Machine learning in credit default risk

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur / Master of Science

im Rahmen des Masterstudiums

Finanz- und Versicherungsmathematik (066 405)

eingereicht von

Alexander Petrov, BSc

Matrikelnummer 01525285

Ausgeführt bei: Institut für Stochastik und Wirtschaftsmathematik
Betreuer: Univ.-Prof. Dipl.-Ing. Dr.techn. Peter Filzmoser

Wien, am 8. Februar 2022

(Unterschrift Verfasser) (Unterschrift Betreuer)

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.

Wien, am 8. Februar 2022

Alexander Petrov

Einverständniserklärung zur Plagiatsprüfung

Ich nehme zur Kenntnis, dass die vorgelegte Arbeit mit geeigneten und dem derzeitigen Stand der Technik entsprechenden Mitteln (Plagiat-Erkennungssoftware) elektronisch-technisch überprüft wird. Dies stellt einerseits sicher, dass bei der Erstellung der vorgelegten Arbeit die hohen Qualitätsvorgaben im Rahmen der ausgegebenen der an der TU Wien geltenden Regeln zur Sicherung guter wissenschaftlicher Praxis - Code of Conduct (Mitteilungsblatt 2007, 26. Stück, Nr. 257 idgF.) an der TU Wien eingehalten wurden. Zum anderen werden durch einen Abgleich mit anderen studentischen Abschlussarbeiten Verletzungen meines persönlichen Urheberrechts vermieden.

Wien, am 8. Februar 2022

Alexander Petrov

Contents

1 Introduction **7**
1.1 Motivation . . . 7
1.2 Overview of Similar Works . . . 8
1.3 Binary Classification Problem . . . 9
1.4 Outlier Treatment . . . 9
1.5 Class Imbalance and Model Evaluation . . . 10

2 Data and Setup **11**
2.1 Sample . . . 11
2.2 Data Partitioning . . . 12
2.3 Categorical Variables . . . 13
2.4 Missing Data . . . 13
2.4.1 Missing Data Classification . . . 13
2.4.2 Fully Conditional Specification . . . 13
2.4.3 Predictive Mean Matching . . . 14

3 Logistic Regression **15**
3.1 Theory . . . 15
3.2 Application . . . 18
3.3 Regularized Logistic Regression . . . 20
3.3.1 Ridge Regression . . . 21
3.3.2 Lasso Regression . . . 23

4 Support Vector Machine **25**
4.1 Margin . . . 25
4.2 Lagrangian Theory . . . 28
4.3 Computing the Classifier . . . 29
4.4 Kernels . . . 31
4.5 Application . . . 31

5 K-Nearest Neighbours **34**
5.1 Theory . . . 34
5.2 Application . . . 35

6 Naive Bayes **37**
6.1 Theory . . . 37
6.2 Application . . . 38

7 Tree-based Methods **40**
7.1 Introduction . . . 40
7.2 Random Forest . . . 41
7.2.1 Theory . . . 41
7.2.2 Application . . . 42
7.3 Boosting Trees . . . 44
7.3.1 Motivation . . . 44

7.3.2 Exponential Loss - AdaBoost 44
7.3.3 Application 46

8 Neural Networks **47**
8.1 General Structure 47
8.2 Backpropagation 48
8.3 Application 50

9 Summary and Conclusions **52**

A Gini-coefficient / Somers' D **59**

B Spearman's Correlation **61**

C Comparison of Similar Works **62**

Kurzfassung

Das rasch zunehmende Volumen an Daten legt den Einsatz von fortgeschrittenen auf maschinellem Lernen basierten Modellen bei Banken nahe. Der im November 2021 erschienene Artikel der Europäischen Bankenaufsichtsbehörde (EBA) könnte demnächst neue Möglichkeiten für die Ermittlung der Kreditrisikoparameter nach dem IRB-Ansatz (internal rating-based) verschaffen.

Die vorliegende Diplomarbeit stellt einen Vergleich der diversen Klassifizierungsmethoden im Kreditrisikobereich auf, die zur Unterscheidung zwischen guten und schlechten Kunden bei der Bestimmung der Ausfallswahrscheinlichkeit dienen. Diese werden auf die Bilanzdaten der Geschäftskunden einer europäischen Bank angewendet, ergänzt durch die Land- und Industriekategorien sowie den binären einjährigen Ausfallsindikator als Zielvariable.

Die Algorithmen werden aus theoretischer Sicht beschrieben und mittels diverser R Pakete angewandt. Dabei bezieht die Datenaufbereitung einige Schritte mit ein, wie die Imputation der fehlenden Daten und die Behandlung von Ausreißern, die eine entscheidende Rolle spielt. Die Methodik umfasst weiters zwei Lösungsansätze für die unbalancierte Zielvariable sowie ein Kreuzvalidierungsverfahren.

Laut den Ergebnissen schneiden manche fortgeschrittene Methoden besser ab, als die logistische Regression und ihre Modifikationen, während die anderen Algorithmen ein vergleichbares Ergebnis liefern. Ein einfaches neuronales Netzwerk mit einer verborgenen Schicht ist das beste Model, wenn man eine uniforme Quantilentransformation auf die Variablen anwendet. Random forest erzielt das beste Ergebnis auf den untransformierten Daten, wobei die Interpretation und die Implementierung des Modells komplizierter werden, als im Falle der logistischen Regression.

Abstract

While the amount of data collected by banks increases exponentially, the introduction of sophisticated machine learning models becomes inevitable in order to keep up with the times. The European Banking Authority (EBA) published a discussion paper in November 2021 which might open new possibilities for the estimation of the risk parameters by the internal rating-based (IRB) approach.

This thesis aims to compare the performance of different machine learning algorithms in the field of credit risk and, more specifically, in the discrimination of good and bad customers as a part of the probability of default (PD) estimation. The data consists of the corporate customers of a European bank and their balance sheet positions enriched by the region and industry information with the 12 months default flag as the target variable.

The binary classification algorithms are described from the theoretical point of view and then applied using R packages. Thereby, the data pre-processing pipeline including an extensive missing data treatment as well as an outlier detection method plays a decisive role because of a significant noise level in the sample, while simultaneously addressing the problem of imbalanced data through undersampling and overweighting. A cross-validation procedure ensures that an adequate out-of-time generalization is achieved.

The results state that some of the advanced machine learning techniques outperform the ordinary logistic regression and its regularized modifications while the others such as support vector machine deliver a comparable performance. A plain neural network with one hidden layer provides the best predictions in terms of gini on the holdout sample using a uniform quantile transformation. Random forest achieves the best performance with the untransformed data, notwithstanding that the interpretation of the results and implementation of the model in production environment are less straightforward than in case of logistic regression.

1 Introduction

1.1 Motivation

While machine learning conquers different areas, the banking sector follows the trend. On the one hand, this can be seen as a logical development since banks are naturally in possession of large amounts of data. On the other hand, the financial institutions (and the supervisors) are trying to use the most state-of-the-art knowledge and technologies to quantify the risks. Most recently the use of machine learning models for the internal rating-based (IRB) models was addressed in the discussion paper [EBA, 2021] by the European Banking Authority (EBA) opening new possibilities and discussing how one can overcome the challenges such as interpreting and understanding the results without running into the risk of developing black box models. This is essential because credit default risk is one of great importance since the assessment of the creditworthiness of a customer contributes significantly to the risk management of the banks.

The spotlight problem is the prediction of the 12 months default flag of a customer. It is important to mention that the quantification of the exact probability of default (calibration) is not in scope of this thesis. The considered approaches aim to rank customers and discriminate good and bad risks.

Since it is of high importance for a risk manager to have a clear understanding of the algorithms used, each of the approaches is described from the theoretical point of view and the deterministic results are then compared. The studied models are:

- logistic regression (logit) and shrinkage methods ridge and lasso;
- Support Vector Machine (SVM);
- k-Nearest Neighbors (KNN);
- Naive Bayes (NB);
- Random Forest (RF);
- AdaBoost;
- Neural Network (NN).

1.2 Overview of Similar Works

Nowadays, the spotlight problem draws more and more attention and similar papers and works can be found. A few of them are summarized in Table 1.

Table 1: Comparison of similar works - summary

Source	Data	Evaluation
[Alonso and Carbo, 2021]	Consumer loans of Banco Santander, 75000 credits, 370 features (2/3 categorical), no time dimension	80/20 with 5-fold cross-validation
[Granström and Abrahamsson, 2019]	Nordea SME, 400 variables including behavioural data, 13 variables selected for the final model	cross-validation on training plus holdout, SMOTE
[Choubey, 2018]	Credit cards, 22 variables and recursive selection	cross-validation on training plus holdout
[Matre, 2019]	Retail mortages 2009-2017, 19 variables	training/test/validation 40/30/30
[Accenture, 2020] (1)	Loans, 23 variables	training/test/validation 70/20/10
[Accenture, 2020] (2)	Loans, 45 variables	training/test/validation 70/20/10

It is important to be aware of the fact that the results measured in gini (can be found in Appendix C) are not directly comparable to each other or to the results of this thesis. The underlying variables as well as the segments are different but also the methods of data pre-processing and sample splitting.

Nevertheless, one can get a rough idea of what can be expected. The tree-based methods random forest, AdaBoost and XGBoost method (which is related to AdaBoost) seem to be performing well in most of the cases. KNN and Naive Bayes have a rather poor performance. The neural networks neither seem to always be the model of choice.

Finally, the logistic regression is often as good or only slightly worse than the other machine learning algorithms but that is compensated by the interpretability and fast fitting of the model.

1.3 Binary Classification Problem

The main framework is a binary classification. The default flag for each of the $i = 1, \ldots, N$ customers is modeled as a binary random variable $Y_i \in \{0, 1\}$ with 0 being non-default and 1 corresponding to the default scenario. The explanatory variables will be presented in the next chapters. For the i-th customer one has a real-valued input $X_i \in \mathbb{R}^p$ with p variables. The evaluation of binary classification problems is described in Appendix A.

1.4 Outlier Treatment

Since the presence of outliers in the balance sheet data draws a great deal of attention, this issue will be addressed in two ways.

On the one hand, some balance sheet entry can lie outside of the range of values where the underlying position would normally be expected to be due to an error during data collection. While some algorithms can handle such situation in a robust manner (e.g. tree-based methods), the others might be strongly affected by those observations (KNN, Naive Bayes). Therefore, it will be addressed before applying those methods by using a suitable variable transformation.

On the other hand, the explanatory variables of some row might have inconspicuous values while the observation is mislabeled. This can be the case e.g. when a loan is incorrectly identified as default / non-default. [Brodley and Friedl, 1999] propose to remove the instances that do not follow the same pattern as the rest of the training data as they might come from a different probability distribution. The filtering is done by either applying one (single algorithm filter) or multiple classifiers (majority vote or consensus filter) on the data and dropping the misclassified data (determined during a cross-validation procedure) and then training the model of choice. While the three filters exhibit on average similar results, the gain in accuracy depends on the data set and the machine learning algorithm as well as the percentage of noise that was introduced artificially.

For the given problem it is proposed to use a similar two-step training procedure. The training takes place first on the full training sample and the mislabeled observations are defined based on the distribution of the scores of the two labels. For example, in case the predicted scores lie between 0 and 1 while the defaults have on average a higher score close to 1, the defaults that have a score lower than 75% of non-defaults and are close to zero (and vice versa) are excluded and the model is trained again. This corresponds to the single algorithm filter from [Brodley and Friedl, 1999] with the same algorithm used for both filtering and training but without additional cross-validation (it is computationally not feasible for most of the models). The proposed definition of outliers has the advantage of doing the job without explicitly defining the cut-off value for the binary classification.

1.5 Class Imbalance and Model Evaluation

Each model is trained on a training set with different hyperparameters. As a basic principle, the training is done on the **undersampled** balanced data sets where the number of defaults equals the number of non-defaults. The added value of undersampling is the following:

- the model is forced to consider the underrepresented class as equally important;
- the fitting of the model is accelerated since less observations are considered.

Another possibility to handle class imbalance is to use **weighting** for the two classes in order to increase the influence of misclassifying the underrepresented group on the model fitting e.g. loss function. This approach will be applied for some algorithms such as logistic regression and tree-based approaches and compared to the results from undersampling.

The main evaluation metric is the **gini** (see Appendix A). The hyperparameters from grid search with the best oot-performance (5-fold cross-validated out-of-time mean gini) are selected for the final model. In case multiple models have similar performances, the one with the least gap between the in-sample and out-of-sample gini is chosen (least overfitting). The model is then trained once more on the **full history** with the best hyperparameters and evaluated on the **holdout** sample previously unseen by the model (the last year Y11), Subsection 2.2 describes the approach in more detail.

At the end, this logic provides, on the one hand, a CV estimate, which is used to tune the model, and on the other hand, an estimate for the real-life performance e.g. when a quantitative analyst trains the given model on the whole available history and predicts the probability of default for the current portfolio with non-observable defaults (per definition the target variable is complete earliest 12 months after the reference date).

With that in mind, the tables containing the results will have the following structure at the end of each chapter:

- **sample**: **original** (output of the general data pre-processing) or **transformed** (additional steps applied to make the data suitable for the particular algorithm);
- **method**: **cv** (cross-validated (mean) results from hyperparameter-tuning) or **holdout** (single value);
- **imb**: the problem of imbalanced data is handled by undersampling (**under**) or overweighting (**weight**);
- **mislab**: mislabeled observations are removed (**1**) or not (**0**);
- **gini_in**: in-sample gini (training set);
- **gini_oot**: out-of-time gini (test set).

2 Data and Setup

2.1 Sample

The data set consists of the clients of a European bank from its corporate segment. The one-year default flag indicates whether a given client defaults in the next 12 months after the reference date. According to Basel II ([BCBS, 2006]), a default occurs when:

- the obligor is unlike to pay its credit obligations in full without giving up a collateral;
- the obligor is 90 days past due on any material credit obligation.

Hence each row corresponds to a customer at some point in time / month and the observations are unique in terms of the customer ID and date. Additionally, two categorical variables split the data set into sub-segments: **regions** and **industries**.

This table is merged with another table containing the balance sheets of the clients. For each observation the latest available balance sheet is selected and attached. If all financials of a given customer are older than **24 months** defined as the gap between the reference date and the balance sheet date and therefore there is no up-to-date financial information, the client is dropped from the sample. The change of the predictive power as the balance sheet gets older is an interesting topic worth investigating but is not in the scope of this thesis.

The data collection as well as the definition of (meaningful) financial ratios and default flags are neither in the scope. Both the anonymized raw balance sheet positions and the ratios are already provided and the differences are considered from the statistical point of view only.

