Lappeenranta-Lahti University of Technology LUT

School of Business and Management

Degree Programme in Strategic Finance and Business Analytics

Master's Thesis

Machine Learning in Predictive Maintenance:

Classification Approach for Remaining Useful Life Prediction

Tuomas Peltola

2020

1st examiner: Mikael Collan

2nd examiner: Jyrki Savolainen

ABSTRACT

Author:	Tuomas Peltola
Title:	Machine Learning in Predictive Maintenance: Classification Approach for Remaining Useful Life Prediction
Faculty:	LUT School of Business and Management
Major:	Strategic Finance and Business Analytics
Year:	2020
Master's thesis	90 pages, 7 tables, 27 figures, 3 appendices
Examiners	Professor Mikael Collan, Post-Doctoral Researcher Jyrki Savolainen
Keywords	predictive maintenance, remaining useful life, machine learning, classification

This thesis focuses on predicting remaining useful life (RUL) with classification approach. The methodology is demonstrated with NASA's turbofan engine degradation dataset. Three classification systems with different multi-class divisions are constructed from RUL's, systems consisting of 10, 6 and 4 classes, respectively. Random forest (RF) and neural network (NN) algorithms are used to train the classification models. The performance comparison between systems and algorithms along with the optimal hyperparameters are the principal questions to be answered in the scope of this study. The performance is measured with mainly Matthews correlation coefficient (MCC) and complemented with interpretations of confusion matrices. The results show that the parameters which were focused on, do not have huge impact on the model performance. The classification system with fewest classes performs much better compared to the other systems, which are closer to each other in terms of MCC. RF slightly outperforms NN in case of every classification system, although the NN parameters would need better optimization. Even though close to perfect classification was not achieved, the results of this study show that the proposed approach has potential, yet the class divisions especially need further consideration.

TIIVISTELMÄ

Tekijä:	Tuomas Peltola
Otsikko:	Koneoppiminen ennakoivassa huollossa: luokittelumallit jäljellä olevan käyttöiän ennustamisessa
Tiedekunta:	LUT School of Business and Management
Maisteriohjelma:	Strategic Finance and Business Analytics
Vuosi:	2020
Pro Gradu:	90 sivua, 7 taulukkoa, 27 kuvaajaa, 3 liitettä
Tarkastajat:	Professori Mikael Collan, Tutkijatohtori Jyrki Savolainen
Hakusanat:	ennakoiva huolto, jäljellä oleva käyttöikä, koneoppiminen, luokittelu

Tämän tutkielman tarkoituksena on ennustaa jäljellä olevaa käyttöikää käyttäen luokittelumalleja. Menetelmää havainnollistetaan NASA:n suihkuturbiinimoottorien vikojen kehittymistä kuvaavalla sensoridatalla. Kolme luokittelujärjestelmää muodostetaan jakamalla jäljellä oleva käyttöikä luokkiin. Järjestelmissä on 10, 6 ja 4 luokkaa. Luokittelumallien kouluttamisessa käytetään random forest (RF) ja neural network (NN) -algoritmeja. Luokittelujärjestelmien ja algoritmien välisen suorituskyvyn vertailu sekä optimaalisten parametrien selvittäminen ovat tämän tutkielman keskiössä. Suorituskykyä mitataan pääasiassa Matthewsin korrelaatiokertoimella (MCC), jonka lisäksi hyödynnetään myös confusion matriiseja. Tulokset osoittavat, että tarkastelun alla olevat parametrit eivät ole kovin tärkeitä mallin suorituskyvyn kannalta. Vähiten luokkia sisältävä luokittelujärjestelmä tuottaa selvästi paremmat tulokset verrattuna kahteen muuhun järjestelmään, jotka ovat suorituskyvyltään lähempänä toisiaan. RF suoriutuu paremmin kuin NN jokaisen luokittelujärjestelmän kohdalla, mutta NN:n parametrien optimointiin tulisi kiinnittää enemmän huomiota parhaan luokittelutarkkuuden varmistamiseksi. Mikään malleista ei kuitenkaan yllä lähelle täydellistä luokittelutulosta. Siitä huolimatta tulokset osoittavat, että menetelmässä on potentiaalia, mutta erityisesti luokkajaotteluun tulisi kiinnittää tarkempaa huomiota.

ACKNOWLEDGEMENTS

Five years passed surprisingly fast. I didn't get to even really realize that this journey was coming to an end. Yet, it seems to be the truth, and now it is time for a change of scenery and new challenges. I certainly enjoyed this episode of my life and it will be warmly remembered in the future.

First of all, I would like to thank my supervisor Mikael for helping to find the subject and for all the guidance he provided. Also, a thank you to Jyrki for the feedback.

A huge thank you to all of my family members for supporting me throughout the studies. Knowing that you always got my back has been essential.

Thanks to all my friends, both old and new ones, for making this journey such a pleasure. And kudos to Teemu, you being there with me was crucial for even making this journey possible. Those long days in Kontula library will not be forgotten.

Finally, enormous special thanks to Juhana for, well, everything. Your influence has been irreplaceable, and I am truly grateful for it. Also, the basketball folks deserve a lot of praise for all the fun times with and without basketball, especially during this very unusual, coronavirus-oriented spring.

In Lappeeranta, May 22nd, 2020

Tuomas Peltola

Table of Contents

1 Introduction 10

1.1 Motivation 11

1.2 Theoretical Framework 11

1.3 Objectives and Research Questions 12

1.4 Structure 13

2 Theory 14

2.1 Predictive Maintenance 14

2.1.1 Remaining Useful Life 16

2.1.2 Prediction Techniques for RUL 17

2.2 Machine Learning 19

2.2.1 Main Paradigms of Machine Learning 20

2.3 Classification Techniques 24

2.3.1 Decision Tree and Random Forest 24

2.3.2 Neural Network 24

2.4 Evaluation of Classification Models 25

2.4.1 Binary Confusion Matrix 25

2.4.2 Multi-class Confusion Matrix 27

2.4.3 Receiving Operating Characteristics Curve 28

2.4.4 Matthews Correlation Coefficient 30

2.5 Validation 31

3 Literature Review 33

3.1 Search Process 33

3.2 Review 35

3.2.1 Previous Research on Binary Classification 37

3.2.2 Previous Research on Multi-class Classification 40
3.3 Summary of Literature Review 41
4 Methodology 44
4.1 Data 45
4.2 Classification Framework 47
4.3 Data Pre-processing and Feature Selection 52
4.4 Classification Algorithms 55
4.5 Evaluation Metrics 56
5 Results 57
5.1 Hyperparameter Optimization 57
5.1.1 Random Forest 58
5.1.2 Neural Network 62
5.2 Final Model Evaluation 66
5.2.1 Final evaluation of Random Forest Models 66
5.2.2 Final evaluation of Neural Network Models 71
5.3 Summary of Results 74
6 Conclusions and Further Research 76
6.1 Limitations and Further Research 79
References 80
Appendices 86

List of Figures

Figure 1. Theoretical Framework of the Study 12
Figure 2. Framework of Predictive Maintenance (after Crespo Marquez, 2007, 70; Jardine et al, 2006) 15
Figure 3. Classification of RUL Prediction Models (after Liao & Kottig, 2014; Fink et al., 2015; Zhang et al., 2018) 18
Figure 4. Major Subfields of AI and Basic ML Paradigms (after Vijipriya et al., 2016) 19
Figure 5. Flow of Supervised Learning 21
Figure 6. Flow of Unsupervised Learning 22
Figure 7. Flow of Reinforcement Learning 23
Figure 8. Binary Confusion Matrix 26
Figure 9. Multi-class Confusion Matrix 28
Figure 10. Example of a ROC Curve 29
Figure 11. The Literature Search Process 34
Figure 12. Model Creation Process 44
Figure 13. Distribution of RUL Values for Train and Test Sets 49
Figure 14. Class Distributions 52
Figure 15. Data Treatment Process 53
Figure 16. Average MCC's of RF System 1 with Different Nodesizes and Number of Trees 59
Figure 17. Average MCC's of RF System 2 with Different Nodesizes and Number of Trees 60
Figure 18. Average MCC's of RF System 3 with Different Nodesizes and Number of Trees 61
Figure 19. Average MCC's of NN System 1 with Different Number of Hidden Layers and Neurons per Layer 63
Figure 20. Average MCC's of NN System 2 with Different Number of Hidden Layers and Neurons per Layer 64
Figure 21. Average MCC's of NN System 3 with Different Number of Hidden Layers and Neurons per Layer 65
Figure 22. Confusion Matrix of RF System 1 with Visualization 67
Figure 23. Confusion Matrix of RF System 2 with Visualization 68
Figure 24. Confusion Matrix of RF System 3 with Visualization 70

Figure 25. Confusion Matrix of NN System 1 with Visualization 71
Figure 26. Confusion Matrix of NN System 2 with Visualization 72
Figure 27. Confusion Matrix of NN System 3 with Visualization 73

List of Tables

Table 1. Evaluation Metrics based on Confusion Matrix (Sokolova & Lapalme, 2009) 27
Table 2. Illustration of 5-fold Cross-validation with Separate Testing Dataset 32
Table 3. Summary of Reviewed Literature 36
Table 4. Concept matrix 42
Table 5. Description of the Turbofan Dataset 45
Table 6. The Structure of the Turbofan Data (Dataset FD001) 46
Table 7. MCC's of All Models 74

List of Appendices

Appendix 1. Average MCC's of All RF Systems 86
Appendix 2. Average MCC's of All NN Systems 88
Appendix 3. Confusion Matrix Absolute Values of All Models 90

List of Abbreviations

AI	Artificial Intelligence
AUC	Area Under Curve
BTA	Boosted Tree Algorithm
CBM	Condition Based Maintenance
CM	Condition Monitoring
C-MAPSS	Commercial Modular Aero-Propulsion System Simulation
DT	Decision Tree
ELM	Extreme Learning Machine
FN	False Negative
FP	False Positive
FPR	False Positive Rate
IoT	Internet of Things
KNN	K-nearest Neighbours
LR	Logistic Regression
MCC	Matthews Correlation Coefficient
ML	Machine Learning
MR	Misclassification Rate
NN	(Artificial) Neural Network
PdM	Predictive Maintenance
RF	Random Forest
ROC	Receiving Operating Characteristics
RUL	Remaining Useful Life
SVM	Support Vector Machine
SVR	Support Vector Regression
TN	True Negative
TP	True Positive
TNR	True Negative Rate
TPR	True Positive Rate

1 Introduction

The awareness and technologies concerning the Internet of Things (IoT) have expanded rapidly during the recent years and more is to come. This has already led to the industrial applications becoming more common, which has allowed the emerging of numerous new possibilities to connect physical objects with each other. Furthermore, these connections enable the collection of data related to the physical objects.

At the same time, there have been major advancements in data science while the computational costs have continued to decrease. Therefore, problems that used to be hard to solve and computationally heavy and costly, have turned into rather feasible tasks due to the improvement of new techniques.

The exponential increase of data provides an opportunity to analyze things with new ways and gain valuable information. One field where this information can offer new kind of benefit is maintenance. It is not cost-effective to perform maintenance actions too much in advance as preventive actions, when there is still operational lifetime remaining. By combining massive amounts of data with advanced analytics, the need of maintenance can be predicted before failures occur which allows actions to be made in more optimal time.

The prediction approach in maintenance is often referred as predictive maintenance (PdM). It can be stated that the main purpose of PdM is to reduce maintenance related costs. In more detail, reducing the operational downtime and direct costs related to maintenance such as labour and spare materials. A popular measure of interest in literature is remaining useful life (RUL), which will be adapted to this study as well.

In this study, the predictive maintenance will be considered from data-driven point of view. Machine learning (ML) techniques are one method to perform PdM as data-driven, and ML will be used as the focal point. The focus will be on classification algorithms, which are rather rarely used in similar kind of problem setting and this will be discussed more later on. NASA's Turbofan Engine Degradation Simulation Dataset about simulated aircraft engines will be used to demonstrate the methodology.

1.1 Motivation

The remaining useful life prediction is about forecasting a defined unit of time. In other words, the target variable is continuous and thus the problem categorizes as so-called regression problem. The usual approach would be to use a model which's function is to model continuous target variable.

The usage of classification model in such situation instead might be not be considered necessary or even reasonable as regression models should exploit the data better and hence provide better results. However, there seem to be some reasons why classification models could be considered over regression models.

Böhm (2017) states that RUL prediction can be quite challenging task and possible problems could be related to uncertainty in measurement data, long horizon of prediction for small units of time. Fink, Zio & Weidman (2015) argue that classification approach is reasonable procedure when the used data consist of many discrete variables as then other methods can be either not applicable, or applicable with notable limitations.

Xue, Williams & Qiu (2011) suggest that reshaping the continuous problem into classification problem by determining "pre-failure" and "normal" states of a system is a commonly used approach in practice. Also Fink et al. (2015) point out that the classification approach matches the needs of practical application as the maintainers need to know whether a failure will occur within a beforehand decided time period, which can be modified easily.

On the other hand, the transformation into classification might stabilize the end user acceptance (Böhm, 2017). Xue et al. (2011) add that especially binary classification has advantages in the form of easy usability and robustness.

The reasons of using classification model instead of regression model might not exist within the used turbofan data. However, there are very few studies concerning the classification approach and therefore more evidence is requisite. Additionally, it is interesting to investigate how well the classification approach performs with this particular data.

1.2 Theoretical Framework

This study will focus on remaining useful life (RUL) prediction with ML classification models, in particular random forest (RF) and neural network (NN) classifiers. The theoretical framework of this study can be better seen from Figure 1.

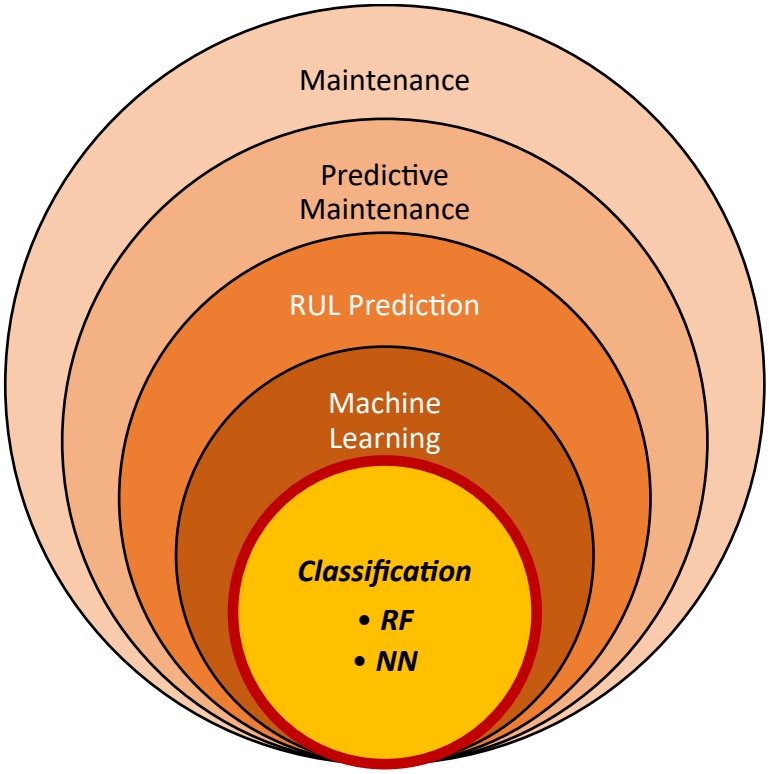


Figure 1. Theoretical Framework of the Study

The literature on RUL prediction with classification models is rather narrow. Thus, some compromises regarding the literature selection might be required. Then, the focus will shift into empirical part focusing on how the classification approach performs with the turbofan data.

1.3 Objectives and Research Questions

The main goal of the study is to predict remaining useful life with classification methods. After some theoretical basics, this study will review previous literature about RUL prediction with classification approach. Regarding the previous literature, the first research question along with two subquestions are formed:

1. **How previous research has approached RUL-type problems with classification?**
   a. Which algorithms have been used the most in RUL classification?
   b. How the performance of classification models has been evaluated?

Additionally, the number of classification models is narrowed to two: random forest (RF) and neural network (NN). The reasons why these models are selected will be discussed in more detail in chapter 4. For both classifiers, parameters will have to be defined and it is one important part of the study to evaluate which parameters lead to the best results. Also, RF and NN performance comparison is essential. Two research questions are formed based on these aspects:

2. **What random forest and neural network parameters lead to the best performance with the turbofan dataset?**
3. **How random forest and neural network compare to each other with the turbofan dataset?**

Finally, time intervals need to be defined for the classification of RUL and this issue will be addressed more specifically later in chapter 4. A good approach would be to evaluate the time required to take action regarding the maintenance issue. However, the purpose of this study is not to hypothesize about what would be a good and sufficient time interval to do so. Instead, the classification will be demonstrated with few different classification systems, which then will be compared. This leads to the final research question:

4. **How random forest and neural network classifiers perform for different classification systems with the turbofan dataset?**

1.4 Structure

The structure of this thesis is now presented shortly. First, in chapter 2, predictive maintenance and machine learning will be examined from theoretical point of view and some important concepts for the purpose of this study are explained. Chapter 3 contains a literature review about classification methods used in PdM. That is followed by the introduction of used methodology in this study in chapter 4. Chapter 5 will focus about the evaluating and discussing the results. Finally, conclusions will be drawn in chapter 6, along with discussion about the limitations concerning this study and suggestions about potential subjects for further research.

2 Theory

In this chapter, the concept of predictive maintenance (PdM) will be introduced. It will be examined how maintenance can be classified into preventive and corrective maintenance, and how predictive maintenance differs from these two.

After reviewing the maintenance concept, a popular time measure in PdM literature, remaining useful life (RUL), will be introduced and discussed. That is followed by a brief introduction of different approaches to implement predictive maintenance.

Following the maintenance concept, the concept of Machine Learning (ML) will be addressed. Aside from the actual basic concept of ML, the three main ML paradigms will be briefly presented, and their differences compared.

Finally, the theory part will focus more on classification which is the used approach and focal point in this study. The selected classification techniques will be discussed, and a short introduction of model evaluation will be included as well.

2.1 Predictive Maintenance

The field of predictive maintenance (PdM) has been studied widely in the past 20 or so years. During this time period, technical development and continuously increasing amount of data has led to better opportunities to utilize data in maintenance. PdM can help decrease maintenance costs and operational downtime, and therefore it is only reasonable that the literature about the subject has been and still is extensive.

First of all, it is reasonable to define predictive maintenance. In order to do so, the concept of maintenance needs to be examined. Crespo Marquez (2007, 69) defines maintenance as a combination of actions which in intend to 1) retain an item in, or 2) restore an item to, a state in which the item can perform a given function. Therefore, the author suggests maintenance classification into two main groups: actions striving for retaining given conditions (Preventive Maintenance) and actions to restore certain conditions (Corrective Maintenance). This division is further demonstrated in Figure 2 (Crespo Marquez, 2007, 70; Jardine, Lin & Banjevic, 2006).

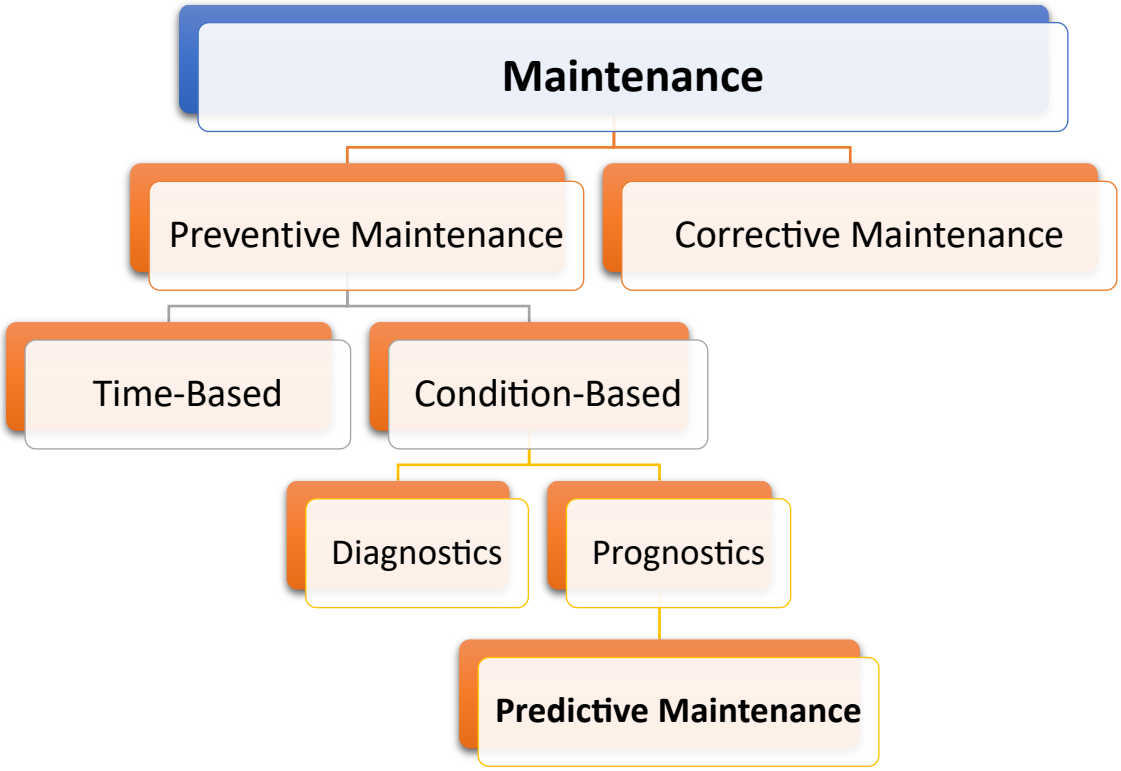


Figure 2. Framework of Predictive Maintenance (after Crespo Marquez, 2007, 70; Jardine et al, 2006)

Corrective maintenance is not the main focus in this study and therefore it is not necessary to investigate it further. That said, the focus will remain on preventive maintenance instead. The function of preventive maintenance is to carry out the maintenance actions before a failure happens. According to Crespo Marquez (2007, 70), preventive maintenance can be carried out based on either time or condition.

Time-based (or predetermined) maintenance is put into practice according to some, usually established, time measure. The time measure can be for example a given time period or number of used units. However, time-based maintenance does not take into account the condition. (Crespo Marquez, 2007, 69-70) Due to the omission of condition, it can be argued that the object of maintenance might still be in sufficiently good condition to be used without maintenance. This could lead to constantly too early maintenance actions and unnecessary maintenance costs.

Condition-based maintenance (CBM) relies on monitoring the performance or the parameters of a certain item. This can be executed in continuous, on-request or scheduled manner.

(Crespo Marquez, 2007, 70) The aim of CBM is to avoid unnecessary maintenance actions and only recommend actions when it is actually necessary (Jardine et al., 2006). Therefore, efficient CBM can reduce maintenance costs by getting rid of unnecessary time-based maintenance actions.

Jardine et al. (2006) propose that CBM can be further divided into diagnostics and prognostic. Diagnostics focuses on fault detection, isolation and identification when they appear. Prognostics concentrates on failure or fault prediction before they emerge. Based on these definitions, it can be said that diagnostics is posterior, and prognostics is prior event analysis. Consequently, prognostics can be argued to be more effective way to minimize operative downtime. Diagnostics becomes beneficial when prognostics (fault prediction) fails and failure actually occur. (Crespo Marquez, 2007, 310-314; Jardine et al., 2006)

Considering everything previously covered, predictive maintenance will fall under prognostics as suggested in Figure 2. PdM applies predictive tools to assess when maintenance actions are required in order to avoid failure (Carvalho et al., 2019). There are different approaches to PdM and those will be discussed more in subchapter 2.1.2.

To summarize, predictive maintenance has an important role in maintenance. If executed efficiently, it can provide great cost-saving benefits. However, the prediction of maintenance need is not an easy task as perfect prediction accuracy is basically impossible to achieve (Carvalho et al., 2019) .

In order to achieve benefits with PdM, a measure to be predicted is required. In this study, remaining useful life will be used as a measure. The concept of remaining useful life in PdM will therefore be introduced next.

2.1.1 Remaining Useful Life

Remaining Useful Life (RUL) is a widely used measure in prognostics and PdM literature when predicting time to machinery fault or failure. RUL measures the time remaining before an error or failure occurs in the machinery, given the current condition profile of the machinery (Jardine et al., 2006).

The definition of RUL can be presented as a conditional random variable (Jardine et al., 2006):

$$RUL = (T - t) \mid T > t, Z(t) \quad (1)$$

In the definition, T signifies a random variable of machinery's time to failure, t stands for the current age of machinery and Z(t) refers to the machinery's condition profile at current time.

RUL obviously is a continuous measure and majority of previous studies have also encountered the problem in that way, meaning regression approach has been used instead of classification approach. However, as discussed earlier, the purpose of this study is to take the less popular approach in the form of classification, as suggested by some studies. The formation of classification framework for this study will be discussed in more detail in chapter 4.

