Master Degree Project

UNIVERSITY OF SKÖVDE

COMPARISON OF UNDERSAMPLING METHODS FOR PREDICTION OF CASTING DEFECTS BASED ON PROCESS PARAMETERS

Master Degree Project
One year Level 18 ECTS
Spring term 2020

Simon Lööv

Supervisors: Sunith Bandaru & Rickard Aston

Examiner: Amos Ng

Abstract

Prediction of both big and small decisions is something most companies have to make on a daily basis. The importance of having a highly accurate technique for different decision-making is not something that is new. However, even though the importance of prediction is a fact to most people, current techniques for estimation are still often highly inaccurate. The consequences of an inaccurate prediction can be huge in the differences between the misclassifications. Not just in the industry but for many different areas. Machine learning have in the recent couple of years improved significantly and are now considered a reliable method to use for prediction.

The main goal of this research is to predict casting defects with the help of a machine-learning algorithm based on process parameters. In order to achieve the main goal, some sub-objectives have been identified to successfully reach those goals. A problem when dealing with machine learning is an unbalanced dataset. When training a network, it is essential that the dataset is balanced. In this research we have successfully balanced the dataset. Undersampling was the method used in our research to establish our balanced dataset. The research compares and evaluates a couple of different undersample methods in order to see which undersampling is best suited for this project. Three different machine models, “random forest”, “artificial neural network”, and “k-nearest neighbor”, are also compared to each other to see what model performs best. The conlcusion reached was that the best method for both undersampling and machine learning model varied due to many different reasons. So, in order to find the best model with the best method for a specific job, all the models and methods need to be tested. However, the undersampling method that provided best performances most times in our research was the NearMiss version 2 model. Artificial Neural Network was the machine learning model that had most success in our research. It performed best in two out of three evaluations and comparisons.

Keywords: Machine learning, casting defects, undersampling, random forest, artificial neural network

HÖGSKOLAN I SKÖVDE

Table of Contents

1 Introduction 1

1.1 Background 1

1.2 Aim and Objectives 2

1.3 Research Methodology 2

1.4 Limitations 4

2 Literature review 5

2.1 Casting Defects 5

2.2 Data Mining 6

2.3 Machine Learning 7

2.3.1 Supervised Learning 8

2.3.2 Unsupervised Learning 8

2.3.3 Reinforcement Learning 9

2.4 Imbalanced Data 9

2.5 Undersampling 10

2.5.1 Random Undersampling 11

2.5.2 Undersampling based on NearMiss methods 12

2.5.3 Undersampling based on the Neighborhood Cleaning Rule 13

2.5.4 Undersampling based on the Edited Nearest Neighbor method 14

2.5.5 Undersampling based on the Repeated Edited Nearest Neighbor method 14

2.5.6 Undersampling based on the Condensed Nearest Neighbor method 15

2.5.7 Undersampling based on the Instance Hardness Threshold 15

2.5.8 Undersampling based on One-Sided Selection method 16

2.5.9 Undersampling by removing Tomek's Links 17

3 Problem Description 18

3.1 Introduction to the Problem 18

3.2 Pre-processing 18

3.3 The Machines 18

4 Experimental Methodology 21
4.1 Data Collection and Aggregation 21
4.2 Evaluation of Classification 22
4.2.1 Cross Validation 22
4.2.2 Data Leakage 23
4.2.3 Classification 23
4.3 Machine Learning Models 25
4.3.1 Random Forest 25
4.3.2 Artificial Neural Network 27
4.3.3 K-NearestNeighbor 29
4.3.4 Hyperparameter Tuning 29
5 Results 31
5.1 Result for Laempe 31
5.2 Result for FCS25 32
5.3 Result for Vridkors 33
5.4 Undersample Method Evaluation 34
5.5 Machine Learning Method Evaluation 35
6 Conclusions 35
6.1 Summary of Results 35
6.2 Future Work 36
6.2.1 Changing the threshold 36
6.2.2 Treating the problem as a regression problem 37
6.2.3 Treating the problem as a multi-task classification problem 37
6.2.4 Large-scale parameter tuning 37
6.2.5 Improving data collection 37
7 References 38
Appendix 41
Appendix A: Results with Hyperparameter Tuning for Laempe 41

Appendix B: Results with Hyperparameter Tuning for FCS25 50

Appendix C: Results with Hyperparameter Tuning for Vridkors 62

HÖGSKOLAN I SKÖVDE

Table of Figures

Figure 1: Time plan for the project
Figure 2: An overview of the data mining process
Figure 3: Imbalanced data for one of the machines
Figure 4: Undersample the imbalanced data in order to balance it
Figure 5: Random undersampling
Figure 6: NearMiss-1 undersampling
Figure 7: NearMiss-2 undersampling
Figure 8: NearMiss-3 undersampling
Figure 9: Neighbourhood Cleaning Rule undersampling
Figure 10: Edited Nearest Neighbor undersampling
Figure 11: Repeated Edited Nearest Neighbor undersampling
Figure 12: Condensed Neasest Neighbor undersampling
Figure 13: Instance Hardness Threshold undersampling
Figure 14: One-Sided Selection undersampling
Figure 15: Tomek's Links undersampling
Figure 16: Missing data in one of the data files
Figure 17: Example of ROC-curves
Figure 18: Random forest classifer
Figure 19: The layout of a typical ANN

HÖGSKOLAN I SKÖVDE

Index of Tables

Table 1: Table of the recorded defected products produced by Laempe during January

Table 2: Table of parameters which caused defected products for Laempe

Table 3: Table of parameters which caused defected products for FSC25

Table 4: Table of parameters which caused defected products for Vridkorsmaskin (swivel machine)

Table 5: Confusion matrix of possible outcomes

Table 6: Result for defect type Trasig Vattenmantel with random forest evaluated with default parameters & with grid CV search

Table 7: Evaluation of undersampling methods for Laempe

Table 8: Evaluation of machine learning methods for Laempe

Table 9: Evaluation of undersampling methods for FSC25

Table 10: Evaluation of machine learning methods for FSC25

Table 11: Evaluation of undersampling methods for Vridkors

Table 12: Evaluation of machine learning methods for Vridkors

Table 13: Result of the best undersampling method for each machine

Table 14: Result of the best machine learning method for each machine

Table 15: Laempe grid CV search on vattenmantel with rf

Table 16: Laempe grid CV search on vattenmantel with ann

Table 17: Laempe grid CV search on vattenmantel with knn

Table 18: Laempe grid CV search on porös kärna with rf

Table 19: Laempe grid CV search on porös kärna with ann

Table 20: Laempe grid CV search on porös kärna with knn

Table 21: Laempe grid CV search on utsötarmärken with rf

Table 22: Laempe grid CV search on utsötarmärken with ann

Table 23: Laempe grid CV search on utsötarmärken with knn

Table 24: Laempe grid CV search on kärna ej fylld with rf

Table 25: Laempe grid CV search on kärna ej fylld with ann

Table 26: Laempe grid CV search on kärna ej fylld with knn

Table 27: Laempe grid CV search on ohärdad kärna with rf

Table 28: Laempe grid CV search on ohärdad kärna with ann

Table 29: Laempe grid CV search on ohärdad kärna with knn

Table 30: Laempe grid CV search on övriga fel with rf

Table 31: Laempe grid CV search on övriga fel with ann

Table 32: Laempe grid CV search on övriga fel with knn

Table 33: FCS25 grid CV search on trasig vattenmantel with rf

Table 34: FCS25 grid CV search on trasig vattenmantel with ann
Table 35: FCS25 grid CV search on trasig vattenmantel with knn
Table 36: FCS25 grid CV search on porös kärna with rf
Table 37: FCS25 grid CV search on porös kärna with ann
Table 38: FCS25 grid CV search on porös kärna with knn
Table 39: FCS25 grid CV search on utstötamärken with rf
Table 40: FCS25 grid CV search on utstötamärken with ann
Table 41: FCS25 grid CV search on utstötamärken with knn
Table 42: FCS25 grid CV search on kärna ej fylld with rf
Table 43: FCS25 grid CV search on kärna ej fylld with ann
Table 44: FCS25 grid CV search on kärna ej fylld with knn
Table 45: FCS25 grid CV search on ohärdad kärna with rf
Table 46: FCS25 grid CV search on ohärdad kärna with ann
Table 47: FCS25 grid CV search on ohärdad kärna with knn
Table 48: FCS25 grid CV search on övriga fel with rf
Table 49: FCS25 grid CV search on övriga fel with ann
Table 50: FCS25 grid CV search on övriga fel with knn
Table 51: FCS25 grid CV search on skadad stöt with rf
Table 52: FCS25 grid CV search on skadad stöt with ann
Table 53: FCS25 grid CV search on skadad stöt with knn
Table 54: FCS25 grid CV search on skadad kanal with rf
Table 55: FCS25 grid CV search on skadad kanal with ann
Table 56: FCS25 grid CV search on skadad kanal with knn
Table 57: Vridkors grid CV search on kavitet 1, 2, 3 with rf
Table 58: Vridkors grid CV search on kavitet 1, 2, 3 with ann
Table 59: Vridkors grid CV search on kavitet 1, 2, 3 with knn
Table 60: Vridkors grid CV search on kavitet 4, 5, 6 with rf
Table 61: Vridkors grid CV search on kavitet 4, 5, 6 with ann
Table 62: Vridkors grid CV search on kavitet 4, 5, 6 with knn
Table 63: Vridkors grid CV search on kavitet 7, 8, 9 with rf
Table 64: Vridkors grid CV search on kavitet 7, 8, 9 with ann
Table 65: Vridkors grid CV search on kavitet 7, 8, 9 with knn
Table 66: Vridkors grid CV search on kavitet 10, 11, 12 with rf
Table 67: Vridkors grid CV search on kavitet 10, 11, 12 with ann
Table 68: Vridkors grid CV search on kavitet 10, 11, 12 with knn

Table 69: Vridkors grid CV search on kavitet 13, 14, 15 with rf
Table 70: Vridkors grid CV search on kavitet 13, 14, 15 with ann
Table 71: Vridkors grid CV search on kavitet 13, 14, 15 with knn
Table 72: Vridkors grid CV search on kavitet 16, 17, 18 with rf
Table 73: Vridkors grid CV search on kavitet 16, 17, 18 with ann
Table 74: Vridkors grid CV search on kavitet 16, 17, 18 with knn
Table 75: Vridkors grid CV search on kavitet 1, 2, 3 (sp) with rf
Table 76: Vridkors grid CV search on kavitet 1, 2, 3 (sp) with ann
Table 77: Vridkors grid CV search on kavitet 1, 2, 3 (sp) with knn
Table 78: Vridkors grid CV search on kavitet 4, 5, 6 (sp) with rf
Table 79: Vridkors grid CV search on kavitet 4, 5, 6 (sp) with ann
Table 80: Vridkors grid CV search on kavitet 4, 5, 6 (sp) with knn
Table 81: Vridkors grid CV search on kavitet 7, 8, 9 (sp) with rf
Table 82: Vridkors grid CV search on kavitet 7, 8, 9 (sp) with ann
Table 83: Vridkors grid CV search on kavitet 7, 8, 9 (sp) with knn
Table 84: Vridkors grid CV search on kavitet 10, 11, 12 (sp) with rf
Table 85: Vridkors grid CV search on kavitet 10, 11, 12 (sp) with ann
Table 86: Vridkors grid CV search on kavitet 10, 11, 12 (sp) with knn
Table 87: Vridkors grid CV search on kavitet 13, 14, 15 (sp) with rf
Table 88: Vridkors grid CV search on kavitet 13, 14, 15 (sp) with ann
Table 89: Vridkors grid CV search on kavitet 13, 14, 15 (sp) with knn
Table 90: Vridkors grid CV search on kavitet 16, 17, 18 (sp) with rf
Table 91: Vridkors grid CV search on kavitet 16, 17, 18 (sp) with ann
Table 92: Vridkors grid CV search on kavitet 16, 17, 18 (sp) with knn
Table 93: Vridkors grid CV search on uppstartskass with rf
Table 94: Vridkors grid CV search on uppstartskass with ann
Table 95: Vridkors grid CV search on uppstartskass with knn

Terminology

A

ANN

Artificial neural network i

C

CNN

Condensed nearest neighbor

CV

Cross-validation i

E

ENN

Edited nearest neighbor

F

FN

False negative i

FP

False positive i

K

KNN

K-NearestNeighbor

N

NCR

Neighborhood cleaning rule i

NM_v1

Nearest neighbor verson 1 i

NM_v2

Nearest neighbor verson 2 i

NM_v3

Nearest neighbor verson 3 i

O

OSS

One sided selection

R

RF

Random forest

RENN

Repeated Edited Nearest Neighbor

RUS

Random Undersampling

T

TN

True negative

TP

True positive i

1 Introduction

In this chapter of the report, a background around the problem is described followed by the aim and objectives. The research methodologies and the limitations of the project is described in the last part of this chapter. The background explains information important to know for the problem description that follows later in the report. A time plan is included in the aim and objectives to show how the work was divided over time. In order to conduct the research, a methodology was needed. The research methodology used in this thesis is presented in this chapter as well.

1.1 Background

Foundry is one of the eldest processes of the human history and it was crucial for the progress of the development of the society many years ago (Pastor-López et al., 2012). Today, foundry and casting still have an important role in the industries around the world. The basics for casting is quite easy. It consists of melting a material and pouring it into a mold. In the mold, the melted material solidifies to the shape of the chosen mold. Even a small defect may cause the product to be un-useful. Manufacturing companies today must keep evolving and innovate due to reason such as, the short lead times, small-quantity batch production, and irregular demand fluctuations in order to stay competitive (Lee et al., 2018). Over the last couple of years, the growing importance of quality in order to have competitive success has been notable by various companies that has successfully translated customer requirements into final products (Robinson & Malhotra, 2005). Predictive and flexible production are innovations manufacturing companies strives to achieve which is contrast to the typical mass production method (Wang et al., 2016). To achieve the expected quality of their products, manufacturing companies also have to deal with rapidly increasing complexity within their manufacturing processes. However, if companies can accomplish to reduce their defected products, then that will help them save both time and money. The fact that companies can rely on their manufacturing processes to produce less defected products also removes a lot of manpower needed for quality controls that is otherwise often necessary.

Scania AB is a major manufacturer of lorries, trucks and buses with its headquarters in Södertälje. One of their manufacturing operations includes casting cylinder heads in their foundry. Within their foundry, defected parts sometimes occur in these machines for different reasons. One of the main reasons being that the combination of parameters, that are essential, in the machines are not always optimum. When the parameters are not optimum it increases the risk of producing defective products.

Because of these different combinations of parameters, it is hard to foresee or to predict when defects will occur in the casting. However, machine learning has evolved and improved a lot in recent years. When it comes to handle huge numbers of predictors, machine learning shines at that. The capacity whose volume or complexity would previously would have made analyzing them extremely hard, now allows the use of new kinds of data (Obermeyer & Emanuel, 2016).

To remove/lower the number of defected products, Scania has implemented a system for collecting and categorizing data of defects by their type at each step in the casting process. This data will be used to predict any casting defects in the future.

1.2 Aim and Objectives

The aim of this project is to implement a machine learning method with the help of undersampling to address the skewed data. The purpose of the machine learning is to predict casting defects in order to prevent defective products to be manufactured in Scanias foundry. Several sub-objectives have been identified to successfully reach the main goal.

1. Develop a methodology for pre-processing, multi-source, multi-format data
    - Implement procedures for correcting faulty, missing, and data collected at different frequencies.
2. Employ and compare undersampling methods to account for class skew
3. Compare and tune various machine learning methods to predict casting defects

The following question should be answered after the project is done: What is the best prediction model for casting defects?

A time plan was created in order to make sure the work is done in time and that the work is done efficiently.

1.3 Research Methodology

When conducting research over an area, it is important to have good information on the topic and its sub-parts. Therefore, a literature review was done by searching in different scholar articles and books, in order to get reliable information. Fink (2019) define a literature review as:

“A literature review surveys books, scholarly articles, and any other sources relevant to a particular issue, area of research, or theory, and by so doing, provides a description, summary,

and critical evaluation of these works in relation to the research problem being investigated. Literature reviews are designed to provide an overview of sources you have explored while researching a particular topic and to demonstrate to your readers how your research fits within a larger field of study."

There are different types of literature reviews. In our research, a narrative literature review was used, which is a comprehensive, critical, and objective analysis of the current knowledge on a topic. A narrative literature review helps the research to establish a theoretical framework. The literature review also aided in choosing one of the certain undersampling techniques that are explained in the 2.3 chapter.

The study uses quantitative observations to gather evidence. According to Creswell (1994), a quantitative research explains phenomena by collecting numerical data that are analyzed using mathematically based methods. The definition can be broken down into three different elements. The first, and perhaps also the key element in quantitative research is explaining phenomena. In the next part is the numerical data that is collected in quantitative research. The last element being mathematically-based methods which are connected to the numerical data. Mathematically-based method refer particularly to statistics for analyzing the data. In our project, the data was collected by observing how many products get defected in a day and because of the defective products were counted in numerical numbers, it qualifies under quantitative data.

The study then continues with the analysis of the data. Wohlin et al. (2012) describe how empirical studies often are divided into three different types: surveys, case studies, or experiments. Since the research uses data derived from actual observations, the next part of the research falls under the category of empirical study. Empirical research relies on evidence obtained through observations or scientific data collection methods. Elements that often are used in empirical studies are:

- Research question
- A particular and planned design
- Primary data that are gathered and analyzed
- Methodology for collecting and analyzing the data
- Limitation of data to a particular group
- Reliability
- Ability to generalize from findings to other samples, and situations

These elements are very much like the empirical cycle which is shown in figure 2. The empirical cycle is used when a researcher designs a research setup and it is structured as a checklist (Condori-Fernandez et al., 2012). However, the research does not necessarily have each item on the checklist. One of the goals with the checklist of the empirical cycle is to help finding justifiable answers to scientific knowledge questions.

