Full Professorship
Financial Accounting

Bachelor's Thesis

for the Attainment of the Degree Bachelor of Science at the

TUM School of Management

of the Technical University of Munich

Predictive Analytics in der Produktion: Eine Fallstudie zur Unterrichtung von Machine Learning Algorithmen

Predictive Analytics in the manufacturing environment: A case of teaching machine learning algorithms

Examiner:	Prof. Dr. Jürgen Ernstberger Full Professorship Financial Accounting
Person in Support:	Dominik Fischer, M. Sc.
Course of Study:	B.Sc. Management & Technology
Submitted by:	Daniel Schroter Schleißheimerstraße 88 80797 München Matriculation Number: 03676544
Submitted on:	21.12.2018

Abstract

The goal of this thesis is to develop a case study for teaching predictive analytics using machine learning algorithms. In times of industry 4.0, artificial intelligence and big data the workforce needs to be well educated in applying those technologies. Within this environment, a leading German manufacturing company faced the challenge of predicting whether a produced part fails the internal quality control. This would enable delivering high-quality products to the end user at lower costs. The target group of the developed case are students with a business background. Therefore, I give weight to certain focal points. The first one is the process of solving those challenges utilising a standard process model for data mining (CRISP DM). The second one is the application of different techniques from the field of machine learning. Finally, I attach weight to the demonstration of how business and statistical knowledge together are needed to improve the performance of a model. This study does neither aim at developing an ideal model for a given dataset nor to provide the mathematical foundations of the applied techniques. Within the resulting case, I present a holistic overview of developing a predictive model for a real-world problem within given constraints. This includes the application of creativity, inventiveness and the need for compromises during the data mining process. With this purpose I recommend teaching the developed case. It is relevant for both academic researches teaching Big Data cases and for decision makers dealing with the topic of predictive analytics.

Table of content

List of abbreviations

CRISP DM	Cross-Industry Standard Process for Data Mining
mcc	Matthews correlation coefficient
t-SNE	t-distributed stochastic neighbour embedding
PCA	principal component analysis
OOB	Out of Bag
TP	True positive
TN	True negative
FP	False positive
FN	False negative
TAP	Total actually positive
TAN	Total actually negative
TPP	Total predicted positive
TPN	Total predicted negative
GB	Gigabyte
GHz	Gigahertz
CPU	Central Processing Unit
NA	not available (=missing value in R)

List of figures

List of tables

1. Introduction

The exponential growth in available data from sensors and increased processing capabilities offer manufacturing firms new opportunities. To gain strategic advantages Industry 4.0 strives for the exploitation of Big Data and the integration of the results into business processes. The optimisation of manufacturing processes plays a vital role in industrial enterprises.

According to a study conducted by the World Economic Forum (2017) do companies need to understand the new technologies of the 4th industrial revolution to remain competitive.[1] They developed a "Production technology radar" to keep track of over 60 technologies impacting the production systems. Managers, who rapidly embrace these technologies and transform their enterprises lay the foundations for success. Core concepts that should be assessed and adopted in nowadays production environment include big data, data mining and artificial intelligence as a key technology. However, the full potential of many of these technologies is yet not being used. Unlocking their value largely depends, besides other factors, on the education of the necessary skilled workforce.[2]

Hence the goal of this thesis is to develop a case study for teaching some of the above-mentioned technologies, namely predictive analytics and machine learning algorithms. The case is developed by means of real-world data from the Robert Bosch GmbH. Teaching a case using real-world data usually requires all steps from data pre-processing to evaluation, which favours the demonstration of the whole data mining process. The problem at hand is furthermore from a real-world scenario which can lead to insights that cannot be provided by a standardised sample case. These insights are assumed to be valuable to prepare students for challenges in their careers. With those expected characteristics I develop a case with real-world data and discuss its advantages and disadvantages compared to teaching a case with sample data.

After pointing out the relevance of the developed case, I introduce the context of the data mining project, what precisely is developed and how this challenge can be approached.

[1] See World Economic Forum ("Publisher") (2017) p.4-7
[2] See World Economic Forum ("Publisher") (2017) p.4-7

Industry 4.0 has still not found to a corresponding definition in academic literature. It refers to the so-called 4th industrial revolution and is shaped by an initiative of the German government.[3] Related terms used in academic literature include smart manufacturing, intelligent manufacturing or smart factories.[4] However, the German government defines Industry 4.0 as the intelligent connection of machines and processes in the industry with the aid of information and communication technology.[5] In other words, a variety of technologies, including machine learning and predictive analytics, enable the optimisation of the production environment. An underlying key capability is the handling of large amounts of data.[6] Besides several objectives industry 4.0 aims at enabling the production of highly individualised products in flexible mass production processes.[7] Within this environment, Bosch, a leading German manufacturing company, faced the challenge of predicting whether a produced part will fail the internal quality control. Bosch is a company that develops and produces a variety of technical parts for different domains. In the interest of delivering high-quality products to the end user at lower costs, they seek for a model to predict those internal failures.[8]

This leads us to the relevant concepts of data mining, predictive analytics and machine learning. As data mining is used as a buzzword, several definitions exist.[9] Larose and Larose (2015) define data mining as the "process of discovering useful patterns and trends in large datasets".[10] Some useful patterns might lead to the ability to make predictions, which is referred to as predictive analytics.[11] Extracting information from the data sets is done by data mining algorithms. According to Kotu and Desphande (2015) does the application of sophisticated algorithms to extract those patterns differentiate data mining from traditional data analysis techniques.[12] Many of the algorithms used for predictive analytics are borrowed from the field of machine learning. Its definition highlights the close relationship of predictive analytics and machine learning. In particular, machine learning is defined as a "set of methods that can automatically detect patterns in data, and

[3] See Roth (2016) p.5
[4] See Thoben et al. (2017)
[5] See Bundesministerium für Wirtschaft und Energie (2018)
[6] See Pereira and Roero (2017)
[7] See Thoben et. al. (2016)
[8] See Bosch ("Publisher") (2016)
[9] See Kotu and Desphande (2015) p.2
[10] See Larose and Larose (2015) p. 4
[11] See Larose and Larose (2015) p. 4
[12] See Kotu and Desphande (2015) p. 4-5

then use the uncovered patterns to predict future data, or to perform other kinds of decision making under uncertainty."[13] Hence the task of predicting internal failures can be addressed by machine learning techniques. One major type of machine learning is called supervised learning. Supervised learning aims at predicting a certain target variable (e.g. failure vs no failure). To do so, a model is built based on training data, where this variable is known. If the target variable embraces categorical values the challenge of predicting those values is called "classification" task.[14] With these definitions at hand, we can describe our task as training a classification model to predict whether a part will fail quality control or not.

To structure the process of developing the model, a standard process model for data mining is applied. Several frameworks exist to support the process of data mining.[15] The most popular are SEMMA and CRISP-DM. According to Palacios et al. (2017) does CRISP-DM have advantages when it comes to a detailed description of the required tasks.[16] Within the scope of this study, it seems therefore reasonable to follow the CRISP-DM model.

The study is structured into three main parts. First the theoretical introduction of the CRISP DM model, which is used as methodology. Afterwards, I lay the theoretical foundation for the applied statistical techniques. The third main part is the application of the CRISP DM Model to the real-world dataset. The primary target group for teaching the developed case are students with a business background. Hence, I give weight to certain focal points. I provide a holistic impression of solving real-world data mining problems utilising a standard process model (CRISP DM). Furthermore, different techniques from the field of machine learning are applied. Finally, I attach weight to the demonstration of how business and statistical knowledge together can be used to improve the performance of a model. The study does not aim at developing an ideal model for a given dataset, but to present a holistic overview of developing a predictive model within given constraints. This includes the application of creativity, inventiveness and the need for compromises during the data mining process.

[13] Murphy (2012) p.1
[14] See Murphy (2012) p. 2
[15] See Kurgan and Musilek (2006)
[16] See Palacios et al. (2017)

2. Methodology: A process model for data mining - CRISP DM

2.1. The CRISP DM: An Introduction

In the late 1990s, the Cross-Industry Standard Process for Data Mining (CRISP DM) was conceived by three leading companies in the field of Data Mining. Specialists from the Daimler AG, SPSS Inc. and NCR corporation saw the need for a standardised process model to face the challenges in the young data mining market. In the following years, they developed and validated a solid process model based on their practical experience.[17] The model is industry and application independent to cover most data mining projects. It should be understood as a framework, that must be adopted to the concrete situation. The data mining process is split into six phases representing the life cycle of a data mining project (Figure 1). Each phase consists out of several tasks, which can usually be executed in varying order and multiple times.[18] The sequence of phases is not fixed but referred to as iterative and adaptive.[19] Arrows illustrate the most significant and frequent dependencies between the phases.

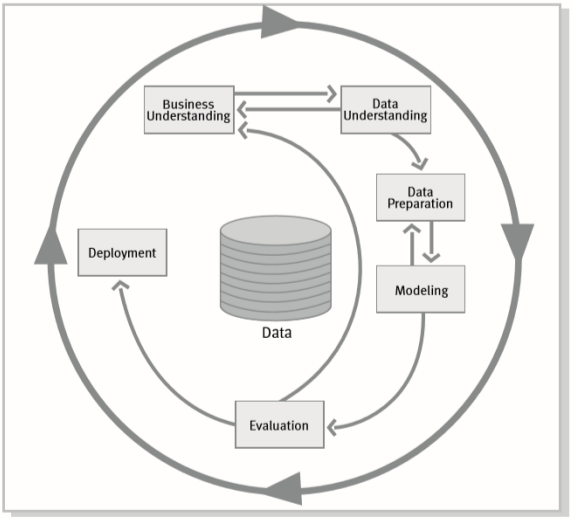


Figure 1: Phases of CRISP-DM

Each phase highly depends on the outcome of a previous phase. The outer circle represents the iterative nature of data mining itself. It indicates the transition of experiences from past data mining projects into following ones.[20] In the following section, we have a close look at the tasks demanded by each phase.

[17] See Chapman et al. (2000), p.1-2
[18] See Chapman et al. (2000), p.6
[19] See Larose and Larose, (2015), p.6
[20] See Chapman et al. (2000), p. 10

2.2. Phases and tasks of the CRISP DM model

Phase 1: Business Understanding

According to Shaerer (2000), business understanding is probably the most critical phase of any data mining project.[21] To deliver a successful project, it is vital to understand the project's objectives from a business or research perspective. This phase comprises the tasks: *Determine business objectives, assess the situation, determine data mining goals* and *produce a project plan*.

To *determine business objectives,* the data analyst must understand the real goal of the proposed project from a business perspective. This step is crucial to avoid that the project produces the right answers to the wrong questions. The business objectives should be related to concrete measures of success. In a next step, he a*ssesses the situation.* This task includes the investigation of available personnel, data, software and computational resources. He clarifies requirements such as a schedule of completion, legal constraints and other risks. A brief cost-benefit analysis is usually conducted to decide whether to proceed with the project.[22] Success should not only be defined from a business perspective but also in technical terms. Therefore, the analyst *determines data mining goals.* They describe the intended output of the project, which enables the achievement of the business objectives. The optimisation of a specific data mining metric is such a goal, for instance achieving a certain level of predictive accuracy.[23] Based on the information gathered so far, the analyst *produces a project plan.* The project plan should determine the steps needed to achieve the data mining goal and thereby the business objectives. The plan should be part of the communication within the team and accessible to stakeholders.[24]

Phase 2: Data Understanding

The Data understanding phase starts with the *initial data collection*. Other tasks include the *description of data, exploration of data* and the *verification of data quality*. In this phase, the data analyst gets familiar with the data, gains first insights about hidden information and impressions about data quality.[25]

---

[21] See Shaerer (2000)
[22] See Chapman (2000)
[23] See Shaerer(2000)
[24] See Lesmeister (2017), Producing a Plan
[25] See Shaerer(2000)

The analyst c*ollects the initial data.* This includes loading and integrating the data into the analytics tool. Occurring problems should be reported, to improve future replications.[26] He then *describes the data.* During this step, the "surface" of the data is investigated. The analyst answers questions about the number of columns and records available, the format of the data or about the features (variables) assumed to be important to solve the problem. In this task, he already achieves a basic understanding of the data.[27] To e*xplore the data,* he already tackles the data mining question. First queries and data visualisations are created. The goal is to gain first findings, initial hypothesis and impressions about the potential impact on the remainder of the project. Finally, the analyst *verifies data quality* and answers questions about missing values and sparsity. Are there features conflicting with common sense, ambiguous or misspelt? Part of this task is to discover possible data quality issues and recommend solutions.[28]

Phase 3: Data Preparation

Data Preparation incorporates the actions which are required to prepare the data to feed it into models. As different models need different formats of data this phase is closely linked to the modelling phase and is executed multiple times. The analyst approaches feature engineering and creates train and test datasets.[29] The data preparation phase has the following five tasks: *Selecting data, cleaning data, construct data, integrate data* and *format data*.[30]

The data scientist *selects the data* for the analysis. He decides on several criteria including relevance to data mining goals, quality limits and technical constraints such as data volume. The decisions for including or excluding data should be explained and includes the selection of rows and columns in a table. Moreover, it is relevant to decide whether some attributes are more relevant than others.[31] C*leaning the data* is crucial for the model's performance. During this task, he addresses the reported data quality issues. The analyst can choose clean subsets of data or estimate missing data. He further c*onstructs data,* which includes the conception of new features. A derived feature would be something like area = length * width. The creation of new derived features usually

[26] See Shaerer(2000)
[27] See Chapman et al. (2000), p. 18
[28] See Shaerer (2000)
[29] See Lesmeister (2017), Data Preparation
[30] See Shaerer (2000)
[31] See Chapmann (2000), p. 21

requires domain knowledge. They should only be added, when they facilitate the modelling algorithm. Values of existing records are transformed if necessary. Within the next task he describes how to *integrate the data.* Information from multiple tables is combined. The analyst can aggregate information. For instance, tables that have a record for each purchase can be aggregated to a table with one record per customer.[32] Finally, he *formats the data* to fit the requirements of the model. This refers to syntactical changes. Some tools require a specific order of the attributes. For instance, a unique identifier in the first column and the label of interest in the last column.[33]

Phase 4: Modelling

During the Modelling phase, various models are selected, applied and modified. The parameters are tuned to provide the best possible results. Often the same data mining problem can be addressed via multiple modelling techniques. The techniques might have different requirements for the structure of the data, which would indicate a step back to the data preparation phase. Modelling consists out of the four tasks: *Select Modelling Technique, Generate Test Design, Build Model* and *Assess Model*.[34]

First, the analyst s*elects the modelling techniques*. Their concrete requirements on the data should be recorded.[35] Before building the model, one should g*enerate a test design.* This includes the definition of mechanisms to test the model's quality and validity. In supervised data mining projects, the dataset is usually split into train and test sets. This decision might require data preparation steps again.[36] Now the analyst is ready to *build the model* by running them on the prepared datasets. Most of the modelling tools have many parameters. The data mining engineer lists the parameters and explains certain values. To complete this phase, he *assesses the model.* He interprets the results according to his domain knowledge and judges the success of the model in technical terms. The models are ranked according to the evaluation criteria, and differences in performance are discussed. Together with domain experts and business analysts, they put the results into business context. The focus of this task is on the models. Other outcomes of the project are evaluated in the evaluation phase.