2.1.2 Prediction Techniques for RUL

The previous literature has identified various different models for RUL prediction. These models can be further categorized and grouped. Zhang, Si, Hu, & Lei (2018) , Jardine et al. (2006), Fink et al. (2015), Schwabacher & Goebel (2007) and Liao & Kottig (2014) have all proposed to categorize the models rather similarly based on the background of the models. The three main categories can be expressed as knowledge-based techniques, model-based techniques and data driven techniques. This division is demonstrated in Figure 3 based on the literature. The figure does not represent an unconditional truth about the divisions as different hybrids between the listed techniques are also possible. It rather serves as a way to showcase the focus of this study.

The knowledge-based techniques usually require special knowledge concerning the observed system. In addition, some failure data is typically needed, which can be expensive to acquire. The most common knowledge-based techniques include expert systems and fuzzy systems. They try to identify similarities between current situations and previous failures. (Zhang et al., 2018)

The model-based techniques commonly rely on the physics of certain system. Predictions can be achieved with mathematical representations of the physics of a system's degradation process. Physical models can be fairly accurate but in order to use them it is required to

possess a good knowledge of the system's physics. The prediction accuracy is directly proportional to the quality and accuracy of the used model. (Zhang et al., 2018)

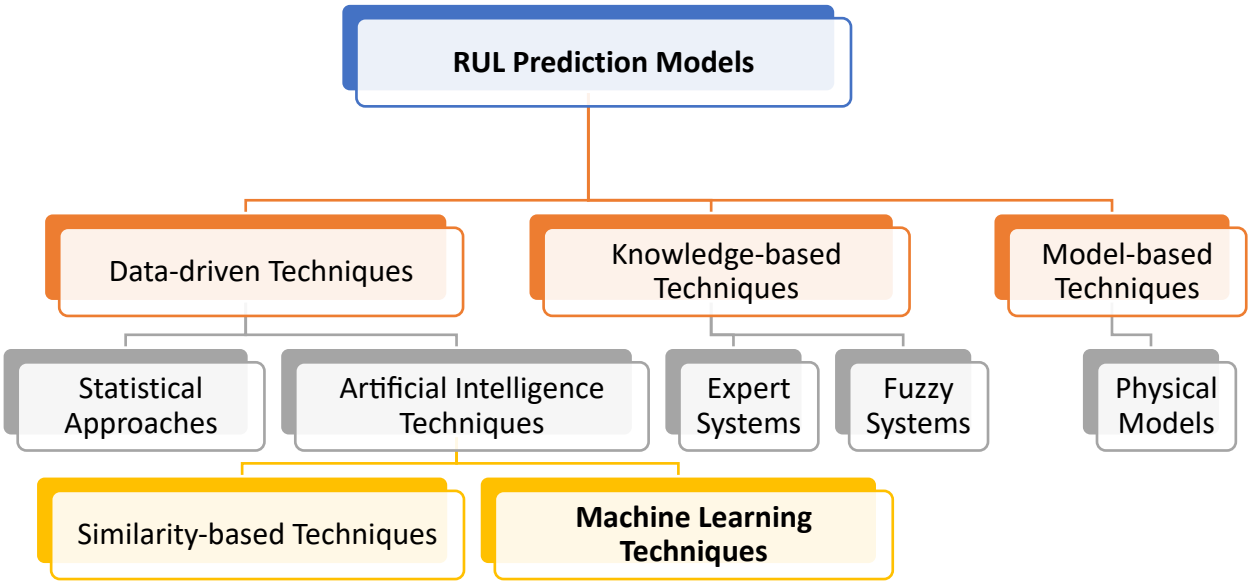


Figure 3. Classification of RUL Prediction Models (after Liao & Kottig, 2014; Fink et al., 2015; Zhang et al., 2018)

The data-driven techniques utilize existing data to provide predictions. Fink et al. (2015) propose that the data could be either failure data or condition monitoring (CM) data. Zhang et al. (2018) argue that data-driven techniques are relatively more flexible compared to the other techniques, thus making data-driven techniques a popular method in RUL prediction.

Data-driven techniques can be further divided into statistical approaches and artificial intelligence techniques. Artificial intelligence techniques can be further split into machine learning (ML) and similarity-based techniques.

According to Si, Wang, Hu & Zhou (2011), statistical approaches can be further divided into two categories based on the CM data and whether it is direct or indirect. Then, the direct CM statistical approaches would include methods such as regression analysis, Wiener processes, Gamma processes and Markovian analysis. Indirect CM methods would cover stochastic filtering, covariate-based hazard analysis and Hidden Markov and Semi-Markov modelling.

Regardless of the technique, in practical applications it is usual that a model/system is once created and then put into use. For machine learning techniques this means that the training is usually stopped at some point not continued after that except for some special cases. Retraining can be done but continuous training is not usual for most applications.

In the scope of this study, the focus is placed on data-driven techniques, more precisely in artificial intelligence and machine learning. Therefore, it is not necessary to examine the other techniques any further, but rather concentrate on ML techniques. Machine learning will be covered more in the next subchapter 2.2.

2.2 Machine Learning

This subsection will introduce the basic idea of Machine Learning and its potential in Predictive Maintenance will be discussed. Then, different types of machine learning techniques (supervised, unsupervised, reinforcement learning) are briefly explained.

Machine Learning (ML) is one of the major subfields of Artificial Intelligence (AI) as can be seen in Figure 4 (Vijipriya, Ashok, & Suppiah, 2016). The concept of ML is not new as it has existed since the 1970's when the first algorithms were introduced (Louridas & Ebert, 2016). The increase in computational power and the persistently growing amount of available data combined with development in ML algorithms and theory has led to ML being one of the most rapidly growing fields in technology (Jordan & Mitchell, 2015).

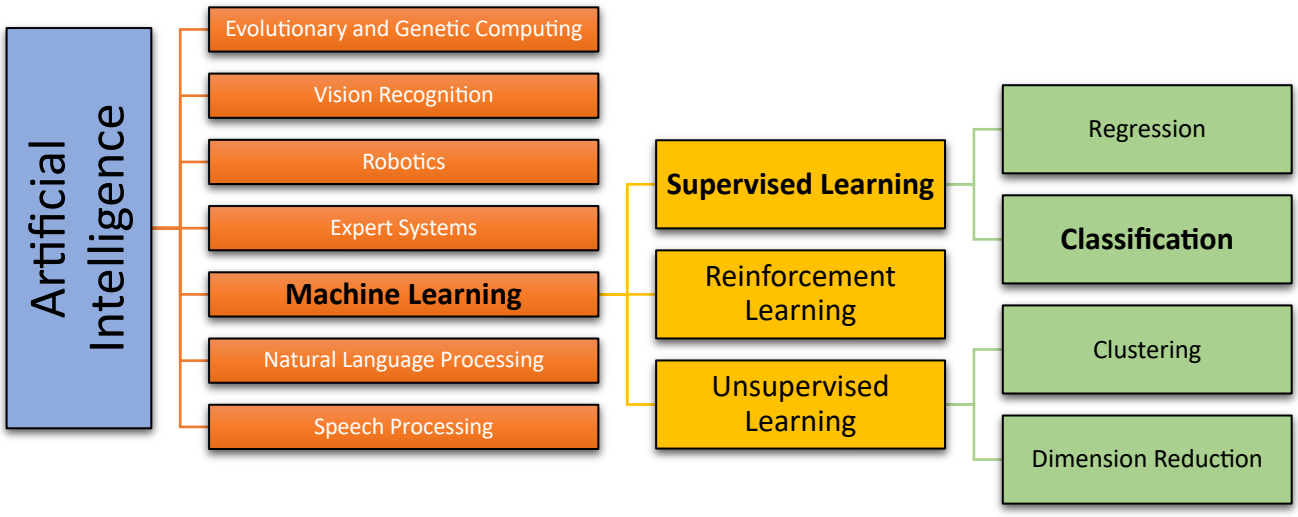


Figure 4. Major Subfields of AI and Basic ML Paradigms (after Vijipriya et al., 2016)

According to Jordan & Mitchell (2015), the field of machine learning is a crossing of computer science and statistics. Machine learning is based on past experience and aims to build computers that able to improve automatically using that experience (Jordan & Mitchell, 2015).

Machine learning can be divided depending on the used type of technique. Jordan & Mitchell (2015) suggest that these techniques can be divided into three main paradigms: supervised learning, unsupervised learning and reinforcement learning. Based on the problem setting, supervised and unsupervised learning can be divided even further, into classification and regression, and into clustering and dimension reduction, respectively (Louridas & Ebert, 2016). The main paradigms of machine learning will be discussed more next.

2.2.1 Main Paradigms of Machine Learning

As mentioned, types of machine learning techniques can be divided into supervised learning, unsupervised learning and reinforcement learning. The differences of these types are in the way they use data. The supervised and unsupervised learning have in common that both use historic data for training phase, whereas reinforcement learning does not use historic data as there is no training phase.

Training the model means that historic data is given to the model as input and the model tries to identify patterns to produce an output. These patterns can then be used in prediction when new data is given as input.

As presented in Figure 4, supervised learning techniques can be divided into regression and classification techniques. Basically, a problem would be regression-type when the desired output variable is continuous. An example for a regression problem would be house price (continuous) prediction or remaining useful life prediction, where certain time measure (continuous) would be used as predicted output variable.

In classification-type of problems, the goal is to find the correct class for given inputs. The classification problem can be binary or multi-class. For example, whether a customer will default or not, is a binary task because the classes would be “yes or “no”. Multi-class task would then obviously have multiple possible classes, for example whether a customer belongs to group 1, 2 or 3.

Supervised learning techniques are applicable, when the correct outputs are known. The outputs can also be called labels, the data may be called labeled if true outputs are known

and unlabeled if not. For regression problems, labeled historic data include the real values of output variable for individual instances. Labeled historic data in classification contain the correct classes of instances.

The flow of supervised learning model is demonstrated in Figure 5. The training data, consisting of the labeled historic data, is used with selected machine learning algorithm in order to create a ML model. Afterwards the model is trained, inputs of a new instance are introduced to the model and it is able to create a prediction of a class or a value, depending on the problem type.

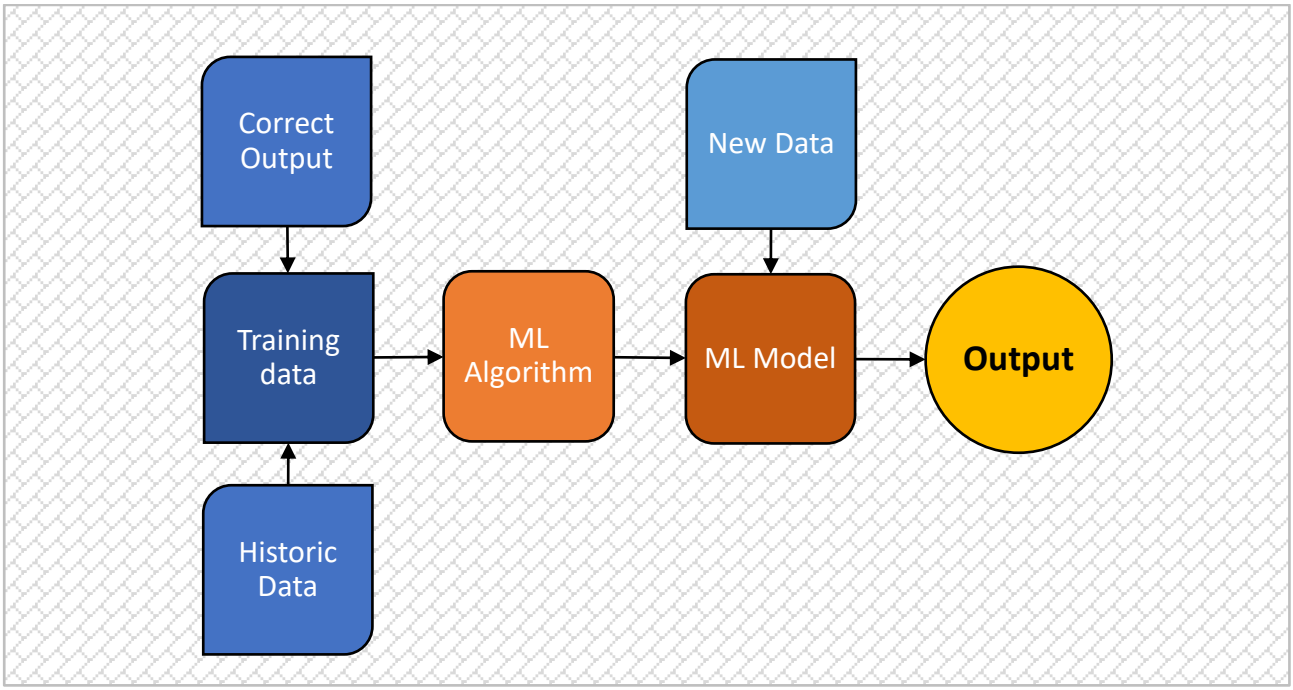


Figure 5. Flow of Supervised Learning

The historic data is usually divided into training and testing data. Training data is used to train the model and testing data is used to evaluate the model performance. Evaluation with training data would result in biased performance metrics as the model has formed the patterns based on that data. Therefore, the testing data can be introduced as new data for the model, but because the correct outputs are known, the accuracy of the model predicting new instances can be addressed. The evaluation of supervised learning will be discussed later in more detail.

Unsupervised learning differs from supervised learning in the way that the training data is unlabeled. With huge amounts of data, the model may be able to find patterns of similarity. Therefore, the purpose is to let the model discover the outputs and apply them to new instances. (Rebala, Ravi, & Churiwala, 2019) The flow of unsupervised learning is illustrated in Figure 6.

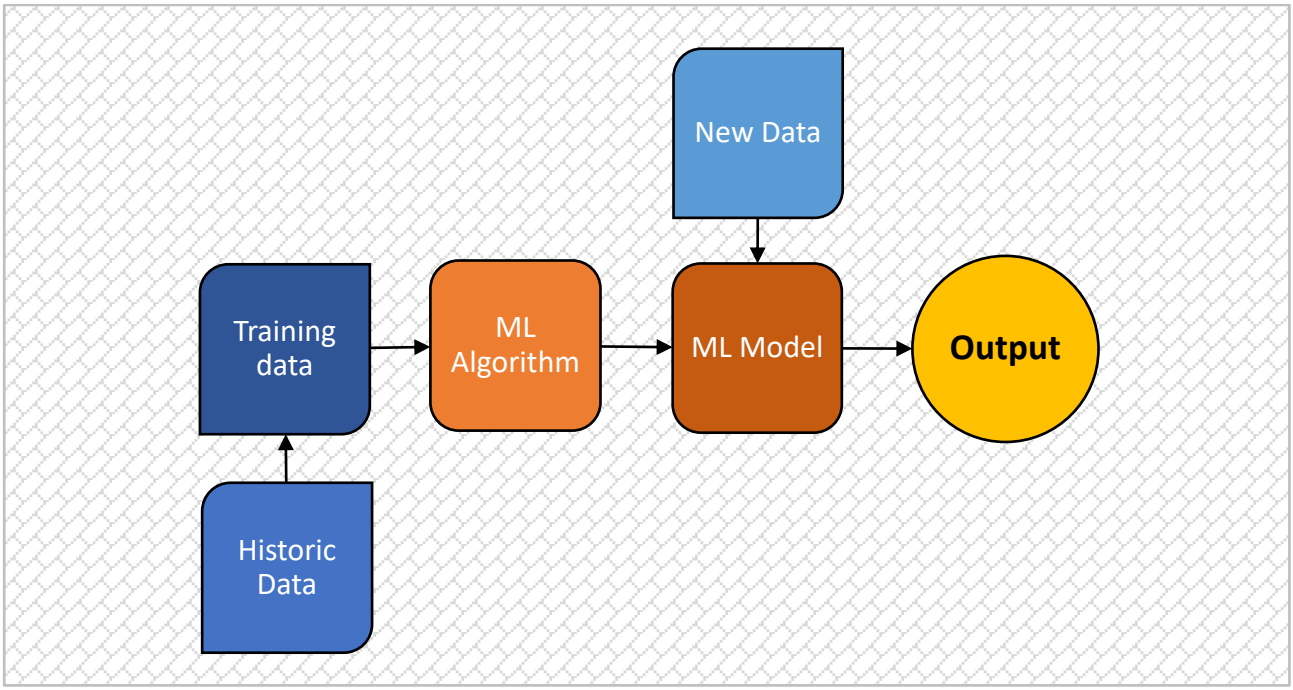


Figure 6. Flow of Unsupervised Learning

One technique of unsupervised learning is to identify and create groups from similar instances. This problem setting is called clustering. Another unsupervised learning technique is called dimensionality reduction. Its function is to take the original set of data with various dimensions, and then lower the number of dimensions so that the aspects of data would be better captured. (Louridas & Ebert, 2016)

The third one of basic machine learning paradigms, reinforcement learning, does not have similar training phase as the other two learning types have. Reinforcement learning is based on the model trying to learn from its own experience. Thus, it is based more on trial and error. The flow of reinforcement learning can be observed in Figure 7.

Reinforcement learning is useful in changing situations and when huge state space is involved. Chess is a good example as the situation (data) is changing continuously whenever a move is made, and the model's proposed next move has to take this changing environment into account. On the other hand, chess has close to infinite number of possible situations and brute force move optimization is not effective. Reinforcement learning models can learn through time to do actions based on the existing situation, aiming to maximize predefined goal. (Rebala et al., 2019, 22)

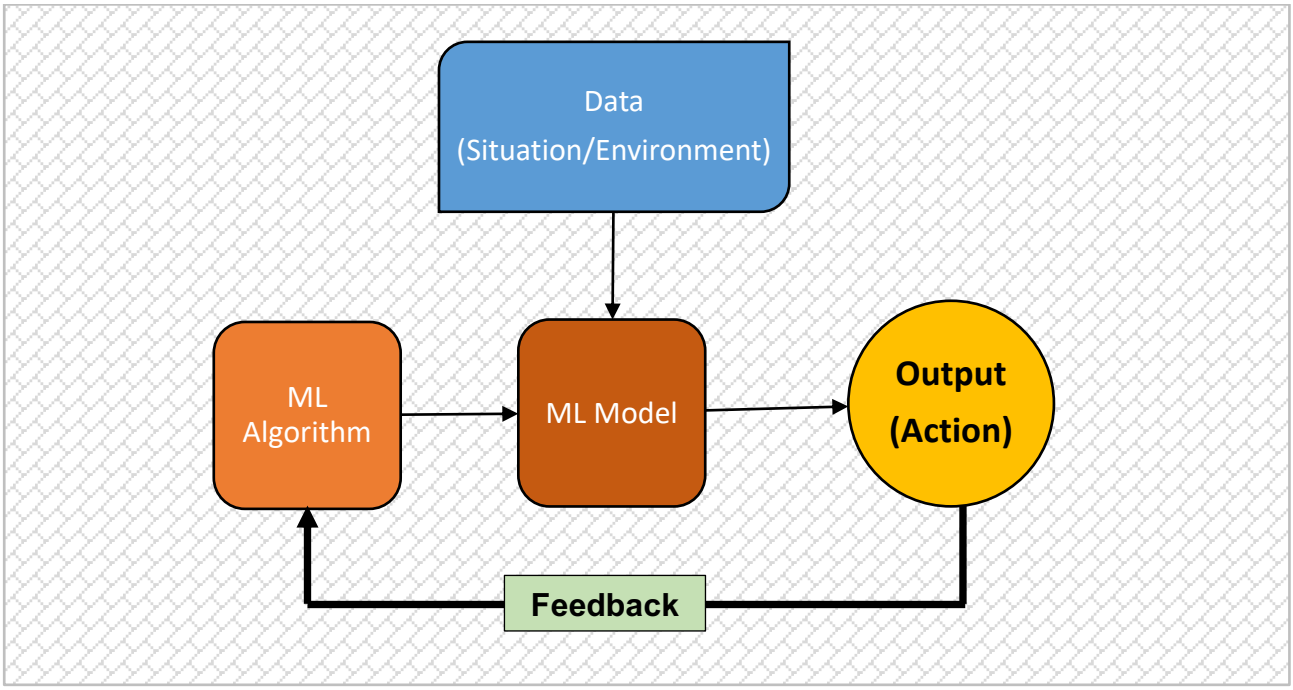


Figure 7. Flow of Reinforcement Learning

Based on all the information just presented, supervised learning is the most suitable ML approach for a remaining useful life prediction as the historic data is usually labeled. That holds true for the dataset used in this study as well. Therefore, the concentration will only be on supervised learning from now on. Also, as already mentioned, classification methods will be utilized in this study. Thus, the regression methods will not be introduced any further as the focus will be kept solely on classification.

2.3 Classification Techniques

This subchapter will introduce the classification techniques used in this study, particularly random forest and neural network. Thus, any other techniques, such as SVM, KNN, LR or naïve Bayes, will not be introduced as those are not the focus of this study. The literature review will cover some other techniques, but it is not required to know the concepts behind these techniques. Decision tree is an exception as it is essential in order to understand the logic behind random forest.

The reasons for selecting to use the techniques in question are discussed later in the methodology chapter. Only a very brief introduction of the basic idea of these techniques will be presented.

2.3.1 Decision Tree and Random Forest

Decision tree (DT) is a non-parametric, simple classifier which adapts the structure of a hierarchical tree and it can be used to perform supervised classification tasks. The logic behind the DT is quite simple. It consists of branches, which are connected with decision nodes. Each decision node tests a value of particular feature, and based on the value, leads to split. This structure is repeated from the starting node until the terminal nodes. The more decision nodes there are, the more complex the DT classifier is. (Dougherty, 2013, 27)

One benefit of DT is the interpretability as the rules from decision nodes can extracted and presented rather intuitively. The usage of DT's is also quite fast and does not require huge computational power in case of classification. However, the classifier tends to overfit easily and it has problems when the number of classes increases. (Dougherty, 2013, 38)

Random forest (RF) is an ensemble method which combines decision trees so that every tree is dependent of a randomly sampled vector of values with same distribution for each tree in the forest. The randomness in the process makes RF's accurate predictors and due to law of large numbers, they do not overfit like DT's. (Breiman, 2001)

2.3.2 Neural Network

Neural networks (NN) are biologically inspired computational models, consisting of elements called neurons and connections with weights between them. An NN typically consists of input layer and output layer, which are separated by number of hidden layers, usually from

one to three. The layers are formed by the neurons, which can connect to the neurons in previous and next layers. The main idea behind the model is that neurons take information, treat it accordingly in regard with selected activation function and then send the information to the next layer. Based on the value on the output layer, the weights between neurons are then adjusted so that desired predictive power is achieved. (Kubat, 2015, 91-93; Shanmuganathan & Samarasinghe, 2016, 4-10)

Neural network is a broad concept and many different modifications fall under the term. Four parameters can be considered to define an NN: the type of neurons, the connection architecture between neurons, the learning algorithm and the recall algorithm (Shanmuganathan & Samarasinghe, 2016, 7-8).

2.4 Evaluation of Classification Models

Evaluation of the models is an important part of the process as it is the only way to compare models with each other. It is possible to evaluate the models by two different aspects: how well they predict, and how efficient they are. However, only the classification power will be considered in this study. Some common evaluation metrics will be introduced and discussed in this subchapter.

2.4.1 Binary Confusion Matrix

Confusion matrix is a very popular approach to evaluate classification models. It provides the basis for many common evaluation metrics. A confusion matrix for binary problem is illustrated in Figure 8. As can be seen, it is a 2x2 matrix and reports the numbers of correctly and incorrectly classified observations. There exist four different possibilities, what an observation could be:

- True Positives (TP): predicted positive, actually positive
- False Positives (FP): predicted positive, actually negative
- False Negatives (FN): predicted negative, actually positive
- True Negatives (TN): predicted negative, actually negative

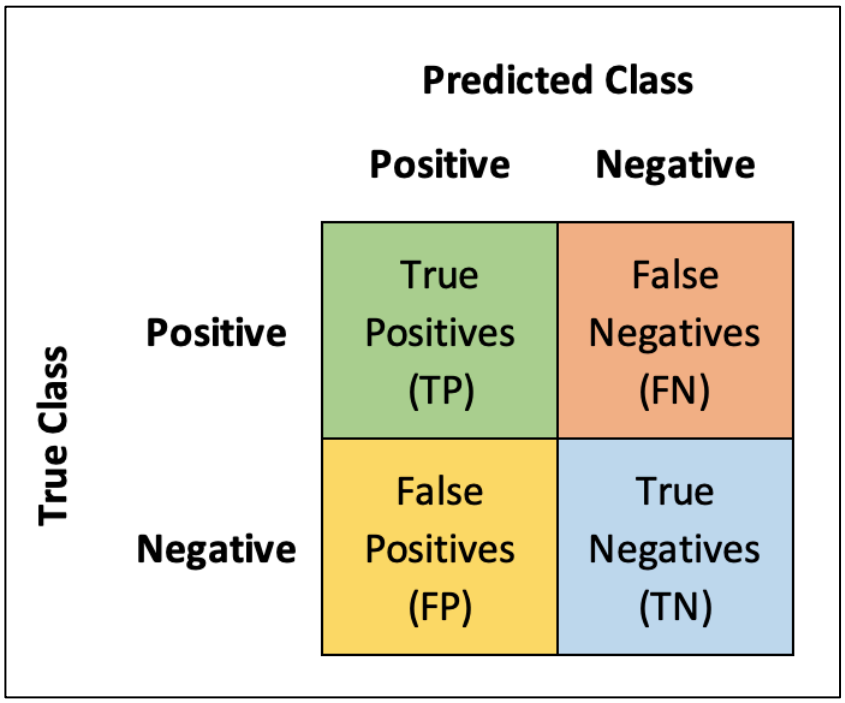


Figure 8. Binary Confusion Matrix

Confusion matrix provides opportunity to calculate different evaluation metrics by using ratios of the four outcomes. Table 1 summarizes some popular confusion matrix -based metrics for binary classification (Sokolova & Lapalme, 2009). According to Hossin & Sulaiman (2015), accuracy and its opposite misclassification rate (MR) are among the most used evaluation metrics. Accuracy and MR have their advantages as those are easy to compute, can be used for both binary and multi-class tasks, and are easy to interpret (Hossin & Sulaiman, 2015). Jurman & Furlanello (2010) argue that the role of accuracy is to roughly give first expressions about classifier goodness.