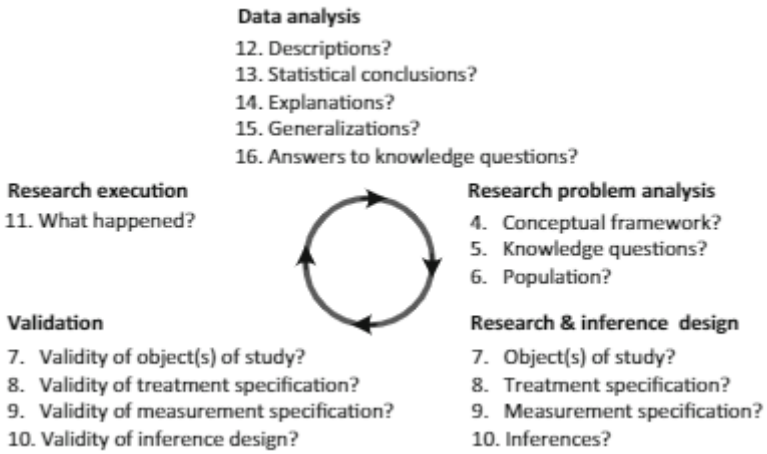

Figure 1: The empirical cycle (Wieringa, 2014)

There is an answer to all seven elements in our research. We have our main aim and objectives which cover the research question. The design of the research is planned with ways of answering the aim and objectives. Primary data has been collected by the company in one year. The methodology used for collecting the data is experimental. Due to research mainly target Scania and their casting of cylinder heads, the limitation of the data is particular targeted for prediction in casting to prevent defects. The ability to recreate the reliability and test the result proves the reliability of the research. Finally, the research can be used for different companies with casting in order to predict the casting defects of their foundry with their own collection of data.

1.4 Limitations

There are several different methods used for both dealing with imbalanced data and predicting casting defects. The research done in machine learning has grown a lot in the last couple of years. Today it is getting more and more acknowledged and used due to the impact it has in root cause analysis and prediction. The computing system chosen for dealing with the machine learning was an artificial neural network (ANN).

To deal with the imbalance problem, a undersampling method was used. Another popular method is oversampling. However, this research will only focus on undersampling.

2 Literature review

This chapter helps the reader get the knowledge needed to be able to understand the core concepts of the research conducted and the discussion and conclusion presented in the end of the report. The literature in this chapter describes casting defects that can occur in a foundry, what imbalance data is and different undersampling techniques used for dealing with the imbalanced data.

2.1 Casting Defects

Casting is a well-used manufacturing method used in many industry companies who is producing a large number of industrial components. In producing a wide variety of aluminum components, high pressure die casting (HPDC) is one of the most productive casting methods. There are many different defects that can occur during casting. Holes and flaws are often the result of the limitation of the manufacturing technique, while porosity and oxide films are two of the major defects. Entrapped gas or shrinkage of the metal during solidification is often what is causing porosity (Skallerud, Iveland & Härkegård, 1993). The defect often results in poor mechanical properties such as limited strength and ductility, variable fracture toughness, irregular crack initiation and crack propagation characteristics, potentially accompanied by a lack of pressure tightness (Ammar, Samuel & Samuel, 2008). These casting defects can lead to catastrophic failure of critical mechanical components if they are still used in production (Ferguson et al., 2018). Many of the key components, such as cylinder heads, are often manufactured under a casting process (Lin et al., 2018). Those components are not unusually interrupted by defects, such as porosity, shrinkage, or oxide films.

Machine learning within manufacturing and detecting casting quality issues has been growing rapidly in the last couple of years. However, the majority of the literature review consider casting defect classification based on image analysis through deep learning and more specific with a X-ray approach. There are several X-ray technologies available when it comes to image and visualization in defect detection in industries such as on-line weld defect inspection, magnetic particle inspection, and ultrasonic inspections (Ferguson et al., 2017). Today, it is even possible to get a three-dimensional characterization of porosity with the help of the recent developments in high resolution X-ray computed tomography (Masad et al., 2002). There are also different methods used in computer vision

techniques. Background subtraction is a very popular method. The method works by an estimated background image is subtracted from the pre-procsessed image to leave a residual image containing the defects and random noise (Gayer et al., 1990). One downside with the background subtraction method is that it tends to be very sensitive to the positioning of the image and random image noise.

In our research, we use casting defect prediction based on process parameters to predict casting defects, in order to prevent defects from happening in the first place. To our knowledge, the existing literature for casting defect prediction based on process parameters are very low with just a few papers found with similar approach.

2.2 Data Mining

There is huge amount of raw data which cannot be directly used by humans or manual applications that surrounds the world. Due to the huge data growth, organized knowledge and information cannot be easily obtained. For these reasons, a well-known discipline which is more and more important in the current world of information, is the development of data mining (Aggarwal, 2015). Data mining aims to discover useful information from large collection of data. These discoveries can appear in different ways. It can be rules which describes properties of the data, frequently occurring patterns, clustering's of the objects in the database, etc (Mannilla, 1996).

Figure 2 present an overview of the data mining process. The selection of data for analysis is the first step of the process. Historical data is normally used for this. The data set undergoes cleaning and pre-processing in the second step. This is done in order to be able to use the data and increase the quality. In the third step, the data is analysed to identify patterns. To ensure the generalizability, the model is then validated with new data sets. When a model or pattern achieve these conditions becomes knowledge. These four steps are done repeatedly until meaningful knowledge is extracted.

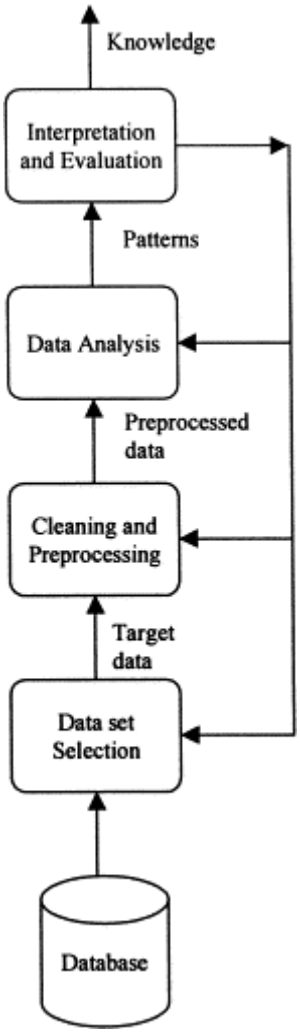

Figure 2: An overview of the data mining process (Fayyad et al., 1996)

2.3 Machine Learning

Machine learning (ML) in prediction is not something that is new. However, in recent years the artificial intelligence and various ML algorithms have advanced rapidly. There are several different algorithms, theories, and methods available in the field of ML but it is often defined as models that can improve automatically through experience. There are three different sub-classes that ML can be split into: supervised-, unsupervised-, and reinforcement learning. All three sub-classes will be described under this chapter. In this paper, we consider the prediction of casting defects with the use of Random Forest (RF), Artificial Neural Network (ANN), and K-Nearest Neighbor (KNN).

2.3.1 Supervised Learning

The goal of supervised learning is to approximate the mapping function with such good result that when new input data (X) can predict the output variables (Y) for that data. This is done by an algorithm to learn the mapping function from the input to the output. The reason why it is called supervised learning is because of the algorithm learns from a training dataset. Ståhl (2021) explained supervised learning as a comparison made from the learning process of a teacher who teaches a student. Several examples and ways of solving the examples are shown to the student from the teacher. The student then has to generalize the knowledge in order to use it in future examples.

Supervised learning can be divided into two different types of problems: classification and regression. Classification uses an algorithm to assign test data to in specific categories. Some common classification algorithms are random forest, k-nearest neighbor and decision-trees. Regression is instead used when the relationship between dependent and independent variables needs to be understood. Some common regression algorithms are linear regression, logistical regression and polynomial regression.

2.3.2 Unsupervised Learning

Unlike supervised learning where finding relationships between input and output is done, unsupervised learning works a little bit different. Unsupervised learning instead tries to find hidden structure in unlabeled data (Lloyd et al., 2013). There have been most success with supervised learning within machine learning. However, unsupervised learning will become more and more important in the future (Bengio, 2009).

There are three main tasks unsupervised learning models are used for: clustering, association, and dimensionally reduction. Based on the similarities and differences, the data mining technique clustering groups unlabeled data. It tries to find a structure in a collection of the unlabeled data. Two useful areas for clustering are in image compression and in market segmentation.

Unsupervised learning has some benefits from supervised learning. It can find some unknown patterns in the data which would not be considered otherwise. It can also find features which can be useful for categorization.

2.3.3 Reinforcement Learning

Reinforcement learning has increased rapidly in the machine learning communities over the last couple of years. In reinforcement learning, a computer agent is connected to its environment via perception and action (Kaelbling et al., 1996). The agent takes an action based on its control policy. The state of the agent is then updated based on the action taken by a "reward". The reward can be positive, neutral or negative, much like praising (positive) or scolding (negative) the agent based on its action. The agent is trained to achieve a certain task without an obvious guidance through the interaction with the environment. The action changes the state of the environment. One of the big advantages of reinforcement learning is that it is not trained to increase performances in the short-term. Instead, it aims to increase performances in the long run and to optimize the long-term achievement (Sugiyama, 2015). The agent learns to do this by a variety for algorithms and by systematic trial and error (Kaelbling et al., 1996).

Reinforcement learning and supervised learning differs in many ways from the more deeply studied supervised learning. In reinforcement learning, there is no presentation of input/output pairs, unlike it is in supervised learning. Some of the common disciplines studied in reinforcement learning is game theory, control theory, operations research, information theory, simulation-based optimization, multi-agent systems and statistics.

2.4 Imbalanced Data

One common problem that often occurs when working in data mining and recognition is the class imbalance problem (Yang & Wu, 2006). Having imbalanced data means that classes are not represented equally. The ratio between the classes can for example be as high as, 100:1, 1000:1 10 000:1 and sometimes even higher (Lemaître et al., 2017). When one of the classes are that much higher than the other then it causes suboptimal classification performances (Chawla et al., 2004). Most of the standard machine learning algorithms expect balanced data to be able to function correctly. The imbalanced data makes the algorithms represent the divided characteristics of the data incorrectly and provides bad accuracies across the classes of the data (He & Garcia, 2009).

Figure 3 present how one of the machines within the foundry of Scania was imbalanced. The machine had six different defect types with five of them being imbalanced and in need for balance in order to be able to use the data for training our machine learning.

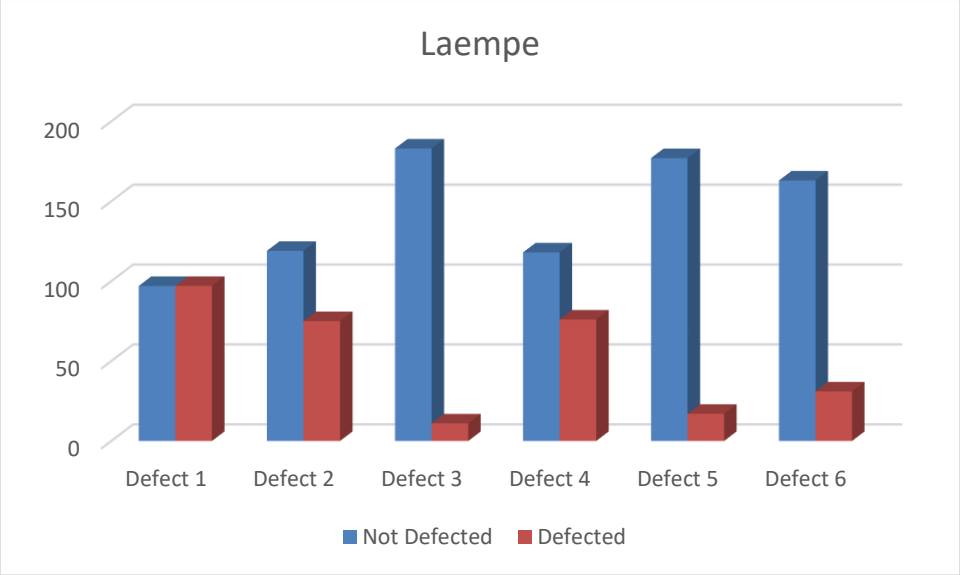

Figure 3: Imbalanced data for one of the machines

There are several different ways of solving a data imbalance problem. We use a undersampling method in this report to train our machine learning algorithm to be as accurate as possible.

2.5 Undersampling

Undersampling is one of several methods of dealing with imbalanced data. The idea of undersampling is to balance the data by filter some of the overrepresented data to have around the same number of examples for each class. Undersampling is a quite fast method when used for imbalanced data due to a lot of majority class examples are ignored. The downside of the method is some of the useful information can be within those ignored examples (Liu et al., 2008).

In our research, the amount of ok products that was produced was a lot more than the products that ended up defected. Figure 4 shows the principle undersampling method were the majority class “Ok products” gets undersampled in order to balance the data in equal amount of “Ok products” as “Defected products”. However, every undersampling method does not always need the data to be perfectly balanced as 50/50. Some methods just decreases the majority class and tries to balance it as good as possible.

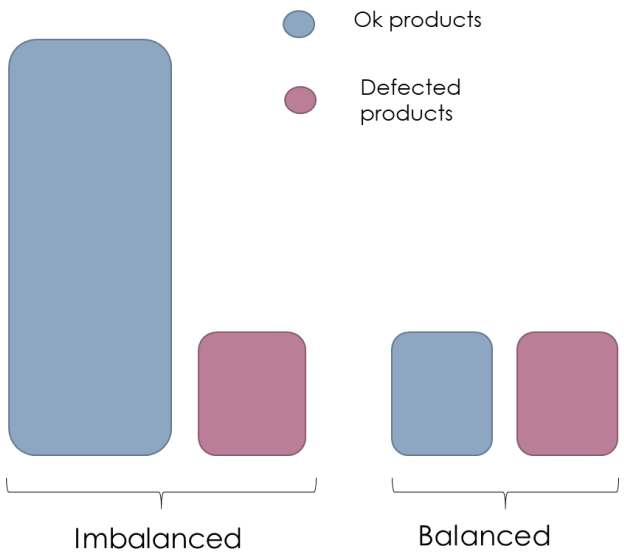

Figure 4: Undersample the imbalanced data in order to balance it

There are several different methods that can be used in undersampling. They all try to balance the data as good as possible, but they all have different approaches to achieve that goal. Because of that, there is not just one method that is superior to the others. The different methods work best for different cases. So, in order to find the best method, they all have to be tested and compared to each other.

In this research, we have tried and compared 11 different methods. Those are:

- Random Undersampling
- NearMiss version 1
- NearMiss version 2
- NearMiss version 3
- Neighborhood Cleaning Rule
- Edited Nearest Neighbor
- Repeated Edited Nearest Neighbor
- Condensed Nearest Neighbor
- Instance Hardness Threshold
- One-Sided Selection
- Tomek's Links

2.5.1 Random Undersampling

The method used in Random Undersampling is to randomly remove samples of the majority class to even the samples with the minority class (He & Ma, 2013). This method is easy to implement and also

fast to execute, which is good for very large and complex datasets. A downside with removing samples randomly is that some of the more important samples could end up being removed. The method can be used for both binary and multi-class classification problems.

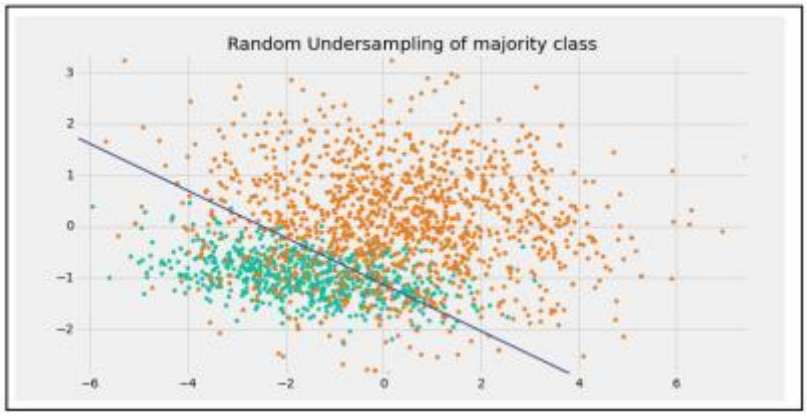

Figure 5: Random undersampling (More, 2016)

2.5.2 Undersampling based on NearMiss methods

There are three different versions of the NearMiss family. The first method being NearMiss-1 which selects examples from the majority class that are close to three of the closests examples from the minority class and removes them.

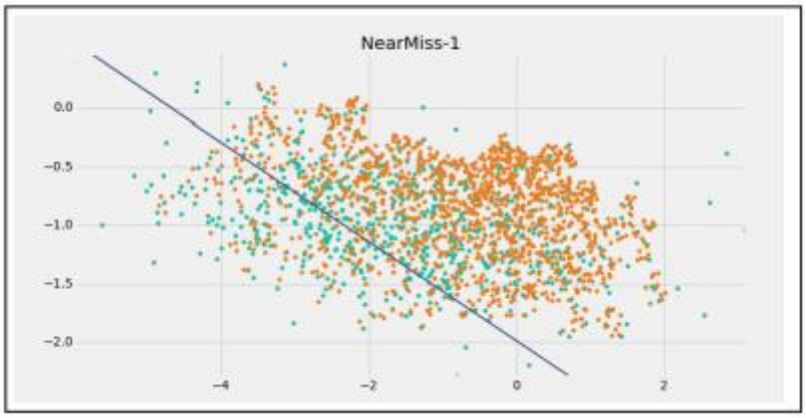

Figure 6: NearMiss-1 undersampling (More, 2016)

The NearMiss-2 method works similar to NearMiss-1. However, instead of choosing the closests examples, it selects examples from the majority class that have the smallest average distance to the three furthest examples from the minority class (Mani & Zhang, 2003).

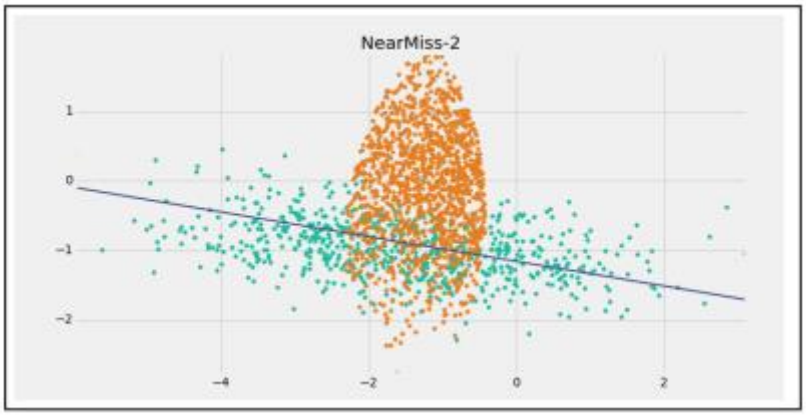

Figure 7: NearMiss-2 undersampling (More, 2016)

In the NearMiss-3 method, a given number of the closest majority examples are selected for each minority example. The strength of NearMiss-3 method is how it guarantees that every minority example is surrounded by some majority samples (Mani & Zhang, 2003).

