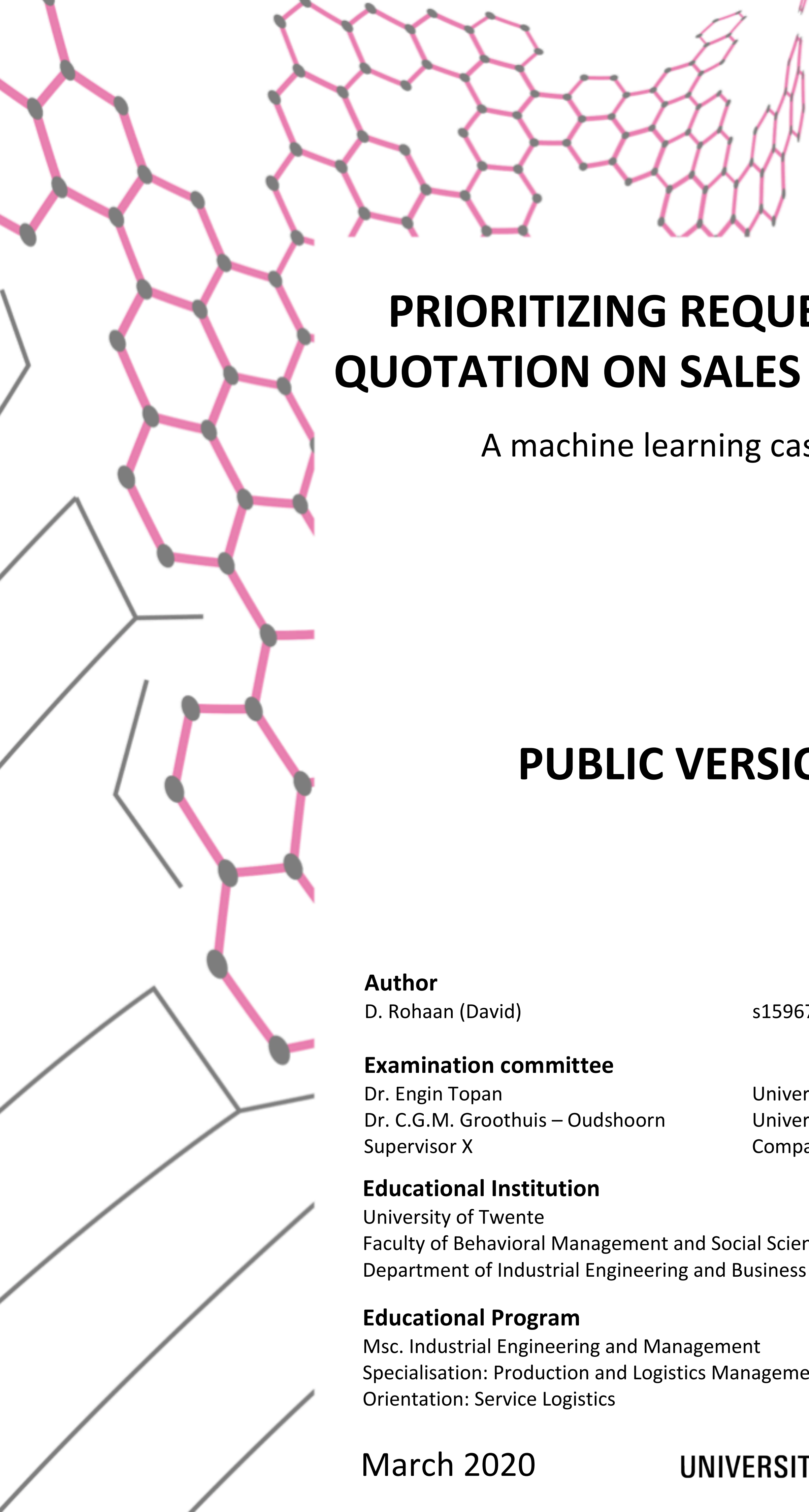


PRIORITIZING REQUESTS FOR QUOTATION ON SALES POTENTIAL

A machine learning case study

PUBLIC VERSION

Author

D. Rohaan (David)	s1596721

Examination committee

Dr. Engin Topan	University of Twente
Dr. C.G.M. Groothuis – Oudshoorn	University of Twente
Supervisor X	Company X

Educational Institution

University of Twente
Faculty of Behavioral Management and Social Sciences
Department of Industrial Engineering and Business Information Systems

Educational Program

Msc. Industrial Engineering and Management
Specialisation: Production and Logistics Management
Orientation: Service Logistics

March 2020

UNIVERSITY OF TWENTE.

Preface

Dear reader,

This thesis is the result of my master graduation project for the *Production & Logistics Management* specialization of the *Industrial Engineering and Management* Master's degree at the University of Twente. Carrying out the research of this thesis has been a tremendous learning curve in which I got the opportunity to deep dive into the fields of Machine Learning and Natural Language Processing and learn how to program in Python, which I had never done before. In this preface I would like to take the opportunity to express my gratitude to the people who helped me with the realization of this thesis.

First, I would like to thank Supervisor X for both allowing me to carry out my master graduation assignment at Company X and for his guidance and feedback throughout.

Second, I would like to thank Dr. Engin Topan and Dr. C.G.M. Groothuis-Oudshoorn for their support, valuable input and feedback on my draft reports.

Finally, I want to thank family and friends for their support and help.

David Rohaan
Amsterdam, The Netherlands
23-03-2020

Management summary

The research of this master thesis has been carried out to enable Company X to prioritize requests for quotation (RFQs), which are noncommittal requests from customers for a quote of spare parts and/or exchange of parts, on sales potential. Currently, Company X receives a large number of RFQs exceeding the capacity of the Customer Support Department, which is responsible for responding to the RFQs. In addition, the Customer Support Department handles many RFQs that have little/no chance of converting into a sale. Company X has already taken some initiative to address these issues by developing a tool in Python which supposedly prioritizes RFQs on sales potential (binary). Yet, this tool, to which will be referred as the concept prioritization tool, has not been implemented in day-to-day operations, nor has its performance been determined and more importantly, Company X has expressed to seriously question its validity. Moreover, even if the prioritization tool would adequately perform, the input data required by the (concept) prioritization tool currently has to be entered manually. Company X has asked for this to be automated through the development of a tool that can autonomously recognize- and extract relevant RFQ data on a selected number of features, to which will be referred as the information extraction module. The research carried out in this thesis has addressed the above by answering the main research seen below.

"How can Company X be enabled to have their incoming RFQs automatically, autonomously and (more) accurately prioritized on their sales potential?"

The main research question, due to its complexity, was split in the following sub-questions:

1) *What tools and/or models are available in literature?*
2) *How does the concept prioritization tool work/perform?*
3) *How does the prototype prioritization tool work/perform?*
4) *How does the operational prioritization tool work/perform?*
5) *How does the information extraction module work/perform?*

RQ1 was answered with a literature review which focused on finding information on both verification/validation and improvement of the prioritization tool, measuring tool performance and automatic/autonomous information extraction. The results of the literature review have been discussed in chapter 2, i.a. a B2B sales potential prediction methodology (Bohanec, et al., 2015), performance metrics and information extraction methods such as Named Entity Recognition (NER), pattern search and vocabulary matching.

RQ2 entailed a current system analysis, which has been discussed in chapter 3. There, a global description of the concept prioritization tool was given, whereas the code and a technical description can be found in Appendix B and C respectively. Analysis of the code behind the concept prioritization tool revealed many defects were present, amongst which data leakage which caused the model to "predict" solely on the presence of certain features rather than the intrinsic information of the data itself, resulting in a (unjust) near perfect performance.

RQ3 was answered in chapter 4 by implementing verification/validation changes addressing the defects of the concept prioritization tool which resulted in the prototype prioritization tool. This part of the research consisted in essence of re-designing and re-building the concept

prioritization tool from scratch, addressing e.g. missing functions, unrepresentative data, unrepresentative quote line-sales order linking, double weighting of features, inability to handle features with a large number of categorial values, faulty data splitting method, etc. In chapter 4, a description of- and arguments for the implemented changes are given, whereas the changes in code can be found in Appendix D.

RQ4 was answered in chapter 5 in which an attempt was made to improve prediction performance of the prototype prioritization tool through application of the B2B sales potential prediction methodology by Bohanec et al. (2015), resulting in the operational prioritization tool. From this chapter followed that the operational prioritization tool is a 13-feature Random Forest model with hyper parameter configuration {'n_estimators': 400, 'min_samples_split': 2, 'min_samples_leaf': 1, 'max_features': 13, 'max_depth': None, 'criterion': 'gini', 'bootstrap': False} and an (optimal) classification threshold of 0.62.

Compared to the historical 17% quote-to-sales order conversion rate of Company X, resulting from the mental decision models of the Customer Support Department, the operational prioritization tool predicts 48.5% correctly to be a sale, which is an increase of $\frac{0.485-0.170}{0.170} * 100\%$ = 185.3%. The need for the operational prioritization tool sprung, amongst other things, from the fact that the Customer Support Department receives more RFQs than they can process. Therefore, use of the operational prioritization tool in the future will allow the Customer Support Department to respond to more RFQs and, under assumption of the same distribution of “Sales”/“No sales” amongst the RFQs that previously could not be processed, generate more sales. In addition, the research carried out in this thesis also contributes to theory by giving an indication of the improvement that can be gained through incorporation of machine learning over manual handling in B2B sales forecasting.

RQ5 was answered in chapter 6 where a description is given of the information extraction module (the code can be found in Appendix F). The information extraction module accesses the Company X RFQ inbox (Exchange outlook) and extracts the part number(s) and partial customer email address from RFQs using an Entity Ruler combined with custom tokenizer and function respectively. This extracted information can theoretically be linked to master part- and master customer data and therefore used to create the machine learning data set required as input for the operational prioritization tool. However, in practice partial customer email address cannot be linked to master customer data due to non-existence of (partial) customer email address data. Consequently, until Company X creates (partial) customer email address data, prioritization of RFQs on sales potential cannot happen automatically and autonomously.

Performance evaluation (manually) of the information extraction module on 100 historical RFQs revealed that part number(s) and partial customer email address were always correctly identified. In addition, besides recognizing the correct part numbers, the information extraction module also recognizes numbers occurring in the RFQs falsely to be part numbers. These numbers are in fact existing part numbers but not in the context of the RFQ. To tackle this issue an array has been created in which frequently falsely identified (part) numbers can be stated which will then not be given to the Entity Ruler and therefore no longer (falsely) recognized.

Finally, chapter 7 addresses the conclusion, limitations and recommendations of this research. Here, it follows that the answer to the main research question is sequential application of the information extraction module and operational prioritization tool, as illustrated in Figure 2. Yet, prioritization cannot happen automatically and autonomously until Company X creates (partial) customer email address data which, given the time saving potential for the Customer Support department, is highly recommended.

Moreover, the following recommendations are made:

- Improve data handling/storage. In Section 7.2 examples of current bad practices are given regarding data handling/storage because of which data products such as the operational prioritization tool suffer (*"Garbage in, garbage out")*. Concrete recommendations to improve data handling/storage are stated in Section 7.3.
- Re-train the prioritization tool regularly using the most recent available data. The machine learning data set of the operational prioritization tool contains multiple ABC classified features whose top contributing categories (to the total sales order revenue) are not encoded. Thus, if a shift in top contributing categories would occur over time, the model would raise an error as it has not seen these categories (during training) before. This can simply be solved by regularly re-training the model using the most recent available data.
- Investigate different recall-precision trade-offs. In this research, improvement of the prioritization tool focussed on metric F1 score, in which precision and recall have equal weighting, whereas the "optimal" weights are subjective. Therefore, it is recommended to have a decision maker investigate the consequence of different trade-off weightings and determine "optimality".
- Prioritize on sales potential and -value. Currently, prioritization happens solely on sales potential and therefore a low-value screw could theoretically be prioritized over a high-value engine with neglectable difference in probability of sale. It is recommended to repeat the research carried out in this thesis where the target feature represents a trade-off between sales potential and -value.
- Investigate information extraction module trade-off between false positives and false negatives. This recommendation more specifically regards the array created to diminish the extend of numbers often falsely recognized to-be part numbers. Elements specified in this array will no longer be (falsely) recognized. Therefore, it is recommended to investigate the consequence of excluding a (part) number and the circumstances (e.g. part number demand rate/part number value) under which it can be justified.

Table of Contents

1. Introduction

In this chapter background information about Company X, for whom the research of this thesis has been carried out, is provided in Section 1.1. In addition, a description of the problem (context) is given in Section 1.2, the research objective is stated in relation to the problem definition in Section 1.3, the research questions are listed in Section 1.4 and finally a table containing an outline for the remainder of this thesis is provided in Section 1.5.

1.1 Company description

This section is left out because of confidentiality reasons.

1.2 Problem description

The problem that Company X currently faces, and that this master thesis aims to solve, is that Company X believes it is missing out on sales. Currently Company X receives a large number of requests for quotation (RFQs). *RFQs* are noncommitted requests for a quote of spare parts and/or exchange of parts that do not necessarily result in a sale. In fact, the current RFQ to sale conversion rate is about 17%, which is the average ratio that an RFQ ever becomes a sale. Due to the large amount of incoming RFQs compared to the number of employees in the *Customer Support Department*, which is responsible for responding to the RFQs, the respond time of Company X to an RFQ is relatively long. As a consequence, customers are complaining and gradually moving to competitors whom already process their RFQs in a 'smart' manner. In addition, the Customer Support Department also picks up RFQs that have less chance of converting into a sale and therefore part of these efforts, which are expensive and scarce (the number of employees in the Customer Support Department), is wasted as well. These problems are visualized in the problem cluster below.

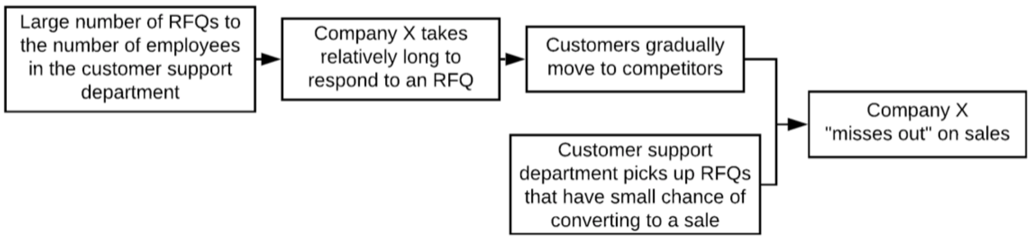

Large number of RFQs to the number of employees in the customer support department
Company X takes relatively long to respond to an RFQ
Customers gradually move to competitors
Customer support department picks up RFQs that have small chance of converting to a sale
Company X "misses out" on sales

Figure 1: Problem cluster Company X.

The problem of Company X missing out on sales cannot be addressed directly and is therefore indirectly addressed in this master thesis by identifying the core problem and tackling it. The core problem is a problem from the problem cluster that can be influenced and is not the consequence of another problem (Heerkens & Van Winden, 2012). The core problem can thus be found at the most left side of the problem cluster for which in this case there are two candidates, namely "Large number of RFQs to the number of employees in the Customer Support Department" and "Customer Support Department picks up RFQs that have small chance of converting to a sale". Here "Customer Support Department picks up RFQs that have small chance of converting to a sale" is chosen to be the core problem as it can clearly be influenced, while the number of RFQs nor the number of employees in the Customer Support Department can be influenced. The latter renders the candidate "Large number of RFQs to the number of employees in the Customer Support Department" infeasible to be a core problem.

1.3 Research objective

The objective of this research, which tackles the core problem defined in Section 1.2, is to enable Company X to prioritize RFQs such that it can select RFQs that are most likely to convert into a sale, and in that way, use its limited resources (number of employees in the Customer Support Department) in the best way to generate maximum sales. Company X has already taken some initiative to prioritize RFQs on sales potential by developing a tool in Phyton that, using RFQ data on a selected number of features, is supposed to binary classify RFQs reflecting their likelihood to convert into a sale. Here, a *feature* is defined as a measurable property or characteristic of a phenomenon being observed/analyzed (Datarobot, sd). However, this tool, to which will be referred throughout the rest of this thesis as the *concept prioritization tool*, is not yet deployed in day to day operations nor has its performance been determined. In addition, and more importantly, Company X has expressed to seriously question the validity of the predictions of the concept prioritization tool and thus its current performance. The latter due to, amongst other things, the fact that the creation of the concept prioritization tool had to be rushed since its creator resigned during its development.

Therefore, the first component of the research objective mentioned above is to verify/validate the concept prioritization tool, resulting in a *prototype prioritization tool*. Next, an attempt will be made to both improve the prediction performance of the prototype prioritization tool as well as enable Company X to prioritize within classes, resulting in an *operational prioritization tool*. Further, the input data for the concept prioritization tool, data of the RFQs on a selected number of features, currently has to be entered manually. Company X has asked for this to be automated through development of a tool that can autonomously recognize- and extract RFQ data on the selected number of features. Therefore, the second component of the research objective is the creation of a tool that can perform *Information Extraction* (IE), which is defined as the automated retrieval of specific information; usually in Natural Language Processing (NLP) to extract structured- from unstructured text (Rouse, 2018). Throughout the rest of this thesis there will be referred to this tool as the *information extraction module*.

At the end of this thesis, after achieving both research objective components, the goal is to create the process as visualized in Figure 2 for Company X. In this process incoming RFQs are read in by the information extraction module from the RFQ inbox. Then, the information extraction module recognizes- and extracts RFQ data which will be used to create the machine learning data set required as input by the operational prioritization tool. Next, the operational prioritization tool will prioritize the RFQs according to their likelihood to convert into a sale.

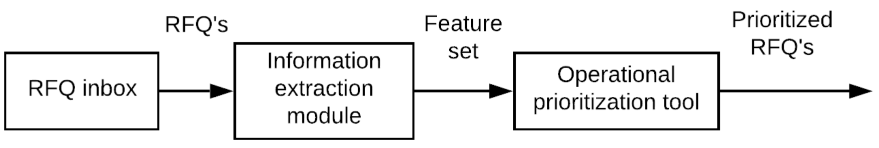

RFQ inbox
RFQ's
Information extraction module
Feature set
Operational prioritization tool
Prioritized RFQ's

Figure 2: To-be process for Company X.

1.4 Research questions

To address the research objective, the following main research question is formulated:

"How can Company X be enabled to have their incoming RFQs automatically, autonomously and (more) accurately prioritized on their sales potential?"

The main research question is split up into the following research questions:

RQ1. What tools and/or models are available in literature to…

1.1 Verify/validate the concept prioritization tool?
1.2 Improve prediction performance of the prototype prioritization tool?
1.3 Measure tool performance?
1.4 Automate data recognition and -extraction?

The answers to these sub-questions will provide a solution framework.

RQ2. How does the concept prioritization tool work/perform?

2.1 How is the concept prioritization tool build?
2.2 How is the performance of the concept prioritization tool?

The objective of sub-question 2.1 is two-fold; it contributes to the understanding of the concept prioritization tool and serves as a starting point for verification/validation which is addressed more elaborately in Chapter 4. Next, sub-question 2.2 gives insight in the performance of the concept prioritization tool.

RQ3. How does the prototype prioritization tool work/perform?

3.1 What changes in the concept prioritization tool are necessary for the purpose of verification/validation?
3.2 How is the performance of the prototype prioritization tool?

Sub-question 3.1 has the objective to identify- and implement changes in the concept prioritization tool which are necessary for the purpose of verification/validation, resulting in the prototype prioritization tool. Sub-question 3.2 gives insight in the performance of the prototype prioritization tool.

RQ4. How does the operational prioritization tool work/perform?

4.1 What historical sales data is available?
4.2 What data- and algorithm combination yields the best prediction performance?
4.3 What data insights can be used for organizational learning?
4.4 How is the performance of the operational prioritization tool?

First, sub-question 4.1 entails an investigation of the available historical sales data and potential creation of additional custom features. Second, sub-question 4.2 has the objective of improving the prediction performance of the prototype prioritization tool. Third, sub-question 4.3 will provide visualizations of data insights which can be used for organizational learning. Finally, sub-question 4.4 will give insight in the performance of the operational prioritization tool.

RQ5. How does the information extraction module work/perform?

5.1 How is the information extraction module build?

5.2 How is the performance of the information extraction module?

The answer to sub-question 5.1 will be the creation of the information extraction module. Next, sub-question 5.2 will provide insight in the quality of the information extraction module.

1.5 Structure of the report

Report structure		
Chapter	**RQ**	**Short description chapter content**
2. Literature review.	1	Relevant literature for achieving the research objective is discussed. Verification/validation and improvement of the prioritization tool, performance metrics and feature recognition/extraction are addressed.
3. Current system analysis.	2	Analysis of the concept prioritization tool. Descriptions of the machine learning data, front end-, back end- and performance of the concept prioritization tool are given in this chapter.
4. Verification/validation of the concept prioritization tool.	3	In this chapter changes necessary for verification/validation of the concept prioritization tool are identified and implemented, resulting in the prototype prioritization tool. In addition, the performance of the prototype prioritization tool is evaluated.
5. Improving the prototype prioritization tool.	4	In this chapter an attempt is made to improve the prediction performance of the prototype prioritization, resulting in the operational prioritization tool. This improvement attempt is made using the methodology by Bohanec et al. (2015). Here, i.a. feature selection, hyper parameter tuning and threshold optimization are carried out.
6. Creation of the information extraction module.	5	The information extraction module is created, analyzed and evaluated performance wise.
7. Conclusion, recommendations and limitations.	-	Self- explanatory.
8. References	-	Self- explanatory.
Appendix A	-	Explanation/description for each feature.
Appendix B	-	The code from the concept prioritization tool is shown. In addition, explanation of code is provided in blue text.
Appendix C	-	Technical description of the concept prioritization tool is provided.
Appendix D	-	Verification/validation changes in concept prioritization tool code, resulting in the prototype prioritization tool, are shown.
Appendix E	-	Tables corresponding to feature selection and hyper parameter tuning are shown here.
Appendix F	-	The code behind the Information Extraction module with corresponding explanation is shown here.

Table 1: Structure report.

2. Literature review

This chapter contains a solution framework to the research objective as seen in Section 1.3. It comprises an introduction to (the variations of) machine learning, a methodology to predict B2B sales potential using supervised machine learning and a guide to autonomous data extraction using Natural Language Processing (NLP). Due to the size of this chapter a conclusion is given at the end of each section and a summary at the end of the chapter.

2.1 Machine learning

The concept prioritization tool has been created using Machine Learning (ML), which is defined as an application of artificial intelligence that provides systems the ability to automatically learn and improve from experience without being explicitly programmed (Expert System, sd). ML differs from traditional programming in the required input and resulting output as seen in Figure 3.

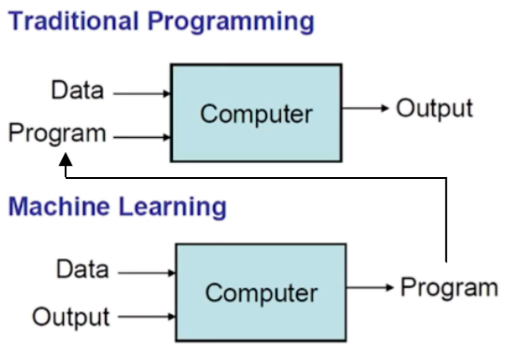


Figure 3: Machine Learning and traditional programming (MIT, 2016).

The basic paradigm for machine learning consists of three steps; observe instances, infer on the process that generated the instances and use inference to predict on previously unseen cases (MIT, 2016). There are two variations on this paradigm: supervised- and unsupervised learning, of which the former yields slightly better results for a two class prediction problem (Kaggle, 2018) (MIT, 2016).

Unsupervised ML

Unsupervised learning tries to infer latent features by clustering training instances into nearby groups (MIT, 2016). Clustering is an optimization problem in which the dissimilarity of all clusters (C) is minimized. In the formula's below c represents a single cluster and e represents a single instance within a cluster (MIT, 2016).

$$Variability(c) = \sum_{e \in c} Distance(mean(c), e)^2$$

$$Dissimilarity(C) = \sum_{c \in C} Variability(c)$$

s.t Minimum distance between clusters <u>or</u> minimum number of clusters

Note that without the constraints the optimization problem is fairly easy; each single instance would be its own cluster as then the variability(c) and dissimilarity(C) would be zero.

Supervised ML

Supervised machine learning, visualized in Figure 4, starts by collecting historical labelled data (data whose class is known) of instances, reflecting features (that could be) of importance in predicting class label. Then, data cleaning takes place after which the cleaned data is split into a training- and testing data set (usually 70/30 or 80/20). Next, different classifiers are trained on different feature subsets of the training data. A classifier is defined as an algorithm that identifies to which sub-population an observation belongs, on the basis of a training data set containing observations whose sub-population memberships are known (Aggarwal, 2014). After, these trained classifiers are applied to the testing data and their predicted class labels are compared to the known true class labels, with which the prediction performance of the feature subset- and classifier combination can be determined. The prediction performance is determined using the measure most appropriate for the task at hand (Section 2.2.3.2). Finally, the combination of feature subset and classifier that yields the best prediction performance is adopted and applied to future/unseen cases.

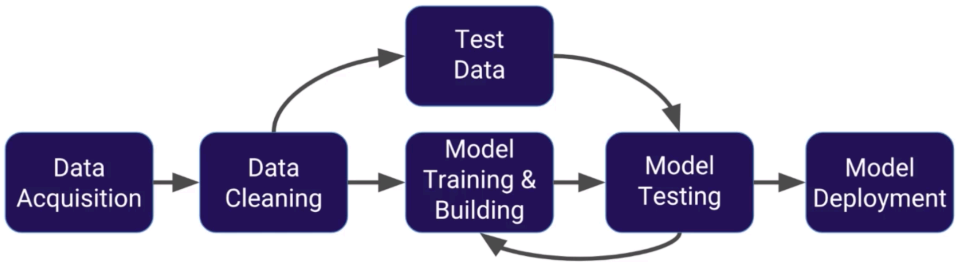


Figure 4: Supervised machine learning process (Udemy, 2019).

Conclusion

From this section it can be concluded that the concept prioritization tool is created using supervised machine learning; it is a classifier trained on historical quote data (of a selected number of features) and their corresponding label (whether the quote converted to a sale or not). Here, the supervised machine learning framework, shown in Figure 4, can be used to verify/validate the concept prioritization tool.

2.2 Predicting sales potential

This section of the literature review is structured according to a methodology from Bohanec et al. (2015) specifically designed for forecasting B2B sales potential using supervised machine learning, which is shown in Figure 5. The methodology yields a heuristic operational solution to the problem as it cannot be solved to optimality without an exhaustive search, which in practice is often impossible due to time constraints.

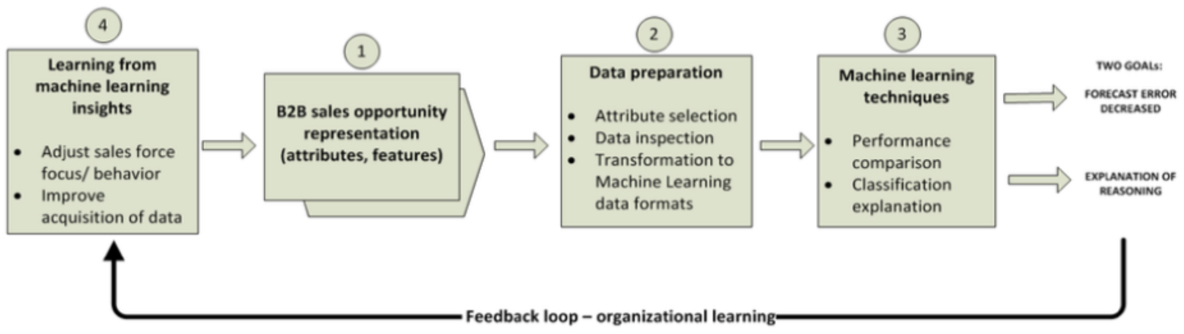

4
Learning from machine learning insights
Adjust sales force focus/ behavior
Improve acquisition of data
1
B2B sales opportunity representation (attributes, features)
2
Data preparation
Attribute selection
Data inspection
Transformation to Machine Learning data formats
3
Machine learning techniques
Performance comparison
Classification explanation
TWO GOALS:
FORECAST ERROR DECREASED
EXPLANATION OF REASONING
Feedback loop – organizational learning

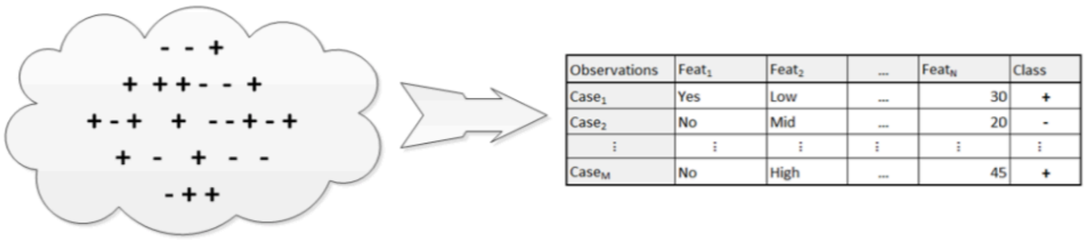
Figure 5: Predicting B2B sales potential methodology (Bohanec, et al., 2015).

2.2.1 Sales opportunity representation

The first step in this methodology is to create a sales opportunity representation, based on historical sales cases, in terms of features. Besides using the available data for feature representation, example cases of B2B sales prediction with machine learning have shown that the introduction of additional custom features (so called 'meta variables') can potentially capture influence which the default features do not, and can be significant predictors (Mortensen, et al., 2019). Examples of such meta variables are:

- Fields Completed — Count of the number of fields completed in one record.
- Task Count — Count of the number of tasks for the customer account associated with a sales opportunity.
- Age-related variables — Analyzes the impact from the age of opportunities.
    - Open Time — The duration that a sales opportunity remained open in the system.
    - Last Action time — The duration from when an opportunity was created to the time of last activity on that sales opportunity.
    - Valid Open Time — A Boolean variable that equals 1 for sales opportunities with positive Open Time and 0 for the remaining sales opportunities.

In addition, in this phase the sales department should be interviewed about the features they deem most important in predicting sales potential, which should then be included in the sales opportunity representation. Based on the selected features, real world cases reflecting sales history need to be described with values for these features (Figure 6).



Observations	$Feat_1$	$Feat_2$	...	$Feat_N$	Class
$Case_1$	Yes	Low	...	30	+
$Case_2$	No	Mid	...	20	-
⋮	⋮	⋮	⋮	⋮	⋮
$Case_M$	No	High	...	45	+

Figure 6: Transforming historical cases into a learning data set.

2.2.2 Data preparation

The second phase is the data preparation which is said to be highly depending on the specific problem at hand (Bohanec, et al., 2015) but if not done properly it can lead to "Garbage in, Garbage out", e.g. incorrect or poor-quality input will produce faulty output and therefore severely impact the validity of the results as the machine learning data set would not accurately reflect the population of interest. During this phase the data that resulted from the sales opportunity representation phase is investigated, potentially cleaned/transformed into a (high-quality) learning set and afterwards split into a train- and test set.

2.2.2.1 Data cleaning

Outlier detection

Outlier detection is the process of finding data instances whose behaviors present significant deviations from the majority patterns (García, et al., 2015). Two methods for detecting outliers in are:

- The unsupervised learning algorithm isolation forest (IF) (Medium, 2019). Isolation forest starts by sampling data for training a model (training data set). Then, a binary decision tree is created for one of the features with a random value for that feature between its minimum and maximum. Next, the last step is repeated for both sub-data sets that resulted from the binary tree split, until all data points are captured. Each data point gets an anomality score [0,1] based on how fast it has been captured. The idea behind the algorithm is that the "fewer" and "different" data points are isolated quicker because of which their anomality score will lie closer to 1 (Medium, 2019). Figure 7 shows the IF algorithm applied to a data set with two features.

