NTNU
Norwegian University of
Science and Technology

Sales prediction in online banking

Mathias Iden
Erlend S. Stenberg

Master of Science in Computer Science
Submission date: June 2018
Supervisor: Jon Atle Gulla, IDI

Norwegian University of Science and Technology
Department of Computer Science

NTNU – Trondheim
Norwegian University of
Science and Technology

Sales prediction in online banking

Mathias Iden

Erlend Stenberg

supervised by

Professor Jon Atle Gulla

Norwegian University of Science and Technology

Department of Computer Science

June 2018

Abstract

This master thesis seeks to explore how machine learning methods can be applied to predict the customers that are likely to purchase a credit card in Sparebank 1 SMN. The sales prediction problem has many similarities with customer churn prediction problems. We examine the current literature of both problems within the banking domain and adapt several techniques to our project. The experiment conducted follows an exploratory, result-driven approach with the primary goal of answering three research questions.

We develop two machine learning models from data based on the event logs from interactions with the bank's online services and from customers' personal attributes. We define two pipelines, one for each dataset. In both pipelines we evaluate multiple classification algorithms. The first pipeline is exploratory of nature as little research has been done examining how sequential event data in the form of customer timelines can be used for training a classification model. The second pipeline is based on a traditional static customer attributes dataset commonly seen in state-of-the-art research. We apply various preprocessing and data aggregation techniques to optimise the datasets for further analysis. By performing sampling and feature selection techniques we measure the effect on model performance in terms of how well the models are able to identify likely credit card purchasers while reducing the number of incorrectly predicted purchasers. After finalising each pipeline, we examine whether a combination of the models produce better results than either model in isolation. Finally, we attempt to uncover customer segments that are likely to produce high confidence predictions.

Our main findings show that the Random Forest algorithm achieves the highest performance for both datasets. The customer event timelines produced a higher performing model than the static customer attributes in terms of identifying likely credit card purchasers. The combination of the two models identifies a slightly lower amount of purchasers than either model in isolation, however greatly reduces the number of incorrectly predicted purchasers. Furthermore, by using sampling techniques to balance the proportion of purchasers to non-purchasers in the datasets, we are able to control the model's ratio between correctly and incorrectly identified purchasers.

Preface

This master thesis is a comprehensive study of various preprocessing and machine learning approaches used in the prediction of product sales from financial customer data. The study was carried out at the Norwegian University of Science and Technology during the spring of 2018. The study was created and completed at the request of Sparebank 1 SMN.

Trondheim, June 19, 2018

Mathias Iden

Erlend Stenberg

Acknowledgements

We would like to extend our thanks to our project supervisor, Jon Atle Gulla, for his counselling and guidance throughout the project. We would also like to thank Lars Ivar Hagfors, representing Sparebank 1 SMN, for providing us with large datasets at the earliest convenience and answering our questions.

Contents

Chapter 1

Introduction

Chapter one will give a brief overview of the problems at hand by presenting our research questions and background for them, describing our limitations, our experiment, and results. The final section of this chapter describes the structure of the report.

1.1 Background

SpareBank 1 SMN maintains a large amount of data on its customers. This is data with attributes such as localisation, age, and products, but also events from clicks (events) in a customer's online bank. Maintaining all of this data is demanding in itself, but extracting useful data and preparing it for training (preprocessing) is non-trivial. This is due to a simple observation; put any ten data analysts on assignment to solve the issue of preprocessing and one may be left with ten completely different approaches.

Today, most banks in Norway utilise various systems of analysis. SpareBank 1 uses machine learning techniques to identify some segments of customers, but most marketing campaigns utilise manual filtering techniques. In some cases, both of these techniques have been applied. To exemplify, a policy in SpareBank 1 has been to annually market towards customers between 18-30 years where account balance and product breadth are within some parameters. This policy aims to remind the segment of customers to fill up their BSU[1]-accounts. Other customers might receive an offer based on the customer's profile by using a machine learning model. Common to all these techniques, is that SpareBank 1 does not use datasets with customer event logs, but rather large sets of customer attributes such as product breadth, account balance, sex, age, and similar.

SpareBank 1 has addressed a wish to examine any existing trends in their datasets and if these trends can be used to predict a sale for a given customer. This is a new analysis to SpareBank 1 as it entails the usage of customer event logs to predict a sale. Determining the value of event logs is of particular interest to SpareBank 1, while methodology through pipelines and preprocessing techniques are also of interest.

The raw datasets obtained from SpareBank 1 are processed according to requirements from NSD[2]. This processing entails aggregating columns such as age into groups of ages spanning

[1]Boligsparing for ungdom (BSU) - Home savings scheme for young people

[2]Norwegian Centre for Research Data

some range, and obfuscating personalia in such a manner that reverse engineering to find an actual person is close to impossible [1]. The entire project spans a large pipeline, from raw datasets to final results.

The findings may help SpareBank 1 discover new technologies and methods to process their data and improve customer sales rates. The intended purpose of the report is as academic research through answering our research questions, but additionally as foundation for further analysis internally by SpareBank 1.

1.2 Research questions

Following are our three research questions,

1. **What preprocessing strategies or steps have the greatest impact on sales prediction from financial data?**
   We wish to attempt several different preprocessing strategies at various stages of our pipeline and compare the results. The greatest impact from preprocessing will be measured through recall and precision scores throughout the pipeline.
2. **How do standard machine learning approaches perform in financial sales prediction and what explains the differences in results?**
   We wish to experiment with several different machine learning approaches and algorithms, and compare the results. We have chosen to examine four algorithms; Naïve Bayes, Decision Trees, Random Forest, and Gradient Boosted Trees. We apply these approaches on both a static attributes dataset and a sequential events dataset with the intent of predicting the likelihood of a credit card sale.
3. **What characterises users for whom sales can be predicted with a high degree of confidence?**
   We partition the final results by the certainty of the models (confidence) into two sets; confidence equal to one and confidence lower than one. We then compare the partitions and attempt to find characteristics that lead to high confidence predictions.

1.3 Approach and limitations

Following are short descriptions of our approach and reductions to the project scope in the form of limitations.

1.3.1 Approach

The datasets used in this project are collected from real events and customer attribute data. Apart from some anonymisation required by NSD, the data is considered raw and unaltered. The actual identity of any given customer is at no point available to us. The datasets are originally not intended to be used for training machine learning classification models.

Our strategy for this project is to utilise two pipelines, each with its own premises. The final approach entails combining the two pipelines. Each of the pipelines utilise their own dataset and thus methods differ. The first pipeline uses customer event data in the form of sequential events, while the other uses customer attributes, containing columns such as age and localisation. By later combining the two models we can obtain a single model built using characteristics initially only offered in separate datasets. The different pipelines and respective approaches are described in chapter 5.

1.3.2 Limitations

There are several limitations to an assignment of this size and nature,

- **GDPR**:
  Because of the limitations from GDPR [2], it was necessary to involve NSD for (1) the formal handover of data from Sparebank 1, and (2) for verifying the masking and anonymisation of data. This can be a slow and cumbersome process potentially spanning months.
- **Data anonymisation**:
  Due to the dataset's processing and degree of obfuscation in coherence with NSD's requirements it is likely that predictions are less accurate. By receiving and training on less accurate data, the predictions will be less accurate as well. Thus, model performance is limited to some extent by the anonymisation of data.
- **Computational**:
  Due to the size of the dataset in our project we had several limitations with computational power. Most of our limitations occurred due to the amount of RAM on our machines and was overcome by renting Google servers of over 180GB of memory. However, this costs money and forced us to maintain a proper structure of server use to minimise loss. While RAM issues are not a direct limitation as problems can still be solved by writing to disk, this likely increases the required time by several orders of magnitude and is not viable.
- **Final predictions are independent of time**:
  Any prediction of a customer as likely to buy a product is not connected to the element of time. Hence, a customer predicted as a potential purchaser in our later experiments may prove due for a purchase in 1 day, 10 days, 100 days, or longer periods. Modelling time till a predicted purchase actually occurs is preferable, but adds a degree of complexity that is outside the scope of this project.

1.4 Results and conclusion

The main results of our experiment are three Models; A, B, and C. They are trained, respectively, on customer event data, customer attributes and the combined output of Models A and B. All of these models are trained to predict whether a customer is likely to buy a credit card or not. We observe that Model C achieves a better balance of precision and recall, vastly outperforming both Model A and B in terms of precision. Furthermore, through the use of under-sampling we are able to control the balance between precision and recall, which lets us maximise the evaluation metrics that more closely aligns with the business goals of SpareBank 1 SMN. The increase in recall from under-sampling the majority class is similar to the behaviour seen in previous studies on customer churn prediction. Furthermore, we observe that Random Forest outperforms less sophisticated algorithms which is also the case in several studies reviewed in this project.

Figure 1.1 contains a ROC-curve of the three models and table 1.1 contains the corresponding precision and recall scores.

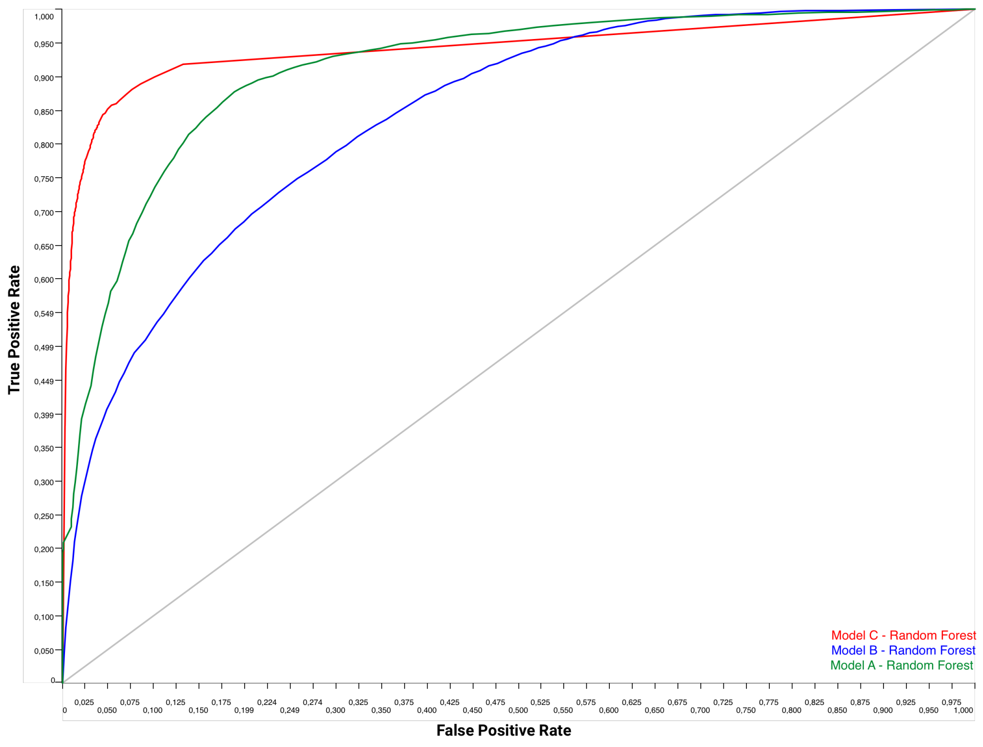


Figure 1.1: ROC-curves for purchaser predictions for the final Models A (customer events), B (customer attributes), and C (combined A and B). All the models are trained using Random Forest

Model	$P_{\text{purchaser}}$	$R_{\text{purchaser}}$	$F_{1\ \text{purchaser}}$	$P_{\text{not purchaser}}$	$R_{\text{not purchaser}}$
A	0.429	0.877	0.576	0.976	0.811
B	0.289	0.811	0.426	0.957	0.677
C	0.798	0.750	0.773	0.960	0.969

Table 1.1: Final performance for Model A, B, and C. $P_{\text{class label}}$ represents precision and $R_{\text{class label}}$ represents recall. $F_{1\ \text{class label}}$ represents the F_1 score. The `class label` represents the binary classes we predict between, either `purchaser` or `not purchaser`

1.5 Structure of the report

The report is structured as follows. The first chapter presents the project and its various aspects. The second chapter establishes a background for the project and presents the experiment. Chapter three describes various related work within the banking domain. Chapter four is a thorough description of the dataset supplied by SpareBank 1 SMN. Chapter five explains our method for the experiments and details a flow from start to finish. The sixth chapter presents the results of the experiments. Chapter seven and eight respectively provides a discussion of our findings, final conclusions with regards to the research questions, value to SpareBank 1, and recommendations for further work.

Chapter 2

Background

This chapter details existing research on machine learning techniques and lay a theoretical foundation for the entire project. Our approach consists of many techniques based on different literature. The domain covered here will not be limited to banking, but rather outline a more general approach to machine learning problems.

2.1 Data mining in the banking domain

Customer retention is becoming a difficult problem in today's market. Banks are facing more competition than ever before. Harvard business review [3] estimates that acquiring new customers is anywhere from five to 25 times as expensive as retaining existing ones. Research done by Reichheld [4] shows that increasing customer retention rates by 5% increases profits by 25% to 95%. For banks to increase profits and enhance their core competitiveness, they must be able to retain existing customer while acquiring new ones.

As data storage and computing power continues to increase in capacity yet still have a relatively low cost, the accumulation of raw data grows. However, raw data does not in itself provide meaningful information. Data mining can be used to discover patterns and relationships in the data which may provide the business with a better basis for their decisions. Specific uses of data mining techniques in the banking domain may include:

- Customer churn prediction
- Market segmentation
- Fraud detection
- Trend analysis
- Credit analysis
- Sales prediction

Sales prediction means analysing data and finding a trend among users who previously have made purchases, and applying that trend to new users. Thus, one can predict if a customer is likely to purchase a product based on information mined from existing records of sales. Conceptually this is highly beneficial, more so for a bank due to products' longevity and core customer relations normally spanning years.

A successful prediction may lead to a customer feeling well taken care of or that the establishment understands its customers. In either case, a customer being contacted with an enticing offer for a product they already seek to purchase is likely positive for the customer relation.

Most sales prediction problems are prevalent in industries that offer products to its users, such as online stores. However, predicting a sale from historical data has similarities with other machine learning problems.

The concept of predicting two outcomes is implemented in the machine learning domain as binary classification. Activities such as fraud detection, customer churn prediction and sales prediction are all examples of binary classification. While the business goals of these activities differ, the implementations are mostly the same. Thus, research on customer churn prediction is highly relevant to a sales prediction problem. Within the machine learning domain we observe far more research on customer churn prediction than sales prediction.

2.2 Classification

The process of identifying which class in a set of classes some arbitrary, new observation belongs to, is called classification. This process is commonly based on training a model on a dataset, often called a training dataset. Such datasets consist of several features, but only one *target label* indicating class membership. In our project all classifications will be binary. Hence, the target label will always either show class membership `1` or not class membership `0`. Conceptually this is similar to attempting to classify an arbitrary, new email received as either spam `1` or not spam `0`.

The key characteristic that distinguishes classification from regression is that the class label is categorical. The definition of classification is the task of learning a target function f that maps each feature set X to one of the predefined class labels Y [5]. This target function f is commonly known as the classification *model*. Several techniques exist to systematically construct such models. Some of them are:

- Naïve Bayes
- Decision Trees
- Random Forest
- Gradient Boosted Trees

Each model attempts to tie together a set of features with a target label. The model that is generated should be able to correctly predict the class label on new, unseen data. These techniques are very different and their performance tends to vary according to the nature of the dataset.

2.3 Supervised vs. unsupervised learning

There are two main categories of machine learning. Supervised learning entails using a training dataset of features and target label to create an approximation function between the two. This function should be able to predict and categorise new data into a category of the target label set (in binary classification the set contains only 0 and 1). At the core of this method is the training data that serves as historical mappings between a set of features, x and a correct label

y. This can be further categorised into classification and regression, where the target label y will be either categorical (e.g. $y \in \{0, 1\}$) or continuous ($y \in \mathbb{R}$). Further examples of each of these methods can be, respectively, classifying mail as spam or not spam, and attempting to predict housing prices.

Unsupervised learning also utilises a training dataset, but the dataset does not contain a mapping of a set of features x and target label y. This means that the dataset has no information on what a set of features point to. To further illustrate using a previous example, the dataset only shows a bunch of emails, but we do not know if they are spam or not as there exists no target label y. Unsupervised learning is generally split into association and clustering [5]. The latter is a technique commonly used in data mining to determine the inherent partitioning patterns of data with no target label. By then using the clusters (groupings) as the target label (category) one can use supervised learning on a dataset that otherwise would have been unsuited.

A third type of learning which is fairly new, is called reinforcement learning. Reinforcement learning is about sequentially rewarding actions performed in a defined environment with the goal of learning the optimal sequence of actions leading to some goal. One could argue it is its own category as it possesses traits from both aforementioned types of learning. Reinforcement learning is part of the technology utilised in Google's AlphaZero, a ground-breaking chess engine smashing previous engines such as Stockfish [6] [7]. In this example the board and pieces act as the environment, constrained by the rules of chess. The objective function accumulates the reward scores of a set of actions, while the reward score at any time step is a combination of the value of remaining pieces and their positions. [8] [9]

2.4 Method

The following sections provide background for a common pipeline for developing prediction models by using supervised machine learning. Each section will describe state-of-the-art concepts that are relevant for the project. Methods used in the experiment are described in chapter 5.

2.5 Data preprocessing

Quite often companies are in possession of large datasets that were collected for purposes other than data mining specifically. Consequently, the quality of the dataset may be poor or even insufficient for common data mining techniques. The task of detecting and correcting poor or insufficient data is called data preprocessing. Some examples of data quality issues are,

- Outliers
- Missing values
- Duplicates
- Inconsistent values

Within the banking domain datasets are generally not subject to human error. This is due to the strict policies set by both banks and governing agencies of the integrity of data. The quality of data is likely to have been slowly improved throughout the years, meaning that data from years ago is likely of worse quality than data from this year. Research has shown that more data is not necessarily key for better results, however starting with a wider dataset is always an advantage [10].

Missing values in these datasets are however very likely. Through the recent years banks have been experiencing an increase in regulations from the government that require them to keep track of more data, or in contrast, remove data due to privacy policies such as GDPR [2]. This in turn may lead to large gaps of missing data.

What follows is a brief description of each of the widely used techniques for correcting poor or insufficient data quality. These techniques heavily depend on the purpose of the data mining application. Consider a binary classification problem with a close-to uniform distribution of the target labels in the training dataset. Such a distribution means that preprocessing steps like removing outliers is natural, as one would want to fit the majority of the data as precise as possible. Any outliers would contaminate the fitting process.

2.5.1 Outliers

Outliers are pieces of data that to some degree do not resemble the characteristics of the majority of the dataset. Commonly extreme outliers considered noise should be removed, but the degree to which outliers are removed can and should be tweaked. This is due to legitimate occurrences of extreme outliers do occur, but may not be of interest. In fact, in cases of fraud or anomaly detection problems, these outliers are exactly what one would want to detect.

2.5.2 Missing values

In real world datasets, data is often full of flaws. In particular, when the size grows, errors commonly follow. Missing values may be purposefully added, or due to information not available at the time, or simply omitted by mistake. Regardless it is very important to take these missing values into consideration during preprocessing.

Tan et al. [5] provides several strategies to tackle missing values during preprocessing. It is important to consider the purpose of the application when picking strategies.

- **Eliminate data objects or attributes**:
  Remove the data objects that contain some missing values. The drawback here is that the data object may contain other values that are useful to later analysis. If the value of a specific attribute is missing across many or the majority of the data objects, one can consider removing the entire attribute from all the objects. This should be done carefully as the eliminated attribute may be impactful to the analysis. This measure is to be considered somewhat extreme, but may be appropriate under the correct circumstances.
- **Estimate missing values**:
  In some cases, missing data fields can be reliably estimated. The goal is reduce the bias that would have been introduced in the model if the value was left unchanged. If the attribute is continuous one can use the average value of the nearest neighbours. If the value is categorical, then the mode of the values for this attribute can be used.

2.5.3 Duplicates

Tan et al. [5] divides duplicate data in two categories:

1. Two or more equal objects that represent a single object
2. Two or more equal objects that represent different objects

In category 1 one should remove the duplicates to prevent the data mining algorithm from fitting the model incorrectly. In category 2 one should leave the duplicated objects as is because they represent a legitimate case of occurrences. An example of duplicate data under category 1 can be two rows that represent the same entity by id, such as two customer attribute rows with the same id and values. While in category 2, the two equal rows represents two different customers.

2.5.4 Inconsistent values

On some occasions, there may be cases where a data object contains values that directly contradicts another object, or simply does not make sense. Take for instance a data object that contains a city in a country, where the city does not exist in the country. Another case of inconsistency may be negative values in columns such as `age`, `weight` and `height`. Limitations such as the latter need to be checked, and sometimes this may require a human to assess. In other cases the inconsistencies may be automatically detected and resolved.

2.5.5 Aggregation

The aggregation of data is also a commonly used data preprocessing technique. Aggregation of data usually means grouping together a set of objects into a single object. A quick example of aggregation may be found if assuming a dataset of names with corresponding ages. If the ages are grouped into two categories, namely 0-49 and 50-100, we can effectively aggregate the dataset into such groupings. This enables us to count frequencies of names per group and similar. The age grouping could also be added as an additional column, useful for later visualisation and manipulation.

In a banking domain it would be natural to assume the existence of datasets of transactions or event logs. Such datasets may be grouped on various categories, drastically reducing the amount of records that the classification algorithm needs to consider, effectively causing less data of a higher quality to be evaluated. A common challenge is *how* to aggregate data and to what degree of granularity. While there are trade-offs in every aggregation, it is important to consider the goals of the application.

2.6 Feature selection

One of the techniques to reduce dimensionality of datasets is known as feature selection. Some of the methods within this technique are often categorised as more of an art than science, as they require good intuition and domain knowledge. A more systematic approach will likely yield better results. Tan et al. [5] describes three standard approaches to feature selection,

- **Filter approaches**: Features are selected before the data mining algorithm itself is chosen.

- **Wrapper**: Uses the target data mining algorithm as a black box to find the best subset of attributes, but without enumerating all the possible subsets of features.
- **Embedded approaches**: The data mining algorithm itself decides which feature to use and which to remove.

A common issue that arises when attempting feature selection on large datasets is *the curse of dimensionality* [11]. A very large set of features may lead to a sparse data matrix and the classification algorithm may not have enough data to build a sufficiently accurate model. The same applies when having too few features in a set. By removing redundant and highly correlated features, one can assure that only the most important features are left. Features such as `id` or auto-generated identifiers are not a natural part of the data and should be removed.

Tan et al. [5] further describes feature selection as the search for an optimal feature subset over all possible subsets of features. The main concern to this approach is to (1) limit the computational cost while (2) searching for the optimal or near-optimal set of features. The problem of generating a powerset (all possible subsets to a set) of features, has the computational cost of 2^n where n is the number of features in the set. Satisfying both (1) and (2) is generally not possible, hence trade-offs are required. In order to find the optimal subset of features in a systematic approach, Tan et al. [5] describes a four-part architecture,

1. A measure for evaluating a feature subset
2. A search strategy that controls generation of feature subsets
3. A stopping criterion
4. A validation procedure

The first item listed of the architecture is complicated because in order to evaluate a subset, the easiest way is to run it through the algorithm we wish to use after having completed feature selection. This paradox is solved by checking baseline scores, such as accuracy or F-measure, from a classification algorithm after one and only one change in a feature. This can be done in several ways, for instance, forward feature selection and backward feature elimination. These methods construct the optimal features by respectively adding features to an empty set if scores go up on addition, or removing from a full set of features if scores go up on removal.

2.7 Sampling

The concept of splitting the dataset into partitions used for training and evaluation may be implemented using several different techniques. Each technique's effectiveness depends on the exact characteristics of the dataset. The most common techniques are,

- Hold-out:
  - From the top
  - Random sampling
  - Stratified sampling
- Cross validation

2.7.1 Hold-out method

The hold-out method entails splitting a dataset into two partitions where one is considered a training set, and the other is considered an evaluation set. This is important as it allows model performance evaluation using data not seen during training. If one were to evaluate a model using the training set, it would not reflect the model's ability to predict unseen data. By using the hold-out method one can effectively "hold out" part of the data as an evaluation set, and thus ensure that testing is done on new, unseen data. A drawback for this approach is that not all available data is used for training the model.

There are several strategies that are commonly applied when using the hold-out method.

Random sampling

Random sampling means extracting the data within each partition in a completely random manner. If one were to split the partitions at 80% for the training set and 20% for the testing set, both sets would consist of rows selected in an arbitrary manner from the original dataset. This approach is useful when the dataset is sorted. By not using random sampling on a sorted dataset one could risk creating partitions that do not represent the original dataset's distribution, which in turn may lead to inaccurate results at a later stage. Random sampling aims to solve this by ensuring that each row in the original dataset has the same probability of being chosen for a partition.

Stratified sampling

Stratified sampling ensures that each partition has the exact same distribution of the target label as the original dataset. To illustrate, lets create an arbitrary dataset consisting of 5 unique IDs and corresponding binary target label as shown.

ID	TARGET LABEL
1	0
2	0
3	0
4	1
5	0

This dataset has a class distribution of 20% True versus 80% False. A stratified sample of such a dataset would ensure that each partition kept the same class distribution to the greatest extent possible. This is not possible in the example illustrated above as it is too small.

A drawback to this approach is that *any* class distribution is kept. Hence a distribution of 99.9% True versus 0.1% False will be kept as is. This is called class imbalance, further described in section 2.7.2.

Cross validation

Cross validation is a technique aiming to remove the chance at randomly obtaining a too good training or testing set. This technique splits the dataset into k partitions where each partition is used once as the test set, and $k-1$ times as part of the training set. This is called `k`-fold cross validation, or a commonly used version of it, 10-fold cross-validation.

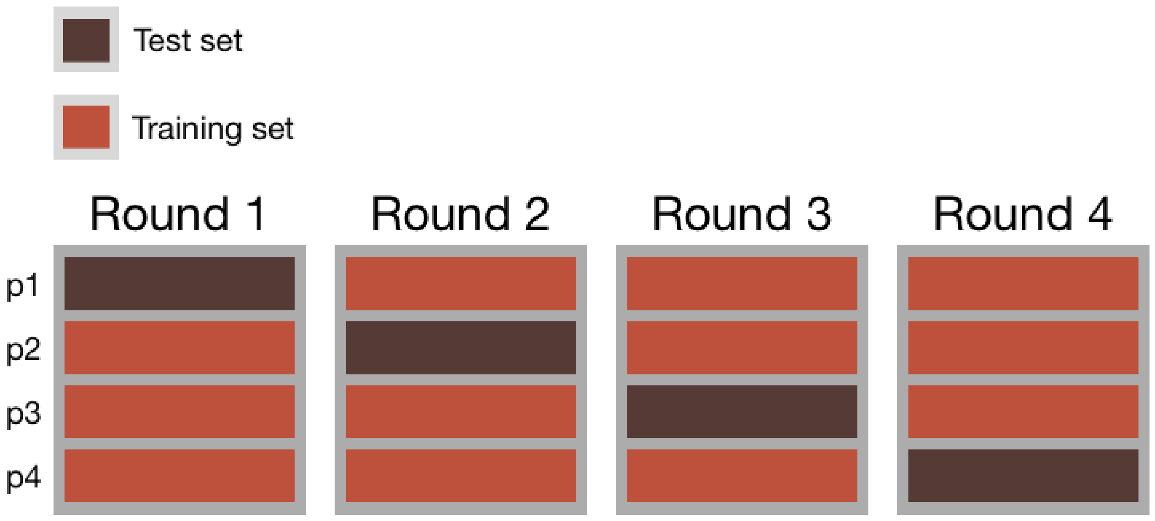


Figure 2.1: Each iteration of a 4-fold cross-validation

Figure 2.1 shows each iteration of a cross validation procedure where $k = 4$. Each partition remains the same in terms of data, meaning that no data is changed or partitioned again. In the first iteration, `p2`, `p3` and `p4` together form one *training set*, whilst `p1` is the *testing set*. For each iteration, the test set is swapped around ensuring that every partition is used both as part of the training set and as the testing set. This method is highly recommended as it utilises more of the information available in the original dataset and thereby improves the quality of the classification model [12].