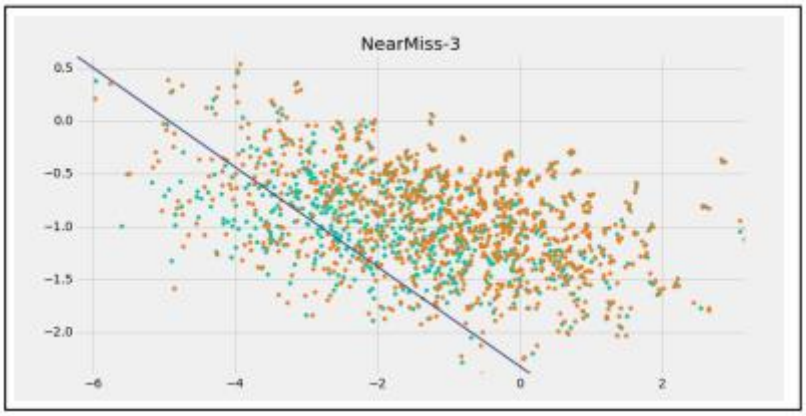

Figure 8: NearMiss-3 undersampling (More, 2016)

2.5.3 Undersampling based on the Neighborhood Cleaning Rule

The method used in Neighborhood Cleaning Rule is similar to the method used in One-Sided Selection. The difference between the two methods is mainly the Neighborhood Cleaning Rule focuses less on data reduction but instead data cleaning (Laurikkala, 2001). The method removes negatives examples which are misclassified by their three nearest neighbors. Then for the majority class the neighbors of each positive example are found and removed.

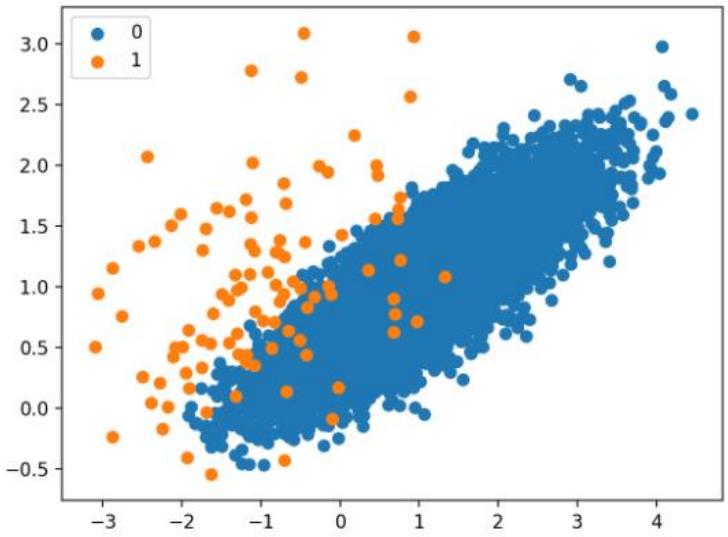

Figure 9: Neighbourhood Cleaning Rule undersampling

2.5.4 Undersampling based on the Edited Nearest Neighbor method

Wilson (1972) was the one who developed the edited nearest neighbor method. The points which class label differs from a majority of its k nearest neighbour, with the rule k=3 (More, 2016). So, if a example of a majority class instance is misclassified by its three nearest neighbors, then the example is removed from the dataset. The method also works in an alternative way. If the example is a minority class instance and is misclassified by its three nearest neighbors, then the instances among the examples neighbors from the majority class are removed (He & Ma, 2013).

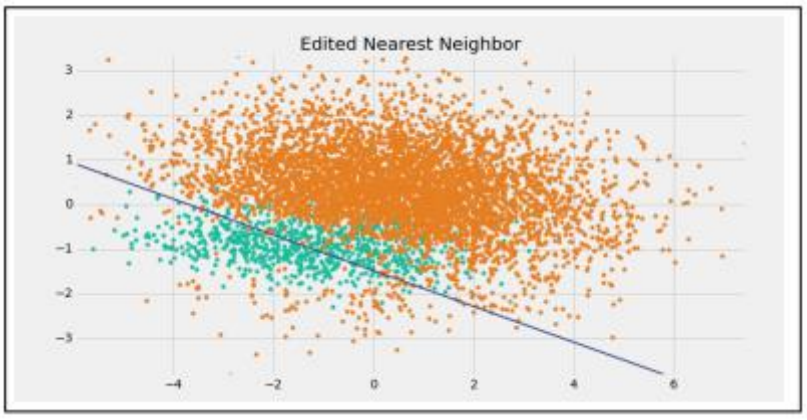

Figure 10: Edited Nearest Neighbour undersampling (More, 2016)

2.5.5 Undersampling based on the Repeated Edited Nearest Neighbor method

In the repeated edited nearest neighbor method, the algorithm from the edited nearest neighbour method is repeated until it cannot remove any further points.

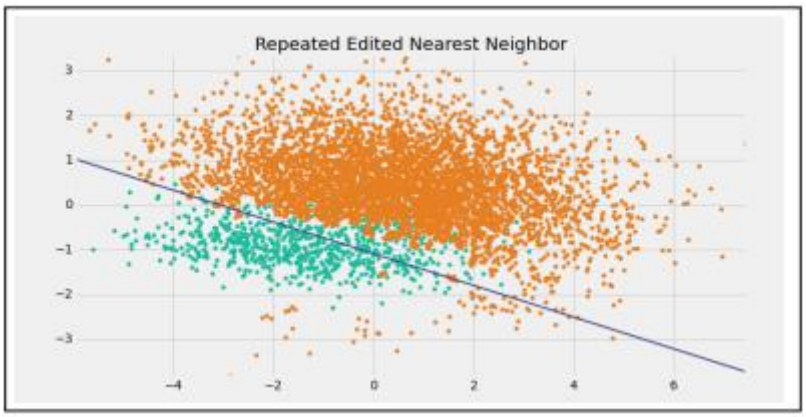

Figure 11: Repeated Edited Nearest Neighbour undersampling (More, 2016)

2.5.6 Undersampling based on the Condensed Nearest Neighbor method

In the Condensed Nearest neighbor method, the aim is to choose a subset of a collection of samples that results in no loss in model performance. The method requires many passes over the training data which can make Condensed Nearest Neighbor a slower method compared to others (More, 2016). The method works by numerating the examples in the dataset and adding them in a so called "store". However, this is done only if they cannot be classified correctly by the current contents of the "store" (Hart, 1968)

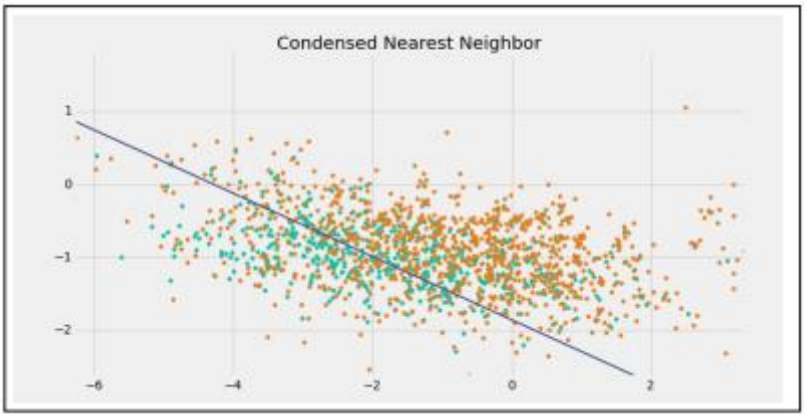

Figure 12: Condensed Nearest Neighbour undersampling (More, 2016)

2.5.7 Undersampling based on the Instance Hardness Threshold

Smith et al. (2014) presented the Instance Hardness for binary classification problems. The method used in undersampling reduce class imbalance by removing hard samples. Misclassification rate for each data is what indicates the hardness. Then data that are frequently misclassified gets eliminated, which is found by a classifier.

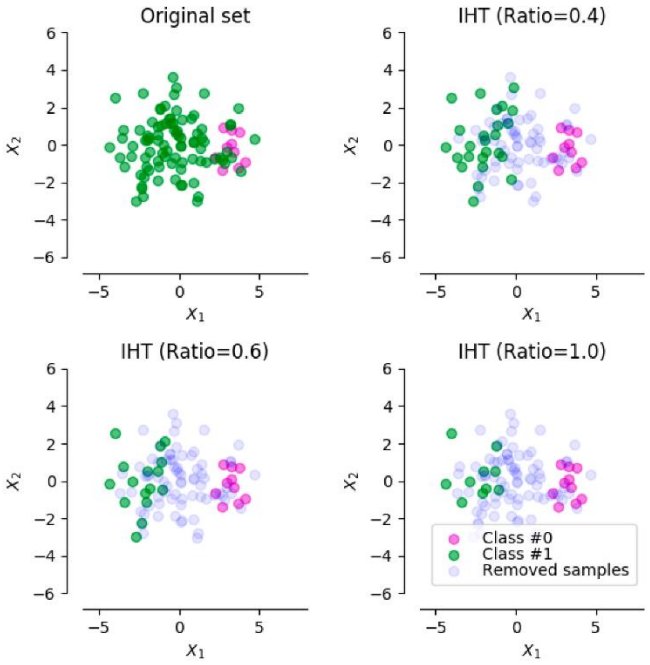

Figure 13: Instance Hardness Threshold undersampling with different ratio (Le et al., 2018)

2.5.8 Undersampling based on One-Sided Selection method

Kubat & Matwin (1997) came up with the one-sided selection. The method is a combination between Tomek's links and the condensed nearest neighbor rule. In the majority class, the Tomek Links points on the boundary are identified and removed. Then the disposable examples from the majority class that are far from the decision boundary are removed by the Condensed nearest neighbor method.

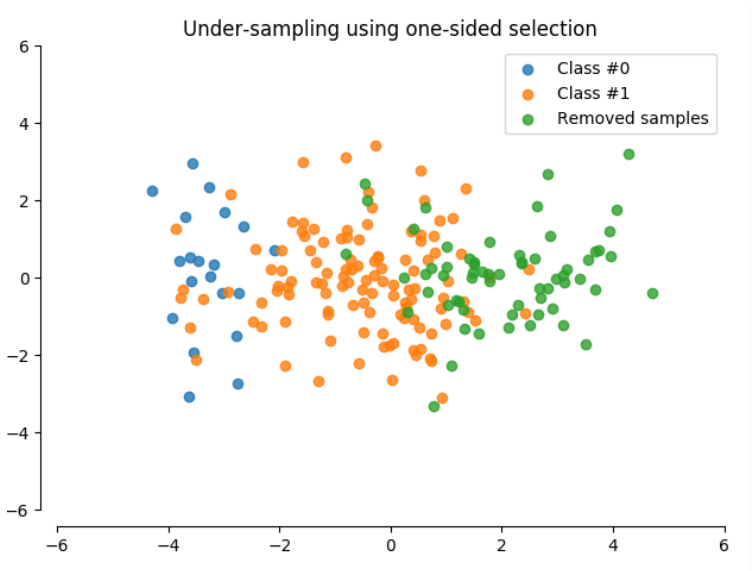

Figure 14: One-Sided Selection undersampling

2.5.9 Undersampling by removing Tomek's Links

Tomek (1976) proposed some modification to the condensed nearest neighbor method which became known as Tomek's links. If a pair are each other's nearest neighbor and belongs to different classes, then they are called Tomek link. One way of undersampling with Tomek's link is to remove all the Tomek links. The other way is to only remove the majority class samples that are part of a Tomek link (Tomek, 1976).

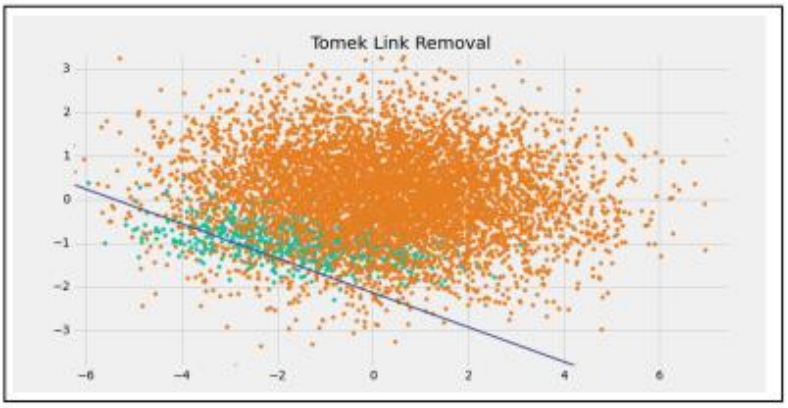

Figure 15: Tomek's Links undersampling (More, 2016)

3 Problem Description

The problem description chapter gives the reader insight on why this research was created and the problems that was faced to conduct the research. Also, the chapter goes through the problems Scania AB's foundry was having that caused some of their products to get defected in their production.

3.1 Introduction to the Problem

The idea is to use the data collected by Scania to predict the defects in the future in order to be able to lower them or to prevent them from occurring in the first place. Machine learning is one method that deals with this problem. Scania has a system for categorizing defects by their types at each stage in the casting process. However, the data is skewed due to the amount of "ok" products that is produced in opposite to how many products gets defected. There are several different methods of dealing with imbalanced data with undersampling being one. This research will be using the undersample method to solve the skewed data. Undersampling is used in many research and are an effective way of dealing with imbalanced data.

3.2 Pre-processing

One of the most important steps, which is also very time consuming, is the preprocessing of data. The preprocessing allows the data to be understandable by the machine learning algorithms. Preprocessing can be split into cleaning, normalizing, transforming, imputation of missing values, integrating, and identification of noise in data (García et al., 2016). One of the most difficult problems in inductive machine learning is the elimination of noise instances (Teng, 1999). However, in our research, the biggest problem was the missing data. Missing data can affect the statistical explanation abilities for traditional data driven modeling methods.

3.3 The Machines

Not every company uses the same machines in their foundry for casting. The set-up and process can be a little bit different from company to company. In Scanias foundry, there were three different processes which caused the most defects of their products.

- Laempe
- FCS25

- Vridkors (Swivel cross)

In these three machines, each one of them had different components or reasons that caused that particular process to produce some defected products in some of their baches. When a defected product occurred, the company observed it and recorded it. This data was recorded from January – December in 2020. Table 1 shows an example of the observed and recorded defected products for each day in January. Each of these defects where a caused from different parameters which also was recorded.

Laempe						
Trasig vattenmantel (broken water mantel)	Porös kärna (porous core)	Utstötarmärken (ejection marks)	Kärna ej fylld (core not filled)	Ohärdad kärna (uncured core)	Övriga fel (other errors)	Totalt (total errors)

Table 1: Table of the recorded defected products produced by Laempe

Table 2 shows the parameters for “Laempe”. Table 3 presents the parameters for “FCS25” and table 4 shows the parameters for “Vridkors” The status and the values of these parameters was observed and recorded three times each day, morning, afternoon and night. When one- or a combination of these parameters were bad, they could cause the machines to produce defected products. So, the parameters in table 3 had consequences that could have caused some of the defects in table 2. The values inside the green colour in table 3 indicated that the values were good. When the values would lie in the yellow zone it meant that the value was on the edge of being in the bad zone. That meant that a combination or a value close to the red zone could cause the processes to produce a defected product. When the colour was red, the value was in a bad zone and the probability of producing a defect product was then very high.

Laempe	Rigg ID		1	to	4
	Blasting pressure sand	3,2	3,54	4,56	4,9
	Gas pressure	2,5	2,7	3,3	3,5
	Gas temp	115	119	131	135
	Temp vent air	85	91	109	115
	Gas pressure in gas hood	1	1,2	1,8	2
	Sand level	15	18	27	30

Table 2: Table of parameters which caused defected products for Laempe

FCS25	Rigg ID		1	to	4
	Blasting pressure sand	3	3,2	3,4	5
	Sand probe level	23	25,8	34,2	37
	Gas starting pressure Program 1		0	0,5	0,7
	Gas pressure 2 venting pressure program 2	3,2	3,46	4,24	4,5
	Gas temp	90	92	112	115
	Sandtemp i tratt 25:an Sand temp in funnel in FC25	16	18	25	28

Table 3: Table of parameters which caused defected products for FCS25

Vridkorsmaskin	Blasting pressure sand	3,5	3,66	4,14	4,3
	Shot valv open (s)	2,5	2,7	3,3	3,5
	Venting time (s)	1,5	1,7	2,3	2,5
	Rigg ID. arm 1		1	to	6
	Rigg ID. arm 2		1	to	6
	Rigg ID. arm 3		1	to	6
	Rigg ID. arm 4		1	to	6
	Temperatur Rigg(C°) Arm 1	165	170	190	195
	Temperatur Rigg(C°) Arm 2	165	170	190	195
	Temperatur Rigg(C°) Arm 3	165	170	190	195
	Temperatur Rigg(C°) Arm 4	165	170	190	195
	Temp watercooling1 (hotplate)	12	20	30	35
	Temp vattenkylning 2 (Blow plate)	12	20	30	35
	Temp vattenkylning 3 (blow pipes)	12	20	30	35
	Gas pressure CO2	1,7	1,9	2,5	2,7
	Co2 Temp	45	50	85	100
	Sand temp funnel to "vridkors"	17	20	29	30

Table 4: Table of parameters which caused defected products for Vridkorsmaskin (swivel machine)

4 Experimental Methodology

The procedures used to conduct this research are presented under this chapter. The first part of the chapter covers how the research started and the basics, such as, what program was used and the initial steps. The chapter then continues with an evaluation over some classification that was compared. When the evaluation was done, the selection of which method was used is explained. In the end of the chapter, two very common machine learning models, random forest and artificial neuron network are explained briefly.

4.1 Data Collection and Aggregation

The data used in this research was observed and collected by Scania AB. There was some flaw in the data which needed to be sorted before the data could be used in the project. The data should had been collected during three occasions for each day, morning, afternoon, and night, for one year. Due to different circumstances, the data was not always available to be observed for those three time periods each day and was left unobserved. This led to the data was missing for some days and having gaps. So, as the second sub-objective states in the research, the need to implement procedures for correcting faulty or missing casting data was required. This was done in the programming language Python. The data was extracted to Python where a mean value for each day was calculated. When this was done, one single value of defected products for each day was recorded.

18-mar			19-mar		
Night	Morning	Afternoon	Night	Morning	Afternoon
	PDE NCG	PDE NCG		PDE NCG	PDE NCG
	ok	Ok		ok	Ok
	ok	Ok		ok	Ok
	ok	Ok		ok	Ok
	ok	Ok		ok	Ok
	84	83		83	84
	170	170		170	170
	3,6	3,6		3,6	3,8
	90	90		90	90
	52	55,2		52	56,6
	20	20		20	20
	20	20		20	20
	175	175		175	175
	175	175		175	175
	185	185		185	185
	185	185		185	185
	22	22		22	22
	175	175		175	175
	175	175		175	175
	185	185		185	185
	185	185		185	185
	23	23		22	23

Figure 16: Missing data in one of the data files

The next step in the project was to address the skewed data, which sub-objective three states in the aim and objectives chapter. When training a machine learning algorithm, it is important to have a balanced dataset. If an unbalanced dataset is used, the accuracy of the algorithm decreases significant. In this research, a method called “undersampling” was used to balance the data. This was done in Python by converting the data to binary values. The days with no defected products was labelled “False” and the days with more than one defected product was labelled “True”. The data was then balanced before applying different undersampling methods in order to evaluate the different methods against each other.