Table 2: Sample preview

cust_id	year	default_1y	region	industry	x_fin_100	x_fin_200
CUST10025	Y9	0	REG4	IND2	0	94
CUST10051	Y1	0	REG4	IND6	2286	12
CUST10077	Y6	0	REG11	IND7	0	NA
CUST10101	Y10	0	REG4	IND2	0	139
CUST10130	Y5	0	REG4	IND1	400188	-9
CUST10160	Y4	0	REG3	IND2	0	19
CUST10187	Y3	0	REG4	IND3	445211	-36
CUST10225	Y10	0	REG4	IND1	0	56
CUST1026	Y10	0	REG2	IND1	7156	3
CUST10278	Y5	0	REG5	IND7	39	4

An important consideration is that balance sheets are not published on a monthly basis and for consistency only those with 12-months reporting period are considered. However, this also leads to the conclusion that the most of the neighboring timeslices of the same customer will naturally contain the same financials figures e.g. both in September and October the latest available data is most likely from December of the previous year. This

information does not bring any additional value for the further development and is therefore dropped: only the **non-overlapping** observations from one selected month of each year with a one-year gap in-between are considered in the next steps. The data structure is presented in Table 2.

At the start of the analysis, there are **626 numerical columns**. Those include balance sheet positions as well as financial ratios which capture some meaningful non-linear interactions between the raw positions e.g. dividing the sum of equities by the total assets results in the equity ratio.

2.2 Data Partitioning

At this stage, one has to keep in mind that a cross-validation approach will be used for model selection. However, it would have been incorrect to select the features on the whole sample by implicitly using the knowledge that the variable performs well or poorly on the test set which contradicts with the purpose of the test set and might lead to overly optimistic results. The goal is to imitate the process for the data which is available up until some point in time and then make predictions on the new data for which the target variable is not observed yet.

Therefore, the **cross-validation partitions** as presented in Figure 1 are defined now and a **separate variable selection** is performed for each of the training sets as recommended by [Hastie et al., 2017].

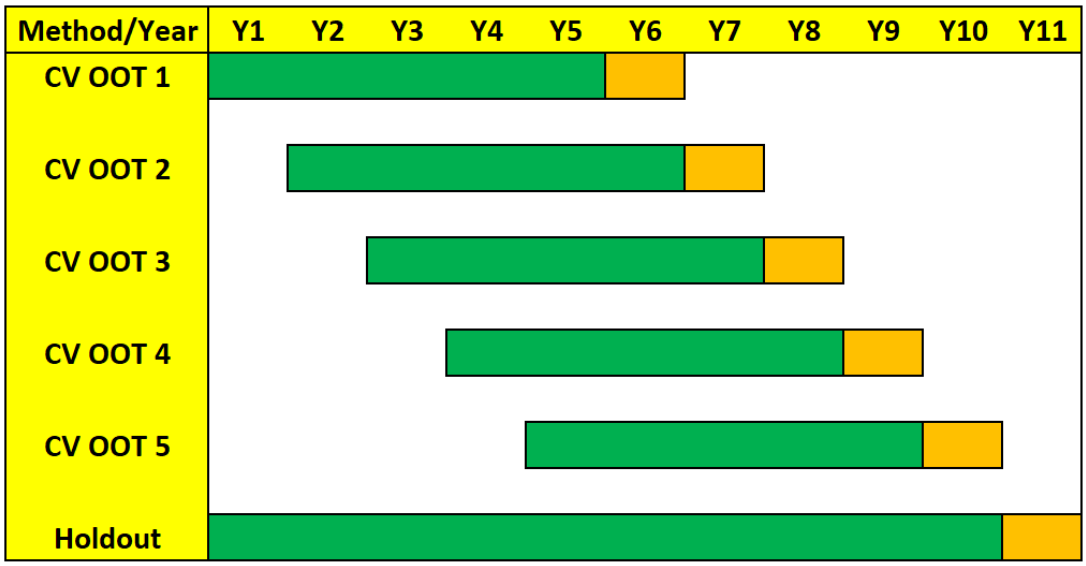

Figure 1: Training / Test structure

Each of the five training / test splits covers five years for the training purposes and a one-year test set. The sixth round is run on the ten years of history to make a prediction on the last year using the best hyperparameters (not to be confused with the model parameters that are re-estimated such as the regression coefficients).

All further steps are performed for each split separately unless stated otherwise.

2.3 Categorical Variables

Some regions and industries contain only few or none defaults. Therefore, the number of subgroups is reduced by combining similar segments (neighbouring countries, related industries). The two variables are then log-odd-transformed: the log-odds of each category are calculated on the training set and replace the factors in the training and test sets. This allows to treat all of the variables as numerical and order the sub-segments in terms of historical riskiness.

2.4 Missing Data

2.4.1 Missing Data Classification

The missingness of the data X can be characterized by defining the indicator matrix M of missing values in X and considering the conditional distribution of m_i given x_i (row i) $f(m_i|x_i, \phi)$ with some unknown parameter ϕ. [Little and Rubin, 2019] distinguish three cases when the data is missing:

- completely at random (MCAR) in case the missingness does not depend on the missing or observed data in x_i;
- at random (MAR) when the missingness depends only on the observed components of x_i;
- not at random (MNAR) if the distribution depends on the unobserved components of x_i.

The MAR assumption is plausible for balance sheets since the available positions might be satisfactory to explain the missingness of the other positions well. This is a good starting point for the further steps ([Van Buuren, 2018]).

2.4.2 Fully Conditional Specification

The given problem of multivariate missing data can be approached by applying the fully conditional specification. The multivariate distribution of the data is specified by the iteration through conditional densities of the missing variables ([Van Buuren, 2018]). Each missing variable is first imputed with a random draw from the observed values. The non-missing values of the given variable are then used as target variable to fit a model with other features used as explanatory variables. This algorithm is known as multivariate imputation by chained equations (MICE).

[Van Buuren, 2018] suggests that a low number of 5 iterations is often enough to reach convergence. However, one has to make sure that there are no high correlations between variables as well as no high rates of missing data. This is ensured by considering only variables with less than 25% percent missing rate and by excluding the variable with the higher missing rate in case two features have a Spearman's correlation of 80% or higher. The advantage of using the Spearman's correlation coefficient is described in Appendix B.

2.4.3 Predictive Mean Matching

After fitting the model, multiple possibilities are open on how to determine the value for imputation. Predictive mean matching (PMM) calculates the fitted value of the target variable for some underlying imputation model and randomly selects the value to be imputed from a set of donors (data points that are closest to the missing entry while the metric can vary). This ensures that the imputed value is realistic and respects the characteristics of the original variable such as skewness. It is, however, best to apply it on large data sets ([Van Buuren, 2018]), which is the case for this application.

At this point the outliers are not treated in any manner hence one needs a robust method. Random forest will be used which satisfies this criteria ([Van Buuren, 2018]). The following code chunks shows how the function from the miceRanger package by [Wilson, 2021] is applied.

```
rf_model <-
  miceRanger(
    dt_train,
    m = 1,
    maxiter = 5,
    num.trees = 10, \\
    returnModels = T,
    valueSelector = "meanMatch",
    meanMatchCandidates = 5
  )
```

m is set to 1 because only one imputation is done. Nevertheless, the imputation uncertainty is considered by doing different imputations across the CV folds. 10 trees are selected based on the results of the study by [Shah et al., 2014]. [Van Buuren, 2018] suggests to use 5 donors (meanMatchCandidates parameter) as the default value. The model is trained on the training sample only and applied on the training and test samples to avoid target leakage.

[Van Buuren, 2018] also discusses the variable selection for the imputation model. Since random forest is capable of choosing the best variable in each split, all a priori available information will be used. This also makes the MAR assumption more realistic.

3 Logistic Regression

Logistic regression is widely used in the credit risk modelling due to its relative simplicity and interpretability of the impact each of the variables has on the result. Therefore the logistic regression model will be considered as the baseline model for this thesis.

The idea is to model the posterior probability of default as a linear function in X. At the same time, the probabilities must sum up to one and lie between zero and one to make the output meaningful.

3.1 Theory

Consider applying the regular linear regression on the binary response variable $Y_i \in \{0, 1\}$ representing the i-th customer with intercept β_0 and coefficient vector β_1. It takes the form

$$Y_i = \beta_0 + \beta_1^T X_i + \epsilon_i \tag{1}$$

and since the error term ϵ_i has a zero expectation, it follows that

$$E[Y_i] = \beta_0 + \beta_1^T X_i \tag{2}$$

and assuming that Y_i is Bernoulli-distributed with $P(Y_i = 1) = PD_i$ the formula is transformed into

$$E[Y_i] = 1 \cdot PD_i + 0 \cdot (1 - PD_i) = PD_i \,. \tag{3}$$

Therefore, the mean response simply equals the probability of default PD_i given some values of the predictor vector X_i. This, however, leads to certain complications as described in [Kutner et al., 2005]:

- the binary response variable Y_i implies that also the error term ϵ_i can only take two values violating the normally distributed error assumption of the ordinary least squares regression;
- the variance of ϵ_i is X_i-dependent and therefore violates the assumption $V[\epsilon_i] = const.$ for all i because:

$$V[\epsilon_i] = V[Y_i] = E[Y_i^2] - E[Y_i]^2 = PD_i \cdot 1^2 + 0 - PD_i^2 = PD_i(1 - PD_i) \tag{4}$$

- PD_i is a probability and therefore the following must hold for the response function of choice:

$$0 \leq E[Y_i] \leq 1 \,. \tag{5}$$

One function which fulfills those constraints is the standard normal cumulative distribution function (or probit):

$$PD_i = \Phi(\beta_0 + \beta_1^T X_i) \,. \tag{6}$$

Alternatively (and frequently), the logistic distribution is used which has heavier tails [Kutner et al., 2005]:

$$PD_i = \frac{\exp(\beta_0 + \beta_1^T X_i)}{1 + \exp(\beta_0 + \beta_1^T X_i)} \,. \tag{7}$$

This results in the logit response function of the PD_i and one gets a linear model for the **log-odds** of the default event:

$$\ln\left(\frac{PD_i}{1 - PD_i}\right) = \beta_0 + \beta_1^T X_i \,. \tag{8}$$

Using the definition of the Bernoulli probability function and the fact that the Y_i are assumed to be independent for all i, one gets for the joint probability function of Y:

$$g(Y_1, \ldots, Y_N) = \prod_{i=1}^{N} PD_i^{Y_i}(1 - PD_i)^{1-Y_i} \,. \tag{9}$$

After applying the logarithm and inserting the Equation (7) from above, one ends up with the following Log-Likelihood function:

$$\begin{aligned}
l(\beta) = \ln(L(\beta)) &= \ln\left(\prod_{i=1}^{N} PD(x_i;\beta)^{y_i}(1 - PD(x_i;\beta))^{1-y_i}\right) \\
&= \sum_{i=1}^{N} \ln(PD(x_i;\beta)^{y_i}(1 - PD(x_i;\beta))^{1-y_i}) \\
&= \sum_{i=1}^{N} \ln(PD(x_i;\beta)^{y_i}) + \ln((1 - PD(x_i;\beta))^{1-y_i}) \\
&= \sum_{i=1}^{N} y_i \cdot \ln(PD(x_i;\beta)) + (1 - y_i) \cdot \ln(1 - PD(x_i;\beta)) \\
&= \sum_{i=1}^{N} y_i \cdot \ln\left(\frac{\exp(\beta^T x_i)}{1 + \exp(\beta^T x_i)}\right) + (1 - y_i) \cdot \ln\left(\frac{1}{1 + \exp(\beta^T x_i)}\right) \\
&= \sum_{i=1}^{N} y_i(\beta^T x_i) - \ln(1 + e^{\beta^T x_i})
\end{aligned} \tag{10}$$

with $\beta = (\beta_0, \beta_1)$ and the first component of x_i is 1. The derivatives are set to zero to maximize the log-likelihood:

$$\frac{\partial l(\beta)}{\partial \beta} = \sum_{i=1}^{N} x_i(y_i - \underbrace{\frac{e^{\beta^T x_i}}{1 + e^{\beta^T x_i}}}_{=PD(x_i;\beta)}) = 0 \tag{11}$$

also known as score equations. Because of the intercept in x_i, the first of the $p + 1$ equations which hide behind the Equation (11) has the form

$$\sum_{i=1}^{N} y_i = \sum_{i=1}^{N} PD(x_i;\beta) = \sum_{i=1}^{N} E[Y_i] \tag{12}$$

hence demands that the observed number of defaults equals the expected number of defaults.

The solution can then be found by iteratively applying the Newton-Raphson algorithm. The second derivative or Hessian matrix is calculated as

$$\frac{\partial^2 l(\beta)}{\partial\beta\partial\beta^T} = -\sum_{i=1}^{N} PD(x_i;\beta)(1-PD(x_i;\beta))x_i x_i^T \tag{13}$$

and β_{old} is updated as described in [Hastie et al., 2017] by

$$\beta_{new} = \beta_{old} - \left(\frac{\partial^2 l(\beta)}{\partial\beta\partial\beta^T}\right)^{-1} \frac{\partial l(\beta)}{\partial\beta}\bigg|_{\beta=\beta_{old}} . \tag{14}$$

To get more insight, one can write these equations in matrix notation. Let:

- y... 0/1 vector with N default flags y_i;
- X... $N \times (p+1)$ matrix with the intercept (1) in the first row and each of the other rows corresponding to a feature vector x_i;
- p... vector with N fitted probabilities of default $PD(x_i;\beta_{old})$;
- W... diagonal matrix with $PD(x_i;\beta)(1-PD(x_i;\beta))$ in the ith diagonal element.

Therefore, Equation (11) can also be written as

$$\frac{\partial l(\beta)}{\partial\beta} = X^T(y-p) \tag{15}$$

and Equation (13) as

$$\frac{\partial^2 l(\beta)}{\partial\beta\partial\beta^T} = -X^T W X \tag{16}$$

thus for the Newton update the following holds

$$\begin{aligned} \beta_{new} &= \beta_{old} + (X^T W X)^{-1} X^T (y-p) \\ &= (X^T W X)^{-1} X^T W (X\beta_{old} + W^{-1}(y-p)) \\ &= (X^T W X)^{-1} X^T W z\,. \end{aligned} \tag{17}$$

This can be seen as a weighted least-squares step with an adjusted response

$$z = X\beta_{old} + W^{-1}(y-p)\,. \tag{18}$$

At each iteration the β changes leading to a new p hence W and z are updated too. The problem that is solved can be expressed as

$$\beta_{new} \leftarrow \underset{\beta}{\operatorname{argmin}} (z-X\beta)^T W (z-X\beta)\,. \tag{19}$$

3.2 Application

Logistic regression requires several assumptions to be fulfilled ([Stoltzfus, 2011]). The independence of error terms is assumed to be fulfilled because each row corresponds to a different customer at a different point in time. Furthermore, one has to make sure that the data contains no strongly influential outliers, taking also into account that the variables have different scales (percentages in the range from 0 to 100, ratios between -1 and 1, absolute values with arbitrarily high or low values). Additionally, the distributions are mostly skewed, some of them are concentrated at different levels and / or show multimodal shape. Based on these facts it is proposed to transform each variable to a uniform distribution between zero and one (this will be referred to later as the uniform transformation). Outliers are then treated as neighbours of normal cases.