2.7.2 Class imbalance

Lets assume a dataset similar to the one shown in section 2.7.1, but expanded to 10 occurrences with a 90%/10% True/False distribution. This large difference in class distribution is referred to as class imbalance [13]. Problems such as these are often found in real-world cases such as customer churn [14] and fraud detection [15].

Some datasets are innately more subjected to extremely unbalanced class distributions than others. In cases where fraud detection through machine learning is attempted, one might expect datasets with close to 99.99%/0.01% distributions.

Two sampling-based techniques are commonly used to counter class imbalance issues; under- and over-sampling (SMOTE) [13].

Synthetic minority over-sampling technique (SMOTE)

This sampling technique is a method of over-sampling where synthetic instances of the minority class are created and added to the dataset. The synthetic objects are created within an adjustable neighbourhood of the existing, actual data.

	id	age	height	width	target
	1	25	198	47	0

Added a synthetic object below

$\longrightarrow$	2	27	189	45	0

As illustrated above, a row is selected from the minority class (assume 0 is the minority class). SMOTE will then search for its k-nearest-neighbours and generate a new, synthetic sample that resides somewhere between the selected row and the neighbours. Exactly where it resides can be adjusted, thus allowing synthetics to be similar to existing data, or less so.

This method may work well on datasets with continuous values opposed to discrete. A drawback of the method is that inherent rules in the dataset may be violated. A rule as simple as this may cause issues:
`if column A between 0-5, column B must be between 0.5-1`
This might cause issues because SMOTE has no way of knowing limitations and rules such as this. Inconsistent data generated due to SMOTE are important to be aware of and solve before further analysis.

A common technique when working with data of a categorical nature is to map each category to corresponding integers. In cases where every feature is categorical, a resulting row will be a vector of integers. Utilising SMOTE to create synthetic rows of such vectors will most likely not produce a desirable result. Each object is mapped to an integer, but the distance between object 1 and object 2 is equal to the distance between object 1 and object 300. A synthetic sequence of events may not reflect any real sequence. A simple example; "words are fun". The mapping is created as follows, $\{"words" \to 1, "are" \to 2, "fun" \to 3\}$. The corresponding vector is $[1, 2, 3]$. If SMOTE was to randomly change any number inside the vector, it could produce a nonsensical sentence.

Under-sampling

Opposed to increasing the amount of minority class occurrences in the dataset, one could cut occurrences of the majority class. This is called under-sampling. By removing rows from the majority class the distribution will shift towards a more balanced ratio and hence reduce any existing imbalance.

2.8 Classification methods

Numerous different classification methods exist, all of which are designed to construct a trained model using different internal techniques. The methods described in this section are Naïve Bayes, Decision Tree, Gradient Boosted Trees, and Artificial Neural Networks. In chapter 3 we will give an overview of common algorithms used in the banking domain.

2.8.1 Naïve Bayes

The Naïve Bayes classification algorithm assumes that all attributes are conditionally independent and estimates class-conditional probability given a class label, y. Tan et al. [5] describes conditional independence by using an example of a person's arm length and reading ability. An observation might be that people with longer arms have greater reading skills. Such a relationship might be due to a confounding factor, such as `age`. Now the relationship between arm length and reading ability seems weaker, as age provides a more probable link. The assumption of conditional independence can be derived from the condition $P(X|Y,Z) = P(X|Z)$ and stated as

$$P(X_1, X_2, ..., X_n|Y) = \prod_{i=1}^{n} P(X_i|Y). \tag{2.1}$$

The Naïve-part of the algorithm comes naturally due to assuming that a feature's existence or lack thereof is unrelated any other feature. An example of this may be found with an apple. Lets assume three features, `round`, `colour` and `diameter`. If a figure is round it most likely has a diameter, hence these two features are dependent of each other to some, unknown degree. Naïve Bayes however, overlooks this dependency, and instead treats all features independently to contribute to a prediction that it is indeed an apple. The algorithm has a trade-off which is its lack of inference of information "between lines", such as sequential features, feature dependencies, or local trends. This trait makes Naïve Bayes more suitable for specific types of problems.

Often Naïve Bayes is associated with being a robust and basic algorithm, proving strong against issues such as isolated data points- often called noise. This is due to the nature of the algorithm. Noise will be weighed down in a statistical distribution because there are very few occurrences. It also handles missing values well, by simply ignoring them. Also, features that are irrelevant to a prediction will not negatively impact the prediction, as the probability would be uniformly distributed (similar to scaling all values with a constant). A weakness however, occurs when features are strongly correlated, as the assumption of conditional independence no longer holds.

The Naïve Bayes classification algorithm is noted for its efficiency on large datasets. This trait is derived from its training time holding linear time complexity, $\mathcal{O}(n)$. Naïve Bayes can outperform more sophisticated algorithms, especially in text classification. [16]

2.8.2 Decision Tree

Decision tree, or variants thereof, are used "under the hood" in different, more complex algorithms such as Gradient Boosted Trees. The core concept can be explained by a case where we have weather data on a set of days in Bergen (a rainy city in Norway). The task is to classify whether a day is `sunny` or `not sunny`, and the features are `temperature`, `season`, `mm of rain` and `cloudy`. By asking a series of questions we can construct a decision tree.

The first question may be if `mm of rain` is more than 50. If it is, the day was likely `not sunny`. If not however, it is still not clear. So we ask another question, whether or not it was `cloudy`. If it was, but little to no rain, the day was likely `not sunny`. If on the other hand there were no clouds and little to no rain, chances are it was `sunny`. The rules created above constitute the decision tree in figure 2.2.

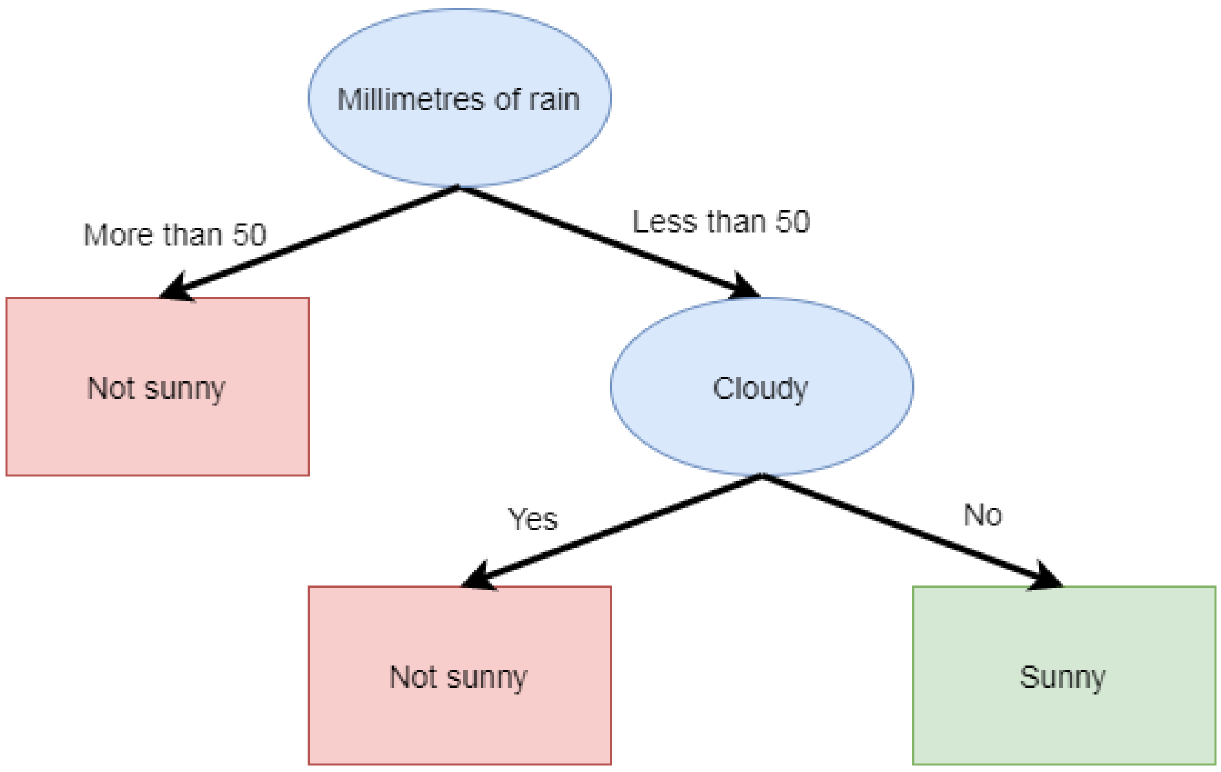


Figure 2.2: A simple decision tree example

To create a model through decision trees, the idea is to separate the largest chunk of data in the root node. By continuing to create child nodes until all nodes eventually point to a leaf node containing a prediction, the model is complete. Optimising the construction of decision trees is however complicated. This is because 2^n nodes may be needed. The worst case of a decision tree is effectively needing one leaf node per row of features, which also creates a completely overfitted model. Creating a decision tree that achieves the lowest classification error whilst keeping it as small as possible is an NP-Hard problem [17].

Several metrics exist to measure the quality of a split, also called impurity measure. The most common metrics are the Gini index, entropy, and information gain, where the latter is the change in entropy measured pre-split and post-split.

Once the model is created, predicting new is done by following the path through the splits in each node until a leaf node is reached. One of the advantages of the decision tree model is its innate opportunity for visualisation and explanation. The entire flow of any given feature input can be visualised. This is valuable for data analysts, and very different from complex algorithms such as neural networks.

An inherent advantage of decision trees is the ability to prune the trees to simplify the model. Issues such as managing the depth of the decision tree is complex, but a common method is called early stopping.

2.8.3 Gradient Boosted Trees

The idea of Gradient Boosted Trees is commonly attributed to Jerome H. Friendman's paper "Greedy Function Approximation: A Gradient Boosting Machine" in 1999 [18], although the idea was conceived two years earlier by Leo Breiman. The idea behind Gradient Boosted Trees is to utilise many, simple Decision Trees thus improving the overall quality of the prediction model. More specifically, the algorithm gradually creates weak learners, - simple decision trees commonly of four to ten levels. n levels contains $1+2^n$ nodes. As one weak learner is created, the algorithm evaluates the overall performance by two commonly used metrics in machine learning algorithms, *training loss function* and *regularisation function.*

Together, the two functions create what is called the *objective function* [19] shown in equation 2.2, a function whose purpose is to determine the value of a given model. Gradient Boosted Trees aims to optimise this function iteratively while constructing its model.

$$\text{obj}(\theta) = L(\theta) + \Omega(\theta) \tag{2.2}$$

$L(\theta)$, the training loss function, is commonly mean squared error or logistic loss for logistic regression. Equation 2.4 shows these two error measures respectively. y_i is the true class label, while $\hat{y}_i$ represents the predicted class label. The predicted class label is commonly derived from some linear model such as shown in equation 2.3,

$$\hat{y}_i = \sum_j (\theta_j x_{ij}) \ , \tag{2.3}$$

where x_i represents training data and all its features. θ represents the weighted input features that accounts for some prediction y_i given x_i.

$$\begin{aligned} L(\theta) &= \sum_i (y_i - \hat{y}_i) \\ L(\theta) &= \sum_i \Big(y_i \ln (1 + e^{-\hat{y}_i}) + (1 - y_i) \ln (1 + e^{\hat{y}_i}) \Big) \end{aligned} \tag{2.4}$$

$\Omega(\theta)$, the regularisation function, is important as it accounts for overfitting. The term overfitting in machine learning means to create a model that too accurately fits the dataset it was trained on, consequently having poor performance on new, unseen data. Avoiding overfitting is so important that not attempting to minimise it may render entire projects completely useless. Take the example in figure 2.3. In this example the green line's model is capable of reaching 100% accuracy, but applying such specific parameters that fit the dataset perfectly will likely lead to very poor performance on unseen data. Visually one can imagine using the green line on a new, unseen dataset. This is likely not a good approach, as it would require a dataset that is exactly equal to the one it was trained on. In figure 2.3 this would entail having a dataset structured the same way with the blue and red dots in the exact same locations. The black line however, represents a regularised function approximating the results of the green line in a smoother manner.

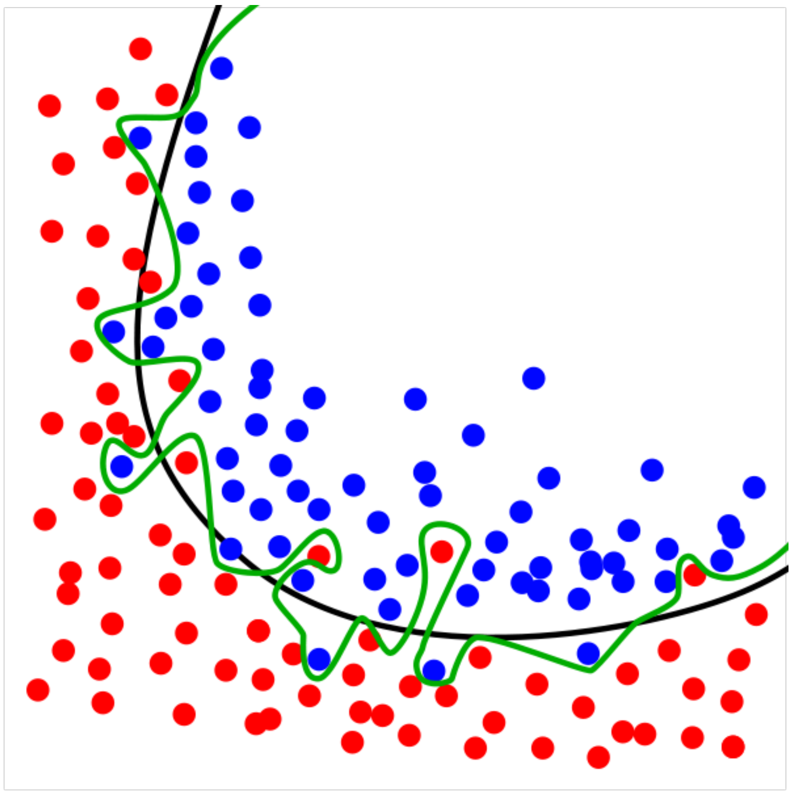


Figure 2.3: An example of overfitting in a binary classification problem. Blue dots are true class labels, red are false. The green line represents an overfitting model, and the black line is a regularised model

To summarise, a more regularised model will outperform an overfitted model on unseen data. This is because overfitted models are tuned to fit only the datasets they were trained on, while being unable to effectively handle new, unseen data. A different example of reducing overfitting is shown in figure 2.4.

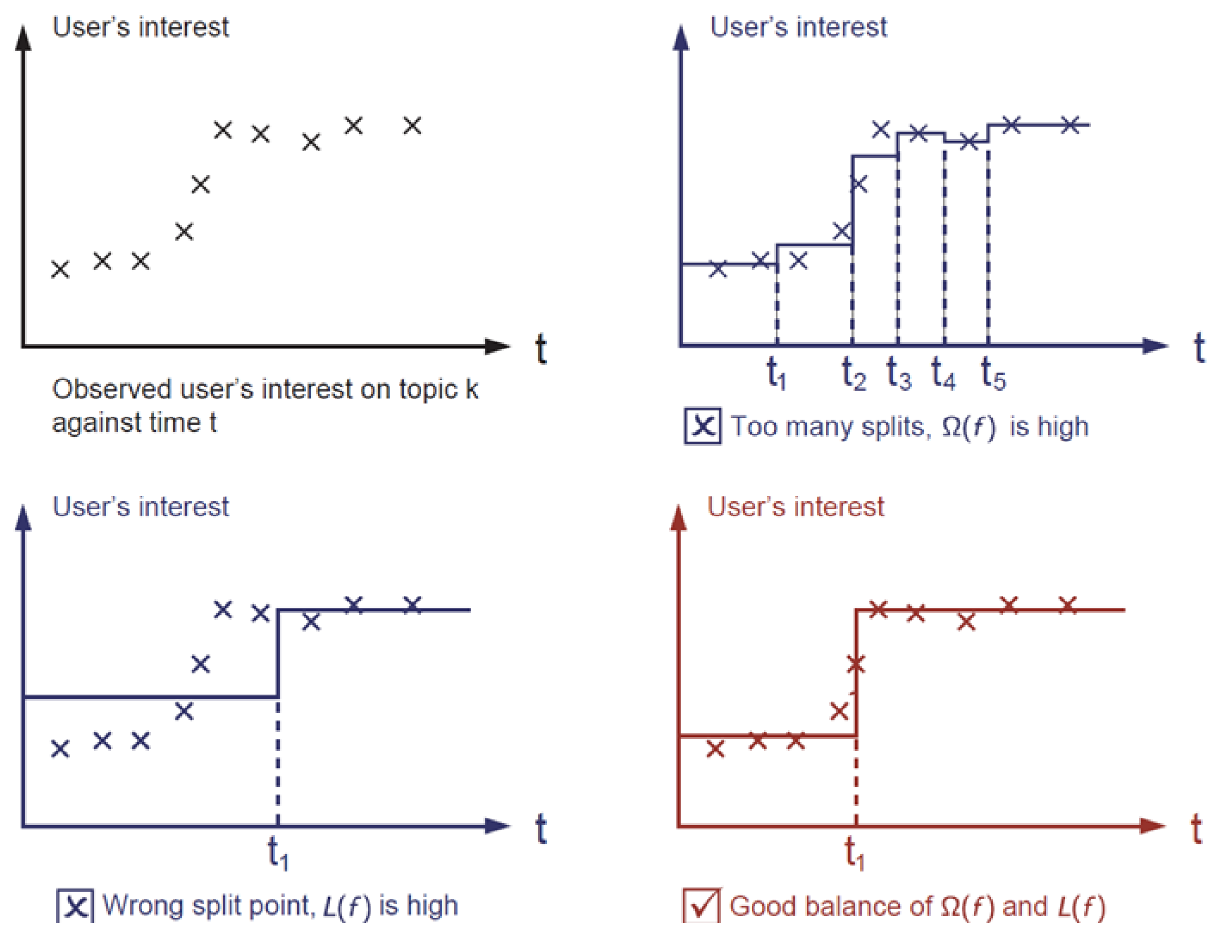


Figure 2.4: Visualising the steps that lead to a proper regularisation function [20]. A split refers to splitting a decision tree

Gradient Boosted Trees is member of the ensemble methods. "Ensembles are machine learning methods for combining predictions from multiple separate models" [21]. Two methods of this technique are common; *bagging* and *boosting*. The Random Forest algorithm uses bagging, while Gradient Boosted Trees uses boosting. The concepts are similar in that bagging utilises strong learners from the start and tries to smooth out their predictions, while boosting utilises weak learners and attempts to boost their predictions [21].

The weak learners utilised in Gradient Boosted Trees are variations of Decision Trees with one important distinction to common Decision Trees (cf. figure 2.2). The leaf nodes do not contain class label `true` or `false`, but rather decision scores as seen in figure 2.5. Trees such as these are called CART- classification and regression trees.

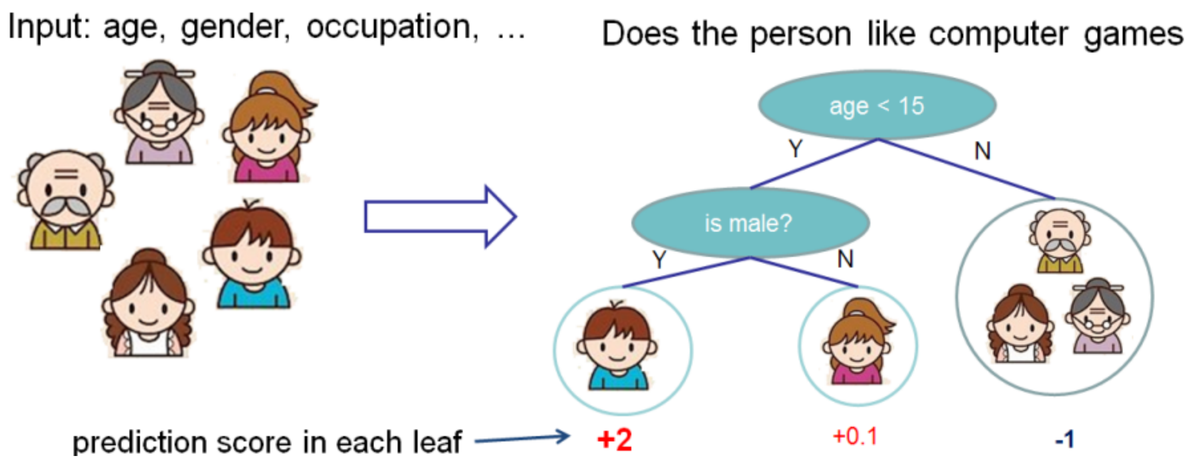

Input: age, gender, occupation, ...
Does the person like computer games
age < 15
Y
N
is male?
Y
N
prediction score in each leaf
+2
+0.1
-1

Figure 2.5: A simple CART example [20]

The leaf nodes' scores alone are never sufficient to make predictions, and as such many CARTs are created sequentially where each one focuses on learning from the mistakes of the previous. Thus, the tree ensemble method attempts to minimise bias-variance through boosting several weak learners and combining them into one strong learner, or rather, the final classification model.

While at the stage seen in figure 2.6, the Gradient Boosted Trees algorithm trains multiple trees containing different information. This may be visualised as optimising several impurity measures and regularisation functions of different trees while attempting to minimise one of the error rates shown in equation 2.4. This details of this process is non-trivial and will not be covered in this report.

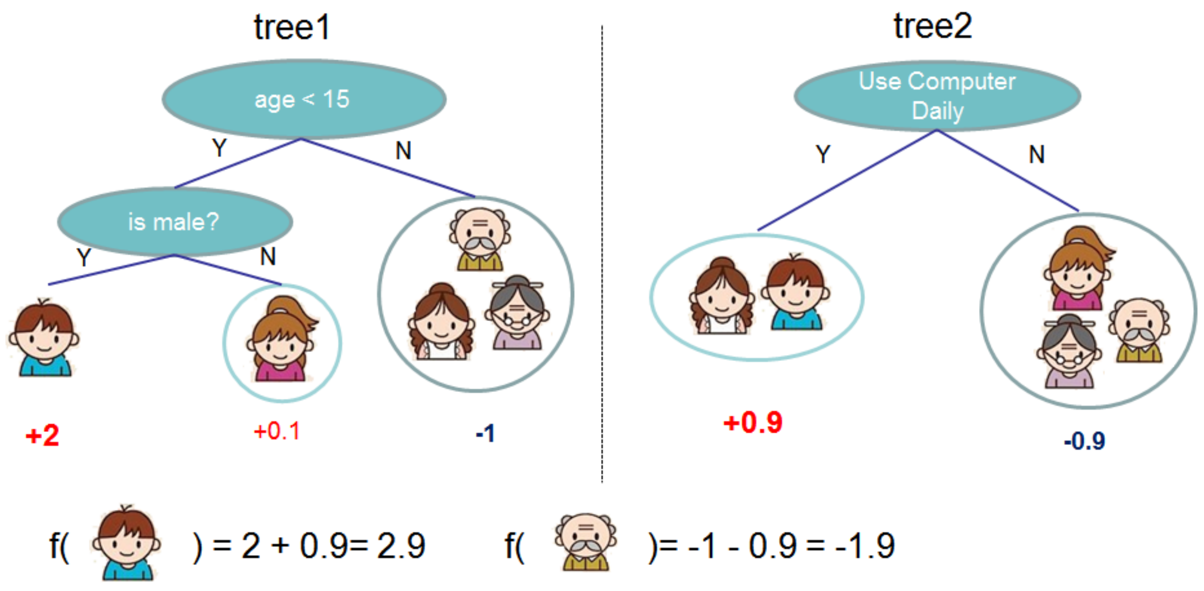


Figure 2.6: Tree ensemble example [20]

2.8.4 Artificial Neural Networks

Artificial Neural Networks are computational models that are loosely inspired by the structure of the human brain. In recent years, major breakthroughs in the research of Neural Networks have drastically improved the state-of-the-art in speech recognition, object detection, drug discovery and genomics [22] [23]. Previous classification methods mentioned are linear, while Neural Networks with at least one hidden layer are able to make non-linear classifications. In essence, a Neural Network mimics a human brain's neurons, while commonly utilising a concept of cost function optimisation. Numerous types of networks exist, varying heavily in complexity and structure.

Figure 2.7 shows the flow in a simple Feed-Forward Neural Network. Here, information in the form of features pass through a hidden layer where different weighting schemes are applied before returning an output prediction. This concept is further built on to create many different networks. Networks are often optimised towards solving specific problems, such as Recurrent Neural Networks performing well on problems where context is required, such as text classification. Long Short-Term Memory Neural Networks are tailored to data where time steps are important (sequentially ordered data in some time continuum, for instance), as they are able to remember information through millions of time steps.

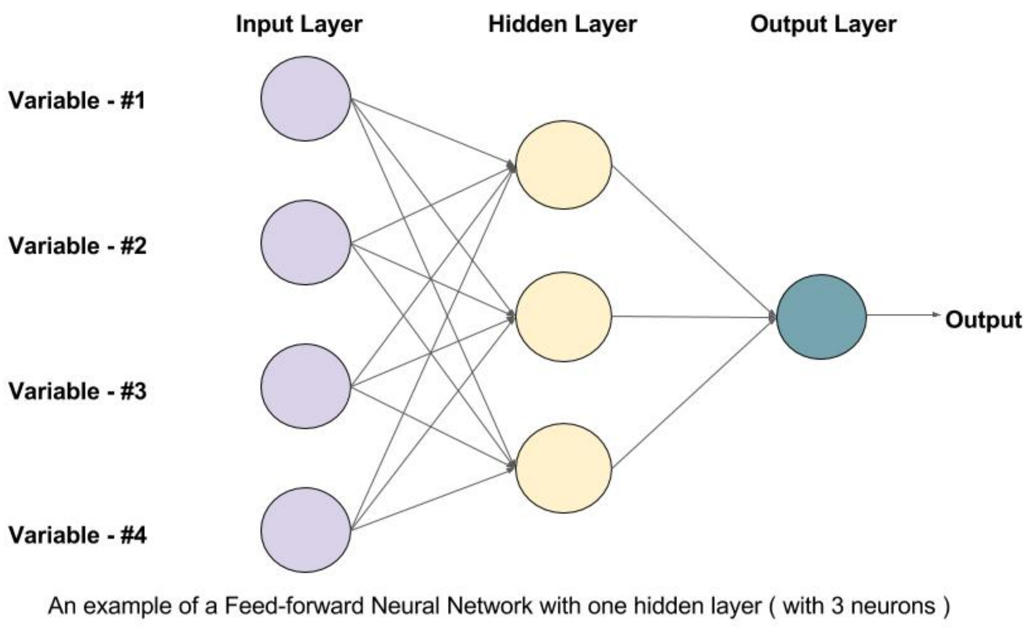


Figure 2.7: Example of Feed-Forward Neural Network with one hidden layer

Research on various Neural Networks is blooming and the expansion of the field is happening rapidly. Within neuroscience, recent studies are steadily approaching a bridge between neuron interactions in the brain versus specialised Neural Networks [23]. While there are many cases where Neural Networks perform well, there are several drawbacks that are natural to mention. The training of networks such as LSTM are extremely costly, both in terms of hardware and time. Commonly, Neural Networks also require a tremendous amount of data to be able to uncover trends as correctly as possible. Furthermore, attempting to debug or examine the resulting function is very challenging. Essentially the models are so complex that they are often treated as black boxes.

