Prediction of quote acceptance in a B2B environment using Random Forests and Gradient Boosting Machines

Jesper Gunnarsson, Jacob Tyrberg

Master's thesis
2021:E4

Lund Institute of Technology
Lund University

Faculty of Engineering
Centre for Mathematical Sciences
Mathematical Statistics

Abstract

For a business to be as successful as possible it needs a sound pricing strategy. A B2B environment allows the business more freedom to tailor each quote to maximize the performance. In order to do this, proper understanding of how probable a quote is to succeed is crucial. This work employs a statistical approach to predict the probability of acceptance based on historical data.

Two different architectures for models were mainly used to compute the probability of acceptance, Gradient Boosting Machines and Random Forests. To improve the models, feature engineering, feature selection, hyperparameter optimization and probability calibration were used. Each step was evaluated in order to determine its success.

Feature engineering, using domain knowledge from sales, significantly improved the results, by 10 percentage points in the models' F1-score. The final binary classification results for the two models are similar, both producing ca 90% F1-score. Where the two models differ is in the behaviour when a single explanatory variable, the price of the quote, is altered. GBM produces probabilities that are more aligned with expectations from experts. The results show that direct price optimization is difficult to use, regardless of the model, as the probabilities are not entirely trustworthy.

The thesis proves the possibility of working with quote prediction using quantitative methods, but also highlights the many challenges it poses for a company.

Keywords— B2B, pricing, Random Forest, Gradient Boosting, calibration, feature engineering

Acknowledgements

We would like to thank Solveigh Plenge from Alfa Laval for her unwavering support and great advice. This thesis would have been half as good and taken twice as long without her.

Furthermore, we would like to thank all the other people we have been in contact with during the course of the thesis at Alfa Laval. Everyone we have been in contact with have always been very accommodating, encouraging and provided valuable insights regarding the company and the sales process. Especially Hans Kristian Knutsson for thoughtful, and at times, challenging inquisitions.

Furthermore, we would also like to thank our supervisor at the university, Magnus Wiktorsson, for his invaluable contributions and feedback, and willingness to discuss interesting ideas and concepts.

We would also like to thank Simba the cat, for making every meeting a bit more interesting.

Lund, March 2021
Jacob Tyrberg & Jesper Gunnarsson

Contents

Chapter 1

Introduction

1.1 Background

Every company aims to have a pricing policy that is the most optimal for their business. What constitutes the most optimal policy in this context can be to maximize profit, revenue, stability or something else that coincides with the company's business model. The pricing strategy differs greatly whether the sector is business to business (B2B) or business to consumer (B2C). In an article the authors mention three distinct key differences between the two sectors [Zhang et al., 2014]:

1. In a B2B setting, sellers can set a price dynamically depending on the specifics of a given transaction. In contrast, in a B2C setting, the sellers are limited in adjusting the price for specific customers.
2. The transaction is typically more complex in a B2B setting. B2B buyers can buy directly (direct order) or ask for a quote. Since quotes can be rejected or accepted, there is data that can be used in order to get a better understanding of the buyer's characteristics. In a B2C setting, there is typically less room for negotiations.
3. Unlike B2C pricing, there is generally no information on the competitors' prices for B2B pricing.

Pricing in B2B markets is infamously complex, with several factors affecting the price decision and affecting each other [Pressey et al., 2011][pp. 271-272]. The purchasing decision in itself is also complex, as the buyers have several factors that affect their decision that are not seen by the sellers. Difficulties encountered in previous research include limitations of the data available. Since pricing and buying strategies have a significant effect on a company's performance, it could be detrimental to the company if these strategies were publicly known [Pressey et al., 2011][p. 40].

This creates the problem of asymmetrical data. If researchers collaborate with one company, the only interaction they have with the customers is a binary response [Qu et al., 2020], either the quote is accepted and turned into an order, or it is declined. The researchers cannot ask the customers if they would have been willing to pay more after a successful transaction is completed, as this is information the seller would utilize in their next transaction. Despite these difficulties, companies today are more and more interested in trying to utilize dynamic and targeted price optimizations in order to increase their revenue and/or margin.

This thesis is a collaboration with a company with an increasing interest in data driven pricing strategies, Alfa Laval. Alfa Laval is a provider of a wide range of products in heat transfer, separation and fluid handling processes [AlfaLaval, 2021]. The company is organized in a matrix organisation divided into three divisions; Energy, Food & Water and Marine. Several Business Units (BU) resides in the divisions, where each BU is in charge of a product. This thesis will focus on Gasketed Plate Heat Exchangers (BU GPHE) in the Energy division.

1.2 Previous work

B2B research account for less than 10% of the papers published in marketing journals [LaPlaca and Katrichis, 2009], and a subset of those papers deal specifically with dynamic pricing within B2B pricing. So the available literature is limited, but there have been some studies specifically trying to use quantitive methods for B2B price modeling.

In a study the authors model the process of accepting the price as a Markov process, using a hidden Markov model. They chose this model due to the sequential nature of a B2B process, where a previous purchase event will affect a coming purchasing event. They discovered that the buyers can be in either a *relaxed* or a *vigilant* state. In a relaxed state the buyers are more likely to accept a quote, but if the seller increases the prices the buyer can transition to the vigilant state, causing them to be more conservative when accepting quotes [Zhang et al., 2014].

In another article the authors attempt to model the demand curve given a certain set of features. The demand curve is then used in a logistic regression model in order to estimate the probability that a quote will be accepted by a customer. The authors state that the opportunity cost for lost quotes is high, which presses the importance of using new observations as effectively as possible, for this they use a Bayesian approach to logistic regression. This is then used in order to create a goal function which maximizes revenue. They show that the goal function is convex, hence simple to optimize using stochastic gradient descent [Qu et al., 2020].

A simple linear regression which tries to mimic the behaviour of a salesperson can also be used, as another article mentioned. They found that the model is able to increase profitability in most cases, however, in situations where humans were able to utilize specific and complex knowledge regarding a transaction, the human outperformed the model. Therefore, they propose a Random Forest model that combines with human input, in order to create a framework which outperforms both the linear model and the human [Karlinsky-Shichor and Netzer, 2019].

1.3 Purpose and problem formulation

The purpose of this thesis is to investigate if it is possible to optimize a price of a quote, using machine learning algorithms to predict the probability of the quote acceptance.

The thesis will answer the following questions:

1. Is it possible to find patterns in historical quotation data in order to predict the probability of acceptance of future quotes?
2. What is a suitable model to use for prediction of the acceptance probability?
3. Is it possible to optimize a price for a future quote, given a predictive quote acceptance model, to maximize revenue?

1.4 Limitations and assumptions

Alfa Laval is a large company that has a diverse product portfolio. In order to decrease the complexity of the problem, only standard spare part quotes for one specific business unit will be studied. The business unit sells parts for Gasketed Plate Heat Exchangers (GPHE).

Historic quote data will be used in machine learning models, with a binary response variable being either that the quote was won or lost. This will be a limitation as many quotes might not be lost, in the sense that a quote was lost to a competitor, even if they are not won. Reasons for a quote not being won can be that the customers changed their minds about their needs, that the customers only wished to know a price to plan for a future purchase etc. These patterns are not visible in the data and can therefore not be captured in a model.

1.5 Thesis outline

The thesis begins with a thorough introduction to the problems posed in B2B pricing. Then, an introduction to the pipeline of the project is provided, followed by a description of the data and the theory related to the machine learning used in the study. Finally the results are presented and discussed. The report concludes with a brief evaluation of the work and suggested research topics for future studies.

Chapter 2

Pipeline

Before any analysis or model training can be done, the data needs to be gathered and processed so that the data is in the right format and have the best conditions to be used later. After the data is processed, more features are added through feature engineering, and finally the most relevant features are selected through a feature selection process. These features can then be used in several machine learning models, where the hyperparameters are tuned for best performance and the models are calibrated to have correct probabilities in a frequentistic sense. All these steps will be further explained in the coming chapters, and can be summarized in a so-called pipeline:

1. Data is extracted from the data source
2. Data processing and feature engineering
   (a) Preprocessing of the data. Quotes which do not have vital information are removed.
   (b) Feature engineering. More explanatory variables are added to the data.
3. Model-training and calibration
   (a) Machine learning model is trained
   (b) Feature selection. Only the relevant features are chosen.
   (c) Random search hypterparameter optimization
   (d) Probability calibration
4. Validation and evaluation

Chapter 3

Data and feature engineering

The data is extracted from an Enterprise Resource Planning (ERP) system, which integrates databases between operations and supply chains [Pressey et al., 2011][p. 50]. Quotes can be created in the ERP system, and the quotes that are won are converted into an order in the same system.

The data originates from four different sales companies, covering several countries, and the samples are combined into one data set.

3.1 Data

The samples in the data are on a quote-line level, meaning that each sample is one line in a quote, i.e. one article. This means that even if an entire quote, with several lines, is either won or lost each line will still be treated separately. This results in the data not being completely independently and identically distributed (i.i.d.), since in theory some samples might depend on each other, which is a fundamental flaw in the data for it to be used in machine learning principles. However, the data shows that there are differences in certain lines for a won quote and the order that the quote is turned into, both in terms of values in the explanatory variables and response variables, which indicates that customers still make individual decisions on each quote-line. Since the differences cover a significant part of the data, the assumption is made that each line is treated individually and that the data is sufficiently independent.

A total of 77 845 samples were used in the data set, with an additional 97 277 samples of order-line data which are not connected to a quote. These order samples are called direct orders, which are orders that the customers order from Alfa Laval without having the need of asking for a quote. Since the direct orders can only be considered as a "won" quote, these samples are not used directly in the training or testing of a model. However, the direct orders have been used when engineering additional explanatory features, as they still contain information of historic transactions, described in more detail in section 3.2. The data is not significantly imbalanced to be an issue.

Each sample in the data contained information about the article, customer and the purchasing event. Three examples of samples can be seen in Table 3.1. Each article has a list price and a net price, as the net price does not necessarily need to be the same as the list price in the sample. All prices are in Euros and the sample prices that were originally set in another currency were converted using the company's internal conversion rates.

Each article has an article ID and an article group ID, which is an explanatory article identifier in a higher hierarchical level.

	Sample 1	**Sample 2**	**Sample 3**
Article ID	12345	12346	12345
Article group ID	123	321	123
Quantity	1	2	10
Unit list price	32	23	32
Line list price	32	46	320
Unit net price	30	25	30
Line list price	30	50	300
Channel	End user	End user	Reseller
Application	Heating	Heating	Cooling
Customer	A	A	B
Sales site	U	U	X
Sales person ID	ABC	ABC	DEF
Date	20210105	20210105	20210103
Quote ID	1	1	2
Quote status	Won	Won	Lost

Table 3.1: An example of three different samples in the data, with example values. Sample 1 and 2 are from the same quote, and sample 3 is a quote on its own.

In the cases where there were missing values, an effort was made in order to infer the missing value based on other values. In the cases where this was not possible, the entries were removed. The entire data set was divided into four parts for training, validation, calibration and testing. 70% was used for training, 10% for calibration, 10% for validation and 10% for testing. Random sampling was used to divide the data.

3.2 Feature engineering

To increase the number of explanatory variables from the data, several features were engineered using domain knowledge about the problem. Discussions with sales controllers, managers and product experts were the main source of the domain knowledge used to construct additional features. This is common practice for machine learning models using tabular data, when artificial neural networks (ANNs) are not used and the available data is not explanatory enough in itself [Géron, 2019][p. 25]. This however, does not mean that new data is used, only that the available data is used to calculate additional features, as means and variances etc. that can be of interest.

Both categorical and numerical features were obtained, where the majority were engineered from the data with regard to customer, article or article group. For all numerical features the mean, μ, standard error, SE, and coefficient of variation, CV, were calculated and added as features as well. The coefficient of variation is defined as [Koopmans et al., 1964]

$$CV = \frac{SE}{\mu}.$$

3.2.1 One-hot encoding

In order to handle the categorical variables, one-hot encoding was used. One-hot encoding is a common approach to handle categorical features [Harris, 2013]. The idea is to create vectors with the length equal to the number of categories. The vector contains zeros everywhere except for one element, and that location is associated with one specific categorical value. An example of a categorical feature converted into one-hot encoded values can be seen in Figure 3.1.

ID	Quarter
1	Q1
2	Q2
3	Q3
4	Q4
5	Q3

ID	Quarter_Q1	Quarter_Q2	Quarter_Q3	Quarter_Q4
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1
5	0	0	1	0

Figure 3.1: Example of one-hot encoding.

One problem with this approach is that the number of features increase exponentially, as each unique categorical value will introduce an additional feature. If this is not handled properly, the dimensionality of the feature space will make the computation unmanageable, which is referred to as the *curse of dimensionality* [Richard, 2015].

As a result of this, some categorical features which contained a large amount of categories were omitted in the modeling. To prevent significant information loss, these features were replaced by "super classes", i.e. a class of a higher hierarchy. An example of this was the Article ID feature. It was removed due to a high number of unique values, and was replaced by Article group ID.

When creating a parametric model using one-hot encoded categorical features, e.g. logistic regression, it is important to drop one of the categories due to multicollinearity. This is due to the fact that the model uses an intercept which, when modelling, will contain the same information as one of the categories. One of the categories will therefore be set as the reference category, and in practice this category will be removed from the data. If this is not done, some of the features will contain redundant information which in worst case could hurt the model [Pantula et al.,][p. 273]. However, non-parametric models, e.g. decision trees or tree-based models, do not use an intercept which means that no categories need to be removed. If categories are removed for non-parametric models, information can be lost and it could hurt the model.

Chapter 4

Machine Learning

Several classifier models were studied, including a logistic regression and several variations of tree classifiers. These models were chosen as they have been proven to be stable for tabular data. Some of the advantages of tree based models are said to be [Hastie et al., 2009][p.352]:

1. Relative fast training times.
2. Easy to combine the usage of categorical and numerical features.
3. Robust against outliers.
4. Robust against the inclusion of irrelevant features.
5. Automatic handling of interaction information.

What follows is a brief overview of the different classifiers that were used. First, Logistic Regression is explained, then a decision tree and bootstrap aggregating classifier are described. Then, Random Forests and Gradient Boosting Machines are described, which were the two models that showed the most promise and will be the methods that the results will focus on.