The next step is to ensure the linearity in the logit for the independent variables. It is fulfilled per definition for the log-odd-transformed categorical variables. The continuous variables are binned and a linear model is fit with the bin log-odds as the target variable and the mean value for each bucket bin as the independent variable. Testing whether the p-value of the model coefficient is below 5% leads to a decision whether to keep the variable for modelling or not.

The last issue to take care of is the multicollinearity / redundancy of the variables. It can be measured by the variance inflation factor (VIF) defined as

$$\mathrm{VIF} = \frac{1}{1 - R_i^2}\,, \tag{20}$$

where R_i^2 is the coefficient of determination when modelling the i-th variable with all other predictors from the model. The common rule of thumb is 5 indicating a high correlation.

Since not all of the selected variables might be statistically significant, the model is fit multiple times and in each step the independent variable with the highest p-value above 0.05 is dropped (stepwise approach).

The removal of mislabeled rows improves the performance on the raw variables, however, one can observe the main increase in the gini when applying the variables pre-processing including the uniform transformation as described earlier. In this case, the mislabeled observations do not have a high impact and only affect the in-sample gini.

Table 3: Logistic regression - results with undersampling

sample	method	mislab	gini_in	gini_oot
original	cv	0	0.55	0.49
original	cv	1	0.66	0.53
transformed	cv	0	0.63	0.58
transformed	cv	1	0.70	0.58

Similar situation is observed when fitting the model on the whole sample and overweighting the default class. The parameter *weights* in the *glm*() function is set equal to 1 for the

default class and to the default rate for non-defaults (inversely proportional to the class imbalance) and drives the overweighting in the log-likelihood function (Equation (10)) which is aquivalent to adding more observations while the weights are not necessarily integers.

Table 4: Logistic regression - results with overweighting

sample	method	mislab	gini_in	gini_oot
original	cv	0	0.39	0.36
original	cv	1	0.45	0.40
transformed	cv	0	0.62	0.59
transformed	cv	1	0.62	0.59
transformed	holdout	1	0.62	0.52

The result is slightly better than for undersampling and consequently this method is selected for the final model which has a significantly lower gini on the holdout sample. However, the final model does not not fully profit from the cross-validation approach because no hyperparameters are selected / tuned. This will be covered in the next subchapters. The goal is to use two enhanced versions of the logistic regression: ridge and lasso. Both of them are shrinkage methods which punish the size of the regression coefficients in a way that the model gives more stable results. Ridge regression will help to find the optimal coefficients for the given set of variables whereas lasso will be used to reduce the number of selected features without losing much of predictive power.

3.3 Regularized Logistic Regression

The idea is to penalize the size of the coefficients and thus tackle the problem of high variance by allowing more bias. This can be the case if one large positive coefficient is cancelled out by a large negative coefficient of another highly correlated variable.

Assume without loss of generality that the observations are standardized $\sum_{i=1}^{N} x_{ij} = 0$, $\frac{1}{N}\sum_{i=1}^{N} x_{ij}^2 = 1$ for each $j = 1, \ldots, p$. The model is fit on the log-likelihood function with additional penalty

$$\max_{\beta} \frac{1}{N} \sum_{i=1}^{N} \left[y_i(\beta^T x_i) - \ln(1 + e^{\beta^T x_i}) \right] - \lambda P_\alpha(\beta)\,, \tag{21}$$

where the penalty term is defined as

$$P_\alpha(\beta) = (1-\alpha)\frac{1}{2}||\beta||_2^2 + \alpha||\beta||_1 \tag{22}$$

with $\alpha = 0$ referring to the ridge regression and $\alpha = 1$ to the lasso regression.

The approach for finding the optimal β is described in [Friedman et al., 2010]. For each λ, a quadratic approximation l_Q of the log-likelihood (Taylor expansion about current estimates) is computed which equals

$$l_Q(\beta_0, \beta) = -\frac{1}{2N} \sum_{i=1}^{N} w_i(z_i - \beta_0 - x_i^T\beta)^2 + C(\beta_{old})^2 \tag{23}$$

with the following notation which is similar to the one from the previous chapter

$$z = X\beta_{old} + W^{-1}(y - p)\,, \tag{24}$$

$$W = diag(PD(x_i;\beta)(1 - PD(x_i;\beta)))\,. \tag{25}$$

For a given lambda, the coordinate descent algorithm is applied on the penalized and weighted least-squares problem

$$\min_{\beta} \left(-l_Q(\beta) + \lambda P_\alpha(\beta)\right) \tag{26}$$

with the update being done coordinate-wise

$$\beta_{new,j} = \frac{S\left(\sum_{i=1}^{N} w_i x_{ij}(y_i - y_{old,i}), \lambda\alpha\right)}{\sum_{i=1}^{N} w_i x_{ij}^2 + \lambda(1-\alpha)}\,, \tag{27}$$

where the $S(z,\gamma)$ is the soft-threshold operator defined by

$$sign(z)(|z| - \gamma)_+ = \begin{cases} z - \gamma & z > 0, \gamma < |z| \\ z + \gamma & z < 0, \gamma < |z| \\ 0 & \gamma \geq |z| \end{cases} \tag{28}$$

and $y_{old,i}$ is the fitted value but with the contribution from x_{ij} excluded

$$y_{old,i} = \beta_{old,0} + \sum_{l \neq j} x_{il}\beta_{old,l} \,. \tag{29}$$

Therefore, $y_i - y_{old,i}$ is the partial residual for fitting β_j.

3.3.1 Ridge Regression

Ridge shrinks the coefficients towards each other. In the (extreme) case of n identical variables, each coefficient will have the $1/n$ of the size it would have had in the univariate model.

The glmnet R package from [Friedman et al., 2010] provides an implementation of the elastic net algorithm which is a generalization of the ridge regression. The alpha is set to zero in order to use the L2-norm as shown in Equation (22). The further parameters of the *cv.glmnet*() function are:

- **family** is set equal to "binomial" to perform the logistic regression;
- **lambda** is a vector of lambdas for which the regression will be fitted;
- **standardize** is set to TRUE;
- **type.measure** defines the metric based on which the optimal lambda will be selected and is set equal to "auc";
- **nfolds** equal 10 indicates that a 10-fold cross-validation is performed for each lambda.

The two vertical dashed lines in the Figure 2 indicate the two optimal lambdas proposed by the package. The left one is the lambda.min value for which the AUC is maximized, the right one is the lambda.1se value that lies within one standard deviation from lambda.min and refers to the most regularized model (highest lambda) with this property.

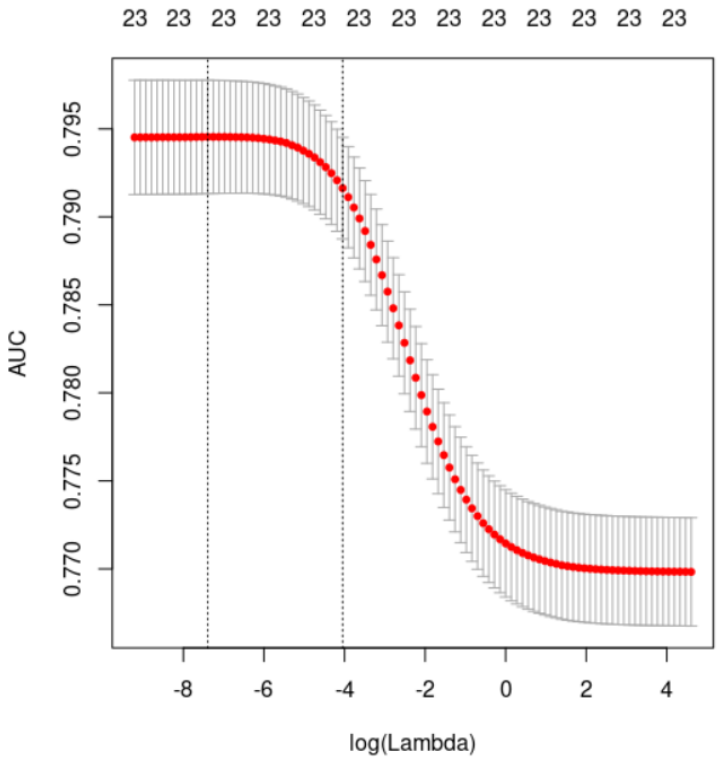

Figure 2: Optimal lambda in ridge - example

The out-of-time predictions are done with lambda.1se and for the holdout sample, the mean of those is used.
The feature preprocessing is the same as the one used for the logistic regression except that the multicollinear variables are not excluded since the elastic net is capable of dealing with such cases. Another difference is that regularization requires standardized feature. Otherwise, the penalty on the coefficients will be uneven for the variables with different scales, see Equation (22). This is handled by the function internally.

Table 5: Ridge regression - results

method	imb	mislab	lambda_1se	gini_in	gini_oot
cv	under	0	0.69	0.60	0.59
cv	under	1	0.30	0.69	0.59
cv	weight	0	0.22	0.62	0.60
cv	weight	1	0.26	0.62	0.60
holdout	weight	1	0.26	0.62	0.54

The performance is better than for the logistic regression. Again, the overweighting provides slightly better results than undersampling while the method seems to be robust against the mislabled observations.

3.3.2 Lasso Regression

The lasso approach is similar to ridge and uses the L1-norm as the penalty term:

$$\max_{\beta} \sum_{i=1}^{N} \left[y_i(\beta^T x_i) - \ln(1 + e^{\beta^T x_i}) \right] - \lambda \sum_{j=1}^{p} |\beta_j| \,. \tag{30}$$

The difference lies in the nature of the L1- and L2-norms as displayed in Figure 3 (from [Hastie et al., 2017]).

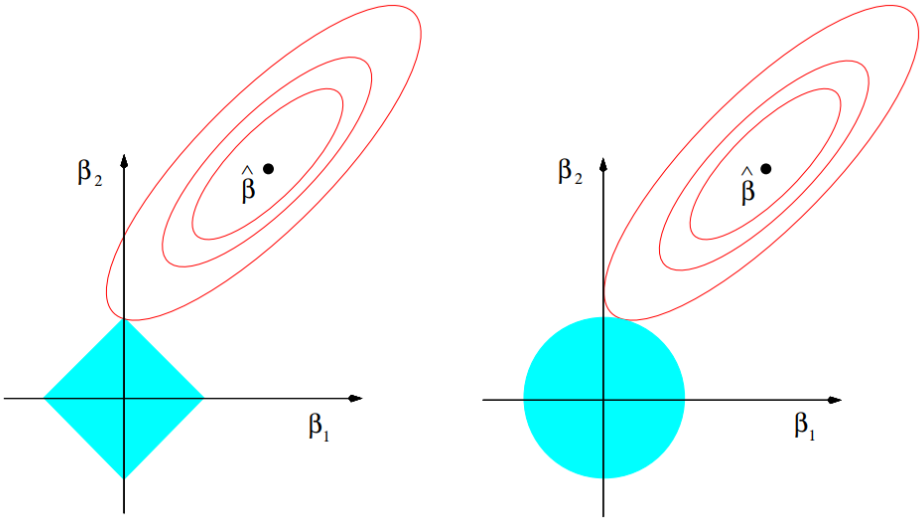

Figure 3: L1 vs L2 norm

The elliptical forms refer to the residual sum of squares (least-squares error function) for different coefficients with the center being the least-squares estimate. The coefficients must lie in the constraint region in the middle, however, those regions have different forms. In case of ridge, one has a circle because of

$$|\beta_1|^2 + |\beta_2|^2 \leq t \,, \tag{31}$$

whereas for lasso this has a diamond form with corners

$$|\beta_1| + |\beta_2| \leq t \,, \tag{32}$$

where the solution is more likely to be found and therefore the method does a kind of variable selection by setting more and more coefficients equal to zero as the lambda gets higher.

Again, the glmnet package will be used. The only difference is that the α parameter will be set to 1.
Figure 4 shows that decreasing the amount of features through increasing of lambda does not lead to any change in the AUC for quite long time, it remains stable.

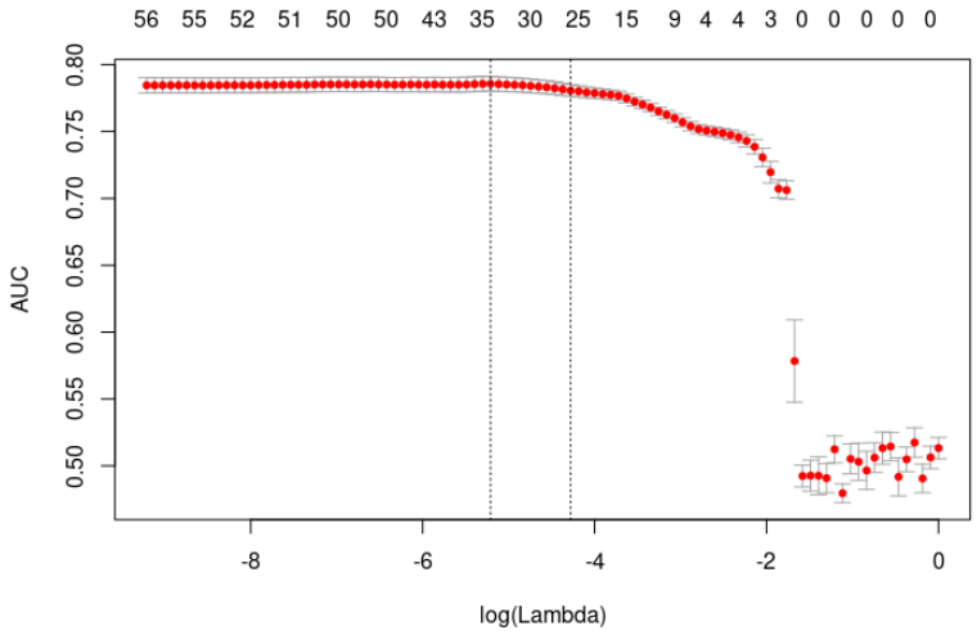

Figure 4: Optimal lambda in lasso - example

The results is as good as for ridge given that the number of variables used is strongly reduced.

Table 6: Lasso regression - results

method	imb	mislab	lambda_1se	gini_in	gini_oot
cv	under	0	0.02	0.61	0.58
cv	under	1	0.02	0.69	0.57
cv	weight	0	0.01	0.62	0.59
cv	weight	1	0.01	0.63	0.60
holdout	weight	1	0.01	0.63	0.54

4 Support Vector Machine

4.1 Margin

Consider a p-dimensional feature space and a two-class problem with the target variable $y_i \in \{-1, 1\}$ consisting of $i = 1, \ldots, N$ observations $x_i \in \mathbb{R}^p$. A hyperplane or affine set L defined by

$$L = \{x : f(x) = \beta_0 + x^T\beta = 0\} \tag{33}$$

with $||\beta|| = 1$ induces a classification rule $sign(f(x))$. In two or three dimensions, it is equivalent to building a line/plain and determining whether some x_i lies above it in case of $f(x_i) > 0$ or below it ($f(x_i) < 0$) [Hastie et al., 2017].