[32] See Shaerer (2000)
[33] See Chapman et al. (2000) p. 22-23
[34] See Shaerer (2000)
[35] See Chapman (2000) p.24
[36] See Chapman (2000) p.24

Phase 5: Evaluation

The evaluation phase describes the activities of evaluating the model against business issues and reviewing the process of creating the model. It is necessary to discuss if there are crucial business issues, that were not considered. The project leader decides how precisely the results should be used. The evaluation consists out of the tasks: *evaluate results, review process* and *determine next steps*.[37]

To *evaluate the results*, the analyst checks whether the model meets the business objectives and if there are some reasons for deficiency. If time and budget constraints permit the model can be checked in a real-world-application test. Additional challenges, information and hints for the future directions are stated. The analyst makes a final statement on whether the project meets the initial business objectives.[38] He carefully *reviews the process* by checking if the models were built correctly or if there are tasks that have been overlooked.[39] Afterwards, he *determines the next steps.* Based on the gathered insight he recommends new projects. With respect to remaining resources, he initiates further improvements or terminates the project and moves on to deployment.[40]

Phase 6: Deployment

In *deployment*, the model is prepared and implemented into organisations decision making processes. Dependent on the business requirements this phase can reach from creating a report to implementing a repeatable data mining process across the company. The demanded tasks include: *plan deployment, plan monitoring and maintenance, produce final report* and *review project*.[41]

During the "*plan deployment"* task, a concrete strategy for deployment is determined. The team moreover develops a *plan for monitoring and maintenance*. Monitoring and maintaining the model ensures the correct use of the results within the day-to-day business. The project leader or project team *produces a final report.* Dependent on the situation, this report can vary between summarising the project and its experience and creating a comprehensive presentation of the results. Finally, they *review the project* and reflect on failures and success in certain situations to improve future projects.[42]

---

[37] See Shaerer (2000)
[38] See Shaerer (2000)
[39] See Chapman (2000) p. 27
[40] See Chapman (2000) p. 27
[41] See Shaerer (2000)
[42] See Chapman (2000) p. 29

3. Theoretical foundations of applied statistical methods

During the application of the CRISP DM Model, we use several statistical techniques. The selection of those techniques was based on the given dataset. It is therefore already a result of applying the process model. However, to introduce the theoretical foundation of the applied techniques, they should be presented together in this section. I cover the theoretical concepts used for data exploration, data preparation, modelling and model assessment. Explanations of the underlying mathematical details would exceed the scope of this study. Thus, I provide an intuitive understanding of the applied techniques.

3.1. Data exploration with t-SNE

Van der Mateen and Hinton (2008) suggests visualising high-dimensional datasets using t-Distributed Stochastic Neighbour Embedding (t-SNE).[43] The method reduces the dimensionality of a dataset and enables the visualisation in a two- or three-dimensional space. It aims at placing similar points in the high dimensional space close to each other in the lower dimensional space. In a first step, the similarity between the observations in the high dimensional space is represented by a certain probability distribution. Afterwards, a second distribution in the low dimensional space is chosen.[44] To find such a second distribution in the low dimensional space, the dissimilarity to the distribution in the high dimensional space is minimised (Kullback-Leibler Divergence).[45] However, t-SNE has a quadratic runtime which makes it difficult to compute if the number of records exceeds for instance 10 000 observations.[46]

3.2. Data preparation with principal component analysis

Principal component analysis (PCA) aims at extracting the underlying principal components of a dataset. A principal component is a linear combination of the original variables. If those variables are highly correlated, they can be consolidated to components. Hence the technique seeks to explain the correlation structure in a set of predictor variables. The different components are built based on the variables containing the highest variability. Those fewer components can often explain a large fraction of the variance within a dataset. Hence the dimensions can be reduced without losing much

[43] See Van der Mateen and Hinton (2008)
[44] See Boschetti and Massaron (2016) t-sne
[45] See Polani (2013) p. 1087-1088
[46] See Van der Mateen and Hinton (2008)

information. The new set of components has some essential properties. The components are uncorrelated to each other. They might explain a large amount of variance within the data and they can be traced back to the original variables.[47]

3.3. Modelling: Important terms and modelling techniques

Several techniques can solve a binary classification task such as predicting failure.[48] During the application of the CRISP DM model, I selected logistic regression and tree-based models. The rationale why specifically those models were applied can be found in section 4.4.1. However, the following sections provide the theoretical foundation for the applied machine learning algorithms.

3.3.1. Important terms for modelling

During modelling, we use some terms that should be understood. *Heteroscedasticity* is the phenomenon of error terms with unequal variances.[49] A *residual* is the difference between the observed and the predicted value.[50] *Overfitting* means that the model is fitted too strong to the training dataset. This leads to poor performance when predicting new data. *Multicollinearity* is the phenomenon, when some of the variables are correlated to each other. This can lead to problems within some algorithms. The *loss function* measures the difference between the model's predicted values and the actual values.[51] There are several different loss functions for different types of problems. The logistic loss function is suitable for binary classification.[52] Usually the objective is to minimise the error from false predictions. Hence there should be simple derivatives of the loss function. For more complex models the derivative of the loss function gets more complicated too. Therefore, solutions are needed to approximate them with iterative methods. One of these methods is the gradient descent method. I omit technical details here but recommend Meister (1999) to the interested reader.[53] To evaluate the model's performance one should understand the concept of *k-fold-cross-validation.* The dataset is divided into k equally sized groups (folds). One of the folds is treated as a validation set whereas the remaining (k-1) folds are used for training the model. Cross-validation is applied to ensure that the

[47] See Kotu and Desphande (2015) p.350
[48] See Lesmeister (2017) Algorithms Flow Chart
[49] See James et al. (2018) p.95
[50] See James et al. (2018) p.62
[51] See Cakmak (2018), Loss and error functions
[52] See Goreman (2017)
[53] See Meister (1999) p.555-556

results of the model are generalizable to an independent, unseen dataset.[54] This process is repeated k times and each time a different fold is used as a validation set.[55]

3.3.2. Logistic regression and tree-based models

Logistic Regression

Kotu and Desphande (2015)[56] describe the logistic regression as a process of obtaining an appropriate nonlinear curve to fit the data. In contrast to linear regression, where the values of the target variable often are continuous, logistic regression works for categorical and especially binary variables (e.g. failure vs no failure). The logistic regression fits a sigmoidal (S-shaped) curve to the data. The curve should classify the data points into two categories: 0 (e.g. no failure) and 1 (e.g. failure). The value range of the curve itself is in between 0 and 1 and may be interpreted as a probability.[57] For a binary variable, we can understand the distance between the curve and 0 as the probability that the observation belongs to the class labelled with 0. To achieve linearity, continuity and a value range from negative infinity to positive infinity the logistic regression is often transformed with a mathematical transformation called "logit". For further interpretations and mathematical details, the reader may refer to Larose and Larose (2015).[58] In contrast to linear regression does the logistic regression not need normally distributed residuals and can better deal with heteroscedasticity. A high degree of multicollinearity and hugely differing feature scales should be avoided.[59]

Decision Tree

In the following section, I will briefly explain the basics of a decision tree for binary classification. A decision tree consists out of decision nodes, branches and leaf nodes. A decision tree is comparable to a decision flow chart. At each decision node, an attribute is tested. The result decides whether to follow the left or right branch to the next decision node. For instance, if value "a" is larger than 0.5 then take the left branch, otherwise, take the right branch. This process is repeated until a leaf node, and its belonging class (e.g. damaged or not damaged) is reached. Each node splits the data into subsets. A split should

[54] See Larose and Larose (2015) p.161
[55] See James et al. (2018) p.181
[56] See Kotu and Desphande (2015) p.182
[57] See Larose and Larose (2015) p.363
[58] See Larose and Larose (2015) p. 362
[59] See Miller and Forte (2017), Assumptions of logistic regression

create subsets, where each record has the same class. This is referred to as purity (Gini Index[60]). The attributes that enable a split with as pure subsets as possible are selected first, when building the tree. If for instance feature "a" perfectly separates a class of observations into damaged and not damaged, it is chosen first. This process is repeated until all the subsets are pure. That is why decision trees provide information about feature importance. Features that have been chosen early are assumed to be more important.[61]

In contrast to regression models are decision trees good in handling complex non-linear relationships between the features and the response variable (class).[62] They are easy to explain to people with no technical background. According to James (2018) do some people say that they more closely represent human decision making compared to other techniques (e.g. logistic regression).[63] Trees can easily be visualised and interpreted. On the other hand, they often do not reach the same level of predicting accuracy as other approaches. Small changes to the data can result in entirely different trees.[64] This is a result of different splits at the beginning of the "growing" process due to the changes in the data. Hence Murphy (2012) describes decision trees as high variance estimators. This means that if we randomly split our data and apply decision trees to each part, the resulting trees could be entirely different. Solutions to this problem are ensemble learning methods like random-forests and boosting.[65]

Ensemble Learning

Ensemble Learning builds a strong model based on many weak models. A weak model is a model, whose predictions are better than guessing by chance. The final model is a weighted combination of the base models.[66]

Random Forest

Random Forest is an approach to achieve a predictor with low variance. The concept of *bagging* should be briefly mentioned in this context. To reduce variance one can build separate models, based on many subsets of the dataset, and average the resulting

[60] See James et al. (2018) p. 311-312
[61] See James et al. (2018) p 311-316
[62] See James et al. (2018) p.314
[63] See James et al. (2018) p.315-316
[64] See Murphy (2012) p. 550
[65] See Murphy (2012) p. 550
[66] See Miller (2017), Weak and Strong Learners

predictions.[67] The subsets are randomly chosen with replacement (=bootstrapping).[68] During this process we create a set of full-grown trees, each having high variance. Averaging the trees reduces the variance. To estimate the test error, we can use out of bag (OOB) observations. Each bagged tree is usually fit to two third of the subset. For the remaining third, the out-of-bag observations, predictions can be made. Finally, each observation has been predicted multiple times, as it appears in several OBBs. For classification problems, the resulting prediction is then the majority vote for each observation. Based on that predictions the classification error can be computed. Using the OOB error is a way of estimating the test error, without performing cross-validation.[69]

Random Forest improves bagging. Let's assume that there are one strong predictor and several moderately strong predictors. Each bagged tree might still use the strong predictor for the first split which would result in trees that look similar to each other. The prediction made by these trees would be highly correlated. Averaging correlated predictions does not lead to the same reduction in variance than averaging uncorrelated predictions. Random Forest addresses this problem, by restricting the available predictors each tree can use for a split. For one split within a decision tree, most predictors are not available. Hence the resulting trees strongly differ in their shape. Those uncorrelated predictions yield to a lower OOB error.[70]

Gradient boosted decision trees

Boosting is another way of improving the predictions of a model. In this section, we restrict ourselves to boosting in the context of decision trees. Bagging is the concept of training decision trees to various subsets of the original dataset. The construction of each tree is independent of the other trees. This is the point where boosting differs from bagging. The trees in boosting are grown sequentially. Each tree uses the information of the previously built tree. Hence the construction of each tree highly depends on the construction of previous trees. Instead of using independent subsets, the trees are constructed with modified versions of the dataset.[71] The trees are usually not trained with a focus on the response variable. They try to predict and correct the systematic errors that

[67] See James et al. (2018) p. 317
[68] See Efron (1979)
[69] See James et al. (2018) p. 317-p.318
[70] See James (2018) p. 320
[71] See James (2018) p. 321

the previous trees made. By adding these trees to an additive model, the prediction quality improves. This leads to a strong model. An advantage of boosting over random forests is that the individual trees are usually smaller because they already consider previously build trees. Adding smaller trees can improve interpretability.[72]

3.4. Model assessment: Contingency table and mcc score

Finally, we look at the theoretical foundation to evaluate a model. In this section, we look at metrics that are based on the contingency table of correct and incorrect classification (Table 2). The contingency table displays the prediction values against the actual values. This results in four categories as displayed in table 1. True positive (TP) and true negative (TN) for correctly classifying an observation as positive and negative respectively. False positive (FP) and False negative (FN) for incorrectly classifying an observation as positive and negative. Several metrics can be defined to determine the performance of an algorithm. Accuracy is defined as the sum of TN and TP divided by the total number of observations. Sensitivity and Specificity are also two common metrics. Sensitivity measures the ability to classify a record positively (TP/TAP). Specificity measures the ability to classify a record negatively (TN/TAN).[73]

		Predicted Category		
		0	1	Total
Actual Category	0	True negatives (TN): predicted: 0 actually: 0	False positives (FP): predicted: 1 actually: 0	Total actually negative (TAN)
	1	False negatives (FN): predicted: 0 actually: 1	True positives (TP): predicted: 1 actually: 1	Total actually positive (TAP)
	Total	Total predicted negative (TPN)	Total predicted positive (TPP)	Grandtotal

Table 1: Contingency table of correct and incorrect classification[74]

Several more metrics can be calculated, but they still must be used consciously. Depended on the structure of the problem and the objectives some metrics are more suitable than others. To deal with an imbalanced dataset, we can use the Matthew Correlation Coefficient (mcc).[75] The mcc score has a range from -1 to 1, where 0 represents guessing and 1 is total agreement. It is defined as:

[72] See James (2018) p. 322
[73] See Larose and Larose (2015) p.456-457
[74] See Larose and Larose (2015) p. 455
[75] See Boughorbel et al. (2017)

$$MCC = \frac{TP * TN - FP * FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \quad (1)$$

4. Application of the CRISP DM Model

The CRISP DM model is a framework and needs to be adapted to the concrete situation. The data mining process requires certain decisions. For instance, which data to use or whether to reduce dimensions. This leads to multiple pathways to deal with the dataset. The presented solution just demonstrates one way of achieving a useful model. However, the introduced mechanisms and principles are transferable. The process is illustrated by means of the production line dataset from Bosch. Due to a lack of knowledge about the circumstances of the real project, I cover the phases 1 to 5 (figure 1). The deployment phase includes a concrete implementation strategy and is therefore not further considered.

4.1. Phase 1: Business Understanding

Determine Business Objectives

Bosch is one of the world's leading manufacturing companies producing advanced mechanical components. They are interested in ensuring that their components meet quality and safety standards. To do so, they challenged data scientists within a competition on Kaggle. Kaggle is the world's largest data science and machine learning community.[76] The business objective for the Kaggle competition is described as "predicting internal failures … to enable Bosch to bring quality products at lower costs to the end user".[77] Further information about the concrete business success criteria is not available. Nevertheless, the objective of this study is to illustrate the process of data mining. In the real-world project, a concrete business success criterion could be something like the reduction of spending for quality control by 10%.