4.1 Logistic regression

For binary classification, a response variable, y_i, is classified as either 0 or 1. This can be interpreted as the response variable having a binomial distribution, $y_i \sim Bin(1, p_i)$ with $p_i = \mathbb{P}(y_i = 1)$ being the probability of a positive response. This probability can be modeled using the logistic function

$$p_i = \frac{e^{\mathbf{x_i}\beta}}{1 + e^{\mathbf{x_i}\beta}} \tag{4.1}$$

by using the odds, defined as

$$\text{odds} = \frac{\mathbb{P}(y_i = 1)}{\mathbb{P}(y_i = 0)} = \frac{p_i}{1 - p_i} = e^{\mathbf{x_i}\beta}. \tag{4.2}$$

By taking the logarithm of the odds, the values will not be bounded, and the result becomes

$$\ln\left(\frac{p_i}{1-p_i}\right) = \mathbf{x_i}\beta \tag{4.3}$$

and is called logit(p_i). The parameters, β, are what needs to be found by the training process, which is done using maximum likelihood estimation. To find the parameters, the maximum of the likelihood function, $l(\beta; \mathbf{Y})$, or the log-likelihood for simplicity,

$$\ln l(\beta; \mathbf{Y}) = \sum_{i=1}^{n} (y_i \mathbf{x_i}\beta - \ln(1 + e^{\mathbf{x_i}\beta})) \tag{4.4}$$

needs to be found by calculating the gradient of the log-likelihood with regard to the parameters, β. One can use the log-likelihood since the logarithmic operator does not affect the location of the maximum of the likelihood function, and it is easier to calculate. One arrives at the normal equations,

$$\mathbf{X'p} = \mathbf{X'Y} \tag{4.5}$$

which are nonlinear in β and thus needs to be solved by an iterative method, e.g. Newton-Raphson [Agresti, 2007][pp. 70-72;106], which is the method used in this thesis. SciKit-learn's implementation for Logistic Regression is used in this thesis, with default parameters except for the solving method and with increased maximum iterations to 500.

What can be noted from the Logistic Regression method, is that the method assumes linearity between the explanatory variables, along with each sample being assumed to be identically and independently distributed. Along with these assumptions, all the parameters need to be added manually, along with any interaction terms.

4.2 Decision Tree

Decision trees are the foundation for two other machine learning models presented in the coming sections, but they are also functional classifiers on their own. A simple tree which demonstrates the basic idea is presented in Figure 4.1.

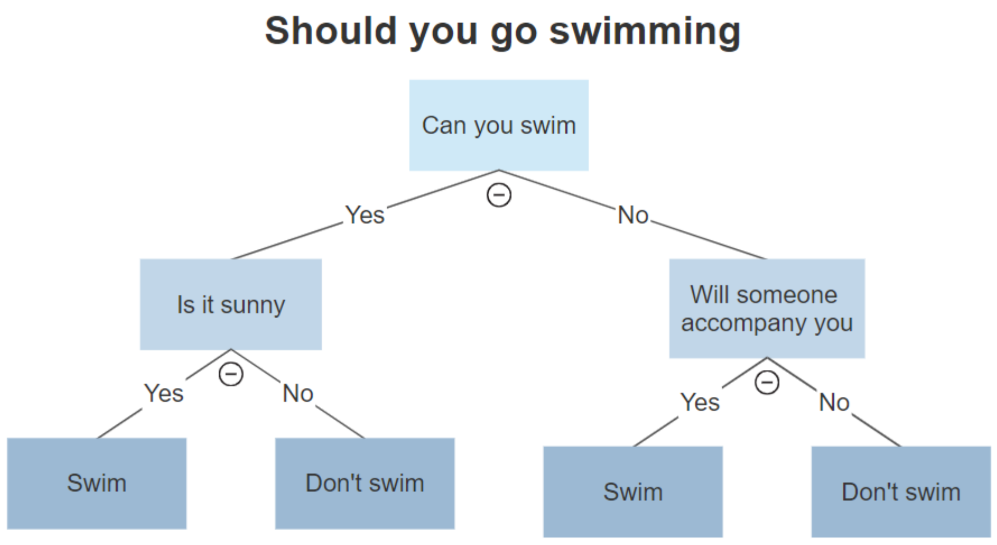


Figure 4.1: A simple decision tree

Decision trees have a high degree of interpretability. If one wishes to understand why an input produced a certain output, one would only have to follow the tree constructed. At every point where the tree is split, only a single feature with a certain condition is considered. Therefore, if the interpretability is highly sought after, simple decision trees can be a good choice. One issue with simple decision trees however is their tendency to produce high variance results, compared to their relatively low bias [Dietterich and Kong, 1995]. In almost all supervised machine learning algorithms, one has to properly balance the bias-variance trade-off. In general, the more complex a model is, the lower the bias and higher variance [Hastie et al., 2009][p.38].

When growing a decision tree, several rules can be used for splitting each node in the tree. In this thesis, information gain is maximised at each node, n, by minimizing each node's entropy, $L(n)$ [Hastie et al., 2009][p. 309], which is defined as

$$L(n) = -\sum_{k=0}^{K} \hat{p}_{nk} \ln(\hat{p}_{nk}) \tag{4.6}$$

where k is the classification outcome, between 0 and K. In this binary case, k can be either 0 or 1. $\hat{p}_{nk}$ is the proportion of class k in node n, representing a region R_n with N_n observations, defined as

$$\hat{p}_{nk} = \frac{1}{N_n} \sum_{x_i \in R_n} \mathbb{1}_{y_i=k} \tag{4.7}$$

where $\mathbb{1}_{y_i=k}$ is the indicator function

$$\mathbb{1}_{y_i=k} = \begin{cases} 1 & \text{if } y_i = k \\ 0 & \text{if } y_i \neq 0. \end{cases} \tag{4.8}$$

The regions in the tree are the space where all joint explanatory variables are partitioned into disjoint regions, $R_j, j = 1, 2, ..., J$ as represented by the terminal nodes, i.e. leaves, in the tree. A constant is assigned to each region, γ_j with the predictive rule

$$x \in R_j \Rightarrow f(x) = \gamma_j \tag{4.9}$$

where $f(x)$ in this case is an approximating function for the samples [Hastie et al., 2009][p. 353]. This function will be discussed more in detail in the chapter about boosting.

The tree is grown until a stopping criteria is met, e.g. until a fixed number of leaves in the tree are reached or a certain tree-depth is reached. Not growing the tree too much can be important to not overfit the model [Hastie et al., 2009][p. 308]. Trees can capture complex interactions in the data, including high-degree interactions [Hastie et al., 2009][p. 587].

4.3 Bootstrap aggregating classifier

In order to address the high variance of simple decision trees, one can utilize bootstrap aggregation, also commonly refered (bagging). Bagging is an ensemble method which trains multiple decision tree classifiers (other classifiers may be used as well, but in this work only decision trees were used). What is crucial is that every tree is trained on a bootstrapped subset of the available data. Each tree classifies the input and then "votes" on what class the input should be classified as, and the class with the most votes wins. This simple procedure generally leaves bias unchanged and decreases the variance [Hastie et al., 2009][pp.283-285].

A unique feature when using bagging is the use of *out-of-bag* (OOB) samples, which are the samples that do not correspond to the bootstrapped samples for the classifiers. For each sample, only the classifiers that have not seen the sample during the training can vote, and then the resulting error is calculated. The averaged error score is called the OOB error. For some classifiers, e.g. Random Forests, this error converges to the cross-valdiation error. Therefore, the OOB error can be used in place of cross-validation, increasing the speed at which the model can be validated [Hastie et al., 2009][p. 592].

4.4 Random Forests

In 1995 the author of a landmark article introduced the concept they called Random Decision Forests [Ho, 1995]. Its distinctive feature is to build decision trees on a random subset of the features available. This concept was further developed upon by combining the concept of bagging to create Random Forests (RF) [Breiman, 2001].

The individual learners created in the bagging process will be highly correlated, as they all operate in the same feature space. Breiman suggests that each individual classifier should not only be trained on a random subset of the data, but also a random subset of the available features [Breiman, 2001]. This results in a greater difference between all the de-correlated trees. This increases the bias and decreases the variance, but generally it leads to a better model overall [Géron, 2019].

The RF predictor creates B de-correlated trees, where each tree, $T(\Theta_i)$ uses a random subset of the available variables, Θ_i, and a random subset of the data, where $i = 1, ..., B$. To make a prediction at point x the predictor, $\hat{T}_{RF}(x)$, takes a majority vote among the B number of tree predictions [Robin and Jean-Michel, 2020][p. 34]

$$\hat{T}_{RF}(x) = \arg \max_{0 \le k \le K} \sum_{i=1}^{B} \mathbb{1}_{T(x,\Theta_i)=k}. \tag{4.10}$$

An example of a RF model can be seen in Figure 4.2.

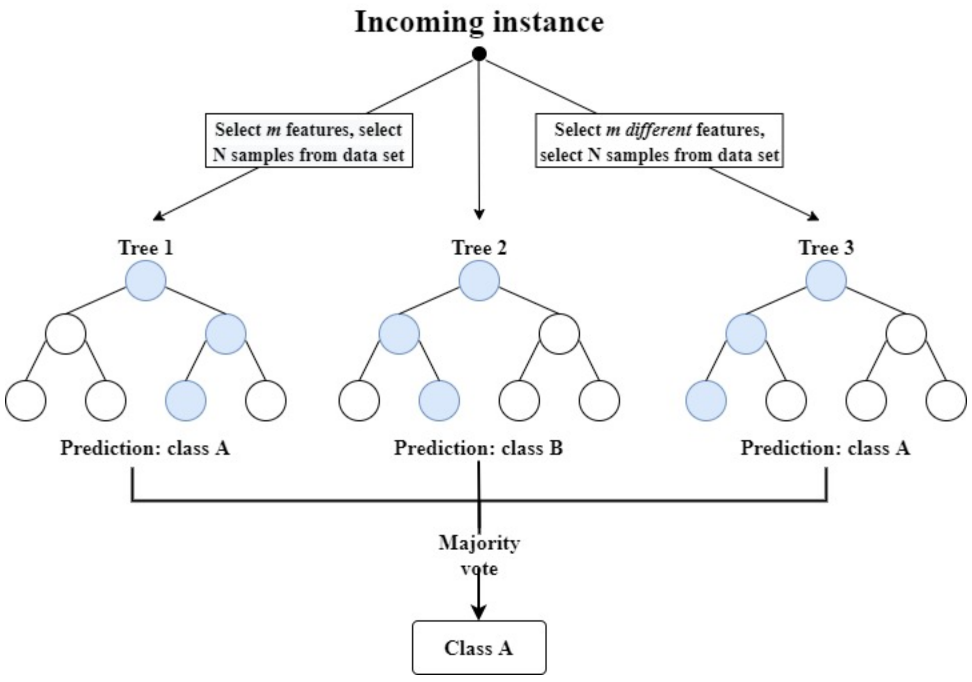

Incoming instance
Select *m* features, select N samples from data set
Select *m different* features, select N samples from data set
Tree 1
Tree 2
Tree 3
Prediction: class A
Prediction: class B
Prediction: class A
Majority vote
Class A

Figure 4.2: Example of a Random Forest using 3 trees, with a binary classification of class A or B.

The conditional class probabilities, i.e. the probability that a certain sample belongs to a class, can be approximated in a RF model by calculating the fraction of trees that vote for a certain class. This, however, need not have anything to do with a class probability, but usually tends to produce good probability estimates in practice [Olson and Wyner, 2018].

4.5 Gradient Boosting

A boosting algorithm is another ensemble method. It differs from RF as it grows trees sequentially. The idea is to create several weak learners, i.e. small decision trees, which combined forms a strong learner. The weak learner is constructed to be better than a random model, but not necessarily much better. Each tree tries to correct the mistakes of the preceding tree [Géron, 2019]. There are several different boosting algorithms, the one used in this work is known as Gradient Boosting Machine (GBM). The foundation for the algorithm was formulated in a landmark paper in 1997 [Breiman, 1997]. What sets GBM apart from its predecessor (Adaboost) is that each new tree is fitted to the residuals of the previous tree by using the gradient of its loss function. The structure of GBM is as follows:

An initial value, $f_0(x)$, is needed to start the algorithm, which is set to

$$f_0(x) = \arg\min_{\gamma} \sum_{i=1}^{N} L(y_i, \gamma) \tag{4.11}$$

where x is the data for samples x_i, $i = 1...N$. $y_i \in \{0, 1\}$ is the response and $L(y_i, \gamma)$ is the loss function, which is the cross-entropy loss as before, but for a GBM is defined as

$$L(y_i, p(x_i)) = -\sum_{k=1}^{K} \mathbb{1}_{y_i=k} \ln(p_k(x_i)) \tag{4.12}$$

for all number of classes, $k = 1...K$, which in this case are two. $p_k(x_i)$ is the probability for sample x_i to be class k, defined as

$$p_k(x_i) = \frac{1}{1 + e^{-f_k(x_i)}} \tag{4.13}$$

where $f_k(x)$ is a sum of weak learner trees, which approximates a function at each sample x_i. A new tree is fitted for each iteration, $m = 1...M$, using the pseudo-residuals as input for growing the tree, using the same procedure as a regular decision tree. The pseudo-residuals are defined as

$$r_{im} = -\left[\frac{\partial L(y_i, f(x_i)))}{\partial f(x_i)}\right]_{f=f_{m-1}} \tag{4.14}$$

which are calculated for each sample x_i, $i = 1...N$. The gradient is calculated with regard to the function from the last iteration, $f_{m-1}(x)$. The sum of the trees can be expressed as

$$f(x) = \sum_{m=1}^{M} T(x; \Theta) \tag{4.15}$$

where each tree, $T(x; \Theta)$, can be expressed as

$$T(x; \Theta) = \sum_{j=1}^{J} \gamma_j \mathbb{1}_{x \in R_j} \tag{4.16}$$

with parameters $\Theta = \{R_j, \gamma_j\}_1^J$ and meta-parameter J. After the tree is fitted a constant is calculated for each region for the current iteration, γ_{jm}, which is used as a weight when adding the trees together to make sure that the better learners that are trained have a more deciding vote in predicting than the worse learners. The constants are calculated as

$$\gamma_{jm} = \arg\min_{\gamma} \sum_{x_i \in R_{jm}} L(y_i, f_{m-1}(x_i) + \gamma) \tag{4.17}$$

for each terminal region $j = 1, ..., J$. The sum of the trees is then updated in each iteration by

$$f_m(x) = f_{m-1}(x) + \nu \sum_{j=1}^{J} \gamma_{jm} \mathbb{1}_{x \in R_{jm}} \tag{4.18}$$

where ν is a chosen learning rate, sometimes called *shrinkage rate*, which is meant to reduce overfitting [Hastie et al., 2009][p. 364]. The final model is then

$$\hat{f}(x) = f_M(x). \tag{4.19}$$

The number of iterations, i.e. number of trees, is a chosen parameter that corresponds to how long the model should be trained. Setting this parameter too low will result in an underfitted model, and setting it too high could result in an overfitted model. To reduce the risk of overfitting in this thesis, *early stopping* was used during training. The idea of early stopping is to continuously evaluate the predictive power of the model during training by testing the model on a validation set after a certain number of iterations. When the performance on the validation set stops improving, the training stops [Hastie et al., 2009][pp. 364-365].

There are many different open-source implementations of GBM available. SciKit learn's GBM and Xgboost are among the most commonly used implementations. In 2017, researchers at Microsoft published a paper where they introduced LightGBM. The researchers claims their implementation is able to achieve accuracies comparable to Xgboost, while being faster and more reliable when handling large data sets [Ke et al., 2017]. In this thesis, LightGBM is used.

