Master Thesis

Improving High-Risk Consumer Credit Scoring with Financial Transaction Data

Jon Stålhammar
jost95@gmail.com

Carl Hvarfner
carl.hvarfner@live.se

Supervisor: Erik Lindström
Examiner: Magnus Wiktorsson

Department of Mathematical Statistics
Lund University, Faculty of Engineering

May 10, 2020

Abstract

Credit scoring, the process of evaluating loan applicants with the help of numerical methods, is widely used in the financial sector. When used correctly, it can aid the decision process and lower default rates on loans. New EU regulation has allowed third-party actors to access personal banking information, following the applicant's consent. The increase in available information that financial transaction data provides can be used to improve credit scoring models, and can thus be of great value to lenders. Using a data set from a Swedish consumer institute containing raw transaction data, the authors have tried to extract useful features which can improve credit scoring precision. The authors describe transaction categorization, feature construction and selection as well as model evaluation. A large number of features were created, which lead to a substantial and involved feature reduction process. Four types of models were evaluated: a Logistic Regression, a Random Forest, an XGBoost and a Neural Network. As reference, the company's old model, a Logistic Regression using socioeconomic factors, was used. After careful evaluation using Bayesian Optimization for hyperparameter tuning, a combination of Logistic Regression models were considered to be the best and most consistent. Of the transaction-derived features evaluated, bailiff expenses, salary and ATM withdrawals were considered to be the most influential across all models. Compared to an optimized Logistic Regression base model, a four percentage point average precision improvement could be observed in the final model.

Keywords: Machine Learning, Scorecard Modelling, Feature Engineering, Feature Selection, Average Precision Score, SHAP, SMOTE, Logistic Regression, Random Forest, XGBoost, Artificial Neural Network

Popular Science Summary

Even though machine learning models have been around for decades, they have risen quickly in popularity thanks to large amounts of data and faster computer processors. Credit institutes, and particularly those who give out consumer loans, are part of those wanting to utilize machine learning in their day-to-day operations to eliminate subjective opinions and boost their profits. Human based decision always inject subjective opinions, based on their personal experiences, and which often are not true. By utilizing automated computer processes, prediction if a customer will not be able to pay their loan, can be done faster and with higher precision. Meanwhile, thanks to new EU regulations, third-parties are allowed at customers' consent to analyze their financial transactions to produce better services. This thesis connects these two dots, machine learning and financial transactions, to improve what is called credit scoring. Multiple models have been evaluated and it can be concluded that the more complex models do not triumph a simple statistical model (Logistic Regression). However, this might change as financial transactions are further evaluated as such data become more widely available.

Acknowledgements

We would first like to thank our thesis advisor, Prof. Erik Lindström, for his involvement not only in ensuring this work came together in a desirable manner, but also for constantly supplying us with new ideas for approaching the problem at hand. Without his ideas, our final models would not have performed nearly as well as they did.

We would also like to thank the people at the Swedish Consumer Credit Institute for their guidance and constructive criticism of our work throughout the process. Their input persistently gave us new areas of improvement, as well as credit when credit was due.

Contents

List of Figures

List of Tables

Chapter 1.

Background

1.1. Introduction

With the emergence of widespread big data usage during the last ten years, the means by which businesses make decisions have fundamentally changed. By incorporating data analytics and machine learning to tame large data sets and find patterns in customer behavior, a number of large and influential tech companies, such as Facebook and Amazon, have risen to prominence in the last decade (Erevelles, Fukawa, and Swayne, 2016). With similar data-driven approaches, companies such as Uber and Airbnb have gone from start-ups to world-leading in their respective industries. The use and value of data has worked as an enabler across industries, and is finding its way into new industries as decision-makers realize its potential value (Van Rijmenam, 2019).

As the use of big data has marked its footprint across industries, the European banking industry has now been set in motion. With the emergence of the second EU Payment Services Directive (PSD2), data is no longer regarded as property of the bank carrying out the transaction, but of the customer (Cortet, Rijks, and Nijland, 2016). In essence, the PSD2 directive puts every consumer in charge of their bank statement data. Consequently, FinTechs and incumbent banks are not at a disadvantage with regard to data supply when compared to larger banks, that tend to hold said costumer's data. FinTechs and incumbent banks are thus guaranteed to receive a customer's transaction data when providing their service, assuming the approval of the customer (Możdżyński, 2017). The dynamic brought along by the PSD2 directive is often referred to as Open Banking (Brodsky and Oakes, 2017).

"From a commercial standpoint, data can serve as a catalyst for new products and business models. The EU has been proactive on this front, setting the rules of engagement through the updated version of the Payment Services Directive (PSD2)." (Brodsky and Oakes, 2017)

The credit market is an important enabling factor in the world economy, by helping both individuals and businesses access capital when needed. In an optimal setting, it can act as a mean of augmenting the market involvement of otherwise illiquid citizens. However, it can also act as a detriment for the individual, causing over-indebtedness and social exclusion (Schraten, 2020). This makes the decision process of providing appropriately sized loans to the right customers a complicated problem, one that credit institutes must continuously work to improve upon. As an incumbent bank focused on consumer credit of relatively small amounts, a Swedish Consumer Credit Institute intends to capitalize on the changes brought on by PSD2, making use of their applicants' transaction data in its decision process.

1.2. Credit Scoring

The term credit scoring is used to describe various statistical methods used for judging an applicant's creditworthiness (Hand and Henley, 1997). The process of credit scoring is reliant on relevant financial data regarding the applicant, and of a larger sample, to put the financial data of the applicant into context. While there are external companies that perform this, legislative measures have provided incentives for financial institutions to internalize the credit scoring process (Siddiqi, 2017).

1.2.1. History of Credit Scoring

As a traditional mean of credit scoring, a majority or all of the scoring processes have been outsourced to a third-party specialized in scoring individuals on socioeconomic factors. FICO and UC are the largest companies in the field in the US and Sweden respectively, and have both been present in the industry for over 40 years. These companies create credit scores on consumers based on numerous economic and social factors, with the final output being a triple-digit integer value. A higher value represents a high creditworthiness for the consumer. Historically, the performance of the consumer in these credit scoring methods have carried significant weight in determining availability for loans and credit limits (Hurley and Adebayo, 2016). However, regulatory directives, such as the Basel II and IFRS 9, incentivized banks of all sizes to develop their own means of credit scoring starting in the mid 2000's (Siddiqi, 2017) (Louzada, Ara, and Fernandes, 2016). Thus, the reliance on third-party scoring methods have subsequently decreased, as scoring methods by individual banks and credit institutes have become increasingly sophisticated (Hurley and Adebayo, 2016).

1.2.2. Credit Scoring using Economic and Demographic Factors

A common method of assessing creditworthiness is through the use of demographic factors in addition to economic ones when constructing a credit scoring model. These can include, but are not limited to, age, gender, household income, public records, residence status and previous loan claims (Azam, Danish, and Akbar, 2012) (Siddiqi, 2017). These attributes are obtained either through public sources, the bank's own records, or provided by the applicant as part of the application process. Thus, the applicant is entrusted to provide accurate information that can be of large importance in the credit evaluation process. Since the applicant is not always able, or lacks the incentive, to provide accurate information, this process can potentially induce errors in the credit scoring evaluation process.

1.2.3. Emergence of New Data Sources

With the introduction of big data to the financial sector, banks and credit institutes are provided with new means of assessing creditworthiness, through more detailed information on each applicant than what has previously been feasible. This could include, but is not limited to, transaction data, social media activity data and other digital footprints. The insights provided by this data has proven to increase the predictory power of credit scoring models, particularly when evaluating applicants with limited financial history. Due to the large volume of data obtained for each customer, machine learning algorithms have become the standard methods for determining creditworthiness in this domain (Onay and Öztürk, 2018).

1.2.4. Explainability Requirement

With any declined credit application, the applicant is entitled to receive the specific reason for the declination. With the use of new or alternative data sources, this requirement must receive special consideration, due to the potentially non-linear and complex relationships generated by certain scoring methods (Johnson, 2019).

1.3. Digital Banking

As the digital capabilities of both companies and consumers have increased in recent years, bank services have seen a shift from local branches to being performed remotely. With this shift, the breadth of banking services has increased, spawning new entrants in the FinTech and niche banking spaces (Zachariadis and Ozcan, 2017). Examples of such players in the Swedish market are Lendo (active in loan consolidation) Marginalen Bank (provider of consumer credit), Klarna and iZettle (payment providers) (Bertsch and Rosenvinge, 2019) (Teigland et al., 2018). As digital services increase their prominence in the banking sector, the opportunity for new entrants and technologies to gain market share as well as for incumbents to reconsider their portfolio of services, becomes imminent (Zachariadis and Ozcan, 2017).

1.3.1. Open Banking

As a consequence of EU's Second Payment Services Directive (PSD2) the concept of open banking has emerged as a means to increase digital banking service capabilities. Open banking can be defined as a collaborative model in which banking data is shared through APIs between two or more unaffiliated parties to deliver enhanced capabilities (Brodsky and Oakes, 2017). In simpler terms, open banking enables third party financial service providers to access consumers' accounts and payment services, following the consent of the account owner. Open banking thus demands collaboration from incumbent banks in the process of providing financial information, when consent is given from an account owner. This is done through the use of Application Programming Interfaces, or APIs, that are able to securely transfer the necessary information (Zachariadis and Ozcan, 2017). Intended to boost innovation and competition in the financial services industry, open banking has effectively made accounts and payments controlled largely by the customer, and not the incumbent bank.

1.3.2. Transaction Data for Credit Scoring

With the changes brought on by PSD2, all banks and credit institutes have the right to obtain applicant transaction data from the applicant's current bank, assuming applicant consent. Consequently, transaction data can now reliably be used as part of a credit decision process. This potentially enables more accurate credit decisions than what was feasible with previously used methods, since economically detrimental behavior can be derived from said data. Typical such behavior could include casino and gambling activity, irregular income and general overspending (Tobback and Martens, 2019).

1.4. Swedish Consumer Credit Institute

The client company in this project is a financial entity where the fraction of 'non-performing' customers, meaning those that fail to completely pay back their loan, ranges from 20-25%. The current credit decision process is partially automated, as most incoming applications are treated by a machine learning algorithm based on economic and demographic factors. For applications where the model is indecisive, a human credit analyst conducts the final judgement. For reasons connected to competitive advantages, the company wished to remain anonymous and will therefore be referred to as a Swedish Consumer Credit Institute.

1.5. Problem Description

This report aims to treat two main topics connected to the use of bank statement transaction data for credit scoring. These topics are intertwined, as the performance in one will determine the success in the other.

1.5.1. Transaction Categorization

1. How should raw financial transactions be categorized to best aid a credit scoring model?
2. How can personal bias be eliminated to avoid incorrect assumptions in this process?

1.5.2. Credit Scoring

1. What metrics from transaction categories are informative in a credit scoring context?
2. Can a transaction data based machine learning model help reduce the fraction of non-performing customers?
3. How can a complex transaction data-based model comply with the explainability requirements?

1.6. Contributions

While the field of credit scoring is well explored, both in terms of using socioeconomic factors and the use of transaction data, our approach attempts to cover the entire transaction data pipeline. Thus, the transaction data will be treated and manipulated with the purpose of extracting the most useful factors for credit scoring. In practice, this means that the categorization of transactions is done in a manner that is believed to solve the problem of credit assessment in the most efficient manner. What said categories constitutes of is based on intuition of the authors, company credit experts and the feasibility of separating distinct categories. With this in mind, the authors believe that the presented approach is novel with regard to the integrated use and treatment of transaction data.

1.7. Limitations

Since the models to be trained require labeled data of known customer performance, the data supply is limited not to all incoming applicants, but to all approved loans with access to transaction data. This is due to the fact that only the approved loans can reliably be assigned a 'performing' or 'non-performing' label. While data from all denied applications could have been assigned a 'non-performing' label or in other ways have their label inferred, the use of this data for model training would potentially be in violation of rules set by the EU in the General Data Protection Regulation (GDPR). Thus, the option to use the data from denied applicants is not explored further.

With the outbreak of the Coronavirus in full effect as of the time of this report, the Swedish economy has entered an unprecedented downturn. Lay-offs, shutdowns and company bankruptcies are rising in prominence, threatening the private economies of Swedish citizens. While the described macroeconomic activity likely has an effect on an applicant's ability to pay back credit, the lack of historic precedent limits the potential explanatory power of such activity. As a result, macroeconomic factors are excluded from the set of explanatory variables.

Chapter 2.

Machine Learning Classifiers

2.1. Objective

The problem of credit scoring constitutes of one main task - to label an applicant as either worthy of a loan, or as unworthy of one. This decision process can be based on any number of economic characteristics displayed by the applicant, and are subject to interpretation by a statistical model. The labeling of applicants mathematically translates to representing the customer as either a zero (predicted non-default) or a one (predicted default). The ability of models to represent customers not only in this binary fashion, but as a probability of default between zero and one based on the values of the economic characteristics, makes it possible to adjust the desired risk level of the lender. As a consequence, the lender could opt only to accept applicants with an extremely low risk of default at the expense of revenues. In the process, likely non-default customers will be denied loans if they are not deemed safe enough based on the set probability threshold. The concept of a desired risk level is an integrated part of the modelling process, and will be explored further in later chapters. However, for a model to be desirably efficient, it is preferred that it displays a degree of certainty in its decisions, predicting probabilities either close to one or close to zero for defaults and non-defaults, respectively.

In addition to the mathematically-oriented objectives described above, the produced solution should be interpretable and traceable in its decision making. As the client company has a relatively limited experience with machine learning based approaches, these traits are essential to the trustworthiness of the model. In addition, being able to trace the decisions to specific feature importances enables the model to be used as decision support to a larger extent. This acts in in contrast to a black-box decision process alternative.

2.2. Reasoning Behind Model Selection

With regard to the objectives and problem description in mind, the model selection will favor simpler designs. Thus, Logistic Regression and Random Forest models seemed reasonable since it is easy to extract parameter influence and grasp how the models work. Furthermore, when researching decision tree models, it was found that XGBoost is a natural extension and that it has performed well in various machine learning competitions (Nielsen, 2016). Finally, as it might be that the data contains more complex relationships than above models can handle, it was decided to also include a Neural Network to capture more complex behavior.

2.3. Common Functions, Algorithms and Metrics

2.3.1. Sigmoid Function

The Sigmoid function maps any real number to a rangehttps://www.overleaf.com/project/5e39201f8133 between 0 and 1, producing the typical logistic curve as seen in figure 2.1. The Sigmoid function is used for Logistic Regression and in Neural Networks (Gareth James and Tibshirani, 2017), and is defined as

$$f(x) = \frac{1}{1+e^{-x}}. \tag{2.1}$$

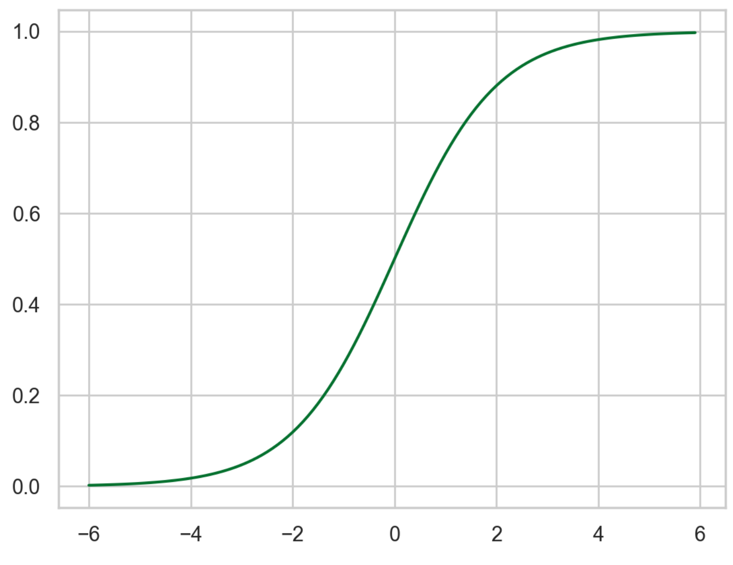

Figure 2.1.: The logistic curve

2.3.2. ReLU Function

The ReLU (Rectified Linear Unit) function is a non-negative function that takes on the value zero for negative values of x, and $f(x) = x$ otherwise giving a desired nonlinear transformation of the output, see figure 2.2. The ReLU function is most frequently used in neural networks (Goodfellow, Bengio, and Courville, 2016) and piecewise linear models (Holmes and Mallick, 2001), and is defined as

$$f(x) = max(0,\ x). \tag{2.2}$$

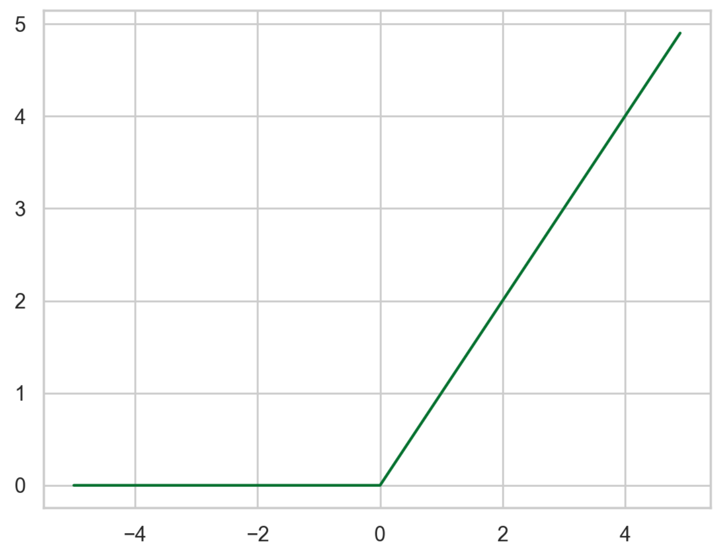

Figure 2.2.: The ReLU function

2.3.3. Maximum Likelihood Estimation

The optimal parameters $\boldsymbol{\theta}$, for a model $y(\boldsymbol{x})$ can be learned by maximizing, or equivalently minimizing, the negative logarithm of the likelihood function (Goodfellow, Bengio, and Courville, 2016) given by

$$\mathcal{L}(\boldsymbol{\theta}) = \log L(\boldsymbol{\theta}) = \sum_{i=1}^{n} y^{(i)} \log y(x^{(i)}) + (1 - y^{(i)}) \log(1 - y(x^{(i)})). \tag{2.3}$$

.

2.3.4. Regularization

All machine learning models have to be trained such that a balance between bias and variance is found so that the total error is minimized, heuristically described in figure 2.3. Generally, with competing models, the simplest one with the fewest assumptions should be selected (Gareth James and Tibshirani, 2017). A good model's parameters (features) $\boldsymbol{\theta}$ should have the following characteristics:

- Few features are relevant (few non-zero θ_i) such that the L1 norm $||\boldsymbol{\theta}||_1$ is small
- The standardized relevant features hold approximately equal importance (most θ_i are uniform) such that the L2 norm $||\boldsymbol{\theta}||_2^2$ is small

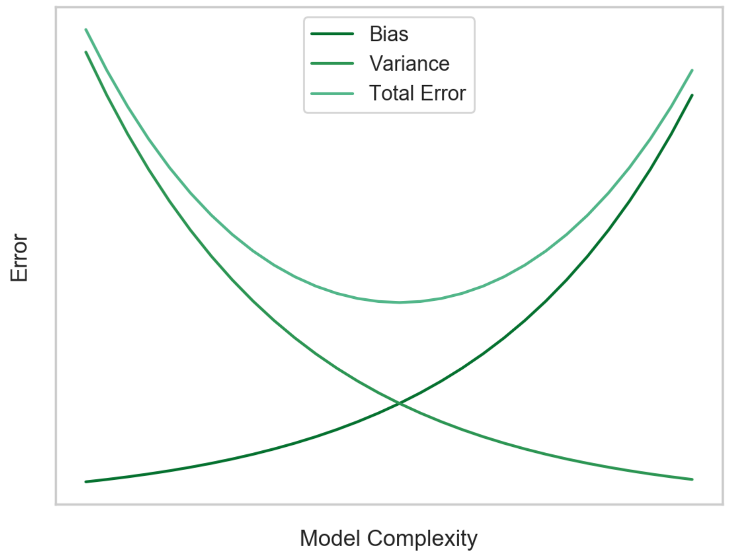

Figure 2.3.: Bias-variance trade-off

In training models that support regularization (e.g. Logistic Regression), the following main approaches exist for adding a regularization term to the optimization problem and thus injecting bias to the model:

- Ridge regression adds a $||\boldsymbol{\theta}||_2$ term
- LASSO regression adds a $||\boldsymbol{\theta}||_1$ term (Tibshirani, 1996)
- ElasticNet adds both an L1 and L2 term (Zou and Hastie, 2005)

Adding a regularization term gives an additional hyperparameter λ, indicating the trade-off between model accuracy and complexity, that needs to be tuned (Gareth James and Tibshirani, 2017).