2.9 Evaluation metrics

In order to properly evaluate a classification model, it is common to use the hold-out method described in section 2.7.1. This requires a dataset that is labelled in advance. The partitioning may be done in many variations and is ultimately at the analysts' discretion. The majority of literature suggests ratios of 60% training set and 40% testing set, or 70%-30%. Once the partitioning is done, a model is trained and evaluated on the unseen testing set. By then comparing the model's predictions with the testing sets actual class labels, several metrics become available.

What follows are some of the metrics available for evaluating the performance of a classification model. Current literature indicates that all of the metrics are common and utilised today. We observe that there is no consensus that any **one** metric is accepted as ideal and optimal. Several metrics may be necessary to evaluate performance and the choice between these metrics depend on the nature of the dataset and the application of the model.

2.9.1 Accuracy

The accuracy of any given classification model is

$$\text{accuracy} = \frac{\text{number of correct predictions}}{\text{number of total predictions}} \tag{2.5}$$

When binary classification models are created and evaluated it is common to refer to the class label as either **positive** or **negative**. The following terminology is important to know to understand the results of an evaluation,

- **True positive**: Actual $\underline{\text{positives}}$ predicted as $\underline{\text{positives}}$
- **False positive**: Actual negatives predicted as positives
- **True negative**: Actual $\underline{\text{negatives}}$ predicted as $\underline{\text{negatives}}$
- **False negative**: Actual positives predicted as negatives

The results of a classification model is commonly described using these terms. The underlined elements indicate the relation within a true prediction, a relation where the model predicts the correct class label. These four categories of predictions are often found in a *confusion matrix.*

To exemplify, assume a model is trained on some training set containing bio-medical measures of a patient. This model is to be tested on an independent test set consisting of 1000 rows of bio-medical measurements. The class label is whether or not a patient is sick. The test set is known to consist of 500 rows where the class label is positive, thus indicating the patient is sick. The remaining 500 rows have class label negative. After running the test set through the model, results are observed in table 2.1. This type of table is known as a confusion matrix.

	Predicted positive	Predicted negative
Actual positive	490	10
Actual negative	50	450

Table 2.1: Confusion matrix for a bio-medical binary classification problem

Using the terminology introduced earlier, the confusion matrix in table 2.1 reads **TP**: 490, **TN**: 450, **FP**: 50, **FN**: 10. The accuracy is $\frac{490+450}{1000} = 0.94$ or 94%.

2.9.2 Precision and recall

Precision and recall are two widely used measures indicating how precise a model is, and the ratio of samples of a certain class it is able to identify. These two measures are considered trade-offs. By maximising precision, recall is likely to suffer and vice versa.

They are defined as follows:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

In binary classification the measures are meant to answer the following questions,

- Precision: "Given a positive prediction, how likely is it to be correct?"
- Recall: "Given an actually positive sample, how likely is the model to detect it?"

This implies that precision scales by the number of false positives, and recall scales by the number of false negatives.

The values of a confusion matrix may also be expressed as percentages as follows,

- **True positive rate (sensitivity)**: $\text{TPR} = \frac{\text{TP}}{\text{TP+FN}}$
- **True negative rate (specificity)**: $\text{TNR} = \frac{\text{TN}}{\text{TN+FP}}$
- **False positive rate (FPR)**: $\text{FPR} = \frac{\text{FP}}{\text{TN+FP}}$
- **False negative rate (FNR)**: $\text{FNR} = \frac{\text{FN}}{\text{TP+FN}}$

Note that the true positive rate is equal to recall. An example confusion matrix from table 2.1 using rates can be seen in table 2.2.

	Predicted positive	Predicted negative
Actual positive	0.98	0.02
Actual negative	0.10	0.90

Table 2.2: Confusion matrix using rates for a bio-medical binary classification problem

2.9.3 F$_\beta$-measure

Precision and recall may be combined into a single measure called the F$_\beta$-measure, shown in equation 2.6.

$$\mathrm{F}_\beta = (1+\beta^2)\,\frac{\text{Precision}\cdot\text{Recall}}{\beta^2\cdot\text{Precision}+\text{Recall}} \tag{2.6}$$

Both precision and recall are bounded in the open unit interval, $\mathrm{F}_\beta \in (0,1)$. Three variations of equation 2.6 are commonly used. F$_{0.5}$-measure weights precision higher than recall, whilst the F$_2$-measure weights recall higher than precision. These measures are less common than the F$_1$-measure, often simply called the F-measure. This measure is found by setting $\beta = 1$ and is called the harmonic mean between precision and recall shown in equation 2.7.

$$\mathrm{F}_\beta = 2\,\frac{\text{Precision}\cdot\text{Recall}}{\text{Precision}+\text{Recall}} = \frac{2}{\frac{1}{\text{Precision}}+\frac{1}{\text{Recall}}} \tag{2.7}$$

2.9.4 ROC-curve

Another metric that is commonly used to compare performance evaluations of classification models is the **R**eceiver **O**perating **C**haracteristic-curve. This measure utilises a more visual approach where a graph is plotted with y-axis as the true positive rate (TPR) and x-axis as the false positive rate (FPR). Figure 2.8 illustrates a ROC-curve for two different models. In essence, a ROC-curve enables a visual evaluation of the trade-off between true and false positive rates.

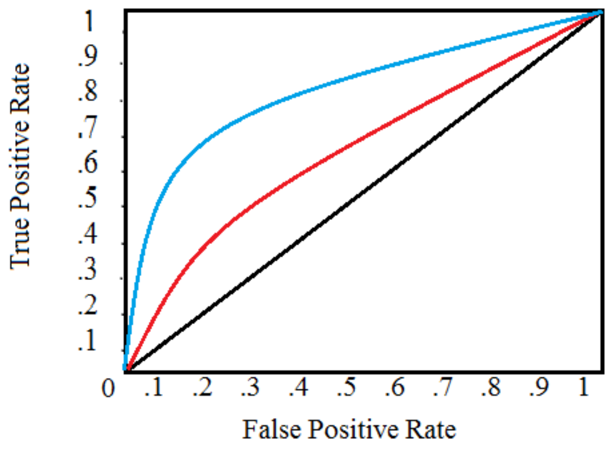


Figure 2.8: Example ROC-curves. Diagonal represents a fixed-probability guessing classifier, red and blue curves are different classification models.

Generally, the closer towards the top-left corner a curve bends, the better. A high performing model should have a high true positive rate while at the same time maintaining a low false positive rate. Figure 2.8 shows that the model represented by the blue curve vastly outperforms the red curve's model. It has both a higher TPR and a lower FPR for every measured point. The diagonal line represents a fixed-probability guessing model as a baseline for the illustration's purposes.

ROC-curves work well as an evaluation method for class imbalanced datasets [5]. Purely relying on accuracy would in class imbalanced cases be misleading, as simply guessing the majority will likely yield very high accuracy, but very low actual value. To exemplify this phenomenon, assume a dataset of 1000 samples with binary class labels of false or positive. If 950 samples have class label of false, any classifier predicting false on all samples would achieve 95% accuracy. However, its recall measure for the class label of positive would zero, as it would fail to find any positive class labels. In scenarios such as this ROC-curves provide valuable insight in evaluating the performance of a classifier.

Chapter 3

Related work

This chapter details related work to sales prediction problems within the banking domain. An explicit connection between churn prediction and sales prediction is described.

3.1 Churn prediction problems vs. sales prediction problems

As briefly mentioned in the final paragraph of section 2.1, there are large similarities between churn prediction and sales prediction. This connection proves useful as previous research on churn prediction is applicable. However, while churn prediction problems may more often be forced to battle severely imbalanced datasets, sales prediction problems are less likely to have the same degree of imbalance. This effect is observable and can be illustrated by an example. Predicting a likely purchase of a product is more efficient when utilising datasets with more occurrences of sales. The more sales of a product, the more training the model gets to find likely purchasers. By selecting products with a high turnover one can maximise the potential value for a sales prediction. Applying the same concept to churn prediction is comparable to predicting churn in a company with a large portion of churning customers. This does not occur often, as companies tend to aim to minimise turnover of customers. As such, the degree of dataset imbalance occurs slightly differently in the two domains.

Being able to draw a connection between churn prediction and sales prediction also opens up more research and related work on a field where breadth of research may prove important. With a concept such as "all datasets are unique", great breadth in research means that it is likelier to observe efficient methods that are applicable for our project. The implementations of churn prediction and sales prediction are also similar enough that they might be called equal. With these factors in mind, we will assume that churn prediction problems are conceptually equal to sales prediction problems throughout our project.

3.2 Churn prediction in the banking domain

Several studies on churn prediction within the banking domain were analysed and can be found in table 3.1. The table shows the applied classification algorithms of each article and what year the study was conducted. Table 3.2 shows each article's utilisation of class distribution-altering sampling techniques. Note that none of these studies detail the usage of event log data. Some of the studies do not properly describe the dataset or any processing, while others describe the datasets in such a way that it is natural to assume some aggregations have been made.

Title of study	**Technique**	**Year**
Predicting credit card customer churn in banks using data mining [24]	Logistic Regression, MLP, Decision Trees (J48), Random Forest, Radial Basis Function, SVM	2008
Application of fuzzyARTMAP for churn prediction in bank credit cards [25]	NN (fuzzyARTMAP)	2009
Customer Churn Prediction Using Improved One-Class Support Vector Machine [26]	SVM	2005
Customer churn prediction using improved balanced random forests [27]	Random Forests	2009
Customer Retention in Banking Sector using Predictive Data Mining Technique [28]	Decision Tree (CART)	2011
Data Mining as a tool to Predict the Churn Behaviour among Indian bank customers [29]	Naïve Bayes, Decision Trees, SVM	2013
Prediction of customer attrition of commercial banks based on SVM model [30]	SVM, Logistic Regression	2014
Predicting customer churn in banking industry using neural networks [31]	Neural Network	2015

Table 3.1: Explored classification techniques in churn prediction literature within the banking domain

Title of study	**Class distribution-altering**
Predicting credit card customer churn in banks using data mining [24]	Over-sampling (SMOTE), under-sampling
Application of fuzzyARTMAP for churn prediction in bank credit cards [25]	Over-sampling (SMOTE), under-sampling
Customer Churn Prediction Using Improved One-Class Support Vector Machine [26]	None
Customer churn prediction using improved balanced random forests [27]	Improved Balanced Random Forests
Customer Retention in Banking Sector using Predictive Data Mining Technique [28]	None
Data Mining as a tool to Predict the Churn Behaviour among Indian bank customers [29]	None
Prediction of customer attrition of commercial banks based on SVM model [30]	Random sampling, over-sampling, under-sampling
Predicting customer churn in banking industry using neural networks [31]	None

Table 3.2: Class distribution-altering sampling techniques for table 3.1

All of the studies listed in table 3.1 apply one or more binary classification algorithms to customer datasets in order to predict whether or not a customer is likely to leave the bank. A common trend in the datasets used is a high degree of class imbalance [24] [25] [28] [30]. This type of imbalance will introduce bias into the models leading them to favour the majority class. To combat this challenge, over- and under-sampling is applied [24] [25] [30]. The studies show that this technique can be highly effective in improving recall for the models produced. The most common metrics to evaluate the results are specificity (true negative rate) and recall [24] [25] [28] [29] [30]. Accuracy is also used by [26] and [27], but as we discussed in section 2.7.2, accuracy may be highly misleading. In the following sections we will give a brief overview of how two studies improved recall and precision from the baseline models using sampling techniques.

3.2.1 Predicting credit card customer churn in banks using data mining

This study [24] uses a customer attributes set which contains general customer characteristics such as age, income, sex, and educational level. The dataset consists of 14 814 rows where 93.24% are non-churners and the remaining 6.76% are churners. In order to smooth out the class imbalance, SMOTE and under-sampling is applied. The study examines the performance of Multilayer Perceptron (a type of Feed-Forward Neural Network), Radial Basis Function, Random Forest, Logistic Regression, Decision Trees and SVM. Their baseline results show that the top two performing algorithms are Decision Tree and Random Forest. They achieve the highest recall scores of 0.63 and 0.62 respectively. Their preliminary sampling strategy shows that by applying

SMOTE they are able to increase the recall by 4.2% for Decision Tree and 12.2% for Random Forest. Using under-sampling the recall is increased by 10.4% for Decision Tree and 23.5% for Random Forest. The results were achieved using stratified, random sampling with a 70:30 split between training and testing data. The precision scores are omitted in this study making us unable to evaluate how the sampling affects precision. However, this study is highly relevant to our project as our customer attributes dataset is similar in nature and exhibits a similar ratio of class imbalance. The size of our dataset however is considerably larger than the one used in this study.

3.2.2 Prediction of customer attrition of commercial banks based on SVM model

This study [30] proves relevant to our project as it tackles the issue of a highly imbalanced dataset. The study presents a dataset containing 50 000 customer records from a Chinese commercial bank. This dataset contains customer attributes such as age, sex, education, income, and occupation. The class distribution ratio between churners and non-churners is 0.91% : 99.09% or 1:109.23.

The study examines and compares the differences in prediction performance by utilising several sampling techniques. Under-sampling accompanied by random sampling is used to obtain and explore five ratios; 2:1, 1:1, 1:2, 1:5 and 1:10 churner to non-churner. The baseline results of the original set achieves zero recall for both Logistic Regression and SVM while the Radial Basis Function achieves 0.26. The best results are achieved at the sampling ratio of 2:1 with a recall score of 0.94, 0.94 and 0.89 for LR, SVM and RBF respectively. In these cases the precision scores range between 0.02 and 0.025.

The study does not go into great depth in explaining their dataset in terms of available features, but seems to have similarities with our customer attributes dataset. The degree of class imbalance of the dataset in this study is around ten times higher than in our datasets.

3.3 Sales prediction in banking

The concept of predicting sales can be a highly complex problem, especially if combined with financial theory of consumer patterns. Li et al. [32] notes the following: “At any point, consumers have demands for multiple products. Economic theory has shown that in the presence of finite resources, there is a ‘priority structure’ among the demand objectives and that this priority structure can be transformed into a priority structure for products.”

The study conducted by Li et al. [32] in 2005 further examines how the financial maturity of a household over time is connected to consumer purchase patterns for products ordered in some sequential structure. Some of their findings indicate that consumers are more likely to purchase some products before others, while other findings detail differences in the demand maturity continuum based on factors such as education and gender.

Another study by Knott et al. [33] in 2002 examines next-product-to-buy models for cross-selling applications. This is similar to the experiments by Li et al. [32], however provides methodological guidelines for implementing NPTB-models and further discussion on several important topics. Examples of the latter are what data to use, what statistical model, and whether it is enough to predict *what* product is to be bought or if *when* the product is bought is also crucial.

Knott et al. [33] conclude that the most crucial predictor to include in an NPTB-model is `current product ownership`. This is equivalent to product breadth in the dataset we received from SpareBank 1 SMN.

Table 3.3 contains an overview of the two studies and their applied techniques. The development of these statistical models is not comparable to the machine learning classification approaches applied in our project. Therefore we do not consider evaluation scores of these studies to be comparable to our results.

Title of study	**Technique**	**Year**
Cross-Selling sequentially Ordered Products: An Application to Consumer Banking Services [32]	Ideal point model, multivariate probit model, explicitly developed statistical model	2005
Next-product-to-buy models for cross-selling applications [33]	Logistic regression, multinomial regression, discriminant analysis, neural networks	2002

Table 3.3: Explored classification techniques in sales prediction within the banking domain

Chapter 4

Data

This chapter describes the dataset received from SpareBank 1 SMN.

4.1 SpareBank 1 SMN's dataset

The dataset received from SpareBank 1 SMN consists of four tables:

- `Kundeinfo` - Customer info (attributes) - 6 519 993 (205 799 unique customers) rows.
- `Logg` - Customer event logs - 368.1 million rows.
- `Salg` - Customer sales - 386 253 rows.
- `VYKODE` - Event codes and explanations - 349 rows.

We will henceforth refer to the customer info table as the *customer attributes* dataset and the VYKODE-table as *event codes*. It is worth noting that the events collected in the customer event logs are primarily used for analysis of how the user navigates on the platform, or even debugging the bank's software, rather than for data mining purposes.

The relations between the tables are seen in figure 4.1. Notice that the `kunde_id`-field is present in the `Sales`, `Log` and `Customer info` table. This acts as a join key which allows us to join and aggregate data. Each `VYKODE` (event code) contains a mapping to a description of the event.

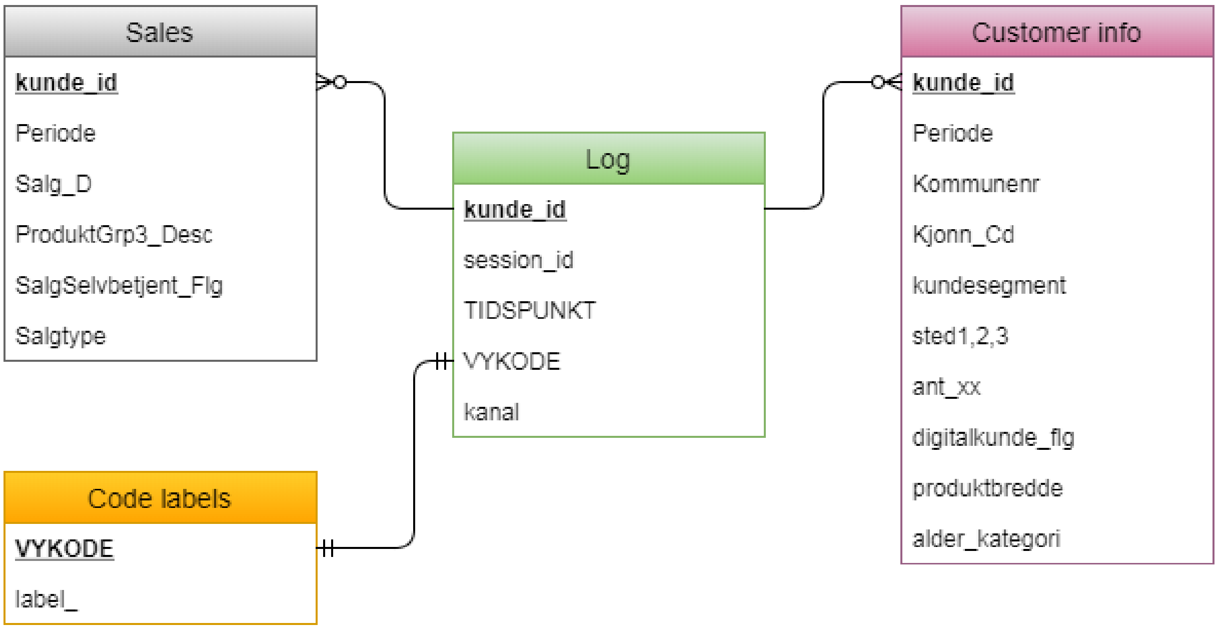


Figure 4.1: ER diagram of the different tables of the SpareBank 1 datasets

What follows is a detailed description of each table. We have chosen to convert some of the column short names into more readable names in addition to translating them from Norwegian to English.

4.1.1 Customer attributes

This table contains a subset of SpareBank 1's customers. For each unique pair of customer_id and period, where period is a month, the customer attributes are stored in several columns. This acts as a snapshot for any customer given a period in time. Each column in the table contains personal data in addition to various subscription and purchase statuses for different products that SpareBank 1 provides. Some of these columns may vary across different periods. Table 4.1 shows a description and data type for each column.

Field name	Description	Type
Customer Id	The id of the customer	Double
Period	The month the record was produced	String
Commune number	The Norwegian commune number for the customer	Integer
Gender	Customer gender	String
Customer segment	Customer segment according to amount of products and volume	String
Location 1-3	Three columns that represent pre-aggregated areas the customer belongs to in SpareBank 1	String
Insurance products	One column for each product; property, vehicle, travel, life, unemployment, children	Boolean
Pension account	If the customer has a pension account	Boolean
Saving account	If the customer has a savings account	Boolean
Mutual fund	If the customer has a mutual fund	Boolean
Saving agreement	If the customer has a saving agreement	Boolean
Active checking account	If the customer has an active checking account	Boolean
Debit card	If the customer has a debit card	Boolean
Credit card	If the customer has a credit card	Boolean
Product breadth	Amount of categories a customer has active products in	Integer
Age category	Age categories of the customer, pre-aggregated buckets	String

Table 4.1: Customer attributes table descriptions

The following columns contain categorical features:

- `Gender`: M/F - Male or female
- `Customer segment`: “Active”, “Total” and “Passive”
- `Age category`: Seven categories (buckets): 0-14, 15-17, 18-24, 25-34, 35-44, 45-59, 60+
- `Location 1-3`: Different levels of granularity for pre-aggregated areas representing an operational unit inside SpareBank 1.

Notable value distributions for some of the fields are shown in table 4.2.

Field name	**Type**	**Comment**
Customer Id	Double	This feature will not be used in the classifier training step
Customer start	Nominal	Earliest date: Jan. 1975, latest: Dec. 2017
Period	Nominal	34 different periods. Period for most rows: January 2015. Period for least amount of rows: December 2017
Commune number	Nominal	269 values. The biggest communes: 1601 (26.5%), 1702 (5.4%) and 1719 (4.1%)
Gender	Nominal	Male (52.8%), Female (47.2%), missing values: 90 338 (1.4%)
Customer segment	Nominal	Active (57.2%), Total (32.3%) and Passive (10.5%)
Active checking account	Nominal	True (71.4%), False (28.6%)
Mutual fund	Nominal	True (20.1%), False (79.1%)
Savings account	Nominal	True (64.8%), False (35.2%)
Pension account	Nominal	True (1.8%), False (98.2%)
Product breadth	Numeric	Average: 4.93, std: 2.81, min: 0, max: 16

Table 4.2: Notable value distributions for customer attributes

4.1.2 Customer event logs

The customer event logs table contains customer events from either SpareBank 1’s website or mobile application. In this project we will not discern between platforms and treat all events equally. The events are generated by various user actions such as `log out`, `log in`, `My overview`, and so on. There are a total of 349 distinct events. This is the largest table in the dataset by far, containing around 368.1 million rows. `Log out`- and `Log in`-events make up a vast majority of the rows, approximately 39.31%. Table 4.3 shows a description of this table.

Field name	**Description**	**Type**
Customer Id	The id of the customer	Double
Session Id	The id used during the sessions	String
Timestamp	Timestamp of the action	String
VYkode	A code that represent a specific action	String
Channel	The channel where that action was performed	String

Table 4.3: Customer event logs table descriptions

The following columns contain categorical values,

- `VYKode`: 349 different actions such as login, logout, cancel subscription, and similar
- `Channel`: “Portal”, “Mobil”. Shows what platform generated the log entry

4.1.3 Customer sales

The sales table contains a row for each product purchased or subscription created in SpareBank 1 SMN. This is not an exhaustive list, but a subset of services and products. Table 4.4 contains a description of this table.

Field name	**Description**	**Type**
Customer Id	The id of the customer	Double
Period	The month the record was produced	String
Sales day	Date of the sale	ISO formatted date string
Product group	The group the product belongs to	String
Self-service	If the sale was done by a customer sales representative or through self-services	Boolean
Sale type	If the customer was new or bought extra products	String

Table 4.4: Customer sales table descriptions

4.1.4 Event codes

This table contains the possible event codes for actions performed on the website or in the mobile application. Each row has a label that gives a textual description of the action that the event code represents. There are four rows with missing values and 27 rows where the label descriptions have the same value as the VYKODE. That means that we cannot know what these codes represent. There are 58 values that are missing a description. Table 4.5 shows a description of this table.

Field name	Description	Type
VYKODE	Event code used in logs	String
label	Label describing the event code	String

Table 4.5: Event codes table descriptions

4.1.5 The most common codes in logs

The most common event codes and their frequencies are shown in table 4.6.

VYKODE	Description	Count
lo099	Log out	80 464 357
lo050	Log in	64 242 658
pm700	My overview	24 083 784
lo001	Login started	21 378 849
pm017	Payment confirmed / signed	14 889 972
pm016	Payment with KID	14 513 012
lo010	OpenAM	13 667 596
loG50	Login with personal number / password	13 401 843
pm890	Unkown - not documented event	13 305 820
pf702	Find retailers based on transaction history	11 336 513
pf701	Read transaction history	11 236 624
pm050	Register a transfer	10 670 849
lo012	Login with BankID completed	7 764 855
co030	Signing	6 916 385
loE50	Login with BankID	6 660 700
pm015	Payment with message	6 642 090
pm011	Accept eInvoice	5 714 399
nb700	My overview	4 048 463
nb001	Login	3 137 276
pm001	Edit payment	3 114 648
pm006	Add payment recipient automatically	2 183 328

loC50	Login with BIM	1 730 216
lo070	Customer elevates authentication level	1 666 514
nb200	Accept one ore several payments	1 535 590
nb005	Selection of agreement	1 109 275
nb019	New payment type	1 104 762
nb990	Log off	1 090 792
loA50	Login with	926 573
pm002	Delete payment	757 259
pm950	Go to external service	717 422

Table 4.6: The top 30 most frequently observed event codes in the customer events logs table

4.1.6 Dataset challenges

Apart from the potential loss of accuracy from the anonymisation process required by NSD, we experienced challenges concerning class imbalance and sparsity. Our dataset contains 10.5% purchasers and 90.5% non-purchasers. Datasets like this tend to produce biased models that favour the majority class. We use various sampling techniques to control this bias.

While attempting to extract customers event timelines based on time periods leading up to a sale, we found that many of the timelines contained mostly zero-events (no event). This sparsity lead to poorly performing models where in some cases the predictions were made purely based on the number of zero-events. We therefore discarded the idea of using time periods as a constraint for generating timelines.

Chapter 5

Methods

This chapter presents our chosen approach in solving a complex, composite problem that is sales prediction. We developed two main pipelines each utilising distinct sets of tools and methods. What follows is a short introduction to our experiment before our method is described.

5.1 Introduction

The activity of targeting customers with marketing campaigns can generally be split into two main strategies,

- Mass marketing
- Direct marketing

Mass marketing aims to reach a large audience with one general message in a *fits-all* fashion. The format and contents of the message is not tailored to any specific customer's dispositions, but to a segment of the customers often based on general characteristics such as age, gender, and country. TV-advertisement, billboards, radio commercials, and mass-mail campaigns are all instances of mass marketing.

Direct marketing on the other hand, aims to tailor the message to a specific customer. A retail bank has information on the service and purchase history of a customer along with personal attributes such as income, owned property, marital status, and so forth. Based on this information, the bank might create a customer persona detailing behavioural patterns. By comparing the modelled persona to other customers, the bank is able to make educated deductions as to their approach in selling a product. Examples of such approaches are calls from a sales-person, or personalised emails.

A challenge with both mass and direct marketing is finding the optimal frequency of marketing contact with the customer. If a customer is constantly being exposed to advertisement by a certain vendor it may cause fatigue or grievance that in turn impacts the vendor in a negative way. Therefore, limiting the frequency of marketing contact to a tolerable level where the customer is likely to respond in a positive way is optimal. By reducing this frequency it becomes even more important to select the most appealing message to the right customer. To account for the reduction of contact frequency, one might wish to increase the precision of customers predicted likely to buy a product.