4.6 Feature selection

In order to reduce the dimensions of the model, the need for a feature selection process was apparent. The total number of original features and created features combined were 36 before the categorical variables were one-hot encoded and 537 after (when not dropping one category). According to a study, a correctly implemented feature selection can improve the model in various aspects. The training time can decrease and the generalization capacity of the model can improve. Another improvement is that as the number of features decrease, the complexity of the model also decreases, making it more approachable for the salesperson who may use the results from the model [Bolón-Canedo et al., 2013].

Broadly speaking, feature selection algorithms can be categorized in three different main classes, based on if the selection is done before or during training: wrapper, filter and embedded [Blum and Langley, 1997]. A wrapper method, which selects the features by retraining several models, known as sequential backward elimination was used in this thesis. In this process a classifier is trained using all available features and then predicting on a validation set to achieve a score, in this case the F1-score which will be defined later, then the importance of each feature is calculated which ranks the features in importance scale. The least important feature is removed from the set and the process restarts with re-training the classifier with the new set of features.

There are several means of calculating the importance of variable, the most common for tree-based models being an impurity estimation, using the Gini-impurity or entropy of the grown tree. This measure however, is strongly biased towards high cardinality features, e.g. numerical features. To reduce bias, a permutation feature importance can be calculated instead, which is what is used in this thesis. Permutation feature importance is calculated for each feature, x_j, for $j = 1...m$, where m is the number of features in the feature set, X, by shuffling the feature vector to generate a corrupted data set, $\tilde{X}$. A score, s_j, which in this case is the F1-score, is computed on a validation part of this corrupted data and compared to a reference score, s, which is calculated before any shuffling of features is done. The shuffling and score computation is repeated K times to compute the importance i_j for feature x_j as [Altmann et al., 2010]

$$i_j = s - \frac{1}{K}\sum_{k=1}^{K} s_{k,j}. \tag{4.20}$$

The outline of the entire feature selection process is presented in Algorithm 1.

Algorithm 1: Sequential backward elimination with permutation feature importance

```
1  Start with the full set of all available features $X$
2  while $X \neq \emptyset$ do
3      Train a model on the feature set $X_{train}$
4      Compute score $s$ of the model $m$ on $X_{val}$
5      for $j = 1...m$ do
6          for $k = 1...K$ do
7              Randomly shuffle feature vector $x_j$ in $X_{val}$ to create the corrupted data set $\tilde{X}_{k,j,val}$
8              Compute the score $s_{k,j}$ of model $m$ on corrupted data $\tilde{X}_{k,j,val}$
9          Compute importance $i_j = s - \frac{1}{K}\sum_{k=1}^{K} s_{k,j}$
10     Remove feature $x$ with least importance from $X$
```

A few things are apparent in the sequential backward selection algorithm. The number of models trained is equal to the number of features present in the feature set minus one. The score function used can be any metric, in this thesis the $F1$-score was used for importance score but the OOB score and AUC score were also calculated. These metrics will be defined in detail in section 4.9.

Backward elimination is a time consuming process, but it has the advantage of taking the interaction of features into consideration when considering the worst feature. This scheme would not be feasible if the number of features were several magnitudes larger, or the model had longer training times. To reduce the computational effort and time, the ten least important features were removed in each iteration instead of removing just one feature. This is done until there are fewer than 50 features, then the only the least important feature is removed until the process is done.

4.7 Hyperparameter optimization

A commonality most machine learning models share is that there exist a set of hyperparameters which can be used to change the behaviour of the chosen model. The hyperparameter optimization problem is defined as [Bergstra and Bengio, 2012]

$$\begin{aligned} \lambda^{(*)} &\approx \underset{\lambda \in \Lambda}{\operatorname{argmin}} \operatorname{mean}_{x \in X^{(\text{valid})}} L\left(x; f_\lambda\left(X^{(\text{train })}\right)\right) \\ &\equiv \underset{\lambda \in \Lambda}{\operatorname{argmin}} \Psi(\lambda) \end{aligned} \tag{4.21}$$

where λ is the set of hyperparameters which will be tuned and Λ is the predefined search space. The second line in the equation above expresses the problem in terms of the hyperparameter response function Ψ. The central issue in hyperparameter optimization is that the characteristics of the Ψ function depends on the model and, more crucially, the given dataset. Therefore, the most common setting is that very little is know about the function. Therefore the most commonly used strategy is a *manual search* [Bergstra and Bengio, 2012].

Manual search is when one manually changes the parameters. This is a time-consuming endeavour, and it is difficult to efficiently test all the combinations available. A *grid search* can also be used, which is when one tries all possible combinations of hyperparameters. This is a very resource consuming approach, as the number of combinations may be large. It has been shown empirically however, that a *random search* is able to find parameters for the model which are as good as those found by a grid search in a fraction of the time [Bergstra and Bengio, 2012].

A random search tries only a fraction of the combinations. Since the computational effort is dramatically decreased, the domain where one searches for the optimal parameters may be expanded, further increasing the benefits of using random search. The strength of random search comes from the fact that it is common that some of the parameters that are being tuned have little to no effect on the performance of the model and that only a few of the parameters have a noticeable effect [Bergstra and Bengio, 2012].

As an illustrative example, consider the following optimization problem

$$f(x, y) = g(x) + h(y) \approx g(x). \tag{4.22}$$

The function $h(y)$ has little effect on the function f, therefore, it is of more interest to focus on $g(x)$. A grid search consisting of 9 trials would try three distinct values of x and three distinct values of y. A random search would try 9 distinct values for both x and y, and would therefore have a higher chance of finding the function values which optimize f [Bergstra and Bengio, 2012].

The chosen parameters can have a significant impact on the performance of the model [Claesen and De Moor, 2015]. While investigating the effects of hyperparameters, a study found that RF is relatively insensitive to the choice of hyperparameters, compared to other models. The same paper found that boosting algorithms have more to gain in hyperparameter optimization, but still not as much as other models such as artificial neural networks or support vector machines [Probst et al., 2018].

4.7.1 Hyperparameters for Random Forests

There are several parameters that can be tuned in a RF. In this thesis, the Scikit-learn implementation of RF is used and therefore the parameter variables are named thereafter. In order to decrease the search space, the results from an article which studies the tunability of the parameters were used [Probst et al., 2018]. The parameters chosen for tuning and optimization are presented below. The explanations of the variables are from the Scikit-learn documentation of the RF classifier [Pedregosa et al., 2011]. The parameters are presented in Table 4.1. The static features are not changed during the hyperparameter optimization, while the dynamic features are.

RF	Parameter	Starting value
Static	n_estimators	150/1000
Static	criterion	entropy
Dynamic	max_depth	None
Dynamic	max_features	$\sqrt{n\_features}$
Dynamic	max_samples	1

Table 4.1: The parameters which are used for Random Forest. The presented static values are the starting values when optimizing the parameters and calibrating the probabilities, for the other models all the presented values are what always were used. The starting value for the number of trees is 150 during feature selection, hyper parameter optimization and calibration, and 1000 during the final evaluation of the model. This was done in order to reduce the computational resources required.

The presented values in the table is also what is used for the feature selection process, for computational reasons.

The number of trees in the forest, *n_estimators*, should be set as high as computationally possible [Probst et al., 2018], as using more trees generally won't increase overfitting [Breiman, 2001]. However, it has been shown that even if the variance of the model decreases with the number of trees, the effect plateaus after a certain number of trees [Probst and Boulesteix, 2017] which means that one needs to balance the number of trees with performance and run time.

The maximum depth of the trees, *max_depth*, can potentially overfit a RF model since the model becomes too complex. This however seems to be a very small effect and especially small for classification forests [Hastie et al., 2009][p. 596].

The number of random features, *max_features*, to be considered in each tree has been shown to be the most important feature in a RF to tune [Probst et al., 2018]. A small amount of features could result in several trees training on irrelevant features, making the entire model perform worse. However, if a large amount of features are used the result can become better but the trees will become more correlated, diminishing the fundamental gain of a RF. The inventors of the RF recommend that the default value for classification forests should be

$$max\_features = \sqrt{\text{Total number of features}} \tag{4.23}$$

but this depends on the data and the problem at hand [Hastie et al., 2009][p. 592]. If many of one-hot encoded categorical features are used, each with many categories, there is a risk that this recommendation is too generous and only irrelevant features will be considered for several trees. One way to counter this effect is to take the percentage of the total number of features that corresponds to Equation 4.23 before one-hot encoding the categorical features. If $max\_features$ is equal to 100%, the model will be the same as a bagging classifier, if the data is bootstrapped as described below.

The number of samples used for each tree, *max_samples*, depends on if bootstrapping is used or not. If bootstrapping is not used, the entire data set is used for each tree and the only stochastic effect in RF will be from the random features chosen. If bootstrapping is used, each sample is drawn with replacement, and one can decide what percentage of of the data set size should be bootstrapped. If a smaller data set is used for each tree, the trees will be even more uncorrelated than with just using random features.

It has recently been shown that even if RF generally are insensitive to hyperparameters for classification tasks compared to other methods, the prediction of class probabilities are more sensitive to the hyperparameters. Specifically, the $max\_features$ parameter has the greatest effect for this optimization [Olson and Wyner, 2018].

The scheme used in order to perform the hyperparameter optimization for the RF model is described in 2.

Algorithm 2: Hyperparameter optimization scheme for Random Forest

```
1 n is the number of random searches
2 for $j = 1...n$ do
3     Draw a random set of parameters, $\lambda$, from the predefined search space $\Lambda$.
4     Construct a model according to $f_\lambda\left(X^{(\text{train})}\right)$
5     Calculate the OOB score of $f$
6 end
7 Choose the set of parameters which yielded the best OOB score
```

4.7.2 Hyperparameters for Gradient Boosting Machines

The chosen parameters which will be optimized are a combination of the features suggested in the previously mentioned study, and recommendations from the LightGBM documentation. The parameters used are presented in Table 4.2. The static features are not changed during the hyperparameter optimization, while the dynamic features are.

GBM	Parameter	Starting value
Static	n_iterations	100000
Static	n_iter_no_change	1000
Static	metric_types	binary logloss
Dynamic	max_depth	-1
Dynamic	bagging_freq	0
Dynamic	bagging_frac	N/A
Dynamic	feature_frac	1
Dynamic	num_leaves	31
Dynamic	learning_rate	0.1

Table 4.2: The parameters which are used for GBM. The presented static values are the starting values when optimizing the parameters and calibrating the probabilities, for the other models all the presented values are what were used.

The learning rate, *learning_rate*, is also known as the shrinkage rate, denoted ν in Equation 4.18. The value of this parameter adjusts how much correction should occur after each subsequent tree. A small learning rate will increase the time it takes to reach convergence but a too large learning rate may not converge at all [Murphy, 2012].

Bagging fraction, *bagging_frac*, and **bagging frequency**, *bagging_freq*, introduce the concept of bagging to GBM. Every tree is trained only on a subset of the data. This introduces additional randomness into the model, which can help prevent overfitting. It also has the advantage of decreasing training times, as trees are fitted to less of the available data. The frequency parameter determines how often bagging should occur and the fraction determines how many samples the tree should be trained on [Ganjisaffar et al., 2011].

Feature fraction, *feature_frac*, takes the concept of subsampling the available features, in the same way as the RF architecture. Every tree is trained on only a subset of the available features [Ke et al., 2017].

The number of trees, *n_iterations*, is how many subsequent trees the model should construct. A large number of trees will increase the training time, and may lead to overfitting. The risk of overfitting is counteracted by routinely monitoring the error on a separate validation data set (that is not present in the training data) and if building additional trees does not yield any improvement for a certain number of trees, the training stops. The number of trees which can be built without any improvement before the training stops is controlled by the **early stopping rounds**, *n_iter_no_change*, and the metric used is controlled by the **metric**, *metric_types*, parameter [Ke et al., 2017].

The max depth, *max_depth*, and **The number of leaves**, *num_leaves*, are the two main parameters used to control how complex the trees can grow. Complex trees are more capable of capturing complex structures in the data, but are also more prone to overfitting [Ke et al., 2017]. The default value of -1 indicates that there is no limit.

The scheme used in order to perform the hyperparameter optimization for the GBM model is described in Algorithm 3. This scheme is a bit more intricate than the one used for RF. There are two reasons for this. The first is that the OOB score for a GBM is not defined if bagging is not enabled, therefore a validation set val_2 is needed to assess the model. The second reason is that the GBM model will utilize early stopping, and for that a separate validation set, val_1, is needed.

Algorithm 3: Hyperparameter optimization scheme for GBM

```
1 n is the number of random searches
2 for $j = 1...n$ do
3     Draw a random set of parameters, $\lambda$, from the predefined search space $\Lambda$.
4     Construct a model with early stopping according to $f_\lambda\left(X^{(train,val_1)}\right)$, training stops when
        the improvement of performance on the $X^{val_1}$ set stops for a certain number of iterations.
5     Calculate the F1-score of the model on $X^{val_2}$
6 end
7 Choose the set of parameters which yielded the best F1-score
```

When comparing Algorithm 2 and Algorithm 3 it is noticeable that the RF scheme needs no separate validation set in order to assess the generalization error of the model, while the GBM scheme needs two disjoint validation sets.

4.8 Probability calibration

Even though the issue at hand is a binary classification problem, it is of interest to know how certain the model is in its predictions. Both RF and GBM are able to assign generated probabilities to the two classes. What is of interest is to know how credible these probabilities are. As an example, if the model sees 100 samples, and they are all assigned a probability of acceptance in the 80-90% range, the observed hit ratio should be (approximately) equal to 85% for these samples [Reich and Ghosh, 2019][p.2]. This frequentistic perspective is also formulated in Equation 4.24 [Kull et al., 2017]. If a probabilistic model's output exhibits this behaviour, it is said to be *calibrated* [Vaicenavicius et al., 2019].

$$\mathbb{E}\left[Y \mid f(X) = s_i\right] = \frac{\sum_{j=1}^{n} y_j \cdot \mathbb{1}_{f(x_j)=s_i}}{\sum_{j=1}^{n} \mathbb{1}_{f(x_j)=s_i}} \tag{4.24}$$

This should be true for the continuous $[0, 1]$. However, since there are a finite number of samples, it is useful to discretize the range into bins. The behaviour can then be examined for each of the bins. This can be visualized using a reliability diagram [Murphy and Winkler, 1977]. How to create a reliability diagram is outlined in Algorithm 4.

Algorithm 4: How to create a reliabiliy diagram

```
1 Let $X$ be a set of the data yet unseen by the model $f$.
2 Sort $\mathbb{P}(y = 1|X, f)$ in ascending order.
3 Bin the elements in $n$ bins
4 for $j = 1...n$ do
5     $\bar{p_j}$ = Predicted fraction of positives in bin $j$
6     $\bar{q_j}$ = True fraction of positives in bin $j$
7 end
8 Plot $\bar{q}$ as a function of $\bar{p}$ as
```

An example of a calibration curve can be seen in Figure 4.3, using an example data set.