The hyperplane as presented in Figure 5 (from [Hastie et al., 2017]) has the following properties:

- for two points x_1 and x_2 which lie on L holds $\beta^T(x_1 - x_2) = 0$ and consequently, $\beta^* = \beta/||\beta||$ is the unit vector normal to the hyperplane;
- the signed distance of any point x to L is proportional to $f(x)$:

$$\begin{aligned} \beta^{*T}(x - x_0) &= \frac{\beta^T(x - x_0)}{||\beta||} \\ &= \frac{\beta^T x - \overbrace{\beta^T x_0}^{=-\beta_0}}{||\beta||} \\ &= \frac{\beta^T x + \beta_0}{||\beta||} \\ &= \frac{f(x)}{||f'(x)||}\,. \end{aligned} \tag{34}$$

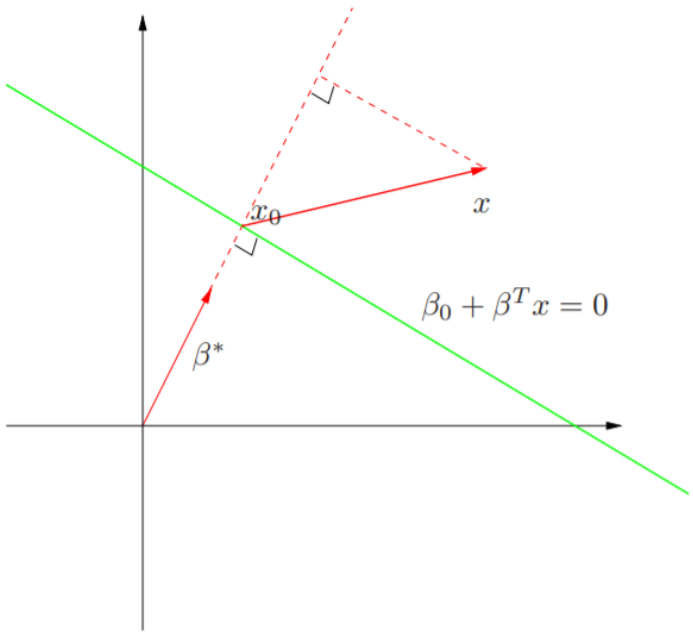

Figure 5: Hyperplane in 2D

It is possible that a hyperplane exists which perfectly separates the two classes which is equivalent to $y_i f(x_i) > 0$ by definition and one can then find an optimal hyperplane with the maximal margin M (distance to the nearest observations) which also defines the band of width $2M$ around $f(x)$ (see the left panel of Figure 6 from [Hastie et al., 2017]).

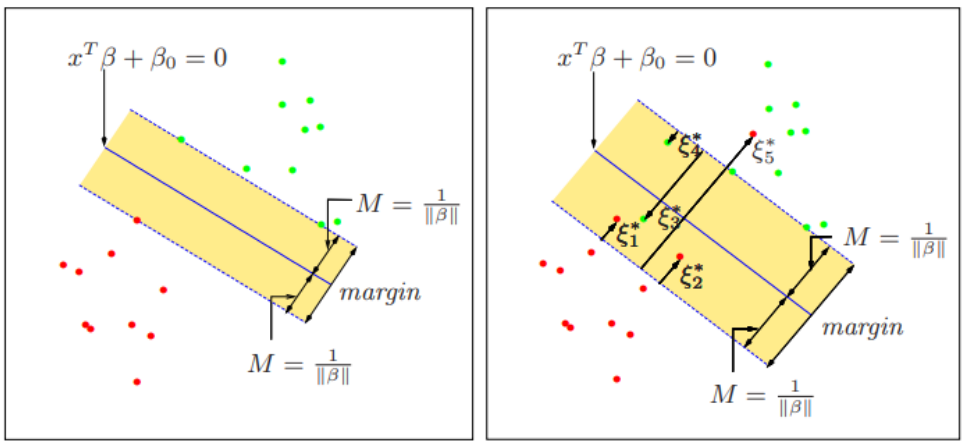

Figure 6: Separable vs. non-separable case

This setup can be written as a optimization problem of the form

$$\begin{aligned} &\max_{\beta,\beta_0,||\beta||=1} M\,, \\ &\text{subject to } y_i(x_i^T\beta + \beta_0) \geq M, i = 1, \ldots, N\,. \end{aligned} \tag{35}$$

In words: maximize the margin M as long as the distance to the points is at least M.

By setting $M = 1/||\beta||$, this can be further re-written as

$$\begin{aligned} &\min_{\beta,\beta_0} ||\beta|| \,, \\ &\text{subject to } y_i(x_i^T\beta + \beta_0) \geq 1, i = 1, \ldots, N \,, \end{aligned} \tag{36}$$

while the constraint $||\beta|| = 1$ is eliminated. This works because multiplying the left-hand side of the constraint in Problem 35 with $1/||\beta||$ eliminates the condition $||\beta|| = 1$. Any positively scaled multiple of some β, β_0 that satisfies the new condition also satisfies it. Hence M can be selected arbitrarily.

However, the classes often overlap in the feature space and one has to allow some observations to violate the rule and consequently to be misclassified leading to the so-called "soft-margin". The extent of the margin violation is given by the so-called slack variables $\epsilon_i, i = 1, \ldots, N$.
The soft-margin can be defined by changing the constraint either to

$$y_i(x_i^T\beta + \beta_0) \geq M - \epsilon_i \tag{37}$$

or

$$y_i(x_i^T\beta + \beta_0) \geq M(1 - \epsilon_i) \tag{38}$$

with $\epsilon_i \geq 0$, $\sum_i \epsilon_i \leq$ const.
The first version measures the overlap in absolute terms which seems to be more natural, however, leads to a non-convex optimization problem. The second option makes use of the relative distance in terms of M or in other words the proportional amount by which the x_i is placed on the wrong half-space. The sum of ϵ_i gives the total misclassification proportion that should be as small as possible. This leads to a convex optimization problem defined as follows ([James et al., 2017]):

$$\begin{aligned} &\max_{\beta,\beta_0} M \,, \\ &\text{subject to } y_i(\beta_0 + x_i^T\beta) \geq M(1 - \epsilon_i) \,, \\ &\epsilon_i \geq 0, \sum_{i=1}^{N} \epsilon_i \leq C, \|\beta\| = 1 \end{aligned} \tag{39}$$

or by getting rid of M

$$\begin{aligned} &\min_{\beta,\beta_0} ||\beta|| \,, \\ &\text{subject to } y_i(\beta_0 + x_i^T\beta) \geq 1 - \epsilon_i \,, \\ &\epsilon_i \geq 0, \sum_{i=1}^{N} \epsilon_i \leq C \,. \end{aligned} \tag{40}$$

In case $\epsilon_i = 0$ the observation is placed on the correct side of the margin (that in fact is equivalent to Problem 36) and is incorrectly classified in case of $\epsilon_i > 1$. The values

in-between represent the case when the points lie between the margin and the hyperplane. Therefore, C is an important tuning parameter:

- it bounds the number of observations allowed to be on the wrong side hence
- higher values less overfitting.

Consequently, only the observations which lie on the margin or violate it, affect the hyperplane and are referred to as "support vectors" [James et al., 2017]. This fact leads to the conclusion that outliers or point lying far inside of one of the half-spaces do not impact the decision boundary.

[Hastie et al., 2017] suggests to use the equivalent form

$$\begin{aligned} \min_{\beta,\beta_0} \frac{1}{2}||\beta||^2 + C\sum_{i=1}^{N} \epsilon_i \,, \\ \text{subject to } y_i(\beta_0 + x_i^T\beta) \geq 1 - \epsilon_i \,, \\ \epsilon_i \geq 0 \,, \end{aligned} \tag{41}$$

which is basically a regularized optimization problem. Note that the parameter C has an inverse effect here compared to previous steps: a higher C makes the algorithm less tolerant of margin violations.

4.2 Lagrangian Theory

The explanation in this subsection follows the lecture by [Palomar, 2020].

The Problem 41 can be generalized as

$$\begin{aligned} \min & f_0(x) \,, \\ \text{subject to } & f_i(x) \leq 0 \; i = 1, \ldots, m \,, \\ & h_j(x) = 0 \; j = 1, \ldots, p \,. \end{aligned} \tag{42}$$

The Lagrangian primal function is defined by

$$L(x, \alpha, \lambda) = f_0(x) + \sum_{i=1}^{m} \alpha_i f_i(x) + \sum_{j=1}^{p} \lambda_i h_i(x) \,. \tag{43}$$

By setting the x derivatives to zero and substituting, one gets the Lagrangian dual function defined by

$$g(\alpha, \lambda) = \inf_x L(x, \alpha, \lambda) \,. \tag{44}$$

This leads to the so-called Lagrangian dual problem

$$\begin{aligned} &\max_{\alpha,\lambda} g(\alpha,\lambda)\,, \\ &\text{subject to } \alpha \geq 0\,, \end{aligned} \tag{45}$$

which is a convex optimization problem (maximization of a concave function with linear constraints).

The weak duality theorem states that maximizing the dual problem produces a bound

$$g(\alpha,\lambda) \leq f_0(x^*) \tag{46}$$

for all $\alpha \geq 0$ and λ. x^* is the optimal solution of the primal problem. Therefore it holds in particular for the optimal dual solution (weak duality)

$$g(\alpha^*,\lambda^*) \leq f_0(x^*)\,. \tag{47}$$

The Slater's Theorem or Strong Duality Theorem says that if the constraint functions are affine, the duality gap (the gap between $g(\alpha^*,\lambda^*)$ and $f_0(x^*)$) is zero. Then the Karush-Kuhn-Tucker conditions provide the sufficient conditions for x^* to be an optimum:

- the first-order derivative of optimality is given by $\frac{\partial L(x,\alpha^*,\lambda^*)}{\partial x}\Big|_{x=x^*} = 0$
- the complementary slackness conditions are defined by $\alpha_i^* f_i(x^*) = 0$
- dual constraints $\alpha_i^* \geq 0$
- and finally the prime constraints $f_i(x^*) \leq 0$ and $h_j(x^*) = 0$.

4.3 Computing the Classifier

With the tool described previously, one can now attack the Problem (41). This is possible because the strong duality obviously holds. The prime problem has $2N$ constraints and the Lagrange primal function equals

$$\begin{aligned} L(\beta,\epsilon,\alpha,\mu) =& \frac{1}{2}||\beta||^2 + C\sum_{i=1}^{N}\epsilon_i \\ &- \sum_{i=1}^{N}\alpha_i(y_i(x_i^T\beta+\beta_0) - (1-\epsilon_i)) - \sum_{i=1}^{N}\mu_i\epsilon_i\,. \end{aligned} \tag{48}$$

This is to be minimized with respect to β, β_0 and ϵ_i, therefore, the respective derivatives are set to zero:

$$\beta = \sum_{i=1}^{N}\alpha_i y_i x_i\,, \tag{49}$$

$$0 = \sum_{i=1}^{N} \alpha_i y_i \,, \tag{50}$$

$$\alpha_i = C - \mu_i \,. \tag{51}$$

The substitution into Equation (48) gives the Lagrangian dual objective function and consequently a lower bound for the objective function ([Hastie et al., 2017]):

$$L_D(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j \,. \tag{52}$$

Additionally to the prime constraints, the Karush-Kuhn-Tucker conditions are the following:

$$\alpha_i (y_i (x_i^T \beta + \beta_0) - (1 - \epsilon_i)) = 0 \,, \tag{53}$$

$$0 \leq \alpha_i \leq C \,, \tag{54}$$

$$\mu_i \epsilon_i = 0 \,. \tag{55}$$

The solution of the optimization problem comes from the Equation (49)

$$\hat{\beta} = \sum_{i=1}^{N} \hat{\alpha}_i y_i x_i \,, \tag{56}$$

where the coefficients $\hat{\alpha}_i$ are non-zero only for the support vectors. The observations which lie on the edge of the margins ($\epsilon_i = 0$) can then be identified by $C > \hat{\alpha}_i > 0$ and the others by $\hat{\alpha}_i = C$ (and $\epsilon_i > 0$). $\hat{\beta}_0$ can be determined with the help of Equation (53): it can be calculated for any point with $\epsilon_i = 0$ and $\hat{\alpha}_i > 0$. For numerical stability, the average of these solutions is used ([Hastie et al., 2017]).

Coming back to the $f(x)$, it means that for some new data u ([Kuhn and Johnson, 2013]):

$$\begin{aligned} f(u) &= \hat{\beta}_0 + u^T \hat{\beta} \\ &= \hat{\beta}_0 + \sum_{j=1}^{p} \hat{\beta}_j u_j \\ &= \hat{\beta}_0 + \sum_{j=1}^{p} \sum_{i=1}^{N} \hat{\alpha}_i x_{ij} u_j \\ &= \hat{\beta}_0 + \sum_{i=1}^{N} \hat{\alpha}_i \Big(\sum_{j=1}^{p} x_{ij} u_j \Big) \,. \end{aligned} \tag{57}$$

The sum in the brackets in the last line is nothing else than a dot / inner product of the new data u_i and the i-th training observation x_i. It also refers to the so-called linear kernel which will be discussed and modified in the subsection below.

4.4 Kernels

The last result can be re-written with the help of the kernel function K which is a generalized version of the inner product of two transformed vectors ([James et al., 2017]):

$$K(x_i, x_{i'}) = \sum_{j=1}^{p} h(x_{ij})h(x_{i'j}) \,, \tag{58}$$

$$f(u) = \hat{\beta}_0 + \sum_{i=1}^{n} \hat{\alpha}_i K(u, x_i) \,, \tag{59}$$

by using a transformation $h(.)$. Therefore, the decision-making is based on the weighted sum where K can be seen as a measure of similarity between new data u and each and every support vector x_i. The optimization problem in fact doesn't involve the $h(x)$ explicitly but only the kernel itself which computes the dot products in some transformed space where the decision boundary is linear but simultaneously non-linear in the original feature space ([Hastie et al., 2017]). In case of the polynomial kernel, one implicitly gets the power transformations of the original variables as well as their interaction terms.

In the R package e1071 by [Meyer et al., 2019] the four popular kernels $K(u, v)$ are available:

- **linear:** $u'v$
- **polynomial:** $(\gamma u'v + \text{coef0})^{\text{degree}}$
- **radial basis:** $e^{-\gamma||u-v||^2}$
- **sigmoid:** $tanh(\gamma u'v + \text{coef0})$

with γ, coef0, degree as well as the cost factor C being the tuning parameters.

4.5 Application

Support vector machine requires variables to be scaled in order to avoid a variable with greater numeric ranges to dominate the inner product ([Hsu et al., 2003]). Furthermore, SVM is not robust for highly skewed variables ([Siddiqui and Ali, 2016]). Therefore, the uniform transformation will be applied on the input data. The best result on the untransformed data (see Table 7) is rather poor, although significantly better when removing mislabeled observations.

