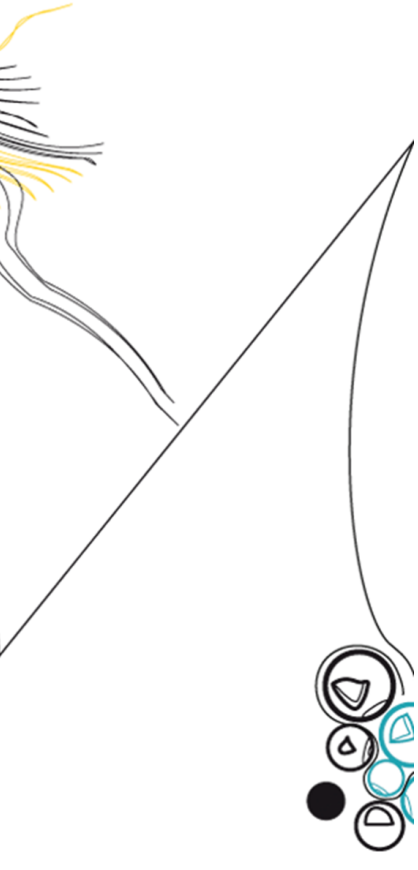
UNIVERSITY OF TWENTE.

Faculty of Electrical Engineering, Mathematics & Computer Science

Customer purchase prediction through machine learning

Hannah Sophia Seippel

M.Sc. Thesis
March 2018



Graduation committee:
Dr. C. Amrit
c.amrit@utwente.nl
Faculty BMS, University of Twente

Dr. M. Poel
m.poel@utwente.nl
Faculty EEMCS, University of Twente

A. K. Ramakrishnan
a.k.ramakrishnan@utwente.nl
Ctit, University of Twente

Preface

This master thesis marks the end of five and a half years of studying, including two universities in two countries and three internships that sent me to five different cities in total. Herewith, I would like to thank everyone that was a part of this journey, making this experience possible and unforgettable.

First of all, I would like to thank my company supervisor, whose name cannot be mentioned due to confidentiality reasons, but who will surely know that he is addressed. Thank you for providing me with the topic for this master thesis that connects Data Science with Business, making it a perfect fit for my degree in Business Information Technology with a focus on Business Analytics. And a further thanks for supporting me along the way with guidance and good advice.

A thank you goes to the University of Twente, for offering this interesting Master program and making an internship and a master thesis abroad possible. Of course, I would also like to thank my two supervisors Dr. Chintan Amrit and Arun Ramakrishnan for providing helpful feedback and support and also Dr. Mannes Poel who agreed on being my second university supervisor on such short notice.

Finally and most importantly, I would like to thank my family and friends, who have always shown great support, especially my parents without whom none of this would have been possible and Fabian who always believes in me.

Abstract

Due to today's transition from visiting physical stores to online shopping, predicting customer behavior in the context of e-commerce is gaining importance. It can increase customer satisfaction and sales, resulting in higher conversion rates and a competitive advantage, by facilitating a more personalized shopping process. By utilizing clickstream and supplementary customer data, models for predicting customer behavior can be built. This study analyzes machine learning models to predict a purchase, which is a relevant use case as applied by a large German clothing retailer. Next, to comparing models this study further gives insight into the performance differences of the models on sequential clickstream and the static customer data, by conducting a descriptive data analysis and separately training the models on the different datasets. The results indicate that a Random Forest algorithm is best suited for the prediction task, showing the best performance results, reasonable latency, offering comprehensibility and a high robustness. Regarding the different data types, models trained on sequential session data outperformed models trained on the static customer data by far. The best results were obtained when combining both datasets.

Management summary

Context

Predicting customer behavior in the context of e-commerce is becoming more important nowadays. It increases customer satisfaction and sales, by facilitating an increase of customer experience through personalization, recommendations and special offers. By utilizing clickstream and additional customer data, predictions can be carried out, ranging from customer classification, purchase prediction, and recommender systems to the detection of customer churn. A variety of machine learning models and data are available to conduct these kinds of predictions.

Research Problem

Categorizing whether a web shop session will end in a purchase or not, is a relevant use case in the context of predictions in e-commerce. This categorization followed by the display of gift cards to non purchasing customers, to convince them of a purchase nonetheless, has proven to increase turnover of a large German clothing retailer. A variety of possible prediction models as well as different data sources exist to carry out such predictions. This paper aims at retrieving well-suited prediction models and comparing their performances across different data types, such as static and dynamic data, to establish how customers can be best classified as buying or no buying. This results in the following research question:

How can a customer in a web shop be categorized as a buying or no buying customer?

Methodology

This research was structured based on the Cross Industry Standard Process for Data Mining (CRISP-DM) methodology. Suitable models, being boosted tree, Ran-

dom Forest (RF), Support Vector Machine (SVM), Feed-forward Neural Network (FNN), Logistic Regression (LR) and Recurrent Neural Networks (RNN), were identified through a literature research. Following, algorithms were trained on three different datasets, the sequential session data, the static customer data and a combined dataset, then evaluated and compared based on different performance metrics, prediction latency and comprehensibility. The RNN was further trained on datasets with varying degrees of required feature engineering. All algorithms as well as the evaluation and comparison were implemented in Python.

Results

The obtained results indicate that the RF performed best while showing reasonable prediction latency. Regarding the comprehensibility, no difference between the different algorithms was observed. The performance of different datasets shows that a combined dataset leads to the best results, where customer information enhances the results only slightly. An overview of the results regarding the different datasets and algorithms concerning the ROC AUC value can be observed in Table 1. Further, a promising effect regarding time-consuming feature engineering was observed for the RNN, where fewer and less engineered features led to better results than a larger amount of more heavily engineered features as used for the other algorithms.

Table 1: ROC AUC results for all algorithms and datasets.

Algorithm/ Data Type	Combined Data	Sequence Data	Customer Data
Boosted tree	0.79	0.78	0.64
RF	**0.82**	**0.81**	**0.67**
FNN	0.73	0.67	**0.67**
SVM	0.67	0.74	0.39
LR	0.8	0.78	0.65
RNN	0.74	0.74	0.66

Conclusion

This study shows that web shop sessions can be well categorized as buying or no buying sessions, with an RF showing the best performance. Further, by training on different datasets this study was able to emphasize that session based data, mainly

generated from the customer clickstream, is most important for predicting purchase probabilities. This indicates that personal customer information, often associated with privacy concerns and regulations, is not necessarily needed to predict customer behavior well. Additionally, by training an RNN on less engineered features, it was displayed that stateful models perform well while requiring less time-consuming feature engineering, when detecting sequential patterns.

List of Figures

List of Tables

List of acronyms

AUC Area under the curve

CRISP-DM Cross Industry Standard Process for Data Mining

DT Decision Trees

FNN Feed-forward Neural Networks

HMC Higher-order Markov Chains

KNN K-nearest Neighbor

LR Logistic Regression

LSTM Long short-term memory

RF Random Forest

ROC Receiver Operating Characteristic

RNN Recurrent Neural Networks

SVM Support Vector Machines

PCA Principal Component Analysis

Contents

Chapter 1

Introduction

The research domain of this master thesis, being the analysis, classification, and prediction of customer behavior in the field of e-commerce, is introduced in this chapter. Research in this area is motivated in the first part (Section 1.1). Resulting, a research question, and sub-questions are formulated in Section 1.2. Finally, the structure of the thesis is outlined in Section 1.3.

1.1 Motivation

As a result of today's knowledge-based economy and the information society, e-commerce is becoming increasingly popular all over the globe. The most common type being the Business-to-Customer trade, typically represented as online stores, displacing physical stores quickly (Suchacka & Chodak, 2016). The transition from physical to online shopping can be inferred by numbers such as an increase in total retail sales in Germany of 3% in 2016, whereas the e-commerce sales rose by an estimated 12.5% to 58.52 billion dollars and are expected to exceed 86 billion dollars at the end of 2021 (*Retail Ecommerce in Germany: A Major Digital Market Growing in Size and Sophistication*, 2017). The rapid growth of e-commerce has transformed the shopping process as a whole and along with it the traditional buyer-merchant relationships. This change is accompanied by challenges that companies need to address. Such challenges entail more competition and a volatile relationship between customers and merchants since customers and their preferences are no longer personally known, resulting in less loyal customers. Therefore, attracting customers, gaining their trust and retaining them becomes a main objective in modern e-commerce (Nakayama, 2009; Salehi, Abdollahbeigi, Langroudi, & Salehi, 2012). Web shop visitors leave more traces than ever before. Large amounts of personal information as well as clickstream data, recorded during each web shop visit, are collected, connected and stored for analysis with data mining techniques. Knowl-

edge retrieved from these analyses can improve customer satisfaction, by making the shopping process more efficient, more engaging and increasingly personalized (Magrabi, 2016), mitigating the risks associated with the aforementioned challenges. In the long run, this can lead to a competitive advantage resulting from a higher conversion rate and increased turnover (Hop, 2013; Suchacka & Chodak, 2016).

The analysis of customer purchase behavior dates back to the beginning of e-commerce (Bellman, Lohse, & Johnson, 1999) and has many applications nowadays. Examples are item recommendations for customers (Y. Tan, Xu, & Liu, 2016; Hidasi, Quadrana, Karatzoglou, & Tikk, 2016), the classification of customers into certain categories such as buyers, visitors, etc. (Moe, 2003; Fajta, 2014), predicting the purchase probability in order to offer a higher quality of service to customers that are more likely to buy (Lo, Frankowsik, & Leskovec, 2014; Korpusik, Sakaki, Chen, & Chen, 2016; Suchacka & Templewski, 2017; Lang & Rettenmeier, 2017) and the timely detection of customer churn in order to prevent such (Xie, Li, Ngai, & Ying, 2008; Castanedo, Valverde, Zaratiegui, & Vazquez, 2014). This thesis is similar to the detection of customer churn, being concerned with predicting the abortion of a shopping process in order to prevent it. Prevention is possible through, for example, showing recommended items or displaying gift cards to motivate a purchase. Predictions of customer actions are often based on personal customer data, clickstream data and supplementary data from other sources. Various machine learning methods exist to perform classification on the collected clickstream data. Examples are regular machine learning models such as Logistic Regression (LR), Support Vector Machines (SVM), Decision Trees (DT), Random Forest (RF) and Feed-forward Neural Networks (FNN) as well as stateful models such as Higher-order Markov Chains (HMC) and Recurrent Neural Networks (RNN).

Motivated by the increasing importance of classifying customer behavior and a large number of possible prediction models and data sources, this thesis aims at implementing and comparing suitable models trained on different datasets to identify the most appropriate one for predicting the abortion probability of a web shop visitor. Hence, a binary classification task of a visitor belonging to the aborting or not aborting category. This classification followed by the display of gift cards in order to persuade the visitor to stay in the web shop has been tested by a large German clothing retailer in an A/B test and showed an increased conversion rate from 8% to 10% under assumptions of a cost-benefit analysis. It is, therefore, a suitable and relevant use case for testing classification models and different data types in the context of e-commerce.

Testing the models on different dataset types was motivated by literature: Previous studies have shown that a combined dataset of clickstream and customer

data leads to the best performance results when predicting buying probabilities as compared to using the datasets separately. Here, the static customer data can only slightly increase the already good performance of the dynamic clickstream data (Bogina, Kuflik, & Mokryn, 2016; Lee, Ha, Han, Rha, & Kwon, 2015; Poggi, Moreno, Berral, Gavald, & Torres, 2007). This thesis aims at replicating the results from literature and additionally enhancing the findings by providing further insight into the predictive power of the different datasets through an exploratory data analysis. This analysis is expected to give reasoning about the bad performance of static customer data compared to dynamic clickstream data.

In the study of Lang and Rettenmeier (2017), an RNN was implemented to predict the probability of a purchase occurring within a web shop session. The study claims that RNNs, being stateful models equipped with a memory, provide a possibility of reducing labor-intensive feature engineering in the context of sequential data. Lang and Rettenmeier do not include a comparison about how RNNs perform on datasets with different degrees of feature engineering, and do, therefore, not prove if less feature engineering does indeed show as good results as heavily engineered features. Resulting from this and since the sequential clickstream data seems to be a natural fit for RNNs, this study aims at training an RNN on datasets with different levels of feature engineering, to establish how well RNNs perform on less engineered features. This shows how RNNs might offer the possibility to reduce feature engineering, which is very important for e-commerce since it is typically a time-consuming task that requires a lot of expert knowledge.

Finally, after obtaining the results, these are tested for their robustness, to display how different conditions, such as the used device or day of the week, could influence the models' performance. This improves on the findings in literature since none of the reviewed papers analyze how the obtained results might behave under different conditions, even though this is important to assess since it helps to understand how a model would perform under real conditions after deployment.

1.2 Problem definition

Resulting from the facts stated in Section 1.1, in this thesis, machine learning models are identified and implemented for solving the task of classifying a web shop visitor as aborting or non aborting, in the following referred to as no buying and buying sessions.

The models are applied to different data types, namely clickstream data generated by each visitor of the web shop as well as customer data if a visitor could be identified. This is done to establish which model and data is best-suited for the task of predicting the buying probability of an online shopping session, in terms of

performance, latency, and comprehensibility. Latency is important since in the use case, predictions have to be conducted in real-time and model comprehensibility is considered since the demand for explaining decisions made by machine learning models is rising. The boosted tree model that is already being used by the German clothing retailer, will act as a baseline for the comparison of the different algorithms. To provide further insight and reasoning about the varying performances on different datasets, an exploratory data analysis on the clickstream and static customer data is conducted.

An RNN, representing the class of stateful machine learning models, is additionally trained on datasets, which required fewer feature engineering, to show if stateful models provide good results while reducing the need for feature engineering.

After obtaining the results of the different models on all datasets, the models are tested for their robustness to different conditions. Examples of such conditions are the gender of the visitor or the device on which the web shop was visited. This analysis indicates how the models will perform under real conditions after deployment.

Deployment is out of scope for this research and can only be regarded as an implication if a tested model outperforms the baseline model.

From all of the previously mentioned points the research question as stated below results:

> ***Research Question:*** *How can a visitor in a web shop be categorized as a buying or no buying?*

Five subquestions are needed to completely answer the main research question and are stated in the following:

> ***Sub Question 1:*** *How can an exploratory data analysis provide insight into the prediction problem?*

> ***Sub Question 2:*** *Which machine learning model is best suited to solve the prediction problem?*

> ***Sub Question 3:*** *How do different data types, such as dynamic clickstream and static customer data, influence the models' performance?*

> ***Sub Question 4:*** *Can stateful models produce good results while requiring less feature engineering?*

Sub Question 5: *How robust is the best-suited algorithm to different conditions?*

1.3 Report organization

The remainder of this report first discusses findings from the literature, analyzing model performance on similar tasks as the one at hand in Chapter 2. Then the methods, consisting of the research framework, the utilized algorithms and software as well as the evaluation metrics are described in Chapter 3. The methodology chapter is followed by the data Chapter 4 discussing data pre-processing, a first data analysis and the used features. Then the implementation including hyper-parameter tuning is described in Chapter 5. The results are presented in Chapter 6 and discussed afterward in Chapter 7. The thesis ends in Chapter 8 with a conclusion, giving answers to all above-stated research questions.

Chapter 2

Literature review

This chapter reviews the results and algorithms used by studies concerned with a similar binary classification problem as the one at hand. To create a broader understanding of the problem and the different algorithms, binary classification and the algorithms used for solving it are each explained before reviewing their application and performance in literature. All reviewed literature is summarized in Table 2.1, showing details on the data and results obtained in each study. Looking at the table one has to keep in mind that comparing results across studies is difficult since different data types, evaluation metrics, evaluation thresholds, etc. were used. Finally, a gap analysis of the reviewed literature is conducted.

2.1 Binary Classification

Data mining has various applications with classification being the most common one. Being a predictive analytics task, the aim of classification is to predict a categorical target variable from a set of input variables. This target variable can be expressed either through various categories or be of binary nature (Kotu & Deshpande, 2014). The task at hand is a binary classification task since the target variable has two categories: *buying* and *no buying*. In order to predict the target variable a generalized relationship between input and target variable is learned from a labeled dataset. It is then applied to new data for classification. Learning algorithms should both fit the training data and generalize well over new data (P. Tan, Steinbach, & Kumar, 2005). Various machine learning algorithms exist that have different methods of extracting this relationship.

Table 2.1: Used data, algorithms and results obtained by different studies.

Authors	Data Information	Data Type	DT	RF	SVM	LR	FNN	KNN	RNN	HMC
Bogina, Kuflik & Mokryn (2016)	9,249,728 transactions	Clickstream	Precision: 0.824 Recall: 0.808 ROC AUC: 0.889							
		Clickstream + Item information	Precision: 0.937 Recall: 0.93 ROC AUC: 0.939							
Korpusik, Sakaki & Chen (2016)	3,655 tweets	Tweets indicating a desire or a purchase of phones/ cameras					Accuracy: 0.734		Accuracy: 0.817	
Hop (2013)	429,013 transactions	Static + Dynamic Session Data	Accuracy: 0.972	Accuracy: 0.903	Accuracy: 0.868		Accuracy: 0.755			
Lang & Rettenmeier (2017)	Several million sessions	Dynamic Session Data + Static Session/ Customer Data				ROC AUC: 0.832	ROC AUC 0.841		ROC AUC: 0.843	
Lee et al. (2015)	76,375,439 transactions	Static Item data			Accuracy: 0.6 ROC AUC: 0.67					
		Dynamic Session Data			Accuracy: 0.8 ROC AUC: 0.78					
		Static item + dynamic session data			Accuracy: 0.8 ROC AUC: 0.79					
Niu, Li & Yu (2013)	6,944,274 transaction 4.9% Buying	Click information, Static + Dynamic Session data, Satic Customer Data	Accuracy: 0.76 Recall: 0.73 Specificity: 0.82			Accuracy: 0.61 Recall: 0.61 Specificity: 0.60				
Poggio et al. (2007)	14,500 transactions 50/50 - buying/no buying	Static Session Data	Accuracy: 0.765 Precision: 0.172 Recall: 0.669			Accuracy: 0.727 Precision: 0.153 Recall: 0.689				Used to generate Data
		Static + Dynamic Session Data	Accuracy: 0.758 Precision: 0.174 Recall: 0.708			Accuracy: 0.742 Precision: 0.159 Recall: 0.681				Input for ML models
Suchacka, Skolimowska-Kulig & Potempa (2015)	26,000 transactions	Static + Dynamic Session data, Satic Customer Data						Accuracy: 0.999 Recall: 0.875		
Suchacka & Templewski (2017)		Static + Dynamic Session data, Satic Customer Data					Accuracy: 0.996 Recall: 0.878			

2.2 Learning algorithms for binary classification

The most common type of machine learning algorithms for a binary classification task are vector-based methods. Belonging to this category are DTs, RFs, SVMs, LR, and FNNs. The baseline model of this study also associates with this category being part of the DT algorithms. Common to these algorithms is to learn supervised with a set of feature vectors and corresponding outputs (Heaton, 2016). They are eager learning models, where a classification model is constructed based on a given training dataset before new data is classified according to the model. The opposites are lazy learners, such as the K-nearest Neighbor (KNN) algorithm, where training data is simply stored and a test data point is awaited for classification (Han, Pei, & Kamber, 2011).

All of the above-mentioned methods are stateless machine learning algorithms; they do not have any memory and always return the same answer given the same input. This is well-suited for most classification tasks, however, it is difficult to model patterns over time, since the different states have to be modeled through complex feature engineering, creating inaccuracies and increasing complexity by raising the number of input features. Nevertheless, the ability to model time sequences and extract patterns over time can be useful for the research at hand, since the click-stream data used in this study is of sequential and time-dependent nature. Fortunately, stateful models exist, equipped with a memory, to remember previous states and to extract sequential patterns without the need for engineering time-dependent features explicitly. Examples of such models are RNNs and HMCs. All of the mentioned algorithms are explained below, each followed by their applications and performances in literature.

2.2.1 Decision Trees

DTs consist of a set of split conditions which divide a heterogeneous population into smaller, more homogeneous subgroups regarding a certain variable. The aim is to create the most homogeneous subgroups. Various algorithms exist to find the best splits such as the Hunts algorithm, which follows a greedy strategy consisting of local optimum decisions (P. Tan et al., 2005). Simple DTs have the advantage of being convertible to simple, understandable classification rules (Han et al., 2011). This comprehensibility decreases as the models grow larger and more unbalanced (Rokach & Maimon, 2014). In general, DTs offer a relatively fast learning and prediction speed. Even though the different types vary regarding comprehensibility, they are still easier to understand than black box models such as FNNs or SVMs (Rokach & Maimon, 2014). Disadvantages are the required feature engineering, the

inability to implicitly model time sequences and an increased complexity regarding trees with categorical variables consisting of many categories. Methods compromising only singular DTs, called non-ensemble DTs, tend to over-fit and be unstable regarding noisy data (Hop, 2013).

Boosted Decision Trees

Boosted DTs belong to the ensemble methods, consisting of more than one DT. Here, a sequence of trees is built, where each tree results from the prediction residuals of the previous tree (Friedman, 2002). An example for such a method is the baseline model used for this study. Boosted DTs have proven to be a very powerful method for predictive analytics by winning a lot of Kaggle machine learning competitions (*Kaggle – The Home of data Science and Machine Learning*, 2017), but are less comprehensible than simple DTs, since they consist of many trees.