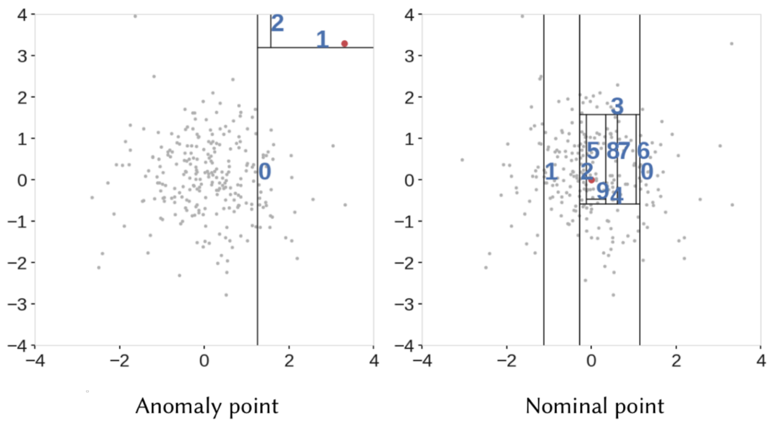


Figure 7: Visualization isolation forest.

- Z-scoring. The Z-score of an instance is calculated as $\frac{Observation-Mean}{Standard\ deviation}$ and represents the number of standard deviations an observation is away from its mean. For normally distributed data, Z-scores less than -3 and more than 3 are considered to be outliers.

Handling of missing data

Usually the treatment of missing data can be handled in one of the following ways (García, et al., 2015):

- The first approach is to discard instances containing missing values (MVs) for their features. It should be noted that this approach can only be safely adopted when data is Missing Completely At Random (MCAR), otherwise bias will be induced. This category of handling missing data also includes deleting features entirely, when their levels of MVs are considered to be too high (rule of thumb is 60-70%+ missing).
- The second approach is to retain- and encode instances with MVs; e.g. for numerical features that are positive in nature '-1' and for categorial features 'other' (Medium, 2019). Note that the former only works well with tree-based models.
- The third approach is the use of maximum likelihood procedures, where the parameters of a model for the complete portion of the data are estimated, and later used for imputation by means of sampling (García, et al., 2015).
- Last, multiple imputation by identifying the relationships between features and estimating the missing values (García, et al., 2015). Here it is important that multiple values are estimated for the missing fields to account for the uncertainty in these fields due to missingness. In this way a number of different complete datasets are formed and the results on these datasets should be pooled to obtain the final results.

Noise reduction

Statistical noise is defined as "unexplained variability within a data sample" and is especially relevant in supervised problems, where it alters the relationship between the informative- and dependent feature (Rouse, 2017) (García, et al., 2015). There are three types of noise (Medium, 2018):

- **Noisy data items**, which entails anomality's in the features and the target. This type of noise can be addressed with outlier detection.
- **Noisy features**, which entail weak/irrelevant features. This type of noise can be addressed with feature selection, on which is elaborated in more detail in Sections 2.2.3.1 and 2.2.3.3.
- **Noisy records**, which entail records (entire instances) that do not follow the form/relation which other records do. This type of noise can be addressed with cross validation by analyzing the folds with poor scores.

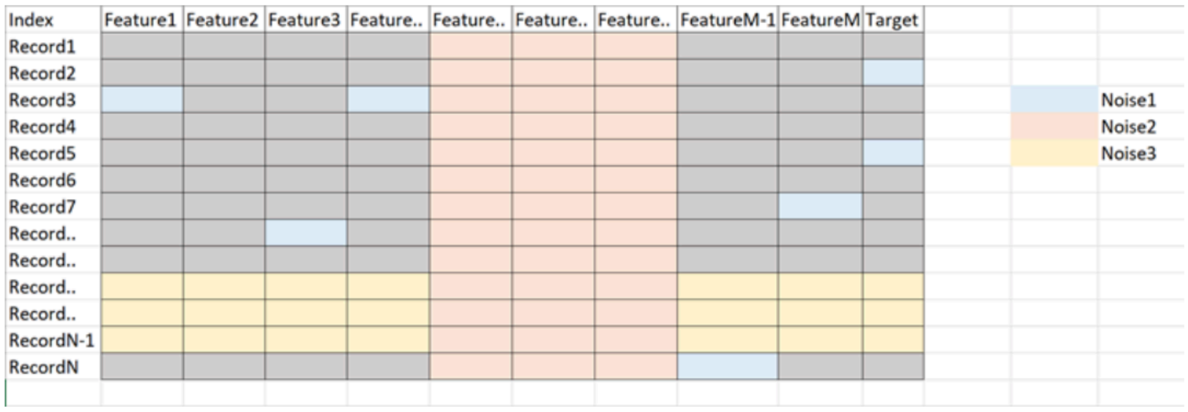


Figure 8: Types of noise.

2.2.2.2 Data transformation

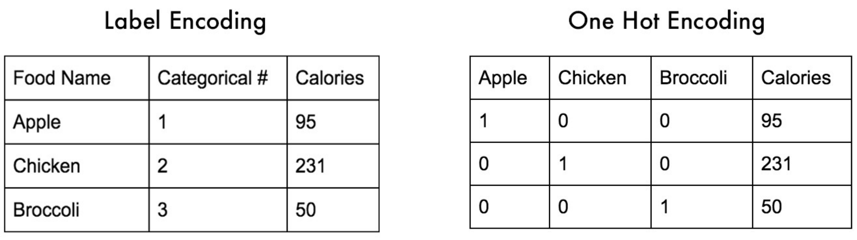
Data normalization

Data normalization is the process of expressing all features in the same measurement units and use a common scale or range (García, et al., 2015). Data normalization has the objective to give all features equal weight to prevent features from intrinsically influencing the result more/less due to their larger/smaller value (Medium, 2018). Three common normalization techniques for numerical features are:

- **Min-Max Normalization** normalizes value v (into v') by applying the following formula: $v' = \frac{v - Min_A}{Max_A - Min_A}(NewMax_A - NewMin_A) + NewMin_A$, in which Max_A/Min_A represent the original maximum- and minimum feature values respectively (García, et al., 2015).
- **Z-score normalization** normalizes value v (into v') by applying the following formula: $v' = \frac{v - \bar{A}}{\sigma_A}$, in which $\bar{A}/\sigma_A$ represent the mean- and standard deviation of values of feature A respectively (García, et al., 2015).
- **Decimal Scaling Normalization** normalizes value v (into v') by applying the following formula: $v' = \frac{v}{10^j}$, in which j is the smallest integer such that NewMax_A < 1.

Encoding categorial features

Categorial features can be encoded using Label encoding and One Hot Encoding (OHE). First, Label encoding ‘translates’ each category of a categorial feature into a number. The example in Figure 9 shows how “Apple”, “Chicken” and “Broccoli” are translated into label 1,2,3 respectively. However, the downside of this approach is that when dealing with more than two categories the model will misunderstand the data to be in some kind of order, e.g. 0 < 1 <2 in Figure 9, which is not the case (Medium, 2018). To overcome this, OHE can be used which transforms the original categorial feature data into a number of columns equal to the number of categories within the original categorial feature (dummy features). Here, each category is represented by its own column containing 0/1 entries depending on the presence of the category (Figure 9). However, one should be aware of the dummy feature trap when using OHE, also known as perfect multicollinearity, which is automatically induced. Perfect multicollinearity occurs when two or more independent features exhibit a perfect linear relationship. OHE automatically induces perfect multicollinearity because only one of the resulting dummy features will have value 1 while the rest will always have value 0. Therefore, when the values of k-1 dummy features are known, the last k^{th} feature value will also be known; e.g. for the example in Figure 9, when $Food_{Apple} = 0$ and $Food_{Chicken} = 0$ then $Food_{Brocolli} = 1$. The consequence of the dummy feature trap is that a single redundant dummy feature will be incorporated in the ML model which takes computational time and power but does not add value. Yet, the dummy feature trap can be easily avoided by simply excluding a single dummy feature for each one hot encoded feature.



Label Encoding

Food Name	Categorical #	Calories
Apple	1	95
Chicken	2	231
Broccoli	3	50

One Hot Encoding

Apple	Chicken	Broccoli	Calories
1	0	0	95
0	1	0	231
0	0	1	50

Figure 9: Label Encoding and One Hot Encoding example.

Data reduction

Data reduction comprises the set of techniques that, in one way or another, obtain a reduced representation of the original data (García, et al., 2015). The three forms of data reduction, as shown in Figure 10, are feature selection, instance selection and discretization.

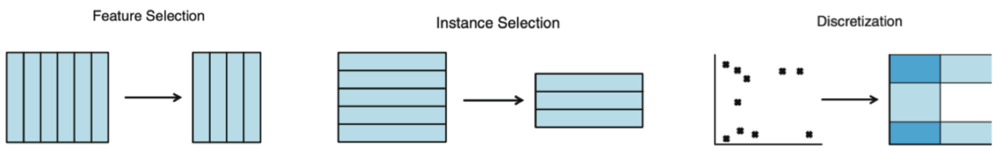


Figure 10: Forms of data reduction (García, et al., 2015).

First, feature selection is explained in more detail in Sections 2.2.3.1 and 2.2.3.3. Second, for instance selection, also known as data sampling, there are multiple methods:

- **Simple random sample with(out) replacement (SRSWOR)** of size s. This method draws s of the N tuples from T (s < N), where the probability of drawing any tuple in T depends on whether sampling is with- or without replacement (García, et al., 2015).

Without replacement means all samples have equal probability to be drawn whereas with replacement means that an instance may be drawn again.

- **Balanced sample**. This method creates a sample that is designed according to the target feature and is forced to have a certain composition according to a predefined criterion (García, et al., 2015). Two balanced sampling sub-methods are oversampling and under-sampling. **Oversampling** entails the replication/creation of new instances (usually minority class) from existing ones and **under-sampling** the exclusion of instances (usually majority class). It has been shown for multiple imbalanced data sets, which are data sets whose instances are not evenly distributed amongst classes, that both oversampling and under-sampling increase classification performance. Between the two balanced sampling sub-methods oversampling yields the highest increase as it increases the minority class recognition rate without sacrificing the majority class recognition rate (Batuwita & Palade, 2010).

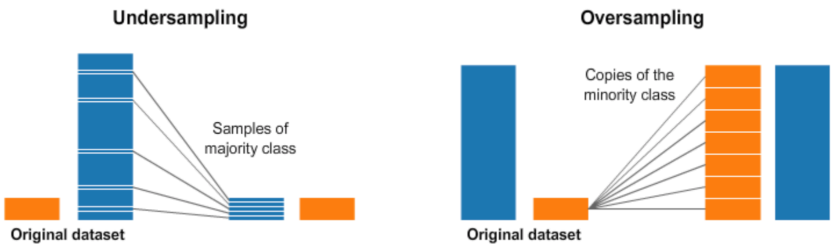


Figure 11: Under-sampling and oversampling.

- **Cluster sample.** If the tuples in T are grouped into G mutually disjointed groups or clusters, then a simple random sample (SRS) of s clusters can be obtained, where s<G (García, et al., 2015).
- **Stratified sample**. If T is divided into mutually disjointed parts called strata, a stratified sample of T is generated by obtaining a SRS at each stratum. This assists in ensuring a representative sample and is frequently used in classification tasks where class imbalance (unequal distribution of instances amongst classes) is present. This method differs from balanced sample in the composition of the sample according to the target feature; rather than on a predefined criterion in now depends on the natural distribution of the target feature (García, et al., 2015).

Last, data discretization transforms quantitative data into qualitative data, that is, numerical features into discrete or nominal features with a finite number of intervals, obtaining a non-overlapping partition of a continuous domain (García, et al., 2015). Discretization can be viewed as a data reduction method since it maps data from a huge spectrum of numeric values to a greatly reduced subset of discrete values. Yet, data discretization of a numerical feature (into k categories) comes with cost of an increased number of features (k-1 dummies) that take computational time and power.

2.2.2.3 Data splitting

After the data is cleaned, it is split into a training- and test data set. This by means of e.g.:

- **Repeat random sampling** is randomly splitting the data set into a train- and test set, usually 80/20 or 70/30 (MIT, 2016). However, this approach can be problematic when dealing with imbalanced data (Kuhn & Johnson, 2019).

- **Stratified random split.** See Section 2.2.2.2 for explanation. This splitting method is useful when dealing with imbalanced data as it ensures equal frequency distribution of classes in the train- and test data sets (Kuhn & Johnson, 2019).
- **Leave One Out** is used when there is little data; it uses all instances minus one to train on, saves the resulting model and repeats this process for different instances until all instances have been excluded once. The final resulting model is than the average of all saved models (MIT, 2016).
- **K-fold Cross Validation** divides the data into a training and testing set. After, the training set is divided into k folds (Figure 12). The model is trained on k-1 folds and tested on the remaining k^{th} fold. This process is repeated for multiple splits, in which the training/testing fold allocation differ. The final performance estimation is the average performance estimation of all k testing folds over all splits.

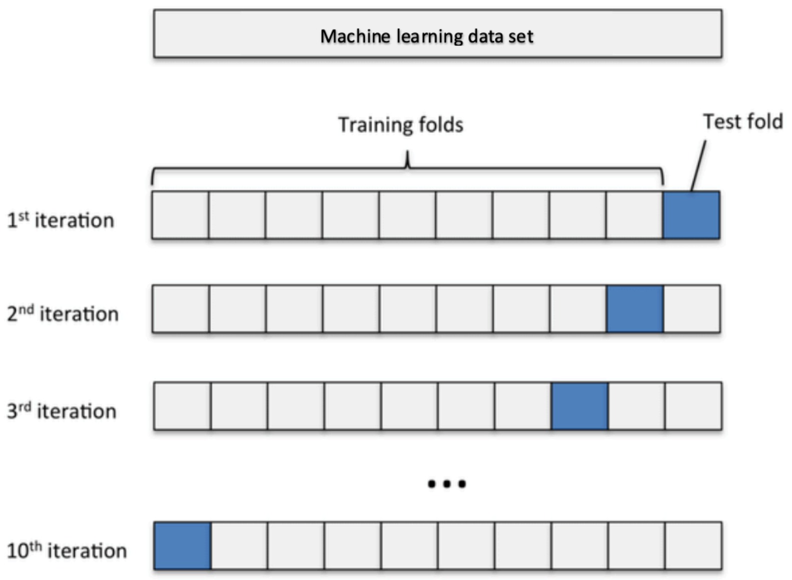


Figure 12: K-fold cross validation.

2.2.3 Machine learning techniques

In the third phase feature- and classifier selection and hyper parameter tuning are addressed. It should be noted that feature- and classifier selection entails two different problems, but are, depending on the feature selection technique addressed either sequentially or simultaneously. In addition, a methodology for feature- and classifier selection is provided that yields a heuristic operational solution. The objective of this phase is to identify the model that is best in predicting sales potential.

2.2.3.1 Feature selection

The feature selection problem is the problem of selecting the most informative feature subset in predicting the dependent feature (label). It arises because not all features are informative in predicting the dependent feature; features can be classified as relevant, irrelevant or redundant. Irrelevant features contain no information about the dependent (to-be predicted) feature, e.g. using someone's hair color to predict the weather. Redundant features provide information for the dependent feature that can be extracted from other features included in the feature subset used for the prediction problem. Redundant features may be relevant on their own but are redundant in the presence of another relevant feature, e.g. "the amount

VAT paid for a product" may be redundant in the presence of "product price" and "percentage VAT". The feature selection problem is therefore about finding an optimal balance between selecting a subset of features that are most relevant to the predictive modeling problem and performance of the machine learning (ML) model trained on the selected subset (Bohanec, et al., 2015).

Feature selection techniques
There are three major techniques in solving the feature selection problem; filter-, wrapper- and embedded methods.

- *Filter methods* try to extract general characteristics of training data and rank features on their relevance to the dependent feature in a pre-processing step without interacting with classification models. The advantage of this approach is that it allows for independence from classifiers and compare performance of different classification models using top ranked features. The downside of this approach is that it is incapable of detecting redundant features as these are likely to have similar rankings.
- *Wrapper methods* take a particular classification model as a part of the feature selection process. The contribution of a feature is determined with the performance of the model for different subsets of features. This approach is computationally expensive as it requires a training re-run for each feature subset; however, it has an advantage of being able to expose low performing- and redundant/noisy features. A wrapper method evaluates each combination of p out of n possible features and stores the best combination (called model(p)) (Analytics University , 2017). Next, this step is repeated for all smaller subset sizes 1,…,p resulting in best combinations model(1),…,model(p) from which again the best performing subset is chosen. The latter is done to detect potential redundancies as smaller subsets with similar performance indicate redundancy.

  It should be noted that for identification of the optimal set of size p, $2^p$of its subsets have to be evaluated and that this thus, due to its exponential size, is impossible for large p. Ways of working around the exponential size problem are forward selection, backward elimination and recursive feature elimination (Kaushik, 2016). First, forward selection starts with a null model after which in each iteration the feature that improves the model most is added until an addition of a new feature does not improve performance. Next, backward elimination starts with all the features and removes the least significant feature in each iteration improving the performance of the model. This is repeated until no more improvement is observed after removal of features. Last, recursive feature elimination is a greedy optimization algorithm aiming at finding the best performing feature subset. It repeatedly creates models and keeps aside the best or the worst performing feature at each iteration. It constructs the next model with the left features until all the features are exhausted. It then ranks the features based on the order of their elimination.
- *Embedded methods* are integrated into a specific classifier and evaluate/select features during the training process. Further, embedded methods may achieve a solution faster by avoiding the re-training of a classifier for each feature subset explored (García, et al., 2015).

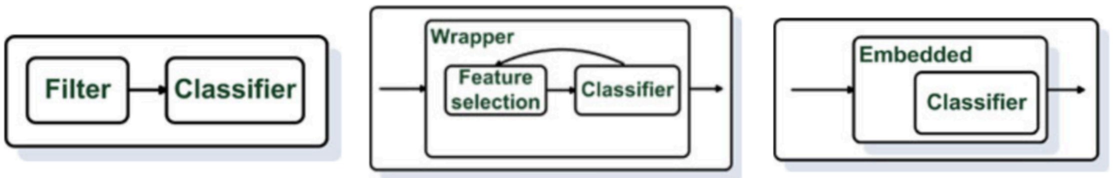


Figure 13: Filter-, wrapper- and embedded method.

2.2.3.2 Classifier selection

Classifier selection, depending on the feature selection method, takes place either before- (wrapper- and embedded method) or after (filter method) feature selection. A few examples of classifiers are the following:

- *RF* (*Random Forest)* creates a large number of decision trees (specified in the classifier), each having access to a random subset of features and instances, which increases diversity in the forest leading to more robust predictions. An instance is classified after a majority vote on the label by the forest (Medium, 2017).
- *Logistic regression*, similar to other types of regression, describes the relationships between one dependent (categorial) feature and one or more nominal, ordinal, interval or ratio-level predictor features (StatisticsSolutions, sd). A logistic regression model assigns, after being given a training data set, weights to features. Then, the logistic regression model uses these weights to predict a probability for an unseen case which is labelled according to the class thresholds that this probability lies in.
- *Gradient Boosting Classifier (GBC)* is a technique that generates (weak) classification models sequentially, where each model focusses on correcting its predecessor's errors. The final output is a combination of the results of all (weak) sequential models.

Metrics to evaluate and/or compare classifier performance on a selected feature subset are the following:

- Accuracy—The percentage of correctly predicted opportunities over the total number of opportunities ($\frac{True\ positive+True\ negative}{Total\ population}$). This metric is unsuitable for imbalanced data sets since it represents the total percentage correctly classified instances of all classes and does not give information about the percentage correctly classified instances per class; a low performance in a minority class easily goes unnoticed.
  - Precision — The percentage of correctly predicted won opportunities over the total number of predicted won opportunities ($\frac{True\ positive}{True\ positive+False\ positive}$).
  - Recall — The percentage of correctly predicted won opportunities over the total number of actual won opportunities ($\frac{True\ positive}{True\ positive+False\ negative}$).
  - F1 Score – This metric entails the harmonic mean of precision and recall and is calculated with the following formula: $F_1 = 2 * \frac{precision*recall}{precision+recall}$. The F1 score is introduced as there is often a trade-off to be made between precision and recall; the F1 score "punishes" extreme values for both.
- Access to feature importance — Certain algorithms provide information to evaluate the importance of features included in the model, e.g. *Percentage increased Mean-squared-error* which implies the loss of accuracy if a certain feature is missing in the model.

- Confusion matrix – A confusion matrix summarizes the performance of a classifier with respect to test data. It is a two-dimensional matrix, indexed in one dimension by the true class of an instance and in the other by the class that the classifier assigns.
- Efficiency — Resources used to build the model including time, memory, and complexity.
- AUC – This stands for Area Under the ROC (Receiver Operating Characteristics) Curve. The ROC curve shows the trade-off between true positives and false positives where AUC indicates the extent to which the ML model can distinguish between classes. An AUC score of 1 means the ML model can perfectly distinguish between classes and an AUC score of 0.5 means the ML model cannot differentiate between classes.

2.2.3.3 Feature selection methodology

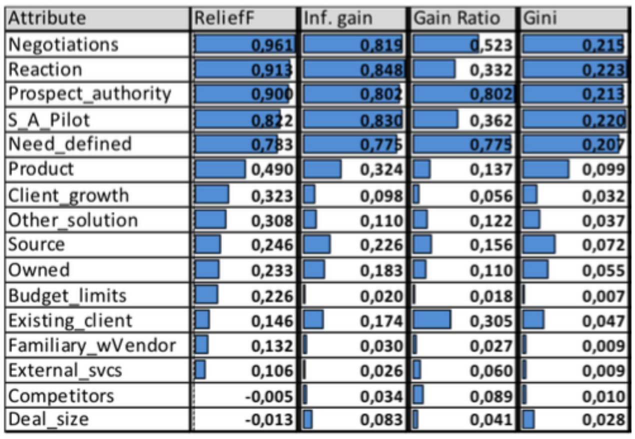
Unfortunately, as explained in Section 2.2.3.1, when addressing the feature- and classifier selection problem sequentially (filter method), noise and/or redundancies are not detected. Yet, on the other hand, an exhaustive search through all possible feature subsets applied to a single classifier (wrapper method), which does eliminate noise and redundancies, is often impossible in practice due to time constraints. Therefore, Bohanec et al. constructed a three step feature selection sub-methodology for B2B sales forecasting that does not require an exhaustive search through all possible feature subsets while still being able to detect and remove redundancies (Bohanec, et al., 2015):

1. Rank features according to relevance using filter methods.

In this step features are ranked using multiple filter methods. A useful tool for testing the significance of- and ranking features is the orange data mining suite which contains the following filter methods (Orange, 2015):

- *ReliefF*: the ability of an attribute to distinguish between classes on similar data instances.
- *Inf. Gain*: reduction of entropy. Entropy is a probability-based measure used to calculate the amount of uncertainty. The downside of Inf. Gain is that it prefers features with more values, while these are not necessarily more informative.
- *Gain Ratio*: a ratio of the Inf. Gain and the attribute's intrinsic information, which reduces the bias towards multivalued features that occurs in Inf. Gain.
- *Gini*: the inequality among values of a frequency distribution.
- *FCBF*: entropy-based measure, which also identifies redundancy due to pairwise correlations between features.
- χ^2: dependence between the feature and the class as measure by the chi-square statistic.
- *ANOVA*: the difference between average values of the feature in different classes.

Figure 14 below shows an example of testing the significance of features using different techniques in the orange data mining suite as described above. The example shows that there are approximately 5 informative- out of the 16 features. Once the features are ranked the results should be discussed with the sales department. Surprising results should be thoroughly analyzed and eventually the learning data set can be updated with additional cases and/or different features can be added. Alternatively, the sales department can view this as a learning opportunity and update their mental decision models.



Attribute	ReliefF	Inf. gain	Gain Ratio	Gini
Negotiations	0,961	0,819	0,523	0,215
Reaction	0,913	0,848	0,332	0,223
Prospect_authority	0,900	0,802	0,802	0,213
S_A_Pilot	0,822	0,830	0,362	0,220
Need_defined	0,783	0,775	0,775	0,207
Product	0,490	0,324	0,137	0,099
Client_growth	0,323	0,098	0,056	0,032
Other_solution	0,308	0,110	0,122	0,037
Source	0,246	0,226	0,156	0,072
Owned	0,233	0,183	0,110	0,055
Budget_limits	0,226	0,020	0,018	0,007
Existing_client	0,146	0,174	0,305	0,047
Familiary_wVendor	0,132	0,030	0,027	0,009
External_svcs	0,106	0,026	0,060	0,009
Competitors	-0,005	0,034	0,089	0,010
Deal_size	-0,013	0,083	0,041	0,028

Figure 14: Testing feature significance in orange data mining suite.

2. Build ML model by incrementally adding ranked features, monitoring maximal performance to detect the cut-off point.

In this step the best ranked features are added one-by-one to the training of the machine learning model and performance is measured each time a feature is added. The number of best ranked features that yields the highest performance is added. The graph below shows an example of how the effect on performance for multiple classifiers may look when incrementally adding features (filter method ReliefF used).

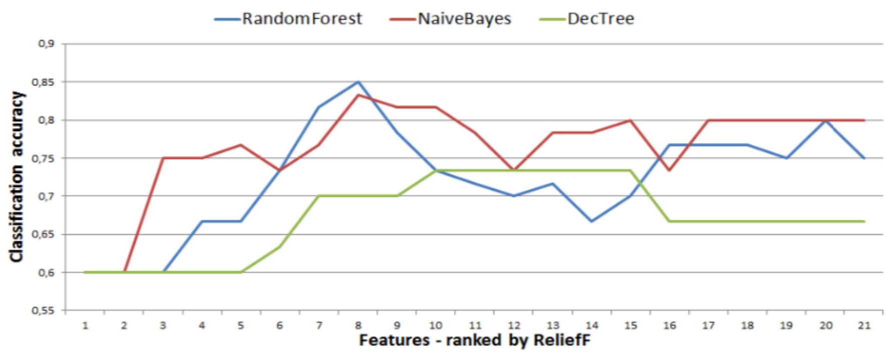


Figure 15: Monitoring CA when adding top ranked features one-by-one.

3. Eliminate noisy and redundant features with wrapper method.

Figure 15 showed it is possible to get negative- or no difference in performance when adding additional features which is an indication of noisy- and/or redundant features. A wrapper method can be used to eliminate such features; features are excluded one-by-one (top down) and performance is monitored. If exclusion of a feature increases performance, it is permanently excluded from the list. After this a minimum feature list for the data set is left.

2.2.3.4 Hyper parameter tuning

Hyper-parameters are the intrinsic parameters of a (machine learning) model which are set before- and used to control learning (TowardsDataScience, 2019). Therefore, hyper-parameter tuning is the problem of choosing a set of hyper-parameter values that yields the best (generalizable) performance improvement.

The most widely used strategies for hyper-parameter optimization are manual- and grid search. In manual search, the operator adjusts the parameters, possibly incorporating knowledge about how those adjustments will influence the behavior of the model and estimation procedure. Next, Grid search is an exhaustive search over a pre-defined grid, returning the best configuration. However, literature rather urges the use of random search, which evaluates a number of random configurations of a pre-defined grid, as it has shown that random search yields equally good or better configurations than grid search in a fraction of the computational time (Bergstra & Bengio, 2012). The idea behind this phenomenon is that not all hyper-parameters are equally important and grid search allocates too many experiments to the exploration of dimensions that do not matter and suffer from poor coverage in important dimensions. Whereas random search has the same efficiency in important dimensions as if it had been used to only search the relevant dimensions, illustrated in Figure 16.

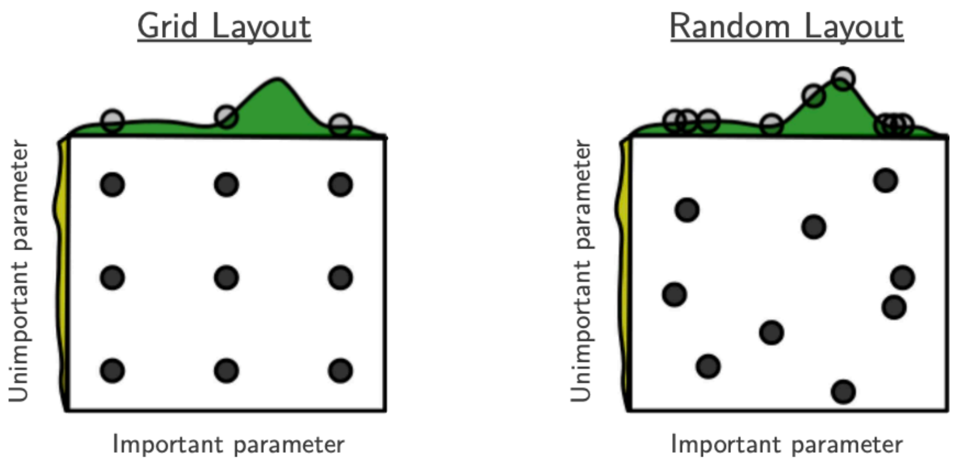


Figure 16: Grid search vs. Random search (Bergstra & Bengio, 2012).

2.2.4 Learning from ML visualizations

Last, in the fourth phase ML techniques and visualizations are used to gain/emphasize insights that the sales department can use to adjust their mental decision models, such as:

- Association rules are if-then statements that help show the probability of relationships between items in a dataset and can be used to reveal “preconditions” for class outcomes (Rouse, 2018). Each association rule is represented with three standard evaluation metrics: support, confidence and lift. The support measures the proportion of cases in the training data set which contain the left side of a rule. The confidence reflects the proportion of the cases in which the left- and right side are satisfied. The lift reports the ratio of the observed support to the expected lhs and rhs being independent (Witten et al., 2011).

- The Sieve multigram (Figure 17) shows correlations between features. A red/blue coloured line indicates negative/positive correlation and the thicker a line, the stronger the correlation.

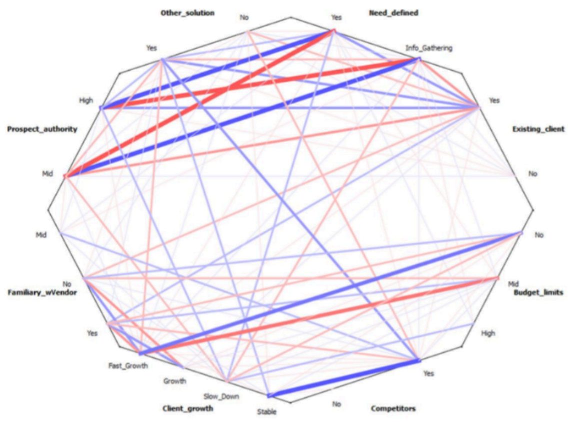


Figure 17: Sieve multigram.

- A classification tree is a structural mapping of decisions that lead to a decision about the class of an object. A classification tree is composed of branches that represent attributes, while the leaves represent decisions (Clark University, sd).