Table 7: SVM - results on the untransformed sample

method	mislab	kernel	cost	gini_in	gini_oot
cv	0	linear	100	0.17	0.15
cv	1	linear	100	0.52	0.30

In case of the uniform transformation, all four kernels lead to very similar cross-validated

results (see Table 8) as well as holdout ginis (see Table 9). The outcome for the polynomial kernel doesn't seem to be significantly better than the performance of the linear kernel. This leads to the conclusion that the power transformation of the variables as well as interaction terms do not give any added value for this particular problem.

Table 8: SVM - CV results with uniform transformation

mislab	kernel	cost	gamma	gini_in	gini_oot
0	linear	1e-02	NA	0.61	0.59
1	linear	1e-02	NA	0.69	0.59
0	polynomial	1e+02	1e-03	0.62	0.60
1	polynomial	1e+02	1e-03	0.69	0.59
0	radial	1e+03	1e-05	0.63	0.60
1	radial	1e+03	1e-05	0.70	0.60
0	sigmoid	1e+02	1e-03	0.64	0.60
1	sigmoid	1e+02	1e-03	0.72	0.60

Table 9: SVM - holdout results with uniform transformation

kernel	gini_in	gini_oot
linear	0.62	0.55
polynomial	0.62	0.55
radial	0.63	0.55
sigmoid	0.64	0.53

Another important observation is that the radial kernel is overfitting for higher gamma values as shown in Figure 7.

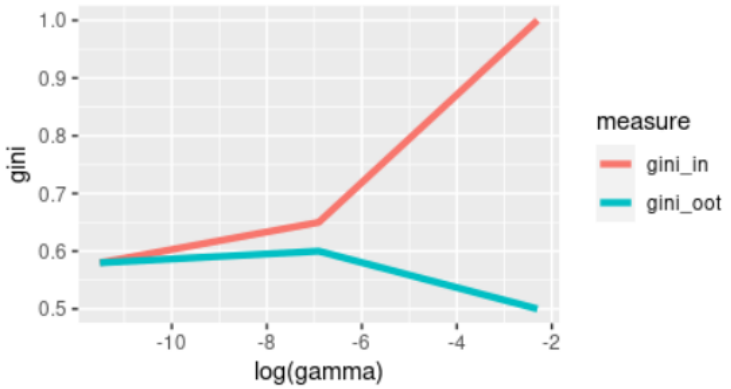

Figure 7: Radial kernel and the gamma parameter - example

A similar situation can be observed for the polynomial kernel, see Figure 8.

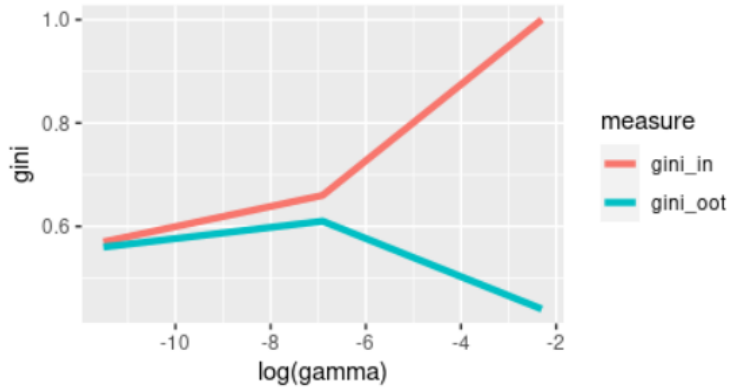

Figure 8: Polynomial kernel and the gamma parameter - example

Hence it is important to select the gamma parameter based on the cross-validated out-of-time gini for a good performance on the holdout sample.

The overweighting is computationally infeasible with the given resources. The function provides the possibility to define the *class.weights* that scale the cost parameter C in Equation (41) in order to increase the misclassification penalty of the underrepresented group.

5 K-Nearest Neighbours

5.1 Theory

The K-nearest neighbours algorithm (or simply KNN) captures the idea that the observations with the same label have similar features in terms of the distance which is measured by some metric. Often the Euclidean distance is used

$$d(x, y) = ||x - y||_2 \,. \tag{60}$$

One computes the distances of each and every observation from the training set to the given test observation. The k closest points are taken and the label is assigned based on the majority voting upon the neighbours. Therefore, the training of the algorithm incorporates both training and test set in order to identify the optimal value of k.

Another important consideration is that heavily imbalanced data might lead to a poor performance on the default class for high k. This is due to the fact that the density of the dominating class is simply so high that it will always dominate the decision and the defaults will be misclassified. For this reason, only the undersampled data set will be considered.

The variables should be standardized in order to assure that the influence of one given variable doesn't come from the larger scale of its values. The presence of outliers makes it necessary to use a robust scaler e.g. the IQR scaler defined as

$$x_t = \frac{x_{raw} - Q_{25}}{Q_{75} - Q_{25}} \tag{61}$$

for the raw data x_{raw} and the quantiles of the variable distribution Q_{25} and Q_{75}.

A further improvement can be achieved by incorporating the covariance structure of the data set by using the Mahalanobis distances defined for two points x and y as

$$M(x, y) = \sqrt{(x - y)^\intercal \Sigma^{-1} (x - y)} \tag{62}$$

with the covariance matrix Σ.

Since the data contains outliers there is a need for a robust covariance estimator. One common approach is to use the minimum covariance determinant (MCD). For a training set of size n one selects a subset of size $0.5 \cdot n \leq h \leq n$ with the lowest determinant of the covariance matrix. Then the scatter estimate is proportional to the covariance matrix of those h points. This is equivalent to finding h rows with a tolerance ellipsoid (for a given level) of the minimal volume around them ([Rousseeuw and Leroy, 1987]). In practice, one cannot consider all possible combinations and can take the best one upon some high number of subsets.

The advantage of using the MCD estimate is visualized in Figure 9 (from [Rousseeuw and Leroy, 1987]). While the classical covariance estimate would give a tolerance ellipse

which is distorted by outliers (dashed line), the MCD estimate ignores the outliers and consideres only the points from the "true" distribution:

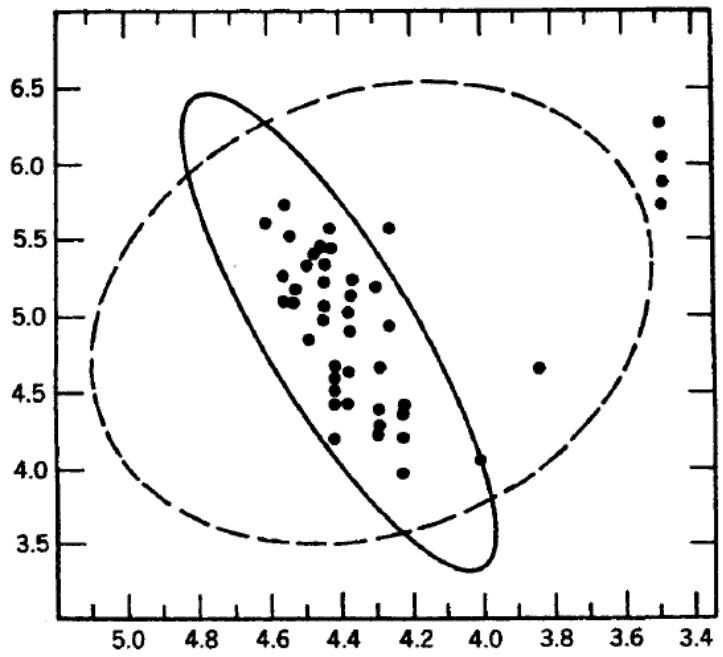

Figure 9: Gini for different k and transformations

The MCD estimator has a high breakdown point (robustness against a certain percentage of outliers). If the fraction of the outliers is at most $0 < \alpha \leq 0.5$ then one can consider subsets up to $h = [n \cdot (1 - \alpha)] + 1$ ([Rousseeuw and Leroy, 1987]).

5.2 Application

The knn function from the R package class by [Venables and Ripley, 2002] is used to run the algorithm with the Euclidean metric. The CV result is only around 24% (for $k = 21$).

The results are significantly further improved by using the Mahalanobis distances with the robust covariance estimator, see Figure 10. Since there is no R implementation of such algorithm, it is programmed from scratch. The MCD covariance estimator is computed based on 1000 runs and a fraction of 75% using the MASS package by [Venables and Ripley, 2002]. The Mahalanobis distances are computed between each and every observation in training and test sets and used to determine a score between 0 and 1 as a proportion of the defaulted cases upon the k nearest observations.

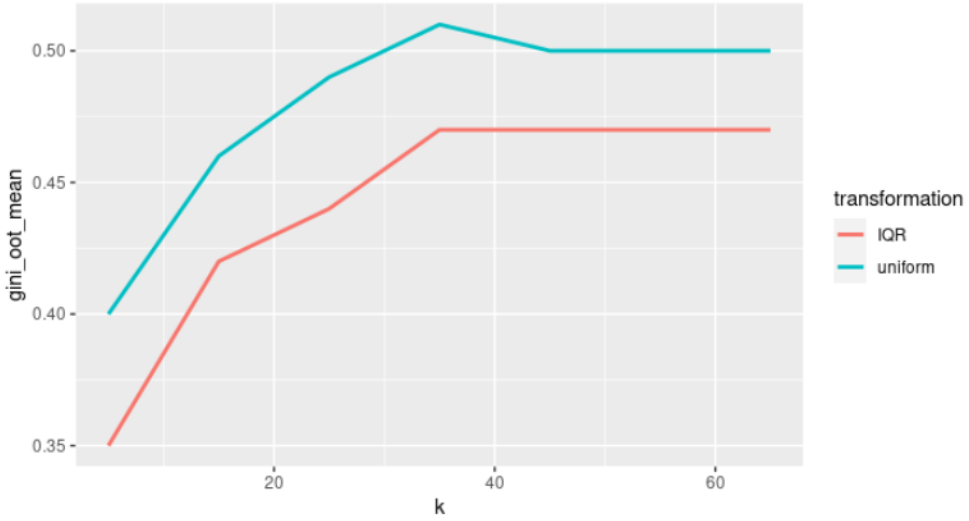

Figure 10: Gini for different k and transformations

The uniform transformation improves the gini even more while removing mislabeled observations improves the result slightly for the optimal $k = 35$.

Table 10: Robust KNN - results (for k=35)

method	sample	mislab	gini_in	gini_oot
cv	IQR	0	0.54	0.47
cv	IQR	1	0.62	0.46
cv	uniform	0	0.58	0.50
cv	uniform	1	0.66	0.51
holdout	uniform	1	0.67	0.46

6 Naive Bayes

6.1 Theory

The Naive Bayes method is based on the famous Bayes formula

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{63}$$

or rather on its generalized form, the Bayes Theorem

$$P(Y = k|X = x) = \frac{p(X = x|Y = k) \cdot P(Y = k)}{p(X = x)}\,. \tag{64}$$

Let Y be the default flag as a random variable and X represent the independent variables. The $P(Y = 1)$ corresponds to the prior probability of default and represents the proportion of defaults in the training sample. The term $p(X = x|Y = k)$ corresponds to the likelihood (conditional density) of the observed feature vector x given the class $k = 1$ (default) or $k = 0$ (non-default), see Figure 11.

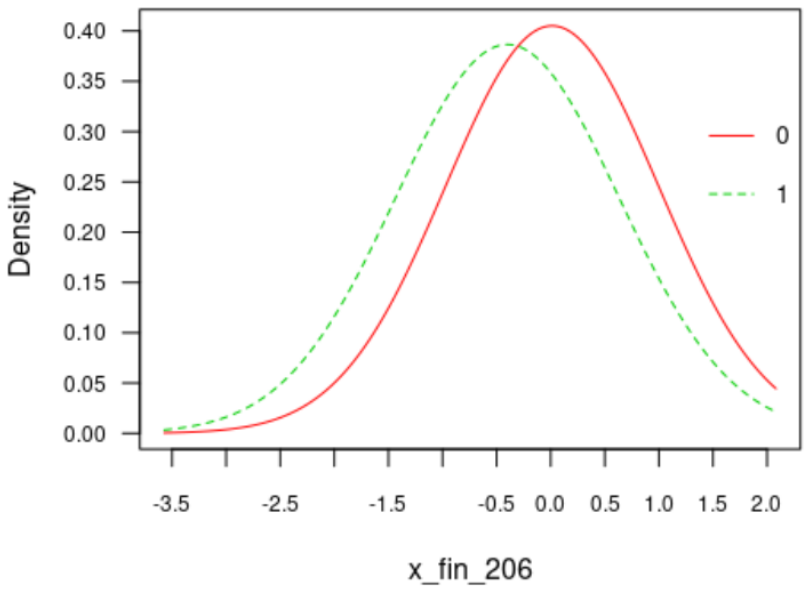

Figure 11: Posterior densities of a variable - example

The "naive" part of the algorithm is that the features are assumed to be conditionally independent which is a strong assumption that is rarely fulfilled. It leads, however, to the significant simplification that the a-posteriori density can be written as a product of the single densities of the p features:

$$p(X = x|Y = k) = \prod_{i=1}^{p} p(X_i = x_i|Y = k)\,. \tag{65}$$

One can take for the single $X_i|Y = k$ a normal distribution (another strong assumption) with μ_i and σ_i being the estimates from the training sample:

$$p(X_i = x|Y = k) = \frac{1}{\sigma_i\sqrt{2\pi}} e^{-(x-\mu_i)^2/2\sigma_i^2} \tag{66}$$

with estimates μ_i and σ_i for the expectance and standard deviation of the variable i. The term $p(X = x)$ in the denominator of the Equation (64) can be computed with the law of total probability via the joint probability of $X = x$ and $Y = k$

$$\begin{aligned} p(X = x) &= p(X = x, Y = 1) + p(X = x, Y = 0) \\ &= p(Y = 1)p(X = x|Y = 1) + p(Y = 0)p(X = x|Y = 0)\,. \end{aligned} \tag{67}$$

A new unlabeled sample is then classified based on the maximum posterior probability $P(Y = k|X = x)$. Often, the denominator can be even dropped since it stays the same for all k and the prediction is done by determining the class k which the numerator is maximal for:

$$y = \text{argmax}_k \left\{ P(Y = k) \cdot \prod_{i=1}^{p} p(X_i = x_i|Y = k) \right\} \tag{68}$$

or equivalently on the logarithmic scale in order to avoid numerical issues arising when multiplying small numbers:

$$y = \text{argmax}_k \left\{ \log(P(Y = k)) + \sum_{i=1}^{p} \log(p(X_i = x_i|Y = k)) \right\}\,. \tag{69}$$

However, this will not be done since the given problem requires the probability of default as output and not only the class prediction.

6.2 Application

The independence of the variables is almost never fulfilled in practice ([Kuhn and Johnson, 2013]). Nevertheless, the variables with the strongest correlations have already been eliminated earlier. Another important aspect to consider is the estimation of the mean and variance parameters because the presence of outliers might bias those significantly. Therefore, the uniform transformation is applied. Alternatively, one can use robust estimates for scale and variance, see e.g. [Ahmed et al., 2017].

The problem of imbalanced classes is of no concern for this algorithm and evaluation metric because the $P(Y = k)$ term in Equation (64) does not change the risk ordering of the customers. Hence the full training sample will be utilized.