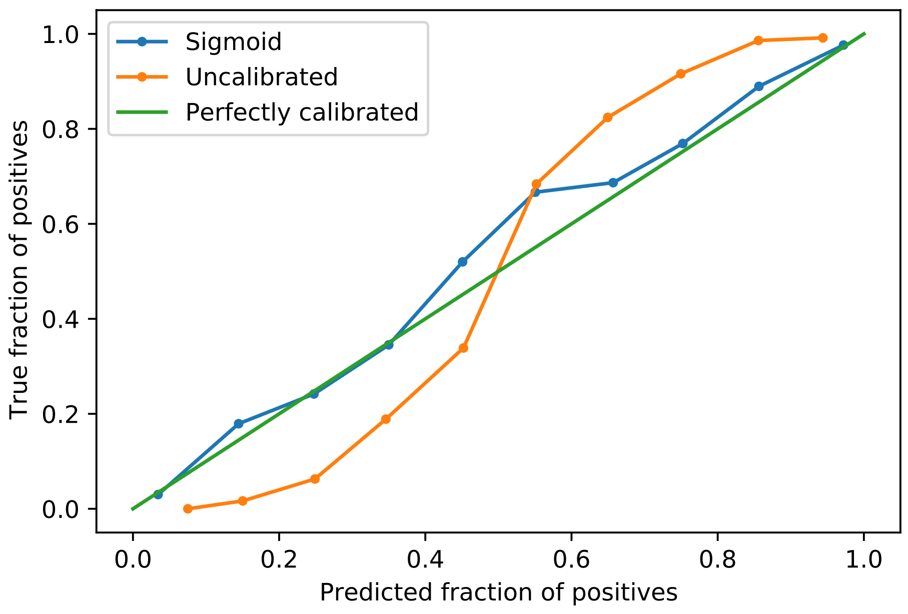


Figure 4.3: Reliability curves for the uncalibrated (orange) and the calibrated probabilities (blue). The data used was synthetically created. The calibration was done by using Sigmoid calibration. A perfectly calibrated classifier would have the shape of the green line.

In the figure above, a RF classifier was fitted to some synthetically created data. The trained model then produces probabilities for a validation set. The orange line represents the yet uncalibrated probabilities. In its fourth bin, it can be seen that the predicted fraction of positive is 38%. In reality, the true fraction of positives is 20%. On the flip-side, in the seventh bin, the predicted fraction of positives is 62%, while the true fraction of positives is 80%. The green line indicates a perfectly calibrated classifier. The blue represents a RF model but after it has been calibrated using Sigmoid calibration. It can be seen that the blue line is more aligned with the green line, hence the calibration was successful in creating more reliable probabilities.

A calibrated model does not implicitly mean that the model is useful in any way. For an example, consider a binary classification problem where the two classes have equal prevalence. A model which assigns each samples a 0.5 probability will be perfectly calibrated. When evaluating a model, it is therefore of importance to consider multiple metrics (these metrics used will be presented in the next section).

The first calibration method used was Sigmoid calibration, also known as Platt scaling. It was introduced in 1999 in the context of support vector machines, but has since then been used liberally for other classifiers as well. The process is simply to fit a Logistic Regression model to the probabilities generated by a model. In the original paper, the author suggests to use one set of data to train the actual model, and another data set to calibrate the model. This method is best suited if the log-odds of the positive response variable are linear [Platt, 1999].

The second calibration method used was isotonic regression which was introduced in 2001 [Zadrozny and Elkan, 2001]. Isotonic regression's basic assumption is that given model probabilities $f(\mathbf{X})$ and responses $\mathbf{Y}$,

$$\mathbf{Y} = m(f(\mathbf{X})) + \xi \tag{4.25}$$

where m is a non-decreasing function [Niculescu-Mizil and Caruana, 2005], and $\xi \sim N(0,1)$ is a Gaussian white noise. To find the optimal isotonic function $\hat{m}$ the optimization problem

$$\hat{m} = \arg\min_{z} \sum (y_i - z(f(x_i))^2) \tag{4.26}$$

is solved. As Equation 4.25 suggests, isotonic regression is able to correct monotonic distortion only. It is more powerful than Sigmoid calibration, but it has been showed that isotonic regression is more prone to overfitting [Niculescu-Mizil and Caruana, 2005].

The third calibration method used is Beta calibration [Kull et al., 2017]. The method is based on the likelihood ratios of two β-distributions, and is defined as

$$f(x_i) = \frac{1}{1 + e^{-m\frac{(1-x_i)^b}{x_i^a}}}, \qquad \{a, b, m\} \in \mathbb{R}. \tag{4.27}$$

The authors suggest that this calibration will at worst perform the same as logistic calibration, and in most cases better. One advantage of the Beta calibration when compared to the Sigmoid calibration is that the identity function is not part of the logistic family, but it is part of the Beta family [Kull et al., 2017].

4.9 Evaluation metrics

In order to asses the results from the classifiers, a number of different metrics are available for use. The most straightforward metric is the accuracy, defined as

$$\text{Accuracy} = \frac{\text{True positives} + \text{True negatives}}{\text{Total samples}}. \tag{4.28}$$

In this thesis, the true positives are the won samples that are correctly classified as won, the false positives are lost samples that are classified as won, the true negatives are the lost samples that are classified as lost and the false negatives are the won samples that are classified as lost.

An issue with accuracy is that a random classifier would get 50% accuracy in a binary classification. Another issue arises if there is significant imbalance in the data. If one of the classes is more prevalent than the other class, a classifier which simply always classifies the input as the first class, would get a high accuracy as well. As previously mentioned, the dataset used in this work is not significantly imbalanced.

There are two metrics which can be used to compliment accuracy; precision and recall. Precision is defined as

$$\text{Precision} = \frac{\text{True positives}}{\text{True positives} + \text{False positives}} \tag{4.29}$$

and recall is defined as

$$\text{Recall} = \frac{\text{True positives}}{\text{True Positives} + \text{False negatives}}. \tag{4.30}$$

To combine the two metrics, one can use the F1-score. It is defined as [Nugues, 2014]

$$F1\text{-score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{4.31}$$

The definition of the false positive rate (FPR) can be seen in

$$\text{False Positive Rate} = \frac{\text{False Positives}}{\text{True Positives} + \text{False Positives}} \tag{4.32}$$

and the true positive rate (TPR) is defined as

$$\text{True Positive Rate} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}. \tag{4.33}$$

The problem considered is a binary classification problem, either a quote line is won or lost (one or zero). However, both of the models used, GBM and RF, produce weights for both labels. Let $f(x)$ be a measure of how confident the model is that the sample x belongs to class 1, where f is one of the models. To get a classification from this weight, one can define a decision rule as

$$\mathbb{1}_{f(x)>\tau} \qquad , \tau \in [0, 1] \tag{4.34}$$

where τ is a threshold value. The threshold value will obviously affect the metrics defined above. A high threshold value will cause a model to classify most samples as negative, and a low threshold will cause most samples to be classified as positive. How the FPR and TPR are affected by the threshold value can be visualized using a receiver operating characteristic (ROC) curve [Murphy, 2012][p.181]. An example of such a curve is presented in Figure 4.4.

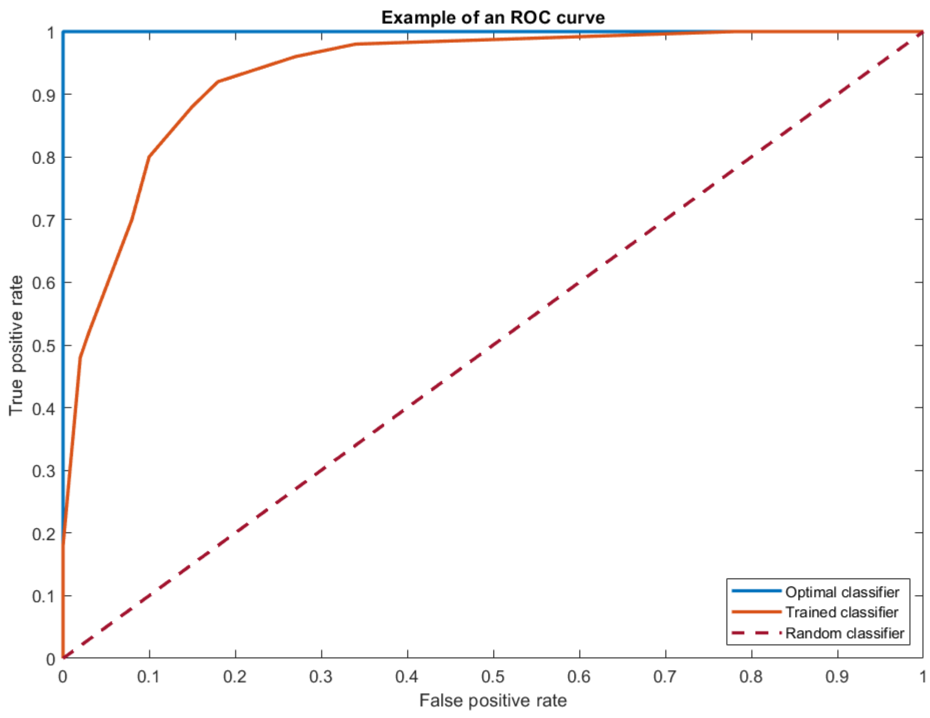


Figure 4.4: The FPR's and TPR's of an optimal classifier, a trained example classifier and a random classifier for different threshold values. The data used was synthetically created.

In order interpret the ROC curve, the area under the curve (AUC) is a measure to evaluate the curve. The optimal classifier has an AUC value of 1. In this case, when using synthetically created data, the trained example classifier has an AUC value of 0.93 and the random classifier 0.5.

To evaluate the probabilities, two metrics will be used along with an additional graphical evaluation. The first is the Brier score, defined as [Brier, 1950]

$$\text{Brier score} = \frac{1}{N}\sum_{n=1}^{N}(f_n - \mathbb{1}_{y_n=1})^2. \tag{4.35}$$

As can be seen in the definition, Brier score does not take the actual classification of the model into consideration (hence, it is independent of the threshold value τ previously discussed).

The second metric which relates to the probabilities is the reliability diagram. For a perfectly calibrated classifier the predicted fraction of positives and the true fraction of positives should align, forming a straight line. The squared vertical distance from the line to the actual position of the bin in the reliability diagram is a measure of how far off the model's probabilities are. One issue with only looking at this distance is that it does not take into account how many samples there are in each bin. To take this into consideration, the *weighted distance* is defined in Equation 4.36.

$$\sum_{j=1}^{n} c_j|\bar{p_j} - \bar{q_j}| \tag{4.36}$$

where $\bar{p_j}$ and $\bar{q_j}$ is the predicted and true fractions of positives in the j:th bin, and c_j is the number of samples in the bin, and n is the total number of bins.

A unique feature for this classification problem is that there is a certain behaviour that is expected from the probabilities. As the price increases for a sample, the probability of acceptance should, according to domain knowledge, decrease. To validate that the constructed models do exhibit a sound behaviour, a test will be conducted.

The test consists of altering the prices of the samples in the test set from a low price to a high price. Expert knowledge will be utilized to determine what the low and high prices are, but will be set in a manner where the experts deem the low price will yield a high probability of acceptance, and the high price will yield a low probability of acceptance. The constructed model will then predict the probability of acceptance for prices ranging from low to high. The average probability of acceptance across all samples in the test will then be calculated, along with the variance of the predicted probabilities. This will be repeated for increased price to obtain new values, to be able to plot all averages and variances as a function of price. An overview of this process to create a graph showing the probability as a function price can be seen in Algorithm 5.

Algorithm 5: How to create a graph showing probability of acceptance as a function of net price

```
1  Train a model on the train set $X_{train}$
2  Predict on a validation set $X_{test}$
3  Change all net prices in $X_{test}$ to a very low price
4  for $j = 1...n$ do
5      Calculate the probability of acceptance for all all samples in the altered $X_{test}$
6      Calculate the average acceptance of probability from the predicted probabilities
7      Calculate the variance from the predicted probabilities
8      Save the all values for plotting
9      In $X_{test}$, increase the price for all samples
10 end
11 Plot all average probabilities as a function of net prices
12 Plot all variances as a function of net prices
```

This test will be done for all samples and for the samples that are won and also correctly classified as won in the unaltered data set by the chosen model.

The graphical test will only be an indication if the probability will decrease or not, and how fast, if the prices increase. This will not indicate that all samples will have the same behaviour, as there could be a variance across the samples where the probability will not have the same behaviour for probability increase/decrease for single samples.

4.10 Combining classifiers

As this work will utilize a few different classification models, the concept of combining classifiers will be attempted. The idea is to use two or more trained classifiers, let them construct predicted probabilities on a test set, and then average the probabilities in order to generate a new prediction. This is the same as ensemble models, even if the models in themselves might be ensemble models. The idea is that if the two classifiers are well calibrated, their predictions will correctly reflect how certain they are. Then, if one classifier is uncertain in one case, and the other classifier is certain, the combined prediction will be that of the certain classifier.

Chapter 5

Results

In this section, the results from the binary classification task, feature selection process, hyperparameter optimization and probability calibration are presented. All results were produced using the same machine. Therefore, the training times presented are comparable across models.

The results for each classifier are first presented separately and later compared to each other. A combined, ensemble classifier using Random Forest (RF) and Gradient Boosting Machine (GBM) is also tested and presented. Results from several different models for each classifier will be presented, and will be referred to as follows:

1. **Standard:** Model trained with standard features obtained from the original data set, according to Table 3.1.
2. **Engineered:** Model trained with standard features and engineered features as described in section 3.2.
3. **Selected:** Model trained with features selected from the feature selection process, as described in section 4.6.
4. **Optimized:** Model trained with selected features and optimized hyperparameters selected from the hyperparameter optimization process, as described in section 4.7.
5. **Calibrated:** Model trained with selected features and optimized hyperparameters, and calibrated probabilities with the method that was the best method according to the calibration evaluation process, as described in section 4.8.

The results for Logistic Regression will only be presented for the first two types of models, as the method requires no calibration. The method also showed little promise compared to the other models, so no feature selection or hyperparameter optimization was done due to limited time.

No feature selection was done for the GBM models due to limited time, so an assumption was made that the selected features for the RF models would be roughly equally important for the GBM models.

5.1 Logistic regression

The ROC curve for the two models using Logistic Regression can be seen in Figure 5.1. The accuracy, F1-score, AUC score, Brier score and training time in seconds can be seen in Table 5.1. The numbers in bold show the best results for each respective category.