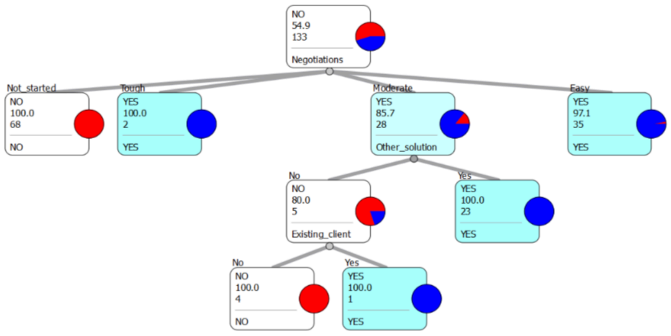


Figure 18: Example classification tree.

- A scatter plot is a graph of plotted points that shows the relationship between two features.

Conclusion

In this section the methodology by Bohanec et al. (2015) was shown for predicting B2B sales potential using supervised machine learning. This methodology can be used to both verify/validate the concept prioritization tool as well as to improve the prototype prioritization tool.

Verification/validation can be performed by retracing the steps in the creation of the concept prioritization tool and comparing these to those of the methodology. Here, especially the sales opportunity- and data preparation phase are useful. First, within the sales opportunity phase, the correct description of historical sales cases should be verified; e.g. quote line-sales order linking, properly merging of data frames, etc. Next, within the data preparation phase it should be verified whether data cleaning, data transformation and data splitting are carried out (correctly).

Attempting to improve the prediction performance of the prototype prioritization tool can be done in multiple ways for the different phases. First, in the sales opportunity phase improvement can be achieved by considering additional features/meta-variables. Second, in the data preparation phase, over-/under-sampling can be incorporated as this has shown to improve prediction performance for imbalanced data. Third, in the machine learning techniques phase, the sub-methodology on feature- and classifier selection (Section 2.2.3.3) and hyper-parameter tuning should be executed in an attempt to improve prediction performance. Last, in the fourth phase, to increase understanding rather than prediction performance, data visualizations can be created to show relationships between features and/or emphasize insights.

Last, multiple metrics for measuring tool performance were shown in Section 2.2.3.2. From this section it followed that, when dealing with imbalanced data, metrics F1 score, confusion matrix and AUC fit best for evaluating tool performance.

2.3 Information extraction module

This part of the literature review has the objective of finding out how the information extraction module, introduced in Section 1.3, can be built.

2.3.1 Natural language processing

In the present era big data is a booming topic. Big data refers to large volumes of data that have the potential to be mined for valuable insights and/or can be used for advanced analytics applications, e.g. machine learning (Rouse, 2019). The majority, an estimated 80-90%, of big data is unstructured data (e.g. emails), which is growing faster than any other type of data (i-scoop, sd). Unstructured data is information that does not have a recognizable structure; it comes in many forms and thus is not a good fit for a mainstream database (Rouse, 2018). A solution to analyzing such big unstructured data is Natural Language Processing (NLP). NLP is defined as a field of Artificial Intelligence that gives machines the ability to read, understand and derive meaning from human languages (Yse, 2019). The solution to the creation of the information extraction module lies in this field as NLP is said to have the ability to automate data extraction from large volumes of unstructured text (Li & Elliot, 2019).

2.3.2 Python vs R

Two programming languages that perform well in both fields of NLP and machine learning are Python and R, of which the best is said to be depending on the task at hand (Wu, 2019).

Python is an open source, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source for all major platforms and can be freely distributed (Python, sd).

R is an open source programming language and software environment for statistical computing and graphics. It provides a wide variety of statistical (linear and nonlinear modelling, classical statistical tests, time-series analysis, classification, clustering, …) and graphical techniques, and is highly extensible. The R language is primarily used amongst statisticians and data miners. One of R's strengths is the ease with which well-designed publication-quality plots can be produced, including mathematical symbols and formulas where needed.

To choose between Python and R, both languages are compared on a set of criteria which are deemed important for the task at hand, shown in Table 2 (Medium, 2018) (Ven, 2018). In Table 2 *Big data handling* represents how data is stored in system memory (memory constraints), *Prototyping* refers to the ease of building a digital product, *Machine learning* to the languages its capabilities in this field, *Syntax difficulty* represents the degree in which a language is similar to other programming languages, *Flexibility* refers to the ease of re-using blocks of code and/or the ability to (inter)connect other software programs/components, *Integration* represents the ease of adopting the language for Company X and/or the ease of connecting with other existing tools, *Speed* refers to the efficiency of large computations and *Visualizations* refers to the quality and options for creating plots.

	Weight	Python	R
Big data handling capability	1	+	-
Prototyping	1	+	-
Machine learning	1	+	-
Syntax difficulty	1	+	-
Flexibility	1	+	-
Integration	1	+	-
Speed	1	-	+
Visualization	1	-	+
	Total	**75%**	**25%**

Table 2: Python vs R comparison.

It should be noted that Table 2 only shows which language performs better for each criterion (awarded a plus sign) but not the extent to which the winner performs better. Moreover, the assumption is made that all criteria are equally important. From Table 2 it can be concluded that Python should be used for the creation of the information extraction module as it received a higher overall score.

2.3.3 NLP Python libraries

In this section the top NLP packages for Python are briefly described and compared (Table 3) on a set of criteria deemed important for the creation of the information extraction module (Elite Data Science, sd) (sunscrapers, 2018).

Natural Language Toolkit (NLTK)
NLTK is a leading platform for building Python programs to work with human language data. NLTK provides easy-to-use interfaces to over 50 corpora (collections of documents) and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning and wrappers for industrial-strength NLP libraries (nltk.org, sd). Downsides of NLTK are its steep learning curve, its speed and it is said to be not production-ready (Elite Data Science, sd).

TextBlob
TextBlob is a Python library for processing textual data. It provides a simple API for common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, translation, and more (TextBlob, sd). Downsides of TextBlob are its speed (sunscrapers, 2018) and its lack of relevant features (e.g. does not include vocabulary matching nor an entity recognizer).

CoreNLP
Stanford CoreNLP provides a set of human language technology tools. It is able to return the base form of words, their parts of speech, recognize entities, normalize dates, times, and numeric quantities, mark up the structure of sentences in terms of phrases and syntactic dependencies, indicate which noun phrases refer to the same entities, indicate sentiment,

extract particular or open-class relations between entity mentions, get the quotes people said, etc. CoreNLP is written in Java but can be used in Python using a wrapper.

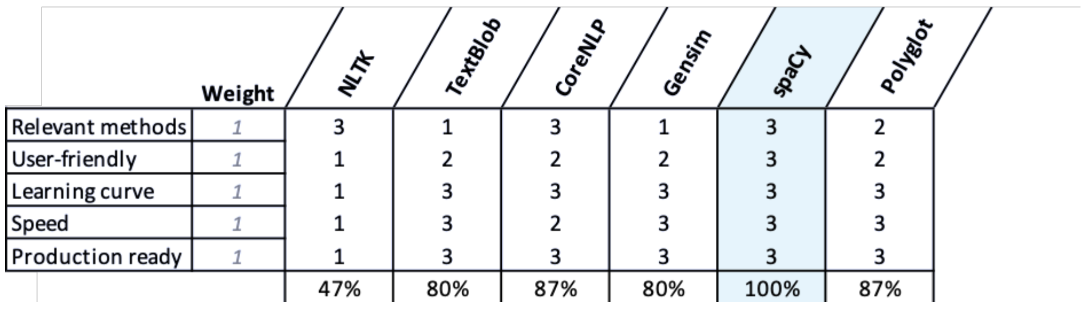
Gensim
Gensim is a well-optimized Python library for topic modelling, document indexing and similarity retrieval with large corpora (pypi.org, 2019). It is the most specialized library listed in this section but unfortunately not relevant for the information extraction- and prioritization tool.

Spacy
Spacy is a free, open-source library for advanced NLP in Python. Spacy forms together with NLTK the most popular NLP libraries. The main difference between Spacy and NLTK is that NLTK offers a wide range of algorithms for a certain problem, whereas Spacy only offers one: the state-of-the-art (Stack Abuse, 2019). Spacy is designed specifically for production use and helps build applications that process and "understand" large volumes of text. It can be used to build information extraction or natural language understanding systems, or to pre-process text for deep learning (spaCy, sd). Spacy has the advantage over NLTK and CoreNLP that its speed is much faster (spaCy, sd).

Polyglot
Polyglot is a natural language pipeline that supports massive multilingual applications. It is said to be similar to Spacy and an excellent choice for projects involving a language Spacy does not support. The only disadvantage compared to Spacy is that polyglot does not support vocabulary matching.

To choose between the top NLP packages, all packages are assigned scores on a set of criteria deemed (equally) important for the creation of the information extraction module, as shown in Table 3. Here, *Relevant methods* refers to the methods the package contains that can be used to recognize/extract data (e.g. entity recognizer, pattern search, vocabulary matching, etc.), *User-friendliness* refers to the ease of working with the package (number of algorithms per problem, documentation, interface, etc.), *Learning curve* refers to the ease of learning to use the package, *Speed* refers to the processing time of functionalities and *Production ready* refers to the ease of building a tool with the package.

In Table 3 each package has been assigned a score 1-3, relative to the best performing package, for each criterion: 1=bad, 2=average and 3=good. From Table 3 it follows that Spacy should be used for the creation of the information extraction module as it got awarded the highest overall score.



	Weight	NLTK	TextBlob	CoreNLP	Gensim	spaCy	Polyglot
Relevant methods	*1*	3	1	3	1	3	2
User-friendly	*1*	1	2	2	2	3	2
Learning curve	*1*	1	3	3	3	3	3
Speed	*1*	1	3	2	3	3	3
Production ready	*1*	1	3	3	3	3	3
		47%	80%	87%	80%	100%	87%

Table 3: Overview comparison top NLP Python packages.

2.3.4 NLP methods in Python

In this section Python NLP methods are described that are relevant for autonomous data recognition and -extraction.

Tokenization

Tokenization is the segmentation of a text into basic units - or tokens - such as words and punctuation, which can then be used as input for other processes, e.g. Named Entity Recognition. Spacy contains a tokenizer whose process is visualized below in Figure 18. Here, prefix means characters at the beginning of a sentence (such as $ (“), suffix characters at the end (such as km , . ! “) and exception means that punctation as part of a known abbreviation will be kept as part of the token, e.g., St. or U.S. (Udemy, 2019).

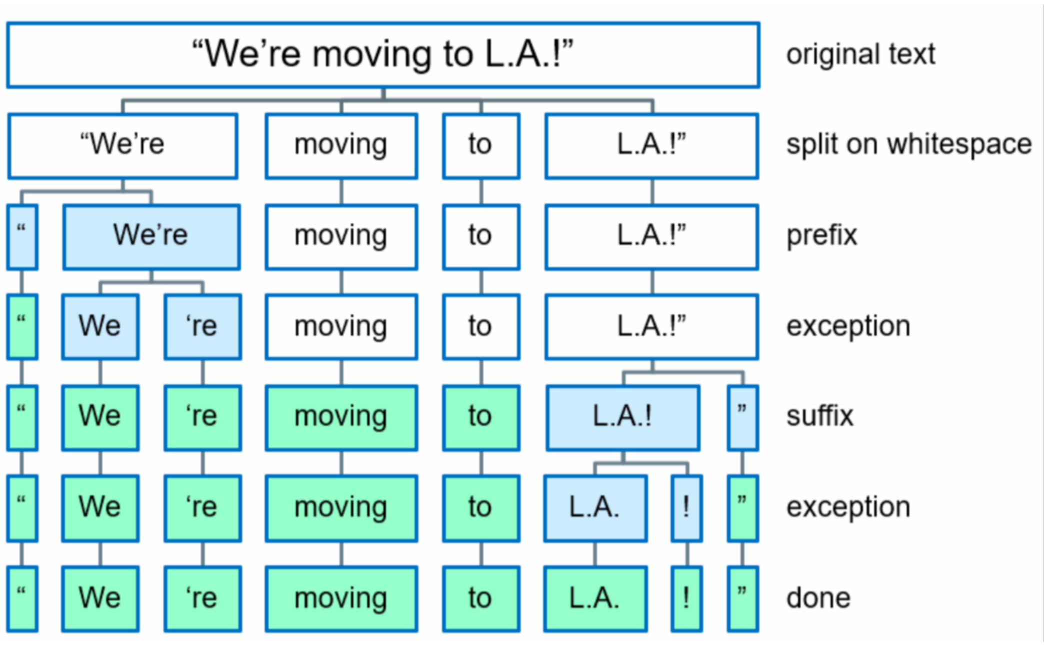


Figure 19: Tokenization process in Spacy visualized.

Spacy will isolate punctuation that does not form an integral part of a word. Quotation marks, commas, and punctuation at the end of a sentence will be assigned their own token. However, punctuation that exists as part of an email address, website or numerical value will be kept as part of the token (Udemy, 2019).

Lemmatization

Lemmatization is a method to reduce words to their root that takes the context of the sentence into account. Lemmatization looks beyond word reduction and considers a language’s full vocabulary to apply a morphological analysis to words (Udemy, 2019).

Stop words

Stop words are frequently occurring words containing little information, which usually are filtered out of the text to be processed. The Spacy package contains 326 stop words.

Pattern searching
Pattern searching is a method for finding parts of a document with an upfront known pattern, e.g. a phone number or email address. Pattern searching can be carried out using the Python package re.

Vocabulary matching
Vocabulary matching is a method in which a matcher object is created containing patterns. A matcher can then be applied to a document and will return the found matches. Vocabulary matching differs from pattern searching as it requires the to-be matched vocabulary to be known upfront, while pattern searching can find any pattern for which the structure is known. Vocabulary matching can be carried out by importing the Matcher from the Spacy package.

Part of Speech tagging (POS)
Part of Speech (POS) is defined as the grammatical class to which a word (token) belongs (Collins Dictionary, sd). Spacy can not only recognize the coarse-grained POS tag per token (grammatical class) but also a fine-grained POS tag (sub-class) within the coarse-grained class.

Named Entity Recognition (NER)
Named-entity recognition (NER) refers to a data extraction task that is responsible for finding, storing and sorting textual content into default categories such as the names of persons, organizations, locations, expressions of times, quantities, monetary values and percentages (Techopedia, sd). The Spacy package contains such an entity recognizer.

Conclusion
From this section can be concluded that the solution to the creation of the information extraction module lies in the field of Natural Language Processing (NLP). Moreover, it followed that in this field, Python- (Table 2), and within, library Spacy (Table 3) fit best for the development of the information extraction module. The NLP methods in Python that are especially relevant for autonomous recognition and -extraction of features are Named Entity Recognition, pattern search and vocabulary matching.

2.4 Summary

First, Section 2.1 gives an introduction to Machine Learning (ML). Machine learning is defined as an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed (Expert System, sd). The basic paradigm for machine learning consists of three steps; observe instances, infer the on process that generated the instances and use inference to predict on previously unseen cases (MIT, 2016). This section concluded that the concept prioritization tool has been created using the supervised machine learning variation, whose framework (illustrated in Figure 4) can be used to verify/validate the concept prioritization tool

Next, in Section 2.2 a methodology is shown for predicting B2B sales potential (Figure 5), which can be used to both verify/validate the concept prioritization tool as well as improve the prototype prioritization tool. The methodology consists of the following phases; sales opportunity representation, data preparation, machine learning techniques and learning from ML insights. First, in the sales opportunity representation phase, real world cases reflecting sales history are described with values for their features; both existing- and additional custom features. Second, in the data preparation phase, data cleaning (outlier detection, handling of missing data and noise reduction), data transformation (data normalization and -reduction) and data splitting takes place. Third, in the machine learning techniques phase, feature- and classifier selection take place, which should be noted are two separate problems, but are, depending on the feature selection technique (Section 2.2.3.1) addressed either sequentially or simultaneously. Moreover, in this phase performance metrics-, a feature selection sub-methodology (Section 2.2.3.3) and hyper-parameter tuning are discussed. Fourth, in the learning from ML insights phase, ML visualizations are created to gain/emphasize data insights which the sales department can use to adjust their mental decision models; e.g. association rules, decision tree visualization and scatter plots.

Last, in Section 2.3 was shown that the solution to the creation of the information extraction module lies in the field of NLP, which is defined as a field of Artificial Intelligence that gives machines the ability to read, understand and derive meaning from human languages (Yse, 2019). Further, in this section Python and R have been compared on a number of criteria deemed import for the development of the information extraction module from which it followed that Python fits best (Table 2). Similarly, a comparison of top NLP Python packages (Table 3) has been made from which it followed that Spacy fits best. Last, NLP methods relevant for autonomous information extraction and recognition are described in Section 2.3.4, e.g. Named Entity Recognition (NER), vocabulary matching and pattern search.

3. Current system analysis

The concept prioritization tool is an initiative by Company X to enable the Customer Support Department to respond more effectively and efficiently to RFQs through (supposedly) binary classification of RFQs on their sales potential. Concluded in Section 2.1, the concept prioritization tool is part of the supervised machine learning variation and in basis a classifier, specifically Gradient Boosting Classifier, trained- and evaluated with historical labelled data. In this chapter a description of the training- and evaluation data is given in Section 3.1. Next, Section 3.2 entails a description/analysis of the concept prioritization tool. Finally, the performance of the concept prioritization tool is evaluated in Section 3.3.

3.1 Machine learning data set

In this section a description of the *machine learning data* is provided, which is defined in this research as the historical labelled quote line data used for training and evaluating the prioritization tool. Here, a *quote* is the response to an RFQ containing the offer (price, lead-time, etc.) by Company X for the given request (Figure 20). A quote may contain multiple (quote) lines, each representing the requested quantity for a specific part (number). Consequently, a *sale* is defined as a line, within a quote, converting into a sales order.

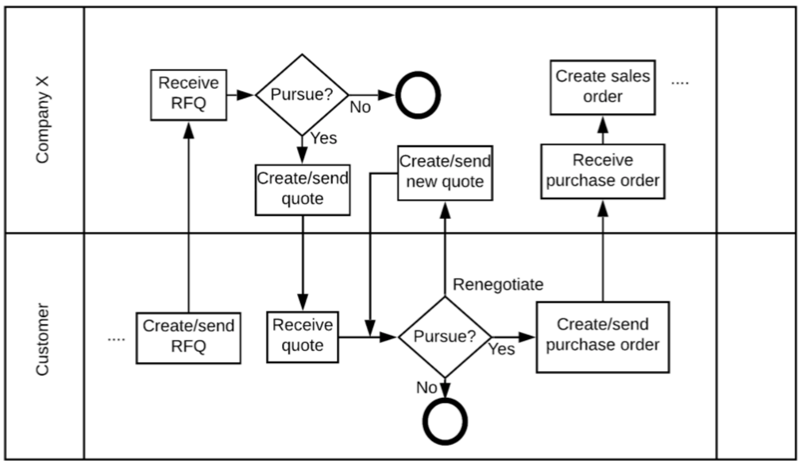


Figure 20: From RFQ to sales order.

The machine learning data set is derived from multiple data sources in preprocessing steps and comprises, in case of the concept prioritization tool, data of 326.429 quote lines with observations for the features shown in Table 4 over the period 2012-01-01 to 2019-01-01. An explanation of each feature can be found in Appendix A.

Machine Learning Data Set						
Column name	Feature name	Data type	% Missing- and Zero values	Data Source		
				Quote	Customer	Part
ACCTNO	Account number	Categorial	0.00%		X	
ACCOUNT_RATE	Account rate	N/A	100.00%		X	
ACCOUNT_TYPE	Account type	Categorial	0.06%		X	
ATA2	Airplane layout	Categorial	46.86%			X
COND	Condition	Categorial	88.20%			X
COUNTRY	Country	Categorial	0.00%		X	
ALT3_CODE_C	Customer type	Categorial	1.47%		X	

RNG	Delivery window time unit	Categorial	0.00030%	X		
DLV	Delivery window width	Numerical	13.67%	X		
MSLP_PRICE	Main Supplier List Price	Numerical	18.70%			X
PARTNUMBER	Part number	Categorial	0.00%			X
CAPABILITY	Repair capability	Categorial	75.47%			X
ROTABLE	Rotable part	Categorial	18.70%			X
REGION	Sales manager	Categorial	0.00%		X	
LINE_TYPE	Sales type	Categorial	0.00%	X		
STD_PART	Standard part	Categorial	18.70%			X
STK_TYPE	Stock type	Categorial	18.70%			X
SUBC	Sub account number	Categorial	0.00%		X	
SUBP	Sub part number	Categorial	0.00%			X
TOTAL_REVENUE	Revenue	Numerical	88.57%	X		

Table 4: Machine learning data set used by the concept prioritization tool.

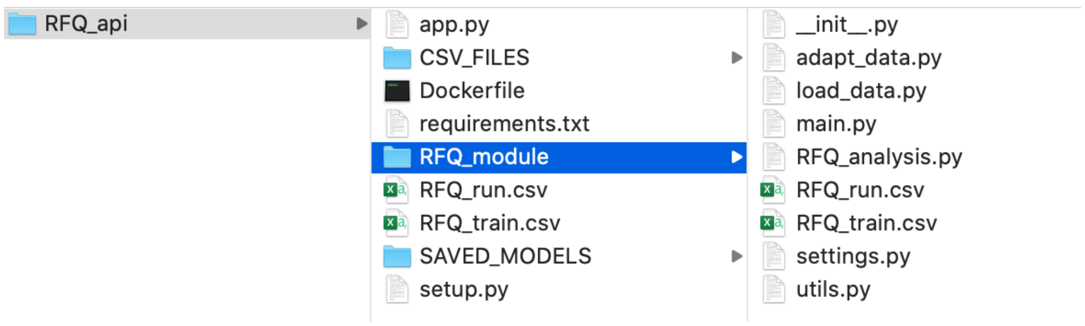
The fact that the concept prioritization tool has been trained and evaluated on quote line data introduces bias when applying the prioritization tool to RFQs. This because an RFQ only becomes a quote when the Customer Support Department deems it has potential to convert into a sale, based on their own mental decision models. It could thus be that the prioritization tool will not perform well on RFQs that would previously not have been selected by the Customer Support Department. However, this bias is unavoidable as it cannot be said for unquoted RFQs whether they would have converted into a sale.

3.2 Concept prioritization tool

In this section a description will be given of the concept prioritization tool, whose code together with a 'translation' can be found in Appendix B.

3.2.1 Front-end concept prioritization tool

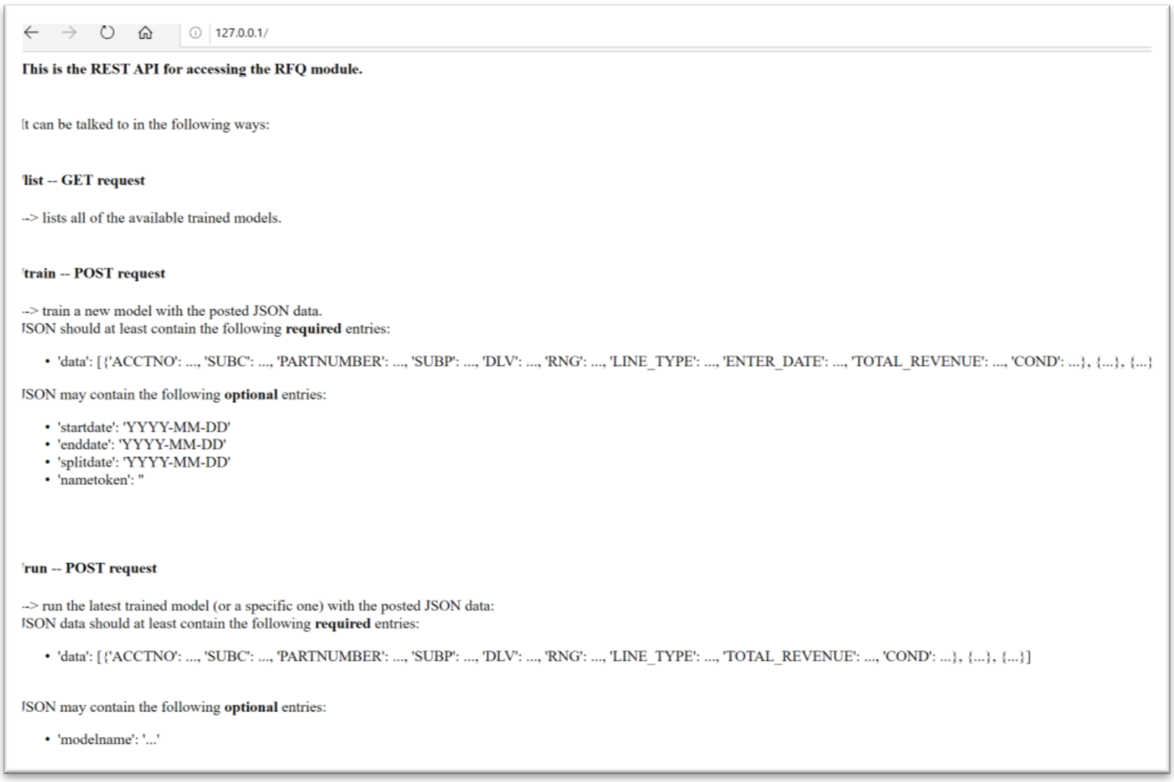
The concept prioritization tool comes in a folder "RFQ_api", which contains the sub-folders/files shown in Figure 21.



```
RFQ_api        ▶   app.py                  __init__.py
                   CSV_FILES         ▶     adapt_data.py
                   Dockerfile              load_data.py
                   requirements.txt        main.py
                   RFQ_module        ▶     RFQ_analysis.py
                   RFQ_run.csv             RFQ_run.csv
                   RFQ_train.csv           RFQ_train.csv
                   SAVED_MODELS      ▶     settings.py
                   setup.py                utils.py
```

Figure 21: Folder contents concept prioritization tool.

The interface of the concept prioritization tool can be activated by opening the command prompt, specifying the path of the file app.py and then typing "python app.py". The command prompt will then return a local host address which, when pasted in an internet browser, brings the user to the interface as seen in Figure 22. It is intended that the user, within this interface, can add /list, /train or /run behind the local host address, press enter and get a list of saved models, train a new model or apply an existing model respectively.



127.0.0.1/

This is the REST API for accessing the RFQ module.

It can be talked to in the following ways:

'list -- GET request

--> lists all of the available trained models.

'train -- POST request

--> train a new model with the posted JSON data.
JSON should at least contain the following **required** entries:

- 'data': [{'ACCTNO': ..., 'SUBC': ..., 'PARTNUMBER': ..., 'SUBP': ..., 'DLV': ..., 'RNG': ..., 'LINE_TYPE': ..., 'ENTER_DATE': ..., 'TOTAL_REVENUE': ..., 'COND': ...}, {...}, {...}

JSON may contain the following **optional** entries:

- 'startdate': 'YYYY-MM-DD'
- 'enddate': 'YYYY-MM-DD'
- 'splitdate': 'YYYY-MM-DD'
- 'nametoken': ''

'run -- POST request

--> run the latest trained model (or a specific one) with the posted JSON data:
JSON data should at least contain the following **required** entries:

- 'data': [{'ACCTNO': ..., 'SUBC': ..., 'PARTNUMBER': ..., 'SUBP': ..., 'DLV': ..., 'RNG': ..., 'LINE_TYPE': ..., 'TOTAL_REVENUE': ..., 'COND': ...}, {...}, {...}]

JSON may contain the following **optional** entries:

- 'modelname': '...'

Figure 22: Interface concept prioritization tool.

3.2.2 Back-end concept prioritization tool

Within the concept prioritization tool, a user can specify whether a new model should be trained, an existing model should be run or a list of existing models should be returned. In addition, the user can specify multiple optional variables, e.g. start date, end date, split date, token to be added to the filename, location of the CSV file used for training the model, which model should be run and whether detailed output should be shown on the screen.

In the sub-sections below, a global description is provided for each of the functionalities of the concept prioritization tool in which variables have been made italic. Moreover, a detailed technical description for each functionality can be found in Appendix C.

3.2.2.1 Train new model

Training a new model takes the input variables *filename_train*, *start_date*, *end_date*, *train_test_splitdate* and *filename* (an optional additional filename token). The function starts by creating data frame df_quote, containing quote data from the file specified in *filename_train* for the period [*start_date*, *end_date*).

Next, arrays are created containing the unique account- and part numbers appearing in df_quote. Then, data frame df_so is created and filled with sales order data retrieved from the Company X SQL server for the period [*start_date*, *end_date*) in which the unique account- and part numbers from df_quote occur.

Afterwards, df_quote and df_so are linked. Here, a distinction is made between quotes/sales orders with account- and part number value combinations occurring only once and multiple times. First, data frames df_quote1 and df_so1 are created containing account- and part number value combinations occurring only once in the quote and sales order data respectively. Then, df_quote1 and df_so1 are merged by means of a left merge after which a boolean mask is applied so that only quote-sales order matches with a difference in days less than 366 or null (no match) remain. Second, data frames df_quote1plus and df_so1plus are created containing account- and part number value combinations occurring multiple times in the quote and sales order data respectively. Here, sales orders are linked to their closest past quote with replacement. Afterwards, a boolean mask is applied so that only quote-sales order matches with a difference in days less than 366 or null remain. Finally, the linked quote-sales order data frames are concatenated, resulting in df_quote_so.

Next, arrays are created containing the unique account- and part numbers occurring in df_quote_so. Using these arrays, master customer and master part number data is retrieved from the Company X SQL server and put in data frames df_mc and df_mpn respectively. These data frames are then (left) merged with df_quote_so, resulting in df_quote_so_mc_mpn.

Then, an additional column is created for df_quote_so_mc_mpn, indicating whether the difference in days between the quote and sales order of a match is less than or equal to variable *TARGETDURATION* (equal to 180). Rows whose difference is less than or equal to variable *TARGETDURATION* get value 1 assigned, rows whose value is more than *TARGETDURATION* or null get value 0 assigned. Note that this is an implicit definition of a sale.

After, df_quote_so_mc_mpn is encoded such that machine learning algorithms can be applied. The encoding depends on the columns type for which a distinction is made between label-, categorial-, binary- and numerical columns. Binary/numerical columns, stored in df_bool and df_num respectively, remain as they originally were, except that missing values are replaced with '-1'. Categorial columns are one hot encoded, meaning that each category of the original feature is replaced by a dummy (variable) column with only 0/1 entries depending on the presence of the category in a row, resulting in df_ohe. Label columns are label encoded, meaning that each category of the original feature is replaced by an integer, resulting in df_label. Missing values for both categorial- and label columns are replaced by 'unknown'. In addition, df_date and df_y are created as subsets of df_quote_so_mc_mpn containing columns QUOTE_DATE_COL and Y_COL respectively. Finally, df_label, df_ohe, df_bool, df_num, df_date and df_y are concatenated column-wise, resulting in df_tot.

Next, df_tot is split into a training- and testing data set. Training data df_train is the subset of df_tot containing all rows whose value for columns QUOTE_DATE_COL is lower than variable *train_test_splitdate.* Further, testing data df_test is the subset of df_tot containing all rows whose value for column QUOTE_DATE_COL is equal to or more than variable *train_test_splitdate*. Then, data frames X_train/X_test are created equal to df_train/df_test respectively without columns QUOTE_DATE_COL and YCOL. In addition, data frames Y_train/Y_test are created equal to column YCOL in df_train/df_test respectively.

Afterwards, variable *clf* is created equal to classifier 'Gradient Boosting Classifier', which generates weak classification models sequentially, where each model focuses on correcting the errors from its predecessor and the final output is a combination of the results from all models. Next, Gradient Boosting Classifier is fit to X_train and Y_train. Then, arrays y_train_pred and y_test_pred are created, containing the predictions of classifier *clf* on data frames X_train and X_test respectively. Next, accuracy and F1 score of the classifier are stored in variable *test_score* and *f1_score_test* respectively.

