Hochschule für Technik
und Wirtschaft Berlin
University of Applied Sciences

MPMD 2.2 Final Report

Predictive Modeling in the Semiconductor Industry: Application of CRISP-DM to the SECOM Case Study

From
King/Catherine; s0572057
Lin/Yi Hsuan; s0572048
Poboon/Pawin; s0572079
Yolagel/Ender; s0572050

17.07.2020

Department 3
Project Management and Data Science
(MPMD)

Supervisor: Prof. Dr. Tilo Wendler

Table of Contents

List of Figures

List of Tables

1. Introduction

Data mining is the process of finding patterns within and building models of data. With companies and organizations collecting more and more data every day, data mining has become that much more important across a myriad of industries in maximizing profits, minimizing delays, and other benefits. In order to maximize results and minimize costs and risks, various processes were developed to supply data scientists with guidance through an otherwise complex endeavor.

The purpose of this paper is to examine the CRISP-DM process, one of the leading procedures for data mining from IBM, and apply it to a specific example from the semiconductor industry. The goal is to build a model which accurately and effectively predict faulty semiconductor wafers using as few inputs as possible, also known as a parsimonious model.

2. CRISP-DM

CRISP-DM is the cross-industry specific process for data mining as published by IBM (IBM, 2019). This process is an iterative, structured framework which clearly defines the steps to be taken in a data mining task. As it is "cross-industry", the process can be applied to any business case. The process consists of 5 iterative steps: business understanding, data understanding, data preparation, modeling, and evaluation. The general sequence between steps should move from 1-6, however it's recommended for users to move back and forth to revisit previous steps as illustrated in *Figure 1*. The CRISP-DM model can be adjusted to each specific business case and is therefore known for its flexibility. CRISP-DM can be used both as a methodology and as a process model to guide all data mining efforts (IBM, 2019).

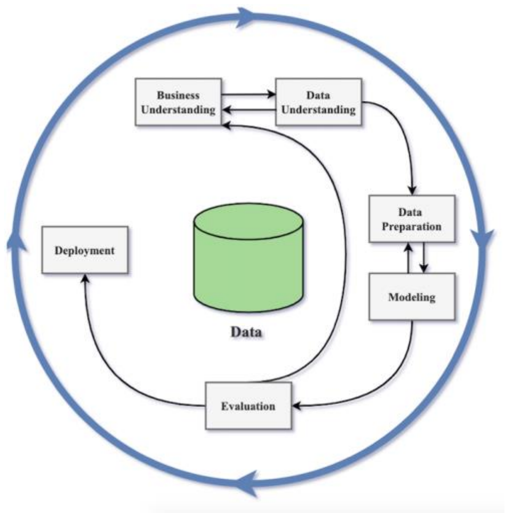
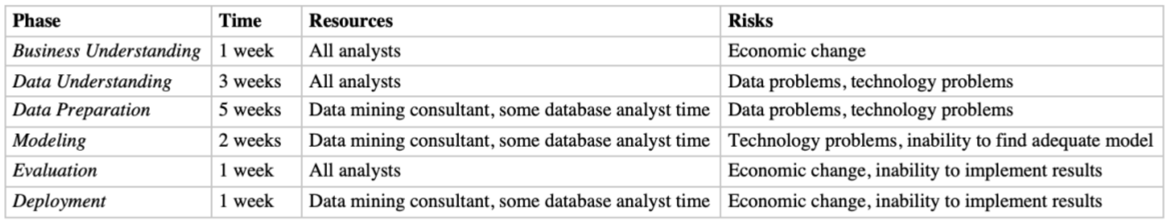


Figure 1 CRISP-DM Process Guide.
Reproduced with modification from IBM SPSS Modeler CRISP-DM Guide

2.1 Business Understanding

The first step of the CRISP-DM process is the business understanding. Before one starts with any aspect of data mining, it's vital to explore the motivation behind the process and what the business or organization hopes to gain from the situation. This will keep all those involved on the aligned and working towards the same goal, which is crucial as the data mining steps become more complex. After the business objective is determined, information about the business background should be collected. This includes determining the organizational structure, describing the problem area, and describing the current situation. These three steps assist in the team's understanding of available resources, problems, and goals (IBM, 2019).

Next, a project plan should be produced, which should serve as a master document for all data mining work (IBM, 2019). This plan should inform all those involved of the project objectives, risks, and timetable throughout the entire data mining process. An example project plan can be seen in *Table 1*.



Phase	**Time**	**Resources**	**Risks**
Business Understanding	1 week	All analysts	Economic change
Data Understanding	3 weeks	All analysts	Data problems, technology problems
Data Preparation	5 weeks	Data mining consultant, some database analyst time	Data problems, technology problems
Modeling	2 weeks	Data mining consultant, some database analyst time	Technology problems, inability to find adequate model
Evaluation	1 week	All analysts	Economic change, inability to implement results
Deployment	1 week	Data mining consultant, some database analyst time	Economic change, inability to implement results

Table 1 CRISP-DM Project Plan reproduced with modification from IBM SPSS Modeler CRISP-DM Guide

Before moving to the next phase of the CRISP-DM process, one must be able to answer and address the follow:

- What is the business goal of this endeavor?
- What is the concrete definition of "done"?
- Are the budget and resources in order to achieve this definition of "done" available?

In addition to the aforementioned business-centric questions, the following questions from a data mining perspective must also be answered:

- Why and how will data mining assist in reaching the objective?
- How will it be clear that the objective has been met?
- How will the results be deployed?

Once these questions are answered, the next step is to take a closer look at the data.

2.2 Data Understanding

In the data understanding step of the CRISP-DM procedure, the data is examined closely to identify any unexpected issues and prepare the user for the next phase, the data preparation, which tends to be the longest and most complex phase of the project. This step involves the collection and exploration of the data, which should give the user a clear idea of the data's contents in terms of quantity and quality (IBM, 2019).

In the initial collection of the data, the following questions shall be addressed:

- What is the structure of the data?
- From where is the data collected?
- Can any attributes be immediately ignored and which seem most important?

Only these broad questions can be answered during the data collection, as the user has not yet had the opportunity to fully dive into the data for analysis. The answers, though, can already be utilized to bring the user to a greater understanding of the data through the writing of a data collection report. The goal of such reports is to seamlessly navigate the user through the next phase of CRISP-DM, the data preparation (IBM, 2019). A data collection report typically focusses on the quality and quantity of the data, including information on amount (size), value types (formats), and coding schemes (representative values).

The information gathered in the data collection report can then be improved upon in form of a data description report. This report should build upon the information previously collected and should help address the data mining objectives defined during the business understanding step, as well as to assist in developing hypotheses about the data. In building on the information collected previously, the following questions can be asked regarding the quantity and quality of the data:

- Which format is the data presented in?
- How was the data extracted?
- How large is the dataset?
- Are there characteristics of the data which already seem relevant to the business understanding?
- What is the type or format of the data (numeric, binary)?
- Can basic statistics be computed and are there any insights from these?
- Can features or attributes already be prioritized in terms of the business understanding?

In order to address and answer the above questions, the data must be explored through use of charts, tables, and visualizations, which should help the user already start to create hypotheses. The quality of the data can be examined through analysis of missing data, data and measurement errors, and coding inconsistencies. Any missing attributes, values, or features should be addressed and analyzed. In order to successfully treat missing values in the future steps of the CRISP-DM process, the user must fully understand the extent of the data's missingness. If possible, the user must attempt to understand the meaning behind the missing values. In addition to missing values, inconsistencies known as "noise", when severe, can also prove problematic in future steps. To identify this "noise", a plausibility check on the values can be run which can distinguish any supposed conflicts within the data. The user can at this point decide to remove or exclude certain data which would not have an impact on the project's hypotheses.

Before moving forward to the next step of the CRISP-DM process, the user must be sure that the data is well understood. All data sources should be clearly distinguished and accessed, and problems or potential problems must be identified. Any issues caused by the data's quality should be recognized, as well as a possible plan to address these issues. Key attributes or features should have also been identified, which should have already assisted in the formulation of initial hypotheses. The hypotheses should have also already been adjusted and reshaped given the information gathered during the data exploration.

2.3 Data Preparation

The next step of the CRISP-DM methodology is arguably the most important and often most time-consuming. With a comprehensive business and data understanding, the user will run into fewer bottlenecks during the data preparation, however it is known for taking about 50-70% of the entire procedure (IBM, 2019). The goal of data preparation is to adapt the data in such a way to later build the best model possible. Data preparation can be broken into two sub-steps: data cleaning and data preprocessing.

In data cleaning, potential issues such as missing data and data errors or inconsistencies are analyzed and mitigated. Mitigation techniques include removal of any feature above a certain threshold of missing values (in percent) and replacing any outlier (any value which is significantly greater or less than the average) with a missing value (NA) (Kuhn and Johnson, 2013). The data quality report from the data understanding phase can guide the user as to which techniques should be implemented. Once the potentially problematic data are mitigated, a new subset of data can be created or constructed. This can be done in two ways: deriving features (columns/characteristics) or generating new records (rows) (IBM, 2019). The tacit knowledge gained from both the business and data understanding should help the user to consider which option would be most beneficial to the model, however statistical approaches may also be taken in form of feature selection.

The data processing phase ensures that the data are in the best possible format before the modelling procedure begins. This includes steps such as balancing methods, scaling or normalization, transformation, and cross validation. Without these steps, all the effort that went into the previous steps would not be recognized, as the data would not be in the best form to create the best model. Each modelling algorithm can assume differences in distribution, in data types, or in rebalanced data, and these aspects should be addressed before moving on to the modelling phase (IBM, 2019).

2.4 Modelling

At this point in the CRISP-DM process, the data should be well prepared to start building models. It typically takes multiple attempts to create and perfect the best possible model for the data. Throughout the modelling process, one must keep track of the parameter settings, the models produced, and the descriptions of the model results in order to adjust and improve after each iteration.

Involved in most modelling techniques are various parameters, also known as settings, which can be fine-tuned to control the modelling process (IBM, 2019). In a standard case, a model will first be built using the default parameters, and then the model will be refined during the next iterations. Detailed notes should be kept during this process in order to keep track of the decisions made during refinement, which will ease the process in choosing the best fit model, as well as in model automation or rebuilding.

Using the information gathered during the modelling phase, a closer look is now required to determine which models are conclusive and sufficient for the specific use case. The business understanding from the first phase of the CRISP-DM procedure comes back into play again as the user must remember the original objective of the data mining task. The definition of this objective will determine what a “final” model means to the user: either the model is ready to deploy or it simply illustrates interesting patterns within the data To judge a model’s success in general, one should analyze and speculate the conclusions found by the model, as well as any new insights or unusual patterns. There must be a clear understanding of the model’s results and these should not include any inconsistencies in need of further explanation.

2.5 Evaluation

The last and final step of the CRISP-DM methodology is the model evaluation and is only achievable if the models built during the previous phase fulfill the data mining success criteria defined in the initial steps. In this phase, the results are assessed to determine whether and to what extent they meet the business success criteria, which was set during the first phase. The results must be clearly stated and in a presentable form, with specific mention of particularly interesting findings. Often time, results from a data mining effort will illuminate new questions, which should be addressed and answered if possible. It's possible that multiple models fulfill the data mining and business goals of the project, possibly in different degrees, and for this reason it's acceptable to evaluate and deploy multiple models.

Given the iterative nature of the CRISP-DM methodology, the evaluation phase gives the user a chance to reflect on the process as a whole and improve for future endeavors. At this point, the user can determine the next steps in light of the business goals for data mining. The model can be deployed or the user can continue to refine or even replace the already developed models, should the results not yet be optimal. In order to make this decision, the accuracy and relevancy of the modelling results should be analyzed.

3. Case Study

3.1 Semiconductor Industry

The semiconductor industry is known for its highly complex and technology-driven manufacturing processes (Munirathinam and Ramadoss, 2016). The production process consists of hundreds of steps involving slight modifications and additions, with sensors recording small amounts of data throughout, producing an enormous amount of monitoring data. Today, these sensors are able to collect real-time data during the production process, which provides the opportunity for quality control and optimization. Fault detection plays a vital role in the semiconductor industry, as it gives way for the introduction of a decision support system (DSS) which could ultimately save resources during the production cycle, should faulty wafers be detected accurately in real-time (Kerdprasop and Kerdprasop, 2011). Given the vast amount of data collected in such a short time span, the analysis and later prescriptive analytics to identify faulty wafers often prove to be too overwhelming.

3.2 SECOM Case Study

The SECOM (Semiconductor Manufacturing) dataset consists of 1567 samples taken from a wafer production line, including all sensor data, timestamps, and classification data (whether or not the wafer was faulty). The data contain measurements across 590 sensors, and are made up of 104 fail cases which are labeled as 1 (less than 7% of all records). Given this imbalance between the pass and fail examples, the dataset proves to be not only a realistic example from the semiconductor industry, but also a complex dataset to analyze using the CRISP-DM process, with the goal of accurately and efficiently predicting faulty wafers.

4. Implementation of CRISP-DM on the SECOM case

4.1 Business Understanding: SECOM

The semiconductor industry is known to have a complex and sophisticated production process with many steps. Thus, default detention plays an important role to smooth the productivity, prevent breakdowns on the production line, and reduce related costs. To be able to predict the defaults, a large number of data must be collected and analyzed. According to recent technologies, the semiconductor industry is able to monitor and store data via sensors or features. Utilizing this data will assist the industry in predicting defaults on fewer sensors and features.