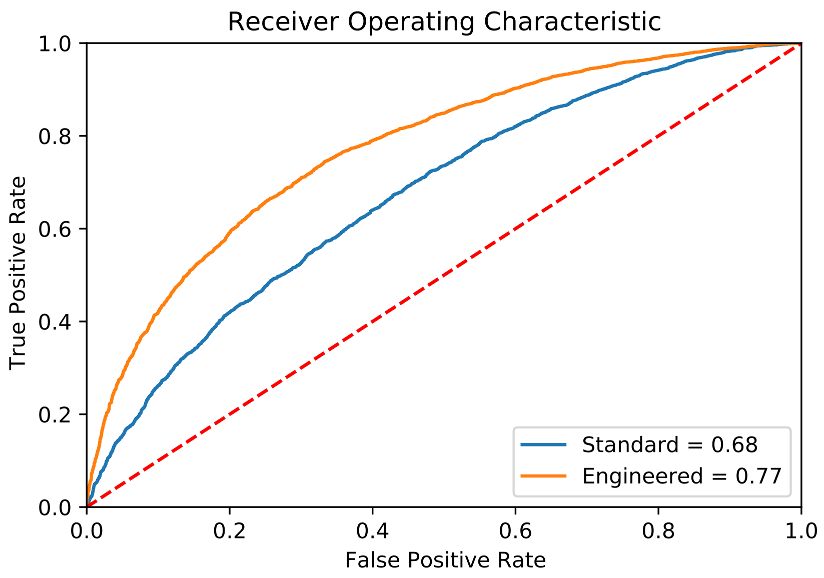
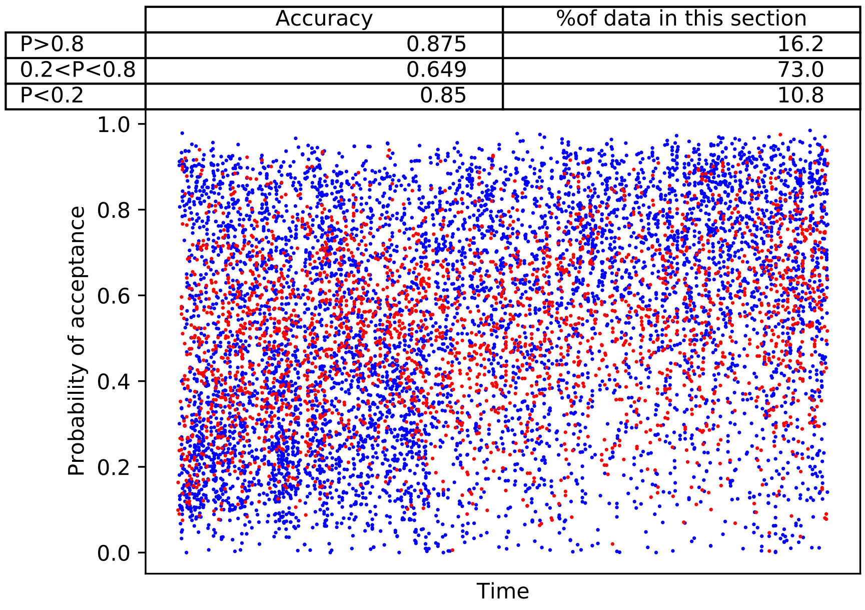


Figure 5.1: ROC curves for the two models using Logistic Regression, one using standard features and one using engineered features. The AUC score is shown in the bottom right corner for the models. The red dashed line shows the ROC curve for a random choice classifier.

Model	**Accuracy**	**F1-score**	**AUC**	**Brier score**	**Training time (s)**
Standard	0.627	0.671	0.675	0.225	**216**
Engineered	**0.707**	**0.732**	**0.774**	**0.193**	972

Table 5.1: Results for the Logistic Regression models showing accuracy, F1-score, AUC score, training time in seconds, and Brier score. The numbers in bold are the best scores in each respective category.

A scatter plot showing the predicted probability of acceptance of each sample in the test set can be seen in Figure 5.2. The blue dots display the correctly classified samples and the red dots display the misclassified samples. The table in the figure shows the accuracy of three sections of the figure, $\mathbb{P}(X) > 0.8$, $\mathbb{P}(X) < 0.2$ and $0.2 < \mathbb{P}(X) < 0.8$, where X is each sample.



	Accuracy	%of data in this section
P>0.8	0.875	16.2
0.2<P<0.8	0.649	73.0
P<0.2	0.85	10.8

Figure 5.2: Scatter plot showing the predicted probability of acceptance of each sample in the test set, using the Logistic Regression model with engineered features. The blue dots display the correctly classified samples and the red dots display the misclassified samples. The accuracy for each section can be seen in the table at the top of the image, along with the percentage of the data set that is in that section.

As can be seen from the figure, most of the misclassified samples lie close to 50% in probability, meaning that the samples that lie closer to 0% or 100% of probability of acceptance are more accurate in prediction of class.

The final Logistic Regression was used to predict the average probability of acceptance when changing the net price of the samples to see if a price optimization would be possible to implement directly on the model. This was done by changing the price successively from very cheap to very expensive, for all samples, and predict the average probability of acceptance. This procedure was used first for all samples, and for the samples that were won and were correctly classified as won. The results can be seen in Figure 5.3, along with the variances in Figure 5.4. The figures also shows a grey histogram of the distribution of the prices in the test data. The averaged probabilities between the two red lines can therefore be compared to how many samples have prices in the same range.

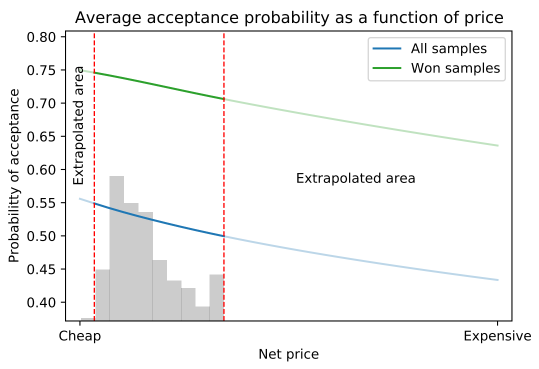


Figure 5.3: Predicted probability of acceptance as a function of price, using the Logistic Regression model with engineered features, for all samples and for the samples that were won and correctly classified as won. The values that are not between the red-dashed lines are extrapolated values, as the data contains no prices in those areas. The grey histogram shows the price distribution of the samples in the unaltered test set.

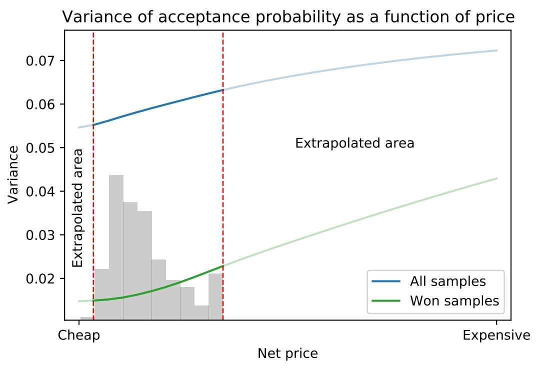


Figure 5.4: Variance of average predicted probabilities of acceptance as a function of price, using the Logistic Regression model with engineered features, for all samples and for the samples that were won and correctly classified as won. The values that are not between the red-dashed lines are extrapolated values, as the data contains no prices in those areas. The grey histogram shows the price distribution of the samples in the unaltered test set.

From Figure 5.3 it can be noticed that the probability of acceptance decreases with increased price, and the average predicted probability for all samples lie closer to 50% as the samples that are predicted as lost are also included. It can also be noticed that the variance increases with decreased probability, seen in Figure 5.4. As can be seen, the extrapolated area shows predicted probabilities for prices that are not present at all in the data.

5.2 Random Forest

Several variants of RF models were tested. First, the results from the feature selection process described in section 4.6 are presented. The features that were chosen were then used in a hyperparameter optimization, described in section section 4.7. Then, the probabilities are calibrated using the methods described in section 4.8 and the method that obtained the best result is used to finally compare the models for all steps of the process. The test set was only used to obtain the results for the final model, with what is presented as the final results. For the other results, the validation set is used instead.

5.2.1 Feature selection

To find the features that were most important, a feature selection process was initiated using Algorithm 1 as described in section 4.6. In each iteration the features' importances are calculated through permutation importance, to rank the features by importance. To increase the speed, the 10 least important features in each iteration were removed until only 50 features remained. At that point the least important feature was removed individually at each iteration.

As one may recall, each category for each categorical feature is regarded as a separate feature. The results from the feature selection process can be seen in Figure 5.5. The red dashed line represents where the maximum score was reached, and the black dashed line represents where the feature selection began selecting each feature individually.

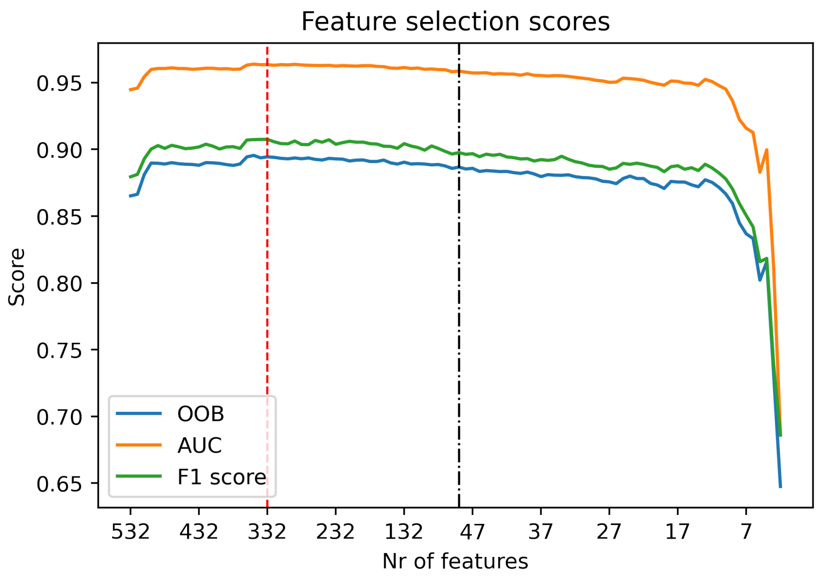


Figure 5.5: Result from feature selection, using Algorithm 1. The green line represents the F1 score, the orange line represents the AUC score and the blue line represents the out-of-bag (OOB) score. The ten worst features were removed at each iteration, up until 50 features were left when only one feature was removed at each iteration. The black dashed line represents where this change was done. The red dashed line shows where the highest score, in all aspects, was reached.

As can be seen from the figure, the best score was reached for about 332 features and for simplicity, these features were used for the model with selected features, and these features are used for optimization and calibration.

By measuring the permutation feature importance for each feature at each iteration, an overview of which features are the most important can be made. Even if price is an important aspect when making a quote and is an important feature, it is not the most important feature in this feature selection process. Other features connected to the customer, product or other transactional data seemed to be more important.

5.2.2 Hyperparameter optimization

To find the optimal hyperparameters for the RF model, a random search was initiated as described in section 4.7. The defined search space can be found in Table A.1. The OOB score of the different models can be seen in Figure 5.6.

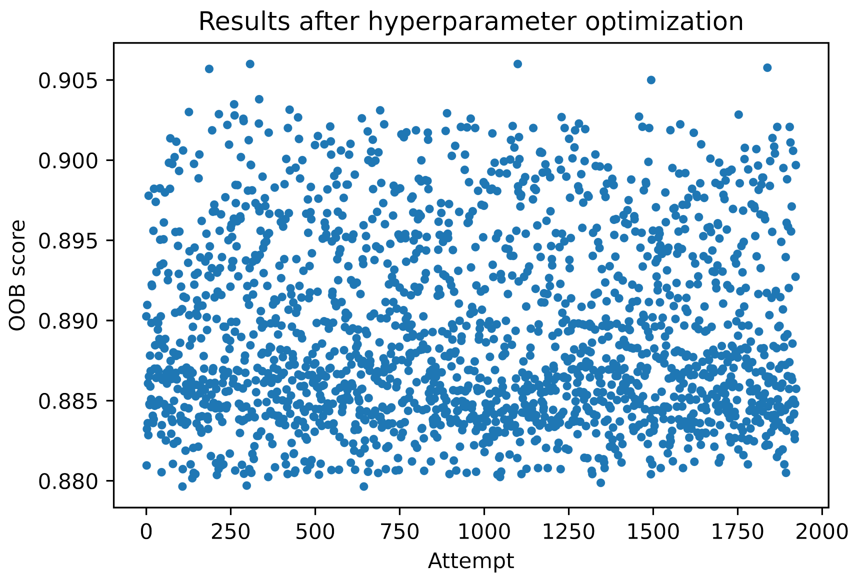


Figure 5.6: The OOB score of the 2000 models trained during the random hyperparameter search.

In Figure 5.6 it can be seen that the OOB score of the models trained differ from 88% to 90.5%.

The optimal parameters that were found are presented in Table 5.2.

Parameter	Optimal value
max_depth	'None'
max_features	0.01
max_samples	0.98

Table 5.2: The optimal parameters for the RF model, found using a random search process.

5.2.3 Probability calibration

The reliability diagrams for Sigmoid, Isotonic and Beta calibration are presented in Figure 5.7 along with the uncalibrated classifier.

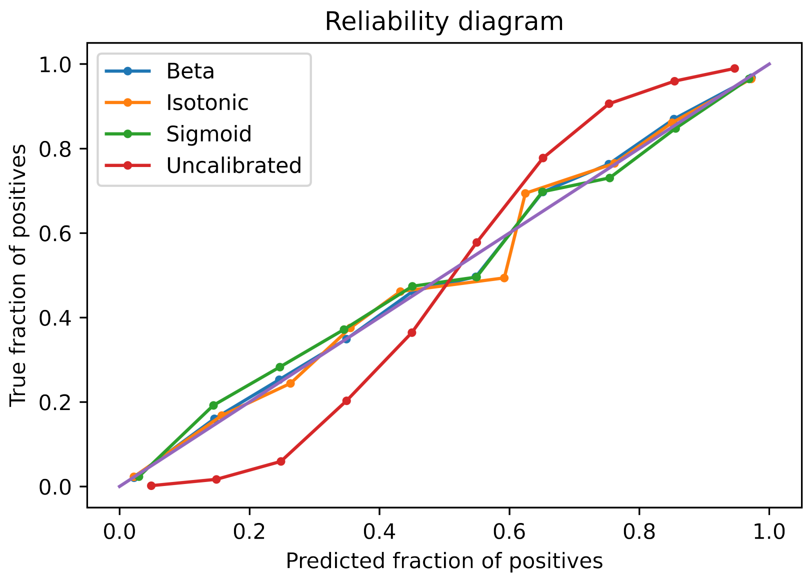


Figure 5.7: Reliability diagram of the three different calibration methods along with the uncalibrated classifier.

The uncalibrated probabilities are clearly not particularly reliable. In the third bin, the model predicts the fraction of positives is 22%, and the true fraction of positives is 5%. The three classification methods have all been able to calibrate the model. In order to determine which of the three methods is the most optimal, more precise data are presented in Table 5.3.

	Beta		**Isotonic**		**Sigmoid**		**Uncalibrated**	
	% of data	Distance	% of data	Distance	% of data	Distance	% of data	Distance
0.0 <P <0.1	30.7	0.017	31.4	0.008	32.1	0.064	11.8	0.468
0.1 <P <0.2	5.9	0.145	5	0.105	5.9	0.476	10.8	1.322
0.2 <P <0.3	3.7	0.07	4	0.19	3.5	0.36	9.2	1.89
0.3 <P <0.4	3.2	0.003	4.1	0.203	2.7	0.261	7.5	1.465
0.4 <P <0.5	2.7	0.107	3.2	0.294	2.5	0.233	6.2	0.857
0.5 <P <0.6	2.9	0.52	0.8	0.985	2.5	0.539	6.2	0.279
0.6 <P <0.7	3.3	0.456	3	0.694	2.7	0.468	7.8	1.258
0.7 <P <0.8	4.5	0.102	6.7	0.029	4	0.239	9.6	1.525
0.8 <P <0.9	7.1	0.165	3.9	0.106	6.5	0.081	14.6	1.052
0.9 <P <1.0	35.9	0.049	37.7	0.072	37.8	0.04	16.4	0.428
Weighted distance	**77.88**		89.46		120.6		951.46	
Brier score	**0.7652**		0.767		0.7668		0.8999	

Table 5.3: Numerical data of the calibration results on the validation set for three different calibration methods and the uncalibrated data.