Random Forest

Bagging is another example for ensemble trees, where many large trees are fit to the bootstrap re-sampled versions of the data and are classified by majority vote (Breiman, 1996). RF improves on Bagging by de-correlating the trees. After each tree split a random sample of features is chosen and only these are considered for the next split. The results are again based on the majority vote of the single trees. By using a large number of classifiers, Bagging and RFs, improve on the weaknesses of non-ensemble DTs, such as robustness and over-fitting. They train faster than the boosted trees but need more time for the prediction (Breiman, 2001). Nevertheless, Bagging and RFs still depend on feature engineering and cannot model time dependencies.

DTs show great results throughout literature being applied to very similar problems as this research. Bogina et al. (2016), for example, used different DT algorithms to classify a session as a buying or no buying session. Next to clickstream data also a setup with additional data about item sales statistics was used, which increased the prediction performance. In the case where only clickstream data was used, being the closest to our use case, a Bagging RepTree showed the best results. It was implemented in Weka, a tool that supports data analysis with different machine learning techniques (Frank, Hall, & Witten, 2016) and showed a precision of 0.824, a recall of 0.808, an F1-score of 0.806 and a ROC Area under the curve (AUC) of 0.889.

Poggi et al. (2007) trained different machine learning models on clickstream data which was transformed into HMCs prior to processing. Only 14,500 transac-

tions were used for training this model. The authors tested separately for static session information and static and dynamic session information combined. The results show that combining static and dynamic session data only increased the prediction performance slightly. The results further show that the J48 model, a DT, outperformed an LR classifier with a recall of 0.708 compared to a recall of 0.681.

In the prudsys Datamining Cup 2013, the challenge was to classify buying versus no buying sessions from clickstream data (*DMC 2013*, 2013), where the training dataset contained 429,013 data points. The winning team from the University of Dortmund achieved an accuracy of 0.972 by using Bagging of 600 C4.5 DTs (Hop, 2013). Hop also competed in the prudsys Datamining Cup 2013 and compared RFs with SVMs and an FNN. RFs outperformed the other two methods with an accuracy of 0.903, the SVM showed the second best accuracy with 0.868, whereas the FNN only achieved an accuracy of 0.755. Next to the high accuracy Hop also mentions other advantages of the RF method: Compared to SVMs the computational effort of training is low and it requires minimal hyper-parameter tuning making it easy and fast to use.

Lastly, Niu, Li, and Yu (2017) also estimated purchase probabilities with RFs while not mainly focusing on clickstream data itself, but on search queries and click positions. For building the model they included information about mouse clicks, static and dynamic session data as well as static customer data. To train and evaluate the model a rather large amount of 1,530,738 records was used. Their study shows that a RF outperformed the LR with an accuracy of 0.76, a sensitivity of 0.73 and a specificity of 0.82, compared to results of 0.61, 0.61 and 0.60 respectively. Next to the prediction results, they also show some descriptive statistics of the customer shopping behavior such as the average amount of formulated queries being 8.3 and that a visitor spends around three minutes on an article detail page.

2.2.2 Support Vector Machines

An SVM separates two classes by fitting a hyperplane between them. Doing this, only one hyperplane is used, which differs from DTs where a hyperplane is added after each split. In cases where multiple separating hyperplanes can be found, the SVM detects the maximum-margin hyperplane, maximizing the distance to data points of both classes. Such a hyperplane is shown in Figure 2.1. This leads to a higher generalizability and therefore to better test accuracies.

If a feature space is not linearly separable, a kernel-function is used to map data on a higher dimensional feature space where the data becomes linear separable (Hofmann, 2006). By mapping inputs to a high-dimensional feature space, it becomes possible for SVMs to not only model linear relationships but to also con-

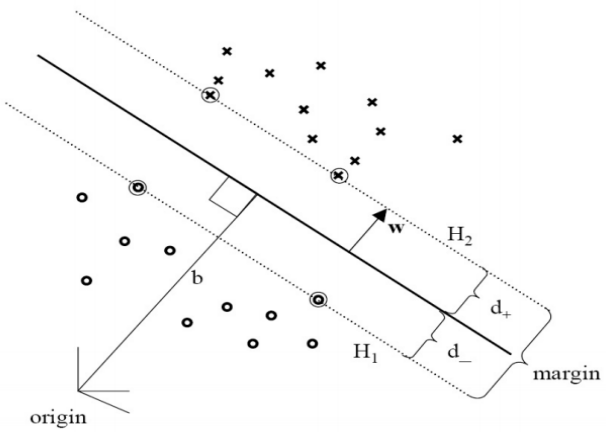


Figure 2.1: Optimal separating hyperplane with maximal margin. Retrieved from Hofmann (2006).

duct non-linear classification (Cortes & Vapnik, 1995). Four basic kernel functions are listed below (Hsu, Chang, & Lin, 2003):

$$Linear = K(x_i, x_j) = x_i^T x_j \tag{2.1}$$

$$Polynomial = K(x_i, x_j) = (\gamma x_i x_j + r)^d, \gamma > 0 \tag{2.2}$$

$$Radial\,basis\,function\,(RBF) = K(x_i, x_j) = exp(-\gamma||x_i - x_j||^2), \gamma > 0 \tag{2.3}$$

$$Sigmoid = K(x_i, x_j) = tanh(\gamma x_i x_j + r) \tag{2.4}$$

Where $K(x_i, x_j)$ is the kernel function, mapping the training vectors x_i into a higher dimensional feature space. y, r and d are kernel specific parameters (Hsu et al., 2003).

Regarding their performance, SVMs show a high accuracy as well as fast prediction times. They further work very well with high-dimensional data input. Disadvantages are long training times and their difficulty to be interpreted. Further, next to feature engineering, also hyper-parameter tuning is required, which can be difficult and time-consuming (Han et al., 2011).

The aforementioned study by Hop (2013) shows that SVMs could not outperform the RF algorithm but performed better than the FNN. Also, Lee et al. (2015) made use of an SVM to predict the purchase probability of a certain item. Next to clickstream data also item information was incorporated. To compare the importance of the different data types on the prediction performance, three models were separately trained and tested on the item data, the clickstream data, and the combined dataset. The results show that the item data generated the worst results whereas the combined dataset performed best. Nevertheless, the results on

the complete dataset and only the session data are very similar, indicating that the static item information does not contain high predictive power. The results on the complete data show an accuracy of 0.8, a ROC of 0.79, a precision of 0.74, a recall of 0.92 and an F1-score of 0.82.

2.2.3 Logistic Regression

LR, also called Logit regression, belongs to the class of generalized linear models and is used to predict categorical target variables. This is achieved through a logistic function, which has the shape of a sigmoid curve, taking values between 0 and 1. This function is modeled by combining input values linearly with coefficients, as shown in Equation 2.5. Where y is the output, b_0 the bias term and b_1 the coefficient for the input value x (Russell & Norvig, 1995).

$$y = \frac{e^{b_0+b_1x}}{1+e^{b_0+b_1x}} \tag{2.5}$$

Every column of the input vector learns a coefficient from the training data through maximum-likelihood estimation (Hastie, Tibshirani, & Friedman, 2002). LR is very fast regarding prediction and training times, it is, hence, one of the most popular machine learning algorithms for binary classification. It, nevertheless, requires feature engineering and the encoding of categorical variables. It is also sensitive to noise, therefore, outliers should be removed before the training. Further, LR does not perform well on highly correlated input factors. Hence, it can be helpful to only use principal components, linearly uncorrelated variables, for the regression (Jolliffe, 1982). This can be achieved through a Principal Component Analysis (PCA), a method to extract the principal components from a set of possibly correlated variables.

LR has been used in a variety of the reviewed papers but has always been outperformed by other machine learning methods, shown in the above-mentioned studies of Poggi et al. (2007) and Niu et al. (2017). Similar findings were obtained by a study from Lang and Rettenmeier (2017), where an RNN outperformed the LR in predicting the purchase probability based on clickstream data, static session data, and customer data with a ROC AUC of 0.843 and 0.832 respectively.

2.2.4 Feed-forward Neural Networks

FNNs are one of the simplest types of Artificial Neural Networks. Input data only flows forward through the network, from the input nodes to the hidden nodes and finally the output nodes (Zell, 1994). Nodes are organized in layers, where every

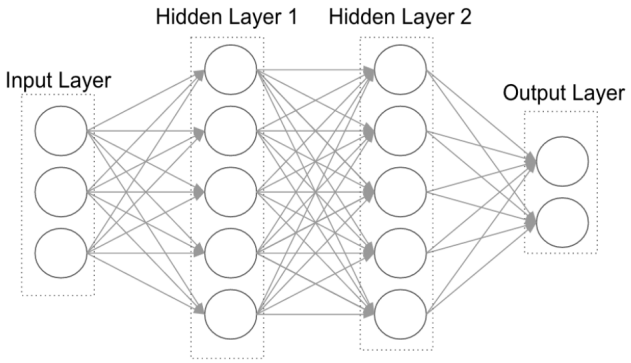


Figure 2.2: An FNN with two hidden layers, three input and two output nodes. Retrieved from *A Practical Introduction to Deep Learning with Caffe and Python* (2016).

node in a layer connects to all nodes of the previous layer. An exemplary architecture can be seen in Figure 2.2. The connections have different weights assigned, which are generated during the learning phase, for example via back-propagation. Every node has an activation function and only fires according to this function. Output is the probability of each class, which add up to one. With enough hidden units, an FNN can approximate any function. From a statistical point of view, FNNs perform a nonlinear regression. Such neural networks have the disadvantage of long training times and little comprehensibility. Next to feature engineering, they require hyper-parameter tuning. Nevertheless, they have shown very good performances throughout literature and generalize well to unseen patterns, while being tolerant to noisy data (Han et al., 2011).

Other than the before mentioned research by Hop (2013), which displayed a bad performance for FNNs, Suchacka and Templewski (2017) were able to achieve a high accuracy (0.996) and recall (0.878) for an FNN on clickstream data to predict purchases in sessions. They used static and dynamic session data as well as static customer data to train the model. The large performance difference to the paper of Hop might be due to the fact that Hop used a rather simple FNN with only one hidden layer and that different data was used to train and test the models in the two papers.

2.2.5 K-nearest Neighbor

As mentioned before, the KNN is a lazy learning algorithm, where training data is simply stored and a test data point is awaited for classification (Han et al., 2011).

All of the stored training instances correspond to points in an *n*-dimensional feature space. A point's nearest neighbors are defined by distance measurements, most commonly Euclidean distance. An unlabeled test data point will be assigned the label most common amongst its *k*-nearest neighbors. The advantages of this method are its robustness to noisy data and a very fast training speed. Disadvantages are an increased complexity of dimensionality through irrelevant features and therefore a decreased performance, highlighting the importance of feature engineering, longer prediction times compared to eager learning models and a low comprehensibility with high-dimensional input.

The KNN model has been applied regularly in e-commerce regarding recommender systems, where products are recommended to a web shop visitor based on the preferences of its nearest neighbors. For classifying customer behavior on the other hand only one study by Suchacka, Skolimowska-Kulig, and Potempa (2015) could be found. The aim of the research was to classify customer web shop sessions in buying or browsing sessions. As a model, a KNN algorithm was used with clickstream data split by sessions as input. 26,000 records were used for training and 13,000 for testing. Different configurations were tested, with an 11-NN model showing the best results, with a sensitivity and accuracy of 0.875 and 0.9985 respectively.

2.2.6 Recurrent Neural Networks

Due to their statefulness, RNNs have been applied to many sequence modeling tasks, such as natural language processing. Their structure is similar to FNNs with additional loops adding memory and feedback to the system as displayed in Figure 2.3. The latent state of the RNN is changed and updated with every new input. This state represents learned features and adds a broader abstraction to the input features, causing less need for feature engineering as the other methods (Brownlee, 2016; Lang & Rettenmeier, 2017). Reducing the amount of feature engineering does not only save time and reduces complexity but is also expected to yield improved results, since feature engineering is an abstraction of the real data through which important information can be lost. Through the RNN's memory, detection of sequential patterns in the data is enabled, making it a good fit for the clickstream data used in this thesis. RNNs are trained with back-propagation through time, which is an adaptation of regular back-propagation, where errors are back-propagated through the network and weights are updated accordingly (Brownlee, 2016). This error gradient vanishes or explodes when the input sequence gets longer, and therefore results in an information loss, known as the vanishing gradient problem (Bengio, Simard, &

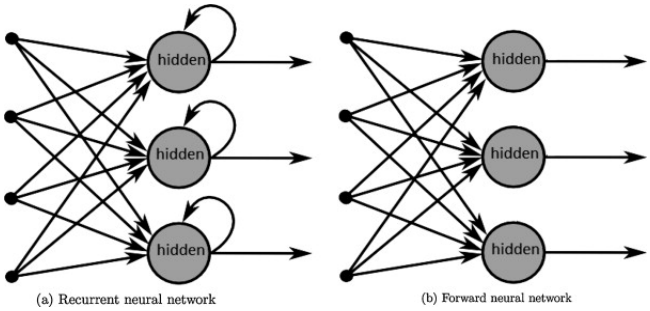


Figure 2.3: On the left an architecture of an RNN showing backward connections compared to an FNN with only forward connections on the right. Retrieved from Mulder et al. (2015).

Frasconi, 1994; Korpusik et al., 2016). A solution to the vanishing gradient problem offers the Long short-term memory (LSTM) network. In LSTMs neurons are replaced by memory cells. These cells operate through gates that manage the cells' input, output and state. These gates are activated with a sigmoid function making the information flow through the cell conditional (Brownlee, 2016). As a result, information that increases prediction accuracy is kept in the memory cell, solving the vanishing gradient problem (Hochreiter & Schmidhuber, 1997).

As mentioned above, RNNs and LSTMs decrease the amount of feature engineering and can model time dependencies of the input data. The paper of Lang and Rettenmeier (2017) shows how the influence of different customer actions on the prediction outcome can be visualized, providing a high interpretability. Drawbacks are both extended prediction and training times compared to other machine learning methods. They further require hyper-parameter tuning, which can make architectural choices complex (Lang & Rettenmeier, 2017).

RNNs are applied in the most recent studies on predicting customer behavior. Lang and Rettenmeier (2017) used clickstream data and customer information to predict the purchase probability of a customer with LSTMs. The model outperformed the also tested FNN and LR with a ROC AUC of 0.841, 0.832 and 0.843 for FNN, LR and RNN respectively. On the downside, RNNs need a lot of data and time to train. The exact numbers are not disclosed in the paper but millions of user histories are mentioned on which the model was trained. Even though RNNs require a lot of training data they decrease the amount of needed feature engineering. Lang and Rettenmeier (2017) express this in an image shown in Figure 2.4. While vector-based methods need information from a lot of handcrafted features, RNNs can extract this information from the sequence of customer actions given to them.

Another study that used LSTMs for purchase prediction was conducted by

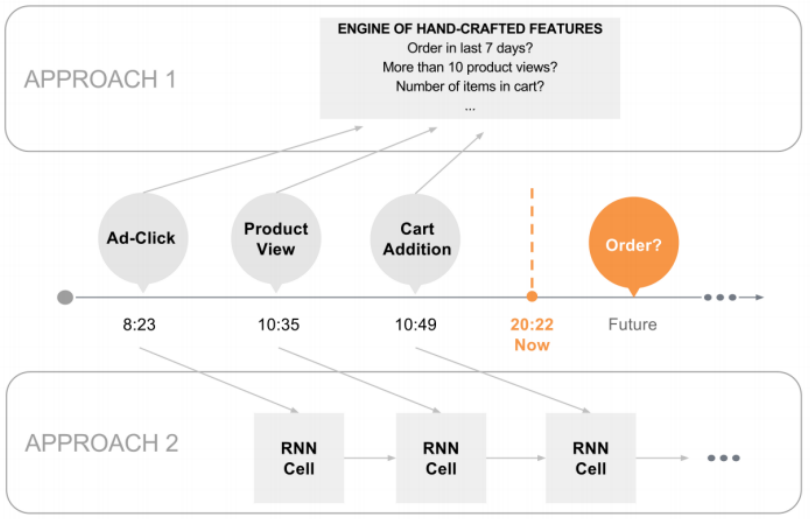


Figure 2.4: On the top Approach 1, common vector-based models that need hand crafted features to explicitly model sequential information compared to Approach 2 on the bottom, an RNN that can extract all important information from a sequence by itself. Retrieved from Lang and Rettenmeier (2017).

Korpusik et al. (2016). Here, successive tweets of people indicating a purchase intent were analyzed. The LSTM outperformed an FNN, with an accuracy of 0.82 versus 0.73. Again, the exact amount of data to train the models was not mentioned. But the model consisted of 50 hidden layers, which is complex and probably required a lot of data to be trained.

2.2.7 Higher-order Markov Chains

A Markov model is a stochastic model, used to model randomly changing systems where it is assumed that the next state is only dependent on the current state. This memoryless property of a stochastic process is called the Markov property and displayed by Equation 2.6. Markov models express the probability to get from a current state into a certain next state (Barbour & Petrelis, 2008; Craven, 2011).

$$P(X_n = x_n | X_{n-1} = x_{n-1}, ..., X_0 = x_0) = P(X_n = x_n | X_{n-1} = x_{n-1}) \tag{2.6}$$

HMCs are used to model dependencies considering previous states. A Markov chain of order 2 for example, will take the previous step into account in order to calculate the transition probability for the next state. Therefore, a Markov chain of order k implies that the probability of switching into state $n + k$ depends on the k previous states. With this, HMCs are often used for sequence classification and time series

analysis and are therefore well suited for the analysis of clickstream data. The main disadvantage is that HMCs can become computationally very expensive as the order increases (Dewey, 2016).

The already mentioned study by Poggi et al. (2007) trained a DT and LR algorithm on weblog data transformed into two HMCs, one for purchasing and one for non-purchasing customers. To train the model the log files of the web shop were used, with seven important variables: Timestamp, session ID, type of accessed page, whether the customer was logged in, whether it was a returning customer and if the customer purchased or not (target variable). Next to this static information, the sequential dependency of the accessed pages by using second-order Markov chains was modeled. For training, only 7,000 transactions were utilized. The best result was obtained by a decision tree algorithm that achieved a recall of 0.78. No study could be found that made use of HMC as the actual prediction model instead of only a data pre-processing tool.

2.3 Implications

This section draws implications from the literature regarding the performance, depending on the used model as well as the type of utilized data. Resulting, gaps in literature are identified.

2.3.1 Algorithm performance

Comparing models' performance across papers, as displayed in Tables 2.1, is difficult since the utilized data differs both regarding content and dataset size. Especially the size differs largely ranging from 3,655 samples used by Korpusik et al. (2016) to 76,375,439 as mentioned in the paper of Lee et al. (2015) and presumably even more used by Lang and Rettenmeier (2017). It is further difficult since the models' implementations regarding hyper-parameters are not known and evaluation metrics and thresholds differ across papers. Looking at performance within the papers one can see that DTs, as well as RFs, show the best results. In Hop (2013) a DT and RF score first and second before SVM and an LR. Also, in the paper of Niu et al. (2017), an LR model is outperformed by an RF model. Similar results can be found in the paper of Poggi et al. (2007) with a DT performing slightly better than an LR. Another algorithm outperforming all other algorithms used on the same dataset is the RNN. In the paper of Lang and Rettenmeier (2017) it was tested against an FNN and LR and in the paper of Korpusik et al. (2016) only against an FNN. Even though these results indicate that DTs in general, as well as RNNs, might be best

suited for the task at hand, Wolpert, Macready, David, and William's (1995) "Free lunch theorem" states that there is no algorithm that universally performs best on all problems. Therefore, all models should be implemented and tested on the data and prediction task of this study to establish which one is best suited for this specific use case. Nevertheless, two models can be excluded, first, the KNN since it has slow prediction times while the use case asks for a real-time prediction. Further, the very good performance shown by Suchacka et al. (2015) is questionable since it is not compared to any other algorithm and only reports the best association rule. Though, this association rule is not useful, if the session step that is to be classified does not contain the same attributes. Secondly, the HMC can be excluded from further analysis since its complexity will be too high if multiple timesteps are considered. It has also not been applied for prediction in literature and was only used as a data preprocessing step by Poggi et al. (2007). As a result, the machine learning models that were being implemented and evaluated in the remainder of this thesis are boosted trees, RF, SVM, LR, FNN, and RNN.

2.3.2 Dataset performance

Not only the different algorithms influenced the performance of the classification, but also different data used for training the models. Three of the above-mentioned studies trained and tested algorithms with different data types and could show differences in the performance. Bogina et al. (2016) displayed that enhancing session data with additional item information could increase the performance slightly. Also, Poggi et al. (2007) observed that combining static and dynamic session data improves the performance compared to only using static information. Lee et al.'s (2015) results suggest that using dynamic session data highly improves the prediction compared to using static item information. Using both led to a slight increase compared to only using the session data. This thesis aims at replicating these results from the reviewed literature. Therefore, algorithms will be separately tested on static customer data, sequential and static session data and on a combined dataset containing both static customer data and static and sequential session data. Analog to the findings in literature the customer dataset is expected to yield the worst results followed by the dataset containing only session data. Combining the two datasets should result in the best prediction performance, where the difference in performance between customer and session data is expected to be much larger than between the session data and the complete dataset.