In our experiment we train several classification models using the datasets provided by SpareBank 1 SMN. Our goal is to identify the customers that are most likely to buy a credit card based on their personal attributes and interaction history with the bank. When evaluating the effectiveness of the generated classification models we look mainly at the recall and precision scores for customers that are predicted to be purchasers. We define the following mappings from the model evaluation metrics to the business goals:

- **Maximising precision**: The model is optimised to discover only customer who are purchasers (true positives), but erroneously discards some purchasers due to a low classification confidence (false negative).
- **Maximising recall**: The model is optimised to discover all customers who are purchasers (true positives) at the cost of erroneously including customers that are not (false positives).

When selecting customers to target in a mass marketing campaign we can allow ourselves to maximise recall in the hopes that the positive response rate will compensate for the possible negative impact the non-interested customers may have. In direct marketing the message needs to be accurate and we therefore need to maximise the precision when selecting the target customers.

Our experiment is focused on maximising recall for `purchasers`, while at the same time monitoring changes in precision. We will include the F_1 measure to provide an overview of the balance between precision and recall. The experiment pipeline which we describe in section 5.3, is comprised of multiple activities. At each step we measure the change in recall. As mentioned in 2.9.2, the cost of improving recall is generally the loss of precision. Throughout the experiment we will show how this balance is altered by applying different classification algorithms and sampling techniques. We will not perform tuning of hyperparameters of any algorithms, but use the default configurations supplied by KNIME and Sci-kit. These configurations can be found in appendix B.

5.2 Data mining and analysis tools

In order to efficiently establish a working data analysis pipeline we have used several data mining tools, some of them open source. Table 5.1 lists the software and various tools used in our experiment,

Library / Framework	**Purpose**
Sci-kit learn	Machine learning algorithms and evaluation
KNIME	Preprocessing, feature selection, training classifier, evaluation
Google Cloud	Infrastructure to run preprocessing jobs and training classifiers
Pandas	Python library for data aggregation and preprocessing
Numpy	Python library for data aggregation and preprocessing

Table 5.1: The software and tools used in our experiment

Sci-kit learn is an open source machine learning library for the Python programming language. It provides implementations of popular classification and regression algorithms such as Random Forest, Gradient Boosted Trees, Naïve Bayes, and Decision Trees. It further features various helper functions to facilitate data preprocessing, sampling and model evaluation. As sci-kit is used with Python we can utilise Python's interoperability by running the scripts on various platforms. This is a major benefit as we can deploy the aggregation and training scripts using computational power on any cloud platform. The file size of the datasets were a major challenge as they exceeded the amount of RAM on commodity hardware by several hundred gigabytes. By moving the aggregation and training steps to the cloud we could adjust the computing power to fit our needs.

KNIME Analytics platform is an open source data analysis platform that features a graphical interface to build pipelines from graphical blocks, or nodes. Similar to sci-kit, it provides implementations of common machine learning algorithms in addition to data transformation and processing functions. In contrast to sci-kit, KNIME is a fully integrated platform that primarily runs on your own hardware. It is possible to run KNIME on a public cloud, but it requires some configuration.

Google Cloud is a public cloud that provides flexible, computational power, data storage, data analytics, and machine learning tools. We primarily use Google Cloud's *Compute Engine* which allows users to provision virtual machines of various configurations. The virtual machines can be configured with up to 624GB of RAM and 96 vCPUs (a single thread on a multi-core processor).

5.3 Approach and pipeline description

Our experiment requires two comprehensive pipelines that inputs the raw datasets, transforms and preprocesses the datasets, trains multiple classifiers, and finally generates evaluations of the classifiers' performance.

Previous research mainly focuses on training classifiers on data such as customer attributes. Examples of such attributes are age, gender, income, and similar customer characteristics that are valid for a certain point in time. Usually a target label such as customer churn or product churn is chosen. A part of our experiment aims to see if sequence data in the form of customer events can provide comparable or better performance than static customer attributes in predicting credit card sales. Our dataset contains both customer attributes data and customer events which allows for a fair comparison of the two approaches. The target label to predict is the customer enrolment of a credit card subscription, which we refer to as *purchaser* in the datasets.

We split the experiment into two pipelines, *A* and *B*. `Pipeline A` describes the activities performed on the customer event logs dataset and `Pipeline B` describes the customer attributes dataset. Each step is denoted with a letter and number describing which dataset and at what step the activity is performed. Throughout the pipeline we move from the raw datasets collected from the bank's data warehouse to data structures that can be used as input for training classification models. We also evaluate multiple classification algorithms and proceed to the next steps wherein we select the best performing algorithms primarily based on the recall values for `purchasers`, but precision is also considered. While accuracy is a popular metric, in the case of imbalanced datasets it may be highly misleading (see section 2.9.4).

At the end of the experiment we combine the models produced by `Pipeline A` and `B`. This makes it necessary to set aside a partition of the dataset that will not be used for training in `Pipeline A` and `B`. A graphical description of how the dataset is partitioned and used for training can be seen in figure 5.1. For all instances of partitioning we use stratified sampling and 10-fold cross validation as it is considered a generally better scheme both in terms of bias and variance when compared to regular cross-validation [34].

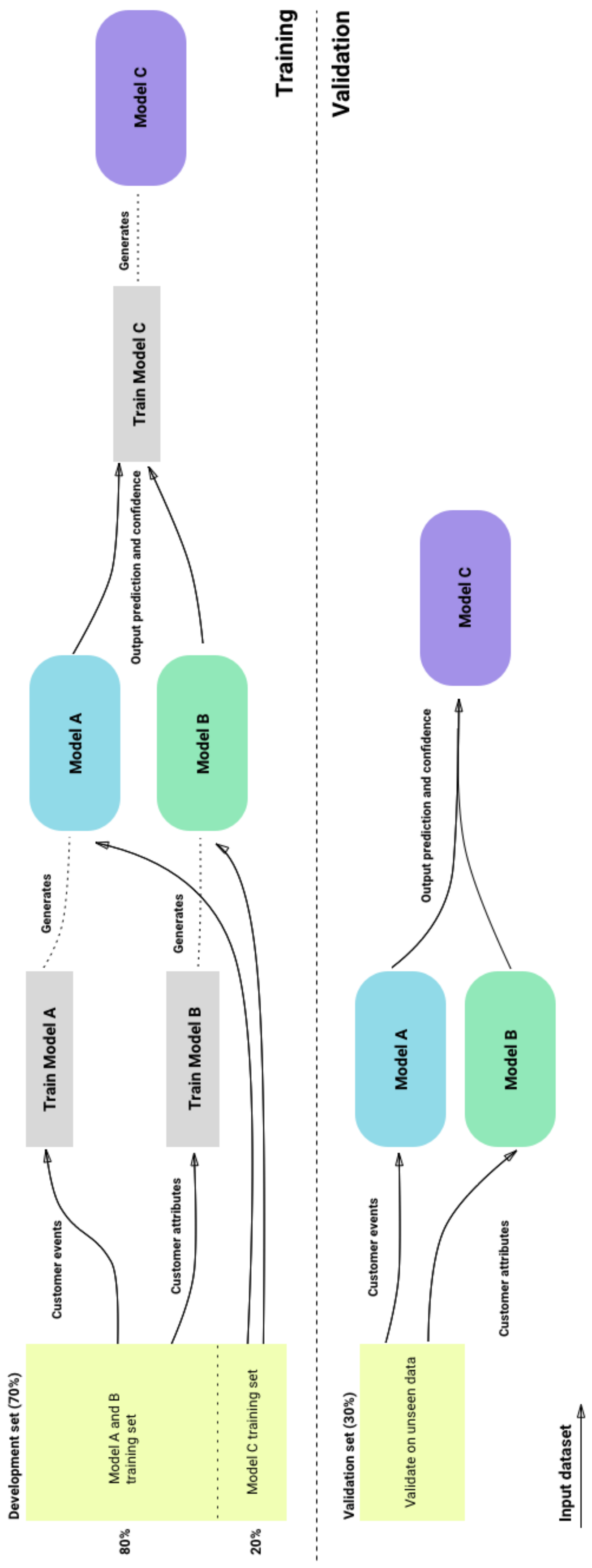
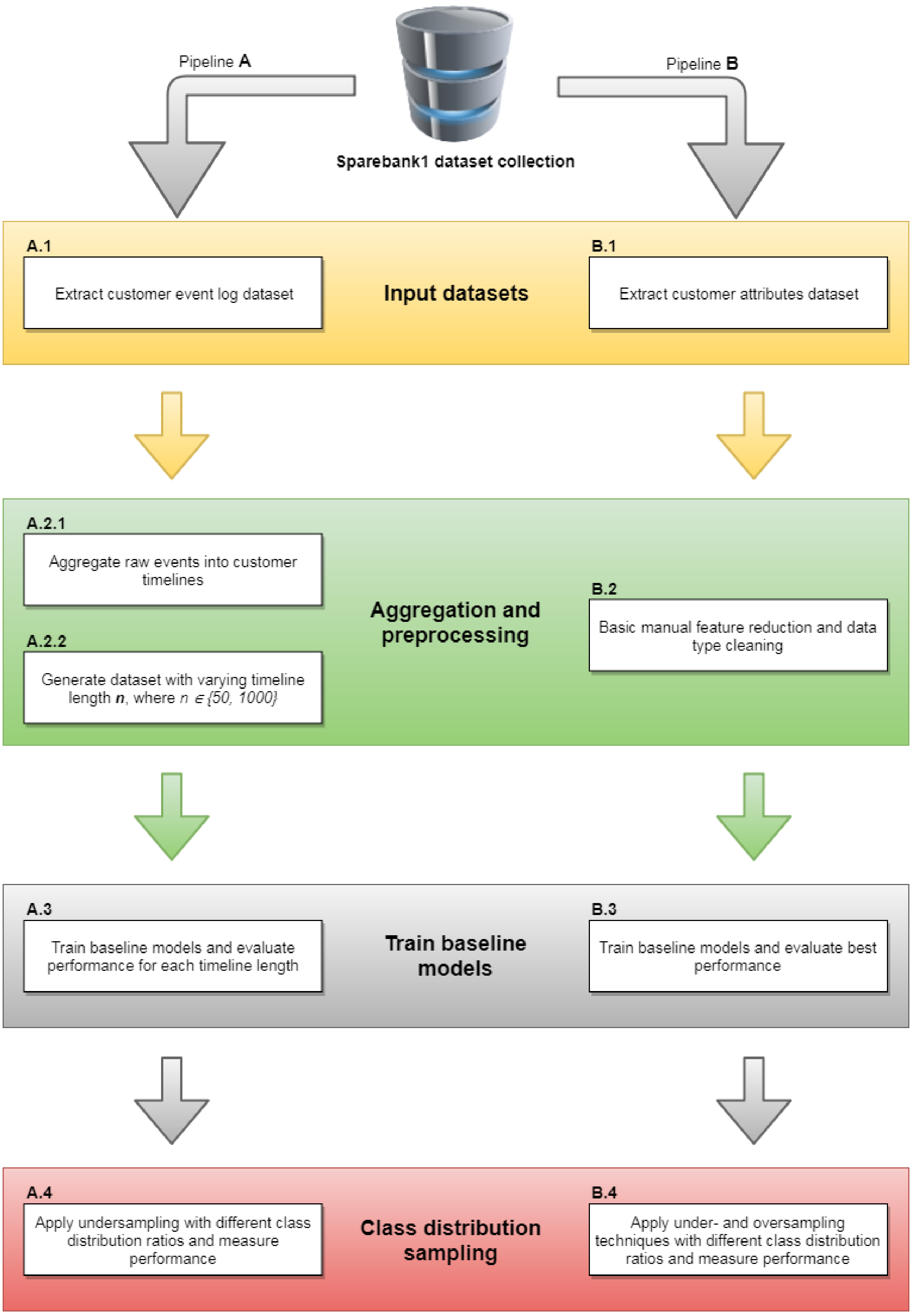


Figure 5.1: High level overview of our project flow. Further illustrates how the dataset is partitioned and used for training and evaluation

A high-level overview of the activities performed in `Pipeline A` and `B` is illustrated in figure 5.2.


Pipeline A
Pipeline B
Sparebank1 dataset collection
A.1
Extract customer event log dataset
Input datasets
B.1
Extract customer attributes dataset
A.2.1
Aggregate raw events into customer timelines
A.2.2
Generate dataset with varying timeline length n, where n ∈ {50, 1000}
Aggregation and preprocessing
B.2
Basic manual feature reduction and data type cleaning
A.3
Train baseline models and evaluate performance for each timeline length
Train baseline models
B.3
Train baseline models and evaluate best performance
A.4
Apply undersampling with different class distribution ratios and measure performance
Class distribution sampling
B.4
Apply under- and oversampling techniques with different class distribution ratios and measure performance

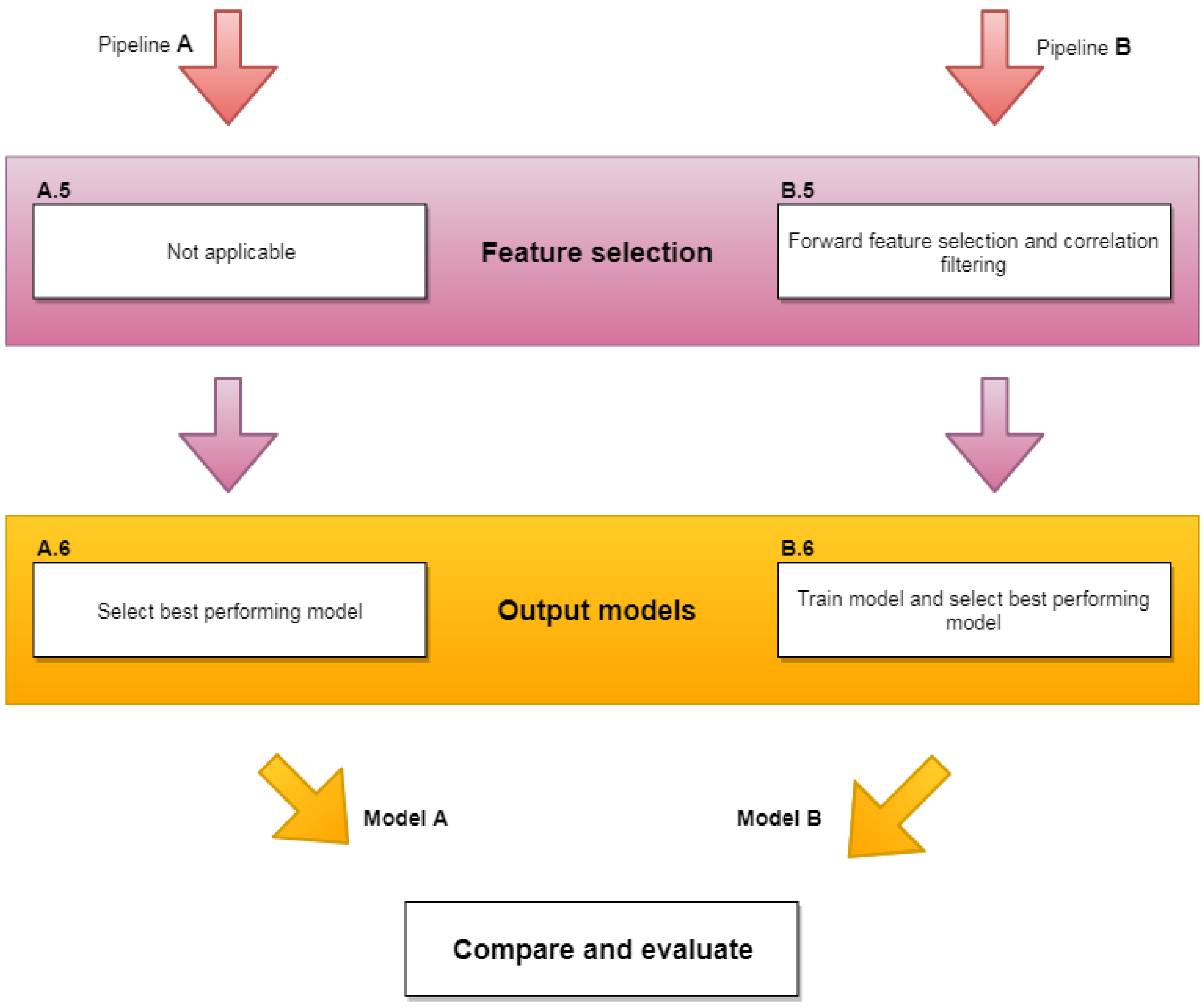

Pipeline A
Pipeline B
A.5
Not applicable
Feature selection
B.5
Forward feature selection and correlation filtering
A.6
Select best performing model
Output models
B.6
Train model and select best performing model
Model A
Model B
Compare and evaluate

Figure 5.2: High-level pipeline illustration

Each step in the pipeline is a set of functions or transformations that produce an output which is used as input for the next step. The two datasets require different types of aggregations, preprocessing and sampling techniques before they are fed to the classifiers. In the last step we evaluate and compare the performance of the classifiers trained on the customer attribute and event logs datasets.

In the next sections we will describe in detail the function and implementations of each step while intermediate results necessary to drive the process to the next step are referenced.

5.4 Experiment

We refer to the two datasets containing customer event logs and customer attributes as `Customer events dataset (Pipeline A)` and `Customer attributes dataset (Pipeline B)`.

5.4.1 Input datasets

A.1

We received a collection of datasets from SpareBank 1. For this step we used the raw customer event logs described in chapter 4.

B.1

From the same collection of datasets we extracted the customer attributes dataset.

5.4.2 Aggregation and preprocessing

A.2.1

In order to train a classifier on the customer event logs dataset, we need to transform it into a data structure that is suitable as input for the classification algorithms. We aggregate the events into a timeline of events for each customer. Table 5.2 shows an excerpt of the raw log table while table 5.3 shows the final, aggregated table. The events are also in order such that timestamp(event$_n$) $<$ timestamp(event$_{n+1}$). However, the timestamp of event$_n$ for customer 1 is not necessarily the same as the timestamp of event$_n$ for customer 2.

We also discard all events that occur after a credit card sale has been conducted. If the customer has not purchased a credit card, we include all the events for the customer.

The size of the raw event table is 37GB which required close to 180GB of RAM to perform the necessary aggregations. We used a compute instance (see section 5.2) on Google Cloud with 200GB of RAM and 2 vCPU cores. Note that this code was not optimised for parallelism which would have greatly reduced the aggregation time.

Customer	**Event**	**Timestamp**
1	Log in	1
1	View balance	2
1	Log out	3
2	Log in	1
2	Transfer funds	2
2	View balance	3
2	Log out	4
3	Fail log in	1
3	Log in	2

Table 5.2: Customer event logs table excerpt

Customer	Event 1	Event 2	Event 3	Event 4
1	0	Log in	View balance	Log out
2	Log in	Transfer funds	View Balance	Log out
3	0	0	Fail log in	Log in

Table 5.3: Aggregated customer timelines table

Finally, we mapped every event code to a number $n \in \{0, 349\}$. The dataset contains a total of 349 distinct events (1 to 349) and 0 refers to no event (zero-event). The actual purchaser class label was appended by joining with the sales dataset described in chapter 4.

A.2.2

In this step we have a dataset with the complete event timelines of every customer. The length of the timelines vary greatly with some containing only ten to twenty events, and others up to thousands.

We generate 13 versions of this dataset by limiting the timeline lengths to the n most recent events leading up to a sale or in the case of no sale, the n latest events, where $n \in \{50, 100, 150, 200, 250, 300, 400, 500, 600, 700, 800, 900, 1000\}$. These datasets will be used to understand how the classifier performance varies according to the timeline length.

We also attempted to limit event timelines by periods of time in days d, where $d \in \{15, 30, 60, 120\}$. However with the constrain of time periods we observed that (1) some customers have a large number of events in short time spans, which led to the necessary step of padding customers with fewer events with 0s (no event). Ultimately this led to a highly sparse dataset where the classifier basically made its prediction by the number of observed zero-events. And (2) the number of days we selected initially may have proved too short, as there are great variances in timelines constructed from a dataset spanning three years. This was also a source for sparsity. Due to both of the factors (1) and (2) we did not continue with the time period constraints when generating datasets.

B.2

The raw customer attributes dataset which is shown in table 4.1, contains several attributes that do not need to be included in the training input,

- Customer id
- Period (date of the snapshot)
- Number of credit cards

We remove `Customer id` as it is redundant for training, `Period` because the most recent period is selected in all cases, and the `Number of credit cards` because they contain the current credit card status of the customers. Instead of the latter, we add a `Purchaser (true/false)` label that is obtained by joining with the sales dataset. This mapping between class label and customer id is the same for both pipelines, meaning that a customer id in `Pipeline A` represents the same customer id in `Pipeline B`. This ensures that there are no inconsistencies that might have been present because of a faulty data collection.

5.4.3 Train baseline models

As described in section 5.3, the datasets in both pipelines are divided into a development set (70%) and validation set (30%). 80% of the development set is used for developing the models in `Pipeline A` and `B`. This means that the same customers are used for training in both models. Prediction output from Models A and B can be joined by the customer ids to compare predictions. The validation set is retained for the final evaluation of the models towards the end of the pipelines.

A.3

We use the customer event logs dataset with varying timeline lengths generated in **step A.2.2** as input. For each dataset with event timeline n we train four models using 10-fold cross validation and compare the precision and recall scores. We have selected the following algorithms,

- Decision Trees
- Naïve Bayes
- Random Forest
- Gradient Boosted Trees

The parameter configurations of each algorithm are the default values for Sci-kit and KNIME and can be found in appendix B.

A summary of the results from this step can be seen in figure 6.1 and 6.2. For the full baseline scores of all timeline lengths and algorithms, see appendix C. We will select the best performing dataset and algorithms, and proceed with further preprocessing techniques to improve the recall and precision scores. We define the following ranking function to select the best performing algorithm,

$$\text{score}_{\text{algorithm}} = \max\left(\text{precision}_{\text{purchaser}}, \text{recall}_{\text{purchaser}}\right) \tag{5.1}$$

The best dataset is selected on the following basis,

- Shortest timeline length, n
- Highest score according to equation 5.1

Ideally we want to select the lowest possible n as the dataset's size decreases and in turn reduces training time. At the same time we want to maintain the highest possible precision and recall scores. In our case the results show little to no fluctuation in precision and recall for all the classification algorithms except for Naïve Bayes (further explored in section 7.1). We select Gradient Boosted Trees and Random Forest because they achieve the highest score according to our ranking function. In the later steps we will attempt to trade precision for higher recall values. Because the precision does not increase considerably as n increases we select $n = 50$ as the dataset we proceed with. The results for this dataset are shown in table 6.1.

B.3

The customer attributes set is used to train models using the same algorithms as in **step A.3**. The results for the classifiers are shown in table 6.2.

5.4.4 Class distribution sampling

In this step we attempt to improve recall by altering the class distribution for the target label. The raw datasets have about 1:10 ratios of purchasers to non-purchasers which might drive the classifiers to favour the majority class. By using under- and over-sampling techniques we can force a more equal class distribution which will reduce the bias during training. When using under-sampling we throw away a number of the samples that are non-purchasers until the pre-selected ratios are reached. We then observe how this affects precision and recall for purchasers by training models at class distribution ratios of `1:1`, `1:2`, `1:5` and `1:10`. We do this for both the customer event logs and attributes datasets.

A.4

We under-sample the customer event logs dataset with timelines of 50 events and train models using Gradient Boosted Trees and Random Forest. 10-fold cross validation is used in order to produce the recall and precision scores seen in tables 6.3 and 6.4. We do not apply over-sampling using SMOTE because of the reasons mentioned in section 2.7.2. As our primary goal is to maximise recall for purchasers we select the best performing algorithm, Random Forest at a 1:1 ratio.

B.4

We apply the same sampling steps to the customer attributes dataset which produces the scores shown in tables 6.5 and 6.6. Additionally we apply over-sampling using SMOTE which produces the scores shown in tables 6.7 and 6.8. As we are looking to maximise recall for purchasers we select Random Forest using the under-sampled dataset at a ratio of 1:1.

5.4.5 Forward feature selection and correlation filtering

After establishing the baseline model scores and the scores for the sampled datasets we apply some feature selection methods to see if we can reduce the dataset's size any further.

A.5

We will not apply this to the customer event logs dataset because of the inherent features of the data structure. Every column in this dataset represents an event at some point in time for a customer. The point in time will most likely differ between the two rows. The only similarity is that the event is the n-th event before a sale, or n-th most recent event in the case where the customer is not a purchaser. Applying any sort of feature reduction methods such as correlation filtering or forward feature selection could remove inherent information in the sequential data structure.

B.5

In the customer attributes dataset the columns represent static features shared by every row. In this case we use correlation filtering and forward feature selection to reduce the dimensions of the dataset, while maintaining its integrity. We analyse the correlation between every feature to see if there are columns that can be removed without impacting the model performance in a negative way. Figure 6.7 illustrates a correlation matrix between every feature. We then perform *forward feature selection* which is an instance of the *wrapper-method* described in section 2.6. We use the change in recall for purchasers to determine whether or not a column should be included.

5.4.6 Output models

The output models are evaluated and compared.

A.6 and B.6

To conclude the `Pipelines A` and `B` we train Models A and B using the following dataset configurations:

- Under-sampling to 1:1 purchaser to non-purchaser ratio for both datasets
- Using Random Forest for both datasets
- $n = 50$ for the customer event logs dataset (A)
- Correlation filtering for the customer attributes dataset (B)

Forward feature selection for the customer attributes dataset was not used in the end, as this led to lower precision and recall measures seen in table 6.9. The reduction in training time was not worth the cost in precision and recall as the training time was already within an acceptable time frame. However, correlation filtering was used as one feature was perfectly correlated and thus removable with no cost.

The models are trained using 80% of the development set. The performance of these models are measured using the validation set. Final results for `Pipeline A` and `B` are listed in table 6.10. These two models function as input for the following step.

5.4.7 Combining Models A and B

We now have two trained models that lets us predict a credit card purchaser based on customer attributes and the 50 most recent events of the same customer. We can select any customer from the database that the model has not seen before and make predictions utilising both models. The output from both models will be the credit card purchaser prediction along with a confidence score for the prediction made.

Seeing as the two models are trained on different data for the same customer we attempt a combination of the two models in order to produce higher precision and recall scores than either one in isolation. Thus, we define a third model, *Model C*, that takes as input the prediction and confidence produced by the two Models A and B.

We train Model C on predicted data which has not been used for training for neither Model A nor B. This is the partition in figure 5.1 marked `Model C training set` and makes up for 20% of the development set. The input format can be seen in table 5.4,

Model A $_{\text{Prediction}}$	Model A $_{\text{Confidence}}$	Model B $_{\text{Prediction}}$	Model B $_{\text{Confidence}}$	Purchaser
1	0.854	0	0.978	1

Table 5.4: One example of a row (customer) of the combined output from Model A and Model B used as input to train Model C. The `Purchaser` column is appended for Model C to be able to train in a supervised manner

We train four models and measure the performance. In this step we do not apply under-sampling to force a 1:1 purchaser to non-purchaser ratio. Our rationale for this choice is that we wanted to keep the algorithm's input as untouched as possible.

The following algorithms are applied,

- Gradient Boosted Trees
- Random Forest
- Decision Tree
- Naïve Bayes

which produces the results seen in table 6.11. Table 6.12 contains the final results of Model C's performance run against the validation set.