4.2 Evaluation of Classification

There is a lot of different undersample methods that can be used for balancing the data. To be able to choose one of them, they need to be evaluated to show which one works best for the data being used. It is important that the evaluation is done instead of just picking one of the methods randomly due to the methods works better for some data and less good for some of the data. The evaluation also provides insights on how good a model performs in a real-world scenario.

4.2.1 Cross Validation

In our research we use cross validation (CV) for our evaluation. CV is used for assessing prediction errors and to prevent overfitting. It evaluates a specific type of model over all available data. The data is split into different subsets which is then trained on some of these subsets. The last set is used as a test set. This procedure is then repeated until all sets have been used as a test set. The procedure has a parameter called K which is the number of groups that a data sample is split into. We used a K-fold cross validation with k = 5 in our research due to the low amount of data in some of the defect types. The dataset was then 5 times split into 5 different sets of learning and testing.

One of the primarily usage of cross validation is within machine learning and more specifically to estimate the skill of a machine learning model on unseen data. This because of estimating the performance in general when used to make prediction on data which is not used during the training of the model. The procedure is generally done:

1. Randomly shuffle the dataset
2. The data set is split into k groups
3. For each group:
    a. Take the group as a hold out or test data set

b. The remaining groups are taken as a training set
c. Fit a model on the training set and the test set is evaluated
d. Retain the evaluation score and discard the model

4. Using the sample of the evaluation scores to summarize the skill of the model

4.2.2 Data Leakage

One major issue when developing machine learning models that are supposed to predict results is data leakage (Nisbet et al., 2009). It is not uncommon that a machine learning model provides results that seem to be very good when evaluating the performance of it, but in fact just are affected by data leakage. Data leakage occurs when information outside the training dataset is used to develop the model. The information provides information to the model that it is not supposed to know. This is typically when splitting a dataset into testing and training sets. It is important to prevent any data from being shared between the two data sets due to the test set is supposed to simulate real world, unseen data. So, when a machine learning model that is affected from data leakage occurs, the evaluation of the performance is inaccurate. The leakage may occur in various steps in the machine learning process and are often hard to detect and eliminate. Three common causes of data leakage are:

1. Pre-processing
2. Duplicates
3. Temporal Data

4.2.3 Classification

In our research two classes was evaluated and converted to a binary classification. Either there where a defected product/products on one day or it was not. Whenever there was one or more defects, the data was shown as true and when there were no defects, the data was shown as false. So, the counting of the system is based on the number of *true positive* (TP), *false positive (FP)*, *true negative* (TN), *false negative* (FN).

	Positive	**Negative**
Positive	True Positive (TP)	False Positive (FP)
Negative	False Negative (FN)	True Negative (TN)

Table 5: Confusion matrix of possible outcomes

Accuracy is a very simple metric. It is a ratio of correctly predict observation to the total observations. However, it might not be the best metric suited for every task. Whenever a dataset is symmetric and have almost the same values of false positive and false negatives accuracy then it works good. However, when that is not the case, the metric can sometimes be misleading.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

F1 is a function of precision and recall. Unlike accuracy, it takes both false positive and false negatives into account. Hence, F1 is better to use if there is an uneven class distribution. A good F1 score means there is a low false positives and low false negatives.

$$F1 = \frac{TP}{TP + \frac{1}{2}(FP + FN)}$$

AUROC stands for Area Under the Receiver Operating Characteristics. AUROC shows how much the model is capable of distinguishing between classes. There are curves that are being created by varying the threshold for when a sample is classified as positive or not. The different threshold levels are recorded from the rate of the true positives and the false positive. The false positive rate is then plotted over the rate of true positives. Therefore, AUROC is very good for calculating the performance of any classification model. Figure 16 shows an example of ROC-curves.

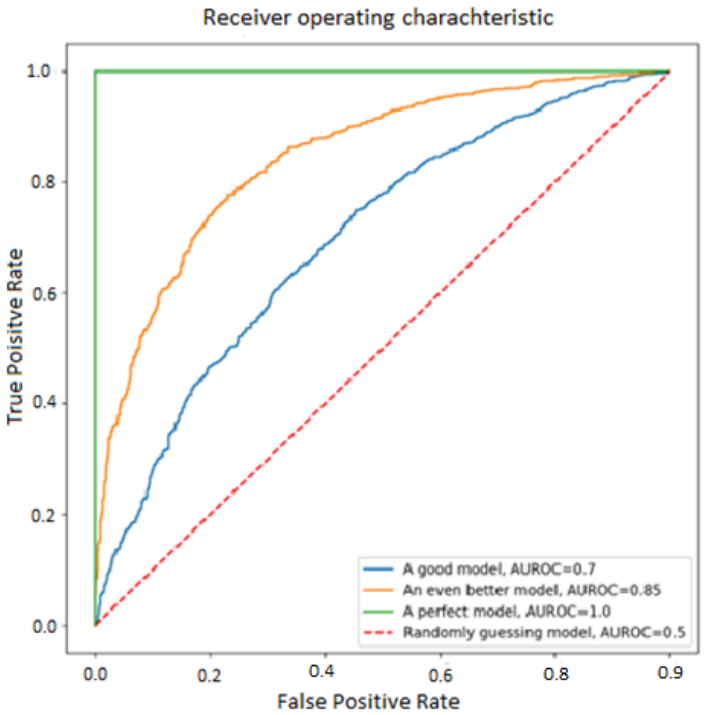

Figure 17: Example of ROC-curves (Gonen, 2007)

4.3 Machine Learning Models

To evaluate the results even further, the comparison of different ML models is done. In this research, we use two of the models to compare the results against each other to achieve the best result. Random forest and neural network are two models that is often used for prediction. They both are described briefly in this chapter.

4.3.1 Random Forest

Breiman (2001) defines random forest as following:

"A random forest is a classifier consisting of a collection of tree-structured classifiers {h(x, k), k = 1,...} where the {k } are independent identically distributed random vectors and each tree casts a unit vote for the most popular class at input x."

The random forest (RF) algorithm can be used for both classification and regression due to its simplicity and diversity. RF have multiple decision trees that are separately trained on random sampled subset of the original dataset. Each one of the trees then predict an outcome and the result is then based on the majority or mean of all the decision trees predictions. RF is often better and have a higher accuracy than a single decision tree due to the prediction of many independently taken predictions by multiple decision trees rather than just one.

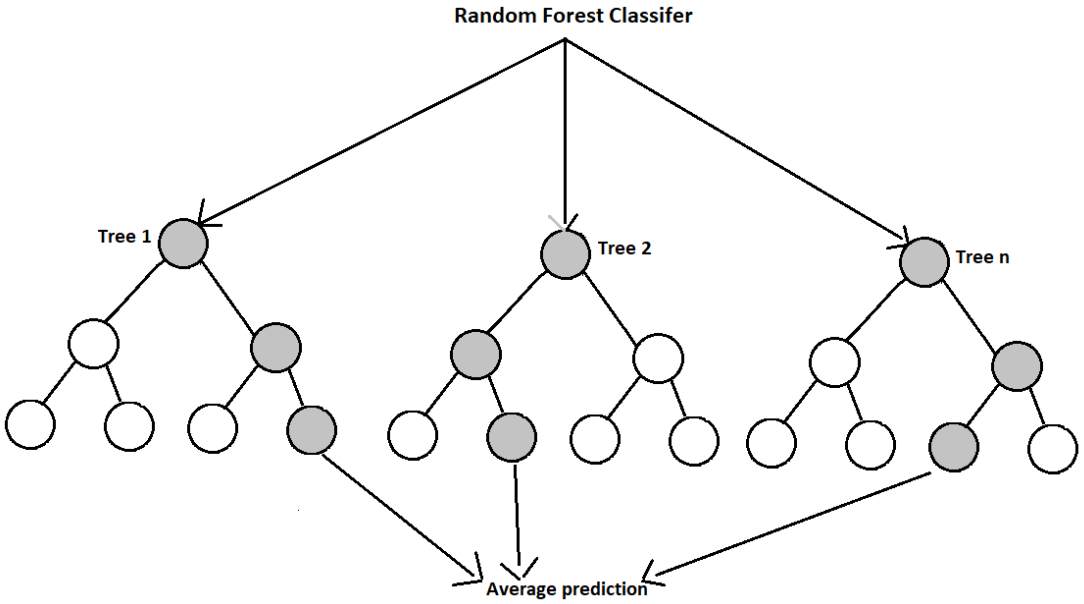

Figure 18: Random forest classifer

As for the artificial neural network, there are also many parameters to consider when implementing random forest in Python. Some of the most common being:

- **n_estimators:** The n_estimators is the number of trees in the forest with the default being "100".
- **Criterion:** Criterion is used to measure the quality of a split. There are two supported criteria, "gini" and "entropy". The criterion parameter is tree-specific and have the default setting "gini".
- **max_depth:** Max_depths is used for the maximum depth of the tree. Default setting being "none".
- **min_samples_split:** Min_samples_split is used to decide the minimum number of samples that is needed to split an internal node. The default value is "2"
- **min_sample_leaf:** Min_sample_leaf is the minimum number of samples required to be at a leaf node. The default value is "1".
- **min_weight_fraction_leaf:** Min_wight_fraction_leaf decides the minimum weighted fraction of the sum total of weights that requires to be at a leaf node. The default value is "0.0".
- **max_leaf_nodes:** Max_leaf_nodes is the max number of samples required to be at a leaf node. It uses the default value "none".
- **max_features:** The max_features is the numbers of features to consider when looking for the best split. The default value is "auto".
- **random_state:** Random_state consider the randomness of the samples used when building trees. The default being "none".
- **bootstrap:** The whole dataset is used to build each tree if bootstrap is set to false. "True" is the default value.
- **oob_score:** Oob_score is only available if bootstrap is true. The default value is "false".
- **n_jobs:** N_jobs decide how many jobs the model runs in parallel. The default value is "none", which means the model only runs 1 job.
- **random_state:** When building trees, random_state controls both the randomness of the bootstrapping of the samples used
- **verbose:** Verbose controls the verbosity when fitting and predicting. "0" is the default value of verbose.
- **warm_start:** Warm_start reuses the solution of the previous call to fit and adds more estimators to the ensemble when set to "true". "False" is the default value.

- **class_weight:** Class_weight decides the weights associated with classes in the form "class_label: weight" with the default value of "none".
- **max_samples:** Max_samples has the default value "none". If set to "none", then draw "X.shape[0]" samples.

4.3.2 Artificial Neural Network

The Artificial Neural Network (ANN) are often compared to a biological neural network of a human brain. There are several artificial neurons that is connected to each other in different layers which is independent of one another. The neurons are the core processing units of the network. The ANN takes in data and is trained to recognize the patterns in the data and then predict the outcome. Figure 18 shows the layout of a typical ANN.

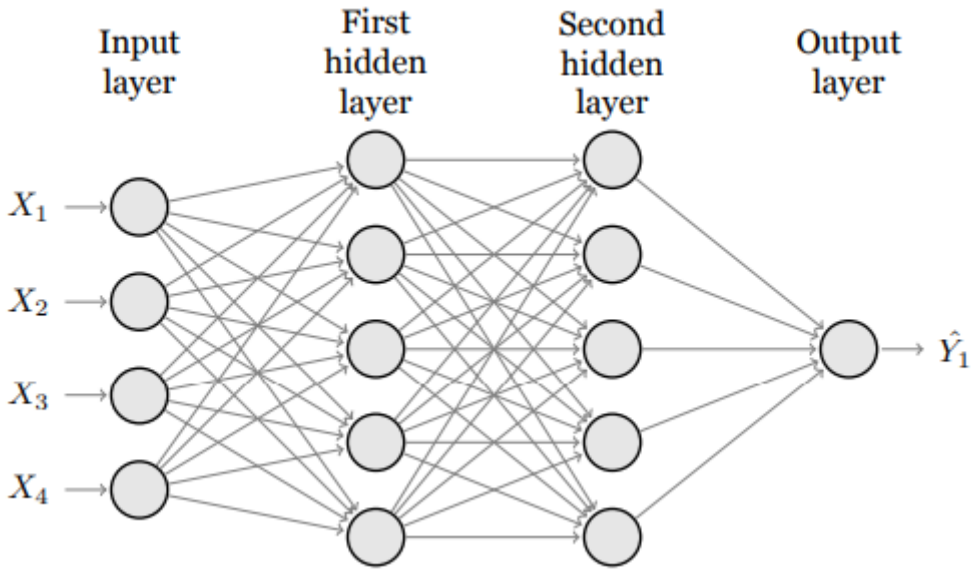

Figure 19: The layout of a typical ANN (Ståhl, 2021)

First is the input layer which receive the input. The input data continues to hidden layers through channels which connects the layers. Each of the channels are assigned a weight. The weight decides how much influence the input will have on the output. Each of the neurons in the hidden layers are associated with a numerical value which is called the bias. Bias units are not influenced by previous layers and are added to the input sum. Most of the computations are made in the hidden layers. In the last layer, the output layer, predicts the final output and the result.

There are many parameters to consider when implementing a artificial neural network in Python in order to make the model work as good as possible. Some of the most common:

- **hidden_layer_sizes:** Hidden_layer_sizes represent the number of neurons in the hidden layers, with default setting being “100” In this research, “5, 5” was used to get 5 different hidden layers with 5 neurons in each layer.
- **solver:** Solver is used for weight optimization, with the default setting being “adam”. “Adam” is a stochastic gradient-based optimizer. However, “lbfgs” was used in our research which is an optimizer in the family of quasi-Newton methods
- **alpha:** Alpha is a penalty parameter with the default setting of “0.0001” For our code, “0.00001” was used.
- **max_iters:** The maximum number of iterations is controlled by max_iters and its default being “200”. Instead of the default setting, “1000” was used in max_iters.
- **random_state:** Random_state is the random number generation for bias and weight initialization. The default being “none” but “10” was used instead.
- **activation:** Activation has the default value “relu” and activation activates the function for the hidden layer.
- **batch_size:** Batch_size decides the size of mini batches for stochastic optimizers, with default “auto”.
- **learning_rate:** Learning_rate rates the schedule for weight updates. The default value for learning_rate is “constant”.
- **learning_rate_init:** The initial learning rate used is controlled by learning_rate_init, with default value of “0.001”.
- **power_t:** The inverse scaling learning rates exponent is decided from power_t. The defualt value is “0.5”.
- **shuffle:** In each iteration, whether to shuffle samples is decided by shuffle and has the default value “true”.
- **tol:** Tol decides the tolerance for the optimization. “1e-4” is the default value. Momentum should be between 0 and 1 and it decide the gradient descent update. “0.9” is the default value.
- **momentum:** Momentum is only used when solver is set to “sgd” and it should be between 0 and 1. It is the momentum for gradient descent update.
- **epsilon:** Epsilon is only used when the solver is set to “adam”. It decides the value for numerical stability in “adam”. “1e-8” is the default value.
- **max_fun:** Max_fun is only used when the solver is set to “lbfgs” and has the default value “15000”.

4.3.3 K-NearestNeighbor

K-NeaestNeighbor (KNN) is a very simple classifier which works both for classification and regression problems. The method works by it find a predefined number of training samples that is closest in distance to the new point. It then predicts the label from these. A strength that KNN has is that the classification can be within the range of being very local or very global. Even though KNN is a very simple method, it has still been successful in many numbers of both classification and regression problems. Commonly used parameters used in KNN:

- **n_neighbors:** N_neighbors is the number of neighbors used. The default value "5", was used in our research.
- **Weights:** Weights is the weight function used in prediction. "Uniform" is the default value, and it weights all points in the neighbordhood equally.
- **Algorithm:** The algorithm used to compute the nearest neighbors has the default value "auto" and was the value used in this project.
- **leaf_size:** Leaf_size was set to the default value "30". The parameter leaf_size can affect the memory required to store the tree and the speed of the construction.
- **P:** P is the power parameter for Minkowski metric. The default is "2".
- **Metric:** To use for the tree, metric is the distance metric with "minkowski" being the default value.
- **metric_params:** Metric_params is an additional keyword argument for the metric function, which has the default value "none".
- **n_jobs:** N_jobs decides how many parallel jobs to run for neighbors to search. "None" is the default value and means 1 job.

4.3.4 Hyperparameter Tuning

There are many types of parameters to consider when designing the machine learning model which will affect the performance of the chosen model. Therefore, it is important to know what parameters are doing what and to build the model with the best possible architecture. Hyperparameter tuning or optimization basically tries different parameters to see what combinations are best to use for the model.

Choosing the approach for hyperparameter tuning can also affect the model's performance. In this research, Grid search was chosen as the approach. Grid search is the most traditional way when

hyperparameter tuning. The approach needs to be guided by some performance metric, which in the research was done by comparing accuracy, F1 and AUC.

The following hyperparameters was used for random forest:

- classify__n_estimators': [50,100]
- classify__max_features': ['auto', 'sqrt', 'log2']
- classify__max_depth': [5,10]
- classify__criterion' :['gini', 'entropy']

In artificial neuron network, the following hyperparameters was used:

- classify__hidden_layer_sizes': [(2),(4),(6),(8),(10)]
- classify__activation' : ['logistic', 'relu']
- classify__max_iter': [500, 1000]

And for K-NearestNeighbor:

- classify__n_neighbors': [2,3,4,5]
- classify__weights': ['uniform', 'distance']

5 Results

The result for the evaluation of each machine is presented in this chapter. Every machine and every defect type in each machine were evaluated with the best undersampling method and with the best machine learning method. The different defect types were evaluated in three different tables with F1-score being the main error metric used for the evaluation. First table present the score of only cross validation applied. Second table present the result with applying default parameters after the cross validation. Third and last table presents the end result with applying hyperparameter tuning after the cross validation and default parameters. These three steps were made for all the machine learning methods presented in the report in order to see the improvement of each step.

Defect type “Trasig Vattenmantel” for the machine “Laempe” will show the tables with the three different steps, cross validation score, default parameters and then the tuned CV grid search. The remaining defect types will be summarized in the improved third step (CV grid search) for each machine with the best undersampling method and the best machine learning model. All the hyperparameter tuned tables used in this research (the tables presented in this chapter included) are presented in the appendix. This was done to not overflow the chapter with tables to confuse the reader. The rows/cells covered in green were the values that performed the best.