Assess the situation

The data was published by Bosch and is publicly available on Kaggle.[78] The data consists out of measurements which were made while the products move through the production line. All the code and models created during this project are computed on an RStudio

[76] See Kaggle ("Publisher") (2018)
[77] Bosch ("Publisher") (2016)
[78] See Bosch Data ("Publisher") (2016)

Server. However, they should be computable with 16 GB of Ram and a Dual Core CPU with 2.00 GHz. As this study provides the foundation for teaching a case in predictive analytics to university students the available resources are restricted to a level which could be made accessible to students. The case is developed within twelve weeks.

Determine data mining goals

The goal of Bosch is to predict whether a particular part will fail quality control. To evaluate the prediction quality, they defined the Matthew correlation score (mcc) as a key metric (see section 3.4).[79] Within the scope of this study, models that perform better than guessing are sufficiently good. Hence, an mcc of above 0.3 is defined as data mining goal.

4.2. Phase 2: Data Understanding

4.2.1. Collection and description of the data

The Bosch Data in total has a size of approximately 14.3 GB. The data is segregated into six files. Due to the large number of features (variables), the dataset was split into smaller ones according to the type of features. There are train and test datasets for numerical, categorical and date features. Within the scope of this study, only the numerical train dataset (train_numeric.csv) is used. The data was directly downloaded from Kaggle.[80]

In our dataset, each part has a unique Id (row). As we apply supervised learning, we have a labelled dataset. This means we have a feature called "Response" in our dataset indicating whether the part failed quality control or not. The features are named according to a naming convention. Each name consists out of three parts: Production line, station number and feature number. "L1_s24_F1512" indicates that the measurement was taken on production line 1, station24, and feature 1512. All columns have numeric values. In total there are 970 features (incl. Id and Response) and 1,183,747 observations. Each observation represents a part that moved through the production line.[81]

4.2.2. Data exploration: First insights and visualisation with t-SNE

First, the value range of the variables should be discovered. The response variable has the value of 1 to indicate that a part failed and 0 otherwise. The Id of the parts does not have

[79] See Bosch Evaluation ("Publisher") (2016)
[80] See Bosch Data ("Publisher") (2016)
[81] See Bosch ("Publisher") (2016)

any predictive power as its purpose is identification. The min and max values for the remaining features are calculated. Those remaining features have a numerical value range of -1 to 1. This leads to the conclusion that Bosch already transformed the data before uploading it on Kaggle. Furthermore, I counted the number of parts that failed quality control. It turns out that only 6,879 parts were damaged, which is a fraction of 0.58 % of the total number of parts.[82] Hence, we face an imbalanced dataset. "A dataset is imbalanced if the classification categories are not approximately equally represented."[83] These first insights already have quite a significant impact on the project. As the single features already are transformed, we do not need to conduct this task. The fact of dealing with a highly imbalanced dataset must be considered during the modelling phase. For instance, a model predicting that each part is working well would automatically achieve an accuracy of 99.42%.

Based on our knowledge about the naming of the features we can increase our understanding of the dataset. If a particular value is missing, the specific part probably does not have this feature. Hence the pattern of missing values might indicate some information about the part. Parts which have similar patterns are probably of the same or similar type. By setting every measured value to 1 and every missing value to 0, we get representations of the characteristic features of the part. By checking for duplicates, we can then filter for parts with the same feature structure and handle them as product groups.

Graphical methods, such as overlay histograms, to explore numeric variables as suggested by Larose and Larose (2015) are not suitable for our case. [84] We do not have any domain knowledge which directs us towards the investigation of certain features. So, we would have to evaluate 968 features, which is not possible within the given time constraints. To still explore the data, we must rely on methods, which statistically analyse the structure of the data. One method that turned out to be useful is t-SNE.

Data visualisation with t-SNE

Laurae (2016) suggests investigating patterns in the missing values of the features using t-SNE. [85] She sets the values to 1 and missing values (= "NA") to 0. Then she calculates a correlation matrix between the features using the phi coefficient. The phi coefficient is

[82] See Appendix B.2
[83] Chawla et al. (2002)
[84] See Larose and Larose (2015) p.65
[85] See Laurea (2016)

a measure for the correlation between two binary coded variables.[86] The resulting 969x969 (without Id) correlation matrix was then visualised using t-SNE.[87] As we can see in Figure 2 (Appendix: A.5), there are patterns recognisable. That means the features can be grouped according to their missing value patterns. Reflecting on our domain knowledge about the data, this seems conclusive. A station might be assigned to a certain assembly step. This assembly step can be split into smaller processes along which the measurements are taken. If a part requires this assembly step, it is likely to run through several subprocesses at this station. Figure 4 displays the sector around the response variable (pink)[88].

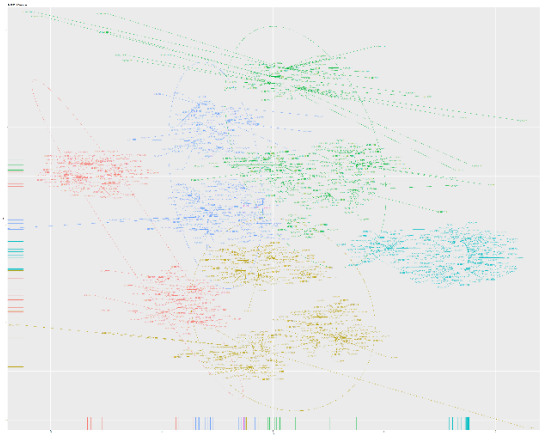


Figure 2: t-SNE overview of missing value patterns

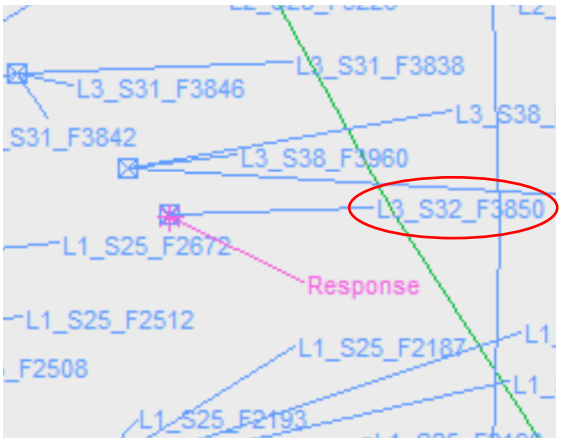


Figure 3: t-SNE sector of response

As we can see the feature L3_S32_F3850 (red) is closest located to the response variable. This is reflected by the blue and pink line pointing towards points, that are close to each other. This leads us towards having a closer look at this station. Station S32 has only one feature. Counting the number of parts that run through this station and failed reveals an interesting result. 4.51 % of the parts from station S32 failed in quality checking.[89] In contrast, only 0.58% of the total parts failed. This might indicate that station 32 is valuable when it comes to predicting failures.

4.2.3. Verification of data quality

With our limited domain knowledge, the reflection about specific values based on common sense is difficult. Therefrom we must look at the structure of the dataset to gather

[86] See Fahrmeier et al. (2016) p.132
[87] See Appendix: B.3
[88] See Laurea (2016)
[89] See Appendix B.4

insights about data quality. In the given dataset all features (excl. Id and Response) have missing values. In 82% per cent of the features, more than 70 % of the values are missing. As we have seen in the previous section, our products have characteristic missing value patterns. Hence our values are probably missing because of the relation to an underlying variable that is not included in the dataset (part-type). This phenomenon is called "*Not missing at Random*" in literature.[90] Winters (2017) suggests two way of dealing with missing values. Firstly, records that have missing values should be excluded. As every record in the dataset has missing values, this is not an appropriate solution.[91] Secondly, missing values can be imputed by the mean, median or mode of the remaining values of the variable. As most variables have more than 70 % missing values it does not seem reasonable to substitute the missing values based on calculations on only 30% of the data.[92] Data scientists who participated in the Kaggle competition suggest setting missing values to 0 and adding 2 to all other values. As the data is normalised to -1 and 1, this would increase their value range to 1 to 3. Most algorithms would be able to handle the difference between the missing and not missing values properly.[93] This sounds like a suitable solution for our problem. With respect to the evaluation metrics, it can be checked whether this solution is appropriate or not.

4.3. Phase 3: Data Preparation

In our case, we have two major points to consider in order to prepare the data. First, we have limited computational resources, which require to reduce data volume. This includes the selection of records and features. Secondly, our dataset is highly imbalanced, which needs special care in data selection. Those problems can be addressed by different techniques such as resampling or adjusting the class weights within the modelling phase.[94] I reduce the data with resampling and by filtering for a product type. The different subsets are evaluated with an initial model.[95] With respect to the initial performance, I choose the subset, with which we proceed. In a next step, we clean the data and look at a method for reducing the features.

[90] See Winters (2017) Missing Values
[91] See Appendix B.2
[92] See Winters (2017) Missing Values
[93] See Waring (2017) Missing Values
[94] See Patreek (2017)
[95] See Appendix B.7

4.3.1. Selection of rows by resampling and filtering for a product group

Select rows by resampling

"Resampling at random with replacement" is an approach, that replicates records from the minority class multiple times. The fraction of the minority class can be increased until a desired threshold.[96] One drawback is the artificial generation of additional data. In our case, we cannot compute the total dataset due to computational limitations. Therefrom generating even more records is not a proper solution to our problem. Another method is called "*Randomly Downsampling"*. This method reduces the number of records in the majority class to a certain degree, by randomly dropping records from the majority class.[97] Larose and Larose (2015) suggest that the proportion still can be relatively low (10%), if the records are sufficiently diverse.[98] A drawback of this method is that much data containing information is not used. The test data should not be balanced to represent the real-world case appropriately. [99] Therefore, we only rebalance the classes in the training datasets and evaluate their performance in an initial model.[100] The best resampled dataset achieves an mcc score of 0.17 (see Appendix A.1).

Select rows according to the product group

As described in the data understanding phase we can filter our dataset for similar parts. The information of the part type is assumed to present in the real-world application and was solely anonymised by Bosch. We create our 0-1 matrix according to the missing values and count the number of duplicates of each unique row. As this operation is computationally intensive, it is done on a randomly chosen subsample of 100,000 records. The row with the largest number of duplicates was assumed to be the largest product group. According to the missing value pattern the total dataset was filtered to collect all records of this product group. The resulting subsample has 11,915 records, with a fraction of 0.62% per cent damaged parts. It should be mentioned that the creation of the product group was computationally expensive and took around six hours.[101] The models created on this product group subsample are not capable of making good predictions for other product groups. Anyhow models for other part types could be trained following the same

[96] See Liu et al. (2006)
[97] See Liu et al. (2006)
[98] See Larose and Larose (2015) p.167
[99] See Larose and Larose (2015) p.167
[100] See Appendix B.5 and B.7
[101] See Appendix B.6

manner. Building the initial model with our product dataset produces an mcc score of 0.44 (see Appendix A.1).

Although the product dataset is not balanced, it outperforms the resampled datasets. For the purpose of this study, I proceed with the product group dataset. The process of building the model for a product group is transferable. The information about the type of a part is assumed to be present in the real-world scenario. Hence this is a creative way to stay within the computational constraints and still create well-performing models.

4.3.2. Cleaning the data

As suggested in the verify data quality section we will not conduct data imputation techniques based on mode, mean or any other advanced modelling technique. When using other datasets than the product dataset, I follow the approach recommended in the verify data quality section (4.2.3). For instance, while modelling the resampled datasets, I set the missing values to 0 and added +2 to all other values. The product subsample has the characteristic that we can easily drop the columns that do not contain values and get a subset which is free of missing values. This reduces the dimensionality from 970 to 211. The features in our dataset have already been transformed, so we do not need to conduct this task.

4.3.3. Selection of columns by reducing dimensionality with PCA

In high dimensional datasets, the number of attributes should be reduced. It is likely that some of the variables are correlated to each other (Multicollinearity). Multicollinearity can lead to an unstable solution, as we will see during the application of logistic regression.[102] The product dataset has 211 columns that contain values. By dropping those, that contain zero variance we remain with 197.[103] Techniques like PCA or t-SNE can further reduce dimensionality. As we have seen in a previous section t-SNE is useful for data visualisation. The question arises if the reduced dimensions can be used as predictive variables. T-SNE cannot reduce the dimension of new data points.[104] To include new data into the model, the whole t-SNE must be conducted again.[105] In our case want to predict whether recently produced parts will fail quality control. The fact, that we

[102] See Larose and Larose (2015), p. 92
[103] See Appendix B.5
[104] See Maaten (2008)
[105] See Maaten (2018)

cannot simply reduce the dimension of the new parts leads to the conclusion that t-SNE is not suitable for our purpose.

Principal component analysis (PCA) on the other hand can operate with new data points. Applying PCA to our product dataset leads to reduced dimensionality. The first component explains around 11 % of the variance within the dataset. The first 106 components explain approximately 99% of the variance. Hence, we can reduce the dimensions from 197 to 106 by only losing around 1 % of the information (Appendix A.2). Thereby multicollinearity is reduced, which will improve our logistic regression model.

Further ways to select features: Feature Engineering

Feature engineering is the process of extracting existing features or creating new features that result in more accurate predictive models. The application of domain knowledge and creativity usually play a crucial role.[106] Due to the lack of domain knowledge, we can't create new variables based on our understanding of the numeric attributes. However, there are ways to construct new features if the other datasets (date) would be considered. Data analysists from Kaggle created a feature, which reflects temporal proximity on the manufacturing line. The idea behind this feature is the assumption that, if one part was a failure it is likely that other parts, which have been processed directly after or in advance, are also more likely to fail. During the competition, this feature turned out to have high predictive power.[107]

4.4. Phase 4: Modelling

To keep the performance of the models comparable I examine this phase utilising the product dataset.

4.4.1. Select modelling techniques and generate test design

Lesmeister (2017) lists several methods to solve classification problems.[108] The four modelling techniques we apply are theoretically described in section 3.3.2. The first one is a logistic regression as it is related to linear regression, which is one of the oldest

[106] See Fuentes (2018) Feature Engineering
[107] See Waring (2017)
[108] See Lesmeister (2017) Algorithms Flow Chart

predictive methodologies.[109] Secondly, we will discuss the decision tree, which is both simple to build and to understand and therefore popular in business applications.[110] Fernandez-Delgado (2014) discover that random forest[111] is most likely to be the best classifier in most datasets.[112] The last one, XGBoost, is an efficient implementation of a gradient boosted tree.[113] It is used by the high performing participants during the Kaggle competition and should, therefore, be evaluated in this study.[114]

Before modelling one must question the test design. The product data was split into train (70%) and test dataset (30%). All models were built based on the product train dataset and evaluated with the product test dataset. To further improve the generalizability of the logistic regression and the decision trees I use k-fold cross-validation. Random Forest and XGBoost were not applied with k-fold cross-validation. First because of computational reasons and secondly because they already are ensemble learning methods which are trained on various subsamples of the dataset (see section: 3.3.2).[115]

4.4.2. Build the models: logistic regression and tree-based models

The process of building the model is closely related to adjusting the data and checking the performance with evaluation metrics. To improve readability, I included some aspects of those phases in the following section. Cross-validation is used within the caret package.[116] The caret package allows us to define a metric to optimise during training. We set this metric to the mcc score.