Chapter 6

Results

This section shows the results from the experiment conducted in section 5.4. We include the most noteworthy results from each step in `Pipeline A` and `B`. Refer to figure 5.2 for an overview of the activities in the two pipelines.

6.1 SpareBank 1 SMN experiment

All following tables will have the same format where $P_{\text{class label}}$ represents precision and $R_{\text{class label}}$ represents recall. $F_{1\ \text{class label}}$ represents the F_1 measure. The `class label` represents the binary classes we predict between, either purchaser or not purchaser.

6.1.1 Baseline models

What follows are results from baseline classifier performances.

Pipeline A: Customer event logs dataset

These results acts as baseline classifiers for the customer event logs dataset from `Pipeline A`. Detailed metrics of all the algorithms for $n \in \{50, 100, 150, 200, 250, 300, 400, 500, 600, 700, 800, 900, 1000\}$ can be found in appendix C. Figures 6.1 and 6.2 show precision and recall as a function of n.

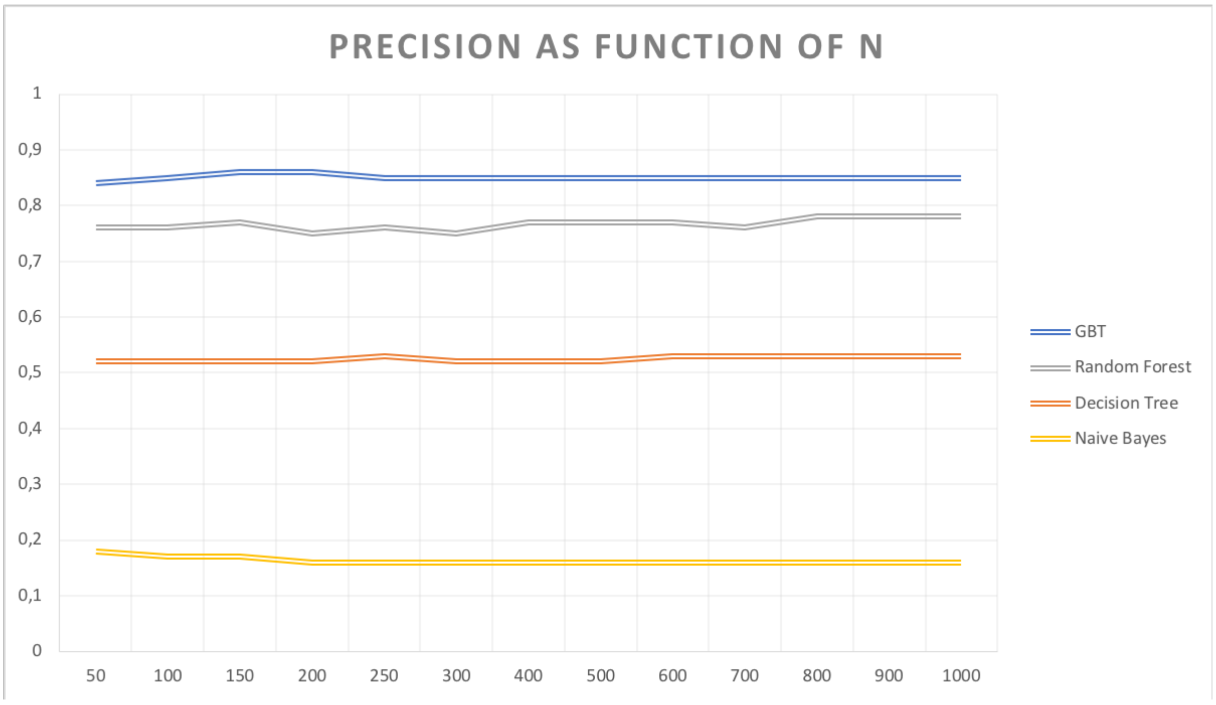


Figure 6.1: Pipeline A: Precision as a function of n (number of events in timeline)

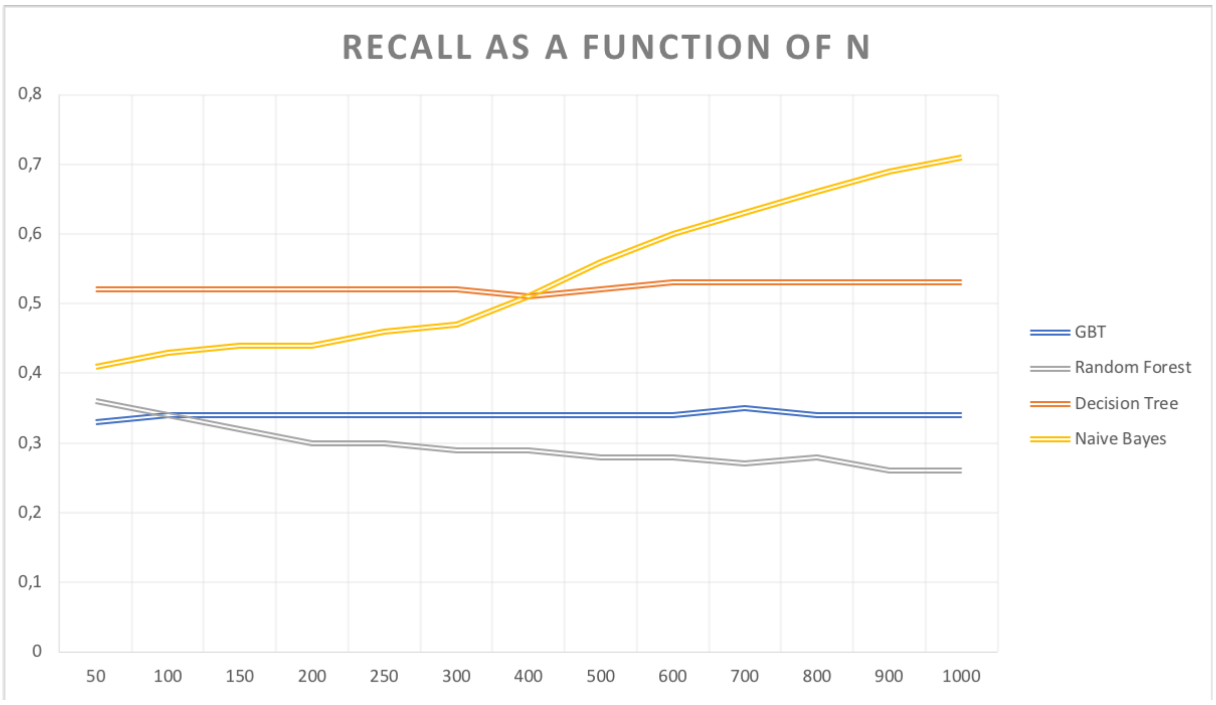


Figure 6.2: Pipeline A: Recall as a function of n (number of events in timeline)

We observe that Gradient Boosted Trees and Random Forest produce the highest precision scores with little to no fluctuation as n increases. Decision Trees produces the best recall score at $n \in [50, 400]$ after which Naïve Bayes takes the lead. This is further analysed in the final paragraphs of section 7.1. The results for $n = 50$ seen in table 6.1 represent the algorithms and sequence length performing the best according to our criteria listed in section 5.4.3.

Algorithm	$P_{\text{purchaser}}$	$R_{\text{purchaser}}$	$F_{1\ \text{purchaser}}$	$P_{\text{not purchaser}}$	$R_{\text{not purchaser}}$
Gradient Boosted Tree	0.84	0.33	0.47	0.91	0.99
Random Forest	0.76	0.36	0.49	0.91	0.98

Table 6.1: Baseline model performance of customer event logs dataset with timelines of 50 most recent events using 10-fold cross validation

Pipeline B: Customer attributes dataset

The results for the baseline performance on the customer attributes dataset can be seen in table 6.2.

Algorithm	$P_{\text{purchaser}}$	$R_{\text{purchaser}}$	$F_{1\ \text{purchaser}}$	$P_{\text{not purchaser}}$	$R_{\text{not purchaser}}$
Gradient boosted tree	0.78	0.24	0.37	0.90	0.99
Random Forest	0.80	0.31	0.45	0.90	0.99
Decision Tree	0.43	0.39	0.41	0.91	0.92
Naive Bayes	0.23	0.43	0.30	0.89	0.77

Table 6.2: Baseline model performance of customer attributes set using 10-fold cross validation

6.1.2 Class distribution sampling

In this section we present the results from under-sampling the majority class of non-purchasers to force specific purchaser to non-purchaser ratios. Over-sampling is also shown for `Pipeline B`. Note that for both of the following tables the final row with a ratio of 1:10 is considered to closely resemble the dataset used to train the baseline models.

Pipeline A: Customer events dataset

Ratio	$P_{\text{purchaser}}$	$R_{\text{purchaser}}$	$F_{1\ \text{purchaser}}$	$P_{\text{not purchaser}}$	$R_{\text{not purchaser}}$
1:1	0.43	0.77	0.55	0.96	0.86
1:2	0.58	0.63	0.60	0.95	0.94
1:5	0.79	0.43	0.56	0.92	0.98
1:10	0.84	0.33	0.47	0.91	0.99

Table 6.3: Gradient Boosted Trees performance on customer event logs dataset using various class distribution ratios by under-sampling

Ratio	$P_{\text{purchaser}}$	$R_{\text{purchaser}}$	$F_{1\ \text{purchaser}}$	$P_{\text{not purchaser}}$	$R_{\text{not purchaser}}$
1:1	0.44	0.82	0.57	0.97	0.85
1:2	0.58	0.72	0.64	0.96	0.93
1:5	0.76	0.53	0.62	0.94	0.98
1:10	0.76	0.36	0.49	0.91	0.98

Table 6.4: Random Forest performance on customer event logs dataset using various class distribution ratios by under-sampling

From scores plotted in figure 6.3 and 6.4 we observe that the recall score increases as the class balance evens out and that precision increases when the opposite occurs. This holds true for both Gradient Boosted Trees and Random Forest.

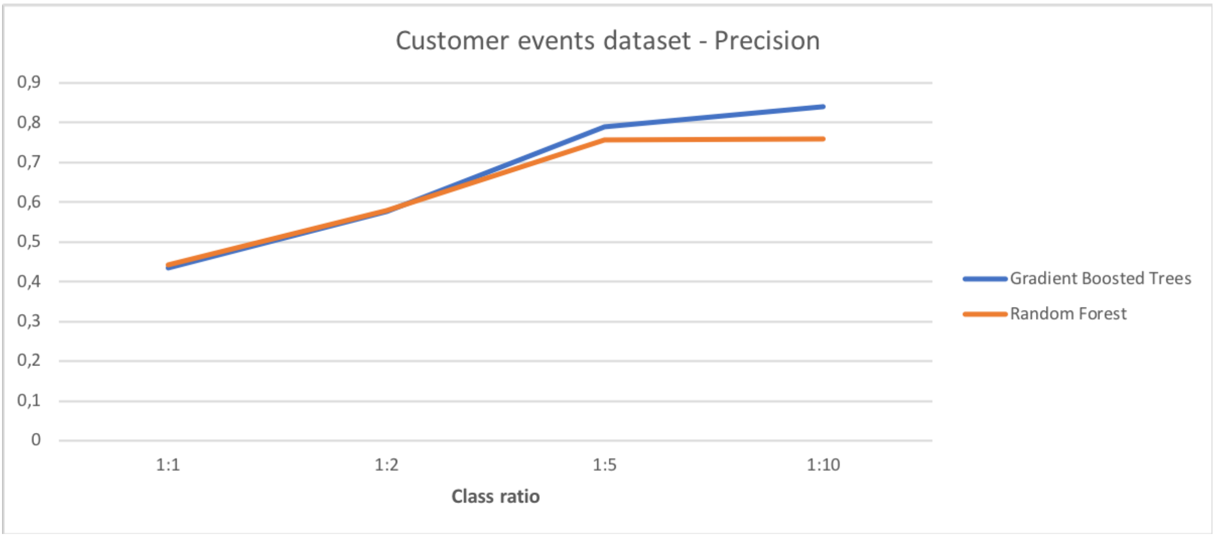


Figure 6.3: Precision as a function of purchaser class distribution ratios for the customer event logs dataset with timeline length of the 50 most recent events using under-sampling

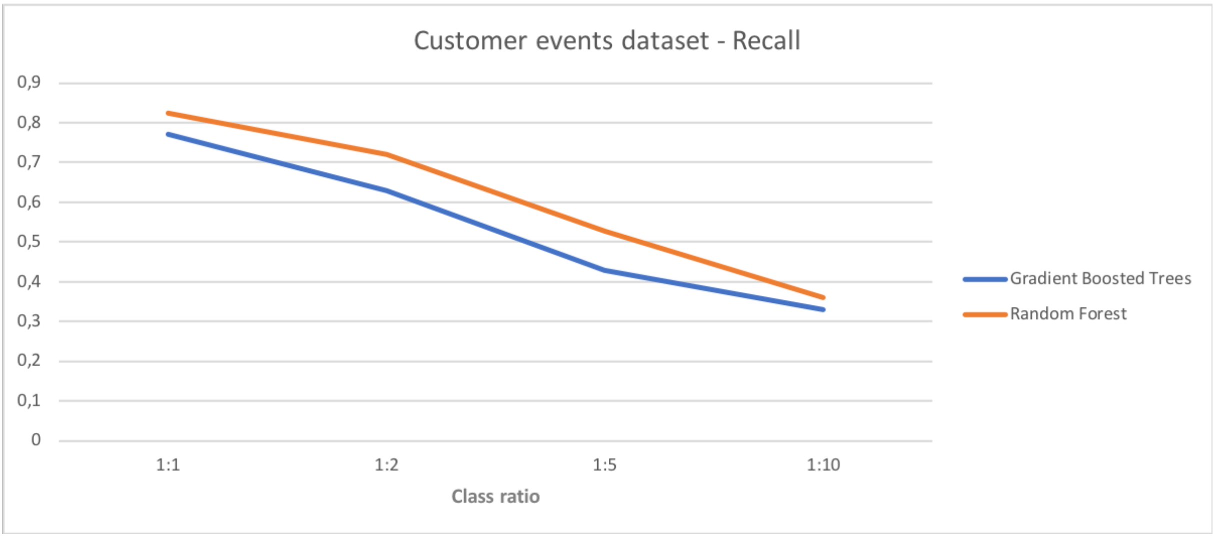


Figure 6.4: Recall as a function of purchaser class distribution ratios for the customer event logs dataset with timeline length of the 50 most recent events using under-sampling

Pipeline B: Customer attributes dataset

Ratio	$P_{\text{purchaser}}$	$R_{\text{purchaser}}$	$F_{1\text{ purchaser}}$	$P_{\text{not purchaser}}$	$R_{\text{not purchaser}}$
1:1	0.31	0.75	0.44	0.95	0.74
1:2	0.41	0.54	0.47	0.92	0.92
1:5	0.73	0.26	0.38	0.89	0.99
1:10	0.78	0.24	0.37	0.89	0.99

Table 6.5: Gradient Boosted Trees performance on the customer attributes dataset using various class distribution ratios by under-sampling

Ratio	$P_{\text{purchaser}}$	$R_{\text{purchaser}}$	$F_{1\text{ purchaser}}$	$P_{\text{not purchaser}}$	$R_{\text{not purchaser}}$
1:1	0.30	0.80	0.44	0.96	0.70
1:2	0.48	0.56	0.52	0.93	0.90
1:5	0.76	0.32	0.45	0.90	0.98
1:10	0.80	0.31	0.45	0.90	0.99

Table 6.6: Random Forest performance on the customer attributes dataset using various class distribution ratios by under-sampling

Ratio	$P_{\text{purchaser}}$	$R_{\text{purchaser}}$	$F_{1\text{ purchaser}}$	$P_{\text{not purchaser}}$	$R_{\text{not purchaser}}$
1:1	0.44	0.49	0.46	0.92	0.90
1:2	0.53	0.36	0.43	0.90	0.95
1:5	0.54	0.30	0.39	0.90	0.96

Table 6.7: Gradient Boosted Trees performance on the customer attributes dataset using various class distribution ratios by over-sampling (SMOTE). Ratio of 1:10 is omitted as it is not achievable without under-sampling

Ratio	$P_{\text{purchaser}}$	$R_{\text{purchaser}}$	$F_{1\text{ purchaser}}$	$P_{\text{not purchaser}}$	$R_{\text{not purchaser}}$
1:1	0.61	0.46	0.52	0.92	0.95
1:2	0.71	0.37	0.49	0.91	0.98
1:5	0.79	0.31	0.45	0.90	0.99

Table 6.8: Random Forest performance on the customer attributes dataset using various class distribution ratios by over-sampling (SMOTE). Ratio of 1:10 is omitted as it is not achievable without under-sampling

From table 6.5, 6.6, 6.7 and 6.8 we see that under-sampling produces the highest recall scores for purchasers. The results further indicate that over-sampling using SMOTE achieves a considerably lower recall score than under-sampling at a ratio of 1:2 and 1:1.

The precision and recall of the under-sampled models are shown in figures 6.5 and 6.6. We notice the same trend as we saw in the customer event logs dataset. Thus, we observe that one can trade precision for recall by under-sampling the majority class and force a more equal class distribution for the target label.

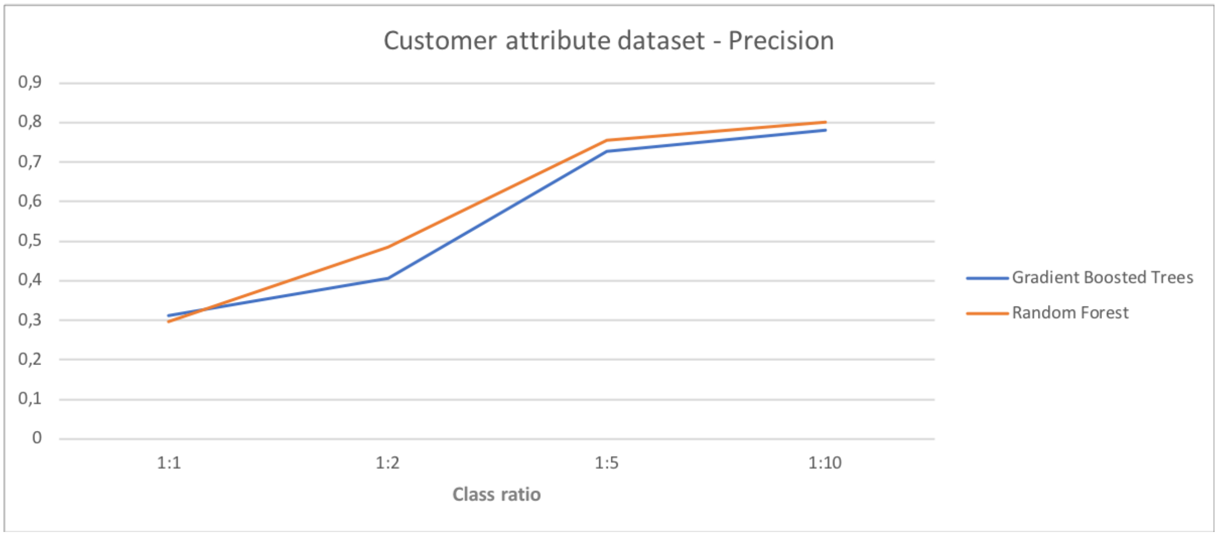


Figure 6.5: Precision as a function of purchaser class distribution ratios for the customer attributes dataset using under-sampling

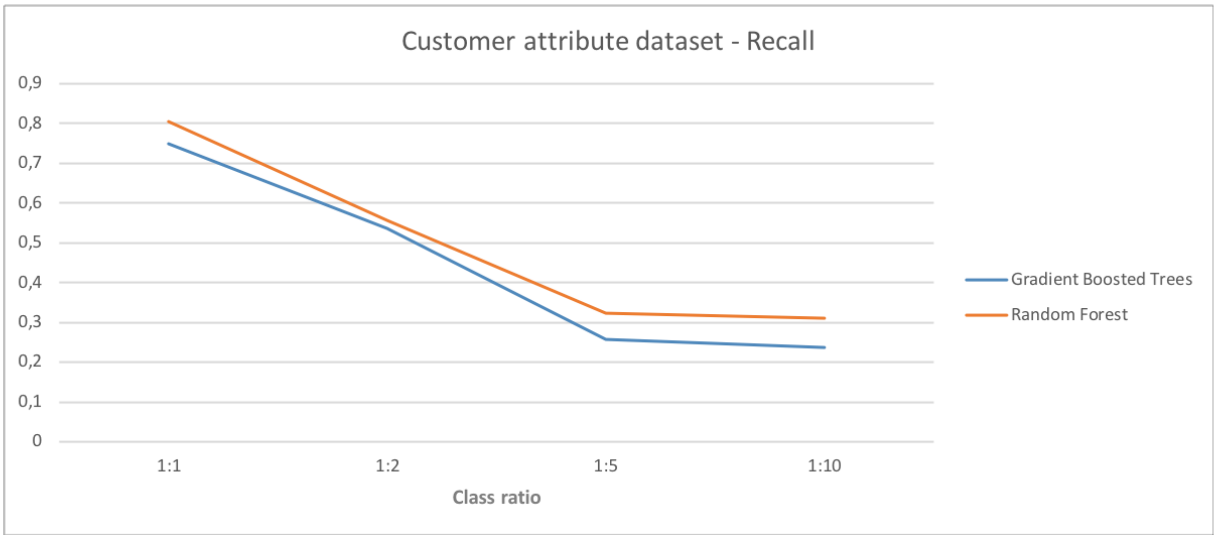


Figure 6.6: Recall as a function of purchaser class distribution ratios for the customer attributes dataset using under-sampling

6.1.3 Feature selection and correlation filtering

In an attempt to reduce the dataset's size to reduce training time we apply forward feature selection and correlation filtering.

A.5

Not applicable, see section 5.4.5.

B.5

We perform feature correlation analysis and forward feature selection for the customer attributes dataset. This was applied to the under-sampled customer attributes dataset with a 1:1 ratio of purchaser to non-purchasers. The correlation values between each column can be seen in figure 6.7.

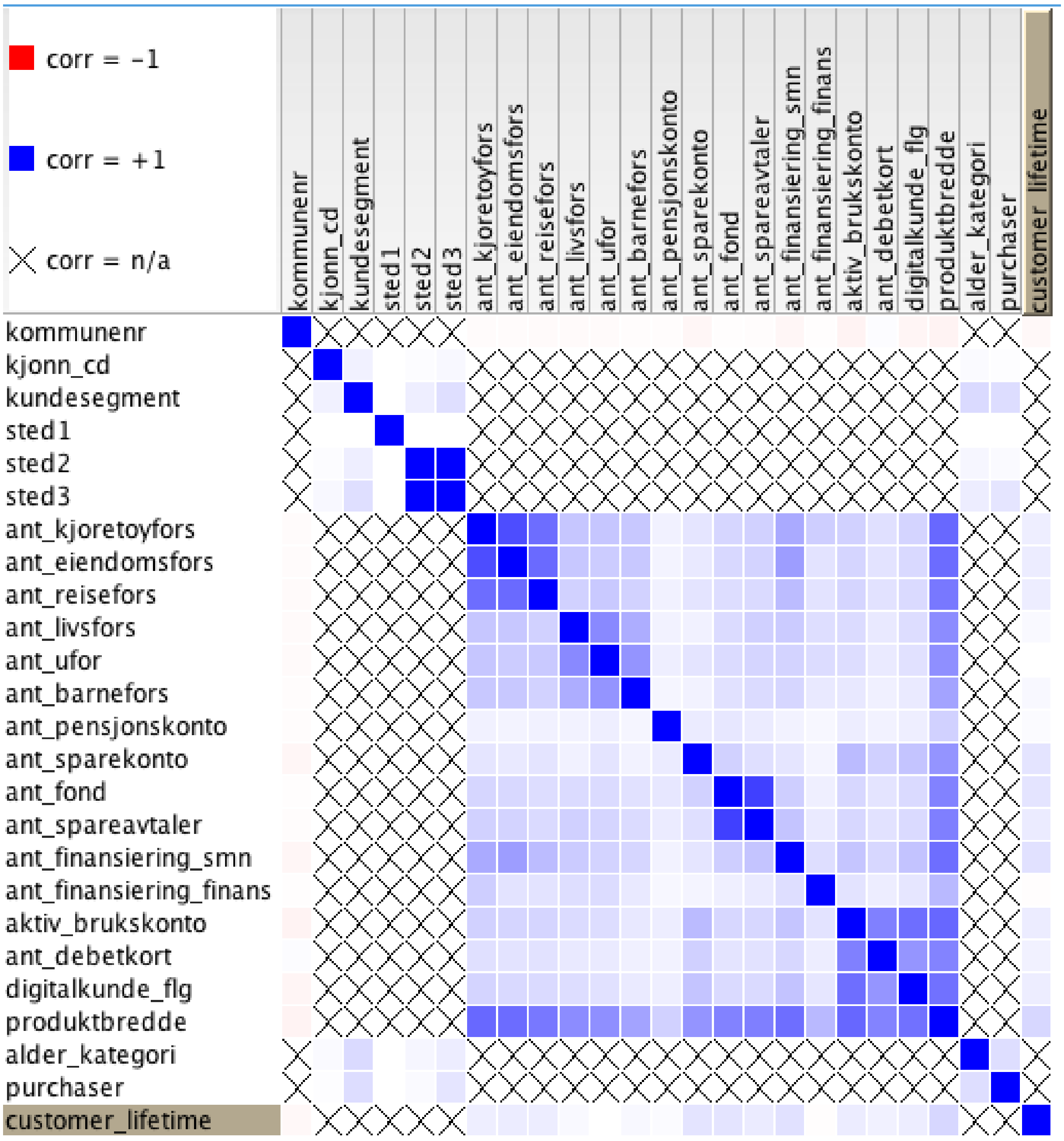


Figure 6.7: Correlation analysis of features in the customer attributes dataset

We see that `sted 2` (location 2) and `sted 3` (location 3) are perfectly correlated. We only remove `sted 2` as there are no other pairs of columns with a correlation coefficient above 0.9, a commonly used threshold.

The results from forward feature selection shows that we can remove the following features with negligible loss of recall,

- `sted 1` (Location 1)
- `ant_livsfors` (Number of life insurances)
- `ant_barnefors` (Number of child insurances)
- `ant_pensjonskonto` (Pension savings account)
- `ant_finansiering_finans` (Private financing)
- `digitalkunde_flg` (Digital customer)

The scores after correlation filtering and forward feature selection can be seen in table 6.9. The recall for purchasers is reduced by 3.5% compared to the non-feature-reduced model from the previous step, while the number of columns is reduced from 23 to 16 (30% reduction).

$P_{\text{purchaser}}$	$R_{\text{purchaser}}$	$F_{1\ \text{purchaser}}$	$P_{\text{not purchaser}}$	$R_{\text{not purchaser}}$
0.30	0.78	0.43	0.95	0.71

Table 6.9: Random Forest with correlation filter and forward feature selection on the customer attributes dataset with 1:1 class distribution ratio using under-sampling

6.1.4 Output models

A comparison between the baseline and the final models of `Pipeline A` and `B` can be seen in table 6.10. `Pipeline A` shows a 144.44% increase in recall when comparing the baseline results to the final model (Model A). The final model of `Pipeline B` increases the recall by 161.30%. We observe that the customer event logs dataset produced a higher performing model for both recall and precision.