2.3.3 Literature gap

Even though the papers from Bogina et al. (2016), Lee et al. (2015) and Poggi et al. (2007) analyzed the impact that different data types can have on the results, none of the papers included reasoning on why static data performed worse compared to sequential data. This paper aims at giving some additional data insight to support these findings through an exploratory data analysis.

While all papers disclose results of their classification performances, no paper gives a detailed insight into the performance of the algorithms under different conditions, such as the difference between weekdays and the weekend, or between using a mobile device or a desktop computer for shopping. This thesis aims at exploring the prediction performance of the models under different conditions, to give a more realistic assessment of the actual performance in the real use case.

Lastly, none of the papers that assess stateful RNN models, such as the research by Korpusik et al. (2016) and Lang and Rettenmeier (2017) explore an analysis of feature engineering efforts, even though it is an important concern in e-commerce, since it is time-consuming and requires a lot of expertise and RNNs offer the possibility of reducing such. Resulting, this thesis will train the stateful RNN model on datasets with two different degrees of required feature engineering, to see how this influences the prediction performance.

Chapter 3

Methodology

This chapter contains an explanation of the methodology framework that was used to structure this study, followed by the explanation of evaluation metrics on which the model comparison will be focused. The chapter ends with the tool selection.

3.1 Research framework

The research framework used in this study is the Cross Industry Standard Process for Data Mining (CRISP-DM). CRISP-DM is a data mining model summarizing all important steps undertaken in a data mining project. It provides a structured approach to the planning and conduction of a data mining project. Being first presented and published in 1999 (Chapman, 1999) it remains one of the standard models today (Piatetsky-Shapiro, 2014). The model splits the data mining process into six phases, as shown in Figure 3.1. The order of steps is arbitrary and largely dependent on the outcome of the previous step. The arrows in the diagram describe the strongest relationships. The outer circle stands for the cyclic nature of data mining tasks: Learned lessons and solutions from a data mining project often lead to new business questions and trigger a new process (Chapman et al., 2000). The six different steps are explained below. A more detailed overview of the six stages can be seen in Table 3.1.

1 **Business understanding:** This phase considers project objectives from a business perspective. Generated insights are transformed into a data mining problem definition.

2 **Data understanding:** During the data understanding phase, data is initially collected and analyzed to generate first insights and for accomplishing familiarity with the data.

Table 3.1: Generic tasks in bold and output in italic of the six different phases of the CRISP-DM model. Retrieved from Chapman et al. (2000).

Business Understanding	Data Understanding	Data Preparation	Modeling	Evaluation	Deployment
Determine Business Objectives	**Collect Initial Data**	*Data Set*	**Select Modeling Technique**	**Evaluate Results**	**Plan Deployment**
Background	*Initial Data Collection Report*	*Data Set Description*	*Modeling Technique*	*Assessment of Data Mining Results w.r.t. Business Success Criteria*	*Deployment Plan*
Business Objectives	**Describe Data**	**Select Data**	*Modeling Assumptions*	*Approved Models*	**Plan Monitoring and Maintenance**
Business Success Criteria	*Data Description Report*	*Rationale for Inclusion / Exclusion*	**Generate Test Design**	**Review Process**	*Monitoring and Maintenance Plan*
Assess Situation	**Explore Data**	**Clean Data**	*Test Design*	*Review of Process*	**Produce Final Report**
Inventory of Resources	*Data Exploration Report*	*Data Cleaning Report*	**Build Model**	**Determine Next Steps**	*Final Report*
Requirements, Assumptions, and Constraints	**Verify Data Quality**	**Construct Data**	*Parameter Settings*	*List of Possible Actions*	*Final Presentation*
Risks and Contingencies	*Data Quality Report*	*Derived Attributes*	*Models*	*Decision*	**Review Project**
Terminology		*Generated Records*	*Model Description*		*Experience Documentation*
Costs and Benefits		**Integrate Data**	**Assess Model**		
Determine Data Mining Goals		*Merged Data*	*Model Assessment*		
Data Mining Goals		**Format Data**	*Revised Parameter Settings*		
Data Mining Success Criteria		*Reformatted Data*			
Produce Project Plan					
Project Plan					
Initial Assessment of Tools and Techniques					

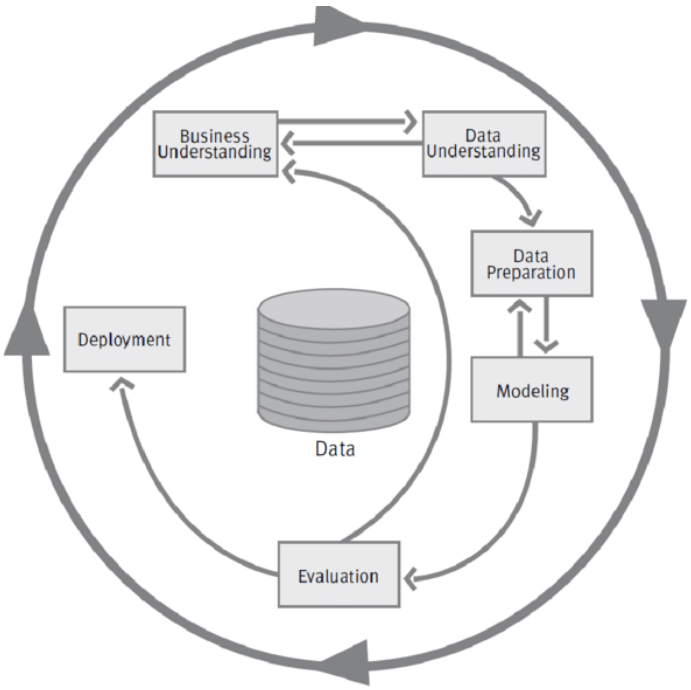


Figure 3.1: Phases of the CRISP-DM reference model. Retrieved from Chapman et al. (2000).

3 **Data preparation:** This phase entails all of the steps undertaken to generate the final dataset of variables from the initial raw data, which will serve as input to the modeling tools.

4 **Modeling:** In the modeling phase, modeling techniques are applied. This involves both model selection and further fine-tuning of the models' parameters. Since several techniques exist for modeling the same data mining problem various models can be considered.

5 **Evaluation:** After implementation, the models' performance has to be evaluated and compared. It is important to assess whether the goals, defined during the business understanding phase, are met.

6 **Deployment:** In order to actually benefit from the model it needs to be deployed. This requires for the model to be integrated in live systems and fed with live data, in order to make valuable predictions.

These six stages provide a guideline to the research at hand: *Business understanding* was established through the background information on customer classification in the e-commerce context in the introductory Chapter 1. This resulted in a data mining problem definition formulated as a research question in Section 1.2. Further, understanding of the actual prediction task was provided in the literature review on classification and machine learning algorithms in Chapter 2. A description of the data and a first insight leading to *Data understanding* will take place in Chapter 4. Chapter 4 also includes *Data preparation* in terms of data pre-processing as well as feature engineering and selection. *Modeling*, including hyper-parameter tuning and training, is described in the implementation Chapter 5. *Evaluation* and comparison of the different algorithms are performed in Chapter 6. The *Deployment* step is out of scope for this research, since it only focuses on comparing different algorithms in respect to the already implemented and deployed baseline model, as described in Section 1.2.

3.2 Evaluation metrics

This section describes the evaluation metrics used to compare the different algorithms. Resulting from the business understanding, the evaluation will be based on performance metrics, interpretability as well as prediction latency.

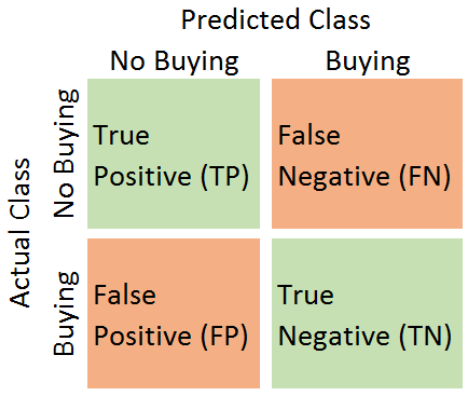


Figure 3.2: Confusion matrix, showing true positives, true negatives, false positives and false negatives.

Performance measures

Various measures exist to assess and compare the performance of machine learning models on a binary classification task. These metrics are based on the so-called confusion matrix, Figure 3.2, from which one can derive the correctly predicted cases, indicated in green, called true positives and true negatives. These are the cases where a visitor did not purchase anything and a no buying session was predicted and sessions where a purchase occurred and was also predicted. Also, the wrongly predicted cases can be identified, as indicated in orange, the false negatives, and the false positives, where a purchase occurred but none was predicted or where no purchase occurred but one was predicted.

From the confusion matrix, various performance metrics can be derived. All metrics calculated from the confusion matrix depend on the chosen classification threshold, based on which the confusion matrix was created. The threshold indicates which of the prediction results, being probabilities ranging from 0 to 1, is transferred to the positive or to the negative class. Choosing the threshold depends on the use case, since it influences the number of false negatives and false positives, therefore changing the values of the performance metrics. The most common metrics are accuracy and error, which are displayed in Equation 3.1 and 3.2. Accuracy describes the percentage of correct results, whereas the error rate is the number of wrongly classified results. Most classification models aim at achieving a high accuracy, or equivalently a low error rate (P. Tan et al., 2005). Accuracy is not always a good measure, especially not for imbalanced datasets. Better estimators which provide more information about the type of error, are precision, recall, and the F1-score. Precision, also called positive predict value, is the number of true positives divided by the number of all positive classified cases, see Equation 3.3. The

recall, also called sensitivity, is the number of true positives divided by all positives in the data set, see Equation 3.4. In the F1-score both recall and precision are considered equally as shown in Equation 3.5. (Manning, Raghavan, & Schuetze, 2008). Another very important measure is the specificity which stands in contrast to the sensitivity and measures the proportion of negatives that are correctly identified as such (Equation 3.6). The trade-off between these two can be modeled through the ROC AUC, which displays the effect of different thresholds on the two metrics. The ROC AUC score is, therefore, threshold-independent. An example of ROC curves is displayed in Figure 3.3. The straight line C displays a ROC AUC of 0.5 which corresponds to the probability of guessing and line A shows a perfect prediction with a ROC AUC value of 1. Line B shows a regular ROC curve with a value of 0.85, which approaches line A as predictions get better (Zou, OMalley, & Mauri, 2007). Most of the reviewed papers use accuracy and the ROC AUC as the performance indicator, since it provides a possibility to consider sensitivity and specificity and to nicely plot their dependency without worrying about the chosen threshold (Castanedo et al., 2014; Lo et al., 2014; Lang & Rettenmeier, 2017; Zhang et al., 2014).

$$Accuracy = \frac{Number\,of correct\,predictions}{Total\,number\,of\,predictions} = \frac{TP + TN}{TP + FP + FN + TN} \tag{3.1}$$

$$Error\,rate = \frac{Number\,of wrong\,predictions}{Total number\,of\,predictions} = \frac{FP + FN}{TP + FP + FN + TN} \tag{3.2}$$

$$Precision = \frac{Number\,of true\,positives}{Total\,number\,of\,positive\,predictions} = \frac{TP}{FP + TP} \tag{3.3}$$

$$Recall = \frac{Number\,of\,true\,positives}{False\,negatives + Number\,of\,true\,positives} = \frac{TP}{FP + TP} \tag{3.4}$$

$$F_1\,Score = \frac{2}{\frac{1}{Recall} + \frac{1}{Precision}} = \frac{2}{\frac{FN+TP}{TP} + \frac{FP+TP}{TP}} \tag{3.5}$$

$$Specificity = \frac{True\,negatives}{True\,negatives + False\,positives} = \frac{TN}{TN + FP} \tag{3.6}$$

There is no static rule to estimate which metric is best suited for a classification task, instead, it depends on the use case. For the case of this study, it is less important to identify every no buying session. It is more important that the ones who are identified as no buying sessions are really such since it would be a wastage to hand a gift card to someone who intended to purchase anyway. This fact is expressed through the precision and the specificity, which are therefore the most important measures. Nevertheless, most preferable are models that also consider the recall, since there are situations, where it is important to find all cases belonging to the positive class. For this, the ROC AUC is a very good measure since it considers

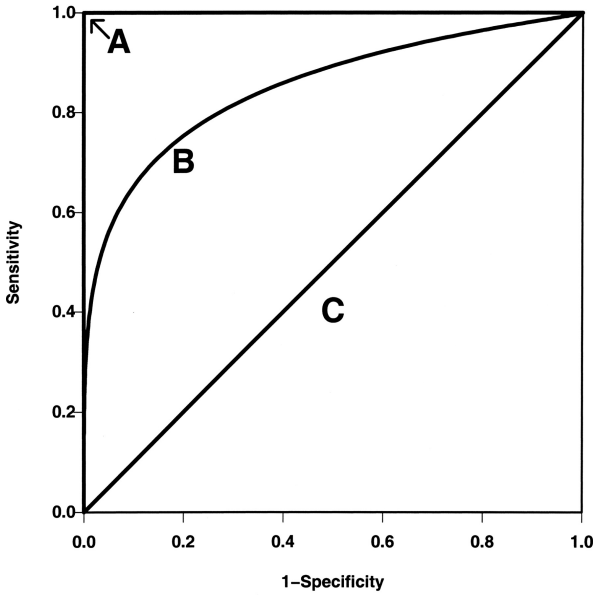


Figure 3.3: Three different ROC curves. A being the perfect curve, B a regular curve and C displaying the chance of guessing. Retrieved from Zou et al. (2007).

both the performance of specificity and recall. To make the results of all algorithms comparable it is important to use the same input data for each. Further, the classification threshold should be the same across all algorithms, in this case, it was decided to be set to 0.5. Lastly, to establish which algorithm performs best on which datatype the algorithms will be tested on three different datasets: The customer dataset, only compromising static customer data, the sequence dataset, containing time-depended clickstream data and static session information and the complete dataset consisting of the combined data.

Latency

Next to the model performance the latency is very important since the algorithms are intended to be used for deployment in the web shop to carry out real-time predictions, in order to immediately react to customer behavior. Here, the training times can be neglected since training is done in an off-line fashion and retraining is only intended on a regular basis with longer time intervals. Classification latency plays a crucial part, being the time between the data input and the models' output.

Comprehensibility

Next, to those different measurements, the models will be compared based on comprehensibility. As machine learning is applied in our everyday lives the demand for understanding the predictions is growing (Ribeiro, Singh, & Guestrin, 2016). With the General Data Protection Regulation law, taking effect in the European Union in 2018, users will even have the 'right for explanation', offering users the possibility to request explanations about algorithmic decisions (Goodman & Flaxman, 2016). Comprehensibility is, therefore, becoming very important. Since comprehensibility can be difficult to define and is very subjective it is not considered to be the main evaluation metric.

3.3 Tool selection

Python will be used for implementing the different machine learning algorithms. Python is a general-purpose high-level programming language (Python, 2017). It is used throughout the machine learning community also due to its many libraries that contain various predictive analytics algorithms. One of the most known libraries is Scikit-learn, which will also be used in this thesis (Sk-learn, 2017). It provides state-of-the-art implementations of many machine learning algorithms, while maintaining an easy-to-use interface and is therefore well suited for the research at hand (Pedregosa et al., 2011). For implementing the FNN and RNN, Keras will be used, a high-level neural network library for Python (*The sequential model API*, n.d.).

Chapter 4

Data

In this part, the data being used for the training and testing of the models is described. The utilized data is clickstream data from the web shop of a large German clothing retailer. The software that records the data, was developed in-house and stores the interactions of visitors with the web shop in high detail. This is in line with the definition of Dumais, Jeffries, Russell, Tang, and Teevan (2014) describing clickstream data as records that store user interactions with an application in a highly detailed manner. Further, a first data analysis is conducted to facilitate data understanding. The chapter concludes with the feature engineering as well as feature selection process.

4.1 Data description

Session data exists for each web shop session of a visitor. Such a session is defined as one visit to the web shop that times out after one hour of inactivity, then a new session starts. This approach of splitting interactions into sessions is a validated approach called gap sessions, described by Chen, Fu, and Tong (2004). Throughout literature, a 30-minute gap is more common (Chen et al., 2004; Stevanovic, Vlajic, & An, 2011; Suchacka & Chodak, 2016). To identify sessions, a cookie-based ID is generated, which changes after one hour of inactivity. Each session is split into a sequence of interactions with the web shop called landmark. A landmark always consists of at least the following structure:

landmark = attribute/value

where a landmark can take n different attributes with n different values. All attributes and values of a landmark belong to the same sequence number with the same timestamp to record different informations about one event. A new landmark is written every time an interaction with the web shop occurs or the web shop renders an action. The goal is to calculate the buying probability each time a new page

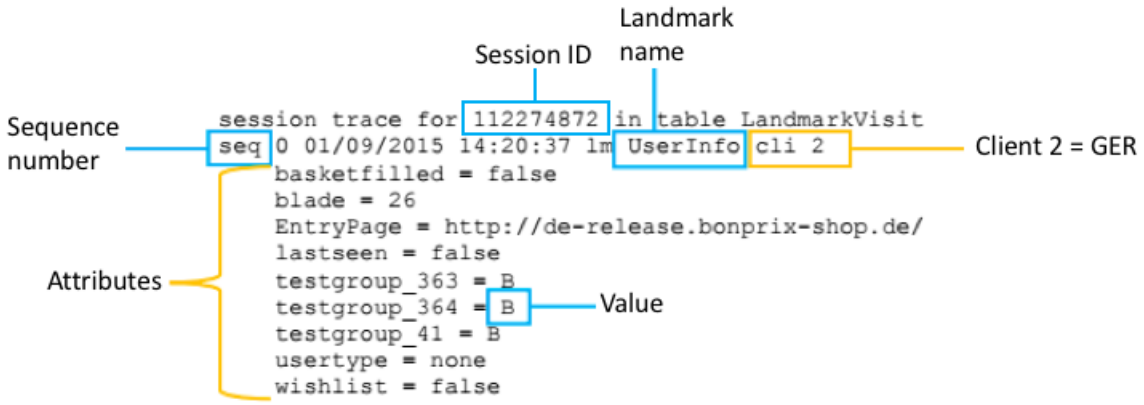


Figure 4.1: Example of a Landmark, with sequence 0, session ID 112274872, name UserInfo and some attributes with different values.

is opened or an item is added to the shopping basket and to then save it as an additional attribute of the current landmark, based on which business decisions, such as displaying gift cards, can be carried out. An example of a landmark can be seen in Figure 4.1. All landmarks are stored with a session ID, a timestamp, a sequence number, their name, attributes and values in a database. For training and testing the models, only the landmarks that described the customer, the session or customer interactions were kept, such as navigating to a page and adding something to the shopping basket, to reduce the size of the dataset.

Customer data only exists if the visitor could be identified. This means if a visitor actively logged in or was automatically logged in due to restoring a cookie. If the visitor is identified various data exists to describe the customer, such as the gender, the age, the postal code as well as purchase and activity history. All of this data is anonymous and does not allow for matching it to a certain person.

4.2 Data pre-processing

The models will be trained with session and customer data of the week from 31.07.2016 until 07.08.2016 since this data was used to train the baseline model and the same data should be used for all algorithms to facilitate a comparison as mentioned in the evaluation section in Chapter 3. Data was transformed such that each visitor action within a session was written to one row of the database.

From the data, the sessions not being caused by humans but by bots had to be removed. The detection and elimination of traffic generated by bots is an important task in clickstream analysis (Suchacka & Chodak, 2016). Fortunately, navigational

patterns generated by bots differ from the ones produced by humans (Stassopoulou & Dikaiakos, 2009; Suchacka, 2016). Resulting, it was decided to remove all sessions without cursor movement, or in the mobile case touch interaction. These sessions can be identified by analyzing the written landmarks, where the device information is stored and one can see that no landmark was written that indicates an interaction. This applied in about 50% of the sessions. It is very plausible that such patterns were caused by bots. In the cases where they have been caused by humans, it is most likely that the page was opened accidentally (e.g. if the tab was still opened on a smart-phone and appeared accidentally when the web browser was accessed). These cases are also not relevant for the use case; therefore, they can be removed along with the traffic caused by bots. Also, all traffic generated by employees caused by testing and development of the web shop, was removed prior to further data pre-processing steps. These cases are marked by specific landmarks that either indicate testing or deployment and can therefore easily be removed.

The first seven visitor interactions of each session were excluded since in the real-time use case prediction commences only after seven interactions to not show gift cards too early to the visitor. Interactions were also removed after a visitor purchased and then continued with the online shopping in the same session. This was done because in the use case, gift cards will not be shown to a visitor after having purchased once since receiving a gift card right after having purchased might upset the customer. This resulted in 3,841,913 remaining visitor interactions having occurred in 237,632 different web shop sessions of which 86.27% ended in no purchase. As input for the algorithms, not sessions as a whole, but each interaction within the session will be used separately. Buying sessions typically consist of more interactions with the web shop, therefore the percentage of interactions from no buying sessions is only 76.71% out of all interactions. This data, consisting of the clickstream generated by the visitor, was used as the sequence dataset. To create the complete dataset, customer information was joined to the sequence dataset once a visitor was identified. Before identification or if no identified occurred the customer values are NULL. Since customer information was solely joined to the clickstream data, the complete dataset consists of the same amount of records as the sequence dataset, but the single records consist of more data. Lastly, to create the customer dataset, only sessions could be used where a visitor was identified. Since this was only the case in 88,789 sessions, the customer dataset only consists of 88,789 records of which 81.28% belong to the no buying category.