2.3.5. Gini Index

The Gini Index is a measure of impurity and is defined for a node t containing different classes j as

$$\text{Gini}(t) = 1 - \sum_{j=1}^{J} p(j|t)^2, \tag{2.4}$$

where J is the total number of classes and $p(j|t)$ the relative frequency of class j (Gareth James and Tibshirani, 2017).

2.3.6. Ensemble Learning

Ensemble learning is typically utilized for classification by using multiple classifiers whose decisions are combined in a certain way. The classifiers are ensured different by randomizing the training data (Brown, 2010). Two general types of ensemble learning exist:

1. **Bagging (Bootstrap Aggregation)**:
   a) Overlapping training sets are created by random sampling with replacement
   b) Separate models are trained on these training sets
   c) New samples are classified by averaging the class predictions of the separate models
2. **Boosting**:
   a) Assign weights to each training sample
   b) Train a weak classifier on the training set
   c) Re-weight the examples such that the classifier focuses on hard examples and train a new weak classifier
   d) New samples are then classified by weighting together all weak classifiers by their accuracy

Among the two techniques, bagging tends to perform better when trained models differ considerably in performance between runs. This is more often the case for high-variance models, such as Random Forests and Neural Networks (Brown, 2010). Conversely, bagging becomes a worse choice for simpler models, such as Logistic Regressions. As for boosting, it is has beneficial over-fitting properties but can be sensitive to outliers, since new classifiers are obliged to compensate for the errors of its predecessors (Nazarenko, Varkentin, and Polyakova, 2019).

2.4. Logistic Regression

The Logistic Regression model is one of the most common for credit scoring (Luo, Desheng Wu, and Dexiang Wu, 2017) as well as in other fields (Dreiseitl and Ohno-Machado, 2002). Arguably, it is also the most simplistic, both in terms of intuition and number of hyperparameters to tune. This statistical model relies on the assumption of a Bernoulli distributed dependent variable, $y \sim Ber(q)$, which produces a binary output, $y \in \{0, 1\}$ with $p(y = 1) = q$, where q is the probability. Given a classifier on the form

$$y_i = \begin{cases} 1, & \text{if } \boldsymbol{\beta^T x_i} \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

,

where $\boldsymbol{\beta^T x_i}$ is our linear hyperplane with parameters $\beta_0, ..., \beta_n$. The probability, conditioned on independent variables, $\boldsymbol{x_1}, ..., \boldsymbol{x_n}$ is given by the logistic function such that

$$p(y = 1|\boldsymbol{x}, \boldsymbol{\beta}) = \frac{1}{1 + e^{-\boldsymbol{\beta^T x}}} \tag{2.5}$$

where the probability $p(y = 1|\boldsymbol{x}, \boldsymbol{\beta})$ describes the probability for y to take on the value 1 given $\boldsymbol{x}$ and the parameters of the model, $\boldsymbol{\beta}$. The binary output is determined by a threshold for the value of $p(y = 1|\boldsymbol{x}, \boldsymbol{\beta})$, commonly set to 0.5.

2.4.1. Training

The parameters β_i are subject to optimization in the learning algorithm and are found by Maximum Likelihood Estimation. Since the minimization of (2.3) has no closed form solution, numerical minimization procedures are used to acquire the optimal β values. This can be done through numerous methods, with one of the most common being gradient descent or stochastic gradient descent (Goodfellow, Bengio, and Courville, 2016). To control model complexity a regularization term can be added as described in section 2.3.4.

2.5. Decision Trees

The key idea behind decision trees is to successively split the data based on the informative properties of the features. Thus, if a feature is estimated to be a good predictor of the output, it will be placed close to the root of the tree. This is done successively in accordance with one or more splitting criteria connected to information gain and the desired shape of the tree. For a data set with n features, an $n - 1$ -dimensional hyperplane is drawn in the space of data points for each partition of the tree, illustrated in figure 2.4 and 2.5. When further partitioning is deemed not useful, the final layer of the tree, known as leaves, predict the output value for data points that satisfy the criteria set out by that specific path of the tree.

The example shows a tree of depth three with six leaves and two input features, x_1 and x_2. The output is binary, $y \in \{0, 1\}$.

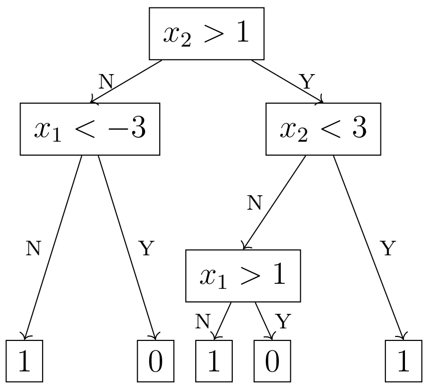

Figure 2.4.: Example of a binary decision tree

The partition above in figure 2.4 has a corresponding representation in the two-dimensional plane (figure 2.5), where the shaded regions correspond to output value 1.

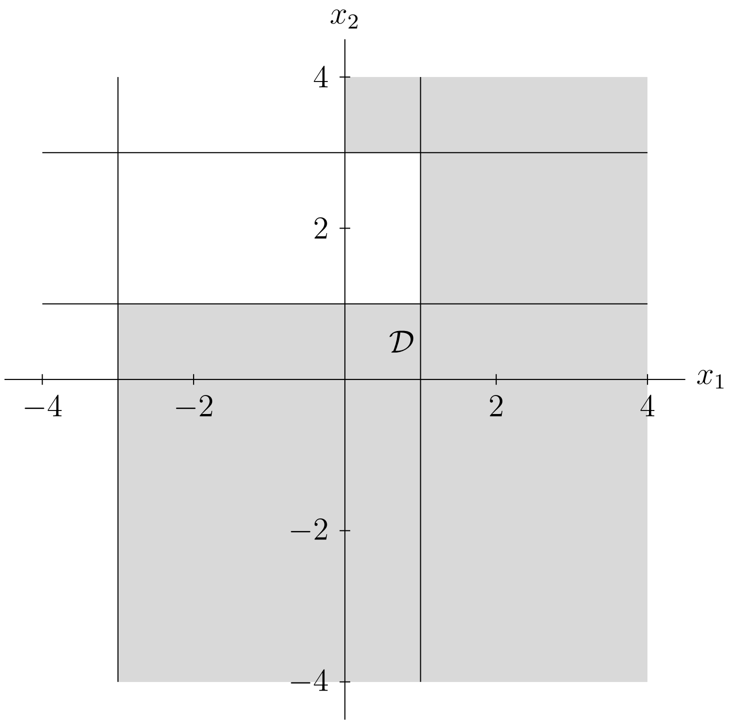

Figure 2.5.: Two-dimensional plane representation of the partition in figure 2.4

Decision trees suit both regression and classification problems. However, as a decision tree is not a state-of-the-art machine learning model, more sophisticated models using bagging, ensemble or boosting methods have been developed to achieve desirable performance (Gareth James and Tibshirani, 2017). Two such methods; Random Forest which is an ensemble method, and XGBoost which is a combined ensemble and boosting method, are covered in sections 2.5.2 and 2.5.3 respectively.

2.5.1. Training

Training a decision tree is done by optimizing the criterion that split the records. This criterion is defined such that the resulting nodes have low impurity, that is the class distributions are as homogeneous as possible. The Gini Index as defined in equation 2.4 is a common metric to achieve this. The criteria is chosen such that when splitting a node t on an attribute A into k children, the quality of the split is computed as

$$\text{Gini}_A = \sum_{i=1}^{k} \frac{n_i}{n} \text{Gini}(i), \tag{2.6}$$

where n_i and n are the number of records in the child and node t respectively. Choosing the attribute A that gives the largest reduction in impurity is then equivalent to choosing the smallest Gini_A (Gareth James and Tibshirani, 2017).

Additionally, a stopping criterion has to be determined to avoid overfitting which can be introduced due to too many branches. Overfitting can be handled through pre-tuning the tree by not splitting if the split quality falls below a certain threshold.

2.5.2. Random Forest

The Random Forest algorithm is a development of bagging described in 2.3.6 using the below algorithm. Consider a data set with N features:

1. Divide the training set into k partitions (trees) by using random sampling with replacement
2. Randomly select $n \leq N$ features and construct an optimized decision tree. Repeat this k times for all trees in the forest
3. Vote on new samples by taking the majority vote or average probability prediction (Gareth James and Tibshirani, 2017)

2.5.3. XGBoost

XGBoost is a newly developed algorithm that builds upon Gradient Boosting Machines (Chen and Guestrin, 2016). Gradient boosting, contrary to classic boosting, does not use re-weighting of training examples to update their weak learners (decision trees). Instead, a user defined differentiable loss function is used so that when a new tree is added it minimizes the loss of the current tree. That is, the parameters of the new tree are chosen so that the residual loss is minimized, hence the name gradient boosting.

XGBoost differs from gradient boosting in a number of ways by implementing both system optimizations and algorithm enhancements. System optimizations allow for more efficient training, drastically reducing the training time compared to gradient boosting. In credit scoring, the two most important algorithm enhancements include:

- Allowing for Ridge and LASSO regularization to prevent overfitting
- Allowing for sparse data by learning missing values. Sparse data may be the cause of one-hot encoding or zero-values (Chen and Guestrin, 2016)

2.6. Artificial Neural Network

An artificial neural network takes inspiration from how our brain works. It consists of the following:

1. An input layer with as many nodes as input features
2. One or more hidden layer consisting of a variable number of nodes
3. An output layer consisting of one (for binary classification) or many (for multi-class classification) output nodes

The input nodes take the value of the corresponding input feature. Any node in the hidden or final layer weights nodes together from the previous layer plus a constant bias. This weighted sum is then parsed through a non-linear activation function, such as the Sigmoid function in figure 2.1 or the ReLU function in figure 2.2. It is the weights in the neural network trees that are changed during training. The weights are changed in such way that an objective/loss function is minimized (Goodfellow, Bengio, and Courville, 2016). Typical objective functions are the Mean Squared Error, corresponding to a linear regression:

$$\text{Error}(f(\boldsymbol{x}^i), y^i) = \frac{1}{2}(y^i - f(\boldsymbol{x}^i))^2 \tag{2.7}$$

or Cross-Entropy, corresponding to a logistic regression:

$$\text{Error}(f(\boldsymbol{x}^i), y^i) = -y^i \log f(\boldsymbol{x}^i) - (1 - y^i)\log(1 - f(\boldsymbol{x}^i)), \tag{2.8}$$

where $f(\boldsymbol{x}^i)$ is the models prediction for sample i and y^i the actual value. For an example of a neural network with one hidden layer (figure 2.6) the output function $f(\boldsymbol{x})$ for input variables $\boldsymbol{x}$ can be written as

$$
\begin{aligned}
f(\boldsymbol{x}) &= v_0 + \sum_{h=1}^{H} v_l z_h \\
&= v_0 + \sum_{h=1}^{H} v_l \sigma\left(w_{h0} + \sum_{j=1}^{p} w_{hj} x_j \right),
\end{aligned} \tag{2.9}
$$

where $\sigma(\cdot)$ is the activation function and w_{ij} the weight from node i to node j. The error is calculated by forward propagation (Goodfellow, Bengio, and Courville, 2016). That is, for a given selection of weights, $f(\boldsymbol{x})$ and the objective function is calculated. Weights are initialized randomly.

2.6.1. Training

The optimal weights are calculated by optimizing the objective function through back-propagation of the error to the different weights by iteratively applying the chain rule. For the example in figure 2.6 one can calculate how much a weight between the input and first hidden layer, w_{hj} affects the objective function by

$$
\frac{\partial E}{\partial w_{hj}} = \frac{\partial E}{\partial f(\boldsymbol{x})} \frac{\partial f(\boldsymbol{x})}{\partial z_h} \frac{\partial z_h}{\partial w_{hj}}. \tag{2.10}
$$

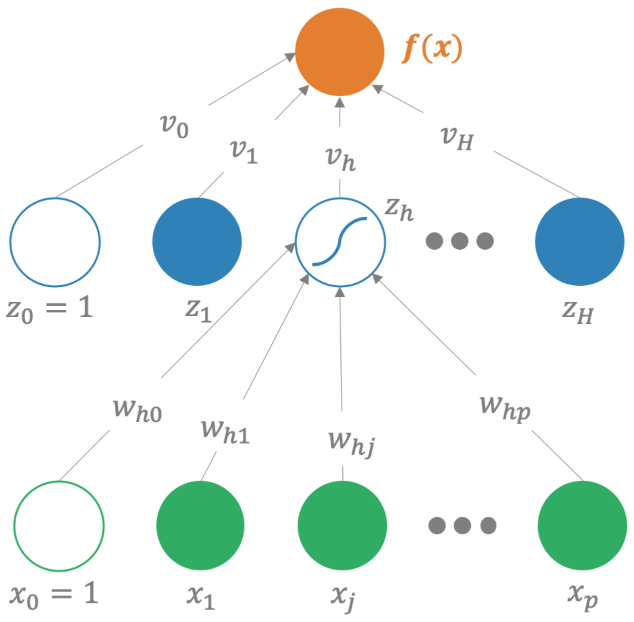

Figure 2.6.: Artificial Neural Network with one hidden layer and binary output

2.7. Bayesian Optimization for Hyperparameter Tuning

Bayesian Optimization (BO) is an optimization framework aimed at expensive black-box functions (Feurer M., 2019), meaning functions that are not known in advance with regard to either their values or derivatives, and are slow to evaluate. It optimizes said function by assuming a surrogate model of the underlying black-box function, which is successively fitted as more data of the underlying function becomes available (Frazier, 2018). In a machine learning context, the black-box function is the objective function value, with the relevant model hyperparameters as the explanatory variables. As models with different hyperparameter configurations are tested, the available information of the black-box function increases. Thus, BO is able to make increasingly informed decisions on which configuration to try next.

To decide whether a certain hyperparameter setting is worth evaluation, BO employs the use of an acquisition function (Frazier, 2018). As acquisition function input, the expected mean and standard deviation of a configuration is used. Of all configurations, the one with the highest acquisition value is the next configuration to be tested (Feurer M., 2019).

2.7.1. Bayes' Theorem, Prior and Posterior

By not assuming any knowledge about the function being optimized, BO instead works by using Bayes' Theorem to guide the search to the optimum. Bayes Theorem, for the conditioned probability of an event, is defined as

$$P(A|B) = P(B|A)\frac{P(A)}{P(B)}. \tag{2.11}$$

One can remove the normalizing quantity $P(B)$ since the actual value of the conditioned is not of interest but rather its relative quantity. This leaves

$$P(A|B) = P(B|A)P(A), \tag{2.12}$$

where $P(A)$ is the probability for an event *prior* to any new evidence, $P(A|B)$ the probability for an event *posterior* to any new evidence and $P(B|A)$ the probability to observe that evidence.

In BO one collects data points $x_1, ..., x_n$ and corresponding function values given by the objective function $f(x_1), ..., f(x_n)$ into a data array $D = \{x_1, f(x_1), ..., x_n, f(x_n)\}$. The function values $f(x_i)$ are now used to define the prior $P(f)$. Furthermore, the likelihood function is defined as the probability of seeing the data D given the function f, that is

$P(D|f)$ (Shahriari et al., 2015). As new points are added to the data, the likelihood function changes. This yields the new equation

$$P(f|D) = P(D|f)P(f). \tag{2.13}$$

The posterior $P(f|D)$ is an approximation of the objective function given our current knowledge D, called a surrogate model.

2.7.2. Surrogate Model

As described in figures 2.7 and 2.8, the surrogate model aims to model the underlying black-box function by fitting to the available function samples. A commonly employed surrogate model is the Gaussian Process, which constructs a joint probability distribution over the variables, on x and $f(x)$. That is, it assumes the variables follows a multivariate Gaussian distribution (Shahriari et al., 2015).

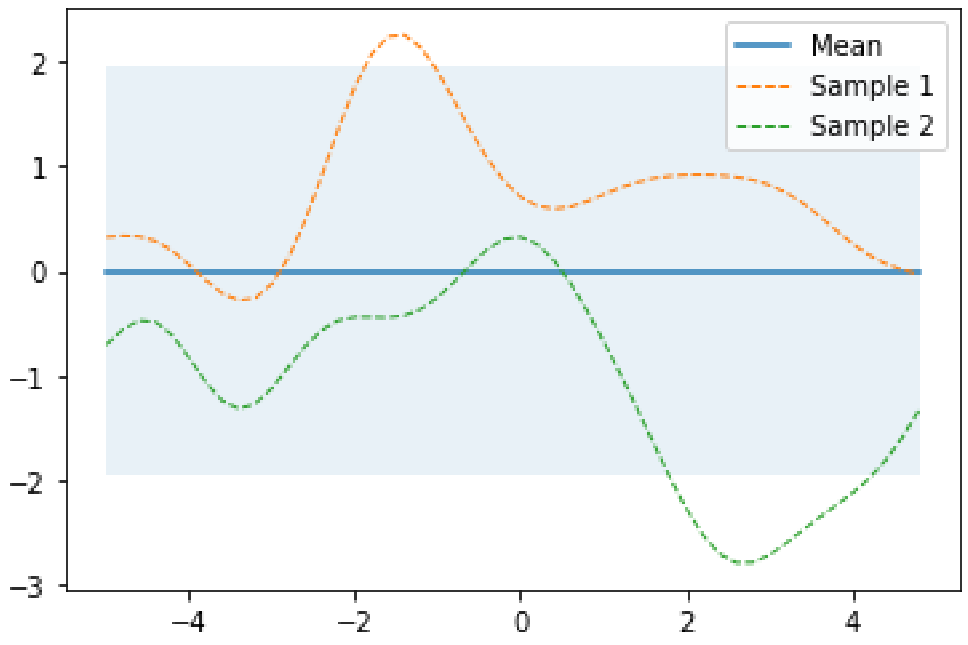

Figure 2.7.: Realizations of a Gaussian Process before evaluation, squared exponential kernel function

Prior to function evaluations, the black-box function is assumed to have a constant mean (blue line) and standard deviation (light blue area). Two sample processes, or potential black-box functions, can be seen in yellow and green (figure 2.7).

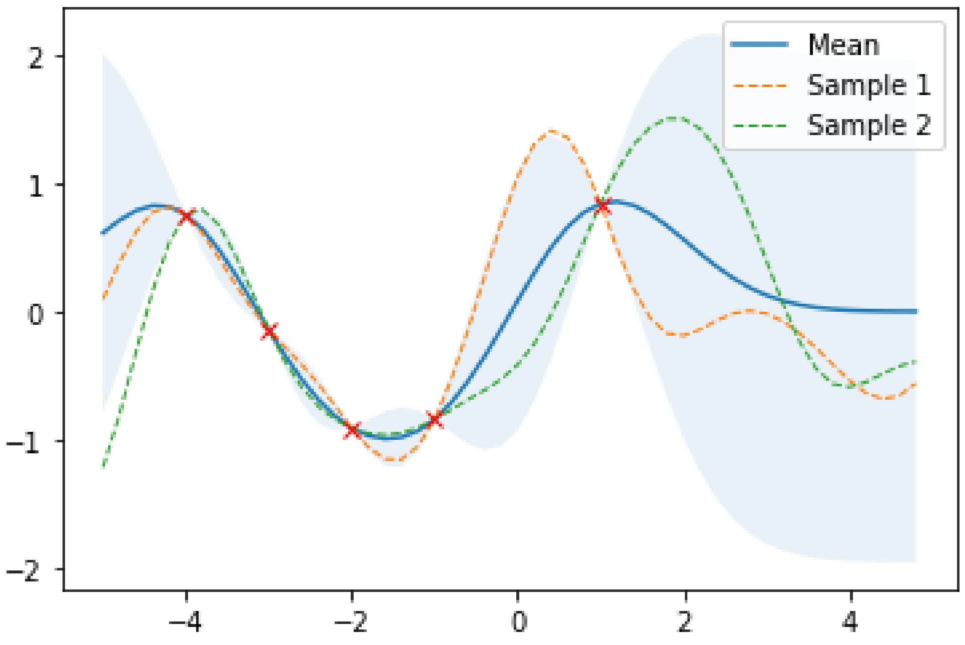

Figure 2.8.: Realizations of a Gaussian Process after five evaluated points, squared exponential kernel function

After adding evaluated points, the Gaussian Process is fitted to the data. This creates regions of higher uncertainty further away from the data, as pictured by the light blue regions (figure 2.8).