Logistic Regression

We start by fitting a simple logistic regression model to our train data and evaluate the predictions with the test dataset. By optimising the classification threshold, we achieve an mcc score of 0.46. In a next step, we apply logistic regression with 10-fold cross validation and generate an mcc score of 0.39. This seems surprising as we would expect better performance with cross validation. One reason might be, that both models produce the warning, that the predictions from a rank deficient fit may be misleading. Almost none of the model's coefficients is significant, and 33 coefficients have no available ("NA")

[109] See Kotu and Desphande (2015) p.167
[110] See Kotu and Desphande (2015) p.64
[111] See Liaw and Wiener (2018) Package 'randomForest'
[112] See Fernandez-Delgado et al. (2014)
[113] See Chen et al. (2018) Package 'xgboost'
[114] See Scnndl (2016)
[115] See James et al. (2018) p. 317-p.318
[116] See Kuhn (2018) Package 'caret'

values (Appendix A.3.1). This is an issue if variables are highly correlated (multicollinearity). As we have seen in a previous section, we can reduce multicollinearity by applying principal component analysis. The first 106 principal components cover 99 % of the explained variance. They are used as input for our models. The single logistic regression remains at an mcc of 0.46. The score of the cross-validated model increases up to 0.46 too. Most of our coefficients are significant, and we do not longer get "NA" values in our coefficients (Appendix A.3.2). Further, we solved the warning that the predictions might be misleading.[117]

Decision Tree

In this section, the decision tree from the rpart package[118] is used. We start by fitting a simple decision tree to the training dataset. The trained model produces an mcc score of 0.44. Applying 10-fold cross validation improves the score up to 0.46. This demonstrates that methods like cross-validation or bagging can improve performance. The trained model was more general and therefore better handled the unseen test data. Whereas the first single decision tree consists out of two decision nodes, does the cross-validation eliminate the second one (Appendix A.4). To verify this impression, we evaluate trees with more depth and see whether they get generalised through cross-validation too. The "minsplit" parameter regulates the minimum number of observations required for a split. In other words, it can control the depth of the grown tree. By default, it is set to 20. With only 51 failures in the train dataset, it might be reasonable to decrease this value to allow further differentiation. By modifying this parameter from 20 to 5, we get a decision tree with three decision nodes. The mcc score is 0.44 again. We then apply cross-validation with the modified parameter, and both additional nodes get eliminated. A closer look at the plots of the trees (Appendix A.4) reveals the decisive feature, namely L1_S24_F1723. The increased depth only added more complexity but did not enhance the model's performance.[119]

Random Forest

The cross-validation of the decision trees only lead to a slight improvement in our prediction quality (mcc from 0.44 to 0.46). This is reasonable as we saw that we have a quite dominant feature. Hence the trees trained during cross-validation probably have a

[117] See Appendix B.8
[118] See Therneau et al. (2018) Package 'rpart'
[119] See Appendix B.9

similar shape. By running a random forest on our train dataset, we can give more value to other predictors. In some of the trees, our dominant feature L1_S24_F1723 cannot be selected for certain splits. Applying random forest further improves the performance and achieves an mcc score of 0.50.[120] The number of trees is set to 400. The modification of parameters such as "sampsize" or "cutoff"[121] did not lead to further improvements concerning the mcc score. Anyhow, we will see how they can be used during the application of business knowledge (section: 4.4.3).

Gradient boosting tree (XGBoost)

Some key parameters must be determined when using boosted decision trees. Firstly, the number of trees. Boosted decision trees can overfit, if the number of trees is chosen too large. The second important parameter is a shrinkage factor (called eta in XGBoost), which describes the learning rate. This factor can slow the process down and provoke different shapes of trees. Typically, this value is 0.01 or 0.001. If it is too small, the number of trees must increase dramatically to achieve good performance. The third critical parameter is the interaction depth d (called max_depth in XGBoost). It controls the complexity of the boosted ensemble model. The parameter d describes the number of splits each tree can make. Often d is equal to one, which means that each tree is a stump consisting of only one split.[122] Within our model, we set the number of trees to 1,000, the learning rate to 0.01 and max_depth to 1. The objective function is set to "binary:logistic", as we want to predict a binary response variable. Applying XGBoost with the mentioned parameters results in an mcc score of 0.51.[123]

Further strategies to handle the total dataset

There are several methods to deal with this vast amount of data. Therefore, I briefly mention another approach. The tree-based algorithms measure the variable importance. XGBoost applied to a randomly chosen subset delivers insights into the importance of the variables.[124] We can now select the critical columns from the original total dataset. With this reduction, the models get computationally applicable. If we now train our XGBoost model again, we achieve an mcc score of around 0.20. A closer look at variable

[120] See Appendix B.10
[121] See Liaw and Wiender (2018) p.19
[122] See James (2018) p. 322
[123] See Appendix B.11
[124] See Lewis (2016)

importance offers another valuable insight. Feature "L3_S32_F3850" and "L1_S24_F1723" are under the top-ranked most important features.[125] This is consistent with the impressions we gained during modelling decision trees and visualising data with t-SNE.

4.4.3. Model assessment with statistical measures and business knowledge

Model assessment with statistical measures

In our test dataset we have 23 failures (1, positive) and 3552 "no failures" (0, negative). Several measures can be used to assess the performance of a model. Nevertheless, in certain situations, some measures might be more useful than others. In our case, we could merely predict all parts as being "no failures", and we would achieve an accuracy of 99.36% (Table 2). Sensitivity would be 0% because no part would be predicted as positive (failure). Specificity, on the other hand, would be 100%, as we would predict all parts as negative (no failure). Hence it is not reasonable to optimise for one metric without considering the concrete structure of the problem. In our case, we should consider the class imbalance in our evaluation metric. Therefore, we use the mcc score as the primary performance measure. All models reached scores above 0.46. Table 3 displays the contingency matrix for the XGBoost model.

		Predicted Category		
		0	1	Total
Actual Category	0	TN: 3552	FP: 0	TAN: 3552
	1	FN: 23	TP: 0	TAP: 23
	Total	TPN: 3552	TPP: 0	3575

Table 2: Only predicting "no failures"

		Predicted Category		
		0	1	Total
Actual Category	0	TN: 3551	FP: 1	TAN: 3552
	1	FN: 16	TP: 7	TAP: 23
	Total	TPN: 3567	TPP: 8	3575

Table 3: Contingency table of XGBoost

It achieved an mcc score of 0.51 and an accuracy of 99.52%. These scores were achieved based on the analysis of a homogenous product group. The winning mcc score during the Kaggle competition was 0.52.[126] They achieved this score on data, that was not filtered by the product group. The XGBoost model performed best, closely followed by the random forest, the cross-validated decision trees and the logistic regression (Appendix A.6).

[125] See Appendix B.12
[126] See Bosch Leaderboard (2016)

Application of business knowledge

Until now we optimised the models for statistical performance measures. However, in real-world projects, the goal is to achieve certain business objectives. This could be for instance cost reduction or quality improvement. It is crucial that the models are discussed and evaluated together with business analysts. It often occurs that the costs for the false positive (FP) and false negative (FN) differ. Consider for instance a manufacturing company, that only checks the quality of a product, when the model classifies the product as damaged. If a product is classified as false positive, it gets checked in quality control but is working well. This is not desirable but still is not a very expensive mistake regarding real costs. On the other hand, if a product is classified as false negative, it gets delivered to the customer even though it is damaged. The part must be replaced, which results in additional transportation costs and a reduced customer satisfaction. This is probably the more damaging mistake. Models can be further optimised with respect to this difference. Thereby the models have different requirements. In the following section, I illustrate this process utilizing the random forest. The random forest is chosen because the optimisation can be intuitively explained. The consideration of business knowledge is crucial to improving the applicability of the model in the concrete business context. However, the business analyst must do a sound cost-benefit analysis to decide about the application of a model. In the following, I will examine such a cost-benefit analysis. Bosch did not explicitly state the real business objectives. Hence, I make certain assumptions to demonstrate the process.

In our case, the application of domain knowledge results into the question whether the costs for incorrectly as working classified products (FN) or the costs for incorrectly as not working classified products are higher (FP). To resolve this question, we make the following assumptions:

1.Assumption: If a predictive model is introduced, Bosch still checks all parts that are classified as a failure.

2.Assumption: We will look at our homogenous product group. Hence the costs x per unit for quality control and the costs y for missing a failure are assumed to be equal for all parts.

3.Assumption: Bosch currently controls every single product. The spending on quality control is assumed to be less than or equal to the cost that would occur if Bosch would

not conduct quality control (assumption of profit maximisation[127]). This leads to the inequation (Table 5):

$$3575 * x \leq 23 * y \quad (2)$$

For simplicity reasons, I illustrate the case of equality. Further, it is assumed that y already includes all relevant costs, from reproducing the part to reputation loss and decreased customer satisfaction. As we have two unknown variables, I choose x = 10 which results to y = 1554 respectively.

4.Assumption: The cost occurring with a model can be calculated as:

$$Cost\_Total = (TP) * Cost(TP) + (TN) * Cost(TN) + (FN) * Cost(FN) + (FP) * Cost(FP) \quad (3)$$

Based on these assumptions we can associate cost or benefits with each of the four possible combinations.[128] Table 4 summarises the assumed cost structure. To calculate the total cost occurring with a model, the values can simply be used in formula 3.

Outcome	Classifi-cation	Actual Value	Cost	Rationale
True Negative (TN)	0	0	0€	No losses
True Positive (TP)	1	1	10€	*x* = The cost of checking the unit in quality control
False Negative (FN)	1	0	1554€	*y* = The cost through the delivery of a damaged product to the customer
False Positive (FP)	0	1	10€	*x* = The cost of checking the unit in quality control

Table 4: Assumed cost structure

The Business analyst and data analyst together discuss the output of the random forest (Table 5). To sum it up: 3 parts would enter quality control, and they would pass ("false alarm"). Whereas 15 parts would be delivered to the customer and he would recognise, that the part fails (“miss”). The business analyst knows about the expensive “misses”. In discussion with the data analysts, they seek for a way to reduce them.

The models can be optimised with respect to this goal. The inclusion of misclassification costs gives weights to specific types of errors and therefore influences the contingency

[127] See Brexer (2008) p.71
[128] See Larose and Larose (2015) p. 462

matrix. This process is called cost-sensitive learning. The interested reader is forwarded to Elkan (2001)[129] or Larose and Larose (2015)[130]. Note that the misclassification costs that are passed to the modelling algorithm should be understood in a way that the algorithm considers a certain error more damaging.[131] The different algorithms have various ways to punish certain mistakes.[132] One way to influence the contingency matrix of the random forest is to adjust the "cutoff" threshold. This threshold finally decides how many votes are needed to classify a part as damaged.[133] By default, it is the majority vote ("cutoff" = (0.5,0.5) for binary variables). Literature suggests manipulating these values to find the best combination suited for the task and business problem at hand.[134] As we want to avoid the expensive "misses" of failures, we adjust the "cutoff"-parameter to (0.93,0.07). To declare a part as damaged, it needs more than 93% of the votes. In other words, the model must be quite sure that the part is damaged in order to classify it so. In Table 5 and Table 6 we can see the predicted values for the random forest without and with the parameter adjustment. Calculating the above-mentioned metrics demonstrates that the mcc score decreases from 0.50 to 0.36. Accuracy drops from 99,50% to 99,02 %.

		Predicted Category		
		0	1	Total
Actual Category	0	TN: 3549	FP: 3	TAN: 3552
	1	FN: 15	TP: 8	TAP: 23
	Total	TPN: 3564	TPP: 11	3575

Table 5: random forest without "cutoff" adjustment

		Predicted Category		
		0	1	Total
Actual Category	0	TN: 3530	FP: 22	TAN: 3552
	1	FN: 13	TP: 10	TAP: 23
	Total	TPN: 3543	TPP: 32	3575

Table 6: random forest with "cutoff" adjustment

On the other hand, the false negative rate (FN/TAP) decreases from 65.23% to 56.52%. We managed to reduce the expensive false negatives, but only with an increase in the false positives. The decision about which model to use still is difficult to make concerning these performance metrics. However, the business analysts must decide whether to apply one of the models and choose one. To get to a decision, he evaluates these models with respect to the anticipated profit or loss. He has three options (Formula (3)):

[129] See Elkan (2001)
[130] See Larose and Larose (2015) p. 471
[131] See Larose and Larose (2015) p. 460
[132] See Larose and Larose (2015) p. 483
[133] See Liaw and Wiener (2018) Package 'randomForest', p.15
[134] See Larose and Larose (2015) p. 460

1. No model is applied. All parts are checked in quality control.
   Cost_Total: 3575*10€ = 35750€
2. The model without parameter adjustment is applied.
   Cost_Total: 3549*0+8*10+15*1554+3*10 = 23420€
3. The model with parameter adjustments.
   Cost_Total: 3534*0+10*10+13*1554+22*10 = 20522€

In this scenario, the business analyst would choose the third model. Although we have less accuracy and a lower mcc score, this model performs best regarding cost reduction. When quality checking is reduced to the predicted failures, the application of the third model reduces expenses by 42.6%. The modification of the "cutoff" parameter added 8.1% compared to the second model. This is a very basic model based on our assumptions above. The assumptions should be subject of discussion and must be determined by the business analyst in the concrete business context. A different cost structure leads to different results. Furthermore, the selection of the "cutoff" parameter is not optimised through parameter tuning. The focal point of this section should be the demonstration of how the models can be improved by applying business knowledge.

4.5. Phase 5: Model Evaluation

As we have seen, model evaluation is interwoven with the task of building models. Evaluating and optimising the model according to data mining and business criteria is explained in the sections above. The models have been built successfully concerning our data mining goal. Our data mining goal was defined as achieving an mcc score of above 0.3. They perform better than guessing and could, therefore, be used for the purpose of illustration. We do not have much information about the real business objectives of Bosch. Therefore, we can hardly evaluate whether to apply a model or not. The developed models during this study achieve good performance within a single product group. It would be further interesting to see whether several models for individual product groups could be combined to a well performing general model. However, other interesting outcomes of the data mining process includes the detection of relevant features. Several different methods and models directed us towards the features "L3_S32_F3850" and "L1_S24_F1723". It can be recommended that those stations are treated with special attention during the manufacturing process. It should be briefly reflected whether the

models have been built correctly. Each of the models still has room for improvement and small issues that should be addressed. When logistic regression is used for predicting the failures the probability of 1 occurs. For the cross-validated decision trees, we get the warning that missing values occurred in the resampled performance measures. Furthermore, it is crucial to mention that with only 23 failures in our test data, one single misclassification already leads to a notable effect on the mcc score of around 0.02. The random split of the train and test data within the product group can lead to small differences in the results. According to the crisp dm model further iterations to improve the model can be initiated to address these problems. Nevertheless, in real-world projects as well as in this study one must decide when to quit, as achieving the perfect model would usually exceed time and budget constraints. In the following section, I will discuss the insights gained from the data mining process.