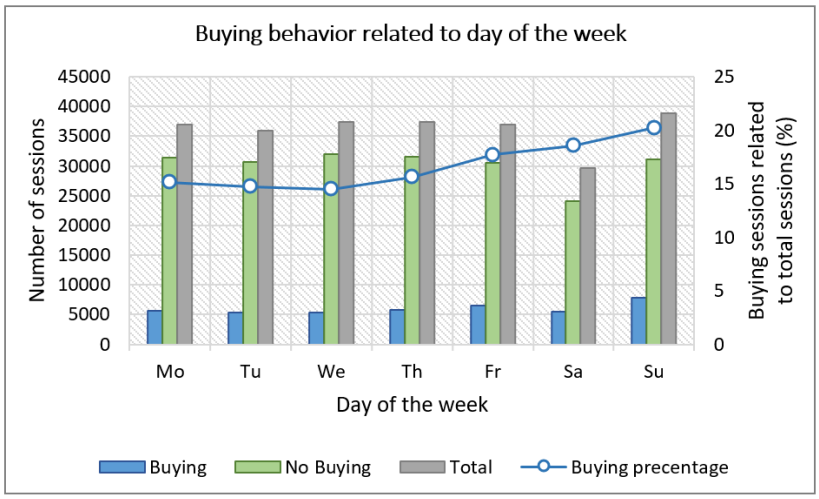


Figure 4.2: The influence of the weekday on the buying and no buying decision.

4.3 Data understanding

To generate a first understanding of both the sequence as well as the customer data a first exploratory data analysis based on Excel was conducted.

4.3.1 Sequence dataset

To provide a first insight into the sequence dataset, sequence data was chosen that remains constant throughout a session. These categorical data columns are weekday, device and entry channel. All of these variables were plotted in relation to the total number of buying and no buying sessions as well as in relation to the percentage of buying sessions out of all sessions. Buying and no buying were chosen since they are the two levels of the target variable. Therefore, the two plots both give an impression on how different variables influence the target variable of the prediction problem at hand.

From the bars displayed in Figure 4.2, one can see that the amount of buying and no buying sessions does not differ largely between the days of the week. For Friday and Sunday the total amount of buying sessions is slightly increased whereas for Saturday the no buying sessions are decreased compared to the other days. Looking at the percentage of buying sessions in regard to the total amount of daily sessions, displayed through the line plot in Figure 4.2, one can see that from Thursday on the percentage of buying sessions rises, with its peak on Sundays before it drops again on Mondays. This indicates that on the weekends including Fridays, the shopping behavior seems to differ from the weekdays. A possible explanation could be that during the week people only browse and look for nice items, hence less buying sessions, whereas on the weekends the actual purchase occurs.

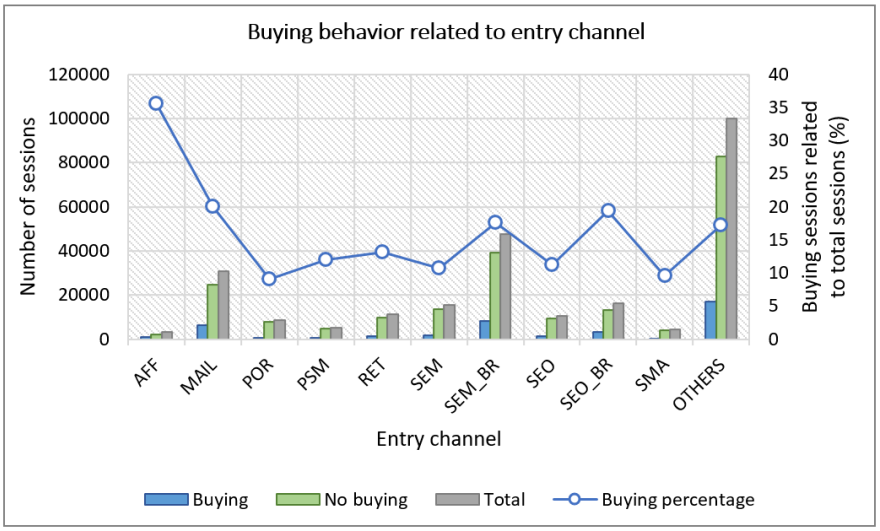


Figure 4.3: The influence of the entry channel on the buying and no buying decision.

Figure 4.3 shows the influence of the different entry channels on the buying versus no buying decision. To understand the graph better the different channels are explained below.

- **AFF:** Affiliate marketing, such as discounts or gift cards provided by third party websites
- **MAIL:** The e-mail channel, such as newsletters, service letters, etc.
- **POR:** Banner advertisement on websites
- **PSM:** Price search engine, a search engine that looks for the lowest prices
- **RET:** Retargeting, personal advertisement on third party pages
- **SEM:** Search engine advertisement, such as Google advertisements
- **SEM_BR:** Search engine advertisement brand specific, such as Google advertisements that are a result of a specific brand search
- **SEO:** Search engine optimization, meaning a web search engine's unpaid results
- **SEO_BR:** Search engine optimization brand specific, meaning a web search engine's unpaid results specific to a brand name
- **SMA:** Social media advertisement, such as advertisements on facebook
- **OTHERS:** Mainly entering the website directly, via entering the URL or bookmarks. Can also be smaller channels

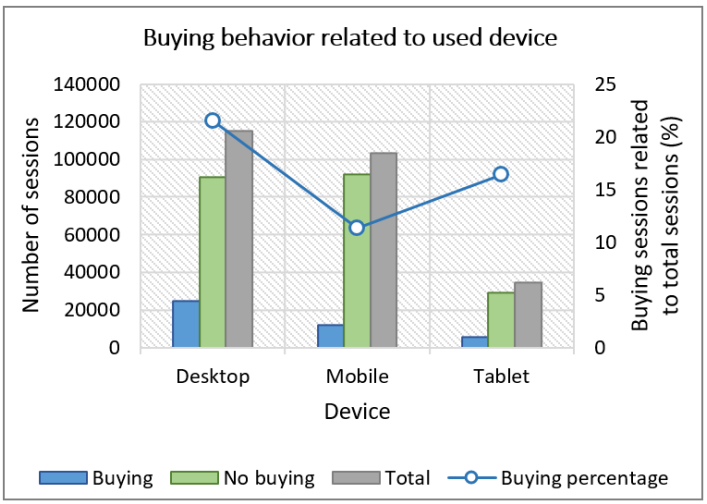


Figure 4.4: The influence of the device on the buying and no buying decision.

One can clearly see from the bars in Figure 4.3 that the MAIL, OTHERS and SEM_BR channel are the most used channels to enter the web shop. The large amount of sessions being entered through the OTHERS channel might be caused by the fact that under the keyword OTHERS a lot of channels are combined. The OTHERS channel also compromises a large number of cases where the web shop URL was entered directly into the browser. The high traffic through the OTHERS channel can therefore also be explained by the high number of catalog subscribers that manually enter the web shop URL in the browser after the catalog raised their interest. Looking at the line chart showing the buying percentage in Figure 4.3, one can see that AFF results in the highest percentage of buying sessions. This very high number is most likely due to the fact, that affiliate marketing entails gift cards and discounts, which result in more buying sessions. AFF is followed by MAIL, SEO_BR, SEM_BR and the OTHERS channel. SEM_BR and SEO_BR result probably in a higher buying ratio compared to SEM and SEO since the visitor was directed towards this specific online shop and not only towards an article that can be found in many other online shops.

In Figure 4.4 the variable device is plotted. Under the keyword mobile, the mobile app, as well as the mobile browser, are condensed. One can see from the bars in Figure 4.4 that desktop PCs and mobile devices are being used more than tablets. The most buying sessions occur on a desktop PC the least on a tablet. Looking at the line chart in Figure 4.4, one can see that the highest purchase percentage is on desktop PCs, followed by tablets. On mobile devices, the lowest percentage of buying sessions occurs. A possible explanation is that a mobile device might be used more often for browsing and selecting items. Whereas, larger devices, such as tablets and desktop computers, are preferred to conduct an actual purchase, possi-

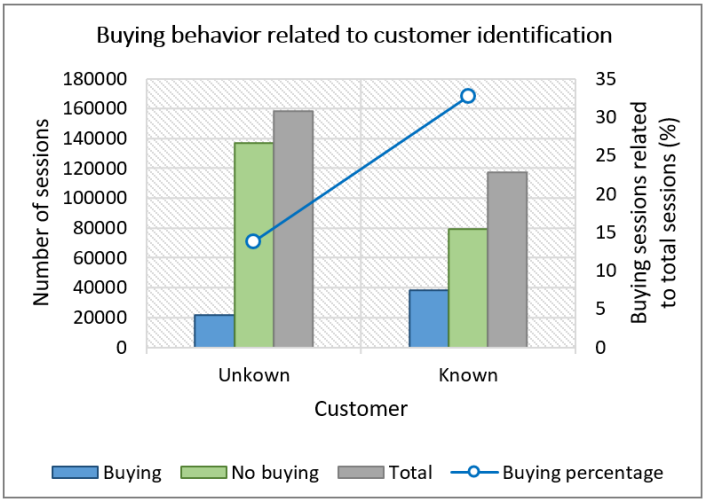


Figure 4.5: The influence of the customer identification on the buying and no buying decision.

bly due to the fact that the size facilitates the input of login and payment information.

Lastly, to analyze which effect the identification of the visitor during the web shop session had on the buying decision, the customer identification variable was plotted. Results are displayed in Figure 4.5. The bars show that the number of identified visitors is smaller than the number of unknown ones, which was expected. It can also be seen that identified visitors buy more often and abort the shopping process less compared to the unknown visitors. The line chart displays that the percentage of buying sessions is more than 15 percent higher with known visitors than with unknown ones. The results indicate that the information whether the visitor could be identified or not can enhance predictive performance since the two different levels of the variable influence the buying decision differently.

4.3.2 Customer dataset

As a next step, some variables from the customer dataset were analyzed in the same fashion, to generate an understanding of the customer data. For analysis, the variables age, gender, and last device were chosen. They can be best plotted since they do not take too many different values. Gender and device are categorical variables and age takes values between 18 and 88.

Figure 4.6 shows the analysis for the last device being used by the visitor in a previous shopping session. Again, the desktop device was most commonly used in previous visits closely followed by the mobile devices, tablets were used the least. Regarding the buying percentages, desktop PCs and tablets result in the highest buying percentage. The pattern matches exactly the results from plotting the variable device. This variable does therefore not add any new information for solving

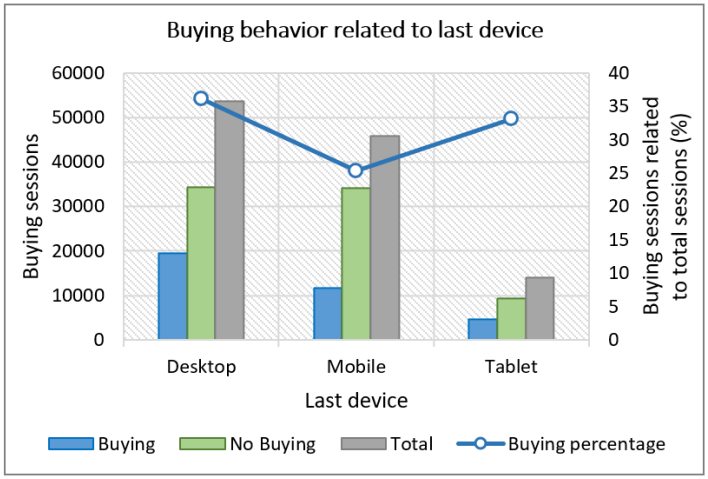


Figure 4.6: Figure (*a*) and (*b*) showing the influence of the last device on the buying and no buying decision.

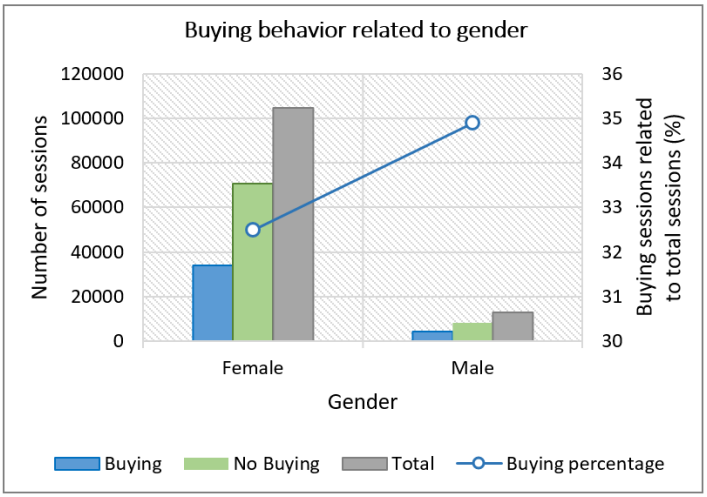


Figure 4.7: The influence of the gender on the buying and no buying decision.

the prediction task.

In Figure 4.7 the influence of the gender on the buying decision is displayed. From the bars in Figure 4.7, one can clearly see that more women visit the web shop than men. The total number of women is about ten times higher than the total number of men, also the number of buying and no buying sessions conducted by women is around ten times as high as the one of men. Looking at the buying percentage, displayed as the line in Figure 4.7, one can see that the percentages differ by less than three percentage points. The buying percentage of men is only slightly higher than the one of women, 32.5 compared to 35 respectively. This variable does therefore also not offer a lot of predictive power about the buying decision.

Lastly, in Figure 4.8 the influence of the age variable is displayed. In Figure 4.8 the bars show that the most customers are between 26 and 52 years old, where

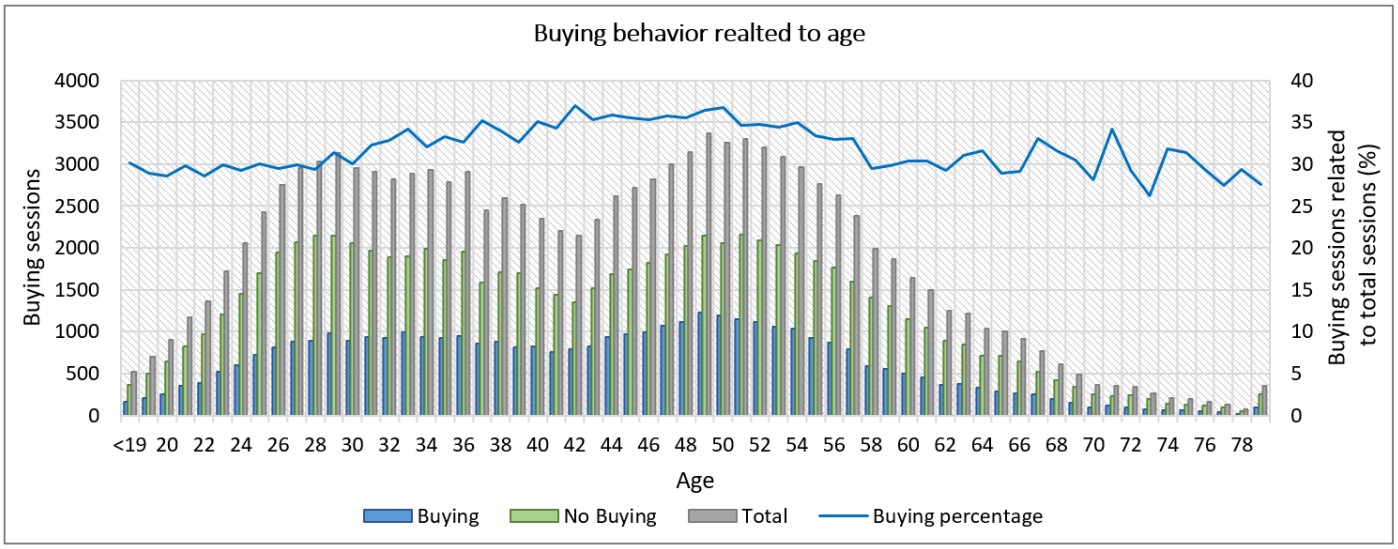


Figure 4.8: The influence of the age on the buying and no buying decision.

around 40 years the number of customers slightly drops, resulting in a binomial distribution with peaks at 26 and 52. For customers younger than 26 the number of customers decreases as the age decreases, analogously for people older than 52 the number of customers decreases as the age increases. The small rise for people around 80, results from aggregating all visits for ages older than 80, a so-called boundary effect. The buying and no buying behavior follows a similar curve as the total number of visitors. This relationship becomes clearer from the plotted line, displaying the buying percentage. The ratio of buying sessions is around 30% for all ages, only between the ages 33 and 57 the ratio rises to around 35%. The fact that the ages do not largely seem to differ in their buying and no buying behavior, especially regarding the ages that occur the most, results in the age variable not having a high predictive power.

In general, looking at the customer features, the data suggests that it is of high importance to the purchase prediction whether a visitor could be identified or not. The specific customer attributes themselves do not add a lot of information and are therefore not very influential on the purchase decision.

4.4 Features

This part describes the feature engineering and feature selection to generate input for the machine learning algorithms.

4.4.1 Feature engineering

Feature engineering is the process of using domain knowledge and expertise to extract features from data that represent the underlying problem to be used as input for machine learning models. It is a not well-defined process, nevertheless, it is one of the most important steps in the development of a machine learning model. No matter how good the model is, it will not be able to predict correctly if the input features are not representative of the task. Feature engineering can be problematic since engineered features are often over-specified and incomplete (Castanedo et al., 2014).

In this study, 28 features were retrieved from the clickstream data to capture session characteristics and 20 features from the customer data to describe the customer. All of the 48 features, split by sequence and customer features, can be seen in Table 4.1. The table shows feature names, a short explanation, the data type and whether a feature can be nullable. It is further indicated which features remain constant throughout a whole session. Values for features that are not constant, always describe events that occurred in the previous step. The green variable is the target variable, it, therefore, does not belong to the feature set.

Except for the RNN, none of the to be implemented algorithms can model time sequences. Therefore, certain features were explicitly created to capture the time dependencies in the data, for example time_delta_lastpage, time_delta_lastWK, and sessiondauer. These algorithms also require heavier feature engineering than the RNN, as a consequence the feature set contains sums, averages, maximum and minimum calculations. Further, each subsequent step in a session contains accumulated information about the previous steps, such as the total of visited pages, the total duration up to this point, the total amount of viewed article detail pages, etc. All of this should not be necessary when using an RNN, since it uses entire sequences as input. Therefore, a second simplified feature set was created which still compromises all customer features but fewer sequence features. It only contains the type of visitor interaction, the visited page and some additional features containing page information such as the article price. These features are related to a certain type of page and are therefore often NULL. For example, adaP, the price of an article, only takes a value if an article detail page was visited. All other features can be retrieved from these. The feature set, therefore, contains the same information as the large dataset, but refrains from explicitly modeling variables, enabling to be closer to the actual clickstream data by using minimal feature engineering. To see whether the RNN can learn all other features by itself, and therefore decrease the feature engineering effort, it will be trained and tested on the regular as well as the RNN dataset and the results will be compared. The feature set used for the RNN can be seen in Table 4.2. Customer features are left out in this table since they are redundant to

Table 4.1: Engineered features, with name, description, value and whether it can be not defined. A feature with * remains constant throughout a session. Orange features were excluded in the feature selection phase. The green feature is the target variable.

	Variable	Description	Values	Nullable
Sequence Data	bb*	Has an order occured in the session?	0, 1	no
	step	Number describing each step	int	no
	device*	Used device (Desktop, Mobile, Tablet)	int (1, 2, 3)	no
	aktion	Customer action	str (page, addWK)	no
	wkss	# of articles in shopping basket at session start	int	no
	page_cnt	# of opened pages up to this point	int	no
	ada_cnt	# of article opened detail pages up to this point	int	no
	starthour*	Hour when session started	0 - 24	no
	sessiondauer	Duration of session up to this point	int	no
	Checkout_login*	Did a checkout occur in the session?	0, 1	no
	time_delta_lastpage	Time difference to last page	int	no
	time_delta_lastWK	Time difference to last shopping basket addition	int	yes
	myaccount_page	Was the my account page visited?	0, 1	no
	myaccount_login	Was the my account login popup filled in?	0, 1	no
	checkout_login	Was the checkout login entered?	0, 1	no
	bestellkarte	Was the order form used?	0, 1	no
	wday*	Day of the week (Mon - Sun)	str (mo, tue, we, ...)	no
	kanal*	Channel through which the page was entred	str (AFF, MAIL, POR, ...)	no
	lastentryrej	In case of reentering, channel used before	str (AFF, MAIL, POR, ...)	yes
	firstpage*	First page that the customer entered	str (home, basket ...)	no
	lastpage	Last page that was visited	str (basket, home ...)	no
	sum_adaP	Sum of prices of all viewed article detail pages	float	no
	avg_adaP	Average price of all viewed article detail pages	float	no
	max_adaP	Max price of all viewed article detail pages	float	no
	min_adaP	Min price of all viewed article detail pages	float	no
	sum_basketP	Sum of article prices in shopping cart	float	no
	avg_basketP	Average price of articles in shopping cart	float	no
	max_basketP	Max price in shopping cart	float	no
	min_basketP	Min price in shopping cart	float	no
Customer Data	cutomer_type*	Type of customer (new/old customer, ...)	0 - 7	yes
	gender*	Gender	0, 1, 2	yes
	PLZ1*	First number of Zip Code	1 - 9	yes
	lastBB*	Last time of an order (in days)	int	yes
	firstBB*	First time of an order (in days)	int	yes
	lastRET*	Last time of a return (in days)	int	yes
	anzBBOL*	# of online orders	int	yes
	anzBB*	# of orders	int	yes
	anzRET*	# of returns	int	yes
	sumRET*	Total price of returned articles	float	yes
	visits12*	# of sessions in last 12 months	int	yes
	lastDevice*	Last device used to visit the page	0 - 3	yes
	lastADS*	# of viewed article details pages in last session	int	yes
	lastPages*	# of visited pages during last visit	int	yes
	lastWK*	Shopping basket filled in last session	0, 1	yes
	lastVisit*	Days since last visit (days)	int	yes
	ALTR*	Age	16, ..., 99	yes
	Kunde_seit*	Customer since (days)	int	yes
	sumKS12*	Total price of bought articles (last 12 month)	int	yes
	sumRET12	Total price returned articles (last 12 month)	int	yes

Table 4.2: RNN features, with name, description, value and whether it can be not defined. A feature with * remains constant throughout a session. The green feature is the target variable. Customer data corresponds to customer data in Table 4.1

.