Finally, variable *xcols*, created as an array containing the column names of the X_train data frame, and the classifier are saved.

3.2.2.2 Run new model

Running an existing model takes as input variables *run_data* and *filename.* It starts by creating data frame df_quote by loading (quote) data from the file specified in *run_data* (by default RFQ_run in *CSV_FOLDER*).

After, arrays are created containing the unique account- and part numbers occurring in df_quote. Next, df_mc and df_mpn are created and filled with master customer and master part number data respectively in which the account- and part numbers from df_quote occur, retrieved from the Company X SQL server. Then, df_mc and df_mpn are sequentially (left) merged with df_quote, resulting in df_quote_mc_mpn.

Further, variables *clf* and *dict_encoders* are created corresponding to the last saved file in the directory behind *MODEL_FOLDER*. Here, *xcols*, *label_encoders* and *ohe_encoders* are created as subsets of *dict_encoders*.

Afterwards, df_quote_mc_mpn is encoded. Binary/numerical columns, stored in df_bool and df_num respectively, remain as they originally were, except that missing values are replaced with '-1'. Categorial columns are one hot encoded, meaning that each category of the original feature is replaced by a dummy (feature) column with only 0/1 entries depending on the presence of the category in a row, resulting in df_ohe. Label columns are label encoded, meaning that each category of the original feature is replaced by an integer, resulting in df_label. Missing values for both categorial- and label columns are replaced by 'unknown'.

Then, data frame df_x is created by concatenating the data frames df_ohe, df_labels, df_bool and df_num column wise (along the x-axis). After, all columns in *xcols* (the columns the model was trained on) but not in df_x are added to df_x and filled with value -1. Next, df_x is set equal to a subset of its former self containing the columns in *xcols* (df_x thus contains its own rows but the columns of xcols).

Last, the trained classifier *clf* is applied to df_x, resulting in array y_pred containing the sales potential predictions.

3.2.2.3 List saved models

The function which returns a list of existing models creates an array containing the items from the directory stored in variable *MODEL_FOLDER* that contain '.pickle'.

3.3 Performance concept prioritization tool

Before the performance of the concept prioritization tool could be determined, bugs were to be eliminated and missing functions to be created (Section 4.1.5.1). Afterwards, performance of the concept prioritization tool was evaluated under the following conditions:

1) Subset of the machine learning data set from 2012-01-01 until 2018-01-01 was used for training the concept prioritization tool.
2) Subset of the machine learning data set from 2018-01-01 until 2019-01-01 was used for evaluating the concept prioritization tool.

From this it followed that the performance of the concept prioritization tool seemed to be 98.87% for the F1 score, 100% for the AUC and following from the confusion matrix a total misclassification rate of 0.24% (Figure 23).

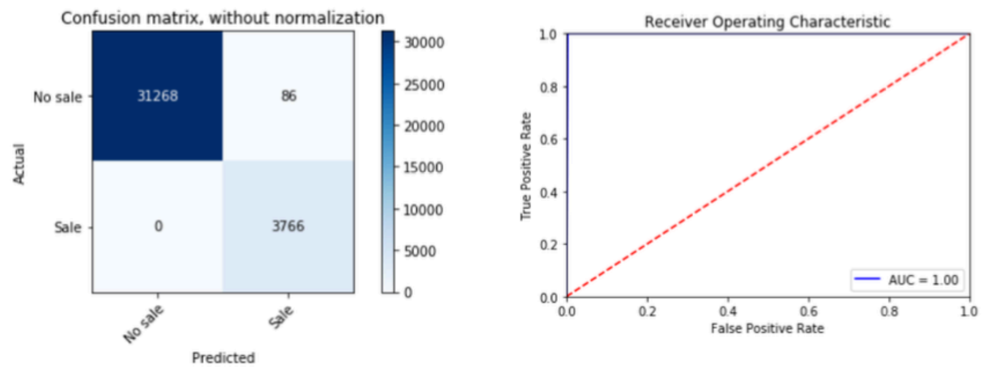


Figure 23: Confusion matrix and ROC curve concept prioritization tool.

However, after analyzing the percentage of missing- and zero values per feature of the machine learning data set (Table 4), it was discovered that the to-be predicted feature class, whether a quote converts into a sale or not, was being leaked, resulting in an overly optimistic performance estimation.

The class distribution of the machine learning data set is 291.322/35.107 for "No sale"/"Sale" respectively, which means that 89.25% ($\frac{35.107}{(35.107+291.322)} * 100\%$) of the quote lines do not convert into a sale. The percentage of missing- and zero values per feature (Table 4) revealed that features revenue and condition have 88.57% and 88.20% missing- and zero values respectively. Since these percentages lie very close to the percentage of quote lines that do not convert into a sale, the question arose whether there was a connection between missing entries for both these features and a quote line not converting into a sale. Therefore, investigation was conducted into the class distribution of the machine learning data subset for which both features revenue and condition did not have missing values. From this followed a class distribution 3401/35.107 for "No sale"/"Sale" respectively. This means that all quote lines from the machine learning data set that convert into a sale are present in the machine learning data subset for which features revenue and condition do not have a missing value; approximately 12% of the entire machine learning data set.

The latter represents a serious data leak since it causes the concept prioritization tool to "predict" solely on the presence of both features revenue and condition rather than the intrinsic information of the machine learning data set, resulting in a (unjust) near perfect performance. After all, approximately 88% of the machine learning data can be correctly classified “No sale” solely on missingness for features revenue and condition, whereas for the remaining approximate 12% machine learning data there is a 91.17% chance ($\frac{35.107}{(35.107+3401)}$) of an instance being “Sale”. After fixing the data leak, through exclusion of features revenue and condition, the true performance estimation of the concept prioritization tool, a significant decrease, appeared to be an F1 score of 6.8%, AUC of 0.52 and a total misclassification rate of 10.55% (96% misclassification in class “Sale”).

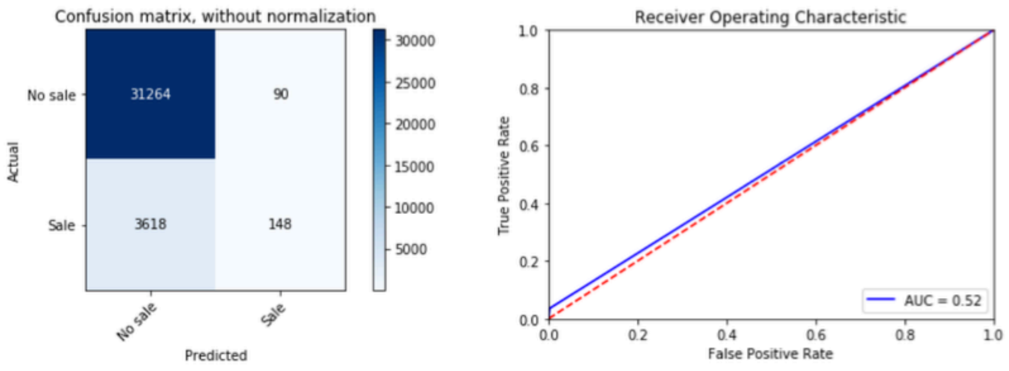


Figure 24: Confusion matrix and ROC curve actual performance concept prioritization tool.

3.4 Conclusion

In this chapter research sub-questions 2.1-2.2 were answered. First, RQ 2.1 *"How is the concept prioritization tool build?"* was addressed in Sections 3.1 and 3.2, in which the machine learning data, front end- and back-end of the concept prioritization tool were described. From this it followed that the concept prioritization tool derives the machine learning data set, the historical labelled quote line data used for training- and evaluating the prioritization tool, in preprocessing steps (Section 3.2) from multiple data sources. The machine learning data set, in case of the concept prioritization tool, consists of 20 features which can be seen in Table 4 and are used to train- and evaluate Gradient Boosting Classifier.

Next, RQ 2.2 *"How is the performance of the concept prioritization tool?"* was addressed in Section 3.3. Here, it was shown that the performance of the concept prioritization tool is significantly lower than initially thought of, due to information leakage of the to-be predicted feature class. The performance of the concept prioritization tool, after fixing the data leak, decreased significantly to an F1 score of 6.8%, AUC of 0.52 and a total misclassification rate of 96% in class "Sale". For the period 2018-01-01 to 2019-01-01, the concept prioritization tool predicted only 148 quote lines correctly to result in a sale out of the 3766 quote lines that actually did. Further, for this period the concept prioritization tool predicted only 238 quote lines to result in a sale from which 148 correctly so. This means that the concept prioritization tool, if it were to be implemented in practice, would only be able to identify a fraction of sales and corresponding revenue/profit. It should be noted that this is only an indication of the concept prioritization tool performance as many more errors, other than the identified data leak, are present which are addressed in chapter 4.

4. Verification/validation of the concept prioritization tool

In the previous chapter the concept prioritization tool was described/analyzed. In this chapter, changes are proposed with regard to verification/validation of the concept prioritization tool, resulting in the *prototype prioritization tool*. Section 4.1 contains a description of- and arguments for the proposed changes. Next, in Section 4.2 the performance of the prototype prioritization tool is evaluated.

4.1 Verification/validation changes

This section contains the proposed changes with regard to verification/validation of the concept prioritization tool and is structured according to the supervised machine learning framework shown in Figure 4. Changes in Python code corresponding to the proposed changes can be found in Appendix D.

4.1.1 Data Acquisition

In Section 3.3 it was revealed that information about the to-be predicted class (feature target) was being leaked to the classifier behind the concept prioritization tool by features revenue and condition. Both these features always had an entry when a quote line converted into a sales order and rarely when a quote line did not; when these features had an entry there was a 91.17% probability of the quote line converting into a sale.

In addition, it was shown in Table 4, that the machine learning data set contained high percentages of missing- and zero values for certain features which should not be possible in reality. Section 3.1 explained that a quote is an offer by Company X for a customer request (RFQ) containing price, lead-time, etc. and becomes a sales order if the customer accepts the offer. Therefore, for certain features (e.g. revenue/condition) it should not be possible in reality and thus in the quote line- and sales order data to have missing entries, giving rise to the question whether the data is representative. Therefore, in this section, the data sources used by the concept prioritization tool are examined one-by-one.

First, the concept prioritization tool acquires quote line data from CSV file RFQ_train, which can be found in the concept prioritization tool folder. The data from this file is described below in Table 5.

Quote Line Data		
Features	**Type**	**% Missing values**
Account number	Categorial	0%
Part number	Categorial	0%
Sub account number	Categorial	0%
Sub part number	Categorial	0%
Delivery window width	Numerical	56.7%
Delivery window time unit	Categorial	0%
Sales type	Categorial	0%
Quote date	Date	0%
Revenue	Numerical	90.8%
Condition	Categorial	90.5%

Table 5: CSV file "RFQ_train".

Sales Order schema		
Features	**Type**	**% Missing values**
Account number	Categorial	0%
Part number	Categorial	0%
Sales order date	Date	0%

Table 6: SQL server Sales_Order schema.

Master Customer schema		
Features	**Type**	**% Missing values**
Account number	Categorial	0%
Sub account number	Categorial	0%
Country	Categorial	0%
Sales manager	Categorial	0.1%
Customer type	Categorial	8.4%
Account type	Categorial	5.3%
Account rate	N/A	100%

Table 7: SQL server Master Customer schema.

Master Part Number schema		
Features	**Type**	**% Missing values**
Part number	Categorial	0%
Sub part number	Categorial	0%
Stock type	Categorial	0%
Airplane layout	Categorial	43.3%
Repair capability	Categorial	96.8%
Rotable part	Categorial	0%
Standard part	Categorial	0%
Main supplier list price	Numerical	0%

Table 8: SQL server Master Part Number schema.

The other data sources, the sales order data and the master customer- and the master part number schema (Table 6 - Table 8), are accessed by the concept prioritization tool on the Company X SQL server. Below tables are provided describing the sales order-, master customer- and master part number schema.

From the data sources the following stands out:

- In the quote line data source, CSV file RFQ_train, features delivery window width, revenue and condition have 56.7%, 90.8% and 90.5% missing values respectively which should not be possible due to the nature of a quote.
- In the customer data source (Master Customer schema) feature account rate has 100% missing data and will therefore not be included in the machine learning data set of the prototype prioritization tool.
- In the part number data source (Master Part Number schema) features airplane layout and repair capability stand out with 43.3% and 96.8% missing values respectively.

Based on analysis of the data sources above, it has been decided to discard RFQ_train as quote line data source and instead use a newly obtained (valid) quote line data set from the Company X SQL Server. In addition, the following restrictions are imposed on the new quote line data:

- Only sales types 1, 14 and 17 are considered, where 1=Direct sale and 14&17=Exchange. The quote line data has been limited to these sales types since these follow the same RFQ process, which can be seen in Figure 20. The concept prioritization tool included additional sales types which follow different processes, e.g. for sales type 4 a sales order is issued before a quote. Inclusion of different sales types (RFQ processes) will distort both the representativeness of the machine learning data set (quote-sales order matching only considers quotes created before sales orders) and the inferences obtained by the machine learning model (feature informativeness might differ for the different RFQ processes).
- Account numbers starting with [redacted] and account numbers CL002 and AO002 are excluded. These accounts are excluded because they follow a different RFQ process than seen in Figure 20; these accounts can order according to agreed-upon pricelists and do not require a quote.
- Quotes with status “no bid” are excluded because these were not sent to the customer and hence no corresponding sales orders exist.

Renewed Quote Data		
Features	**Type**	**% MV**
Account number	Categorial	0%
Part number	Categorial	0%
Sub account number	Categorial	0%
Sub part number	Categorial	0%
Delivery window width	Numerical	6.9%
Delivery window time unit	Categorial	0%
Sales type	Categorial	0%
Quote line date	Date	0%
Revenue	Numerical	0%
Condition	Categorial	0.1%

Table 9: Quote data Company X SQL Server.

Renewed Sales Order Schema		
Features	**Type**	**% MV**
Account number	Categorial	0%
Part number	Categorial	0%
Sub account number	Categorial	0%
Sub part number	Categorial	0%
Sales type	Categorial	0%
Sales order date	Date	0%
Sales order revenue	Numerical	0%

Table 10: Sales order data Company X SQL server.

The sales order data has also been "renewed" with the quote line restrictions above to make sure that it is not possible for sales orders outside these restrictions to match the renewed quote line data. In addition, sales orders with value zero for both features quantity ordered and sales order revenue are excluded as these were sales orders were cancelled and therefore should not be linked to a quote line.

Further, features sub account number, sub part number, sales type and sales order revenue have been added to the sales order data source to decrease mismatching probability and enable ABC classification, upon which will be further elaborated in Section 4.1.2.

4.1.2 Data Cleaning

4.1.2.1 Data source linking

In Section 3.2.2.1 the code behind training a new model in the concept prioritization tool was described, including the preprocessing steps that resulted in the machine learning data set. In Section 4.1.1 the different data sources, which contributed to the creation of the machine learning data set, were described. These data sources are, in the basis, linked together as shown in Figure 25.

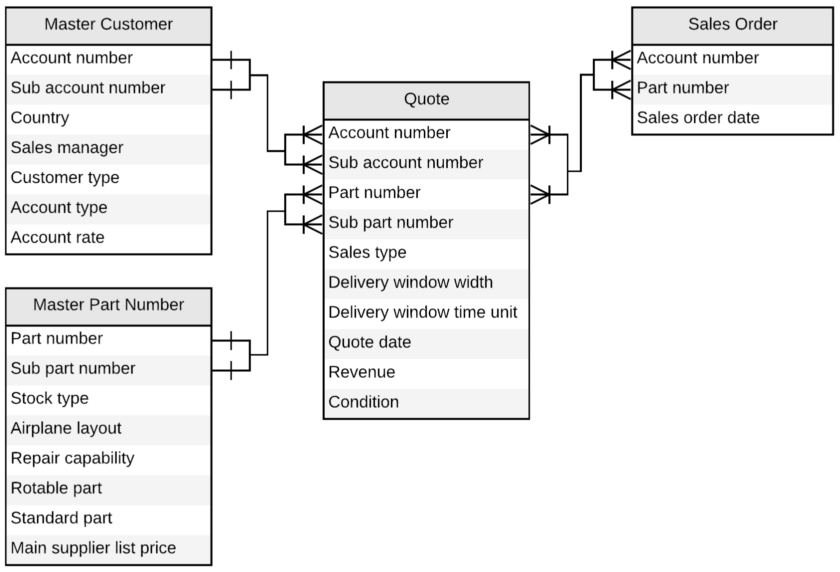


Figure 25: Data source linking concept prioritization tool.

Linking customer- and part number data to quotes
From Figure 25 it can be seen that both the Master Customer and Master Part Number schema are linked to the quote lines with relation one to one-or-more. This relationship is possible because account number combined with sub account number and part number combined with sub part number represent a unique customer and part number respectively and will therefore present no duplicate issues.

Yet, this data source link is only valid when customers and part numbers are recognized in the Master Customer and Master Part Number schema respectively. The Master Customer- and Master Part Number schema contain information about customers and parts respectively that have been sold at least once in the past. Therefore, when a quote (line) in the machine learning data set has missing values for all features of the Master Customer and/or Master Part Number data source, it means by definition of the Master Customer- and Master Part Number schema, that the quote (line) did not convert into a sales order. Missing values for all features of the Master Customer and/or Master Part Number data source thus leak the to-be predicted feature class.

It has been decided to discard data of quote lines whose customer- and/or part data is not recognized in the Master Customer- and Master Part Number schema, rather than to impute the missing values. The reason being that even a sophisticated imputation method, modeling each feature with missing values as a function of other features, would mean that the missing customer/part data is imputed based on data which contains very little information about the to-be imputed customer/part. Further, the true inferences of the machine learning model for the features in the Master Customer- and/or Master Part Number schema would get distorted since imputation induces uncertainty.

Linking sales orders to quotes
Next, Figure 25 shows that in the concept prioritization tool, the sales order- and quote line data sources are linked on features account number and part number with relation one-or-many to one-or-many. This relation is in line with reality because a quote may be ordered multiple times by the customer within a certain time frame in which the quote is (still) valid. Hence, a quote line can be linked to 0, 1 or 1+ sales order(s) although rare cases exist in which a sales order is created without a quote line. Yet, the problem with linking quote lines to sales orders is that there exists no unique identifier which a quote line and sales order have in common.

In the concept prioritization tool, when linking quote lines to sales orders, a distinction is made between account number- and part number feature value combinations occurring once and account number- and part number feature value combinations occurring multiple times:

- Quote lines with account number- and part number feature value combinations occurring once are linked to sales orders with account number- and part number feature value combinations occurring once by means of a left merge. This link assumes that for every account number- and part number feature value combination in the quote lines either 0 or 1 sales order exist. Yet, it could be possible that there are multiple sales orders for an account number- and part number feature value combination occurring once in the quote lines, which is currently not considered for matching.

- Quotes with account number- and part number feature value combinations that occur multiple times in the quote lines are linked to account number- and part number feature value combinations that occur multiple times in the sales orders. For each account number- and part number feature value combination, a quote date- and sales order date list is created. After, the matching quote line is calculated for each sales order, being the quote line whose quote date is closest and prior to the sales order date. This linking method allows for 0, 1 or 1+ sales orders to be linked to a single quote line, which corresponds to reality, but assumes that for every account number- and part number feature value combination occurring multiple times in the quote line data, multiple sales orders exist, while this can also be 1.

The following changes are proposed with regard to linking the quote lines and sales orders:

1) The sales order- and quote line data sources will be linked on features account number, part number, sub account number, sub part number and sales type in the prototype prioritization tool. Sub account number and sub part number are included because these, combined with account number and part number, represent a unique customer and part number respectively. In addition, increasing the number of features that quote lines and sales orders have to match on decreases the mismatching probability, hence inclusion of sales type.
2) The quote line subset with account number-, part number-, sub account number-, sub part number- and sales type feature value combinations occurring only once are linked in the prototype prioritization tool to a subset of the sales order data in which these exact account number-, part number-, sub account number-, sub part number- and sales type feature value combinations occur. This allows 0, 1 or 1+ sales orders to be linked to a quote line with an account number-, part number-, sub account number-, sub part number- and sales type feature value combination occurring only once.
3) The quote line subset with account number-, part number- sub account number-, sub part number- and sales type feature value combinations occurring multiple times are linked in the prototype prioritization tool to a subset of the sales order data in which these exact account number-, part number-, sub account number-, sub part number- and sales type feature value combinations occur. After, the matching quote line is calculated for each sales order, (still) being the quote line whose quote date is closest and prior to the sales order date. Therefore, in the prototype prioritization tool quote lines with account number-, part number- sub account number-, sub part number- and sales type feature value combinations occurring multiple times can be linked to 0,1 or 1+ sales orders.

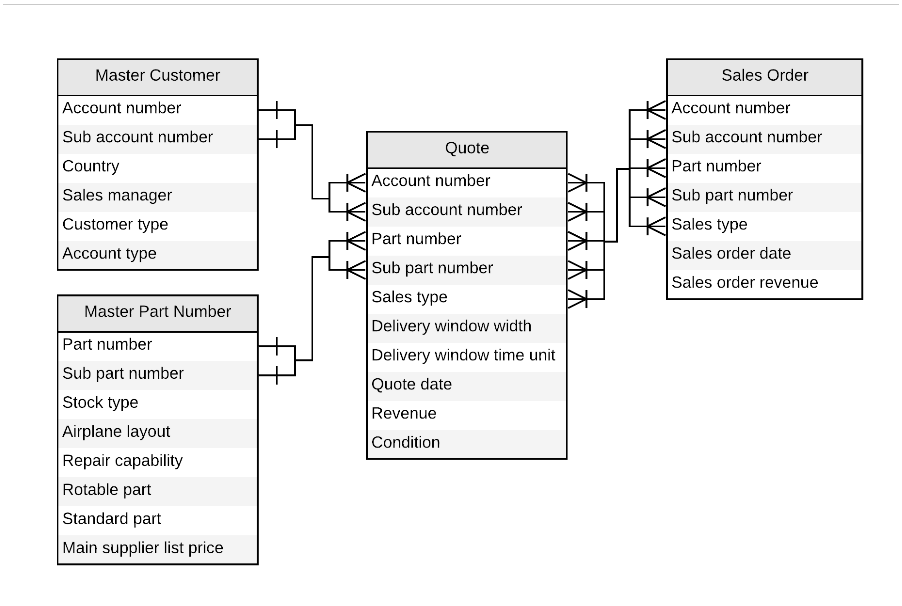


Figure 26: Data source linking prototype prioritization tool.

Definition of sale

As mentioned in Section 3.1, the machine learning data set consists of quote lines, each representing the requested amount for a specific part by a customer. There, a sale was defined as a line, within a quote, converting into a sales order. This definition translates in the concept prioritization tool to a quote line which is linked to a sales order, using the linking procedure described in Section 3.2.2.1, and has a difference in days between the quote date and sales order date of at maximum 180 days. This thus implies that the period in which a quote (line) is valid on average is 180 days, while this is in reality usually a month.

In the prototype prioritization tool, a sale is defined as a quote line which is linked to a sales order, using the linking procedure as described above, and has a difference in quote- and sales order date within the period in which a quote is still valid. For the period in which a quote is valid a distinction is made between regular customers and governments. For regular customers, the period in which a quote is valid is set equal to 30 days. For governments, which are known to take significant time between a quote and sales order (due to bureaucracy and/or approvals required), the period in which a quote is valid is changed to a year.

4.1.2.2 Merge feature pairs

Currently the features delivery window width and delivery window time unit combined represent the allowed delivery window, which means that the delivery window is weighted twice in classifying instances. Therefore, it is proposed to merge these two features into feature delivery window. This by translating the categories of feature delivery window time unit {Y,M,W,D,H} into a corresponding number of days and multiplying this with the value for feature delivery window width.

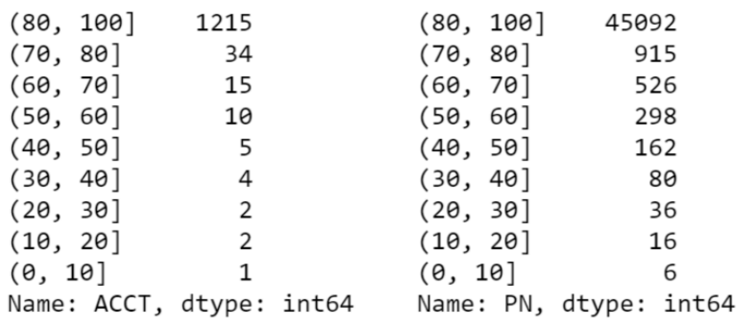
Similarly, account number and sub account number and part number and sub part number are merged so that they represent a unique account number and part number respectively.

4.1.2.3 Handling large number of categorial values

The concept prioritization tool encodes categorial features using either label- or one hot encoder (Section 2.2.2.2). The categorial features account number and part number, which both have a large number of categorial values, are currently label encoded. However, when applying label encoding to a non-binary feature, the machine learning model will have a false sense of order amongst the categories (0<1<2<etc.). In addition, label encoder cannot handle data it has never seen before, thus if a category occurs in the test data which did not occur in the train data, an error will be raised. On the other hand, when applying One Hot Encoding to account number and part number, python raises a memory error due to the large resulting (sparse) matrix.

Thus, for features account number and part number, a sweet spot is sought between the (number of) categories to include, to avoid a memory error, and capturing as much of the information the features entail as possible. Therefore, it is proposed to apply a form of ABC classification to merged features account number and sub account number and part number and sub account number, which represent a unique customer and part respectively.

First, for each customer/part the sum of the sales order revenue of all individual sales orders corresponding to that customer/part was calculated. Then, the total sales order revenue of all sales orders for all customers/parts was calculated and the sales order revenue for a specific customer/part was expressed as a percentage of the total sales order revenue. This list was then sorted (descending) after which the cumulative contribution to the total sales order revenue was calculated for each customer/part. Afterwards, customers/parts were assigned a bin corresponding to their cumulative contribution, which can be seen in Figure 27 together with a count per bin.



```
(80, 100]    1215          (80, 100]    45092
(70, 80]       34          (70, 80]       915
(60, 70]       15          (60, 70]       526
(50, 60]       10          (50, 60]       298
(40, 50]        5          (40, 50]       162
(30, 40]        4          (30, 40]        80
(20, 30]        2          (20, 30]        36
(10, 20]        2          (10, 20]        16
(0, 10]         1          (0, 10]          6
Name: ACCT, dtype: int64   Name: PN, dtype: int64
```

Figure 27: Number of accounts (left) and -parts (right) per corresponding cumulative contribution to the total sales order revenue.

Based on Figure 27, it was decided to not encode the top 40% and 20% customers and parts respectively, contributing the most to the total sales order revenue. The remaining customers/parts were encoded by being assigned a bin corresponding to their cumulative contribution to the total sales order revenue. This reduces the number of categories of features account number merged with sub account number and part number merged with sub part number from 1288 and 47131 to 14 and 29 respectively. Customers/parts which never converted to a sale were assigned bin (80, 100]. Now, One Hot Encoding can be applied to these features without resulting in a memory issue.

4.1.2.4 Handling of missing values

Investigation into feature missingness of the prototype prioritization tool machine learning data set (Table 11) revealed that features repair capability, airplane layout, delivery window, customer type, condition, and account type have missing values (MVs).

Missingness Machine Learning Data			
Feature	**Type**	**Missing values**	**% MV**
Repair capability	Categorial	134769	71.3%
Airplane layout	Categorial	64529	34.1%
Delivery window	Numerical	6628	3.5%
Customer type	Categorial	1328	0.7%
Condition	Categorial	110	0.1%
Account type	Categorial	91	0%
Account	Categorial	0	0%
Part	Categorial	0	0%
Standard part	Categorial	0	0%
Rotable part	Categorial	0	0%
Main supplier list price	Numerical	0	0%
Revenue	Numerical	0	0%
Sales type	Categorial	0	0%
Country	Categorial	0	0%
Sales manager	Categorial	0	0%
Stock type	Categorial	0	0%

Table 11: Missing values machine learning data set.

For features repair capability and airplane layout, missing values are either truly missing or imply that the part cannot be repaired and/or that the part has no airplane layout code (e.g. does not belong to an aircraft). For these features missing values are thus a combination of Missing Not At Random and Missing (Completely) at Random. On one side, Missing Not At Random data, which is the case when data is missing because of its own value, should not be discarded as it would introduce bias into the model. On the other hand, imputing values would also not be wise because missingness can imply two categories; truly missing or non-existing.

Therefore, it has been decided to replace missing values for repair capability and airplane layout with 'unknown' and remove instances with missing values for features delivery window, customer type, account type and condition, which are Missing (Completely) At Random.

4.1.2.5 Dummy feature trap

In the concept prioritization tool perfect multicollinearity is induced after applying One Hot Encoding to categorial features. This perfect multicollinearity introduces a redundant dummy feature for each OHE encoded categorial feature, which takes computational time and power without adding value to the model. This is in the prototype prioritization tool eliminated by dropping the first dummy feature for every OHE encoded feature.

4.1.3 Data Splitting

The concept prioritization tool splits the machine learning data set into a train- and test set using a *split date* variable. Data from [*start date*, *split date*) is used as train data and the data from [*split date*, *end date*] as test data. Thus, both the train- and test data set consists of consecutive instances. However, literature revealed that taking consecutive instances severely impacts the validity of the results as the population of interest is not reflected accurately enough as consecutive instances may contain events such as promotions, seasonality, etc. that distort the weights that the machine learning model assigns to features.

In Section 1.3 it was mentioned that historically only 17% of all quotes convert into a sale, which in the train data set is approximately 19%, confirming that the issue of data imbalance is present. According to literature, when dealing with imbalanced data, a stratified random split should be used. Moreover, literature recommends in general the use of (k-fold) cross validation to prevent overfitting. Therefore, a k-fold stratified split is incorporated in the prototype prioritization tool rather than splitting the machine learning data based on a date.

4.1.4 Model Training & Building

4.1.4.1 Outlier detection

Outliers only exist by definition in non-categorial features (Medium, 2018). The concept prioritization tool machine learning data set, shown in Table 4, contains three numerical features, namely; revenue, delivery window and main supplier list price. Values for these features takes significant ranges due to the nature of the data, e.g. the spare part data includes both bolts and engines.

Box-plots were created for the numerical features of the machine learning data set, shown on the left-side of Figure 28, Figure 29 and Figure 30, which reveal that few instances take extreme values. Investigation of these instances in the ERP system revealed that these extreme values are due to human error; e.g. a quote line exists with value of 800 million for feature revenue but is due to an accidental quantity times 1000. Such human errors should be removed by means of outlier detection before training the machine learning model since they greatly affect normalization.

Therefore, it is proposed to apply the outlier detection method Z-scoring to the training data, which represents the number of standard deviations a feature observation is away from its mean. Numerical feature values with a Z-score of less than -3 or more than 3 are considered to be an outlier. Table 12 below shows per numerical feature the mean, standard deviation, threshold for exclusion and the number of records that are excluded by using this method of outlier detection. Next, the right-side of Figure 28, Figure 29 and Figure 30 show the box-plots of the numerical features after outlier detection for the train data.

Numerical features outliers				
Feature	**Mean**	**Std deviation**	**Excluded from**	**Records excluded**
Revenue	5539.64	49014.40	152582.84 Euro	648
Delivery window	23.65	507.22	1545.31 days	9
Main supplier list price	14527.71	62231.21	201221.34 Euro	2450
Total				**2755**

Table 12: Numerical features outlier detection.