The R package naivebayes by [Majka, 2019] is used. The performance on the raw data is poor and improved drastically by the uniform transformation, see Table 11.

Table 11: Naive Bayes - results

method	sample	mislab	gini_in	gini_oot
cv	original	0	0.03	0.05
cv	original	1	0.03	0.04
cv	transformed	0	0.57	0.57
cv	transformed	1	0.57	0.57
holdout	transformed	0	0.57	0.56

A further attempt to improve the performance is done by using only the best MCD subset from the previous chapter, see Table 12. The idea is to exclude outliers and allow a robust estimation of the parameters for the normal distribution. That works for the raw data but the gini of the transformed sample remains unchanged. Same holds for the removal of mislabeled data.

Table 12: Naive Bayes - results with MCD

method	sample	mislab	gini_in	gini_oot
cv	original	0	0.38	0.37
cv	original	1	0.39	0.38
cv	transformed	0	0.58	0.57
cv	transformed	1	0.57	0.57

7 Tree-based Methods

7.1 Introduction

A decision tree can be interpreted as the well-known if/elseif/else statement in the world of algorithms: the prediction is based on a chain of yes/no checks which divide the feature space into rectangular decision areas. Figure 12 shows a simple example of a such decision tree.

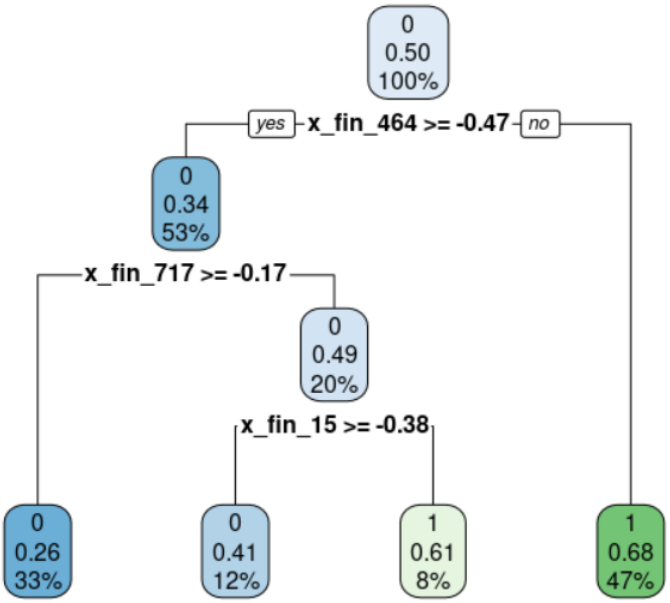

Figure 12: Decision tree - example

Each split is shown as a threshold of a certain variable based on which the decision is made. Furthermore, every node contains three figures:

1. the predicted class of the observations in the node based on the proportions of the classes;
2. the proportion of the defaults;
3. the percentage of the observations which the node contains out of all observations.

This results in the following strength of such models [Kuhn and Johnson, 2013]:

- high interpretability;
- no need to pre-process variables, the splits can handle categoricals;
- graphical visualization.

Nevertheless, one single tree is often insufficient due to its instability when introducing even slight changes in the train data resulting in poor performance on the test set (high degree of overfitting) [Kuhn and Johnson, 2013]. To solve this issue, different algorithms

were invented which use ensembling techniques to stabilize the results. One can also use a test set / cross-validation approach to determine the right tree size for the optimal out-of-sample performance. The process of growing a tree is discribed in the next chapter.

Due to their structure, the decision-tree-based algorithms are robust against outliers in the explanatory variables and highly skewed data, which can also be proven by empirical studies e.g. [Siddiqui and Ali, 2016]. Therefore, no transformation will be applied on the explanatory variables since any order-preserving transformation will not affect the results.

7.2 Random Forest

7.2.1 Theory

Random forest is a special case of a more general approach called bagging (**B**ootstrap **AGG**regation), where on each of the bootstrap samples, a separate tree is generated and the average of their predictions gives the final prediction. Random forest goes even further and introduces a new random component: a random selection of variables considered at each split. The resulting trees are even more decorrelated leading to a variance reduction ([Kuhn and Johnson, 2013]).

The splits are made based on the gini impurity (not to be confused with Somers' D). Let $p_{m,k}$ represent the proportion of the class k in the node m, then the impurity is a measure of how frequently some element from the node would be incorrectly classified if it was randomly labeled according to the distribution of classes in the node or ([James et al., 2017])

$$Imp_m = \sum_{k=1}^{K} p_{m,k}(1 - p_{m,k})\,. \tag{70}$$

For each of the splits considered in the given step, the ginis of the two resulting nodes are calculated and weighted based on the number of observations. The split with the lowest weighted impurity is selected (gini gain below certain threshold can be used as a stopping criteria for growing a single tree). This further provides a basis for a natural variable importance measure: a high mean gini gain (impurity of the root node minus the weighted leaf impurity) of a given variable indicates a high predictive power, see Figure 13.

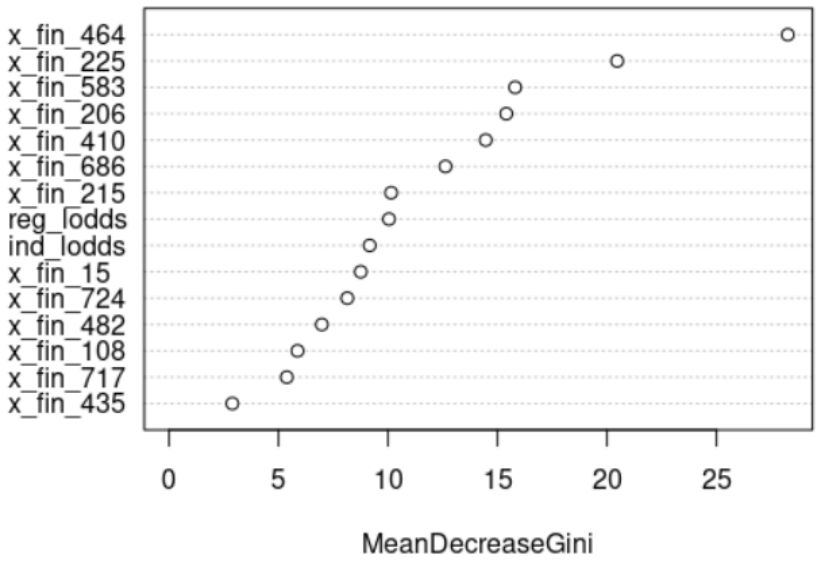

Figure 13: Gini impurity - example

An alternative approach suggests to score the out-of-bag (OOB: data that was not used in the particular bootstrap sample, 1/3 on average) and then repeat it but with the selected variable permuted. The decrease in accuracy (averaged over all trees) provides a measure of variable importance ("OOB randomization", [Hastie et al., 2017]).

Random forest can be used for regression as well as classification tasks. In case of classification, the prediction of each tree counts as a vote and the proportion of the outcomes provides the predicted probabilities for the classes.

7.2.2 Application

The R package randomForest by [Liaw and Wiener, 2002] is one of the many implementations of this algorithm in R. The user can tune the following (hyper-)parameters:

- **ntree** defines the number of trees to grow. The increasing ntree does not lead to overfitting because of the decorrelation. Therefore, it should be high enough so that adding more trees does not further improve the oos performance but keeping in mind that the computational power is limited;
- **mtry** is the number of predictors randomly chosen at each split as described previously. The default value for classification is $\sqrt{p}$ (and p/3 for regression). A low value leads to a higher variability across the trees since sub optimal variables are selected more frequently while a high mtry implies that the strong predictors do not let the others to influence the model. Hence it is of high importance to tune this parameter carefully;
- **replace and sampsize** control the bootstrapping. Those values will be left at the default settings (bootstrap with replacement of training sample size);

- **nodesize and maxnodes** define the minimum size and the maximum number of the terminal nodes and therefore are the main parameters to control the tree depth hence to combat the overfitting. Although they are closely related to each other and e.g. increasing nodesize / decreasing maxnodes both lead to smaller trees, the interaction between them is an interesting aspect which will be analysed empirically on the data.

- **classwt** will be used for overweighting of the default class by applying a higher weight on those observations during the voting process as well as for the selection of the best split.

While 200 trees are enough to achieve optimal results with undersampling, it takes 5000 trees for the full sample while the gini is significantly worse. In the latter case, the removal of the mislabeled observations helped to achieve a minor improvement in the gini.

Table 13: Random Forest - results

method	imb	mislab	ntree	nodesize	maxnodes	gini_in	gini_oot
cv	under	0	200	300	50	0.64	0.58
cv	under	1	200	300	50	0.69	0.58
cv	weighted	0	5000	300	5	0.54	0.43
cv	weighted	1	5000	300	5	0.56	0.45
holdout	under	0	200	300	50	0.65	0.60

In both cases, the optimal trees are rather small judging by the nodesize and maxnodes parameters. This can be justified by Figure 14 which shows how Random Forest overfits: if other parameters are assumed to be fixed, the increasing number of nodes does not lead to a (significant) improvement of the out-of-sample performance while the in-sample gini increases rapidly. Therefore the parameters selected by the grid search on the mean out-of-time gini are those leading to smaller trees and less difference between the in-sample and out-of-time ginis.

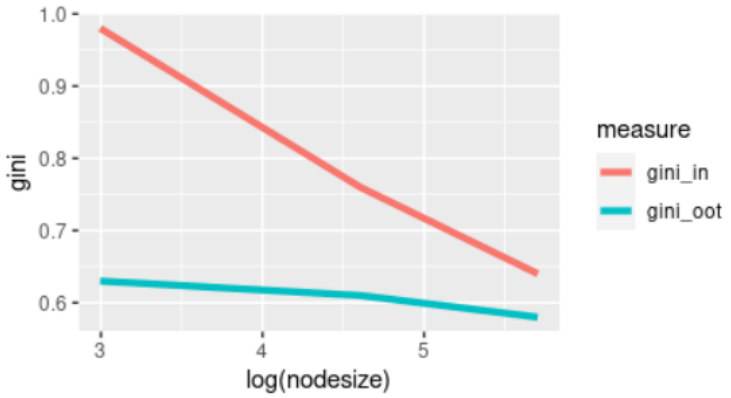

Figure 14: Overfitting and maxnodes - example

7.3 Boosting Trees

7.3.1 Motivation

Similarly to the bagging approach, the boosting technique takes advantage of aggregating many weak learners (models with a classification error which is only slightly better than a coin toss) to one classifier by majority voting. However, in each iteration the training set gets adjusted and also the voting does not follow the rule one model - one vote, but is weighted.

Each of N observation gets a weight ω_i assigned which for the first iteration simply equals $1/N$. The idea is to update the weights in a way that the misclassified data points attain more weight in the next iteration. The majority vote also considers the individual misclassification error of each tree by applying the weights α_m ([Hastie et al., 2017]):

$$G^{(M)}(x) = \text{sgn} \sum_{m=1}^{M} \alpha_m G_m(x)\,, \tag{71}$$

where $G^{(M)}$ represents a boosting classifier consisting of M G_m trees. The algorithm, how α_m and ω_i are determined and G_m are grown, is described in the next section.

7.3.2 Exponential Loss - AdaBoost

One can fit the model by minimizing a loss function L summed over the training observations:

$$\arg \min_{\alpha_m, G_m} \sum_{i=1}^{N} L(y_i, \sum_{m=1}^{M} \alpha_m G_m(x))\,. \tag{72}$$

This is often a computationally intensive problem but can be approximated by the forward stagewise approach which boosting is based on. Forward stagewise means that at each iteration m, only the new basis function / tree G_m and the corresponding coefficient α_m are estimated whereas the previous parameters are not adjusted ([Hastie et al., 2017]).

Following this steps with the exponential loss function for a predictor $f(x)$

$$L(y, f(x)) = \exp(-yf(x)) \tag{73}$$

leads to the AdaBoost algorithm which searches for the solution of

$$\begin{aligned}(\beta_m, G_m) &= \arg \min_{\beta, G} \sum_{i=1}^{N} \exp(-y_i(G^{(m-1)}(x_i) + \beta G(x_i))) \\ &= \arg \min_{\beta, G} \sum_{i=1}^{N} \omega_i^{(m)} \exp(-y_i \beta G(x_i))\end{aligned} \tag{74}$$

with

$$\omega_i^{(m)} = \exp(-y_i G^{(m-1)}(x_i))\,. \tag{75}$$

This coefficient does not depend on the step m but only on the previous iterations because of $G^{(m-1)}$ hence is updated in each step.
This problem can be solved by firstly fixing a $\beta > 0$ and re-writing the minimized term in Equation (74) for $y \in \{-1, 1\}$ as

$$e^{-\beta} \cdot \sum_{y_i = G(x_i)} \omega_i^{(m)} + e^{\beta} \cdot \sum_{y_i \neq G(x_i)} \omega_i^{(m)} \tag{76}$$

and further

$$(e^{\beta} - e^{-\beta}) \sum_{i=1}^{N} \omega_i^{(m)} I(y_i \neq G(x_i)) + e^{-\beta} \cdot \sum_{i=1}^{N} \omega_i^{(m)} . \tag{77}$$

For a fixed $\beta > 0$ the solution of Equation (74) is then

$$G_m = \arg\min_{G} \sum_{i=1}^{N} \omega_i^{(m)} I(y_i \neq G(x_i)) . \tag{78}$$

G_m minimizes the weighted classification errors. Plugging G_m and Equation (77) into Equation (74) results in the following solution ([Hastie et al., 2017]):

$$\beta_m = 0.5 \log \frac{1 - \text{err}_m}{\text{err}_m} . \tag{79}$$

err_m stands for the minimized weighted error rate:

$$\text{err}_m = \frac{\sum_{i=1}^{N} \omega_i^{(m)} I(y_i \neq G_m(x))}{\sum_{i=1}^{N} \omega_i^{(m)}} . \tag{80}$$

The approximation for the next step is therefore

$$G^{(m)} = G^{(m-1)} + \beta_m G_m \tag{81}$$

hence the weights for the $m + 1$st iteration are computed as

$$\begin{aligned} \omega_i^{(m+1)} &= \omega_i^{(m)} \cdot e^{-\beta_m y_i G_m(x_i)} \\ &= \omega_i^{(m)} \cdot e^{2\beta_m I(y_i \neq G_m(x))} \cdot e^{-\beta_m} . \end{aligned} \tag{82}$$

$e^{-\beta_m}$ multiplies every single weight thus can be dropped whereas $\alpha_m = 2\beta_m$.

Thus, AdaBoost can be summarized as follows: for each $m = 1, \ldots, M$

- Fit a tree $G_m(x)$ which minimizes the loss function;
- Calculate the error rate err_m and the coefficient $\alpha_m = \log \frac{1-\text{err}_m}{\text{err}_m}$;
- Update the weights with $\omega_i^{(m+1)} = \omega_i^{(m)} \cdot e^{\alpha_m I(y_i \neq G_m(x))}$.