Variable	Description	Values	Nullable
bb*	Has an order occured in the session?	0, 1	no
event_type	Which customer interaction occured	str(page, basketadd, ident, ...)	no
page_category	Category of visited page	str(home, canal, search, ...)	yes
adaP	Price of a viewed product	float	no
basketP	Price of article put in shopping cart	float	no
entry_kanal	Channel through which customer entered	str (page, addWK, ...)	yes
device*	Used device (Desktop, Mobile, Tablet, App)	0 - 3	no
wkss	Number of articles in shopping cart at start	int	no
sessiondauer	Time difference to last event	int	no
wday*	Day of the week (Mon - Sun)	str (mo, tue, we, ...)	no
Customer Data	All customer data	various	yes

the ones used in the other feature set. Next to engineering the features, the dataset also requires further processing to make it suitable as input for the machine learning models. All algorithms, except the ones belonging to the DT category, require to replace missing values and to encode categorical variables, which are both described in the following. Some algorithms even require further data pre-processing, these specific steps will be described in the implementation section of each algorithm.

Categorical variables

Most algorithms require for categorical variables to be encoded. This concerns the features device, aktion, wday, kanal, lastentryrej, firstpage, lastpage, gender, lastDevice and the RNN features event_type, page_category and entry_kanal. Most commonly two different methods are used for encoding, namely label and one-hot encoding. With label encoding each unique category of a feature gets an integer value assigned. These values have a naturally ordered relationship, which machine learning algorithms are able to learn. This is therefore well suited for ordinal variables. For categorical variables without an ordinal relationship, label encoding is not suitable. It could result in wrong predictions since an order of categories is assumed which is nonexistent. Here, one-hot encoding can be applied. This entails that a binary variable is added for each unique category (Brownlee, 2017). Since all of the categorical variables, in this case, are on a nominal scale and some algorithms

such as SVMs perform better on one-hot encoded data (Hsu et al., 2003), one-hot encoding was applied to the data.

Missing values

Lastly, missing values had to be imputed. The most common techniques to solve the problem of missing values is to either remove, predict or impute them. Removing records with missing values is only possible if the missing values occur completely at random (Gelman & Hill, 2016). This is not the case here since the absence of customer data does not occur randomly. Whether predicting or imputing leads to better results depends on the use case. Here imputation was chosen since it showed the best results in the paper of Hop (2013). For algorithms that do not require normalization of the data such as boosted trees and RFs unique value imputation was chosen analogously to the paper of Hop (2013). This means replacing missing values by a value that does not occur in the dataset, in this case -100. Next to good results this method provides speed and simplicity. Unique value imputation should not be used for algorithms that require normalization of the data such as SVMs and FNNs, since the imputed values, being outliers, have a large influence on the normalization process. For these algorithms, missing values were imputed with the mean values of the corresponding features. Hence, a mean value imputation was performed.

4.4.2 Feature selection

Feature selection is the process of retrieving a subset of relevant features from the before engineered features. The aim is to remove redundant or irrelevant features to simplify the model, shorten training times, and reduce dimensionality and the chance of over-fitting (James, Witten, Hastie, & Tibshirani, 2013).

After closely analyzing the above-constructed features, six features were excluded resulting in a feature set of 43 features, the excluded features are marked in Table 4.1 with orange color. Step was removed since it is redundant with page_cnt. Customer_type, PLZ1 and start hour were removed because all of them are categorical variables with many categories that increase the complexity of the algorithms a lot, while not increasing prediction performance. Further, Customer_type contained too detailed information about the customers, raising privacy concerns. Checkout_login had to be removed since it was a trivial solution to the classification task. If it is known that someone logged into the checkout process, it is very likely that a purchase indeed occurred and the session was not aborted. It is also not available in the real-time implementation of the model. Trees facilitate the analysis

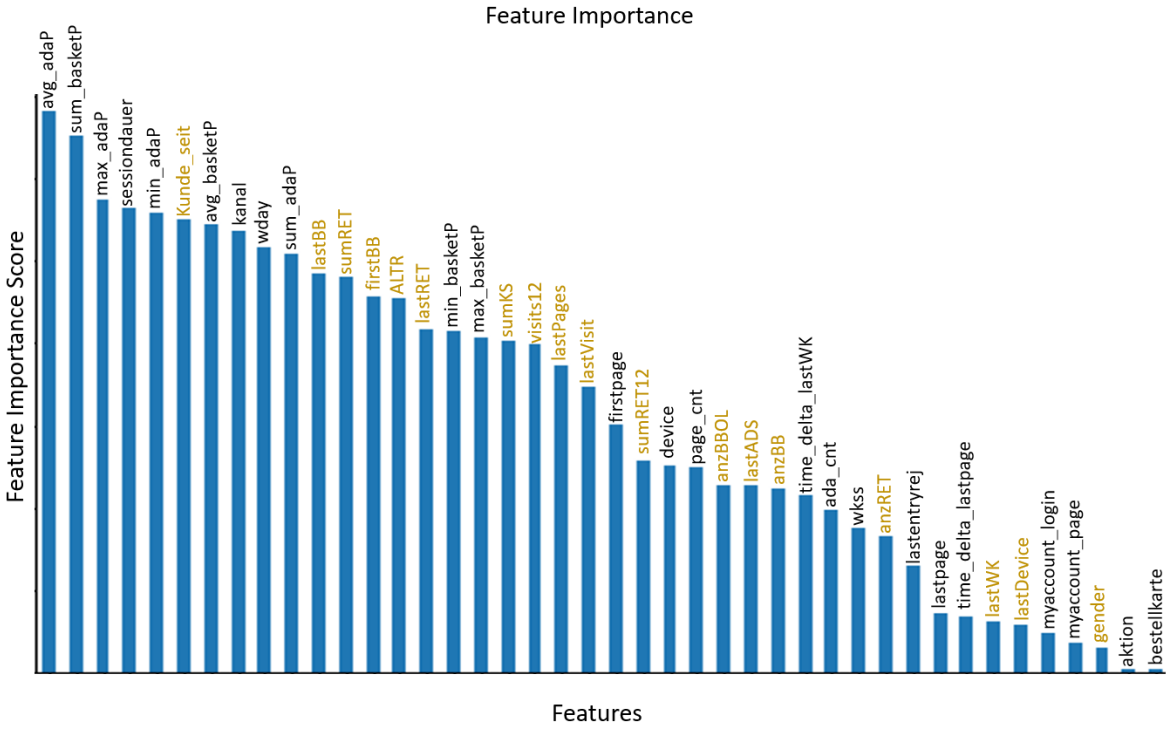


Figure 4.9: Importance of the 43 selected features as shown by the boosted tree baseline model. Customer features are marked in yellow font and session features in black.

of variable importance. The 43 final features were inputted into the already existing boosted tree model and variable importance was assessed. Results are shown in Figure 4.9. It can be seen that certain features are by far more important than others. Most important is the session feature *avg_adaP*, which is the average price of all viewed articles. *Aktion* (the type of interaction) and *bestellkarte* (if an order form was used) are the least important. Nevertheless, since all features seem to provide some information gain, it was decided to keep all of them.

Chapter 5

Implementation

5.1 Implementation and hyper-parameter tuning

This section describes the implementation as well as hyper-parameter tuning for all implemented algorithms. Hyper-parameter tuning is only conducted on the complete dataset, since it is the dataset that, based on literature, is expected to yield the best results. Running hyper-parameter tuning on all datasets was impossible due to time and computational constraints.

Boosted Tree

The boosted tree was built analogously to the already implemented boosted tree model since it is the baseline to which other algorithms will be compared. The already used model was built in R, a programming language used for statistical computing and graphics (*R: The R Project for Statistical Computing*, 2017). Since all other algorithms will be built in Python it was decided to rebuild the baseline model, to ensure that performance differences are not due to used software and libraries, but to actual differences of the algorithms. The original model was built with the XGBoost library for R, which is a gradient boosting framework including a linear model and a tree learning algorithm (*XGBoost R Tutorial*, 2016). Fortunately, the same algorithm exists in Python (*Scalable and flexible gradient boosting*, 2016). The model could therefore simply be transferred to the new programming language. Hyper-parameter tuning was already done when the algorithm was first engineered through grid search. These same parameter settings were used in Python and are explained below. As already mentioned in the literature research, boosted trees have the advantage of not requiring too much hyper-parameter tuning.

- max_depth = 12: Describes the maximum depth of a tree, which is used to

control over-fitting, since higher depths will result in learning very specific relationships (Jain, 2016).

- learning_rate = 0.2: Describes the learning rate of the model. It leads to an increased robustness by shrinking the weights in each step (Jain, 2016).
- min_child_weight = 100: Defines the required minimum sum of weights of observations in a child node. It is useful to control over-fitting and under-fitting. Higher values prevent the model from learning too specific relationships, whereas too high values could result in under-fitting (Jain, 2016).
- num_rounds = 400: Number of learning iterations carried out by the algorithm. A larger number results in a better performance while increasing training times.

All other tunable parameters were set to their default settings. For completion, the Python code of setting the hyper-parameters can be found in Appendix A Figure A.1.

Random Forest

To build the RF the implementation of the scikitlearn library RandomForestClassifier was used (*RandomForestClassifier*, 2017). Since RFs are not very prone to over-fitting, (Breiman, 2001) they do not require a lot of hyper-parameter tuning. Mainly, there are three different parameters influencing prediction performance:

- max_features: Describes the maximum number of features the algorithm tries for an individual tree. Generally, a higher number of features increases performance since the number of options at each node is increased. However, this does not necessarily hold true since an increase in features will also lead to a higher correlation of the individual trees (Srivastava, 2015). According to Geurts, Ernst, and Wehenkel (2016) a good solution for classification problems lies around the square root of the number of features. However, to find the most optimal solution each case has to be tested.
- min_samples_leaf: Meaning the minimum amount of samples in the end nodes of the tree, where a smaller leaf size can lead to the effect of capturing more noise in the data.
- n_estimators: Describes the number of trees to be trained. This parameter has no local optimum; therefore a larger number always leads to an increase in performance (Hop, 2013). This is in contrast with the goal of decreasing training and prediction speed (Srivastava, 2015).

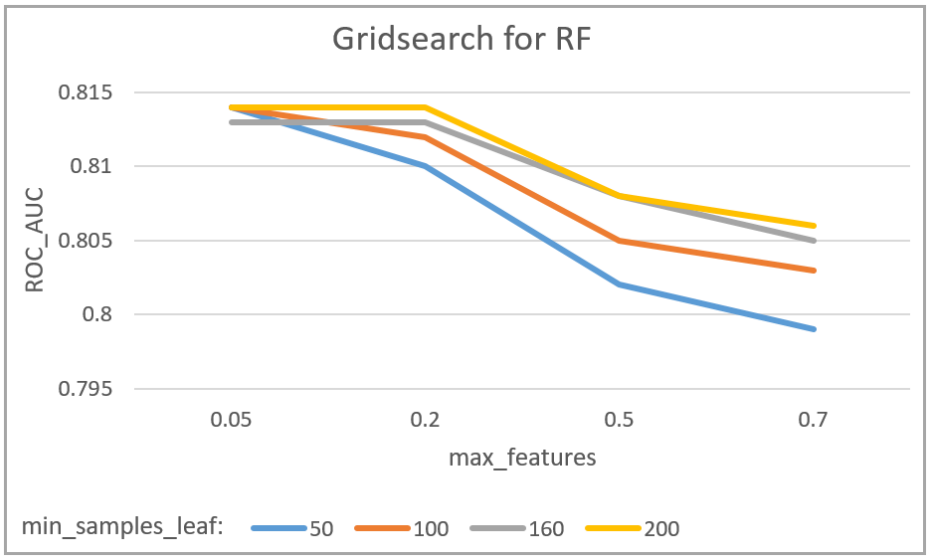


Figure 5.1: ROC AUC results of the grid search for the parameters max_features and min_samples_leaf for the RF algorithm.

For n_estimators, a number was chosen large enough to show a good performance while still resulting in reasonable training times. This number was 400, which is slightly bigger than the 300 trees chosen by Hop (2013). To estimate the values for max_features and min_samples_leaf that maximize the ROC AUC score, a grid search was carried out with values of 0.05, 0.2, 0.5 and 0.7 for max_features and 50, 100, 160 and 200 for min_samples_leaf. A grid search is the testing of all possible parameter combinations and retrieving the best setting through cross-validation (Hsu et al., 2003). Here, grid search was conducted with only 300 estimator trees. This leads to overall worse results than training with the 400 trees, but still shows the effect of the different parameter settings on the performance, while maintaining reasonable training times. From the results, displayed in Figure 5.1, one can see that increasing the number of samples per leaf led to a slight performance increase. This effect becomes less for smaller numbers of max_features. The yellow line representing 200 samples per leaf which showed the overall highest ROC AUC score reaches a plateau between a factor of 0.2 and 0.05 for max_features, resulting in 9 or 2 features of the 43 features respectively. These numbers are close to the suggested square root rule of Geurts et al. (2016), which would result in 7 features. Based on these results min_samples_leaf was set to 0.2 and max_features to 200 for training the final RF model. For completion, the Python code for setting the RF hyper-parameters can be found in Appendix A in Figure A.2.

Support Vector Machine

The implementation of the SVM follows the main constraint that training time complexity is more than quadratic with the number of data samples. This results in the fact that the utilized scikitlearn implementation of the algorithm does not scale well to datasets with more than a couple of 10,000 samples (*SVC*, 2017). It was decided to use 100,000 data samples to train the algorithm since it was the highest number of samples still leading to reasonable training times. The input data needed to be scaled to values ranging from 0 to 1. Since SVMs are prone to misclassifying the minority class in unbalanced datasets (Liu, An, & Huang, 2006) and only a subset of the data was used to train the SVM anyways, the subset was sampled such that buying and no buying sessions were equally present.

The effectiveness of an SVM depends on the used kernel, its parameters, and the soft margin parameter C. Hsu et al. (2003) suggest to use the RGB kernel. This kernel only has one kernel parameter, namely γ. The according formula can be found in Chapter 2 in Section 2.2.2. This, therefore, leaves C and γ for hyper-parameter tuning, which was again accomplished by grid search. Values for γ were 0.0001, 0.001, 0.01 and 0.1 and C was tested with values 0.0001, 0.001, 0.01 0.1 and 1. The results in Figure 5.2 show that the highest value was obtained with a C value of 1 and a γ of 0.1. In general, the configurations with a C value of 1 resulted in the highest ROC AUC score, the smaller C got the further the score decreased. For the final implementation C of 1 and γ of 0.1 were chosen. For completion, the Python code for setting the SVM hyper-parameters can be found in Appendix A in Figure A.3.

Feed-forward Neural Network

For building the FNN the Sequential model implementation of Keras was used (*The sequential model API*, n.d.). This implementation provides a framework, which leaves the possibility for intensive hyper-parameter tuning. Hyper-parameters used in FNNs can be split into two categories, determining the training and the structure of the network. Important training parameters are the loss function, the number of epochs and the batch size. The loss function compares the network's output against the intended ground truth output (*Neural Network Hyperparameters*, 2015). Here, the chosen loss function is binary_crossentropy, which is commonly used for binary classification problems. The epochs describe the number of times the dataset is passed through the network, where too small numbers result in under-fitting and

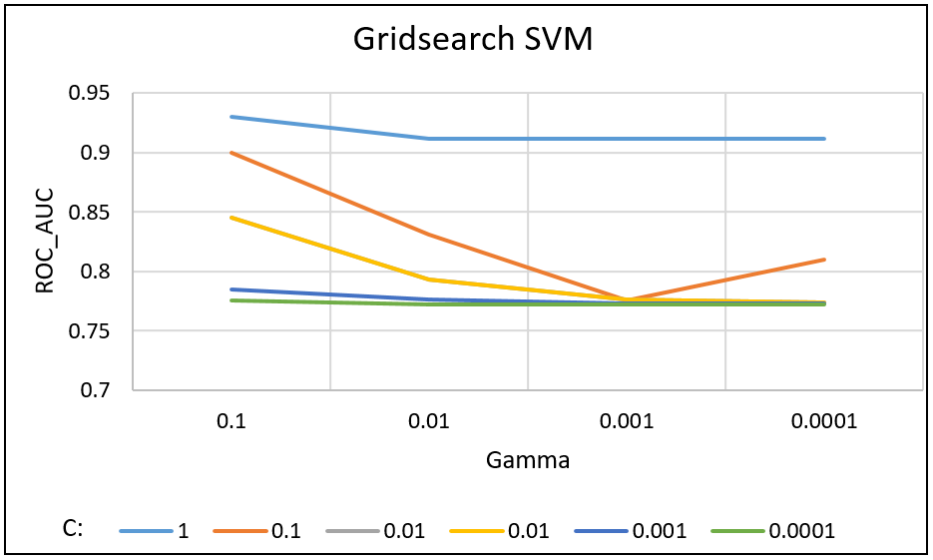


Figure 5.2: ROC AUC results of the grid search for the parameters C and γ.

too high numbers in over-fitting. After testing various configurations the number of epochs was set to 50 since for higher numbers the prediction performance on the test dataset decreased again. Lastly, since, due to memory constraints, not all data can be passed through the network at once, a batch size has to be chosen. If multiple data points are passed to the network, the average of all resulting gradients is used for updating. This yields the advantage of not capturing as much noise as updating for every single data point. The FNN was tested with different batch sizes where a batch size of 1012 yielded the best results and was still computationally feasible. Hyper-parameters regarding the structure of the FNN are concerned with the size and non-linearity of each layer as well as the total amount and configuration of different layers. Tuning these parameters results in too many different configurations for conducting a grid search. Therefore, a coordinate descent was used, where only one parameter is adjusted at a time to reduce the validation error (*Neural Network Hyperparameters*, 2015). This resulted in a final architecture with five dense layers of sizes 700, 350, 200, 64 and 1 separated by three dropout layers, to reduce over-fitting. The complete architecture in Python code can be found in Figure A.5 in Appendix A.5.

Logistic Regression

For implementing the LR the LogisticRegression model from sklearn was used. LRs make certain assumptions about the input data in order to perform well. One of which is that there is only a little or no multicollinearity among the independent vari-

ables (*Assumptions of Logistic Regression*, 2017). Therefore, a PCA was carried out before inputting the data into the algorithm. PCA is used to convert a set of possibly correlated variables into linearly uncorrelated variables, which are called principal components. Unfortunately, the PCA did not improve model performance, therefore, it was excluded in the final implementation. Regarding hyper-parameters, only C can be tuned. Small values of C result in a higher regularization strength which under-fits the data. Whereas a higher value for C results in more complex models that are likely to over-fit the data. Conducting a grid search with values of 0.1, 1, 10 and 100 revealed that C did not influence the model's performance. Therefore, C was set to 1 for the final implementation. For completion, the Python code for setting the LR hyper-parameters can be found in Appendix A in Figure A.4.

Recurrent Neural Network

To build the RNN the recurrent layers implementation of Keras was used (*RNN*, 2018). As for most other algorithms the data has to be normalized before it can be used as input. But the RNN requires even more data pre-processing because it needs sequences as input instead of feature vectors. While Lang and Rettenmeier (2017) used the complete length of a session as the length for the input sequences, where shorter sessions are padded with zeros to achieve the same length for all inputs, this method is computationally infeasible in this study. Here, the most occurring interactions within one session were 204,177, to pad all other sequences to this length is impossible on a regular desktop computer. It further does not contain a lot of information content since most sessions consisted of fewer interactions. Finally, the sequence length was set to 20 since it was the highest number that still led to reasonable computation times. For each session sequences had now to be formed. The first interaction in a session was chosen, represented as a feature vector, just like the input for the other models, and padded with 19 zeros. This resulted in the first input of a session. Then the feature vectors of the first and second interaction were chosen and stacked behind each other, padded with 18 zeros they served as second input to the RNN. This was repeated so forth for all interactions of all sessions. If a session was longer than 20 interactions, the oldest interactions were removed from the input sequence as a new feature vector was added. This resulted in input matrices of the shape: number of samples x sequence length (20) x number of features.