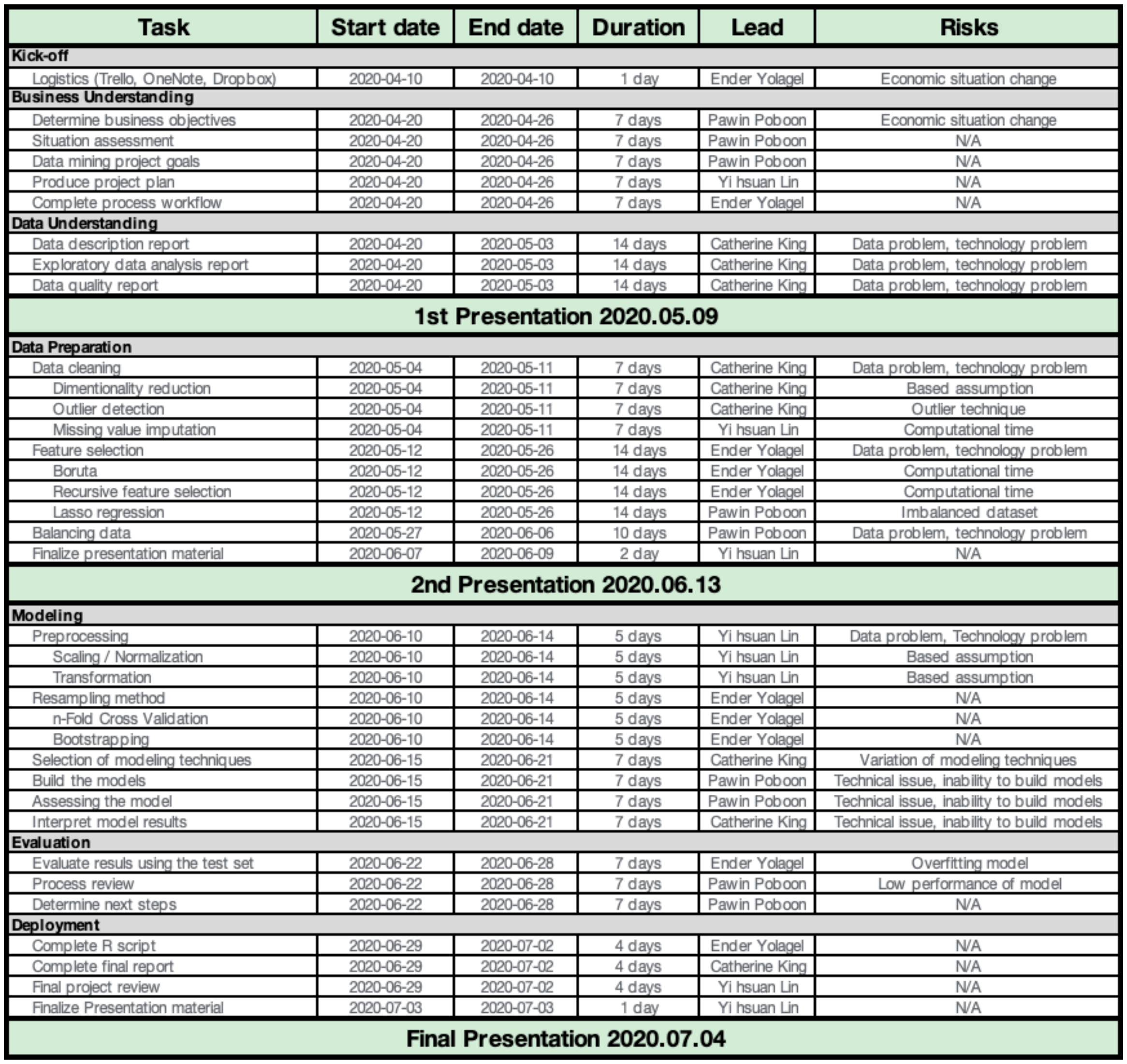
The business success criteria of this case study are to proceed through the CRISP-DM methodology to build a parsimonious model which accurately and efficiently predicts defaults in the SECOM dataset. A parsimonious model uses the least amount of input variables as possible and runs as quickly as possible.

The available resources for this case study are:

- Four data scientists
- 10 working hours per scientist per week with possibility of additional overtime
- Project start 10.04.2020 – project end 04.07.2020
- 4 MacBook Pro Personal Computers (2-core Intel Core i5, RAM 8, SSD 256GB)
- R Server 3.6.3

The data mining success criteria include the running R-Script which follows the CRISP-DM methodology to analyze, clean, prepare, model, evaluate and deploy a parsimonious model with as high accuracy and efficiency as possible, proper and complete documentation, and a timely delivery by or before the deadline.

An initial project plan was developed as per the CRISP-DM guidelines, which can be found in *Table 2*. Given the methodology's iterative nature, the project plan must not be adhered to exactly, however it will function as a helpful guideline and reminder of future steps.



Task	Start date	End date	Duration	Lead	Risks
Kick-off					
Logistics (Trello, OneNote, Dropbox)	2020-04-10	2020-04-10	1 day	Ender Yolagel	Economic situation change
Business Understanding					
Determine business objectives	2020-04-20	2020-04-26	7 days	Pawin Poboon	Economic situation change
Situation assessment	2020-04-20	2020-04-26	7 days	Pawin Poboon	N/A
Data mining project goals	2020-04-20	2020-04-26	7 days	Pawin Poboon	N/A
Produce project plan	2020-04-20	2020-04-26	7 days	Yi hsuan Lin	N/A
Complete process workflow	2020-04-20	2020-04-26	7 days	Ender Yolagel	N/A
Data Understanding					
Data description report	2020-04-20	2020-05-03	14 days	Catherine King	Data problem, technology problem
Exploratory data analysis report	2020-04-20	2020-05-03	14 days	Catherine King	Data problem, technology problem
Data quality report	2020-04-20	2020-05-03	14 days	Catherine King	Data problem, technology problem
1st Presentation 2020.05.09					
Data Preparation					
Data cleaning	2020-05-04	2020-05-11	7 days	Catherine King	Data problem, technology problem
Dimentionality reduction	2020-05-04	2020-05-11	7 days	Catherine King	Based assumption
Outlier detection	2020-05-04	2020-05-11	7 days	Catherine King	Outlier technique
Missing value imputation	2020-05-04	2020-05-11	7 days	Yi hsuan Lin	Computational time
Feature selection	2020-05-12	2020-05-26	14 days	Ender Yolagel	Data problem, technology problem
Boruta	2020-05-12	2020-05-26	14 days	Ender Yolagel	Computational time
Recursive feature selection	2020-05-12	2020-05-26	14 days	Ender Yolagel	Computational time
Lasso regression	2020-05-12	2020-05-26	14 days	Pawin Poboon	Imbalanced dataset
Balancing data	2020-05-27	2020-06-06	10 days	Pawin Poboon	Data problem, technology problem
Finalize presentation material	2020-06-07	2020-06-09	2 day	Yi hsuan Lin	N/A
2nd Presentation 2020.06.13					
Modeling					
Preprocessing	2020-06-10	2020-06-14	5 days	Yi hsuan Lin	Data problem, Technology problem
Scaling / Normalization	2020-06-10	2020-06-14	5 days	Yi hsuan Lin	Based assumption
Transformation	2020-06-10	2020-06-14	5 days	Yi hsuan Lin	Based assumption
Resampling method	2020-06-10	2020-06-14	5 days	Ender Yolagel	N/A
n-Fold Cross Validation	2020-06-10	2020-06-14	5 days	Ender Yolagel	N/A
Bootstrapping	2020-06-10	2020-06-14	5 days	Ender Yolagel	N/A
Selection of modeling techniques	2020-06-15	2020-06-21	7 days	Catherine King	Variation of modeling techniques
Build the models	2020-06-15	2020-06-21	7 days	Pawin Poboon	Technical issue, inability to build models
Assessing the model	2020-06-15	2020-06-21	7 days	Pawin Poboon	Technical issue, inability to build models
Interpret model results	2020-06-15	2020-06-21	7 days	Catherine King	Technical issue, inability to build models
Evaluation					
Evaluate resuls using the test set	2020-06-22	2020-06-28	7 days	Ender Yolagel	Overfitting model
Process review	2020-06-22	2020-06-28	7 days	Pawin Poboon	Low performance of model
Determine next steps	2020-06-22	2020-06-28	7 days	Pawin Poboon	N/A
Deployment					
Complete R script	2020-06-29	2020-07-02	4 days	Ender Yolagel	N/A
Complete final report	2020-06-29	2020-07-02	4 days	Catherine King	N/A
Final project review	2020-06-29	2020-07-02	4 days	Yi hsuan Lin	N/A
Finalize Presentation material	2020-07-03	2020-07-03	1 day	Yi hsuan Lin	N/A
Final Presentation 2020.07.04					

Table 2 Project Plan for SECOM Case Study

In addition to the project plan, a detailed CRISP-DM process workflow specific to the SECOM case study was developed in *Figure 2*. This follows the same general idea as the original CRISP-DM process flow as shown in *Figure 1* with additional details and steps specific to the case study.

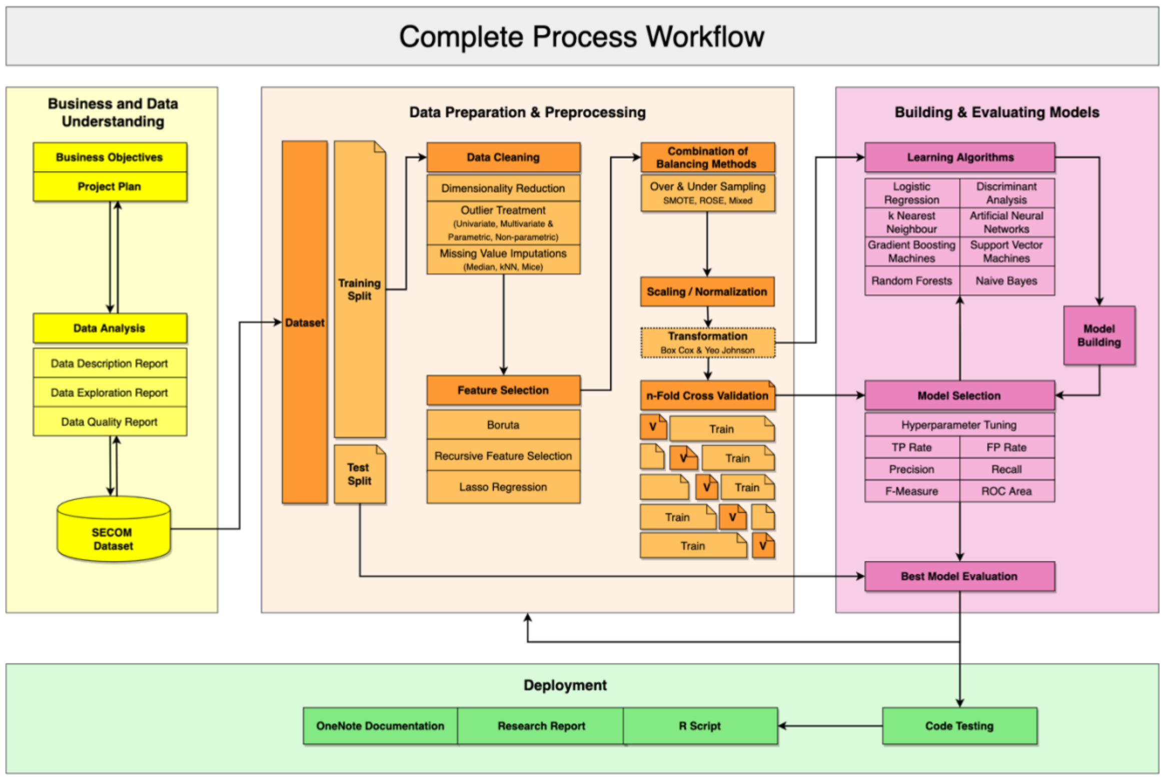


Figure 2 CRISP-DM Process Workflow Adjusted for SECOM Case Study

4.2 Data Understanding: SECOM

4.2.1 Data Collection Report

In order to fully prepare for future steps in the CRISP-DM process, the data was examined and analyzed to give a deeper understanding of its contents. Initially, the data quantity is examined. The SECOM dataset, as mentioned previously, can be found in one .sav file and includes 593 columns and 1567 rows. The columns are made up of 590 features (presumably various sensors on the production line), plus the row ID, a timestamp, and a binary classification (0 = pass, 1 = fail). Out of the 1567 records, about 93% are classified as “pass” (1472 records) and about 6% as “fail” (95 records). This information was gathered into a Data Quantity Chart in *Table 3*, as instructed by the IBM CRISP-DM guide.

Criteria	**Results**
What is the format of the data?	Data comes in one .sav file. Feature names no given.
How is data captured?	Data is in .sav format. This is not the original export of the data, but a merged version.
How large is the data (in numbers of rows and columns)?	590 features plus 3 (ID, timestamp, and class [0 = pass, 1 = fail). 1567 entries, with 1472 "pass" and 95 "fail”.

Table 3 Data Quantity Chart

Next, the data quality was analyzed, as seen in *Table 4*. All features (excluding the ID, timestamp, and classification) were found to be numeric and the classification was found to be binary (either 0 or 1). Many missing values were identified, with up to 1429 missing values per feature. The features are identifiable by number values and no additional insight is given as the meaning or relationship behind or between the features. Given this lack of business insight, the missing values are assumed to be missing at random (MAR).

