University of Economics in Prague

Faculty of Finance and Accounting

Department of Banking and Insurance

Field of study: Financial Engineering


# Credit Scoring Using Ensemble Machine Learning Methods

Author of the Master Thesis: Bc. Jan Helcl

Supervisor of the Master Thesis: prof. RNDr. Jiří Witzany, PhD.

Year of Defence: 2018

# The Declaration of Authorship

I hereby declare that I carried out the master thesis “Credit Scoring Using Ensemble Machine Learning Methods” independently, using only the resources and literature properly marked and included in the bibliography.

Prague, XXX, 2018 signature of the author

# Acknowledgements

I would like to thank prof. RNDr. Jiří Witzany, Ph.D. for his comments and valuable feedback, my family and my partner Karin for her continual moral support.

**Abstract:**

The aim of this thesis is to examine the performance of selected decision tree-based ensemble machine learning models and to assess their applicability in credit scoring. In the first part, general principles behind supervised machine learning were introduced. Selected ensemble models were then explained in detail, special focus was placed on decision trees, the building blocks of these models. The standard approach to credit scoring function development was introduced and used to develop a benchmark model. In last section, the ensemble models were trained in several different ways. Their out-of-sample performance was measured and compared to that of the benchmark.

**Keywords:**

ensemble machine learning, credit scoring, random forest, extremely randomized trees, AdaBoost, gradient boosting, logistic regression

**Abstrakt:**

Cílem této práce je zhodnotit prediktivní sílu zvolených metod skupinového strojového učení založeného na rozhodovacích stromech a posoudit aplikovatelnost těchto modelů v kreditním skóringu. V první části byly vysvětleny základní principy strojového učení s učitelem. Zvolené metody skupinového strojového učení byly poté detailně popsány, zvláštní důraz byl kladen na popis rozhodovacích stromů jakožto základních stavebních prvků daných modelů. Standardní přístup ke kreditnímu skóringu byl popsán a použit pro vývoj referenčního modelu. Skupinové modely byly vyvinuty několika způsoby a jejich predikční síla byla porovnána s referenčním modelem.

**Klíčová slova:**

skupinové strojové učení, kreditní skóring, náhodný les, extrémně náhodné stromy, AdaBoost, gradient boosting, logistická regrese

# CONTENTS

# INTRODUCTION

Assessing the creditworthiness of potential clients is a classic problem faced by all institutions involved in the lending business, banks and non-bank providers alike. Historically, applicants were assessed individually based on expertise and experience of analysts responsible for the approval process. Up today risks associated with the borrowers' ability to repay their liabilities are dealt with in similar ways when dealing with large corporate debts. When we are, on the other hand, dealing with large amounts of smaller loans such approaches easily become infeasible. Over time and with the advancements in mathematical and statistical modelling as well as with the availability of modern computational tools much of this process gradually became more and more automated. Without at least partial automation, institutions would hardly be able to provide retail loans, auto loans or credit cards at today's scale.

What the whole world has been experiencing during the last decade or two is a rapid growth of the amount of data collected by all kinds of businesses that became possible thanks to new ways of storing the massive datasets efficiently. Hand in hand with this phenomenon came the fast progress in advanced data analytics and machine learning. Today virtually no industry is unaffected by these advancements. Machine learning based automated systems are being deployed in marketing, transaction monitoring, supply chain management and other areas at all sorts of companies ranging from e-commerce to manufacturing. Financial sector is no exception to this rule.

Financial institutions and finance scholars are experimenting with alternative datasets (geospatial data, online footprint etc.) as well as with alternative predictive models. These are used in anti-money laundering, trading or risk modelling.

The main goal of this thesis is to examine the performance of selected decision tree-based ensemble machine learning models in credit scoring and to compare their capabilities to those of the industry standard model represented by logistic regression. My secondary goal is to address other practical issues as well. Namely, how complicated and timely it is to develop such models and especially how difficult is to find the optimal hyper-parameter settings.

In the first three chapters, I will briefly introduce the reader to the basics of supervised machine learning. Special emphasis will be placed on the nature of the errors in models' predictions. These will be explained in the context of bias-variance tradeoff.

Fourth chapter will be dedicated to in-depth analysis of decision tree models with focus on classification and regression trees (CART). Overview of other decision tree building algorithms will be provided as well. A whole section will be devoted to explaining the geometrical interpretation of decision trees.

Chapter Five will cover the main principles of combining many models into ensembles. Four different ensemble machine learning models, falling into two distinct model families: model averaging and boosting, will be introduced. All four of these models are based on CART. At the end of this chapter the issue of explainability of such models will be addressed briefly.

Chapter Six will introduce the reader to the general principles of credit scoring. The industry standard model, logistic regression, will be explained in detail and the whole scoring function development process will be described, including methods of model validation and performance measurement.

In Chapter Seven, all the above mentioned concepts will be applied to a real-world credit dataset. The performance of the selected ensemble models will be measured and compared to that of the benchmark model, logistic regression. The dataset will come from a peer-to-peer (P2P) lending platform Bondora. P2P lending presents an opportunity for using alternative approaches to credit scoring as traditional banks are unlikely to adopt these methods any soon due to heavy regulation.

Since the Bondora platform is open to general public, all the findings are immediately applicable in practice. This thesis is written solely for academic purposes and the author bears no responsibility for any losses caused by using the models he developed.

# 1 INTRODUCTION TO MACHINE LEARNING

Machine learning is a very broad field with no clear and simple definition. Generally, it is a subfield of Computer Science, more specifically a subfield of artificial intelligence, concerned with algorithms that allow machines to learn. These algorithms can be classified into four wide categories by the nature of inputs they require during training.

- **Supervised learning** deals with labeled data. Each observation used during training is pre-classified. The aim is to infer a function that links an object to a specific output value.
- **Unsupervised learning** uses unlabeled data only. The goal is to find a structure that describes the data reasonably well.
- **Reinforcement learning** is inspired by behavioural psychology. There are no explicit input-output pairs during the training. The system learns through interaction with its environment based on rewards/punishments for its behaviour.
- **Apprenticeship learning** uses indirect hints derived from a teacher´s behaviour.

Credit scoring is, in principle, quite simple. We need a model, which can be used to discriminate between good and bad debtors. This model is built using data about past applications where we know, whether these credits turned out to be good or bad. This, in other words, resembles a classic supervised machine learning problem. In the next section we will investigate the principles behind supervised machine learning a bit further.

# 2 SUPERVISED MACHINE LEARNING

As mentioned above, supervised machine learning uses labeled data to extract patterns that can be used to predict an unknown label of a previously unseen observation. Typical use cases include, among others, image recognition, natural language processing (e.g. spam detection) or housing prices prediction.

First, let’s define the concept more formally. When talking about supervised learning, analogies to human reasoning or (especially in case of artificial neural networks) the way brain biologically works are often made. This phenomenon can, however, be understood

mathematically as a function approximation problem[1]. Suppose $f(x)$ is some underlying function that links input $x$ to output $y$. Our goal is to find $\hat{f}(x)$ that approximates $f(x)$ well enough for the given purpose.

We can further classify supervised learning problems as follows:

- For $y \in \mathbb{R}$ i.e. the output variable is a real number we speak about regression.
- For $y \in G$ where $G$ is a finite set we speak about classification.

The input $x$ is a row vector of explanatory variables.[2] During the training phase we use $X$ an $n$ by $d$ matrix (where $n$ is the number of observations and $d$ is the number of explanatory variables) and $y$ a vector of outputs to find $\hat{f}(x)$.

There are numerous machine learning algorithms that can be used to find such a function. Before diving into details of some of these methods selected for the purpose of this thesis, we will first closely examine the error of this approximation.

# 3 Bias-Variance Tradeoff

Ultimately, our goal is to build a model that can generalize well. This means we need a model that shows good prediction capability on previously unseen observations. The importance of testing on samples that were not used during training cannot be exaggerated. As illustrated bellow, it is actually easy to find a model that can label 100% of the training set correctly, given we can choose models that are arbitrarily complex.

## 3.1 Intuition Behind Bias-Variance Tradeoff

To get some intuition first, let's set up a simple simulation to demonstrate the relationship between generalization error and model complexity. To do so we need to introduce two sources of error (these will be more formally defined in the next section):

[1] Hastie, Trevor; Tibshirani, Robert; Friedman, Jerome (2008). *The Elements of Statistical Learning (2nd ed.)*. Springer. ISBN 0-387-95284-5.

[2] Output $y$ can also be a vector, consider for example a case of predicting a point in two dimensions (described by latitude and longitude). We won't deal with this kind of problems in this thesis.

- **Bias** is an error arising from erroneous assumptions in the model. High bias can cause an algorithm to miss the relevant relations between explanatory variables and target output (this is often referred to as underfitting).
- **Variance** is an error coming from sensitivity to small fluctuations in the training set. High variance can cause an algorithm to fit the random noise in the training data, rather than the true relations (often referred to as overfitting).

Our data will be generated by a simple statistical model:

$$y = \sin x + \varepsilon.$$

Where $\varepsilon$ is normally distributed with 0 mean and 0.5 variance and is independent of $x$. We will repeatedly draw samples of 25 observations from this population, 90% of the sample will be used as a training set to fit a polynomial of a given degree. The out-of-sample performance will be measured using the other 10%. Mean squared error (MSE) will be used as a metric for assessing the performance:

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

Where $y_i$ is the i-th observed value of the output variable, $\hat{y}_i$ is the corresponding value predicted by the model and $n$ is the number of observations.

We will fit polynomials up to 12 degrees (50 iterations for each degree).

Figure 1 shows fitted curves for polynomials of selected degrees. In the upper left corner we can see the results of trying to fit a simple line to the generated data. What we observe is a clear example of underfitting. The model is not complex enough to capture the non-linear nature of the data. To put it in a different way, we can see that this model results in very high bias (due to the wrong assumption of linearity) but small variance as all the lines look very similar. The upper right figure shows fits of polynomials of degree 3. Intuitively, this seems to be the best model among the four. The blue line representing the average fit is quite close to the underlying function $\sin x$. Moreover, most of the green curves look very similar to each other. In the bottom of Figure 1 we can see two clear examples of overfitting. While the blue curve is close to the underlying function (i.e. we see low bias), the models are very sensitive to small variations in the data and each fitted curve is very different from the others.

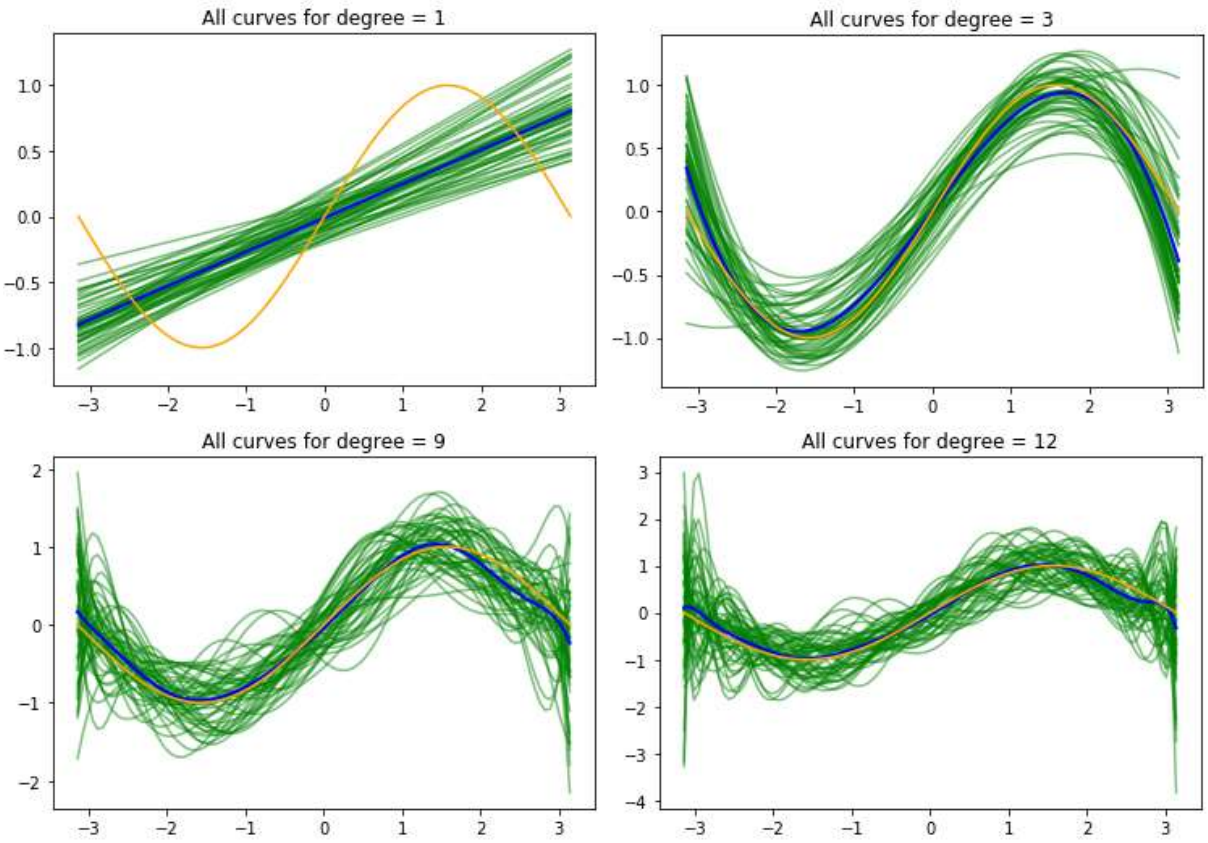


*Figure 1: Curves for selected degrees of polynomials. Underlying function $sin(x)$ is shown in orange. Curves fitted during all 50 iterations are shown in green. The blue curve represents the average fit.*

Next, we will have a look at the performance of the models. Figure 2 shows average train and test scores of polynomials of different degrees.

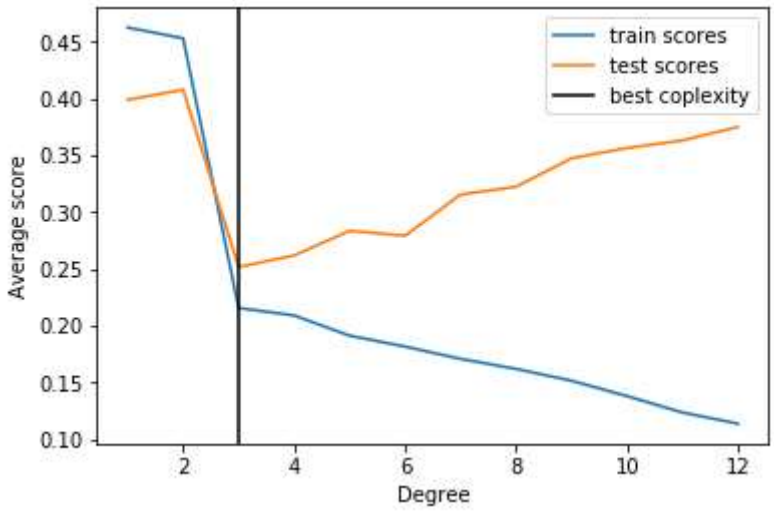


*Figure 2: Average train and test performance of polynomials of different degrees (each point represents the average performance of all 50 models of the given degree).*

As expected, we see that the polynomial of degree 3 provides the best out-of-sample performance. Note, how the training errors decrease with increasing model complexity while test errors start to go up after reaching certain degree of complexity.

We will get back to this simulation one more time after exploring the relation between bias and variance more formally in the next section.

## 3.2 Bias–Variance Decomposition

Bias-variance decomposition was first introduced in the context of linear regression. If we assume $y = f(x) + \varepsilon$, where $E[\varepsilon] = 0$ and $Var(\varepsilon) = \sigma_\varepsilon^2$, we can derive the expected prediction error of $\hat{f}(x)$. The derivation proceeds as follows[3]:

$$E\left[\left(y - \hat{f}(x)\right)^2\right] = E\left[y^2 + \hat{f}(x)^2 - 2y\hat{f}(x)\right] = E[y^2] + E\left[\hat{f}(x)^2\right] - E\left[2y\hat{f}(x)\right],$$

$$E\left[2y\hat{f}(x)\right] = 2E\left[(f(x) + \varepsilon)\hat{f}(x)\right] = 2E\left[f(x)\hat{f}(x)\right] + E[\varepsilon]E\left[\hat{f}(x)\right],$$

$$E\left[\left(y - \hat{f}(x)\right)^2\right] = Var(y) + E[y]^2 + Var\left(\hat{f}(x)\right) + E\left[\hat{f}(x)\right]^2 - 2E\left[f(x)\hat{f}(x)\right].$$

As $E[xy] = E[x]E[y] + Cov(x, y)$:

$$E\left[\left(y - \hat{f}(x)\right)^2\right] = Var\left(f(x)\right) + Var(\varepsilon) + E[y]^2 + Var\left(\hat{f}(x)\right) + E\left[\hat{f}(x)\right]^2$$

$$-2E[f(x)]E\left[\hat{f}(x)\right] - 2Cov\left(f(x), \hat{f}(x)\right).$$

Since $Var(x - y) = Var(x) + Var(y) - 2Cov(x, y)$:

$$E\left[\left(y - \hat{f}(x)\right)^2\right] = Var(\varepsilon) + Var\left(f(x) - \hat{f}(x)\right) + E[y]^2 + E\left[\hat{f}(x)\right]^2 - 2E[f(x)]E\left[\hat{f}(x)\right]$$

$$= Var(\varepsilon) + Var\left(f(x) - \hat{f}(x)\right) + E[f(x)]^2 + E\left[\hat{f}(x)\right]^2 - 2E[f(x)]E\left[\hat{f}(x)\right],$$

$$E\left[\left(y - \hat{f}(x)\right)^2\right] = \sigma_\varepsilon^2 + Var\left(f(x) - \hat{f}(x)\right) + \left(E[f(x)] - E\left[\hat{f}(x)\right]\right)^2,$$

$$Prediction\ error\ (MSE) = irreducible\ error + variance + bias^2.$$

[3] Vijayakumar, Sethu (2007). *The Bias–Variance Tradeoff* . University Edinburgh. Available at http://www.inf.ed.ac.uk/teaching/courses/mlsc/Notes/Lecture4/BiasVariance.pdf

Irreducible error is inherent to the problem and therefore cannot be reduced. When searching for the right model, we are looking for optimal balance between bias and variance. Models with low level of complexity tend to have high bias and low variance, whereas largely complex models usually have low bias but suffer from very high variance. The relationship is demonstrated by Figure 3.