5.1 Result for Laempe

Table 6 shows the result of “Trasig Vattenmatel” for “Laempe”. The tables also provide insight on the procedure how every machine and defect type was evaluated with default parameters and grid CV search. Every defect type was also tested with ANN and KNN in the same way as for random forest.

Test Score With RF (Default Parameters)				Grid CV Search With RF			
Method	**Accuracy**	**F1**	**AUROC**	**Method**	**Accuracy**	**F1**	**AUROC**
NM_v1	0.5567	0.5567	0.5568	NM_v1	0.5876	0.5918	0.5876
NM_v2	0.6392	0.6465	0.6390	NM_v2	0.5670	0.5532	0.5674
NM_v3	0.6185	0.6105	0.6189	NM_v3	0.5876	0.5652	0.5882
RUS	0.6082	0.6274	0.6078	RUS	0.6082	0.6346	0.6076
ENN	0.5773	0.6771	0.5741	ENN	0.5876	0.6825	0.5846
CNN	0.5154	0.6240	0.5125	CNN	0.5361	0.6511	0.5327
RENN	0.5773	0.6772	0.5742	RENN	0.5876	0.6825	0.5846
Tomek	0.6185	0.6542	0.6175	Tomek	0.5979	0.6286	0.5971
OSS	0.6495	0.6909	0.6482	OSS	0.6185	0.6542	0.6175
NCR	0.5773	0.6772	0.5742	NCR	0.5567	0.6560	0.5538

Table 6: Result for defect type Trasig Vattenmantel with random forest evaluated with default parameters & with grid CV search

Table 7 presents the evaluation for which undersampling method performed best in machine "Laempe". NearMiss version 1 and NearMiss version 2 outperformed the remaining undersampling methods. Both of them won in five occasions in the evaluation.

Laempe	RF	ANN	KNN
Defect 1	ENN/RENN	All	NCR
Defect 2	TOMEK	NM/RUS/TOMEK/OSS	RUS
Defect 3	NM_v3	NM_v1	NM_v1
Defect 4	NCR	NM/RUS/ENN/CNN/RENN	NCR
Defect 5	RUS	NM_V2	NM_v1
Defect 6	NM_v2	NM_v2	NM_v3

Table 7: Evaluation of undersampling methods for Laempe

The different machine learning methods was also evaluated for machine "Laempe". Table 8 shows how Artificial Neuron Network performed the best in most of the defect types.

Laempe	NM_v1	NM_v2	NM_v3	RUS	ENN	CNN	RENN	Tomek	OSS	NCR
Defect 1	ANN	ANN	ANN	ANN	KNN	ANN	RF	ANN	ANN	KNN
Defect 2	ANN	ANN	ANN	ANN	ANN	KNN	KNN	ANN	ANN	ANN
Defect 3	ANN	ANN	RF	ANN	RF	ANN	RF	RF	RF	RF
Defect 4	ANN	ANN	ANN	ANN	ANN	ANN	ANN	KNN	KNN	RF
Defect 5	KNN	KNN	ANN	KNN	KNN	RF	KNN	KNN	KNN	KNN
Defect 6	ANN	ANN	ANN	ANN	ANN	ANN	ANN	KNN	ANN	ANN

Table 8: Evaluation of machine learning methods for Laempe

5.2 Result for FCS25

For machine "FCS25", Random Undersampling and Tomek's links were the undersampling methods who had the best overall score. Both of them won six times when comparing the undersampling methods to each other. NearMiss version 1 and NearMiss version 2 also performed well but not as good as Random Undersampling and Tomek's links.

FCS25	RF	ANN	KNN
Defect 1	NM_v1	NM_v1	RUS
Defect 2	NM_v2	NM_v3	NM_v1
Defect 3	RUS	RUS/ENN/CNN/RENN/Tomek/OSS/NCR	All
Defect 4	NM_v2	NM_v2	ENN
Defect 5	NM_v3	RUS	NM_v2
Defect 6	CNN	RENN	RUS
Defect 7	Tomek	Tomek/OSS	NM_v1
Defect 8	OSS	RENN/Tomek/OSS/NCR	Tomek

Table 9: Evaluation of undersampling methods for FCS25

In the comparison between machine learning methods for the machine "FCS25", Artificial Neuron Network was the one who performed best. However, K-NearestNeighbor came in second place with ANN just outperforming KNN.

FCS25	RUS	NM_v1	NM_v2	NM_v3	ENN	CNN	RENN	Tomek	OSS	NCR
Defect 1	KNN	RF	KNN	ANN	KNN	ANN	KNN	ANN	KNN	KNN
Defect 2	RF	ANN	ANN	ANN	KNN	KNN	KNN	KNN	RF	KNN
Defect 3	RF	KNN	KNN	KNN	All	All	All	All	All	All
Defect 4	ANN	KNN	ANN	RF	KNN	RF	RF	RF	RF	RF
Defect 5	KNN	ANN	KNN	RF	RF/KNN	KNN	All	RF/KNN	RF/KNN	RF/KNN
Defect 6	ANN	ANN	ANN	ANN	RF	RF	ANN	KNN	KNN	ANN
Defect 7	ANN	ANN	ANN	ANN	RF	ANN	RF	ANN	ANN	ANN
Defect8	KNN	ANN	ANN	ANN	ANN	KNN	ANN	ANN	ANN	ANN

Table 10: Evaluation of machine learning methods for FCS25

5.3 Result for Vridkors

NearMiss version 2 did perform best in the evaluation between undersampling methods for the machine "Vridkors". Tomek's Links was the method who scored the best results after NearMiss version 2.

Vridkors	RF	ANN	KNN
Defect 1	Tomek	Tomek	OSS
Defect 2	Tomek	Tomek/OSS	OSS
Defect 3	OSS	OSS	Tomek
Defect 4	NM_v1	NM_v2	NM_v3
Defect 5	RUS	NCR	NCR
Defect 6	NM_v3	NM_v1	NM_v3
Defect 7	NM_v2	NM_v2	NM_v1
Defect 8	RENN	RENN	NM_v2
Defect 9	NM_v2	NM_v3	CNN
Defect 10	NM_v2	NM_v1	NM_v3
Defect 11	NM_v2	NM_v2	NM_v1
Defect 12	ENN/RENN/NCR	NM_v2	NCR
Defect 13	ENN/CNN/RENN/Tomek/OSS/NCR	NCR	ENN/CNN/RENN/Tomek/OSS/NCR

Table 11: Evaluation of undersampling methods for Vridkors

Random forest was the clear winner in the evaluation of machine learning methods for "Vridkors". As can be seen in table 12, random forest was almost performed best in more than half of the defect types. K-NearestNeighbor and Artificial Neuron Network performed similar to each other.

Vridkors	RUS	NM_v1	NM_v2	NM_v3	ENN	CNN	RENN	Tomek	OSS	NCR
Defect 1	ANN	ANN	RF	ANN	ANN	ANN	ANN	ANN	ANN	RF
Defect 2	KNN	ANN	RF	RF	RF	ANN	RF	RF	RF	RF
Defect 3	RF	ANN	RF	KNN	KNN	RF	KNN	KNN	RF	KNN
Defect 4	ANN	RF	ANN	ANN	ANN	ANN	ANN	RF	RF	ANN
Defect 5	RF	KNN	RF	RF	KNN	RF	KNN	RF	RF	RF
Defect 6	RF	ANN	RF	RF	RF	RF	RF	RF	KNN	RF
Defect 7	KNN	RF	ANN	ANN	RF/KNN	KNN	KNN	RF/KNN	RF/KNN	RF
Defect 8	KNN	ANN	KNN	KNN	ANN	RF	KNN	RF	RF	RF
Defect 9	KNN	RF	RF	ANN	KNN	KNN	RF	KNN	KNN	KNN
Defect 10	KNN	RF	RF	KNN	RF/KNN	RF/KNN	RF	RF/KNN	RF/KNN/ANN	RF/KNN
Defect 11	ANN	ANN	RF	ANN	RF/ANN	ANN	ANN	RF/ANN	RF/ANN	RF
Defect 12	RF	RF	RF	RF	RF	RF/KNN	RF	KNN	KNN	KNN
Defect 13	KNN	RF	KNN	RF	RF/KNN	RF/KNN	RF/KNN	RF/KNN	RF/KNN	RF/KNN

Table 12: Evaluation of machine learning methods for Vridkors

5.4 Undersample Method Evaluation

Table 13 present the result of the undersampling method that performed best for most of the defect types in each of the machines. NearMiss version 2 came out as the winning method in two of the three

machines. For the third machine, “FCS25”, Random Undersampling and Tomek’s Links equally performed best in most of the defect types.

Machine	Undersampling Method
Laempe	NM_v1/NM_v2
FCS25	RUS/Tomek
Vridkors	NM_v2

Table 13: Result of the best undersampling method for each machine

5.5 Machine Learning Method Evaluation

The evaluation for the machine learning methods was done in a similar way as for the undersampling methods. The best performing machine learning method was recorded for each undersampling method in all of the three machines. Next step was to see how many of the machine learning methods had the overall best result for each machine (table 14). Artificial Neural Network came out in the top with having the best overall performance for “Laempe” and for “FCS25”. Random forest had the best overall performance for “Vridkors”.

Machine	Undersampling Method
Laempe	NM_v1/NM_v2
FCS25	RUS/Tomek
Vridkors	NM_v2

Table 14: Result of the best machine learning method for each machine

6 Conclusions

The purpose of this chapter is to summarize and discuss the results. The research questions are answered and some final conclusions regarding the aim are covered. The chapter ends with some suggestions about future work.

6.1 Summary of Results

The aim of this research was to build a machine learning model and evaluate the best prediction model that could predict casting defects. With the help of the programming language Python, we could achieve these sub-objectives and build our machine learning model. The problem of having multi-

source, multi-format data was solved by extracting all the data to Python, where we also could address the fact that the data was faulty or missing in some parts.

We have also compared ten different undersampling methods to evaluate which method would have the most success addressing our skewed data. As can be seen in the result, there is not always one best method to use for every task. The best performed undersample method varied in the research between the different machines and defect types. We can therefore conclude that the best undersample method varies depending on many variables, such as the data and how it is used. However, if we were to crown one method that had best performances most times, then it would be NearMiss version 2. It performed best in two of the three machines and was the second-best method in the third.

The same thing can be said when evaluating the best machine learning method. There are many variables that affect the performance of the method and different data will provide different results. Artificial Neural Network was the machine learning method that had the most success in our research. It performed best in two of the three machines. Random forest was the method which was most successful in the other machine. However, K-NearestNeighbor should not be labelled as a bad machine learning method just because it did not win in any of these evaluations. The method did provide some good results and was close to outperforming the other two methods in of the machines.

6.2 Future Work

In this chapter, we suggest some changes or improvements that could be done in order to improve the research and its result.

6.2.1 Changing the threshold

There are several possible changes or improvements that could be done for any future work which could result in better results for the research. When conducting this research, the defect limitation was set to “>0”, which means that every defective unit was considered defected. Even though this is true, one could argue that 1 defective product could had been caused by other reasons that had nothing to do with the process parameters. Random defective causes sometimes occurs that can not be predicted with our model. So, changing the threshold for what is considered as a defective day or not could be changed to something greater than “>0”. That would help take the random defectives in consideration.

6.2.2 Treating the problem as a regression problem

In our research, we look at the problem as a classification problem. That means we are using a predictive model where a class label is predicted for a given example of input data. However, this research could also be seen as a regression problem. A regression problem looks to the relationship between dependant and independent variables with one or more independent variables.

6.2.3 Treating the problem as a multi-task classification problem

Another interesting approach would be to treat the problem like a multi-class classification problem. Our research has been looking at the problem as a binary classification problem. Hence, we sorted the data as either a defective- or a not defective product. A multi-class classification approach uses more than two classes. Using a multi-class classification approach could sort the data as a defective product, not a defective product and as a suggestion: a random defective product which hasn't been caused by any process parameters.

6.2.4 Large-scale parameter tuning

Hyperparameter tuning is a great tool in machine learning. It helps by allowing the model to try out different architecture in order to find the best suited parameters for the model. In our research, we used a few hyperparameters for each classifier to increase our results. This could be increased to a large-scale parameter tuning in order to increase the results even further. However, when tuning more hyperparameters often results in slower tuning process.

6.2.5 Improving data collection

One important part when conducting a research like this, is the consistency of the data collection. As been mentioned in our report, the data collection was not always collected three times a day it should had been. Even though we used methods in order to fix this problem, the accuracy of the model still gets affected from it. So, for future work, a good way to improve the accuracy would be to more precise with the collection of the data. The data collection could also be improved by measuring the process parameters multiple times a day. More measurements of the process parameters would secure a higher accuracy of the model.

7 References

Aggarwal, C. C. (2015). *Data mining: the textbook*. Springer.

Ammar, H. R., Samuel, A. M., & Samuel, F. H. (2008). Porosity and the fatigue behavior of hypoeutectic and hypereutectic aluminum–silicon casting alloys. *International journal of Fatigue*, *30*(6), 1024-1035.

Bengio, Y. (2009). Learning Deep Architectures for AI. Foundations Trends Machine Learning, vol. 2 (1).

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

Chawla, N. V., Japkowicz, N., & Kotcz, A. (2004). Special issue on learning from imbalanced data sets. *ACM SIGKDD explorations newsletter*, *6*(1), 1-6.

Condori-Fernandez, N., Wieringa, R. J., Daneva, M., Mutschler, B. B., & Pastor, O. (2012). An experimental evaluation of a unified checklist for designing and reporting empirical research in software engineering.

Creswell, J.W., Research Design: Qualitative & Quantitative Approaches, London: *SAGE Publications*, 1994

Fayyad, U., Piatetsky-Shapiro, G., & Smyth, P. (1996). From data mining to knowledge discovery in databases. *AI magazine*, *17*(3), 37-37.

Ferguson, M., Ak, R., Lee, Y. T. T., & Law, K. H. (2017, December). Automatic localization of casting defects with convolutional neural networks. In *2017 IEEE international conference on big data (big data)* (pp. 1726-1735). IEEE.

Ferguson, M. K., Ronay, A. K., Lee, Y. T. T., & Law, K. H. (2018). Detection and segmentation of manufacturing defects with convolutional neural networks and transfer learning. *Smart and sustainable manufacturing systems*, *2*.

Fink, A. (2019). *Conducting research literature reviews: From the internet to paper*. Sage publications.

García, S., Ramírez-Gallego, S., Luengo, J., Benítez, J. M., & Herrera, F. (2016). Big data preprocessing: methods and prospects. *Big Data Analytics*, *1*(1), 1-22.

Gayer, A., Saya, A., & Shiloh, A. (1990). Automatic recognition of welding defects in real-time radiography. *Ndt International*, *23*(3), 131-136.

Gonen, M. (2007). *Analyzing receiver operating characteristic curves with SAS*. SAS publishing.

Hart, P. (1968). The condensed nearest neighbor rule (corresp.). *IEEE transactions on information theory*, *14*(3), 515-516.

He, H., & Garcia, E. A. (2009). Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, *21*(9), 1263-1284.

He, H., & Ma, Y. (Eds.). (2013). Imbalanced learning: foundations, algorithms, and applications.

Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of artificial intelligence research*, *4*, 237-285

Kubat, M., & Matwin, S. (1997, July). Addressing the curse of imbalanced training sets: one-sided selection. In *Icml* (Vol. 97, pp. 179-186).

Laurikkala, J. (2001, July). Improving identification of difficult small classes by balancing class distribution. In *Conference on Artificial Intelligence in Medicine in Europe* (pp. 63-66). Springer, Berlin, Heidelberg.

Le, T., Le Son, H., Vo, M. T., Lee, M. Y., & Baik, S. W. (2018). A cluster-based boosting algorithm for bankruptcy prediction in a highly imbalanced dataset. *Symmetry*, *10*(7), 250.

Lee, J., Noh, S. D., Kim, H. J., & Kang, Y. S. (2018). Implementation of cyber-physical production systems for quality prediction and operation control in metal casting. *Sensors*, *18*(5), 1428.

Lemaître, G., Nogueira, F., & Aridas, C. K. (2017). Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *The Journal of Machine Learning Research*, *18*(1), 559-563.

Lin, J., Yao, Y., Ma, L., & Wang, Y. (2018). Detection of a casting defect tracked by deep convolution neural network. *The International Journal of Advanced Manufacturing Technology*, *97*(1), 573-581.

Liu, X. Y., Wu, J., & Zhou, Z. H. (2008). Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *39*(2), 539-550.

Lloyd, S., Mohseni, M., & Rebentrost, P. (2013). Quantum algorithms for supervised and unsupervised machine learning. *arXiv preprint arXiv:1307.0411*.

Mair, C., Kadoda, G., Lefley, M., Phalp, K., Schofield, C., Shepperd, M., & Webster, S. (2000). An investigation of machine learning based prediction systems. *Journal of systems and software*, *53*(1), 23-29.

Mani, I., & Zhang, I. (2003, August). kNN approach to unbalanced data distributions: a case study involving information extraction. In *Proceedings of workshop on learning from imbalanced datasets* (Vol. 126). United States: ICML.

Mannila, H. (1996, June). Data mining: machine learning, statistics, and databases. In *Proceedings of 8th International Conference on Scientific and Statistical Data Base Management* (pp. 2-9). IEEE.

Masad, E., Jandhyala, V. K., Dasgupta, N., Somadevan, N., & Shashidhar, N. (2002). Characterization of air void distribution in asphalt mixes using X-ray computed tomography. *Journal of materials in civil engineering*, *14*(2), 122-129.

More, A. (2016). Survey of resampling techniques for improving classification performance in unbalanced datasets. *arXiv preprint arXiv:1608.06048*.

Nisbet, R., Elder, J., & Miner, G. (2009). *Handbook of statistical analysis and data mining applications*. Academic press.

Obermeyer, Z., & Emanuel, E. J. (2016). Predicting the future—big data, machine learning, and clinical medicine. *The New England journal of medicine*, *375*(13), 1216.

Pastor-López, I., Santos, I., Santamaría-Ibirika, A., Salazar, M., De-la-Peña-Sordo, J., & Bringas, P. G. (2012, July). Machine-learning-based surface defect detection and categorisation in high-precision foundry. In *2012 7th IEEE Conference on Industrial Electronics and Applications (ICIEA)* (pp. 1359-1364). IEEE.

Robinson, C. J., & Malhotra, M. K. (2005). Defining the concept of supply chain quality management and its relevance to academic and industrial practice. *International journal of production economics*, *96*(3), 315-337.