After preparing the input for the RNN, hyper-parameters could be tuned. As for the FNN, one can differentiate between hyper-parameters that determine the training and hyper-parameters that determine the structure of the network. The most

important parameters for training, are again the loss function, the number of epochs and the batch size. For the loss function this time categorical_crossentropy was chosen since it showed better results than the binary_crossentropy. The target variable was therefore transformed into a categorical variable, with the two categories buying and no buying. The batch size was set to 1200 because setting the batch size higher or lower showed worse results. Lastly, the number of epochs was set to 16, for an optimal training this amount is not sufficient but due to computational constraints, this number could not be set any higher. It can be expected that more training epochs would yield better results. For determining the training parameters coordinate descent was used as for the FNN. This resulted in a final architecture of one LSTM layer of size 400, three dense layers with sizes 300, 100 and 2 and two dropout layers to reduce over-fitting. The complete architecture can be seen in Figure A.6 in Appendix A.

Chapter 6

Results

In this section, the prediction results achieved by the different algorithms are displayed. First the performance results on the three datasets are explained, followed by a robustness analysis for the best algorithm. Then the prediction speed of the different algorithms is evaluated and finally, the comprehensibility is discussed.

6.1 Performance results

The performance results are evaluated separately for the three different datasets. Further, an analysis of the less feature engineered RNN data is conducted. The metrics were chosen as discussed in Section 3.2. Precision, recall, F1-score, accuracy and ROC AUC score were evaluated with precision being the most important metric for the use case and ROC AUC giving the best impression on the overall performance across different thresholds. The chosen threshold was the same across all datasets and algorithms and was set to 0.5. For completion, the confusion matrices of all algorithms on all datasets from which the performance results were calculated can be found in Appendix B.1 in Figure B.1.

In Table 6.1 the results for the complete dataset are displayed. The results for the buying and no buying class as well as the overall results are shown. The use case asks for predicting the no buying class, hence performance on this class is more important. Overall, it can be seen that the predictions for the no buying class were better than for the buying class, which most likely results from the unbalanced dataset, but can also be due to the fact that the no buying class contains 'easier to detect cases' such as short sessions or sessions with empty baskets. The LR achieved the highest precision score for the no buying class with 0.86. The SVM could achieve the highest recall for this class with a result of 0.96, which results from neglecting the buying class, with a recall of only 0.1. The best F1-score for the no buying class, as well as the highest accuracy and ROC AUC was achieved by the

Table 6.1: Results of all algorithms for the complete dataset with threshold 0.5.

All Data	Threshold: 0.5					
Algorithm	Class	Precision	Recall	F1-score	Accuracy	Roc Auc
Boosted tree	**no buy**	0.79	0.86	0.82		
	buy	0.62	0.5	0.56		
	total	0.73	**0.75**	0.74	0.75	0.79
RF	**no buy**	0.8	0.86	**0.83**		
	buy	**0.64**	0.64	0.59		
	total	0.75	**0.75**	**0.75**	**0.76**	**0.82**
FNN	**no buy**	0.78	0.84	0.81		
	buy	0.6	0.5	0.55		
	total	0.72	0.73	0.72	0.73	0.73
SVM	**no buy**	0.7	**0.96**	0.81		
	buy	0.51	0.1	0.17		
	total	0.64	0.68	0.6	0.68	0.67
LR	**no buy**	**0.86**	0.71	0.78		
	buy	0.56	**0.76**	**0.64**		
	total	**0.76**	0.73	0.74	0.73	0.8
RNN	**no buy**	0.78	0.83	0.81		
	buy	0.56	0.49	0.52		
	total	0.72	0.72	0.72	0.72	0.74

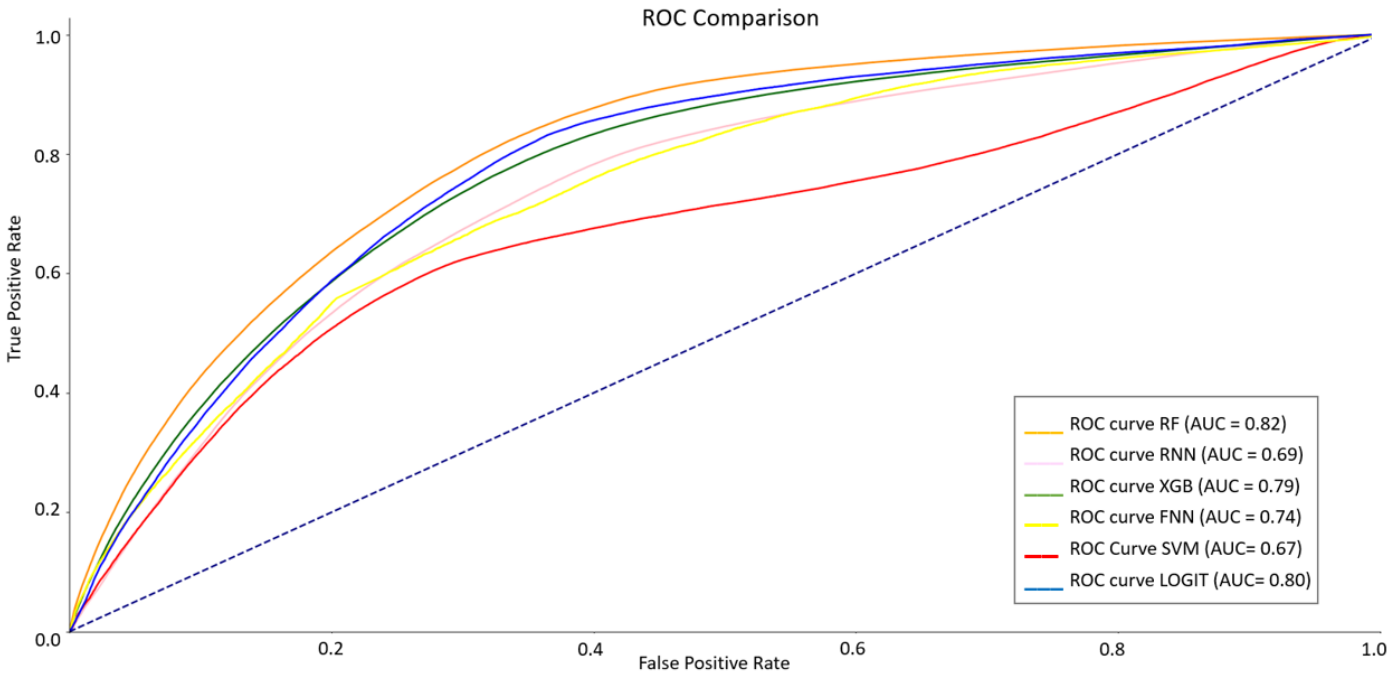


Figure 6.1: The ROC AUC curve for the complete dataset tested with all algorithms.

RF with 0.83, 0.76 and 0.82 respectively. It was also the only algorithm that could outperform the baseline model, the boosted tree, regarding all metrics. The worst performance is obtained by the SVM, it achieves the lowest accuracy and ROC AUC with values of 0.68 and 0.67. It further achieves the lowest total scores for precision, recall, and f1-score since it is very bad in predicting the buying class, even though it was trained with a balanced sub-dataset. The results obtained by the SVM are worse than the ones obtained during the SVM grid search. This might be caused by the not balanced test dataset. For the grid search, a 10-fold cross validation was used, which means that since the training dataset had to be balanced also the test dataset was balanced. For obtaining the final results, the SVM was also trained on a balanced dataset but tested with a dataset showing the same distribution of buying and no buying classes as the actual dataset. This might have led to the discrepancies in the performance results. Figure 6.1 shows the ROC AUC for each algorithm in more detail, by displaying the entire ROC curve. One can see that the ROC curve for the RF, displayed by the orange line results in a higher recall (true positive rate) and specificity (false positive rate) across all thresholds compared to all other algorithms. For all other algorithms, it is dependent on the threshold which algorithm outperforms the others regarding recall and specificity, which results from the curves crossing each other.

In Table 6.2 the results for the sequence dataset are displayed. The results show the same pattern as for the complete dataset. The LR has the best precision

Table 6.2: Results of all algorithms for the sequence dataset with threshold 0.5.

Sequence Data	Threshold: 0.5					
Algorithm	Class	Precision	Recall	F1-score	Accuracy	Roc Auc
Boosted tree	**no buy**	0.79	0.85	0.82		
	buy	0.59	0.49	0.53		
	total	0.73	0.74	0.73	0.74	0.78
RF	**no buy**	0.8	0.87	**0.83**		
	buy	**0.62**	0.51	0.56		
	total	0.75	**0.76**	**0.75**	**0.76**	**0.81**
FNN	**no buy**	0.77	0.85	0.81		
	buy	0.55	0.42	0.48		
	total	0.7	0.72	0.71	0.71	0.67
SVM	**no buy**	0.75	**0.92**	0.82		
	buy	0.61	0.29	0.39		
	total	0.7	0.73	0.69	0.73	0.74
LR	**no buy**	**0.86**	0.7	0.77		
	buy	0.53	**0.75**	**0.62**		
	total	**0.76**	0.72	0.73	0.72	0.78
RNN	**no buy**	0.77	0.85	0.81		
	buy	0.58	0.46	0.51		
	Total	0.71	0.72	0.71	0.72	0.74

results for the no buying class with 0.86. The SVM results in the best recall, with a value of 0.92 and the RF scores the best F1-score, the best accuracy, and the best ROC AUC, with values of 0.83, 0.76 and 0.81 respectively. The RF is again better than the baseline model regarding all measures. The worst performing algorithm is the FNN with the lowest accuracy and ROC AUC of 0.71 and 0.67. It also displayed very low scores for precision, recall, and F1-score. In general, the results for the sequence dataset are slightly lower than the ones of the complete dataset. The ROC AUC and recall of the RF algorithm have for example dropped by 1%, in contrast, the accuracy and F1-score remained the same.

Table 6.3 displays the results achieved on the customer dataset. Immediately one can see that the obtained results are much worse than the results achieved on the complete and the sequence dataset. Here, the best ROC AUC results lie at 0.67, compared to values of 0.81 and 0.82 for the sequence and complete dataset. Also, the results for the buying class are very low, especially regarding recall and F1-score. The only algorithm that showed decent results on the buying class was the LR. The best precision results for the non buying class were obtained by the LR, which resulted in a value of 0.85. The highest recall was at 1 achieved by RF, FNN, SVM and RNN, due to neglecting the buying class, which resulted in a recall of 0

Table 6.3: Results of all algorithms for the customer dataset with threshold 0.5.

Customer Data	Threshold: 0.5					
Algorithm	Class	Precision	Recall	F1-score	Accuracy	Roc Auc
Boosted tree	**no buy**	0.79	0.97	0.87		
	buy	0.39	0.07	**0.12**		
	total	0.7	0.78	**0.71**	0.78	0.64
RF	**no buy**	0.78	**1**	**0.88**		
	buy	**0.53**	0.01	0.01		
	total	**0.73**	0.78	0.69	0.78	**0.67**
FNN	**no buy**	0.79	**1**	**0.88**		
	buy	0.51	0.01	0.02		
	total	**0.73**	**0.79**	0.69	**0.79**	**0.67**
SVM	**no buy**	0.78	**1**	**0.88**		
	buy	0	0	0		
	total	0.61	0.78	0.69	0.78	0.39
LR	**no buy**	**0.85**	0.61	0.71		
	buy	0.3	**0.61**	0.4		
	total	**0.73**	0.61	0.64	0.61	0.65
RNN	**no buy**	0.78	**1**	**0.88**		
	buy	0.43	0	0		
	total	0.71	0.78	0.69	0.78	0.66

or 0.01. The same algorithms achieved the highest F1-score with a value of 0.88, with again very low scores for the buying class ranging between 0 and 0.2. The best accuracy score of 0.79 was achieved by the FNN, closely followed by the boosted tree, RF, SVM and RNN which all obtained a score of 0.78. The highest ROC AUC of 0.67 was achieved by RF and FNN. The lowest ROC AUC value of 0.39 showed the SVM. This value is worse than the guessing probability of 0.5, which indicates that the algorithm learned wrong patterns. In general, the FNN showed the best results, closely followed by the RF. The best results regarding both classes could be achieved by the LR since it was the only algorithm that did not neglect the buying class.

6.1.1 Results on RNN data

Next to training and testing all algorithms on the three different datasets, the RNN was additionally trained on data containing the same amount of information but fewer features, requiring less elaborate and time-consuming feature engineering. This effort focused solely on the sequence data, as described in Chapter 4. Therefore, the RNN was tested on a new complete dataset and a new sequence dataset, where sequence features were replaced by the new features and customer features remained the same. This effort was undertaken to analyze whether RNNs offer a way to decrease the amount of feature engineering, by learning more complex relationships in the data, especially regarding the sequential input, implicitly.

The obtained results are displayed in Table 6.4. The table shows that for both the complete as well as the sequence dataset the predictions on the no buying class achieved higher results for all measures than on the buying class. Further, the results obtained on the sequence dataset were better than on the complete dataset, with values of 0.8, 0.82, 0.81, 0.82 and 0.7 for precision, recall, F1-score, accuracy and ROC AUC on the sequence dataset and values of 0.75, 0.7, 0.76, 0.78 and 0.69 respectively on the complete dataset.

6.1.2 Robustness of results

To understand how the algorithms perform under different conditions, the RF results, being the best results, were analyzed in regard to certain variables. The chosen variables are categorical variables from the feature set, being of high importance to the business decision makers, namely, the day of the week, gender, and device. It was analyzed how the ROC AUC changes under different levels of these variables. The ROC AUC was chosen since it is the metric that is not influenced by the selected prediction threshold. For the analysis, only the complete dataset was used since it

Table 6.4: Results of the RNN on complete and sequence RNN dataset with a threshold of 0.5.

RNN with RNN Data	Threshold: 0.5					
Data Type	Class	Precision	Recall	F1-score	Accuracy	Roc Auc
All Data	**no buy**	0.83	0.91	0.87		
	buy	0.46	0.3	0.37		
	total	0.75	0.7	0.76	0.78	0.69
Sequence Data	**no buy**	0.87	0.93	0.9		
	buy	0.45	0.29	0.35		
	total	0.8	0.82	0.81	0.82	0.7

yielded the best prediction results and allows for an analysis of both the sequence as well as the customer features.

Table 6.5: The ROC AUC results in regard to different levels of the day of the week variable.

Day	ROC AUC
Mo	0.82
Tu	0.81
We	0.82
Th	0.83
Fr	0.82
Sa	0.81
Su	0.81

The results of the analysis of the different days of the week can be seen in Table 6.5. The results imply, that the algorithm performs similarly well for all days of the week. The best ROC AUC result could be obtained for Wednesdays with a value of 0.83. The lowest value of 0.81 was achieved on Tuesdays, Saturdays, and Sundays. All other days resulted in a ROC AUC of 0.82.

Table 6.6 shows the results of the analysis of the feature device. The differences here are as minimal as with the day of the week variable. The best ROC AUC results are obtained with the desktop device with a value of 0.82. The tablet and the mobile devices resulted in slightly worse results with a ROC AUC of 0.80.

Lastly, in Table 6.7 the effects of the gender on the prediction outcome can be observed. The best results could be obtained in the cases where the gender was unknown, which is surprising since it was expected that additional information would yield better prediction results. In this case, the ROC AUC was at 0.82, whereas for

Table 6.6: Percentage of correct results in regard to the device.

Device	ROC AUC
desktop	0.82
mobile	0.80
tablet	0.80

Table 6.7: Percentage of correct results in regard to the gender.

Gender	ROC AUC
unknown	0.82
women	0.79
men	0.79

women and men it could only reach a value of 0.79.

6.2 Latency results

The latency analysis was conducted on the complete dataset. One has to keep in mind that all algorithms that required one hot encoding, such as the FNN, the LR, the SVM and the RNN, need to process more features per sample than the RF and the boosted tree, namely 97 versus 42. Also, since the analysis was carried out locally on a desktop computer, results can only be understood as indications for performance of the algorithms in relation to each other. An actual implementation of the algorithms in the real-time environment will lead to much faster prediction times.

To analyze the latency, prediction times for each algorithm on 100 feature vectors were determined. The analysis was conducted in bulk mode, which means passing multiple data points to the algorithms at once, in this case, all 100. The opposite to this would be an atomic mode, where every data point is processed successively. Bulk mode was chosen because it offers a more realistic scenario for the real-time implementation since it is faster than the atomic mode due to optimization, branching, etc.. Predictions were repeated five times, the resulting boxplots are displayed in Figure 6.2. They show that the SVM displayed the longest prediction times with a mean of 0.85 seconds for predicting the 100 feature vectors, followed by the RNN with a mean of 0.4 seconds. All other algorithms showed very fast prediction times, with the best produced by the boosted tree and the LR on average only needing 0.01 and 0.009 seconds respectively for predicting all 100 instances. The RF and the FNN required a mean of 0.05 and 0.06 seconds for the predictions.

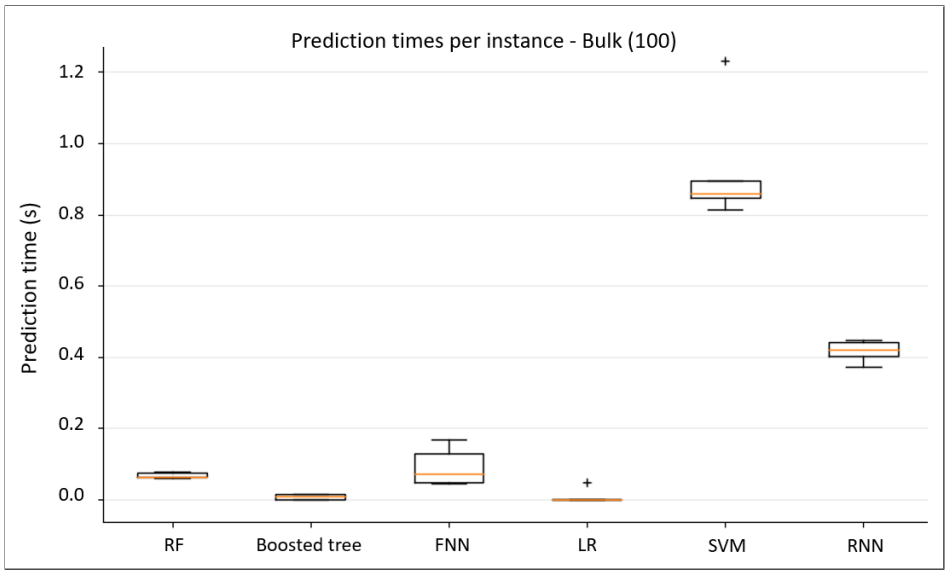


Figure 6.2: Computed latency for each algorithm performing predictions on 100 data points in bulk mode.

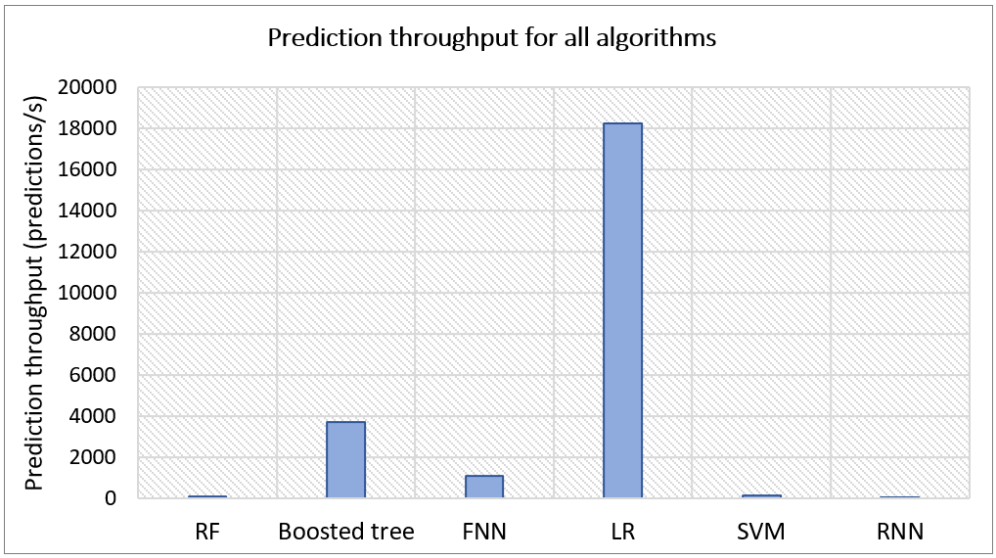


Figure 6.3: Prediction throughput (predictions per second) for each of the tested algorithms.

Further, the prediction throughput was analyzed. Prediction throughput describes how many predictions can be completed within one second by each algorithm. The results are displayed in Figure 6.3. The LR clearly has the highest throughput with 18220 feature vectors per second. It is followed by the boosted tree algorithm that can throughput 3700 samples in a second. The FNN was able to throughput 1090 samples per second. All other algorithms showed a very low throughput with 60, 100 and 10 for the RF, the SVM, and the RNN respectively.