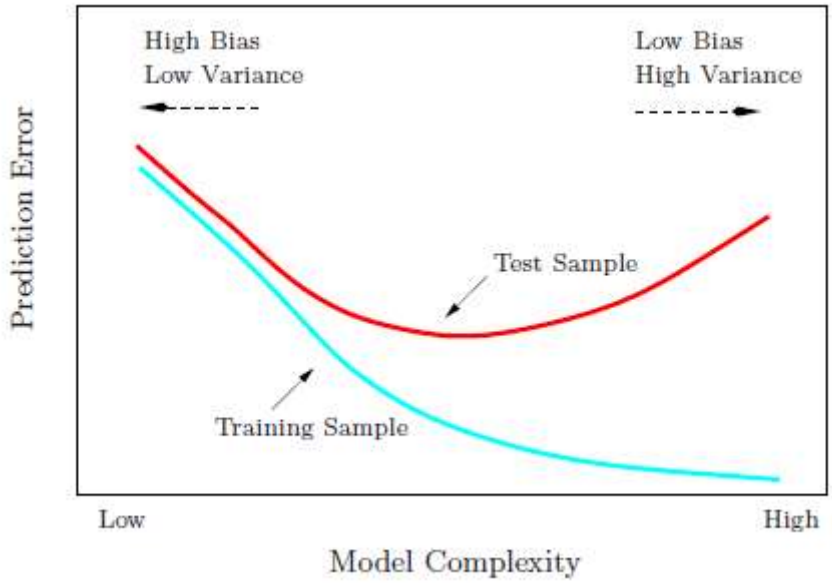


*Figure 3: Bias-variance tradeoff, taken from Elements of statistical learning[4].*

Now, let´s return to our previous simulation and examine the test results more closely. In Figure 4 we can clearly see the variance (orange curve) increasing with the model complexity (degree of polynomial), bias (in blue) jumping to almost zero between degree 2 and 3 and the irreducible error as the difference between the green curve representing total error and the red curve ($bias^2$ + variance).

**Note:** I introduced the bias-variance tradeoff concept in the context of regression as I find it more intuitive and easier to visualise. We could, however, derive it for classification problems as well. For derivation of a general decomposition of an arbitrary loss function see Domingos 2000[5]. Alternatively, in case of probabilistic classification, we could use the expected squared

[4] Hastie, Trevor; Tibshirani, Robert; Friedman, Jerome (2008). *The Elements of Statistical Learning (2nd ed.).* Springer. ISBN 0-387-95284-5.
[5] Domingos, Pedro (2000). *A unified bias-variance decomposition* . ICML. Available at https://homes.cs.washington.edu/~pedrod/bvd.pdf

error of the predicted probabilities with respect to the true probabilities and decompose it as was shown above[6].

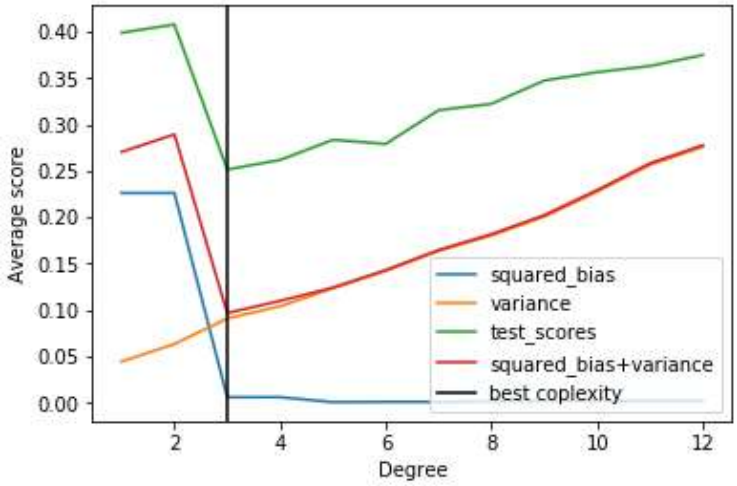


*Figure 4: Bias-variance decomposition of the experiment's results*

# 4 DECISION TREES

Representations of knowledge as tree-like structures can be found across many fields. It is a simple and very intuitive way to visualise what we know about some phenomena in the form of if-then statements resulting in a readable flowchart. Figure 5 shows an example of a very simple decision tree.

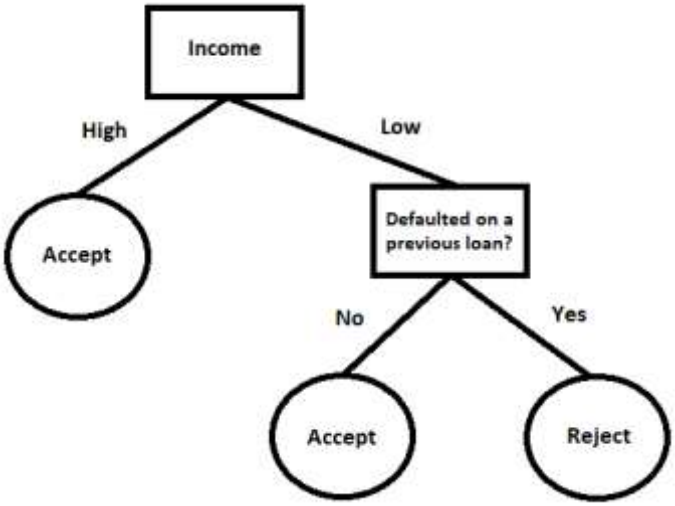


*Figure 5: Very simplistic example of a tree that could be used during loan approval process*

[6] Manning, Christopher D.; Raghavan, Prabhakar; Schütze, Hinrich (2008). *Introduction to Information Retrieval*. Cambridge University Press. pp. 308–314.

Decision trees are also among the most widely used supervised machine learning models. Building a decision tree can be seen as specialization in the space of hypotheses (trees) starting from an empty tree. The goal is to find a tree consistent with the data. The training data are iteratively partitioned into smaller and smaller subsets that are increasingly more homogenous. This procedure is known as *„top down induction of decision trees"*(TDIDT), sometimes also called "*divide and conquer"* method [7]. Below is the general algorithm described in pseudocode.

1. Select one input variable as the root of a new sub-tree.
2. Partition the data according to values of this variable into subsets and create a node for each subset.
3. For each node:
    a. If stopping criterion is not met: go to point 1.
    b. Else: stop.
4. Output the trained model.

*Algorithm 1: General tree building procedure*

Various approaches differing in the way the input variable is selected, the partitions are made and stopping conditions has been proposed. Next, I will introduce one particular decision tree building algorithm that will later be used to construct more complex ensemble models. I will briefly introduce some of the other important approaches to decision tree building in the next section.

## 4.1 CLASSIFICATION AND REGRESSION TREES

Classification and regression trees (CART) are a non-parametric decision tree learning technique that produces either classification or regression trees, depending on whether the output variable is categorical or numeric, respectively. It is probably the most commonly used decision tree algorithm.

CART use numeric inputs only, all categorical variables have to be converted to numerical. Most common approach is to use dummy variables (creating a set of binary variables indicating which category is present). At each node a binary split is made.

[7] BERKA, P (2003). *Dobývání znalostí z databází*, Praha Academia, 80-200-1062-9

### 4.1.1 Splitting Criteria

At each node the first task is to select the best input variable and the cutting point to base the split on. This is achieved in the following way.

1. For each input variable:
    a. Find all potential cutting points by sorting values of the input variable in ascending order, leaving out duplicate values and the last value.
    b. Evaluate all potential cutting points by splitting the data accordingly and measuring impurity of resulting subsets (impurity in output variable is measured).
    c. Remember the best split (cutting point and value of the metric used to measure impurity).
2. Choose the input variable that resulted in the best split, create nodes for both resulting subsets.

*Algorithm 2: CART splitting criteria*

For classification tasks two measures of impurity are commonly used: Gini index and Information gain.

Gini index is defined as:

$$G = \sum_{c=1}^{C} w_c * G_c,$$

$$G_c = 1 - \sum_{k=1}^{K} P_{ck}^2.$$

Where: C is the number of subsets (C = 2 in case of CART), K is the number of categories of the output variable (K = 2 for binary classification), $P_{ck}$ is the probability of k-th value of input variable occurring in c-th subset, $G_c$ is the Gini index of c-th subset, $w_c$ is the weight computed as number of observations belonging to c-th subset divided by total number of observations in the original set. The lower the impurity (Gini coefficient) the better the split.

Information gain is a concept based on information theory. It is the difference between Information entropy of the original set and the weighted sum of entropies of the resulting subsets. Entropy is defined as:

$$H = -\sum_{k=1}^{K} p_k * \log_2 p_k.$$

Where: K is the number of categories of the output variable (K = 2 for binary classification) and $p_k$ is the percentage of the given class of the input variable in the set. Information gain is then simply:

$$IG = H(original\ set) - \sum_{c=1}^{C} H(subset\ c).$$

Where C is again the number of subsets (C = 2 in case of CART). Naturally, the higher the Information gain the better the split.

For regression problems Mean squared error (MSE) defined earlier is often used. Split that resulted in the lowest weighted sum of MSE is considered the best.

Note, how the algorithm proceeds in a greedy fashion. At each node a locally optimal split is made. This selection is conditional on all previous splits and doesn't take into account any later splits.

#### 4.1.2 Stopping Criteria

Various stopping criteria can be specified by the user to control the complexity of resulting models.

- **Maximum depth** refers to the number of levels on which splits are made. (Figure 5 shows a tree with depth = 2). The higher the maximum depth the more complex the final model.
- **Minimum number of samples in node** is the number of samples required to be in a node to perform a split. The higher the number the simpler tree we get.
- **Minimum number of samples in leaf nodes.** Leaves are the final nodes that are not being split any further (the tree in Figure 5 has 3 leaves). Same logic applies, the higher the number of samples required to be in leaves the simpler the model.
- **Minimum impurity decrease** is a user defined threshold. All splits are required to decrease the impurity at least by this amount. This is a way to avoid overfitting as only splits with significant gain in accuracy are being made.

There is one natural case when the algorithm stops. It is when a node achieves the lowest possible impurity. For classification tasks this means that all observations in the node are of the same class, for regression the value of output variable in the node is constant. Now it is

easy to see that in case none of the above mentioned stopping criteria is defined the model is trained to label all training data with 100% accuracy. Such models, however, suffer from high generalization errors as they have very high variance (but low bias).

Several ways to simplify such fully grown trees has been developed. These methods are commonly called tree pruning. I won't describe these as I won't need them for building the ensembles. Interested reader can see for example Hastie, Tibshirani, Friedman 2008[8].

### 4.1.3 Prediction

Making prediction using a built decision tree is quite straightforward. At each node the value of respective variable is tested and sent to the next node accordingly until a leaf node is reached. In case of classification, majority vote is performed in the leaves. In other words, the prediction represents the most frequent class among the training observations that fell into the given leaf node. For regression problems average value of the output variable of these observations is used.

**Note:** When doing classification, we are often more interested in the probability that the given observation falls into the respective class. Credit scoring (estimation of probability of default) represents such a case. Proportions of classes in the leaf node can be used as estimates of these probabilities.

## 4.2 Overview of Other Decision Tree Learning Algorithms

As mentioned earlier there are many ways in which decision trees can be built. What follows next is a brief overview of some of the alternatives to classification and regression trees.

### 4.2.1 Iterative Dichotomiser 3 (ID3)

ID3 is an algorithm invented by Ross Quinlan. It can be applied to discrete data only. At each node a variable resulting in the highest Information gain is selected and a new node for each category of the variable is created. Naturally, every variable can be used only once, when there are no variables left the algorithm stops. This procedure results in relatively simple decision trees. For more information refer to the Quinlan's 1986 book[9].

[8] Hastie, Trevor; Tibshirani, Robert; Friedman, Jerome (2008). *The Elements of Statistical Learning (2nd ed.)*. Springer. ISBN 0-387-95284-5.

[9] Quinlan, J. R. (1986). *Induction of Decision Trees*. Machine Learning 1: 81-106, Kluwer Academic Publishers. Available at http://hunch.net/~coms-4771/quinlan.pdf

### 4.2.2 C4.5 and C5.0

C4.5 is an algorithm developed by the same author, it is an extension to ID3. It can handle continuous inputs by performing online discretization, procedure similar to that used in CART. It also handles missing values by ignoring them when computing the Information gain. C4.5 uses a pruning method to simplify the final tree[10]. C5.0 is an improvement to C4.5 that Quinlan is commercially selling[11].

### 4.2.3 Chi-square Automatic Interaction Detection (CHAID)

CHAID is a technique that uses Person's Chi-square tests of independence when performing the splits. Important advantage of this method and similar approaches is that it results in simpler trees as only statistically significant split are made (there is no need for pruning). As in ID3 and C4.5/C5.0 the splits are multiway. For more details see Magidson 1994[12].

### 4.2.4 Multivariate Adaptive Regression Splines (MARS)

MARS can be viewed as generalization of stepwise linear regression or as a modification of CART. It produces complex binary trees (need for pruning) with linear function at each leaf node (instead of a constant as in CART).For detailed discussion on MARS refer to Hastie T., Tibshirani R., and Friedman J.H. (2009)[13].

Other less commonly used techniques are available as well: evolutionary programing-based methods[14], Markov chain Monte Carlo[15] or bottom-up search[16].

---

[10] Quinlan, J. R. (1986). *Induction of Decision Trees*. Machine Learning 1: 81-106, Kluwer Academic Publishers. Available at http://hunch.net/~coms-4771/quinlan.pdf

[11] Pandya R., Pandya J. (2015). C5.0 *Algorithm to Improved Decision Tree with Feature Selection and Reduced Error Pruning.* Available at https://research.ijcaonline.org/volume117/number16/pxc3903318.pdf

[12] Magidson, Jay (1994) *The CHAID approach to segmentation modeling: chi-squared automatic interaction detection*, in Bagozzi, Richard P. (ed); *Advanced Methods of Marketing Research*, Blackwell, Oxford, GB, pp. 118–159

[13] Hastie, Trevor; Tibshirani, Robert; Friedman, Jerome (2008). *The Elements of Statistical Learning (2nd ed.)*. Springer. ISBN 0-387-95284-5.

[14] Papagelis A., Kalles D.(2001). *Breeding Decision Trees Using Evolutionary Techniques*, Proceedings of the Eighteenth International Conference on Machine Learning. pp. 393–400. http://www.gatree.com/data/BreedinDecisioTreeUsinEvo.pdf

[15] Chipman, Hugh A., Edward I. George, and Robert E. McCulloch (1998). *Bayesian CART model search*. Journal of the American Statistical Association 93.443: 935–948.

[16] Barros R. C., Cerri R., Jaskowiak P. A., Carvalho, A. C. P. L. F., *A bottom-up oblique decision tree induction algorithm*. Proceedings of the 11th International Conference on Intelligent Systems Design and Applications (ISDA 2011).

## 4.3 Decision Trees: Summary

Decision trees comes with several important advantages.

- They are **simple and easy to interpret**. Decision trees can be easily visualized and can be interpreted intuitively.
- Decision trees are **computationally cheap**. Even large amounts of data can be analysed using standard resources in reasonable time.
- Most approaches represent **non-parametric methods**. There are no distributional assumptions on training data or prediction errors. There are also no independence or constant variance assumptions.
- There is **in-built feature selection**. Trees with limited depth (or limited using another above mentioned stopping criteria) are relatively robust to presence of irrelevant input variables, these are not used in any split.

Naturally, it also comes with certain limitations.

- Standalone decision trees tend to be **less accurate** than some of the other machine learning models[17].
- Fully grown trees are **very sensitive to small variations in training data**. This often leads to the need for pruning.
- Some decision tree learning algorithms require **discretization of input variables**.

Later chapters on ensemble machine learning will show how some of these advantages can be leveraged to overcome the limitations by combining more decision trees into a single predictive model.

## 4.4 Geometrical Interpretation of Decision Trees

### 4.4.1 Different Visual Representations of Decision Trees

Finally, we will have a closer look at the geometrical interpretation of decision trees. From now on, when I am discussing decision trees I refer to CART unless explicitly stated otherwise.

[17] Gareth, James; Witten, Daniela; Hastie, Trevor; Tibshirani, Robert (2015). *An Introduction to Statistical Learning*. New York: Springer. p. 315. ISBN 978-1-4614-7137-0.

By performing binary splits the feature space is partitioned by lines parallel to axis. In two dimensions this results in creating areas of rectangular shape as shown in Figure 6. In more dimensions we speak of hyperrectangles.

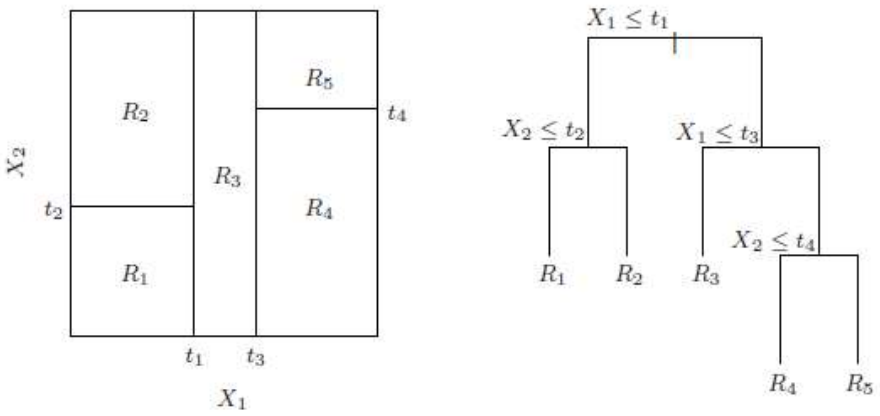


*Figure 6: Partitions of two dimensional feature space by a decision tree. Taken from Elements of Statistical Learning.*[18]

In Figure 6 we can see two graphical representations of the same decision tree trained using two input variables $X_1$ and $X_2$. On the right, we see the now familiar tree-like structure. On the left, the partitions the model makes are visualized.

In case of classification the most frequent class within each region is being predicted. When performing regression, prediction is constant within the given region. Figure 7 shows prediction surface of the same model. This is done by extending the space by a third dimension representing the output variable.

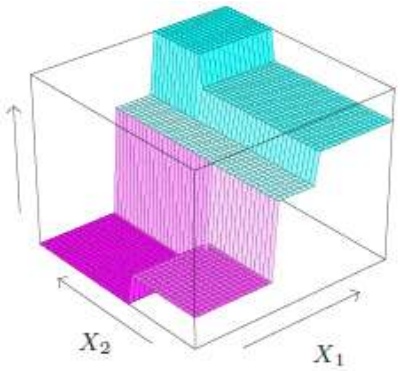


*Figure 7: Prediction surface of the same regression tree. Taken from Elements of Statistical Learning*[19].

[18,19] Hastie, Trevor; Tibshirani, Robert; Friedman, Jerome (2008). *The Elements of Statistical Learning (2nd ed.)*. Springer. ISBN 0-387-95284-5.

### 4.4.2 Visual Examples of Decision Tree Models of Different Complexities

I will end the discussion on standalone decision trees by showing visual examples of decision tree models of different complexities.

The first problem is a regression one. The data are generated by the same statistical model as the one used in Chapter 2 on bias-variance tradeoff ($\sin x$ + normally distributed noise), so there is only one input variable. Figure 8 shows prediction curves for regression trees with different maximum depths.