	$P_{\text{purchaser}}$	$R_{\text{purchaser}}$	$F_{1\ \text{purchaser}}$	$P_{\text{not purchaser}}$	$R_{\text{not purchaser}}$	$\Delta R_{\text{purchaser}}$
Baseline A	0.76	0.36	0.49	0.91	0.98	
Model A	0.43	**0.88**	0.58	0.98	0.81	144.44%
Baseline B	0.80	0.31	0.45	0.90	0.99	
Model B	0.29	0.81	0.43	0.96	0.68	161.30%

Table 6.10: Final model performance using Random Forest. Customer event logs model (`Pipeline A`) defined as *Model A*, customer attributes model (`Pipeline B`) defined as *Model B*

6.1.5 Combining Model A and B

The models produced by `Pipeline A` and `B` were combined by taking the output prediction and confidence scores from both models and using them as input to train *Model C*, seen in table 6.11.

Algorithm	$P_{\text{purchaser}}$	$R_{\text{purchaser}}$	$F_{1\text{ purchaser}}$	$P_{\text{not purchaser}}$	$R_{\text{not purchaser}}$
Gradient Boosted Trees	0.80	0.69	0.74	0.91	0.88
Random Forest	0.85	**0.74**	0.79	0.96	0.98
Decision Tree	0.80	0.73	0.76	0.96	0.97
Naive Bayes	0.83	0.70	0.76	0.95	0.98

Table 6.11: Model C baseline performances using 10-fold cross validation

We observe that Random Forest has the highest scores and proceed using this algorithm. The resulting table shows our final and concluding results of `Pipeline A` and `B`, and the combined Model C. The final performance of Model C where all models were run against the unseen validation set is shown in table 6.12. This means that performance scores will slightly deviate from performance scores in the previous steps using 10-fold cross validation.

Model	$P_{\text{purchaser}}$	$R_{\text{purchaser}}$	$F_{1\text{ purchaser}}$	$P_{\text{not purchaser}}$	$R_{\text{not purchaser}}$
A	0.43	**0.88**	0.58	0.98	0.81
B	0.29	0.81	0.43	0.96	0.68
C	0.80	0.75	0.77	0.96	0.97

Table 6.12: Final performance for Models A, B, and C

We observe that Model A has the highest recall. The combined model achieves lower recall for purchasers than Models A and B, but considerably higher precision. This can be due to not under-sampling and forcing a 1:1 class ratio when training Model C. Higher recall values for Model C could possibly be achieved by applying the same under-sampling ratio as done on the datasets that were used for training Model A and B. Figure 6.8 plots the scores from table 6.12, while figure 6.9 shows the ROC-curves of all three models.

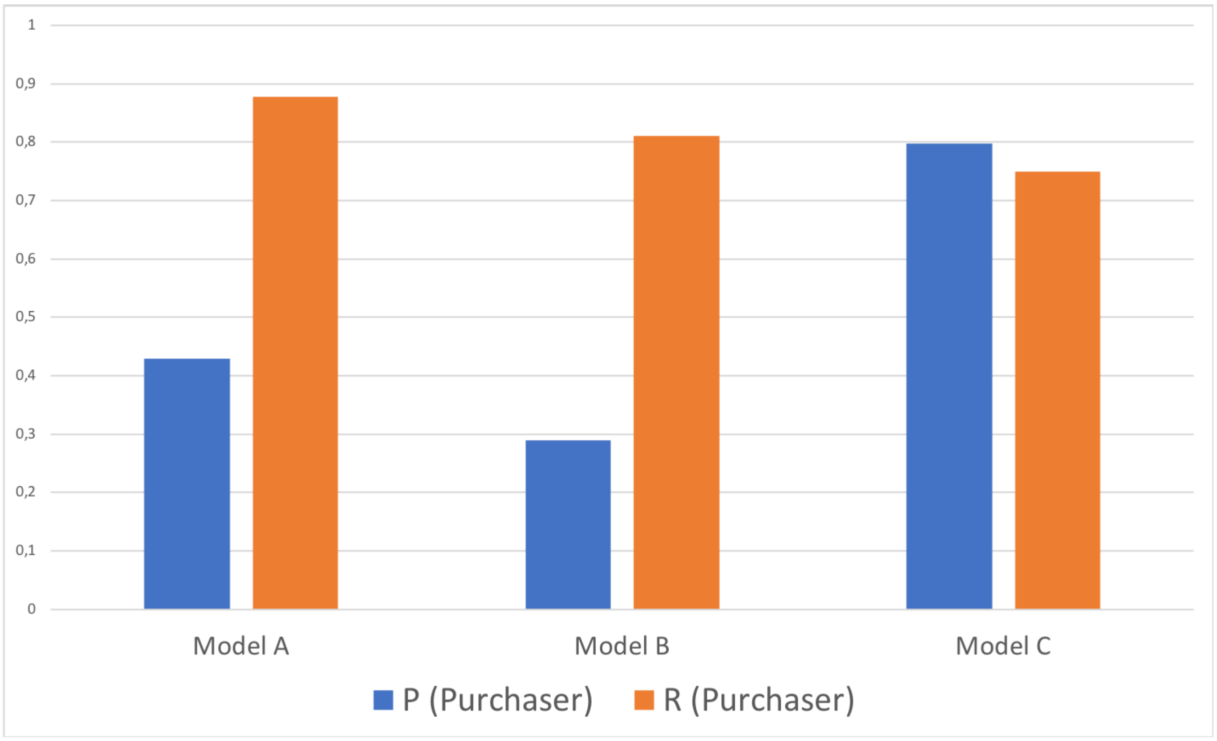


Figure 6.8: Comparison of precision and recall for purchasers for Models A, B, and C

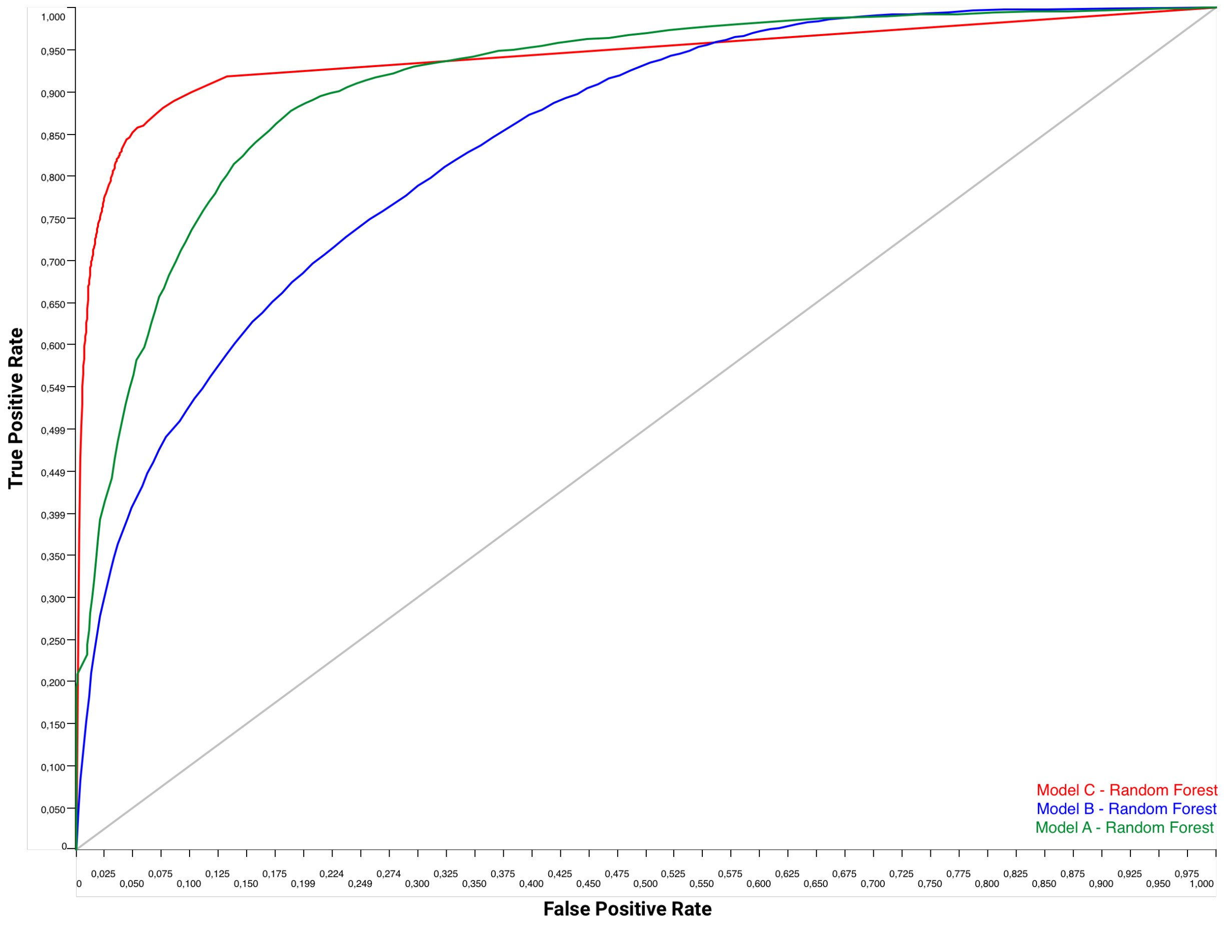


Figure 6.9: ROC-curves for Models A, B, and C for purchasers

6.1.6 Analysing customers with high confidence predictions

After evaluating Model C we end up with predictions for 59 541 customers, 7253 purchasers and 52 288 non-purchasers. In addition to the predicted target label we get a confidence score which is the probability of a customer belonging to the predicted class. We split these results into two partitions where `partition 1` contains all customers with predictions of confidence = 1 and `partition 2`, confidence < 1. We are unable to further segment the confidence scores because 81.3% of the customers already belong to `partition 1`. These are the customers the model is most confident will be predicted correctly.

Partition	No. of customers	% of customers	No. of actual purchasers	% of actual purchasers
1 (conf = 1)	48 432	81.3%	3777	52%
2 (conf < 1)	11 109	18.7%	3476	48%

Table 6.13: Distribution of customers in validation set for `partition 1` and `partition 2`

`Partition 1` includes 52% of the customers that were labelled as actual purchasers in the validation set. The recall and precision scores for each partition are shown in table 6.14.

Partition	$P_{\text{purchaser}}$	$R_{\text{purchaser}}$	$F_{1\ \text{purchaser}}$	$P_{\text{not purchaser}}$	$R_{\text{not purchaser}}$
1 (conf = 1)	0.96	0.82	0.88	0.99	0.99
2 (conf < 1)	0.74	0.64	0.69	0.76	0.84

Table 6.14: Recall and precision for predictions on `partition 1` and `partition 2`

We note that the precision and recall scores for both purchasers and non-purchasers are both considerably higher in `partition 1` than `2`. This is to be expected as the partition contains the most confident predictions. What is not obvious is that the set also contains a vast majority of the predictions.

Having partitioned the predictions by confidence score we would like to explore whether there are general statistical trends that differ between the two partitions. Gradient Boosted Trees and Random Forest do not provide the ability to generate a visual or interpretable graphical representation of their trained models such as Decision Trees.

Instead we can look at the feature importance of Model B as it is trained on the customer attributes and may help us discover what characterises the customers of `partition 1`. We select the three most important attributes from the feature importance illustration seen in figure 6.10 for further analysis.

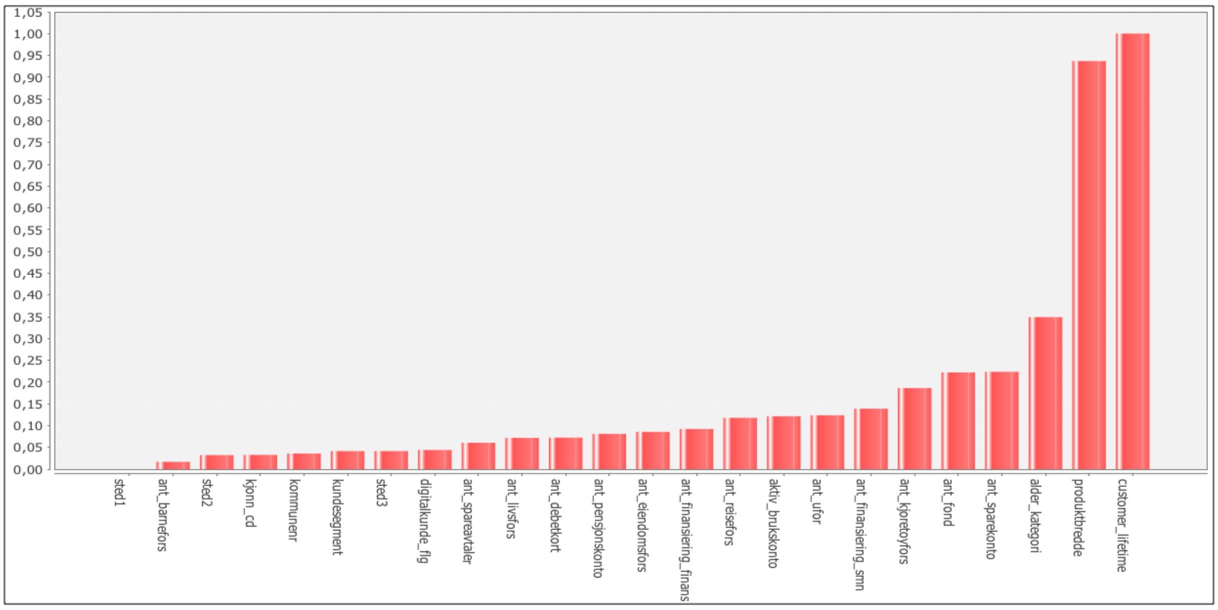


Figure 6.10: Feature importance of the customer attribute dataset, created using the entire development set seen in figure 5.1

Figure 6.10 shows how informative and important each feature is. Customer lifetime (how long the customer has been with the bank) and product breadth are clearly the most influential. Age category also stands out as important compared to the remaining features though it is considerably less influential than the top two. In addition we append the confidence scores from previous Models A and B for the same customers of both partitions. We then compare the differences of average values and distributions between both partitions, seen in table 6.15.

Metric	**Attribute**	**Partition 1**	**Partition 2**
Mean $\pm$ std	Model A confidence	$80.8\% \pm 13.6\%$	$71.6\% \pm 13.7\%$
Mean $\pm$ std	Model B confidence	$79.0\% \pm 16.9\%$	$70.8\% \pm 14,5\%$
Mean $\pm$ std	Customer lifetime (days)	7304 ± 4748	7033 ± 4606
Mean $\pm$ std	Product breadth	4.75 ± 2.90	5.38 ± 2.60
Mean $\pm$ std	Age	41.55 ± 15.41	42.08 ± 13.87

Table 6.15: A comparison of the distributions of a subset of attributes for `partition 1` and `2`. Age is calculated with the assumption that pre-aggregated buckets of age are defined as $\text{age}_{\text{bucket}} = \frac{\text{age}_{\text{upper bound}} - \text{age}_{\text{lower bound}}}{2}$. See figures 6.15 and 6.16 for the actual buckets

We observe higher average confidence levels from Model A and B in `partition 1` than in `partition 2`. This may indicate that Model C is trained correctly.

Customer lifetime distribution

There are no major differences between the general trend of the distributions for `partition 1` and 2 seen in figures 6.11 and 6.12. We observe some pairs of buckets that are more equal in size for `partition 1` such as $[0, 900]$ and $(900, 1800]$ [1]. Customers that have been with the bank for less than 900 days constitute 9%, about the same as customers with a lifetime between 900 and 1800. From table 6.15 we note that there is a difference of about 300 days in average lifetime between the partitions.

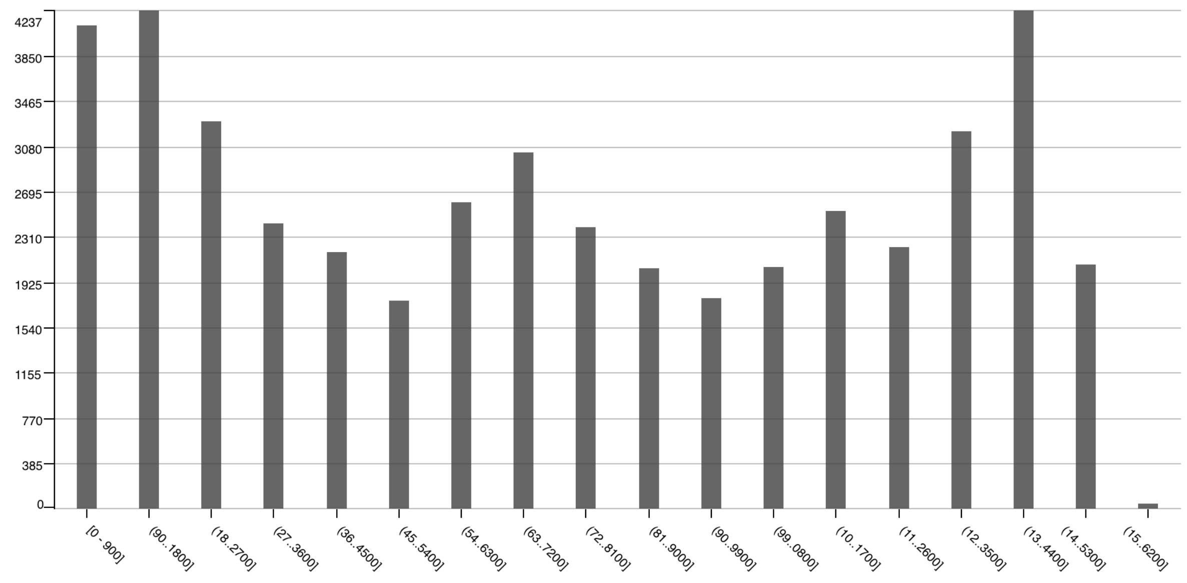


Figure 6.11: Customer lifetime distribution for customers in `partition 1`

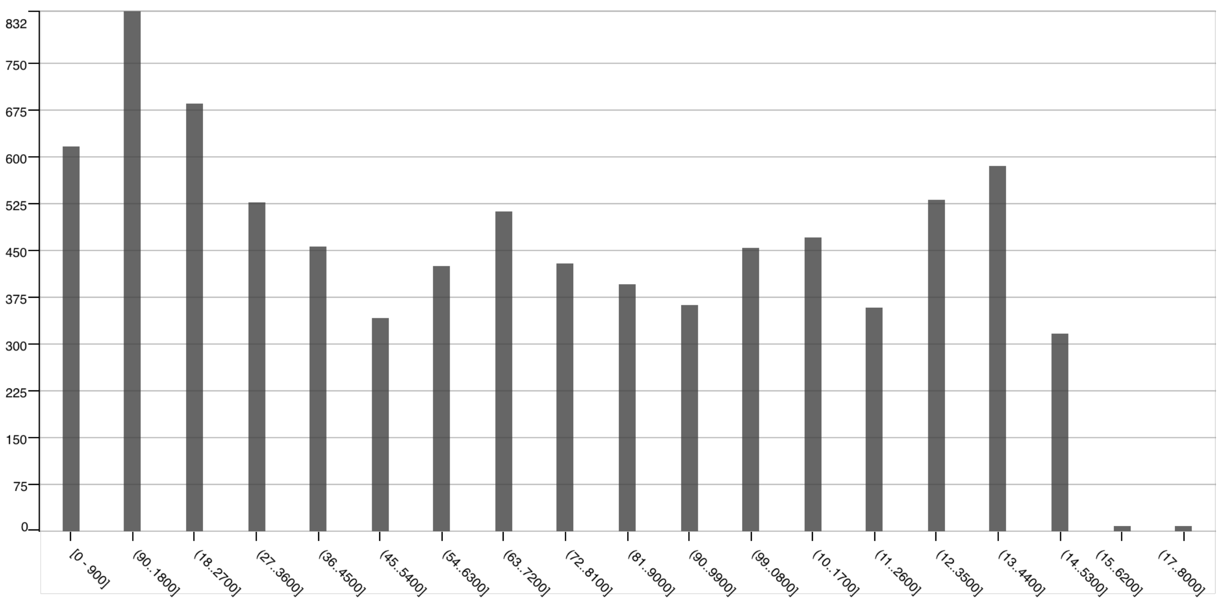


Figure 6.12: Customer lifetime distribution for customers in `partition 2`

[1]The graph exported from KNIME crops the number for the lower bound of each partition.

Product breadth distribution

We note from table 6.15 that the product breadth average of `partition 1` is lower than that of `partition 2` by almost one product. This is also reflected in the bucket sizes of $[0, 2]$, $(2, 4]$ and $(4, 6]$ for both partitions. In `partition 2` the $(4, 6]$-bucket contains more customers than any other bucket while in `partition 1` $(2, 4]$ is the largest.

The group of customers with less than five products constitute 51% of `partition 1`, but only 39% of `partition 2`. This is a notable difference and suggests that customers with relatively few products tend to lead to high confidence predictions.

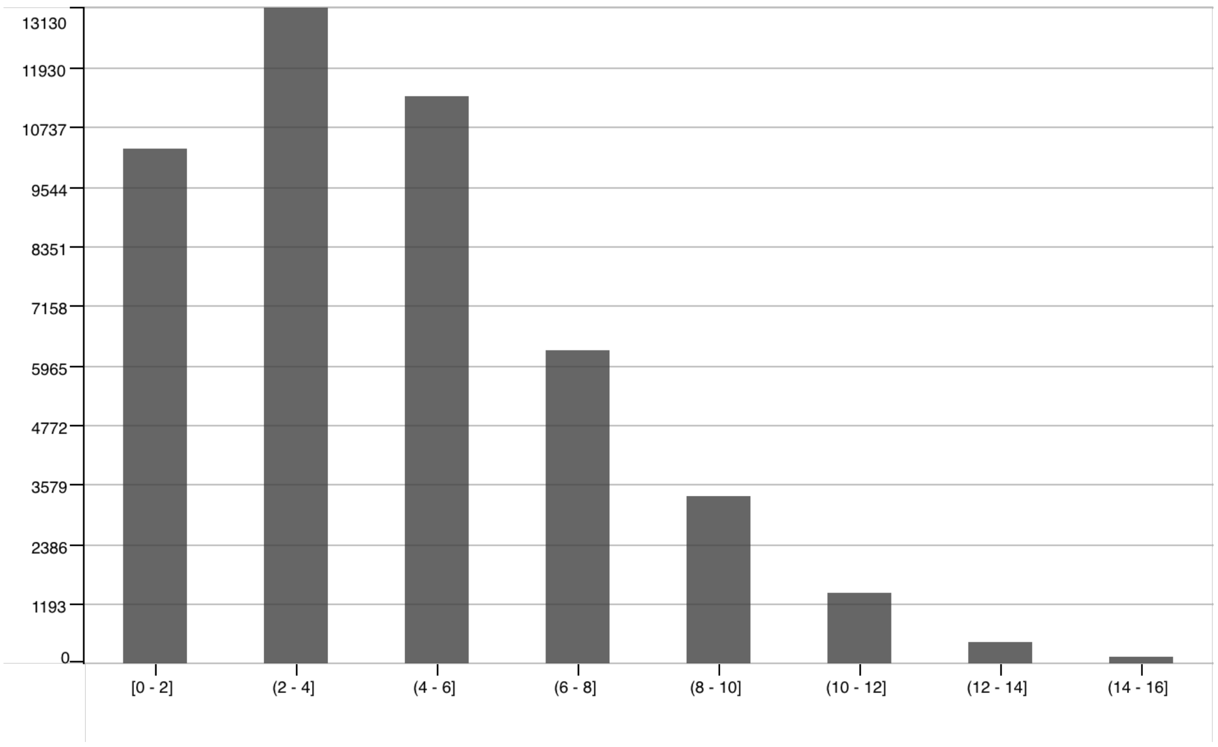


Figure 6.13: Product breadth distribution for customers in `partition 1`

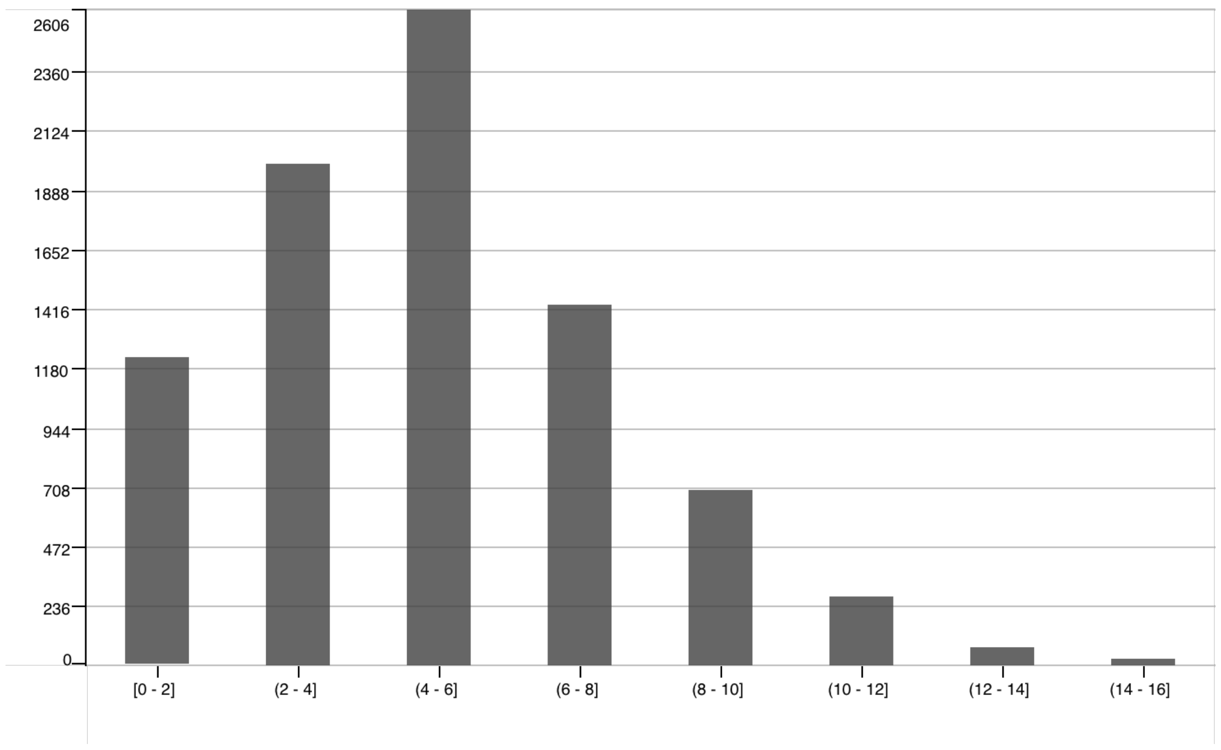


Figure 6.14: Product breadth distribution for customers in `partition 2`

Customer age distribution

We observe some interesting characteristics from figures 6.15 and 6.16. One of which is that the group of customers younger than 18 years is more represented in `partition 1` than `2`, likely due to being unable to own a credit card at this age. Thus, seeing a large portion of under-aged customers in `partition 1` and not in `partition 2` may indicate that our algorithm can successfully discern cases of under-aged customers not likely to buy one. However, one could argue whether or not it is prudent to include these customers in the training set when attempting to predict sales of credit cards. Both partitions have a mean age of around 42 years (see table 6.15).