5. Discussion

In this section, I will briefly reflect on the insights gained from the developed case and further discuss advantages and disadvantages of using this case for teaching. As expected, it turned out, that the process is not linear but iterative. In between modelling and data preparation exists a close relationship. During several iterations, the model and data are adopted concerning the evaluation metrics. While applying the CRISP DM model, it emerged that some methods explained by literature are not always straightforward applicable. For instance, the application of logistic regression produced the warning that the interpretation of our predictions might be misleading. Probably most people would not use this example to illustrate logistic regression in the first place. Nevertheless, the process of addressing warnings produced within a real-world project provokes valuable insights. In this case, the insight that applying PCA in advance leads to less multicollinearity and then to a better performing model. Moreover, we have seen how creative solutions and inventiveness can enable a successful project considering available resources. T-SNE, for instance, can be applied to the correlation matrix of the features but not to the parts themselves. The need to reduce data volume resulted in the creation of subsamples according to the part-type. And while dealing with missing values, it turned out that the addition of (+2) to transformed data and the replacement of missing values with 0 is an applicable solution. We have further seen how to improve our models when

business knowledge comes into play. This underlines the importance of diverse teams to conduct a successful data mining project. We have seen that optimising only for statistical metrics might lead to a well-performing model in terms of accuracy, but enterprises seek for the optimisation of business objectives such as cost reduction or quality improvement. Hence the models need to be optimised with respect to those business objectives. To evaluate the pros and cons of teaching this case I compare it to teaching a case with specially created sample data. Both options have some advantages and disadvantages.

Cases with specially prepared datasets can be utilised to explain and illustrate specific concepts. The data can be prepared for the illustration of specific machine learning techniques. Hence the output of these techniques is probably perfectly interpretable. This supports the understanding of how specific methods work technically. As they are created for illustration purpose, they have only small computational requirements. This ensures that most people can profit from the sample case. This approach can be found in a variety of educational books, such as Larose and Larose (2015)[135], Kotu and Desphande (2015)[136] or James et al. (2017)[137].

The developed case, on the other hand, displays the process within a real-world project. Thereby it delivers insights that are valuable for real-world challenges. By not always providing the expected results in the first place it leads towards a more profound understanding of the algorithms. The models get improved by iteratively changing the data to fit the specific technique better. A process, which is typical within a work-environment in industry. It moreover shows how, besides the technical knowledge about an algorithm, creativity and adaptability enable the creation of successful models. Not only concerning certain constraints but also considering the underlying data structure. Teaching a case with real-world data can provide a more holistic illustration of the data mining process.

The question arises which case should be preferred. The answer to this question depends on the addressed learning objective. If the goal is to illustrate the functionality of a particular algorithm or the basics of a technical concept a sample case should be chosen. It can be prepared to allow interpretable results supporting the understanding of the concept. On the other hand, the developed real-world case can provoke a more advanced

[135] See Larose and Larose (2015)
[136] See Kotu and Desphande (2015)
[137] See James et al. (2017)

understanding of the applied methods and challenges that might occur with real-world data. Moreover, this case should be used when the goal is to illustrate the CRISP DM model from a management perspective. The process of finding solutions within the presence of certain constraints is essential in project management. Managing real-world projects always include the consideration of computational, time and budget constraints. Challenges that students face lately when working in a practical environment. Besides that, it demonstrates that not only technical knowledge from a specific discipline is needed but also creativity and the expertise from different domains.

6. Conclusion

Within this study, I developed a case for teaching machine learning algorithms with a real-world dataset. Limitations of the developed models are stated within the section model evaluation (4.5). Furthermore, it should be mentioned that this case itself is highly specific. The applied models and techniques are adapted to the dataset and therefore cannot be directly transferred to other problems. The presented case only represents one of multiple ways of dealing with the dataset. Nevertheless, the learnings and experiences that can be gained from teaching this case are transferable.

Whether teaching a real-world case or using sample data depends on the learning objectives. This case might not be suitable when the learning goal is the basic understanding of certain machine learning algorithms. It does not contain mathematical details and dealing with real-world data can sometimes lead to confusing results. However, the case might better reflect and prepare the students towards the challenges of a data mining project in the industry. We have seen how creativity and diverse knowledge enable the creation of a successful model, meanwhile considering certain constraints. Those characteristics make the case especially interesting from a management perspective. In accordance with the pathway developed by the World Economic Forum (2017), I would like to close with the words: Real-world projects will require workers who, besides formal education, can show creativity, inventiveness and adaptability to the concrete problems at hand.[138] With this purpose, I recommend teaching the developed case.

[138] World Economic Forum ("Publisher") (2017) p. 33

Appendix

Table of Content

A Appendix: Model performance and visualizations

A.1 Initial performance of resampled subsets and product group.

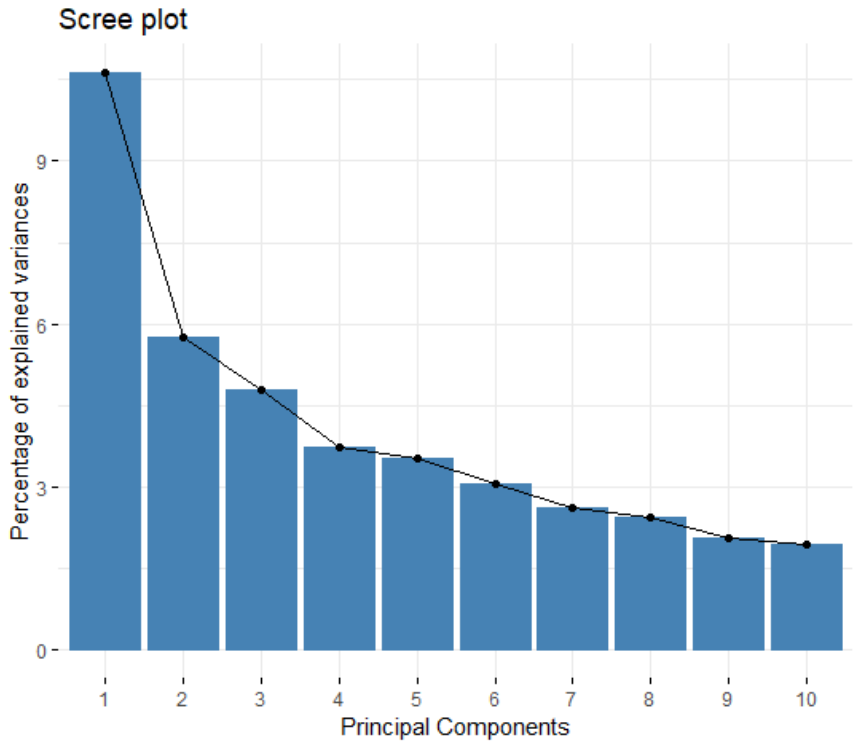
Calculations are done with the Script: Resample_vs_Product_Performance.R

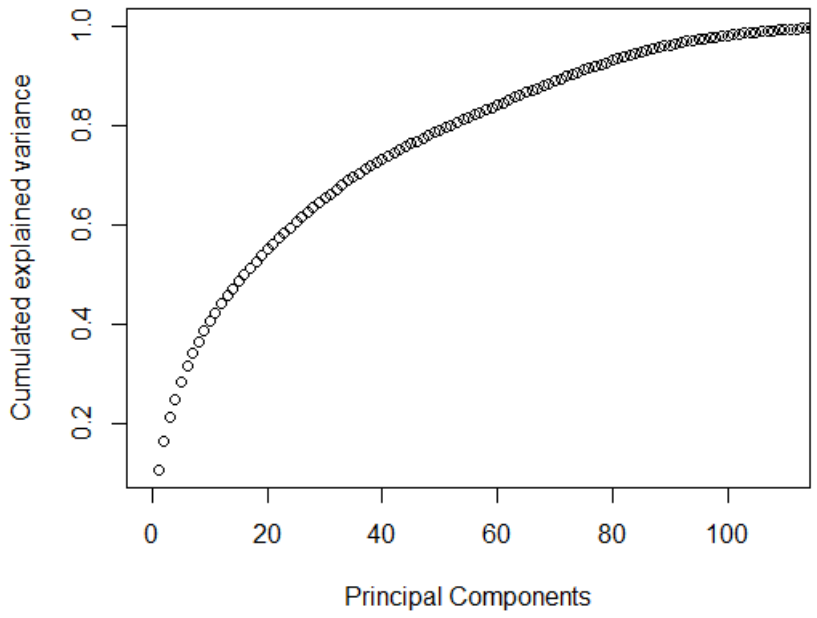
Name	Fraction of Failures	Number of Records	MCC in XGBoost
Fifty_damaged_train.csv	50%	11,006	0.153
Ten_damaged_train.csv	10%	55,030	0.169
Sample_50k.csv	0.57%	50,000	0.149
TestData.csv	0.58112%	236,784	
Product_train.csv	0,6151%	8,340	0.440
Product_test.csv	0,6434%	3,575	

A.2 Visualisation of PCA

The first plot displays the variance explained by each component. The second plot displays the cumulated values of the explained variance.

The plots are generated in the script: logReg.R





A.3 Logistic Regression Output

Due to the large size of the table I provide only a sector of the first 10 variables (components). The script logReg.R reproduces the result for closer investigation.

A.3.1 Output without PCA

As we can see, 33 coefficients are not defined.

```
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0160  -0.0453  -0.0163  -0.0048   3.5853
##
## Coefficients: (33 not defined because of singularities)
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -8.635e+01  3.105e+05   0.000   0.9998
## Ll_S24_Fl512 -6.955e+00  1.729e+01  -0.402   0.6875
## Ll_S24_Fl514  1.949e+00  1.951e+00   0.999   0.3179
## Ll_S24_Fl516 -2.333e+00  2.696e+00  -0.865   0.3869
## Ll_S24_Fl518 -2.290e+00  7.079e+00  -0.324   0.7463
## Ll_S24_Fl520  1.040e+00  2.559e+00   0.407   0.6843
## Ll_S24_Fl539  9.704e+02  6.867e+02   1.413   0.1576
## Ll_S24_Fl544  1.643e+01  1.095e+02   0.150   0.8807
## Ll_S24_Fl565  1.334e+03  1.392e+03   0.959   0.3377
## Ll_S24_Fl567 -1.678e+04  1.311e+07  -0.001   0.9990
## Ll_S24_Fl569 -9.191e-01  2.233e+01  -0.041   0.9672

## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 621.58  on 8339  degrees of freedom
## Residual deviance: 300.64  on 8176  degrees of freedom
## AIC: 628.64
##
## Number of Fisher Scoring iterations: 22
```

A.3.2 Output with PCA

With PCA most coefficients are significant. Moreover, we do not get coefficients that are not defined.

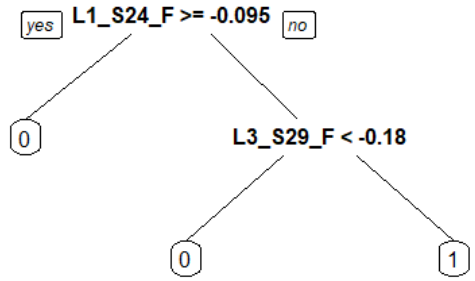
```
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
##  -8.49    0.00    0.00    0.00    8.49
##
## Coefficients:
##               Estimate Std. Error    z value Pr(>|z|)
## (Intercept) -1.447e+15  7.348e+05 -1.969e+09   <2e-16 ***
## PC1         -9.079e+12  1.610e+05 -5.638e+07   <2e-16 ***
## PC2         -1.275e+13  2.190e+05 -5.820e+07   <2e-16 ***
## PC3         -2.446e+12  2.400e+05 -1.019e+07   <2e-16 ***
## PC4         -4.916e+12  2.723e+05 -1.805e+07   <2e-16 ***
## PC5          6.255e+12  2.797e+05  2.236e+07   <2e-16 ***
## PC6          1.363e+13  3.011e+05  4.526e+07   <2e-16 ***
## PC7         -1.783e+12  3.246e+05 -5.491e+06   <2e-16 ***
## PC8         -4.332e+11  3.367e+05 -1.286e+06   <2e-16 ***
## PC9          1.793e+13  3.676e+05  4.878e+07   <2e-16 ***
## PC10         3.964e+12  3.784e+05  1.048e+07   <2e-16 ***

## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance:  621.58  on 8339  degrees of freedom
## Residual deviance: 3243.93  on 8233  degrees of freedom
## AIC: 3457.9
##
## Number of Fisher Scoring iterations: 17
```

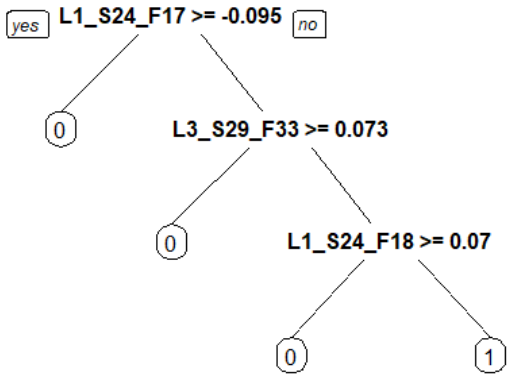
A.4 Plot of decision trees

The following plots are created within the RPART.R Script.