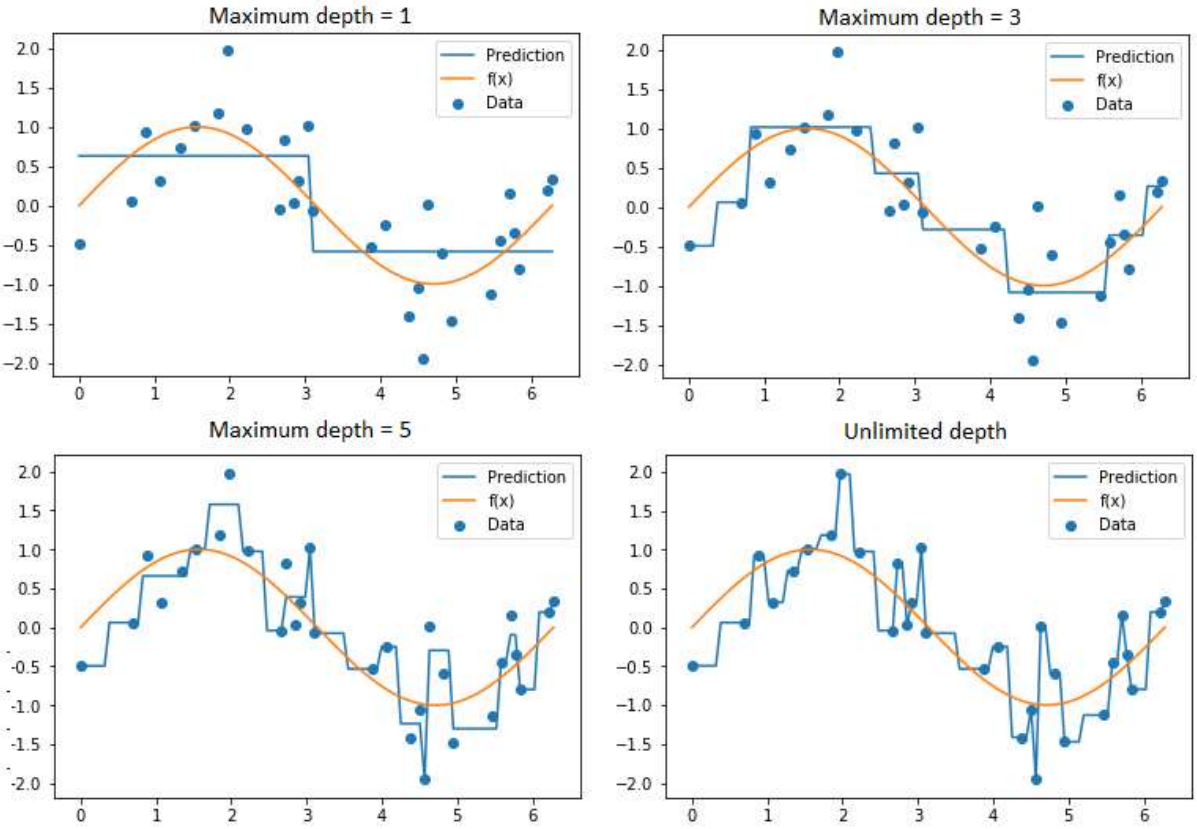


*Figure 8: Predictions of regression decision tree models of different depths*

The smooth curve is approximated by a stepwise constant function. Note, how with increasing depth the model starts to overfit the data.

Second, we will have a quick look at a simple, artificially created 2D binary classification problem. The data are generated as two interleaving half circles with some normally distributed noise added. This would be a difficult problem for some types of classifiers.

Figure 9 shows the results, two classes are represented by red and blue dots. The areas in which given classes are predicted are filled with respective colours.

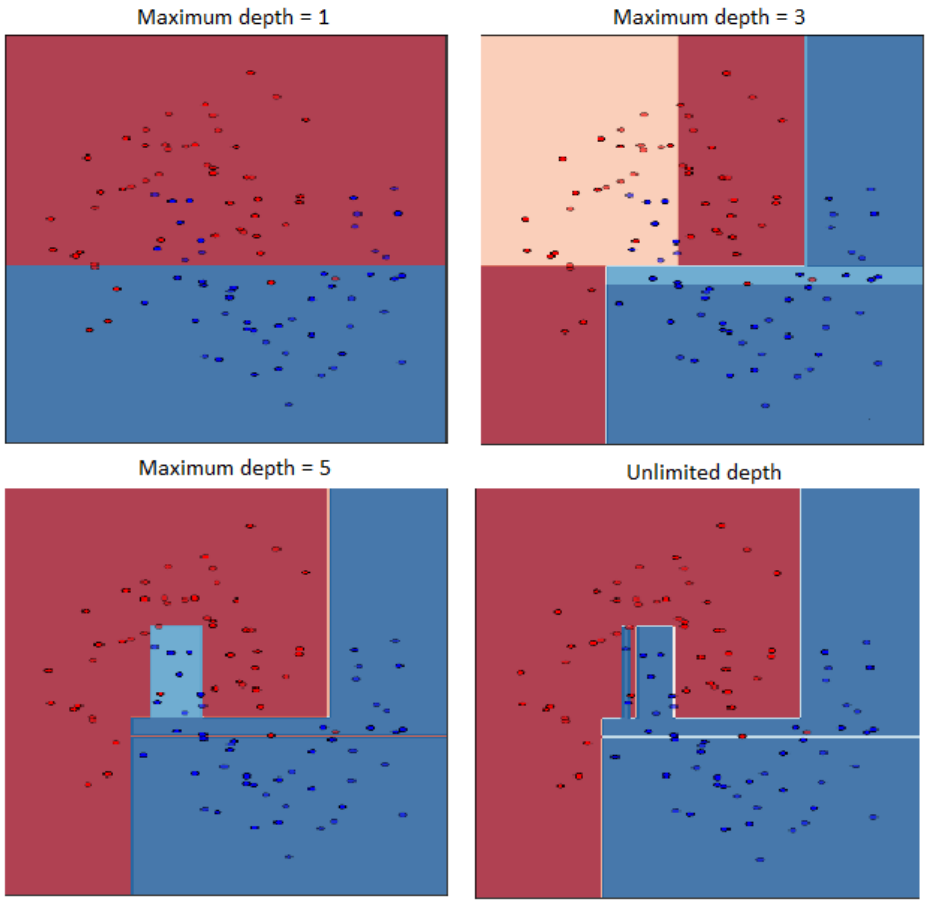


*Figure 9: Partitions made by decision tree classifiers of different depths*

Again we see the clear underfitting of the tree with maximum depth limited to 1 and signs of overfitting of the unrestricted model.

# 5 Ensembles of Decision Trees

## 5.1 Ensemble Machine Learning

The goal of ensemble machine learning is to enhance the predictive capability by combining more models into one more powerful one. The underlying models are called “base learners”, the ensemble is often referred to as a “strong learner”. Generally, the base learners within one ensemble can be of different types. Most common approaches, however, combine more models of the same type. Linear models and decision trees are often used as base learners. There are several ways in which the models can be combined. What comes next is a brief overview of the most common approaches.

- The principle behind **model averaging** is to combine models trained independently and average their predictions (or perform voting in case of classification) to achieve better accuracy and/or robustness.
- In **boosting**, on the other hand, the base learners are trained sequentially. Each successive model is trying to correct the errors made by previously built models.
- **Stacking** presents yet another common approach. Predictions made by the base learners are used as input variables to a meta-learner (sometimes in combination with some of the variables used to train the base learners), predictions of the meta-learner are used as predictions of the ensemble.

For the purpose of this thesis two examples of model averaging and two examples of boosting have been selected. In all four cases, the base learners are of the same type - CART.

## 5.2 Decision Trees as Base Learners

CART and some other decision tree learning algorithms are often used as base learners. They are not computationally expensive, so even large ensembles can be trained in reasonable time. The base learners are required to be either simple (low variance, high bias) or complex (low bias, high variance) depending on the ensemble technique used. By controlling the depth of the trees we can easily adjust the properties of decision trees to fit the given ensemble learning method. Decision trees can also be further adjusted easily to fit the specific needs of the ensembles. Decision trees can be used to both classification and regression, so the same ensemble techniques built upon them can also be used for both of these tasks. By combining trees into ensembles we lose their easy interpretation, other visualization techniques has been proposed to, at least partially, overcome this difficulty.

## 5.3 Model Averaging

Two ensemble learning methods based on model averaging will now be introduced: random forest and extremely randomized trees. Before diving into details of these particular methods let's first have a look on how model averaging methods work in the context of bias variance tradeoff.

### 5.3.1 Variance Reduction

The following models use averaging as means of reducing the variance. Recall that if we have $B$ independent and identically distributed variables with variance $\sigma^2$ the variance of their average equals to:

$$Var(avg) = \frac{1}{B}\,\sigma^2.$$

In case the variables are not independent we get:

$$Var(avg) = \rho\sigma^2 + \frac{1-\rho}{B}\sigma^2.$$

Where $\rho$ is pairwise correlation. In the context of model averaging, the random variables represent the predictions of base learners. These are identically distributed as they are of the same type (CART) trained using the same dataset. We can see that by adding more learners we can eliminate the second term. The correlation among the base learners limits the benefits of averaging.

### 5.3.2 Random Forest

Random forest is one of the most widely used supervised machine learning algorithms. It shows very high predictive power in many areas. It also scores high in many studies comparing the performance of different learning methods[20]. The algorithm proceeds as follows[21].

1. For b = 1, 2, …, $B$:
    a. Draw $N$ observations from the original dataset with replacement.
    b. Train a randomized CART using the samples drawn in previous step.
    c. Store the trained tree.
2. Output the trees.

*Algorithm 3: Random forest*

Where $B$ is the number of base learners to be trained and $N$ is the number of samples in the training dataset.

[20] Stefan Lessmann, Bart Baesens, Hsin-Vonn Seow and Lyn C. Thomas (2015). European Journal of Operational Research, 2015, vol. 247, issue 1, 124-136

[21] L. Breiman, and A. Cutler, *Random Forests*, Available at http://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm

#### *5.3.2.1 Randomized CART*

Random forests use a special version of decision trees. These differ from those introduced in Chapter 5 in only one aspect. At each node when selecting the next split only a subset of $p$ input variables is considered. Typical values of $p$ are $\sqrt{d}$ or $\log_2 d$, where $d$ is the number of input variables in the training set.

#### *5.3.2.2 Hyper-parameters of Random Forest*

The user needs to specify the number of base learners to be trained ($B$), the value of $p$ (in absolute terms or, more often, as a function of $d$) and the parameters of the trees: the stopping conditions and the metric used to evaluate the potential cutting points when making splits.

As explained earlier, the idea behind model averaging is to reduce the variance of the resulting ensemble as compared to single base learners. Therefore, we need an ensemble of very deep decision trees with low bias and high variance (which is lowered by averaging). To de-correlate the trees random forests use two sources of randomness. Each tree uses a different input data (drawn from the same empirical distribution) and a different subset of input variables is used on each split.

Generally, the higher the $B$ the better the model's performance as increasing $B$ does not lead to overfitting. When choosing $B$ we want to find a value that results in good performance of the model while still being computationally feasible (naturally, the more decision trees we have in the ensemble the more computationally expensive the task becomes).

To avoid overfitting the complexity of base learners can be bound. In spite of that, unlimited depth is used in most cases, on same occasions limited (but still relatively deep) trees show better performance[22].

#### *5.3.2.3 Prediction*

When doing classification majority vote among the trees in the ensemble is performed. In regression settings the final prediction is simply the average of predictions of the base

[22] Segal, M. (2004). *Machine learning benchmarks and random forest regression*, Technical report, eScholarship Repository, University of California. Available at http://repositories.edlib.org/cbmb/bench rf regn.

learners. In case of probabilistic classification the probabilities predicted by the base learners can be averaged.

#### *5.3.2.4 Special cases*

Certain settings of parameters lead to special cases that are described in literature as distinct machine learning models.

##### *5.3.2.4.1 Bagging*

When $p = d$ we get the bootstrap aggregating algorithm (commonly shortened to bagging). It is an ensemble machine learning method that can be used to combine base learners of any type. Decision trees are a natural choice thanks to their properties. It usually shows lower predictive capability as the trees are more similar to each other so there is lower reduction in variance.

##### *5.3.2.4.2 Random Subspace Method*

If we, on the other hand, disable the random resampling (the first step before training the randomized CART) we get to the (local) random subspace method (also called attribute bagging or feature bagging), one of the predecessors of random forests. As with bagging random subspace method usually underperforms random forest because of higher correlation among the underlying trees.

#### *5.3.2.5 Random Forest: Summary*

Random forests present probably the best known ensemble machine learning method. This is because they come with many advantages.

- They tend to be very **accurate**.
- They are still more **computationally feasible** than some other advanced machine learning models (such as neural networks).
- They are relatively **robust to overfitting** and **outliers**.
- They are a **standard part of** many **data analytics tools** (both commercial and open source).

Although complex in nature, random forests are relatively easy to use. There are not as many hyper-parameters to be tuned as when using some other supervised learning methods. They

are quite robust to outliers as these gets isolated in small "boxes" in the feature space[23] and only some of the base learners are trained using these observations. Random forests are also robust to overfitting, when necessary overfitting can be controlled by tuning a single parameter – the depth of the trees. As always, there are some limitations as well.

- On some problems, random forests **get outperformed by other approaches**.
- They are **more computationally demanding** then standalone models.
- They **lose the natural visual interpretation** of single decision trees.

### 5.3.3 Extremely Randomized Trees

Extremely randomized trees (ExtraTrees) are based on the same principles as random forest. They also belong to the model averaging family of ensemble machine learning models and use decision trees as weak learners. They differ in the sources of randomness used to de-correlate the individual trees in the ensemble.

The ensemble is composed of $B$ extremely randomized CART fitted to the same data (no resampling in this case).

#### *5.3.3.1 Extremely Randomized CART*

As with randomized trees used in random forest, only a subset of $p$ input variables is considered at each node. Moreover, for each of these variables a cutting point is selected at random. These potential splits are then evaluated using a specified metric (e.g. Gini index, Information gain, MSE).

The rationale behind this approach, according to the authors, is to take full advantage of model averaging by further randomization of the base learners[24].

#### *5.3.3.2 Hyper-parameters of Extra Trees*

In general, ExtraTrees have the same hyper-parameters as random forest. The authors, however, propose certain settings as default and claim that these perform well in most cases. The depth of the trees is expected to be unlimited. To avoid overfitting the authors suggest setting a minimum number of samples in node. The parameter $p$ determines the strength of

[23] In fact, there is a random tree-based unsupervised learning algorithm used to detect anomalies in the data based on this idea. It is called isolation forest. For more details see: Liu, Fei Tony, Ting, Kai Ming and Zhou, Zhi-Hua. "Isolation forest." Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on.

[24] P. Geurts, D. Ernst., and L. Wehenkel, *Extremely randomized trees*, Machine Learning, 63(1), 3-42, 2006.

the variable selection process (the degree of randomization), recommended values are $\sqrt{d}$ for classification and $p = d$ for regression[25]. The number of trees in the ensemble $B$ regulates the strength of the variance reduction of the ensemble model aggregation. Same logic as with random forest applies, the higher the $B$ the stronger the reduction in variance. Although, after certain number of trees the marginal gain can become negligible. The goal is, again, to find a value that results in good model while maintaining computational feasibility.

#### *5.3.3.3 Special Case: Totally Random Trees*

When we set $p = 1$ we get an ensemble of decision trees in which all splits are completely independent of the output variable. The predictive power of such an ensemble would, naturally, be poor unless there are very many trees in the ensemble. Totally random trees can, however, be used in anomaly detection. A technique called isolation forest (mentioned earlier in a footnote) uses the average number of splits needed to separate an observation from all the others as a measure of normality[26].

#### *5.3.3.4 Extra Trees: Summary*

Basically, extremely randomized trees can be thought of as a variant of random forest. Therefore, it has very similar advantages and limitations. The authors claim that ExtraTrees perform better than random forest on many problems. ExtraTrees are computationally less demanding as there are less splits to be evaluated. It is a newer and less known machine learning method and is not implemented in many analytics tools.

## 5.4 BOOSTING

Boosting algorithms are a family of ensemble machine learning methods that primarily focuses on reducing the bias (it has been shown that boosting also reduces variance[27]). Unlike in model averaging, the base learners within these ensembles are very simple. They are often referred to as weak learners and, in themselves, perform only slightly better than random guessing. The goal is to combine such weak learners to produce a strong ensemble model.

---

[25] P. Geurts, D. Ernst., and L. Wehenkel, *Extremely randomized trees*, Machine Learning, 63(1), 3-42, 2006.
[26] Liu, Fei Tony, Ting, Kai Ming and Zhou, Zhi-Hua. "Isolation forest." Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on.
[26] P. Geurts, D. Ernst., and L. Wehenkel, *Extremely randomized trees*, Machine Learning, 63(1), 3-42, 2006.
[27] Leo Breiman (1996). *BIAS, VARIANCE, AND ARCING CLASSIFIERS,* TECHNICAL REPORT. Available at http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.115.7931&rep=rep1&type=pdf

### 5.4.1 Additive Modelling

Boosting is a way of fitting an additive model in the form:

$$F(X) = \sum_{m=1}^{M} \beta_m b(x; \gamma_m).$$

Where $b(x; \gamma)$ is a simple basis function of multivariate input $x$ and parameters $\gamma$ and $\beta_m$ is the weight of m-th basis function in the additive model. These models are usually fit by minimizing a loss function $L$ averaged over $N$ training data points[28],

$$\min_{\beta,\gamma} \sum_{i=1}^{N} L\big(y_i, \beta b(x_i, \gamma)\big).$$

Solving such problems can be a computationally intensive task. Forward stagewise additive modelling approximates the solution by sequentially adding new basis functions to the model without adjusting the parameters and weights of those that have already been added[29].

Both of the boosting algorithms used in this thesis are examples of forward stagewise modelling and use simple CART models as basis functions (parameter $\gamma$ refers to the splitting variables and cutting points).

### 5.4.2 AdaBoost

Adaptive boosting, commonly shortened to AdaBoost, is an ensemble machine learning algorithm developed by Yoav Freund and Robert Schapire, who won the 2003 Gödel Prize for their work. It was first introduced in the context of binary classification, but it can be generalized to multiple classification and regression as well. It can be used with weak learners of any type, decision trees are the most common choice.

AdaBoost is adaptive in the sense that at each stage it focuses on samples that has been mislabelled by previous weak learners. This is done by reweighting the samples at each iteration. The algorithm proceeds as follows[30], assuming binary classification where $y \in \{-1, 1\}$.

---

[28] Hastie, Trevor; Tibshirani, Robert; Friedman, Jerome (2008). *The Elements of Statistical Learning (2nd ed.)*. Springer. ISBN 0-387-95284-5

[29] Hastie, Trevor; Tibshirani, Robert; Friedman, Jerome (2008). *The Elements of Statistical Learning (2nd ed.)*. Springer. ISBN 0-387-95284-5

[30] Robert E. Schapire, *Explaining AdaBoost*, Available at http://rob.schapire.net/papers/explaining-adaboost.pdf

1. Initialize the observation weights:

$$w_1(i) = \frac{1}{N}, i = 1, 2, \dots, N.$$

2. For m = 1, 2, …, M:
    a. Train a weak learner $h_m$ using $w_m$.
    b. Compute the in-sample weighted error $\varepsilon_m$:

$$\varepsilon_m = Pr_{i \sim w_m}[h_m(x_i) \neq y_i].$$

    c. Set the coefficient $\alpha_m$ to:

$$\alpha_m = \frac{1}{2}\ln\left(\frac{1 - \varepsilon_m}{\varepsilon_m}\right).$$

    d. Update the model:

$$F_m(x) = F_{m-1}(x) + \alpha_m h_m(x).$$

    e. For $i = 1, 2, \dots, N$ update the weights:

$$w_{m+1} = \frac{w_m(i)\exp\left(-\alpha_m y_i h_m(x_i)\right)}{Z_m}.$$

    Where $Z_m$ is a normalization factor chosen so that $w_m$ is a valid probability distribution.

3. Output the model $F_M(x)$.

*Algorithm 4: AdaBoost*

The predictions are simply computed as the sign of the weighted sum of the weak learners' predictions,

$$F(x) = sign\left(\sum_{m=1}^{M} \alpha_m h_m(x)\right).$$

In case of probabilistic prediction, the class probabilities of an input sample are computed as the weighted mean predicted class probabilities of the classifiers in the ensemble.