6.3 Comprehensibility

As explained in the literature review in Chapter 2 some algorithms are more comprehensible than others. Whereas simple DTs can be transformed into easy to comprehend if-else statements, it is impossible to break down the reasoning of more complex algorithms such as SVMs into these easy rules. Even the DTs used in this thesis, being a boosted DT and the RF algorithm, cannot be translated into such rules anymore since they belong to the group of ensemble trees. Nevertheless, the DTs have the advantage that they can assess the variable importance, such as shown in Section 4.4.2. Lang and Rettenmeier (2017) could show that also RNN results can be made more comprehensible through visualization, achieved by plotting how the purchase probability changes when the consumer takes certain actions. This same method was applied here on the RNN results as well as on the RF results, to replicate the findings from Lang and Rettenmeier and to see if visualizing non stateful algorithms works just as well. The results can be seen in Figure 6.4 and Figure 6.5. Figure 6.4 shows the buying behavior for a specific buying session as predicted by the RF. On the x-axis the number of viewed pages is listed. Adding something to the shopping basket does not result in opening a new page, therefore the page number remains constant in these cases. One can see that returning from an article detail page back to the shop overview, without adding something to the basket results in a decrease of the purchase probability, as well as returning to the homepage after visiting the order site. In contrast, adding something to the shopping basket as well as going to the order website increases the purchase probability. Especially, the first basket addition increases the purchase probability a lot, whereas, basket additions do only slightly increase the purchase probability if the latter is already high. These findings make sense and through the visualization, the reasoning of the algorithm can be better understood. Similar results can be seen in Figure 6.5, where the buying behavior of a different buying session as predicted by the RNN can be seen. The results also show that returning to a shop overview page, as well as going back to the homepage, decreases the purchase probability, as can be seen for viewed pages 8, 10 and 15. As in the results displayed by the RF, adding an item to the shopping basket as well as visiting the order page increases the purchase probability. Again, the first basket addition has the highest impact on the purchase probability. This method of visualization can also be applied to the other algorithms. Therefore, this method as shown by Lang and Rettenmeier helps to increase comprehensibility of all algorithms.

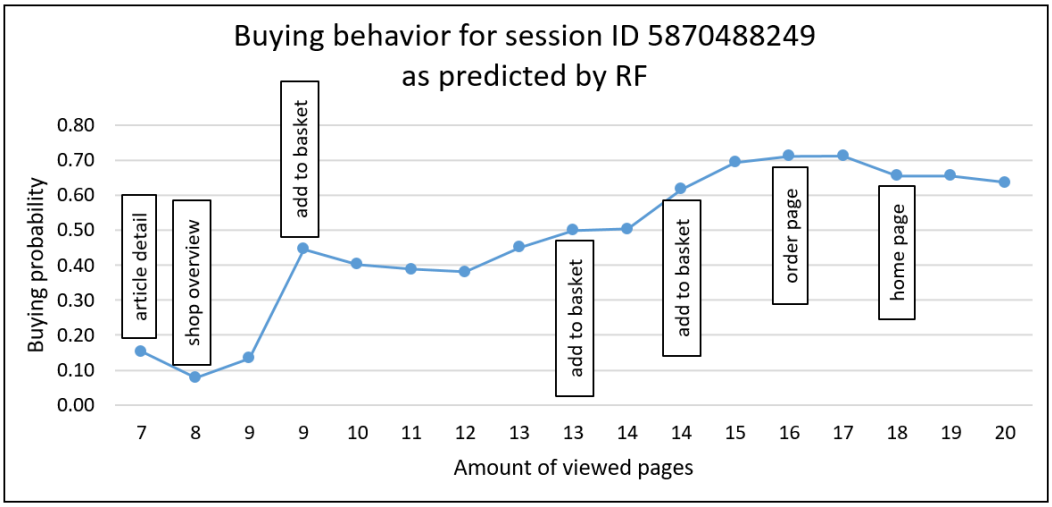


Figure 6.4: Buying probability for session ID 5870488249, a buying session, as predicted by the RF in relation to certain actions.

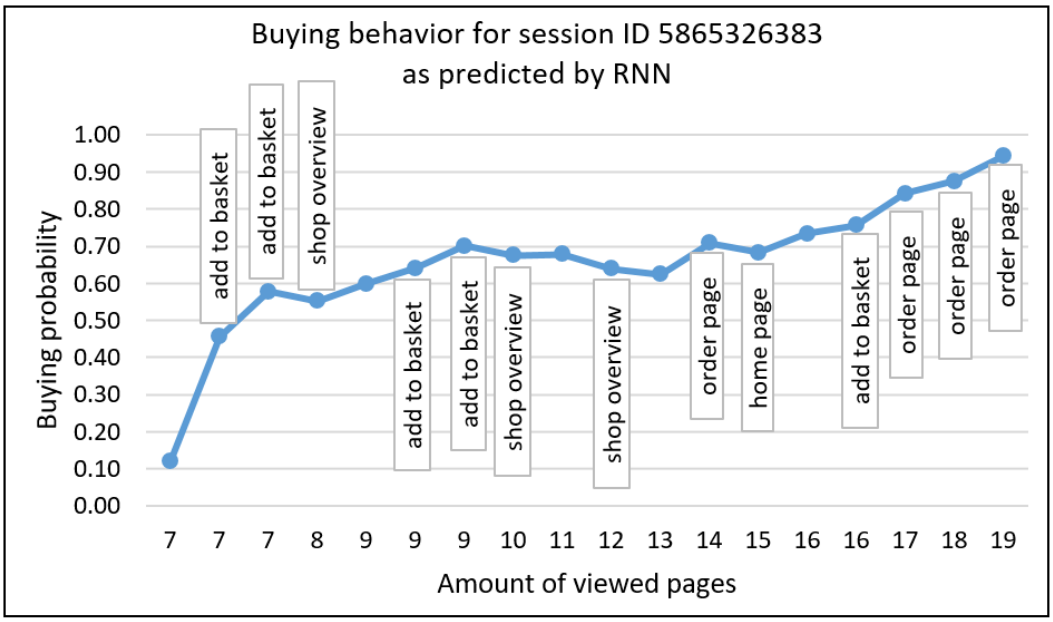


Figure 6.5: Buying probability for session ID 5865326383, a buying session, as predicted by the RNN in relation to certain actions.

Chapter 7

Discussion

This chapter discusses the results, demonstrates the limitations of this research and gives advice for future work.

7.1 Principal findings

This section discusses the principal findings of the results as displayed in Chapter 6 split by the performance results, the latency results, and the comprehensibility analysis. Finally, a comparison with the baseline model is drawn.

Performance results

In general, the performance results, especially on the sequence and complete dataset, were very satisfactory for most algorithms yielding good results that are of value for the management and decision makers, building a basis on which business decisions can be made. Looking more closely at the performance of the different algorithms on the three datasets, one can see that some algorithms performed better on all datasets than others. The RF performed best of all algorithms on the sequence and the complete dataset, achieving the second highest score on the precision metric and the highest score on the ROC AUC, which were identified as the most important metrics in Chapter 3. It can, therefore, be said to be the best algorithm regarding the performance, obtaining the highest scores out of all algorithms regarding all datasets. These results are in line with the findings from Hop (2013) and Niu et al. (2017) where the RF also outperformed other algorithms, as reviewed in Chapter 2. The second best performing algorithm was the LR displaying the highest precision and second highest ROC AUC on the complete and sequence dataset. Further, it was the only algorithm not neglecting the buying class when tested on the customer

dataset. This is in contrast to all reviewed papers, where the LR performed worst compared to all other tested algorithms (Hop, 2013; Niu et al., 2017; Poggi et al., 2007). The worst performance shows the SVM algorithm, completely neglecting the buying class with the customer dataset, and also showing the worst results on the complete dataset, due to also neglecting the buying class. It is hard to explain these results since especially for the SVM a balanced dataset was used for training the algorithm. A possible explanation is that the choice of the kernel function was wrong. Choosing the correct kernel is the most difficult challenge about SVMs, according to Burges (1998). For the RNN and FNN, one can expect that performance results would have increased with using more training iterations, which was not possible due to computational constraints. Especially, regarding the RNN, 16 training epochs is a very little amount. Concluding, for the performance results one could see that the well-performing algorithms did not show large differences in their achieved results. This indicates that if the data is well prepared and features are chosen and selected thoroughly, most algorithms show good results. Decision makers need to consider that next to choosing the right algorithm data preparation is one of the most important steps in a data mining project.

Looking at the results based on the different datasets, the complete dataset led to the best results, closely followed by the sequence dataset. This applies for each algorithm. By far the worst results with every algorithm were obtained on the customer dataset. These results were expected since they correspond to the findings from the reviewed literature. The high increase of performance when using dynamic session data compared to static data was also observed by Lee et al. (2015). That using both static and dynamic session data enhances the results as opposed to only using dynamic session data was also found to be true in the studies of Bogina et al. (2016) and Poggi et al. (2007), where the differences observed by Poggi et al. (2007) were similarly small as in this study. Nevertheless, none of these studies aimed at further explaining these findings. A possible explanation can be found in this thesis in Chapter 4, where different factors from the sequence and the customer dataset were analyzed. One could see that most analyzed customer features did not yield high predictive power, such as the gender or the age since across all categories the percentage of buying sessions was almost constant. This can explain why predictions solely based on the customer dataset showed such bad results. Nevertheless, the pure existence of the customer features, indicating that a visitor could be identified, yielded very high predictive power with 14% of buying sessions if the visitor was unknown compared to 32% if the visitor could be identified. This can explain the improved prediction results if both session and static customer data were combined. The increase might therefore not be due to the customer data it-

self, but can be attributed to the fact that customer information was available, giving information about the identification of the visitor. Another reason why the customer data by itself displayed bad results, can be the small number of training samples. Since the customer dataset only consisted of the sessions where a customer identification occurred, it yielded less data than the complete and sequence dataset. This decreased amount might not have been enough to train the different algorithms. In general, for the analysis of the sequence compared to the customer data one has to keep in mind that the sequence dataset did not only contain sequential session data, but also static session data. So, the results mostly allow for a differentiation between the performance of session and customer data, but since most of the session data was sequential and customer data only of static nature one can assume to be distinguishing between sequential and static datasets. Concluding, for future deployments the complete dataset should be used since even though the difference to the sequence dataset was small, it led to the best results. This is in line with the current deployment where the boosted tree is also running on the complete dataset.

The results achieved by the RNN tested on the sequential and complete RNN dataset are surprising. Unlike for the other datasets, the results obtained on the sequential dataset clearly outperform the results as achieved by predictions carried out on the complete dataset. This cannot be explained by findings in literature, since none of the reviewed papers, tested predictions with RNNs on sequential and static data separately. Lang and Rettenmeier (2017) for example only utilized a combination of dynamic and static session data and static customer data for training and testing the RNN. The results obtained in this study might suggest that Lang and Rettenmeier could have achieved a better performance by only training on the sequential session data since the results clearly indicate that RNNs perform better on only sequential data. This might be due to their capability of predicting entire sequences. Another interesting observation can be drawn from the results obtained by the RNN. When being trained on the feature engineered dataset, the best results were obtained with the complete dataset with values of 0.72 for precision, recall, f1-score and accuracy and a value of 0.74 for the ROC AUC score. When trained on the less feature engineered RNN dataset the best results were achieved on the sequential data with values of 0.8, 0.82, 0.82, 0.82 and 0.7 for precision, recall, F1-score, accuracy and ROC AUC score. These results show that the predictions on the RNN data outperformed the ones on the feature engineered dataset clearly in all measures except for the ROC AUC score. These findings are astonishing, especially when looking at the number of used features. Where 43 features are used for the complete feature engineered dataset and only nine for the sequential RNN dataset. This shows that the amount of used features is not a good predictor

of how well a model will perform, but rather the nature of the features themselves seems to be important. Implying that an RNN trained on a small amount of sequential features with a small degree of feature engineering provides better results than being trained on a larger amount of static and sequential heavily engineered features. These findings are of high importance since no other reviewed study made an effort of comparing types and different degrees of feature engineering of features for sequential machine learning models, even though this can have a strong impact on predictions conducted in the context of e-commerce. The current belief in e-commerce seems to be that more features mean better predictions, as can be seen from the study of Lang and Rettenmeier, conducted for the large German clothing retailer Zalando. Here, smaller companies with less computational resources seem to have a disadvantage. With the results of this study now, this does not seem to be true anymore, since fewer features, meaning less computational effort, resulted in better findings. Making it possible to carry out precise predictions with a limited amount of computational resources. Also, with RNNs the usage of static customer features, that can lead to privacy concerns and require strict storing regulations, is not necessary anymore, especially, since these have even decreased the prediction performance. This is a further advantage for e-commerce companies carrying out predictions.

Latency

Both regarding the prediction times per instance in bulk mode as well as the throughput per second the LR outperformed the other algorithms by showing the fastest prediction times, followed by the boosted tree algorithm. This is in line with the reviewed literature in Chapter 2 where LR was said to be having fast prediction and training times. Also, the boosted tree was reported to have faster prediction times than the RF (Breiman, 2001). The bad latency results of the RNN especially to be seen in the long prediction times per instance in the bulk mode, also correspond to the reviewed literature. There it was stated that RNNs display extended prediction times compared to other machine learning algorithms (Lang & Rettenmeier, 2017). The even worse results obtained by the SVM are on the other hand surprisingly, since fast prediction times were listed as an advantage of SVMs in the literature (Han et al., 2011). Concluding, most of the obtained latency results were expected since they correspond to the findings in the literature. Decision makers should keep these results in mind when deploying the models in the real-time environment since a high latency in this context can yield unwanted results. Nevertheless, to truly establish which latency results are acceptable and which ones are too high, one has

to deploy the algorithms in the real-time environment to see how a more powerful machine decreases the latency.

Comprehensibility

The analysis of the comprehensibility was based on a visualization technique as applied by Lang and Rettenmeier (2017). This method shows that the buying probability for each step of the shopping process can be plotted, making it easy to understand which visitor actions yield an increase or decrease in the buying probability. Visualizing a session as a whole makes the effect of actions on the prediction outcome more understandable, hence increases the comprehensibility of a model. It was applied to both a stateful and stateless machine learning algorithm, the RNN and RF, to see if also models that do not directly predict sequences can be visualized in a sequential fashion. The results show that for both models the prediction outcomes can be well visualized in a way that intuitively makes sense. Though this technique showed that comprehensibility of all models can be equally increased, it can only be applied to sequential data, since it plots the changes in the buying probability over time. There are other techniques that help to understand black box models trained on any kind of data. One such example is the LIME technique. It can explain predictions of any classifier by approaching single predictions locally through visualizing which features have a positive or a negative influence on the prediction outcome and how large this influence is (Ribeiro et al., 2016). This visualization helps to understand predictions better by displaying the factors that led to it. These factors can be of sequential or static nature, allowing for explanations of predictions for any kind of data. Through these two techniques, one can see that all machine learning algorithms can be made comprehensible through visualization. Resulting, all algorithms perform equally well in this category and can help decision makers to understand and explain prediction outcomes.

Comparison with baseline model

After the results of all algorithms in regard to all different measures have been discussed a comparison to the already implemented baseline model is drawn to establish how the algorithms perform in relation to the implemented standard. Looking at the performance, especially the RF could improve on the results of the boosted tree. For the important measures such as precision and ROC AUC results were increased by up to 3% on all datasets. All other algorithms were not able to outperform the

Boosted tree regarding all datasets and most performance metrics. Looking, at the latency the only algorithm being able to show even faster prediction times than the baseline model was the LR, all other models displayed a smaller prediction throughput as well as higher prediction times when processing bulks of data. Concerning the comprehensibility this thesis showed that all algorithms can be interpreted equally well, therefore no difference to the baseline model resulted. Concluding, the RF is the algorithm that could replace the currently existing model to increase performance. Whether the slower prediction times are sufficient for real-time predictions has to be assessed after implementation and deployment in the real-time environment. The RF shows a further advantage compared to the other algorithms. It is the only algorithm that requires exactly the same data pre-processing steps as the boosted tree, therefore, processes focused on data preparation in the real-time environment would not have to be adapted to the new algorithm, which would make the actual deployment procedure fairly easy.

7.2 Limitations

This section discusses the limitations of this research, which are mostly related to the fact that the study was carried out locally on a desktop computer and could not be run on a server due to missing infrastructure and data protection constraints.

Due to the memory and computational constraints of a local computer, factors such as the amount of training data, creating the padded sequence data for the RNN input, training iterations, and hyper-parameter tuning were influenced. Regarding the training data especially for the FNN and the RNN, which are algorithms that require large amounts of data to properly find patterns in the data and carry out good predictions, the used training data was not enough. Using more data would have most likely led to better prediction results, but had resulted in too long training times. Regarding the sequence creation for the RNN exceptional constraints were met. Due to the fact, that the padded sequences get very large they could not be stored in the memory of the computer anymore, therefore they had to be saved on disk after processing every 5,000 sequences. The limited memory did further not allow for using sequences longer than 20 time steps since the computer could not process longer sequences through the RNN anymore. Therefore, it was not possible to explore if considering longer sequences had led to better predictions. Regarding all algorithms the training iterations had to be kept low, where more training iterations had resulted in better predictions but also in longer training times. This again influenced the FNN and RNN the most, since these algorithms require a lot of training epochs.

Further constraints concern the utilized data, resulting from the fact that the

data for this study had to be the same as for the already implemented baseline model to allow for comparability. Due to this, additional feature engineering could not be explored. It is imaginable that additional feature engineering had improved the results. One could have made use of additional data such as click positions, and hover duration as in the study of Niu et al. (2017). Further, for the customer dataset, it probably had been better to use an equal amount of data samples as for the complete and sequence dataset, even if this means that additional weeks of data have to be used that were not used in the complete and sequence dataset, since the training set size of the customer data was probably too small for all algorithms.

Concerning the analysis of the sequence compared to the static data, the analysis would have been clearer if the sequence dataset had only consisted of sequential, time-dependent data. In this setting, the sequence dataset described the session data consisting of both static and sequential features. This does allow for a distinction between session and customer data, but it does not clearly separate sequential from static data. So, to truly analyze the influence of sequential versus static data, the data should be more clearly separated. Further, hyper-parameters were optimized to fit the complete dataset, there is the possibility that results on the sequence dataset would have been just as good or even better than the complete dataset if hyper-parameter tuning had been done for this dataset.

Lastly, the way how the current model is implemented, trained and deployed, asks for session-based predictions. Therefore, this study was also restricted to conducting session-based predictions. Unfortunately, this approach has some shortcomings. With session based predictions for example purchases that might have taken place on a later point in time, can be prefetched by displaying a gift card. So, if someone is currently only browsing because he plans one purchasing on the weekend, the algorithm correctly detects a no buying session. Resulting a gift card is displayed, which is only valid for this current session. Most likely the visitor will make a purchase now instead of, as planned, on the weekend. Even though it looks like the correct prediction was a success and turnover for that particular day was increased, the profit was taken away from the weekend when the visitor might have even purchased without the need of a gift card. This so-called 'front-loading' effect is not desirable, and can be avoided if predictions are conducted across sessions, because then visitors can be identified that do not plan on purchasing in the current session or in the near future.

7.3 Future work

From the above-mentioned limitations, some recommendations result for future research. First, any follow-up study should be concerned with running the different

algorithms on a more computationally powerful machine instead of locally on a regular desktop computer. All algorithms offer the possibility of parallelization, which can also be utilized on a better machine. This can already help to create more powerful models, through using more data, conducting a more thorough hyper-parameter tuning and running more training iterations to fully exploit and analyze the algorithms' potential.

Regarding the data, it would be interesting if using other types of features especially ones that regard aspects such as click information, results in better predictions. Therefore, a follow up study could focus on using more information of the data that is available anyways to see if performance can also be enhanced by improving the data basis instead of tuning and exploring the potentials of different algorithms. Further, to really establish the difference between sequential and static customer data future studies should differentiate between the sequential and the static session data.

Lastly, it would be very interesting to exploit the possibility of reducing feature engineering and the usage of static customer data with stateful models such as RNNs further. Especially, since the already obtained results, that implicate the RNNs perform well on less engineered features of sequential nature, were already very promising. It is imaginable to solely conduct a study based on comparing the results of for example the RF algorithm with an RNN which are trained on data containing the same information but a different number of features, just like it was done in this study. But here one should train the RNN on a server, so more data and more epochs can be used. It is imaginable that saving time and human brain power during feature engineering will, on the other hand, result in longer training times with more training data to extract important features implicitly and achieve similar results as the RF trained on the complete dataset. This, therefore, has to be facilitated through a more thorough training on more powerful and possibly parallel machines.

Chapter 8

Conclusion

This chapter concludes the report by offering answers to the sub-questions and the main research question.

Sub Question 1: *How can an exploratory data analysis provide insight into the prediction problem?*

An exploratory data analysis as conducted in Chapter 4 in Section 4.3, can help to generate a first insight into the prediction problem, by visualizing important variables such as gender of the customer or day of the week in relation to the target variable, in this case, whether a purchase occurred or not. This analysis could already show that customer-specific attributes might not have a large influence on the prediction outcome, since the purchase probabilities across the different levels of the customer variables did not vary largely. Additionally, visualizing data in this way can help the management to create an understanding of which website visitors actually purchase and can already prompt decisions on targeting a specific customer segment with personalized advertisements or special offers.

Sub Question 2: *Which machine learning model is best suited to solve the prediction problem?*

Based on the results displayed in Chapter 6 the RF model showed the best performance, by displaying good results for all evaluation metrics and offering a reasonable latency. Regarding the comprehensibility, it did not display any differences to the other algorithms. The RF algorithm offers a good possibility for the German clothing retailer to increase the performance of the currently implemented model while causing minimal deployment effort, since it requires the same data and data pre-processing steps as the current model. In terms of feature engineering efforts, another well-suited algorithm for this sequential prediction task is the RNN. While

not having displayed the best results with the heavily engineered features, it showed a very good performance on the sequential RNN features. It offers the advantages of requiring less feature engineering and fewer features in general, displaying no need for customer-specific features, which are often concerned with privacy issues.