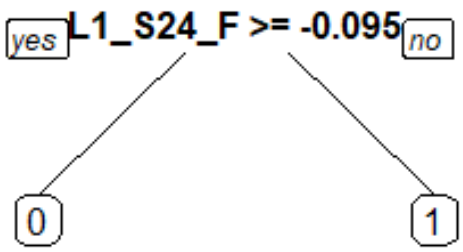
Single Rpart Model with minsplit = 20:



Single Rpart Model with minsplit = 5:

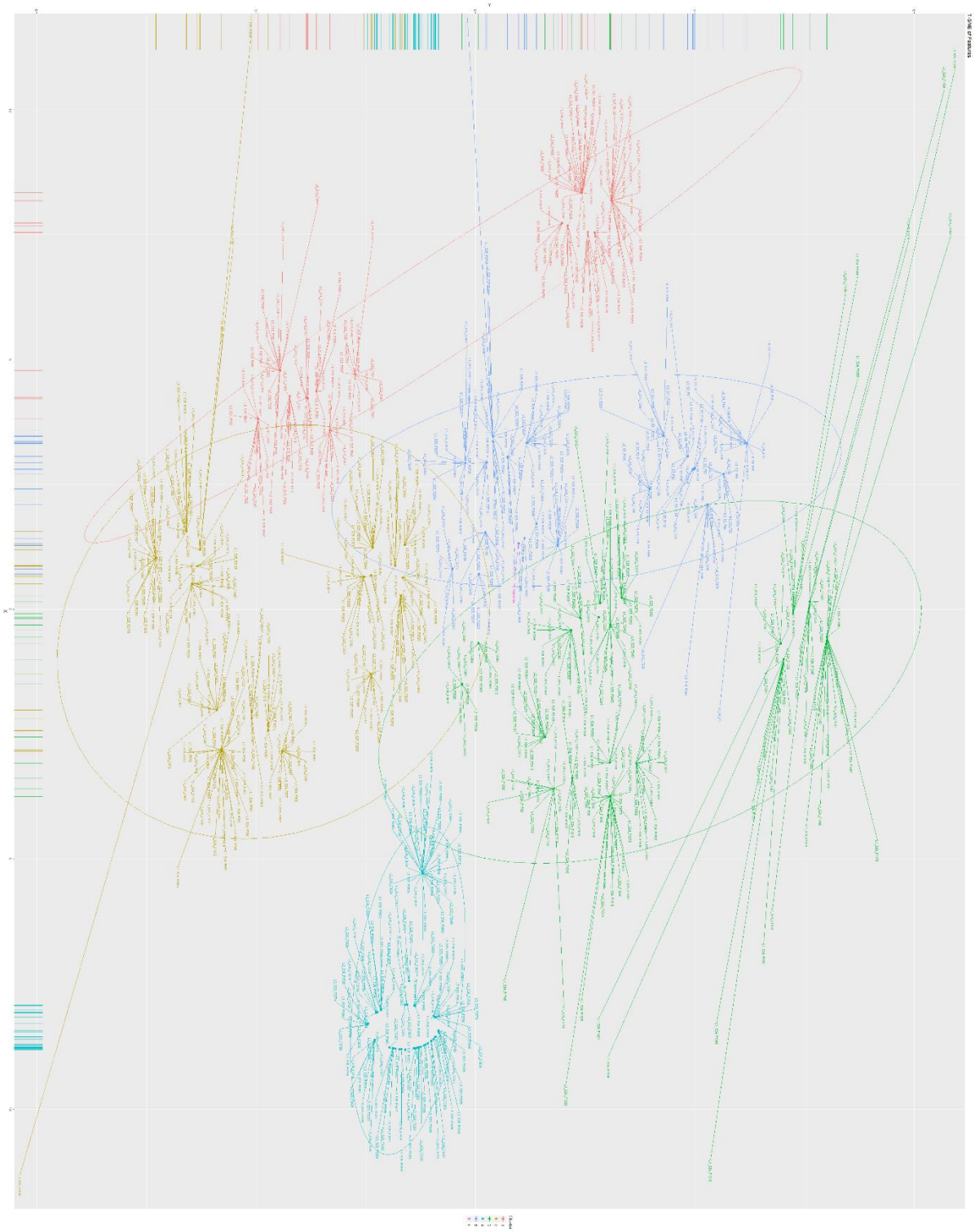


Cross Validated Rpart model with minsplit = 20 or 5



A.5 T-SNE Visualization,

The visualization was based on the Script: tsne-from-Kaggle, created by Laurea[139]



[139] See Laurea (2016)

A.6 Model Performance

Model	MCC
XGBoost	0.51
Random forest	0.50
Cross validated decision trees	0.46
Logistic regression	0.46

B Appendix: R Scripts and Data

The code and subsamples developed during this study are on an SD Card in the back of this Paper. The constructed data includes the subsamples as displayed in Appendix A.1. However, all models and subsamples can be recreated with the R Scripts presented in the following sections.

B.1 Packages

The following packages are required to execute the RScripts:

- data.table
- ggrepel
- Matrix
- factoextra
- randomForest
- Rtsne
- readr
- xgboost
- rpart
- ggplot2
- plyr
- mltools
- rpart.plot

B.2 Data_Exploration.R

```
library(data.table)


#Memory limit setup
memory.limit(20000)

# Load the data into the workspace

trainData <- fread(file = "train_numeric.csv", header = TRUE,
                   data.table = FALSE)

# Discover the value range of the variables,
# apply function does not work with given storage constraints.
# We write a function that computes the min and max values for each
column

min_max_calc <- function()
{
  mins <- numeric(0)
  for(i in 1:ncol(trainData)){
    mins <- c(mins, min(trainData[,i], na.rm = TRUE))
  }
  mins <- as.data.frame(mins)

  maxs <- numeric(0)
  for(i in 1:ncol(trainData)){
    maxs <- c(maxs, max(trainData[,i], na.rm = TRUE))
  }
  maxs <- as.data.frame(maxs)

  result <- cbind(mins, maxs)
  result <- as.data.frame(t(result))
  names(result) <- names(trainData)

  return(result)
}


#global_min and global_max. Take into consideration, that first column
are id values

min_max <- min_max_calc()
str(min_max)
min_max$Id <- NULL

min(min_max)
max(min_max)

# Verify Data Quality
# Count defect parts
defect_Parts <- sum(trainData$Response)
defect_Parts/nrow(trainData)

#ratio of NA in Features
values_Na <- sapply(trainData, function(x)
sum(length(which(is.na(x)))))
ratio_Na <- values_Na/nrow(trainData)

ratio_Na <- t(ratio_Na)
```

```
ratio_Na <- as.data.frame(ratio_Na)

colnames(ratio_Na) <- colnames(trainData)
columns <- colnames(trainData)

#Number of values with ratio_Na < 0.8, (-1) beacuse Id is not
considered
x <- (sum(ratio_Na>0.7)-1)
x/ncol(trainData)

#Ensure that Response does not have missing values
ratio_Na$Response

#Calculate records number of records, that are complete (No missing
values)
comp <- trainData[complete.cases(trainData),]
nrow(comp)
```

B.3 t-SNE-from-Kaggle.R

```
# This script is developed by Laurae (2016) and
# retrieved from: https://www.kaggle.com/c/bosch-production-line-
performance/discussion/23067
# Request Date was 10.11.2018

library(data.table)
library(Rtsne)
library(ggplot2)
library(ggrepel)

# The calculation of the correlation matrix, needs some time.
# Therefore it is provided by Laurae and can be downloaded on the
above mentioned link.


cor_out <- as.matrix(fread("cor_train_numeric.csv", header = TRUE,
                           sep = ","))
gc(verbose = FALSE)
set.seed(78)
tsne_model <- Rtsne(data.frame(cor_out),
                    dims = 2,
                    #initial_dims = 50,
                    initial_dims = ncol(cor_out),
                    perplexity = 322, #floor((ncol(cor_out)-1)/3)
                    theta = 0.00,
                    check_duplicates = FALSE,
                    pca = FALSE,
                    max_iter = 1350,
                    verbose = TRUE,
                    is_distance = FALSE)
corMatrix_out <- as.data.frame(tsne_model$Y)
cor_kmeans <- kmeans(corMatrix_out, centers = 5, iter.max = 10,
                     nstart = 3)
corMatrix_outclust <- as.factor(c(cor_kmeans$cluster[1:968], 6))
corMatrix_names <- colnames(cor_out)

# Dependend on the display, the plot is not analyzable.
# Therefore it should be exportet to a image file with width: 4212 ,
height: 3321

ggplot(corMatrix_out, aes(x = V1, y = V2, color = corMatrix_outclust))
+ geom_point(size = 2.5) + geom_rug() + stat_ellipse(type = "norm") +
ggtitle("T-SNE of Features") + xlab("X") + ylab("Y") + labs(color =
"Cluster", shape = "Cluster") + geom_text_repel(aes(x = V1, y = V2,
label = corMatrix_names), size = 2.8)
```

B.4 count_failures_s32.R

```
library(data.table)

memory.limit(20000)
#Load train_numeric and safe Response

trainData <- fread(file = "train_numeric.csv", header = TRUE,
                   data.table = FALSE)
Response <- trainData$Response
trainData$Response <- NULL

#Set values to 1 and na to 0, add Response column

trainData[trainData <= 1 & trainData >= -1] <- 1
trainData[is.na(trainData)] <- 0
trainData <- cbind(trainData, Response)

#Drop id
trainData  <- trainData[,2:970]

#Get subset of records wich passed through s32

records_s32 <- trainData[trainData$L3_S32_F3850 == 1,]

#calculate fraction of failed parts
num_failures_s32 <- sum(records_s32$Response == 1)
fraction_failures_s32 <- num_failures_s32/nrow(records_s32)
fraction_failures_s32
```

B.5 subsample_construction.R

```
library(data.table)
library(readr)

trainData <- fread("case4_train_numeric.csv", header= TRUE,
                   data.table = FALSE)


#select the damaged parts
damaged_only <- trainData[trainData$Response == 1,]
write_csv(damaged_only,"damaged_only.csv", col_names = TRUE)

#select the functioning parts
functioning_only <- trainData[trainData$Response == 0,]

#trainSize equals to 80% of the data
trainSize = round(0.8*nrow(damaged_only))


#random selection of train and test data
set.seed(123)
training_indices <- sample(seq_len(nrow(damaged_only)),
                           size = trainSize)
trainDamaged <- damaged_only[training_indices,]
testDamaged <- damaged_only[-training_indices,]

#Create A subsample where a fraction of 10% is damaged
set.seed(123)
Ten_damaged_train <- functioning_only[sample(nrow(functioning_only),
                                             49527),]
Ten_damaged_train <- rbind(trainDamaged, Ten_damaged_train)
set.seed(123)
Ten_damaged_train <-
Ten_damaged_train[sample(nrow(Ten_damaged_train)),]
write_csv(Ten_damaged_train,"ten_damaged_train.csv", col_names = TRUE)


#Create a subsample where a fraction of 50% is damaged
set.seed(123)
Fifty_damaged_train <- functioning_only[sample(nrow(functioning_only),
5503),]
Fifty_damaged_train <- rbind(trainDamaged, Fifty_damaged_train)
set.seed(123)
Fifty_damaged_train <-
Fifty_damaged_train[sample(nrow(Fifty_damaged_train)),]
write_csv(Fifty_damaged_train,"fifty_damaged_train.csv",
          col_names = TRUE)

#Create a test subsample with the original balance of 0.58112%
set.seed(321)
testData <- functioning_only[sample(nrow(functioning_only), 235408),]
testData <- rbind(testDamaged, testData)
set.seed(123)
testData <- testData[sample(nrow(testData)),]
write_csv(testData,"testData.csv", col_names = TRUE)
```

```
#Create Train data from Product Group
product<-fread("product1_numeric.csv", header = TRUE,
               data.table = FALSE)

#Drop Columns that only contain missing values
nas <- apply(product,2,function(x) sum(length(which(is.na(x)))))
product <- product[, nas != nrow(product)]

#drop column that contain 0 variance
product <- product[,apply(product, 2, function(x) var(x)!= 0)]

trainSize = round(0.7*nrow(product))
set.seed(123)
training_indices <- sample(seq_len(nrow(product)), size = trainSize)
product_train <- product[training_indices,]
product_test <- product[-training_indices,]

write_csv(product_train, "product_train.csv", col_names = TRUE)
write_csv(product_test, "product_test.csv", col_names = TRUE)

#make a subsample of 50k records
set.seed(123)
sample_50k <- trainData[sample(nrow(trainData),50000),]
sample_50k_train <-
write_csv(sample_50k,"sample_50k.csv", col_names = TRUE)
```

B.6 create_productGroup.R

```
library(data.table)
library(plyr)

trainData <- fread("train_numeric.csv", header = TRUE,
                   data.table = FALSE)

#Create a subset for the first 500k records
set.seed(123)
training_indices <- sample(seq_len(nrow(trainData)), size = 500000)
sample_500k <- trainData[training_indices,]

#Set values to 1 and missing values to 0
sample_500k[sample_500k <= 1 & sample_500k >= -1] <- 1
sample_500k[is.na(sample_500k)] <- 0

#Drop Id and Response
sample_500k_duplicates  <- sample_500k[,2:969]

#Count the number of duplicates of each unique row based on a subset
batch <- sample_500k_duplicates[1:100000,]
aggregation <- aggregate(list(numdup = rep(1,nrow(batch))),
                         batch, length)

#Check the max, mean and min number of duplicates
max(aggregation$numdup)
min(aggregation$numdup)
mean(aggregation$numdup)

#Select the characteristic pattern of 0 and 1 from the major product
group

select_max_row <- aggregation[aggregation$numdup ==
                                        max(aggregation$numdup),]

select_max_row_search <- select_max_row

select_max_row_search$numdup

select_max_row_search$numdup <- NULL

#Extract the records from the total numeric dataset, This is done with
the first 500k records
batch_size = 20000
res_duplicates<-data.frame()

for(i in 1:25){
  batch <- sample_500k[1+((i-1)*batch_size):((i)*batch_size),]
  res <- batch[which(apply(batch,1,
            function(x) all(select_max_row_search == x[2:969]))),]
  res_duplicates <- rbind(res_duplicates, res)
}

#Afterwards it is done for the next 680k parts
sample_500k <- trainData[-training_indices,]

for(i in 1:34){
  batch <- sample_500k[1+((i-1)*batch_size):((i)*batch_size),]
```

```
  res <- batch[which(apply(batch,1,
                 function(x)all(select_max_row_search == x[2:969]))),]
  res_duplicates <- rbind(res_duplicates, res)
}

#Include remaining 3.747 records
batch <- batch <- sample_500k[1+((59)*batch_size):1183747,]
res <- batch[which(apply(batch,1,
            function(x) all(select_max_row_search == x[2:969]))),]
res_duplicates <- rbind(res_duplicates, res)


#Drop Columns that only contain missing values
nas <- apply(res_duplicates,2,function(x)
sum(length(which(is.na(x)))))
data <- res_duplicates[, nas != nrow(res_duplicates)]

#drop column that contain 0 variance
data <- data[,apply(data, 2, function(x) var(x)!= 0)]

write_csv(data, "product1_numeric.csv", col_names = TRUE)
```