#### *5.4.2.1 Exponential Loss Function*

Many machine learning and statistical methods can be viewed as procedures for minimizing a loss function (also called cost function or objective function). Although not originally derived this way, AdaBoost can be shown to minimize the exponential loss function,

$$\frac{1}{N}\sum_{i=1}^{N} exp\big(y_i F(x_i)\big).$$

This is done in a greedy fashion as outlined in the section on additive modelling. It is easy to see that by minimizing this function we strongly penalize the misclassified cases (cases where the sings of $y_i$ and $H(x_i)$ does not match). AdaBoost can be viewed as a form of functional gradient descent, as observed by Mason et al.[31]. This understanding has led to the generalization of boosting to a wide range of other loss functions and learning problems, such as multiclass classification or regression[32].

#### *5.4.2.2 Hyper-parameters of AdaBoost*

As usual in ensemble learning, the user has to specify the maximum number of base learners to be trained. In case of prefect classification the algorithm stops prematurely. Unlike in previous cases with increasing number of base learners in the ensemble the model starts to overfit the training data. One way of avoiding overfitting is to specify a learning rate. The contribution of each base learner is then shrank by this proportion. There is a natural trade-off between learning rate and the number of base learners.

Parameters of the base learners can also be specified. When using CART as base learners, maximum depth is usually set to 1. Such trees are sometimes called decision stumps and represent the simplest form of decision trees with only one split.

#### *5.4.2.3 AdaBoost: Summary*

AdaBoost is a classic boosting algorithm. It has several major advantages.

- AdaBoost tend to be **very accurate** on many problems.
- It is **easy to use** as there are not many hyper-parameters to be tuned.
- AdaBoost is **relatively fast** as the base learners are usually very simple.

As always there are certain limitations.

- AdaBoost **is not very robust to overfitting** and, unlike in random forest, the number of weak learners needs to be chosen carefully.

[31] Mason, L., Baxter, J., Bartlett, P., Frean, M.: *Functional gradient techniques for combining hypotheses*. In: Advances in Large Margin Classifiers. MIT Press (2000)
[32] Robert E. Schapire, *Explaining AdaBoost*, Available at http://rob.schapire.net/papers/explaining-adaboost.pdf

- AdaBoost is **sensitive to outliers** as it focuses primarily on hard-to-label samples.

Many alternative boosting methods has been proposed since the introduction of AdaBoost. These methods are developed to minimize other loss functions (e.g. LogitBoost) or to be more robust (e.g. BrownBoost).

### 5.4.3 Gradient Boosting

Gradient boosting is a very popular ensemble machine learning method. It proved to preform extremely well on many different data sets. In recent years gradient boosting based solutions have won many machine learning competitions[33].

Gradient boosting is based on the idea that boosting can be interpreted as a loss function minimization problem. It can be seen as a generalization of AdaBoost that can be used to minimize an arbitrary differentiable loss function. Generally, gradient boosting can be performed using weak learners of any type, decision trees are, again, the most common choice.

Similarly to AdaBoost gradient boosting updates the model iteratively in a greedy fashion. The way each weak learner is constructed is, however, very different. The logic behind gradient boosting is best explained in the context of least-squares regression. At each iteration $m, 1 \leq m \leq M$ what we have is an imperfect model $F_m$ and want to improve this model by adding a new weak learner $h$ such that:

$$F_{m+1}(x) = F_m(x) + h(x).$$

Gradient boosting solution to this problem is based on the observation that perfect choice of $h$ would mean:

$$F_{m+1}(x) = F_m(x) + h(x) = y,$$

$$h(x) = y - F_m(x).$$

Therefore at each stage gradient boosting fits a regression model with the vector of residuals $y - F_m(x)$ as the output variable. We can generalize this idea to loss functions other than $MSE$ and to classification problems as well. This follows from the fact residuals $y - F_m(x)$

[33] D. (2018, August 07). Dmlc/xgboost. Retrieved from https://github.com/dmlc/xgboost/tree/master/demo#machine-learning-challenge-winning-solutions

are the negative gradients of the loss function $\frac{1}{2}\left(y - F_m(x)\right)^2$ with respect to $F_m(x)$. When using different loss functions at each stage a regression model is fit to the so called pseudo-residuals computed as the negative gradient of the respective loss function. The general algorithm proceeds as follows[34].

1. Initialize a model with constant value:

$$F_0(x) = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{N} L(y_i, \gamma).$$

2. For m = 1, 2, …, M:
   a. Compute the pseudo-residuals:

$$r_{mi} = -\left[\frac{\partial L(y_i, F_{m-1}(x_i))}{\partial F_{m-1}(x_i)}\right] \text{ for } i = 1, 2, \ldots, N.$$

   b. Fit a base learner $h_m(x)$ to the pseudo-residuals (using original input variables as inputs and vector of pseudo-residuals as the output variable)
   c. Compute the multiplier $\gamma_m$:

$$\gamma_m = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{N} L\left(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)\right).$$

   d. Update the model:

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x_i).$$

3. Output the model $F_M(x)$.

*Algorithm 5: Gradient boosting*

Predictions are, again, the weighted sum of base learners' outputs.

#### *5.4.3.1 TreeBoost*

As mentioned above gradient boosting is typically used with fixed size decision trees as weak learners. In this case Friedman proposes a modification to the general algorithm. At each stage $m$ a regression tree is fit to the pseudo-residuals giving $J$ hyperrectangular terminal regions $R_{jm}$ (as explained in the section on geometric interpretation of decision trees). An optimal

[34] Friedman, J. H. *Greedy Function Approximation: A Gradient Boosting Machine*. (February 1999) Available at https://statweb.stanford.edu/~jhf/ftp/trebst.pdf

value of $\gamma_{jm}$ is chosen for each terminal region $j = 1, 2, \ldots J$ instead of having one single $\gamma_m$ for the whole tree. This is done by solving:

$$\gamma_{jm} = \underset{\gamma}{\operatorname{argmin}} \sum_{x_i \in R_{jm}} L(y_i, F_{m-1}(x) + \gamma).$$

And so the model is updated as:

$$F_m(x) = F_{m-1}(x) + \sum_{j=1}^{J} \gamma_{jm} I(x_i \in R_{jm}).$$

In his paper Friedman also derives specific gradient boosting algorithms for several different loss functions for regression as well as for both binary and multiclass classification[35].

#### *5.4.3.2 Hyper-parameters of Gradient Boosting*

When training a gradient boosting model we need to specify the loss function suitable for the problem at hand. As usual in ensemble learning we need to choose the number of base learners to be trained. Again a learning rate can be set to shrink the contributions of each weak learner.

We also have to specify the parameters of the base learners. Maximum depth (or, alternatively, maximum number of leaves) controls the level of interaction allowed among the input variables. Generally, we only want to model dominant interactions so we usually use very simple decision trees. According to Hastie et al.[36] experience indicates that $4 \leq J \leq 8$ works well for most problems. Moreover, the results are usually fairly insensitive to particular choices within this range.

There is also another modification of the general algorithm which is inspired by the bagging method already mentioned. At each iteration $m$ only a subset of the original training data (drawn at random without replacement) is supplied to the weak learner. The subsample size is defined as constant fraction $f$ of the original training set. When $f < 1$ we speak of stochastic gradient boosting. This modification often leads to significant improvement of the

[35] Friedman, J. H. "Greedy Function Approximation: A Gradient Boosting Machine." (February 1999) Available at https://statweb.stanford.edu/~jhf/ftp/trebst.pdf

[36] Hastie, Trevor; Tibshirani, Robert; Friedman, Jerome (2008). *The Elements of Statistical Learning (2nd ed.)*. Springer. ISBN 0-387-95284-5

results. Friedman observed that $0.5 \leq f \leq 0.8$ leads to good results for most data sets[37]. This modification also improves computational efficiency.

#### *5.4.3.3 Gradient Boosting: Summary*

Gradient boosting has become very popular among machine learning practitioners and is used in many different areas. Here are some of its main advantages.

- Gradient boosting **performs very well on many data sets**.
- It is **very flexible** as it can be used to minimize an arbitrary differentiable loss function. By selecting a proper loss function it can be made more robust than AdaBoost.
- It is **still easier to use** than some of the most complex machine learning models (e.g. artificial neural networks).

Naturally, gradient boosting has its own limitations.

- It is **more difficult to use** than other before mentioned ensemble methods as it has more hyper-parameters to tune.
- Gradient boosting is **more computationally demanding**.
- It is **less prone to overfitting** than random forest or ExtraTrees.

Gradient boosting gained its popularity thanks to a good balance of its complexity, computational efficiency and accuracy. It is much easier to use and faster to train than some of the modern (deep) learning techniques while often being competitive in terms of predictive capability.

## 5.5 Uncovering the Black Box

As I already mentioned, one of the major disadvantages of ensemble machine learning methods in general is that they lose the natural interpretation of standalone models. This has practical impact as companies in different industries might be obliged to explain the behaviour of their models to a regulator (banking presents such a sector). Such hard to understand models can also be difficult to deploy, debug, monitor etc. When dealing with single decision trees we can visualize them easily as shown in figures 5 and 6. These visualizations are very

[37] Friedman, J. H. *Stochastic Gradient Boosting*. (March 1999) Available at https://statweb.stanford.edu/~jhf/ftp/stobst.pdf

intuitive and their interpretation is straightforward. If we work with ensembles than can easily contain hundreds of decision trees, things get much more complicated. Theoretically, we could visualize all the trees in the ensemble, but such visualization would be completely useless. It can also be tempting to have a look at a single tree in the ensemble (or a few of them) just to get some basic idea what is going on inside. For boosting models this makes no sense as the trees are trained sequentially and cannot be assessed as individual models. In random forest and extra trees the base models are trained in parallel and are independent of each other but all the splits are randomized so such a picture also does not say much.

### 5.5.1 Variable Importance

One simple and very practical approach to get at least some basic idea what is going on inside the ensemble models is to look at the variable importance. These could be calculated as follows[38]: every time a split is made on a particular input variable the resulting nodes has lower impurities than the parent node, by adding the decreases for each individual variable over all the trees in the ensemble (measured by a selected metric: Gini index, IG etc.) we get a simple estimate of importance of individual variables. Since absolute numbers have no meaningful interpretation we can rather plot relative variable importance in the form of a simple bar chart as demonstrated bellow.

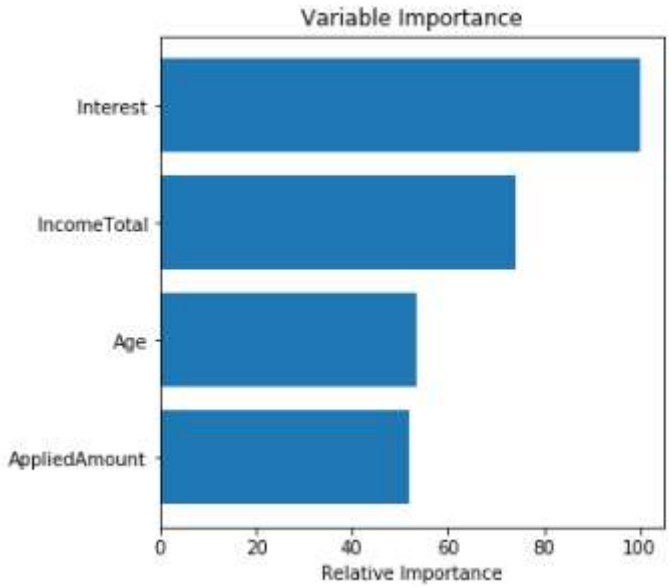


*Figure 10: Relative variable importance of a very simple gradient boosting model*

[38] L. Breiman, and A. Cutler, *Random Forests*, http://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm

When using random forest model we can use a little more intricate version of this. This has to do with an interesting feature of random forest (or eventually other models using random resampling with replacement) that is called out-of-bag (OOB) samples. These are simply the observations left out when training particular base learner due to resampling. When the $b$th tree in the ensemble is trained, the OOB samples can be used to measure this base learner's out-of-sample accuracy. After doing so the values of a particular variable are randomly permuted and the accuracy is measured again on the OOB samples using this permuted variable instead of the original one. The decrease in accuracy as a result of this permuting is averaged over all trees, and is used as a measure of the importance of this particular variable in the model[39]. Note, that permuting the variable is not the same as leaving it out completely. If the model was re-trained without this variable, other variables could compensate for this loss. We can visualize this the same way as in Figure 10.

### 5.5.2 Partial Dependence Plots

Identifying important variables does usually not suffice to truly understand how the model works. We are interested in how the (most important) variables affect the output variable and ideally how they interact among themselves. This is by no means a simple task and there are unfortunately no straightforward solutions to this problem. One way we can address this problem is to use partial dependence plots. These could be used with any black box model. Partial dependence function is defined as:

$$f_S(X_S) = E_{X_C}[f(X_S, X_C)].$$

Where $f(X)$ is a general function of all the inputs, $X_S$ is the set of variables for which the partial dependence function should be plotted and $X_C$ are the other variables that were used when training the model. Partial dependence functions can be estimated as[40]:

$$\hat{f}_S(X_S) = \frac{1}{N}\sum_{i=1}^{N} f(X_S, x_{iC}).$$

[39] Hastie, Trevor; Tibshirani, Robert; Friedman, Jerome (2008). *The Elements of Statistical Learning (2nd ed.)*. Springer. ISBN 0-387-95284-5.

[40] Christoph Molnar (2018), *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable,* https://christophm.github.io/interpretable-ml-book/

Where $\{x_{1C,}x_{2C}, \dots, x_{NC}\}$ are the values of $X_C$ occurring in the training data. This can be a computationally very expensive task, with tree-based models we can, fortunately, perform this computation in a relatively efficient way.

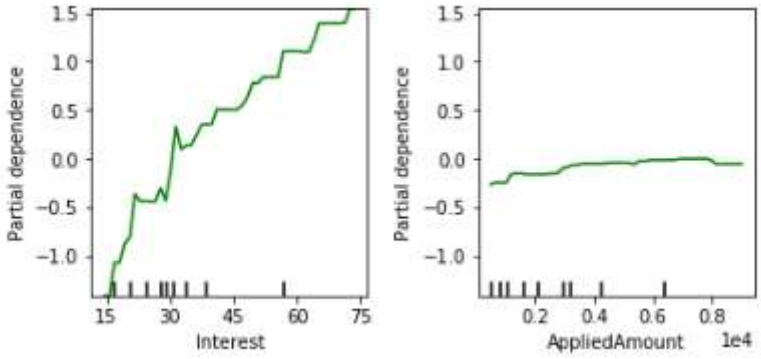


*Figure 11: Two univariate partial dependence functions of the same simple GB model (log B/G odds on y-axis)*

Naturally, we are limited only to low-dimensional views. In Figure 11 we can see estimates of two partial dependence functions with only one explanatory variable, in Figure 12 we see a partial dependence function of both of these variables.

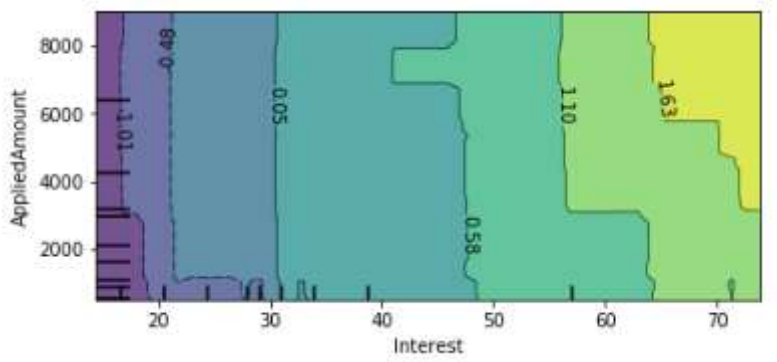


*Figure 12: Partial dependence function of two inputs (same model as before)*

There are other methods of understanding black box models such as those based on simulating the prediction surface. All of these are indirect and often computationally demanding, complex nature of these models present a great difficulty and is probably among the main reasons why simpler models (e.g. logistic regression) are used instead.

# 6 CREDIT SCORING

In this chapter I am going to introduce the reader to the main concepts of credit risk management with the focus on credit rating and scoring. After explaining the basic concepts and showing why these are important, I will describe the standard approach to credit scoring function development. This generally accepted approach will later serve as a benchmark with which each of the ensemble models is going to be compared.

## 6.1 CREDIT RISK MANAGEMENT

Lending money has always been a risky business, the borrowers can cease to make the scheduled payments for a myriad of reasons. These are usually related to financial distress of the borrower, but some borrowers can also exhibit fraudulent behaviour and be simply unwilling to repay the loan. When approving loans, banks naturally want to be able to discriminate between good and bad debtors. But that is not the end of the process, after the loan is granted banks need to monitor it and communicate with the borrower, in particular when the borrower defaults on the loan (e.g. misses a payment). The riskiness of the loan also needs to be regularly reassessed. Based on this riskiness (and other parameters as briefly explained below) the banks estimate the expected losses which is an important regulatory requirement.

As outlined above, managing credit risk is by no means an easy task. Traditionally, credit risk management was based on experience and expertise of the lenders and most loans were assessed by human judgement. In modern days much of this process is automated thanks to the advancements in mathematical and statistical modelling as well as thanks to the availability of efficient computational tools. Without such advancements banks would hardly be able to provide loans at today's scale, this is especially true for large portfolios of retail loans.

## 6.2 RATING AND SCORING SYSTEMS

To express the riskiness of a given credit we use a score or a rating scale. Scores are continuous variables that can sometimes be interpreted as probabilities. For practical purposes scores are usually standardized (rounded, multiplied by a constant etc.) and are therefore expressed on

a fine but finite scale, most often ranging from 1 to 1000[41]. Ratings use discrete scales of approximately 7 to 25 grades. Rating grades are usually denoted by letters or sometimes numbers. Well-known examples of scales are those of the external rating agencies, e.g., Standard & Poor's, starting with the worst non-default rating: C, and going up to the best rating AAA, or the very similar one of Moody's, starting with C and going up to the best Aaa[42]. The quality of rating systems can be assessed by comparing observed rates of default with the expected rates within the grades. In case the rating system is built in such a way that there are no expected rates of default assigned to the rating grades, we can assess the basic capability of the system based on simple logic: the observed rate of default within the grades should be monotonically increasing from the best grade to the worst. Alternatively, methods based on the discriminatory power of the systems can be used, these are explained in a later section on model evaluation.

## 6.3 Probability of Default

Estimated probability that the borrower will default during a given time horizon (PD) is often used as the score. This is a very practical approach as such score is interpretable and can be used as an input to other credit risk models. When modelling PDs we need to beforehand precisely define[43]:

- the time horizon (usually, we model one year default, but we might be interested in different time frames),
- what do we mean by default (legally proclaimed bankruptcy, any delay in payments, delay in payments exceeding a given number of days etc.),
- whether we are scoring the debtor or only a certain facility (e.g. bond issue),
- whether we model the current riskiness or riskiness conditional on the new loan,
- if we are interested in Through the Cycle (TTC) or Point in Time (PIT) score.

PIT score takes into account current economic conditions and is dependent on the business cycle, this kind of score is desirable from the business point of view. TTC score should be independent of the business cycle and is more important for the regulator. PDs can be

[41,42,43] WITZANY, J. Credit risk management : pricing, measurement, and modeling. Cham: Springer, 2017. ISBN 978-3-319-49799-0.

estimated using various statistical or machine learning techniques. Note, that in this approach we completely ignore the default dynamics during the selected time horizon, methods of survival analysis can be used to address this problem.