2.7.3. Acquisition Function

To interpret the modeled mean and standard deviation produced by the surrogate, an acquisition function is used. The main purpose of the acquisition function is to balance exploration (testing uncertain configurations) and exploitation (testing configurations that are likely to perform well). The next configuration to evaluate is found through the $argmax$ of the acquisition function on the domain. Common acquisition functions are listed below.

Upper Confidence Bound

The Upper Confidence Bound (UCB) function weights the standard deviation by a coefficient β to enable a varied trade-off between exploration or exploitation. β can either be set throughout the BO process or vary according to a schedule (Shahriari et al., 2015).

$$UCB(\mathbf{x}) = \mu(\mathbf{x}) + \beta\sigma(\mathbf{x}) \tag{2.14}$$

where $\mathbf{x}$ is a valid configuration.

Expected Improvement

The Expected Improvement(EI) function makes use of the standard probability density function, ϕ, and cumulative distribution function, Φ, to compute the acquisition value accordingly (Feurer M., 2019),

$$Z = \frac{f_{min} - \mu(\mathbf{x})}{\sigma(\mathbf{x})} \tag{2.15}$$

and the expected function value at configuration $\mathbf{x}$. Let

$$EI(\mathbf{x}) = Z\sigma(\mathbf{x})\phi(Z) + \sigma(\mathbf{x})\Phi(Z) \tag{2.16}$$

where Z represents the normalized difference between the current function optimum and the expected function value at configuration $\mathbf{x}$.

Chapter 3.

Resampling of Data

A problem in ours and many other classification problems is that the data set is highly imbalanced with relatively few samples of the minority class. Furthermore, it is often the minority class that is of interest for classification. Various techniques exists to counter this, of which two has been used in this report.

3.1. Upsampling using SMOTE

SMOTE or Synthetic Minority Oversampling Technique is an upsampling technique that instead of duplicating samples in the minority class creates artificial new ones. It does this by randomly selecting one of k neighbors to a sample and generate a new sample on the line between the sample and its neighbor (figure 3.1). One can continue to d o this until the desired class distribution is achieved (Chawla et al., 2002).

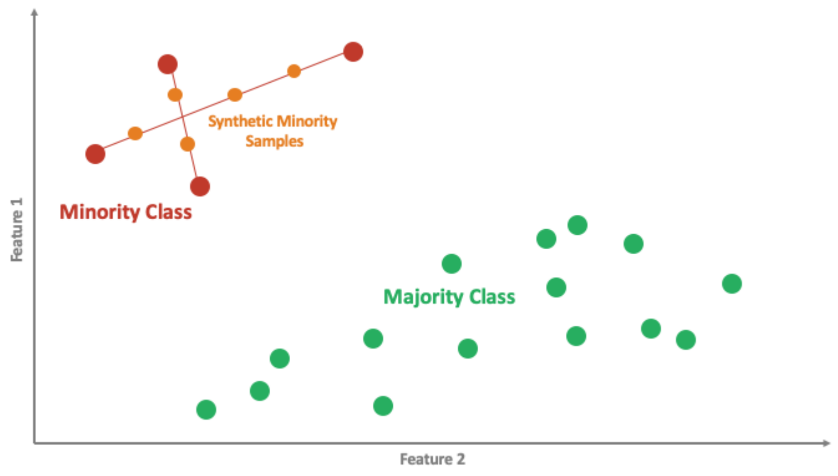

Figure 3.1.: Example of SMOTE synthesizing new samples between existing samples

One of the main disadvantages with SMOTE is that it creates new samples of the minority class without considering the majority class, which can result in overlapping examples. This makes it more difficult to draw a clear decision boundary.

3.2. Random Downsampling

Random downsampling involves randomly selecting samples from the training data and removing them. The advantage is that it balances the class distribution without altering the information of the minority class. However, valuable samples in the majority class may be lost resulting in a less robust decision boundary. As described in (Chawla et al., 2002), it has been proven effective for highly imbalanced sets to first randomly downsample (trim) the majority class followed by applying SMOTE to the minority class.

Chapter 4.

Evaluation

4.1. Feature Metrics

The feature selection metrics that were used were based on individual feature importance and correlation. By evaluating on feature importance one can study how much an individual feature affects the predicted outcome. The exact implementation routine is described in chapter 5.

4.1.1. Feature Significance for Logistic Regression

The coefficients β_j in a Logistic Regression model, where all continuous features have been standardized, give an influence of how individual features affect new predictions and in which direction.

4.1.2. Feature Significance for Random Forest and XGBoost

For a single decision tree feature importance is calculated by how much each feature's split point in the tree improves the performance metric, in our case the Gini Index. The total feature importance is then given by the weighted sum of these improvements, where each weight is the number of samples the node is accountable for. For ensemble models such as Random Forest and XGBoost, the final feature importance is calculated by averaging the importance given by each decision tree. The direction of importance is not possible to extract with this method alone.

4.1.3. Feature Significance for Neural Networks

As the weights of a Neural Network are not sufficient to explain the importance of a certain feature, due to their cross linking nature, another approach is needed. SHAP or SHapley Additive exPlanations is a method developed in 2016 (Lundberg and Lee, 2017) that builds upon game theoretic Shapley values (Winter et al., 2002). In itself, Shapley values calculates the contribution from each player in a collaborative game. However, applied to machine learning model interpretability, SHAP values lets us evaluate how much a single feature contributes to the performance of the model, relative to the model's baseline prediction. Let F be the full set of features and take $S \subseteq F$ where feature i is excluded. Now, compute the difference of a model $f_{S \bigcup i}$ where i is present and f_S where i is excluded by

$$\phi_i = \sum_{S \subseteq F} \frac{|S|!(|F| - |S| - 1)!}{F!} \left[f_{S \bigcup i}(x_{S \bigcup i}) - f_S(x_S) \right], \tag{4.1}$$

where x_S represents the input feature values. SHAP values can be used for both global and local interpretability, that is for the results as a whole and for a specific individual, making them very useful on a case by case basis to understand what drives a certain outcome (Lundberg and Lee, 2017).

4.1.4. Feature Reduction using Correlation

One can use correlation plots to determine how related changes are between different features. Some models such as Logistic Regression have poor performance with highly correlated variables (Courvoisier et al., 2011). Thus, for these models the least significant of two highly correlated features needs to be removed. The correlation between two features X and Y is calculated as

$$\rho_{X,Y} = \frac{E[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y} \tag{4.2}$$

where $E(\cdot)$ is the expected value, μ the mean value and σ the standard deviation. Correlation values vary between -1 and 1, where 1 indicates that the two features vary completely together while 0 indicate no correlation at all.

4.2. Model Metrics

For a trained model in binary classification (0 or 1), predicting the outcome of unseen samples yields either (Davis and Goadrich, 2006):

- **True Positive (TP):** Correctly classifying a sample as 1
- **False Positive (FP):** Incorrectly classifying a sample as 1
- **False Negative (FN):** Incorrectly classifying a sample as 0
- **True Negative (TN):** Correctly classifying a sample as 0

This information is captured in what is called a confusion matrix, which for the case of credit scoring can be seen in figure 4.1.

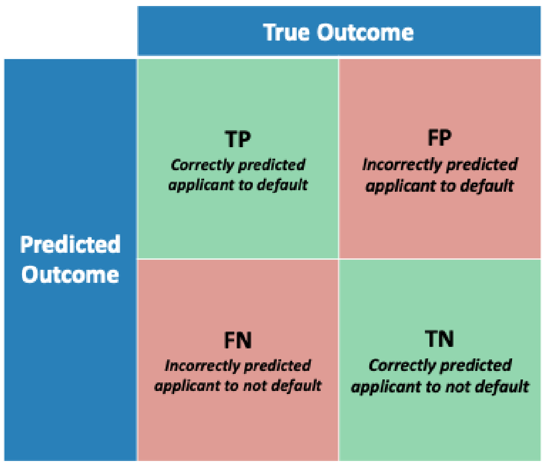
True Outcome
Predicted Outcome
TP
Correctly predicted applicant to default
FP
Incorrectly predicted applicant to default
FN
Incorrectly predicted applicant to not default
TN
Correctly predicted applicant to not default

Figure 4.1.: Confusion Matrix

From the confusion matrix a certain number of basic performance metrics can be extracted. The most common ones are:

$$\begin{aligned}
\text{Recall/True Positive Rate (TPR)} &= \frac{TP}{TP + FN} \\
\text{Precision/Positive Predicted Value (PPV)} &= \frac{TP}{TP + FP} \\
\text{Accuracy (ACC)} &= \frac{TP + TN}{TP + TN + FP + FN} \\
\text{F1 score (F1)} &= 2 \cdot \frac{PPV \cdot TPR}{PPV + TPR}
\end{aligned} \tag{4.3}$$

which in more simple terms for our problem can be characterized as:

- **Recall**: how many of the *actual* defaults did the model predict to default
- **Precision:** how many did *actually* default of the ones the model predicted to default
- **Accuracy:** how many correct guesses did the model have overall
- **F1 score:** harmonic mean of precision and recall

Based on these fundamental metrics a number of additional metrics can be constructed. However, not all metrics are well suited for all problems. In this case, with only 22% of the applicants defaulting, the data set is highly imbalanced. Thus, if one would use accuracy as the evaluation metric to tune models on said data, one of the best models would be the one guessing for non-default all the time, since that would yield an overall accuracy of 78%. However, it would not catch a single defaulting customer and would be useless for our purposes.

Instead, from a business perspective one typically looks at the contribution margin of a customer, which for high risk loans becomes "how many non-defaulting customers do we need to have for each defaulting". Translated in to precision and recall, this becomes: "given that we detect a certain share x of *actually defaulting customers* (recall level of x), how large is the share y of *actually non-defaulting customers* that we lose due to incorrect predictions (precision level of $1 - y$). For the detection of non-defaulting customers, the logic is reversed. In total one would like to keep all occurrences of FP (loss in potential revenue) and FN (increased cost due to defaults) to a minimum. However, due to the nature of the unbalanced data set, one needs to target the minority class to achieve this. Using the Average Precision (AP) balances the precision and recall for our targeted class (defaulting applicants) (Su, Yuan, and Zhu, 2015). Average Precision is defined as

$$\text{AP} = \sum_{n=1}^{N}(R_n - R_{n-1})P_n, \tag{4.4}$$

where R_n and P_n is recall and precision at threshold n. Average Precision gives the weighted mean of precisions with the recall delta as weights, for all thresholds N. Different thresholds is obtained by changing the limit for which the model predicts either 0 or 1. For example by raising the threshold from 0.5 to 0.6, the model would need to be more certain before predicting 1 and vice versa by lowering the threshold to 0.4. The precision and recall values at each threshold can also be plotted in what is called a precision-recall curve to evaluate in which recall ranges the model drops in precision and where it is fairly stable. Furthermore, tying Average Precision to above discussion, one is typically more interested in "for a certain level or range of recall, what is our precision", which are also the metrics that are in place currently at the credit institute. Now, let

- **AP1** be the average precision in recall range 20-35%,
- **AP2** be the average precision in recall range 35-50%,
- **PRX** be the precision at exactly 35% recall.

These metrics are jointly going to be the criterion on which all models are evaluated, along with their simplicity and interpretability.

4.3. Base Model

To evaluate the impact of transaction based features, one would like to compare the performance with an optimized model only using data from the loan application form. Thus, any new constructed features could then be added to this base model to check for a significant improvement. The features used in the base models are not included in this report since they are not the focus. Instead, our aim is to evaluate and find significant transaction based features which improve upon a given base model. Note that a base model needs to be constructed and optimized for all types of models.

4.4. Cross Validation

Firstly, all models should be trained and tested on completely separated sets and testing should be done on completely unseen samples. Furthermore, if one tunes and compares different models using the testing data set it is impossible to know the true generalization error, since the test data was in fact used to train the model (through optimization of hyperparameters or comparison of different models). Thus, one always needs three sets: a training, validation and testing set. However, when working with small data sets, to avoid using a fairly large dedicated portion of the data to validate the models one can use cross validation. Cross validation is performed by keeping a test set for final evaluation but instead of using a static validation set one splits the training set into smaller chunks. Typically, k-fold cross validation is used (Gareth James and Tibshirani, 2017) as shown in figure 4.2, which works as:

1. Split the training data into k blocks
2. Train the model on $k - 1$ blocks and test on the remaining block
3. Iteratively change the blocks to train and test k times
4. Average the performance from each iteration

The number of folds varies depending on the goal of the cross validation and the size of the training data set.

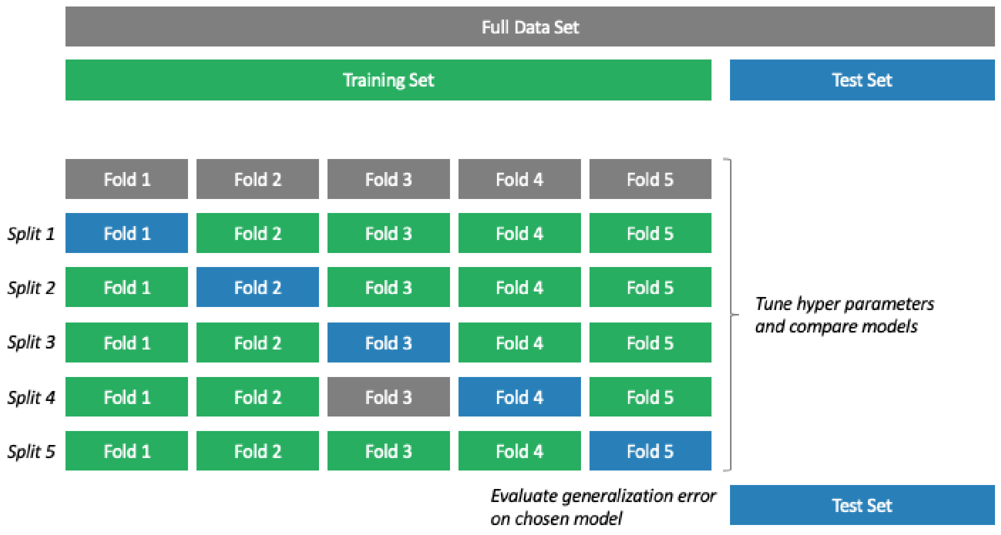

Figure 4.2.: K-fold cross validation scheme

Chapter 5.

Implementation

5.1. Setup details

For the tests conducted, a number of software libraries were used to simplify the process. For all programming and most data analysis, Python was the language of choice, with some additional data analysis and plotting taking place in Microsoft Excel.

5.1.1. Pandas

For all manipulation and storage of data, the Pandas library (*pandas - Python Data Analysis Library* n.d.) was used. Pandas is a purpose-built tool for conducting data analysis in Python.

5.1.2. Scikit-Learn

Scikit-Learn (Pedregosa et al., 2011) is an open-source machine learning library for Python. The library was used for preprocessing and utilities, as well as for the Logistic Regression and Random Forest models.

5.1.3. XGBoost

The XGBoost(*About* n.d.) model was implemented through the official library produced by the model creator (Chen and Guestrin, 2016).

5.1.4. Tensorflow

Tensorflow (*Why TensorFlow* n.d.) is a machine learning library that offers custom model-building through its Keras API. Due to its superior Neural Network functionality over Scikit-Learn, it was used specifically for those models.

5.1.5. Bayes-Opt

Bayes-Opt (Nogueira, 2014) offers an out-of-the-box implementation of Bayesian Optimization.

5.1.6. SHAP

SHAP (*slundberg/shap: A game theoretic approach to explain the output of any machine learning model.* N.d.) is the official library for producing model interpretability, created by the author of the original SHAP paper (Lundberg and Lee, 2017).

5.2. Transaction Categorization

After a quick analysis of different machine learning clustering models it was decided that a pure machine learning approach to categorization was neither the most effective way nor the most accurate. One of the main issues was that these method did not take advantage of our or the credit institute's previous know-how of transactions. Instead a conventional approach was taken as the foundation for this thesis. However, as described in section 8.1, a machine learning aid for categorization was developed for the credit institute but was not studied enough to be part of this thesis.

5.2.1. Raw Data

The raw transaction data that was stored in a tree like format. The following information was extracted, transformed and used in future steps:

- Errand ID
- Report date
- Bank information
- Account information
- Transactions per account with amount, value date, balance and description

5.2.2. Categorization Pipeline

The following high-level actions were taking to categorize the data:

1. Transformation of raw tree data into a table based format containing the most important information
2. Cleaning of description field and removal of certain nonsensical stop words

3. A per word prioritized lookup matching either a word (e.g. "aftonbladet"), combination of words (e.g. "a kassa"), part of a word (e.g. "försäk") or by a certain format using a regular expression (e.g. "(\s1,|$\hat{)}$((0|46)(70|72|73|76|79))\d7"). The lookup terms are fetched from an external spreadsheet making addition, edits and deletions simple. In total, approximately 1400 match terms were used.
4. Usage of logical rules to capture
   a) Transactions not categorized in previous step (e.g. rent not containing any of the words used for rent)
   b) Wrongly categorized transactions (e.g. credit card expenses which are actually credit loan expenses)
   c) Internal transactions

5.2.3. Proposed Categories

After several iterations, to facilitate feature construction and model evaluation, only categories which the authors hypothesized to be important were included and evaluated. This did inject personal bias but was concluded to be the only feasible option. While the underlying hypotheses for each category's effect on default rate can be argued, they were ultimately deemed the most credible by the authors and the company. The final selection and underlying assumptions regarding categories was:

- **Allowance**: any non-salary and non-transfer income (e.g. "a kassa"). The hypothesis was that people with large allowances relative to their salary have a higher default rate.
- **ATM Withdrawal / atm deposit**: negative and positive transactions from an ATM. The hypothesis was that frequent usage and/or large sums from ATMs show suspect behavior in a country like Sweden were most transaction are done by card. Suspect behavior as this could point to other underlying problems and a higher default rate.
- **Consumption, Fixed**: any purchases deemed necessary for a basic living standard. The hypothesis was that this should be fairly consistent and limited spend, and would thus indicate a low default rate.
- **Consumption, Leisure**: any non-essential purchases. The hypothesis was that high frequency and large sums indicate volatile behavior and a higher default rate.
- **Credit Loan Expense / Credit Loan Income**: any credit loan expense or income. Credit income can take the form of several withdrawals from a credit account or of one single payment. Hypothesis was that high frequency and total sum indicate a higher default rate.

- **Credit Card**: any credit card expense. Credit cards are common but should, given an errand, be consistent and non-volatile. The hypothesis was that high volatility and large expenses indicate a higher default rate.
- **Gambling Expense / Gambling Profit**: any gambling related expense or profit. The hypothesis was that frequent and large gambling expenses indicate an addiction and thus a higher default rate.
- **Debt Collector**: any debt collector expense, originating from unpaid bills. The hypothesis was that large number of debt collector errands with larger sums indicate a higher default rate.
- **Insurance**: any insurance expense. The hypothesis was that consistent and non-volatile insurance expenses indicate a lower default rate.
- **Kronofogden**: governmental enforcement agency which takes over if bills are not paid to a debt collector company. The hypothesis was that any Kronofogden errand indicates a higher default rate.
- **Rent**: any rent expense. The hypothesis was that relatively small, consistent and non-volatile rent expenses indicate a stable economy and a lower default rate.
- **Salary**: any salary income. The hypothesis was that large, consistent and non-volatile salaries indicate a stable economy and a lower default rate.
- **Swish Expense / Swish Income**: any Swish expense or income. Swish is an application for direct money transfer widely used in Sweden. The hypothesis was that many large Swish expenses or incomes indicate a higher default rate.

Additional categories were available but were not used in the modelling. The purpose of making a thorough categorization beyond what is used in the credit scoring process is that the information can also be used by the credit institute's handlers when making human judgement and by a machine learning categorization algorithm as as it yields a larger and more accurate data set.

5.2.4. Evaluation of Proposed Categories

After analyzing the number and total sum of transactions in different linear amount based bins, it was concluded that in almost all categories, outlying behaviors would make it difficult for any model to find a reasonable decision boundary. Thus, it was decided to analyze and divide the transactions by amount in the different categories, corresponding to different hypothesized behaviors. Taking *credit loan income* as an example:

- Small incomes of <1000 SEK should reflect credit withdrawals. Frequency and sum was of high interest.