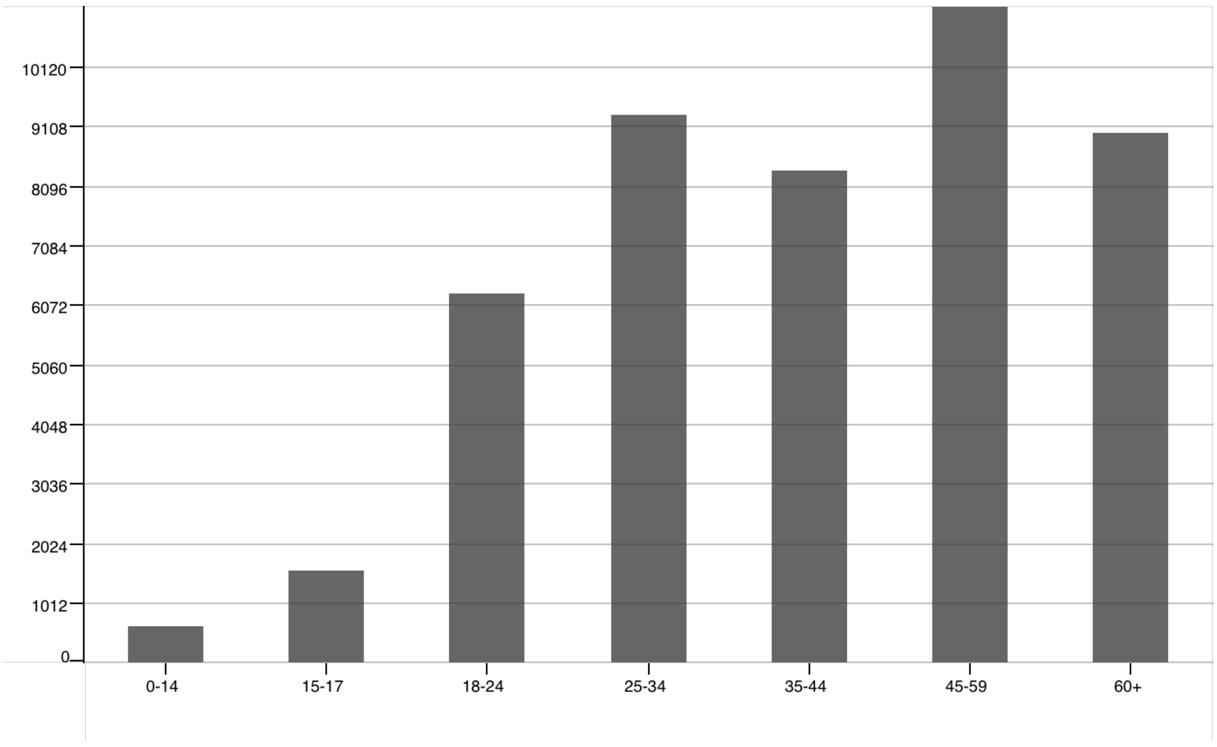


Figure 6.15: Customer age distribution for customers in `partition 1`

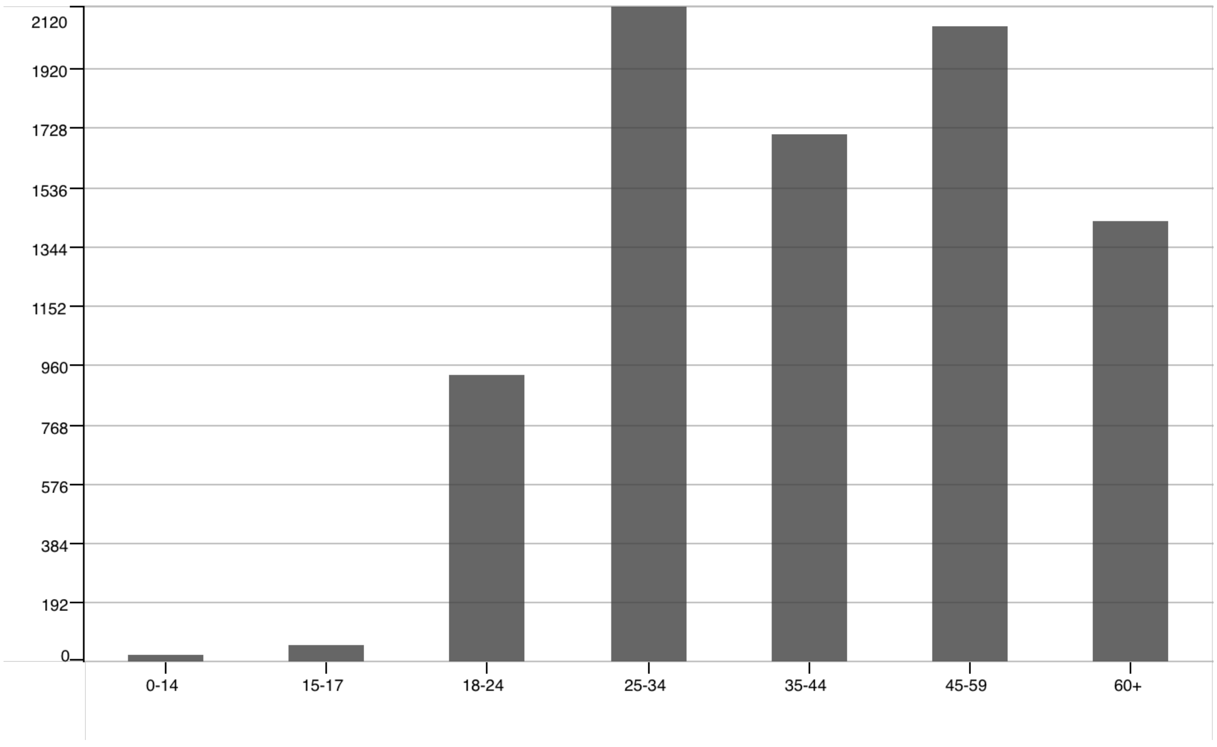


Figure 6.16: Customer age distribution for customers in `partition 2`

Our findings show that product breadth and customer lifetime are the two strongest predictors of a customer purchasing a credit card. Comparing the product breadth distribution between `partition 1` and `partition 2` we find that the group of customers with less than five products are more represented in `partition 1`. This means that customers with fewer than five products tend to lead to high confidence predictions.

We find little to no variation in distribution between the partitions for age and customer lifetime. In `partition 1`, 9% of the customers have been with the bank for less than 900 days. The corresponding percentage in `partition 2` is 7%. In terms of age we find that 14% of the customers in `partition 1` are between 18 and 24, compared to 11% in `partition 2`. Beyond this it is challenging to draw any conclusions on what characterises customers predicted with high confidence based on the customers' attributes.

Chapter 7

Discussion

This chapter provides a discussion of our results. The chapter is structured into several subsections discussing notable features and elements of our experiment and results.

7.1 Sequential events as training data

Throughout this experiment we have shown how various preprocessing techniques and different types of datasets can be applied to create models with high recall and precision for predicting likely credit cards purchasers. An interesting discovery is how the models perform on the two different datasets for customer event logs (`Pipeline A`) and customer attributes (`Pipeline B`). Most of the previous work on machine learning classification problems within the banking domain uses datasets that are similar to our customer attributes set. These types of datasets are perhaps the simplest to attain and have proven to produce useful models in multiple domains. The customer event logs dataset is fundamentally different as it contains sequential data in the form of customer timelines of events.

When we started out we wanted to measure the change in precision and recall as a function of the number of events (timeline length) of each customer. We also extracted timelines based on a set time period, such as the events for the last 1-12 months. A major challenge with the former approach was that highly active customers had up to 100 times as many events as the more inactive customers. This resulted in sparse datasets where a majority of the event columns contained zero-events (no event). We realised during the early stages that the models became biased, basically making their predictions off of how many zeroes each customer's timeline contains. In order to counter this, we applied a threshold to filter out the customers that had a ratio of over 70% and 80% zero-events. However, this discarded too many purchasers and reduced the class label distribution to an unacceptable ratio ($< 0.5\%$). We therefore choose to proceed with setting a fixed number of events to include for each customer's timeline. The resulting performance from training the models on timeline lengths of $n \in \{50, 100, 150, 200, 250, 300, 400, 500, 600, 700, 800, 900, 1000\}$ shows little to no change in the precision and recall scores except for Naïve Bayes.

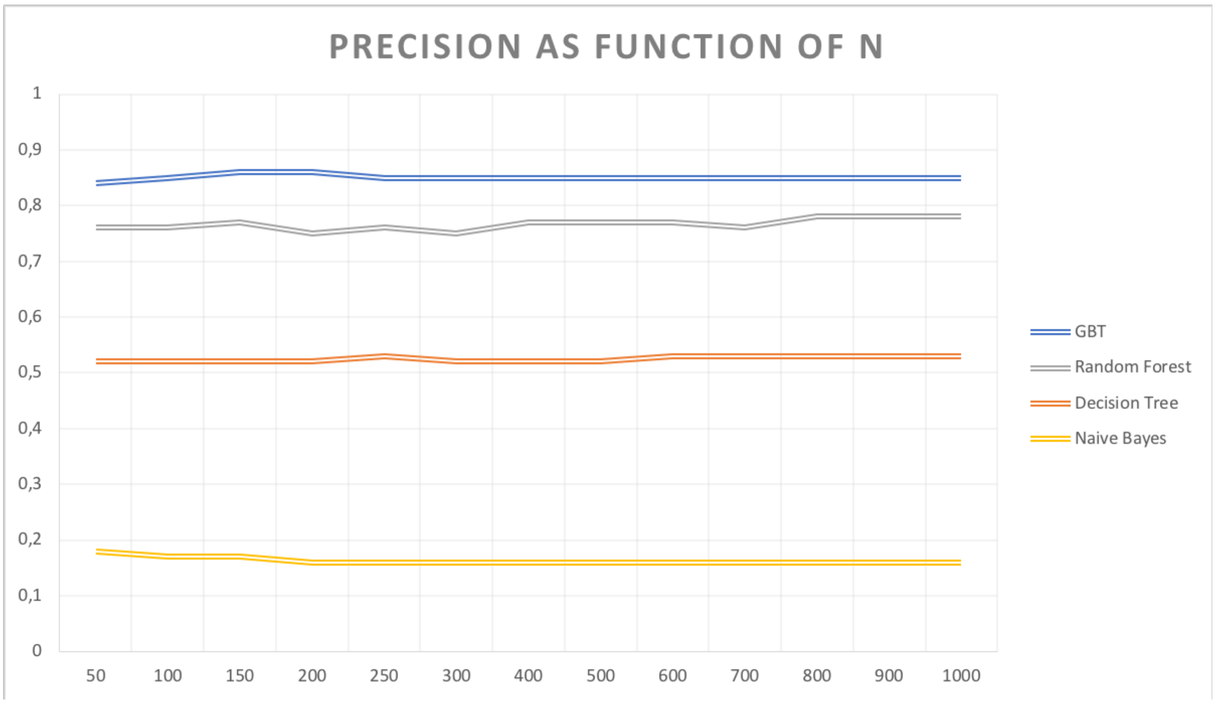


Figure 7.1: Pipeline A: Precision as a function of n (number of events in timeline)

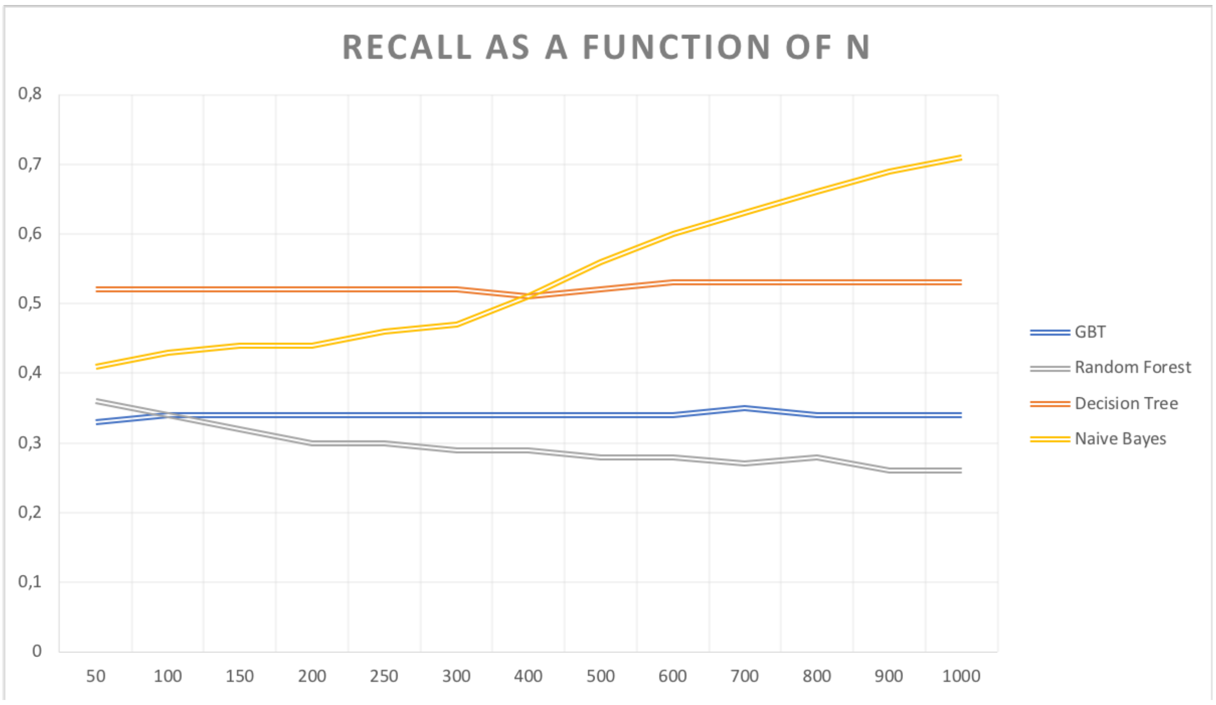


Figure 7.2: Pipeline A: Recall as a function of n (number of events in timeline)

We realise that figures 7.1 and 7.2 may represent a local minima and so we make no conclusions as to whether or not the scores could be improved by selecting longer customer timelines. Initially we expected this dataset to produce worse performing models compared to the customer attributes dataset. However, the performance turned out to be comparable and even better in some cases. From a human point of view, it appears unlikely that sequences of event codes (VYKODEs) such as `viewing account balances` and `confirming payments` can contain clues as to whether or not a customer is interested in attaining a credit card. Random Forest and Gradient Boosted Trees are able to produce models that make predictions exceeding guesses based on class label distribution probabilities. Due to the complex internals of RF and GBT it is hard to interpret exactly which events or subsets of events that increase the confidence of their predictions.

We also observe that the recall score for Naïve Bayes starts to rapidly increase after $n = 300$, seen in figure 7.2. The recall score both surpasses the other algorithms and behaves distinctly different. From figure 7.1 a corresponding change in precision is not observed. This behaviour stands out, requiring further analysis.

Due to the mechanics of Naïve Bayes we speculated that the left-padding of non-events as 0s might negatively impact the predictions. As the timeline length of customers grow, the number of customers requiring padding increases. This leads to event columns with a high ratio of zero-events. An example of this padding is shown in table 7.1 and 7.2. `Event 1` is the oldest event in the extracted timeline, while the highest event number is the most recent one. As the timeline length is increased, the number of zero-events (0s) in the `Event 1`-column increases. This is due to the fact that several customers do not have more events to include.

Customer id	Event 1	Event 2	Event 3
1	**0**	43	364
2	34	54	125
3	43	25	64

Table 7.1: Example customer event timeline of length $n = 3$. Columns are sequential events, while numbers other than 0 are mapped to an actual event code

Customer id	Event 1	Event 2	Event 3	Event 4	Event 5	Event 6
1	**0**	**0**	**0**	**0**	43	364
2	**0**	**0**	76	34	54	125
3	**0**	102	99	43	25	64

Table 7.2: Example customer event timeline of length $n = 6$

In figure 7.3 we plot the percentage of customers with a zero-event in the `Event 1`-column for each timeline length n. We observe that the percentage of customers with a zero-event reaches 50% at $n = 500$. This means that 50% of the factors in the Naïve Bayes probability formula will be calculated from the relation between a zero-event and the target label.

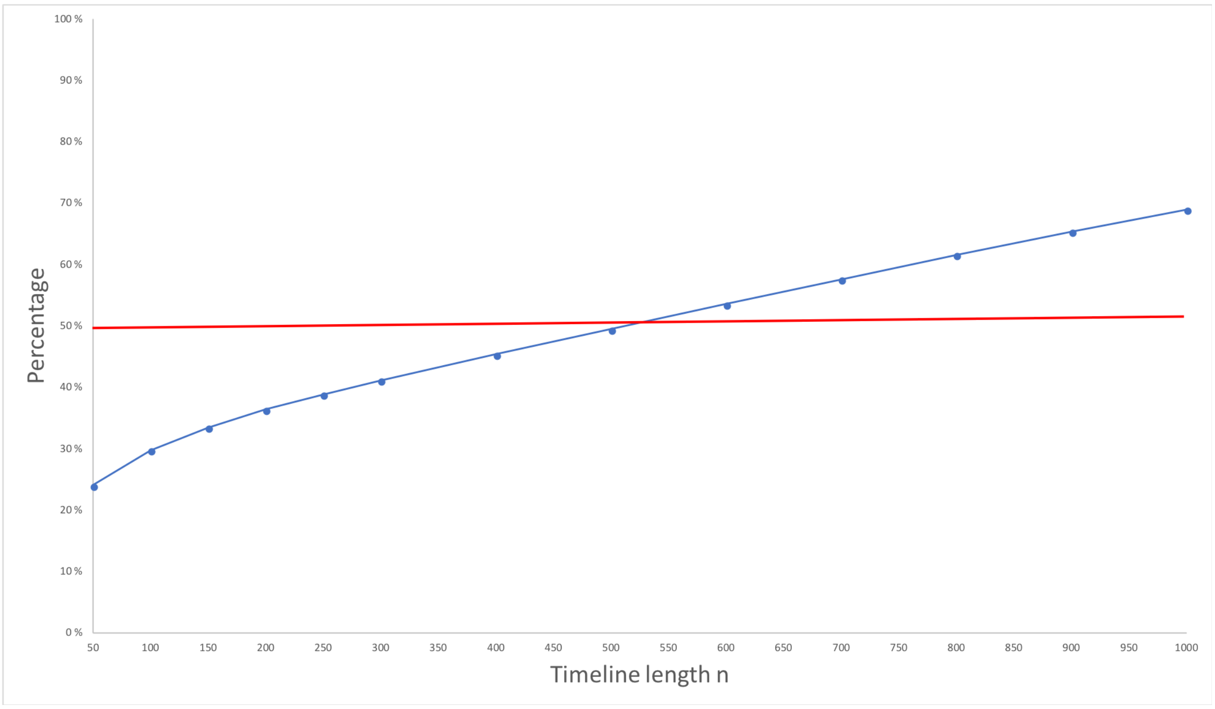


Figure 7.3: Percentage of customers with zero-events in the `Event 1`-column at different timeline lengths, n

From equation 7.1 we can assume that at least X_1 to $X_{\frac{n}{2}}$ is purely calculated from zero-events at timeline lengths $n \in [500, 1000]$. Based on this explanation we find it unlikely that the recall scores for $n \geq 500$ holds any real value.

$$P(X_1, X_2, ..., X_n|Y) = \prod_{i=1}^{n} P(X_i|Y). \tag{7.1}$$

7.2 Trading precision and recall using sampling

A common challenge when working with classification problems within the banking domain is handling datasets with a high degree of class imbalance. In our case the dataset contained 89.5% non-purchasers and 10.5% purchasers. This imbalance tends to introduce bias into the models which leads to favouring the majority class when making predictions. For our dataset, a trivial model predicting only the majority class would achieve an accuracy score of 89.5%. The impressive accuracy score is highly misleading and is not an appropriate metric to use as evaluation. Instead we use recall and precision for the purchaser class to evaluate our model. These metrics give a much better understanding of how well the models are performing in predicting purchasers.

As mentioned in section 5.1, optimising recall and precision serves different business goals. We work under the premise that maximising one of these metrics comes at the cost of reducing the other. This can be illustrated by altering the class label distribution. By under-sampling the majority class we can examine how the precision and recall balance is shifted at different ratios of purchaser to non-purchaser, seen in figures 7.4 and 7.5.

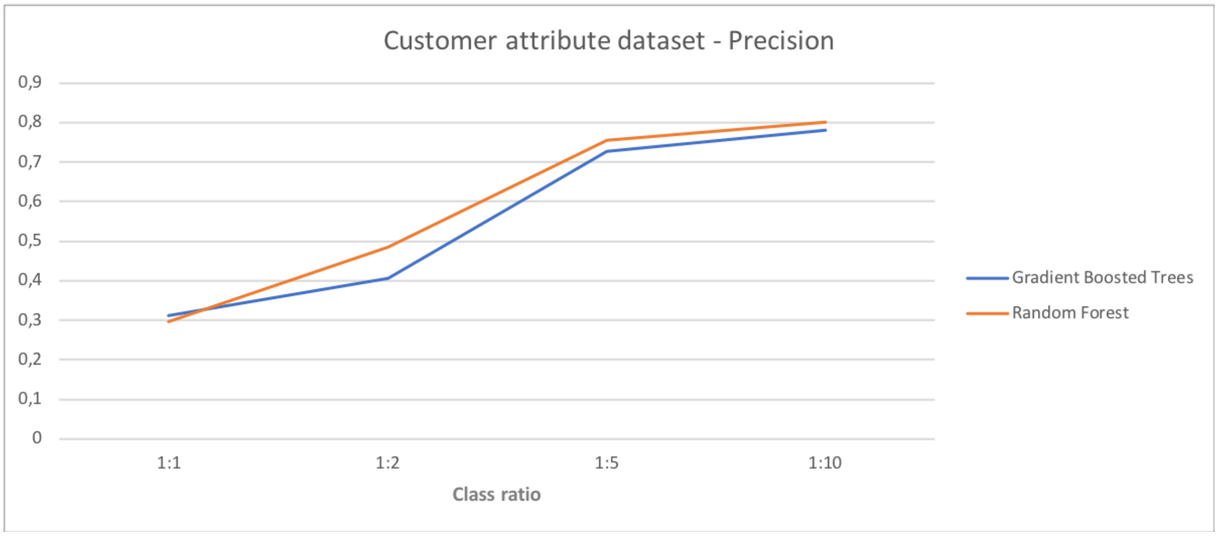


Figure 7.4: Precision scores at various ratios of purchaser to non-purchaser

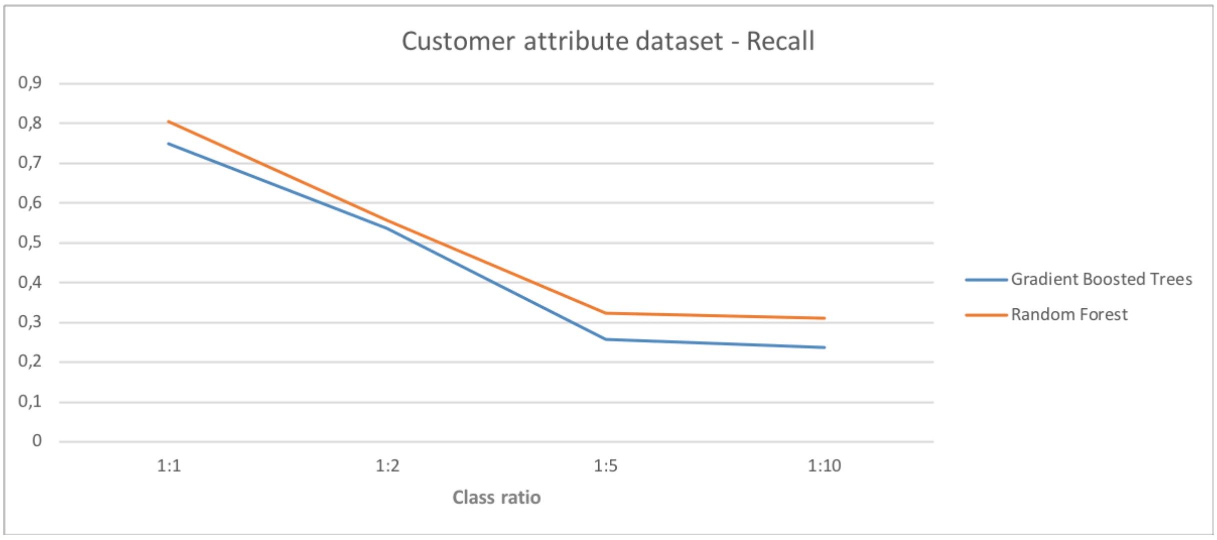


Figure 7.5: Recall scores at various ratios of purchaser to non-purchaser

When smoothing out the class imbalance we counter the majority bias introduced to the models. It is also possible to configure the parameters of each classification algorithm to weight the class labels unequally, but this requires algorithm-specific knowledge and tuning. We consider under-sampling to be a more general approach that appears to produce predictable effects with the algorithms applied in this project. The general trend we observe in this experiment is that by smoothing out the class imbalance we increase recall at the cost of precision. This observation enables us to maximise the metric that is more closely aligned with the business goals.

7.3 Selecting classification algorithm

One of the main goals for this project was to uncover which set of classification algorithms that perform best on our dataset from SpareBank 1 SMN. We chose to apply four algorithms,

- Naïve Bayes
- Decision Trees
- Random Forest
- Gradient Boosted Trees

Naïve Bayes and Decision Trees are simpler algorithms than Random Forest and Gradient Boosted Trees in terms of internal complexity. We expected that both Random Forest and Gradient Boosted Trees would outperform Naïve Bayes and Decision Trees which turned out to be partially correct. For both datasets GBT and RF achieved the highest precision scores while NB and DT achieved the highest recall. We selected the best performing algorithms based on the highest score of either precision or recall, as we anticipated the potential for trading off one metric for the other. We therefore ranked the algorithms by the following scoring function seen in equation 7.2.

$$\text{score}_{\text{algorithm}} = \max\left(\text{precision}_{\text{purchaser}}, \text{recall}_{\text{purchaser}}\right) \tag{7.2}$$

This lead to the selection of GBT and RF to build the final model for `Pipeline A` and `B`.

7.4 Combining Model A and B

`Pipeline A` and `B` output two models trained on the customer event logs and customer attributes dataset respectively. The models are trained on data from the same customers, but on different characteristics. The results from the different pipelines and combination can be seen in 7.6. We observe that the Models A and B have similar performance scores. This leads us to believe that the likelihood of a customer purchasing a credit card is reflected in both dimensions covering different characteristics.

With our primary goal of maximising recall, Model A is the highest scoring model. If SpareBank 1 were to launch marketing campaigns that would benefit from high recall scores, Model A would be the best fit. However, instead of completely discarding Model B we find that by combining the predictions and confidence scores from both models we are able to smooth out the imbalance between precision and recall. This may be desirable for some business goals such as in direct marketing approaches.