Probability of default is important not only to discriminate between good and bad debtors, it also plays important role when pricing the loan. The loan's interest rate can be decomposed as follows:

$$Loan\ interest\ rate = internal\ cost\ of\ funds + administrative\ cost$$

$$+ risk\ premium + profit\ margin$$

Risk premium (RP) covers the losses suffered because of the debtors that default on their loans:

$$RP = \frac{EL}{1 - PD}, EL = PD * LGD.$$

Where $EL$ is relative expected loss, $LGD$ stands for Loss Given Default defined as:

$$LGD = E[L \mid Default].$$

It is expected relative loss conditional on default occurring ($L$ refers to relative loss). These equations can be further refined by taking into account the stochastic nature of time to default, time value of money and possibly other factors.

## 6.4 Standard Approach to PD Estimation: Logistic Regression

Logistic regression (LR, logit) is a linear classifier that is usually applied to a binary target variable (generalization that allow us to apply LR to multiclass problems is available as well). In the context of credit scoring, logistic regression can be explained as follows[44]. What we are trying to model is a future latent credit score $y^*$such that:

$$y_i^* = \beta' * x_i + \varepsilon_i.$$

[44] WITZANY, J. Credit risk management : pricing, measurement, and modeling. Cham: Springer, 2017. ISBN 978-3-319-49799-0.

Where $x$ is a vector of explanatory variables (including a constant), $\beta$ is a vector of associated weights and $\varepsilon$ is a random error. Since $\varepsilon$ is not observable $y^*$ is not directly observable as well, thus the word latent. What we can observe and therefore use to estimate $\beta$ are explanatory variables ($x$) and a binary variable $y$ indicating whether the default occurred or not.

$$y_i = \begin{cases} 1 \; if \; y_i^* < 0 \; (default \; occured) \\ else \; 0 \; (default \; did \; not \; occur) \end{cases}$$

The probability of default conditional on observed values of explanatory variables can be expressed as:

$$PD_i = P[y_i = 1 \mid x_i] = P[\beta' * x_i + \varepsilon_i \leq 0] = F_i(\beta' * x_i).$$

Where $F$ is the cumulative distribution function (CDF) of the random variable $\varepsilon$. In case we assume $\varepsilon$ to be normally distributed we get the probit model[45]. If the residual $\varepsilon$ follows logistic distribution:

$$F(x) = \frac{e^x}{1 + e^x} = \frac{1}{1 + e^{-x}},$$

we call the model logistic regression. Values of $\beta$ are obviously unknown, to find $b$, an estimate of $\beta$, Maximum Likelihood Estimation (MLE) method is usually used[46]. Assuming the observations are independent, the total likelihood function ($L$) of all the observations given vector $b$ can be expressed as:

$$L(b) = \prod_i [F(-b' * x_i)^{y_i} * (1 - F(-b' * x_i))^{1-y_i}].$$

For numerical reasons, what actually gets maximized in practice is the log-likelihood function.

$$\ln L(b) = l(b) = \sum_i [y_i * \ln F(-b' * x_i) + (1 - y_i) * \ln(1 - F(-b' * x_i))].$$

The maximum is computed numerically using Newton-Raphson's algorithm. Logistic regression model comes with several important advantages. It is computationally feasible and easily interpretable. The interpretability of the coefficients is based on the fact that:

[45] Greene. W. H. (2003). *Econometric Analysis (5th ed),* Upper Saddle River, New Jersey 07458

[46] Alternatively, we can purposefully introduce bias to the estimation via l1 or l2 penalty to fight overfitting.

$$\frac{PD_i}{1 - PD_i} = e^{-\beta' * x_i}, \ln\frac{B}{G} = \ln\frac{1 - PD_i}{PD_i} = \beta' * x_i.$$

The coefficients $\beta$ therefore express the impact of the respective inputs on the log odds $\ln\frac{B}{G}$. Another useful property of the coefficients is that these are asymptotically normally distributed which allows us statistically test their significance. We can also easily get confidence interval for the true coefficient $\beta_i$ given its estimate $b_i$. There are also other statistics based on the likelihood function that can be used to assess the quality of the model.

Logistic regression comes with important drawbacks as well. As mentioned at the beginning of this section, LR is a linear classifier. It assumes a linear relationship between input variables and the log odds. The model's inputs should also show little or no multicollinearity, no interactions among the inputs are modelled unless specially designed variables representing these interactions are added to the model (for example the product of two inputs).

## 6.5 Scoring Function Development

In this section I am going to describe the process of developing a scoring function. While some of the points are very general, I will mostly focus on the standard approach using LR.

### 6.5.1 Categorical Variables

Most algorithms, including LR and the earlier introduced tree-based ensemble models, require the input variables to be numeric. Categorical variables must be encoded into numerical values. Most common approach is to use dummy variables (in the machine learning community also called one-hot-encoding) a set of binary variables indicating which category is present. The concept is demonstrated bellow using first five rows of a fictive dataset.

| | Original variable | Dummy variables | | |
|---|---|---|---|---|
| **Customer id** | **Gender** | **Gender_M** | **Gender_F** | **Gender_U** |
| 4569 | Male | 1 | 0 | 0 |
| 1353 | Unknown | 0 | 0 | 1 |
| 5462 | Female | 0 | 1 | 0 |
| 4984 | Female | 0 | 1 | 0 |
| 4698 | Male | 1 | 0 | 0 |

*Table 1: Dummy variables example*

In machine learning we usually create $n$ dummy variables, one for each category (as in the table above), traditionally when using LR, we create only $n - 1$ dummies. Alternatively, we

can encode the categories with their respective Weight of Evidence (WoE) values. WoE value for a given category $c$ can be estimated as[47]:

$$WoE(c) = \ln P[c \mid Good] - \ln P[c \mid Bad].$$

Where $P[c \mid Good]$ is estimated as relative frequency of category $c$ among the good loans, $P[c \mid Bad]$ is estimated analogously. The concept of WoE comes also handy when reducing the cardinality of categorical variables or during correlation analysis.

Continuous variables sometimes need to be addressed as well, especially when using linear models, it might be useful to propose such transformations as to make variables with non-linear relationship with the output variable linear.

### 6.5.2 Variable Selection

An important step when developing any kind of model is to select the optimal subset of input variables from the list of all potential predictors at hand. Generally, different algorithms require different approaches to variable selection as they differ in how they can handle irrelevant inputs, multicollinearity or high dimensionality. We also want the model as simple as possible a principle often referred to as Occam's razor.

When using logistic regression we usually start by preselecting a short list (approx. 20 – 30 variables) of predictors from all the potential ones. This can be done by measuring their predictive power using a simple LR model with just one input or by their Information Value (IV) a metric based on WoE:

$$IV = \sum_{c=1}^{C} WoE(c) * (P[c \mid Good] - P[c \mid Bad]).$$

Where $C$ is the number of categories of given predictor. Variables from the short list are further analysed and transformed. Special treatment is given to categorical variables, categories with similar riskiness (measured for example by their WoE values) are merged together. This results in simpler and more stable models.

It should be obvious from the previous section that the highest value of likelihood function is achieved when using all the available input variables. As explained earlier our goal is to

[47] WITZANY, J. Credit risk management : pricing, measurement, and modeling. Cham: Springer, 2017. ISBN 978-3-319-49799-0.

develop a model that can generalize well and can be used to score previously unseen data point with reasonable accuracy. To achieve that we need to select only subset of the variables from the short list, 7 to 15 inputs are usually selected for the final model. Since LR assumes there is no multicollinearity among the predictors correlation matrix of all the potential inputs is often calculated and variables that show high correlation (e.g. 60%>) with other variables are eliminated, potential predictor with lower univariate predictive power or lower IV is left out.

When using all the available inputs, many coefficients often becomes statistically insignificant. In other words, we end up using inputs were we are not even sure which sign the coefficient has. This can obviously cause systematic error when scoring new observations that were not used during training. Therefore we want all the coefficient to be statistically significant. Moreover, the values of the coefficient should not contradict our business understanding of the problem. For example having higher income should not increase a borrower's riskiness etc. Statistical significance of individual input variables can be tested using Wald statistic (W):

$$W = \frac{b_j}{s.e.(b_j)}.$$

Where $s.e.(b_j)$ is the standard error of the tested coefficient $b_j$. All coefficient should be significant on some reasonable probability level (e.g. 90%, 95% or even 99%).

*Predictors* = set of all the available predictors

1) Until a stopping criterion is met:
    a) Fit a model with *Predictors* as inputs
    b) Select the input with lowest predictive power
    c) Remove the selected predictor from *Predictors*

*Algorithm 6: Backward variable selection*

There are multiple methods, which can be used to find the optimal final subset of the inputs. One could try the brute force approach of fitting a model for each of the combination of variables and simply select the best one. This would be a computationally very demanding task. The final subset is usually found using a greedy algorithm. One way to do that is to perform backward selection.

The predictive power of inputs can be measured by their significance, alternatively we can iteratively re-fit the model leaving one input variable at time and compare the resulting models with the one using all input from $Predictors$. Statistics derived from the likelihood function can be used to assess the models in this case.

Forward selection is analogous to backward selection. It starts from an empty set of selected predictors and these are iteratively added to the model based on similar criteria. Stepwise selection presents a combination of both of these methods. Simply put, when more variables have been added to the model, the non-significant ones get eliminated. This way we explore more of the possible combinations of inputs at the price of higher computational cost.

Akaike information criterion (AIC) is one of the metrics that can be used to estimate the relative quality of models based on their likelihood. It is defined as follows[48]:

$$AIC = 2k - 2\ln L.$$

Where $k$ is the number of inputs of the model. AIC is based on information theory, it is an estimate of how much more (or less) information was lost when using one model or another as opposed to knowing the underlying function with certainty[49]. It penalizes models with higher number of inputs, lower value of AIC means a better model.

### 6.5.3 Model Validation and Performance Measurement

As emphasized repeatedly, we are interested mainly in performance of the model measured on data points that were not used during the training. When developing a scoring function a proportion of the data (usually 20% or 30%) is left out and used to measure the out-of-sample performance. Alternatively, a method called k-fold cross-validation (CV) can be used. The training set is randomly split into $k$ subsets. At each of $k$ iterations $k - 1$ subsets is used to train the model and one to measure its out-of-sample performance. Model with best average out-of-sample performance is considered the best. One advantage of CV, as opposed to simple train/test split, is that we not only get the estimate of expected performance (the average out-of-sample performance) but we also get the estimate of its variance. Ideally, we would like to estimate the expected performance conditional on the dataset we have at hand. This

[48, 49] Akaike, H. (1973), *Information theory and an extension of the maximum likelihood principle,* in Petrov, B. N.; Csáki, F., 2nd International Symposium on Information Theory, Tsahkadsor, Armenia, USSR, September 2-8, 1971, Budapest: Akadémiai Kiadó, pp. 267–281

can be achieved by choosing very high values of $k$ (the special case when $k$ equals the number of observations is referred to as leave-one-out cross-validation), the variance of such estimate is unfortunately very high. Typical values of $k$ range between 5 and 10[50].

There are many metrics that could potentially be used to measure the performance of the trained models. A set of simple metrics can be derived from the so called confusion matrix. It basically is a special kind of contingency table. Table 2 shows a general example of a confusion matrix for a binary classification problem. Each cell of the table contains the number of observations satisfying the given conditions.

| | | **Predicted value** | |
|---|---|---|---|
| | | **Positive** | **Negative** |
| **Actual value** | **Positive** | True positive (TP) | False negative (FN) |
| | **Negative** | False positive (FP) | True negative (TN) |

*Table 2: Confusion matrix for a binary classification problem*

Simple metrics derived from such a table are for example:

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)},$$

$$Sensitivity = \frac{TP}{(TP + FN)},$$

$$Specificity = \frac{TN}{(FP + TN)}$$

and many others. This comes with certain problems, first of all we generally only have data on loans that were approved and do not know whether the rejected applications would turn out to be good or bad, so these need to be estimated from the underlying probabilities[51]. Secondly, a threshold score to decide if the application is considered to be good or bad needs

[50] Hastie, Trevor; Tibshirani, Robert; Friedman, Jerome (2008). *The Elements of Statistical Learning (2nd ed.)*. Springer. ISBN 0-387-95284-5.
[51] WITZANY, J. Credit risk management : pricing, measurement, and modeling. Cham: Springer, 2017. ISBN 978-3-319-49799-0.

to be set. This is an optimization task involving not only the probabilities of errors occurring (FP, FN) but also the costs associated with these errors.

A more comprehensive metric is called Receiver Operating Characteristic (ROC) curve. It is a curve in the $[0,1] \times [0,1]$ rectangle that connects points on $x$ and $y$ axes defined as:

$$Hit\ rate\ (cumulative\ Sensitivity) = x_s = F(s|G) = P[score(X) \leq s \mid X\ is\ good],$$

$$False\ alarm\ rate\ (cumul.\ 1 - Specificity) = y_s = F(s|B) = P[score(Y) \leq s|Y\ is\ bad].$$

This curve gets aggregated into a single number by computing the area under the ROC curve (often simply called Area Under Curve, AUC). The values of AUC vary between 0 and 1, value of 0.5 is equivalent to random guessing, value of 1 represents a perfect model.

Very similar and in credit scoring often used metric is called Gini coefficient or Accuracy ratio (AR). It is defined in a very similar way as ROC. It actually can be proved that[52]:

$$AR = 2 * (AUC - 0.5).$$

In other words these two metrics contain the same information about the model's performance. I will be showing both of them most of the time just for the convenience of the reader.

### 6.5.4 Grid Search

Complex models, as those introduced in the chapter on ensemble machine learning, usually have quite a few hyper-parameters. These are parameters that cannot be learnt from the data directly and need to be set beforehand. Choosing the right hyper-parameters generally consist of iteratively trying to train the model with different settings and comparing the results. Grid search is a systematic and, in principle, very simple way of doing that. First, a range of acceptable values of each hyper-parameter is set, then for each of the possible combinations (points on the grid) a model is trained and validated (using k-fold cross-validation or simple train/test split), finally the highest performing model gets selected. This obviously is a very demanding task in terms of computational time. Two step approach is sometimes used to at least partially overcome this difficulty. First a rough grid is used to identify the best performing area on the grid. Second round of grid search within the best performing region is then run

[52] WITZANY, J. Credit risk management : pricing, measurement, and modeling. Cham: Springer, 2017. ISBN 978-3-319-49799-0.

using a finer grid. Alternatively, we can set up a fine grid and perform a randomized grid search (using only a given proportion of the points on the grid sampled at random) and then doing deterministic search on the same grid limited to the best region.

# 7 APPLICATION OF ENSEMBLE MACHINE LEARNING IN CREDIT SCORING

Finally, we get to the most important part of this work which is the application of the selected ensemble machine learning models to a real-world credit dataset. The results of these models will be compared to those of the standard approach represented by a logistic regression model.

While the main goal is to compare the selected approaches in terms of out-of-sample predictive capability, my aim is to answer other important questions as well. I am interested in how difficult and time consuming it is to train the models, especially how hard is it to find the right hyper-parameter settings. I also want to demonstrate how input variables selection affects the performance of the models. Along the way I will also make comments on the practicality of these approaches and suggestions for further research.

Default over one year horizon will be modelled. Bondora defines defaulted loans as those where the borrower missed a part of the scheduled payment by more than 60 days. I will stick to this definition.

All the predictions in coming sections are probabilistic in the sense that the scores are real numbers between 0 and 1. These should be properly calibrated in order to be interpreted as valid PDs. I will not address the issue of calibration in this thesis. The scores should therefore be interpreted as relative "confidence" of the models not probabilities.

The whole development was executed within the Python environment. Python is an interpreted high-level general purpose programming language[53]. It is known for its design philosophy that emphasise code readability and availability of many specialized high quality open-source libraries. Libraries that were most important during the development are: Pandas[54] and NumPy[55] for data handling, scikit-learn[56], StatsModels[57] and XGBoost[58] for modelling and Matplotlib[59] and Seaborn[60] for visualizations.

[53] https://www.python.org
[54] https://pandas.pydata.org
[55] http://www.numpy.org
[56] scikit-learn.org
[57] http://www.statsmodels.org
[58] https://xgboost.readthedocs.io
[59] https://matplotlib.org
[60] https://seaborn.pydata.org

## 7.1 DATASET DESCRIPTION

The dataset used for this thesis comes from an Estonian peer-to-peer lending platform Bondora. The dataset was obtained from the company's website[61]. It is the company's policy to present the (potential) investors with detailed data so they can make informed decisions while investing their money.

As of June 2018, the data set contains information on more than 58 000 loans provided from 2009 to 2018. Since my goal is to predict default over one year horizon, only loans granted before 1st June 2017 were used during modelling. There were total of 36 565 of such loans.

The binary target variable "1Y_default" was created using the provided columns "DefaultDate" (day on which the default occurred) and "LoanDate" (day of loan origination). When the difference between these two dates is lower than or equal to 365 days "1Y_default" equals 1, it equals 0 otherwise.

The loans provided through Bondora platform show a high-risk profile. Of the loans used in modelling 29.732% defaulted by the end of the first year. We can also clearly see the rapid increase in riskiness during recent years. Number of credits provided also showed sharp increase during this period. In Figure 13 we see the number of loans granted during given years and average one year default rate of loans that originated during that year.

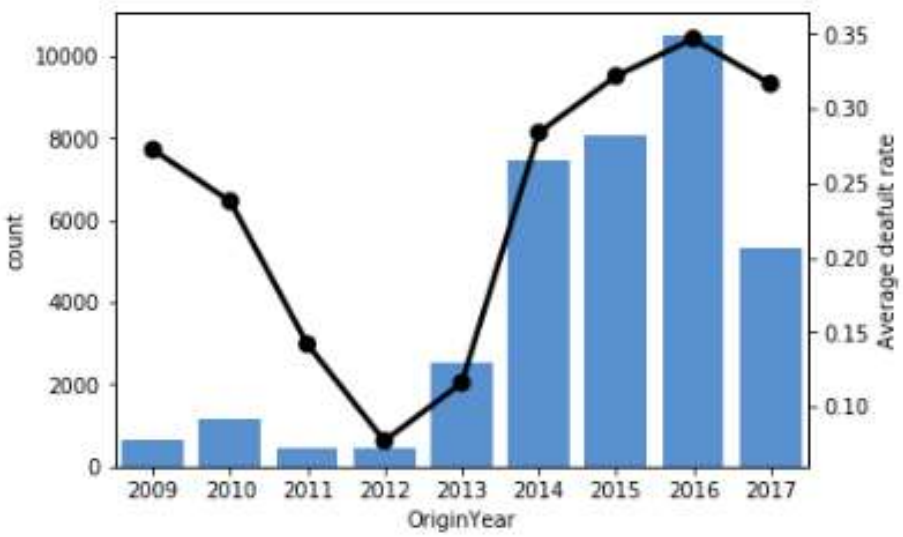


*Figure 13: Number of loans granted and average default rate over years*

[61] https://www.bondora.com/public-reports

Between years 2014 to 2017 we observe average default rate that was higher than that of post-crisis years. Note, that only loans granted before 1st June 2017 are shown in the figure.

Information provided by Bondora is quite throughout. The original dataset contains total of 112 columns. My aim is to develop score cards that can be used to score newly granted loans. Therefore many columns providing information on what happened during the credit's lifetime (collection, recovery, restructuring, secondary market info etc.) were not used during modelling. The dataset contains several scores estimated and provided by Bondora platform, these were not used as predictors as my goal is to assess whether the selected approaches can be used to develop standalone scoring functions. Other columns were dropped as well for different reasons (identifiers, various dates etc.). The full dataset description can be found on the company's website[62].