Skallerud, B., Iveland, T., & Härkegård, G. (1993). Fatigue life assessment of aluminum alloys with casting defects. *Engineering Fracture Mechanics*, *44*(6), 857-874.

Smith, M. R., Martinez, T., & Giraud-Carrier, C. (2014). An instance level analysis of data complexity. *Machine learning*, *95*(2), 225-256.

Ståhl, N. (2021). *Integrating domain knowledge into deep learning: Increasing model performance through human expertise* (Doctoral dissertation, Högskolan i Skövde).

Sugiyama, M. (2015). *Statistical reinforcement learning: modern machine learning approaches*. CRC Press.

Teng, C. M. (1999, June). Correcting Noisy Data. In *ICML* (pp. 239-248).

Tomek, I. (1976). Two modifications of CNN.

Wang, S., Wan, J., Zhang, D., Li, D., & Zhang, C. (2016). Towards smart factory for industry 4.0: a self-organized multi-agent system with big data based feedback and coordination. *Computer Networks*, *101*, 158-168.

Wieringa, R. J. (2014). *Design science methodology for information systems and software engineering*. Springer.

Wilson, D. L. (1972). Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man, and Cybernetics*, (3), 408-421.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., & Wesslén, A. (2012). *Experimentation in software engineering*. Springer Science & Business Media.

Yang, Q., & Wu, X. (2006). 10 challenging problems in data mining research. *International Journal of Information Technology & Decision Making*, *5*(04), 597-604.

Appendix

Appendix A: Results with Hyperparameter Tuning for Laempe

Laempe			
Trasig Vattenmantel			
Grid CV Search With RF			
Method	**Accuracy**	**F1**	**AUROC**
NM_v1	0.5876	0.5918	0.5875
NM_v2	0.5670	0.5531	0.5673
NM_v3	0.5876	0.5652	0.5882
RUS	0.6082	0.6346	0.6075
ENN	0.5876	0.6825	0.5846
CNN	0.5360	0.6511	0.5327
RENN	0.5876	0.6825	0.5846
Tomek	0.5979	0.6285	0.5971
OSS	0.6185	0.6542	0.6175
NCR	0.5567	0.656	0.5537

Table 15: Laempe grid CV search on vattenmantel with rf

Laempe			
Trasig Vattenmantel			
Grid CV Search With ANN			
Method	**Accuracy**	**F1**	**AUROC**
NM_v1	0.5051	0.6712	0.5
NM_v2	0.5051	0.6712	0.5
NM_v3	0.5051	0.6712	0.5
RUS	0.5051	0.6712	0.5
ENN	0.5051	0.6712	0.5
CNN	0.5051	0.6712	0.5
RENN	0.5051	0.6712	0.5
Tomek	0.5051	0.6712	0.5
OSS	0.5051	0.6712	0.5
NCR	0.5051	0.6712	0.5

Table 16: Laempe grid CV search on vattenmantel with ann

Laempe			
Trasig Vattenmantel			
Grid CV Search With KNN			
Method	**Accuracy**	**F1**	**AUROC**
NM_v1	0.5463	0.5686	0.5459
NM_v2	0.5360	0.5544	0.5357
NM_v3	0.5360	0.5544	0.5357
RUS	0.5257	0.5660	0.5248
ENN	0.6082	0.6935	0.6054
CNN	0.5154	0.6178	0.5127
RENN	0.5876	0.6666	0.5852
Tomek	0.5670	0.6181	0.5656
OSS	0.5670	0.6181	0.5656
NCR	0.5979	0.6829	0.5952

Table 17: Laempe grid CV search on vattenmantel with knn

Laempe			
Porös kärna			
Grid CV Search With RF			
Method	**Accuracy**	**F1**	**AUROC**
NM_v1	0.5360	0.5871	0.5369
NM_v2	0.4948	0.5333	0.5036
NM_v3	0.5257	0.5660	0.5337
RUS	0.5360	0.5945	0.5317
ENN	0.4742	0.3376	0.5542
CNN	0.4742	0.4950	0.4921
RENN	0.4742	0.3376	0.5542
Tomek	0.6082	0.6984	0.5693
OSS	0.5979	0.6829	0.5662
NCR	0.5051	0.4782	0.5481

Table 18: Laempe grid CV search on porös kärna with rf

Laempe			
Porös kärna			
Grid CV Search With ANN			
Method	**Accuracy**	**F1**	**AUROC**
NM_v1	0.6185	0.7643	0.5
NM_v2	0.6185	0.7643	0.5
NM_v3	0.6185	0.7643	0.5
RUS	0.6185	0.7643	0.5
ENN	0.5360	0.4943	0.5887
CNN	0.3917	0.2337	0.4668
RENN	0.5360	0.4943	0.5887
Tomek	0.6185	0.7643	0.5
OSS	0.6185	0.7643	0.5
NCR	0.5051	0.6307	0.4497

Table 19: Laempe grid CV search on porös kärna with ann

Laempe			
Porös kärna			
Grid CV Search With KNN			
Method	**Accuracy**	**F1**	**AUROC**
NM_v1	0.4123	0.4356	0.4265
NM_v2	0.5051	0.5471	0.5119
NM_v3	0.4845	0.5370	0.4849
RUS	0.6082	0.6607	0.6056
ENN	0.4432	0.3571	0.5033
CNN	0.4639	0.5094	0.4682
RENN	0.4432	0.3571	0.5033
Tomek	0.5979	0.6548	0.5921
OSS	0.5876	0.6428	0.5837
NCR	0.4845	0.4680	0.5211

Table 20: Laempe grid CV search on porös kärna with knn

Laempe			
Utstötamärken			
Grid CV Search With RF			
Method	**Accuracy**	**F1**	**AUROC**
NM_v1	0.5360	0.0816	0.4413
NM_v2	0.4432	0.0999	0.4697
NM_v3	0.6391	0.1025	0.4963
RUS	0.6288	0.0526	0.4130
ENN	0.9381	0.0	0.5
CNN	0.8969	0.0	0.4780
RENN	0.9381	0.0	0.5
Tomek	0.9381	0.0	0.5
OSS	0.9381	0.0	0.5
NCR	0.9381	0.0	0.5

Table 21: Laempe grid CV search on utstötamärken with rf

Laempe			
Utstötarmärken			
Grid CV Search With ANN			
Method	**Accuracy**	**F1**	**AUROC**
NM_v1	0.5979	0.1702	0.6300
NM_v2	0.5979	0.1333	0.5521
NM_v3	0.6494	0.0555	0.4239
RUS	0.2474	0.0987	0.4432
ENN	0.8762	0.0	0.4670
CNN	0.9278	0.0	0.4945
RENN	0.8762	0.0	0.4670
Tomek	0.9381	0.0	0.5
OSS	0.8969	0.0	0.4780
NCR	0.8865	0.0	0.4725

Table 22: Laempe grid CV search on utstötamärken with ann

Laempe			
Utstötamärken			
Grid CV Search With KNN			
Method	**Accuracy**	**F1**	**AUROC**
NM_v1	0.6082	0.1363	0.5576
NM_v2	0.6907	0.1176	0.5238
NM_v3	0.7010	0.0	0.3736
RUS	0.3711	0.0615	0.3534
ENN	0.8041	0.0	0.4285
CNN	0.9278	0.0	0.4945
RENN	0.8041	0.0	0.4285
Tomek	0.8453	0.0	0.4505
OSS	0.8453	0.0	0.4505
NCR	0.8144	0.0	0.4340

Table 23: Laempe grid CV search on utstötamärken with knn

Laempe			
Kärna ej fylld			
Grid CV Search With RF			
Method	**Accuracy**	**F1**	**AUROC**
NM_v1	0.4536	0.4044	0.4571
NM_v2	0.5154	0.4197	0.5033
NM_v3	0.4639	0.4347	0.4750
RUS	0.4845	0.4565	0.4966
ENN	0.4020	0.5468	0.4944
CNN	0.5154	0.5252	0.5454
RENN	0.4020	0.5468	0.4944
Tomek	0.4329	0.3956	0.4402
OSS	0.4536	0.4044	0.4571
NCR	0.4329	0.5669	0.5245

Table 24: Laempe grid CV search on kärna ej fylld with rf

Laempe			
Kärna ej fylld			
Grid CV Search With ANN			
Method	**Accuracy**	**F1**	**AUROC**
NM_v1	0.3917	0.5629	0.5
NM_v2	0.3917	0.5629	0.5
NM_v3	0.3917	0.5629	0.5
RUS	0.3917	0.5629	0.5
ENN	0.3917	0.5629	0.5
CNN	0.3917	0.5629	0.5
RENN	0.3917	0.5629	0.5
Tomek	0.4536	0.2739	0.4197
OSS	0.4536	0.2739	0.4197
NCR	0.3917	0.5629	0.5

Table 25: Laempe grid CV search on kärna ej fylld with ann

Laempe			
Kärna ej fylld			
Grid CV Search With KNN			
Method	**Accuracy**	**F1**	**AUROC**
NM_v1	0.5567	0.5376	0.5747
NM_v2	0.5567	0.5274	0.5700
NM_v3	0.4536	0.4175	0.4618
RUS	0.5154	0.4946	0.5314
ENN	0.4329	0.4954	0.4823
CNN	0.4948	0.5050	0.5238
RENN	0.4329	0.4954	0.4823
Tomek	0.5257	0.5	0.5399
OSS	0.5257	0.5	0.5399
NCR	0.4536	0.5391	0.5180

Table 26: Laempe grid CV search on kärna ej fylld with knn

Laempe			
Ohärdad kärna			
Grid CV Search With RF			
Method	**Accuracy**	**F1**	**AUROC**
NM_v1	0.6082	0.24	0.6344
NM_v2	0.4948	0.1694	0.5220
NM_v3	0.5257	0.1153	0.4393
RUS	0.4123	0.1739	0.5265
ENN	0.8659	0.0	0.4772
CNN	0.8350	0.0	0.4602
RENN	0.8659	0.0	0.4772
Tomek	0.8865	0.0	0.4886
OSS	0.8659	0.0	0.4772
NCR	0.8453	0.0	0.4659

Table 27: Laempe grid CV search ohärdad kärna with rf

Laempe			
Ohärdad kärna			
Grid CV Search With ANN			
Method	**Accuracy**	**F1**	**AUROC**
NM_v1	0.7216	0.1290	0.4974
NM_v2	0.0927	0.1698	0.5
NM_v3	0.4226	0.125	0.4324
RUS	0.4639	0.1612	0.5050
ENN	0.9072	0.0	0.5
CNN	0.8659	0.0	0.4772
RENN	0.9072	0.0	0.5
Tomek	0.9072	0.0	0.5
OSS	0.9072	0.0	0.5
NCR	0.9072	0.0	0.5

Table 28: Laempe grid CV search ohärdad kärna with ann

Laempe			
Ohärdad kärna			
Grid CV Search With KNN			
Method	**Accuracy**	**F1**	**AUROC**
NM_v1	0.6185	0.2745	0.6900
NM_v2	0.6288	0.1818	0.5460
NM_v3	0.7113	0.0666	0.4419
RUS	0.5567	0.1886	0.5561
ENN	0.8247	0.2608	0.6041
CNN	0.8350	0.0	0.4602
RENN	0.8247	0.2608	0.6041
Tomek	0.8659	0.1333	0.5271
OSS	0.8659	0.1333	0.5271
NCR	0.8144	0.1818	0.5486

Table 29: Laempe grid CV search ohärdad kärna with knn

Laempe			
Övriga fel			
Grid CV Search With RF			
Method	**Accuracy**	**F1**	**AUROC**
NM_v1	0.5773	0.2264	0.4961
NM_v2	0.4639	0.2777	0.5285
NM_v3	0.4123	0.2400	0.4726
RUS	0.4639	0.2121	0.4533
ENN	0.7113	0.0666	0.4510
CNN	0.6804	0.1142	0.4575
RENN	0.3711	0.2077	0.4228
Tomek	0.7422	0.0740	0.4695
OSS	0.7731	0.1538	0.5131
NCR	0.6597	0.0571	0.4201

Table 30: Laempe grid CV search övriga fel with rf

Laempe			
Övriga fel			
Grid CV Search With ANN			
Method	**Accuracy**	**F1**	**AUROC**
NM_v1	0.6288	0.3076	0.5771
NM_v2	0.5051	0.3142	0.5783
NM_v3	0.5360	0.3076	0.5717
RUS	0.4536	0.2318	0.4722
ENN	0.8247	0.1052	0.5189
CNN	0.7319	0.1333	0.4884
RENN	0.4432	0.2500	0.4911
Tomek	0.8350	0.0	0.5
OSS	0.8350	0.2	0.5501
NCR	0.8247	0.1904	0.5439

Table 31: Laempe grid CV search övriga fel with ann

Laempe			
Övriga fel			
Grid CV Search With KNN			
Method	**Accuracy**	**F1**	**AUROC**
NM_v1	0.4948	0.2222	0.4718
NM_v2	0.6597	0.1538	0.4702
NM_v3	0.4742	0.2608	0.5096
RUS	0.4639	0.1875	0.4282
ENN	0.6804	0.2051	0.5077
CNN	0.7835	0.0	0.4691
RENN	0.4845	0.2187	0.4656
Tomek	0.7628	0.08	0.4818
OSS	0.7113	0.125	0.4760
NCR	0.5979	0.1333	0.4332

Table 32: Laempe grid CV search övriga fel with knn

Appendix B: Results with Hyperparameter Tuning for FCS25

FCS25			
Trasig Vattenmantel			
Grid CV Search With RF			
Method	**Accuracy**	**F1**	**AUROC**
RUS	0.5463	0.3888	0.5781
NM_v1	0.5154	0.4337	0.6224
NM_v2	0.3505	0.3076	0.4515
NM_v3	0.3711	0.3440	0.4969
ENN	0.5463	0.2666	0.4818
CNN	0.6701	0.1578	0.4815
RENN	0.2989	0.3333	0.4663
Tomek	0.7113	0.0	0.46
OSS	0.7525	0.1999	0.5348
NCR	0.6288	0.28	0.5190

Table 33: FCS25 grid CV search trasig vattenmantel with rf

FCS25			
Trasig Vattenmantel			
Grid CV Search With ANN			
Method	**Accuracy**	**F1**	**AUROC**
RUS	0.6288	0.3571	0.5672
NM_v1	0.5360	0.3999	0.5875
NM_v2	0.4948	0.3098	0.4966
NM_v3	0.2474	0.3760	0.5133
ENN	0.6185	0.3934	0.5927
CNN	0.5590	0.3181	0.6907
RENN	0.3298	0.3434	0.4863
Tomek	0.8041	0.3448	0.6003
OSS	0.7731	0.0	0.5
NCR	0.6494	0.3703	0.5806

Table 34: FCS25 grid CV search trasig vattenmantel with ann

FCS25			
Trasig Vattenmantel			
Grid CV Search With KNN			
Method	**Accuracy**	**F1**	**AUROC**
RUS	0.6288	0.4545	0.6475
NM_v1	0.5051	0.3513	0.5354
NM_v2	0.4226	0.3170	0.4821
NM_v3	0.3917	0.2716	0.4300
ENN	0.5257	0.4102	0.5969
CNN	0.6185	0.2745	0.5124
RENN	0.4123	0.3736	0.5396
Tomek	0.6288	0.2173	0.4869
OSS	0.6185	0.2127	0.4803
NCR	0.5876	0.3939	0.5887

Table 35: FCS25 grid CV search trasig vattenmantel with knn

FCS25			
Porös kärna			
Grid CV Search With RF			
Method	**Accuracy**	**F1**	**AUROC**
RUS	0.4639	0.4901	0.4624
NM_v1	0.4536	0.4854	0.4516
NM_v2	0.4948	0.5148	0.4940
NM_v3	0.4848	0.4791	0.4863
ENN	0.4845	0.0386	0.5098
CNN	0.4639	0.2777	0.4784
RENN	0.4845	0.0384	0.5098
Tomek	0.4432	0.4130	0.4471
OSS	0.5051	0.5	0.5070
NCR	0.4948	0.1403	0.5174

Table 36: FCS25 grid CV search porös kärna with rf

FCS25			
Porös kärna			
Grid CV Search With ANN			
Method	**Accuracy**	**F1**	**AUROC**
RUS	0.5154	0.4050	0.5264
NM_v1	0.5257	0.6891	0.5
NM_v2	0.5257	0.6891	0.5
NM_v3	0.5154	0.6802	0.4901
ENN	0.4742	0.0	0.5
CNN	0.4432	0.1	0.4641
RENN	0.4742	0.0	0.5
Tomek	0.4948	0.1967	0.5153
OSS	0.4639	0.2121	0.4816
NCR	0.5051	0.1724	0.5272

Table 37: FCS25 grid CV search porös kärna with ann

FCS25			
Porös kärna			
Grid CV Search With KNN			
Method	**Accuracy**	**F1**	**AUROC**
RUS	0.4639	0.45833	0.4656
NM_v1	0.4845	0.51923	0.4820
NM_v2	0.5154	0.49462	0.5189
NM_v3	0.4742	0.4631	0.4765
ENN	0.5051	0.2	0.5262
CNN	0.4742	0.4	0.4818
RENN	0.5051	0.2	0.5262
Tomek	0.5257	0.4772	0.5319
OSS	0.5257	0.4651	0.5330
NCR	0.5051	0.2941	0.5219

Table 38: FCS25 grid CV search porös kärna with knn

FCS25			
Utstötarmärken			
Grid CV Search With RF			
Method	**Accuracy**	**F1**	**AUROC**
RUS	0.5154	0.0784	0.7526
NM_v1	0.2886	0.0	0.1473
NM_v2	0.2886	0.0	0.1473
NM_v3	0.2886	0.0	0.1473
ENN	0.9793	0.0	0.5
CNN	0.9793	0.0	0.5
RENN	0.9793	0.0	0.5
Tomek	0.9793	0.0	0.5
OSS	0.9793	0.0	0.5
NCR	0.9793	0.0	0.5

Table 39: FCS25 grid CV search utstötamärken with rf

FCS25			
Utstötarmärken			
Grid CV Search With ANN			
Method	**Accuracy**	**F1**	**AUROC**
RUS	0.9793	0.0	0.5
NM_v1	0.113	0.0	0.0578
NM_v2	0.113	0.0	0.0578
NM_v3	0.113	0.0	0.0578
ENN	0.9793	0.0	0.5
CNN	0.9793	0.0	0.5
RENN	0.9793	0.0	0.5
Tomek	0.9793	0.0	0.5
OSS	0.9793	0.0	0.5
NCR	0.9793	0.0	0.5