Criteria	**Results**
What data types are present?	Beside time stamps, all features are numeric; class or target variable is binary.
How are key attributes calculated? What insight does this provide into the business question?	Data for each feature determines "pass" or "fail" result.
Possible to prioritize specific attributes? Is there anyone to provide more insight?	No additional business insight possible.
Have you identified missing attributes and blank fields? If so, is there meaning behind such missing values?	First screening shows missing values, up to 1429 missing values per feature. Missing values classified as MAR.
Are there spelling inconsistencies that may cause problems in later merges or transformations?	No, all data is numerical

Table 4 Data Quality Chart

4.2.2 Univariate Analysis

A univariate analysis assisted in gaining a deeper insight on the individual features. Given the high number of features, the analysis remained quite broad, however it was still able to identify important aspects of the data. The data contains neither duplicated records, nor records which contain only missing values. 538 of the 590 features contain at least one missing value, with 28 features containing more than 50% missing values. 116 features contain 100% constant values (with 0% variance), and 473 out of the 474 remaining features with variance are not normally distributed. This information will prove to be vital in the future steps, specifically during dimensionality reduction and missing value imputation.

4.2.3 Multivariate Analysis

A brief multivariate analysis was also conducted given the volume of features. Over 220,000 correlations can be seen in the histogram in *Figure 3* through analyzing the Pearson Correlations between all features. 48 correlations of exactly +1 and 2 correlations of exactly -1 were identified, which is vital to consider for future statistical approaches, most notably the missing value imputation in section 4.3.3.

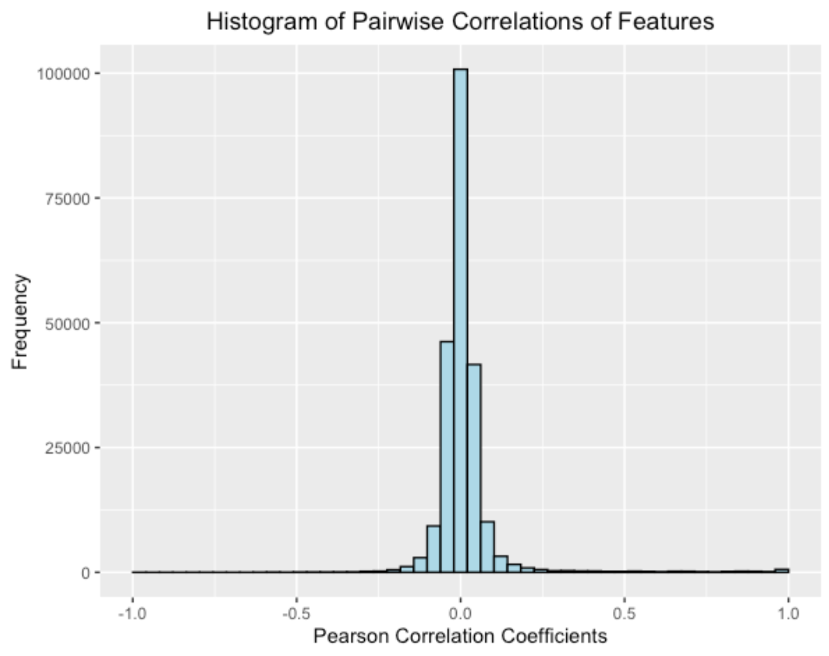


Figure 3 Histogram of Pairwise Correlations

4.2.4 Outlier Analysis

Outliers are values in a dataset which are notably far from the average data and can negatively affect predictive models (Kuhn and Johnson, 2013). Outlier treatment in the data mining process can drastically improve a model's performance, as a model will not be confused or mislead by this so-called "noise" (Kuhn and Johnson, 2013). The specific outlier treatment for the SECOM case will be discussed in section 4.3.2.2, but first the outliers in the dataset must be identified and analyzed to properly determine the most appropriate course of treatment.

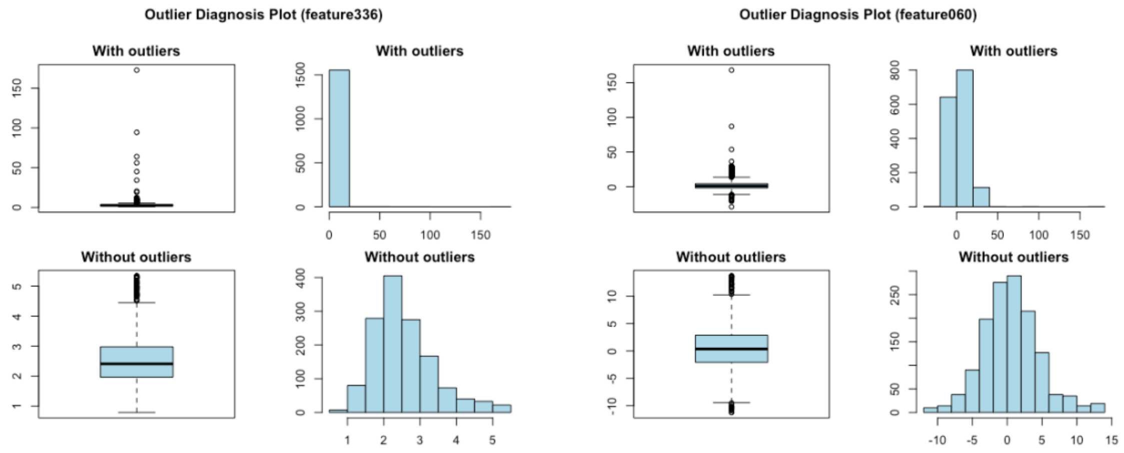


Figure 4 Outlier Diagnosis Plots for Features 336 and 060

In applying the 3-s rule (classifying outliers as any value outside a range of the mean plus and minus 3 standard deviations), over 2000 outliers were identified (Wendler and Gröttrup, 2016). A quick overview of some features were examined to display the effect these outliers have on these features' distribution, as seen in *Figure 4*. Please note, a box plot is designed using the interquartile range, not the 3-s rule, to identify outliers, which explains why all four boxplots below still technically identify outliers.

4.2.5 Missing Value Analysis

Like outlier treatment, identification and treatment of missing values will have a radical effect on the model's success. The missing value treatment will be handled in section 4.3.3, however it's important to first gain a broad understanding in this phase in order to determine an appropriate method for this treatment. A histogram of all features with between 0 and 100 missing features was created to gain oversight of the distribution of missing values in the SECOM dataset, as seen in *Figure 5*.

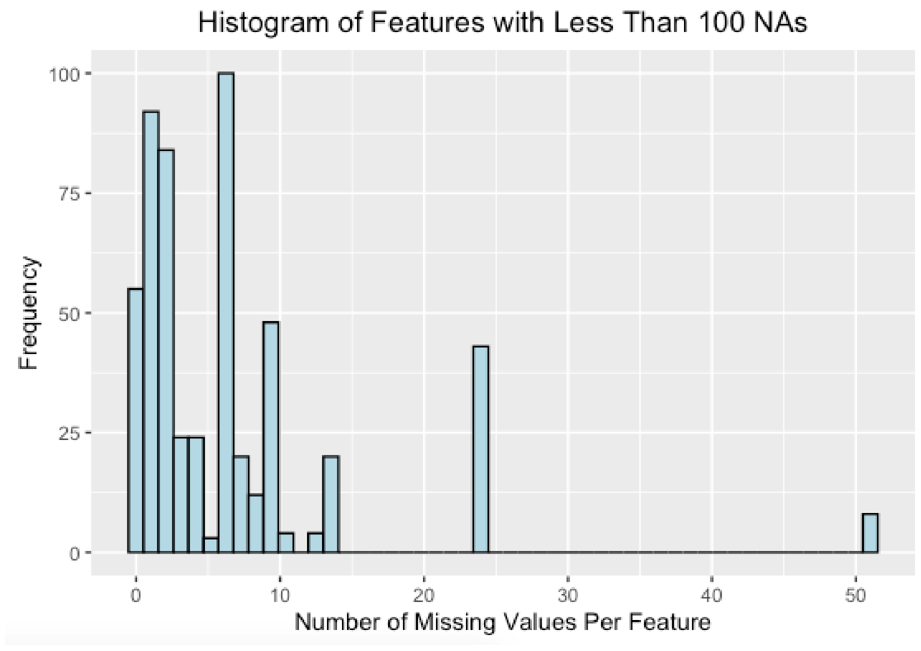


Figure 5 Histogram of Features with less than 100 missing values

4.3 Data Preparation: SECOM

The data preparation phase of CRISP-DM is, as previously mentioned, arguably the most important step in the model building process and therefore usually accounts for 50-70% of the entire procedure. The goal of data preparation is to adapt the data in such a way to later build the best model possible.

4.3.1 Separation of Test and Training Dataset

The very first step in the data preparation phase is the separation of the train and test datasets. The training dataset will be used to build the model and estimate the best parameters, while the test dataset is used only to examine the accuracy and exactitude of the model (Wendler and Gröttrup, 2016). Given these roles, once separated, the test dataset is put aside until after the models are built and the best are selected to test. There is discussion as to the perfect ratio to split the dataset and, given the size of the SECOM dataset, it was decided to split it 80% train and 20% test. In order to avoid sampling bias, the class proportion of the target feature in the original dataset (93% pass, 6% fail) was maintained in both the test and train datasets through the use of stratified random sampling (Kuhn and Johnson, 2013).

4.3.2 Data Cleaning

As previously mentioned, data cleaning is a vital step in the data preparation phase of CRISP-DM as it mitigates any potential issues within the data before the data preprocessing starts. For the purpose of the SECOM case study, the data cleaning includes dimensionality reduction and outlier treatment.

4.3.2.1 Dimensionality Reduction

As opposed to feature selection, which uses statistical approaches and algorithms during the data preprocessing step to select the most important features for a model, dimensionality reduction removes features which the user already knows will not add value to the model (Kerdprasop and Kerdprasop, 2011). In the SECOM dataset, 24 features containing 55% or more missing values were removed, as well as 116 features with 0% variance (Munirathinam and Ramadoss, 2016). All 140 of these features will not add value to the model and could greatly reduce effectiveness and accuracy, and are therefore removed without further testing. The dataset now contains 453 features.

4.3.2.2 Outlier Treatment

As determined in the data understanding, the SECOM dataset contains 2,732 outliers when applying the 3-s rule. Once these outliers are identified, these values are replaced by missing values (NAs) and will be imputed as all other missing values in the next step of the CRISP-DM procedure (Kuhn and Johnson, 2013).

4.3.3 Missing Value Imputation

Missing value imputation is one of the critical steps in the modelling process that has an indirect effect on the predictive power of the models (Kuhn and Johnson, 2019). In the SECOM case, ad-hoc solutions are not implemented due to their limitations of assumptions and weaknesses against non-normal distributions. Instead, three major imputation methods, Multiple Imputation, k-Nearest Neighbor and Random Forest, are implemented and their performances are measured. Due to its superior advantages, the multiple imputation method will be further discussed and explained.

Before any imputation operation, the missing data mechanism is inspected. For this purpose, a plot for missingness pattern is created to further examine features that have missing values more than 20% in *Figure 6*. While all eight features are missing together in nearly 25% of all observations, the first and second half of the features are missing together in another nearly 25% and 21% of observations respectively. This indicates data is missing at random (MAR) since the probability of being missing is same only within groups defined by the observed data (Van Buuren, 2018).

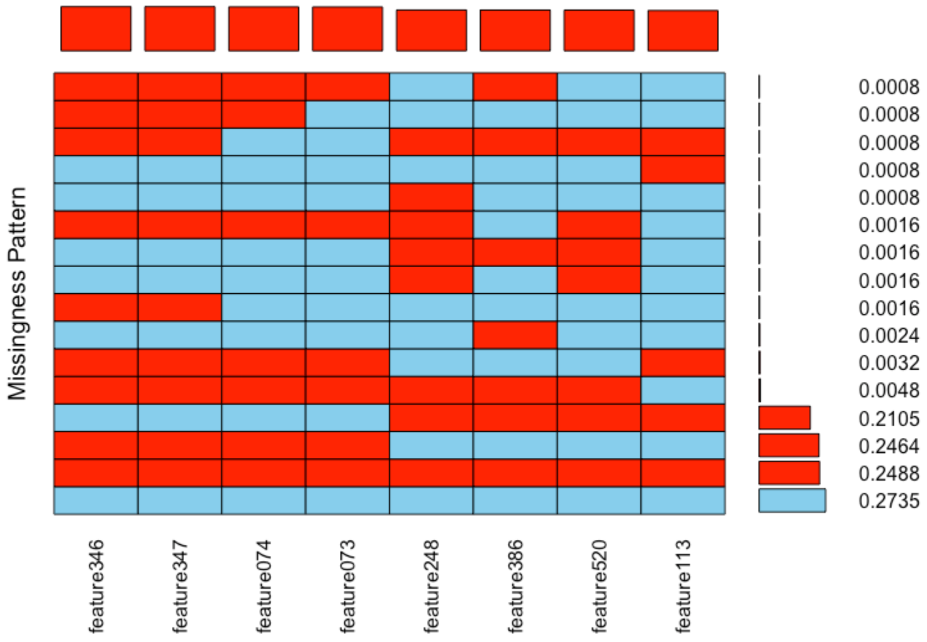


Figure 6 Features have missing values more than 20%.

Well-known ad-hoc solutions listed *Table 5* below are not implemented due to their limited ability of producing unbiased estimates (Van Buuren, 2018). It is seen that listwise deletion, pairwise deletion, and mean imputation produces an unbiased estimate of the mean if data is missing completely at random (MCAR), and these methods also produce standard error estimations that are too large.