In Table 5.3, in the column showing the percentage of the data that the bin contains, it can be seen that all three calibration methods have the major part of the samples in the first and last bin, while the uncalibrated probabilities are more evenly distributed across the bins. The weighted distance and the Brier score have all also been improved by the calibration methods, the largest improvement of both metrics was achieved by Beta calibration. Therefore, Beta calibration was chosen as the choice of calibration methods for RF.

5.2.4 Final results for Random Forest

The ROC curves for the five final different RF models can be seen in figure 5.8. The accuracy, F1-score, AUC score, training time in seconds and the Brier score can be seen in Table 5.4 for the five different models.

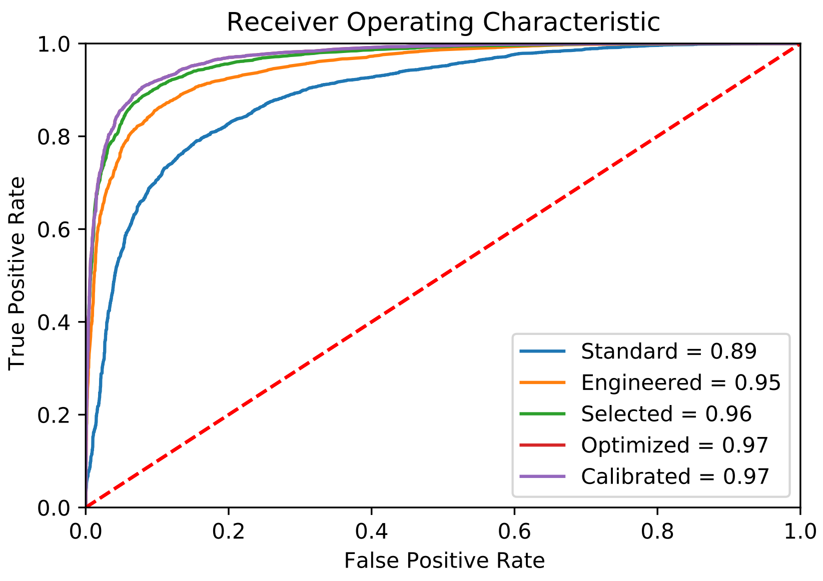


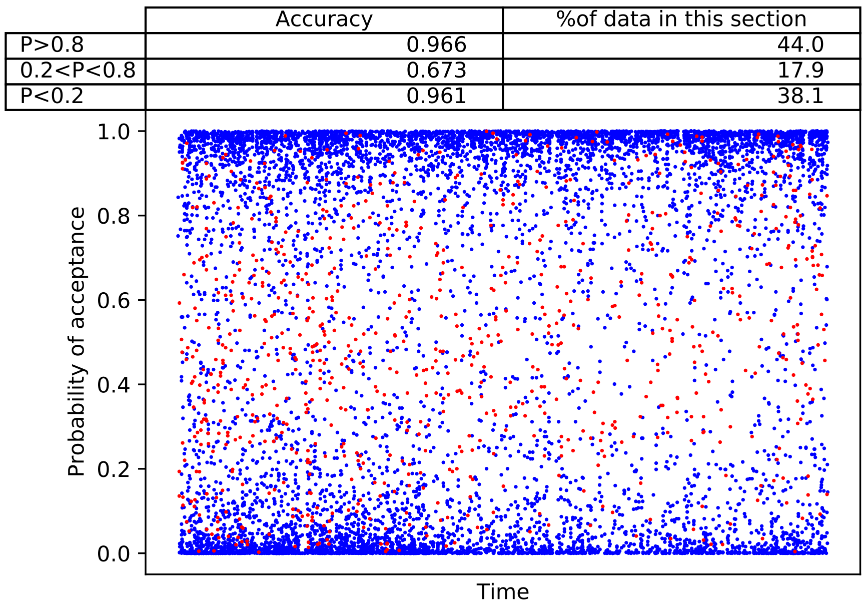
Figure 5.8: ROC curves for the five different RF models; using the standard features from the original data set, including the engineered featuers, using the selected features, using the optimized hyperparameters and using Beta-calibration for the probabilities. The model with the optimized hyperparameters and with the calibrated probabilities achieved the same ROC curve and same AUC score. All AUC scores can be seen in the bottom right corner. The red dashed line represents a random choice classifier.

Model	Accuracy	F1-score	Training time (s)	AUC	Brier score	Weighted distance
Standard	0.808	0.827	89.3799	0.889	0.136	470.33
Engineered	0.876	0.884	70.2242	0.945	0.105	1056.2
Selected	0.903	0.909	47.6926	0.964	0.085	943.78
Optimized	**0.912**	0.917	**30.0526**	**0.969**	0.08	961.18
Calibrated	0.912	**0.918**	30.7781	**0.969**	**0.066**	**114.85**

Table 5.4: Results for the RF models showing accuracy, F1-score, AUC score, training time in seconds, Brier score and the weighted distance. The numbers in bold are the best scores in each respective category.

From the results it can be noticed that the best RF in terms of accuracy and training time was the uncalibrated model with optimized hyperparameters. In all other aspects the calibrated model performed best, and is the model which is chosen to be the best RF model to compare with the other models. The AUC score was identical for the Optimized model and Calibrated model, which is the reason that both values are in bold.

A scatter plot showing the predicted probability of acceptance of each sample in the test set, using the calibrated RF model, can be seen in Figure 5.9. The blue dots display the correctly classified samples and the red dots display the misclassified samples. The table in the figure shows the accuracy of three sections of the figure, $\mathbb{P}(X) > 0.8$, $\mathbb{P}(X) < 0.2$ and $0.2 < \mathbb{P}(X) < 0.8$, where X is each sample.



	Accuracy	%of data in this section
P>0.8	0.966	44.0
0.2<P<0.8	0.673	17.9
P<0.2	0.961	38.1

Figure 5.9: Scatter plot showing the predicted probability of acceptance of each sample in the test set, using the calibrated RF model using Beta calibration. The blue dots display the correctly classified samples and the red dots display the misclassified samples. The accuracy for each section can be seen in the table at the top of the image, along with the percentage of the data set that is in that section.

As for Logistic Regression, the calibrated RF model was used to predict the average probability of acceptance for different net prices, ranging from very cheap to very expensive, both for all the samples and for the samples that are won and correctly classified as won. The results can be seen in Figure 5.10 along with the variances in Figure 5.11. And as for Logistic Regression, the figures also shows a grey histogram of the distribution of the prices in the test data. The averaged probabilities in the two lines can therefore be compared to how many samples have prices in the same range.

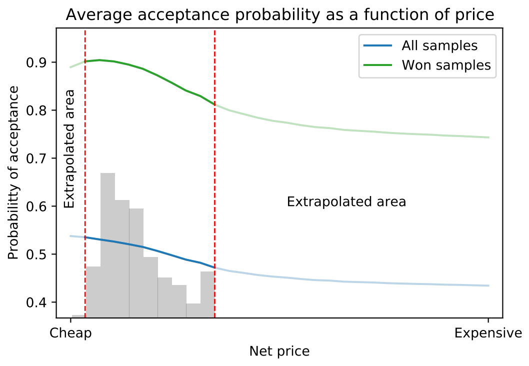


Figure 5.10: Predicted probability of acceptance as a function of price, using the calibrated RF model, for all samples and for the samples that were won and correctly classified as won. The values that are not between the red-dashed lines are extrapolated values, as the data contains no prices in those areas. The grey histogram shows the price distribution of the samples in the unaltered test set.

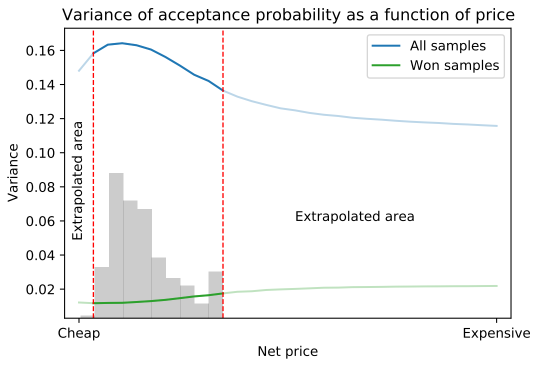


Figure 5.11: Variance of average predicted probabilities of acceptance as a function of price, using the calibrated RF model, for all samples and for the samples that were won and correctly classified as won. The values that are not between the red-dashed lines are extrapolated values, as the data contains no prices in those areas. The grey histogram shows the price distribution of the samples in the unaltered test set.

As for the Logistic Regression, the probability decreases with increased price, seen in Figure 5.10. The variance does not increase that much for the won samples as for the Logistic Regression, seen in Figure 5.11. The probability of acceptance also seem to reach a plateau for the values in the extrapolated area. As can be seen, the extrapolated area shows predicted probabilities for prices that are not present at all in the data.

5.3 Gradient Boosting

Several variants of GBM models were tested as well. No feature selection was done for the GBM models, and the features selected for the RF model were assumed to be the best features for the GBM models.

5.3.1 Hyperparameter optimization

A random search was initiated to find the optimal hyperparameters for the GBM models. The results can be seen in Figure 5.12. The reason for using fewer attempts for GBM models compared to RF was due to training time. The features used for hyperparameter tuning were the features selected from the RF feature selection.

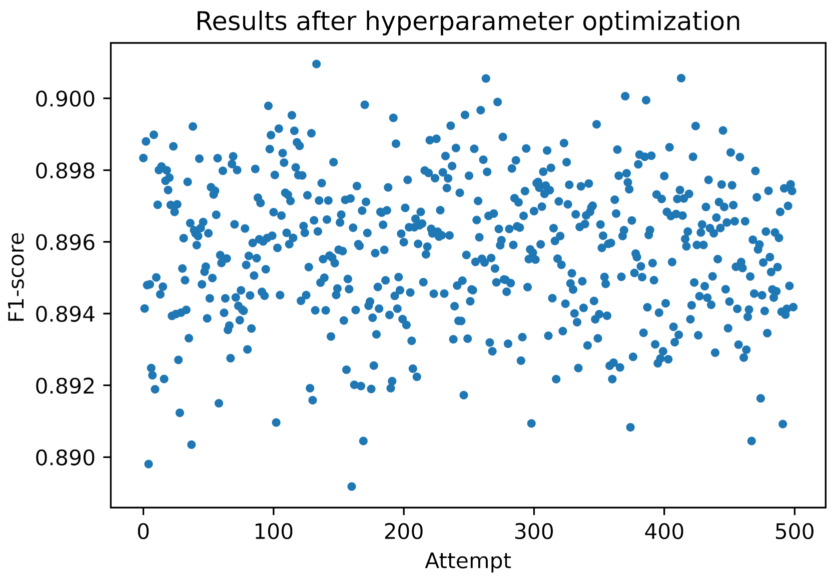


Figure 5.12: The F1-score of the 500 models trained during the random hyperparameter search.

In Figure 5 it can be seen that the F1-score of the models trained differ from 89% to 90.5%. None of the models utilized all 100 000 of the iterations, meaning that they all stopped early due to the early stopping option.

The optimal parameters found by the random search are presented in Table 5.5.

Parameter	Optimal value
max_depth	19
bagging_freq	5
bagging_fraction	0.95
feature_fraction	0.65
num_leaves	29
learning_rate	0.039

Table 5.5: The optimal parameters for the GBM model.

The found learning rate of 0.039 is less than half the size of the default value (0.1). The low learning rate is enabled by the high number of maximum iterations, since even though the steps are small, as long as the iterations keep improving the score on the validation data, the iterations will not stop.

5.3.2 Probability calibration

The reliability diagrams for Sigmoid, Isotonic and Beta calibration are presented in Figure 5.13 along with the uncalibrated GBM model.

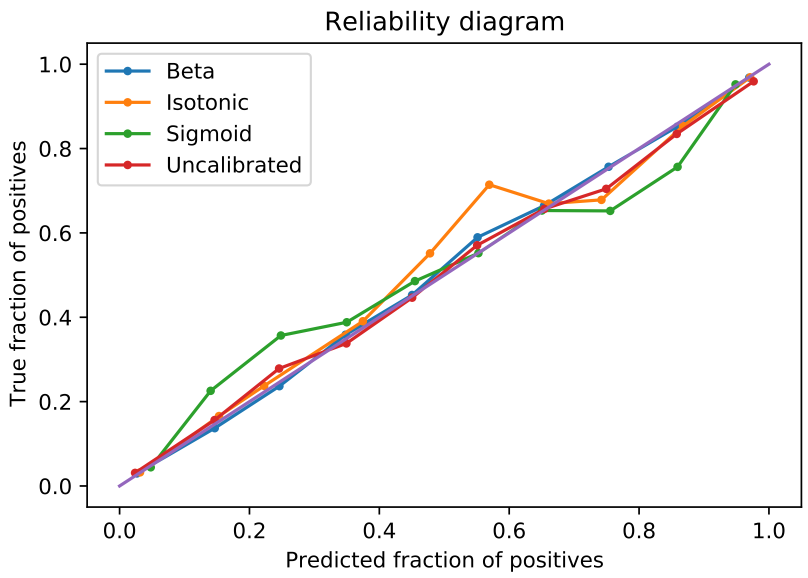


Figure 5.13: Reliability diagram of the three different calibration methods and the uncalibrated classifier.

As can be seen in Figure 5.13, the probabilities from the uncalibrated GBM are quite well calibrated in themselves, at least when one compares to the uncalibrated probabilities that were produced by the RF model. More precise data for the three methods are presented in Table 5.6.

	Beta		**Isotonic**		**Sigmoid**		**Uncalibrated**	
	% of data	Distance	% of data	Distance	% of data	Distance	% of data	Distance
0.0 <P <0.1	31.7	0.031	33.6	0.008	36.6	0.037	32.8	0.072
0.1 <P <0.2	5.8	0.095	3.9	0.126	4.9	0.854	5.5	0.1
0.2 <P <0.3	3.8	0.095	3.6	0.148	2	1.083	3.4	0.329
0.3 <P <0.4	2.5	0.105	5.2	0.158	1.6	0.383	2.3	0.113
0.4 <P <0.5	2.7	0.023	2.1	0.735	1.5	0.309	2.4	0.041
0.5 <P <0.6	2.5	0.383	0.4	1.449	1.3	0.006	2.2	0.201
0.6 <P <0.7	3.1	0.107	3.9	0.084	1.6	0.027	2.6	0.027
0.7 <P <0.8	3.6	0.037	1.5	0.638	2.5	1.034	3.1	0.45
0.8 <P <0.9	7.5	0.069	9.7	0.142	4.5	1.028	5.4	0.233
0.9 <P <1.0	36.7	0.015	36.1	0.023	43.5	0.036	40.3	0.171
Weighted distance	**43.74**		71.67		164.34		134.88	
Brier score	**0.751**		0.7552		0.7715		0.7533	

Table 5.6: Numerical data of the calibration results on the validation set for three different calibration methods and the uncalibrated data.