B.7 Resample_vs_Product_Performance.R

```
library(data.table)
library(Matrix)
library(caret)
library(xgboost)
library(mltools)

#Load TrainData from ten_damaged. Can be replaced with fifty_damaged
trainData <- fread("ten_damaged_train.csv", header = TRUE,
                   data.table = FALSE)

Response <- trainData$Response

#Drop ID and Response
trainData<-trainData[,2:969]

#Add 2 to all values, and set missing values to 0
for(col in names(trainData)) set(trainData, j = col, value =
                                                  trainData[[col]] + 2)
for(col in names(trainData)) set(trainData,
                                 which(is.na(trainData[[col]])), col, 0)
trainData <- cbind(trainData, Response)

#load testData
testData <- fread("testData.csv", header = TRUE, data.table = FALSE)

Response_test<- testData$Response

#Drop ID and Response
testData<-testData[,2:969]

for(col in names(testData)) set(testData, j = col, value =
                                                  testData[[col]] + 2)
for(col in names(testData)) set(testData,
                                 which(is.na(testData[[col]])), col, 0)

# Prepare xgboost trainData

xgb_train <- trainData
xgb_train_Response <- xgb_train$Response
xgb_train$Response<- NULL

#Parameter for xgBoost
params <- list(objective = "binary:logistic",
               eval_metric = "auc",
               eta = 0.01,
               max_depth = 2,
               colsample_bytree = 0.5,
               base_score = 0.005)

#Train model
xgb <- xgboost(data.matrix(xgb_train),
               label = xgb_train_Response,
                params = params, nrounds = 200,
                early_stopping_rounds = 50, verbose = T)
```

```
#Make Predictions
pred_xgb <- predict(xgb, data.matrix(testData))

#Define a sequence of possible threshold values
matt <- data.table(thresh = seq(0.0, 0.998, by = 0.001))

#Calculate mcc scores to the threshold values
matt$scores <- sapply(matt$thresh, FUN =
                        function(x) mcc(Response_test, (pred_xgb > x)
                        * 1))

# Print the optimal result
opt <- matt[which.max(matt$scores), ]
print(opt)
pred_bin <- ifelse((pred_xgb > opt$thresh), 1, 0)
table(Response_test, pred_bin)
mcc(Response_test,pred_bin)

# Evaluate performance of product group
#Load TrainData & TestData
trainData <- fread("product_train.csv", header = TRUE,
                    data.table = FALSE)
testData <- fread("product_test.csv",
                   header = TRUE, data.table = FALSE)

#Drop ID
trainData<-trainData[,2:198]
testData <- testData[,2:198]

#Drop Response column of test Data
Response_test <-testData$Response
testData$Response <- NULL

xgb_train <- trainData
xgb_train_Response <- xgb_train$Response
xgb_train$Response<- NULL

params <- list(objective = "binary:logistic",
               eval_metric = "auc",
               eta = 0.01,
               max_depth = 2,
               colsample_bytree = 0.5,
               base_score = 0.005)

set.seed(123)
xgb <- xgboost(data.matrix(xgb_train),
               label = xgb_train_Response,
                params = params, nrounds = 200, early_stopping=50,
                verbose = T)

pred_xgb <- predict(xgb, data.matrix(testData))

matt <- data.table(thresh = seq(0.0, 0.999, by = 0.001))

matt$scores <- sapply(matt$thresh, FUN =
                            function(x) mcc(Response_test,(pred_xgb >
                            x)*1))

opt <- matt[which.max(matt$scores), ]
print(opt)
pred_bin <- ifelse((pred_xgb > opt$thresh), 1, 0)
```

```
table(Response_test, pred_bin)
mcc(pred_bin,Response_test)
```

B.8 logReg.R

```
library(data.table)
library(caret)
library(mltools)
library(factoextra)
library(plyr)

#Load TrainData & Test Data
trainData <- fread("product_train.csv", header = TRUE,
                   data.table = FALSE)
testData <- fread("product_test.csv", header = TRUE,
                  data.table = FALSE)

#Drop ID
trainData<-trainData[,2:198]
testData <- testData[,2:198]


#Drop Response column of test Data
Response_test <-testData$Response
testData$Response <- NULL

#Apply PCA

pca_data <- trainData
Response <- trainData$Response
pca_data$Response <- NULL

#Apply PCA first,
pca <- prcomp(pca_data, scale = TRUE)

#Explore output of PCA
fviz_eig(pca)

#Calculate predicted variances
pr_var = (pca$sdev)^2
pro_var_ex = pr_var/sum(pr_var)

#Plot predicted variance for each component
plot(pro_var_ex, xlim=c(0,60), type = "b")
plot(cumsum(pro_var_ex), xlim = c(0,60),
      ylab = "Cumulated explained variance",
      xlab = "Principal Components")
cumsum(pro_var_ex)

#create the dataframes with the principal components
trainData$Response <- NULL
trainData <- data.frame(Response = Response, pca$x)
testData <- as.data.frame(predict(pca, newdata = testData))
rm(pca_data)

#Only take 106 features, to cover 99% of explained varaince
trainData <- trainData[,1:107]
testData <- testData[,1:106]


###### Apply Logistic Regression
```

```
#Transform the Response variable into a vector
trainData$Response <- factor(trainData$Response)

###### logistic Regression Without Cross Validation

model_logReg <- glm(formula = Response~.,
                    family = binomial(link = "logit"),
                    data = trainData)
summary(model_logReg)

#Type response leads to probabilites instead of the logOdds
predictions_logReg <- predict(model_logReg, newdata = testData,
                              type = "response")

matt <- data.table(thresh = seq(0.0, 0.999, by = 0.001))

matt$scores <- sapply(matt$thresh, FUN =
                        function(x) mcc(Response_test,
(predictions_logReg > x) * 1))

#Select max. mcc score and threshold

opt <- matt[which.max(matt$scores), ]
print(opt)
pred_bin <- ifelse((predictions_logReg > opt$thresh), 1, 0)
table(Response_test, pred_bin)
mcc(pred_bin,Response_test)

#Probabilities of 1 occurs.
max(predictions_logReg)
sum(predictions_logReg==max(predictions_logReg))

##### Apply logistic regression with cross validation using the caret
package

#Define our metrics which should be optimized, here Matthew
correlation coefficient
mccSummary <- function (data, lev = NULL, model = NULL){

  tp <- as.numeric(sum(data$obs == 1 & data$pred == 1))
  tn <- as.numeric(sum(data$obs == 0 & data$pred == 0))
  fp <- as.numeric(sum(data$obs == 0 & data$pred == 1))
  fn <- as.numeric(sum(data$obs == 1 & data$pred == 0))

  numer <- (tp * tn) - (fp * fn)
  denom <- ((tp + fp) * (tp + fn) * (tn + fp) * (tn + fn)) ^ 0.5
  out <- numer/denom
  names(out) <-"mcc"
  out
}

# define traininControl with 10-fold-cross validation
train_control<- trainControl(method="cv", number=10,
                             summaryFunction = mccSummary)

# train the model, define family as binomial for logistic regression
model<- train(Response~., data=trainData, metric = "mcc",
              trControl=train_control, method="glm", family =
              binomial(link = "logit"), maximize = T)

# print cv scores
```

```
model
summary(model)
varImp(model)


#make Predictions and calculate mcc
predictions_logReg <- predict(model, newdata = testData)
predictions_logReg <- as.numeric(predictions_logReg)
predictions_logReg[predictions_logReg == 1] <- 0
predictions_logReg[predictions_logReg == 2] <- 1

table(Response_test, predictions_logReg)
mcc(predictions_logReg,Response_test)
```

B.9 RPART.R

```
library(data.table)
library(caret)
library(mltools)
library(rpart)
library(rpart.plot)

#Load TrainData & Test Data
trainData <- fread("product_train.csv", header = TRUE,
                   data.table = FALSE)
testData <- fread("product_test.csv", header = TRUE,
                   data.table = FALSE)

#Drop ID
trainData<-trainData[,2:198]
testData <- testData[,2:198]

#Drop Response column of test Data
Response_test <-testData$Response
testData$Response <- NULL

###### Apply Decision Tree

#Transform the Response variable into a vector
trainData$Response <- factor(trainData$Response)


####### Rpart Without Cross Validation

#Train model, minsplit can be changed to 5
model <- rpart(formula = Response~., data = trainData,
               minsplit=20, method = "class")
prp(model)

#VarImp
varImp(model)

#Make Predictions and clalulate mcc
predictions <- predict(model, newdata = testData,  type = "class")
predictions <- as.numeric(predictions)
predictions[predictions == 1] <- 0
predictions[predictions == 2] <- 1

table(Response_test, predictions)
mcc(predictions,Response_test)

####### Rpart with Cross validation

#Define our metrics which should be optimized, here Matthew
correlation coefficient
mccSummary <- function (data, lev = NULL, model = NULL){

  tp <- as.numeric(sum(data$obs == 1 & data$pred == 1))
  tn <- as.numeric(sum(data$obs == 0 & data$pred == 0))
  fp <- as.numeric(sum(data$obs == 0 & data$pred == 1))
  fn <- as.numeric(sum(data$obs == 1 & data$pred == 0))

  numer <- (tp * tn) - (fp * fn)
```

```
  denom <- ((tp + fp) * (tp + fn) * (tn + fp) * (tn + fn)) ^ 0.5
  out <- numer/denom
  names(out) <-"mcc"
  out
}


# define traininControl with 10-fold-cross validation
train_control<- trainControl(method="cv", number=10,
                             summaryFunction =mccSummary,
                             savePredictions = T)

# train the model, define family as binomial for logistic regression,
minsplit can be changed to 5
model<- train(Response~., data=trainData, metric = "mcc",
              trControl=train_control, method="rpart", minsplit=20,
              maximize = T)

# print cv scores
model

prp(model$finalModel)
varImp(model)

# Make Predictions and Calculate MCC
predictions <- predict(model, newdata = testData)
predictions <- as.numeric(predictions)
predictions[predictions == 1] <- 0
predictions[predictions == 2] <- 1

table(Response_test, predictions)
mcc(predictions,Response_test)


####### Rpart with missclassification cost adjustments

#Define lossMatrix
lossMatrix <- matrix(c(0,6,1,0), nrow = 2)
(t(lossMatrix))

model <- rpart(formula = Response~., data = trainData,
               method = "class", parms=list(split = "gini",
               loss = lossMatrix))

prp(model)

#VarImp
varImp(model)

#Make Predictions and clalulate mcc
predictions <- predict(model, newdata = testData, type = "class")
predictions <- as.numeric(predictions)
predictions[predictions == 1] <- 0
predictions[predictions == 2] <- 1

table(predictions, Response_test)
mcc(predictions,Response_test)
```

B.10 RandomForest.R

```
library(data.table)
library(caret)
library(mltools)
library(randomForest)

#Load TrainData & Test Data
trainData <- fread("product_train.csv", header = TRUE,
                   data.table = FALSE)
testData <- fread("product_test.csv", header = TRUE,
                   data.table = FALSE)

#Drop ID
trainData<-trainData[,2:198]
testData <- testData[,2:198]


#Drop Response column of test Data
Response_test <-testData$Response
testData$Response <- NULL


###### Apply Random Forest

#Transform the Response variable into a vector
trainData$Response <- factor(trainData$Response)

set.seed(123)
model <- randomForest(formula = Response~., ntree=400, data =
trainData, importance = T, do.trace = T)

varImp(model)

#Make Predictions and clalulate mcc
predictions <- predict(model, newdata = testData,  type = "class")
predictions <- as.numeric(predictions)
predictions[predictions == 1] <- 0
predictions[predictions == 2] <- 1


table(Response_test, predictions)
mcc(predictions,Response_test)

###### Apply Random Forest with missclassification cost adjustments
set.seed(123)
model <- randomForest(formula = Response~., ntree=400, data =
trainData, importance = T, do.trace = T, cutoff=c(0.93,0.07))

predictions <- predict(model, newdata = testData,  type = "class")
predictions <- as.numeric(predictions)
predictions[predictions == 1] <- 0
predictions[predictions == 2] <- 1

table(Response_test, predictions)
mcc(predictions,Response_test)
```

B.11 XGBoost_on_Product.R

```
library(data.table)
library(caret)
library(xgboost)

#Load TrainData & Test Data
trainData <- fread("product_train.csv", header = TRUE,
                   data.table = FALSE)
testData <- fread("product_test.csv", header = TRUE,
                   data.table = FALSE)

#Drop ID
trainData<-trainData[,2:198]
testData <- testData[,2:198]


#Drop Response column of test Data
Response_test <-testData$Response
testData$Response <- NULL

#Prepare the data for the xgboost model
xgb_train <- trainData
xgb_train_Response <- xgb_train$Response
xgb_train$Response<- NULL

#Choose parameters, base_score can be used to represent class
imbalance
params <- list(objective = "binary:logistic",
               eval_metric = "auc",
               eta = 0.01,
               max_depth = 1,
               colsample_bytree = 0.5,
               base_score = 0.005)

#Train the model
set.seed(123)
xgb <- xgboost(data.matrix(xgb_train),
               label = xgb_train_Response, params = params,
                  nrounds = 1000, verbose = T)

#Make Predictions
pred_xgb <- predict(xgb, data.matrix(testData))

#Define a sequence of possible thresholds
matt <- data.table(thresh = seq(0.0, 0.999, by = 0.001))

# Calculate the mcc score to the given thresholds
matt$scores <- sapply(matt$thresh, FUN =
                        function(x) mcc(Response_test,
                        (pred_xgb > x) * 1))

#Print the optimal values
opt <- matt[which.max(matt$scores), ]
print(opt)
pred_bin <- ifelse((pred_xgb > opt$thresh), 1, 0)
table(Response_test, pred_bin)
mcc(pred_bin, Response_test)
```

B.12 XGBoost.R

```
#This script was copied from:
https://www.kaggle.com/cartographic/bish-bash-xgboost, and slightly
modified

library(data.table)
library(Matrix)
library(caret)
library(xgboost)

#Load the data, and create a subsample of 200,000
dt <- fread("train_numeric.csv",header= TRUE)
set.seed(123)
dt <- dt[sample(nrow(dt), 200000),]

#Save response in Y and set the column in dt to NULL
Y  <- dt$Response
dt[ , Response := NULL]


# Add 2 to the values and replace missing values with 0
for(col in names(dt)) set(dt, j = col, value = dt[[col]] + 2)
for(col in names(dt)) set(dt, which(is.na(dt[[col]])), col, 0)

#Matrix with sparse = T reduces the storage needed
dt[1:5, 1:5]
X <- Matrix(as.matrix(dt), sparse = T)
rm(dt)

#Create train and test indices
folds <- createFolds(as.factor(Y), k = 6)
valid <- folds$Fold1
model <- c(1:length(Y))[-valid]

#Param for XGBoost, learning rate 0.01, base score (default 0.5) as we
have fifty damaged dataset,
param <- list(objective = "binary:logistic",
              eval_metric = "auc",
              eta = 0.01,
              base_score = 0.005,
              col_sample = 0.5)

#Transformations into DMatrix to fulfill the requirements of XGBoost
dmodel <- xgb.DMatrix(X[model,], label = Y[model])
dvalid <- xgb.DMatrix(X[valid,], label = Y[valid])

#Train the model
m1 <- xgb.train(data = dmodel, param, nrounds = 20,
                watchlist = list(mod = dmodel, val = dvalid),
                verbose = 1)


#Investigate variable importance
imp <- xgb.importance(model = m1, feature_names = colnames(X))
cols <- imp$Feature
imp[1:10]
length(cols)

head(cols, 10)
```

```
#Remove variables except cols
rm(list = setdiff(ls(), "cols"))

###Apply xgboost on good features

#Only read the detected important cols from the total dataset
dt <- fread("train_numeric.csv",
            select = c(cols, "Response"),
            showProgress = T)

Y  <- dt$Response
dt[ , Response := NULL]

# Add +2 to all values and set missing values to 0
for(col in names(dt)) set(dt, j = col, value = dt[[col]] + 2)
for(col in names(dt)) set(dt, which(is.na(dt[[col]])), col, 0)

X <- Matrix(as.matrix(dt), sparse = T)
rm(dt)


#Apply XGBoost

set.seed(7579)
folds <- createFolds(as.factor(Y), k = 6)
valid <- folds$Fold3
model <- c(1:length(Y))[-valid]

param <- list(objective = "binary:logistic",
              eval_metric = "auc",
              eta = 0.01,
              max_depth = 2,
              colsample_bytree = 0.5,
              base_score = 0.005)

dmodel <- xgb.DMatrix(X[model,], label = Y[model])
dvalid <- xgb.DMatrix(X[valid,], label = Y[valid])


m1 <- xgb.train(data = dmodel, param, nrounds = 50,
                watchlist = list(mod = dmodel, val = dvalid))

pred <- predict(m1, dvalid)
summary(pred)

imp <- xgb.importance(model = m1, feature_names = colnames(X))

head(imp, 30)


## Select a sequence of threshold values
matt <- data.table(thresh = seq(0.0, 0.998, by = 0.001))

# Calculate mcc scores for the threshold values
matt$scores <- sapply(matt$thresh, FUN =
                        function(x) mcc(Y[valid], (pred > x) * 1))


# Select the max mcc score
opt <- matt[which.max(matt$scores), ]
print(opt)
```

```
pred_bin <- ifelse((pred > opt$thresh), 1, 0)
table(Y[valid], pred_bin)
mcc(Y[valid],pred_bin)
```

References

Bendel, 0. (2018): Industrie 4.0, online in the Internet: https://wirtschaftslexikon.gabler.de/definition/industrie-40-54032/version-277087 , *(Gabler Wirtschaftslexikon)*, Date: 19.02.2018, Request Date: 01.12.2018, 19.30.