Table 40: FCS25 grid CV search utstötamärken with ann

FCS25			
Utstötarmärken			
Grid CV Search With KNN			
Method	**Accuracy**	**F1**	**AUROC**
RUS	0.9793	0.0	0.5
NM_v1	0.9793	0.0	0.5
NM_v2	0.9793	0.0	0.5
NM_v3	0.9793	0.0	0.5
ENN	0.9793	0.0	0.5
CNN	0.9793	0.0	0.5
RENN	0.9793	0.0	0.5
Tomek	0.9793	0.0	0.5
OSS	0.9793	0.0	0.5
NCR	0.9793	0.0	0.5

Table 41: FCS25 grid CV search utstötamärken with knn

FCS25			
Kärna ej Fylld			
Grid CV Search With RF			
Method	**Accuracy**	**F1**	**AUROC**
RUS	0.5257	0.1153	0.4258
NM_v1	0.4845	0.2187	0.5798
NM_v2	0.4329	0.2253	0.5954
NM_v3	0.3298	0.1772	0.4936
ENN	0.8969	0.0	0.5
CNN	0.8762	0.25	0.5770
RENN	0.8762	0.25	0.5770
Tomek	0.8969	0.0	0.5
OSS	0.8969	0.0	0.5
NCR	0.9072	0.1818	0.55

Table 42: FCS25 grid CV search kärna ej fylld with rf

FCS25			
Kärna ej Fylld			
Grid CV Search With ANN			
Method	**Accuracy**	**F1**	**AUROC**
RUS	0.5773	0.2807	0.6758
NM_v1	0.4845	0.2187	0.5798
NM_v2	0.8762	0.3333	0.6212
NM_v3	0.3092	0.1728	0.4821
ENN	0.8659	0.0	0.4827
CNN	0.8969	0.0	0.5
RENN	0.8969	0.0	0.5
Tomek	0.8865	0.0	0.4942
OSS	0.8865	0.0	0.4942
NCR	0.8969	0.0	0.5

Table 43: FCS25 grid CV search kärna ej fylld with ann

FCS25			
Kärna ej Fylld			
Grid CV Search With KNN			
Method	**Accuracy**	**F1**	**AUROC**
RUS	0.6804	0.0606	0.4235
NM_v1	0.7113	0.2222	0.5735
NM_v2	0.5670	0.125	0.4488
NM_v3	0.2371	0.1777	0.4862
ENN	0.8865	0.2666	0.5827
CNN	0.8969	0.0	0.5
RENN	0.8350	0.1111	0.5097
Tomek	0.8969	0.0	0.5
OSS	0.8969	0.0	0.5
NCR	0.8453	0.1176	0.5155

Table 44: FCS25 grid CV search kärna ej fylld with knn

FCS25			
Ohärdad Kärna			
Grid CV Search With RF			
Method	**Accuracy**	**F1**	**AUROC**
RUS	0.4123	0.0338	0.4552
NM_v1	0.5360	0.0	0.2736
NM_v2	0.5360	0.0	0.2736
NM_v3	0.4948	0.0392	0.4973
ENN	0.9793	0.0	0.5
CNN	0.8969	0.0	0.4578
RENN	0.9793	0.0	0.5
Tomek	0.9793	0.0	0.5
OSS	0.9793	0.0	0.5
NCR	0.9793	0.0	0.5

Table 45: FCS25 grid CV search ohärdad kärna with rf

FCS25			
Ohärdad Kärna			
Grid CV Search With ANN			
Method	**Accuracy**	**F1**	**AUROC**
RUS	0.4948	0.0392	0.4973
NM_v1	0.9587	0.0	0.4894
NM_v2	0.9587	0.0	0.4894
NM_v3	0.9587	0.0	0.4894
ENN	0.9690	0.0	0.4947
CNN	0.7628	0.0	0.3894
RENN	0.9793	0.0	0.5
Tomek	0.9690	0.0	0.4947
OSS	0.9175	0.0	0.4684
NCR	0.9175	0.0	0.4684

Table 46: FCS25 grid CV search ohärdad kärna with ann

FCS25			
Ohärdad Kärna			
Grid CV Search With KNN			
Method	**Accuracy**	**F1**	**AUROC**
RUS	0.6391	0.0540	0.5710
NM_v1	0.2474	0.0266	0.3710
NM_v2	0.7216	0.1290	0.8578
NM_v3	0.2474	0.0266	0.3710
ENN	0.9793	0.0	0.5
CNN	0.9793	0.0	0.5
RENN	0.9793	0.0	0.5
Tomek	0.9793	0.0	0.5
OSS	0.9793	0.0	0.5
NCR	0.9793	0.0	0.5

Table 47: FCS25 grid CV search ohärdad kärna with knn

FCS25			
Övriga fel			
Grid CV Search With RF			
Method	**Accuracy**	**F1**	**AUROC**
RUS	0.3814	0.25	0.4253
NM_v1	0.4020	0.2564	0.4383
NM_v2	0.3814	0.2682	0.4438
NM_v3	0.4226	0.2222	0.4142
ENN	0.3402	0.3333	0.5103
CNN	0.7010	0.3555	0.5896
RENN	0.2268	0.3362	0.4944
Tomek	0.7938	0.2307	0.5555
OSS	0.6701	0.2000	0.4961
NCR	0.5051	0.2941	0.5032

Table 48: FCS25 grid CV search övriga fel with rf

FCS25			
Övriga fel			
Grid CV Search With ANN			
Method	**Accuracy**	**F1**	**AUROC**
RUS	0.2680	0.3364	0.5019
NM_v1	0.3814	0.3181	0.4993
NM_v2	0.2164	0.3448	0.5064
NM_v3	0.2164	0.3448	0.5064
ENN	0.4123	0.3132	0.5003
CNN	0.7938	0.0	0.5
RENN	0.2164	0.3448	0.5064
Tomek	0.7938	0.0	0.5
OSS	0.7113	0.125	0.4850
NCR	0.3505	0.3076	0.4798

Table 49: FCS25 grid CV search övriga fel with ann

FCS25			
Övriga fel			
Grid CV Search With KNN			
Method	**Accuracy**	**F1**	**AUROC**
RUS	0.4020	0.3255	0.5123
NM_v1	0.3917	0.3058	0.4873
NM_v2	0.3917	0.2891	0.4688
NM_v3	0.4329	0.2857	0.4762
ENN	0.3505	0.3225	0.4983
CNN	0.6701	0.3043	0.5516
RENN	0.2268	0.3243	0.4759
Tomek	0.6907	0.2857	0.5461
OSS	0.6391	0.2222	0.4951
NCR	0.4639	0.2972	0.4957

Table 50: FCS25 grid CV search övriga fel with knn

FCS25			
Skadad Stöt			
Grid CV Search With RF			
Method	**Accuracy**	**F1**	**AUROC**
RUS	0.6597	0.7441	0.6162
NM_v1	0.5463	0.6206	0.5357
NM_v2	0.4742	0.5405	0.4733
NM_v3	0.4845	0.6093	0.4271
ENN	0.5876	0.6491	0.5877
CNN	0.5979	0.6486	0.6092
RENN	0.5876	0.6491	0.5877
Tomek	0.6597	0.7555	0.5959
OSS	0.6391	0.7328	0.5868
NCR	0.5463	0.6071	0.5492

Table 51: FCS25 grid CV search skadad stöt with rf

FCS25			
Skadad Stöt			
Grid CV Search With ANN			
Method	**Accuracy**	**F1**	**AUROC**
RUS	0.6494	0.7848	0.5067
NM_v1	0.6494	0.7848	0.5067
NM_v2	0.6494	0.7848	0.5067
NM_v3	0.6494	0.7848	0.5067
ENN	0.5051	0.4893	0.5648
CNN	0.5773	0.6611	0.5527
RENN	0.5154	0.4835	0.5863
Tomek	0.6494	0.7875	0.5
OSS	0.6494	0.7875	0.5
NCR	0.5670	0.6769	0.5109

Table 52: FCS25 grid CV search skadad stöt with ann

FCS25			
Skadad Stöt			
Grid CV Search With KNN			
Method	**Accuracy**	**F1**	**AUROC**
RUS	0.5360	0.5945	0.5413
NM_v1	0.5979	0.6776	0.5753
NM_v2	0.5360	0.6086	0.5277
NM_v3	0.5670	0.6557	0.5380
ENN	0.5567	0.6126	0.5639
CNN	0.6082	0.6545	0.6239
RENN	0.5567	0.6126	0.5639
Tomek	0.5463	0.6393	0.5154
OSS	0.5463	0.6393	0.5154
NCR	0.5051	0.5636	0.5107

Table 53: FCS25 grid CV search skadad stöt with knn

FCS25			
Skadad Kanal			
Grid CV Search With RF			
Method	**Accuracy**	**F1**	**AUROC**
RUS	0.5463	0.6333	0.5828
NM_v1	0.4226	0.4909	0.4867
NM_v2	0.4948	0.5420	0.6089
NM_v3	0.5051	0.6000	0.5258
ENN	0.5773	0.6719	0.5881
CNN	0.5463	0.6562	0.5229
RENN	0.5567	0.6386	0.6045
Tomek	0.6804	0.8050	0.4759
OSS	0.7422	0.8427	0.5613
NCR	0.7113	0.8082	0.6160

Table 54: FCS25 grid CV search skadad kanal with rf

FCS25			
Skadad Kanal			
Grid CV Search With ANN			
Method	**Accuracy**	**F1**	**AUROC**
RUS	0.3917	0.3655	0.5713
NM_v1	0.7525	0.8588	0.4932
NM_v2	0.7525	0.8588	0.4932
NM_v3	0.7525	0.8588	0.4932
ENN	0.7628	0.8654	0.5
CNN	0.2371	0.0	0.5
RENN	0.7628	0.8654	0.5
Tomek	0.7628	0.8654	0.5
OSS	0.7628	0.8654	0.5
NCR	0.7628	0.8654	0.5

Table 55: FCS25 grid CV search skadad kanal with ann

FCS25			
Skadad Kanal			
Grid CV Search With KNN			
Method	**Accuracy**	**F1**	**AUROC**
RUS	0.6494	0.7213	0.7103
NM_v1	0.5257	0.6229	0.5393
NM_v2	0.4536	0.4752	0.5969
NM_v3	0.5463	0.6140	0.6277
ENN	0.5773	0.6371	0.6780
CNN	0.6391	0.7407	0.5987
RENN	0.5773	0.6495	0.6480
Tomek	0.7319	0.8333	0.5696
OSS	0.7216	0.8258	0.5628
NCR	0.6597	0.7591	0.6122

Table 56: FCS25 grid CV search skadad kanal with knn

Appendix C: Results with Hyperparameter Tuning for Vridkors

Vridkors			
Kavitet 1,2,3			
Grid CV Search With RF			
Method	**Accuracy**	**F1**	**AUROC**
RUS	0.5567	0.6194	0.5466
NM_v1	0.6288	0.6727	0.6293
NM_v2	0.5876	0.6363	0.5860
NM_v3	0.6185	0.6942	0.5927
ENN	0.4536	0.3456	0.5133
CNN	0.6288	0.6896	0.6152
RENN	0.4536	0.3456	0.5133
Tomek	0.6288	0.7230	0.5825
OSS	0.5670	0.6769	0.5176
NCR	0.4742	0.4742	0.4975

Table 57: Vridkors grid CV search kavitet 1,2,3 with rf

Vridkors			
Kavitet 1,2,3			
Grid CV Search With ANN			
Method	**Accuracy**	**F1**	**AUROC**
RUS	0.7113	0.7878	0.6643
NM_v1	0.6288	0.6949	0.6106
NM_v2	0.5773	0.6238	0.5776
NM_v3	0.6907	0.7794	0.6333
ENN	0.4845	0.4047	0.5388
CNN	0.7113	0.7878	0.6643
RENN	0.4845	0.4047	0.5388
Tomek	0.7010	0.7883	0.6418
OSS	0.6907	0.7826	0.6286
NCR	0.4948	0.4615	0.5332

Table 58: Vridkors grid CV search kavitet 1,2,3 with ann

Vridkors			
Kavitet 1,2,3			
Grid CV Search With KNN			
Method	**Accuracy**	**F1**	**AUROC**
RUS	0.6185	0.6605	0.6208
NM_v1	0.5257	0.5660	0.5305
NM_v2	0.5360	0.5714	0.5437
NM_v3	0.5463	0.6140	0.5334
ENN	0.4536	0.3291	0.5180
CNN	0.6185	0.6605	0.6208
RENN	0.4536	0.3291	0.5180
Tomek	0.5876	0.6875	0.5439
OSS	0.6082	0.6984	0.5702
NCR	0.4845	0.4680	0.5153

Table 59: Vridkors grid CV search kavitet 1,2,3 with knn

Vridkors			
Kavitet 4,5,6			
Grid CV Search With RF			
Method	**Accuracy**	**F1**	**AUROC**
RUS	0.6288	0.7096	0.6116
NM_v1	0.6082	0.6935	0.5875
NM_v2	0.6082	0.6666	0.6335
NM_v3	0.7319	0.7968	0.7139
ENN	0.6391	0.6846	0.6835
CNN	0.6288	0.7096	0.6116
RENN	0.6391	0.6846	0.6835
Tomek	0.7938	0.8591	0.7218
OSS	0.7731	0.8382	0.7253
NCR	0.7010	0.7603	0.7099

Table 60: Vridkors grid CV search kavitet 4,5,6 with rf

Vridkors			
Kavitet 4,5,6			
Grid CV Search With ANN			
Method	**Accuracy**	**F1**	**AUROC**
RUS	0.6494	0.7118	0.6634
NM_v1	0.6288	0.7313	0.5656
NM_v2	0.5463	0.6140	0.5611
NM_v3	0.6597	0.7027	0.7077
ENN	0.5773	0.6434	0.5927
CNN	0.6391	0.7798	0.4626
RENN	0.5773	0.6434	0.5927
Tomek	0.6907	0.8170	0.5
OSS	0.6907	0.8170	0.5
NCR	0.6082	0.7121	0.5507

Table 61: Vridkors grid CV search kavitet 4,5,6 with ann

Vridkors			
Kavitet 4,5,6			
Grid CV Search With KNN			
Method	**Accuracy**	**F1**	**AUROC**
RUS	0.6701	0.7288	0.6875
NM_v1	0.6391	0.7244	0.6099
NM_v2	0.5979	0.6548	0.6261
NM_v3	0.6907	0.7656	0.6656
ENN	0.5979	0.6422	0.6445
CNN	0.6082	0.6935	0.5875
RENN	0.5979	0.6422	0.6445
Tomek	0.7113	0.7846	0.6805
OSS	0.7010	0.7751	0.6731
NCR	0.6701	0.7377	0.6691

Table 62: Vridkors grid CV search kavitet 4,5,6 with knn

Vridkors			
Kavitet 7,8,9			
Grid CV Search With RF			
Method	**Accuracy**	**F1**	**AUROC**
RUS	0.7731	0.7555	0.7749
NM_v1	0.7525	0.7142	0.7469
NM_v2	0.7731	0.7499	0.7726
NM_v3	0.7216	0.6823	0.7168
ENN	0.6391	0.7058	0.6735
CNN	0.7731	0.7499	0.7726
RENN	0.6391	0.7058	0.6735
Tomek	0.7113	0.6666	0.7052
OSS	0.7835	0.7640	0.7842
NCR	0.7216	0.7567	0.7476

Table 63: Vridkors grid CV search kavitet 7,8,9 with rf

Vridkors			
Kavitet 7,8,9			
Grid CV Search With ANN			
Method	**Accuracy**	**F1**	**AUROC**
RUS	0.6804	0.6436	0.6774
NM_v1	0.6701	0.7241	0.7013
NM_v2	0.7422	0.7252	0.7448
NM_v3	0.7216	0.6823	0.7168
ENN	0.6288	0.7	0.6642
CNN	0.6804	0.6666	0.6845
RENN	0.6288	0.7	0.6642
Tomek	0.7319	0.7450	0.7474
OSS	0.7422	0.7619	0.7614
NCR	0.7113	0.7358	0.7312

Table 64: Vridkors grid CV search kavitet 7,8,9 with ann

Vridkors			
Kavitet 7,8,9			
Grid CV Search With KNN			
Method	**Accuracy**	**F1**	**AUROC**
RUS	0.7319	0.7111	0.7332
NM_v1	0.7422	0.7126	0.7400
NM_v2	0.6804	0.6516	0.6798
NM_v3	0.7319	0.6904	0.7260
ENN	0.6907	0.7321	0.7174
CNN	0.7010	0.7010	0.7101
RENN	0.6907	0.7321	0.7174
Tomek	0.7835	0.7878	0.7960
OSS	0.7731	0.7499	0.7726
NCR	0.7525	0.7735	0.7730

Table 65: Vridkors grid CV search kavitet 7,8,9 with knn

Vridkors			
Kavitet 10,11,12			
Grid CV Search With RF			
Method	**Accuracy**	**F1**	**AUROC**
RUS	0.5979	0.6355	0.5934
NM_v1	0.6288	0.6727	0.6217
NM_v2	0.5876	0.6153	0.5859
NM_v3	0.6082	0.6346	0.6067
ENN	0.5567	0.4415	0.5808
CNN	0.5463	0.5319	0.5540
RENN	0.5567	0.4415	0.5808
Tomek	0.6288	0.6666	0.6237
OSS	0.5979	0.6285	0.5954
NCR	0.5773	0.5393	0.5900

Table 66: Vridkors grid CV search kavitet 10,11,12 with rf

Vridkors			
Kavitet 10,11,12			
Grid CV Search With ANN			
Method	**Accuracy**	**F1**	**AUROC**
RUS	0.6597	0.6972	0.6539
NM_v1	0.6391	0.6666	0.6370
NM_v2	0.6701	0.7090	0.6633
NM_v3	0.6288	0.6538	0.6275
ENN	0.5670	0.5434	0.5767
CNN	0.6082	0.6274	0.6087
RENN	0.5670	0.5434	0.5767
Tomek	0.5876	0.6	0.5898
OSS	0.5773	0.6019	0.5765
NCR	0.5670	0.5531	0.5748

Table 67: Vridkors grid CV search kavitet 10,11,12 with ann

Vridkors			
Kavitet 10,11,12			
Grid CV Search With KNN			
Method	**Accuracy**	**F1**	**AUROC**
RUS	0.6082	0.62	0.6106
NM_v1	0.5979	0.6138	0.5992
NM_v2	0.5567	0.5742	0.5576
NM_v3	0.6185	0.6476	0.6162
ENN	0.5773	0.4810	0.5996
CNN	0.5876	0.5454	0.6014
RENN	0.5773	0.4810	0.5996
Tomek	0.5876	0.6	0.5898
OSS	0.5876	0.5918	0.5917
NCR	0.5876	0.5348	0.6033