Both the weighted distance and Brier score were negatively affected by the Sigmoid calibration. It can also be seen in the results that the Brier score was negatively affected by the Isotonic calibration, but it still improved the weighted distance. The best scores were achieved by Beta calibration, and therefore that is the calibration method which will be used in the Calibrated model.

5.3.3 Final results for Gradient Boosting

As for RF, the feature importances were calculated for the GBM using permutation. The importance ranking of the features were very similar to RF with minor differences. The net price was not the most important predictor for the GBM either, however, it was slightly more important than for the RF.

The ROC curves for the five final different GBM models can be seen in figure 5.14. The accuracy, F1-score, AUC score, training time in seconds and the Brier score can be seen in Table 5.7 for the five different models.

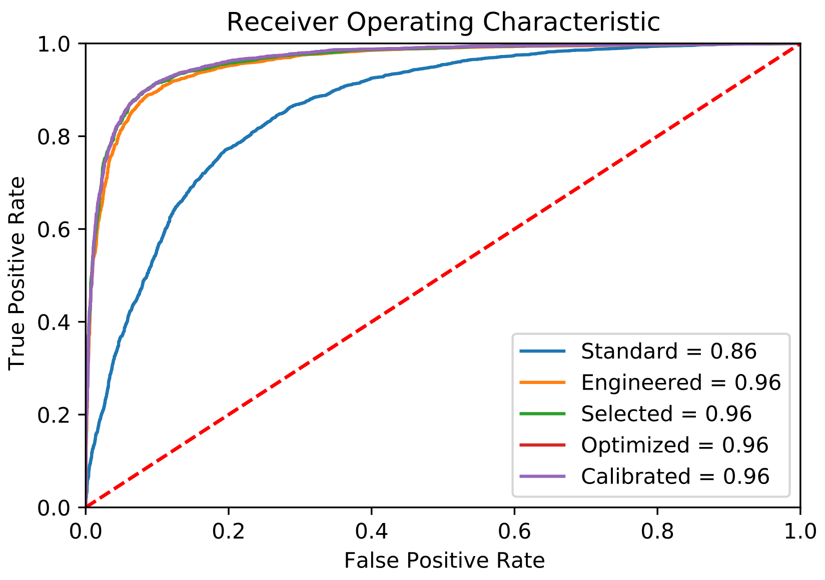


Figure 5.14: ROC curves for the five different GBM models; using the standard features from the original data set, including the engineered features, using the selected features, using the optimized hyperparameters and using Beta-calibration for the probabilities. All models except the Standard model achieved the same AUC score. All AUC scores can be seen in the bottom right corner. The red dashed line represents a random choice classifier.

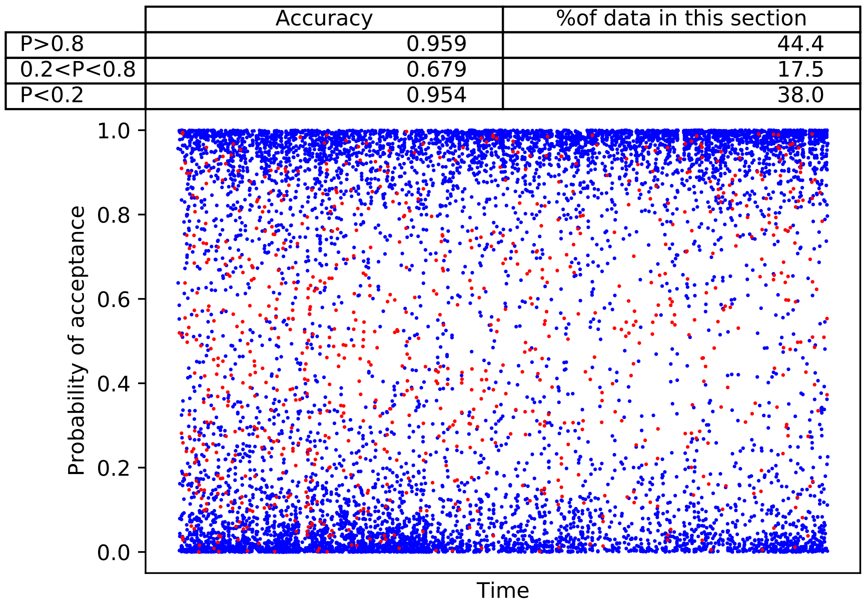
Name	**Accuracy**	**F1-score**	**Training time (s)**	**AUC**	**Brier score**	**Weighted distance**
Standard	0.791	0.812	**17.921**	0.862	0.149	230.68
Engineered	0.9	0.906	41.9738	0.959	0.076	163.92
Selected	0.908	0.913	30.1941	0.963	0.071	107.43
Optimized	**0.909**	0.914	154.555	**0.965**	**0.069**	89.46
Calibrated	0.908	**0.914**	160.336	**0.965**	0.07	**60.80**

Table 5.7: Results for the GBM models showing accuracy, F1-score, AUC score, training time in seconds, Brier score and the weighted distance. The numbers in bold are the best scores in each respective category.

From the results it can be noticed that the best GBM model in terms of accuracy and Brier score was the uncalibrated model with optimized hyperparameters, but it was only slightly better than the calibrated model. The model with standard features was the fastest to train but had significantly worse performance than the other models. The calibrated model had the best F1-score and identical

AUC score as the uncalibrated model. Where the calibrated model is significantly better is when it comes to the weighted distance. The calibrated model is what was chosen to be the best GBM model to compare with the other models.

As for the other models, a scatter plot showing the predicted probability of acceptance of each sample, using the calibrated GBM model, can be seen in Figure 5.15. The table in the figure shows the accuracy of three sections of the figure, where the sections are the same as for the other models.



	Accuracy	%of data in this section
P>0.8	0.959	44.4
0.2<P<0.8	0.679	17.5
P<0.2	0.954	38.0

Figure 5.15: Scatter plot showing the predicted probability of acceptance of each sample in the test set, using the calibrated GBM-model. The blue dots display the correctly classified samples and the red dots display the misclassified samples. The accuracy for each section can be seen in the table at the top of the image, along with the percentage of the data set that is in that section.

The calibrated GBM model was used to predict the average probability of acceptance for different net prices, in the same procedure as for the other models, ranging from very cheap to very expensive. The results can be seen in Figure 5.16 along with the variances in Figure 5.17. As for the two previous models, the figures also shows a grey histogram of the distribution of the prices in the test data. The averaged probabilities in the two lines can therefore be compared to how many samples have prices in the same range.

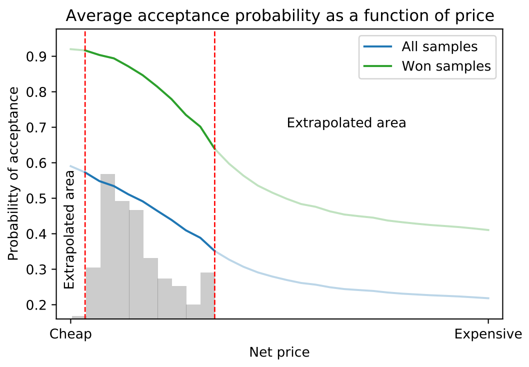


Figure 5.16: Predicted probability of acceptance as a function of price, using the calibrated GBM model, for all samples and for the samples that were won and correctly classified as won. The values that are not between the red-dashed lines are extrapolated values, as the data contains no prices in those areas. The grey histogram shows the price distribution of the samples in the unaltered test set.

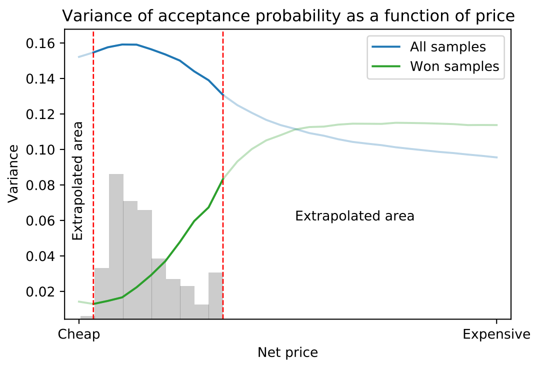


Figure 5.17: Variance of average predicted probabilities of acceptance as a function of price, using the calibrated GBM model, for all samples and for the samples that were won and correctly classified as won. The values that are not between the red-dashed lines are extrapolated values, as the data contains no prices in those areas. The grey histogram shows the price distribution of the samples in the unaltered test set.

From the results it can be noticed that the probability of acceptance decreases faster with increased price than for Logistic Regression and RF in Figure 5.3 and 5.10. It can also be seen that the probabilities and variance reach a plateau in the extrapolated area. As can be seen, the extrapolated area shows predicted probabilities for prices that are not present at all in the data.

5.4 Comparing and combining classifiers

To compare the results from the three different methods of modeling, the results from Engineered Logistic Regression, Calibrated RF and Calibrated GBM are here presented next to each other. In addition, the results from the Combined classifier (RF+GBM) are also presented here. The results can be seen in Table 5.8.

Name	**Accuracy**	**F1-score**	**Training time (s)**	**AUC**	**Brier score**	**Weighted distance**
LR	0.707	0.732	972	0.774	0.193	108
RF	0.912	0.918	**30**	0.969	0.066	158
GBM	0.908	0.914	160	0.965	0.07	**60**
RF + GBM	**0.914**	**0.919**	197	**0.971**	**0.0639**	161
RF + GBM + LR	0.907	0.913	1169	0.960	0.0824	898

Table 5.8: The results from the Engineered Logistic Regression model (LR), Calibrated RF, Calibrated GBM, along with the combined models; RF+GBM and RF+GBM+LR, where the RF and GBM used were calibrated.

The best accuracy, F1-score, AUC and Brier score were all achieved by the RF+GBM model. The ROC curves for all presented models can be seen in Figure 5.18

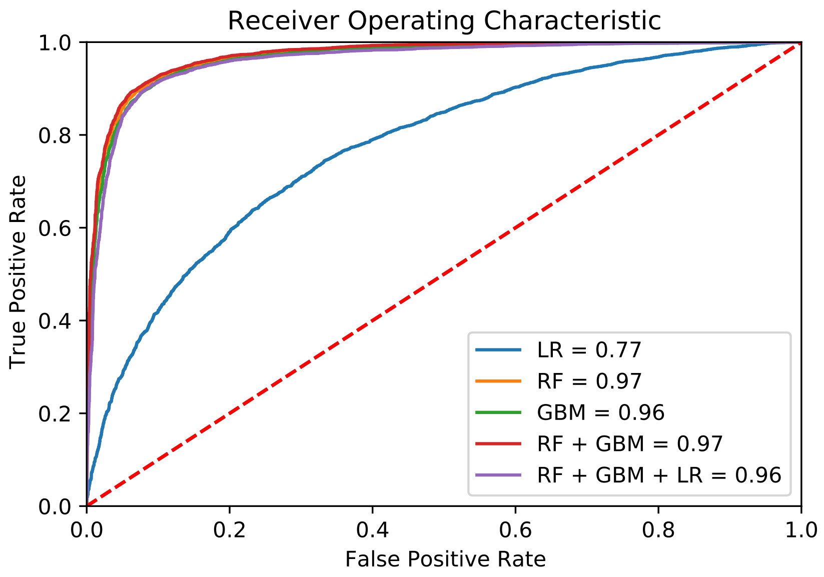


Figure 5.18: ROC curves for the three different models; Logistic Regression (LR), RF, GBM, along with two combined models; RF+GBM and RF+GBM+LR. AUC scores can be seen in the bottom right corner. The red dashed line represents a random choice classifier.

A scatter plot showing the predicted probabilities of acceptance for each sample in the test set, using the combined model with RF and GBM, can be seen in Figure 5.19. The blue dots display the correctly classified samples and the red dots display the misclassified samples. The table in the figure shows the accuracy of three sections of the figure, using the same sections as for the other models.

	Accuracy	%of data in this section
P>0.8	0.966	44.1
0.2<P<0.8	0.692	18.8
P<0.2	0.965	37.1

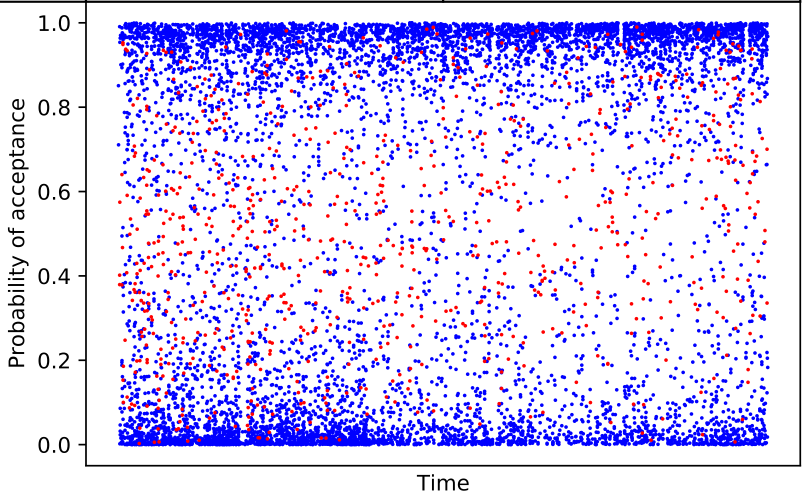


Figure 5.19: Scatter plot showing the predicted probability of acceptance of each sample in the test set, using the combined model with RF and GBM. The blue dots display the correctly classified samples and the red dots display the misclassified samples. The accuracy for each section can be seen in the table at the top of the image, along with the percentage of the data set that is in that section.

Chapter 6

Discussion

This thesis was carried out during the Covid-19 pandemic, which in itself at times was challenging, with the available data containing quotes and orders made during the pandemic. The choice of not having data from March 2020 and forwards was made to reduce the risk of the possible effect of the pandemic on the model. This could be an issue, however, if the resulting model from the thesis should be implemented as a decision support system for sales people. There could namely be a difference in buying behaviour, price elasticity etc., during and directly after the pandemic. This should be taken into consideration when analyzing the results from this thesis.

6.1 Machine learning model comparisons

All models that were tested proved to be better than a random model, which is as expected. Using Logistic Regression, however, seemed to be significantly worse than using tree-based models. One reason for this is that no interaction variables are present in the Logistic Regression model, since they have to be added manually for each interaction variable that is deemed to be important. This is only reasonable when it is possible for the user to think which possible interactions will be important, which was not the case in this thesis. Since B2B sales are so complex it was deemed that the possible important interactions are endless, and it would be better to automatically capture the interaction effects, in this case with tree-based models. It is possible to automatically calculate the interaction variables for Logistic Regression as well, but for the high dimensional data used in this thesis the computational memory and power needed would be too high, due to the curse of dimensionality. An automatic calculation for interaction variables for Logistic Regression would also only capture first degree interactions, or a specified degree, while the tree-based models can capture interaction variable degrees corresponding to the depth of the tree which makes it more powerful for this complex problem. Logistic Regression also makes the assumption of linear variables, which is not necessarily the case in this problem. Tree-based models are non-parametric and therefore makes fewer assumptions on the data, including the assumption of linear variables, which might be better suited for this problem and could be a reason why the tree-based models performed better.