- Larger incomes of 1000 - 25000 SEK should reflect traditional one-time loan payments. Frequency and sum was of high interest.
- Outlier incomes of $>$ 25000 SEK is suspect, unknown and outlying behavior that needs separate treatment. Only a count was of interest.

Similar analysis was conducted for all categories to determine suitable amount ranges. All corresponding graphs can be found in appendix A.

5.3. Credit Scoring of Applicants

When using transaction data for credit scoring, a number of approaches have been proposed. The approach proposed in this thesis attempts to consider the frequency, volatility and overall spending amounts for all relevant categories. It should be noted that all considered transactions have an affiliated category. However, as the accuracy of the transaction categorization is not perfect, the affiliated categories are not always accurate.

5.3.1. Feature Construction for Credit Scoring

Features are generated based on

- 17 main categories based on different types of spend
- 2-4 amount based sub-categories accounting for different types of behavior
- 2-3 metrics further capturing different types of behavior

Moreover, as these metric for all customers might show linear, log-linear or other types of patterns, resulting in several hundreds of potential features, it was decided only to considered average spend per month and not cut the data in to different time bins. This simplifies the process but fails to capture any recency in the data. One or more of the following metrics were calculated for each sub-category:

- **Frequency**: number of occurrences per month
- **Mean**: average transaction sum per month
- **Standard Deviation**: standard deviation of monthly transaction sum

Derived Features

In addition to features being generated directly based on spend categories, some additional features were generated to capture a certain behavior, for example:

- **swish_net**: monthly average net of received and sent Swish payments

It deserves to be said that when evaluating the corresponding balance in the raw data, a strange behavior was observed. It could be seen that over 90% of all customers did have a negative balance sometime during the time period. Thus, it was decided to not include any balance-based metric in the analysis.

Feature Processing

As the different metrics does not need to show a linear behavior, a logarithmic counterpart was added for each metric. Additionally, for models that take advantage of standardized data (such as any linear model) all metrics were standardized by subtracting the mean and dividing by the standard deviation for that feature. Also, to boost performance in the feature selection, some of the features were discretized into an ordinal categorical variable.

5.3.2. Feature Selection and Model Evaluation

As the number of features was very large, feature selection was performed in two steps. Common pitfalls when working with raw data with no clear features is that too many features are created which may or may not lack logical reasoning but which together does not improve model performance and make it difficult to find where a problem lies. A more controlled approach involves starting with a base model and in a controlled manner adding more features. In this thesis, feature selection was performed in two steps.

As a first step, individual feature importances were determined for each of the Logistic Regression, Random Forest and XGBoost models. For Logistic Regression, the feature importance was represented by the coefficient value of the feature when running it as the only transaction data-derived feature along with the socioeconomic variables. Combinations of high-scoring variable's were then tested in various combinations, successively adding complexity to the model. This was done while tracking the product of the features' coefficient values as a measure of combined feature importance, excluding unimportant features from further contention in a silo-like process. Features were not tested with their logarithmic counterpart.

As a second step in the process, the least significant of highly correlated features were removed with regard to their coefficient value. Such correlation plot can be seen in figure 5.1. The resulting set of models included one to eight transaction data-related features, and were tuned on their regularization parameter through Bayesian Optimization simultaneously with the evaluation. The top three models in each bracket of features were evaluated on their performance on the test set.

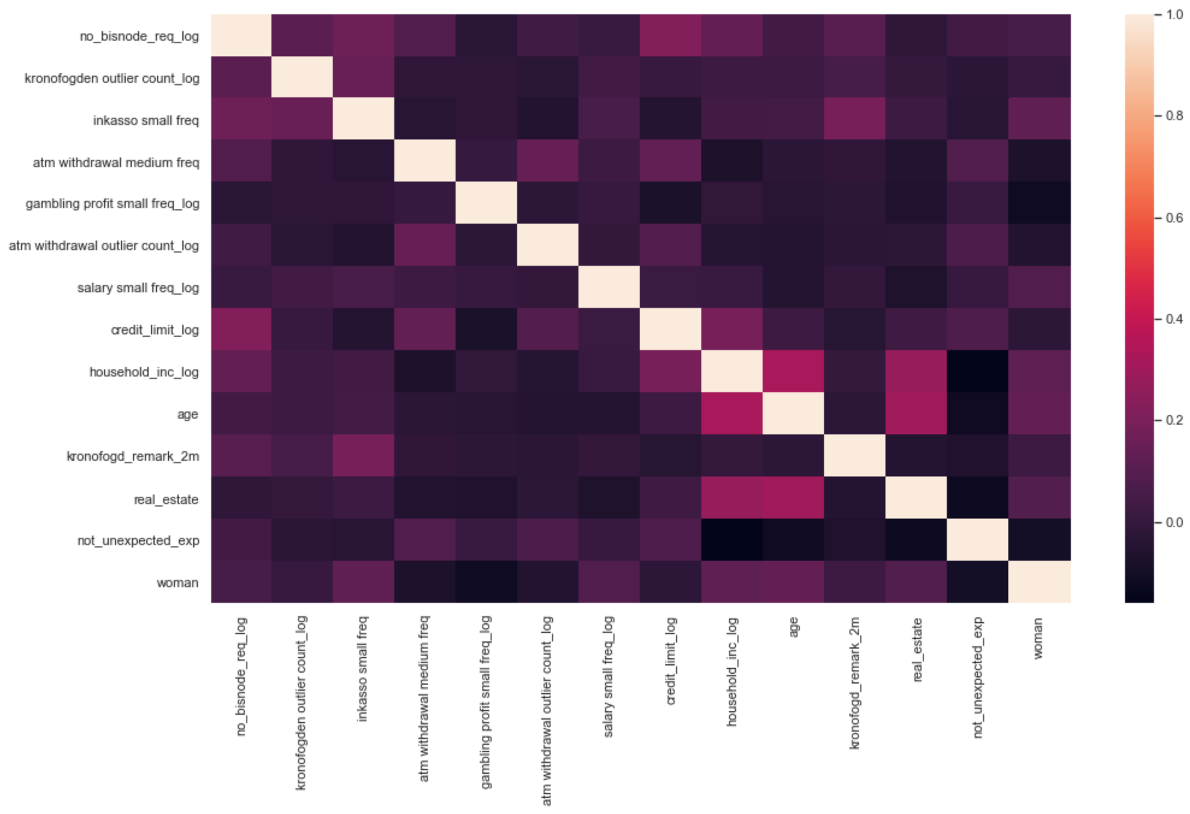

Figure 5.1.: Correlation plot for the logistic regression model features

For the tree-based models, Random Forest and XGBoost, the feature importance was determined by running a cross validated, not-tuned Random Forest algorithm with a very large number of trees. This process excluded logarithmic features, as these were expected to add no additional information for the tree-based models. From the not-tuned Random Forest, feature importances were extracted according to an impurity criterion and the features ranked thereafter. Models were then created by including a varying number of the most important features, from 5 to 60, and tuned through Bayesian Optimization. The final Random Forest was picked by evaluating average precision on the test set. This process was replicated in its entirety for XGBoost.

For the Neural Network, no feature selection was deemed necessary as this was believed to restrict the model's own ability to interpret the data. However, features with high correlation were removed prior to model training. To allow the model to restrict the number of features used for evaluation, an L1 regularization parameter was used as part of the tuning for the first hidden layer. This would allow the model to make the decision of restricting the number of used features itself.

The resulting selection of metrics was used as input features for the different models. Section 6.3 show the final features used in the different models excluding any metrics included in the base model.

Validation of Selected Features

All features' discriminatory power was continuously and validated throughout the process by plotting the default frequency and count in each amount bin. An example for the frequency of small *inkasso* expenses is shown in figure 5.2. As with many other features, it was difficult to find features which give a high number of occurrences in each linear bin.

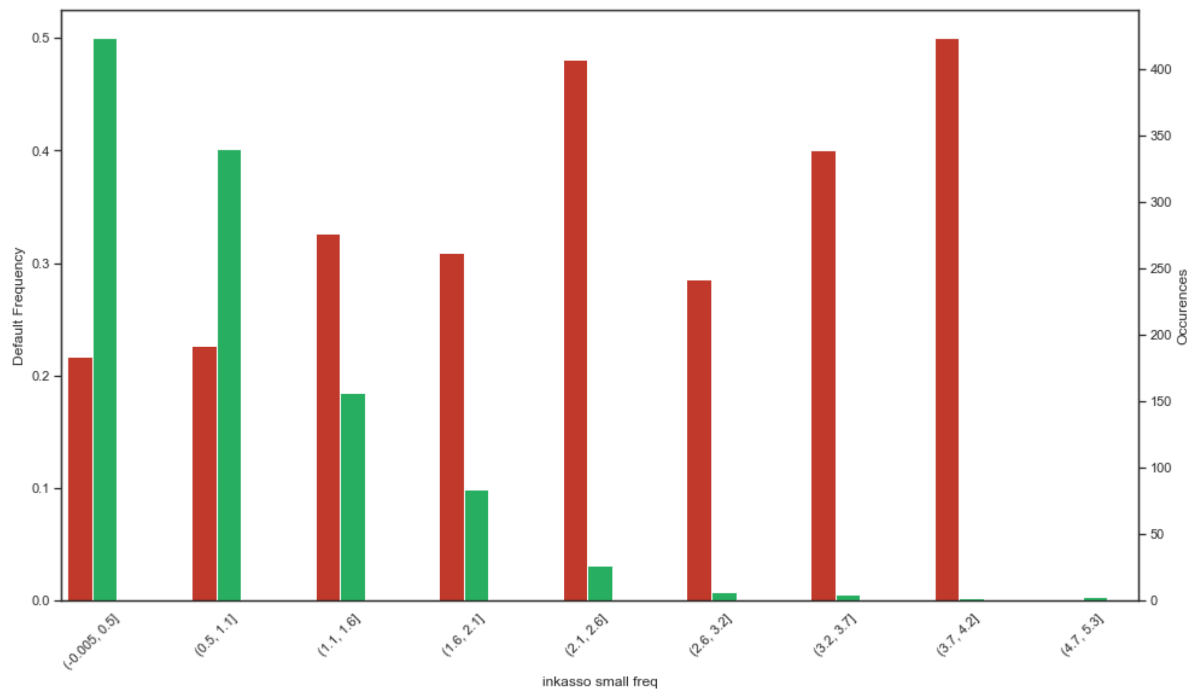

Figure 5.2.: Frequency of small inkasso expenses

5.3.3. Prior Knowledge Splits

By experimenting with manual splits of the data based on prior knowledge, an income-based split proved to give superior results to any other attempted split as well as non-split models and models excluding customers with data deemed insufficient. The split consisted of the following:

- Remove any applicant that had no traceable income that could be categorized as either salary or allowance
- Separate applicants with *both* allowance and salary from those with *either* allowance or salary
- Train models for the two resulting data sets with a common feature set

Throughout the training process, this split was compared to the baseline and non-split approach. In the results section, the split approach models will be referred to as 'Both' and 'Either', referring to the set of customers with *both* salary and allowance and *either* salary or allowance, respectively.

Chapter 6.

Credit Scoring Results

After selecting the most relevant features and tuning the models according to the scheme in chapter 5, they were compared to each other. All comparisons between models were done using SMOTE and a 10-fold cross validation scheme. For the final chosen model, the generalization error was calculated using the held-out test data. The evaluation metrics that were used focused on average precision, more precisely:

- **AP**: average weighted precision in recall range 0-100% using recall delta as weights
- **AP1**: average precision in recall range 20-35%
- **AP2**: average precision in recall range 35-50%
- **PRX**: precision at exactly 35% recall

Furthermore, since the results for all models had a tendency to fluctuate, all simulations were run 10 times. Average performance and standard deviation for all metrics was calculated in an effort to identify the most consistent model. For each metric; μ indicates metric mean and σ standard deviation. The following abbreviations for the different models is used:

Name	**Abbreviation**
Logistic Regression	LOG
Random Forest	RFO
XGBoost	XGB
Neural Network	NET

Table 6.1.: Model Name Abbreviations

For each metric shown in the tables below, the highest performer is indicated with green, the lowest with red and the rest with yellow.

6.1. Overall Performance

Tables 6.2 and 6.3 show the overall performances of all unsplit and split models, as well as the performance of the base model over 10 runs. For the split models, the performance metrics are computed for each model and averaged with weights corresponding to the number of test samples for each model. In the split data table, the base model is kept as reference.

Model	**AP**		**AP1**		**AP2**		**PRX**	
	μ	σ	μ	σ	μ	σ	μ	σ
BAS	0.410	6.7%	0.445	7.3%	0.424	6.5%	0.429	6.1%
LOG	0.430	6.7%	0.480	7.3%	0.452	7.4%	0.472	7.1%
RFO	0.424	5.9%	0.470	8.3%	0.435	6.9%	0.450	8.8%
XGB	0.409	5.7%	0.464	8.1%	0.429	7.5%	0.444	7.3%
NET	0.402	5.3%	0.443	9.8%	0.419	7.0%	0.436	9.0%

Table 6.2.: Overall Model Performance, unsplit data

Model	**AP**		**AP1**		**AP2**		**PRX**	
	μ	σ	μ	σ	μ	σ	μ	σ
BAS	0.410	6.7%	0.445	7.3%	0.424	6.5%	0.429	6.1%
LOG	0.450	7.3%	0.514	10.0%	0.478	8.1%	0.494	9.1%
RFO	0.408	6.2%	0.454	7.7%	0.417	6.3%	0.436	6.5%
XGB	0.425	6.8%	0.478	9.7%	0.455	9.1%	0.466	9.1%
NET	0.426	4.1%	0.473	6.8%	0.424	3.9%	0.444	6.5%

Table 6.3.: Overall Model Performance, split data

From the results tables above, it can be seen that the single-model runs are slightly more stable than the multi-model runs, but produce a lower AP value for all models but XGBoost. However, the single-model Logistic Regression produce the highest PRX-, AP1- and AP2-values, making it the best model over its best runs. However, its inconsistency is a drawback.

6.1.1. Precision-Recall Curves

To better visualize above metrics a precision-recall curve can be plotted to see how much and where the precision drops when recall increases for the different models. Figure 6.1 shows the precision-recall curves for all models given unsplit data. For the split model graphs, a weighted average of the included models were computed, with the weights being proportional to the number of samples in each model. For all plots, a run that has been deemed representative of the model's average performance has been used.

Unsplit data

As seen in Figure 6.1, the logistic regression including transaction features performs marginally better than the base model and pulling ahead significantly in the AP2-range. Other models are lagging behind by a slight margin.

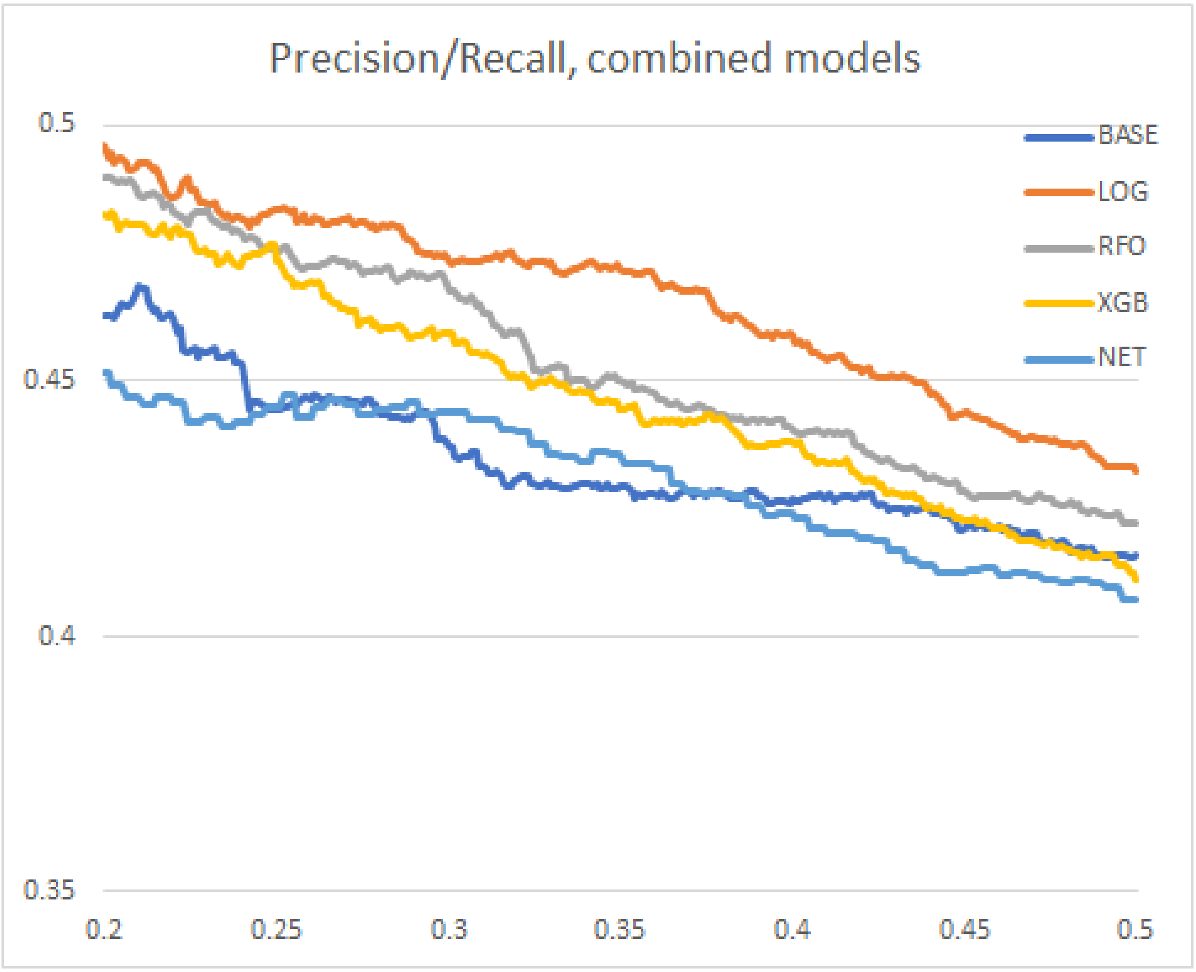

Figure 6.1.: Precision-recall curves for all models using unsplit data

Split data

As seen in Figure 6.2, the combined Logistic Regressions (yellow) pulls ahead of the other models. The base model is kept for reference. For all combined models, the performance of the Both-models were significantly higher than that of the Either-models, as exemplified by the two models included in the split model for Logistic Regression, Figure 6.3. The complete set of Both- and Either-model performances can be found in appendix C.