Accuracy and MR also have their limitations. Neither metric offers very distinctive or distinguishable values. Additionally, both metrics are not informative about classes and tend to favor larger class if the data is imbalanced. (Hossin & Sulaiman, 2015)

True positive rate (TPR), true negative rate (TNR), false positive rate (FPR) and precision pay attention to only one evaluation task as those focus on either positive or negative prediction at a time. These are more informative about the classes and might be useful in more specific cases when certain class is preferred over others. On the other hand, there usually exists notable trade-offs between these metrics and imbalanced class distribution aggravates situation even further.

Table 1. Evaluation Metrics based on Confusion Matrix (Sokolova & Lapalme, 2009)

Metric	Formula	Evaluation focus
Accuracy	$\frac{TP + TN}{TP + TN + FP + FN}$	Ratio of correct predictions over all predictions
Misclassification Rate (MR)	1 – Accuracy	Ratio of incorrect predictions over all predictions
True Positive Rate (TPR) or Sensitivity	$\frac{TP}{TP + FN}$	Share of correctly classified positive instances
True Negative Rate (TNR) or Specificity	$\frac{TN}{TN + FP}$	Share of correctly classified negative instances
False Positive Rate (FPR)	$\frac{FP}{FP + TN}$	Share of incorrectly classified negative instances
Precision	$\frac{TP}{TP + FP}$	Share of correct predictions over predicted positive instances
F1-score	$2 \times \frac{TPR \times Precision}{TPR + Precision}$	Harmonic mean of specificity and sensitivity

F1-score is a metric that combines the information benefits of TPR and precision in a single value. It also allows to favor either TPR or precision if one is more desirable feature than the other. Nevertheless, F1-score is not as simple to understand compared to the previously introduced metrics. The simpler metrics provide a clear, easy to understand value that allows the interpreter to comprehend what it actually means in reality. F1-score is a good measure to compare models, but its transferal into reality is more difficult.

Thus, it can be said that each evaluation metric has its use. There are differences in their informativeness and interpretability. Different metrics suit for different situations and problems.

2.4.2 Multi-class Confusion Matrix

Confusion matrix can be crafted also for multi-class classification problems. An example of n-class confusion matrix is represented in Figure 9. The interpretation of multi-class confusion matrix differs slightly from binary confusion matrix.

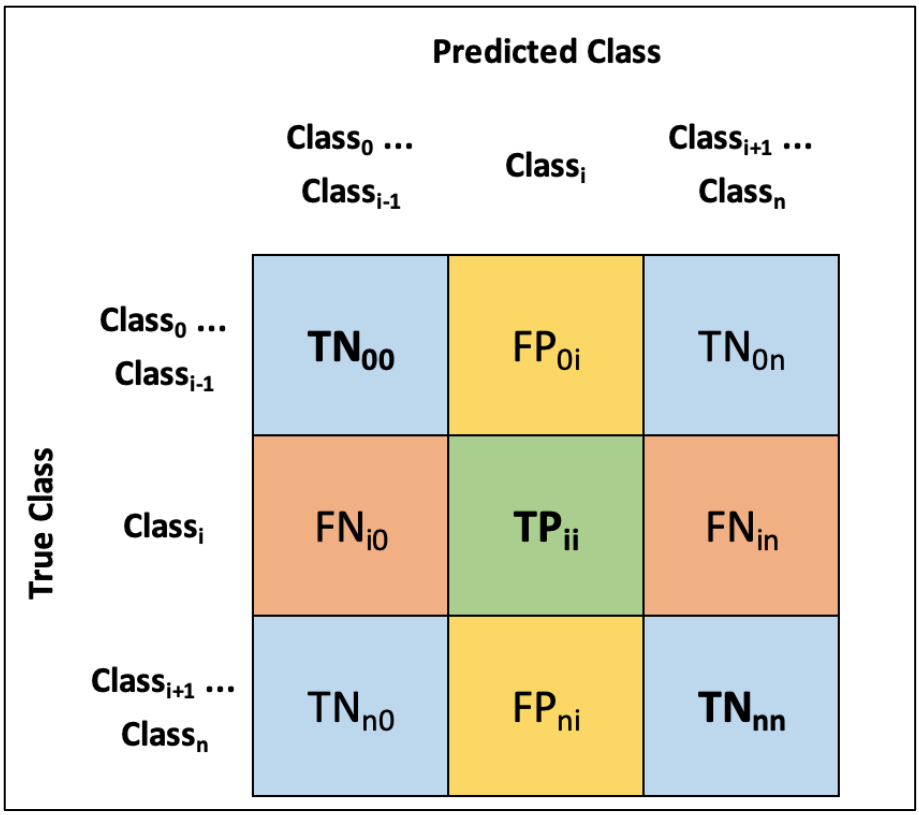


Figure 9. Multi-class Confusion Matrix

Now each class needs to be addressed separately. The true positives lie on the main diagonal where each predicted class counterparts its true class. In Figure 9, $Class_i$ is selected to be under investigation and similarly to binary confusion matrix, false positives are located in the same column and false negatives in the same row.

The metrics introduced while ago for binary confusion matrix can be generalized for multi-class cases. The calculation differs so that a metric will be calculated individually for each class which will be used to calculate an average. This can be conducted by either micro- or macro-averaging. Micro-averaging tends to favor bigger classes, whereas macro-averaging treats the classes equally regardless of the size. (Sokolova & Lapalme, 2009)

2.4.3 Receiving Operating Characteristics Curve

Receiving operating characteristic (ROC) curve is another metric that can be used to evaluate classification models. The curve is constructed by plotting true positive rate against false positive rate. Therefore, the curve visualizes the performance of the classifier by

showing the trade-off between TPR and FPR. (Fawcett, 2006) Figure 10 provides an example of a ROC curve.

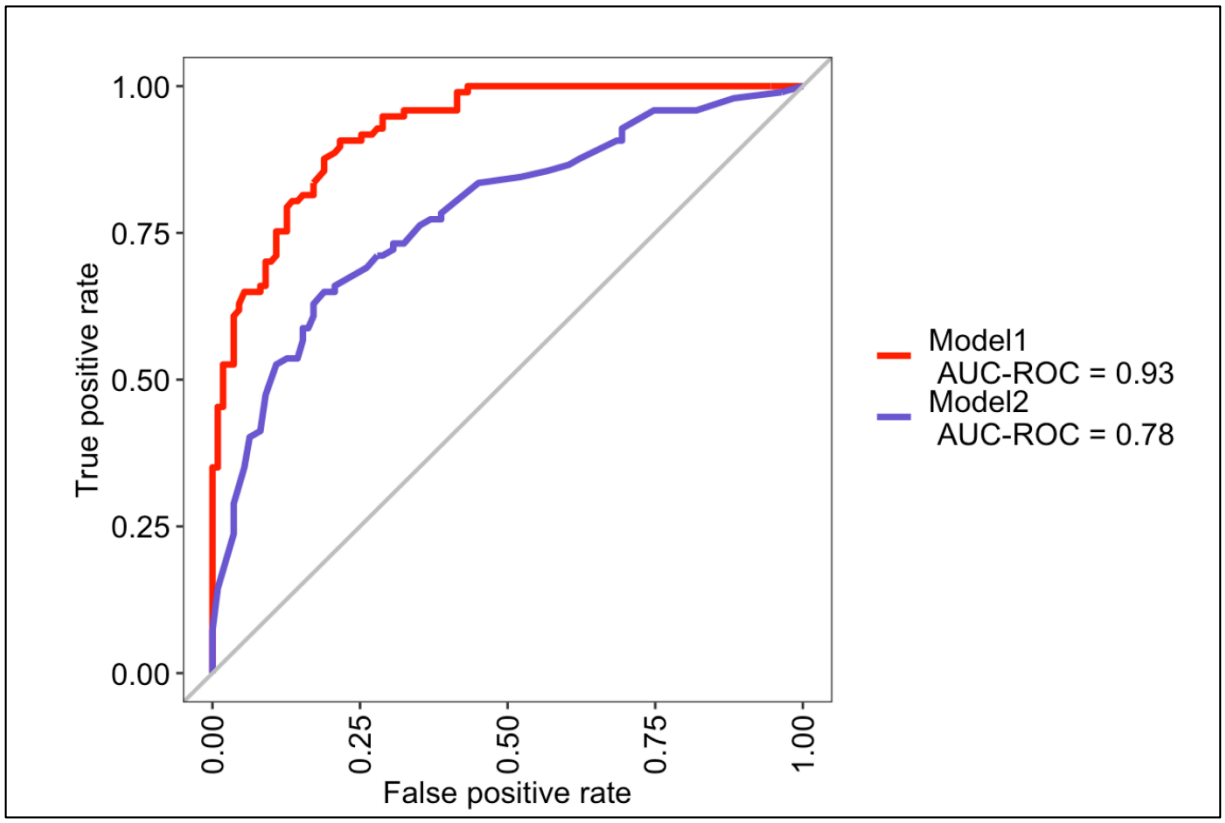


Figure 10. Example of a ROC Curve

As seen in Figure 10, a threshold for a random classifier can be drawn with a simple line on the diagonal. If the curve is above that threshold, the classifier performs better than coin flip. Perfect classifier would be obtained in the top left corner where TPR is 1 and FPR is 0.

However, only graphical metric can make it troublesome to compare between models and a single number would be more convenient. Area under ROC curve (AUC) provides that by simply calculating the area below the curve, thus returning a single value between 0 and 1. AUC value of 1 therefore refers to perfect classifier and 0,5 to random classifier. (Fawcett, 2006)

AUC has been proven to be better metric than accuracy in performance evaluation (Hossin & Sulaiman, 2015). According to Bradley (1997), AUC also has some desirable benefits over accuracy. However, AUC do not have well established extension into multi-class classification (Jurman & Furlanello, 2010; Sokolova & Lapalme, 2009), which limits its usability in more complex problems.

2.4.4 Matthews Correlation Coefficient

Matthews correlation coefficient (MCC) was first introduced by Matthews (1975) for binary contingency table. According to Jurman & Furlanello (2010), MCC has then increased its popularity among machine learning applications as a good single value metric to summarize confusion matrix in binary classification. The MCC for binary confusion matrix can be calculated as follows (Jurman & Furlanello, 2010):

$$MCC_{binary} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \quad (2)$$

However, MCC is not limited only to binary classification such as AUC is. Gorodkin (2004) proposed a generalization for MCC which can then be applied to multi-class confusion matrix with additional indices:

$$MCC_{multi-class} = \frac{\sum_{i,l,m=1}^{N} C_{ii}C_{ml} - C_{li}C_{im}}{\sqrt{\sum_{i=1}^{N}(\sum_{l=1}^{N} C_{li})(\sum_{\substack{f,g=1 \\ f \neq i}}^{N} C_{gf})} \sqrt{\sum_{i=1}^{N}(\sum_{l=1}^{N} C_{il})(\sum_{\substack{f,g=1 \\ f \neq i}}^{N} C_{fg})}} \quad (3)$$

, where *C* means class, *N* is number of classes and indices *i*, *l*, *m*, *f* and *g* are referring to classes.

MCC gets values between [-1,1], 1 meaning perfect classification and -1 perfect misclassification. If the confusion matrix is all zeros except for one column (all instances classified to the same class), the MCC gets value 0. MCC takes the class distribution better into consideration compared to the previously introduced metrics, although it does not do it perfectly. On the other hand, MCC is rather easy to interpret compared to some more complex metrics. Thus, MCC compromises well between interpretability and discriminating the classes, making it a good evaluation metric for general purposes. (Jurman & Furlanello, 2010)

MCC will be used as the primary evaluation metric in this study. It will also be complemented with visualizations of confusion matrices. Chapter 4 will provide the reasons for this selection in more detail.

2.5 Validation

The classification models, as any machine learning models, need to be validated in order to ensure that the model under examination can be generalized, i.e. the model is not overfitting and the results are not biased. The overfitting problem is highly likely if all of the data is used for training the model, and then conclusions are made using the same data.

The so-called holdout method is one way to validate the model. The data is divided into two subsets, namely training set and testing (or holdout) set. There is no exact correct split ratio for that but usually something around 2/3 for training set and 1/3 for testing set is used. The model is trained with the training set and testing set will be used to evaluate the model. (Kohavi, 1995) This allows to test the model performance with data that is completely new to the model and therefore gives a better overview about the generalizability.

The downside of leaving data for testing is that it reduces the amount of available data for training. There exists a trade-off between bias and variance when splitting the data. Larger training set decreases the bias but makes the testing set smaller, thus increasing the variance of test error estimate. (Kohavi, 1995)

Also, if the testing set is only one random sample of the whole data, there is a possibility that it happens to be substantially good or bad sample, thus leading to misleading results. Then, the generability of the model might suffer.

The model construction might have certain interphases, such as parameter optimization. The performance with different parameter combinations should be validated with some other data than the testing set in order to avoid specifically favorable parameters for the testing set. Separating an individual validation set would solve this, but it would reduce the number of observations in the training set.

K-fold cross-validation can be used to validate the results during the interphases. It allows the usage of the whole training data, while still leaving the testing data independent. The k-fold cross-validation method functions so that the training set will be divided into k subsamples which are equal size. That is followed by the model training by using k-1 folds as training data and the additional subsample as testing data. This procedure is repeated k times, each time the testing sample being different. (Kohavi, 1995) Table 2 demonstrates 5-fold cross-validation with additional testing set left out.

Table 2. Illustration of 5-fold Cross-validation with Separate Testing Dataset

k = 5	Whole Dataset					
	Training Dataset					Testing Dataset
	Training				Validation	
Split 1	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Evaluation Metric 1
Split 2	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Evaluation Metric 2
Split 3	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Evaluation Metric 3
Split 4	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Evaluation Metric 4
Split 5	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Evaluation Metric 5
						Average Evaluation Metric

The performance obtained with cross-validation can be defined with average of all combinations' evaluation metrics (Arlot & Celisse, 2010). The procedure illustrated in Table 2 reflect the cross-validation to be used during the model construction phase while the final performance is evaluated using the separate testing set.

This study will utilize the validation strategy presented in Table 2. Thus, the data will be divided into training and testing sets. Then, 5-fold cross-validation will be performed with the training set to optimize the parameters. Finally, the final models will be trained with whole training set and the evaluation will be done using the testing set.

3 Literature Review

This chapter will focus on reviewing the earlier literature related to remaining useful life estimation with classification methods. First, the process of finding and selecting related literature will be shortly explained. That will be followed by the actual reviewing and discussing section.

3.1 Search Process

As it has been already mentioned earlier, the existing amount of research concerning RUL prediction with classification approach is rather limited. This fact made the construction of the literature review quite challenging as it should be about articles that somehow relate to the subject under investigation.

Regardless of the challenges and limitations, a literature review has to be conducted. Webster and Watson (2002) suggest that the searching process should be done using a structured approach in the determination of selected literature. The procedure used in this study is demonstrated in Figure 11.

Finna was used as primary searching platform to begin the process but as explained soon, some other resources had to be used as well in order to collect somewhat sufficient amount of related literature. Finna is a platform that among other things has access to multiple databases containing articles published in scientific publications. Finna was used as the first option for searching as it covers wide range of publications and also grants access to them.

Finna search provided 47 articles. Titles and abstracts of these articles were reviewed and based on that the relevant articles were chosen. Classification was the matter that was especially looked to be included in these articles. However, majority of them did not have anything to with classification. After reviewing this set of articles, only three articles turned out to be relevant.

Google Scholar was used next. Keywords “classification in RUL prediction” were used and the based on the title some potential articles were reviewed. Again, classification aspect was a demanded feature, and thus this action resulted in one more related article.

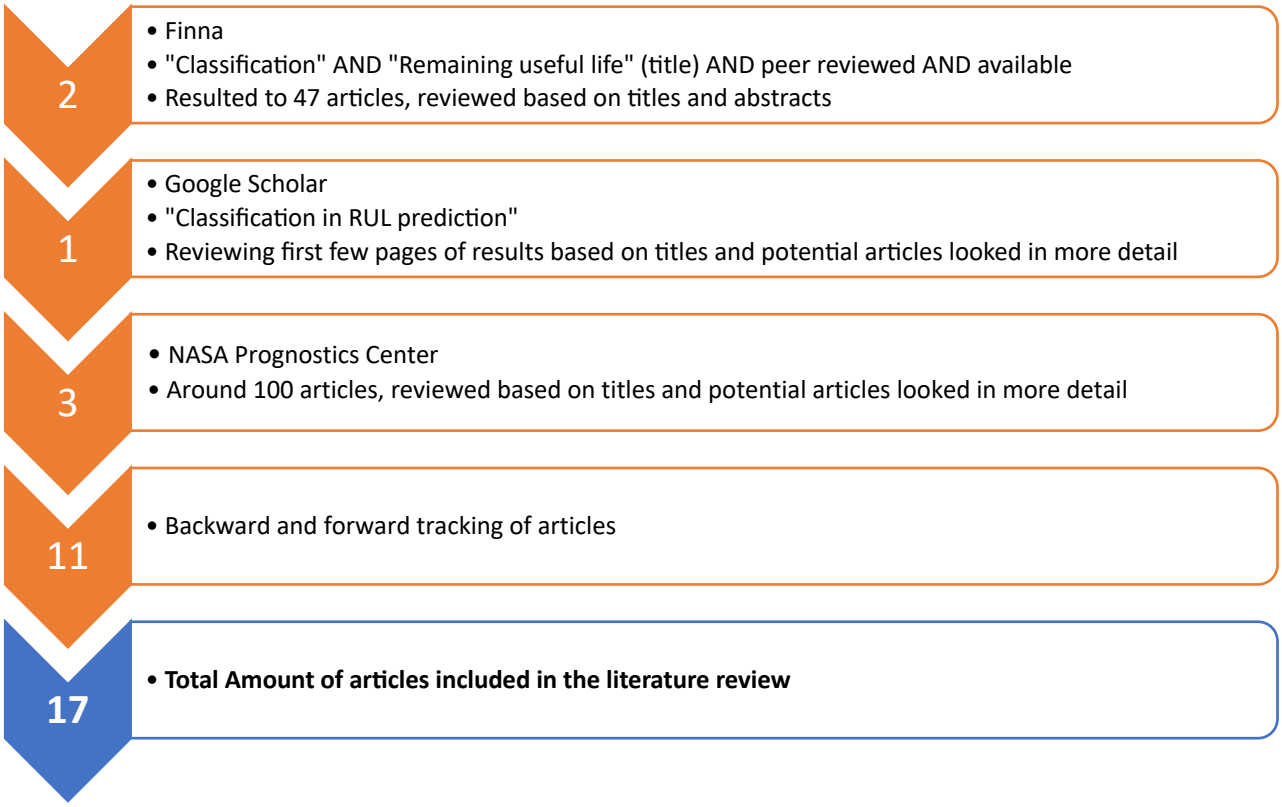


Figure 11. The Literature Search Process

Lastly, the NASA prognostics center's list of publications was reviewed. There is a total of around 100 articles related to prognostics, which of 68 are related to the turbofan dataset. Based on the title, the articles were reviewed, and potentially related articles were taken into further reviewing. Total of three articles were chosen from the NASA's list.

As suggested by Webster & Watson (2002), the articles were backward and forward tracked as well. This resulted into 11 new articles, making the total number of articles 17. Such a low number of articles is not desired but on the other hand the usage of classification in RUL prediction has clearly not been very popular method. Therefore, the literature review will have to be conducted with rather small amount of literature. The collection of literature could be considered to be representative sample of the related research, but not all-inclusive in any means.

An observation from the search process is that very few of the articles actually mention classification in title or abstract, even though classification is the only methodology used. This makes the literature gathering rather difficult as there really is not any keywords these articles have in common. Too broad searches with tens of thousands of matches are not

ideal as it is overly time consuming. Therefore, the more reasonable option was to find some articles and then backward and forward track them.

As the supply of literature was narrow, some of the selected articles' main focus is not solely in creating a classification framework for RUL prediction. Classification might have been used as a framework to investigate some other subjects, such as feature selection. On the other hand, most of the articles are not focused exactly on RUL prediction but are still handling with very similar type of problem, such as health state or fault prediction.

3.2 Review

The reviewing will be carried out by going through all of the selected articles. The contents, methods and relevant results will be discussed, and especially matters that can be used to help create the methodology for this study will be evaluated with particular consideration.

The literature review could be done as either author-centric or concept-centric (Webster & Watson, 2002). They argue that the review should be compiled as concept-centric as the concepts better determine the review's framework. However, in the case of this study, the literature cannot be divided in clear concepts and therefore using purely concept-centric approach would not provide any more clearness than a mixture of the two approaches.

As the number of articles is so scarce, every article will be reviewed independently. Table 3 is composed to give a quick overview of the selected articles. After reviewing all articles, the findings will be summarized in a concept matrix which allows to examine the literature more from the concept perspective.

Table 3 shows that only a few of the articles in principle focus on RUL prediction. There are different definitions for the problems such as failure, fault or maintenance need prediction. In practice, the objective can be viewed to be the same or at least very similar as these studies try to predict whether a failure will happen in a given amount of time. Phrases time interval and time frame will be used when discussing this amount of time.

Table 3. Summary of Reviewed Literature

Author(s), year	Objective	Used data
Letourneau, Famili & Matwin, 1999	Aircraft component replacement need prediction	Data of 34 aircrafts
Yang & Letourneau, 2005	Failure prediction with multiple classifier system	Train wheel data
Yang & Letourneau, 2009	Two-stage classification to estimate time-to-failure	Train wheel data
Georgescu, Berger, Willett, Azam & Ghoshal, 2010	Feature reduction	NASA's turbofan dataset, breast cancer dataset, ionosphere dataset
Kusiak & Li, 2011	Fault and fault category prediction with classification	Wind turbine data
Xue, Williams & Qiu, 2011	Creating a noise-label framework for classification models in fault prediction	Bearing dataset and NASA's turbofan dataset
Zaluski, Letourneau, Bird & Yang, 2011	Predicting the need of component replacement with classification	Aircraft component data
Zhao, Georgescu & Willett, 2011	Feature reduction	NASA's turbofan dataset
Bluvband, Porotsky & Tropper, 2014	Comparison of regression and classification models for critical zone prediction	NASA's turbofan dataset
Kauschke, Schweizer, Fiebrig & Janssen, 2014	Failure prediction with classification	DB Schenker Rail data
Li, Parikh, He, Qian, Li, Fang & Hampapur, 2014	Alarm and failure prediction with classification	Train-, maintenance-, weather- and schedule data
Fink, Zio & Weidmann, 2015	RUL prediction with classification	Discrete-event data of railway operations disruptions
Kauschke, Janssen & Schweizer, 2015	Failure prediction with classification	DB Schenker Rail system log data
Zhao, Al Iqbal, Bennett & Ji, 2016	Fault prediction with classification	Wind turbine data
Böhm, 2017	RUL prediction with classification	Railway switch engine data
Al Iqbal, Zhao, Ji & Bennett, 2018	Fault detection and prediction with classification	Wind turbine data
Allah Bukhsh, Saeed, Stipanovic & Doree, 2019	Maintenance need and type prediction	Railway switch engine data

The review is divided into two parts. The first part focuses on binary classification and the second part include the review on multi-class classification. This procedure allows the literature review to take a step closer into concept-centric approach.

3.2.1 Previous Research on Binary Classification

Letourneau, Famili & Matwin (1999) used classification technique to predict the need of component replacements in aircrafts. They viewed the problem as a binary classification task, replace or do not replace component. Data of 16 different components was used and the time interval was determined separately for each component, ranging between 10 and 40 units of time. They used KNN, C4.5 decision tree and naïve Bayes as algorithms and calculated own scoring function for evaluation. It was founded that none of the algorithms outperforms other algorithms for every component.