Total of 35 variables were identified as potential predictors. Out of these three more ("County", "City" and "EmploymentPosition") were dropped because of extremely high cardinality. The following table shows all the input variables selected for further analysis. Most names are self-explanatory, brief descriptions can be found in Appendix A. or, again, on the website.

| Feature | Type | #categories |
|---|---|---|
| Interest | Num | N/A |
| LanguageCode | Cat | 13 |
| Country | Cat | 4 |
| MonthlyPayment | Cat | 3 |
| HomeOwnershipType | Cat | 11 |
| MaritalStatus | Cat | 5 |
| NoOfPreviousLoansBeforeLoan | Num | N/A |
| Gender | Cat | 3 |
| AmountOfPreviousLoansBeforeLoan | Num | N/A |
| NewCreditCustomer | Cat | 2 |
| VerificationType | Cat | 4 |
| Education | Cat | 5 |
| ExistingLiabilities | Num | N/A |
| OccupationArea | Cat | 20 |
| UseOfLoan | Cat | 16 |
| NrOfDependants | Cat | 4 |
| EmploymentStatus | Cat | 6 |

[62] https://www.bondora.com/public-reports

| Feature | Type | #categories |
|---|---|---|
| AppliedAmount | Num | N/A |
| EmploymentDurationCurrentEmployer | Cat | 7 |
| Age | Num | N/A |
| WorkExperience | Cat | 6 |
| FreeCash | Num | N/A |
| IncomeOther | Num | N/A |
| DebtToIncome | Num | N/A |
| Amount | Num | N/A |
| IncomeFromLeavePay | Num | N/A |
| IncomeFromChildSupport | Num | N/A |
| IncomeFromPension | Num | N/A |
| IncomeFromSocialWelfare | Num | N/A |
| IncomeFromPrincipalEmployer | Num | N/A |
| IncomeFromFamilyAllowance | Num | N/A |
| IncomeTotal | Num | N/A |

*Table 3: Variables selected for further analysis*

The data contains variables with some missing values. These were treated in several different ways depending on the variable type (categorical/numeric) and the number of observations were the value is missing.

| Feature | #missing |
|---|---|
| DebtToIncome | 45 |
| Education | 53 |
| EmploymentDurationCurrentEmployer | 875 |
| EmploymentStatus | 197 |
| FreeCash | 45 |
| Gender | 45 |
| HomeOwnershipType | 1652 |
| MaritalStatus | 53 |
| MonthlyPayment | 6685 |
| NrOfDependants | 983 |
| OccupationArea | 137 |
| VerificationType | 53 |
| WorkExperience | 61 |

*Table 4: Missing values*

Monthly payment, originally continuous variable, was discretized into three categories “100<”, “<100” and “N/A” indicating missing values. Number of dependants was treated in a similar way. In “HomeOwnershipType” missing values were replaced by new category “N/A”.

All other missing values were replaced by median in case of numerical variables and mode in case of categorical variables. All categorical variables were encoded into dummy variables.

For the purpose of performance testing, the dataset was split into training and testing sets. Of all observations 70% was used during development and 30% was used for measuring out-of-sample performance. The split was stratified over the target variable to ensure the same proportion of defaulted loans is present in both sets.

## 7.2 Benchmark Model: Logistic Regression

Logistic regression (LR) represents the industry standard and was therefore used to build a benchmark model. Generally accepted development process was performed: preselection, univariate analysis, coarse classification, correlation analysis, forward selection from the short list and in- and out-of-sample performance measurement.

### 7.2.1 Preselection: Univariate Scoring

The preselection was performed by measuring in-sample Gini coefficient of a logistic regression model trained using one input variable at a time. All decisions regarding variable selection and later transformations were based on the training set only.

| Feature | Gini | Type | #categories |
|---|---|---|---|
| Interest | 0.420 | Num | N/A |
| LanguageCode | 0.412 | Cat | 12 |
| Country | 0.396 | Cat | 4 |
| MonthlyPayment | 0.208 | Cat | 3 |
| HomeOwnershipType | 0.195 | Cat | 11 |
| Gender | 0.178 | Cat | 3 |
| NoOfPreviousLoansBeforeLoan | 0.176 | Num | N/A |
| VerificationType | 0.173 | Cat | 4 |
| NewCreditCustomer | 0.172 | Cat | 2 |
| AmountOfPreviousLoansBeforeLoan | 0.171 | Num | N/A |
| Education | 0.171 | Cat | 5 |
| MaritalStatus | 0.171 | Cat | 5 |
| ExistingLiabilities | 0.149 | Num | N/A |
| OccupationArea | 0.142 | Cat | 20 |
| UseOfLoan | 0.109 | Cat | 16 |

*Table 5: Variables with in-sample Gini higher than 10%*

Table 3 shows top 15 variables sorted by in-sample Gini coefficient, these are all the variables that achieved in-sample Gini of more than 10%.

### 7.2.2 Univariate Analysis and Coarse Classification

Variables preselected this way were inspected graphically. Special treatment was given to categorical variables with high cardinality. Categories with similar logarithmic bad/good odds ratio were merged into one category to make the model estimation more stable as explained in section 6.5.2. For example “HomeOwnershipType” originally had 11 categories many of them being similar in terms of log B/G odds.

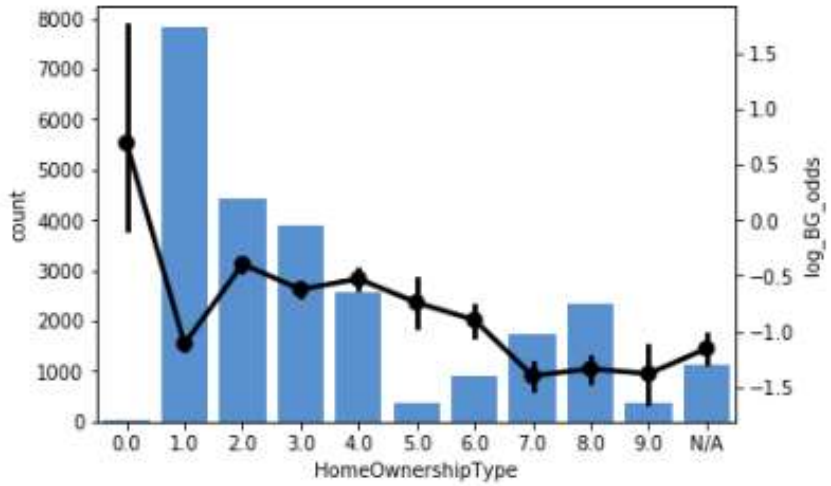


*Figure 14: Original HomeOwnershipType*

Categories showing similar riskiness were combined together. Category “0” (meaning the applicant is homeless) has only a few observations in it and was therefore merged into the most risky category despite having significantly higher log B/G odds. The resulting variable is shown in the following figure.

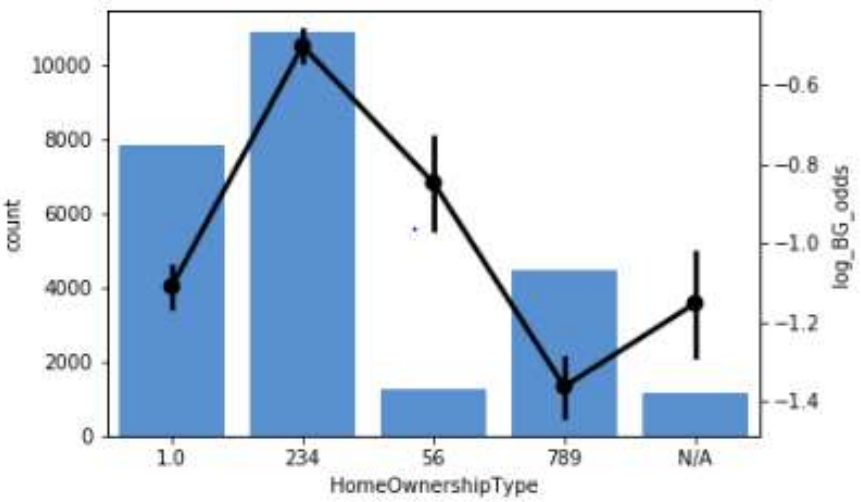


*Figure 15: HomeOwnershipType coarse classified*

Other categorical variables were treated the same way. In-sample Gini coefficient was measured after the transformation to ensure that not much information has been lost.

### 7.2.3 Multivariate Analysis: Pairwise Correlation

One of the assumptions of logistic regression model is independence of input variables. High multicollinearity can cause severe problems. Pairwise correlation was measured using Pearson correlation coefficient and high multicollinearity (corr. coeff. > |60%|) was eliminated. For the purpose of this analysis values of categorical variables were replaced by the corresponding weight of evidence values.

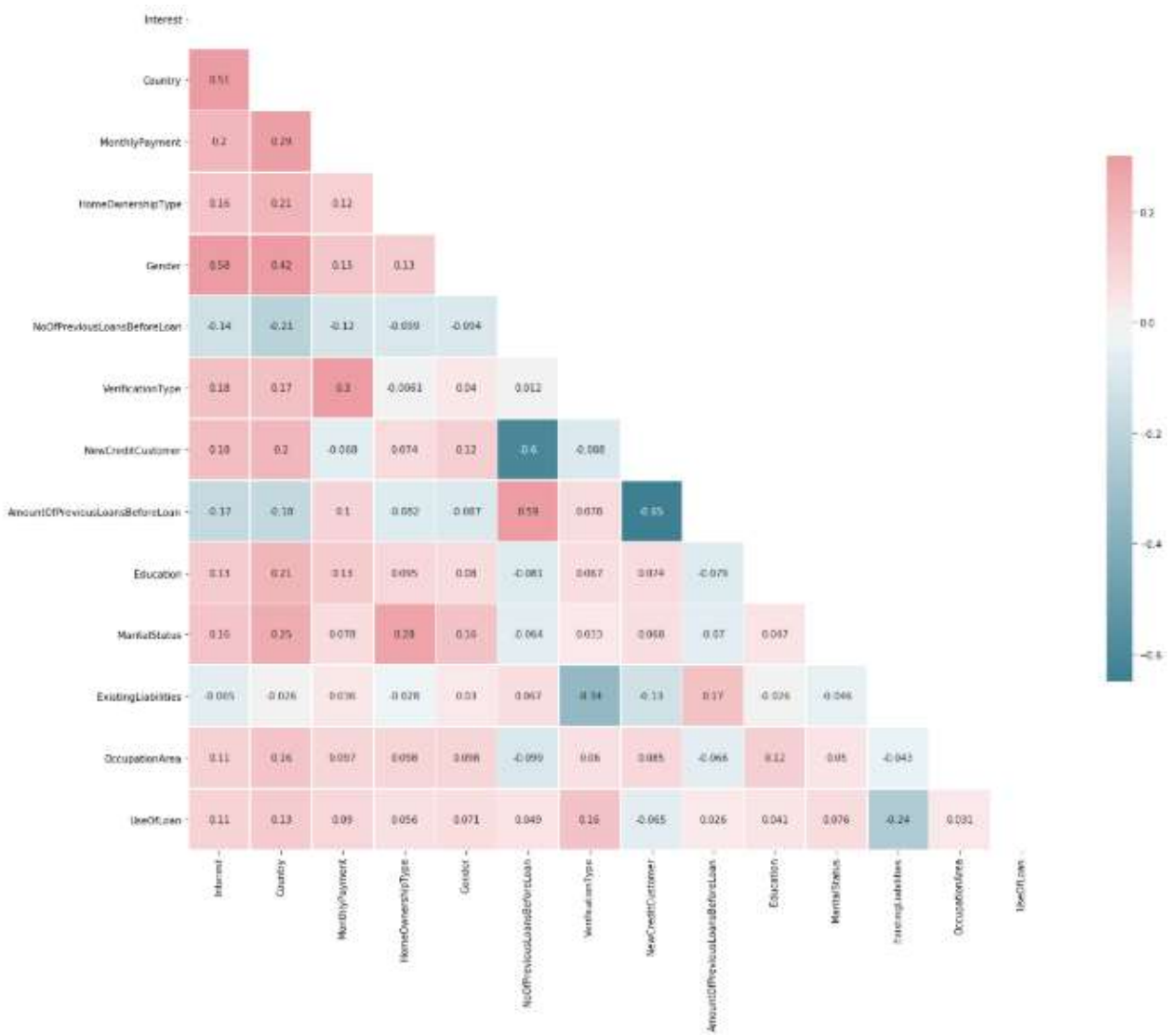


*Figure 16: Correlation matrix*

Variables “NoOfPreviousLoansBeforeLoan”, “AmountOfPreviousLoansBeforeLoan” and “NewCreditCustomer” show pairwise correlations about |60%|. These bear basically the

same information, binary variable “NewCreditCustomer” was kept as it is the simplest one and the Gini coefficients are at about the same for all of them.

### 7.2.4 Model Estimation and Evaluation

Forward selection with Bayesian information criterion (BIC) as the selection criterion was used to select the variables used in the final model. BIC is a metric closely related to AIC introduced in section 6.5.2., it was used as the forward selection procedure with AIC did not reject any input variables. It is defined as:

$$BIC = \ln(n)k - 2\ln L\,.$$

Where $n$ is the number of observations. BIC is based on Bayesian theorem and penalizes complex models more strongly than AIC[63]. The following table shows the final set of variables used during training.

| Feature | Gini | Type | #categories | Final model |
|---|---|---|---|---|
| Interest | 0.420 | Num | N/A | Yes |
| Country | 0.396 | Cat | 3 | Yes |
| MonthlyPayment | 0.208 | Cat | 3 | Yes |
| HomeOwnershipType | 0.184 | Cat | 5 | Yes |
| Gender | 0.178 | Cat | 3 | Yes |
| VerificationType | 0.173 | Cat | 4 | Yes |
| NewCreditCustomer | 0.172 | Cat | 2 | Yes |
| Education | 0.171 | Cat | 5 | Yes |
| MaritalStatus | 0.171 | Cat | 4 | Yes |
| ExistingLiabilities | 0.149 | Num | N/A | No |
| OccupationArea | 0.129 | Cat | 4 | No |
| UseOfLoan | 0.102 | Cat | 3 | No |

*Table 6: Final set of input variables*

The complete model summary including model coefficients, p-values and other statistics can be found in Appendix B. The model showed solid and stable performance with Gini coefficient above 50%.

| Set | Gini | AUC |
|---|---|---|
| Train | 0.5342 | 0.7671 |
| Test | 0.5287 | 0.7644 |

*Table 7: LR model performance*

[63] Bhat, H. S.; Kumar, N (2010). *On the derivation of the Bayesian Information Criterion, https://www.researchgate.net/publication/265739543_On_the_Derivation_of_the_Bayesian_Information_Criterion*

This model will serve as a baseline for comparison with all the other approaches developed in coming sections.

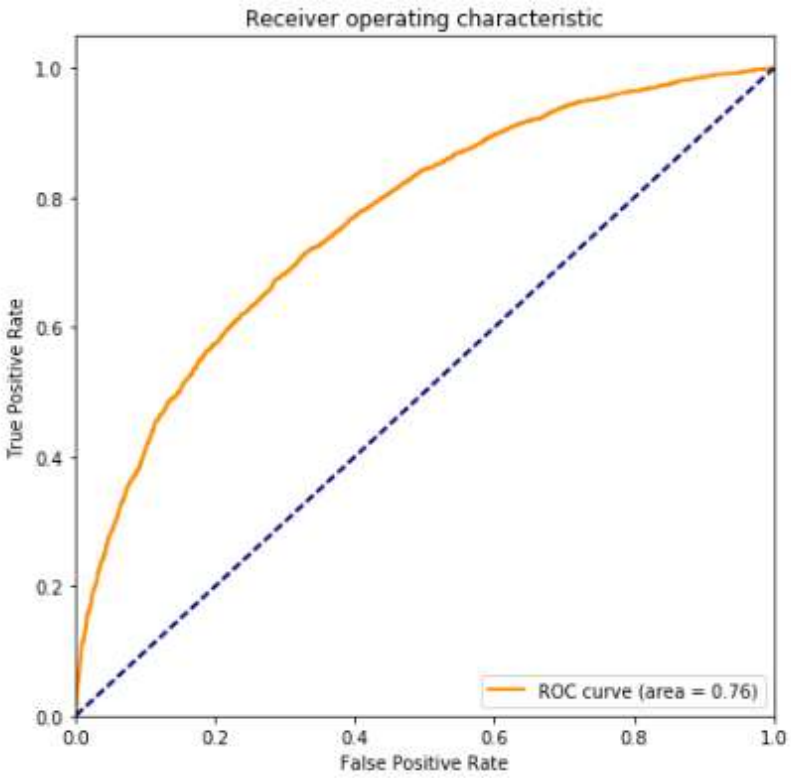


*Figure 17: LR ROC curve*

## 7.3 Basic Comparison of Selected Models

Before diving deep into different hyper-parameter settings of the given ensemble models, I will run two experiments first to demonstrate some properties of these models. These experiments will help us answer some of the questions outlined in the introduction to this chapter and to get some more intuition for these methods before developing the final models.

The experiments consist of running all the models with parameter settings that can be considered usual or generally acceptable. Of course, choosing such parameters is to some extend arbitrary. My aim here is not primarily to compere the models in terms of predictive capability (this will be done after parameter tuning in coming sections), but to show how these models behave in different conditions. The parameter settings used are described in Appendix C.

### 7.3.1 Unrestricted Experiment

In the first version of the experiment all models were trained using all the variables available (as listed in Table 3). So there is multicollinearity present in the dataset as well as several largely irrelevant variables. CARTs of three different depths (3, 7 and unlimited) were included to demonstrate how the base learners behave when used as standalone models.

| Model | test Gini | train Gini | test AUC | train AUC |
|---|---|---|---|---|
| ExtraTrees | 0.5895 | 1.0000 | 0.7947 | 1.0000 |
| RandomForest | 0.5853 | 1.0000 | 0.7927 | 1.0000 |
| XGBoost | 0.5583 | 0.5981 | 0.7792 | 0.7990 |
| AdaBoost | 0.5446 | 0.5725 | 0.7723 | 0.7863 |
| DecisionTree_7 | 0.5147 | 0.5786 | 0.7573 | 0.7893 |
| LogisticRegression | 0.4853 | 0.5053 | 0.7427 | 0.7527 |
| DecisionTree_3 | 0.4684 | 0.4764 | 0.7342 | 0.7382 |
| DecisionTree | 0.2413 | 1.0000 | 0.6207 | 1.0000 |

*Table 8: Results of the unrestricted experiment*

DecisionTree_3 refers to decision tree with maximum depth set to 3 (analogously for DecisionTree_7), DecisionTree refers to the tree with unlimited depth. XGBoost is a popular implementation of gradient boosting, for more details see Appendix C. Other names should be self-explanatory.

In conditions described above, the assumptions of logistic regression are severely violated. It should come as no surprise that the model performed relatively poorly. All four ensembles, on the other hand, showed very good results, all of them outperforming the benchmark model developed in the previous section. Methods based on model averaging performed better than boosting. We can also see how the unlimited CART severely overfits the data. Also note how model averaging did its job in combining the overfit models into a strong learner. Figure 18 compares the models graphically.