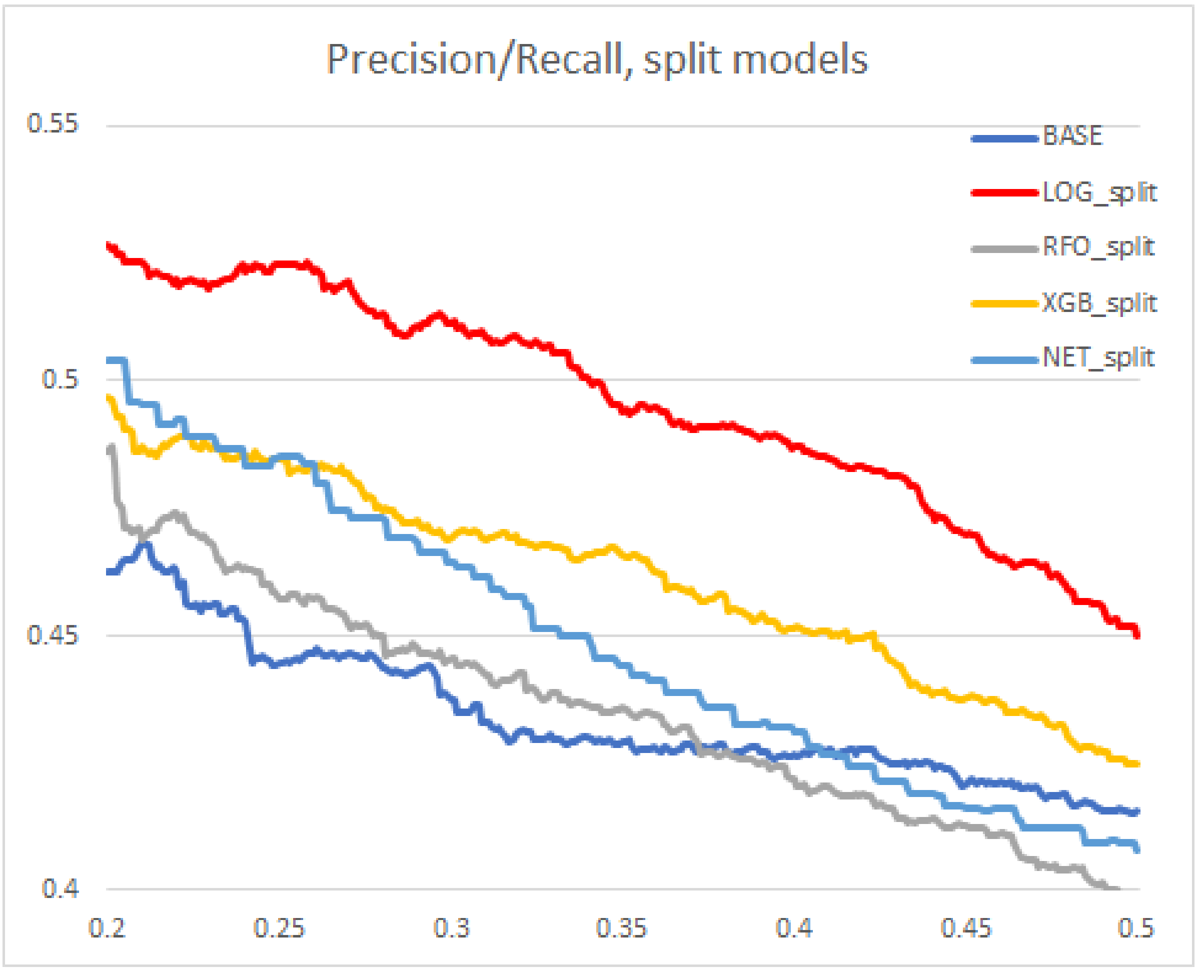

Figure 6.2.: Precision-recall curves for all models using split data

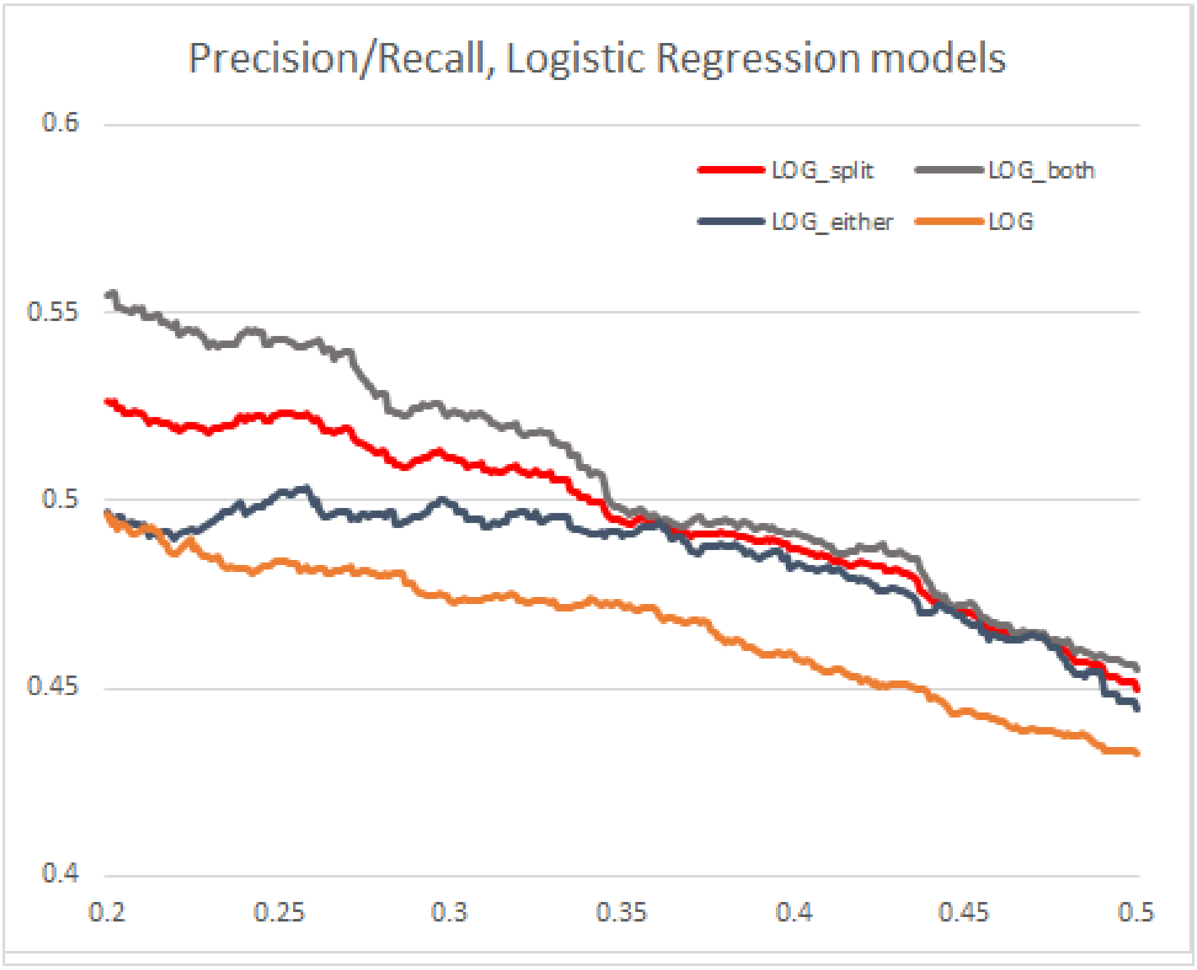

Figure 6.3.: Precision-recall curves for the logistic regression transaction models

6.2. Varying Loan Amounts

As the credit limit (how much the applicant wants to loan) was an important feature in the base model, the performance for different loan amounts was also important to evaluate. Below tables, 6.4 and 6.5, show the performance for small (0-4k SEK) and medium (5k-20k SEK) loans. There were not sufficient data to test large loans (21k-25k SEK). Still, as can be seen there is a large performance difference between small and medium loan amounts. The loan amount tests are conducted on the unsplit models due to issues of data set size.

Model	AP		AP1		AP2		PRX	
	μ	σ	μ	σ	μ	σ	μ	σ
BAS	0.348	8%	0.395	14%	0.367	13%	0.368	13%
LOG	0.395	5%	0.421	7%	0.402	9%	0.411	5%
RFO	0.372	6%	0.402	10%	0.371	8%	0.380	7%
XGB	0.388	5%	0.400	9%	0.390	6%	0.392	9%
NET	0.329	7%	0.351	10%	0.384	11%	0.306	7%

Table 6.4.: Model performances for small loans: 0-5k SEK

Model	AP		AP1		AP2		PRX	
	μ	σ	μ	σ	μ	σ	μ	σ
BAS	0.392	5%	0.415	7%	0.441	9%	0.389	5%
LOG	0.431	3%	0.459	6%	0.422	6%	0.445	4%
RFO	0.403	7%	0.433	10%	0.413	10%	0.417	8%
XGB	0.420	6%	0.442	8%	0.419	8%	0.427	6%
NET	0.421	5%	0.443	8%	0.413	8%	0.422	7%

Table 6.5.: Model performances for medium loans: 5k-20k SEK

6.3. Feature Importance

As the models use different features and feature importances, it was interesting to see both on a population and individual level how the different features of the models affect the final outcome. The evaluation metrics for the different models were used for below plots.

6.3.1. Global Importance

Figures 6.4 and 6.5 show the global feature importance for the pair of top performing models. The SHAP value is not directly connected to the predicted default probability and the actual values have limited relevance. Thicker colored regions represent a high concentration of sample values, and the red color indicates that the value for the specific sample(s) is positive following normalization. Features are shown in order of descending importance. Thus, *credit_limit_log* is the most important feature, and a high value in that category indicates an increased default frequency. Additionally, there is a large number of samples that hold the maximum value in that category. This naturally coincides with the maximum loan amount given by the company. The single largest effect on predictions by any variable and sample are in *inkasso small freq* and *no_bisnode_req* for model Both and model Either, respectively.

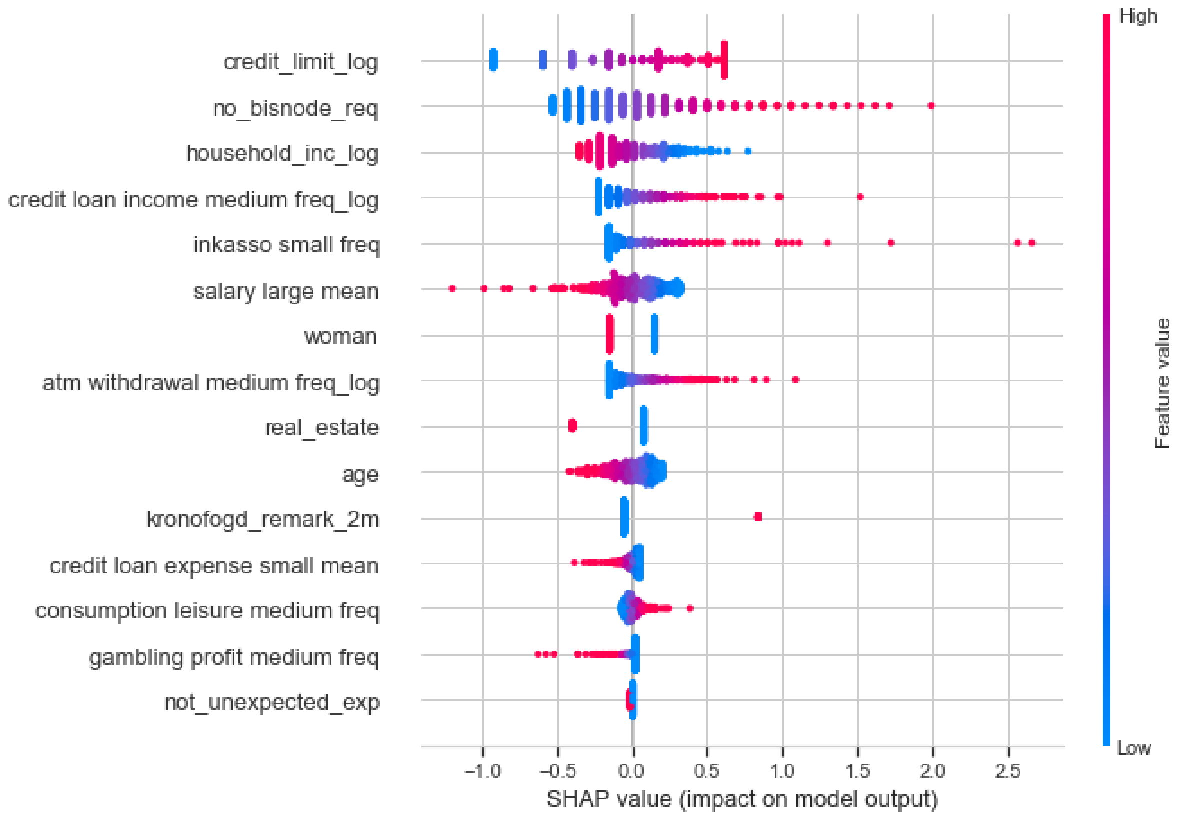

Figure 6.4.: SHAP values for features in the Both-model

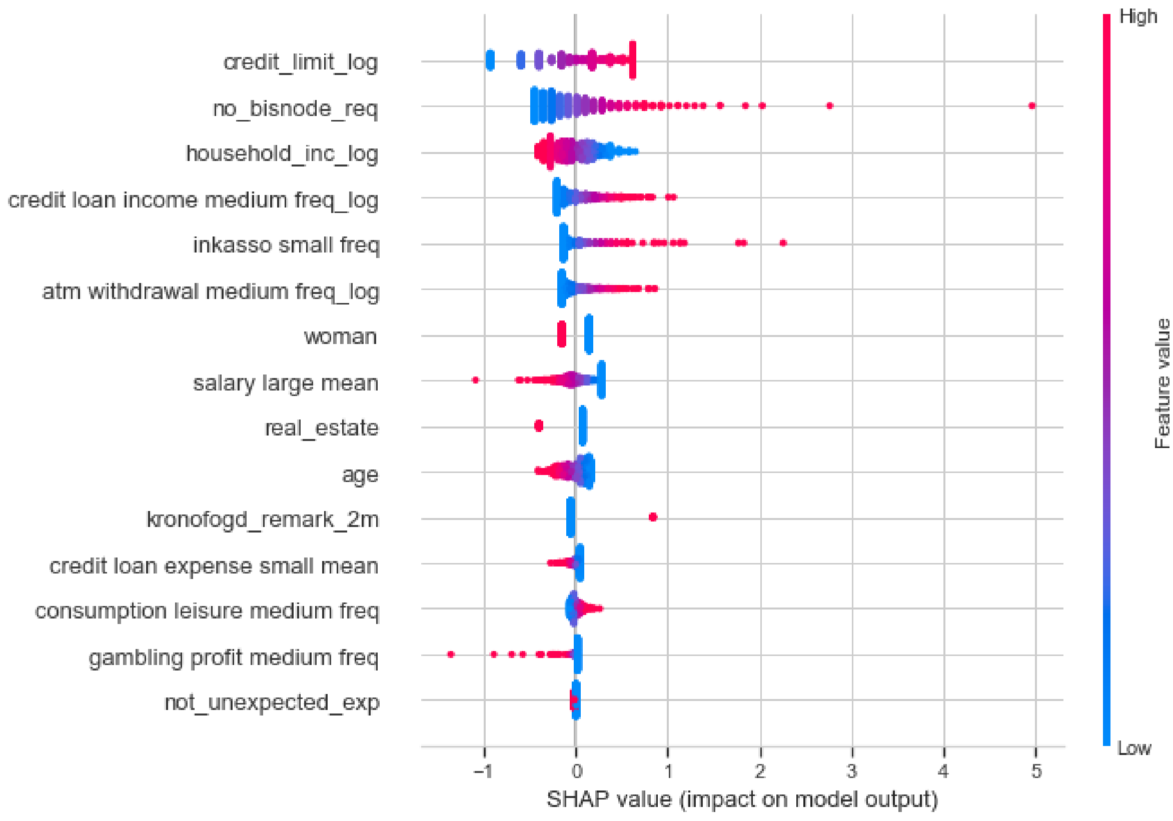

Figure 6.5.: SHAP values for features in the Either-model

<

6.3.2. Individual Importance

To evaluate how the models treat specific individuals, three types of applicants were considered:

- **The Good applicant**: a non-defaulting applicant
- **The Bad applicant**: a defaulting applicant
- **The Average applicant**: an average applicant, typically approved

Figures 6.6 - 6.8 show explainability only for model Both due to its superior predictory power. All applicants have been correctly classified and were respectively viewed by the model as obvious default, obvious non-default and borderline non-default. For spacing reasons, the names have been shortened. A high score represents a high predicted probability of default. Consequently, the red arrows show feature contributions towards a high default probability, while blue arrows do the opposite. Figure 6.6 shows an applicant with a 92% probability of default, and figure 6.7 shows an applicant with a 10% probability.

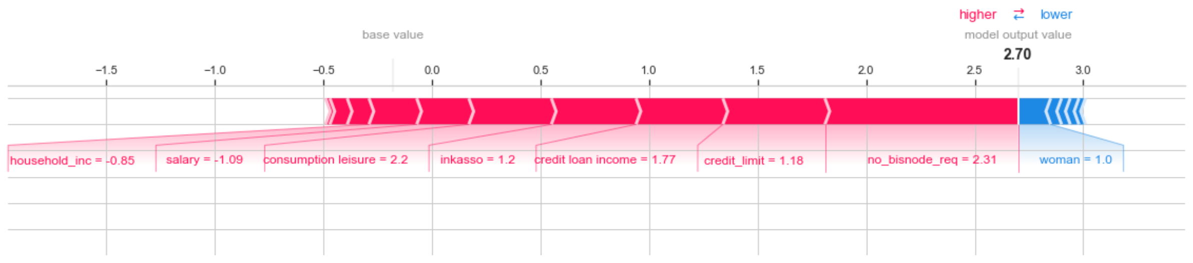

Figure 6.6.: Feature importances for obvious defaulting (92%) customer

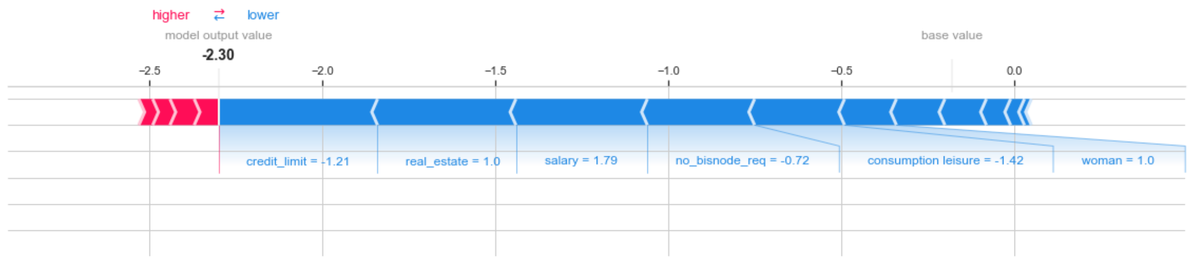

Figure 6.7.: Feature importances for obvious non-defaulting (10%) customer

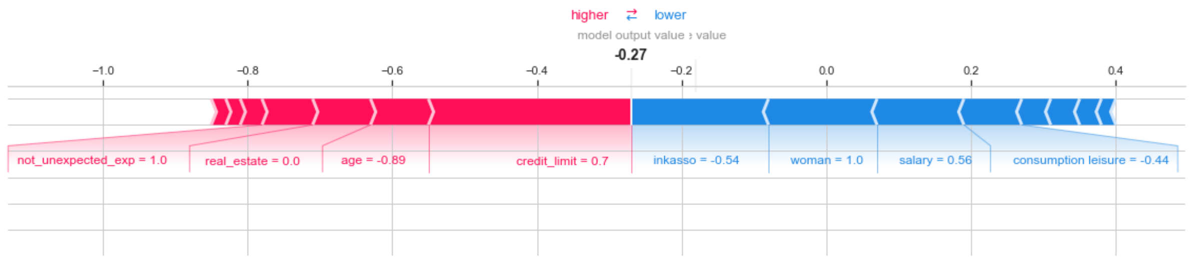

Figure 6.8.: Feature importances for borderline non-defaulting (48%) customer

6.4. Effect of SMOTE

SMOTE does not seem to significantly improve the overall average precision of the models. It does however, for a decision threshold of 0.5, shift the balance between precision and recall for defaulting customers. For example, by evaluating the precision and recall at threshold 0.5 for the Logistic Regression model, precision and recall change from 0.7 to 0.4 and 0.1 to 0.6 respectively, which is quite a significant change. SMOTE might therefore be useful depending on the objectives of the credit institute. From the authors point of view, it is definitely preferred to have a somewhat balanced model with similar precision and recall values.

Chapter 7.

Discussion

7.1. Credit Scoring

7.1.1. Final Model

The results presented above show that the logistic regression models are slightly better than the rest. In addition, it has the benefit of being simple to explain and gives clear insights to the credit institute and their handlers in how to consider the output from the model. The split-data models proved to be more stable, which can be attributed to the fact that the result is a weighted average of two models' performances.

7.1.2. Model Non-Complexity

From the process of trying out new models, many attempts were made at getting other models than a Logistic Regression to perform well. However, only in rare cases were other models, particularly Random Forests, able to perform equally well. This fact highlights that there are likely very few or no significant non-linear relationships in the data, that helps better predict default rate. For the more complex models to perform better, a change in feature construction or data set size would likely need to take place.

7.1.3. Lack of New Explanatory Features

As can be observed by the above results, models using transaction-derived features do not improve drastically upon the base model. In addition, the SHAP values in 6.4 and 6.5 show that transaction-derived features have far lower importance than the socioeconomic factors used in the base model. This could be attributed to sub-optimal feature construction or noise stemming from the transaction categorization. Alternatively, it could be the case that transaction data does not contain much information with explanatory power to begin with.

To test the possible explanations for the relatively poor performance of transaction data, one could run models with solely transaction-derived features to better gauge the explanatory power of these features.