Yang & Letourneau (2005) implemented a multiple classifier system to predict train wheel failures using decision trees and naïve Bayes along with additional cost information. The multiple classifier system was created by first conducting base-level models for different features, which were used to create new training dataset. Then, meta-level models were made with the new training set. The meta-level model predicted 97% of failures with an 8% false alert rate. They did not find the algorithms to be much different in terms of performance.

Yang & Letourneau (2009) developed a two-stage classification system to predict train wheel failures. The first stage was a binary classification determining whether fault would occur and if positive, the second stage would predict when the fault occurs with a 4-class classification. As in their previous study, decision tree and naïve Bayes were used as algorithms. Regarding the binary classification, it was found out that 97% of failures could be predicted with an 8% false positive rate, and these results very similar compared to the previously obtained results.

Georgescu, Berger, Willett, Azam & Ghoshal (2010) and Zhao, Georgescu & Willett (2011) were investigating feature reduction for classification using NASA's turbofan dataset among other datasets. Support vector machine (SVM) and proximal support vector machine were the used algorithms and accuracy the used performance metric for both studies.

Regarding these two studies, the interest lies in the results obtained with the turbofan dataset as it is the only one related to predictive maintenance. The earlier study used a time

interval of 15 units of time before failure to create binary classes, and around 70% accuracy was reached (Georgescu et al., 2010). The later study used mean RUL as the time interval for binary classification and resulted in similar results with 70% accuracy (Zhao et al., 2011). Both studies discovered that better accuracy is achieved when less features were included in the models for the turbofan dataset and principal component analysis (PCA) performed as the best feature reduction algorithm.

Kusiak & Li (2011) performed binary classification to predict faults of wind turbines with artificial neural network (NN), NN ensemble, boosted tree algorithm (BTA) and SVM. Performance was measured with accuracy, true positive rate (TPR, also sensitivity or recall) and true negative rate (TNR, also specificity). NN ensemble was found to be the best one with 74% accuracy, 83% TPR and 65% TNR but the differences to regular NN were minor and partially mixed.

Xue, Williams & Qiu (2011) also performed fault prediction with binary classification. Their main objective was to build a framework for noisy labels, and this was demonstrated with bearing and turbofan datasets using different time intervals and portions of training data. Logistic regression (LR) was used to train models and area under curve (AUC) to evaluate them. They were able to obtain rather good results with their framework as the reported AUC values were over 0.85 for most of the combinations of time intervals and training data portions and some combinations were close to 1.

Zaluski, Letourneau, Bird & Yang (2011) predicted the need of component replacement for aircraft by the means of classification. They tested a total of 18 different algorithms for two different datasets and used two custom evaluation metrics, scoring function and problem detection rate, along with TPR and precision. The results show that none of the utilized algorithms generated accurate predictions in terms of TPR (5-87%) or precision (31-85%), while both never being on a high level at the same time.

Bluvband, Porotsky & Tropper (2014) compared classification and regression approach in critical zone prediction and the comparison was carried out by using the turbofan dataset. They utilized SVM and SVR (support vector regression) as algorithms and for classification time interval of 20 units of time before failure to separate the binary classes. Three custom scoring functions which emphasize early and late predictions were used to compare SVM and SVR models. It was found that classification outperforms regression in terms of these scoring functions.

Kauschke, Schweizer, Fiebrig & Janssen (2014) researched train component failure prediction by utilizing system log data. Binary classification was applied to predict failure of a single component and the experiments were produced for different timeframes. They used random forest (RF), sequential minimal optimization (SMO), JRip and J48 as algorithms and performance was measured with AUC, F1-score, precision and TPR. The result showed that RF produces the best outcomes, even if the data is highly imbalanced. However, by addressing the imbalance by increasing the timeframe, better results can be achieved.

Li, Parikh, He, Qian, Li, Fang & Hampapur (2014) applied binary classification to predict failure alarms and train component faults. The alarm prediction for two different time intervals was done with SVM and DT as a benchmark while TPR and FPR were used to measure performance. It was found that SVM performs better than DT. However, they focused on minimizing the FPR on the expense of TPR. The portion of false positives was very close to zero but then only 45,4% of the true positives could be identified. The fault prediction was carried out with only DT with focus on rule simplification and user interpretability which lead to TPR of 97% and FPR of 0,23%.

To the best knowledge, Fink et al. (2015) are one of the few who actually approached the problem especially as remaining useful life prediction and then reshaped it into a classification problem. They predicted the RUL of train components using binary classification and extreme learning machine (ELM) and neural network as algorithms. The performance of the models was measured by TPR, TNR and misclassification rate (MR). ELM outperformed NN by far with this data, and NN performed almost as a random classifier with about 50% MR.

Kauschke, Janssen & Schweizer (2015) utilized binary classification to predict train failures. Random forest, JRip and Bayesian network were the used algorithms with different time intervals and feature selection algorithms. AUC, accuracy, TPR and precision were the employed performance metrics. RF turned out to be performing the best out the tested algorithms.

Zhao, Al Iqbal, Bennett & Ji (2016) and Al Iqbal, Zhao, Ji & Bennett (2018) proposed a soft label binary classification framework to wind predict wind turbine faults. The soft labeling was implemented to deal with uncertainty in the label information. Zhao et al. (2016) used SVM and k-nearest neighbors (KNN) with different loss functions. SVM with the proposed soft labeling technique performed the best. Al Iqbal et al. (2018) added RF along with SVM

and KNN, and it was found that RF outperformed the other algorithms. Both studies utilized AUC to measure performance.

Allah Bukhsh, Saeed, Stipanovic & Doree (2019) applied binary classification to predict maintenance need of railway switches. They used tree-based algorithms DT, RF and gradient boosted tree. Performance was measured using accuracy, MR, F1-score and kappa. Gradient boosted tree slightly outperformed the other two algorithms in terms of every metric, but the differences were rather marginal.

3.2.2 Previous Research on Multi-class Classification

Yang & Letourneau (2009) as discussed in the previous subchapter, included a 4-class classification as the second stage of their model for train wheel fault prediction. Similar to the first stage binary classification, decision tree and naïve Bayes were used as algorithms. The multi-class classification was executed both individually and as part of two-stage classification. Comparison with the binary model shows that the problem detection rate was significantly lower and the false alert rate higher for the multi-class model. However, the two-stage classifier resulted in lower false alert rate than either binary or multi-class model but was the worst one to detect problems. Based on these results, it can be stated that binary model seems to perform better than multi-class model but including two stages into the model do not provide improvement in the results.

Zhao et al. (2011) also experimented multi-class classification along with binary model. As discussed earlier, they worked with the turbofan dataset and created three different 4-class classification systems. It is notable that the lowest thresholds for all three systems are quite high, 75, 125 and 150, meaning that the first class includes all the observations with RUL lower than these.

To consider these from PdM point of view, it is reasonable to question whether it provides any useful information about the remaining useful life and maintenance need. On the other hand, Zhao et al. (2011) were not focusing on PdM rather feature selection. Regardless, it was found that the performance increases when the lowest threshold is increased. Additionally, fewer number of features mainly resulted in better results.

Böhm (2017) performed prediction of railway switch engine remaining useful life with multi-class classification. Two classification systems were constructed and the time intervals to

separate classes differed significantly to any previous study. Geometric sequence (2^x) was used to determine the thresholds and this resulted into 14 classes, the smallest class containing observations with less than 1,5 hours to failure and largest class containing observations with over 256 days to failure. The other system's class boundaries were manually picked, including 10 classes.

Böhm (2017) used NN, SVM, DT and KNN algorithms to create models. Matthews correlation coefficient (MCC) was used as a single value metric and confusion matrix as a more comprehensive metric to measure performance of the models. It was found that SVM with radial basis function kernel worked the best as MCC values were close to one for both class formations. The confusion matrices showed that the smaller classes were more difficult to get right compared to the larger classes.

3.3 Summary of Literature Review

The literature review provided answers for the first research question *"How previous research has approached RUL-type problems with classification?"* and for the subquestions under it. Table 4 represents a concept matrix which summarizes the main findings.

It can be seen from Table 4 that both binary and multi-class classification have been utilized when approaching RUL-type problems. However, the binary approach has been much more popular as only three out of 17 reviewed studies applied multi-class classification. Therefore, it looks like multi-class approach is more in need of further research.

Binary classification is arguable the simpler, and probably also the more robust way to solve the problem. On the other hand, the multi-class approach provides more informative knowledge about the remaining useful life but as said, the complexity increases which presumably affects the performance.

Other observation regarding the first research question is that the used time intervals varied a lot between studies. Some studies had rather short time intervals, and some had very large. Obviously, it is dependent on the data and there is no correct time interval to choose that would be ideal for each RUL prediction case. The time intervals need to be determined by considering the real-life applications so that maintenance actions can be made in proper time.

Author(s), year	Algorithm							Classification		Evaluation Metric							
	SVM	NN	KNN	DT	Bayes	RF	Other	Binary	N-Class	ACC	MR	TPR	TNR	FPR	Precision	AUC	Other
Letourneau, Famili & Matwin, 1999			X	X	X			X									Scoring function
Yang & Letorneau, 2005				X	X			X						X			Scoring function, Fault detection rate
Yang & Letourneau, 2009				X	X			X	4					X			Scoring function, Fault detection rate
Georgescu, Berger, Willett, Azam & Ghoshal, 2010	X							X		X							
Kusiak & Li, 2011	X	X					NN Ensemble, Boosted Tree	X		X		X	X				
Xue, Williams & Qiu, 2011							LR	X								X	
Zaluski, Letourneau, Bird & Yang, 2011				X	X		Many algorithms	X				X			X		Scoring function, Fault detection rate
Zhao, Georgescu & Willett, 2011	X							X	4	X							
Bluvband, Porotsky & Tropper, 2014	X							X		X							Scoring functions
Kauschke, Schweizer, Fiebrig & Janssen, 2014						X	SMO, Jrip, J48	X				X			X	X	F1-score
Li, Parikh, He, Qian, Li, Fang & Hampapur, 2014	X			X				X				X		X			
Fink, Zio & Weidmann, 2015		X					ELM	X			X	X	X				
Kauschke, Janssen & Schweizer, 2015					X	X		X		X		X			X	X	
Zhao, Al Iqbal, Bennett & Ji, 2016	X		X					X								X	
Böhm, 2017	X	X	X*	X*					14 & 10								MCC, Confusion Matrix Visualization
Al Iqbal, Zhao, Ji & Bennett, 2018	X		X			X		X								X	
Allah Bukhsh, Saeed, Stipanovic & Doree, 2019				X		X	Gradient Boosted Tree	X		X	X						Kappa

*Results not reported																
	SVM	Support Vector Machine		ELM	Extreme Learning Machine			ACC	Accuracy		TNR	True Negative Rate		MCC	Matthews Correlation	
	NN	(Artificial) Neural Network		Bayes	Bayesian Network			MR	Misclassification Rate		FPR	False Positive Rate			Coefficient	
	KNN	K Nearest Neighbours		LR	Logistic Regression			TPR	True Positive Rate		AUC	Area Under Curve				
	DT	Decision Tree		RF	Regression Forest											

Table 4. Concept matrix

The time interval selection also affects the class balance. This is especially true for binary classification as if the time interval is short, it means there are lot less observations in that class. The imbalance of classes might cause some trouble, and therefore it can be said there is at least some sort of trade-off.

Table 4 also provides an answer to the subquestion *"Which algorithms have been used most in RUL classification?"*. Wide range of different algorithms have been used. Support vector machine (SVM) and decision tree (DT) are among the most used algorithms and those have been used quite steadily during the whole time period covered in the review. It is also noticeable that the usage of random forest (RF) has increased during the recent years.

Based on the received results in the literature, a conclusion can be made that no algorithm is superior to others. The results for different algorithms have proved to fluctuate a lot between and even within studies. Different algorithms suit better for different data, in other words, the algorithms are data specific and cannot be generalized too broadly.

Finally, the answer for the second subquestion *"How the performance of classification models has been evaluated?"* can as well be found from Table 4. Accuracy, true positive rate (TPR) and area under curve (AUC) have been popular choices to measure performance. Additionally, other simple metrics based on confusion matrix such as TNR, FPR, MR or precision have been used quite a lot. These are quite straightforward and easy to use and interpret which is probably why these have been extensively selected even though they might have some weaknesses.

In some studies, own scoring functions have been calculated or more complex confusion matrix -based metrics such as F1-score and Matthews correlation coefficient (MCC) have been applied to measure performance. Using more advanced metrics might provide better understanding about the performance but can be more difficult to interpret in real-life cases.

4 Methodology

This chapter will introduce the methodology that will be used in the empirical part of the study. Figure 12 visualize the process of creating a ML model. First, the data used to illustrate the methodology has to be selected and gathered.

Secondly, the problem setting itself needs to be addressed. The continuous RUL prediction needs to be converted into a form of classification problem. Then, the data probably requires some pre-processing before it can used to train the model.

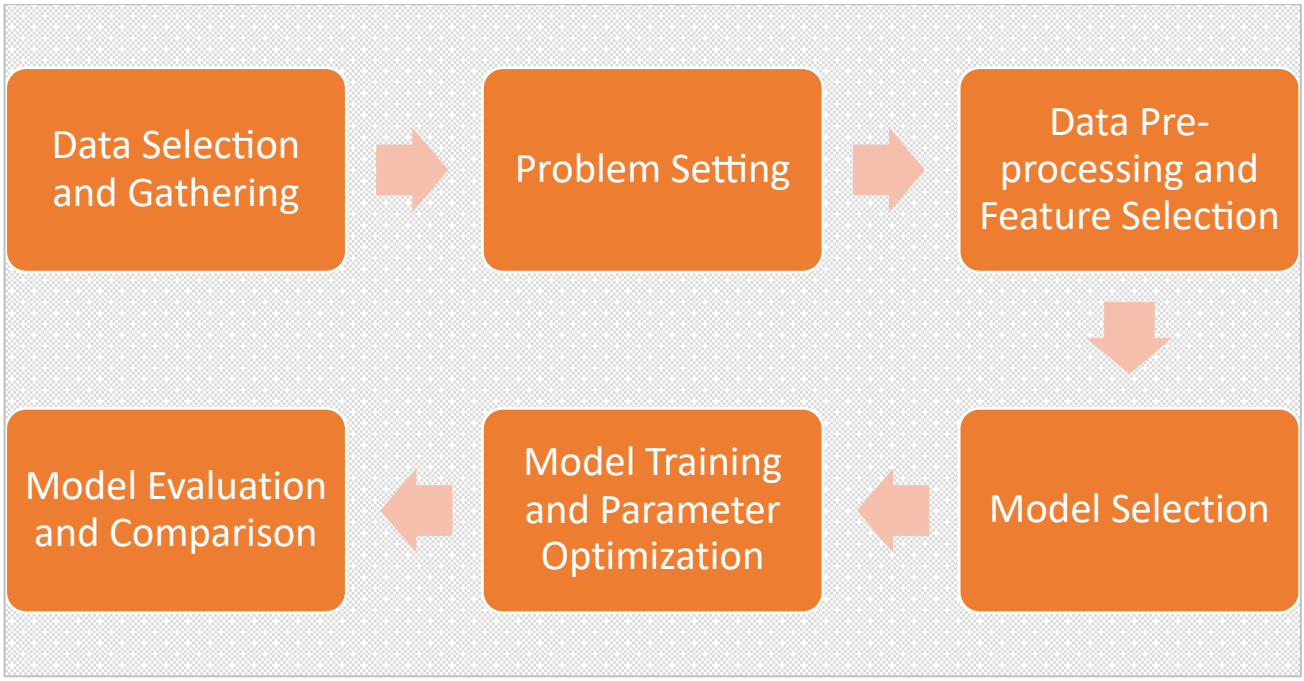


Figure 12. Model Creation Process

Additionally, the selection of classification models used will be discussed. After all these preparations, the models can be trained and validated. Finally, evaluation can be done, and models compared with each other. The results will be presented and discussed in chapter 5.

4.1 Data

NASA's Turbofan Engine Degradation Simulation Dataset was selected because the dataset had good amount of observations and it was easy to access. The dataset is publicly available on The Prognostics Data Repository, a dataset collection which focuses only on prognostics data (NASA Prognostics Center, 2020).

The simulation of turbofan engine degradation has been carried out by using Commercial Modular Aero-Propulsion System Simulation (C-MAPSS), which is a turbofan engine simulation environment. The dataset consists of four independent subsets, that were each simulated under unequal combinations of conditions and fault modes. (Saxena & Goebel, 2008)

The descriptions of the four independent datasets are presented in Table 5. As mentioned, the combination of conditions and fault modes is different between each set. Also, the number of simulated units is different, thus leading to different number of observations in each subset.

Table 5. Description of the Turbofan Dataset

Data subset	Number of Train Units	Number of Test Units	Conditions	Fault Modes
FD001	100	100	ONE (Sea Level)	HPC Degradation
FD002	260	259	SIX	HPC Degradation
FD003	100	100	ONE (Sea Level)	HPC Degradation, Fan Degradation
FD004	248	249	SIX	HPC Degradation, Fan Degradation

The structure of the dataset is shown in Table 6. Each data subset consists of particular number of units representing individual engines, and a time series of cycles is provided for each unit. Thus, the data can be considered to represent a fleet of similar type engines. At the start of the time series, every unit starts with different, unknown degree of initial usage and variation of manufacturing. However, the usage and variation are normal and not considered to be a fault condition. (Saxena & Goebel, 2008)

At the start of each time series, the units are operating normally, and the faults develop over time. The difference between train and test sets is that the train sets have the complete time series until failure, whereas time series of test sets end before failure happens and there is still remaining useful life left. The test sets also include the true RUL values (bolded in Table 6) at the last included cycle. (Saxena & Goebel, 2008)

Additionally, the time series contain three operational setting values and 21 sensors values. The operational settings illustrate the conditions and affect the performance of an engine significantly. The sensor values contain measurement values from actual sensors in an engine and the fault should be recognized from these values. (Saxena & Goebel, 2008)

Table 6. The Structure of the Turbofan Data (Dataset FD001)

FD001 train											
Unit	Cycle	Setting 1	Setting 2	Setting 3	Sensor 1	Sensor 2	Sensor 3	...	Sensor 20	Sensor 21	RUL
1	1	-0,0007	-0,0004	100	518,67	641,82	1589,7	...	39,06	23,419	191
1	2	0,0019	-0,0003	100	518,67	642,15	1591,82	...	39	23,4236	190
1	3	-0,0043	0,0003	100	518,67	642,35	1587,99	...	38,95	23,3442	189
...	...	...	...	...	...	...	...	...	...	...	...
1	191	0	-0,0004	100	518,67	643,34	1602,36	...	38,45	23,1295	1
1	192	0,0009	0	100	518,67	643,54	1601,41	...	38,48	22,9649	0
2	1	-0,0018	0,0006	100	518,67	641,89	1583,84	...	38,94	23,4585	286
2	2	0,0043	-0,0003	100	518,67	641,82	1587,05	...	39,06	23,4085	285
2	3	0,0018	0,0003	100	518,67	641,55	1588,32	...	39,11	23,425	284
...	...	...	...	...	...	...	...	...	...	...	...
2	286	-0,001	-0,0003	100	518,67	643,44	1603,63	...	38,33	23,0169	1
2	287	-0,0005	0,0006	100	518,67	643,85	1608,5	...	38,43	23,0848	0
100	1	-0,0033	0,0003	100	518,67	642,25	1596,57	...	38,72	23,3899	199
100	2	0,001	-0,0004	100	518,67	642,37	1589,43	...	39,18	23,246	198
100	3	0,0024	-0,0004	100	518,67	643,13	1588,55	...	38,95	23,2034	197
...	...	...	...	...	...	...	...	...	...	...	...
100	199	-0,0011	0,0003	100	518,67	643,23	1605,26	...	38,29	23,064	1
100	200	-0,0032	-0,0005	100	518,67	643,85	1600,38	...	38,37	23,0522	0
FD001 Test											
Unit	Cycle	Setting 1	Setting 2	Setting 3	Sensor 1	Sensor 2	Sensor 3	...	Sensor 20	Sensor 21	RUL
1	1	0,0023	0,0003	100	518,67	643,02	1585,29	...	38,86	23,3735	142
1	2	-0,0027	-0,0003	100	518,67	641,71	1588,45	...	39,02	23,3916	141
1	3	0,0003	0,0001	100	518,67	642,46	1586,94	...	39,08	23,4166	140
...	...	...	...	...	...	...	...	...	...	...	...
1	30	-0,0025	0,0004	100	518,67	642,79	1585,72	...	39,09	23,4069	113
1	31	-0,0006	0,0004	100	518,67	642,58	1581,22	...	38,81	23,3552	**112**
2	1	-0,0009	0,0004	100	518,67	642,66	1589,3	...	39	23,3923	146
2	2	-0,0011	0,0002	100	518,67	642,51	1588,43	...	38,84	23,2902	145
2	3	0,0002	0,0003	100	518,67	642,58	1595,6	...	39,02	23,4064	144
...	...	...	...	...	...	...	...	...	...	...	...
2	48	0,0011	-0,0001	100	518,67	642,64	1587,71	...	38,99	23,2918	99
2	49	0,0018	-0,0001	100	518,67	642,55	1586,59	...	38,81	23,2618	**98**
100	1	0,0014	0,0003	100	518,67	641,65	1591,5	...	39,01	23,3087	217
100	2	0,0031	0,0001	100	518,67	642,2	1588,99	...	38,97	23,351	216
100	3	0	0,0001	100	518,67	642,27	1587,47	...	39,14	23,3636	215
...	...	...	...	...	...	...	...	...	...	...	...
100	197	-0,0038	0,0001	100	518,67	643,26	1594,99	...	38,66	23,2699	21
100	198	0,0013	0,0003	100	518,67	642,95	1601,62	...	38,7	23,1855	**20**

The RUL column on the right-hand side was not provided with the dataset hence it is calculated based on the cycle. For train sets, the last cycle gets RUL of zero as the failure occurs during that cycle. Then, RUL is raised by one for each cycle until the beginning of time series is reached. For test sets, the RUL was provided for the last given cycle. Then similarly to the train set treatment, it was raised by one until the first observation.

4.2 Classification Framework

As already made clear earlier, this study takes a rather rarely used approach for RUL prediction in form of classification. Predicting remaining useful life can be rather challenging task, especially if uncertainty is involved in measurement data and the prediction horizon is long compared to the unit of time (Böhm, 2017). It can be stated that at least the prediction horizon in the case of this study is rather long for the most parts as can be observed from Figure 13. Majority of the observations have RUL over 100 cycles.

In addition, it is not too informative to know the exact RUL when it is high. Classification approach can address this issue if the class structure is defined accordingly. Therefore, a well-defined classification model could provide near the same amount of useful information as a regression model would. However, it must be addressed that converting the problem into classification form will cause losing the time series feature from the data set.

In order to use classification methods for RUL prediction, the continuous problem has to be first converted into discrete problem. That said, the RUL has to be divided into intervals covering certain time periods. However, defining the intervals can be challenging.

The best approach for defining the intervals should be considered to be based on the necessary lead time to predict failures (Böhm, 2017). After all, the reason to predict RUL in the first place is to know when to take maintenance actions before it is too late.

If considering turbofans, the maintenance actions involve at least workforce and spare parts. Furthermore, some spare parts might require ordering if those are not being held at stock. This would have to be considered in the required lead time as time needed before maintenance could be performed gets longer. Also, the maintenance means that the turbofan in question cannot be used. Hence, scheduling of maintenance is important as it is assumed that having the whole fleet maintained simultaneously is not desirable.

A modifiable binary classification model like suggested by Fink et al. (2015) could be considered if the lead time is known. Nevertheless, it could be considered slightly uncertain and less informative as it does not tell whether the selected threshold is close or not and therefore might require rather quick actions.

On the other hand, it must be addressed that the approach which requires modifying would not be practical. In most real applications, it is expected that predictive models are trained for some time and then put into practice whereupon the training will stop. The models might be retrained occasionally, but constant, continuing training is not usually the case, except for some special applications. Thus, the approach which requires a lot of continuous modifying would not fit the needs of most practical applications.

The multiclass framework proposed by Böhm (2017) should be considered more informative, thus not requiring the changing of time intervals. Even when using a multiclass model, the information about lead time should be considered and the intervals set in accordance with those. But as mentioned earlier, the purpose of this study is not to investigate the required lead time to perform maintenance on turbofan engines, rather the focus is on showcasing the classification approach to the RUL problem.

Thus, the intervals in this study will not take into account the actual required lead time. However, three different multiclass interval settings will be formed, and it is reasonable to assume that at least one of these should include close to realistic lead times. Nevertheless, none of the three formed classification systems cannot be justified from practical standpoint. The first two systems are based purely on mathematics which will be discussed more shortly. The last system will be formed manually with focus on creating some distinction to the other two systems, thus no precise practical aspect is considered.