The final classifier is then $G(x) = \text{sgn}(\sum_{m=1}^{M} \alpha_m G_m(x))$. The sign comes from the fact that if one wants to minimize the expected exponential loss, the following holds as shown in [Friedman et al., 2000]:

$$E(e^{-yf(x)} \mid x) = P(y = 1 \mid x) \cdot e^{-f(x)} + P(y = -1 \mid x) \cdot e^{f(x)} \,. \tag{83}$$

After deriving and setting to zero

$$\frac{\partial E(e^{-yf(x)})}{\partial f(x)} = -P(y = 1 \mid x) \cdot e^{-f(x)} + P(y = -1 \mid x) \cdot e^{f(x)} = 0 \,, \tag{84}$$

one sees that AdaBoost approximates one half log-odds

$$f(x) = 0.5 \cdot \ln \left(\frac{PD_x}{1 - PD_x} \right) . \tag{85}$$

7.3.3 Application

The R package gbm by [Greenwell et al., 2020] has the AdaBoost algorithm implemented with the following hyperparameters:

- **distribution** is set to "AdaBoost", other loss functions are available too;
- **n.trees** which refers to M as the total number of trees;
- **n.minobsinnode** controls the minimal number of observations in the terminal nodes similarly to the random forest;
- **shrinkage** is applied to every tree coefficient α_m and slows down the learning rate which might require more trees;
- **bag.fraction** defines a subset of the training set on which each tree is fit. This introduces randomness and also reduces the computation time (is set to default which is 0.5);
- **weights** impose a heavier cost if misclassification occurs in the default class (overweighting).

Neither overweighting nor the removal of outliers improves the outcome of undersampling. In all setups the same optimal hyperparameters are selected (200 trees with at least 200 observations per node and a shrinkage parameter of 0.001).

Table 14: AdaBoost - results

method	imb	mislab	gini_in	gini_oot
cv	under	0	0.58	0.55
cv	under	1	0.67	0.55
cv	weighted	0	0.57	0.55
cv	weighted	1	0.67	0.54
holdout	under	0	0.57	0.51

8 Neural Networks

8.1 General Structure

A classical neural net which can be used for the purposes of this thesis consists of

- an input layer where every neuron refers to one of the variables
- an output layer with only one neuron which represents the probability of default
- one or multiple hidden layers with every neuron being connected to each and every neuron of the previous and the next layer.

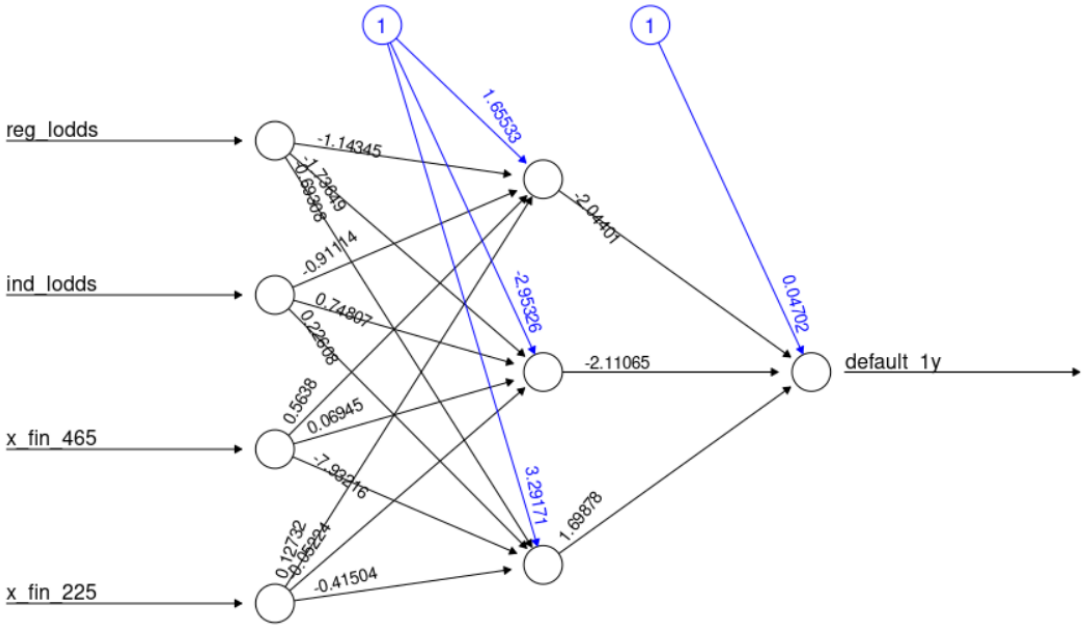

Figure 15: Single-layer Neural Network

In order to calculate the j^{th} neuron of the of the l^{th} layer a_j^l, the k^{th} neuron of the $(l-1)$st layer a_k^{l-1} is firstly multiplied with the corresponding weight w_{jk}^l. Those are summed up in order to get an intermediate quantity

$$z_j^l = \sum_k w_{jk}^l a_k^{l-1} + b_j^l \,, \tag{86}$$

which is simply a linear combination of the previous layer plus the intercept term b_j^l (can be identified by the blue color in the plot). Finally, an activation function σ is applied

$$a_j^l = \sigma(z_j^l) \tag{87}$$

because otherwise one would end up with an ordinary logistic regression model. The logistic/sigmoid function $\sigma(z) = \frac{1}{1+e^{-z}}$ will be used as the activation function for each layer including the output layer in order to end up with a PD value at the end. The cross-entropy will be used as the error (or cost) function

$$E = -\sum_{n=1}^{N} (y_n \log(\hat{y}_n) + (1 - y_n) \log(1 - \hat{y}_n)) \tag{88}$$

with the weights being adjusted so that the cross-entropy value becomes as low as possible. This is done by the backpropogation algorithm and its modifications as implemented in the R package neuralnet by [Fritsch et al., 2019].

8.2 Backpropagation

The algorithm is presented graphically in the Figure 16 (from [Günther and Fritsch, 2010]). In the one-dimensional case, the derivative of the error function E is computed and the weight w_t is shifted towards the minimum of the error function, resulting in the updated weight w_{t+1}.

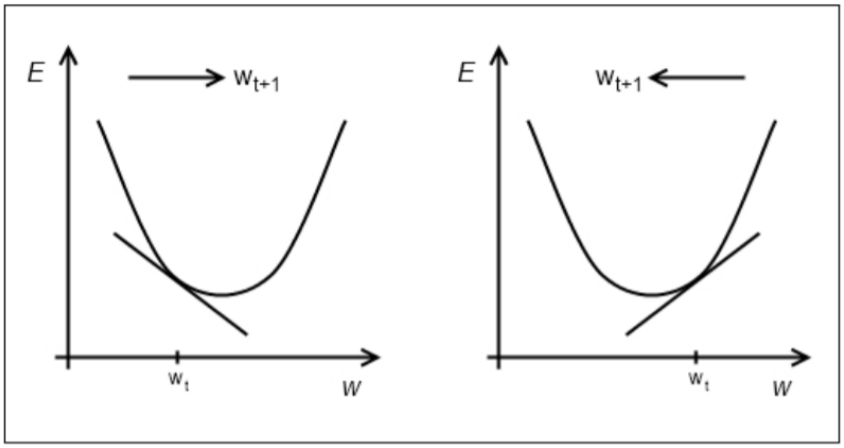

Figure 16: Backpropagation algorithm

The mathematical motivation is presented below, the explanation follows [Nielsen, 2015]. The main idea is to compute iteratively the partial derivatives $\frac{\partial E}{\partial w_j^l}$ of the error function with respect to each and every weight by using the chain rule. A further intermediate quantity is introduced to make the computation easier

$$\delta_j^l = \frac{\partial E}{\partial z_j^l}\,, \tag{89}$$

which refers to the error in the j^{th} weighted input of the l_{th} layer. By making use of the chain rule this can be written for the output neuron (layer L, index skipped) as

$$\begin{aligned} \delta^L &= \frac{\partial E}{\partial z^L} \\ &= \frac{\partial E}{\partial a^L} \frac{\partial a_L}{\partial z^L} \\ &= \frac{\partial E}{\partial a^L} \sigma'(z^L)\,. \end{aligned} \tag{90}$$

For any layer l this can be computed by using the derivatives from the next layer. One starts with the chain rule:

$$\begin{aligned}\delta_j^l &= \frac{\partial E}{\partial z_j^l} \\ &= \sum_k \frac{\partial E}{\partial z_k^{l+1}} \frac{\partial z_k^{l+1}}{\partial z_j^l} \\ &= \sum_k \delta_k^{l+1} \frac{\partial z_k^{l+1}}{\partial z_j^l} \,.\end{aligned} \tag{91}$$

The weighted input can be written as

$$z_k^{l+1} = \sum_j w_{kj}^{l+1} a_j^l + b_k^{l+1} = \sum_j w_{kj}^{l+1} \sigma(z_j^l) + b_k^{l+1} \tag{92}$$

and the derivative is then simply

$$\frac{\partial z_k^{l+1}}{\partial z_j^l} = w_{kj}^{l+1} \sigma'(z_j^l) \,. \tag{93}$$

By combining Equation (91) and Equation (93), the final equation has the form

$$\delta_j^l = \sigma'(z_j^l) \sum_k \delta_k^{l+1} w_{kj}^{l+1} \,. \tag{94}$$

The further steps are not proved by [Nielsen, 2015] but follow the same idea. One can start by computing the derivative with respect to the bias which in fact equals δ_k^l that has already been calculated because

$$\begin{aligned}\delta_j^l &= \frac{\partial E}{\partial z_j^l} \\ &= \sum_k \frac{\partial E}{\partial b_k^l} \frac{\partial b_k^l}{\partial z_j^l} \\ &= \frac{\partial E}{\partial b_j^l} \underbrace{\frac{\partial b_j^l}{\partial z_j^l}}_{=1}\end{aligned} \tag{95}$$

due to the second derivative term being zero for $j \neq k$ and 1 for $j = k$ since the bias can be written as a linear function of z

$$b_k^l = z_k^l - \sum_j w_{kj}^l a_j^{l-1} \,. \tag{96}$$

Last but not least, the partial derivative with respect to the weight w_{jk}^l equals

$$\begin{aligned}\frac{\partial E}{\partial w_{jk}^l} &= \sum_m \frac{\partial E}{\partial z_m^l}\frac{\partial z_m^l}{\partial w_{jk}^l} \\ &= \frac{\partial E}{\partial z_j^l}\frac{\partial z_j^l}{\partial w_{jk}^l} \\ &= \delta_j^l a_k^{l-1}\,.\end{aligned} \tag{97}$$

The regular backpropagation algorithm is applied in order to minimize the error function. This is done by applying a simple gradient descent procedure. From iteration t to iteration $t+1$ the weights are updated as follows

$$w_{jk}^l(t+1) = w_{jk}^l(t) - \epsilon\frac{\partial E}{\partial w_{jk}^l}(t)\,. \tag{98}$$

The learning rate ϵ is constant and its selection is essential for the performance. A too large value might lead to oscillation which would prevent the error to fall below a certain level. On the other hand, a too low value could lead to a slow convergence requiring a high number of iterations in order to reach an optimal solution.

[Riedmiller and Braun, 1993] propose the so-called resilient backpropagation which is also implemented in [Fritsch et al., 2019]. The weight adjustment does not depend on the absolute value of the derivative and the learning rate changes over the time based on the change of the derivative's sign. If the sign remains unchanged

$$\frac{\partial E}{\partial w_{jk}^l}(t)\cdot\frac{\partial E}{\partial w_{jk}^l}(t-1) > 0\,, \tag{99}$$

the step size from the previous epoch/iteration is increased (multiplied with some $\epsilon^+ > 1$) in order to accelerate convergence. Otherwise, the previous iteration is undone and a smaller step size is applied in the succeeding iteration (multiplied by $0 < \epsilon^- < 1$). This part is referred to as weight backtracking and is implemented in [Fritsch et al., 2019] as RPROP+. The parameters proposed by the authors are $\epsilon^+ = 1.2$ and $\epsilon^- = 0.5$ and remain unchanged in the application.

8.3 Application

Some theoretical studies suggest that a variable transformation can improve the results. [Siddiqui and Ali, 2016] show that artificial neural networks performed poorly on skewed data. [Khamis et al., 2005] claim that a high percentage of outliers has a significant impact on the performance, as it could be expected for a parametric method. Therefore, the features are again uniformly-transformed.

Since there is no way to determine the optimal number and the size of the hidden layers a priori, one to three hidden layers of different sizes are combined and tuned as hyperparameters. Another hyperparameter is the threshold which defines the stopping criteria in terms of the change in the partial derivatives of the error function (a lower value leads to a higher number of iterations and significant overfitting, see Figure 17).

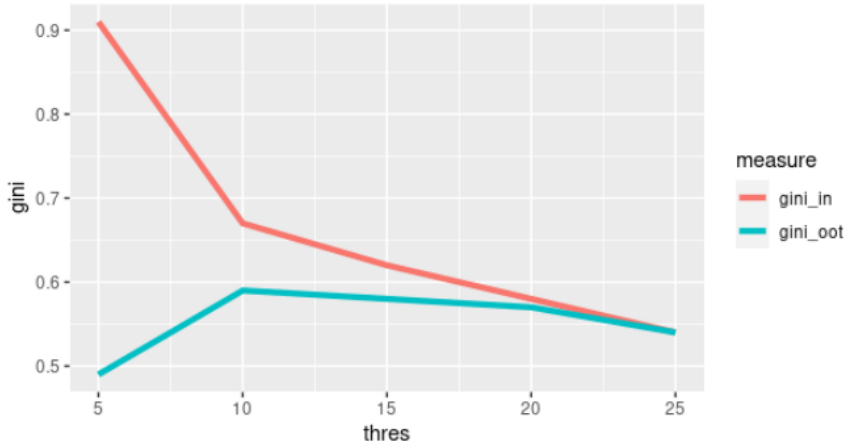

Figure 17: Neural network - effect of the threshold

The results indicated that only one hidden layer with few nodes is enough to achieve the best gini. The transformation also improves the performance significantly while the removal of mislabeled instances has a low but positive impact.

Table 15: Neural network - results for one hidden layer

sample	method	mislab	threshold	nodes	gini_in	gini_oot
original	cv	0	10	30	0.46	0.41
original	cv	1	10	30	0.62	0.40
transformed	cv	0	20	20	0.58	0.57
transformed	cv	1	20	20	0.67	0.58
transformed	holdout	1	20	20	0.70	0.63

The class weighting is not implemented in the neuralnet package hence not applied. Similarly to the other algorithms, one would adjust the loss function Equation (88) and choose the weights inversely proportional to the class distribution in the training sample.

9 Summary and Conclusions

In this section the results from previous chapters are summarized and the best model is selected. Additionally, few possible steps are proposed that could improve the performance further. Table 16 provides an overview of the ginis, calculated on the holdout sample (last available year).