7.1.4. Risk of Bias with Raw Data

As with all machine learning tasks, inserting personal bias and assumptions in the model processing can ruin the performance, especially for more complex processes. At the same time, when working with raw data one must use some bias to generate desired features. Thus, it is vital, and something that the authors learned the hard way, to spend a long time pre-processing the data and continuously analyzing all output to ensure that no faulty assumptions are introduced. Furthermore, one should always ask oneself if this is a reasonable assumption or if it will inject faulty bias. In the authors' opinion, 80% of the time should be spent on data analysis and 20% on model implementation and evaluation.

7.2. Transaction Categorization

Even if the rules for mapping a transaction to a certain category is clear if all information is known, it is sometimes very difficult given the limited information provided on a bank statement. This is certainly the case for payment services such as Klarna and Kivra which may be used for various purposes. As for a machine learning model, this leads to the question of precision versus recall in the different categories. If one aims to map more of the transactions in to certain categories one inevitably ends up with a lot of noise and miss-classifications. On the other hand, if one is too strict on what to include in the different categories one misses out on potentially important information.

In this report, both approaches were tested and it was concluded that it was better to have high precision in the different categories than high recall.

Chapter 8.

Parallel and Future Work

8.1. Machine Learning Categorization

An important aspect when working with big data is the possibility to automation. The credit institute also told us that having a good knowledge of how their applicants spend their money is equally important to knowing if that category is significant in a machine learning model or not. Additionally, we have only worked with a subset of all available data. At the time of writing more than 10 times the amount of data ($>$10GB) exists making it even more important to automate the process of transaction categorization. Thus, large efforts have been made in parallel to automate the categorization process by constructing metrics based on the amount, date and description similarities. Preliminary results show promise but due to time restraints and requirements for high precision in the different transaction categories no in-depth analysis was made in this thesis.

8.2. Profitability Analysis

As briefly discussed in model evaluation metrics, the question for the credit institute to accept or decline a customer is purely a financial one. Thus, future work should explore the possibility to connect the performance of a classifier to financial data metrics such as

- how much is made on a non-defaulting customer
- how much can be recovered from a defaulting customer and what is the total loss

Chapter 9.

References

[1] *About.* https://xgboost.ai/about. (Accessed on 05/05/2020).

[2] Rehan Azam, Muhammad Danish, and Suleman Akbar. "The Significance of Socioeconomic Factors on Personal Loan Decision (A Study of Consumer Banking in Local Private Banks in Pakistan)". In: *Available at SSRN 2167960* (2012).

[3] Christoph Bertsch and Carl-Johan Rosenvinge. "FinTech credit: Online lending platforms in Sweden and beyond". In: *Sveriges Riksbank Economic Review (Sweden)* 2 (2019), pp. 42–70.

[4] Laura Brodsky and Liz Oakes. "Data sharing and open banking". In: *McKinsey on Payments July* (2017).

[5] Gavin Brown. "Ensemble Learning." In: *Encyclopedia of Machine Learning* 312 (2010).

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

[7] Tianqi Chen and Carlos Guestrin. "Xgboost: A scalable tree boosting system". In: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining.* 2016, pp. 785–794.

[8] Mounaim Cortet, Tom Rijks, and Shikko Nijland. "PSD2: The digital transformation accelerator for banks". In: *Journal of Payments Strategy & Systems* 10.1 (2016), pp. 13–27.

[9] Delphine S Courvoisier et al. "Performance of logistic regression modeling: beyond the number of events per variable, the role of data structure". In: *Journal of clinical epidemiology* 64.9 (2011), pp. 993–1000.

[10] Jesse Davis and Mark Goadrich. "The relationship between Precision-Recall and ROC curves". In: *Proceedings of the 23rd international conference on Machine learning.* 2006, pp. 233–240.

[11] Stephan Dreiseitl and Lucila Ohno-Machado. “Logistic regression and artificial neural network classification models: a methodology review”. In: *Journal of biomedical informatics* 35.5-6 (2002), pp. 352–359.

[12] Sunil Erevelles, Nobuyuki Fukawa, and Linda Swayne. “Big Data consumer analytics and the transformation of marketing”. In: *Journal of Business Research* 69.2 (2016), pp. 897–904.

[13] Hutter F Feurer M. *Hyperparameter Optimization.* Vol. The Springer Series on Challenges in Machine Learning. 2019.

[14] Peter I Frazier. “A tutorial on bayesian optimization”. In: *arXiv preprint arXiv:1807.02811* (2018).

[15] Trevor Hastie Gareth James Daniela Witten and Robert Tibshirani. *An Introduction to Statistical Learning.* Springer, 2017.

[16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning.* `http://www.deeplearningbook.org`. MIT Press, 2016.

[17] David J Hand and William E Henley. “Statistical classification methods in consumer credit scoring: a review”. In: *Journal of the Royal Statistical Society: Series A (Statistics in Society)* 160.3 (1997), pp. 523–541.

[18] C. C. Holmes and B. K. Mallick. “Bayesian regression with multivariate linear splines”. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 63.1 (2001), pp. 3–17. DOI: `10.1111/1467-9868.00272`. eprint: `https://rss.onlinelibrary.wiley.com/doi/pdf/10.1111/1467-9868.00272`. URL: `https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/1467-9868.00272`.

[19] Mikella Hurley and Julius Adebayo. “Credit scoring in the era of big data”. In: *Yale JL & Tech.* 18 (2016), p. 148.

[20] Kristin N Johnson. “Examining the Use of Alternative Data in Underwriting and Credit Scoring to Expand Access to Credit”. In: *Available at SSRN 3481102* (2019).

[21] Francisco Louzada, Anderson Ara, and Guilherme B Fernandes. “Classification methods applied to credit scoring: Systematic review and overall comparison”. In: *Surveys in Operations Research and Management Science* 21.2 (2016), pp. 117–134.

[22] Scott M Lundberg and Su-In Lee. “A unified approach to interpreting model predictions”. In: *Advances in neural information processing systems.* 2017, pp. 4765–4774.

[23] Cuicui Luo, Desheng Wu, and Dexiang Wu. “A deep learning approach for credit scoring using credit default swaps”. In: *Engineering Applications of Artificial Intelligence* 65 (2017), pp. 465–470.

[24] Daniel Możdżyński. “The conceptions of new payment methods based on revised payment services directive (PSD2)”. In: *Information Systems in Management* 6 (2017).

[25] E Nazarenko, V Varkentin, and T Polyakova. “Features of Application of Machine Learning Methods for Classification of Network Traffic (Features, Advantages, Disadvantages)”. In: *2019 International Multi-Conference on Industrial Engineering and Modern Technologies (FarEastCon)*. IEEE. 2019, pp. 1–5.

[26] Didrik Nielsen. “Tree boosting with xgboost-why does xgboost win" every" machine learning competition?” MA thesis. NTNU, 2016.

[27] Fernando Nogueira. *Bayesian Optimization: Open source constrained global optimization tool for Python*. 2014. URL: `https://github.com/fmfn/BayesianOptimization`.

[28] Ceylan Onay and Elif Öztürk. “A review of credit scoring research in the age of Big Data”. In: *Journal of Financial Regulation and Compliance* (2018).

[29] *pandas - Python Data Analysis Library*. `https://pandas.pydata.org/about/`. (Accessed on 05/05/2020).

[30] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[31] Jürgen Schraten. *Credit and Debt in an Unequal Society: Establishing a Consumer Credit Market in South Africa*. Vol. 7. Berghahn Books, 2020.

[32] Bobak Shahriari et al. “Taking the human out of the loop: A review of Bayesian optimization”. In: *Proceedings of the IEEE* 104.1 (2015), pp. 148–175.

[33] Naeem Siddiqi. *Intelligent credit scoring: Building and implementing better credit risk scorecards*. John Wiley & Sons, 2017.

[34] *slundberg/shap: A game theoretic approach to explain the output of any machine learning model*. `https://github.com/slundberg/shap`. (Accessed on 05/05/2020).

[35] Wanhua Su, Yan Yuan, and Mu Zhu. “A relationship between the average precision and the area under the ROC curve”. In: *Proceedings of the 2015 International Conference on The Theory of Information Retrieval*. 2015, pp. 349–352.

[36] Robin Teigland et al. *The Rise and Development of FinTech (Open Access): Accounts of Disruption from Sweden and Beyond*. Routledge, 2018.

[37] Robert Tibshirani. “Regression shrinkage and selection via the lasso”. In: *Journal of the Royal Statistical Society: Series B (Methodological)* 58.1 (1996), pp. 267–288.

[38] Ellen Tobback and David Martens. “Retail credit scoring using fine-grained payment data”. In: *Journal of the Royal Statistical Society: Series A (Statistics in Society)* 182.4 (2019), pp. 1227–1246.

[39] Mark Van Rijmenam. *The Organisation of Tomorrow: How AI, blockchain and analytics turn your business into a data organisation*. Routledge, 2019.

[40] *Why TensorFlow*. `https://www.tensorflow.org/about`. (Accessed on 05/05/2020).

[41] Eyal Winter et al. “The shapley value”. In: *Handbook of game theory with economic applications* 3.2 (2002), pp. 2025–2054.

[42] Markos Zachariadis and Pinar Ozcan. “The API economy and digital transformation in financial services: The case of open banking”. In: (2017).

[43] Hui Zou and Trevor Hastie. “Regularization and variable selection via the elastic net”. In: *Journal of the royal statistical society: series B (statistical methodology)* 67.2 (2005), pp. 301–320.

Appendix A.

Transaction Categorization

Please note that many main categories include very large outliers for which plots are not included below. They were however included in the credit scoring analysis.

A.1. Summary

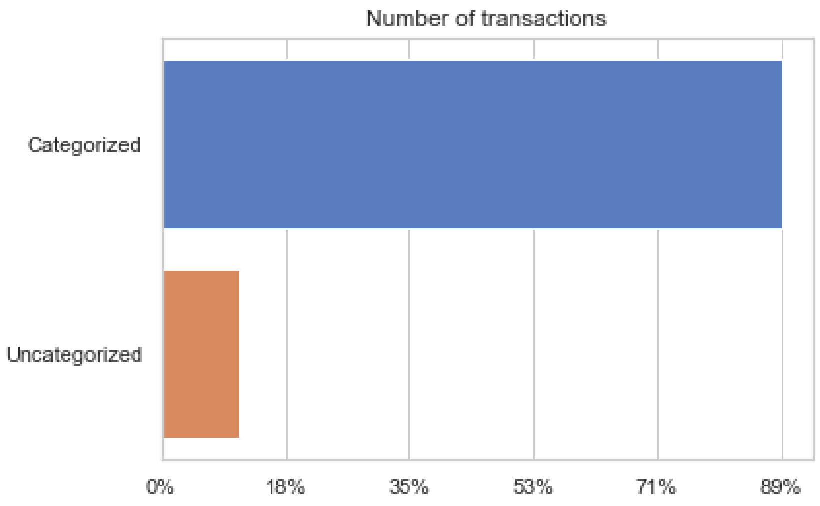

Figure A.1.: About 90% of roughly 6 million transactions were categorized

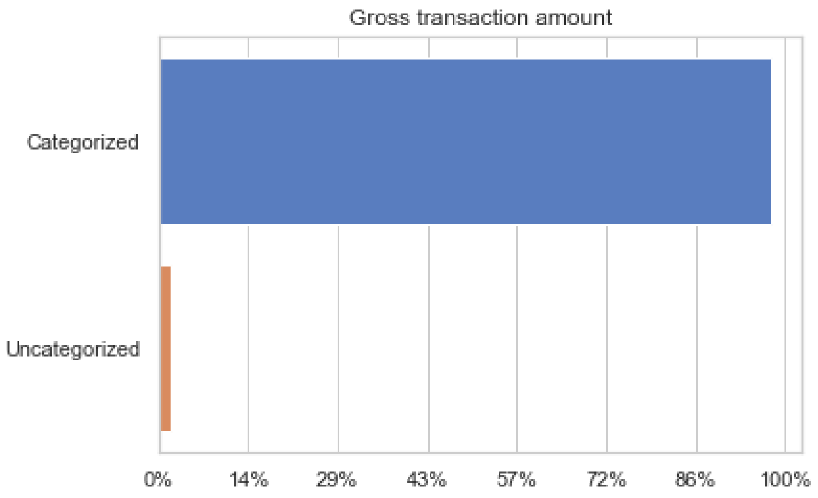

Figure A.2.: More than 99% of the total gross value was categorized

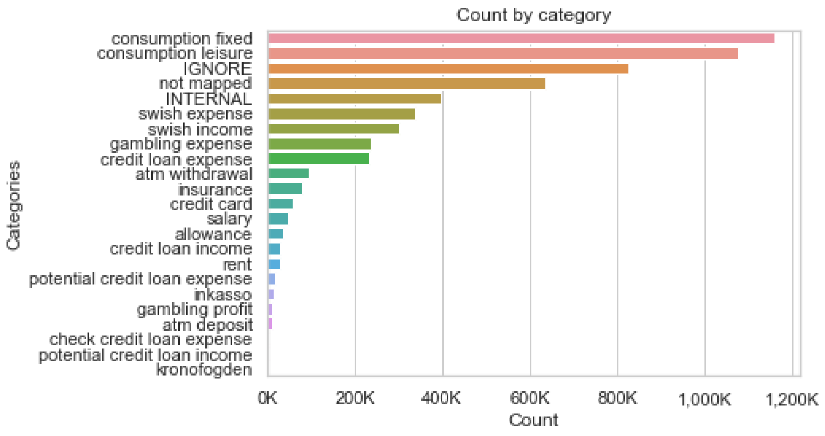

Figure A.3.: Total number of transactions per category

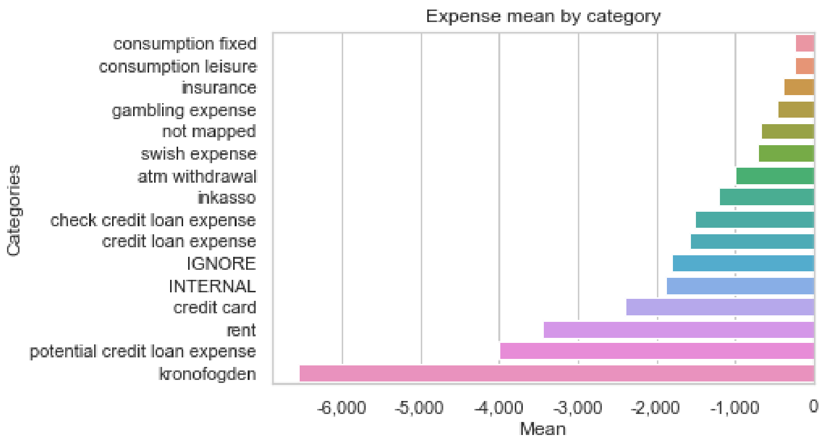

Figure A.4.: Expense *mean* per category

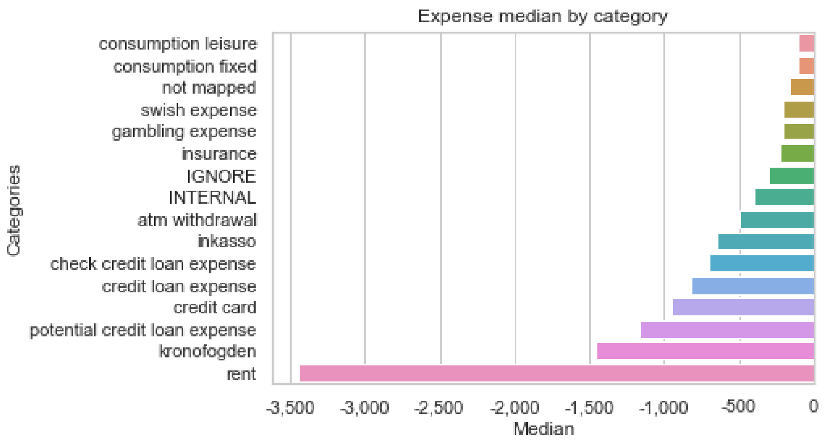

Figure A.5.: Expense *median* per category

A.2. ATM

Large ATM deposits (>10k SEK) not included). All figures show total total number of transactions in amount bins divided by corresponding description match term.

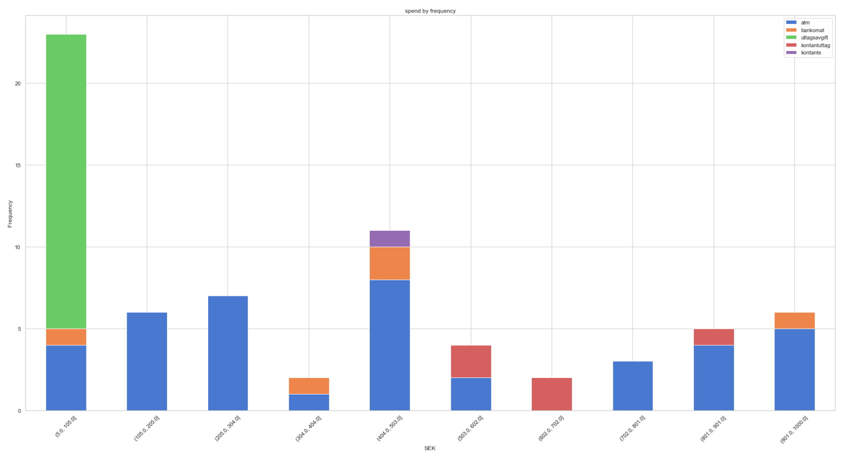

Figure A.6.: Small ATM deposits: 0-1k SEK

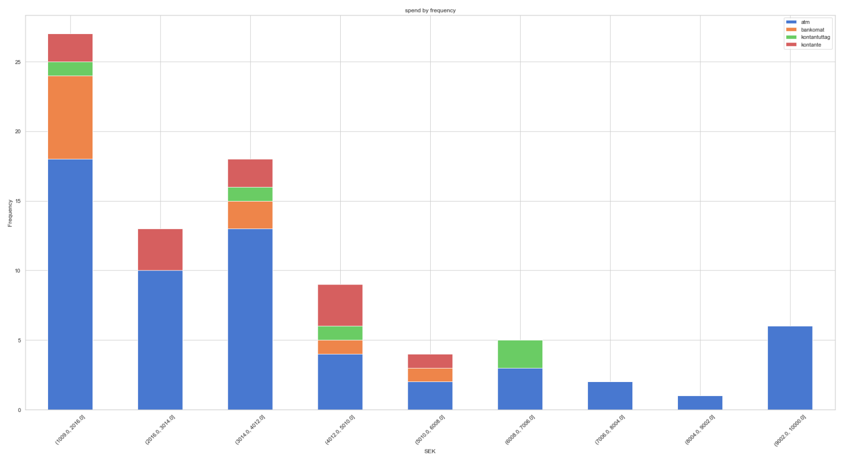

Figure A.7.: Medium ATM deposits: 1-10k SEK

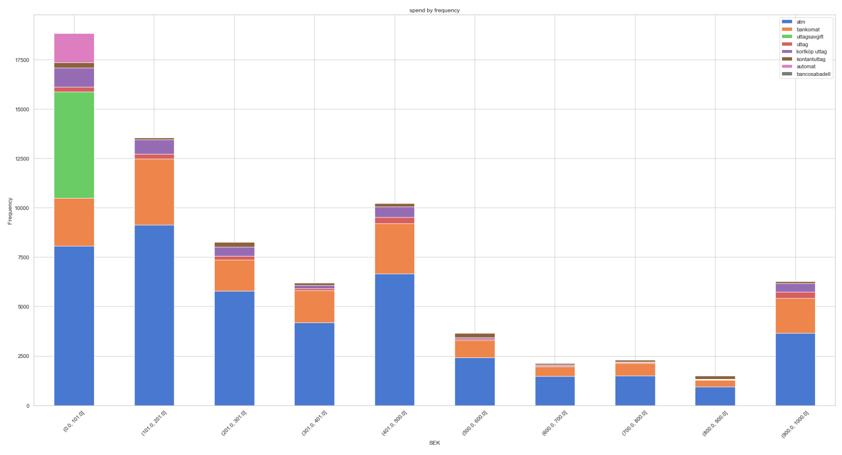

Figure A.8.: Small ATM withdrawal: 0-1k SEK

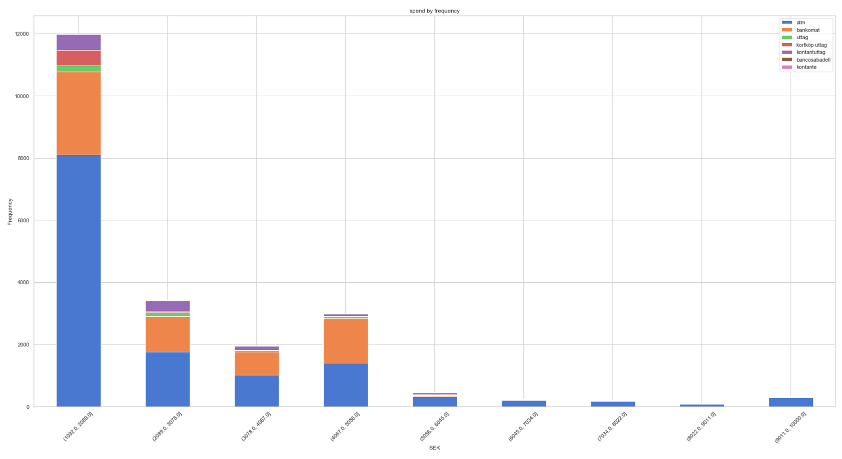

Figure A.9.: Medium ATM withdrawal: 1-10k SEK

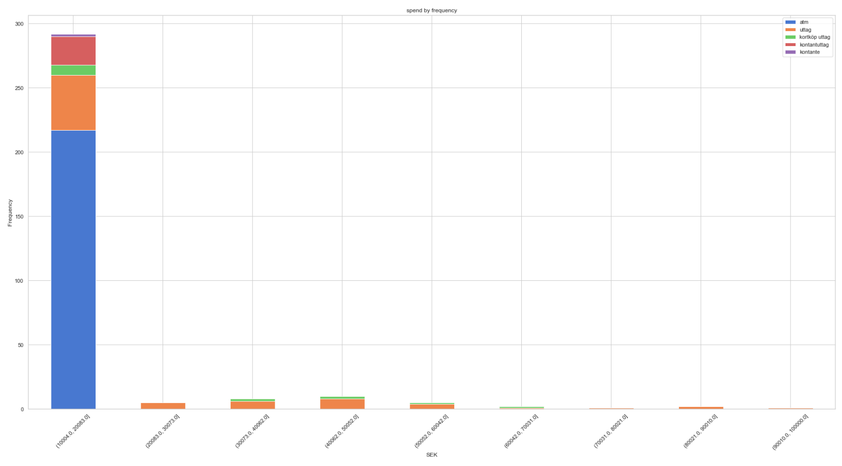

Figure A.10.: Large ATM withdrawal: >10k SEK

A.3. Bailiff

All plots show bailiff (inkasso) expense per receiving party.

