000 0.6203296
5          1         1 0.7573593 0.7600000 0.5827191
6          2         2 9.2426407 0.7600000 0.6609924
7          3         3 5.0000000 0.6185786 0.6250844
8          4         4 5.0000000 0.9014214 0.5880985
9          5         5 5.0000000 0.7600000 0.6629251
10         6         6 5.0000000 0.7600000 0.6593970
11         7         7 5.0000000 0.7600000 0.6621889

Data are stored in coded form using these coding formulas ...
x1 ~ (mtry - 5)/3
x2 ~ (cut_off - 0.76)/0.1
```

The model summary of the second-order response surface model fitted to the design:

```
Call:
rsm(formula = y ~ SO(x1, x2), data = ccd)

              Estimate Std. Error t value  Pr(>|t|)
(Intercept)  0.6615037  0.0167174 39.5698 1.943e-07 ***
x1           0.0091882  0.0102373  0.8975    0.4106
x2          -0.0113706  0.0102373 -1.1107    0.3172
x1:x2       -0.0016539  0.0144777 -0.1142    0.9135
x1^2        -0.0131435  0.0121848 -1.0787    0.3300
x2^2        -0.0207756  0.0121848 -1.7050    0.1489
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Multiple R-squared:  0.5157,Adjusted R-squared:  0.0314
F-statistic: 1.065 on 5 and 5 DF,  p-value: 0.4734

Analysis of Variance Table

Response: y
            Df    Sum Sq    Mean Sq  F value   Pr(>F)
FO(x1, x2)   2 0.0017097 0.00085485   1.0196 0.425221
TWI(x1, x2)  1 0.0000109 0.00001094   0.0130 0.913497
PQ(x1, x2)   2 0.0027432 0.00137160   1.6359 0.284062
Residuals    5 0.0041921 0.00083841
Lack of fit  3 0.0041851 0.00139504 402.7306 0.002478
Pure error   2 0.0000069 0.00000346

Stationary point of response surface:
        x1         x2
 0.3676726 -0.2882857

Stationary point in original units:
     mtry   cut_off
6.1030177 0.7311714

Eigenanalysis:
eigen() decomposition
$values
[1] -0.01305492 -0.02086421

$vectors
         [,1]      [,2]
x1 -0.9943131 0.1064965
x2  0.1064965 0.9943131
```

The corresponding estimated canonical path:

```
  dist     x1     x2 |   mtry cut_off |  yhat
1   -2  2.356 -0.501 | 12.068  0.7099 | 0.613
2   -1  1.362 -0.395 |  9.086  0.7205 | 0.652
3    0  0.368 -0.288 |  6.104  0.7312 | 0.665
4    1 -0.627 -0.182 |  3.119  0.7418 | 0.652
5    2 -1.621 -0.075 |  0.137  0.7525 | 0.613
6    3 -2.615  0.031 | -2.845  0.7631 | 0.547
7    4 -3.610  0.138 | -5.830  0.7738 | 0.456
8    5 -4.604  0.244 | -8.812  0.7844 | 0.338
```

D.3 Feature Importance

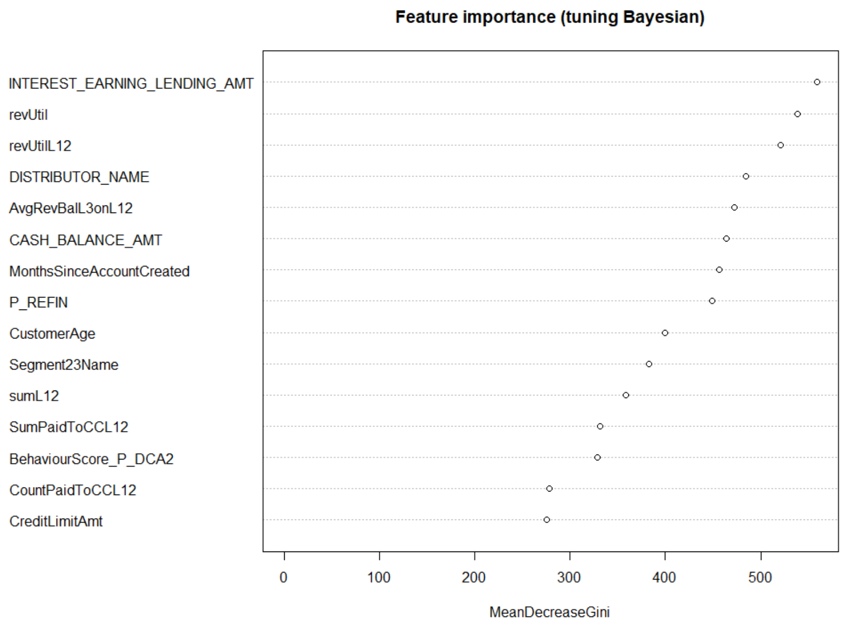


Figure D.4: Feature importance calculated for Random Forests trained on the training set with tuned hyperparameters from Bayesian optimization with acquisition function UCB (Table 5.55).

NTNU
Kunnskap for en bedre verden

SpareBank 1