	Unbiased			Standard Error
	Mean	**Reg Weight**	**Correlation**	
Listwise	MCAR	MCAR	MCAR	Too large
Pairwise	MCAR	MCAR	MCAR	Complicated
Mean	MCAR	–	–	Too small
Regression	MAR	MAR	–	Too small
Stochastic	MAR	MAR	MAR	Too small
LOCF	–	–	–	Too small
Indicator	–	–	–	Too small

Table 5 Overview of assumptions made by ad-hoc methods

Reproduced with modification from Van Buuren's Flexible Imputation of Missing Data - html version.

Alternatively, regression imputation and stochastic regression imputation have the assumption of MAR, yet they fail to understand and account for the uncertainty in the imputations (Van Buuren, 2018). They produce imputations drawn from a normal distribution while the SECOM dataset has many severely skewed variables with significant deviations from normality. Last observation carried forward (LOCF), baseline observation carried forward (BOCF), and indicator methods are also not considered due to their tendency of producing standard errors that are too small. Instead Multiple Imputation, k-Nearest Neighbor (kNN), and Random Forest imputation methods are implemented as they work with data missing at random and non-normally distributed.

4.3.3.1 Non-parametric Random Forest Imputation

The first method used is the non-parametric random forest imputation using the missForest package. It takes complex interactions and nonlinear relations into account and needs no tuning parameter. It does not require prior knowledge about the data as well. Random Forest often shows better predictive performance compared to simple statistical models. It provides an out-of-bag (OOB) imputation error estimate which is normalized root mean squared error (NRMSE) for continuous variables (Stekhoven and Bühlmann, 2011).

There are two disadvantages of missForest. The first is its computational cost; finding the optimum error in a speed-accuracy trade-off can be challenging as growing multiple random forests can be time-consuming. In the SECOM case, using 20 trees with 128 variable to split gives the most optimum NRMSE of 0.1848 with the runtime of near 75 minutes in a trial-error effort of 635 minutes in total as revealed in *Table 6*. The second disadvantage, as explained by Van Buuren, is exacerbated by a high number of missing values, like in the SECOM dataset for example. Missing data causes a large amount of uncertainty and often influences various algorithms to understand imputed data as if they were never missing. This places a higher significance on p-values and strengthens relations between variables (Van Buuren, 2018).

ntree	mtry	Time (min)	error
10	2	1,95	0.5052
40	4	14,71	0.4289
20	4	5,63	0.4422
20	8	9,23	0.4180
20	16	12,33	0.3671
20	32	38,16	0.2955
20	64	68,40	0.2261
20	128	74,94	0.1848
20	256	250,20	0.1610
50	2	8,75	0.4449
100	1	13,31	0.4514
100	2	30,26	0.4342
100	21	38,00	0.3077
500	2	82,20	0.4337

Table 6 Overview of assumptions made by ad-hoc methods

4.3.3.2 k-Nearest Neighbor Imputation

The second method used is k-Nearest Neighbor Imputation (kNN) which is a non-parametric and useful method to deal with all kinds of missing data (Beretta and Santaniello, 2016). The main assumption of the method is the missing values could be approximately estimated by the non-missing values that are close to them in value. kNN represents an improvement of mean imputation that reflects the observed data structure with underlying correlation structure of the data (Mandel J, 2015). It is also relatively quick in performance time; with the SECOM dataset, kNN imputation takes approximately 7 mins with k = 21. It matches a datapoint with its closest k neighbors in a multi-dimensional space. Variables with skewed distributions need to be temporarily scaled to deal with range differences between variables as they are computing based on Euclidean distance. Imputed features has been visually inspected by marginplots. In *Figure 7*, 51 missing values are imputed for feature 091, revealing the interquartile range for imputed values and actual observations differ, but the overall structure appears to be preserved.

In their book on data mining with use of SPSS Modeler, the Wendler and Gröttrup discuss the number of dimensions influencing kNN's performance, the so-called "curse of dimensionality”. With higher dimensions, the relations and tendencies between multiple variables are counterbalanced (Wendler and Gröttrup, 2016).

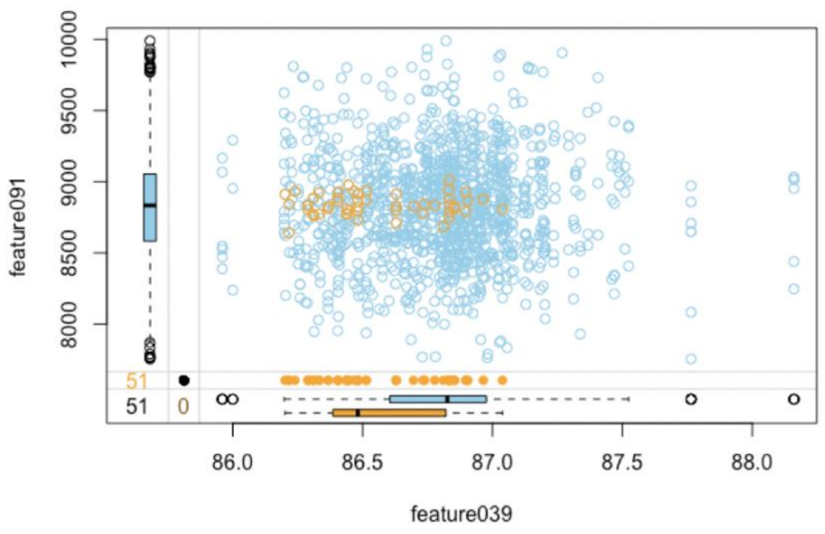


Figure 7 The marginplot of features 039 and 091 with imputed values of features 091 by kNN.

4.3.3.3 Multivariate Imputation by Chained Equations

The third method used is multivariate imputation by chained equations (MICE). This method creates multiple complete datasets by filling missing values multiple times based on Fully Conditional Specification, where each incomplete variable is imputed by a separate model and the analyses of the data consider the uncertainty in the imputations and yield accurate standard errors (Azur et al., 2011). The MICE methodology attempts to be as realistic as possible, considering the process which lead to the missing data in the first place. This ideally retains not only the relations, but also the noise between the data (Van Buuren, 2018).

“Blind imputation” is a term coined by Van Buuren alluding to using all variables as predictors in imputation model as it is not feasible due to multicollinearity and computational problems. In the SECOM case, this means trying to calculate analyses with 440 explanatory variables and to repeat that for every incomplete variable, where the operations took 48 hours (Van Buuren, 2018).

An automatic predictor matrix is generated which uses a four-step variable selection procedure to prevent blind imputation. This matrix contains an encoded list of features to be used as predictors on imputation of each missing features. As further explained in detail by Buuren, this procedure includes all variables that appear in the complete-data model and response model and include variables that explain a considerable amount of variance of the target and removes variables selected in previous steps from those that have too many missing values within the subgroup of incomplete cases (Buuren et al., 1999, p. 687). The correlation parameters of the prediction matrix is optimized to get 21 predictors on average for the imputation of each feature. The Spearman method has been chosen as it does not assume linearity and the majority of the SECOM dataset is not normally distributed. The underlying motivation for this optimization is the increase in explained variance becoming negligible after 15 variables have been included, and sub-selection of no more than 25 variables is important (Buuren and Groothuis-Oudshoorn, 2011).

Instead of Predictive Mean Matching, which is the default method of MICE for continuous predictors, the CART algorithm (classification tree as target feature is discrete) has been implemented. Van Buuren explains this further as "CART methods have properties that make them attractive for imputation: they are robust against outliers, can deal with multicollinearity and skewed distributions, and are flexible enough to fit interactions and nonlinear relations." (Van Buuren, 2018, p. 85).

Using CART with the auto-generated predictor matrix for the SECOM dataset reduced the total computation time to approximately 15 minutes. For the SECOM dataset, multiple stripplots of features with imputed values represented by different color across original and 5 different datasets is shown in *Figure 8*.

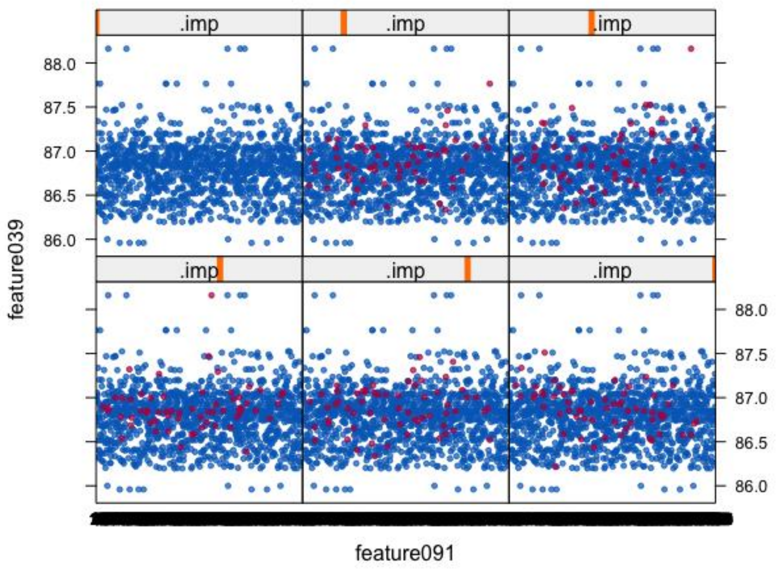


Figure 8 The stripplot of features 091 and 039 with imputed values of feature 039 by MICE

A random forest model with the same hyper parameters has been fit on two datasets imputed by kNN and MICE algorithms. The model performance results for both are shown in *Figure 9*, with MICE outperforming kNN in all metrics. While AUC and sensitivity values are slightly different, MICE's specificity performs significantly higher than kNN. As an additional advantage, MICE detects collinear and constant features if any, and does not impute them. As a result, the MICE algorithm has been chosen to proceed in the SECOM case.

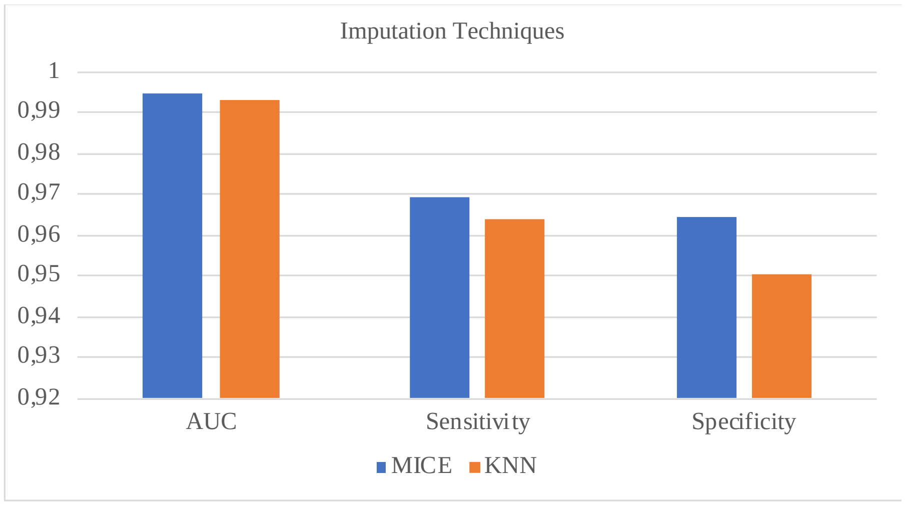


Figure 9 The stripplot of features 091 and 039 with imputed values of feature 039 by CART of MICE.

4.3.4 Feature Selection: SECOM

Feature selection is a vital step prior to building the predictive models for three main reasons: First, it assists in removing potentially redundant variables to increase the accuracy of the predictive models Kohavi and John state that when the model is not at its optimal features, its performance is also not optimal (Kohavi and John, 1997). Moreover, some models' performance such as Support Vector Machines and Neural Networks are affected by irrelevant variables (Kuhn and Johnson, 2019, p. 228). Additionally, to reduce the number of features can alleviate the requirement of computation power. Third, to defy the curse of dimensionality and to improve the prediction performance (Guyon and Elisseeff, 2003).

4.3.4.1 Feature Selection Methods

There are in general three feature selection methods: Filter, Wrapper and Embedded methods. Boruta and Recursive Feature Elimination (RFE) from Wrapper methods and Lasso Regression from Embedded are used in this study.

The wrapper method uses chosen learning algorithms to identify variables based on its predictive power (Guyon and Elisseeff, 2003). Statistical hypotheses are used to identify the significance of the variables before or after adding it into the predictors (Kuhn and Johnson, 2013, pp. 491–498). Boruta is a backward selection wrapper method with an underlying random forest classification algorithm to identify all relevant variables (Kursa and Rudnicki, 2010). Recursive Feature Elimination (RFE) is a backward selection wrapper method introduced by Guyon which starts by using all the variables to build and compute every variable's statistically significance with scores and remove the least important variable in each iteration (Guyon et al., 2002; Kuhn and Johnson, 2019, p. 248).

The embedded method conducts feature selection while training the model. In this study, Lasso is selected during trial phase, but performs poorly with the SECOM. Lasso, the Least Absolute Shrinkage and Selection Operator, which was first introduced by Robert Tibshirani in 1996 (Tibshirani, 1996), performs regularization (shrinking) and feature selection with the primary goal of minimizing the prediction error.

During the trial stage, RFE is implemented but it does not extract as many features as Boruta. Lasso does not perform well without implementing balancing techniques (Fonti, 2017). Furthermore, SECOM dataset's EPV (ratio between number of features and the number of observations) is too low (van Smeden et al., 2019). Therefore, Boruta is eventually chosen and proceed to next step – Model Building.