Bonaccorso, G. (2018): Mastering Machine Learning Algorithms, E-Book, Birmingham, 2018, Web-ISBN-13: 978-1-78862-590-6 , Retrieved from: https://proquest.safaribooksonline.com/book/programming/machine-learning/9781788621113, Request date: 08.11.2018, 11.00

Bonaccorso, G. (2018): Machine Learning Algorithms -Second Edition, E-Book, 2nd Edition, Birmingham, 2018, Web-ISBN-13: 978-1-78934-548-3 , Retrieved from: https://proquest.tech.safaribooksonline.de/book/programming/machine-learning/9781789347999, Request Date: 08.11.2018, 12.00.

Bonnin, R. (2017): Machine Learning for Developers, E-Book, Birmingham, 2017, Web ISBN-13: 978-1-78646-696-9, Retrieved from: https://proquest.safaribooksonline.com/book/programming/machine-learning/9781786469878, Request Date: 21.11.2018, 9.30.

Bosch (2018): Bosch Production Line Performance, online in the internet: https://www.kaggle.com/c/bosch-production-line-performance, (*Kaggle)*, Date: 17.08.2016, Request Date: 15.11.2018, 16.45.

Boschetti, A. and Massaron, L. (2016): Python Data Science Essentials - Second Edition, E-Book, 2nd Edition, Birmingham, 2016, Web ISBN-13: 978-1-78646-283-1, Retrieved from: https://proquest.tech.safaribooksonline.de/book/programming/python/9781786462138, Request Date: 08.11.2018, 16.30.

Bosch Data (2016): Bosch Production Line Performance: Data, online in the internet: https://www.kaggle.com/c/bosch-production-line-performance/data, *(Kaggle)*, Date: 17.08.2016, Request Date: 27.09.2018, 9.15.

Bosch Data (2016): Bosch Production Line Performance: Data, online in the internet: https://www.kaggle.com/c/bosch-production-line-performance#evaluation, *(Kaggle)*, Date: 17.08.2016, Request Date: 20.12.2018, 00.15.

Bosch Leaderboard (2016): Bosch Production Line Performance: Leaderboard, online in the internet: https://www.kaggle.com/c/bosch-production-line-performance/leaderboard, (*Kaggle)*, Date: 17.08.2016, Request Date: 05.12.2018, 13.30.

Boughorbel, S., Jarray, F. and El-Anbari, M. (2017): Optimal classifier for imbalanced data using Matthews Correlation Coefficient metric, in: *PLoSONE*, Volume 12, Issue: 6, pp. e0177678.

Brexer, F. (2008): Mikroökonomik, (Eine Einführung), 4th Edition, Heidelberg, 2008

Bundesministerium für Wirtschaft und Energie (2018): Was ist Industrie 4.0?, online in the internet: https://www.plattform-

i40.de/I40/Navigation/DE/Industrie40/WasIndustrie40/was-ist-industrie-40.html, *(Plattform Industrie 4.0)*, Date: n.a., Request Date: 05.12.2018, 10.15.

Cakmak, M. U. (2018): Mastering Numerical Computing with NumPy, E-Book, Birmingham, 2018, Web ISBN-13: 978-1-78899-684-6, Retrieved from: https://proquest.safaribooksonline.com/book/databases/business-intelligence/9781788993357, Request Date: 10.12.2018, 21.15.

Chen, T., He, T., Benesty, M., Kohtilovich, V., Tang, Y., Cho, H., Chen, K., Mitchel, R., Cano, I., Zhou, T., Li, M., Xie, J., Lin, M., Geng, Y. and Li, Y. (2018): Package 'xgboost', online in the internet: https://cran.r-project.org/web/packages/xgboost/xgboost.pdf, *(r-project)*, Date: 08.06.2018, Request Date: 12.12.2018, 15.00.

Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C. and Wirth, R. (2000): CRISP-DM 1.0 Step-by-step data mining guide, *Working Paper*, Date: 2000.

Chawla, N. V., Bowyer, K. W., Hall, L. O. and Kegelmeyer, W. P. (2002): SMOTE: synthetic minority over-sampling technique, in: *Journal of artificial intelligence research*, Volume 16, pp. 321-357.

Efron, B. (1979): Bootstrap methods: another look at the jackknife. in: *The Annals of Statistics*, Volume 7, Issue 1, pp. 1-26.

Elkan, C. (2001): The foundations of cost-sensitive learning. In: *International joint conference on artificial intelligence*, Volume 2, pp. 973-978.

Erl, T., Khattak, W. and Buhler, P. (2016): Big Data Fundamentals: Concepts, Drivers & Techniques, E-book, Crawfordsville, 2016, Web-ISBN: 978-0-13-429118-5, Retrieved from: https://proquest.tech.safaribooksonline.de/book/databases/business-intelligence/9780134291185, Request date: 06.11.2018, 21.30.

Fahrmeier, L., Heumann, C., Künstler, R., Pigeot, I. and Tutz, G. (2016): Statistik – Der Weg zur Datenanalyse, 8th Edition, Heidelberg, 2016.

Fernández-Delgado, M., Cernadas, E., Barro, S. and Amorim, D. (2014): Do we need hundreds of classifiers to solve real world classification problems?, in: *The Journal of Machine Learning Research*, Volume: 15, Issue: 1, pp. 3133-3181.

Fuentes, A. (2018): Mastering Predictive Analytics with scikit-learn and TensorFlow, E-Book, Birmingham, 2018, Web-ISBN: 978-1-78961-224-0, Retrieved from: https://proquest.tech.safaribooksonline.de/book/programming/machine-learning/9781789617740, Request date: 04.12.2018, 11.45.

Goreman, B. (2016): A Kaggle Master Explains Gradient Boosting, online in the internet: http://blog.kaggle.com/2017/01/23/a-kaggle-master-explains-gradient-boosting/, *(Kaggle)*, Date: 01.23.2017, Request Date: 21.11.2018, 19.45.

James, G., Witten, D., Hastie, T., and Tibshirani, R. (2017): An introduction to statistical learning, 8th Edition, New York, 2017.

Kaggle (2018): Kaggle is the place to do data science projects, online in the internet: https://www.kaggle.com/, *(Kaggle)*, Date: n.a., Request Date: 10.12.2018, 17.00.

Kuhn, M. (2018): Package ‘caret’, online in the internet https://cran.r-project.org/web/packages/caret/caret.pdf, *(r-project)*, Date: 20.11.2018, Request: 12.12.2018, 18.00.

Kotu, V. and Desphande, B. (2015): Predictive Analytics and Data Mining, (Concepts and Practice with RapidMiner), Waltham, 2015.

Kurgan, L. and Musilek, P. (2006): A survey of Knowledge Discovery and Data Mining process models. *The Knowledge Engineering Review*, Volume 21, Issue 1, pp. 1-24.

Larose, D.T. and Larose, C. D. (2015): Data mining and predictive analytics, 2nd Edition, Hoboken, 2015.

Laurae (2016): Numeric Features: 2-D missing patterns (t-SNE), online in the internet: https://www.kaggle.com/c/bosch-production-line-performance/discussion/23067, *(Kaggle)*, Date: 2016, Request Date: 09.11.2018, 18.45.

Lesmeister, C. (2017): Mastering Machine Learning with R – Second Edition, E-Book, 2nd Edition, Birmingham, 2017, Web ISBN-13**:** 978-1-78728-448-7, Retrieved from: https://proquest.tech.safaribooksonline.de/book/programming/machine-learning/9781787287471, Request date: 06.11.2018, 14.00.

Lewis (2016): bish, bash, xgboost, Online in the internet: https://www.kaggle.com/cartographic/bish-bash-xgboost, *(Kaggle)*, Date: mid of 2016, Request Date: 12.12.2018, 15.30.

Liaw, A. and Wiener, M. (2018): Package ‘randomForest‘, online in the internet https://cran.r-project.org/web/packages/randomForest/randomForest.pdf, *(r-project)*, Date: 22.03.2018, Request Date: 12.12.2018, 14.15.

Liu, Y., Chawla, N. V., Harper, M. P., Shriberg, E., and Stolcke, A. (2006): A study in machine learning from imbalanced data for sentence boundary detection in speech, in: *Computer Speech & Language*, Volume 20, pp. 468-494.

Maaten, L. V. D. (2018): FAQ : Once I have a t-SNE map, how can I embed incoming test points in that map?, Online in the internet: https://lvdmaaten.github.io/tsne/ , *(Ivdmaaten)*, Date: 2018, Request: 11.11.2018, 15.00.

Maaten, L. V. D. and Hinton, G. (2008): Visualizing data using t-SNE, in: *Journal of machine learning research*, Volume 9, pp. 2579-2605.

Meister, A. (1999): Numerik linearer Gleichungssysteme, (Eine Einführung in moderne Verfahren), Wiesbaden, 1999.

Miller, J. D. (2017): Statistics for Data Science, E-Book, Birmingham, 2017, Web ISBN-13: 978-1-78829-534-5, Retrieved from: https://proquest.safaribooksonline.com/book/statistics/9781788290678, Request Date: 20.11.2018, 12.15.

Miller, J. D. and Forte, R. M. (2017): Mastering Predictive Analytics with R – Second Edition, E-book, 2nd Edition, Birmingham, 2017, Web ISBN-13**:** 978-1-78712-435-6, Retrieved from: https://proquest-tech-safaribooksonline-

de.eaccess.ub.tum.de/book/programming/r/9781787121393, Request date: 14.11.2018, 18.30.

Patreek, J. (2017): Artificial Intelligence with Python, E-Book, Birmingham, 2017, Web ISBN-13: 978-1-78646-967-0, Retrieved from: https://proquest.safaribooksonline.com/book/programming/python/9781786464392, Request Date: 09.11.2018, 10.45.

Pereira, A. C., and Romero, F. (2017): A review of the meanings and the implications of the Industry 4.0 concept, in: *Procedia Manufacturing*, Volume 13, pp. 1206-1214.

Polani D. (2013): Kullback-Leibler Divergence, in: (eds) Dubitzky W., Wolkenhauer O., Cho KH. and Yokota H. (2013): *Encyclopedia of Systems Biology*, New York, 2013.

Roth, A. (2016): Einführung und Umsetzung von Industrie 4.0, (Grundlagen, Vorgehensmodell und Use Cases aus der Praxis), Heidelberg, 2016.

Scnndl (2016): 4th place solution, Online in the internet: https://www.kaggle.com/c/bosch-production-line-performance/discussion/25370, *(Kaggle)*, Date: mid of 2016, Request Date: 20.12.2018, 10.30.

Segaran, T. (2007): Programming collective intelligence, (Building smart web 2.0 applications), Sebastopol, 2007.

Shearer, C. (2000): The CRISP-DM model: the new blueprint for data mining. *Journal of data warehousing*, Volume *5*, Issue 4, pp. 13-22.

Therneau, T. M.,Atkinson, B. and Ripley, B. (2018): Package 'rpart', online in the internet: https://cran.r-project.org/web/packages/rpart/rpart.pdf, *(r-project)*, Date: 23.02.2018, Request Date: 12.12.2018, 14.15.

Therneau, T.M., Atkinson, E.J. and Mayo Foundation (2018): An Introduction to Recursive Partitioning Using the RPART Routines, online in the internet: https://cran.r-project.org/web/packages/rpart/vignettes/longintro.pdf, *(r-project)*, Request Date: 17.11.2018, 17.00.

Thoben, K. D., Wiesner, S. and Wuest, T. (2017): „Industrie 4.0" and smart manufacturing–a review of research issues and application examples, in: *International Journal of Automation Technology*, Volume 11, Issue 1, pp 4-16.

Waring, C. (2017): Kaggle Bosch Competition, online in the internet: https://github.com/waringc/Kaggle-Bosch-Competition, *(GitHub)*, Date: 08.01.2017, Request Date: 07.11.2018, 09.15.

Winters, R. (2017): Practical Predictive Analytics, E-Book, Birmingham, 2017, Web-ISBN-13: 978-1-78588-046-9, Retrieved from: https://proquest.tech.safaribooksonline.de/book/databases/business-intelligence/9781785886188, Request Date: 07.11.2018, 10.15.

World Economic Forum (2017): Technology and Innovation for the Future of Production: Accelerating Value Creation, *White Paper*, Switzerland, 2017

Affirmation

Declaration of Authorship

I hereby declare that the thesis submitted is my own unaided work. All direct or indirect sources used are acknowledged as references.

I am aware that the thesis in digital form can be examined for the use of unauthorized aid and in order to determine whether the thesis as a whole or parts incorporated in it may be deemed as plagiarism. For the comparison of my work with existing sources I agree that it shall be entered in a database where it shall also remain after examination, to enable comparison with future theses submitted. Further rights of reproduction and usage, however, are not granted here.

This paper was not previously presented to another examination board and has not been published.

Ehrenwörtliche Erklärung

Ich erkläre hiermit ehrenwörtlich, dass ich die vorliegende Arbeit selbständig angefertigt habe. Die aus fremden Quellen direkt und indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

Ich weiß, dass die Arbeit in digitalisierter Form daraufhin überprüft werden kann, ob unerlaubte Hilfsmittel verwendet wurden und ob es sich – insgesamt oder in Teilen – um ein Plagiat handelt. Zum Vergleich meiner Arbeit mit existierenden Quellen darf sie in eine Datenbank eingestellt werden und nach der Überprüfung zum Vergleich mit künftig eingehenden Arbeiten dort verbleiben. Weitere Vervielfältigungs- und Verwertungsrechte werden dadurch nicht eingeräumt.

Die Arbeit wurde weder einer anderen Prüfungsbehörde vorgelegt noch veröffentlicht.

_____ _____

(Location, Date) (Name)