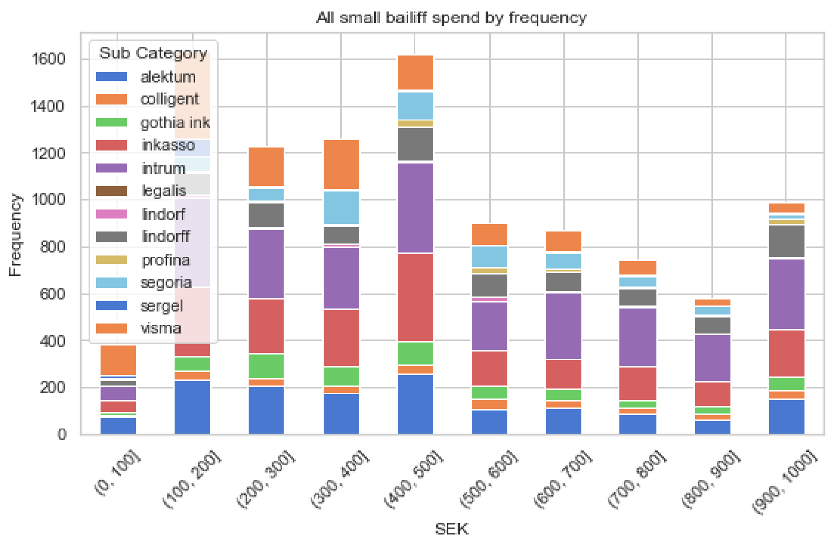

Figure A.11.: Small inkasso: 0-1k SEK

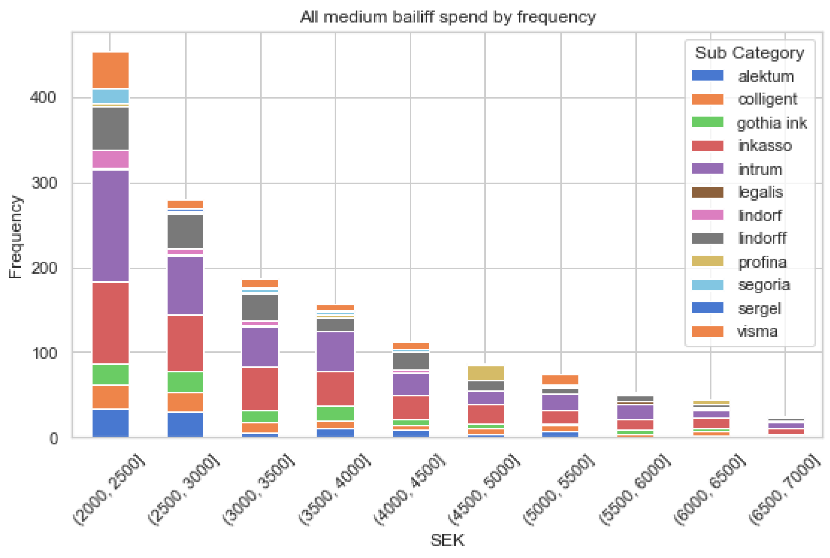

Figure A.12.: Medium inkasso: 2k-7k SEK

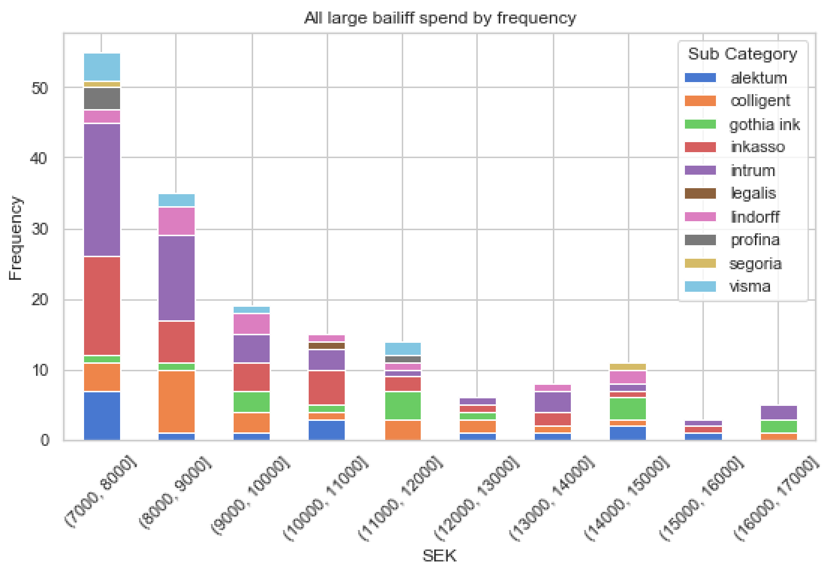

Figure A.13.: Large inkasso: 7k-17k SEK

A.4. Credit Income/Expense

Credit income/expenses include credit card payments and credit loan income/payments. Absolute amount is considered for expenses. Additional categories for which not certain mapping could be made are included for reference. These are prefixed with "potential".

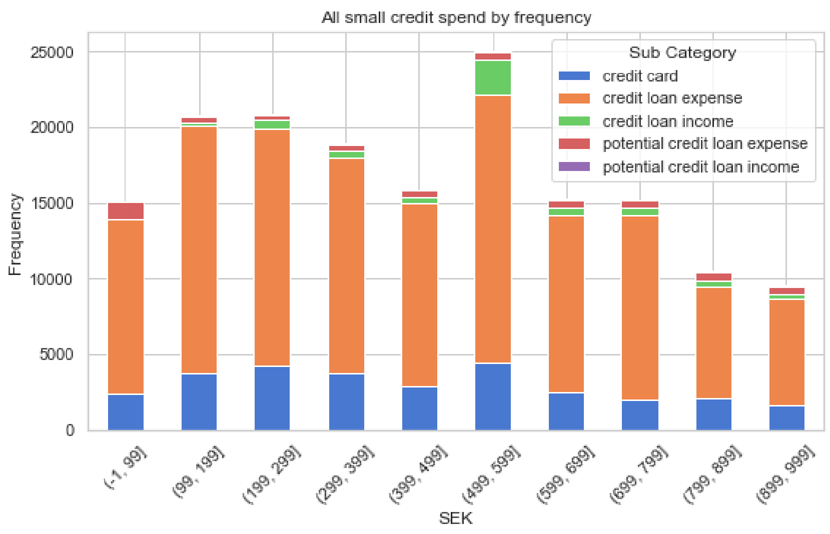

Figure A.14.: Small credit: 0-1k SEK

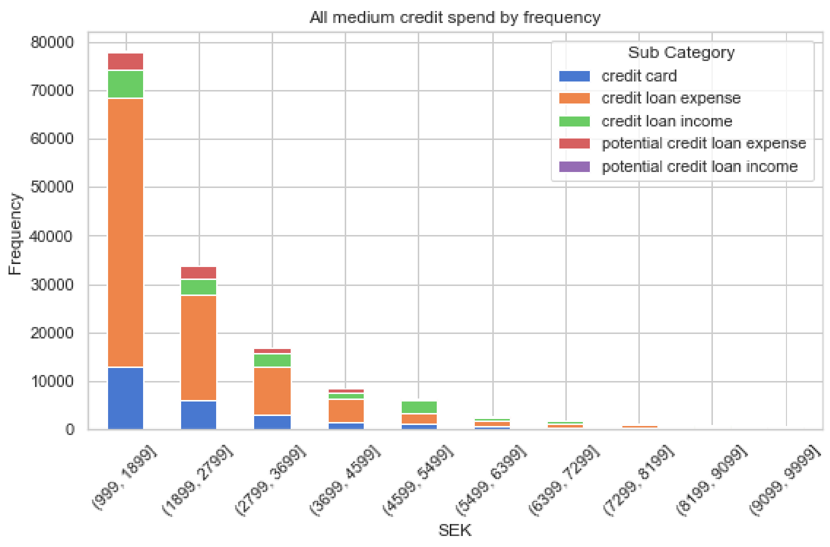

Figure A.15.: Medium credit: 1-10k SEK

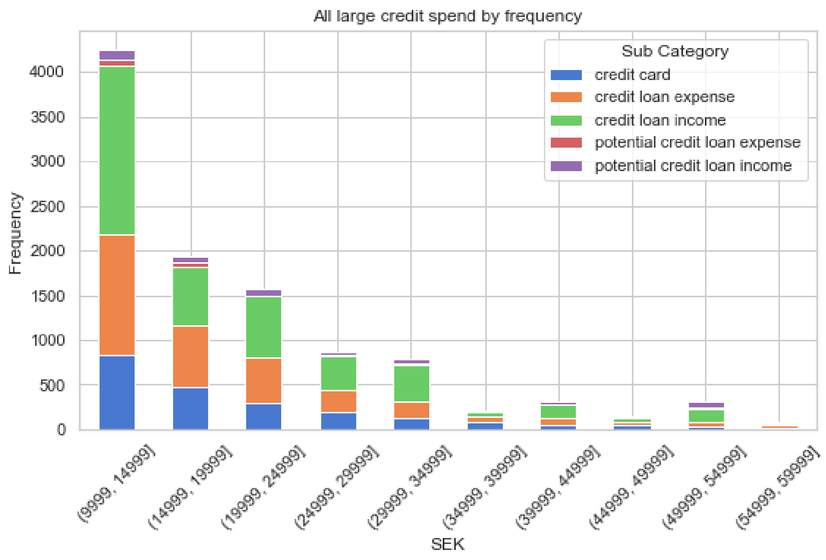

Figure A.16.: Large credit: >10k SEK

A.5. Consumption

Consumption is divided in essential consumption (fixed) and non-essential consumption (leisure). The definition was made by the authors and can roughly be determined by looking at the sub-categories in the plots below. Please note that roughly 90% of the transactions in the respective categories were categorized.

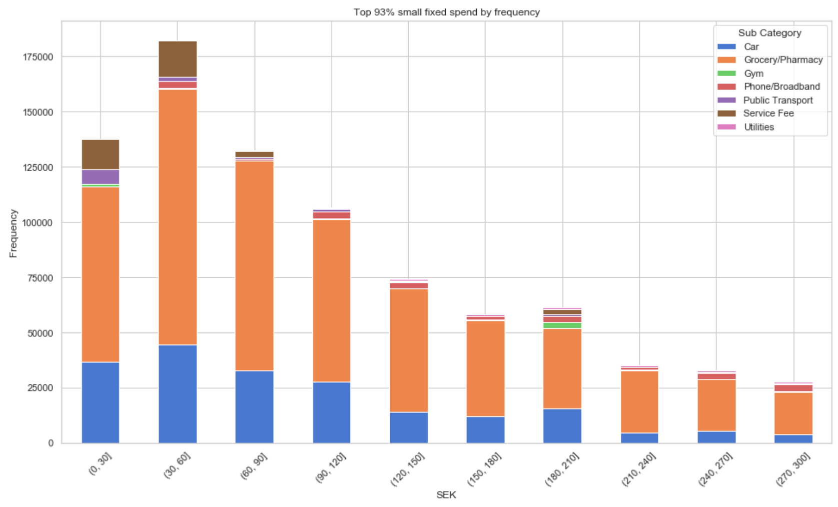

Figure A.17.: Small fixed consumption: 0-300 SEK

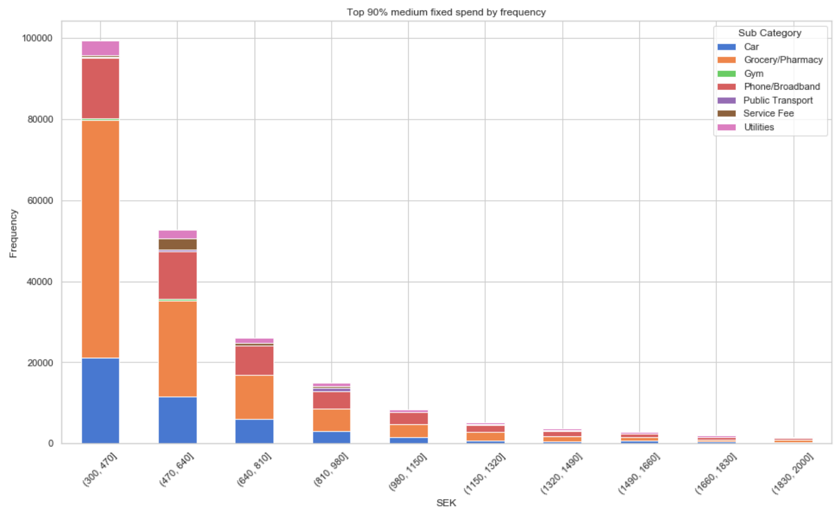

Figure A.18.: Medium fixed consumption: 300-2k SEK

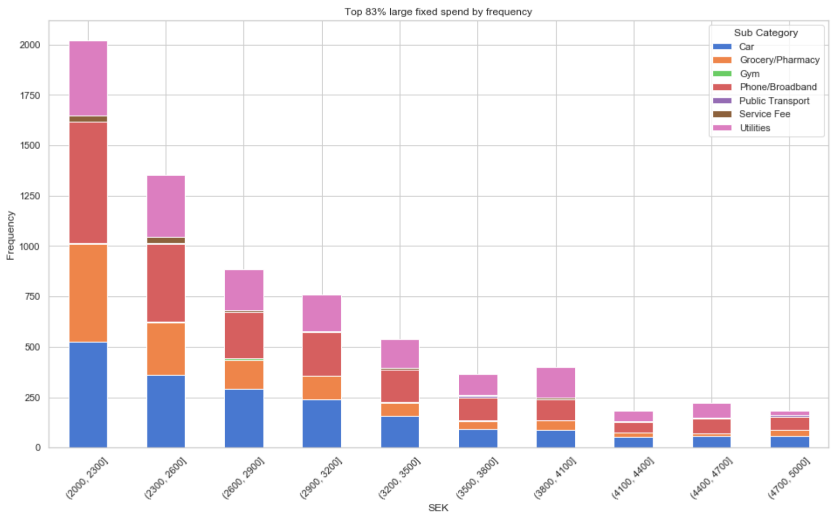

Figure A.19.: Large fixed consumption: 1k-5k SEK

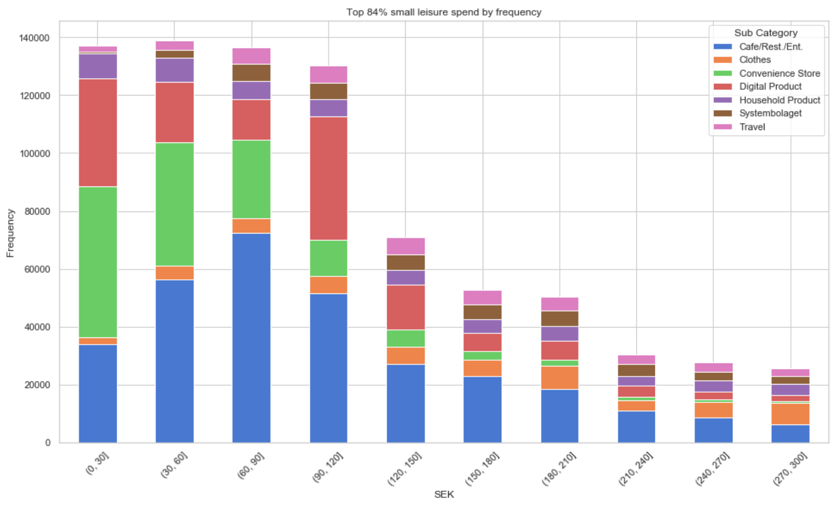

Figure A.20.: Small leisure consumption: 0-300 SEK

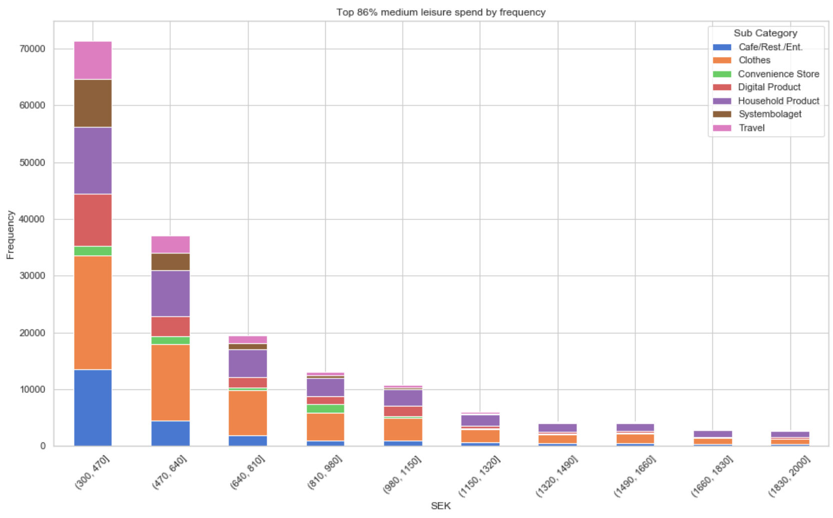

Figure A.21.: Medium leisure consumption: 300-2k SEK

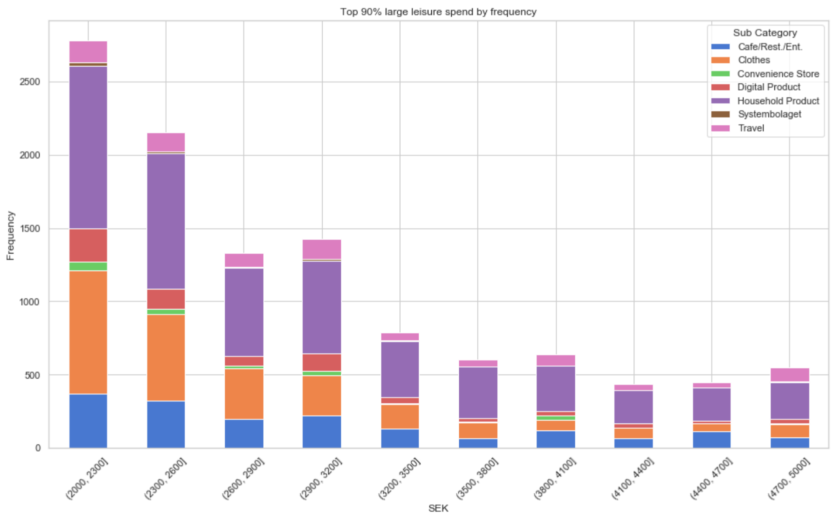

Figure A.22.: Large leisure consumption: 2k-5k SEK

A.6. Gambling expenses

Please note that no plots is provided for gambling profit even though it was analyzed and used in the credit scoring process.