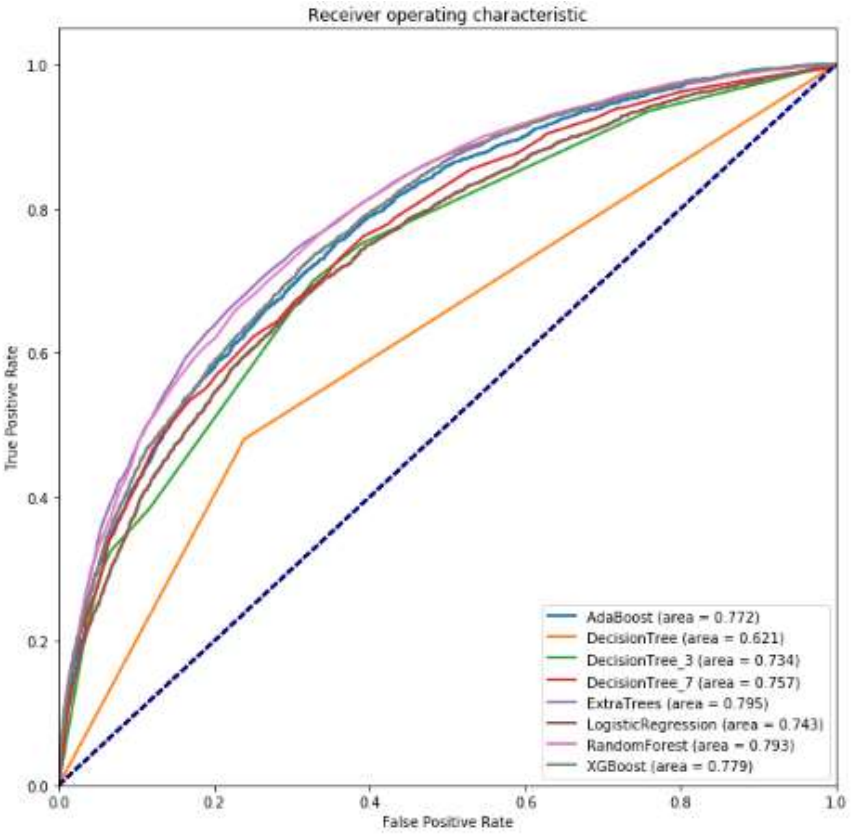


*Figure 18: ROC curves of models in unrestricted experiment*

Next we will have a look on how computationally expensive it is to train the models. This plays especially big role when searching for the right hyper-parameters via grid search where we have to fit each model many times.

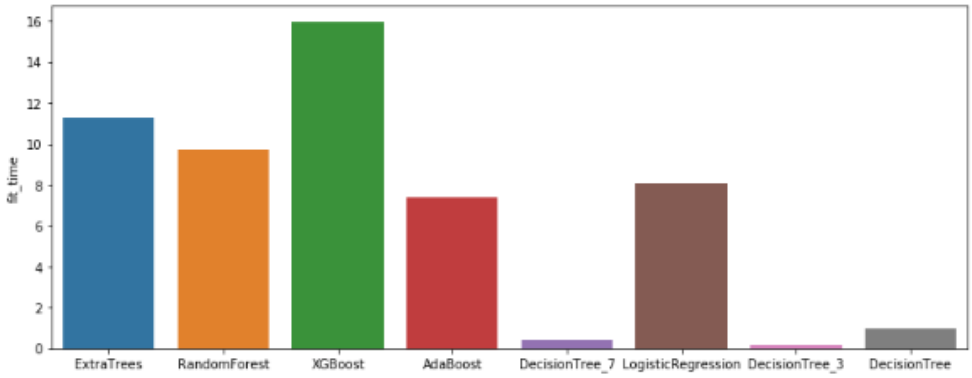


*Figure 19: Training times (in seconds)*

The time needed to fit the models is obviously determined by the machine used so only relative comparison is meaningful in this context. Different implementations of the same algorithm can also show different efficiency. Note for example that ExtraTrees which should theoretically be faster to train than random forest took more time in this experiment. Also the

time needed to fit the ensemble model is largely determined by the number of base learners (all ensembles for now consist of 100 decision trees).

### 7.3.2 Experiment with Preselected Variables

In the second version of the experiment the same models were trained using the variables preselected during the development of the benchmark model (as listed in Table 4). There are no irrelevant variables and no very high correlation. Some information was, however, lost due to coarse classification. Variables with strongly non-linear relationship with the target variable were also lost as simple logit model was responsible for the selection.

| Model | test Gini | train Gini | test AUC | train AUC |
|---|---|---|---|---|
| XGBoost | 0.5485 | 0.5723 | 0.7743 | 0.7862 |
| AdaBoost | 0.5390 | 0.5472 | 0.7695 | 0.7736 |
| LogisticRegression | 0.5313 | 0.5362 | 0.7656 | 0.7681 |
| RandomForest | 0.5219 | 0.9994 | 0.7609 | 0.9997 |
| DecisionTree_7 | 0.5144 | 0.5594 | 0.7572 | 0.7797 |
| DecisionTree_3 | 0.4563 | 0.4658 | 0.7282 | 0.7329 |
| ExtraTrees | 0.4406 | 0.9998 | 0.7203 | 0.9999 |
| DecisionTree_unl | 0.2360 | 0.9998 | 0.6180 | 0.9999 |

*Table 9: Results of experiment with preselected variables*

Performance of logistic regression model improved significantly, it is actually slightly better than the performance of benchmark model, selection based on BIC doesn't seem to help in this case (from the practical standpoint of view one could, however, argue that selecting simpler model giving comparable results is in fact beneficial). Performance of all ensemble models not only didn't improve but dropped rapidly, standalone CARTs performed about the same as previously. Especially interesting are the poor results of extremely randomized trees, these got outperformed even by two of the standalone decision trees. When introducing ExtraTrees the authors argue that the splitting points tend to be very unstable and so making them random doesn't hurt the performance but can actually improve it (recall that extremely randomized trees use random splitting points instead of resampling as means of decorrelation of the base learners)[64]. This simply didn't work in this case, even after rerunning the experiment with different random seeds ExtraTrees achieved Gini coefficients about 0.44.

[64] P. Geurts, D. Ernst., and L. Wehenkel, *Extremely randomized trees*, Machine Learning, 63(1), 3-42, 2006.

When using resampling (as in random forest) the performance grew to Gini coefficient of 0.48, which is still relatively poor.

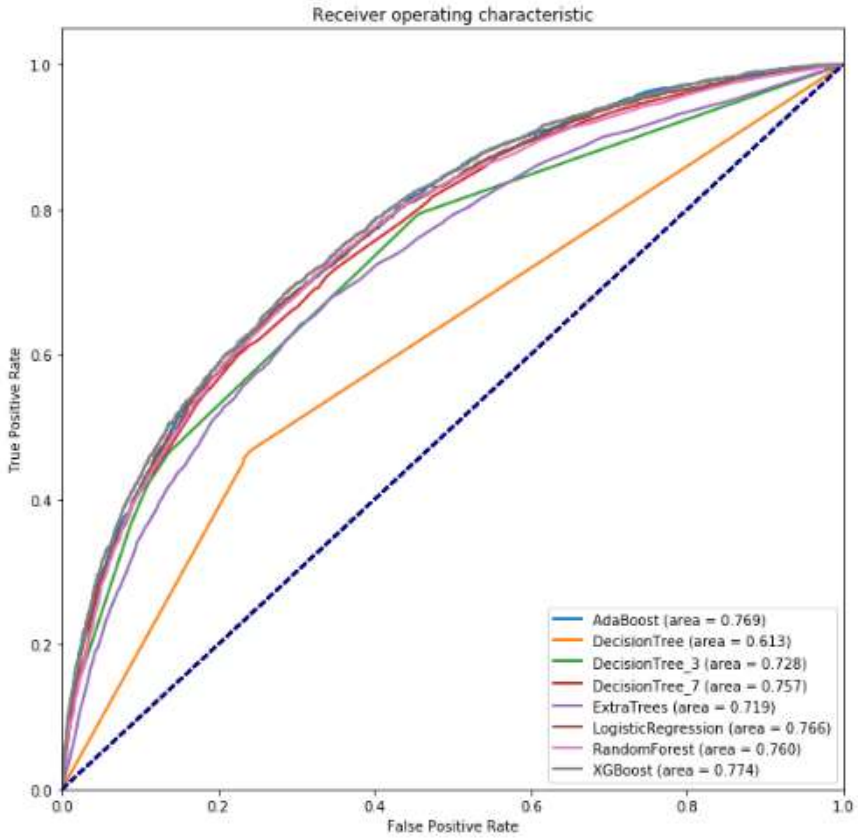


*Figure 20: ROC curves of model in experiment with preselected predictors*

All of this reminds us about the fundamental differences between linear and tree-based models. It is now also obvious that the whole data preparation and variable selection process needs to be re-though.

## 7.4 Variable Selection with Tree-Based Models

When using ensemble models based on decision trees we need to approach variable selection differently. Univariate selection does not make much sense as there are no independence assumptions. When dealing with high-dimensional datasets one would probably need to discard some potential inputs based on business logic (variables where there is no reasonable hypothesis why this could predict default, variables that are expensive to acquire or otherwise hard to get, legal issues etc.) or some other criteria (very low variance, extremely high cardinality etc.), but to get most of the data one has to take into account the interactions among inputs when selecting variables for the final model. Coarse classification also seems

unnecessary as decision trees perform implicit variable selection (not all the inputs need to be used for prediction) so dummy variables representing irrelevant categories are simply ignored. Categories with low number of observations (or irrelevant categories chosen by randomized trees) can cause problems. Unlike linear models decision trees are local models, therefore only certain areas of the feature space are influenced by such cases. Discretization of continuous variables is unnecessary as well as CART performs discretization online.

Naïve approach would be to try all possible combinations of input variables, test them out-of-sample (or even cross-validate them) and select the best combination. This brute force solution would be extremely computationally demanding and therefore totally impractical.

More sensible solution is to use variable importance measure introduced in chapter X. We can simply select $k$ best variables based on their importance, select variables with importance above some predefined threshold or recursively eliminate variables with lowest importance. Cross-validation can be used to select the best value of $k$.

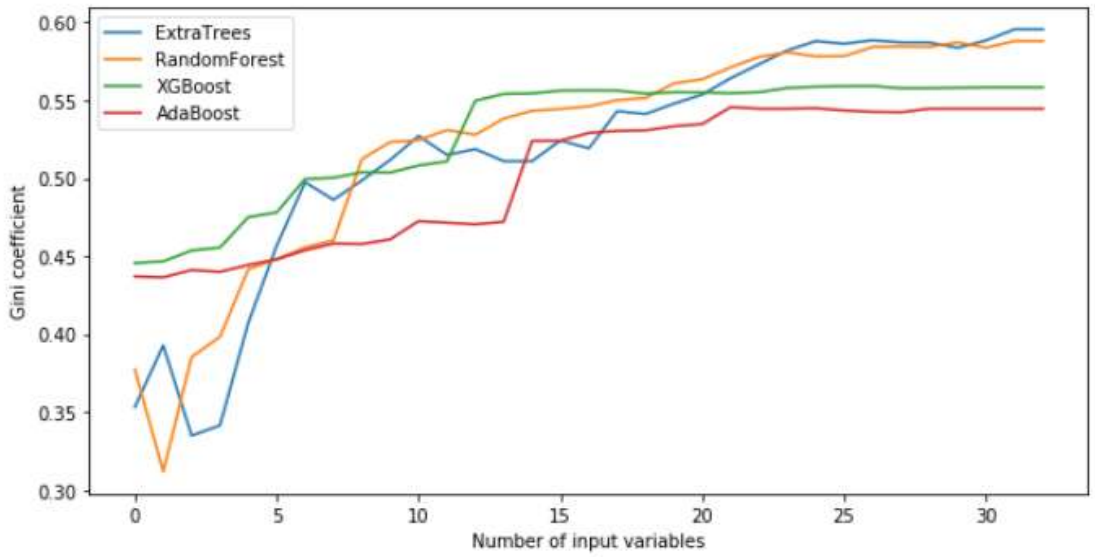


*Figure 21: Performance of models with different number of input variables*

In Figure 21 we see out-of-sample performance of models trained using different number of inputs. These were selected as best variables based on variable importance of a model trained using all available variables. All models use the same hyper-parameters as in previous section (see Appendix C). Variable selection does not improve performance of the models. After reaching certain number of inputs, very different for different models, the out-of-sample Gini coefficient (almost) stops increasing.

From all of the above it seems that the best practical approach would be to fine tune the model using all predictors available and then use feature importance to prune some variables from the original set of input variables. Naturally, we want the model to be as simple as possible. Models with lower number of predictors are easier to deploy and eventually debug. The lower the number of inputs the easier is to monitor the model while being used in production. The behaviour of some variables can change over time and can make the model's performance to drop. Such cases are hard to recognize if the number of predictors is high. In practice, we should therefore eliminate all the features that does not improve the model's predictive power, we may even be willing to sacrifice some of the predictive capability to have the model simpler and easier to manage.

As my main goal is to compare the models by their out-of-sample performance and because the variables selection process does not seem to improve the performance, I will use all the available predictors in the coming sections.

## 7.5 Ensemble Models with Tuned Hyper-Parameters

Finally, we can compare the models in terms of their out-of-sample predictive capability. To find optimal hyper-parameter settings exhaustive grid search was performed for each model. This was done in two steps, during the first run rougher grid was used to find promising hyper-parameters, these were further refined using another run of grind search with finer grid.

Such a task is computationally very demanding and given my resources I was forced to use relatively rough grids. I also focused on the most important parameters only. For example I only used the depth of the trees as means of limiting their complexity and used default values of other stopping criteria. Three-fold cross-validation stratified over the output variable was used during the search.

### 7.5.1 Random Forest

When searching for optimal settings of the random forest model three hyper-parameters were tuned: number of base learners, maximum depth of the trees and $p$ (the number of input considered at each split.

| Random forest | | |
|---|---|---|
| Number of base learners | 300 | |
| Maximum depth | Unlimited | |
| $p$ | $\log_2 d$ | |
| | **Benchmark** | **Final** |
| train Gini | 1.0000 | 1.0000 |
| test Gini | 0.5853 | 0.5960 |

*Table 10: Random forest – final model*

Two values of $p$ were considered: $p = \sqrt{d}$ and $p = \log_2 d$. Number of base learners tried during the search ranged from 20 to 300. As expected, increasing number of trees in the ensemble meant better performance. Several values higher than 300 were tested to see whether the performance can be further enhanced. Having more base learners in the model did not significantly improve the performance. Limiting the maximum depth of the trees did not lead to any improvement (values between 5 and 20 were considered). Random forest proved to be robust to different hyper-parameter settings.

### 7.5.2 Extremely Randomized Trees

The same procedure was performed with the extremely randomized trees model.

| Extremely randomized trees | | |
|---|---|---|
| Number of base learners | 300 | |
| Maximum depth | Unlimited | |
| $p$ | $\log_2 d$ | |
| | **Benchmark** | **Final** |
| train Gini | 1.0000 | 1.0000 |
| test Gini | 0.5895 | 0.6004 |

*Table 11: Extremely randomized trees – final model*

The results are almost identical to those of random forest. Both models are based on the same principles and extremely randomized trees as a newer variant of this model did not proved to be notably better than the traditional one.

### 7.5.3 AdaBoost

Only two hyper-parameters were played with while searching for the best adaptive boosting model. These were the number of base learners and learning rate. The selected model look as follows.

| AdaBoost | | |
|---|---|---|
| Number of base learners | 480 | |
| Learning rate | 0.2 | |
| | **Benchmark** | **Final** |
| train Gini | 0.5725 | 0.5773 |
| test Gini | 0.5446 | 0.5497 |

*Table 12: AdaBoost – final model*

As originally recommended base learners with only one split were used, number of trees in the ensemble up to 600 and learning rate ranging from 0.1 to 1 were tried during the search (it is feasible to try larger ensembles as the base learners are very simple and computationally cheap). Similarly to previous models AdaBoost also shows robustness to hyper-parameter settings, results of both versions of the model are virtually identical.

### 7.5.4 Gradient Boosting

Four hyper-parameters were tuned during the search for the best gradient boosting model: number of base learners, learning rate, maximum depth and subsample size.

| Gradient boosting | | |
|---|---|---|
| Number of base learners | 220 | |
| Learning rate | 0.05 | |
| Maximum depth | 6 | |
| Subsample (*f*) | 0.7 | |
| | **Benchmark** | **Final** |
| train Gini | 0.5981 | 0.7665 |
| test Gini | 0.5583 | 0.5826 |

*Table 13: Gradient boosting – final model*

Maximum depth up to 6, number of base learners up to 500, learning rate ranging between 0.025 and 0.3 and subsample between 0.5 and 1 were tried during the search. Lower learning rates were chosen as more complex models are being combined in this case. For the first time we see noticeable difference in performance between benchmark and tuned models. What is quite surprising is the relatively high maximum depth (larger than generally recommended).

### 7.5.5 Performance Comparison: Summary

All four models showed strong out-of-sample performance, all outperforming the benchmark model developed in chapter X. Random forest, extremely randomized trees and gradient boosting showed very similar performance, adaptive boosting's performance was closer to

the logistic regression benchmark. Performance of all the models is visualized in Figure 22 on the next page.

**One final note:** although the models' performances were often almost identical, they differ in how they use the input variables. When looking at variable importance, we see that, for example, extremely randomized trees model has these much more evenly distributed than random forest. The same is true for both boosting models before and after hyper-parameter optimization. This might have practical impact as we want the models to be as simple as possible (for reasons explained earlier). It seems reasonable to assume that models with more evenly distributed importance would be harder to prune (by leaving out variables with lowest variable importance). This hypothesis would, however, need to be further researched. Variable importance of final models can be found in electronic files attached to this thesis described in Appendix D.

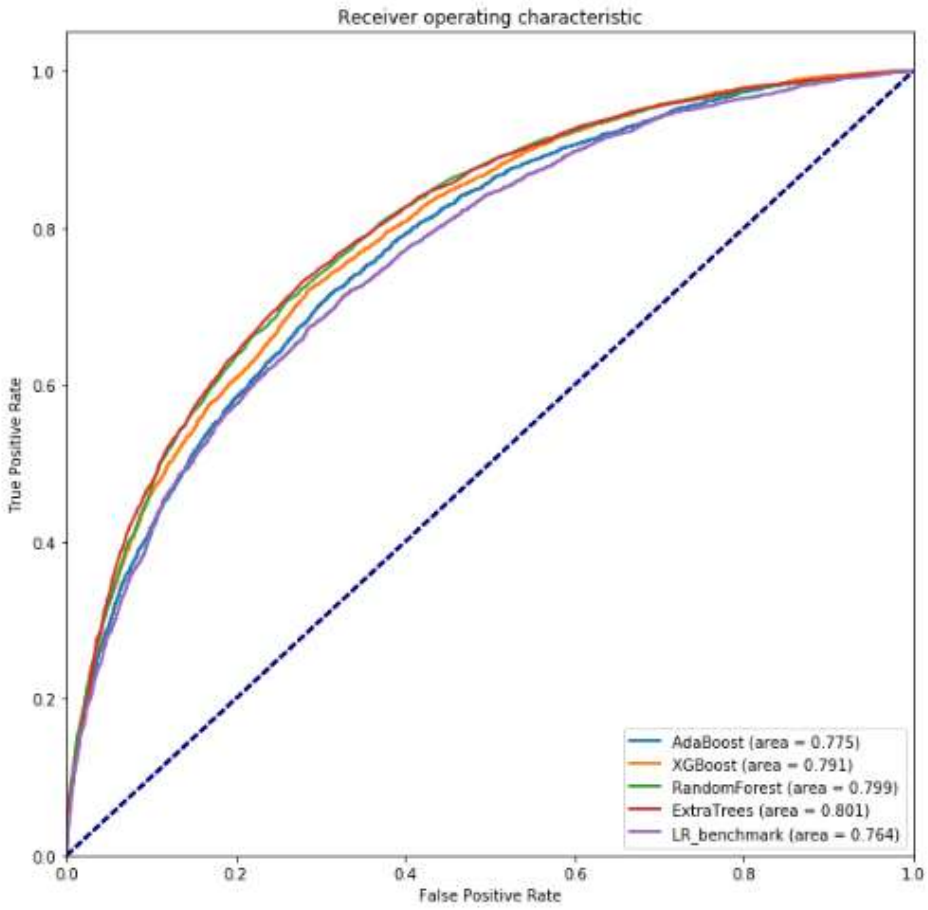


*Figure 22: ROC curve of final models*

# CONCLUSIONS

The main goal of this work was to explore the possibility and feasibility of using ensemble machine learning methods for credit scoring tasks. Four ensemble models were selected for this purpose: random forest, extremely randomized trees, adaptive boosting and gradient boosting, all using classification and regression trees (CART) as base models.