4.3.4.2 Implementing Feature Selection

In the dimensional reduction steps, 140 features are removed and at missing value imputation steps, another 26 features are removed. The data now has 424 features. *Figure 10* shows that with Boruta methods, 13 important features and 2 tentative features are identified with various trials as *Table 7* displays the summary.

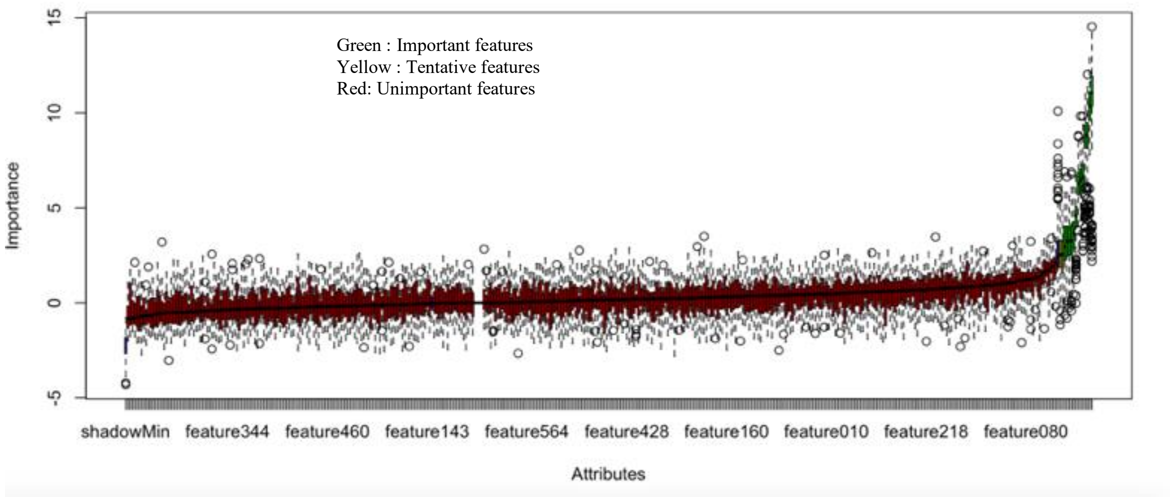


Figure 10 Boruta selected features distribution based on the importance

maxRuns	iterations	Important	unimportant	tentative	Duration (mins)
100 (default)	99	11	409	4	7.305315
101	100	11	409	4	6.600742
76	75	11	409	4	6.486325
150	149	12	409	3	6.993041
250	249	13	409	2	9.19243
500	499	13	409	2	10.80016

Table 7 Overview of results of different trials made by Boruta method

One defect of the Boruta method is its lack of ability to handle multicollinearity. Fortunately, this was already dealt with by MICE during the previous section. To illustrate that feature selection does help to improve the model's performance in the SECOM case, *Figure 11* shows the comparison of the performance of the model built using all features and the model using Boruta-selected 13 important features. It shows that AUC and sensitivity improve dramatically. Specifically, sensitivity improves from 0.76 to 0.96 with the implementation of Boruta feature selection prior to model building.

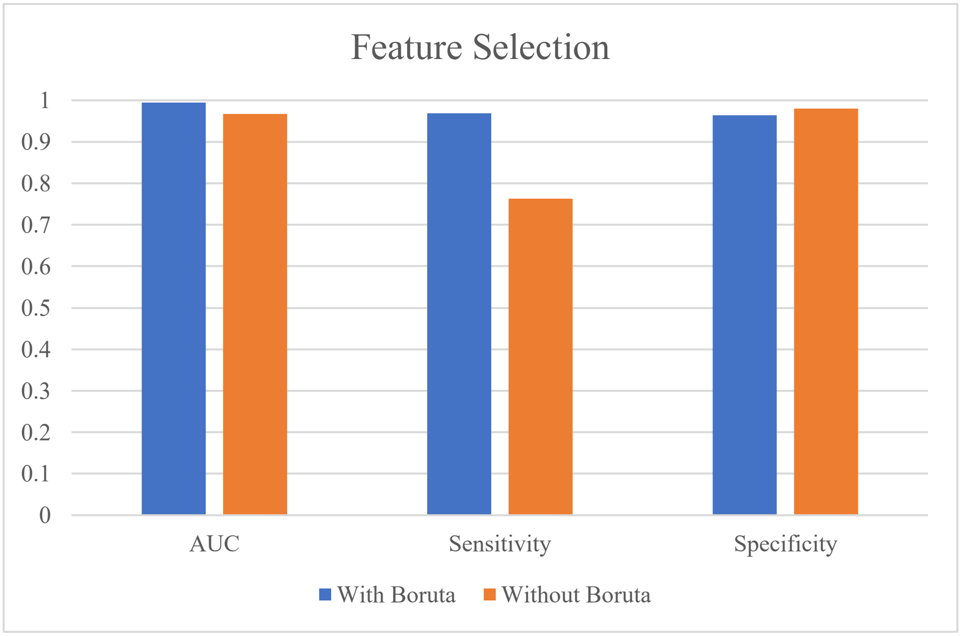


Figure 11 Effects of Boruta on Model's Performance

4.3.5 Balancing Methods

According to the SECOM dataset, it is clear that the binary target variable is highly imbalanced with a majority class as "passing" wafers and a minority class as "failed" wafers. An imbalanced dataset occurs when the minority class of the target variable has a low proportion compared to the majority class (Aggarwal, 2015). Consequently, the effectiveness and accuracy of the model can be highly impacted because of the severely skewed class distribution. This issue can be mitigated by various methods, namely by balancing methods, adjusting the model's alternate cutoffs, and tuning the model (Kuhn and Johnson, 2013). This chapter explains the impact of imbalanced data on a model's performance, and how the issue is mitigated by applying balancing methods.

To check the remedies of having a severe class imbalance, three well-known predictive models (Random Forest, Logistic Regression, and K Nearest Neighbor) can be applied on an imbalanced train set and the model's results are analyzed. According to the model's results, three main performance measures are emphasized, namely AUC, sensitivity, and precision. Three models are built by using a 10-fold cross validation along with their best hyperparameters.

Model	**Accuracy**	**AUC**	**Sensitivity**	**Specificity**	**Precision**
Logistic Regression	0.936	0.521	0.049	0.994	0.333
Random Forest	0.945	0.557	0.115	0.999	0.875
K Nearest Neighbor	0.939	0.505	0.011	0.999	0.500

Table 8 Results from three predictive models using the evaluation set

According to *Table 8*, the three models yield similar poor performances, as they are highly impacted by the imbalanced class distribution. Especially for sensitivity, the results are extremely low due to the fact that the majority class of target feature overwhelms the minority class when building the models. Moreover, sensitivity is highly important in SECOM case, because sensitivity can be interpreted as the model's ability to correctly detect faulty wafers which actually are. To reduce the impact of the issues introduced by imbalanced data, the next chapter will discuss the available balancing methods which can be used to handle class imbalances.

4.3.5.1 Introduction of Balancing Methods

Instead of building the models with an imbalanced training set, balancing methods can be applied to control the proportion of class in the training set. However, only the training set should be balanced, not test set, as the test set should be consistent with its actual proportion to enable the model to yield a realistic result with any future dataset (Kuhn and Johnson, 2013). There are several advanced methods to balance the data, however, can be combined into two fundamental approaches: over-sampling and under-sampling (Chawla, 2009). Over-sampling methods balance the dataset by randomly duplicating observations in the minority class, while under-sampling randomly remove observations from the majority class. As both methods contain disadvantages, such as overfitting, changes in class distribution, and loss of valuable information, the combination of these methods lead to more effective and sophisticated approaches, including SMOTE and ROSE. SMOTE, the Synthetic Minority Over-sampling Technique, embraces both over-sampling and under-sampling methods, and applies k Nearest Neighbor (kNN) to generate new samples (Chawla et al., 2002). According to the SMOTE algorithm, datapoints in minority class are randomly chosen and their kNN are to be determined. Thus, the new synthetic datapoints are created on the line segment between the chosen minority datapoint and its nearest neighbor. Additionally, SMOTE undersamples the datapoints from the majority class (Fernandez et al., 2018). The combination of these datapoints are added into the training set and the proportion can be defined by setting three parameters in SMOTE: the number of over-sampling points, under-sampling points, and number of kNN. There are several advantages introduced by applying SMOTE, such as a reduction in overfitting and more relatable minority class samples.

The other well-known balancing method to be discussed, ROSE, known as Random Over-Sampling Examples, can be used in competition with SMOTE. ROSE uses bootstrapping to create samples from the feature space neighborhood around the minority class by applying an under-sampling approach to remove modules of majority class, while under-sampling of repeat modules in the minority class by a conditional density estimate of the two classes. The greatest benefit of applying ROSE is that it maintains the distribution of the data, as it creates synthetic datapoints from an estimate of the conditional densities of the two classes (Lunardon et al., 2014). The next chapter will show the results of applying the aforementioned balancing methods to the SECOM training set while providing convincing justification on which method should be applied.

4.3.5.2 Implementing Balancing Methods

From the previous section, two balancing methods, SMOTE and ROSE, are used and their results are compared against the original result from imbalanced training set. Over-sampling and under-sampling method are to be excluded in this chapter, as each method itself has several drawbacks, including the severe impact from data loss. This is extremely critical when considering the low number of SECOM observations. Another disadvantageous impact from the over-sampling method is having a heavily overfitted model (Barandela et al., 2004).

Method	**Accuracy**	**AUC**	**Sensitivity**	**Specificity**	**Precision**
Original Set	0.945	0.557	0.115	0.999	0.875
SMOTE	0.967	0.967	0.969	0.965	0.962
ROSE	0.781	0.780	0.817	0.744	0.768

Table 9 Results from two balancing methods using the evaluation set

To compare these balancing methods, a Random Forest algorithm is applied by using the bootstrapping approach, along with its best hyperparameters. A model built with the original, imbalanced dataset has a sensitivity less than 0.15, as seen in *Table 9*. It improves significantly when balancing methods, such as ROSE and SMOTE, are applied, with the sensitivity ranging now from 0.78-0.96. According to the current results, SMOTE performs better than ROSE in term of AUC, sensitivity, specificity, and other performance metrics. Given these results, SMOTE is determined to be used for further processes as it is the best balancing method. These results can be changed in other different cases as all processes, such as imputation, feature selection, and other related preprocessing can heavily affect the balancing results.

In conclusion, an imbalanced training dataset can affect a model's performances, and it is suggested to apply balancing methods to mitigate the impact of this imbalance issue. By considering the current pre-process, SMOTE is determined to be the best balancing approach as it yields the best result in term of AUC, sensitivity, and precision, which are key factors in analyzing SECOM dataset.

4.4 Modeling

4.4.1 Resampling Methods

The stratified SECOM test set is considerably smaller with 327 observations. This indicates the necessity of resampling methods, as such test set's power to make judgements is questionable. A test set should possibly be avoided, should the sample size be small, as the train set needs to be large enough to produce a sufficient model (Kuhn and Johnson, 2013). To avoid issues with smaller datasets, cross-validation and other resampling methods may be utilized. These methods may be able to accurate estimate of the model's performance and tend to be more reliable than approaches without resampling (Hawkins et al., 2003; Kuhn and Johnson, 2013, p. 67).

Resampling approaches to evaluate model performance work similarly: a subset of samples are randomly split and used to fit a model, and the remaining samples are used to evaluate the model's effectiveness. This is repeated several times, and the results are aggregated and summarized (Kuhn and Johnson, 2013). A visual representation of this concept for an example of 10-fold cross-validation is shown in *Figure 12*. In the SECOM case, k-fold cross validation, repeated k-fold cross validation and the bootstrap resampling methods have been used with Caret package.

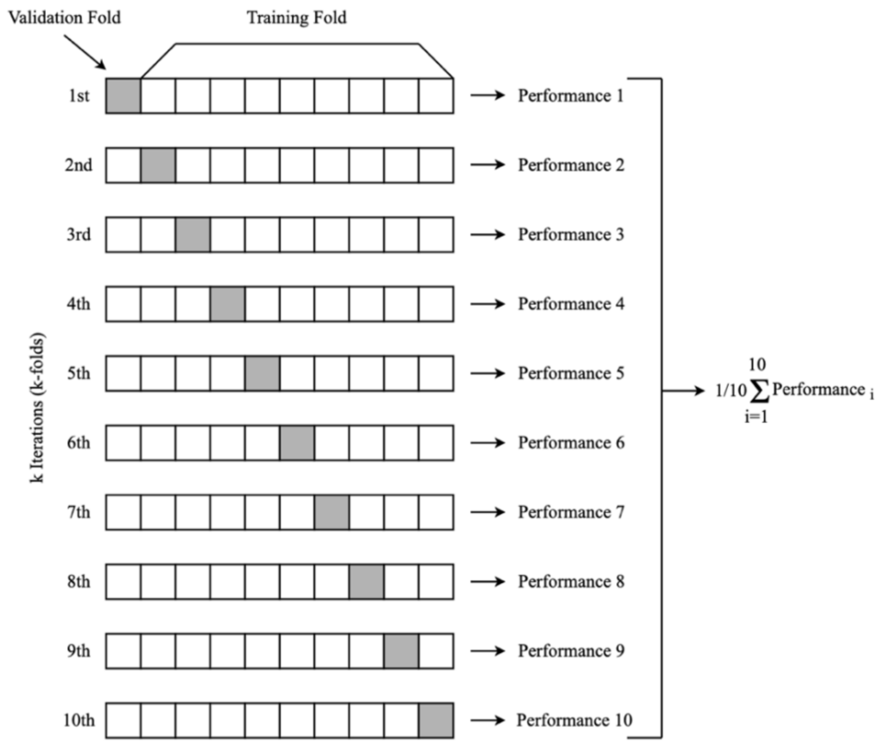


Figure 12 Performance estimate in 10-fold cross-validation
Reproduced with modification from Sebastian Raschka's paper licensed under CC BY 4.0

4.4.1.1 k-fold Cross Validation

Sometimes called as "rotation estimation", k-fold cross-validation is one of mostly used resampling method to validate models in modern machine learning workflows, where data is randomly split into k mutually exclusive subsets (Kohavi, 1995). Using 10-fold cross-validation, 10 different train and validation sets was created from the balanced training data to tune the hyperparameters and evaluate the models in the SECOM case.