Before defining the intervals, the distribution of RUL should be examined. The distribution of whole dataset divided into train and test sets can be seen in Figure 13. Both the train sets combined, and the test sets combined contain the time series of 708 units. As the train set consisted of time series until the failure, there are fixed number (708) of observations for RUL from zero to 127, and after that the number of observations per RUL value begins to decline rather rapidly.

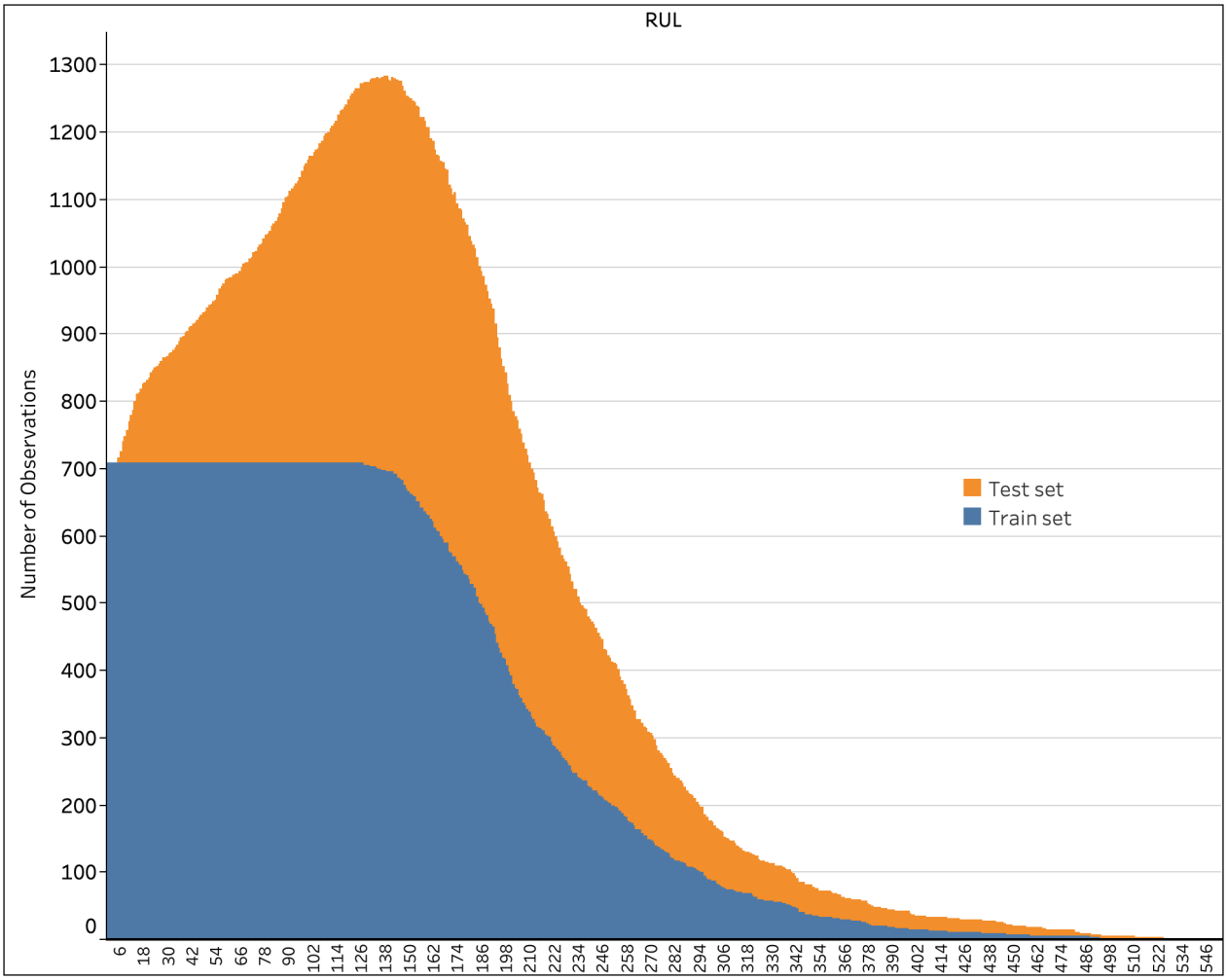


Figure 13. Distribution of RUL Values for Train and Test Sets

The time series in test sets were cut before failure occurred and thus there are less observations with small RUL values. The smallest value is six and it appears only eight times. The number of smaller values (RUL under 50) is significantly lower than the number of most common values (around RUL of 100-200). That could be problematic when converted into classes if the given train and test sets would be preserved as they are because train and test sets would be completely differently distributed.

The options would be to use only the train set and divide it into again into train and test set, which would keep the class distribution more even. Another option is to combine train and test sets together, and then randomly split them into train and test sets. The latter option seems more reasonable because in reality, it is probable that the new data introduced for the model is more similar to the current test set than to the train set. Another reason why

both sets should be included is that the overall number of data points the model can learn from is much larger. If test set was left out, the data would be 104897 instances smaller whereas with test set, it consists of 265256 observations.

It is reasonable to assume that the desired intervals are higher resolution for small RUL values and lower resolution for long RUL values. This meaning that it is more important to know whether the RUL is 5 or 15 than it is to know whether the RUL is 150 or 200. Here, the assumption is that the time needed to react is on the lower side of the RUL distribution, probably for most applications something between one and 50 units of time, depending from the unit of time. Therefore, the considered intervals should be shorter near zero and become longer when RUL increases.

Böhm (2017) suggests a geometric sequence (2^x) to represent intervals, which is easy to implement, mathematically precise and fits well to the requirement of better resolution for small RUL values. This will be used as way to create the intervals. The classes will be formed with geometric sequence as follows:

$$RUL\ class = \begin{cases} 1: 0 & \leq\ RUL\ \leq\ 2^0 \\ i: 2^{i-2} & <\ RUL\ \leq\ 2^{i-1} \\ 10: 2^8 & <\ RUL \end{cases} \quad (4)$$

$$, where\ i = 2, 3, 4, \dots, 9$$

With this class division, there are total of 10 classes. The numbers of observations in each class are shown in Figure 14. The number of observations in the first classes is very small compared to the latter classes, thus the data is highly unbalanced, which was supposed.

Another classification system will be constructed so the performances can be compared. The second system should be a little simpler at the beginning when RUL values are small, but at the same time still have somewhat shorter intervals for smaller values. Then, the classes with small RUL values should not be as highly unbalanced as the previous classification system created, and it should still have the ability to focus on smaller RUL values. Again, the geometric sequence can be utilized with the modification to the first interval bound:

$$RUL\ class = \begin{cases} 1: 0 & \leq\ RUL\ \leq\ 2^0 \times 10 \\ i: 2^{i-2} \times 10 & <\ RUL\ \leq\ 2^{i-1} \times 10 \\ 6: 2^4 \times 10 & <\ RUL \end{cases} \quad (5)$$

$$, where\ i = 2, 3, 4, 5$$

This kind of class formation leads into six classes. The class distribution is visualized in Figure 14 along with the first system's distribution. Like said, the second classification system has slightly more equal classes. Of course, there still is huge difference in class size between the first two and the last two classes, but compared to the differences in system 1, it is much more acceptable.

One more classification system will be created in order to reduce the number of classes even further. The last system is not based on any clear sequence. It is constructed by simply selecting thresholds so that the lowest interval is wider compared to system 2 and the next few classes would still be relevantly informative. The system 3 has four classes which are determined in the following way:

$$RUL\ class = \begin{cases} 1\colon\ 0 \leq RUL \leq 20 \\ 2\colon\ 20 < RUL \leq 60 \\ 3\colon\ 60 < RUL \leq 100 \\ 4\colon\ 100 < RUL \end{cases} \quad (6)$$

As the interval of the smallest class was lengthened, the number of observations in that class obviously increases which can be seen from Figure 14. The time intervals of the next two classes are equal length but rather small so that some information regarding the maintenance need is provided. This leads to the last class being very large both in terms of time and number of observations.

The classes could be shaped so that all classes would be close to equal. However, in order to do so, the first classes would have to cover longer interval, which would lead to weakening resolution. For example, it could be assumed to be important to know whether there is 10 or 30 days (or cycles) before failure occurs. On the other hand, the classes with high RUL could be adjusted to shorter intervals. This would lead to a lot higher number of classes and those classes not providing too much information as in practice it should not be too relevant to know whether the RUL is 300 or 400. Therefore, these three classification systems are used and investigated in this study.

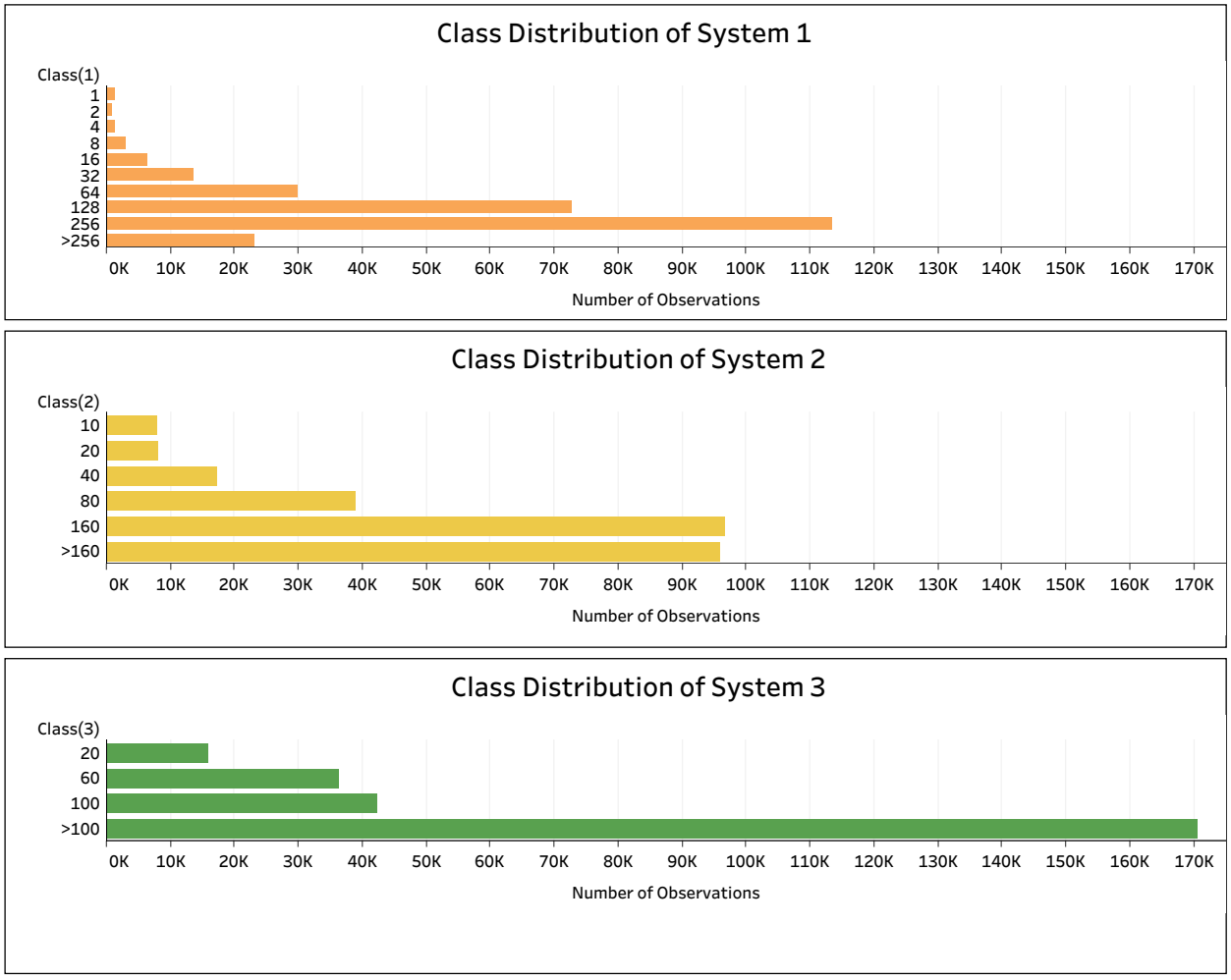


Figure 14. Class Distributions

Supposedly, system 2 should be able to perform better than system 1. That is because system 2 has lesser number of classes and the distributions are better. Furthermore, it is expected that system 3 will perform better than system 2, and even though it has one class being much more unequal, the sensor values should be relatively constant at higher RUL values. It remains to be seen whether these presumptions are correct when the models are created and compared. And if so, it is also interesting to see how much differences there are in terms of performance.

4.3 Data Pre-processing and Feature Selection

The next step is to perform data pre-processing. Figure 15 illustrates the data treatment done in this study. The dataset was complete in respect to no missing data. Thus, no further actions were required to deal with any incomplete instances.

A portion of the pre-processing is already done. First, the RUL variable was calculated. Like mentioned before, for train set this was done using the cycle variable by reversing it. For test set, the provided RUL for the last cycle was used to generate RUL's for anterior cycles. Then, three classification system variables were created by using the RUL variable. This process has been already explained in more detail in the previous subchapter.

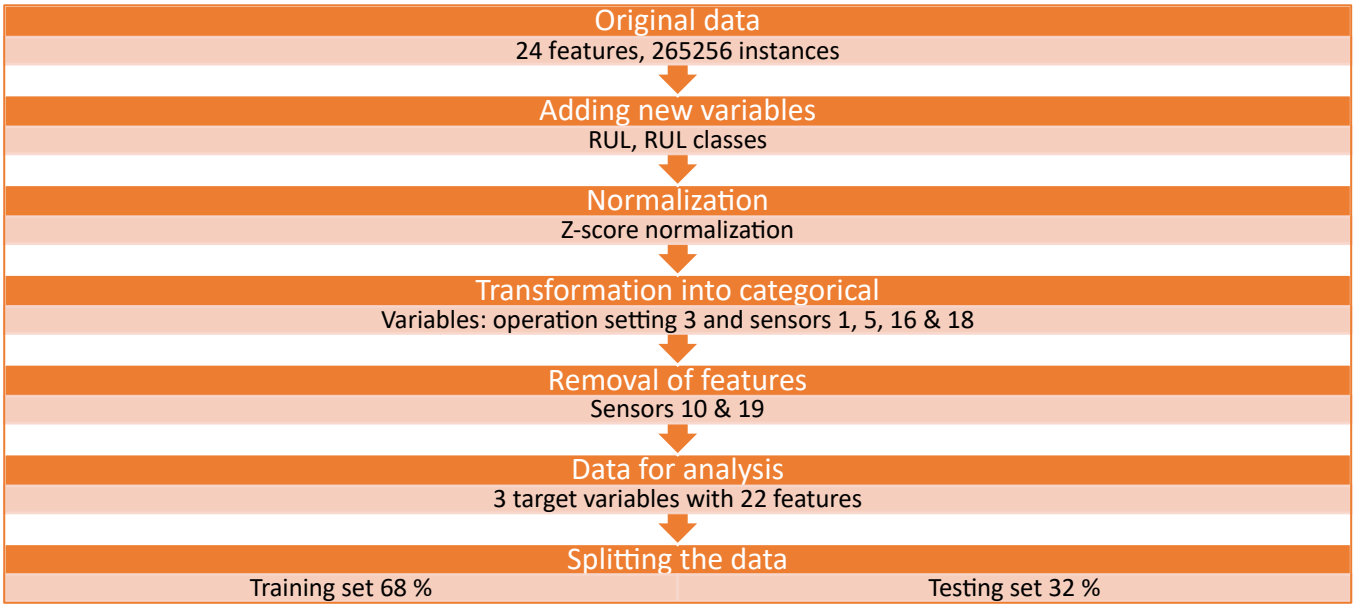


Figure 15. Data Treatment Process

The range of values for different features were on quite diverse scales as could be noticed earlier on Table 6. Therefore, data normalization is recommended in order to avoid possible problems with biased variable importance due to high absolute differences in values. Normalization changes the structure so that the relative differences remain the same, but the values lie on a smaller scale.

The normalization is performed by using the so-called z-score normalization. The z-score normalization calculates new values by subtracting the column mean from the original value and then dividing the remainder by column standard deviation.

Technically, all of the provided features were continuous and therefore normalization was applied to all features. However, closer looks at features showed that some of them were more categorical than continuous as each of them did have only 2-6 different values across all observations.

Considering the fairly large number of observations, it could be assumed that these values are the only ones that can ever appear. However, this can turn out to be quite naive expectation if other values are actually possible in reality. On the other hand, the algorithms were not able to handle these variables as continuous and therefore total of five features were turned into categorical variables. These features were operational setting 3 and sensors 1, 5, 16 and 18.

Finally, two features were cut out due to similarity and limited amount of values. Sensor 19 was perfectly correlated with operational setting 3 and thus was not providing any new information. Sensor 10 had similar problems with continuity as the few other features. 21 different values existed across the observations hence making it closer to a continuous variable than the previous ones but again, the algorithms could not handle it as continuous. Therefore, it was left out as it seemed more probable that a value outside these 21 values could occur in future which would then be unknown to the model.

The processed data consists of 22 features and three dependent variables which represent the different classification systems. Some further feature reduction could have been done but as the number of features is rather low anyway, it was decided to keep the rest of the features to get as much information as possible. Fewer number of features could be more efficient in terms of computational requirements and in practice could be the better option if it would lead to similar results. Nevertheless, this study will take advantage of all the usable features.

As the last pre-processing step, the data will be divided into training and testing sets. As all of the data is used and the pre-determined datasets have different combinations of operating conditions and fault modes, certain amount of observations will be taken from each dataset for training and testing sets.

The division will not be done as random sampling because testing set is desired to be consisting of full unit data. This approach aims to create the testing set so that it would represent data of new engines that the model has not encountered before. Also, it keeps class distribution somewhat similar between training and testing sets as both have full RUL cycles of units.

The division is executed by selecting the first 70 units of FD001 and FD003, and the first 170 units of FD002 and FD004 for the training set and the rest for the testing set. This leads approximately to 68/32 % -split.

4.4 Classification Algorithms

The problem type has now been changed as the continuous RUL is converted into discrete classes and the faced problem is now a multiclass classification task. Like discussed in chapter 2, supervised methods can be used because the data set is labeled. Various different classification techniques exist which can be used to solve a classification problem.

In essence, there is no generally best algorithm to solve a problem, which is also known as the “No Free Lunch” -theorem. This holds true even for algorithms that are generally suitable. A technique’s ability to solve a certain problem can be heavily dependent about the used algorithm and parameters, thus the configuration might determine the success of a technique for a given problem. (Duda, Hart, & Stork, 2012, 454-458)

Many different algorithms could be included and see how they perform with the data used in this study. However, as just discussed, some algorithms could be very dependent of parameters, thus adequate testing would be computationally exhaustive and time consuming. Due to this, only two different algorithms will be considered.

Fernandez-Delgado, Cernadas & Barro (2014) performed extensive testing with 179 algorithms from 17 different families and found that RF’s and SVM’s are the top two families while the different versions of NN’s also performed well. Even though there is no real reason why these techniques should work the best with this data, they should be considered to be tested out as they generally seem to perform well.

SVM is computationally quite heavy technique to use, especially when non-linear kernels are considered. Additionally, SVM’s are much more prone to parameters compared to some other algorithms (Probst, Boulesteix, & Bischl, 2019a). Hence, experimentations with SVM are not included in this study due to lack of sufficient computing power.

The models will therefore be created using RF and NN. The computations are done using R and there are various packages which can be used. For RF model, *randomForest* -function of the package with the same name is selected. There exist a lot of options for NN model, but rather basic feed-forward NN is used. The model will be created with *h2o* -package’s *deeplearning* -function as it allows to connect to an open source external parallel computing system which makes the computations more efficient.

4.5 Evaluation Metrics

The evaluation metrics were introduced and discussed in more detail earlier in chapter 2. Böhm (2017) put together a good set of properties a metric should have for this kind of problem, and these properties should largely hold true for this dataset as well:

- The metric desirably would be applicable to both binary and multi-class cases. Even though it is not crucial for this particular study, for future comparisons it might be a useful feature to have.
- It should be invariable to class distributions as there is no certainty about future class distributions.
- The metric should be able to tolerate noise in class memberships as the intervals are manually designed and not absolute truths.
- The classes with smaller number of observations should be treated equally to classes with higher number of observations.
- The metric should also be able to distinguish small changes in performance.

Apparently, there is no perfect metric that would tackle all of these issues. Thus, MCC will be used as it can be extended for both binary and multi-class cases, it handles unbalanced class distributions and noise well, and also express smaller changes in confusion matrix rather well. It is not completely invariable against changing class distributions but is less sensitive compared to most of the other metrics. (Jurman & Furlanello, 2010)

MCC gives a single number to compare the model performance. However, the interpretation of MCC for real world is rather difficult and hence simple visualizations of confusion matrices will be used to see how different classes are treated. The confusion matrix approach will only be used for the final model evaluation, not for parameter optimization.

The hyperparameter optimization will be performed with training data and 5-fold cross-validation. Then, the final evaluation will be performed by using the whole training set to train the models, and testing set will be used to evaluate the performance.

5 Results

This chapter will present the results found out in this study. First, the issue of hyperparameter optimization will be addressed. After tuning the parameters, the final models will be created, and the results will be displayed for each classification system and both chosen algorithms.

5.1 Hyperparameter Optimization

The simplest way to train the models would be by just using the default hyperparameters. However, it is expected that using these presets would lead to inferior performance compared to well optimized parameters. At least it is useful to test how the different hyperparameters affect the performance and see if better performance can be achieved.

The two chosen classification algorithms both provide an opportunity to change multiple hyperparameters. Hyperparameter optimization can be computationally a rather heavy task, especially if done systematically. One way to perform it would be using ready-made grid search algorithms to find the optimal combination of hyperparameters. However, in the scope of this study, there exists a problem with this grid search approach. As MCC will be used as the only numerical evaluation metric, it would be desirable to use it to optimize the hyperparameters as well. The grid search algorithms usually use some simpler metric such as error to determine the best hyperparameter combination.

Another issue with grid searches is that those simply provide the founded best hyperparameters. Therefore, comparisons cannot be made, and the user needs to adapt the offered hyperparameters without better knowledge. Thus, it is decided to do the grid search manually even though it might be more time consuming and require some compromises in number of optimized hyperparameters.

The hyperparameter optimization will be performed with the training data and validated with 5-fold cross-validation. Each classification system will be considered individually for both algorithms which means total of six optimizations. MCC will be used as the metric to compare different parameter combinations.

5.1.1 Random Forest

Random forest algorithm is not as dependent from hyperparameters as it is able to generate good results with default settings (Fernandez-Delgado et al., 2014). Probst et al. (2019a) argue that RF is far less tunable compared to some other algorithms such as SVM in terms of achieving better performance.

The studies of Probst et al. (2019a) and Probst, Wright & Boulesteix (2019b) show that the hyperparameter tuning of RF lead to only marginal performance improvements. Regardless of that, some testing will be done to see whether it holds true for this specific data.

Random forest has several hyperparameters that can be tuned. These parameters are mtry, nodesize, sample size, replacement, number of trees and splitting rule (Probst et al., 2019b). This study will only consider tuning of nodesize and number of trees while the other parameters will be kept at default values. Nodesize defines the number of observations in the terminal node and the meaning of number of trees is quite self-evident.

The number of trees should rather large, but the optimal number depends on the properties of the data (Probst et al., 2019b). Oshiro, Perez & Baranauskas (2012) concluded with large number of datasets that 100 trees is usually sufficient number of trees in order to achieve good performance. Therefore, 100 trees will be used as the minimum number of trees and larger numbers will be experimented to see if any improvements can be obtained. Number of trees from 100 to 500 with interval of 100 trees were chosen to be used.

The default value for nodesize is usually 1 and Probst et al. (2019a) suggest that it generally results in good performance but in some cases increasing it might be optimal. To find out how it affects the results with this particular data, few different, increasing values will be experimented. Nodesizes 1, 100, 1000 and 10000 will be used as the number of observations is quite large.

Figures 16-18 display visualizations of average MCC's of cross-validation for random forest systems 1-3 with different parameter combinations. Appendix 1 provides the exact average values for every system and each combination along with the average standard deviations within number of trees.

Figure 16 shows and Appendix 1 confirms that the difference between number of trees is almost meaningless as MCC changes only in the third decimal. The nodesize on the other hand is much more significant. There is a wide difference between nodesizes 10000 and 1,

although it was expected. However, the difference between 1 and 100 is very small, only in the third decimal. The MCC of system 1 hovers around 0,4 with nodesizes 1 and 100.