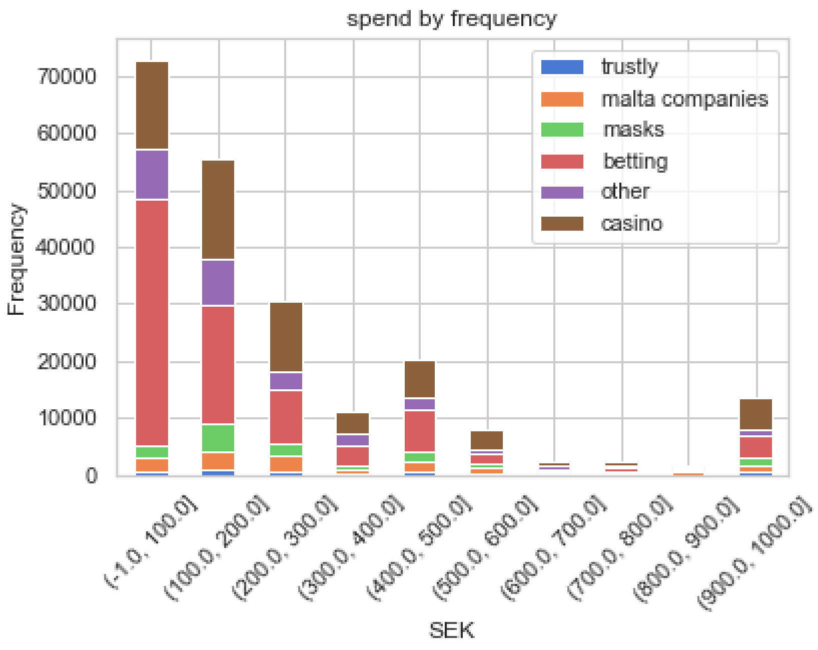

Figure A.23.: Small gambling: 0-1k SEK

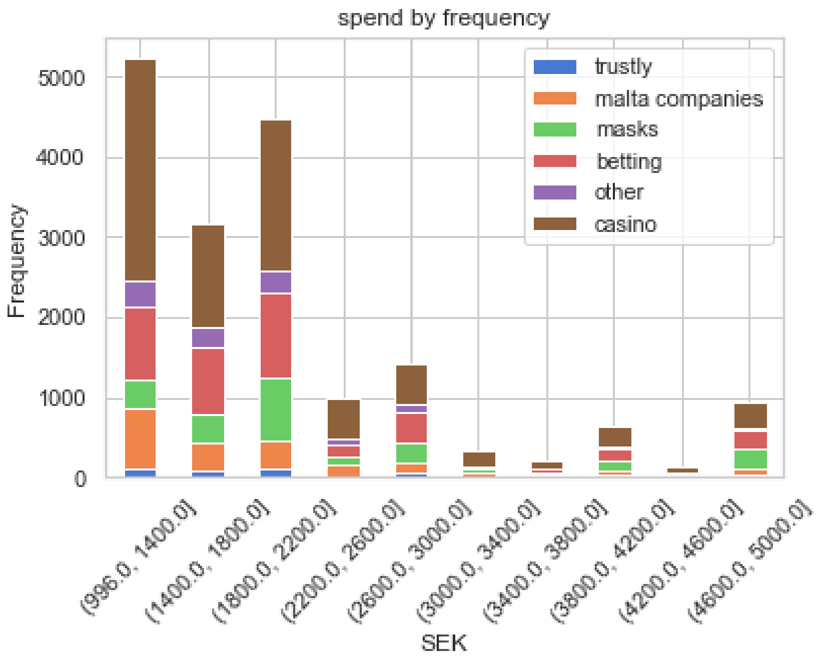

Figure A.24.: Medium gambling: 1k-5k SEK

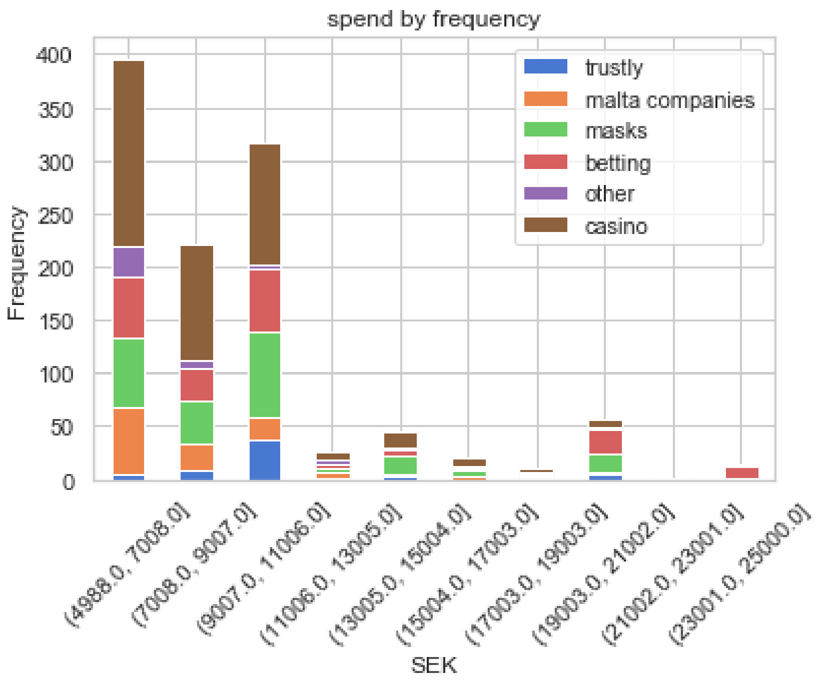

Figure A.25.: Large gambling: 5k-25k SEK

A.7. Rent

Any rent less than 1000 SEK was excluded. Outlying rent was any rent larger than 15 000 SEK.

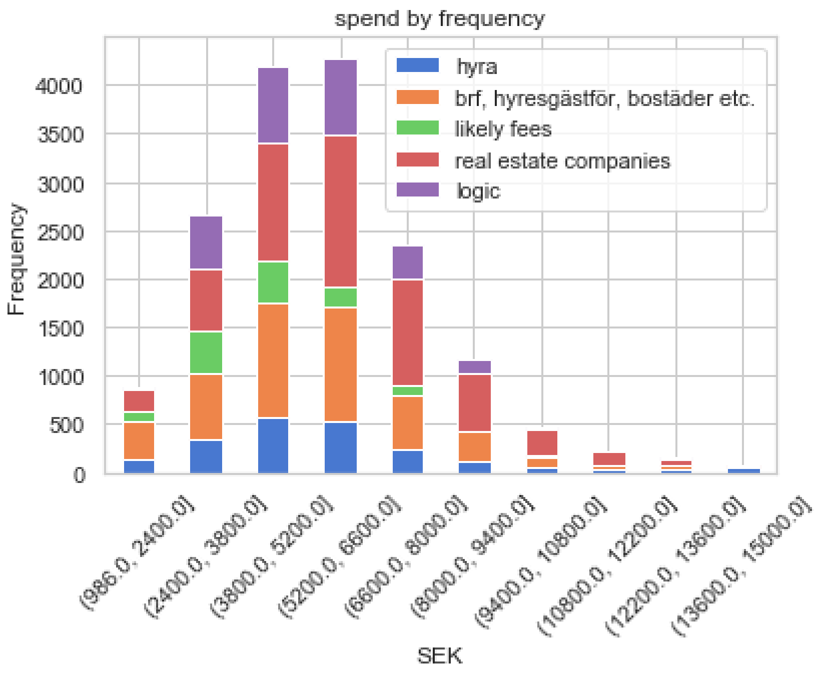

Figure A.26.: Normal rent: 1k-15k SEK

A.8. Swish

Swish is a widely used direct payment service in Sweden mostly used for person-to-person debts.

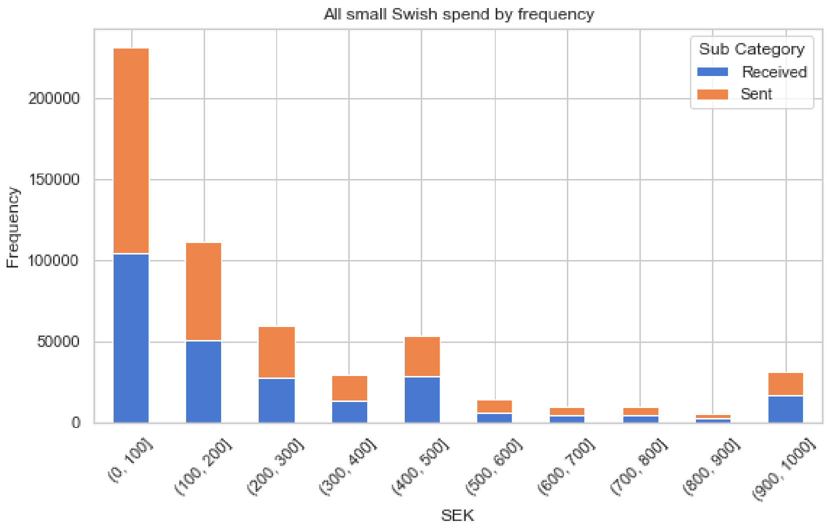

Figure A.27.: Small Swish: 0-1k SEK

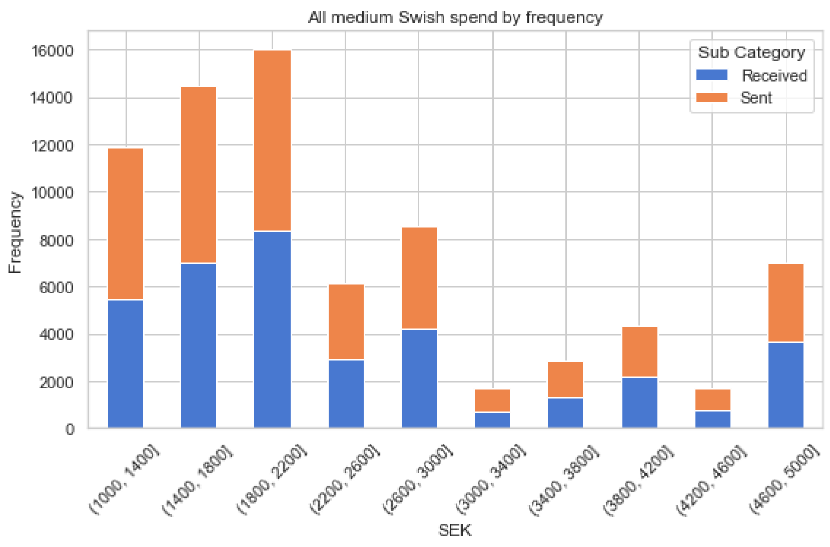

Figure A.28.: Medium Swish: 1k-5k SEK

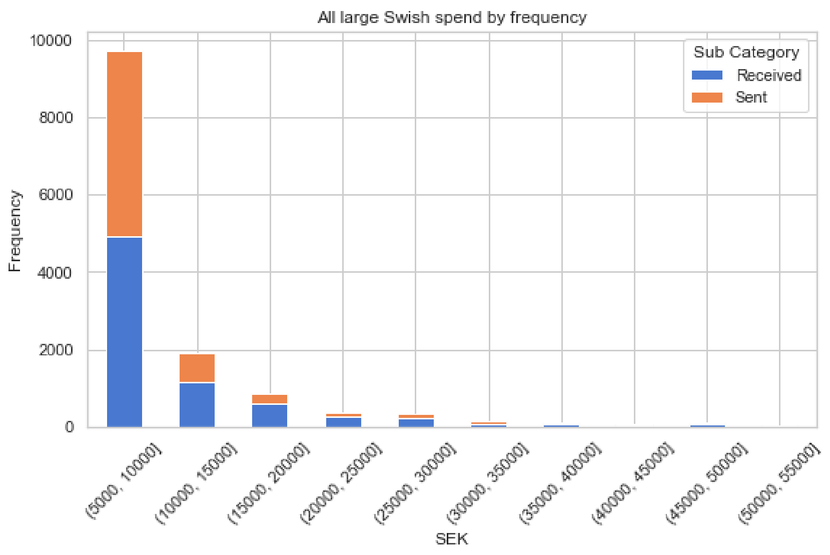

Figure A.29.: Large Swish: >5k SEK

Appendix B.

Model Hyperparameters

All base settings refer to settings that remained constant during training but that were different than the default value in the respective programming libraries.

B.1. Logistic Regression

Name	Value	Comment
max_iter	8000	Max. iterations for solver to converge
penalty	l2	L2-norm as regularizer

Table B.1.: Base settings for Logistic Regression

Name	Boundaries	Comment
C	[0.001, 10]	Inverse of regularization strength

Table B.2.: Hyperparameter for Logistic Regression

B.2. Random Forest

No specific base settings were used during training of the Random Forest model.

Name	Boundaries	Comment
max_depth	[5, 40] (I)	Max. depth of each decision tree
n_estimators	[30, 300] (I)	Number of decision trees
min_samples_split	[2, 50] (I)	Min. samples required to split an internal node
min_samples_leaf	[0.001, 0.5]	Min. samples share required in a leaf node

Table B.3.: Hyperparameters for Random Forest

B.3. XGBoost

Name	Value	Comment
learning_rate	0.01 | 0.3	Shrinks feat. weights for more conservative process
		0.3 used during hyper parameter tuning for speed
		0.01 used when trained with optimal parameters
objective	binary:logistic	For binary classification, outputs a probability

Table B.4.: Base settings for XGBoost

Name	Boundaries	Comment
n_estimators	[30, 500] (I)	Number of decision trees
max_depth	[5, 40] (I)	Max. depth of each tree
min_child_weight	[1, 6] (I)	For linear regression, instances in each node
gamma	[0, 0.5]	Min. loss reduction to make partition in leaf
subsample	[0.57, 0.9]	Subsampling of data before growing each tree
colsample_bytree	[0.75, 0.9]	Subsample ratio of feats. before growing tree
reg_alpha	[25, 200]	L1 regularization strength

Table B.5.: Hyperparameters for XGBoost

B.4. Keras Sequential (Neural Network)

Name	Value	Comment
epochs	20	number of epochs
batch_size	32	number of samples per batch

Table B.6.: Base settings for Neural Network

Name	Boundaries	Comment
dropout	[0, 0.5]	the dropout percentage for all layers
hidden_nodes_1	[0, no.features]	number of nodes for hidden layer 1
hidden_nodes_2	[0, no.features]	number of nodes for hidden layer 2, if applicable
hidden_nodes_3	[0, no.features]	number of nodes for hidden layer 3, if applicable
learning_rate	[0.001, 0.2]	the learning rate when adjusting parameter weights
l1	[0, 0.2]	The magnitude of the L1-regularization term

Table B.7.: Hyperparameters for Neural Network

Appendix C.

Additional Results

C.1. Both-Model Precision/Recall

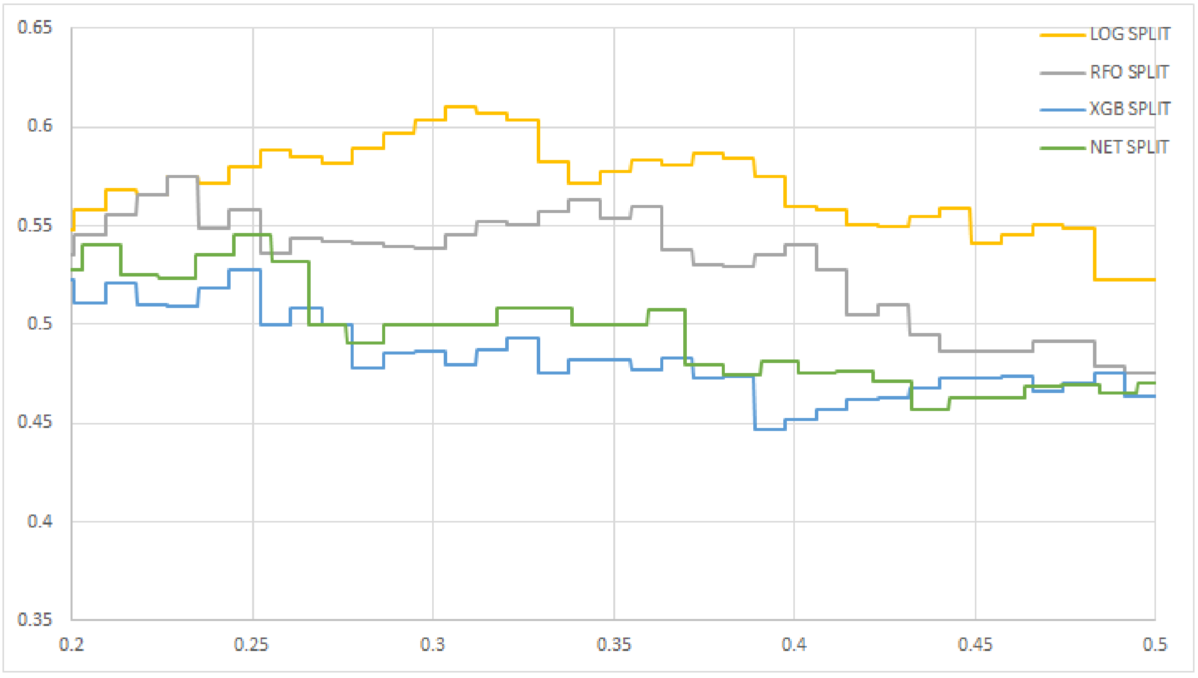

Figure C.1.: Precision-recall results for all Both-models

C.2. Either-Model Precision/Recall

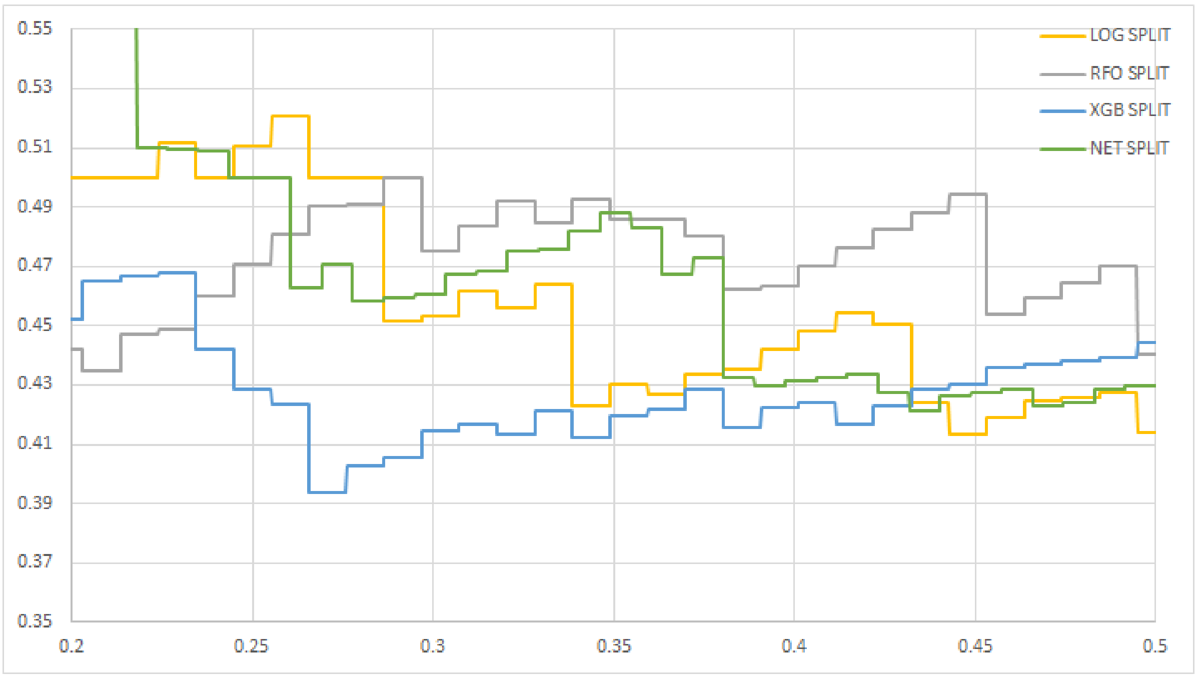

Figure C.2.: Precision-r ecall results for all Either-models