Like many viable statistical methods, incorporation of the uncertainty is important for a proper implementation as well as the selection of the number of k. The choice for k was 10, due to increasing k significantly reduces the difference in size between subsets and bias of the implementation becomes smaller. An unbiased k-fold cross-validation may be estimating the true theoretical performance but it might be costly in terms of uncertainty due to cross-validation may have high variance compared to other resampling methods (Kuhn and Johnson, 2013). This bias-variance trade-off is also further discussed by James et al (2003): ". . . there is a bias-variance trade-off associated with the choice of k in k-fold cross-validation. Typically, given these considerations, one performs k-fold cross-validation using k = 5 or k = 10, as these values have been shown empirically to yield test error rate estimates that suffer neither from excessively high bias nor from very high variance." (James et al., 2013, p. 184).

4.4.1.2 Repeated Cross Validation

Also known as "leave-group-out cross validation" or "Monte Carlo cross-validation", repeated cross validation randomly splits the sample into a train and test set numerous times (e.g. 20, 50 or 1000 iterations) as the number of repetitions depends on the practitioner, with the only difference with k-fold cross-validation is that samples can be represented in multiple (held-out) test subsets (Kuhn and Johnson, 2013; Molinaro et al., 2005). This means data is shuffled before each repetition to ensure splits are different.

Two researches show that repeating k-fold cross validation can produce more accurate estimates while maintaining a small bias. (Kim, 2009; Molinaro et al., 2005). This concludes that the number of repetitions is important for bias-variance trade-off as Kuhn and Johnson further discussed this The uncertainty of performance estimates decreases as the number of subset increases. This is also linked to the random allocations of samples to the prediction set, as more repetitions are required to reduce uncertainty (Kuhn and Johnson, 2013, p. 72).

In the SECOM case, 5 times repeated 10-fold cross validation is used on the balanced training data for creating validation sets to tune the hyperparameters and evaluate the models but results will not be shared for the simplicity of the paper, since the bootstrapping produced better results.

4.4.1.3 Bootstrapping

Bootstrapping is a random sampling procedure where some of the data taken with replacement (Efron and Tibshirani, 1986). Due to the bootstrapped sample is same size as the original data, this means some of the samples will be represented multiple times while some of others will not be represented at all. This situation again leads us to bias-variance trade-off. Bootstrapping gives more pessimistic "aka biased" results as a consequence of its tendency to dramatically reduce the variance. There are also some bootstrapping techniques such as "632 method" invented to deal with the bootstrap bias. (Efron, 1983; Kuhn and Johnson, 2013).

In the SECOM case, 20 times bootstrapped validation sets derived from the balanced training data were used to evaluate the models. The models trained on bootstrapped samples produces the best results. Modified bootstrapping methods are not implemented to reduce the complexity of the workflow. A more appealing schematic to compare all three methods is shown in *Figure 13*.

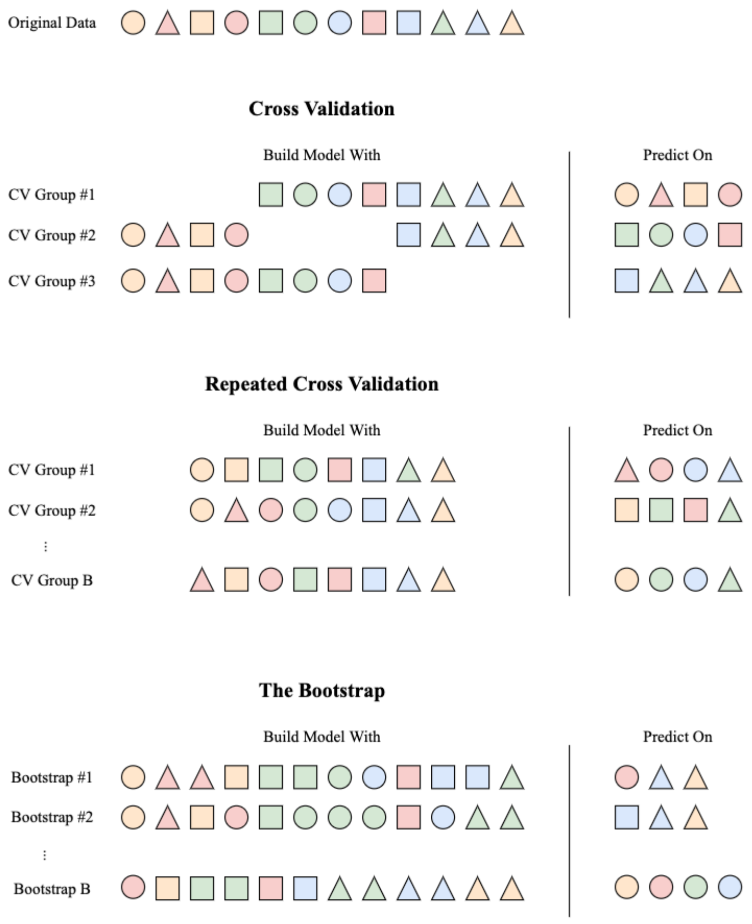


Figure 13 A holistic schematic of three resampling methods.
Reproduced with modification from (Kuhn and Johnson, 2013, pp. 71, 72, 73). All rights are reserved by Springer Science+Business Media LLC New York.

4.4.2 Model Training and Hyperparameter Optimization

Most of the non-linear models have tuning parameters which are vital to optimize model performance to yield the most realistic predictive power without falling into the trap of "overfitting". In the SECOM case, the training process suggested by (Kuhn and Johnson, 2013) has been employed. It started with models that are the least interpretable and most flexible such as Random Forest and Support Vector Machines and then considerably simpler, less opaque models such as Naive Bayes are investigated. At the end, the simplest model that might possibly match the performance of the more complex models are trained such as Logistic Regression.

As discussed by previous chapter, multiple training / test splits are created with the help of resampling methods. These validations sets, that are created from the training data that was split before with stratified random sampling, used to build and tune the models. For this purpose, pre-defined resampling folds are being used during model training to make fair comparisons between models. 21

pre-defined lists of seed values are stored and used to allow parallel processing without errors in tree-based models and to ensure the reproducibility of the research. Kuhn and Johnson's parameter tuning process represented in *Figure 14* has been employed as the main hyperparameter tuning strategy.

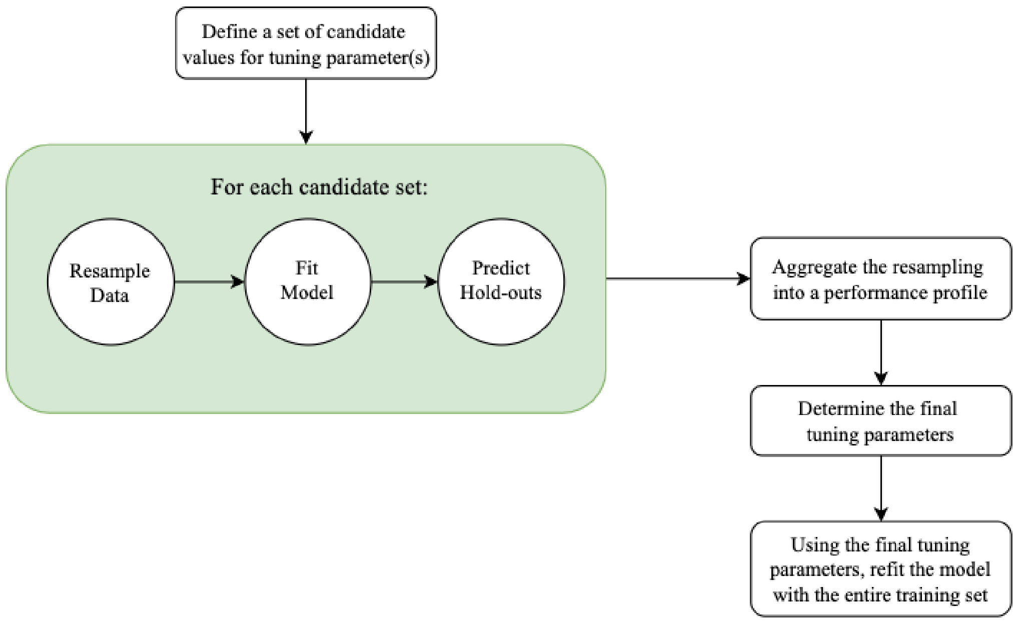

Define a set of candidate values for tuning parameter(s)
For each candidate set:
Resample Data
Fit Model
Predict Hold-outs
Aggregate the resampling into a performance profile
Determine the final tuning parameters
Using the final tuning parameters, refit the model with the entire training set

Figure 14 Hyperparameter Tuning Process
Reproduced with modification from (Kuhn and Johnson, 2013, p. 66).
All rights are reserved by Springer Science+Business Media LLC New York.

In this context, 10 set of hyperparameter candidates for each model has been created by using Caret package's default grid search routine. As a brute force algorithm, grid search was preferred due over random search and genetic algorithm due to random search provides speed but does not guarantee the best results and genetic algorithm is too much expensive in terms of computation (Liashchynskyi and Liashchynskyi, 2019). Each candidate set has been fit on the same 20-fold of bootstrapped samples and the predictions coming from each hold-out are aggregated into a performance profile for each model. Performance profiles for Random Forest, Support Vector Machine, Gradient Boosting Machine and k-Nearest Neighbor models are represented in *Figure 15*. And finally, each model has been refitted with the entire training set using the final tuning parameters that yielded the best results.

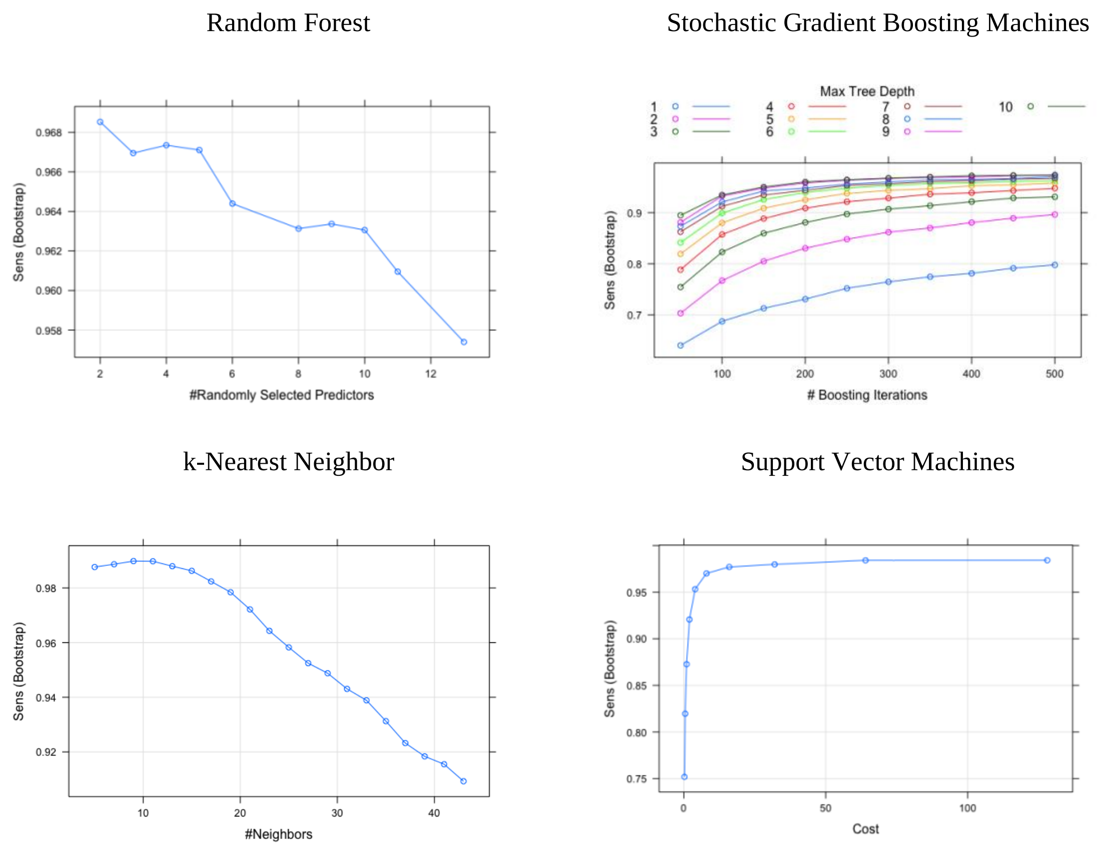


Figure 15 Performance profiles of hyperparameters for Random Forest, SVM, GBM and kNN models

After these evaluations, some of the best performer models is tested on test "yet-to-seen" dataset that preserves the class imbalance. Models trained on the bootstrapped samples found the best balance between bias and variance, as it is further illustrated in *Figure 16* to better understand the importance of it in model evaluation and selection process (Raschka, 2018).