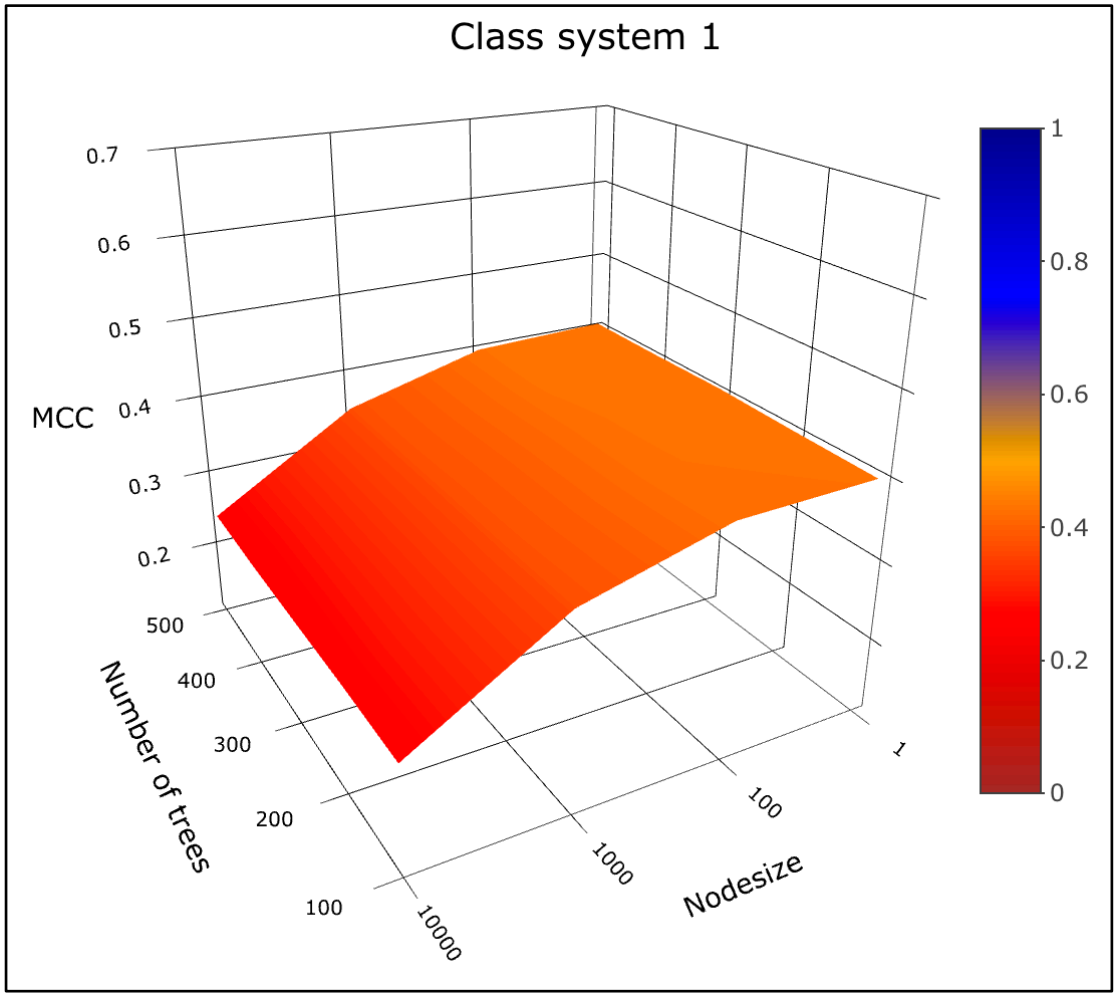


Figure 16. Average MCC's of RF System 1 with Different Nodesizes and Number of Trees

From Figure 17 it can be noted that the overall performance of system 2 seem to be on a slightly higher level with MCC around 0,45 compared to system 1. Otherwise, the same observations can be done as number of trees does not have an effect on the performance and different nodesizes influence similarly to system 1.

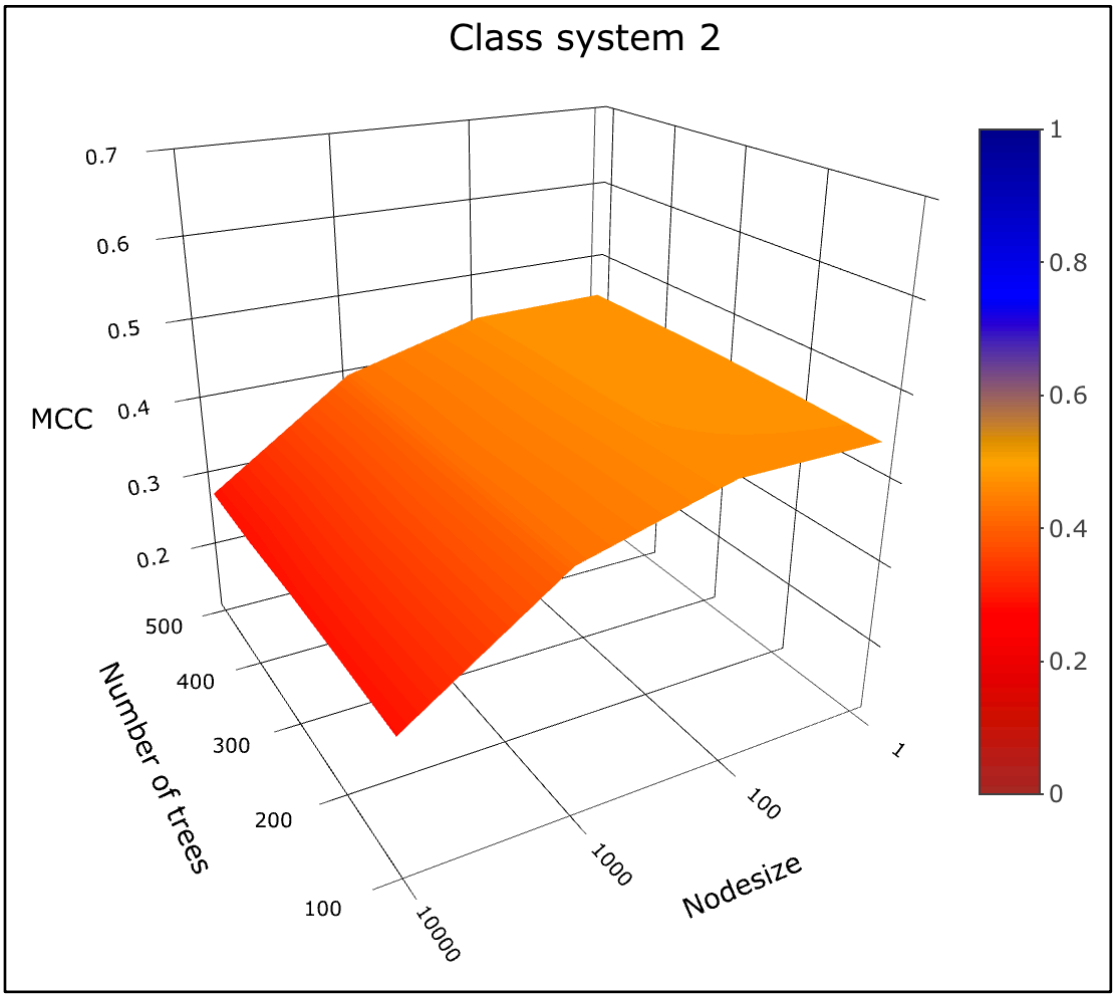


Figure 17. Average MCC's of RF System 2 with Different Nodesizes and Number of Trees

Figure 18 indicates that the similar conclusions can also be extended to class system 3. The MCC with nodesizes 1 and 100 are close to 0,6 which is much higher than the other two systems.

Some additional random experiments with the other hyperparameters were also done to see if those would have had more impact. Any significantly different results were not found and due to the computational limits, any further systematic experimentations were consequently not done. The low average standard deviations imply that the differences between cross-validation folds were low, which indicates that the model was working steadily regardless of the fold.

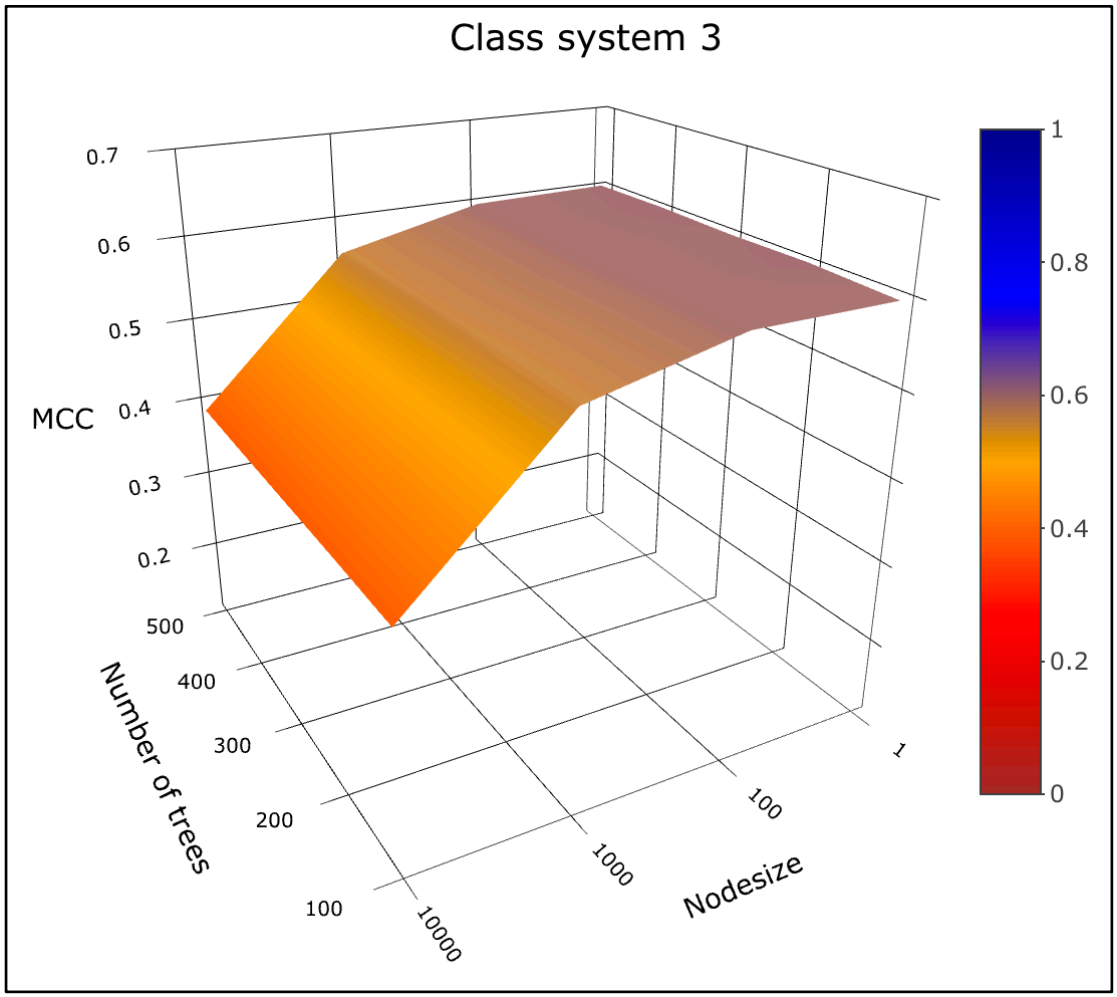


Figure 18. Average MCC's of RF System 3 with Different Nodesizes and Number of Trees

Based on these findings, the final models will be trained with nodesize of 1. As seen, the number of trees did not have huge effect and therefore it should not matter too much which number is chosen. Nevertheless, because this testing was done, it could might as well be used to choose the number of trees which resulted in best MCC, even though the differences are marginal. This means 300 trees for systems 1 and 2, and 400 trees for system 3, if the lowest maximums are chosen from Appendix 1.

The second research question *"What random forest and neural network parameters lead to the best performance for the turbofan dataset?"* can now be answered for the part of random forest. The answer is not perfectly complete due to the issues discussed but it can be said that smaller nodesize seems to lead to better results and the number of trees does not have consistent best value, given that the other used parameters are set to default values.

5.1.2 Neural Network

Neural networks have much more hyperparameters that can be tuned. For example, the *deeplearning*-function of h2o-package which is used in this study to train the NN has tens of different parameters, although not all them are relevant for the model created in this study. Regardless, there are way too many options to systematically investigate in the context of this study and automatic grid search algorithms are not ideal as discussed earlier.

Even though comprehensive parameter tuning cannot be done, it still good to do some testing like in case of RF. The way to perform parameter optimization for NN is adopted from Böhm (2017), thus it is chosen to try different combinations of number of hidden layers and neurons per layer. The meaning of these parameters easy to understand as it is clear to understand what their role is in the method. Number of layers is kept rather small as only 1-3 layers are tested while the neurons per layer will vary between 10 and 200, increasing by 10.

Like in the case of RF, Figures 19-21 display visualizations of average MCC's of cross-validation for neural network systems 1-3 with different parameter combinations. Appendix 2 provides the exact average values for every system and each combination along with the average standard deviations within the number of hidden layers.

Figure 19 shows that system 1 has no huge differences between different combinations of hidden layers and number of neurons. The average MCC is around 0,35 for each tested combination. The exact numbers of Appendix 2 show that NN with 3 layers perform slightly better as the it has highest minimum and maximum MCC's.

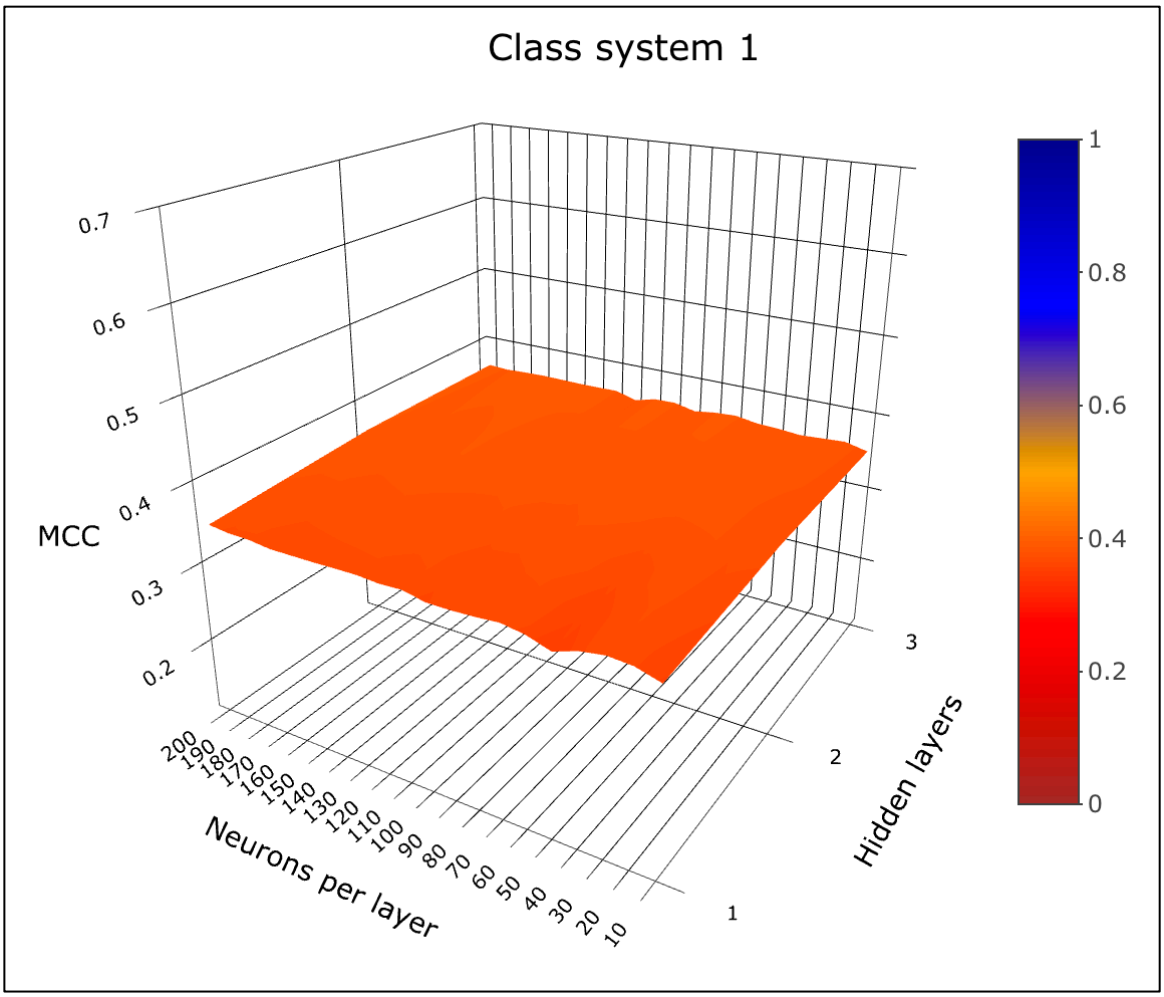


Figure 19. Average MCC's of NN System 1 with Different Number of Hidden Layers and Neurons per Layer

Also, system 2 has very flat area displayed in Figure 20. The surface is at MCC around 0,4 which is little higher than system 1. The same effect was observed with RF as well. Again, quick look at Appendix 2 shows that three layers slightly outperform the lesser number of layers, but the difference is very small.

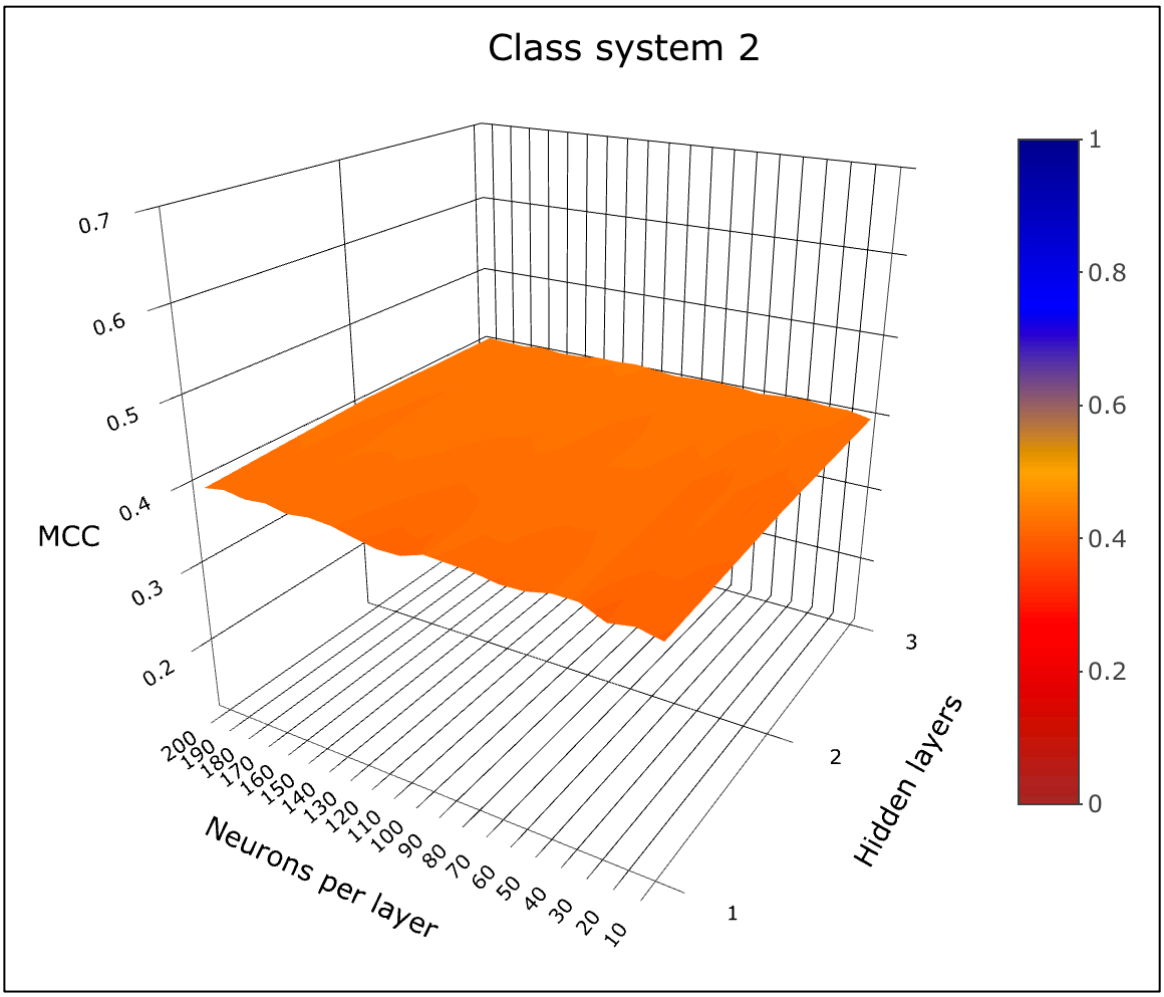


Figure 20. Average MCC's of NN System 2 with Different Number of Hidden Layers and Neurons per Layer

Figure 21 shows the results for system 3. It undergoes a substantial increase in performance compared to the other two systems as the again quite flat surface is located between MCC's of 0,55 and 0,6. The differences the first two systems had between number of layers has now almost completely disappeared. Two layers provide the maximum MCC, even though it should not matter much as the difference is so small.

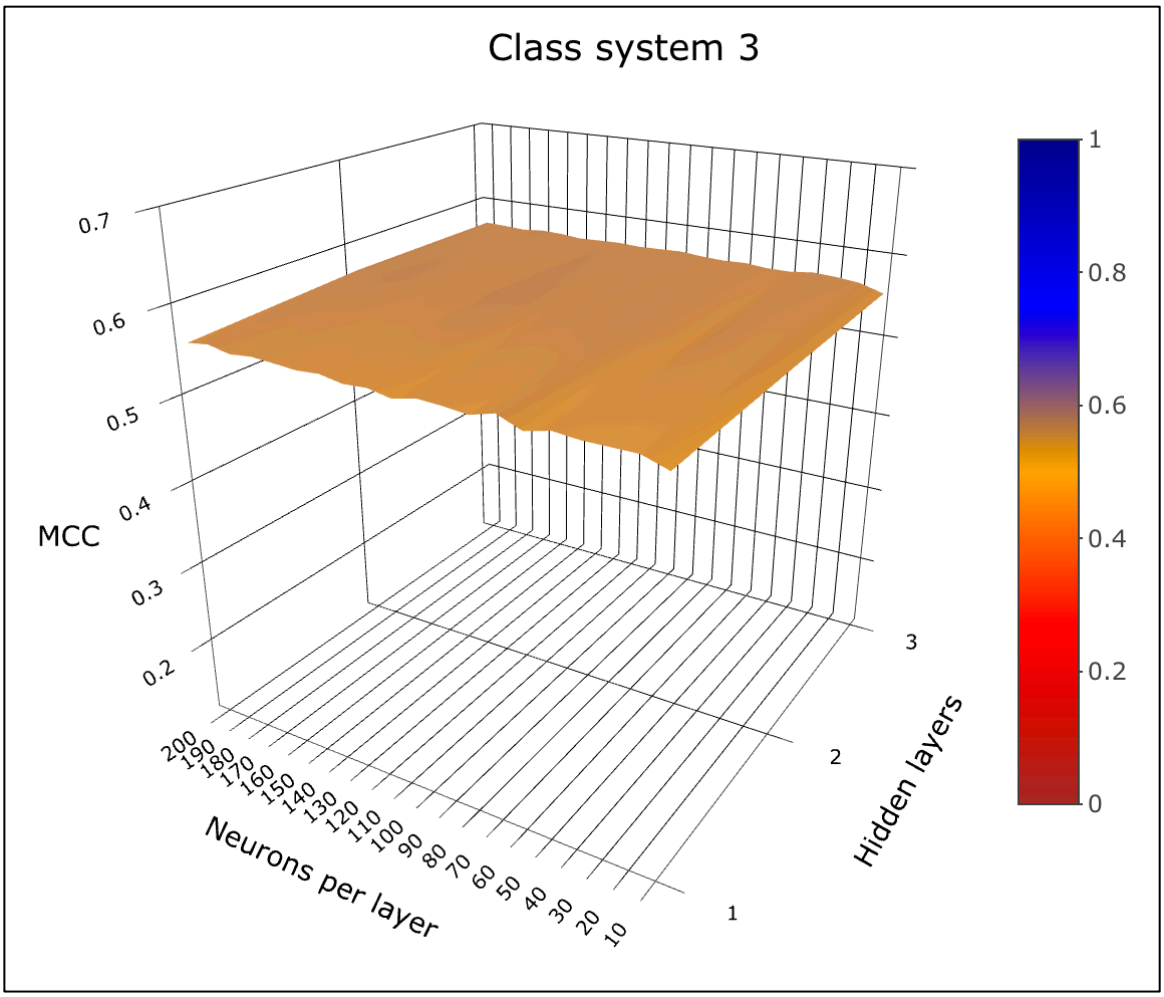


Figure 21. Average MCC's of NN System 3 with Different Number of Hidden Layers and Neurons per Layer

As there are so many different parameters to be changed with NN, even any random experimentations with other parameters were not done. This could be interesting topic to investigate more, but it is so large topic that it should considered to focus solely on that. As with RF, the average standards deviations were low, which indicates that the model was working steadily regardless of the fold.