Both the Random Forest (RF) models and the Gradient Boosting Machine (GBM) models performed well. The Engineered model increased the F1-score by 8pp (percentage points) for the Logistic Regression, 6pp for the RF and 11pp for the GBM. That is a sharp increase in all three models. The increase was more modest by the Selected features, but still significant. The RF model increased the most, almost 2.5pp. However, this is expected since the feature selection was optimized using the RF. The GBM model saw only a small increase, 0.7pp. It is possible that the increase in performance would have been higher for the GBM if a separate feature selection was made using GBM. The graph shown in Figure 5.5, showing the results from the feature selection process, mostly looks as one would expect, where the performance gradually gets worse using fewer features. However, it seems that some features have a negative impact on the performance, as the performance gets better by removing some features in the beginning of the graph. Although, this could also be an affect of that too many features could be detrimental to the model, as described in section 4.6. As can be seen in the graph, a much harsher selection can be made as about 10 features achieve a score which is not much worse than with about 300 features. Even if simpler models with less complexity and dimensionality is what usually is desired in a machine learning model, tree-based models like a RF can usually handle models with more complexity without a significant cost of inference time, which can make it worthwhile if the model also performs better.

The hyper parameter optimization yielded 1pp increase for the RF model and only a negligible increase, 0.1pp, for the GBM. It is possible that the hyper parameter search for the GBM would have been more successful if a wider search space was defined. The calibration made only small changes to the F1-score of both models, as is expected. The calibration instead made the probabilities more trustworthy, seen in the improvement of the Brier scores and distance to the reliability curve. However, one cannot draw the conclusion that feature engineering is necessarily the best course of action in order to improve model performance. As feature engineering was the first step taken to improve the performance, it had the most room for improvement. Every subsequent attempt at performance had less and less room for improvement. So if the hyperparameter optimization was the first step, it may have yielded greater improvement in performance than it did being the second to last step.

The calibration using three different methods for the RF model, as can be seen in Figure 5.7 and Table 5.3, were all successful. The weighted distance, the Brier score and the figure itself all reveal that the RF has notoriously poor calibrated probabilities. On the contrary, for the probabilities predicted by the GBM, seen in Figure 5.13 and Table 5.6, the need for calibration is not as apparent. In the case of one of the methods, Sigmoid calibration, the result is actually worse than the uncalibrated probabilities. Since the GBM produces probabilities that are already quite calibrated, seen to the Brier score and weighted distance, and that the identity map does not belong to the logistic family [Kull et al., 2017], the Sigmoid calibration actually uncalibrates the probabilities. This is an example of why one cannot blindly include Sigmoid calibration without evaluating the results. The Isotonic and Beta calibration methods do not suffer from this issue. When comparing the performance of Beta calibration and Isotonic calibration, Beta calibration performs slightly better with respect to the two chosen metrics.

Another interesting finding is that combining the RF and GBM actually improved the classification results compared to each model individually. Adding the Logistic Regression model, however, worsened the performance instead which is not unreasonable since the Logistic Regression model was not that good in itself. If the goal is to use a binary classification, then the best approach might then be to use an ensemble of RF and GBM. If the goal is to use trustworthy probabilities, then this is not necessarily the case if one model is considered to produce more trustworthy probabilities than the other, which might be the case for GBM when comparing Figure 5.10 and Figure 5.10.

Since the initial assumptions of won and lost quotes is a large limitation of what possible patterns can be found in the available data, it would be difficult to get a significantly higher performance until the data is more representable compared to what can be considered as a lost quote line and what is just an expired quote line. Since all data is inserted by people, with little limitations of how the data can be or not be inserted, there is also a possibility that some of the data might be wrong due to human error.

6.2 Price optimization

One of the questions that this thesis was meant to answer was if it is possible to optimize a price for a future quote, given a predictive quote acceptance model. The answer to that question seems to be more intricate than what was expected. Using the method proposed in this work, a price optimization would rely heavily on the probability of acceptance which implies that the probabilities that the model predicts needs to be trustworthy. Since the different methods predict different probabilities when the price is changed, as seen in Figure 5.3, 5.10 and 5.16, it is not completely obvious that the probabilities can be relied upon, since the behaviour of the probabilities vary so extensively between the models. Especially considering RF and GBM, which have similar classification performance but very different probability prediction. This might be because the price was slightly more important for the GBM than the RF model, but could also be an affect from the model in itself. The figures, however, show an expected behaviour of decreasing probability with increased price but it is questionable that the probability of acceptance for e.g. the RF model is still above 70% for a very expensive quote line. The results indicate that the probabilities plateau in the extrapolated area, meaning for prices that are generally not seen in the data itself. The probabilities calculated in this area should therefore be even less reliable, as the model basically guesses the behaviour. This can also be seen in the variances in Figure 5.4, 5.11 and 5.17, where the variance for the won samples increases with decreased probability and increased price for all models, and increases in the extrapolated area. It is however interesting that the variance for the GBM model, and somewhat for the RF model, only increase slightly in the extrapolated area, where the model should be more unsure of the probabilities. It is also interesting that the GBM has a higher increase of variance, while it has a larger decrease of probability which would be the expected behaviour. The probability increases with increased price for cheap samples for the RF model as well, which is not an expected behaviour.

Even if the probabilities are calibrated they are only calibrated in a frequentist sense, using the available data. This could be considered to be naive since prior knowledge in a Bayesian setting might give a more complete picture. The available data might also not capture the complete distribution that is necessary to model the effect of a price change. The reason for that could be that the sales people that have made the historic quotes are *rational* and will set a price that they believe will be accepted. This means that there is no data for very cheap or very expensive quote lines, which implies that the distribution is not complete. One possibility to counter this problem is to introduce augmented data using expert knowledge to try to capture the distribution. This, however, will induce a huge bias and skew the real data which could make the model even more unreliable.

Another interesting finding from the results is that the price is not the most important predictor, which could also be a reason that the probability of acceptance is not as expected when increasing the price. This also makes it difficult to use the models developed in this study to directly optimize the price.

The scatter plots in Figure 5.2, 5.9, 5.15 and 5.19 all reveal that the samples that have low or high probability of acceptance are very often classified correctly. This implies that when the model predicts a sample to be very probable of success or failure, it is very often correct, which makes it possible to use in an application for pricing. Since the price is not the most important predictor, it might sometimes be possible to increase the price for quote lines that are highly probable to be accepted.

An interesting future work is to study a more sequential model, like discussed in the introduction, but instead use a LSTM network. The problem with this approach is that a lot of data is needed, where the data needs to formatted in a way that encapsulates the dimension of the customer and customers' purchase events. This would mean that the model would only be satisfactory for customers with a large amount of historic purchase event.

Appendix A

Supplemental material

A.1 Hyperparameter search space for Random Forest

max_depth	max_features	max_samples
45	0.1	1
47	0.12	0.99
49	0.14	0.98
51	0.16	0.97
52	0.18	0.96
54	0.2	0.95
56	0.22	0.94
58	0.24	0.93
60	0.26	0.92
62	0.28	0.91
64	0.3	0.9
66	0.32	0.89
68	0.34	0.88
70	0.36	0.87
72	0.38	0.86
74	0.4	0.85
76	0.42	0.84
77	0.44	0.83
79	0.46	0.82
81	0.48	0.81
83	0.5	0.8
None	sqrt	None
	log2	

Table A.1: Search space for random forest

A.2 Hyperparameter search space for Gradient Boosting

max_depth	bagging_freq	bagging_fraction	feature_fraction	num_leaves	learning_rate
16	1	1.0	1.0	16	0.03
18	2	0.99	0.95	18	0.039
19	3	0.98	0.9	19	0.048
21	4	0.97	0.85	21	0.057
22	5	0.96	0.8	22	0.066
24	6	0.95	0.75	24	0.075
26	7	0.94	0.7	26	0.084
27	8	0.93	0.65	27	0.093
29	9	0.92	0.6	29	0.102
30	10	0.91	0.55	30	0.111
32	11	0.9	0.5	32	0.12
34					
35					
37					
38					
40					
42					
43					
45					
46					
48					
-1					

Table A.2: Search space for hyper parameter optimization for GBM

Bibliography

[Agresti, 2007] Agresti, A. (2007). *An Introduction to Categorical Data Analysis, Second Edition.* John Wiley Sons Incorporated.

[AlfaLaval, 2021] AlfaLaval (2021). Alfa laval in brief – who we are and what we do.

[Altmann et al., 2010] Altmann, A., Toloşi, L., Sander, O., and Lengauer, T. (2010). Permutation importance: a corrected feature importance measure. *Bioinformatics*, 26(10):1340–1347.

[Bergstra and Bengio, 2012] Bergstra, J. and Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(2):281–305.

[Blum and Langley, 1997] Blum, A. L. and Langley, P. (1997). Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97(1):245–271.

[Bolón-Canedo et al., 2013] Bolón-Canedo, V., Sánchez-Maroño, N., and Alonso-Betanzos, A. (2013). A review of feature selection methods on synthetic data. *Knowledge and Information Systems*, 34(3):483–519.

[Breiman, 1997] Breiman, L. (1997). Arcing the edge. Report, Technical Report 486, Statistics Department, University of California at....

[Breiman, 2001] Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

[Brier, 1950] Brier, G. W. (1950). Verification of forecasts expressed in terms of probability. *Monthly weather review*, 78(1):1–3.

[Claesen and De Moor, 2015] Claesen, M. and De Moor, B. (2015). Hyperparameter search in machine learning. *arXiv preprint arXiv:1502.02127.*

[Dietterich and Kong, 1995] Dietterich, T. G. and Kong, E. B. (1995). Machine learning bias, statistical bias, and statistical variance of decision tree algorithms. Report, Technical report, Department of Computer Science, Oregon State University.

[Ganjisaffar et al., 2011] Ganjisaffar, Y., Caruana, R., and Lopes, C. V. (2011). Bagging gradient-boosted trees for high precision, low variance ranking models. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 85–94.

[Géron, 2019] Géron, A. (2019). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow : concepts, tools, and techniques to build intelligent systems.* O'Reilly Media, Inc., second edition edition.

[Harris, 2013] Harris, D. (2013). *Digital Design and Computer Architecture.*

[Hastie et al., 2009] Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning. Data Mining, Inference, and Prediction.* Springer Series in Statistics. Springer New York, second. edition.

[Ho, 1995] Ho, T. K. (1995). Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282. IEEE.

[Karlinsky-Shichor and Netzer, 2019] Karlinsky-Shichor, Y. and Netzer, O. (2019). Automating the b2b salesperson pricing decisions: A human-machine hybrid approach. Report ID 3368402, Social Science Research Network.

[Ke et al., 2017] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30:3146–3154.

[Koopmans et al., 1964] Koopmans, L. H., Owen, D. B., and Rosenblatt, J. I. (1964). Confidence intervals for the coefficient of variation for the normal and log normal distributions. *Biometrika*, 51(1-2):25–32.

[Kull et al., 2017] Kull, M., Silva Filho, T., and Flach, P. (2017). Beta calibration: a well-founded and easily implemented improvement on logistic calibration for binary classifiers. 54:623–631.

[LaPlaca and Katrichis, 2009] LaPlaca, P. J. and Katrichis, J. M. (2009). Relative presence of business-to-business research in the marketing literature. *Journal of Business-to-Business Marketing*, 16(1-2):1–22.

[Murphy and Winkler, 1977] Murphy, A. and Winkler, R. (1977). Reliability of subjective probability forecasts of precipitation and temperature. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 26(1):41–47.

[Murphy, 2012] Murphy, K. P. (2012). *Machine learning : a probabilistic perspective.* Adaptive computation and machine learning series. MIT Press.

[Niculescu-Mizil and Caruana, 2005] Niculescu-Mizil, A. and Caruana, R. (2005). Predicting good probabilities with supervised learning. In *Proceedings of the 22nd international conference on Machine learning*, pages 625–632.

[Nugues, 2014] Nugues, P. M. (2014). *Language processing with Perl and Prolog : theories, implementation, and application.* Cognitive technologies. Springer, 2. ed. edition.

[Olson and Wyner, 2018] Olson, M. A. and Wyner, A. J. (2018). Making sense of random forest probabilities: a kernel perspective.

[Pantula et al.,] Pantula, S. G., Rawlings, J. O., and Dickey, D. A. *Applied Regression Analysis: A Research Tool (Springer texts in statistics).* Springer.

[Pedregosa et al., 2011] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

[Platt, 1999] Platt, J. (1999). Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74.

[Pressey et al., 2011] Pressey, A., Tadajewski, M., and Ellis, N. (2011). *Business-to-business marketing*. SAGE library in marketing. SAGE.

[Probst et al., 2018] Probst, P., Bischl, B., and Boulesteix, A.-L. (2018). Tunability: Importance of hyperparameters of machine learning algorithms. *arXiv preprint arXiv:1802.09596*.

[Probst and Boulesteix, 2017] Probst, P. and Boulesteix, A.-L. (2017). To tune or not to tune the number of trees in random forest?

[Qu et al., 2020] Qu, H., Ryzhov, I. O., Fu, M. C., Bergerson, E., Kurka, M., and Kopacek, L. (2020). Learning demand curves in b2b pricing: A new framework and case study. *Production and Operations Management*, 29(5):1287–1306.

[Reich and Ghosh, 2019] Reich, B. J. and Ghosh, S. K. (2019). *Bayesian statistical methods*. Chapman Hall/CRC texts in statistical science series. CRC Press, Taylor Francis Group.

[Richard, 2015] Richard, B. (2015). *Adaptive Control Processes: A Guided Tour*. Princeton University Press.

[Robin and Jean-Michel, 2020] Robin, G. and Jean-Michel, P. (2020). *Random Forests with R*. Springer.

[Vaicenavicius et al., 2019] Vaicenavicius, J., Widmann, D., Andersson, C., Lindsten, F., Roll, J., and Schön, T. (2019). Evaluating model calibration in classification. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 3459–3467. PMLR.

[Zadrozny and Elkan, 2001] Zadrozny, B. and Elkan, C. (2001). Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In *Icml*, volume 1, pages 609–616. Citeseer.

[Zhang et al., 2014] Zhang, J. Z., Netzer, O., and Ansari, A. (2014). Dynamic targeted pricing in b2b relationships. *Marketing Science*, 33(3):317–337.

Master's Theses in Mathematical Sciences 2021:E4
ISSN 1404-6342

LUTFMS-3406-2021

Mathematical Statistics
Centre for Mathematical Sciences
Lund University
Box 118, SE-221 00 Lund, Sweden

http://www.maths.lth.se/