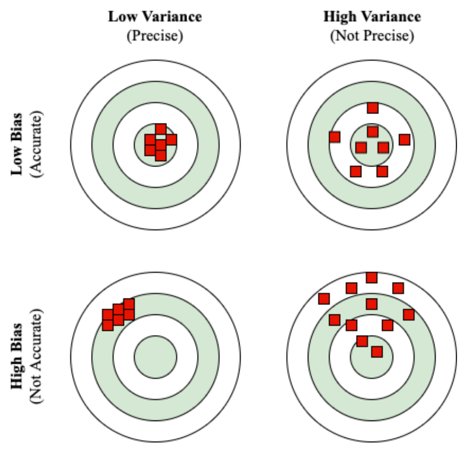


Figure 16 Illustration of bias and variance
Reproduced with modification from Sebastian Raschka's paper licensed under CC BY 4.0.

4.4.3 Learning Algorithms

This chapter focuses on building and evaluating models by using various algorithms. As the target variable in SECOM case is discrete, a classification model should be applied instead of having normal regression models. Even though both regression and classification models produce a continuous result, the results from classification models are in term of probability which can be integrated with a cut-off threshold to get a predicted class. To find the model which yields a best performance for the SECOM dataset, seven algorithms are considered and built on the training set. These algorithms include Random Forest, Gradient Boosting Machine, Support Vector Machine, K Nearest Neighbor, Naïve Bayes, Discriminant Analysis, and Logistic Regression. Each model built by these algorithms uses Bootstrapping, along with its best hyperparameters. Then, the model's performance is evaluated by several metrics, such as AUC, sensitivity, precision.

4.4.3.1 Random Forest

The Random Forest classifier, one of the most powerful supervised learning algorithms, is built on the foundation of decision tress, as it creates several decision tress by using random subsets from the training set (Breiman, 2001). The aim of having several decision tress is to partition the training set into smaller homogeneous groups which possess pure nodes by containing larger proportion of any specific target class (Liaw and Wiener, 2001). Another important aspect of the Random Forest classifier is that it uses a bagging method to combine all decision trees to get a more accurate and stable prediction. Even though this classifier tends to perform well without having normalized, centered, and scaled the training set as an input, it still carries some assumptions to fully leverage its result, such as correlation between created trees and predictive power of input features. For correlation between trees, Random Forest handles this by randomly picking the subset of features which leads to its tree having low correlation between one another, as well as being more diversified. With this approach, the algorithm introduces another benefit in avoiding an overfitted model. There are two hyperparameters for the Random Forest classifier, including the number of trees and number of randomly-selected features (Breiman, 2001). The number of trees refers to the total number built by the Random Forest classifier. The increase in trees results in a higher stability of the models, as well as trees to vote for the model's final result, however, it also requires higher computational time. The number of randomly selected feature refers to number of features which are used to split each tree node.

4.4.3.2 Gradient Boosting Machine

Whereas the Random Forest classifier uses an ensemble of several independent trees, a Gradient Boosting Machine uses an ensemble of weak trees by improving these trees in sequential iterations. The logic behind Gradient Boosting Machine is this sequential process of building trees which lowers the loss function in each iteration. Loss functions can be different in several cases, namely mean squared error or misclassification. The standard steps include fitting a simple model and calculating its loss function, identifying weak trees and adding extra weights, the finally aggregating all predicted features by adding weights to some of them (Friedman, 2001). However, as the algorithm continues to minimize the errors, this can also lead to an overfitted model. This issue can be controlled by having appropriate hyperparameters. The main hyperparameters of Gradient Boosting Machine include the number of trees, the learning rate or shrinkage, and the minimum number of features to split the tree node. Unlike the Random Forest classifier, Gradient Boosting Machine requires higher numbers of tree with the same training set to avoid overfitting issue. The shrinkage rate controls how fast the algorithm proceeds with its gradient descent, while the minimum number of features to split the tree node controls the complexity of the ensemble (Natekin and Knoll, 2013).

4.4.3.3 Support Vector Machine

Despite of being known as black box algorithm, with complex and results which are difficult to interpret, Support Vector Machines bring several advantages, such as being robust, being effective in high dimensional space, and yielding high precision (Vapnik, 1998). Support Vector Machine is an algorithm which builds a set of hyperplanes to be used for classification. A hyperplane is created on the concept that it possesses the largest distance to the nearest datapoints between the target classes. The data transformation which produces the hyperplane is determined by its kernel function. The kernel function consists of several types, such as Linear, Polynomial, Sigmoid, Radial basis function (RBF) (Wendler and Gröttrup, 2016, pp. 808–812). So far, there has been no concrete rule on which type of kernel function works well or performs best on a specific type of dataset. Thus, most of the time all types of kernel function are applied and chosen based on their respective performances. However, the Radial basis function (RBF) can be a good starting point as it is known to perform well with most kinds of dataset. Because of a variety of kernel functions, there are several hyperparameters used differently for each kernel function, however, the common hyperparameters include Gamma, Cost, and Degree (Evgeniou and Pontil, 2001).

4.4.3.4 k-Nearest Neighbors

The k-Nearest Neighbor classifier is one of the most intuitive and simplest machine learning algorithms which has been used in many fields including fault classification (He and Wang, 2007). kNN classifier is a non-linear, unsupervised classification algorithm. It utilized the geographical attribute of the observation to classify the target observation based on the k nearest neighbors of the observations and calculate the frequency of each class from the neighbors (Wendler and Gröttrup, 2016, p. 878). As kNN classification is based on the distance between observations, it's recommended to scale and center the observations prior to implementing the algorithm (Kuhn and Johnson, 2013, p. 352). Additionally, among different distance calculations in this study, Euclidian distance is used. The number of k is the tuning parameter for this algorithm as if too few neighbors might overfit the training data while too many neighbors are chosen might not yield optimal performance (Kuhn and Johnson, 2013, p. 64). *Figure 17* shows the number of k being tuned via cross-validation on bootstrapped samples while training the model and for Secom training set k appears to reach maximum sensitivity around k = 10.

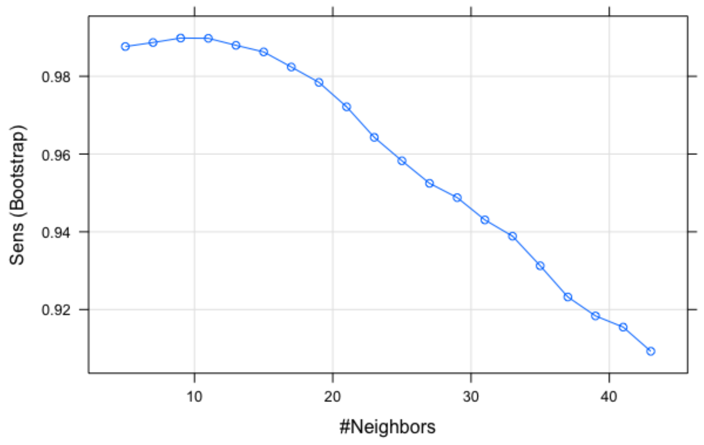


Figure 17 kNN tuning parameter k

4.4.3.5 Neural Network

The Neural Network classifier has recently attracted public's attention with its ability to handle complicated problems, with no restriction of number of inputs and layers (Bourg and Seemann, 2004, pp. 269–315). It is therefore especially applicable in the deep learning field for applications such as image recognition, nature language processing, speech recognition and pattern detection. However, it is famous for being a black box algorithm with results that are difficult to interpret. The performance and ease of interpretation trade-offs would therefore need to be considered when deciding which algorithm to implement. A Neural Network is created with the inspiration from human brain's neuron structure. A non-linear classification technique factoring hidden units which linearly combine some or all of the predictors and then transform using a nonlinear function such as sigmoidal to the value between zero and one (Kuhn and Johnson, 2013, p. 141). A Neural Network classification has a high potential for overfitting since NN is very flexible (Kuhn and Johnson, 2019, p. 7). Therefore, some remedies such as weight decay lessens the parameter estimates and model averaging (Kuhn and Johnson, 2013, p. 335). A Neural Network's tuning parameters are "decay" which is the amount of weight decay and "size" which is the number of hidden units. Moreover, it is recommended to center and scale and apply spatial sign transformation on the observations prior to model building to improve the model's performance.

4.4.3.6 Naïve Bayes

The Naïve Bayes classifier is built based on the Bayesian probability factoring in conditional probability with a strong assumption that every predictor variable is independent from each other to reduce the complexity of the computation. However, this strong assumption is likely to potentially overestimate the probability. As the equation of probability might be affected if any frequency of the variable from the training set is zero for specific class, the multiple of conditional probability would be coerced to zero. To avoid this, Laplace smoothing can be included if necessary and it is one of the tuning parameter of Naïve Bayes named fL. In the SECOM case, fL is set from 0 to 5 and the optimal model is when Laplace smoothing is equal to zero which is identical to the default. Another tuning parameter is the different use of density type and the parameter is "usekernel" in train function. When "usekernel" is set to TRUE which is SECOM case for the trained model indicates that kernel density estimation is used to estimate the probability of class distribution. Also, the bandwidth of kernel density can also be adjusted accordingly (Kuhn and Johnson, 2013, pp. 353–358).

4.4.3.7 Logistic Regression

One of the most commonly used Machine Learning algorithms is the Logistic Regression classifier (LR). It is famous for its intuition, simplicity in interpretation, and quick computational speed. The major difference between LR and linear regression is the categorical nature of target variable. LR integrates the probability concept to predict the target variable by transforming the linear regression equations into probability to predict the target variables and use threshold to classify. In the SECOM case, the target variable is binary 0 and 1. Therefore, Logistic regression would be predicting and classifying the result into either 0 or 1 respectively for each observation. LR has its disadvantages that it has difficulty predicting collinear variables due to LR would not be able to distinguish the impact of variables separately (Wendler and Gröttrup, 2016, pp. 733–736).

In R, to perform Logistic Regression, the glm function is chosen at method parameter in train function to perform generalized linear models (GLM) which includes both Logistic regression and Ordinary Linear Regression (Kuhn and Johnson, 2013, p. 284). For SECOM case it would perform Logistic Regression due to the scale of measurement of target variable – class is categorical, binary.

4.5 Model Evaluation

All the aforementioned algorithms are applied to the training set by using bootstrapping as the sampling method, along with their best hyperparameters. 20 different bootstrapped samples were generated to validate all the models to obtain the best performing hyperparameters. Additionally, their results are analyzed to define the algorithm which yields the best performance measures regarding the SECOM case. In preprocessing, the outliers in the training set were treated using the 3-s rule, missing values were imputed by MICE, and its important features are determined by BORUTA. However, there are some algorithms which require further preprocessing steps, such as scaling, centering, and transformation. For example, should an algorithm assume the data is distributed normally, the data would be transformed accordingly.

Model	**AUC**	**Sensitivity**	**Specificity**	**Precision**	**F1**	**FN**	**FP**
Random Forest	0.996	0.969	0.965	0.970	0.965	245	303
GBM	0.994	0.974	0.950	0.962	0.960	205	437
SVM	0.990	0.984	0.955	0.981	0.968	123	389
kNN	0.977	0.990	0.798	0.793	0.895	79	1748
Neural Network	0.959	0.929	0.883	0.931	0.903	557	1010
Naïve Bayes	0.827	0.674	0.775	0.808	0.701	2551	1948
GLM	0.778	0.587	0.792	0.737	0.646	3234	1797

Table 10 Results from seven algorithms using the training set

As illustrated in *Table 10*, the three models which outperformed the others in terms of overall performance measures are Random Forest, Gradient Boosting Machine, and Support Vector Machine. The result from these models yields a very low number of False Negative and False Positive compared to the total sampling size which is 16,473. The Random Forest produces the highest AUC and specificity with the lowest number of false positives, while Support Vector Machine gives the highest sensitivity and precision, a result of producing the lowest number of false negatives between the three algorithms. Apart from the two previously mentioned models, Gradient Boosting Machine performs well across all metrics, but doesn’t have the highest value in any one measurement. Thus, with the given results, these three models are to be further analyzed through the application on the test set in order to define the best model.

4.5.1 Model Selection (Test-set)

According to previous chapter, three models, using Random Forest, Gradient Boosting Machine, and Support Vector Machine, are chosen as the best performing algorithms when applied to the training set. To finally determine the best model, these models are to be fit with test set to see how well the models works across different samples. Aside from evaluating the model's performance by using normal accuracy-based metrics, non-accuracy-based metrics are also to be used to reflect the real-world situation. As per the SECOM case, accuracy is not the only primary goal of a successful model; the main goal of the predictive model is to detect as many true faulty products as possible, while having an acceptable amount of incorrect predictions (false positives). Because incorrect predictions can lead to additional costs, a cost formula can be applied to calculate the theoretical impact of false negatives and false positives, which will assist in defining and evaluating the models.

As the test set is highly imbalanced, the model is severely impacted by this skewed class distribution. One way to mitigate this issue is by applying an alternate cut-off approach. Even though the alternative cut-off doesn't increase the overall predictive performance, it can be used to make meaningful trade-offs between types of errors (Kuhn and Johnson, 2013).