At first, general principles behind supervised machine learning were explained. The concept of bias-variance decomposition was derived and used as a tool for understanding how model complexity relates to out-of-sample predictive capability of the models.

Chapter Four provided detailed explanation of how decision tree models work. Special focus was placed on geometrical interpretation of decision trees and their behaviour in the context of bias-variance tradeoff. In Chapter Five two different approaches to combining models into ensembles were introduced: model averaging and boosting. All four selected ensemble models were then described in detail.

Chapter Six focused on common practices in credit scoring and the general scoring function development process. Methods for model validation and performance measurement were introduced.

At the beginning of Chapter Seven, a benchmark model in accordance with the generally accepted development process was created using the industry standard model: logistic regression. The ensemble models were then developed in a number of ways to not only test their out-of-sample performance but to demonstrate other properties as well.

All four models proved to be robust to presence of irrelevant input variables and to multicollinearity among the inputs. Even the generic versions of the ensembles, without any hyper-parameter optimization, trained using all the inputs available outperformed the benchmark model. This is remarkable as the data preparation phase, when working with logistic regression, is time consuming and many decisions have to be made, some of them are, at least to some extent, arbitrary. The ability to handle “messy” data may also come handy when experimenting with alternative datasets.

All of the selected ensembles (even though gradient boosting to a lesser extent than other models) seem to also be robust to different hyper-parameter settings. The computationally

very expensive hyper-parameter tuning via grid search did not lead to big improvements in models' performances.

Both methods based on model averaging, random forest and extremely randomized trees, showed better performance than boosting algorithms. As extremely randomized trees is a new model with little coverage in literature and not many software implementations available, I would recommend using random forest when looking for a tree-based ensemble machine learning solution to credit scoring. It is, however, fair to note that this conclusion is based on one single dataset. On different datasets (e.g. different kinds of credit) boosting models can outperform random forest[65]. The final conclusion is therefore a more general one, when only out-of-sample performance is considered, tree-based ensemble machine learning models present a promising alternative to the standard modelling approach.

The main disadvantage of such ensembles is their complex nature. On one hand, these models are, in a way, easy to use as strong performance can be achieved with little to no data preparation and hyper-parameter tuning. On the other hand, such models would be very impractical. The more inputs the model has the more difficult it is to deploy, monitor and eventually debug it. A straightforward approach to simplifying the ensembles based on variable importance was described in section 7.4.

Other big issue related to the models' complexity is their interpretability. Financial institutions are required to be able to explain the behaviour of their models. Interpretability is also closely related to monitoring the model's behaviour once the model is deployed. Unlike in logistic regression or standalone decision trees, the coefficients (splitting points of the base learners and their weights in the ensemble) in the ensemble models cannot be interpreted directly. Two of the indirect methods for understanding the models, variable importance and partial dependence plots, were explained in section 5.5. There are also other methods based on similar principles available such as individual conditional expectation plots or Friedman's H-statistic for understanding the interactions among the inputs.

The main challenge and an interesting topic for further research is to find a standardized way of developing scoring functions using these models in such a way as to maintain as much of

[65] As indicated by the author's experience after brief experimenting with a dataset provided by a different P2P lending platform that is not part of this thesis.

their discriminatory power as possible while reducing them to a simpler and more manageable forms. Until such a procedure is proposed and thoroughly tested, financial institutions and their regulators are unlikely to consider replacing logistic regression with ensemble models. For now, these models may present an interesting alternative for private funds that are not bounded by the regulation and eventually for individuals investing through P2P platforms like Bondora.

# BIBLIOGRAPHY


Hastie, Trevor; Tibshirani, Robert; Friedman, Jerome (2008). *The Elements of Statistical Learning (2nd ed.)*. Springer. ISBN 0-387-95284-5.

Vijayakumar, Sethu (2007). *The Bias–Variance Tradeoff* . University Edinburgh. Available at: http://www.inf.ed.ac.uk/teaching/courses/mlsc/Notes/Lecture4/BiasVariance.pdf

Domingos, Pedro (2000). *A unified bias-variance decomposition* . ICML. Available at: https://homes.cs.washington.edu/~pedrod/bvd.pdf

Manning, Christopher D.; Raghavan, Prabhakar; Schütze, Hinrich (2008). *Introduction to Information Retrieval*. Cambridge University Press. pp. 308–314.

BERKA, P (2003). *Dobývání znalostí z databází*, Praha Academia, 80-200-1062-9

Quinlan, J. R. (1986). *Induction of Decision Trees*. Machine Learning 1: 81-106, Kluwer Academic Publishers. Available at: http://hunch.net/~coms-4771/quinlan.pdf

Pandya R., Pandya J. (2015). C5.0 *Algorithm to Improved Decision Tree with Feature Selection and Reduced Error Pruning.* Available at: https://research.ijcaonline.org/volume117/number16/pxc3903318.pdf

Magidson, Jay (1994) *The CHAID approach to segmentation modeling: chi-squared automatic interaction detection*, in Bagozzi, Richard P. (ed); *Advanced Methods of Marketing Research*, Blackwell, Oxford, GB, pp. 118–159

Papagelis A., Kalles D.(2001). *Breeding Decision Trees Using Evolutionary Techniques*, Proceedings of the Eighteenth International Conference on Machine Learning. pp. 393–400. Available at: http://www.gatree.com/data/BreedinDecisioTreeUsinEvo.pdf

Chipman, Hugh A., Edward I. George, and Robert E. McCulloch (1998). *Bayesian CART model search*. Journal of the American Statistical Association 93.443: 935–948.

Barros R. C., Cerri R., Jaskowiak P. A., Carvalho, A. C. P. L. F., *A bottom-up oblique decision tree induction algorithm*. Proceedings of the 11th International Conference on Intelligent Systems Design and Applications (ISDA 2011).

Gareth, James; Witten, Daniela; Hastie, Trevor; Tibshirani, Robert (2015). *An Introduction to Statistical Learning*. New York: Springer. p. 315. ISBN 978-1-4614-7137-0.

Stefan Lessmann, Bart Baesens, Hsin-Vonn Seow and Lyn C. Thomas (2015). European Journal of Operational Research, 2015, vol. 247, issue 1, 124-136

L. Breiman, and A. Cutler, *Random Forests*, Available at: http://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm

Segal, M. (2004). *Machine learning benchmarks and random forest regression*, Technical report, eScholarship Repository, University of California. Available at: http://repositories.edlib.org/cbmb/bench rf regn.

Liu, Fei Tony, Ting, Kai Ming and Zhou, Zhi-Hua. *Isolation forest*. Data Mining, 2008. ICDM‘08. Eighth IEEE International Conference on.

P. Geurts, D. Ernst., and L. Wehenkel, *Extremely randomized trees*, Machine Learning, 63(1), 3-42, 2006.

Leo Breiman (1996). *BIAS, VARIANCE, AND ARCING CLASSIFIERS,* TECHNICAL REPORT. Available at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.115.7931&rep=rep1&type=pdf

Robert E. Schapire, *Explaining AdaBoost*, Available at: http://rob.schapire.net/papers/explaining-adaboost.pdf

Mason, L., Baxter, J., Bartlett, P., Frean, M.*: Functional gradient techniques for combining hypotheses*. In: Advances in Large Margin Classifiers. MIT Press (2000)

D. (2018, August 07). Dmlc/xgboost. Retrieved from https://github.com/dmlc/xgboost/tree/master/demo#machine-learning-challenge-winning-solutions

Friedman, J. H. *Greedy Function Approximation: A Gradient Boosting Machine*. (February 1999) Available at: https://statweb.stanford.edu/~jhf/ftp/trebst.pdf

Friedman, J. H. *Stochastic Gradient Boosting*. (March 1999) Available at: https://statweb.stanford.edu/~jhf/ftp/stobst.pdf

Christoph Molnar (2018), *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable,* Available at: https://christophm.github.io/interpretable-ml-book/

WITZANY, J. Credit risk management: pricing, measurement, and modeling. Cham: Springer, 2017. ISBN 978-3-319-49799-0.

Greene. W. H. (2003). *Econometric Analysis (5th ed),* Upper Saddle River, New Jersey 07458

Akaike, H. (1973), *Information theory and an extension of the maximum likelihood principle,* in Petrov, B. N.; Csáki, F., 2nd International Symposium on Information Theory, Tsahkadsor, Armenia, USSR, September 2-8, 1971, Budapest: Akadémiai Kiadó, pp. 267–281.

Bhat, H. S.; Kumar, N (2010). *On the derivation of the Bayesian Information Criterion,* Available at: https://www.researchgate.net/publication/265739543_On_the_Derivation_of_the_Bayesian_Information_Criterion

# LIST OF TABLES

# LIST OF FIGURES

# LIST OF ALGORITHMS

# APPENDIX A: DATA DICTIONARY

| Feature | Description |
|---|---|
| Interest | Maximum interest rate accepted in the loan application |
| LanguageCode | 1 Estonian 2 English 3 Russian 4 Finnish 5 German 6 Spanish 9 Slovakian |
| Country | Residency of the borrower |
| MonthlyPayment | Estimated amount the borrower has to pay every month |
| HomeOwnershipType | 0 Homeless 1 Owner 2 Living with parents 3 Tenant, pre-furnished property 4 Tenant, unfurnished property 5 Council house 6 Joint tenant 7 Joint ownership 8 Mortgage 9 Owner with encumbrance 10 Other |
| MaritalStatus | 1 Married 2 Cohabitant 3 Single 4 Divorced 5 Widow |
| NoOfPreviousLoansBeforeLoan | Number of previous loans |
| Gender | 0 Male 1 Woman 2 Undefined |
| AmountOfPreviousLoansBeforeLoan | Value of previous loans |
| NewCreditCustomer | Did the customer have prior credit history in Bondora 0 Customer had at least 3 months of credit history in Bondora 1 No prior credit history in Bondora |
| VerificationType | Method used for loan application data verification 1 Income unverified 2 Income unverified, cross-referenced by phone 3 Income verified 4 Income and expenses verified |
| Education | 1 Primary education 2 Basic education 3 Vocational education 4 Secondary education 5 Higher education |
| ExistingLiabilities | Borrower's number of existing liabilities |

| | |
|---|---|
| OccupationArea | 1 Other 2 Mining 3 Processing 4 Energy 5 Utilities 6 Construction 7 Retail and wholesale 8 Transport and warehousing 9 Hospitality and catering 10 Info and telecom 11 Finance and insurance 12 Real-estate 13 Research 14 Administrative 15 Civil service & military 16 Education 17 Healthcare and social help 18 Art and entertainment 19 Agriculture, forestry and fishing |
| UseOfLoan | 0 Loan consolidation 1 Real estate 2 Home improvement 3 Business 4 Education 5 Travel 6 Vehicle 7 Other 8 Health 101 Working capital financing 102 Purchase of machinery equipment 103 Renovation of real estate 104 Accounts receivable financing 105 Acquisition of means of transport 106 Construction finance 107 Acquisition of stocks 108 Acquisition of real estate 109 Guaranteeing obligation 110 Other business All codes in format 1XX are for business loans that are not supported since October 2012 |
| NrOfDependants | Number of children or other dependants |
| EmploymentStatus | 1 Unemployed 2 Partially employed 3 Fully employed 4 Self-employed 5 Entrepreneur 6 Retiree |
| AppliedAmount | The amount borrower applied for originally |
| EmploymentDurationCurrentEmployer | Employment time with the current employer |
| Age | The age of the borrower when signing the loan application |
| WorkExperience | Borrower's overall work experience in years |
| FreeCash | Discretionary income after monthly liabilities |
| IncomeOther | Borrower's income from other sources |

| | |
|---|---|
| DebtToIncome | Ratio of borrower's monthly gross income that goes toward paying loans |
| Amount | Amount the borrower received on the Primary Market. This is the principal balance of your purchase from Secondary Market |
| IncomeFromLeavePay | Borrower's income from paternity leave |
| IncomeFromChildSupport | Borrower's income from alimony payments |
| IncomeFromPension | Borrower's income from pension |
| IncomeFromSocialWelfare | Borrower's income from social support |
| IncomeFromPrincipalEmployer | Borrower's income from its employer |
| IncomeFromFamilyAllowance | Borrower's income from child support |
| IncomeTotal | Borrower's total income |

# APPENDIX B: BENCHMARK MODEL DETAILS

Logit Regression Results

| | | | |
|---|---|---|---|
| Dep. Variable: | default | No. Observations: | 25597 |
| Model: | Logit | Df Residuals: | 25575 |
| Method: | MLE | Df Model: | 21 |
| Date: | Mon, 16 Jul 2018 | Pseudo R-squ.: | 0.1661 |
| Time: | 23:13:45 | Log-Likelihood: | -12991. |
| Converged: | True | LL-Null: | -15579. |
| | | LLR p-value: | 0.000 |

| | coef | std err | z | P>\|z\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| Intercept | -1.5535 | 0.149 | -10.415 | 0.000 | -1.846 | -1.261 |
| C(Country)[T.ES] | 1.1691 | 0.047 | 25.044 | 0.000 | 1.078 | 1.261 |
| C(Country)[T.FI] | 0.5347 | 0.044 | 12.265 | 0.000 | 0.449 | 0.620 |
| C(NewCreditCustomer)[T.True] | 0.6039 | 0.036 | 16.929 | 0.000 | 0.534 | 0.674 |
| C(MonthlyPayment)[T.<100] | -0.0591 | 0.034 | -1.724 | 0.085 | -0.126 | 0.008 |
| C(MonthlyPayment)[T.N/A] | -0.9643 | 0.065 | -14.756 | 0.000 | -1.092 | -0.836 |
| C(HomeOwnershipType)[T.234] | 0.2931 | 0.038 | 7.792 | 0.000 | 0.219 | 0.367 |
| C(HomeOwnershipType)[T.56] | 0.2057 | 0.073 | 2.808 | 0.005 | 0.062 | 0.349 |
| C(HomeOwnershipType)[T.789] | -0.2029 | 0.050 | -4.052 | 0.000 | -0.301 | -0.105 |
| C(HomeOwnershipType)[T.N/A] | 0.5815 | 0.168 | 3.456 | 0.001 | 0.252 | 0.911 |
| C(Education)[T.2.0] | -0.1389 | 0.141 | -0.986 | 0.324 | -0.415 | 0.137 |
| C(Education)[T.3.0] | -0.1951 | 0.138 | -1.410 | 0.159 | -0.466 | 0.076 |
| C(Education)[T.4.0] | -0.4858 | 0.137 | -3.543 | 0.000 | -0.755 | -0.217 |
| C(Education)[T.5.0] | -0.6324 | 0.138 | -4.586 | 0.000 | -0.903 | -0.362 |
| C(VerificationType)[T.2.0] | 0.3826 | 0.152 | 2.511 | 0.012 | 0.084 | 0.681 |
| C(VerificationType)[T.3.0] | -0.1090 | 0.041 | -2.648 | 0.008 | -0.190 | -0.028 |
| C(VerificationType)[T.4.0] | -0.4006 | 0.040 | -10.064 | 0.000 | -0.479 | -0.323 |
| C(Gender)[T.1.0] | -0.1540 | 0.032 | -4.759 | 0.000 | -0.217 | -0.091 |
| C(Gender)[T.2.0] | 0.0359 | 0.077 | 0.464 | 0.643 | -0.116 | 0.188 |
| C(UseOfLoan)[T.low] | -0.1190 | 0.047 | -2.554 | 0.011 | -0.210 | -0.028 |
| C(UseOfLoan)[T.medium] | 0.0171 | 0.035 | 0.492 | 0.623 | -0.051 | 0.085 |
| Interest | 0.0150 | 0.001 | 15.020 | 0.000 | 0.013 | 0.017 |

# APPENDIX C: IMPLEMENTATIONS OF ENSEMBLE MODELS USED

Popular Python machine learning libraries were used for modelling in this thesis. For random forest, extremely randomized trees and adaptive boosting scikit-learn library was used. The details, parameters and links to source code can be found in the official documentation at http://scikit-learn.org/stable/modules/classes.html#module-sklearn.ensemble.

For gradient boosting a popular implementation called XGBoost was used. It is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. All the details can, again, be found in the official documentation that can be accessed at https://xgboost.readthedocs.io/en/latest/python/index.html.

By referring to "default settings" anywhere in the text, the default settings of algorithms as implemented in these libraries are meant with one exception, the number of base learners was set to 100 for all the models.

# APPENDIX D: ELECTRONIC ATTACHMENTS

Electronic attachments present an important part of this work. These can be divided into several different categories.

There are nine HTML files, these are exports from Jupyter notebooks (Python 3) and can be used to explore the data and the general workflow using any standard web browser.

- BondoraProfile.html – Convenient tool for exploring the raw dataset.
- DataPreparation.html – Data preparation, pre-selection, exploration.
- ExperimentUnrestricted.html – Experiment from section 7.3.1.
- ExperimentPreselected.html – Experiment from section 7.3.2.
- Benchmark.html – Benchmark model development.
- RandomForest.html – Random forest development.
- ExtraTrees.html – Extremely randomized trees development.
- AdaBoost.html – Adaptive boosting development.
- XGBoost.html – Gradient boosting development.

Eight Jupyter notebooks, the working versions of the above.

- DataPreparation.ipnb
- ExperimentUnrestricted.ipnb
- ExperimentPreselected.ipnb
- Benchmark.ipnb
- RandomForest.ipnb
- ExtraTrees.ipnb
- AdaBoost.ipnb
- XGBoost.ipnb

To keep the notebooks as readable as possible some of the developed functions were moved to the background. These can be found in following files.

- transformers.py – Data transformation utilities.
- plots.py – Plotting utilities based on Seaborn and Matplotlib.
- utils.py – Other general utility functions.

Other files:

- Profile.ipnb – Used to generate the BondoraProfile.html.
- BiasVar.py – Simulation from section 3.1.
- TreeExamples.py – Decision trees from section 4.4.2.
- ParDep.ipnb – Model from section 5.5.

Data:

- LoanData.csv – Raw dataset.

All the above mentioned files including this text can be found on the author's personal GitHub profile, available at: https://github.com/janhelcl.