These findings suggest that systems 1 and 2 should be trained with three hidden layers and in the case of system 3 this parameter is not too important but the maximum MCC was found with two layers which will be used. There is some variance of MCC within the layers with different number of neurons and each system will be trained with the number of neurons which performed the best. Therefore, based on Appendix 2, 130, 120 and 130 neurons per

layer are chosen respectively to systems 1, 2 and 3 to keep the number of neurons similar for all models.

The second research question *"What random forest and neural network parameters lead to the best performance for the turbofan dataset?"* can be addressed now by the part of neural network as well. Similarly to RF, no clear answer can be given as further testing should be done in order to discover more meaningful answers. For the more complicated class systems, 3 three hidden layers provided slightly better results while the number of neurons was very specific to the system.

5.2 Final Model Evaluation

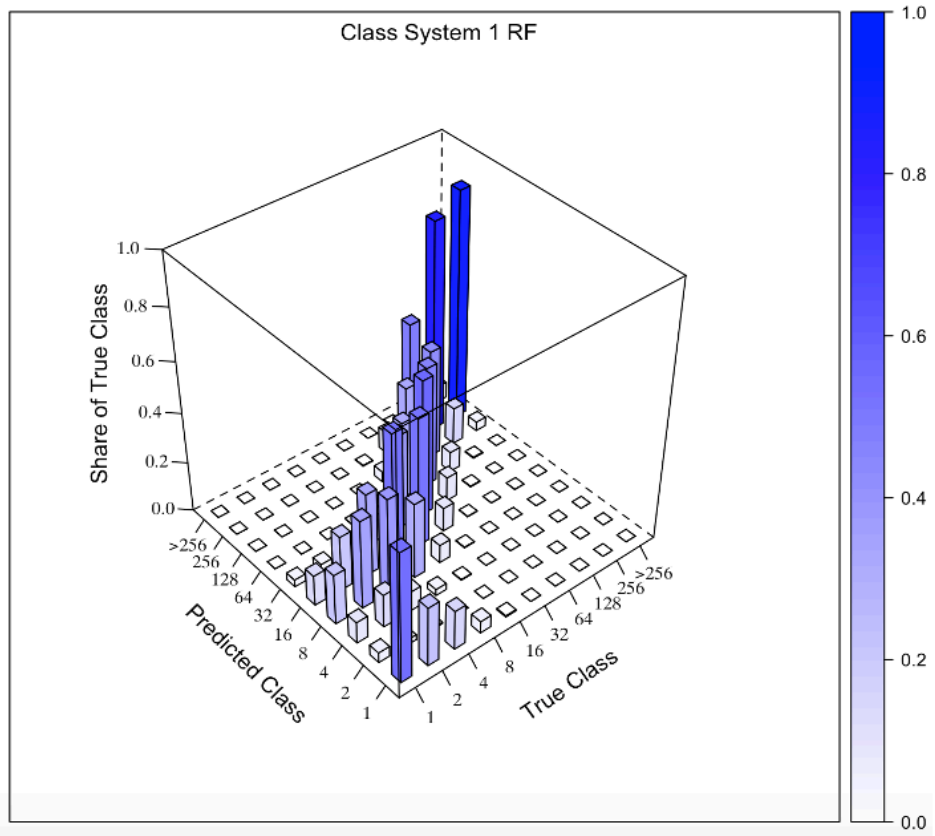
The results of final models are presented and discussed in this subchapter. The models are trained with the training set which includes 179661 observations and the evaluated with the testing set which consists of 85595 observations. After reviewing the final models, the last two research questions will be answered.

As discussed, the final models will be evaluated and compared with the MCC's and additionally, confusion matrices with appropriate visualizations will be reported to further investigate the models. The confusion matrix values are presented as ratios to the real class. The absolute numbers can be found from Appendix 3.

5.2.1 Final evaluation of Random Forest Models

The confusion matrix with visualization for RF with class system 1 is presented in Figure 22. The confusion matrix shows the share of true class each node has. The MCC for this model is 0,391, which is at the same level than it was when parameters were optimized. The MCC itself reveals that the model is not performing very well as it is quite far from the best value 1.

The confusion matrix shows that there is a lot of deviations between the predicted and actual classes. The perfect case would have ones on the diagonal, thus meaning that each class was predicted correctly. Interestingly, around half of the observations of the smallest class are predicted correctly, whereas respectively, only 2,1% and 7,9% of the next two classes were predicted correctly. This could indicate that the sensor values during the last cycle are clearly different and identifiable from the sensor values with few more cycles of lifetime left.



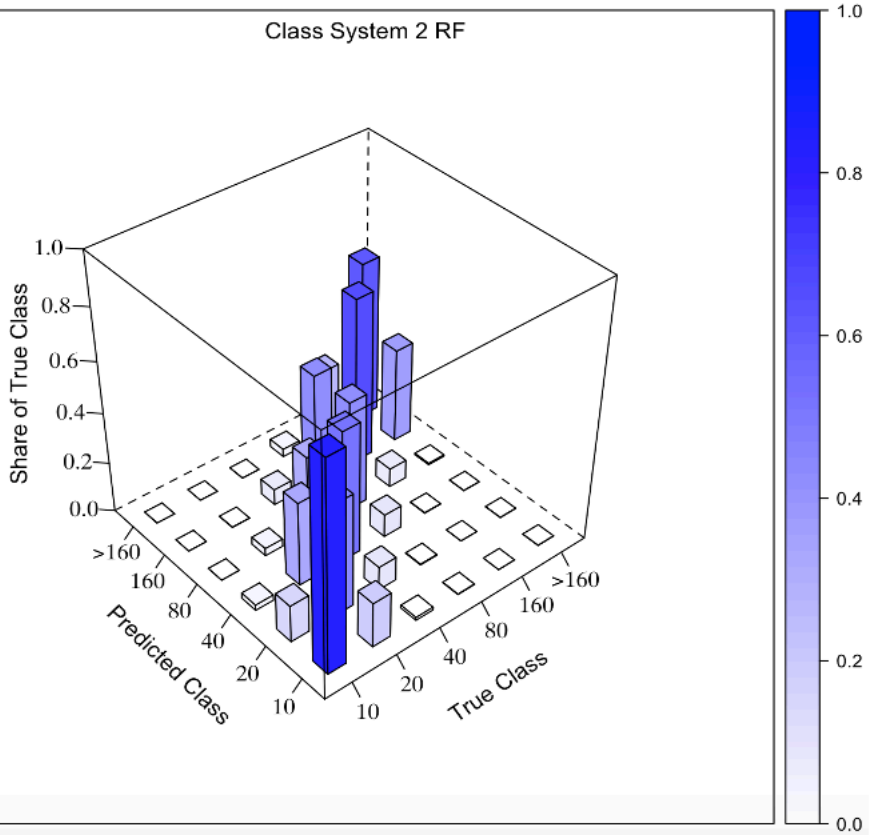
MCC = 0,391		Predicted Class									
		1	2	4	8	16	32	64	128	256	>256
True Class	1	0,521	0,041	0,086	0,202	0,122	0,028	0	0	0	0
	2	0,24	0,021	0,133	0,356	0,219	0,03	0	0	0	0
	4	0,159	0,006	0,079	0,382	0,33	0,043	0	0	0	0
	8	0,048	0,001	0,025	0,303	0,511	0,11	0,002	0	0	0
	16	0,005	0	0,003	0,073	0,514	0,38	0,025	0	0	0
	32	0	0	0	0,003	0,096	0,543	0,312	0,043	0,003	0
	64	0	0	0	0	0,002	0,097	0,478	0,336	0,087	0
	128	0	0	0	0	0,001	0,003	0,081	0,432	0,481	0,002
	256	0	0	0	0	0,001	0,001	0,004	0,148	0,829	0,017
	>256	0	0	0	0	0	0	0	0,037	0,899	0,064

Figure 22. Confusion Matrix of RF System 1 with Visualization

Another observation can be made regarding the deviations between classes. The smaller classes have more clearly spread distribution compared to the higher classes. Especially, the first three true classes have significant portions of predictions to five different classes, and the as can be seen from the visualization, the bars are low. The bars seem to grow, and the predictions are made to fewer classes as the classes get higher. This is somewhat expected as the intervals are increasing, meaning that number of observations per class increase. Also, the sensor values are expected to be more constant as the units are considered to be healthier the higher the class is.

Regardless, the share of correct predictions per class is not very high in general. Only the 256-class has relatively high true positive rate with 82,9 %. The last class should have definitely been included in the previous class as almost 90 % of the observations were predicted to belong to that.

Figure 23 provides the same information about RF system 2. The MCC of 0,429 for this model is also approximately on the same level as it was for for the parameter optimization trials. The MCC is only slightly higher than it was for RF system 1 but the confusion matrix looks much better. There are not as much deviations as there was with system 1 and the shares of correct predictions are on a sligthly higher level.



MCC = 0,429		Predicted Class					
		10	20	40	80	160	>160
True Class	10	0,829	0,149	0,022	0	0	0
	20	0,181	0,451	0,336	0,03	0,001	0
	40	0,012	0,095	0,52	0,309	0,063	0
	80	0,002	0,003	0,091	0,435	0,438	0,031
	160	0	0	0,002	0,075	0,652	0,27
	>160	0,001	0,001	0,001	0,006	0,372	0,619

Figure 23. Confusion Matrix of RF System 2 with Visualization

The most important take is the 82,9 % true positive rate for the first class which is much better than in the case of the larger system. After all, the smallest class is the most interesting class in order to take maintenance actions before failure. Therefore, the rate of correct classifications should desirably be as high as possible. The model still misclassifies 17,1 % of the first class, meaning that a lot of observations near a failure would have not been recognised. On the other hand, the amount of misclassified early predictions is not extremely high, only 18,1 % of true 20-class observations and very low for other classes, hence meaning that too early maintenance actions would not lead to wasted remaining useful life.

Aside from the first class, the model distinctly struggles to separate the other classes from their neighbouring classes. The true positive rate for the other classes vary between 44-65 %, while the neighbouring classes practically add up to the rest.

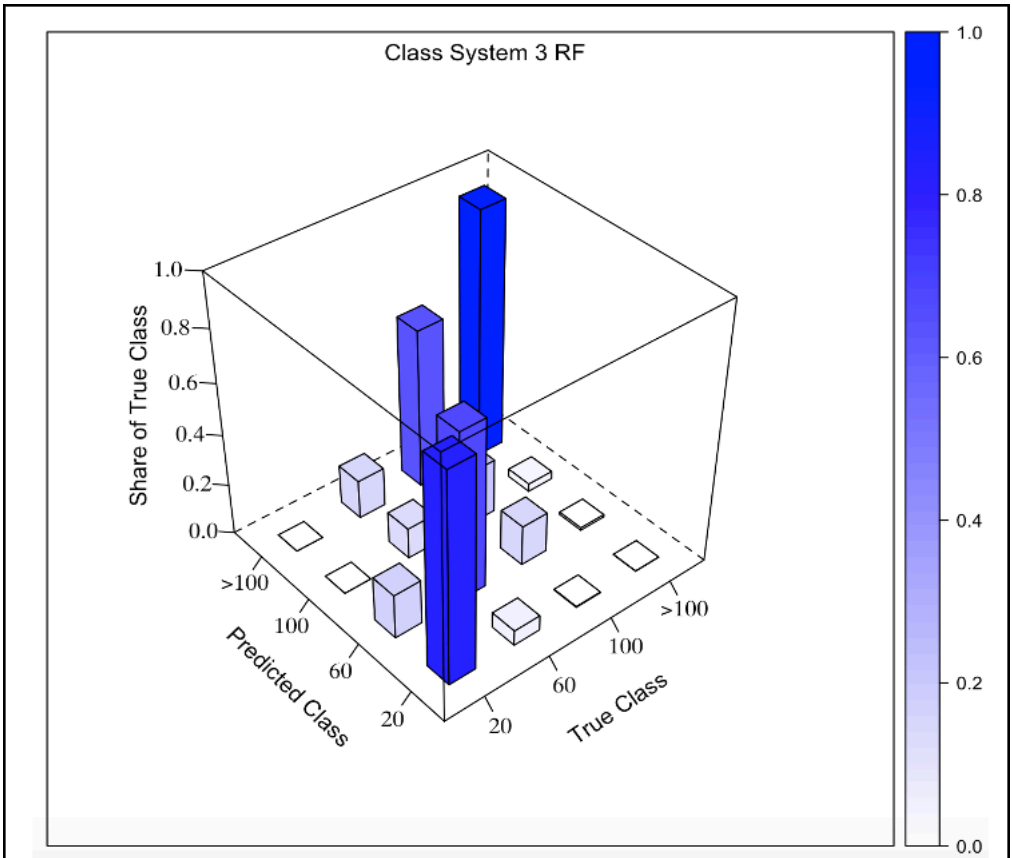
The confusion matrix of the final RF model with classification system 3 is presented in Figure 24. As for the previous models, the MCC of 0,59 is on similar level than for the parameter tuning experimentations. Furthermore, the MCC of RF system 3 is undeniably the highest so far thus implying dominance over the other two RF models. Reducing the number of classes undoutebly improved the overall performance, which was expected due to reduced number of classes.

Even so, the true positive rate of the first class is slightly lower than it was for RF system 2. This is interesting and somewhat confusing, as the interval was increased from 10 to 20 RUL. One explanation could be that for most instances, the sensors would still be giving rather normal values closer RUL of 20. Therefore, it is possible that the model would not be able to recognize as precisely the difference for example between RUL's of 15 and 30 cycles as well as it could for example the difference between RUL's of 8 and 15 cycles.

Further testing would be required to find the optimal threshold where the model could find the difference better. However, pushing the first threshold even further away from failure might not lead to any improvements as similar issue could arise at other thresholds as well. Also, pushing the threshold further would reduce the benefits from the maintenance point of view.

The model misclassifies 17,6 % of the true first class observations which again is quite a lot. However, the system 3 model produces far less early predictions for the first class compared to the previous model.

The model has difficulties with the 100-class which is mostly misclassified to belong into the last class. Removing the 100-class could make the model better, or it might lead to the 60-class being misclassified into the last class. Thus, keeping the 100-class might be a good idea as it is probably not as important for maintenance actions and scheduling.



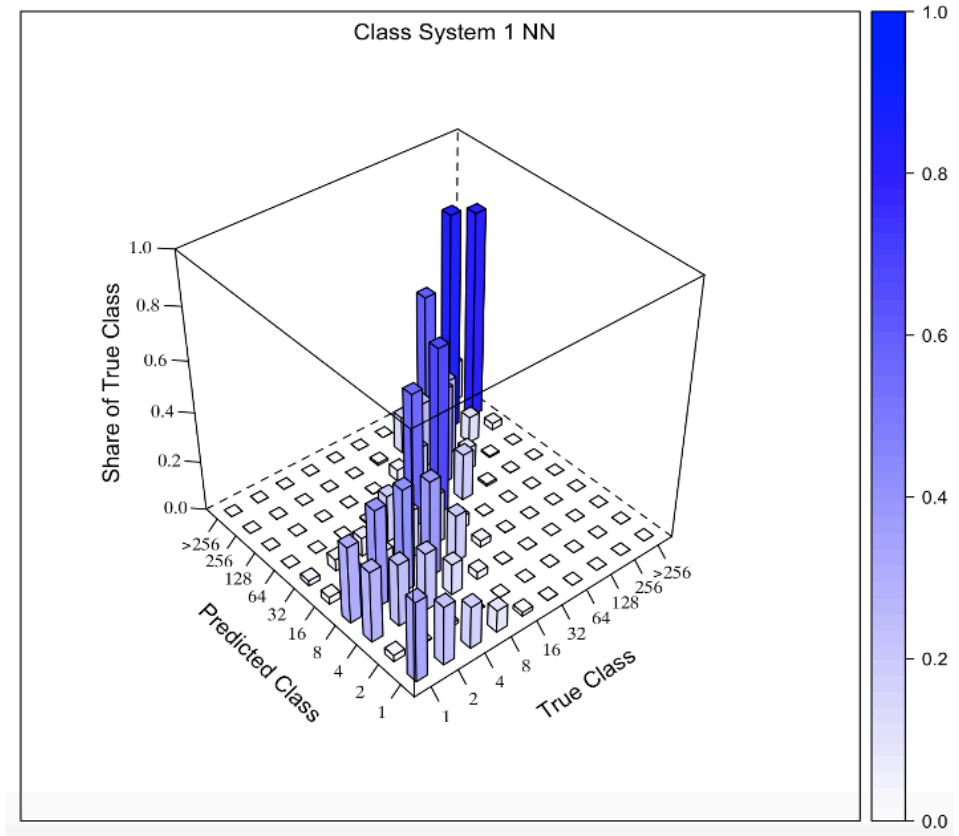
MCC = 0,590		Predicted Class			
		20	60	100	>100
True Class	20	0,824	0,176	0	0
	60	0,06	0,666	0,122	0,153
	100	0,003	0,159	0,214	0,624
	>100	0,001	0,01	0,031	0,958

Figure 24. Confusion Matrix of RF System 3 with Visualization

Based on both the MCC's and confusion matrices, a conclusion that RF system 3 outperforms the other two models fairly easily can be made. The remaining research questions will be answered after exploring the final results of NN models as well.

5.2.2 Final evaluation of Neural Network Models

The confusion matrix with visualization for NN with class system 1 is presented in Figure 25. The 0,354 MCC is approximately on the same level as it was for the parameter optimization models. It is slightly lower than it was for the RF system 1 model, which was the case for the parameter optimization results as well.

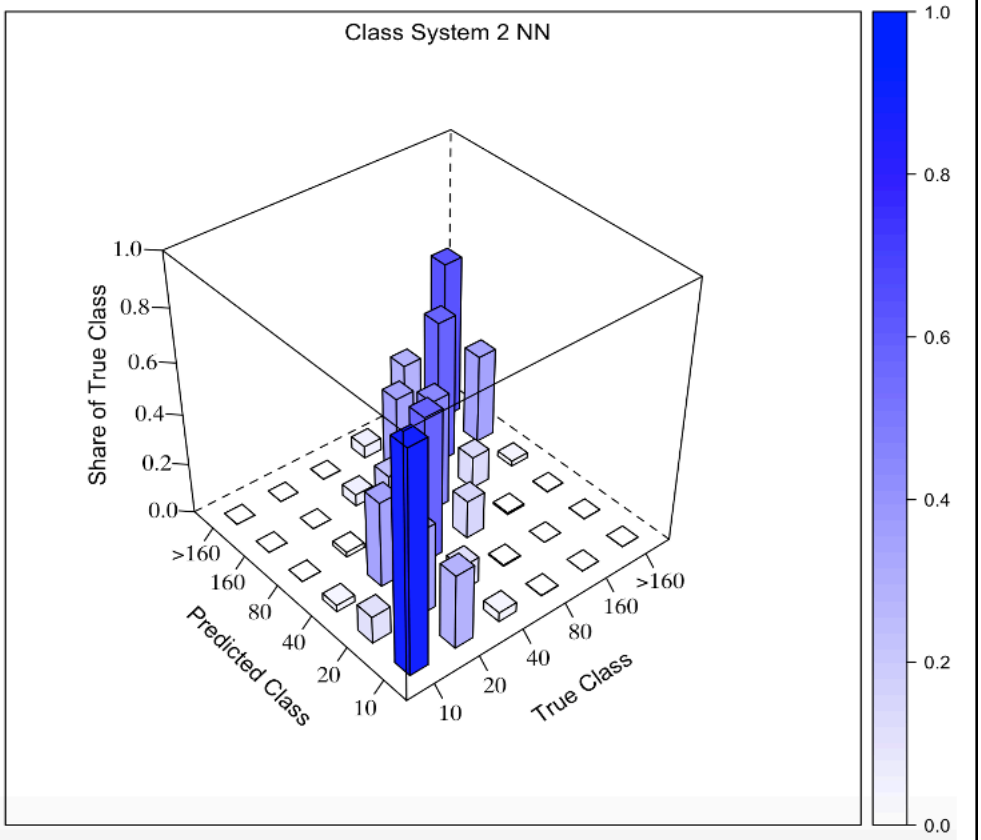


MCC = 0,354		Predicted Class									
		1	2	4	8	16	32	64	128	256	>256
True Class	1	0,326	0,026	0,288	0,311	0,028	0,021	0	0	0	0
	2	0,24	0,004	0,253	0,395	0,06	0,047	0	0	0	0
	4	0,172	0,013	0,234	0,416	0,09	0,075	0	0	0	0
	8	0,09	0,005	0,126	0,389	0,194	0,19	0,005	0,001	0	0
	16	0,016	0	0,03	0,192	0,202	0,541	0,017	0,002	0	0
	32	0,001	0	0,001	0,026	0,05	0,662	0,203	0,047	0,009	0
	64	0	0	0	0,001	0,001	0,19	0,389	0,27	0,149	0
	128	0	0	0	0	0	0,009	0,091	0,31	0,585	0,005
	256	0	0	0	0	0	0,001	0,006	0,107	0,848	0,037
	>256	0	0	0	0	0	0	0	0,027	0,813	0,16

Figure 25. Confusion Matrix of NN System 1 with Visualization

The confusion matrix looks somewhat similar than it did for the equivalent RF model. Especially the smaller classes are very deviated, and the bar height is low. However, the true positive rate of the smallest class is much lower than it was for RF, only 32,6 %. Otherwise, any major observations cannot be made that would differentiate system 1 RF and NN models from each other.

The results of NN classification system 2 are showcased in Figure 26. The MCC of 0,397 is in line with the earlier experiments. Therefore, the MCC for NN system 2 is slightly higher than it was for NN system 1, but slightly lower than it was for RF system 2. Similarly to RF, the deviations are much more subtle compared to system 1.

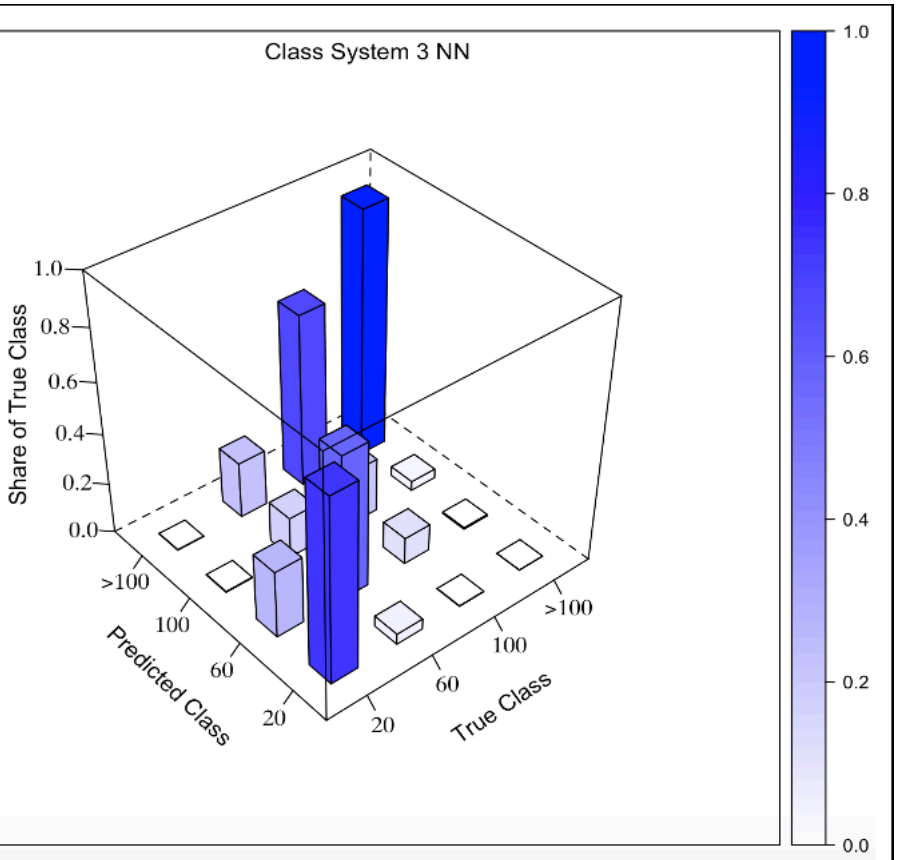


MCC = 0,397		Predicted Class					
		10	20	40	80	160	>160
True Class	10	0,866	0,108	0,026	0	0	0
	20	0,297	0,342	0,342	0,019	0	0
	40	0,037	0,101	0,577	0,238	0,046	0,001
	80	0,002	0,005	0,152	0,443	0,351	0,048
	160	0	0	0,007	0,128	0,566	0,299
	>160	0,001	0,001	0,002	0,021	0,352	0,623

Figure 26. Confusion Matrix of NN System 2 with Visualization

The smallest class has so far the highest true positive rate, 86,6 %. That said, this model has problems to predict the second class, as only 34,2 % of the true observations are correctly predicted. The ratios on the diagonal are comparable to the equivalent RF model, and in general higher than NN system 1. Rather than wide range deviations, each class' neighbouring classes seem to include most of the misclassifications which was the case for the equivalent RF model as well. Hence, system 2 seems to perform better than system 1 also with NN.