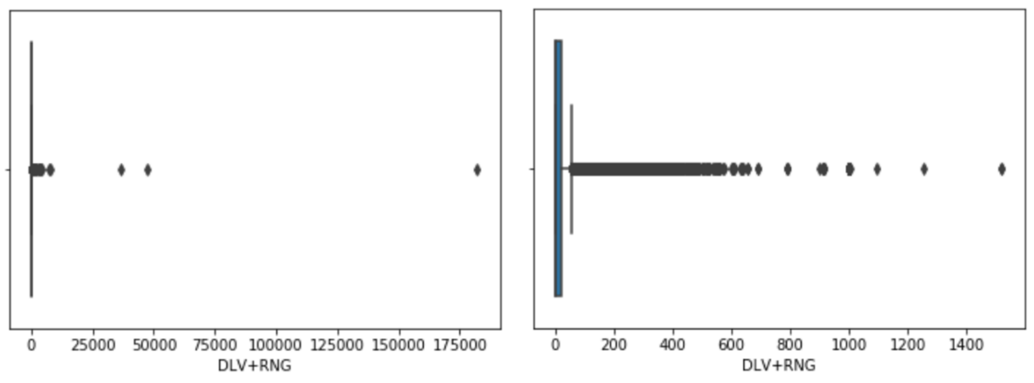


Figure 28: Boxplot feature delivery window before (left) and after (right) outlier detection.

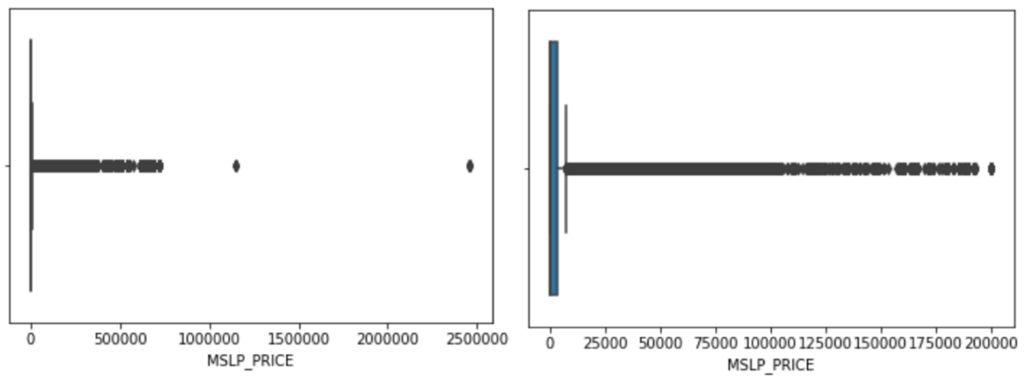


Figure 29: Boxplot feature main supplier list price before (left) and after (right) outlier detection.

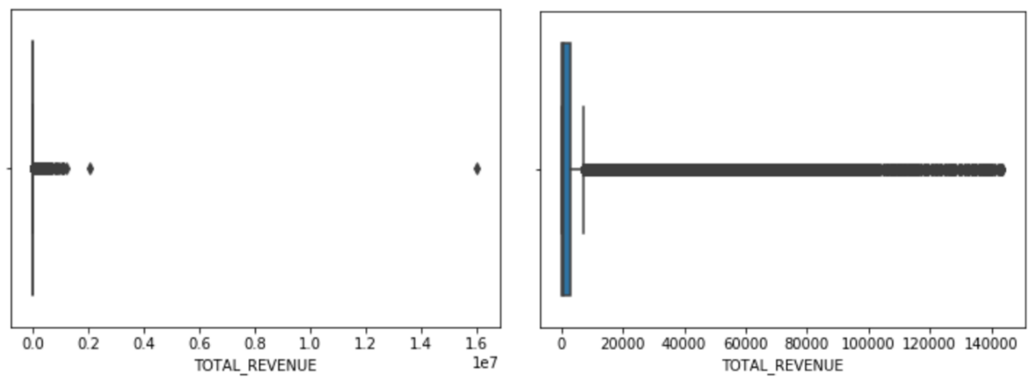


Figure 30:Boxplot feature revenue before (left) and after (right) outlier detection.

4.1.4.2 Under-sampling

From Section 2.2.2.2 it followed that for imbalanced datasets under-sampling can improve classifier performance by increasing the minority class recognition rate. Within under-sampling, a distinction can be made between instance generating- and instance selecting algorithms (Imbalanced-Learn, sd). Further, within instance selecting algorithms, a distinction can be made between controlled- and cleaning algorithms. Controlled algorithms allow the number of samples to be specified by the user while cleaning algorithms do not allow this specification and are meant for cleaning the feature space. An overview of existing under-sampling algorithms is given below (TowardsDataScience, 2018):

Instance generating algorithms:

- **Cluster Centroids**, which applies the k-means method to obtain the class centroids after which it synthesizes instances for each class instead of using the original samples. This method requires data to be grouped in clusters and the number of centroids to be set such that the under sampled clusters are representative of the original data.

Instance selecting algorithms:

- Controlled algorithms:
    - **Random under-sampling**, which under samples the majority class randomly and uniformly. This method can lead to potential loss of information.
    - **NearMiss**, which under samples using heuristic rules based on the nearest neighbor algorithm. This method is sensitive to noise.
- Cleaning algorithms:
    - **Tomek's links**, which detects- and removes Tomek's links. A *Tomek's link* is defined as two instances, x and y, of different class such that for any sample z: d(x,y)<d(x,z) and d(x,y)<d(y,z) holds where d() is the distance between two instances. This thus means that Tomek's links are nearest neighbors of different classes.
    - **Edited Nearest Neighbors**, which applies the nearest-neighbors algorithm to instances of the to-be under sampled class and removes those which do not agree "enough" with the neighborhood (majority or all neighbors).
    - **Condensed Nearest Neighbors**, which uses a 1 nearest-neighbor rule to iteratively decide whether a sample should be removed or not. This method starts by putting all minority instances in set A and all majority instances in set B. Then, one instances from B is added to A after which instances in B are classified using 1 nearest neighbor. Instances which are classified wrong are moved to A and this is repeated until no more instances are moved. This method is sensitive to noise and will by definition add noisy samples.

It is decided to consider instance selecting algorithms only since generation of data, when plenty of data is available, can only lead to a less representative sample. Within instance selecting algorithms, only the controlled under-sampling techniques are considered. This because data preparation has already been carried out and an equal recognition rate for both classes is desired. Here, random under-sampling is preferred over NearMiss because it is less influenced by noise. Oversampling techniques have also been considered but use significantly more computation time while yielding similar performance.

4.1.5 Model Testing

4.1.5.1 Missing function

The concept prioritization tool, when training a new model**,** calls upon a function that supposedly plots a confusion matrix, yet has not been defined. Now, this function has been created and plots a confusion matrix using the predicted- and true labels, which it then saves as .png file. In addition, this function now also plots a ROC curve and displays performance metric AUC in the legend of the graph. Examples of the confusion matrix and ROC curve, created by the newly defined function, can be seen in Figure 23.

4.1.5.2 Noise reduction

In the literature review three types of noise were identified; noisy data items, noisy features and noisy records. Noisy data items were addressed with outlier detection. Next, Noisy features will be addressed in chapter 5 by means of feature selection. Finally, the performance of the prototype prioritization tool, shown in Section 4.2, was ran using 5-fold cross validation, from which the different fold scores (Table 14) did not differ significantly, indicating that noisy records do not influence the results.

4.2 Performance prototype prioritization tool

The performance of the prototype prioritization tool has been evaluated after incorporating all the changes proposed in this chapter. The order in which certain algorithms/preprocessing steps, proposed in the aforementioned changes, are applied in the machine learning model is crucial for its validation/verification. For example, if under-sampling were to be carried out before data splitting, the class distribution of testing data would be 50/50 “No sale”/”Sale” rather than the (approximate) true class distribution 83/17 for “No sale”/”Sale” and therefore no longer accurately reflect reality. In the prototype prioritization tool the order of such algorithms/preprocessing steps, whose order of application could potentially affect verification/validation, is the following: handling missing values, data encoding, splitting data, detecting- and removal of outliers and under-sampling.

The performance of the prototype prioritization tool was evaluated using 5-fold stratified cross validation with (machine learning) data from the period 2012-01-01 to 2019-01-01, where 80%/20% was allocated for training- and evaluation respectively. The prototype prioritization tool machine learning data set consisted of features country, sales manager, customer type, account type, stock type, airplane layout, account, part, repair capability, sales type, condition, standard part, rotable part, main supplier list price, delivery window and revenue. From this follows that the performance of the prototype prioritization tool is 48.6% for the F1 score, 80% for the AUC and, following from the confusion matrix, a total misclassification rate of 31.61%.

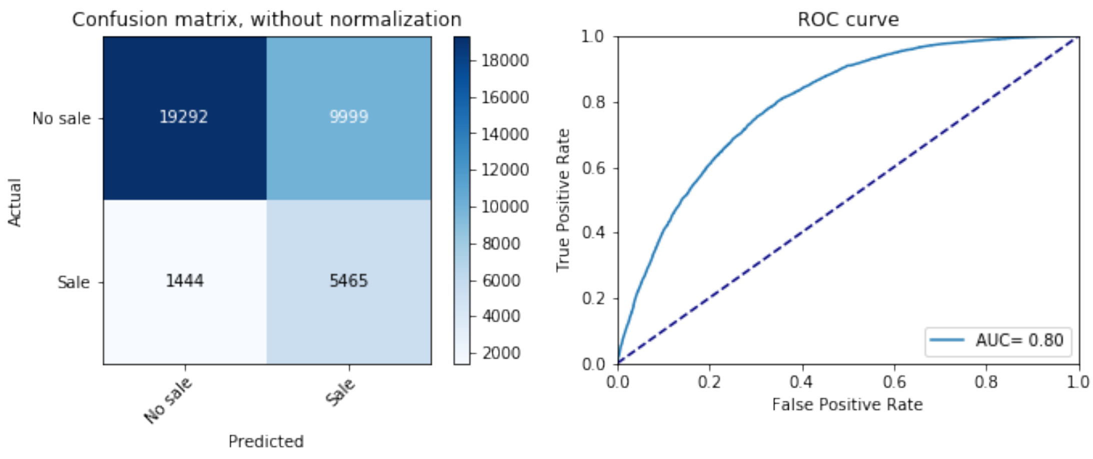


Figure 31: Confusion matrix and ROC curve prototype prioritization tool, including quote features.

	Accuracy	Precision	Recall	F1-score
Fold 1	0.685	0.354	0.785	0.488
Fold 2	0.678	0.348	0.788	0.483
Fold 3	0.683	0.353	0.792	0.488
Fold 4	0.683	0.351	0.775	0.483
Fold 5	0.684	0.353	0.791	0.489
Average	**0.683**	**0.352**	**0.786**	**0.486**
Max Δ fold	**0.007**	**0.006**	**0.016**	**0.006**

Table 13: 5-fold cross validation prototype prioritization tool, including quote features.

However, the objective of the prioritization tool is to prioritize RFQs on sales potential. Yet, its machine learning data set contains features from data source quote; revenue, condition, sales type and delivery window. These features are only known after customer negotiation/consultation (Figure 20) and thus not present in RFQs. Consequently, these features should be excluded from the prototype prioritization tool machine learning data set, resulting in 45.5% for the F1 score, 77% for the AUC and, following from the confusion matrix, a total misclassification rate of 34.51%.

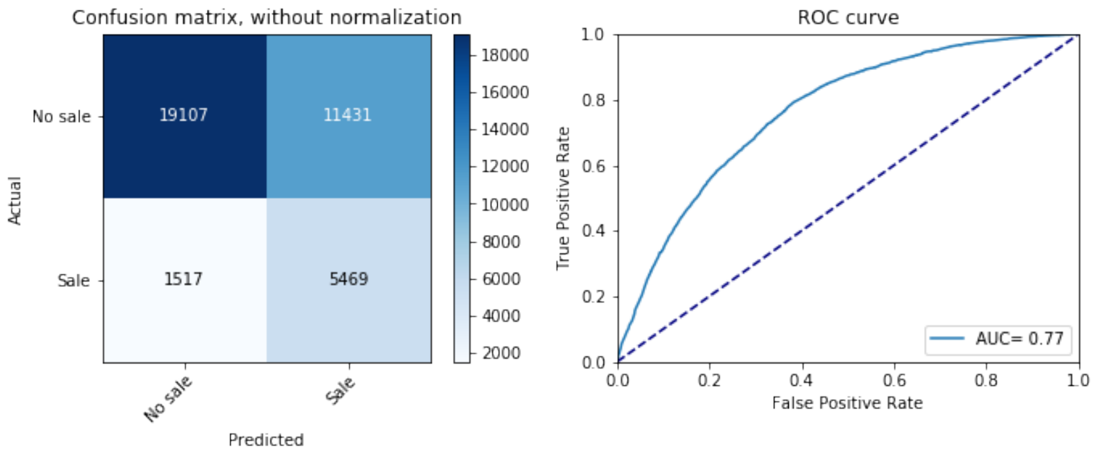


Figure 32: Confusion matrix and ROC curve prototype prioritization tool.

	Accuracy	Precision	Recall	F1-score
Fold 1	0.650	0.318	0.768	0.450
Fold 2	0.655	0.322	0.771	0.454
Fold 3	0.661	0.326	0.774	0.459
Fold 4	0.660	0.323	0.756	0.453
Fold 5	0.655	0.324	0.783	0.458
Average	**0.656**	**0.323**	**0.770**	**0.455**
Max Δ fold	**0.011**	**0.008**	**0.027**	**0.009**

Table 14: 5-fold cross validation prototype prioritization tool.

4.3 Conclusion

In this chapter research sub-questions 3.1-3.2 were answered. First, RQ 3.1 *"What changes in the concept prioritization tool are necessary for the purpose of verification/validation?"* was addressed in Section 4.1. Here, according to the supervised machine learning framework shown in Figure 4, the following changes were proposed; renew quote line- and sales order data, remove customers and parts that do not occur in the master customer- and master part schema respectively (data leak), change quote line-sales order matching and the definition of a sale, merge features to prevent double weighting, use ABC classification for features with large number of categories, remove instances with missing values (MCAR/MAR), eliminate the dummy feature trap, change the data splitting method from date based to stratified k-fold cross validation, incorporate outlier detection and under-sampling and create a function which plots a confusion matrix and ROC curve.

Next, RQ 3.1 *"How is the performance of the prototype prioritization tool?"* was answered in Section 4.2. Here, it was shown that the performance of the prototype was 45.5% for the F1 score, 77% for the AUC and, following from the confusion matrix, a total misclassification rate of 34.51%. Figure 32 revealed that the prototype prioritization tool predicted 5469 quote lines correctly to result in a sale out of the 7013 quote lines that actually did. Further, the prototype prioritization tool predicted 16.900 quote lines to result in a sale from which only 5469 correctly so. This performance is not bad considering approximately 80% of actual sales were identified, yet this through falsely predicting a large number of quote lines to result in a sale. Consequently, an attempt will be made to improve the performance of the prototype prioritization tool in chapter 5.

5. Improving the prototype prioritization tool

This chapter entails an attempt to improve the performance of the prototype prioritization tool by following (the structure of) the B2B sales potential prediction methodology shown in Figure 5, resulting in the *operational prioritization tool*. In Section 5.1 the phases of the aforementioned methodology are carried out; sales opportunity representation, data preparation, machine learning techniques and data insights. Next, in Section 5.2 the performance of the operational prioritization tool is evaluated.

5.1 B2B sales potential prediction methodology

In this section the B2B sales potential prediction methodology by Bohanec et al. (2015) is carried out. Here, improvement focus is on metric F1 score (Section 2.2.3.2) since both wrongly predicted sales (concerning precision), which waste time, and wrongly predicted non-sales (concerning recall), which cause sales to be missed, ought to be minimized. The F1 score metric represents an equally weighted precision-recall trade-off, for which is deliberatly chosen as "optimal" weights are merely subjective and would require additional research to be quantified.

5.1.1 Sales opportunity representation

The B2B sales potential prediction methodology starts with the sales opportunity representation phase. In this phase, available historical (quote line) data is collected that could potentially contribute to predicting sales potential. Rather, since chapter 4 already concluded with a working prototype prioritization tool, in this phase additional features are considered. The additional features have been incorporated in consultation with the product management department, whom are considered to be experts on the RFQ process (Figure 20) and quote line data itself.

Next, literature revealed that meta variables can also capture information which the default features do not, and can be significant predictors. In the context of this assignment, meta features stock, hit rate account, hit rate part, frequency account and frequency part were created and incorporated as they were believed to be potential contributors. An overview of the additional (meta) features can be seen in Table 15, whereas an explanation/description per (meta) feature can be found in Appendix A.

Additional (meta) features machine learning data set

Column name	Feature name	Data type	% Missing values	Data Source			
				Customer	Part	Vendor	Meta
ALT3_CODE_V	Supplier type	Categorial	65.4%			X	
STOCK	Stock	Categorial	3.5%				X
TERM_CODE	Credit period	Categorial	0%	X			
COMPANYNO	Company number	Categorial	0%	X			
EXCH_CORE_RETURN	Exchange return period	Numerical	0%	X			
MFG	Part manufacturer	Categorial	0%		X		
MFG_SUBC	Part manufacturer sub code	Categorial	0%		X		
PIECEPART_SUBASSY	Piece part	Categorial	0%		X		
PN_PROGRAM	Aircraft type	Categorial	0%		X		
AUTO_VENDOR	Default supplier	Categorial	0%		X		
AUTO_VENDOR_SUBC	Default supplier sub code	Categorial	0%		X		
SELL_TYPE	Mark-up	Categorial	0%		X		

AUTO_PRICE_ENABLED	Fixed vendor price	Categorial	0%		X		
ONCONDITION	Obligatory safety check	Categorial	0%		X		
HIT_RATE_ACCT	Hit rate account	Numerical	0%				X
HIT_RATE_PN	Hit rate part	Numerical	0%				X
FREQ_ACCT	Frequency account	Numerical	0%				X
FREQ_PN	Frequency part	Numerical	0%				X

Table 15: Additional (meta) features machine learning data set.

Considering the additional (meta) features, data sources are now linked as shown in Figure 33.

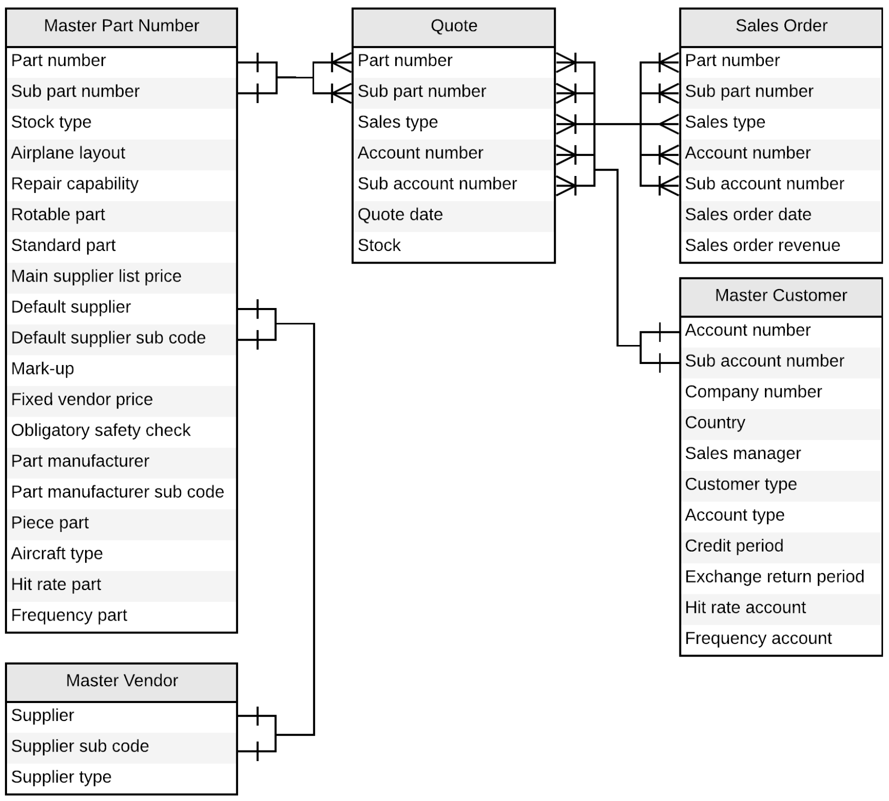


Figure 33: Data source linking machine learning data set.

5.1.2 Data preparation

The second phase of the B2B sales potential prediction methodology is the data preparation phase. This phase has already been carried out- and described in chapter 4. The additional (meta) features resulting from the sales opportunity representation phase are prepared similarly.

5.1.3 Machine learning techniques

The third phase of the B2B sales potential prediction methodology is the machine learning techniques phase. In this phase the feature selection, hyper-parameter tuning and classification threshold optimization are carried out.

5.1.3.1 Feature selection sub-methodology

A major problem for large data sets, in which many potential predictor features are present, is the curse of dimensionality. The *curse of dimensionality* is the consequence of high dimensionality of the input, which increases both the size of the search space in an exponential manner as well as the chance to obtain invalid models. In this sub-section the curse of dimensionality will be addressed via feature selection, which will speed up computation time, improve input data quality, and potentially increase model performance while simultaneously decreasing model complexity. Feature selection is carried out according to the sub-methodology of Bohanec et al. (2015) shown in Section 2.2.3.3, and consists of the following (sequential) steps:

1) Ranking feature importance.
2) Monitoring model performance when incrementally adding top ranked features to detect the optimal cut-off point.
3) Eliminating noise/redundancy using a wrapper method.

It should be noted that literature recommends feature selection to be carried out **only if** the *number of events per variable (EPV)*, which is the smallest of the number of positive/negative cases divided by the number of independent variables, is at least 50 (Heinze & Dunkler, 2016). In the training data, used below for feature selection, the $\text{EPV} = \frac{27437}{28} = 979.89$ which justifies its use.

Feature importance

Feature ranking is carried out using the Orange datamining suite, of which the workflow can be seen in Figure 34. Here, data is loaded into Orange via the File widget. The data used to rank features is the training data, except that now it is not dummy-encoded. The latter because the interest is in the overall most important features rather than the most important categories within features. Test data is not included in feature ranking as it would introduce bias in the performance estimators.

Next, the File widget is connected to the Rank widget, in which multiple filter methods are applied (Figure 56). It should be noted that these filter methods require categorial features and that Orange datamining suite by default applies equal-frequency discretization (4 intervals) to numerical features. Besides standard filter methods from the Rank widget, classifiers Random Forest and Logistic Regression were also used to rank the data by connecting their widgets to the Rank widget.

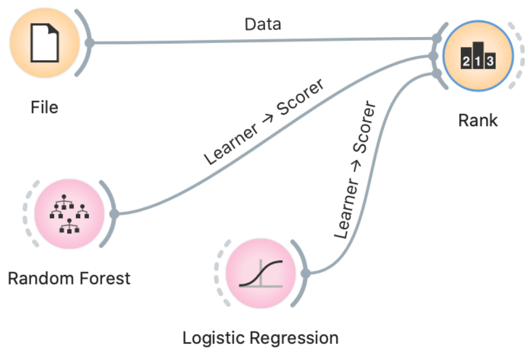


Figure 34: Workflow Orange datamining suite filter methods.

Following Figure 56, shown in Appendix D, the top 15 features were further investigated in the second- and third step of the feature selection sub-methodology.

Monitoring performance to detect optimal cut-off point

Next, the top ranked features, different for each filter method, are added one-by-one from highest- to lowest ranked and F1 score performance of multiple classifiers on the test data was monitored, resulting in Figure 35-Figure 37 (corresponding to Table 22-Table 24 in Appendix D). In this phase filter methods Information gain, Gain ratio, Logistic Regression, Chi-square and Random Forest were considered and classifiers Gradient Boosting Classifier (used in the concept- and prototype prioritization tool), Random Forest and Logistic Regression.

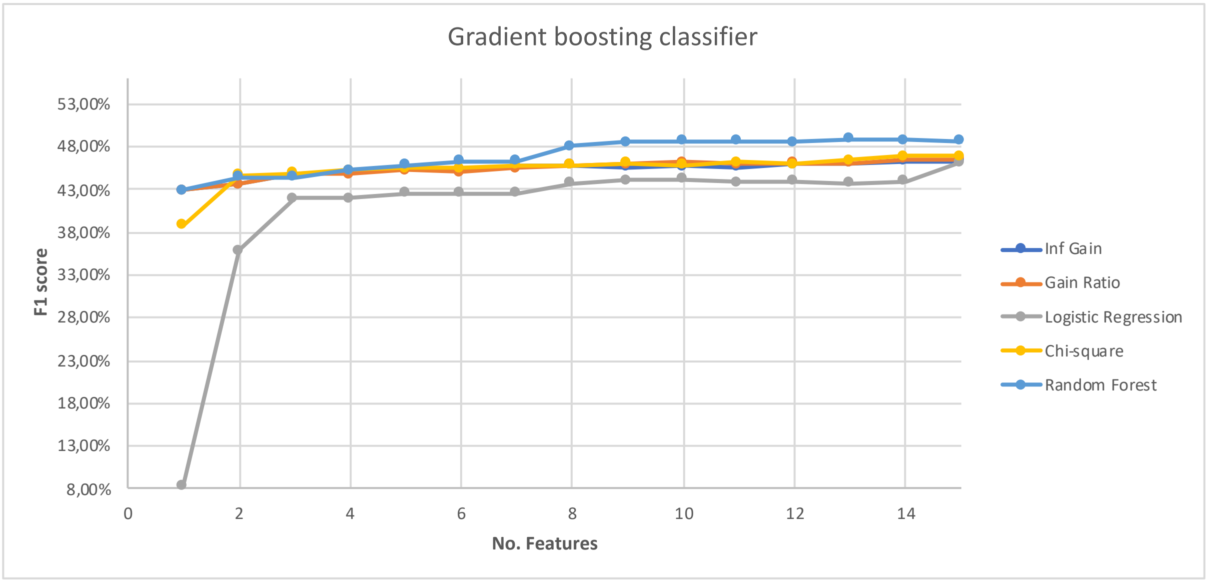


Figure 35: Monitoring F1 score performance Gradient Boosting Classifier using multiple filter methods.

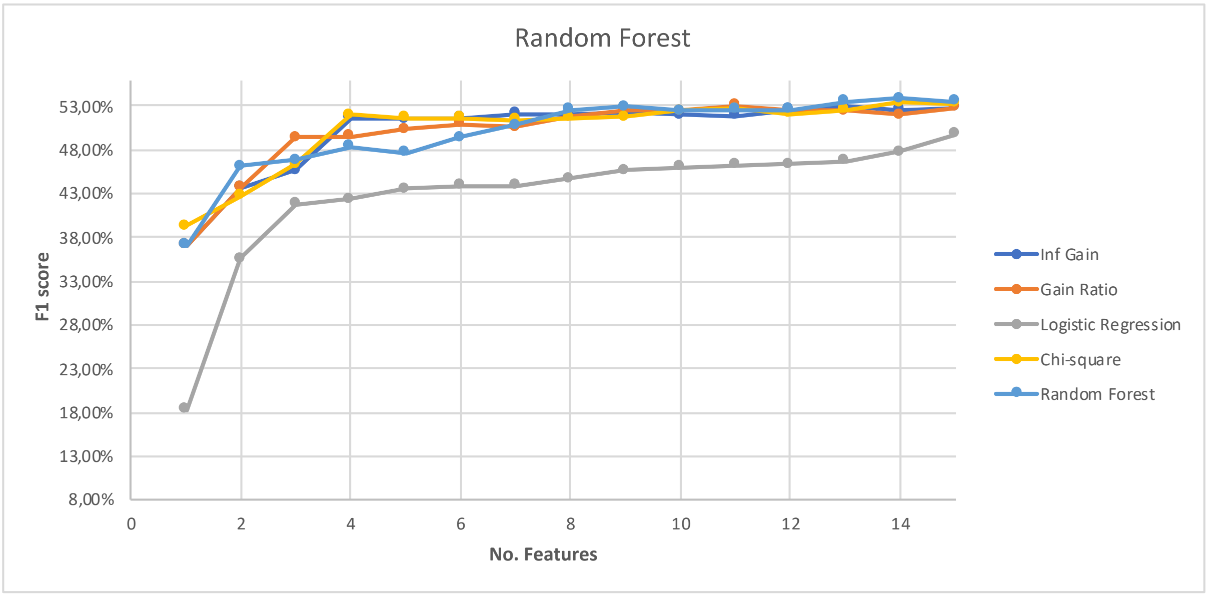


Figure 36: Monitoring F1 score performance Random Forest using multiple filter methods.

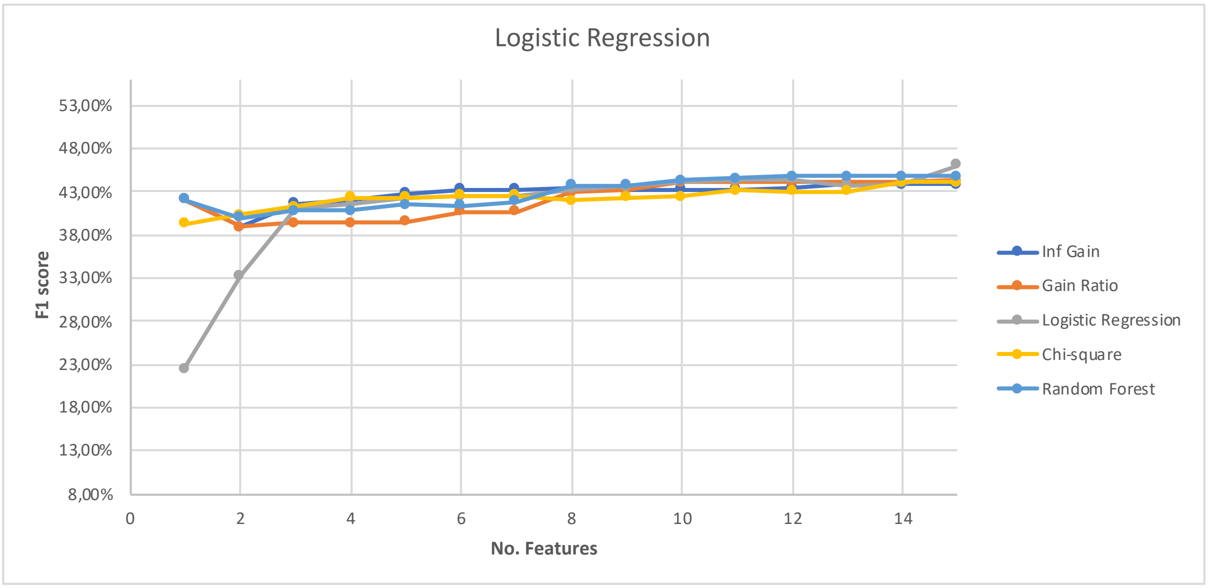


Figure 37: Monitoring F1 score performance Logistic Regression using multiple filter methods.

From these figures, it followed that the optimal cut-off point for each classifier, being the combination of ranking method and the number of top ranked features that yields the highest overall F1-score, were the following:

- For Gradient Boosting Classifier the optimal cut-off point is at the top 13 ranked features using filter method Random Forest with an F1 score of 48.84%.
- For Random Forest the optimal cut-off point is at the top 14 ranked features using filter method Random Forest with an F1 score of 53.82%.
- For Logistic Regression the optimal cut-off point is at the top 15 ranked features using filter method Logistic Regression with an F1 score of 46.05%.