Table 16: Overview of the results on the holdout sample

model	kernel	gini
Neural Network		62.74%
Random Forest		59.99%
Naive Bayes		56.43%
Support Vector Machine	linear	55.30%
Support Vector Machine	polynomial	55.22%
Support Vector Machine	radial	54.82%
Ridge Regression		54.48%
Lasso Regression		53.64%
Support Vector Machine	sigmoid	53.04%
Logistic Regression		51.98%
AdaBoost		50.85%
k-Nearest Neighbours		46.27%

The special treatment of the (potentially) mislabeled observation does not improve the results significantly. On the one hand, a more conservative threshold than the one proposed in Subsection 1.4 might lead to a different outcome. On the other hand, it might be an indication of a good quality of the target variable. Furthermore, most of the algorithms, where both approaches of dealing with imbalanced data are applied, do not profit from the overweighting significantly while the increased number of observations triggers a longer search for the optimal hyperparameters. For most of the methods, the best results are achieved after applying the uniform transformation which leads to some loss of information but the outcome verifies that the elimination of outliers and skewness, achieved by doing so, leads to a better performance.

K-nearest neighbours are strongly underperforming compared to the other methods. That fact is supported by the overview of related works in Table 17. On the contrary, Naive Bayes achieves solid results putting it in the upper half of the ranking while it is highly attractive because of the fast fitting process. Both of those methods profit massively from the variable transformation and might be further enhanced in terms of the out-of-time gini by applying some variable selection procedure e.g. backward elimination.

It is confirmed that the logistic regression and its regularized versions may offer models that are comparable to some of the more sophisticated or less widespread approaches such as support vector machine. In case of SVM, the four different kernels provide similar results proving that the best model can be obtained with the linear kernel.

The tree-based methods which are applied on original untransformed data are placed in

the opposite tails of the evaluation table. While random forest almost reaches impressive 60%, AdaBoost has the second worst gini. This is rather unexpected when looking at the other sources in Table 17 and might indicate a need for a further hyperparameter search.

Finally, a neural network with one hidden layer consisting of twenty nodes outperformed the multi-layer networks as well as other methods. This method is easier to implement in production environment than e.g. a random forest model but the structure makes the interpretation of the impact of each of the variables on the score less intuitive than in logistic regression. In addition, a careful training with an evaluation on the test set is required in order to avoid overfitting. The early stopping procedure does the job well, nevertheless, other techniques like regularization might be considered for this purpose.

References

Accenture (2020). The future of default prediction: A comparison of machine learning model performance. https://www.accenture.com/nl-en/blogs/insights/the-future-of-default-prediction-a-comparison-of-machine-learning-model-performance.

Ahmed, M., Shahjaman, M., Rana, M., and Mollah, M. N. (2017). Robustification of naive bayes classifier and its application for microarray gene expression data analysis. *BioMed Research International*, 2017:1–17.

Alonso, A. and Carbo, J. (2021). Understanding the performance of machine learning models to predict credit default: A novel approach for supervisory evaluation. *Banco de Espana Working Paper No. 2105.*

BCBS (2006). International convergence of capital measurement and capital standards. https://www.bis.org/publ/bcbs128.pdf.

Brodley, C. and Friedl, M. (1999). Identifying mislabeled training data. *Journal of Artificial Intelligence Research - JAIR*, 11:131–167.

Choubey, A. (2018). Predicting credit default risk via statistical model and machine learning algorithms. Master's thesis, University of North Caroline, Charlotte.

EBA (2021). EBA discussion paper on machine learning for IRB models. https://www.eba.europa.eu/regulation-and-policy/model-validation/discussion-paper-machine-learning-irb-models.

Friedman, J., Hastie, T., and Tibshirani, R. (2000). Additive logistic regression: A statistical view of boosting. *The Annals of Statistics*, 28:337–407.

Friedman, J., Hastie, T., and Tibshirani, R. (2010). Regularized paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22.

Fritsch, S., Guenther, F., and Wright, M. N. (2019). neuralnet: Training of neural networks. CRAN.R-project.org/package=neuralnet. R package version 1.44.2.

Granström, D. and Abrahamsson, J. (2019). Loan default prediction using supervised machine learning algorithms. Master's thesis, KTH Royal Institute of Technology.

Greenwell, B., Boehmke, B., and Cunningham, J. (2020). gbm: Generalized boosted regression models. CRAN.R-project.org/package=gbm. R package version 2.1.8.

Günther, F. and Fritsch, S. (2010). neuralnet: Training of neural networks. *The R Journal*, 2(1):30–38.

Hastie, T., Tibshirani, R., and Friedman, J. (2017). *The Elements of Statistical Learning*. Springer, 2nd edition.

Hsu, C.-W., Chang, C.-C., and Lin, C.-J. (2003). A practical guide to support vector classification. https://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf.

James, G., Witten, D., Hastie, T., and Tibshirani, R. (2017). *An Introduction to Statistical Learning*. Springer.

Khamis, A., Ismail, Z., Khalid, H., and Mohammed, A. (2005). The effects of outliers data on neural network performance. *Journal of Applied Sciences*, 5(8):1394–1398.

Kuhn, M. and Johnson, K. (2013). *Applied Predictive Modeling*. Springer.

Kutner, M., Nachtsheim, C., Neter, J., and Li, W. (2005). *Applied Linear Statistical Models*. McGraw-Hill Irwin.

Liaw, A. and Wiener, M. (2002). Classification and regression by randomforest. *R News*, 2(3):18–22.

Little, R. and Rubin, D. (2019). *Statistical Analysis with Missing Data*. John Wiley & Sons, 3rd edition.

Majka, M. (2019). naivebayes: High performance implementation of the naive bayes algorithm in R. `CRAN.R-project.org/package=naivebayes`. R package version 0.9.7.

Matre, A. (2019). Machine learning in default prediction: The incremental power of machine learning techniques in mortage default prediction. Master's thesis, Norwegian School of Economics.

Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., and Leisch, F. (2019). e1071: Misc functions of the department of statistics, probability theory group (formerly: E1071), TU Wien. `CRAN.R-project.org/package=e1071`. R package version 1.7-1.

Nielsen, M. (2015). *Neural Networks and Deep Learning*. Determination Press.

Palomar, D. (2020). Lagrange duality. `https://palomar.home.ece.ust.hk/ELEC5470_lectures/slides_Lagrange_duality.pdf`.

Riedmiller, M. and Braun, H. (1993). A direct adaptive method for faster backpropagation learning: The RPROP algorithm. *IEEE Int. Conf. Neural Networks*, 1:586–591.

Rousseeuw, P. and Leroy, A. (1987). *Robust Regression & Outlier Detection*. John Wiley & Sons.

Shah, A., Bartlett, J., Carpenter, J., Nicholas, O., and Hemingway, H. (2014). Comparison of random forest and parametric imputation models for imputing missing data using mice: A caliber study. *American Journal of Epidemiology*, 179(6):764–774.

Siddiqui, F. and Ali, Q. (2016). Performance of non-parametric classifiers on highly skewed data. *Global Journal of Pure and Applied Mathematics*, 12:1547–1565.

Stoltzfus, J. (2011). Logistic regression: A brief primer. *Academic Emergency Medicine*, 18:1099–1104.

Van Buuren, S. (2018). *Flexible Imputation of Missing Data*. Chapman and Hall, 2nd edition.

Venables, W. N. and Ripley, B. D. (2002). *Modern Applied Statistics with S*. Springer, 4th edition.

Wilson, S. (2021). miceranger: Multiple imputation by chained equations with random forests. `CRAN.R-project.org/package=miceRanger`. R package version 1.5.0.

List of Figures

1 Training / Test structure . . . 12
2 Optimal lambda in ridge - example . . . 22
3 L1 vs L2 norm . . . 23
4 Optimal lambda in lasso - example . . . 24
5 Hyperplane in 2D . . . 26
6 Separable vs. non-separable case . . . 26
7 Radial kernel and the gamma parameter - example . . . 32
8 Polynomial kernel and the gamma parameter - example . . . 33
9 Gini for different k and transformations . . . 35
10 Gini for different k and transformations . . . 36
11 Posterior densities of a variable - example . . . 37
12 Decision tree - example . . . 40
13 Gini impurity - example . . . 42
14 Overfitting and maxnodes - example . . . 43
15 Single-layer Neural Network . . . 47
16 Backpropagation algorithm . . . 48
17 Neural network - effect of the threshold . . . 51
18 Confusion matrix . . . 59
19 ROC-curve - example . . . 60
20 High spearman correlation - example . . . 61

List of Tables

1 Comparison of similar works - summary . . . 8
2 Sample preview . . . 11
3 Logistic regression - results with undersampling . . . 18
4 Logistic regression - results with overweighting . . . 19
5 Ridge regression - results . . . 22
6 Lasso regression - results . . . 24
7 SVM - results on the untransformed sample . . . 31
8 SVM - CV results with uniform transformation . . . 32
9 SVM - holdout results with uniform transformation . . . 32
10 Robust KNN - results (for k=35) . . . 36
11 Naive Bayes - results . . . 39
12 Naive Bayes - results with MCD . . . 39
13 Random Forest - results . . . 43
14 AdaBoost - results . . . 46
15 Neural network - results for one hidden layer . . . 51
16 Overview of the results on the holdout sample . . . 52
17 Comparison of similar works - results . . . 62

A Gini-coefficient / Somers' D

The given setup is a binary classification problem which can be evaluated by the so-called confusion matrix (Figure 18). The output of a model will be some real number and based on the selected threshold / cut-off value, one ultimately decides whether some observation gets classified as 0 or 1. In real-life problems a perfect separation is almost never possible and one can work with measures derived from the confusion matrix.

		Predicted	
		1	0
Actual	1	TP	FN
	0	FP	TN

Figure 18: Confusion matrix

True and **F**alse refer to correctly and incorrectly predicted classes while **P**ositive and **N**egative are the classes of the target variable e.g. TN is the number of the observations which belong to the negative class and were also classified as negative.

The **sensitivity** (also known as true positive rate or recall) is the proportion of correctly classified positive observations

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{100}$$

Similarly, the **specificity** (or true negative rate) is defined as the proportion of correctly classified negative observations

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}} \tag{101}$$

If one imagines moving the threshold along the real axis, it gets clear that the one of the two measures will always increase and the second one will simultaneously decrease. To cover all of the possible thresholds at once, the **ROC-curve** is used which displays the sensitivity and the **F**alse **P**ositive **R**ate (or 1 - Specificity) for all possible cut-off values (see Figure 19). The base "random" model refers to the diagonal in this plot which has an area under the curve (AUC) of 0.5.

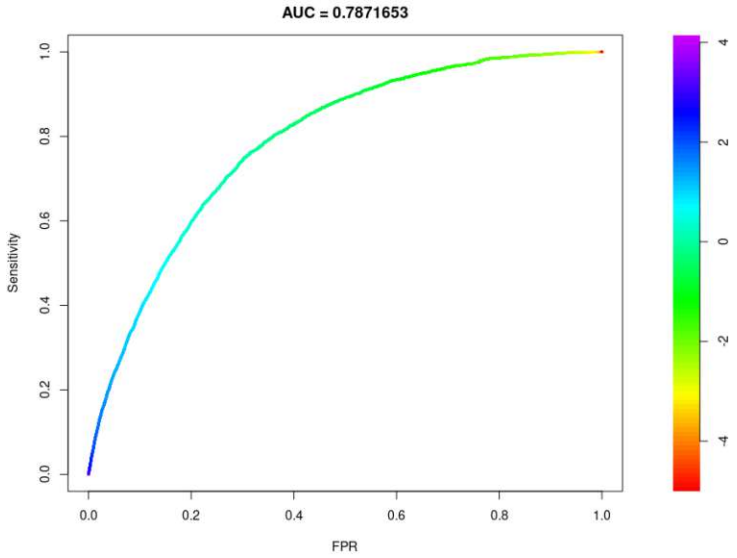

Figure 19: ROC-curve - example

The area under the ROC-curve can also serve for the univariate analysis and evaluate the performance of each and every financial variable as if it was a standalone model score. Since an AUC value under 50% would indicate that the sign of the financial should be changed in order to achieve consistency in terms of economic interpretation (e.g. higher values imply a better credit worthiness and a lower probability of default), the AUC is transformed as

$$\text{Gini} = \text{AUC} \cdot 2 - 1. \tag{102}$$

The resulting figure is the so-called **Gini-coefficient** or **Somers' D** and is a measure of ordinal association between an independent variable and a binary target variable. The sign of the feature is selected in a way that the gini is always **negative** for consistency purposes.

B Spearman's Correlation

The Spearman's correlation of two continious variables x and y is formulated as the Pearson's correlation coefficient of the rank transformed variables rg_x and rg_y

$$r_s = \rho(rg_x, rg_y) = \frac{Cov(rg_x, rg_y)}{\sigma_{rg_x} \cdot \sigma_{rg_y}} \tag{103}$$

where σ_{rg_x} and σ_{rg_y} denote the standard deviations of the rank variables.

This is more general than the ordinary linear Pearson's correlation because only the monotonicity of the relationship is considered and the outliers do not have a high influence on the outcome. This can be seen in the Figure 20. The robust Spearman's correlation is 89% whereas the Pearson correlation is only 56% because of the outlier.

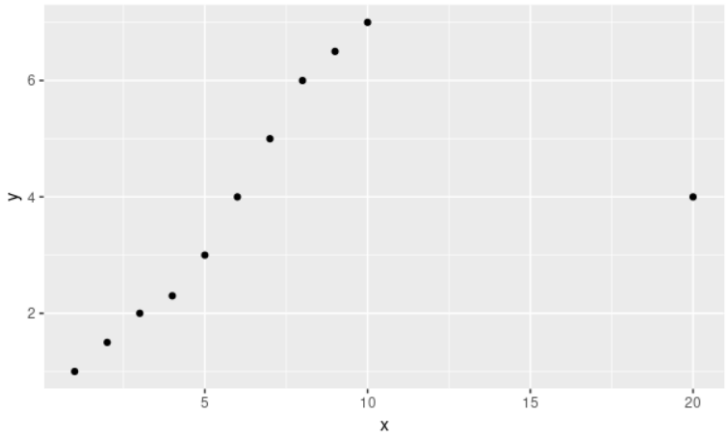

Figure 20: High spearman correlation - example

C Comparison of Similar Works

Table 17: Comparison of similar works - results

Source	Logit	NN	RF	XGBoost	AdaBoost	SVM	NB	KNN
[Alonso and Carbo, 2021]	0.56	0.62	0.66	0.68	NA	NA	NA	NA
[Granström and Abrahamsson, 2019]	0.76	0.76	0.76	0.78	0.76	0.48	NA	NA
[Choubey, 2018]	0.42	0.34	0.52	NA	NA	NA	NA	NA
[Matre, 2019]	0.77	0.80	0.80	0.76	NA	0.80	NA	0.59
[Accenture, 2020] (1)	0.54	0.56	0.54	0.56	0.54	0.52	0.48	0.52
[Accenture, 2020] (2)	0.54	0.62	0.60	0.62	0.60	NA	0.32	0.50