Finally, the confusion matrix of NN system 3 is presented in Figure 27. Once again, the MCC (0,543) is on the corresponding level to the parameter optimization results. It falls a bit behind the equivalent RF model's MCC, but the visualized confusion matrix looks to some extent alike.



MCC = 0,543		Predicted Class			
		20	60	100	>100
True Class	20	0,729	0,266	0,003	0,002
	60	0,042	0,574	0,161	0,222
	100	0,002	0,106	0,214	0,679
	>100	0,001	0,006	0,037	0,956

Figur 27. Confusion Matrix of NN System 3 with Visualization

The most interesting take out of this model is the clearly lower ratio of correctly predicted values for the first class as it is almost 10 and 15 percentage points below the RF system 3 and NN system 2, respectively. This combined with lower MCC leaves the NN system 3 lagging behind the RF system 3, even though rest of the class specific true positive rates are rather similar.

5.3 Summary of Results

The results consisted of the parameter optimization part and the final model evaluation part. The second research question concerning the best parameter combinations was already addressed earlier and thus will not be discussed again in this section.

The research question 3 and 4 are still not answered. The newly presented results on final models allow to cover these questions. The third research question was *"How random forest and neural network compare to each other with the turbofan dataset?"*. The simplest way to compare RF and NN in general is to take a look at the MCC's of equivalent class systems. Table X summarizes the MCC's of all trained models.

Table 7. MCC's of All Models

MCC	RF	NN
System 1	**0,392**	0,354
System 2	**0,429**	0,397
System 3	**0,590**	0,543

Comparison shows that RF outperforms NN with every system. However, the differences between equivalent systems are not drastically huge, only 0,032-0,047 MCC. It might be possible to further narrow these differences or even flip them around with additional parameter optimization for NN. Although, with only these few parameters optimized, RF is able to outperform NN by a fraction, thus making it more suitable for the turbofan dataset.

The last research question *"How random forest and neural network classifiers perform for different classification systems with the turbofan dataset?"* can be answered by comparing the MCC's between different class systems within an algorithm. System 3 was by far the best out the three systems. This is most likely due to the much simpler class structure.

System 2 seems to perform slightly better than system 1, even though the difference is not very massive in terms of MCC. This is an interesting finding because system 1 is after all much more specific for the small classes. This could indicate that better, more thoughtful construction of the smaller class intervals could lead to improved performance.

Comparison between system 3 and the other two systems reveal that it is reasonable to aggregate the higher classes together to achieve more accurate overall predictions. Systems 2 and 3 tend to mix higher classes with neighbouring classes, hence the benefit of having separate higher classes disappears and having just one or maybe two high classes could be enough. Of course, this also depends about the specific practical needs concerning the classification as in some applications it might be desired to get knowledge about the higher RUL as well.

One could argue that these are just results of one random sample as the final evaluation was based only on the test set performance, and not for example cross-validation. However, the results obtained during parameter optimization are in line with the final results, and since 5-fold cross-validation was used to optimize the parameters, it can be suggested that the test set represent the whole data rather well and the results should be reliable enough.

6 Conclusions and Further Research

This study's purpose was to showcase a classification approach for remaining useful life (RUL) prediction in the predictive maintenance framework. The methodology was demonstrated by using the NASA turbofan dataset.

First, some theory concerning predictive maintenance and machine learning was introduced to give a basic understanding about the methods and concepts used later on. Then, a literature review concentrating on classification in RUL prediction was conducted. The first set of research questions were tied specifically to the literature review:

1. **How previous research has approached RUL-type problems with classification?**
a. **Which algorithms have been used the most in RUL classification?**
b. **How the performance of classification models has been evaluated?**

By conducting the review, these questions were able to be answered. First of all, there were no single exact way to form a classification framework for RUL prediction problem. The approach is heavily dependent about the used data, which leads back to the practical application. Both Binary and multi-class classification has been used to convert the continuous problem into discrete problem, yet binary approach has been widely more popular. The time interval settings have also varied a lot, which is only logical as those are and should be highly dependent on the application.

The review also revealed that wide range of classification algorithms have been used to solve the problem. Support vector machine and decision tree were among the most used ones, and the popularity of random forest (RF) had increased during the recent years. No ultimately best algorithm could be identified as different data usually requires different algorithms for the best results. The obtained results have also been varying a lot between studies.

Previous studies have used many different metrics to evaluate the performance. Accuracy, true positive rate (TPR) and area under curve (AUC) have been the most popular measures. Also, some other simple confusion matrix -based metrics have been used commonly. Less frequently some more complex confusion matrix -based metrics such as Matthews correlation coefficient (MCC), or some custom scoring functions have been used as well.

Following the literature review, the methodology used in this study was introduced and discussed. This included introduction to the used turbofan dataset, construction of three different classification systems, data pre-processing and selection of the classification algorithms and appropriate evaluation metrics. It was chosen to use RF and neural network (NN) classifiers to train the models while MCC and confusion matrix with visualization were chosen to be used in performance evaluation.

Lastly, the results were presented. To begin the results section, hyperparameter optimization was performed for all three systems and both algorithms. The second research question was formed with respect to the parameter optimization:

2. **What random forest and neural network parameters lead to the best performance with the turbofan dataset?**

Due to limited computing power, some simplification had to be done within the optimization. Thus, only two parameters per algorithms were chose to be focused on. For RF, number of trees and nodesize, and for NN, number of hidden layers and number of neurons per layer were concentrated on. In the case of RF, the number of trees had close to no effect on the performance and best results were obtained when nodesize was set to 1. The optimal NN parameters changed depending on the class system, but the differences were rather marginal in case of both chosen parameters.

The final models were trained and evaluated after deciding the optimal parameter combinations. The last two research question were formed to address the final model evaluation and comparison:

3. **How random forest and neural network compare to each other with the turbofan dataset?**
4. **How random forest and neural network classifiers perform for different classification systems with the turbofan dataset?**

The overall comparisons of performance between both the algorithms and the systems were done by MCC. More detailed analysis with respect to the confusion matrices was provided in chapter 5. It was found that with these exact parameters considered, the RF performed better than NN with all class systems, although the differences between equivalent class systems were rather low, the difference in MCC varying between 0,032-0,047. Thus, further NN parameter tuning might change the relationship.

The differences between systems were found to be more important. System 3 with the lowest number of classes easily outperformed the other two systems. The difference between systems 1 and 2 was much smaller in terms of MCC but confusion matrices showed that there was much more deviation with system 1.

A conclusion can be made that the choice of algorithm is not as important as the proper construction of the class system, at least when these two algorithms are examined. The MCC's for system 3 were 0,59 (RF) and 0,54 (NN), meaning that both models were quite far from perfect classification, but there definitely exist potential with this framework. Further experimenting with different class intervals, better parameter optimization and possibly other algorithms should be considered.

A good division of the classes should consider the practical application and provide useful information for the user. Thus, each application should have unique classification system depending on the aspects that define the maintenance process, such as the required time to perform maintenance.

Also, it is important to understand the meaning of different false predictions. False positive, or early predictions in this context, for the small classes can be considered to lead to early maintenance actions, meaning that some remaining useful life is wasted. But if the misclassification is not far, the loss is rather insignificant. However, false negatives (late predictions) could turn out to be fatal, especially for the smaller classes. It could mean that failure happens before the need of maintenance is noticed. These factors should be considered when forming and testing class systems.

Comparison to previous studies is rather difficult due to different data and evaluation metrics. The only study that used MCC as metric was Böhm (2017). That study resulted in very high MCC's close to one but as said, different data was used. However, the it shows that for some datasets, this approach can work very well. This study was not able to reach that high level of performance, hence further research is required. The results of this study cannot be generalized to be applied to other datasets.

When considering classification to predict remaining useful life, the focus should lie on the practical application. Mainly, the classes should be formed with maintenance requirements in mind. Essentially, that is the only way to actually reduce costs and prevent failures in real life.

6.1 Limitations and Further Research

Some limitations concerning this study need to be addressed as they affected the methodologies used. These limitations also provide some ideas about future research possibilities. First of all, computational limitations proved to be one of the most significant limiting aspects.

Computational limitations affected especially the parameter optimization, where compromises had to be done. It is possible that the models were not performing at their best possible level, due to non-optimal parameters. For RF, it is unlikely that this was the case, but for NN it is more likely. Additionally, some classification algorithms, such as SVM, could not be considered due to limited computing power. For future research, extensive parameter testing in addition to experimenting with other classification algorithms is suggested to investigate whether improved results can be achieved.

Another limitation is related to the class constructions of the systems. No practical justifications were used when creating the systems. This limits the practical usefulness of the results, although the data was simulated and provided time units (cycles) might not be transferrable into real world anyways. Also, the class systems were created without preliminary testing, meaning that it is probable that more optimal systems could be created via extensive experimentations. Therefore, future research could make the classification system construction the focal point either by taking the practical view into consideration, or alternatively testing different systems more comprehensively in quest for finding the most optimal solution. This could include also binary classification to be compared with multi-class cases.

The empirical part of this study was conducted by using the whole turbofan dataset. Initially, it was provided as four different datasets with different settings and fault modes. The aggregation of the datasets into one resulted in decent outcome, but each dataset could have treated individually. This could have led to better results in general. Also, almost all of the features were used to train the models. Future studies using this dataset could consider different approaches to the mentioned issues.

Even though the results of this study show that there exists some potential about the methodology for this kind of task, it cannot be generalized to be applicable for other datasets. The results might be very good or on the other hand very bad, depending on the specific data under consideration and some other factors. Experiments using similar methodology for different data could be considered to get more evidence about the effectiveness and generability of this approach.

References

Al Iqbal, M., Zhao, R., Ji, Q., & Bennett, K. (2018). A generalized method for fault detection and diagnosis in SCADA sensor data via classification with uncertain labels. Paper presented at the *International Conference on Data Science ICDATA'18,*

Allah Bukhsh, Z., Saeed, A., Stipanovic, I., & Doree, A. G. (2019). Predictive maintenance using tree-based classification techniques: A case of railway switches. *Transportation Research Part C, 101*, 35-54.

Arlot, S., & Celisse, A. (2010). A survey of cross-validation procedures for model selection. *Statistics Surveys, 4*, 40-79.

Bluvband, Z., Porotsky, S., & Tropper, S. (2014). Critical zone recognition: Classification vs. regression. Paper presented at the *2014 International Conference on Prognostics and Health Management,* 1-5.

Bohm, T. (2017). Remaining useful life prediction for railway switch engines using classification techniques. *International Journal of Prognostics and Health Management 8,*

Bradley, A. P. (1997). The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition, 30*(7), 1145-1159.

Breiman, L. (2001). Random forests. *Machine Learning, 45*(1), 5-32.

Carvalho, T. P., Soares, Fabrízzio A. A. M. N, Vita, R., Francisco, R. d. P., Basto, J. P., & Alcalá, S. G. S. (2019). A systematic literature review of machine learning methods applied to predictive maintenance. *Computers & Industrial Engineering, 137*, 106024.

Crespo Marquez, A. (2007). *The maintenance management framework: Models and methods for complex systems maintenance*. London: Springer-Verlag London Limited.

Dougherty, G. (2013). *Pattern recognition and classification* Springer Science & Business Media.

Duda, R. O., Hart, P. E., & Stork, D. G. (2012). *Pattern classification* (2nd ed.) John Wiley & Sons.

Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters, 27*(8), 861-874.

Fernandez-Delgado, M., Cernadas, E., & Barro, S. (2014). Do we need hundreds of classifiers to solve real world classification problems? *Journal of Machine Learning Research, 15*(1), 3133-3181.

Fink, O., Zio, E., & Weidmann, U. (2015). A classification framework for predicting components' remaining useful life based on discrete-event diagnostic data. *IEEE Transactions on Reliability, 64*(3), 1049-1056.

Georgescu, R., Berger, C. R., Willett, P., Azam, M., & Ghoshal, S. (2010). Comparison of data reduction techniques based on the performance of SVM-type classifiers. Paper presented at the *2010 IEEE Aerospace Conference,* 1-9.

Gorodkin, J. (2004). Comparing two K-category assignments by a K-category correlation coefficient. *Computational Biology and Chemistry, 28*(5), 367-374.

Hossin, M., & Sulaiman, M. N. (2015). A review on evaluation metrics for data classification evaluations. *International Journal of Data Mining & Knowledge Management Process, 5*(2)

Jardine, A. K. S., Lin, D., & Banjevic, D. (2006). A review on machinery diagnostics and prognostics implementing condition-based maintenance. *Mechanical Systems and Signal Processing, 20*(7), 1483-1510.

Jordan, M. I., & Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. *Science (New York, N.Y.), 349*(6245), 255-260.

Jurman, G., & Furlanello, C. (2010). A unifying view for performance measures in multiclass prediction. *arXiv Preprint*, arXiv:1008.2908.

Kauschke, S., Janssen, F., & Schweizer, I. (2015, Advances in predictive maintenance for a railway scenario - project techlok. *Knowledge Engineering Group, University of Darmstadt,*

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. *Ijcai, 14*(2), 1137–1143.

Kubat, M. (2015). *An introduction to machine learning* Springer International Publishing.

Kusiak, A., & Li, W. (2011).

The prediction and diagnosis of wind turbine faults. *Renewable Energy, 36*(1), 16-23.

Letourneau, S., Famili, F., & Matwin, S. (1999). Data mining to predict aircraft component replacement. *IEEE Intelligent Systems and their Applications, 14*(6), 59-66.

Li, H., Parikh, D., He, Q., Qian, B., Li, Z., Fang, D., & Hampapur, A. (2014). Improving rail network velocity: A machine learning approach to predictive maintenance. *Transportation Research Part C, 45*, 17-26.

Liao, L., & Kottig, F. (2014). Review of hybrid prognostics approaches for remaining useful life prediction of engineered systems, and an application to battery life prediction. *IEEE Transactions on Reliability, 63*(1), 191-207.

Louridas, P., & Ebert, C. (2016). Machine learning. *IEEE Software, 33*(5), 110-115.

Matthews, B. W. (1975). Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochimica Et Biophysica Acta - Protein Structure, 405*(2), 442–451.

NASA Prognostics Center. (2020). PCoE datasets. Retrieved from https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/#turbofan

Oshiro, T. M., Perez, P. S., & Baranauskas, J. A. (2012). How many trees in a random forest? Paper presented at the *International Workshop on Machine Learning and Data Mining in Pattern Recognition,* 154-168.

Probst, P., Boulesteix, A., & Bischl, B. (2019). Hyperparameters, tuning and meta-learning for random forest and other machine learning algorithms. *Journal of Machine Learning Research, 20*(1), 1-32.

Probst, P., Wright, M. N., & Boulesteix, A. (2019). Hyperparameters and tuning strategies for random forest. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 9*(3), e1301.

Rebala, G., Ravi, A., & Churiwala, S. (2019). *An introduction to machine learning* Springer.

Saxena, A., & Goebel, K. (2008). Turbofan engine degradation simulation data set. *NASA Ames Prognostics Data Repository,*

Schwabacher, M., & Goebel, K. (2007). A survey of artificial intelligence for prognostics. Paper presented at the *AAAI Fall Symposium: Artificial Intelligence for Prognostics,* 108–115.

Sebastian Kauschke, Immanuel Schweizer, Michael Fiebrig, & Frederik Janssen. (2014). Learning to predict component failures in trains. Paper presented at the *Proceedings of the LWA 2014 Workshop,* 71-82.

Shanmuganathan, S., & Samarasinghe, S. (2016). *Artificial neural network modelling* Springer.

Si, X., Wang, W., Hu, C., & Zhou, D. (2011). Remaining useful life estimation – A review on the statistical data driven approaches. *European Journal of Operational Research, 213*(1), 1-14.

Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing and Management, 45*, 427–437.

Vijipriya, J., Ashok, J., & Suppiah, S. (2016). A review on significance of sub fields in artificial intelligence. *International Journal of Latest Trends in Engineering and Technology, 6*(3), 542-548.

Xue, Y., Williams, D., & Qiu, H. (2011). Classification with imperfect labels for fault prediction. Paper presented at the *Proceedings of the First International Workshop on Data Mining for Service and Maintenance,* 12-16.

Yang, C., & Létourneau, S. (2005). Learning to predict train wheel failures. Paper presented at the *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining,* 516-525.

Yang, C., & Létourneau, S. (2009). Two-stage classifications for improving time-to-failure estimates: A case study in prognostic of train wheels. *Applied Intelligence, 31*(3), 255-266.

Zaluski, M., Létourneau, S., Bird, J., & Yang, C. (2011). Developing data mining-based prognostic models for CF-18 aircraft. *Journal of Engineering for Gas Turbines and Power, 133*(10)

Zhang, Z., Si, X., Hu, C., & Lei, Y. (2018). Degradation data analysis and remaining useful life estimation: A review on wiener-process-based methods. *European Journal of Operational Research, 271*(3), 775-796.

Zhao, D., Georgescu, R., & Willett, P. (2011). Comparison of data reduction techniques based on SVM classifier and SVR performance. *Signal and Data Processing of Small Targets, 8137*, 81370X.

Zhao, R., Al Iqbal, M. R., Bennett, K. P., & Qiang Ji. (2016). Wind turbine fault prediction using soft label SVM. Paper presented at the *2016 23rd International Conference on Pattern Recognition (ICPR),* 3192-3197.

Appendices

Class System 1		Number of Trees					Average STD of CV Folds
		100	200	300	400	500	
Nodesize	1	0,401	0,404	**0,407**	0,406	**0,407**	0,002
	100	0,398	0,400	0,400	0,401	0,401	0,003
	1000	0,352	0,353	0,353	0,354	0,353	0,004
	10000	0,243	0,244	0,243	0,243	0,245	0,003
Min		0,243	0,244	0,243	0,243	0,245	
Max		0,401	0,404	**0,407**	0,406	**0,407**	

Class System 2		Number of Trees					Average STD of CV Folds
		100	200	300	400	500	
Nodesize	1	0,445	0,449	**0,451**	**0,451**	0,450	0,002
	100	0,445	0,446	0,447	0,446	0,447	0,004
	1000	0,399	0,398	0,399	0,399	0,400	0,004
	10000	0,274	0,276	0,277	0,275	0,277	0,005
Min		0,274	0,276	0,277	0,275	0,277	
Max		0,445	0,449	**0,451**	**0,451**	0,450	

Class System 3		Number of Trees					Average STD of CV Folds
		100	200	300	400	500	
Nodesize	1	0,598	0,601	0,600	**0,602**	0,601	0,004
	100	0,597	0,597	0,598	0,597	0,598	0,004
	1000	0,558	0,558	0,561	0,560	0,559	0,002
	10000	0,388	0,389	0,389	0,389	0,389	0,004
Min		0,388	0,389	0,389	0,389	0,389	
Max		0,598	0,601	0,600	**0,602**	0,601	

Appendix 1. Average MCC's of All RF Systems

Class System 1		Hidden Layers 1	2	3	Average STD of CV Folds
Neurons per Layer	10	0,335	0,352	0,355	0,009
	20	0,344	0,351	0,362	0,007
	30	0,345	0,358	0,360	0,006
	40	0,341	0,357	0,358	0,008
	50	0,331	0,357	0,360	0,005
	60	0,340	0,359	0,362	0,008
	70	0,344	0,360	**0,366**	0,005
	80	0,342	0,362	0,364	0,007
	90	0,340	0,359	0,360	0,007
	100	0,340	0,360	**0,366**	0,008
	110	0,346	0,361	**0,366**	0,007
	120	0,345	0,360	0,358	0,008
	130	0,347	0,359	**0,366**	0,006
	140	0,346	0,361	0,365	0,005
	150	0,345	0,364	0,365	0,007
	160	0,345	0,363	**0,366**	0,005
	170	0,345	0,361	**0,366**	0,007
	180	0,348	0,363	0,365	0,006
	190	0,348	0,363	0,364	0,006
	200	0,351	0,364	0,365	0,006
Min		0,331	0,351	0,355	
Max		0,351	0,364	**0,366**	

Class System 2		Hidden Layers 1	2	3	Average STD of CV Folds
Neurons per Layer	10	0,379	0,392	0,398	0,009
	20	0,385	0,389	0,404	0,009
	30	0,380	0,400	0,404	0,009
	40	0,393	0,390	0,407	0,008
	50	0,393	0,400	0,405	0,006
	60	0,387	0,400	0,402	0,005
	70	0,386	0,401	0,405	0,006
	80	0,391	0,401	0,406	0,007
	90	0,392	0,396	0,405	0,006
	100	0,394	0,400	0,403	0,006
	110	0,384	0,403	0,406	0,006
	120	0,384	0,400	**0,409**	0,008

130	0,389	0,401	0,408	0,006
140	0,395	0,403	0,408	0,003
150	0,397	0,404	0,404	0,007
160	0,393	0,406	0,404	0,006
170	0,398	0,404	0,407	0,004
180	0,396	0,400	0,403	0,005
190	0,400	0,402	0,404	0,006
200	0,395	0,406	0,405	0,007
Min	0,379	0,389	0,398	
Max	0,400	0,406	**0,409**	

Class System 3	Hidden Layers			Average STD of CV Folds
Neurons per Layer	1	2	3	
10	0,544	0,551	0,556	0,007
20	0,553	0,562	0,564	0,007
30	0,553	0,562	0,567	0,006
40	0,553	0,567	0,568	0,006
50	0,556	0,556	0,565	0,006
60	0,549	0,563	0,565	0,010
70	0,560	0,569	0,568	0,007
80	0,553	0,570	0,567	0,005
90	0,556	0,567	0,568	0,006
100	0,557	0,568	0,570	0,005
110	0,551	0,565	0,569	0,007
120	0,556	0,571	0,569	0,007
130	0,554	**0,573**	0,570	0,006
140	0,559	0,566	0,569	0,006
150	0,557	0,567	0,568	0,010
160	0,559	0,565	0,570	0,005
170	0,561	0,568	0,571	0,006
180	0,558	0,572	0,569	0,004
190	0,564	0,570	0,570	0,007
200	0,560	0,571	0,571	0,005
Min	0,544	0,551	0,556	
Max	0,564	**0,573**	0,571	

Appendix 2. Average MCC's of All NN Systems

RF System 1 N = 85595		Predicted Class									
		1	2	4	8	16	32	64	128	256	>256
True Class	1	243	19	40	94	57	13	0	0	0	0
	2	56	5	31	83	51	7	0	0	0	0
	4	74	3	37	178	154	20	0	0	0	0
	8	45	1	24	287	484	104	2	0	0	0
	16	11	0	6	151	1059	783	51	0	0	0
	32	1	0	0	12	436	2460	1414	197	12	0
	64	4	2	1	2	23	966	4765	3346	864	0
	128	1	0	2	4	18	64	1918	10186	11339	48
	256	8	3	0	12	22	32	148	5387	30138	626
	>256	0	0	0	0	0	0	1	255	6261	444

RF System 2 N = 85595		Predicted Class					
		10	20	40	80	160	>160
True Class	10	2158	387	58	0	0	0
	20	485	1205	899	81	3	0
	40	70	555	3022	1795	369	2
	80	28	39	1181	5650	5690	404
	160	0	4	76	2334	20251	8401
	>160	28	24	40	182	11318	18856

RF System 3 N = 85595		Predicted Class			
		20	60	100	>100
True Class	20	4345	927	2	2
	60	724	8050	1471	1846
	100	42	2206	2975	8685
	>100	54	565	1689	52012

NN System 1 N = 85595		Predicted Class									
		1	2	4	8	16	32	64	128	256	>256
True Class	1	152	12	134	145	13	10	0	0	0	0
	2	56	1	59	92	14	11	0	0	0	0
	4	80	6	109	194	42	35	0	0	0	0
	8	85	5	119	368	184	180	5	1	0	0
	16	34	0	61	396	416	1114	35	5	0	0
	32	6	0	6	117	226	3002	920	213	42	0
	64	0	0	1	11	13	1890	3881	2695	1482	0
	128	0	0	1	3	7	207	2154	7310	13791	107
	256	0	0	2	17	8	49	233	3907	30831	1329
	>256	0	0	0	0	0	0	2	186	5662	1111

NN System 2 N = 85595		Predicted Class					
		10	20	40	80	160	>160
True Class	10	2253	282	68	0	0	0
	20	794	915	914	50	0	0
	40	213	587	3355	1385	268	5
	80	27	59	1973	5753	4561	619
	160	0	4	218	3988	17572	9284
	>160	27	20	55	648	10726	18972

NN System 3 N = 85595		Predicted Class			
		20	60	100	>100
True Class	20	3845	1404	18	9
	60	513	6941	1952	2685
	100	26	1470	2975	9437
	>100	41	344	2006	51929

Appendix 3. Confusion Matrix Absolute Values of All Models