Eliminating noise and redundancy

Figure 35-Figure 37 above showed negative or no difference in F1 score during incremental addition of certain features, which indicates the presence of noisy and/or redundant features. In the final step of the feature selection sub-methodology these noisy and/or redundant features are eliminated using a wrapper method.

For the optimal cut-off point per classifier, features were excluded one-by-one from highest- to lowest ranked and permanently excluded when exclusion increased F1 score performance, resulting in a minimum list of features. The results can be found in Table 25-Table 27 in Appendix D, from which followed:

- For gradient boosting classifier the top 13 features all remain, resulting in an F1-score of 48.84%.
- For Random Forest feature hit rate part is excluded and the remaining 13 features result in an F1-score of 54.26%.
- For Logistic Regression, features aircraft type, sales manager and mark-up are excluded and the remaining 12 features result in an F1-score of 46.69%.

5.1.3.2 Comparing classifier performance

In this section the performance of the best models following the feature selection sub-methodology are compared using a k-fold cross validation approach (Tan, et al., 2006).

Let M_{ij} denote the classification model using classification technique T_i during the j^{th} fold. Let p_{1j} and p_{2j} be the respective performance estimations of the models in the jth fold. The difference between these performance estimations in the j^{th} fold can be written as $d_j = p_{1j} - p_{2j}$. Then, if k is sufficiently large, d_j is normally distributed with mean d_{true}^{CV}, which is the true difference in performance estimation, and standard deviation σ_{CV}. The variance of the observed differences is estimated using $\sigma_{CV}^2 = \frac{\sum_{j=1}^{k}(d_j-\bar{d})^2}{k(k-1)}$, where $\bar{d}$ is the average difference in performance estimations of all folds. Then, a confidence interval can be calculated for the true difference in performance estimation between the two models: $d_{true}^{CV} = \bar{d} \pm \sigma_{CV} * t_{(1-a)(k-1)}.$, where $t_{(1-a)(k-1)}$ can be obtained from the probability table of the t-distribution.

Comparing the 13-feature random forest, 13-feature gradient boosting classifier and 12-feature logistic regression model, Table 16 and Table 17 show that for confidence levels 80-99% zero does not lie in the confidence intervals [Lower bound (LB), Upper bound (UB)] of the mean difference between folds. Therefore, it can be concluded that all three models statistically differ in performance and that the 13-feature Random Forest model performs best.

Fold	Difference per fold
1	5,42%
2	5,58%
3	5,18%
4	5,24%
5	5,57%
Average	**5,40%**
Variance	**0,000068%**

CI	Mean	±	Std*$t_{(1-a)(k-1)}$	LB	UB
99%	0,0540	±	0,00126	0,0527	0,0552
98%	0,0540	±	0,00175	0,0522	0,0557
95%	0,0540	±	0,00229	0,0517	0,0563
90%	0,0540	±	0,00308	0,0509	0,0571
80%	0,0540	±	0,00378	0,0502	0,0578

Table 16: 13-feature Random Forest versus 13-feature Gradient Boosting Classifier.

Fold	Difference per fold
1	2,15%
2	1,64%
3	2,82%
4	2,25%
5	2,03%
Average	**2,18%**
Variance	**0,000364%**

CI	Mean	±	Std*$t_{(1-a)(k-1)}$	LB	UB
99%	0,0218	±	0,00292	0,0188	0,0247
98%	0,0218	±	0,00406	0,0177	0,0258
95%	0,0218	±	0,00530	0,0165	0,0271
90%	0,0218	±	0,00716	0,0146	0,0289
80%	0,0218	±	0,00878	0,0130	0,0305

Table 17: 13-feature Gradient Boosting Classifier versus 12-feature Logistic Regression.

5.1.3.3 Hyper-parameter tuning

As mentioned in Section 2.2.3.4, hyper-parameter tuning is the problem of choosing a set of hyper-parameter values that yields the best generalizable performance improvement. To obtain a generalizable performance improvement an additional (stratified) split, resulting in a validation data set, is required. If hyper-parameter tuning would be conducted using solely a train- and test set, there would be a risk of overfitting since parameters can be tweaked until the model performs optimally on the test set. That way, knowledge about the test set leaks and consequently evaluation metrics do no longer report on general performance (Jordan, 2017). Therefore, the objective of this section is to find the hyper-parameter configuration that yields the best performance on the validation set performance metric (TowardsDataScience, 2018).

Hyper-parameter tuning is carried out for the 13-feature Random Forest model resulting from Section 5.1.3.1, of which the to-be tuned hyper-parameters are shown in Table 18, together with a description and their default value (TowardsDataScience, 2018) (Medium, 2017).

Hyper-parameter	Explanation	Default value
Criterion	Function to measure the quality of a split.	'Gini'
N_estimators	The number of trees in the forest.	100
Max_depth	The maximum depth of the tree.	None
Min_samples_split	The minimum number of samples required to split a node.	2
Min_samples_leaf	The minimum number of samples required to be at a leaf node.	1
Max_features	The number of features to consider when looking for the best split.	$\sqrt{\text{max features}}$
Bootstrap	Whether bootstrap samples are used when building trees.	True

Table 18: Hyper-parameters Random Forest classifier.

It has been decided to carry hyper-parameter tuning out using random search with cross validation. In addition, *validation curves* (Figure 38-Figure 44) are plotted, which show the effect of different values for a single hyper-parameter on classification performance on both the train- and validation data set*, allowing a more promising hyper-parameter grid to be identified (TowardsDataScience, 2019). Moreover, the validation curves reveal that the model overfits as there exists a large gap between train- and validation set performance. Yet, the latter is not a problem since overfitting is prevented by measuring (generalizable) performance on a kept-aside test set combined with k-fold cross validation.

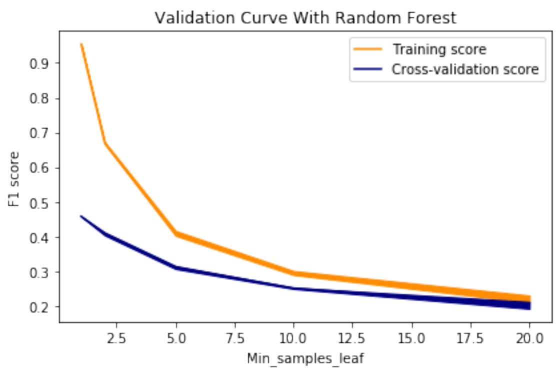


Figure 38: Validation curve min_samples_leaf.

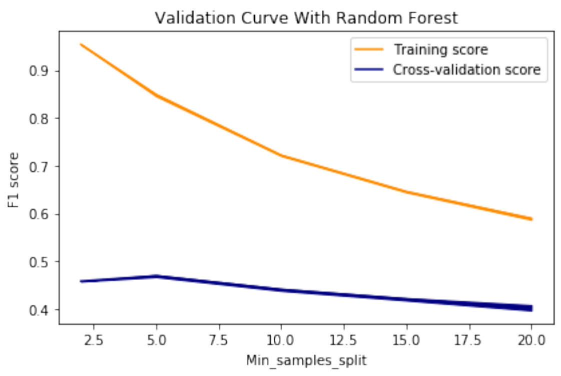


Figure 39: Validation curve min_samples_split.

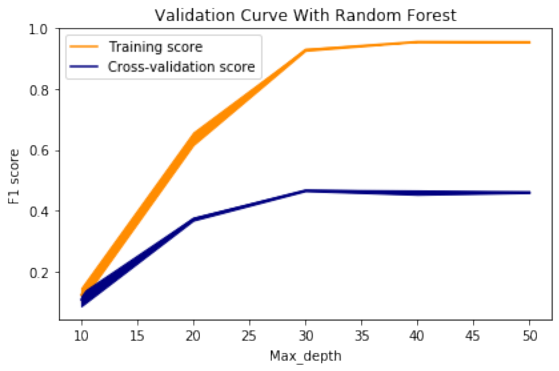


Figure 40: Validation curve max_depth.

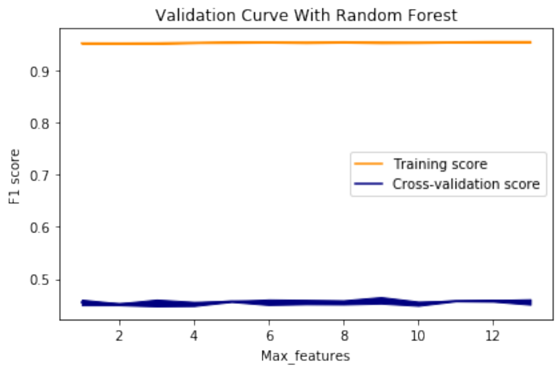


Figure 41: Validation curve max_features.

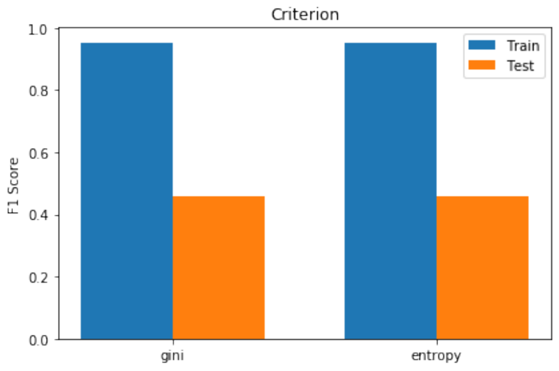


Figure 42: Validation plot criterion.

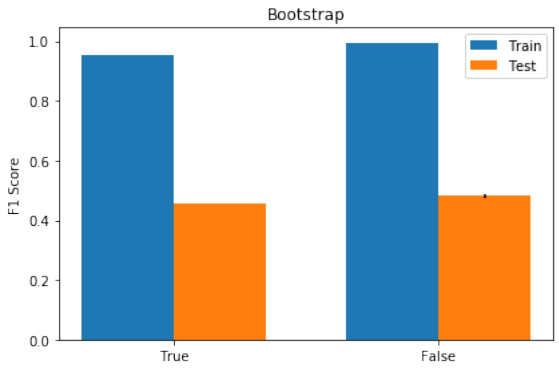


Figure 43: Validation plot bootstrap.

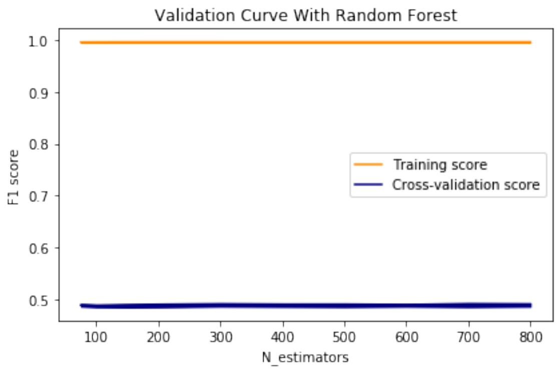


Figure 44: Validation curve n_estimators.

*The maximum F1 score in the validation curves seems to be below the performance shown during feature selection. The reason for this is that under-sampling was temporarily turned off during hyper-parameter tuning as the class distribution of the train-, validation- and test set have to match in order for the improvement in validation set performance, due to hyper-parameter tuning, to be generalizable to that of the test set.

Although the validation curves show promising areas per hyper-parameter, they do not account for *sequential effects*, differences due to the order of hyper-parameter tuning. Therefore, an exploratory random search with cross validation is ran over the entire grid shown in Figure 38-Figure 44. The top 20 configurations resulting from the exploratory random search can be seen in Table 28 (Appendix D) and confirm the promising areas per hyper-parameter seen in the validation curves. Moreover, Figure 45, in which the average F1 score is plotted against the number of iterations, confirms the need of an initial exploratory random search as it shows a steep improvement curve, meaning a lot of “bad” configurations were explored.

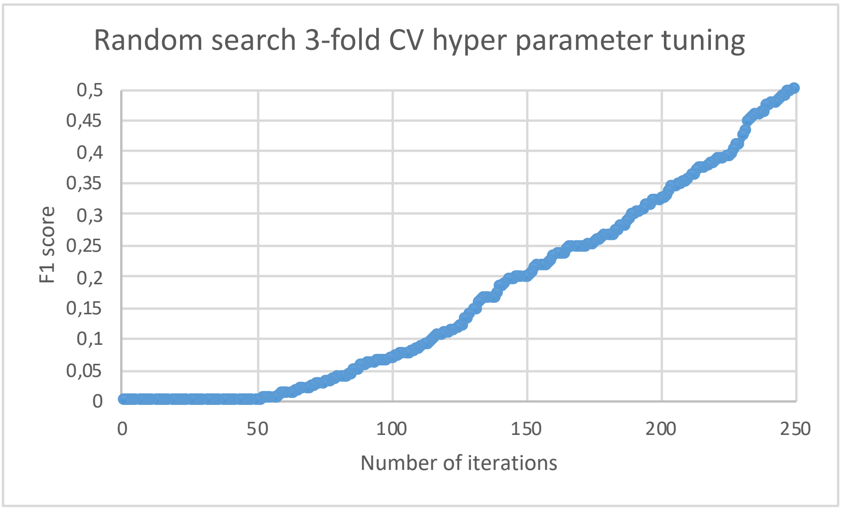


Figure 45: Exploratory random search 3-fold CV for 250 iterations.

Next, using the results of the exploratory search, the grid search space is narrowed down as shown in Table 19. Here, hyper-parameter values were excluded that appeared ≤ 2 in the top 20 configurations of the exploratory search, resulting in a targeted grid.

Hyper-parameter	**Grid values**	**Excluded**
Criterion	['Gini', 'Entropy']	[]
N_estimators	[75, 100, 150, 200, 300, 400, 500]	[600,700,800]
Max_depth	[None, 40, 50]	[10,20,30]
Min_samples_split	[2, 5, 10]	[15,20]
Min_samples_leaf	[1]	[2,5,10,15,20]
Max_features	[4, 5, 6, 7, 8, 9, 10, 11, 12, 13]	[1,2,3]
Bootstrap	[True,False]	[]

Table 19: Targeted grid random search with cross validation (k=3) of 500 configurations.

Running random search with cross validation (k=3) for 500 iterations over the targeted grid resulted in the top 20 configurations as seen in Table 29 in Appendix D. From this followed that configuration {'n_estimators': 400, 'min_samples_split': 2, 'min_samples_leaf': 1, 'max_features': 13, 'max_depth': None, 'criterion': 'gini', 'bootstrap': False} yields the best F1-score performance on the validation set. Moreover, the plot of the average F1 score against the number of iterations of the random search over the targeted grid (Figure 46) shows a flat improvement curve, implying that increasing the number of iterations is not likely to result in a (much) better configuration.

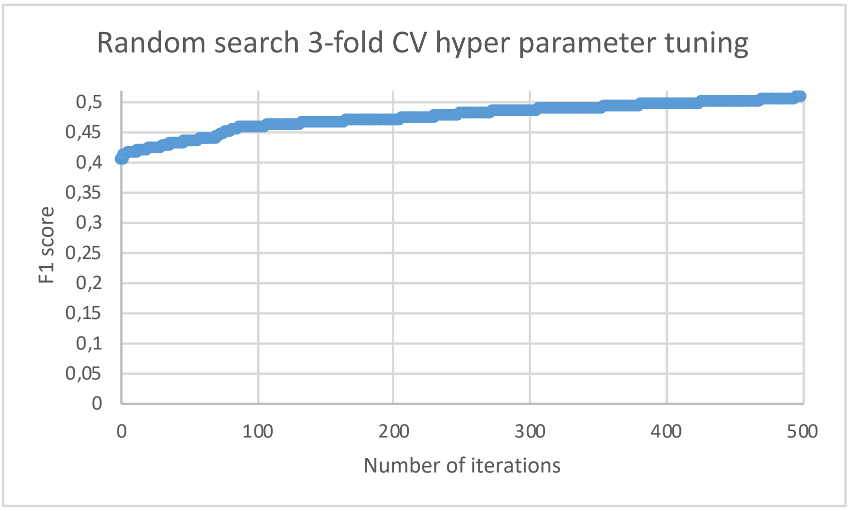


Figure 46: Targeted random search 3-fold CV for 500 iterations.

5.1.3.4 Classification threshold optimization

Classification is based on a probability that the classifier assigns to an observation using the inferences learned from the training data set. By default, this probability is set to 0.5 meaning in this case, that an observation with a predicted probability >0.5 is classified as "Sale" and ≤ 0.5 as "No sale". Yet, this classification threshold can be varied, and therefore optimized. The classification threshold can be perceived as a hyper-parameter and should be treated accordingly although it cannot be set within the classifier itself and is rather manually coded. Below, in Figure 47 the effect of different thresholds on the train- and validation set is shown. From Figure 47 it follows that the validation set F1 score performance is optimal in the area of [0.55,0.65]. Zooming-in on this area (Figure 48), it is shown that the optimal threshold on the validation set is 0.62.

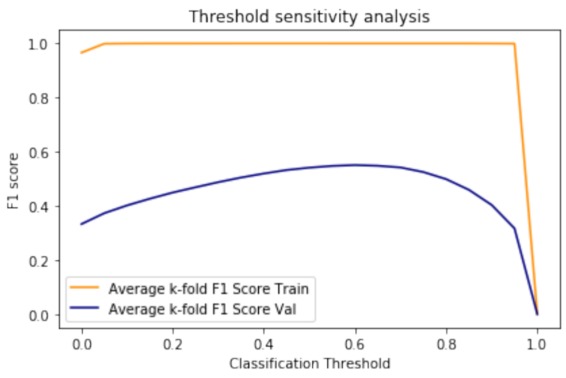


Figure 47: Validation curve classification threshold.

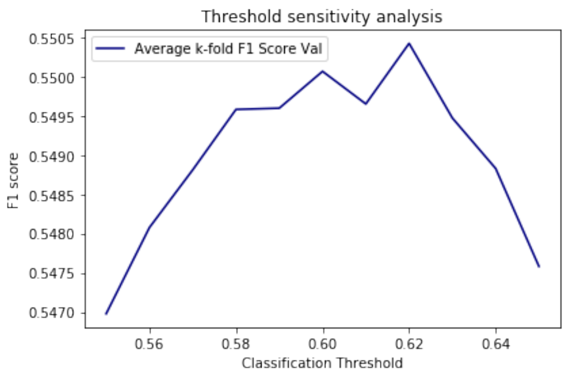


Figure 48: Classification threshold optimization zoom-in.

5.1.4 Data insights

In this section insights are given into the operational prioritization tool and machine learning data set via feature distributions (frequency/probability), association rules and a classification tree visualization. First, Figure 49 shows the number of quote lines that converted to sales orders per (anonymized) sales manager. From this it becomes clear that there are significant differences in sales performance; not only in sales conversion rate but also in frequency. From this stands out that quotes from sales manager 3, 5, 7, 8, 9, 11, 14, 15, 16 and 17 are more likely to miss, whereas sales manager 2 and 13 are best performers in both frequency and conversion rate.

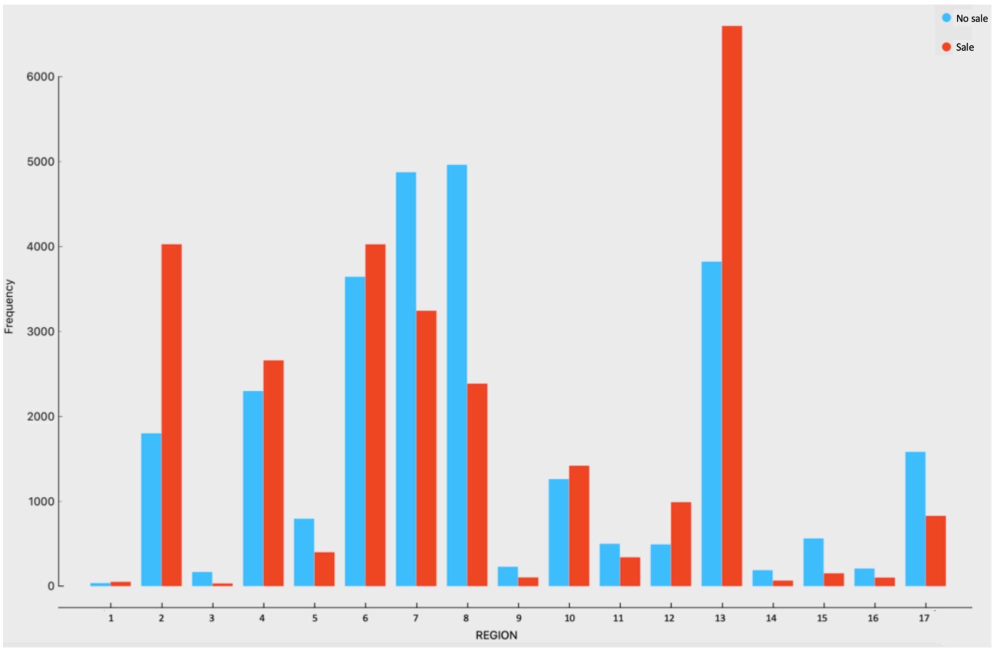


Figure 49: Frequency (not) sold per sales manager.

Next, Figure 50 shows the probability of a quote line converting into a sale per account (ABC classified account number). From this can be seen that the lower the bin range, the higher the probability of sale, as is expected. Moreover, we see that although GL0021 is one of the top contributors, it has low probability of sale, indicating that this customer has bought expensive parts and/or frequently requests quotations.

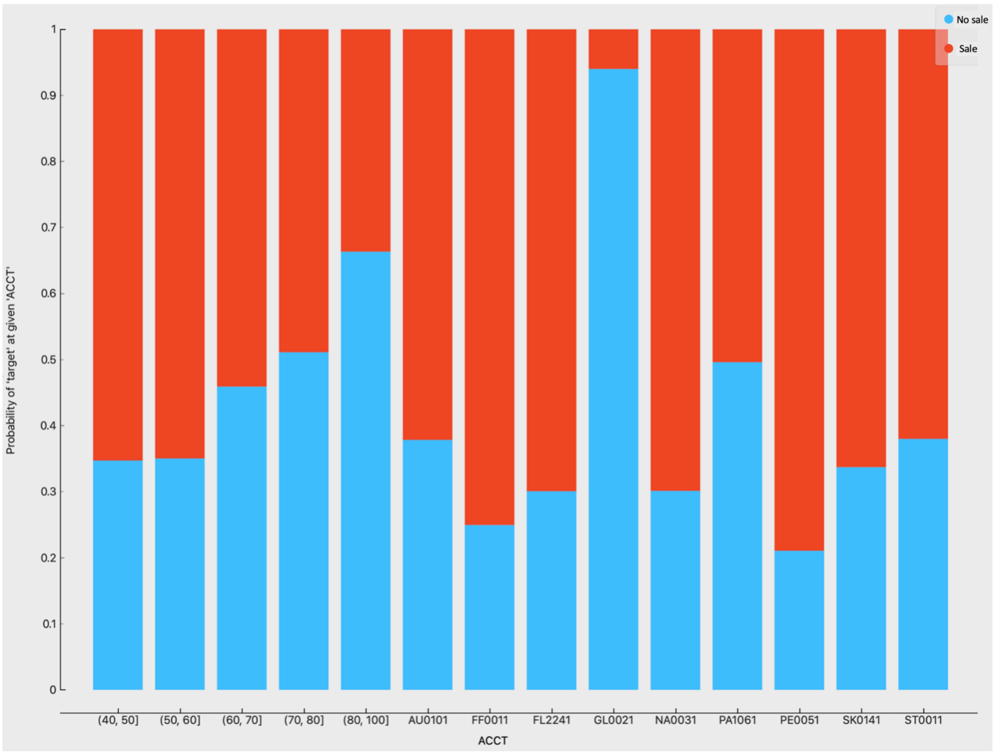
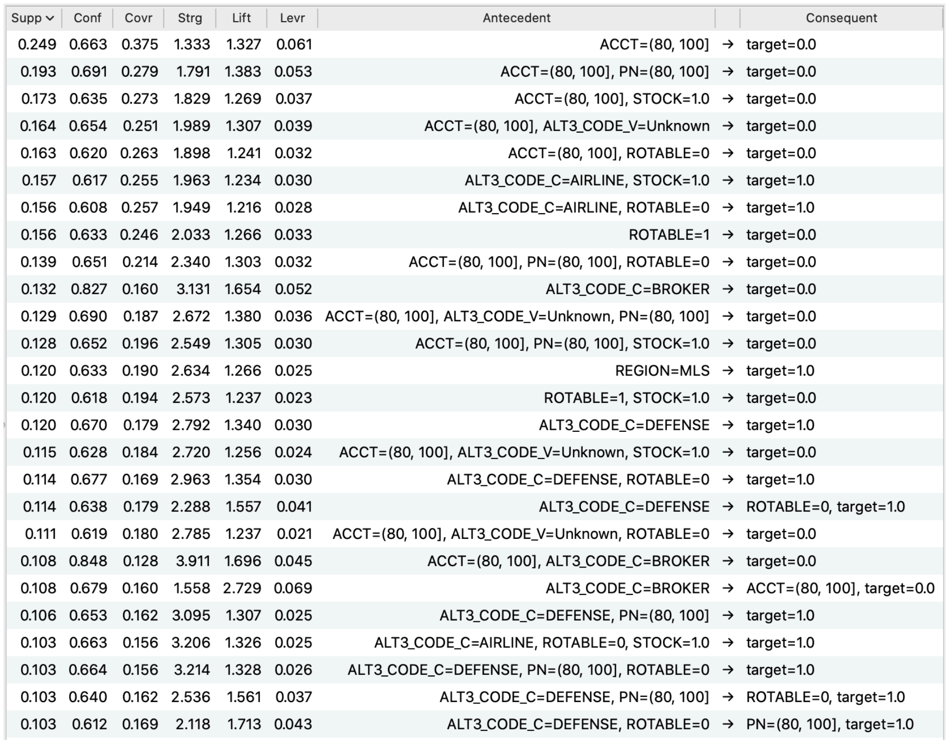


Figure 50: Probability of (no) sale per account.

Further, association rules (Section 2.4) have been derived regarding feature class, whether a quote converts into a sales order (or not). The association rules can be seen in Figure 51 and were extracted under restriction of ≥10% and ≥60% for minimal support and minimal confidence respectively. Support represents the fraction of the data set that contains the association rule and confidence the probability of occurrence of the consequent given the antecedent.



Supp	Conf	Covr	Strg	Lift	Levr	Antecedent		Consequent
0.249	0.663	0.375	1.333	1.327	0.061	ACCT=(80, 100]	→	target=0.0
0.193	0.691	0.279	1.791	1.383	0.053	ACCT=(80, 100], PN=(80, 100]	→	target=0.0
0.173	0.635	0.273	1.829	1.269	0.037	ACCT=(80, 100], STOCK=1.0	→	target=0.0
0.164	0.654	0.251	1.989	1.307	0.039	ACCT=(80, 100], ALT3_CODE_V=Unknown	→	target=0.0
0.163	0.620	0.263	1.898	1.241	0.032	ACCT=(80, 100], ROTABLE=0	→	target=0.0
0.157	0.617	0.255	1.963	1.234	0.030	ALT3_CODE_C=AIRLINE, STOCK=1.0	→	target=1.0
0.156	0.608	0.257	1.949	1.216	0.028	ALT3_CODE_C=AIRLINE, ROTABLE=0	→	target=1.0
0.156	0.633	0.246	2.033	1.266	0.033	ROTABLE=1	→	target=0.0
0.139	0.651	0.214	2.340	1.303	0.032	ACCT=(80, 100], PN=(80, 100], ROTABLE=0	→	target=0.0
0.132	0.827	0.160	3.131	1.654	0.052	ALT3_CODE_C=BROKER	→	target=0.0
0.129	0.690	0.187	2.672	1.380	0.036	ACCT=(80, 100], ALT3_CODE_V=Unknown, PN=(80, 100]	→	target=0.0
0.128	0.652	0.196	2.549	1.305	0.030	ACCT=(80, 100], PN=(80, 100], STOCK=1.0	→	target=0.0
0.120	0.633	0.190	2.634	1.266	0.025	REGION=MLS	→	target=1.0
0.120	0.618	0.194	2.573	1.237	0.023	ROTABLE=1, STOCK=1.0	→	target=0.0
0.120	0.670	0.179	2.792	1.340	0.030	ALT3_CODE_C=DEFENSE	→	target=1.0
0.115	0.628	0.184	2.720	1.256	0.024	ACCT=(80, 100], ALT3_CODE_V=Unknown, STOCK=1.0	→	target=0.0
0.114	0.677	0.169	2.963	1.354	0.030	ALT3_CODE_C=DEFENSE, ROTABLE=0	→	target=1.0
0.114	0.638	0.179	2.288	1.557	0.041	ALT3_CODE_C=DEFENSE	→	ROTABLE=0, target=1.0
0.111	0.619	0.180	2.785	1.237	0.021	ACCT=(80, 100], ALT3_CODE_V=Unknown, ROTABLE=0	→	target=0.0
0.108	0.848	0.128	3.911	1.696	0.045	ACCT=(80, 100], ALT3_CODE_C=BROKER	→	target=0.0
0.108	0.679	0.160	1.558	2.729	0.069	ALT3_CODE_C=BROKER	→	ACCT=(80, 100], target=0.0
0.106	0.653	0.162	3.095	1.307	0.025	ALT3_CODE_C=DEFENSE, PN=(80, 100]	→	target=1.0
0.103	0.663	0.156	3.206	1.326	0.025	ALT3_CODE_C=AIRLINE, ROTABLE=0, STOCK=1.0	→	target=1.0
0.103	0.664	0.156	3.214	1.328	0.026	ALT3_CODE_C=DEFENSE, PN=(80, 100], ROTABLE=0	→	target=1.0
0.103	0.640	0.162	2.536	1.561	0.037	ALT3_CODE_C=DEFENSE, PN=(80, 100]	→	ROTABLE=0, target=1.0
0.103	0.612	0.169	2.118	1.713	0.043	ALT3_CODE_C=DEFENSE, ROTABLE=0	→	PN=(80, 100], target=1.0

Figure 51: Association rules.

From Figure 51 some intuitive relations follow, e.g. when features account and/or part have value (80,100] as (part of) antecedent, often (part of) the consequent is feature class to be "No sale". This makes sense since features account and part are ABC classified on contribution to the total sales order revenue where (80,100] is the lowest contributing bin. On the other hand, it also follows that e.g. selling low contributing non-rotable parts to defense and selling non-rotable parts from stock to airlines often result in a sale.

Finally, insight into how the operational prioritization tool predicts sales potential is given through visualization of a single random tree from the Random Forest (Figure 52), upon which is zoomed-in on levels 1-3 (Figure 53). Giving this insight into what most will perceive to be a "black-box" is expected to foster adoption in practice. The operational prioritization tool consists of 400 trees similar to the one shown in Figure 52, from which the majority vote determines the prediction outcome.

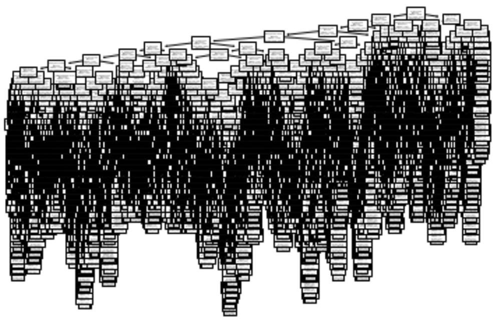


Figure 52: Single tree from Random Forest.