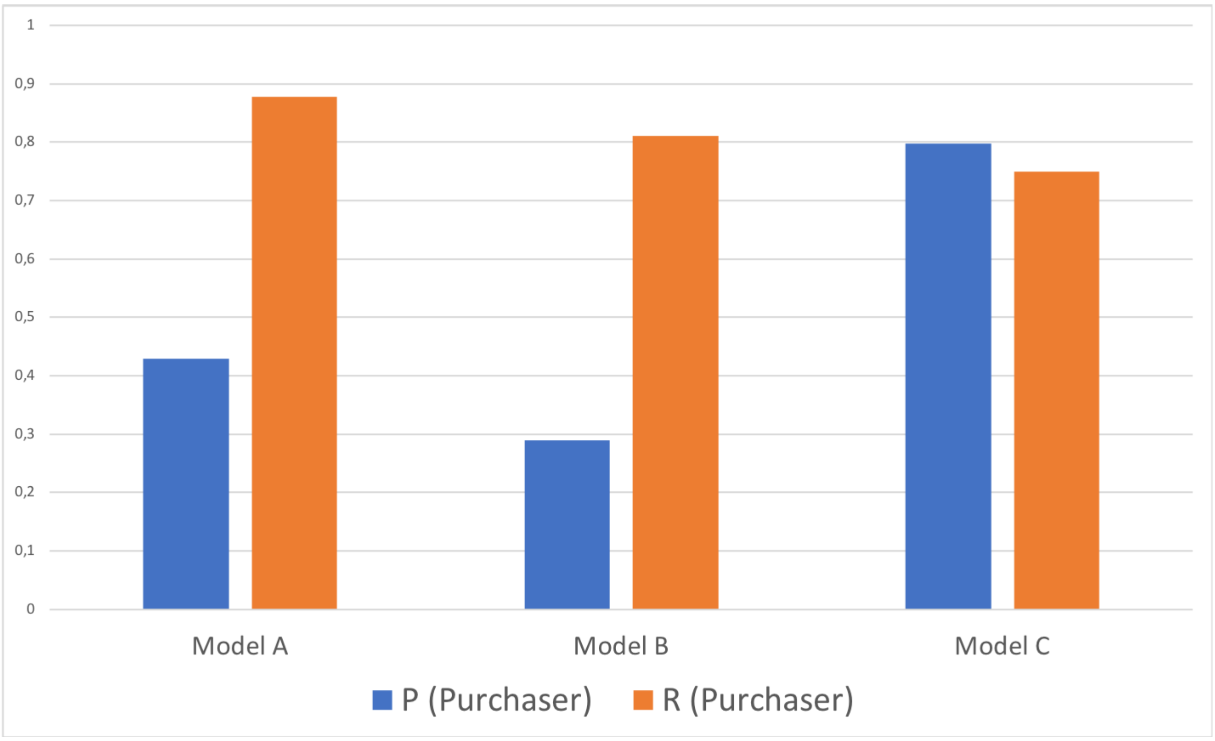


Figure 7.6: Comparison of precision and recall for Models A, B, and C

The recall score of Model C drops below both Model A and B, but we observe a considerable increase in precision. While this is not our primary goal, the increase in precision would allow us to further increase recall by applying the same under-sampling technique as we effectively did with Model A and B.

Moreover, we view Model C as a model that takes prediction "votes" from Model A and B along with their respective confidence scores, and then estimates a weighting scheme that finally settles the vote to a prediction. In table 7.3 we examine various combinations of correct and incorrect predictions between the models. These are the results from making predictions on the validation set containing 59 541 customers whereas 8289 are purchasers.

From table 7.3 we observe that if both Model A and B predict incorrectly, Model C also predicts incorrectly. However, in the scenario where Model A and B disagree, Model C attempts to settle the vote based on the confidence scores. There are 2170 (#2 + #3) cases where Model A and B disagree. 393 (#6 + #7) of these cases are settled correctly by Model C. In the case where both Model A and B predict correctly, we observe that there are 145 (#4 - #8) cases where Model C does not predict correctly. This is an unfortunate effect and we would suggest to only apply Model C if Model A and B do not make the same prediction.

#	**Model A**	**Model B**	**Model C**	**Number of cases**
1	Incorrect	Incorrect		208
2	Incorrect	Correct		809
3	Correct	Incorrect		1361
4	Correct	Correct		5911
5	Incorrect	Incorrect	Incorrect	208
6	Incorrect	Correct	Correct	110
7	Correct	Incorrect	Correct	283
8	Correct	Correct	Correct	5766

Table 7.3: Notable prediction combinations of all models for actual purchasers. Structured in the common binary truth table format

7.5 Selecting high confidence customers for marketing campaigns

Based on our findings in 6.1.6 it is hard to make any definitive conclusions about which customer attributes lead to a high confidence prediction. However, our findings suggest that customers with fewer than five products tend to lead to high confidence predictions. We recommend two guidelines when analysing the results of the predictions,

- Select only confidence = 1 predictions
- Only apply Model C when Model A and B do not agree in their predictions

Selecting confidence = 1 filters out 52% of purchasers, but improves the precision by 17% and recall by 10%. It is also possible for SpareBank 1 to further filter out customers that they consider to be more interesting, for instance, customers between 18-24 with a product breadth of 2-4. This can be useful for more expensive marketing campaigns where profit heavily relies on a high positive response rate from the customers. Also, cases #2+#3 and #6+#7 in table 7.3 indicate that Model C is able to resolve 393 cases out of 2170 otherwise unsettled predictions. As mentioned in the final paragraph of section 7.4, applying Model C when Model A and B agree may lead to incorrect predictions.

7.6 Comparison to related work

"**Predicting credit card customer churn in banks using data mining**" by Kumar and Ravi [24] aims to develop models that predict customer credit card churn. It is important to note that the business goals in our experiment and the study are different, but as we mentioned in section 3.1 the solution is implemented in a similar fashion. The study uses a dataset similar to the customer attributes dataset used in our experiment. However, our dataset contains about 200 000 customers compared to their 14 814. In addition we have a slightly less imbalanced dataset with a minority class of 10.5% to their 6.8%.

The only classifiers we share with the study are Random Forest and Decision Trees. Similar to our sampling approach, Kumar and Ravi [24] experiment with various class distribution ratios, using SMOTE and under-sampling. The highest increase in recall for the minority class is 23%, observed at a ratio of 1:3 churner to non-churner using Random Forest and under-sampling. The recall decreases when the ratio of churner to non-churner shifts from 1:3 to 1:6. This is similar to the behaviour we experienced in our project. However, we are unable to compare the change in precision as it is not included in the study. Table 7.4 shows a comparison between the models produced by Kumar and Ravi [24] and `Pipeline B`. The minority to majority class ratios are also displayed.

Model and class ratio	$R_{\text{minority class}}$	$\Delta R_{\text{purchaser}}$
Baseline B (1:10)	0.31	
Model B (1:1)	0.81	**162%**
Kumar and Ravi (1:13)	0.62	
Kumar and Ravi (1:3)	0.76	**23%**

Table 7.4: Comparison of recall in the minority class for under-sampled Random Forest models in Kumar and Ravi [24] and Model B (customer attributes dataset)

It is important to note that our baseline model initially achieves a much lower recall score than the baseline models by Kumar and Ravi [24]. This is an interesting result as their dataset is even more imbalanced than the one we use for `Pipeline B`. We speculate that the reason for this may be that some features of their dataset are very strong predictors of churning customers, or that the entire dataset is inherently more fit for the prediction problem. The study does not apply under-sampling at a ratio of 1:1 which potentially could have increased the recall score further. Thus, a direct comparison of the increase in recall between the experiments may not be suitable. However, we do note that under-sampling to smooth out the class imbalance yields higher recall scores in both experiments.

"**Prediction of customer attrition of commercial banks based on SVM model**" by Benlan et al. [30] examines the use of Logistic Regression (LR) and Support Vector Machines (SVM) to predict bank customer churn. The study's dataset consists of 50 000 customers with a churner to non-churner ratio of close to 1:109. This class imbalance is considerably larger than that of the dataset used in our experiment. As we do not apply LR or SVM, we will only compare how under-sampling affects the recall of the minority class. The precision and recall scores achieved by Benlan et al. [30] using Logistic Regression for various class ratios can be seen in table 7.5.

Ratio	$P_{\text{minority class}}$	$R_{\text{minority class}}$
1:109		0
1:10	0.238	0.388
1:5	0.124	0.496
1:2	0.049	0.734
1:1	0.034	0.861
2:1	0.024	0.941

Table 7.5: Recall and precision scores for churners using Logistic Regression for various class distribution ratios using under-sampling in Benlan et al. [30]

The baseline model is trained using the non-sampled dataset at a ratio of 1:109 churner to non-churner. At this ratio the model is unable to identify any churning customers because of the majority bias introduced by the class imbalance. Furthermore, we observe the same trend as in our experiment where recall for the minority class increases as the ratio becomes more balanced. We also note that this comes at the cost of decreased precision. The highest recall value the study achieved was 0.94 at a ratio of 2:1 churners to non-churners.

A comparison between the final model from `Pipeline B` and the similar under-sampled models by Benlan et al. [30] is shown in table 7.6. At 1:1 minority to majority ratio, the recall scores are close to equal, slightly favouring the model by Benlan et al. [30]. However, our model vastly outperforms the study's model in terms of precision and consequently the F_1-measure. The study has an increase in recall of 121% from altering the class distribution ratio from 1:10 to 1:1. This is comparable to our increase of 162% for the same change in ratio. The corresponding precision decreases by 64% in our experiment and 86% in the study by Benlan et al. [30].

Model and class ratio	$P_{\text{minority class}}$	$R_{\text{minority class}}$	$F_{1\ \text{minority class}}$
Benlan et al. [30] (2:1)	0.02	0.94	0.05
Benlan et al. [30] (1:1)	0.03	0.86	0.07
Model B (1:1)	0.29	0.81	0.43

Table 7.6: Comparison of recall and precision scores for the minority class using under-sampled Logistic Regression models in Benlan et al. and Model B

Chapter 8

Conclusions and further work

This chapter answers the research questions posed in section 1.2, followed by a brief discussion of the project's value to SpareBank 1 SMN. Finally we outline some suggestions for further work.

8.1 Research questions

What preprocessing strategies or steps have the greatest impact on sales prediction from financial data?

We applied various steps for two different datasets; customer attributes and customer event logs. The customer attributes dataset was subjected to basic manual feature reduction and cleaning, over- and under-sampling, correlation filtering, and forward feature selection. We found that under-sampling the purchaser to non-purchaser ratio to 1:1 had the greatest impact on recall for purchasers. This increased the score by 161.30% from the non-sampled baseline. We also found that by altering the class distribution ratio we were able to control the balance between precision and recall.

The customer event logs dataset was transformed into customer timelines with varying length. Our findings suggest that the length of a timeline has little to no impact on model performance. We observed that under-sampling had the same effect as with the customer attributes dataset, increasing the recall score by 144.44%. Finally, we combine the two models trained on the two datasets and observe an increase in precision with slight loss of recall. Thus achieving a considerably higher F_1 score compared to either model in isolation.

How do standard machine learning approaches perform in financial sales prediction and what explains the differences in results?

We applied four classification algorithms, Gradient Boosted Trees, Random Forest, Decision Trees and Naïve Bayes, to both the customer attributes dataset and the customer event logs dataset. We observed that Gradient Boosted Trees and Random Forest had greater performance than Decision Trees and Naïve Bayes according to our evaluation criteria. The bagging and boosting techniques applied in Random Forest and Gradient Boosted Trees likely captures the complex patterns of the datasets more effectively than the less sophisticated algorithms. The results suggest that these algorithms better handle the issue of class imbalance. Our two final models, Model A and B, are under-sampled to a ratio of 1:1 and trained using Random Forest. Model A achieved the highest recall score of 0.88 and a precision score of 0.43. Model B achieved a recall score of 0.81 and a precision score of 0.29. The final combined model, Model C, achieved more balanced scores with a recall score of 0.75 and a precision score of 0.80.

What characterises users for whom sales can be predicted with a high degree of confidence?

From the predictions made by the combined model on the validation set, we partition predictions of customers into sets of high and lower confidence. The model produced from the customer attributes dataset marks product breadth and customer lifetime as the strongest predictors. Comparing the average values and distributions of these features, we find that the group of customers having less than five active products constitute a large portion of the high confidence predictions. Furthermore, we find little to no variation in distribution between the partitions for age and customer lifetime. It is challenging to draw any definite conclusions on what characterises customers predicted with high confidence based on the customers' attributes.

8.2 Value to SpareBank 1 SMN

Several fields within the banking domain are today automated or on the verge of being automated. Domains such as fraud detection, anti money laundering, and vetting customers for credit card acquisition have all been subjected to early adaptations of machine learning techniques leading to a higher precision than before. All of the aforementioned domains were previously based on the conclusions of one or more employees who manually sifted through lists and accounts - a very cumbersome process. With the advances in the field of machine learning these domains have become more efficient than ever before, increasing profitability and saving man hours.

SpareBank 1 did not expect we would achieve the results that we did, and representatives were pleased with the quality of our results. In particular, our findings suggest that the customer event logs contain valuable information that should not be discarded in the process of predicting product sales. Our results show that model performance scores within product prediction are comparable to the scores of state-of-the-art methods found within churn prediction research. Although comparisons across different datasets are innately difficult, preprocessing steps such as sampling produce interesting findings like being able to control the balance of precision and recall. These findings align closely with previous literature on prediction using binary classification.

In our project we have looked at sales of credit cards, but our approach in its entirety is also applicable to any other products SpareBank 1 may wish to predict sales for. Our system may be integrated into existing processes and pipelines of SpareBank 1, such as targeted marketing through banners in the online banking solution. Furthermore, being able to suggest products that a customer is likely to buy may prove a valuable aid to customer advisors in conversation with new or existing customers. Data analysts at SpareBank 1 may also benefit from learning new or improved methods and techniques to utilise the vast amount of data available for problems such as sales or churn prediction.

8.3 Further work

8.3.1 Using LSTM for sequential data

In parts of our experiment we examined how sequential event data in the form of customer timelines could be used as input for training a classification model. We applied various traditional machine learning algorithms and preprocessing techniques and observed comparable performance to models trained on a common attributes dataset. We believe that sequential input data such as the customer event logs dataset would be a great candidate for Recurrent Neural Networks using Long Short-Term Memory (LSTM) units. LSTM has been proven to perform well on sequence data structures and have been applied to problems such as anomaly detection from time series [35], sequence prediction [36], and predicting time to an event based on sequences of events [37].

8.3.2 Cost-sensitive model evaluation metrics

Throughout this experiment we have been applying various preprocessing and classification algorithms to improve the precision and recall of our model. The end goal is a model that accurately predicts the customers that are likely to buy a credit card and then target them with marketing campaigns. Ultimately, the success of this marketing approach is measured by the cost and profit associated with the campaigns. It is not obvious how precision and recall affects the metrics used within the business domain. We fear that optimising model performance metrics do not necessarily align with the business goals.

We observe the need for a metric within the machine learning domain that takes the cost and profits of a potential marketing campaign into account when developing the model. The metric should be able to give meaningful answers to questions like "how much profit does 5% increase in recall account for when the precision is reduced by 10%?", or "what is the worst-case cost of a campaign targeting customers predicted with 90-95% confidence?". Correa Bahnsen et al. [38] presents a cost-sensitive framework for customer churn predictive modelling. Their framework proposal considers factors such as available portfolio offers, probability of positive marketing response and individual financial cost.

A similar framework could be adapted to sales prediction, incorporating factors such as product profitability, whether the product increases customer loyalty, and if the product is likely to lead to cross-selling. In addition to solving the metrics mismatch, this bridging of metrics could also facilitate the communication between data scientists and marketing experts.

Marketing experts who are not familiar with machine learning evaluation metrics will struggle to utilise their expertise when not fully understanding the implications of the model development. On the other hand, data scientists who build models are not necessarily aware of how or even if, the model optimisation leads to achieving the business goals. By developing an explicit, cost-sensitive metric, the border between data scientists and marketing experts diminishes. This could lead to more effective collaboration and higher profitability.

Appendices

Appendix A

Acronyms

GBT: Gradient Boosted Trees
RF: Random Forest
DT: Decision Trees
NB: Naïve Bayes
NN: Neural Networks
RNN: Recurrent Neural Networks
LSTM: Long Short-Term Memory
SVM: Support Vector Machine
RBF: Radial Basis Function
NPTB: Next Product To Buy-model
SB1: SpareBank 1 SMN
NSD: Norsk Senter for Forskningsdata
GDPR: General Data Protection Regulation (new EU directive)
TP: True Positive
FP: False Positive
TN: True Negative
FN: False Negative
ROC: Receiver Operating Characteristic
$P_{\text{class label}}$: Precision for some class label
$R_{\text{class label}}$: Recall for some class label
$\Delta R_{\text{class label}}$: Change in recall for some class label
$F_{1\text{ class label}}$: F_1-measure for some class label. See definition, section 2.9.3.
FFS: Forward feature selection
CF: Correlation filter
SMOTE: Synthetic Minority Oversampling Technique
CART: Classification And Regression Trees

Appendix B

Algorithm configurations

B.1 Random Forest

- **Split criterion**: Information Gain Ratio
- **Number of models**: 100
- No minimum node size
- No limit to number of levels (tree depth)

B.2 Gradient Boosted Trees

- **Limit to number of levels (tree depth)**: 4
- **Number of models**: 100
- **Learning rate**: 0.1
- Using mid point splits for numeric attributes
- Using binary splits for nominal columns
- **Missing value handling**: XGBoost
- **Attribute sampling**: All columns (no sampling)
- **Attribute selection**: Use same set of attributes for entire tree

B.3 Decision Tree

- **Split quality measure**: Gini index
- **Pruning method**: None
- **Minimum number of records pr node**: 2
- **Average split point**: True
- **Binary nominal splits**: False

B.4 Naïve Bayes

- **Default probability**: 0.0
- **Maximum number of unique nominal values per attribute**: 20

Appendix C

Baseline algorithm performance

n	$P_{\text{purchaser}}$	$R_{\text{purchaser}}$	$P_{\text{not purchaser}}$	$R_{\text{not purchaser}}$
50	0.84	0.33	0.91	0.95
100	0.85	0.34	0.91	0.95
150	0.86	0.34	0.91	0.95
200	0.86	0.34	0.91	0.95
250	0.85	0.34	0.91	0.95
300	0.85	0.34	0.91	0.95
400	0.85	0.34	0.91	0.95
500	0.85	0.34	0.91	0.95
600	0.85	0.34	0.91	0.95
700	0.85	0.35	0.91	0.95
800	0.85	0.34	0.91	0.95
900	0.85	0.34	0.91	0.95
1000	0.85	0.34	0.91	0.95

Table C.1: Gradient Boosted Trees baseline performance for customer timelines of length n

n	$P_{\text{purchaser}}$	$R_{\text{purchaser}}$	$P_{\text{not purchaser}}$	$R_{\text{not purchaser}}$
50	0.76	0.36	0.91	0.98
100	0.76	0.34	0.91	0.98
150	0.77	0.32	0.91	0.99
200	0.75	0.30	0.91	0.99
250	0.76	0.30	0.91	0.99
300	0.75	0.29	0.91	0.99
400	0.77	0.29	0.91	0.99
500	0.77	0.28	0.90	0.99
600	0.77	0.28	0.90	0.99
700	0.76	0.27	0.90	0.99
800	0.78	0.28	0.90	0.99
900	0.78	0.26	0.90	0.99
1000	0.78	0.26	0.90	0.99

Table C.2: Random Forest baseline performance for customer timelines of length n

n	$P_{\text{purchaser}}$	$R_{\text{purchaser}}$	$P_{\text{not purchaser}}$	$R_{\text{not purchaser}}$
50	0.52	0.52	0.93	0.93
100	0.52	0.52	0.93	0.93
150	0.52	0.52	0.93	0.93
200	0.52	0.52	0.93	0.93
250	0.53	0.52	0.93	0.93
300	0.52	0.52	0.93	0.93
400	0.52	0.51	0.93	0.93
500	0.52	0.52	0.93	0.93
600	0.53	0.53	0.93	0.93
700	0.53	0.53	0.93	0.93
800	0.53	0.53	0.93	0.93
900	0.53	0.53	0.93	0.93
1000	0.53	0.53	0.93	0.93

Table C.3: Decision Tree baseline performance for customer timelines of length n

n	$P_{\text{purchaser}}$	$R_{\text{purchaser}}$	$P_{\text{not purchaser}}$	$R_{\text{not purchaser}}$
50	0.18	0.41	0.90	0.73
100	0.17	0.43	0.90	0.71
150	0.17	0.44	0.89	0.69
200	0.16	0.44	0.89	0.67
250	0.16	0.46	0.89	0.66
300	0.16	0.47	0.89	0.65
400	0.16	0.51	0.90	0.62
500	0.16	0.56	0.90	0.58
600	0.16	0.60	0.90	0.55
700	0.16	0.63	0.91	0.53
800	0.16	0.66	0.91	0.50
900	0.16	0.69	0.91	0.48
1000	0.16	0.71	0.92	0.46

Table C.4: Naïve Bayes baseline performance for customer timelines of length n

Bibliography

[1] NSD. (2017) De-identified personal data. Norsk senter for forskningsdata. [Online]. Available: http://www.nsd.uib.no/personvernombud/en/help/vocabulary.html?id=8

[2] Datatilsynet. (2016, juni) Hva betyr de nye personvernreglene for din virksomhet? [Online]. Available: https://www.datatilsynet.no/regelverk-og-skjema/veiledere/hva-betyr/

[3] A. Gallo. (2014, oktober) The value of keeping the right customers. Harvard Business Review. [Online]. Available: https://hbr.org/2014/10/the-value-of-keeping-the-right-customers

[4] F. Reichheld. (2001, October) Prescription for cutting costs. Bain & Company. [Online]. Available: http://www.bain.com/Images/BB_Prescription_cutting_costs.pdf

[5] Tan, Steinbach, and Kumar, *Introduction to Data Mining*, 1st ed. Pearson, January 2016, new international edition.

[6] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. P. Lillicrap, K. Simonyan, and D. Hassabis, "Mastering chess and shogi by self-play with a general reinforcement learning algorithm," *CoRR*, vol. abs/1712.01815, 2017. [Online]. Available: http://arxiv.org/abs/1712.01815

[7] DeepMind. (2018) Deepmind - solve intelligence. Google. [Online]. Available: https://deepmind.com/about/

[8] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*, 2nd ed., ser. A Bradford book. Bradford Book, 5 2018. [Online]. Available: https://books.google.no/books?id=CAFR6IBF4xYC

[9] E. D. D. Team. (2018) A beginner's guide to deep reinforcement learning. Deeplearning4j. [Online]. Available: https://deeplearning4j.org/deepreinforcementlearning#define

[10] P. Z. Maymin, "Wage against the machine: A generalized deep-learning market test of dataset value," *International Journal of Forecasting*, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0169207017301152

[11] R. B. MARIMONT and M. B. SHAPIRO, "Nearest neighbour searches and the curse of dimensionality," *IMA Journal of Applied Mathematics*, vol. 24, no. 1, pp. 59–70, 1979. [Online]. Available: http://dx.doi.org/10.1093/imamat/24.1.59

[12] openML team. (2017) 10-fold crossvalidation. Open Machine Learning. [Online]. Available: https://www.openml.org/a/estimation-procedures/1

[13] R. Longadge and S. Dongre, "Class imbalance problem in data mining review," *CoRR*, vol. abs/1305.1707, 2013. [Online]. Available: http://arxiv.org/abs/1305.1707

[14] J. Burez and D. V. den Poel, “Handling class imbalance in customer churn prediction,” *Expert Systems with Applications*, vol. 36, no. 3, Part 1, pp. 4626 – 4636, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417408002121

[15] A. Agrawal, S. Kumar, and A. K. Mishra, “Credit card fraud detection: A case study,” in *2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom)*, March 2015, pp. 5–7.

[16] T. Pranckevicius and V. Marcinkevičius, “Comparison of naive bayes, random forest, decision tree, support vector machines, and logistic regression classifiers for text reviews classification,” *Baltic Journal of Modern Computing*, vol. 5, 01 2017.

[17] X. Fern. (2011) Decision trees. Oregon State University. [Online]. Available: http://web.engr.oregonstate.edu/~xfern/classes/cs534/notes/decision-tree-7-11.pdf

[18] J. H. Friedman. (1999, February) Greedy function approximation: A gradient boosting machine. Dept. of Statistics, Stanford University. [Online]. Available: https://statweb.stanford.edu/~jhf/ftp/trebst.pdf

[19] xgboost. (2016) Introduction to boosted trees. [Online]. Available: http://xgboost.readthedocs.io/en/latest/model.html

[20] T. Chen. (2014, October) Introduction to boosted trees. University of Washington. [Online]. Available: https://homes.cs.washington.edu/~tqchen/pdf/BoostedTree.pdf

[21] E. team. (2018) Overfitting in machine learning. EliteDataScience. [Online]. Available: https://elitedatascience.com/overfitting-in-machine-learning

[22] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *nature*, vol. 521, no. 7553, p. 436, 2015.

[23] M. van Gerven and S. Bohte, “Editorial: Artificial neural networks as models of neural information processing,” *Frontiers in Computational Neuroscience*, vol. 11, 12 2017.

[24] D. A. Kumar and V. Ravi, “Predicting credit card customer churn in banks using data mining,” *Int. J. Data Anal. Tech. Strateg.*, vol. 1, no. 1, pp. 4–28, Aug. 2008. [Online]. Available: http://dx.doi.org/10.1504/IJDATS.2008.020020

[25] N. Naveen, V. Ravi, and D. A. Kumar, *Application of fuzzyARTMAP for churn prediction in bank credit cards.* Inderscience Publishers, 01 2009, vol. 1, no. 4, pp. 428–444.

[26] Y. Zhao, B. Li, X. Li, W. Liu, and S. Ren, *Customer Churn Prediction Using Improved One-Class Support Vector Machine.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 300–306. [Online]. Available: https://doi.org/10.1007/11527503_36

[27] Y. Xie, X. Li, E. Ngai, and W. Ying, *Customer churn prediction using improved balanced random forests.* Elsevier, 2009, vol. 36, no. 3, Part 1, pp. 5445 – 5449. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417408004326

[28] B. K. Chitra, “Customer retention in banking sector using predictive data mining technique,” *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 9, 2011.

[29] D. N. S. Manjit Kaur, Dr. Kawaljeet Singh, “Data mining as a tool to predict the churn behaviour among indian bank customers,” *International Journal on Recent and Innovation Trends in Computing and Communication*, 2013.

[30] B. He, Y. Shi, Q. Wan, and X. Zhao, *Prediction of Customer Attrition of Commercial Banks based on SVM Model.* Elsevier, 2014, vol. 31, no. Supplement C, pp. 423

– 430, 2nd International Conference on Information Technology and Quantitative Management, ITQM 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050914004633

[31] A. Bilal Zorić, *Predicting customer churn in banking industry using neural networks.* Hrvatsko interdisciplinarno društvo, 2016, vol. 14, no. 2, pp. 116–124.

[32] S. Li, B. Sun, and R. T. Wilcox, "Cross-selling sequentially ordered products: An application to consumer banking services," *Journal of Marketing Research*, vol. 42, no. 2, pp. 233–239, 2005.

[33] A. Knott, A. Hayes, and S. A. Neslin, "Next-product-to-buy models for cross-selling applications," *Journal of interactive Marketing*, vol. 16, no. 3, pp. 59–75, 2002.

[34] R. Kohavi *et al.*, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Ijcai*, vol. 14, no. 2. Montreal, Canada, 1995, pp. 1137–1145.

[35] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, "Long short term memory networks for anomaly detection in time series," in *Proceedings.* Presses universitaires de Louvain, 2015, p. 89.

[36] J. Schmidhuber, D. Wierstra, and F. J. Gomez, "Evolino: Hybrid neuroevolution/optimal linear search for sequence learning," in *IJCAI*, 2005.

[37] E. Martinsson, "WTTE-RNN : Weibull Time To Event Recurrent Neural Network," Master's thesis, Chalmers University Of Technology, 2016.

[38] A. C. Bahnsen, D. Aouada, and B. Ottersten, "A novel cost-sensitive framework for customer churn predictive modeling," *Decision Analytics*, vol. 2, no. 1, p. 5, Jun 2015. [Online]. Available: https://doi.org/10.1186/s40165-015-0014-6