Sub Question 3: *How do different data types, such as dynamic clickstream and static customer data, influence the models' performance?*

Different input data influence the performance of the models differently. Regarding the heavily engineered datasets, functioning as input for the stateless machine learning models, the customer data showed by far the worst results. It could also only slightly increase the performance when combined with the sequence data. This showed clearly, that sequence data is most important for predicting purchase decisions in an online store. This implies that future feature engineering efforts should be focused on the sequence data. It is further imaginable to trim the input feature set to solely the sequence data, to reduce computational complexity and to decrease privacy concerns. The importance of the sequential data became even clearer when training the RNN on the less engineered RNN features. Where the RNN performed better on only the sequential data than on the combined dataset. It this case the customer data even reduced the prediction performance, indicating that especially stateful machine learning models should only be trained with sequential data.

Sub Question 4: *Can stateful models produce good results while requiring less feature engineering?*

By training the RNN on the regular, engineered and a less engineered feature set, one could display that stateful models produce good results with less feature engineering. The results even showed that the less engineered feature set produced better results than the other feature set, while consisting of fewer features.

Sub Question 5: *How robust is the best-suited algorithm to different conditions?*

The performance of the best-suited algorithm, being the RF as explained in the discussion of the results, was analyzed regarding the days of the week, the utilized device, and the gender of the customer. One could see that for none of the tested variables the performance differences were drastically, and the algorithm can be said to be performing well under various conditions. The only surprising effect that was observed, was that not knowing the gender yielded the best results regarding the gender variable, which again shows that adding the customer data can even de-

crease the prediction performance. Further, it is advantageous for decision makers to know how different settings can influence the correctness of predictions.

Research Question: *How can a visitor in a web shop be categorized as a buying or no buying?*

When commencing such a binary classification task, it can be helpful to plot categorical variables in relation to the target variable, for generating a first insight into the predictive power of the tested variables. For the actual task of classifying a web shop visitor as buying or no buying the RF seems to be best suited concerning the class of the stateless machine learning models. It requires a minimal amount of data pre-processing, showed good results on all performance measures, displayed a decent latency and degree of comprehensibility and was robust to various conditions such as utilized device or gender of the customer. The best performance, as for all other stateless models, was achieved on a combined dataset containing static and dynamic session data as well as static customer data, where the static customer data only led to a slight increase in performance. Hence, when concerned with computational capacity and privacy issues, training on a smaller feature set, containing only session specific data, will still yield powerful results. The only shortcoming of this model, as with all other stateless models, is that it requires a fair amount of feature engineering to model the sequential nature of the dynamic session data explicitly. To avoid this time consuming and expertise requiring task, stateful models, such as RNNs, can be used due to their capacity of modeling sequences. In this study, the tested RNN showed promising results, when being tested on fewer, less engineered features as compared to the regular dataset. The shortcomings with RNNs are that they require a lot of data pre-processing, such as one-hot-encoding, normalization, missing value imputation and the conversion of single data points to sequences. They further require a lot of time and computational power to be trained and fine-tuned and show longer prediction times. Nevertheless, they hold great potential for e-commerce, since a lot of important tasks in e-commerce are of sequential nature.

References

Assumptions of logistic regression. (2017). Retrieved from http://www.statisticssolutions.com/assumptions-of-logistic-regression/

Barbour, A., & Petrelis, N. (2008). *Lecture 3: Markov chains.* Retrieved from http://www.math.uzh.ch/?file&key1=9151

Bellman, S., Lohse, G. L., & Johnson, E. J. (1999). Predictors of online buying behavior. *Communications of the ACM*, *42*(12).

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, *5*(2), 157 – 166.

Bogina, V., Kuflik, T., & Mokryn, O. (2016, 3). Learning item temporal dynamics for predicting buying sessions. *ACM*.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*(2).

Breiman, L. (2001). Random forests. *Machine learning*, *45*(1).

Brownlee, J. (2016, 7). *Crash course in recurrent neural networks for deep learning.* Retrieved from https://machinelearningmastery.com/crashcourse-recurrent-neural-networks-deep-learning/

Brownlee, J. (2017). *Why one-hot encode data in machine learning?* Retrieved from https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/

Burges, C. J. (1998). A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, *2*(2), 121–167.

Castanedo, F., Valverde, G., Zaratiegui, J., & Vazquez, A. (2014). Using deep learning to predict customer churn in a mobile telecommunication network. *wiseathena*.

Chapman, P. (1999). The crisp-dm user guide. *NCR Syst. Eng. Copenhagen*.

Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., & Wirth, R. (2000). Crisp-dm 1.0 step-by-step data mining guides. *NCR Syst. Eng. Copenhagen*.

Chen, Z., Fu, A. W.-C., & Tong, C.-H. (2004). Optimal algorithms for finding user access sessions from very large web logs. *World Wide Web*, *6*, 259–279.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273 – 297.

Craven, M. (2011). *Markov chain models.* Retrieved from https://cw.fel.cvut.cz/wiki/_media/courses/a6m33bin/markov-chains-2.pdf

Dewey, C. (2016). *Markov chain models.* Retrieved from https://www.biostat.wisc.edu/bmi576/lectures/markov-chains.pdf

Dmc 2013. (2013). Retrieved from `http://www.data-miningcup.de/en/review/goto/article/dmc-2013.html`

Dumais, S., Jeffries, R., Russell, D., Tang, D., & Teevan, J. (2014). *Understanding user behavior through log data and analysis*. New York: Springer.

Fajta, J. (2014). *Online visitor classification based on mbti model* (Unpublished master's thesis). Technische Universiteit Eindhoven.

Feng, J. (2017). *gcforest github repository.* Retrieved from `https://github.com/kingfengji/gcForest`

Frank, E., Hall, M., & Witten, I. (2016). *The weka workbench. online appendix for "data mining: Practical machine learning tools and techniques* (4th ed.). Morgan Kaufmann.

Friedman, H. (2002). Random forests. *Stochastic gradient boosting*, *38*(4).

Gelman, A., & Hill, J. (2016). *Data analysis using regression and multilevel/hierarchical models*. Cambridge University Press.

Geurts, P., Ernst, D., & Wehenkel, L. (2016). Extremely randomized trees. *Springer Science + Business Media, Inc.*.

Goodman, B., & Flaxman, S. (2016). European union regulations on algorithmic decision- making and a right to explanation. *In ICML Workshop on Human Interpretability in Machine Learning*.

Han, J., Pei, J., & Kamber, M. (2011). Data mining: concepts and techniques. *Elsevier*.

Hastie, T., Tibshirani, R., & Friedman, J. (2002). The elements of statistical learning: Data mining, inference, and prediction. *Biometrics*.

Heaton, J. (2016, 3). An empirical analysis of feature engineering for predictive modeling. *IEEE*, 1–6.

Hidasi, B., Quadrana, M., Karatzoglou, A., & Tikk, D. (2016). Parallel recurrent neural network architectures for feature-rich session-based recommendations. *In Proceedings of the 10th ACM Conference on Recommender Systems*, 241–248.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8).

Hofmann, M. (2006). Support vector machines-kernels and the kernel trick. *An elaboration for the hauptseminar reading club:support vector machines*, 1 – 16.

Hop, W. (2013). *Web-shop order prediction using machine learning* (Unpublished master's thesis). Erasmus University Rotterdam.

Hsu, C.-W., Chang, C.-C., & Lin, C.-J. (2003). *A practical guide to support vector classification* (Tech. Rep.). Taipei 106, Taiwan: Department of Computer Science National Taiwan University.

Jain, A. (2016, 3). *Complete guide to parameter tuning in xgboost (with codes in python).* Retrieved from https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning* (Vol. 112). New York: Spring.

Jolliffe, I. (1982). A note on the use of principal components in regressio. *Journal of the Royal Statistical Society*, *31*(3), 300 – 303.

Kaggle – the home of data science and machine learning. (2017). Retrieved from https://www.kaggle.com/

Korpusik, M., Sakaki, S., Chen, F., & Chen, Y. (2016). Recurrent neural networks for customer purchase prediction on twitter. *In Proceedings of the CBRecSys*.

Kotu, V., & Deshpande, B. (2014). Predictive analytics and data mining: concepts and practice with rapidminer. *Morgan Kaufmann*, 562–572.

Lang, T., & Rettenmeier, M. (2017). Understanding consumer behavior with recurrent neural networks. *In Proceedings of the 3rd International Workshop on Machine Learning Methods for Recommender Systems*.

Lee, M., Ha, T., Han, J., Rha, J., & Kwon, T. (2015). Online footsteps to purchase: Exploring consumer behaviors on online shopping sites. *In Proceedings of the ACM Web Science Conference*.

Liu, Y., An, A., & Huang, X. (2006, April). Boosting prediction accuracy on imbalanced datasets with svm ensembles. *PAKDD*, *6*, 107 – 118.

Lo, C., Frankowsik, D., & Leskovec, J. (2014). Understanding behaviors that lead to purchasing: A case study of pinterest. *In Proceedings of the KDD 16, San Francisco, CA, USA*.

Magrabi, A. (2016, 11). *Top 5 machine learning applications for e-commerce.* Retrieved from https://techblog.commercetools.com/top-5-machine-learning-applications-for-e-commerce-268eb1c89607

Manning, C., Raghavan, P., & Schuetze, H. (2008). Introduction to information retrieval. *Cambridge University Press*.

Moe, W. (2003). Buying, searching, or browsing: Differentiating between online shoppers using in-store navigational clickstream. *Journal of Consumer Psychology*, *13*(1–2), 29–39.

Mulder, W. D., Bethard, S., & Moens, M.-F. (2015). A survey on the application of recurrent neural networks to statistical language modeling. *Computer Speech & Language*, *30*(1), 61 – 98.

Nakayama, Y. (2009). The impact of e-commerce: It always benefits consumers, but may reduce social welfare. *Japan and the World Economy*, *21*(3), 239–247.

Neural network hyperparameters. (2015, 12). Retrieved from http://colinraffel.com/wiki/neural_network_hyperparameters

Niu, X., Li, C., & Yu, X. (2017). Predicitive analytics of e-commerce search behavior for conversion.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duchesnay, E. (2011). Scikit-learn: machine learning in python. *Journal of Machine Learning Research*.

Piatetsky-Shapiro, G. (2014). Kdnuggets methodology poll.

Poggi, N., Moreno, T., Berral, J., Gavald, R., & Torres, J. (2007, 7). Web customer modeling for automated session prioritization on high traffic sites. *International Conference on User Modeling*.

A practical introduction to deep learning with caffe and python. (2016, 6). Retrieved from http://adilmoujahid.com/posts/2016/06/introduction-deep-learning-python-caffe/

Python. (2017). *Python website.* Retrieved from https://www.python.org/

Randomforestclassifier. (2017). Retrieved from http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

Retail ecommerce in germany: A major digital market growing in size and sophistication. (2017, 7). Retrieved from https://www.emarketer.com/Report/Retail-Ecommerce-Germany-Major-Digital-Market-Growing-Size-Sophistication/2002102

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). *Why should i trust you?: Explaining the predictions of any classifier.*

Rnn. (2018). Retrieved from https://keras.io/layers/recurrent/

Rokach, L., & Maimon, O. (2014). Data mining with decision trees: theory and applications. *World scientific*.

R: The r project for statistical computing. (2017). Retrieved from https://www.r-project.org/

Russell, S., & Norvig, P. (1995). *Artificial intelligence - a modern approach*. Prentice-Hall, Englewood Cliffs: Artificial Intelligence.

Salehi, F., Abdollahbeigi, B., Langroudi, A. C., & Salehi, F. (2012). The impact of website information convenience on e-commerce success of companies. *Procedia Social and Behavioral Sciences*, *57*, 381–387.

Scalable and flexible gradient boosting. (2016). Retrieved from http://xgboost.readthedocs.io/en/latest/

The sequential model api. (n.d.). Retrieved from https://keras.io/models/sequential/

Sk-learn. (2017). *Sckit-learn website.* Retrieved from http://scikit-learn.org/stable/

Srivastava, T. (2015, 6). *Tuning the parameters of your random forest model.* Retrieved from https://www.analyticsvidhya.com/blog/2015/06/

tuning-random-forest-model/

Stassopoulou, A., & Dikaiakos, M. (2009). Web robot detection: a probabilistic reasoning approach. *Comput Netw*, *53*(3), 265 – 278.

Stevanovic, D., Vlajic, N., & An, A. (2011). Unsupervised clustering of web sessions to detect malicious and non-malicious website users. *Procedia Comput Sci*, *5*, 123–131.

Suchacka, G. (2016). Analysis of aggregated bot and human traffic on e-commerce sites. *Proceedings of IEEE FedCSIS14*, *2*, 1123 – 1130.

Suchacka, G., & Chodak, G. (2016). Using association rules to assess purchase probability. *Information Systems and e-Business Management*, 1–30.

Suchacka, G., Skolimowska-Kulig, M., & Potempa, A. (2015). A k-nearest neighbors method for classifying user sessions in e-commerce scenario. *Telecommunications and Information Technology*, *64*(3).

Suchacka, G., & Templewski, S. (2017). Applications of neural network to predict purchases in online store. *In Information Systems Architecture and Technology: Proceedings of 37th International Conference on Information Systems Architecture and Technology*.

Svc. (2017). Retrieved from http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

Tan, P., Steinbach, M., & Kumar, V. (2005). *Introduction to data mining*. Boston: Addison-Wesley Longman.

Tan, Y., Xu, X., & Liu, Y. (2016). Improved recurrent neural networks for session-based recommendations. *In Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, 17–22.

Wolpert, D., Macready, W., David, H., & William, G. (1995). *No free lunch theorems for search* (Tech. Rep.). 1399 Hyde Park Road, Santa Fe, NM, 87501: Santa Fe Institute.

Xgboost r tutorial. (2016). Retrieved from http://xgboost.readthedocs.io/en/latest/R-package/xgboostPresentation.html

Xie, Y., Li, X., Ngai, E., & Ying, W. (2008). Customer churn prediction using improved balanced random forests. *Expert Systems with Applications*.

Zell, A. (1994). *Simulation neuronaler netze*. Addison-Wesley.

Zhang, Y., Dai, H., Xu, C., Feng, J., Wang, T., Bian, J., ... Liu, T.-Y. (2014). Sequential click prediction for sponsored search with recurrent neural networks. *Association for the Advancement of Artificial Intelligence*.

Zhou, Z.-H., & Feng, J. (2017). Deep forest: Towards an alternative to deep neural networks. *IJCAI-2017*.

Zou, K. H., OMalley, A. J., & Mauri, L. (2007). Receiver-operating characteristic analysis for evaluating diagnostic tests and predictive models. *Circulation*, *115*,

654–657.

Appendix A

Algorithm Implementation

A.1 Boosted tree hyper-parameters

```
param_xgboost = {'max_depth': 12,
                 'min_child_weight': 100,
                 'base_score': base_score,
                 'learning_rate': 0.2,
                 'objective': 'multi:softprob',
                 'num_class': 2,
                 'eval_metric': 'auc',
                 'num_round': 400}
```

Figure A.1: The hyper-parameter settings of the boosted tree in Python code.

A.2 Random Forest hyper-parameters

```
rf_model = RandomForestClassifier(n_estimators=400,
                                  max_depth=200,
                                  random_state=1,
                                  verbose=2,
                                  min_samples_leaf=70,
                                  max_features=0.2)
```

Figure A.2: The hyper-parameter settings of the RF in Python code.

A.3 Support Vector Machine hyper-parameters

```
svm_model = SVC(kernel='rbf',
                C=0.1,
                gamma=1,
                verbose=True,
                probability=True)
```

Figure A.3: The hyper-parameter settings of the SVM in Python code.

A.4 Logistic Regression hyper-parameters

```
svm_model = SVC(kernel='rbf',
                C=0.1,
                gamma=1,
                verbose=True,
                probability=True)
```

Figure A.4: The hyper-parameter settings of the LR in Python code.

A.5 Feedforward Neural Network Architecture

```
model = Sequential()
model.add(Dense(700, activation='relu', input_shape=(X.shape[1],)))
model.add(Dropout(0.3))
model.add(Dense(350, activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(200, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(64))
model.add(Dense(1, activation='sigmoid'))
model.summary()
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
model.fit(X_train, y_train, epochs=50, batch_size=1012, verbose=1)
model.save(MODEL_DIRECTORY)
```

Figure A.5: The architecture of the FNN in Python code implemented with the Keras library.

A.6 Recurrent Neural Network Architecture

```
model = Sequential()
model.add(LSTM(400, input_shape=(20, 97)))
model.add(Dropout(0.4))
model.add(Dense(300, activation='relu'))
model.add(Dropout(0.4))
model.add(Dense(100, activation='relu'))
model.add(Dense(2, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Figure A.6: The architecture of the RNN in Python code implemented with the Keras library.

Appendix B

Results

B.1 Confusion Matrices

Table B.1: Confusion matrices for all algorithms on the complete, the sequence and the customer dataset.

Confusion Matrices Threshold 0.5

Algorithm		All Data		Sequence Data		Customer Data	
		Predicted Class		Predicted Class		Predicted Class	
	Actual Class	No buying	Buying	No buying	Buying	No buying	Buying
XGBoost	No buying	927458	152914	529498	93424	111695	3400
	buying	251618	254566	136917	132178	29655	2176
RF	No buying	924758	155614	539744	83178	114674	185
	buying	229440	276744	132803	136292	31855	212
FNN	No buying	905711	172506	166670	29913	115136	265
	buying	253155	255184	50355	36918	31246	279
SVM	No buying	3453391	158269	602990	53844	383745	0
	buying	1511591	165268	205553	83799	106008	0
Logit	No buying	769344	308873	137308	59275	70030	44999
	buying	122508	385831	21498	65775	12435	19462
RNN	No buying	85814	17851	261321	47327	114991	4
	buying	23566	22769	77035	64317	31928	3

Appendix C

Deep Forest

Towards the end of the processing time of this master thesis, another promising algorithm, namely Deep Forest, was discovered. Due to time constraints, a quick implementation and testing of the complete dataset were conducted. It showed that the algorithm did not perform better than the previously tested models and was therefore not included in the main body of this thesis. For completion, the algorithm, the results and the implications are discussed below.

Deep Forest, as proposed by Zhou and Feng (2017), is a machine learning model based on a decision tree ensemble approach. The model consists of a cascading forest structure. This structure, being similar to an FNN, processes features by propagating them through multiple levels, where outputs of one level are the inputs for the preceding level. The different levels consist of multiple independent RFs. It further makes use of multi-crained scanning, which means applying sliding windows to the features, to allow for sequence processing. Regarding the results, it has proven to be highly competitive with other deep learning models such as RNNs. It offers the advantages of a certain robustness to different hyper-parameters, performing well on small-scale training data, the possibility of running in parallel and being able to process sequences just like RNNs (Zhou & Feng, 2017). All of this makes it a good fit for the task of this thesis.

Data

The Deep Forest model was tested on the complete dataset first. Since this dataset yielded the best results for the other algorithms it was also expected to show good results with the Deep Forest. The goal was then to continue with the RNN dataset to establish whether the algorithm also performs well on sequences, without heavy feature engineering. Nevertheless, since the results on the complete dataset, as

Table C.1: Results of the Deep Forest implementation on the complete dataset.

Deep Forest Results - Complete Dataset					
Class	Precision	Recall	F1-score	Accuracy	Roc Auc
No buy	0.79	0.85	0.82		
Buy	0.62	0.53	0.57		
Total	0.74	0.75	0.74	0.74	0.68

displayed in the results section, were not good compared to the other algorithms, it was refrained from further analyzing this.

Implementation

For implementing the algorithm the official gcForest implementation from GitHub was utilized (Feng, 2017). For a first basic model, the structure and hyper-parameters found in the exemplary code on GitHub were used. This was already expected to yield good results since in their paper Zhou and Feng (2017) claim a high robustness to different hyper-parameters. After this implementation, a slightly more complex model was chosen, with an increased amount of RFs in each cascading layer. Unfortunately, due to memory constraints, this model was too large to be saved and loaded and could therefore not be assessed. Hence, the result section only contains to results of the first implementation.

Results

The results as displayed in Table C.1, show that the algorithm performed better on the no buying than the buying class. The precision, recall, and F1-score of the buying class are 0.79, 0.85 and 0.82 respectively. For the no buying class, the results are 0.62, 0.53 and 0.57. The achieved accuracy is 0.74 and the obtained ROC AUC score is at 0.68.

Discussion and Conclusion

Looking at the results one can see that precision, recall, F1-score, and accuracy are similar to the other tested algorithms. Regarding the ROC AUC score the obtained results are bad compared to the other algorithms, with only the SVM model showing a lower value. This, therefore, indicates that with other thresholds, the ob-

tained results would be worse than for the other algorithms. An increase of model complexity might increase the results, since the here implemented architecture was kept very simple. Unfortunately, this was not possible due to memory constraints of the utilized computer. If more resources or a parallel architecture are available the model could be tested with more complexity and the sequence approach, being also computationally expensive, could be implemented and analyzed. Even though the obtained results were not as good as expected, the model should not be neglected in future studies since it offers the possibility of sequence processing while requiring fewer training data than deep neural networks.