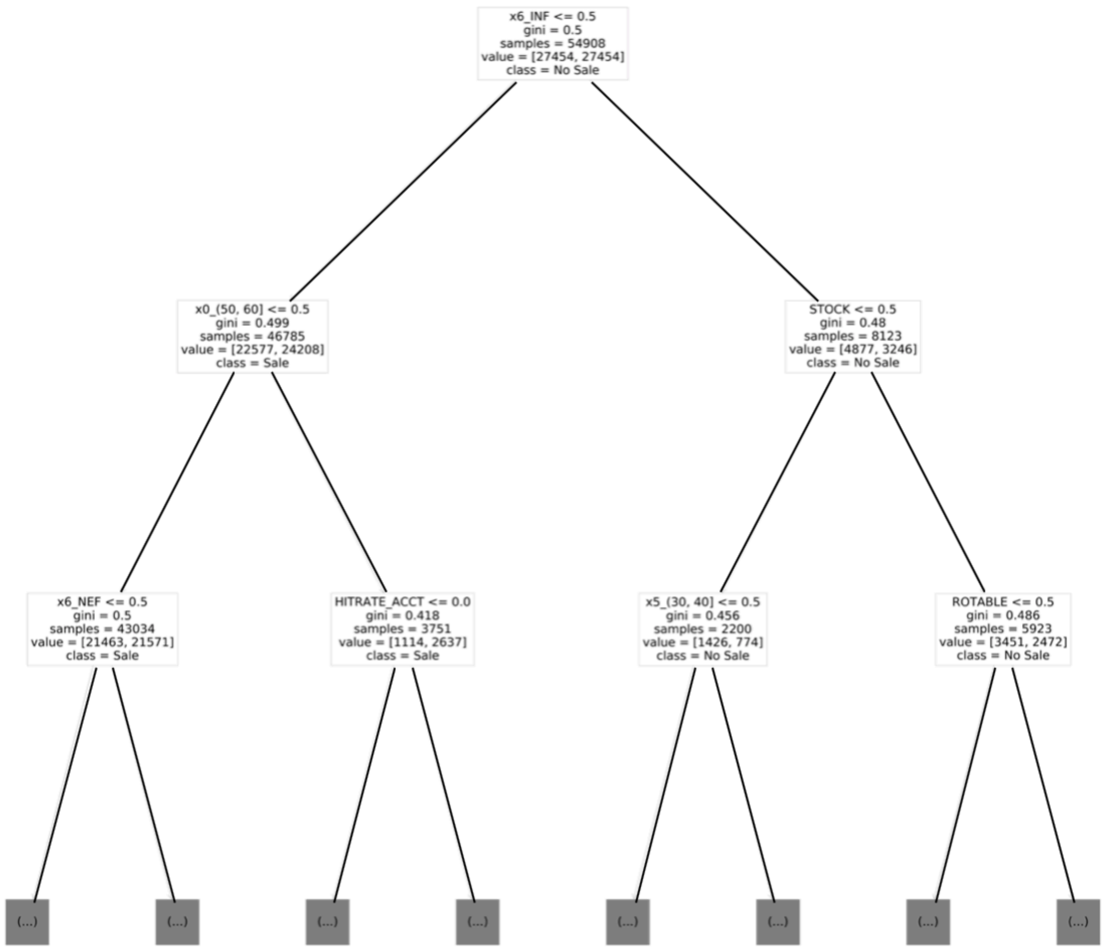


Figure 53: Single (random) tree from Random Forest, restricted max_depth =2.

5.2 Performance

In this section the performance of the operational prioritization tool has been evaluated, resulting from application of the B2B sales potential prediction methodology (Bohanec, et al., 2015) to the prototype prioritization tool. From this followed that the operational prioritization tool is a Random Forest model with hyper-parameter configuration {'n_estimators': 400, 'min_samples_split': 2, 'min_samples_leaf': 1, 'max_features': 13, 'max_depth': None, 'criterion': 'gini', 'bootstrap': False} and optimal classification threshold 0.62.

The performance of was evaluated using 5-fold stratified cross validation with (machine learning) data from the period 2012-01-01 to 2019-01-01, where 80%/20% was allocated for training- and evaluation respectively. The operational prioritization tool machine learning data set consists of features; hit rate account, frequency account, main supplier list price, frequency part, account, customer type, stock, part, default supplier, part manufacturer, rotable part, sales manager and supplier type. From this follows that the performance of the prototype prioritization tool is 56.24% for the F1 score, 83% for the AUC and, following from the confusion matrix, a total misclassification rate of 19.77%.

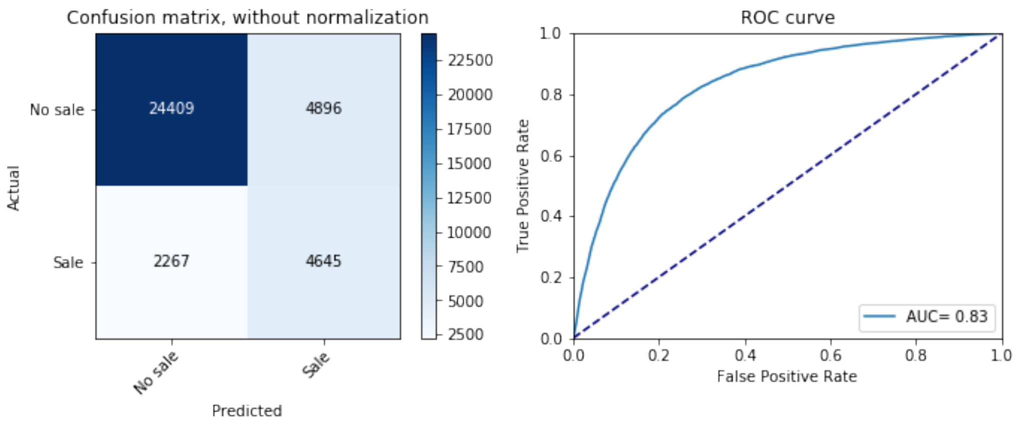


Figure 54: Confusion matrix and ROC curve operational prioritization tool.*

	Accuracy	**Precision**	**Recall**	**F1-score**
Fold 1	0.802	0.487	0.672	0.565
Fold 2	0.802	0.486	0.670	0.563
Fold 3	0.801	0.485	0.670	0.563
Fold 4	0.801	0.484	0.670	0.562
Fold 5	0.800	0.483	0.665	0.559
Average	**0.801**	**0.485**	**0.669**	**0.562**
Max Δ fold	**0.002**	**0.004**	**0.007**	**0.006**

Table 20: 5-fold cross validation operational prioritization tool.

*It might strike that the numbers in the confusion matrices of the prototype- and operational prioritization tool do not amount to the same. This is because, as part of data preparation, quotes containing missing values (MAR/MCAR) are excluded (Section 4.1.2.4). The prototype prioritization tool did not include feature delivery window width (Section 4.2) which had 3.5% missing values (Table 11) whereas feature stock, a meta feature based on delivery window width (which therefore also has 3.5% missing values), is included in the operational prioritization tool.

5.3 Conclusion

In this chapter research sub-questions 4.1 to 4.4 were answered. RQ 4.1 *"What historical sales data is available?"* was answered in Section 5.1.1. There, additional (meta) features were shown in Table 15 that have been added to the machine learning data set in consultation with the product management department, whom are considered to be experts on the RFQ process and quote data itself.

RQ 4.2 *"What data- and algorithm combination yields the best prediction performance?"* was addressed in Section 5.1.3 Here, feature selection, hyper-parameter tuning and classification threshold optimization were carried out. Feature selection was carried out according to the sub-methodology by Bohanec et al. (2015). First, features were ranked according to multiple filter methods (and classifiers) using Orange datamining suite. Next, for classifiers Random Forest, Gradient Boosting Classifier and Logistic Regression, top ranked features were added one-by-one and F1 score performance on the test set was monitored. This, to detect the optimal cut-off point, being the combination of classifier and number of top ranked features that yields the highest overall F1 score performance. Next, for these three classifiers and their corresponding optimal cut-off points, features were excluded one-by-one from highest to lowest ranked. When exclusion increased performance, the concerned feature was permanently excluded, resulting in a minimum list of features. Executing the feature selection methodology resulted in an optimal model for each classifier. After feature selection, a performance comparison of the optimal models was carried out using a k-fold cross validation approach, from which followed that the 13-feature Random Forest model performed best.

Next, hyper-parameter tuning was carried out for the 13-feature Random Forest model. Validation curves per hyper-parameter were plotted, showing promising areas per hyper-parameter, yet not accounting for sequential effects. Therefore, an initial random search 3-fold of 250 configurations was executed. Using these results, a targeted grid was identified and further explored with an additional random search 3-fold of 500 configurations. From this additional search followed that configuration {'n_estimators': 400, 'min_samples_split': 2, 'min_samples_leaf': 1, 'max_features': 13, 'max_depth': None, 'criterion': 'gini', 'bootstrap': False} yields the highest generalizable F1 score performance improvement.

Last, classification threshold optimization was carried out for the 13-feature Random Forest model with the best hyper-parameter configuration. The classification threshold can be seen as a hyper-parameter and should be treated accordingly, although it cannot be set within the classifier and rather has to be coded manually. From Figure 47 and Figure 48 followed that classification threshold that yields the best F1 score on the validation set (3-fold CV) is 0.62.

RQ 4.3 *"What data insights can be used for organizational learning?"* was answered in Section 5.1.4 where feature distributions, association rules and a classification tree visualization were shown.

RQ 4.4 *"How is the performance of the operational prioritization tool?"* was answered in Section 5.2. From this followed that the performance of the operational prioritization tool is 56.24% for the F1 score, 83% for the AUC and, following from the confusion matrix, a total misclassification rate of 19.77%.

Comparing the operational prioritization tool (Figure 54) to the prototype prioritization tool (Figure 32), it might strike that the number of quote lines correctly predicted to result in a sale decreased after improvement. The reason for this is that in improving the prototype prioritization tool, focus was on metric F1 score, which represents the trade-off between recall, the percentage of actual sales correctly predicted to be a sale, and precision, the percentage of predicted sales that is an actual sale, both weighted equally. The improvement of the operational- over the prototype prioritization tool is made by sacrificing little recall for a significant increase in precision. This means, that the operational prioritization tool now predicts approximately 50/50 correctly to be a sale, compared to the historical 17% quote-to-sales order conversion rate resulting from the mental decision models of the Customer Support Department (increase of $\frac{0.485-0.17}{0.17} * 100\%$ = 185.29%). The need for the operational prioritization tool sprung, amongst other things, from the fact that the Customer Support Department receives more RFQs than they can process (Figure 1). Using the operational prioritization tool in the future will allow the Customer Support Department to respond to more RFQs and, assuming the same distribution of "Sales"/"No sales" amongst the RFQs that previously could not be processed, generate more sales.

Besides a clear practical contribution, the research carried out in this thesis also contributes to theory. This by giving an indication of the improvement that can be gained through incorporation of machine learning over manual handling in B2B sales forecasting.

6. Creation of the Information Extraction module

The operational prioritization tool has proven to more accurately- and efficiently predict which RFQs have the potential to convert into a sale. Yet, the RFQs still have to be manually described in terms of the features used by the operational prioritization tool to predict sales potential (Table 26). In this chapter an attempt is made to automate this process through the creation of the Information Extraction module, which theoretically, in combination with the RFQ inbox and operational prioritization tool allows RFQs to be prioritized on sales potential without the need of human input. In Section 6.1 a discrepancy between RFQ data and the customer data source is described due to which prioritization currently cannot happen automatically and autonomously. Next, Section 6.2 contains a description of the Information Extraction module, whose code with “translation” can be seen in Appendix F. Finally, in Section 6.3 the performance of the Information Extraction module is evaluated.

6.1 Discrepancy between RFQ and -Master Customer data

The operational prioritization tool takes input features shown in Table 26 which can all be gathered if features account- and part number are known. Each RFQ contains, amongst other things, the requested part number(s), the name of the requesting company and the customer email address of which the latter two both theoretically can be linked to feature account number.

However, in reality there appears to be a significant discrepancy between customer names as they occur in the master customer schema and as they occur in the RFQs consisting of different wording rather than punctuation, capital letters, etc. Moreover, customer email address data is currently not available. Consequently, both customer company name and customer email address cannot be used to retrieve the feature account number and corresponding (customer) features due to which the operational prioritization tool cannot automatically and autonomously be applied to RFQs.

After consulting with Company X on this limitation, which is due to unavailability/non-existence of the required data, it was decided to still create the information extraction module but hold-off implementation of the flow depicted in Figure 2 until Company X in the future resolves the discrepancy. Further, Company X expressed preference for use of partial customer email address, from “@” until the end of the email address which (usually) contains the company name, to retrieve customer account number and corresponding features. Here, the argument is that customers may spell their company name differently in RFQs whereas their partial email address will always remain the exact same.

6.2 Information extraction module

The Company X RFQ inbox (Exchange outlook) can be accessed in python using the package ExchangeLib (Medium, 2019). Once connection has been established the X most recent emails are retrieved for which a data frame is created containing their corresponding text bodies, sender email addresses and conversation ids.

Next, the part number(s) and partial customer email address are extracted from RFQ text bodies. First, part numbers are extracted using an *Entity Ruler*, which is a matcher based on a dictionary of patterns and corresponding (entity) labels. The Entity Ruler identifies entities in texts which can be called upon per label. In the case of Company X, the Entity Ruler is given a dictionary of all part numbers (approximately 1 million) as seen in the Master Part schema with label “PART”. Now, when the Entity Ruler is applied to historical and/or future RFQs, it will find any part number in the RFQ that also occurs in the Master Part schema. Note that a custom tokenizer had to be defined to extract part numbers as the (spacy) default tokenizer interprets “-“ to be an infix and as a consequence splits part numbers, which often contain multiple ‘-‘ symbols, into multiple tokens allowing the entity ruler to falsely identify part numbers within the requested part number(s) (Longest, 2018).

Second, partial customer email addresses are recognized at the moment using a function but can, once Company X creates customer email address data, be recognized similarly to part number(s) by applying this function to the customer email address data and feeding the resulting partial customer email addresses into the Entity Ruler with label “CUSTOMER_EMAIL”. In addition, in extraction of partial customer email addresses a distinction is made between direct- and indirect customer RFQs, the latter entailing (online) spare part marketplaces. For the direct RFQs the partial customer email address is extracted from the sender email address and for the indirect RFQs the partial customer email address is extracted from the RFQ text body.

6.3 Performance

Performance of the information extraction module has been manually evaluated on 100 random historical RFQs (due to time constraints), from which followed:

- For 100/100 RFQs the correct part number(s) are recognized.
- For 99/100 RFQs a single correct partial customer email address has been recognized. For 1/100 RFQs two partial customer email addresses have been recognized, the reason being that the customer stated two different email addresses, yet still correctly.

However, besides identifying the correct part number(s), the entity ruler also falsely identifies other numbers occurring in the RFQs to be part numbers. These wrongly identified part numbers are in fact existing part numbers but not in the context of the RFQ, e.g. street number, tracking number, quantity required, etc. This behavior is expected for the Entity Ruler as it doesn’t account for context. Still, the Entity Ruler remains the best choice for the task at hand due to its high accuracy performance. This is desired because the extracted part number(s) and partial customer email address will serve as input for the operational prioritization tool in a fully autonomous/automated flow, meaning that any unrecognized part number by definition will never become a sale.

To diminish the issue of falsely identified part numbers, an array has been created in which such frequently wrongly identified (part) numbers can be specified. The elements of this array will not be given to the Entity Ruler and will therefore no longer be (falsely) recognized. It should be noted that this is a trade-off between falsely identified (part) numbers and not recognizing a requested (part) number, into which further research is recommended.

6.4 Conclusion

In this chapter the information extraction module has been addressed, which has the objective of extracting part number(s) and partial customer email address from RFQs. Then, using the extracted information, features from the machine learning data set can be gathered after which the operational prioritization tool can be applied to predict the sales potential of the concerned RFQs (illustrated in Figure 55). In this chapter research sub-questions 5.1 and 5.2 have been answered.

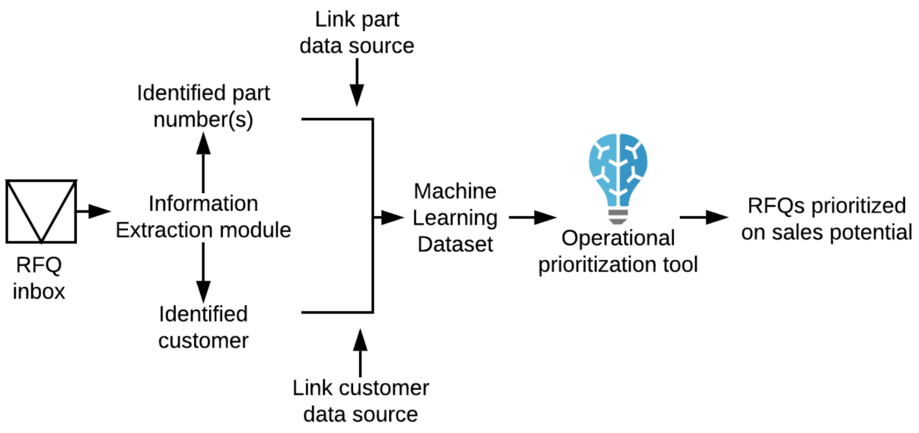

Link part data source
Identified part number(s)
RFQ inbox
Information Extraction module
Identified customer
Machine Learning Dataset
Operational prioritization tool
RFQs prioritized on sales potential
Link customer data source

Figure 55:Illustration theoretical flow Information Extraction module combined with the operational prioritization tool.

RQ 5.1 *"How is the information extraction module build?"* has been answered in Section 6.2. The Company X RFQ inbox (Exchange Outlook) is accessed using package ExchangeLib. After, a data frame is created in which the text bodies, sender email addresses and conversation ids of the X most recent emails are stored. Next, part number(s) and partial customer email address are extracted from RFQ text bodies using an Entity Ruler combined with custom tokenizer and function respectively. Currently, due to a discrepancy between RFQ data and master customer data and/or non-existence of data, prioritization of RFQs on sales potential cannot happen automatically and autonomously.

RQ 5.2 *"How is the performance of the information extraction module?"* was answered in Section 6.3. There, performance was manually evaluated on 100 historical RFQs from which followed that all requested part numbers- and the correct partial customer email address were extracted. It was also revealed that Entity Ruler, besides recognizing the requested (correct) part numbers, falsely identifies numbers occurring in the RFQs to be part numbers. The reason for this is that these falsely identified (part) numbers are in fact existing part numbers but not in the context of the RFQ, for which the Entity Ruler cannot account. To diminish this issue, an array has been created in which frequently wrongly identified (part) numbers can be specified which will no longer (falsely) be recognized.

7. Conclusion, limitations and recommendations

7.1 Conclusion

In this section focus is on answering the main research question *"How can Company X be enabled to have their incoming RFQs automatically, autonomously and (more) accurately prioritized on their sales potential?"*. From the research carried out in this thesis followed that the answer to the main research question is sequential application of the information extraction module and operational prioritization tool, as illustrated in Figure 2. Yet, prioritization cannot happen automatically and autonomously until Company X creates (partial) customer email address data (Section 6.1).

7.1.1 Information extraction module

The Company X RFQ inbox (Exchange Outlook) is accessed by the information extraction module. After this, a data frame is created in which text bodies, sender email addresses and conversation ids of the X most recent emails are stored.

Next, part number(s) and partial customer email address are extracted from RFQ text bodies using an Entity Ruler combined with custom tokenizer and function respectively. This extracted information can then theoretically be linked to master part- and master customer data and therefore used to create the machine learning data set required as input for the operational prioritization tool. However, in practice partial customer email address cannot be linked to master customer data due to non-existence of (partial) customer email address data. Consequently, until Company X creates this data, prioritization of RFQs on sales potential cannot happen automatically and autonomously.

Performance evaluation (manually) of the information extraction module on 100 historical RFQs proved that all requested part numbers and correct partial customer email addresses were recognized. However, in addition, "other" numbers occurring in the RFQs were falsely recognized as part numbers. These numbers are in fact existing part numbers but not in the context of the RFQ. This issue is addressed via an array in which frequently falsely identified (part) numbers can be stated which will then not be given to the Entity Ruler and therefore no longer (falsely) recognized.

7.1.2 Operational prioritization tool

At the start of this research Company X had already created a tool in Python which supposedly would (binary) prioritize RFQs on sales potential. Yet, this tool, to which is referred as the concept prioritization tool, had not been implemented in day-to-day operations, nor had its performance been determined and more importantly, Company X had expressed to seriously question its validity. The latter justly so, as analysis of the code behind the concept prioritization tool revealed many defects to be present, amongst which data leakage caused by features revenue and condition. All quotes with missing values for both features revenue and condition would not convert into a sale, causing the model to "predict" solely on the presence of these features rather than the intrinsic information of the data, resulting in an (unjust) near perfect performance. After fixing this data leak, the concept prioritization tool performance significantly dropped from previously assumed 98.87% F1-score and 100% AUC to merely 6.8% F1-score and 52% AUC (Figure 24).

Afterwards, validation/verification changes were implemented addressing the identified defects in the concept prioritization tool, resulting in the prototype prioritization tool. This part of the research consisted in essence of re-designing and re-building the concept prioritization tool from scratch, addressing e.g. missing functions, unrepresentative data, unrepresentative linking of quotes to sales orders, double weighting of features, inability to handle features with a large number of categorial values, faulty data splitting method, etc. Performance evaluation of the prototype prioritization tool revealed an F1-score of 45.5% and AUC of 77% (Figure 32).

Finally, an attempt was made to improve prediction performance of the prototype prioritization tool through through application of a B2B sales potential prediction methodology (Bohanec, et al., 2015), resulting in the operational prioritization tool. From this followed that the operational prioritization tool is a 13-feature Random Forest model (Table 26) with hyper parameter configuration shown in Section 5.1.3.3 and (optimal) classification threshold 0.62. Performance evaluation of the operational prioritization tool revealed an F1-score of 56.24% and AUC of 83% (Figure 54).

Compared to the historical 17% quote-to-sales order conversion rate of Company X, resulting from the mental decision models of the Customer Support Department, the operational prioritization tool predicts 48.5% correctly to be a sale, which is an increase of $\frac{0.485-0.170}{0.170} * 100\%$ = 185.3%. The need for the operational prioritization tool sprung, amongst other things, from the fact that the Customer Support Department receives more RFQs than they can process. Therefore, use of the operational prioritization tool in the future will allow the Customer Support Department to respond to more RFQs and, under assumption of the same distribution of “No sale”/“Sale” amongst the RFQs that previously could not be processed, generate more sales. In addition, the research carried out in this thesis also contributed to theory. This by giving an indication of the improvement that can be gained through incorporation of machine learning over manual handling in B2B sales forecasting.

7.2 Limitations

In this section the research limitations, in (chronological) order of occurrence, are discussed.

Data bias

The first limitation of the research described in this thesis is (historical) data bias. The data used for training- and evaluating the prioritization tool is that of quotes while the desired application is RFQs. Figure 20 shows that a quote is the response of Company X to an RFQ, containing price, lead-time, documentation, etc. and is thus one step ahead in the process. The RFQs prior to the quotes, whose data is used to train- and evaluate the prioritization tool, were thus deemed to have sales potential by the Customer Support Department. Therefore, the inferences which the prioritization tool picks up from the training data and uses to make predictions thus also contain this bias.

Data quality issues

The second limitation of this research regards data quality issues, of which the following forms occurred:

- Data dispersion. Quotes and corresponding sales orders are created separately, although best practice would be to create a sales order from the prior quote, due to which their linkage is now approximated.
- Data categorization is not standardized. Different employees categorize observations differently (subjectively), causing affected features to be “contaminated”. As a consequence, the potential predictive power of these contaminated features is diminished.

Existing customers/parts only
The third limitation is that the prioritization tool can only be applied to existing customers and parts. The reason for this is that the prioritization tool predicts using features that are linked to the customer account- and part number. As a result, when either the customer account- and/or part number is missing, all corresponding features are missing as well. Consequently, the operational prioritization tool cannot be applied to new customers which therefore have to be handled manually instead. If not, no sales would be made anymore outside the current customer base. Regarding parts this limitation is not an issue since Company X only sells parts occurring in the master part number schema.

Direct sales and exchanges only
The fourth limitation is that the prioritization tool should only be used to predict sales potential of RFQs regarding a direct sale or exchange. The reason for this is that the prioritization tool has been trained solely on historical quote data regarding direct sales/exchanges because these follow the same process (Figure 20). Therefore, the inferences picked up by the prioritization tool from the training data apply to direct sales/exchanges but may not be generalizable to other sales types.

Interaction effects
The fifth limitation is the failure to account for potential interaction effects of feature selection, hyper-parameter tuning and classification threshold optimization, which are executed sequentially (in that order), each taking input from its predecessor. A logical “solution” would seem to be an additional iteration of the concerned sequential steps, yet this is not possible in practice due to hyper-parameter restrictions. Hyper parameter max_features does not allow the feature selection methodology to be executed for the best configuration resulting from hyper-parameter tuning. Specifically, adding top ranked features one-by-one and monitoring performance cannot be carried out when the number of features in the machine learning data set is below max_features, thus making it impossible to measure the true potential interaction effects.

Information extraction module recognizes without context
The final limitation is that the information extraction module, besides recognizing the correct part number(s), also falsely identifies other numbers occurring in the RFQs to be part numbers. These wrongly identified part numbers are in fact existing part numbers but not in the context of the RFQ, e.g. street number, tracking number, quantity required, etc. This behavior is expected for the Entity Ruler as it doesn’t account for context. Yet, the Entity Ruler remains the best choice for the task at hand due to its high accuracy performance, which is desired as the extracted part number(s) and partial customer email address serve as input for

the operational prioritization tool in a fully autonomous/automated flow, meaning that any unrecognized part number by definition will never become a sale.

Moreover, training a NER model (Section 2.3.4), which does account for context, was also attempted but has shown high losses during training indicating, and following from performance, that the NER model was unable to learn properly and thus unsuitable for the task at hand.

7.3 Practical recommendations

Improve data handling/storage
As mentioned in the limitations, current practice involves data dispersion and subjective data categorization. Consequently, data products, such as the operational prioritization tool, suffer as the quality of output information cannot be better than the input information (*"Garbage in, garbage out"*).

To improve data handling and -storage I would suggest standardizing data logging processes:

- Enforce sales orders to-be created from its preceding quote, tackling data dispersion.
- Standardize data categorization via e.g. categorization rules, tackling subjective data categorization and consequently "contamination" of features.

Re-train the prioritization tool regularly
Features account, part, part manufacturer and default supplier from the operational prioritization tool machine learning data set are all encoded using ABC classification (Section 4.1.5). This means, that the categories within these features that contribute the most to the total sales order revenue are not encoded and others are assigned a bin according to their contribution. Thus, if over time a shift would occur in the top contributing categories, the prioritization tool would not be able to recognize the category (as it has not seen it during training) and raise an error. To prevent this, the model should be re-trained regularly using the most recent available data (when re-training the prioritization tool set input parameter end_date = dt.datetime.today())

7.4 Further research recommendations

Investigate different recall-precision trade-offs
In this thesis the focus was on improving the F1 score metric of the prioritization tool. The F1 score metric represents the trade-off between recall, the percentage of actual sales correctly predicted to be a sale, and precision, the percentage of predicted sales that is an actual sale, both weighted equally (Section 2.2.3.2). This because both wrongly predicted sales (concerning precision), which waste time, and wrongly predicted non-sales (concerning recall), which cause sales to be missed, ought to be minimized. Within this trade-off equal weight was chosen, as "optimal" weights are merely subjective. I would recommend Company X to repeat the research in this thesis for different trade-off weightings, so that the effect of different trade-off weightings on the confusion matrix distribution can be understood and an (subjective) optimal trade-off weighting can be found.

Prioritization on sales potential and -value
Currently, the operational prioritization tool solely predicts sales potential and disregards the value corresponding to a sale. In practice this means that, for example, a bolt (low-value) with a predicted sales probability of 0.95 would be prioritized over an engine (high-value) with predicted sales probability of 0.94, which is not desired as the engine obviously would make significantly more profit for a neglectable difference in sales potential. Therefore, I would recommend Company X to establish a feature which represents a desired the trade-off between sales potential and -value and repeat the research from this thesis on this feature.

Investigate trade-off false positive and false negative
The information extraction tool, besides recognizing the correct part number(s), also falsely recognizes numbers occurring in the RFQs to be part numbers. The reason being that an Entity Ruler was used, which doesn't account for context, and that these numbers are in fact existing part numbers. Still, the Entity Ruler remains the best choice for the task at hand due to its high accuracy performance. This is desired because the extracted part number(s) and partial customer email address will serve as input for the operational prioritization tool in a fully autonomous/automated flow, meaning that any unrecognized part number by definition will never become a sale.

To diminish the extend of this issue an array has been created in which frequently falsely identified (part) numbers can be specified, which will then not be fed to the Entity Ruler and therefore no longer (falsely) recognized. However, this is a trade-off between false positives (falsely recognized (part) numbers) and false negatives (part numbers in the array will no longer be recognized).

8. References

Yse, D. L., 2019. *Your Guide to Natural Language Processing (NLP).* [Online] Available at: https://towardsdatascience.com/your-guide-to-natural-language-processing-nlp-48ea2511f6e1
[Accessed 20 08 2019].

Bohanec, M., Borštnar, M. K. & Robnik - Šikonja, M., 2015. *Integration of machine learning insights into organizational learning: a case of B2B sales forecasting.* Bled, s.n.

Rouse, M., 2018. *Association rules (in data mining).* [Online] Available at: https://searchbusinessanalytics.techtarget.com/definition/association-rules-in-data-mining
[Accessed 23 08 2019].

Clark University, n.d. *Classification Tree Analysis.* [Online] Available at: https://clarklabs.org/classification-tree-analysis/
[Accessed 23 08 2019].

Bohanec, M., Borštnar, M. K. & Robnik-Šikonja, M., 2015. *Feature subset selection for b2b sales forecasting.* Bled, The 13th International Symposium on Operations Research.

Mortensen, S. et al., 2019. *Predicting and Defining B2B Sales Success with Machine Learning.* Charlottesville, IEEE.

Analytics University , 2017. *Feature Selection in Machine learning| Variable selection| Dimension Reduction.* [Online] Available at: https://www.youtube.com/watch?v=TsqTuwTKFSs
[Accessed 28 08 2019].

Udemy, 2019. *NLP - Natural Language Processing with Python.* [Online] Available at: https://www.udemy.com/nlp-natural-language-processing-with-python/
[Accessed 6 09 2019].

Techopedia, n.d. *Named-Entity Recognition (NER).* [Online] Available at: https://www.techopedia.com/definition/13825/named-entity-recognition-ner
[Accessed 10 9 2019].

Heerkens, H. & Van Winden, A., 2012. De probleemidentificatie. In: *Geen probleem.* Nieuwegein: Van Winden Communicatie, pp. 44-55.

Expert System, n.d. *What is Machine Learning? A definition.* [Online] Available at: https://www.expertsystem.com/machine-learning-definition/
[Accessed 13 9 2019].

MIT, 2016. *Lecture 11: Introduction to Machine Learning.* [Online] Available at: https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0002-introduction-to-computational-thinking-and-data-science-fall-2016/lecture-videos/lecture-11-introduction-to-machine-learning/
[Accessed 13 9 2019].