Table 68: Vridkors grid CV search kavitet 10,11,12 with knn

Vridkors			
Kavitet 13,14,15			
Grid CV Search With RF			
Method	**Accuracy**	**F1**	**AUROC**
RUS	0.7319	0.7173	0.7458
NM_v1	0.6391	0.5679	0.6296
NM_v2	0.6288	0.5714	0.6245
NM_v3	0.6391	0.5783	0.6333
ENN	0.4639	0.5666	0.5214
CNN	0.6597	0.6451	0.6732
RENN	0.4639	0.5666	0.5214
Tomek	0.6701	0.5789	0.6521
OSS	0.6804	0.6352	0.6796
NCR	0.6185	0.6837	0.6754

Table 69: Vridkors grid CV search kavitet 13,14,15 with rf

Vridkors			
Kavitet 13,14,15			
Grid CV Search With ANN			
Method	**Accuracy**	**F1**	**AUROC**
RUS	0.5979	0.5894	0.6131
NM_v1	0.6185	0.4477	0.5822
NM_v2	0.4742	0.5486	0.5190
NM_v3	0.4536	0.5309	0.4978
ENN	0.5051	0.5862	0.5565
CNN	0.6082	0.5777	0.6144
RENN	0.5051	0.5862	0.5565
Tomek	0.4742	0.5486	0.5190
OSS	0.4742	0.5486	0.5190
NCR	0.4845	0.6093	0.5576

Table 70: Vridkors grid CV search kavitet 13,14,15 with ann

Vridkors			
Kavitet 13,14,15			
Grid CV Search With KNN			
Method	**Accuracy**	**F1**	**AUROC**
RUS	0.6288	0.5909	0.6320
NM_v1	0.6701	0.5789	0.6521
NM_v2	0.6494	0.5526	0.6309
NM_v3	0.6391	0.5569	0.6258
ENN	0.6494	0.6304	0.6607
CNN	0.6494	0.6136	0.6532
RENN	0.6494	0.6304	0.6607
Tomek	0.6597	0.5714	0.6434
OSS	0.6391	0.5454	0.6221
NCR	0.6288	0.6538	0.6618

Table 71: Vridkors grid CV search kavitet 13,14,15 with knn

Vridkors			
Kavitet 16,17,18			
Grid CV Search With RF			
Method	**Accuracy**	**F1**	**AUROC**
RUS	0.7938	0.7142	0.7803
NM_v1	0.7525	0.7	0.7689
NM_v2	0.8144	0.775	0.8368
NM_v3	0.8762	0.8181	0.8573
ENN	0.8041	0.7397	0.8018
CNN	0.7835	0.7042	0.7724
RENN	0.8247	0.7733	0.8312
Tomek	0.8247	0.7536	0.8109
OSS	0.7835	0.6769	0.7521
NCR	0.7319	0.6829	0.7530

Table 72: Vridkors grid CV search kavitet 16,17,18 with rf

Vridkors			
Kavitet 16,17,18			
Grid CV Search With ANN			
Method	**Accuracy**	**F1**	**AUROC**
RUS	0.7628	0.7088	0.7768
NM_v1	0.7731	0.725	0.7915
NM_v2	0.7216	0.6582	0.7315
NM_v3	0.7422	0.6575	0.7338
ENN	0.7422	0.6987	0.7677
CNN	0.6907	0.6428	0.7145
RENN	0.7319	0.5666	0.6785
Tomek	0.7628	0.7228	0.7903
OSS	0.6288	0.5609	0.6398
NCR	0.6597	0.6117	0.6839

Table 73: Vridkors grid CV search kavitet 16,17,18 with ann

Vridkors			
Kavitet 16,17,18			
Grid CV Search With KNN			
Method	**Accuracy**	**F1**	**AUROC**
RUS	0.7319	0.6486	0.7259
NM_v1	0.7319	0.5937	0.6921
NM_v2	0.7525	0.7073	0.7756
NM_v3	0.8247	0.7462	0.8041
ENN	0.6907	0.6739	0.7415
CNN	0.7525	0.6666	0.7418
RENN	0.6907	0.5945	0.6806
Tomek	0.7835	0.7042	0.7724
OSS	0.7835	0.7123	0.7791
NCR	0.7010	0.6233	0.7021

Table 74: Vridkors grid CV search kavitet 16,17,18 with knn

Vridkors			
Kavitet 1,2,3 (SP)			
Grid CV Search With RF			
Method	**Accuracy**	**F1**	**AUROC**
RUS	0.5154	0.0408	0.3377
NM_v1	0.2061	0.1720	0.5674
NM_v2	0.5463	0.1851	0.5821
NM_v3	0.5979	0.0930	0.4396
ENN	0.9175	0.0	0.5
CNN	0.8969	0.0	0.4887
RENN	0.8350	0.0	0.4550
Tomek	0.9175	0.0	0.5
OSS	0.9175	0.0	0.5
NCR	0.8969	0.0	0.4887

Table 75: Vridkors grid CV search kavitet 1,2,3 (sp) with rf

Vridkors			
Kavitet 1,2,3 (SP)			
Grid CV Search With ANN			
Method	**Accuracy**	**F1**	**AUROC**
RUS	0.5257	0.1153	0.4571
NM_v1	0.2371	0.1395	0.4705
NM_v2	0.5567	0.2181	0.6446
NM_v3	0.7422	0.1379	0.5182
ENN	0.8659	0.0	0.4719
CNN	0.8453	0.0	0.4606
RENN	0.7525	0.076	0.4669
Tomek	0.8865	0.0	0.4831
OSS	0.8762	0.0	0.4775
NCR	0.8556	0.125	0.5231

Table 76: Vridkors grid CV search kavitet 1,2,3 (sp) with ann

Vridkors			
Kavitet 1,2,3 (SP)			
Grid CV Search With KNN			
Method	**Accuracy**	**F1**	**AUROC**
RUS	0.4536	0.1587	0.5316
NM_v1	0.2371	0.1590	0.5273
NM_v2	0.7628	0.1481	0.5294
NM_v3	0.5876	0.090	0.4339
ENN	0.9175	0.0	0.5
CNN	0.9175	0.0	0.5
RENN	0.7010	0.1212	0.4957
Tomek	0.9175	0.0	0.5
OSS	0.9175	0.0	0.5
NCR	0.8247	0.0	0.4494

Table 77: Vridkors grid CV search kavitet 1,2,3 (sp) with knn

Vridkors			
Kavitet 4,5,6 (SP)			
Grid CV Search With RF			
Method	**Accuracy**	**F1**	**AUROC**
RUS	0.4742	0.1355	0.5190
NM_v1	0.2989	0.1282	0.4904
NM_v2	0.4020	0.0937	0.4142
NM_v3	0.2783	0.1666	0.6111
ENN	0.8865	0.0	0.4777
CNN	0.9072	0.1818	0.5547
RENN	0.7216	0.1818	0.5865
Tomek	0.9278	0.0	0.5
OSS	0.9278	0.0	0.5
NCR	0.9278	0.0	0.5

Table 78: Vridkors grid CV search kavitet 4,5,6 (sp) with rf

Vridkors			
Kavitet 4,5,6 (SP)			
Grid CV Search With ANN			
Method	**Accuracy**	**F1**	**AUROC**
RUS	0.4226	0.0344	0.2936
NM_v1	0.4432	0.1	0.4365
NM_v2	0.4845	0.1379	0.5246
NM_v3	0.2989	0.1707	0.6222
ENN	0.7731	0.0833	0.4825
CNN	0.7835	0.0869	0.4880
RENN	0.6288	0.1818	0.6023
Tomek	0.8556	0.0	0.4611
OSS	0.8144	0.0	0.4388
NCR	0.7835	0.0	0.4222

Table 79: Vridkors grid CV search kavitet 4,5,6 (sp) with ann

Vridkors			
Kavitet 4,5,6 (SP)			
Grid CV Search With KNN			
Method	**Accuracy**	**F1**	**AUROC**
RUS	0.4123	0.1739	0.6174
NM_v1	0.5463	0.0	0.2944
NM_v2	0.7319	0.2777	0.7238
NM_v3	0.5670	0.2222	0.7007
ENN	0.9278	0.0	0.5
CNN	0.8865	0.1538	0.5436
RENN	0.7010	0.2162	0.6412
Tomek	0.8453	0.0	0.4555
OSS	0.8453	0.0	0.4555
NCR	0.7731	0.0	0.4166

Table 80: Vridkors grid CV search kavitet 4,5,6 (sp) with knn

Vridkors			
Kavitet 7,8,9 (SP)			
Grid CV Search With RF			
Method	**Accuracy**	**F1**	**AUROC**
RUS	0.5360	0.1176	0.5192
NM_v1	0.4639	0.1333	0.5586
NM_v2	0.5154	0.1454	0.5860
NM_v3	0.2577	0.1428	0.6043
ENN	0.8659	0.0	0.4615
CNN	0.9381	0.0	0.5
RENN	0.8762	0.25	0.6227
Tomek	0.9072	0.0	0.4835
OSS	0.9278	0.0	0.4945
NCR	0.8350	0.0	0.4450

Table 81: Vridkors grid CV search kavitet 7,8,9 (sp) with rf

Vridkors			
Kavitet 7,8,9 (SP)			
Grid CV Search With ANN			
Method	**Accuracy**	**F1**	**AUROC**
RUS	0.5257	0.08	0.4358
NM_v1	0.3608	0.0882	0.4258
NM_v2	0.4226	0.0967	0.4587
NM_v3	0.3608	0.1621	0.6593
ENN	0.8350	0.0	0.4450
CNN	0.6288	0.0526	0.4130
RENN	0.7319	0.1333	0.5457
Tomek	0.8659	0.0	0.4615
OSS	0.7835	0.0	0.4175
NCR	0.8041	0.0	0.4285

Table 82: Vridkors grid CV search kavitet 7,8,9 (sp) with ann

Vridkors			
Kavitet 7,8,9 (SP)			
Grid CV Search With KNN			
Method	**Accuracy**	**F1**	**AUROC**
RUS	0.5463	0.1200	0.5247
NM_v1	0.6907	0.0625	0.4459
NM_v2	0.6185	0.1395	0.5631
NM_v3	0.3505	0.1126	0.4981
ENN	0.8659	0.2352	0.6172
CNN	0.8865	0.2666	0.6282
RENN	0.8247	0.1904	0.5952
Tomek	0.8247	0.1052	0.5173
OSS	0.8247	0.1052	0.5173
NCR	0.8556	0.2222	0.6117

Table 83: Vridkors grid CV search kavitet 7,8,9 (sp) with knn

Vridkors			
Kavitet 10,11,12 (SP)			
Grid CV Search With RF			
Method	**Accuracy**	**F1**	**AUROC**
RUS	0.4432	0.0357	0.3508
NM_v1	0.1649	0.0898	0.5645
NM_v2	0.2061	0.0941	0.5860
NM_v3	0.4226	0.0666	0.4596
ENN	0.9587	0.0	0.5
CNN	0.9587	0.0	0.5
RENN	0.8556	0.0	0.4462
Tomek	0.9587	0.0	0.5
OSS	0.9587	0.0	0.5
NCR	0.9587	0.0	0.5

Table 84: Vridkors grid CV search kavitet 10,11,12 (sp) with rf

Vridkors			
Kavitet 10,11,12 (SP)			
Grid CV Search With ANN			
Method	**Accuracy**	**F1**	**AUROC**
RUS	0.7319	0.0	0.3817
NM_v1	0.5567	0.0444	0.4099
NM_v2	0.6804	0.0	0.3548
NM_v3	0.7319	0.0	0.3817
ENN	0.8969	0.0	0.4677
CNN	0.9278	0.0	0.4838
RENN	0.7731	0.0	0.4032
Tomek	0.9484	0.0	0.4946
OSS	0.9587	0.0	0.5
NCR	0.8969	0.0	0.4677

Table 85: Vridkors grid CV search kavitet 10,11,12 (sp) with ann

Vridkors			
Kavitet 10,11,12 (SP)			
Grid CV Search With KNN			
Method	**Accuracy**	**F1**	**AUROC**
RUS	0.5463	0.0833	0.5241
NM_v1	0.9587	0.0	0.5
NM_v2	0.6288	0.0526	0.4475
NM_v3	0.4948	0.1090	0.6169
ENN	0.9587	0.0	0.5
CNN	0.9587	0.0	0.5
RENN	0.7628	0.0	0.3978
Tomek	0.9587	0.0	0.5
OSS	0.9587	0.0	0.5
NCR	0.9587	0.0	0.5

Table 86: Vridkors grid CV search kavitet 10,11,12 (sp) with knn

Vridkors			
Kavitet 13,14,15 (SP)			
Grid CV Search With RF			
Method	**Accuracy**	**F1**	**AUROC**
RUS	0.5876	0.0476	0.4260
NM_v1	0.3092	0.0821	0.5201
NM_v2	0.3608	0.1142	0.6666
NM_v3	0.3505	0.0307	0.3024
ENN	0.9484	0.0	0.4946
CNN	0.9278	0.0	0.4838
RENN	0.8762	0.0	0.4569
Tomek	0.9484	0.0	0.4946
OSS	0.9484	0.0	0.4946
NCR	0.9484	0.0	0.4946

Table 87: Vridkors grid CV search kavitet 13,14,15 (sp) with rf

Vridkors			
Kavitet 13,14,15 (SP)			
Grid CV Search With ANN			
Method	**Accuracy**	**F1**	**AUROC**
RUS	0.2783	0.0789	0.5040
NM_v1	0.3195	0.0571	0.4059
NM_v2	0.4742	0.1052	0.6061
NM_v3	0.7113	0.0666	0.4905
ENN	0.9484	0.0	0.4946
CNN	0.9381	0.0	0.4892
RENN	0.9381	0.0	0.4892
Tomek	0.9484	0.0	0.4946
OSS	0.9484	0.0	0.4946
NCR	0.8969	0.0	0.4677

Table 88: Vridkors grid CV search kavitet 13,14,15 (sp) with ann

Vridkors			
Kavitet 13,14,15 (SP)			
Grid CV Search With KNN			
Method	**Accuracy**	**F1**	**AUROC**
RUS	0.4226	0.0666	0.4596
NM_v1	0.5051	0.1111	0.6223
NM_v2	0.4845	0.1071	0.6115
NM_v3	0.6701	0.0588	0.4690
ENN	0.8762	0.0	0.4569
CNN	0.9072	0.0	0.4731
RENN	0.7938	0.0	0.4139
Tomek	0.9072	0.0	0.4731
OSS	0.9072	0.0	0.4731
NCR	0.9175	0.0	0.4784

Table 89: Vridkors grid CV search kavitet 13,14,15 (sp) with knn

Vridkors			
Kavitet 16,17,18 (SP)			
Grid CV Search With RF			
Method	**Accuracy**	**F1**	**AUROC**
RUS	0.2989	0.0555	0.3467
NM_v1	0.2783	0.1025	0.525
NM_v2	0.5154	0.1754	0.7445
NM_v3	0.7113	0.2631	0.8478
ENN	0.9587	0.3333	0.6
CNN	0.9484	0.0	0.5
RENN	0.9587	0.3333	0.6
Tomek	0.9484	0.0	0.5
OSS	0.9484	0.0	0.5
NCR	0.9587	0.3333	0.6

Table 90: Vridkors grid CV search kavitet 16,17,18 (sp) with rf

Vridkors			
Kavitet 16,17,18 (SP)			
Grid CV Search With ANN			
Method	**Accuracy**	**F1**	**AUROC**
RUS	0.5463	0.0434	0.4045
NM_v1	0.3195	0.0571	0.4059
NM_v2	0.4742	0.1052	0.6061
NM_v3	0.7113	0.0666	0.4905
ENN	0.9484	0.0	0.4946
CNN	0.9484	0.0	0.4946
RENN	0.9381	0.0	0.4892
Tomek	0.9484	0.0	0.4946
OSS	0.9175	0.0	0.4784
NCR	0.8969	0.0	0.4677

Table 91: Vridkors grid CV search kavitet 16,17,18 (sp) with ann

Vridkors			
Kavitet 16,17,18 (SP)			
Grid CV Search With KNN			
Method	**Accuracy**	**F1**	**AUROC**
RUS	0.8865	0.1538	0.5619
NM_v1	0.7731	0.0	0.4076
NM_v2	0.8762	0.1428	0.5565
NM_v3	0.6804	0.1621	0.6423
ENN	0.9072	0.3076	0.6673
CNN	0.9484	0.0	0.5
RENN	0.9072	0.3076	0.6673
Tomek	0.9587	0.3333	0.6
OSS	0.9484	0.2857	0.5945
NCR	0.9484	0.4444	0.6891

Table 92: Vridkors grid CV search kavitet 16,17,18 (sp) with knn

Vridkors			
Uppstartskass			
Grid CV Search With RF			
Method	**Accuracy**	**F1**	**AUROC**
RUS	0.2061	0.3304	0.35
NM_v1	0.2680	0.4033	0.6263
NM_v2	0.2680	0.4036	0.6263
NM_v3	0.2989	0.4426	0.6421
ENN	0.9793	0.9895	0.5
CNN	0.9793	0.9895	0.5
RENN	0.9793	0.9895	0.5
Tomek	0.9793	0.9895	0.5
OSS	0.9793	0.9895	0.5
NCR	0.9793	0.9895	0.5

Table 93: Vridkors grid CV search kavitet uppstartskass with rf

Vridkors			
Uppstartskass			
Grid CV Search With ANN			
Method	**Accuracy**	**F1**	**AUROC**
RUS	0.5051	0.1428	0.6445
NM_v1	0.2783	0.0789	0.4304
NM_v2	0.8556	0.1250	0.5456
NM_v3	0.8453	0.2105	0.6347
ENN	0.9278	0.3636	0.6782
CNN	0.8247	0.1052	0.5293
RENN	0.9278	0.3636	0.6782
Tomek	0.9484	0.0	0.5
OSS	0.9278	0.0	0.4891
NCR	0.9381	0.4000	0.6836

Table 94: Vridkors grid CV search kavitet uppstartskass with ann

Vridkors			
Uppstartskass			
Grid CV Search With KNN			
Method	**Accuracy**	**F1**	**AUROC**
RUS	0.5463	0.7027	0.5236
NM_v1	0.0618	0.0808	0.5210
NM_v2	0.6082	0.7532	0.5552
NM_v3	0.1340	0.2075	0.5578
ENN	0.9793	0.9895	0.5
CNN	0.9793	0.9895	0.5
RENN	0.9793	0.9895	0.5
Tomek	0.9793	0.9895	0.5
OSS	0.9793	0.9895	0.5
NCR	0.9793	0.9895	0.5

Table 95: Vridkors grid CV search kavitet uppstartskass with knn