Model	**AUC**	**Sensitivity**	**Specificity**	**Precision**	**F1**	**FN**	**FP**	**Total Cost**
Random Forest	0.721	0.579	0.864	0.216	0.314	8	40	160
GBM	0.725	0.789	0.660	0.130	0.224	4	100	160
SVM	0.712	0.632	0.793	0.164	0.261	7	61	166

Table 11 Results from three algorithms using the test set

For the SECOM case, a cost ratio of fifteen to one is applied to false negatives and false positives respectively, as false negatives are much more costly than false positive. False prediction of a silicon wafer as not failed when it actually is would not only result in customers receiving defective products, and worse if customers apply to their production that would incur major losses in cost and time, whereas a false positive, while also an incorrect prediction, would cause more minor impacts such as delays in inspections and additional testing costs. As per *Table 11*, the Random Forest and Gradient Boosting Machine models perform equally in terms of total cost, while Random Forest yields a better result in specificity, precision, and F1, and Gradient Boosting Machine gives a better value in the remaining metrics. The alternate cut-offs of both models can be adjusted to give a slightly different result. However, these won't change the fact that both models are quite similar in term of performance. As Gradient Boosting Machine requires a much higher computational time with complex hyperparameters, Random Forest is chosen as the best model in term of performance, cost, and computational time.

4.5.2 Model Deployment

As the last step, the chosen Random Forest model's consistency was tested on 500 different datasets created from the raw data using Bootstrapping. A procedure was designed to calculate the cost of false predictions for each dataset with a 95% confidence interval. The cost formula was:

*15 * False Negatives + False Positives*

In predictive maintenance like in the SECOM case study, type II errors (false negatives) are considered much more severe than type I errors (false positives). Results of the Random Forest model's prediction on 500 different datasets shown in a histogram overlaid with a kernel density curve in *Figure 18*. Each sample contains 1567 observations which is the same as the raw data, resulting in the higher total cost than in the *Table 11*. A near leptokurtic curve indicates that the model produced consistently stable results and it would be reliable in a production environment.

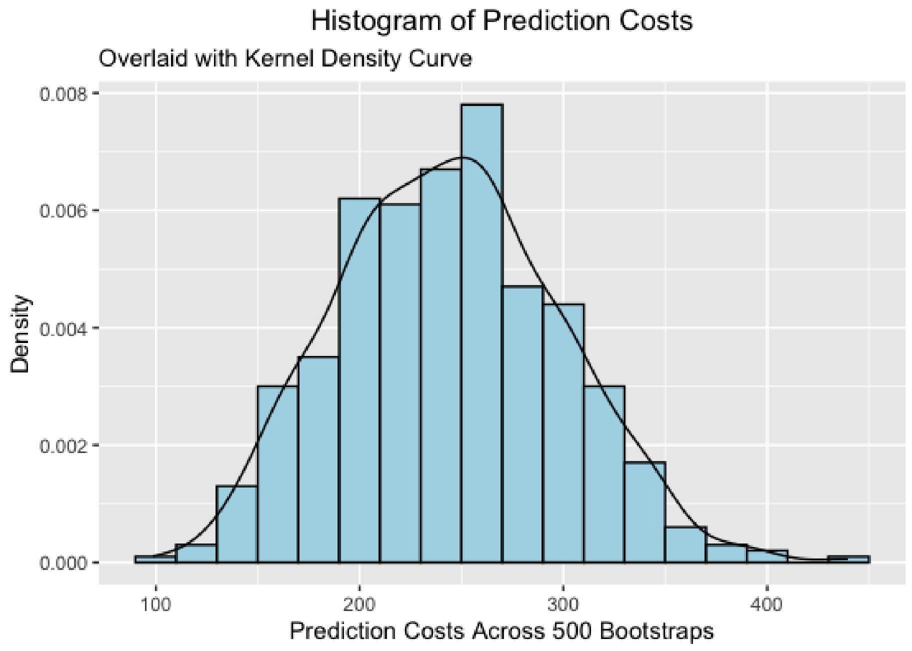


Figure 18 Prediction Costs across 500 predictions

5. Conclusion

With its high complexity and access to vast amounts of data, the semiconductor industry proves to be an excellent use case to apply data mining techniques with the goal of improving its processes and accuracy control, lowering potential risks and costs, and eventually raising profit margins. The CRISP-DM methodology supplied a useful structure for this data mining endeavor and ensured that not only the data mining goals were achieved, but also that the entire project was useful to the business case and would bring value to the company or organization.

The CRISP-DM provided an iterative structure to cross-check decisions and choose the best possible option. Through this data mining endeavor, the most reliable and stable model was found using the following procedure:

- MICE imputation
- Boruta feature selection
- SMOTE balancing
- Random Forest learning algorithm

These were the four major decisions during the procedure, however there are countless other minor decisions which could impact the results of the model. This is a crucial element to data mining, that the results can differ significantly from one attempt to another like a chained effect, particularly when presented with new or adapted data. For this reason, the stable model built during this research, although not perfect, proves to be a reliable resource for future endeavors.

References

Aggarwal, C.C., 2015. Data mining: the textbook. Springer, Cham Heidelberg New York Dordrecht London.

Azur, M.J., Stuart, E.A., Frangakis, C., Leaf, P.J., 2011. Multiple imputation by chained equations: what is it and how does it work? Int. J. Methods Psychiatr. Res. 20, 40–49. https://doi.org/10.1002/mpr.329

Barandela, R., Valdovinos, R.M., Sánchez, J.S., Ferri, F.J., 2004. The Imbalanced Training Sample Problem: Under or over Sampling?, in: Fred, A., Caelli, T.M., Duin, R.P.W., Campilho, A.C., de Ridder, D. (Eds.), Structural, Syntactic, and Statistical Pattern Recognition, Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, pp. 806–814. https://doi.org/10.1007/978-3-540-27868-9_88

Beretta, L., Santaniello, A., 2016. Nearest neighbor imputation algorithms: a critical evaluation. BMC Med. Inform. Decis. Mak. 16, 74. https://doi.org/10.1186/s12911-016-0318-z

Bourg, D.M., Seemann, G., 2004. AI for game developers, 1st ed. ed. O'Reilly, Sebastopol, CA.

Breiman, L., 2001. Random Forests. Mach. Learn. 45, 5–32. https://doi.org/10.1023/A:1010933404324

Buuren, S.V., Boshuizen, H.C., Knook, D.L., 1999. Multiple imputation of missing blood pressure covariates in survival analysis 14.

Buuren, S. van, Groothuis-Oudshoorn, K., 2011. mice: Multivariate Imputation by Chained Equations in R. J. Stat. Softw. 45, 1–67. https://doi.org/10.18637/jss.v045.i03

Chawla, N.V., 2009. Data Mining for Imbalanced Datasets: An Overview, in: Maimon, O., Rokach, L. (Eds.), Data Mining and Knowledge Discovery Handbook. Springer US, Boston, MA, pp. 875–886. https://doi.org/10.1007/978-0-387-09823-4_45

Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P., 2002. SMOTE: Synthetic Minority Over-sampling Technique. J. Artif. Intell. Res. 16, 321–357. https://doi.org/10.1613/jair.953

Efron, B., 1983. Estimating the Error Rate of a Prediction Rule: Improvement on Cross-Validation. J. Am. Stat. Assoc. 78, 316–331. https://doi.org/10.2307/2288636

Efron, B., Tibshirani, R., 1986. Bootstrap Methods for Standard Errors, Confidence Intervals, and Other Measures of Statistical Accuracy. Stat. Sci. 1, 54–75.

Evgeniou, T., Pontil, M., 2001. Support Vector Machines: Theory and Applications, in: Paliouras, G., Karkaletsis, V., Spyropoulos, C.D. (Eds.), Machine Learning and Its Applications, Lecture Notes in Computer Science. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 249–257. https://doi.org/10.1007/3-540-44673-7_12

Fernandez, A., Garcia, S., Herrera, F., Chawla, N.V., 2018. SMOTE for Learning from Imbalanced Data: Progress and Challenges, Marking the 15-year Anniversary. J. Artif. Intell. Res. 61, 863–905. https://doi.org/10.1613/jair.1.11192

Fonti, V., 2017. Feature Selection using LASSO.pdf [WWW Document]. URL https://beta.vu.nl/nl/Images/werkstuk-fonti_tcm235-836234.pdf (accessed 7.10.20).

Friedman, J.H., 2001. Greedy Function Approximation: A Gradient Boosting Machine. Ann. Stat. 29, 1189–1232.

Guyon, I., Elisseeff, A., 2003. An Introduction to Variable and Feature Selection 26. https://doi.org/10.1162/153244303322753616

Guyon, I., Weston, J., Barnhill, S., 2002. Gene Selection for Cancer Classification using Support Vector Machines 34. https://doi.org/10.1023/A:1012487302797

Hawkins, D.M., Basak, S.C., Mills, D., 2003. Assessing Model Fit by Cross-Validation. J. Chem. Inf. Comput. Sci. 43, 579–586. https://doi.org/10.1021/ci025626i

He, Q.P., Wang, J., 2007. Fault Detection Using the k-Nearest Neighbor Rule for Semiconductor Manufacturing Processes. IEEE Trans. Semicond. Manuf. 20, 345–354. https://doi.org/10.1109/TSM.2007.907607

IBM, 2019. IBM SPSS Modeler CRISP-DM Guide to Modeler 18.2 [WWW Document]. URL ftp://ftp.software.ibm.com/software/analytics/spss/documentation/modeler/18.2.1/en/ModelerCRISPDM.pdf (accessed 7.7.20).

James, G., Witten, D., Hastie, T., Tibshirani, R., 2013. An Introduction to Statistical Learning: with Applications in R. Springer Science & Business Media.

Kerdprasop, K., Kerdprasop, N., 2011. Feature Selection and Boosting Techniques to Improve Fault Detection Accuracy in the Semiconductor Manufacturing Process. Int. MultiConference Eng. Comput. Sci. 1.

Kim, J.-H., 2009. Estimating classification error rate: Repeated cross-validation, repeated hold-out and bootstrap. Comput. Stat. Data Anal. 53, 3735–3745. https://doi.org/10.1016/j.csda.2009.04.009

Kohavi, R., 1995. A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. Morgan Kaufmann, pp. 1137–1143.

Kohavi, R., John, G.H., 1997. Wrappers for feature subset selection. Artif. Intell. 97, 273–324. https://doi.org/10.1016/S0004-3702(97)00043-X

Kuhn, M., Johnson, K., 2019. Feature Engineering and Selection: A Practical Approach for Predictive Models. CRC Press.

Kuhn, M., Johnson, K., 2013. Applied Predictive Modeling. Springer New York, New York, NY. https://doi.org/10.1007/978-1-4614-6849-3

Kursa, M.B., Rudnicki, W.R., 2010. Feature Selection with the Boruta Package. J. Stat. Softw. 36. https://doi.org/10.18637/jss.v036.i11

Liashchynskyi, Petro, Liashchynskyi, Pavlo, 2019. Grid Search, Random Search, Genetic Algorithm: A Big Comparison for NAS. ArXiv191206059 Cs Stat.

Liaw, A., Wiener, M., 2001. Classification and Regression by RandomForest. Forest 23.

Lunardon, N., Menardi, G., Torelli, N., 2014. ROSE: a Package for Binary Imbalanced Learning. R J. 6, 79. https://doi.org/10.32614/RJ-2014-008

Mandel J, S.P., 2015. A Comparison of Six Methods for Missing Data Imputation. J. Biom. Biostat. 06. https://doi.org/10.4172/2155-6180.1000224

Molinaro, A.M., Simon, R., Pfeiffer, R.M., 2005. Prediction error estimation: a comparison of resampling methods. Bioinformatics 21, 3301–3307. https://doi.org/10.1093/bioinformatics/bti499

Munirathinam, S., Ramadoss, B., 2016. Predictive Models for Equipment Fault Detection in the Semiconductor Manufacturing Process. IACSIT Int. J. Eng. Technol. 8.

Natekin, A., Knoll, A., 2013. Gradient boosting machines, a tutorial. Front. Neurorobotics 7. https://doi.org/10.3389/fnbot.2013.00021

Raschka, S., 2018. Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning. ArXiv181112808 Cs Stat.

Stekhoven, D.J., Bühlmann, P., 2011. MissForest—non-parametric missing value imputation for mixed-type data. Bioinformatics 28, 112–118. https://doi.org/10.1093/bioinformatics/btr597

Tibshirani, R., 1996. Regression Shrinkage and Selection Via the Lasso. J. R. Stat. Soc. Ser. B Methodol. 58, 267–288. https://doi.org/10.1111/j.2517-6161.1996.tb02080.x

Van Buuren, S., 2018. Flexible imputation of missing data, Second edition. ed, Chapman and Hall/CRC interdisciplinary statistics series. CRC Press, Taylor & Francis Group, Boca Raton.

van Smeden, M., Moons, K.G., de Groot, J.A., Collins, G.S., Altman, D.G., Eijkemans, M.J., Reitsma, J.B., 2019. Sample size for binary logistic prediction models: Beyond events per variable criteria. Stat. Methods Med. Res. 28, 2455–2474. https://doi.org/10.1177/0962280218784726

Vapnik, V., 1998. The Support Vector Method of Function Estimation, in: Suykens, J.A.K., Vandewalle, J. (Eds.), Nonlinear Modeling: Advanced Black-Box Techniques. Springer US, Boston, MA, pp. 55–85. https://doi.org/10.1007/978-1-4615-5703-6_3

Wendler, T., Gröttrup, S., 2016. Data mining with SPSS Modeler: Theory, exercises and solutions. Springer, Switzerland.