Elite Data Science, n.d. *5 Heroic Python NLP Libraries.* [Online] Available at: https://elitedatascience.com/python-nlp-libraries
[Accessed 16 09 2019].

sunscrapers, 2018. *6 best Python Natural Language Processing (NLP) libraries.* [Online] Available at: https://sunscrapers.com/blog/6-best-python-natural-language-processing-nlp-libraries/
[Accessed 16 09 2019].

nltk.org, n.d. *Natural Language Toolkit.* [Online]
Available at: http://www.nltk.org
[Accessed 16 09 2019].
TextBlob, n.d. *TextBlob: Simplified Text Processing.* [Online]
Available at: https://textblob.readthedocs.io/en/dev/
[Accessed 16 09 2019].
pypi.org, 2019. *gensim 3.8.0.* [Online]
Available at: https://pypi.org/project/gensim/
[Accessed 16 09 2019].
spaCy, n.d. *spaCy 101: Everything you need to know.* [Online]
Available at: https://spacy.io/usage/spacy-101
[Accessed 16 09 2019].
Stack Abuse, 2019. *Python for NLP: Tokenization, Stemming, and Lemmatization with SpaCy Library.* [Online]
Available at: https://stackabuse.com/python-for-nlp-tokenization-stemming-and-lemmatization-with-spacy-library/
[Accessed 16 09 2019].
spaCy, n.d. *Detailed speed comparison.* [Online]
Available at: https://spacy.io/usage/facts-figures
[Accessed 16 09 2019].
MIT, 2016. *Lecture 13: Classification.* [Online]
Available at: https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0002-introduction-to-computational-thinking-and-data-science-fall-2016/lecture-slides-and-files/MIT6_0002F16_lec13.pdf
[Accessed 17 09 2019].
MIT, 2016. *Lecture 12: Clustering.* [Online]
Available at: https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0002-introduction-to-computational-thinking-and-data-science-fall-2016/lecture-slides-and-files/MIT6_0002F16_lec12.pdf
[Accessed 17 09 2019].
Kaggle, 2018. *K-Means Clustering vs. Logistic Regression.* [Online]
Available at: https://www.kaggle.com/minc33/k-means-clustering-vs-logistic-regression
[Accessed 18 09 2019].
Li, A. Y. & Elliot, N., 2019. Natural language processing to identify ureteric stones inradiology reports. *Journal of Medical Imaging and Radiation Oncology,* Volume 63, p. 307–310.
Rouse, M., 2019. *Big data.* [Online]
Available at: https://searchdatamanagement.techtarget.com/definition/big-data
[Accessed 18 09 2019].
i-scoop, n.d. *Unstructured data: turning data into actionable intelligence.* [Online]
Available at: https://www.i-scoop.eu/big-data-action-value-context/unstructured-data/#Unstructured_data_the_untapped_majority_of_data_which_grows_faster_than_any_other_type_of_data
[Accessed 18 09 2019].
Rouse, M., 2018. *Unstructured data.* [Online]
Available at: https://searchbusinessanalytics.techtarget.com/definition/unstructured-data
[Accessed 18 09 2019].

Python, n.d. *What is Python? Executive Summary.* [Online]
Available at: https://www.python.org/doc/essays/blurb/
[Accessed 19 09 2019].
Wu, J., 2019. *Python vs. R — Choosing the Best Programming Language for Data Science.* [Online]
Available at: https://towardsdatascience.com/python-vs-r-choosing-the-best-programming-languages-for-data-science-b1327f01f6bf
[Accessed 19 09 2019].
Medium, 2018. *Python vs R for Data Science: And the winner is...* [Online]
Available at: https://medium.com/@data_driven/python-vs-r-for-data-science-and-the-winner-is-3ebb1a968197
[Accessed 19 09 2019].
Ven, R., 2018. *Python vs. R: Which Should You Choose For Your Next ML Project?.* [Online]
Available at: https://dzone.com/articles/python-or-r-which-should-you-choose-for-your-next
[Accessed 19 09 2019].
Medium, 2017. *Random Forest Simple Explanation.* [Online]
Available at: https://medium.com/@williamkoehrsen/random-forest-simple-explanation-377895a60d2d
[Accessed 21 09 2019].
StatisticsSolutions, n.d. *What is Logistic Regression?.* [Online]
Available at: https://www.statisticssolutions.com/what-is-logistic-regression/
[Accessed 21 09 2019].
Orange, 2015. *Rank.* [Online]
Available at: https://docs.biolab.si//3/visual-programming/widgets/data/rank.html
[Accessed 22 09 2019].
Kaushik, S., 2016. *Introduction to Feature Selection methods with an example (or how to select the right variables?).* [Online]
Available at: https://www.analyticsvidhya.com/blog/2016/12/introduction-to-feature-selection-methods-with-an-example-or-how-to-select-the-right-variables/
[Accessed 22 09 2019].
Aggarwal, C. C., 2014. *Data Classification: Algorithms and Applications.* New York, USA: CRC Press.
García, S., Luengo, J. & Herrera, F., 2015. *Data Preprocessing in Data Mining.* s.l.:Springer.
Medium, 2018. *Label Encoder vs. One Hot Encoder in Machine Learning.* [Online]
Available at: https://medium.com/@contactsunny/label-encoder-vs-one-hot-encoder-in-machine-learning-3fc273365621
[Accessed 21 10 2019].
Medium, 2019. *5 Ways to Detect Outliers/Anomalies That Every Data Scientist Should Know.* [Online]
Available at: https://towardsdatascience.com/5-ways-to-detect-outliers-that-every-data-scientist-should-know-python-code-70a54335a623
[Accessed 21 10 2019].
Medium, 2019. *Dealing with Missing Data.* [Online]
Available at: https://medium.com/@danberdov/dealing-with-missing-data-8b71cd819501
[Accessed 21 10 2019].

Rouse, M., 2017. *statistical noise.* [Online]
Available at: https://whatis.techtarget.com/definition/statistical-noise
[Accessed 21 10 2019].
Medium, 2019. *Isolation Forest Step by Step.* [Online]
Available at: https://medium.com/@hyunsukim_9320/isolation-forest-step-by-step-341b82923168
[Accessed 21 10 2019].
Batuwita, R. & Palade, V., 2010. *Efficient resampling methods for training support vector machines with imbalanced datasets.* Barcelona, IEEE.
Kuhn, M. & Johnson, K., 2019. *Feature Engineering and Selection: A Practical Approach for Predictive Models.* 1 ed. s.l.:CRC press.
Medium, 2018. *Why Data Normalization is necessary for Machine Learning models.* [Online]
Available at: https://medium.com/@urvashilluniya/why-data-normalization-is-necessary-for-machine-learning-models-681b65a05029
[Accessed 11 11 2019].
Datarobot, n.d. *What is a Feature Variable in Machine Learning?.* [Online]
Available at: https://www.datarobot.com/wiki/feature/
[Accessed 11 11 2019].
Rouse, M., 2018. *information extraction (IE).* [Online]
Available at: https://whatis.techtarget.com/definition/information-extraction-IE
[Accessed 11 11 2019].
Medium, 2018. *Dealing with Noisy Data in Data Science.* [Online]
Available at: https://medium.com/analytics-vidhya/dealing-with-noisy-data-in-data-science-e177a4e32621
[Accessed 5 12 2019].
Medium, 2018. *Categorical Outliers Don't Exist.* [Online]
Available at: https://medium.com/owl-analytics/categorical-outliers-dont-exist-8f4e82070cb2
[Accessed 7 12 2019].
Imbalanced-Learn, n.d. *Under-sampling.* [Online]
Available at: https://imbalanced-learn.readthedocs.io/en/stable/under_sampling.html
[Accessed 17 12 2019].
TowardsDataScience, 2018. *Using Under-Sampling Techniques for Extremely Imbalanced Data.* [Online]
Available at: https://towardsdatascience.com/sampling-techniques-for-extremely-imbalanced-data-part-i-under-sampling-a8dbc3d8d6d8
[Accessed 17 12 2019].
Medium, 2018. *Why Data Normalization is necessary for Machine Learning models.* [Online]
Available at: https://medium.com/@urvashilluniya/why-data-normalization-is-necessary-for-machine-learning-models-681b65a05029
[Accessed 6 1 2020].
Tan, P.-N., Steinbach, M. & Kumar, V., 2006. Classification: Basic Concepts, Decision Trees, and Model Evaluation. In: *Introduction to Data Mining.* s.l.:Pearson Addison-Wesley, pp. 25-44.
TowardsDataScience, 2019. *Hyperparameter Tuning.* [Online]
Available at: https://towardsdatascience.com/hyperparameter-tuning-c5619e7e6624
[Accessed 15 1 2020].

Jordan, J., 2017. *Hyperparameter tuning for machine learning models..* [Online] Available at: https://www.jeremyjordan.me/hyperparameter-tuning/ [Accessed 15 1 2020].

TowardsDataScience, 2018. *A Conceptual Explanation of Bayesian Hyperparameter Optimization for Machine Learning.* [Online] Available at: https://towardsdatascience.com/a-conceptual-explanation-of-bayesian-model-based-hyperparameter-optimization-for-machine-learning-b8172278050f [Accessed 15 1 2020].

Medium, 2018. *Hyperparameter Tuning the Random Forest in Python.* [Online] Available at: https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74 [Accessed 19 1 2020].

Bergstra, J. & Bengio, Y., 2012. Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research ,* Volume 13, pp. 1-25.

TowardsDataScience, 2019. *Optimizing Hyperparameters in Random Forest Classification.* [Online] Available at: https://towardsdatascience.com/optimizing-hyperparameters-in-random-forest-classification-ec7741f9d3f6 [Accessed 21 1 2020].

Heinze, G. & Dunkler, D., 2016. Five myths about variable selection. *Transplant International,* 30(1), pp. 1-6.

TowardsDataScience, 2018. *Hyperparameter Tuning the Random Forest in Python.* [Online] Available at: https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74 [Accessed 25 1 2020].

Medium, 2017. *In Depth: Parameter tuning for Random Forest.* [Online] Available at: https://medium.com/all-things-ai/in-depth-parameter-tuning-for-random-forest-d67bb7e920d [Accessed 25 1 2020].

Longest, D., 2018. *Adding Custom Tokenization Rules to spaCy.* [Online] Available at: http://www.longest.io/2018/01/27/spacy-custom-tokenization.html [Accessed 31 1 2020].

Medium, 2019. *You've Got Mail: Email analytics with Python and Exchange.* [Online] Available at: https://medium.com/thg-tech-blog/youve-got-mail-email-analytics-with-python-and-exchange-7ae152443957 [Accessed 11 2 2020].

Appendix A – Feature overview

Description/explanation features		
Column name	**Feature name**	**Feature description**
ACCT	Account	This feature represents ABC classified feature account number.
ACCTNO	Account number	Internal account number that Company X creates for customers. Customers can have multiple accounts corresponding to their account number (e.g. when they are both supplier and customer) which are distinguished via a sub-customer code (SUBC).
ACCOUNT_RATE	Account rate	All entries for this feature are “Unknown”.
ACCOUNT_TYPE	Account type	Company X may employ different prices (markups) for different customers. This feature determines the price (markup) for a (new) part for a customer. Note: for secondhand parts the part price is equal to its day value and equal for all customers.
PN_PROGRAM	Aircraft type	Aircraft type to which a part belongs.
ATA2	Airplane layout	Number corresponding to the area in an airplane layout where the part belongs (e.g. 52 = doors).
Y_COL, target	Class	This feature entails a binary representation of the class, where 0=”No sale” and 1=“Sale”.
COMPANYNO	Company number	The regarding branch of Company X.
COND	Condition	Condition of the part.
COUNTRY	Country	The country in which the customer resides.
TERM_CODE	Credit period	Customer specific *credit period;* the number of days a customer is allowed to wait before paying an invoice.
ALT3_CODE_C	Customer type	The customer company type, e.g. airline, defense, OEM, etc. Note: different sales managers might categorize customers differently as there are no categorization rules because of which the feature might be distorted.
AUTO_VENDOR	Default supplier	Default (part) supplier account number.
AUTO_VENDOR_SUBC	Default supplier sub code	Default (part) supplier sub account number.
DLV+RNG	Delivery window	The delivery window in days (see Section 4.1.2.2).
RNG	Delivery window time unit	The delivery window time unit; hours, days, weeks, months or years.
DLV	Delivery window width	The number of time units (days, weeks, etc.) allowed for delivery.
EXCH_CORE_RETURN	Exchange return period	Period (part specific) in which a broken part, in case of an exchange, needs to be delivered at Company X (by default 15 days).
AUTO_PRICE_ENABLED	Fixed vendor price	Binary, whether the vendor part price is fixed.
FREQ_ACCT	Frequency account	Frequency count of an account number, and thus customer, in the sales order data at the moment in time that the quote was issued.
FREQ_PN	Frequency part	Frequency count of a part number, and thus part, in the sales order data at the moment in time that the quote was issued.
HIT_RATE_ACCT	Hit rate account	The percentage of the total quoted value for an account that converted to a sale at the moment in time that the quote was issued.
HIT_RATE_PN	Hit rate part	The percentage of the total quoted value for a part that converted to a sale at the moment in time that the quote was issued.
MSLP_PRICE	Main Supplier List Price	Unnegotiated supplier part price which is visible to the entire world. This feature is an indication of the purchase/sales price as Company X can sometimes negotiate a better price.
SELL_TYPE	Mark-up	Type of part linked to mark-up structure.
ONCONDITION	Obligatory safety check	Binary, whether the manufacturer demands the part to be subjected to a safety check after certain time/usage.

PN	Part	This feature represents ABC classified feature part number.
MFG	Part manufacturer	Part manufacturer account number.
MFG_SUBC	Part manufacturer sub code	Part manufacturer sub account number.
PARTNUMBER	Part number	The part number requested by the customer in the quote. Generally, a part number represents a unique part although it is possible for multiple parts to have the same part number which are then distinguished via a sub part number code (SUBP).
PIECEPART_SUBASSY	Piece part	Binary whether a part belongs to another (larger) part.
QUOTE_DATE_COL, ENTER_DATE	Quote date	This feature represents the date and time at which the quote (line) was issued.
CAPABILITY	Repair capability	The repair capability represents which party is able to perform a repair.
ROTABLE	Rotable part	Binary feature representing whether the part is a rotable or not. Rotable parts are parts that can be repaired (second-hand).
REGION	Sales manager	The regional sales manager.
DOC_DATE	Sales order date	This feature represents the date and time at which the sales order was issued.
LINE_TYPE	Sales type	Type of sale (e.g. direct sale, repair, exchange, etc.).
STD_PART	Standard part	Binary feature representing whether the part is standardized. Standardized parts are parts that are not system specific.
STOCK	Stock	Whether the requested part was on stock; approximated under assumption that if a part was on stock, the quote feature DLV+RNG would be ≤ 7 days.
STK_TYPE	Stock type	Type of part; e.g. proprietary part, vendor part, standard part, etc.
SUBC	Sub account number	Used to distinguish between multiple accounts corresponding to the same customer.
SUBP	Sub part number	Used to distinguish between multiple parts having the same part number.
ALT3_CODE_V	Supplier type	The type of supplier.
TOTAL_REVENUE	Total revenue	The revenue corresponding to the quote.

Table 21: Feature description overview.

Appendix B – Concept prioritization tool

This Appendix is left out because of confidentiality reasons.

Appendix C – Technical description concept prioritization tool

This Appendix is left out because of confidentiality reasons.

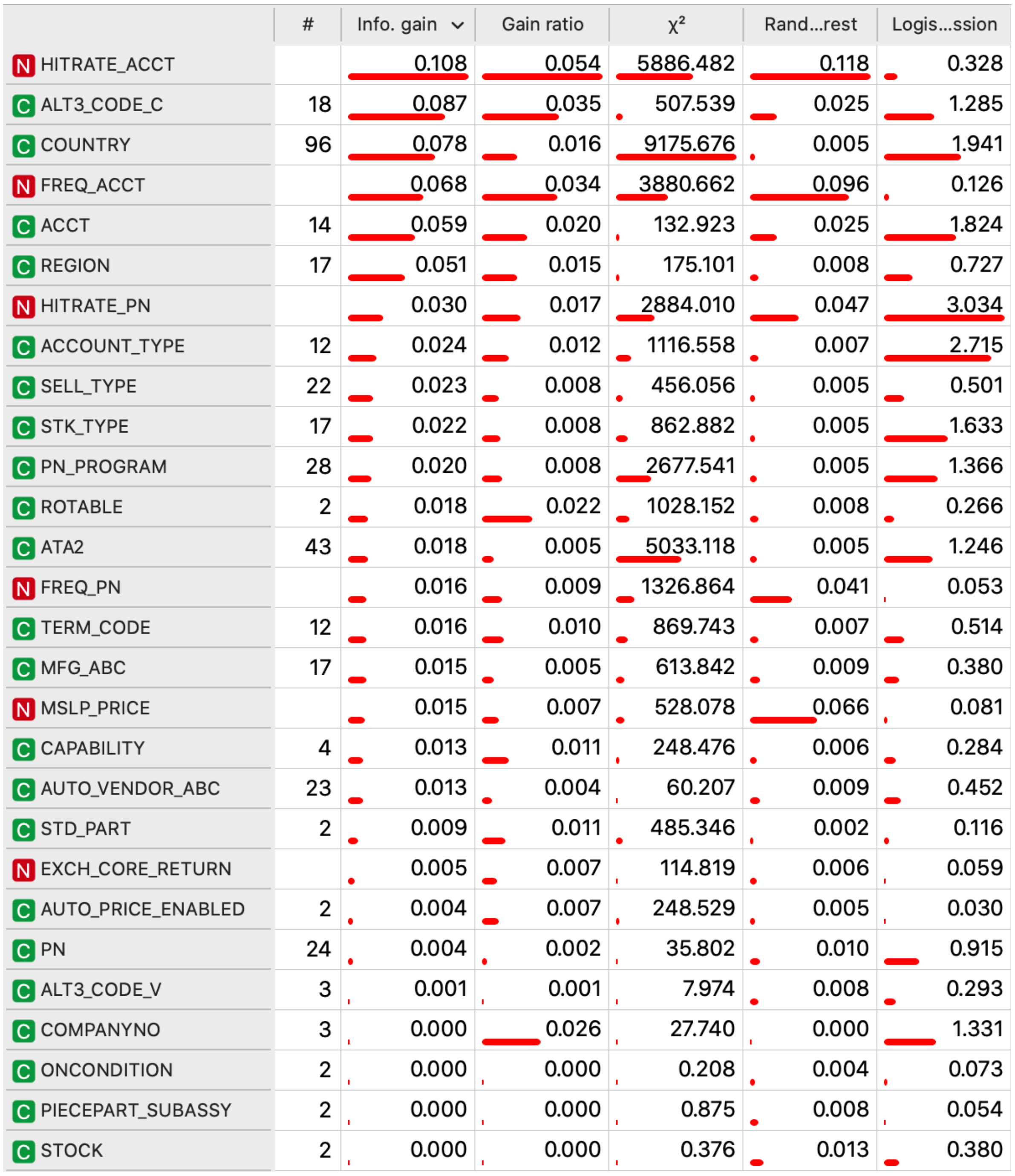
Appendix D – Validation/verification concept prioritization tool.

This Appendix is left out because of confidentiality reasons.

Appendix E – Improving the prototype prioritization tool

E.1 Feature selection sub-methodology

The feature ranking results of the machine learning data set are shown below in Figure 56, from which further investigation of the top 15 features for each filter method/classifier followed.



	#	Info. gain	Gain ratio	χ²	Rand...rest	Logis...ssion
N HITRATE_ACCT		0.108	0.054	5886.482	0.118	0.328
C ALT3_CODE_C	18	0.087	0.035	507.539	0.025	1.285
C COUNTRY	96	0.078	0.016	9175.676	0.005	1.941
N FREQ_ACCT		0.068	0.034	3880.662	0.096	0.126
C ACCT	14	0.059	0.020	132.923	0.025	1.824
C REGION	17	0.051	0.015	175.101	0.008	0.727
N HITRATE_PN		0.030	0.017	2884.010	0.047	3.034
C ACCOUNT_TYPE	12	0.024	0.012	1116.558	0.007	2.715
C SELL_TYPE	22	0.023	0.008	456.056	0.005	0.501
C STK_TYPE	17	0.022	0.008	862.882	0.005	1.633
C PN_PROGRAM	28	0.020	0.008	2677.541	0.005	1.366
C ROTABLE	2	0.018	0.022	1028.152	0.008	0.266
C ATA2	43	0.018	0.005	5033.118	0.005	1.246
N FREQ_PN		0.016	0.009	1326.864	0.041	0.053
C TERM_CODE	12	0.016	0.010	869.743	0.007	0.514
C MFG_ABC	17	0.015	0.005	613.842	0.009	0.380
N MSLP_PRICE		0.015	0.007	528.078	0.066	0.081
C CAPABILITY	4	0.013	0.011	248.476	0.006	0.284
C AUTO_VENDOR_ABC	23	0.013	0.004	60.207	0.009	0.452
C STD_PART	2	0.009	0.011	485.346	0.002	0.116
N EXCH_CORE_RETURN		0.005	0.007	114.819	0.006	0.059
C AUTO_PRICE_ENABLED	2	0.004	0.007	248.529	0.005	0.030
C PN	24	0.004	0.002	35.802	0.010	0.915
C ALT3_CODE_V	3	0.001	0.001	7.974	0.008	0.293
C COMPANYNO	3	0.000	0.026	27.740	0.000	1.331
C ONCONDITION	2	0.000	0.000	0.208	0.004	0.073
C PIECEPART_SUBASSY	2	0.000	0.000	0.875	0.008	0.054
C STOCK	2	0.000	0.000	0.376	0.013	0.380

Figure 56: Feature ranking using Orange datamining suite.

Next, Table 22-Table 24 show the F1 performance of different classifiers when top ranked features, according to multiple (different) filter methods, are incrementally added.

Gradient boosting classifier F1					
# Features	**Inf Gain**	**Gain ratio**	**Log reg**	**Chi-square**	**Random Forest**
28	48,89%	48,89%	48,89%	48,89%	48,89%
15	46,27%	46,56%	46,18%	46,87%	48,72%
14	46,27%	46,56%	43,96%	46,87%	48,75%
13	46,06%	46,05%	43,72%	46,39%	48,84%
12	45,96%	46,03%	43,98%	45,99%	48,57%
11	45,62%	45,97%	43,86%	46,20%	48,62%
10	45,70%	46,17%	44,15%	45,85%	48,63%
9	45,64%	46,00%	44,04%	46,07%	48,52%
8	45,79%	45,87%	43,74%	45,85%	48,09%
7	45,74%	45,48%	42,58%	45,84%	46,37%
6	45,29%	45,05%	42,58%	45,46%	46,31%
5	45,45%	45,20%	42,62%	45,53%	45,85%
4	45,03%	44,74%	41,95%	45,27%	45,21%
3	44,77%	44,74%	41,98%	44,88%	44,48%
2	43,60%	43,60%	35,85%	44,63%	44,48%
1	42,91%	42,91%	8,26%	38,77%	42,91%

Table 22: Monitoring performance (F1 score) when incrementally adding top ranked features according to different filter methods for Gradient Boosting classifier.

Random Forest F1					
# Features	**Inf Gain**	**Gain ratio**	**Log reg**	**Chi-square**	**Random Forest**
28	54,14%	54,14%	54,14%	54,14%	54,14%
15	52,82%	52,78%	49,76%	53,31%	53,48%
14	52,64%	52,02%	47,80%	53,42%	53,82%
13	53,01%	52,46%	46,75%	52,55%	53,55%
12	52,53%	52,53%	46,39%	52,13%	52,60%
11	51,87%	53,01%	46,22%	52,75%	52,57%
10	51,96%	52,41%	46,03%	52,42%	52,51%
9	52,22%	52,44%	45,69%	51,73%	52,91%
8	52,03%	51,81%	44,67%	51,67%	52,64%
7	52,15%	50,63%	43,96%	51,42%	50,87%
6	51,67%	50,98%	43,96%	51,63%	49,44%
5	51,52%	50,38%	43,51%	51,60%	47,68%
4	51,61%	49,48%	42,33%	52,03%	48,38%
3	45,70%	49,43%	41,83%	46,41%	46,88%
2	43,66%	43,66%	35,60%	42,72%	46,11%
1	37,09%	37,09%	18,28%	39,34%	37,09%

Table 23: Monitoring performance (F1 score) when incrementally adding top ranked features according to different filter methods for Random Forest classifier.

Logistic regression F1					
# Features	Inf Gain	Gain ratio	Log reg	Chi-square	Random Forest
28	46,80%	46,80%	46,80%	46,80%	46,80%
15	43,92%	44,32%	46,05%	44,15%	44,75%
14	43,84%	44,13%	43,98%	44,15%	44,77%
13	44,03%	44,06%	43,77%	43,05%	44,84%
12	43,49%	44,05%	44,43%	43,02%	44,84%
11	43,33%	44,08%	44,29%	43,19%	44,52%
10	43,33%	44,10%	44,18%	42,46%	44,32%
9	43,23%	43,15%	43,62%	42,37%	43,74%
8	43,44%	42,98%	43,26%	41,99%	43,74%
7	43,22%	40,71%	42,40%	42,60%	41,90%
6	43,22%	40,71%	42,40%	42,63%	41,40%
5	42,81%	39,55%	42,24%	42,35%	41,47%
4	42,08%	39,46%	41,55%	42,36%	40,82%
3	41,61%	39,40%	41,11%	41,28%	40,82%
2	38,93%	38,93%	33,29%	40,29%	40,03%
1	42,16%	42,16%	22,51%	39,32%	42,16%

Table 24: Monitoring performance (F1 score) when incrementally adding top ranked features according to different filter methods for Logistic Regression classifier.

Finally, for the optimal cut-off point per classifier, features were excluded one-by-one from highest to lowest ranked and permanently excluded when exclusion increased performance, thereby eliminating redundancy/noise.

Gradient Boosting Classifier - Ranked by Random Forest				
Feature	Rank	F1 score all features	F1 score feature excluded	Difference
HITRATE_ACCT	1	48,84%	47,61%	-1,23%
FREQ_ACCT	2	48,84%	48,42%	-0,42%
MSLP_PRICE	3	48,84%	48,06%	-0,78%
HITRATE_PN	4	48,84%	48,74%	-0,10%
FREQ_PN	5	48,84%	48,62%	-0,22%
ACCT	6	48,84%	48,72%	-0,12%
ALT3_CODE_C	7	48,84%	48,30%	-0,54%
STOCK	8	48,84%	47,52%	-1,32%
PN	9	48,84%	48,56%	-0,28%
AUTO_VENDOR_ABC	10	48,84%	48,73%	-0,11%
MFG_ABC	11	48,84%	48,66%	-0,18%
ROTABLE	12	48,84%	48,82%	-0,02%
REGION	13	48,84%	48,57%	-0,27%

Table 25: Eliminating noise/redundancy by excluding features one-by-one from highest- to lowest rank for the optimal cut-off point of Gradient Boosting classifier.

Random Forest Classifier - Ranked by Random Forest				
Feature	**Rank**	**F1 score all features**	**F1 score feature excluded**	**Difference**
HITRATE_ACCT	1	53,82%	51,37%	-2,46%
FREQ_ACCT	2	53,82%	51,37%	-2,45%
MSLP_PRICE	3	53,82%	53,00%	-0,82%
HITRATE_PN	4	53,82%	54,26%	0,44%
FREQ_PN	5	54,26%	54,03%	-0,23%
ACCT	6	54,26%	53,62%	-0,64%
ALT3_CODE_C	7	54,26%	53,33%	-0,93%
STOCK	8	54,26%	52,01%	-2,25%
PN	9	54,26%	53,98%	-0,28%
AUTO_VENDOR_	10	54,26%	53,62%	-0,64%
MFG_ABC	11	54,26%	54,07%	-0,20%
ROTABLE	12	54,26%	53,85%	-0,42%
REGION	13	54,26%	53,08%	-1,18%
ALT3_CODE_V	14	54,26%	54,25%	-0,01%

Table 26: Eliminating noise/redundancy by excluding features one-by-one from highest- to lowest rank for the optimal cut-off point of Random Forest classifier.

Logistic Regression Classifier - Ranked by Logistic Regression				
Feature	**Rank**	**F1 score all features**	**F1 score feature excluded**	**Difference**
HITRATE_PN	1	46,05%	46,05%	0,00%
ACCOUNT_TYPE	2	46,05%	45,85%	-0,20%
COUNTRY	3	46,05%	45,38%	-0,67%
ACCT	4	46,05%	45,78%	-0,27%
STK_TYPE	5	46,05%	46,04%	-0,01%
PN_PROGRAM	6	46,05%	46,12%	0,07%
COMPANYNO	7	46,12%	46,11%	-0,01%
ALT3_CODE_C	8	46,12%	45,52%	-0,60%
ATA2	9	46,12%	45,63%	-0,50%
PN	10	46,12%	45,70%	-0,43%
REGION	11	46,12%	46,14%	0,01%
TERM_CODE	12	46,14%	46,05%	-0,09%
SELL_TYPE	13	46,14%	46,69%	0,55%
AUTO_VENDOR_ABC	14	46,69%	46,31%	-0,38%
STOCK	15	46,69%	44,34%	-2,35%

Table 27: Eliminating noise/redundancy by excluding features one-by-one from highest- to lowest rank for the optimal cut-off point of Logistic Regression classifier.

E.2 Hyper-parameter tuning

param_n_estimators	param_min_samples_split	param_min_samples_leaf	param_max_features	param_max_depth	param_criterion	param_bootstrap	rank_test_score
300	5	1	11		gini	FALSE	1
300	5	1	10		entropy	FALSE	2
400	5	1	8	50	entropy	FALSE	3
150	2	1	12	40	gini	TRUE	4
500	2	1	9		entropy	TRUE	5
700	2	1	4		entropy	TRUE	6
100	2	1	4	50	gini	TRUE	7
75	10	1	12	40	entropy	FALSE	8
600	10	1	12	40	gini	FALSE	9
400	10	1	11	40	entropy	FALSE	10
150	2	1	11	30	gini	TRUE	11
200	5	1	7	50	entropy	TRUE	12
100	10	1	4	50	entropy	FALSE	13
400	10	1	2		entropy	FALSE	14
75	15	1	12	50	gini	FALSE	15
75	15	1	11	50	entropy	FALSE	16
400	2	2	12	50	gini	FALSE	17
500	20	1	13		entropy	FALSE	18
150	5	1	1	40	gini	FALSE	19
600	10	1	8	40	gini	TRUE	20

Table 28: Top 20 configurations resulting from exploratory random search cross validation (k=3) of 250 iterations.

param_n_estimators	param_min_samples_split	param_min_samples_leaf	param_max_features	param_max_depth	param_criterion	param_bootstrap	rank_test_score
400	2	1	13		gini	FALSE	1
150	2	1	13	50	gini	FALSE	2
300	2	1	13		entropy	FALSE	3
75	2	1	13		entropy	FALSE	4
150	2	1	13	40	gini	FALSE	5
100	2	1	12	40	gini	FALSE	6
400	2	1	13		entropy	FALSE	7
200	2	1	13		entropy	FALSE	8
300	2	1	13	40	entropy	FALSE	9
400	2	1	12		gini	FALSE	10
150	2	1	13	50	entropy	FALSE	11
200	2	1	12	50	gini	FALSE	12
75	2	1	13	40	entropy	FALSE	13
75	2	1	12	50	gini	FALSE	14
300	2	1	11	40	gini	FALSE	15
150	2	1	11	50	entropy	FALSE	16
75	2	1	11		gini	FALSE	17
100	2	1	13		entropy	FALSE	18
150	2	1	10		gini	FALSE	19
75	2	1	12		entropy	FALSE	20
200	2	1	12	40	entropy	FALSE	21

Table 29: Top 20 configurations resulting from targeted random search cross validation (k=3) of 500 iterations.

Appendix F – Information Extraction module.

This Appendix is left out because of confidentiality reasons.