Detection and Analysis of Molten Aluminium Cleanliness Using a Pulsed Ultrasound System

Presented by:

Kgothatso Matlala

MHLKGO007

Prepared for:

Prof. Amit Mishra

Department of Electrical Engineering

University of Cape Town

Submitted to the Department of Electrical Engineering at the University of Cape Town in partial fulfilment of the academic requirements for the degree of Master of Engineering.

September 2022

Declaration

1. I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.
2. I have used the IEEE convention for citation and referencing. Each contribution to, and quotation in, this project report from the work(s) of other people, has been attributed and has been cited and referenced.
3. This project report is my own work.
4. I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as their own work or part thereof.

Signature of Author.....

Cape Town

Abstract

This document presents the development of a solution for analysis and detection of molten metal quality deviations. The data is generated by an MV20/20, an ultrasound sensor that detects inclusions - molten metal defects that affect the quality of the product. The data is then labelled by assessing the sample using metallography. The analysis provides the sample outcome and dominant inclusion. The business objectives for the project include the real-time classification of anomalous events by means of a supervised classifier for the metal quality outcome, and a classifier for the inclusion type responsible for low quality. The adopted methodology involves descriptive, diagnostic and predictive analytics. Once the data is statistically profiled, it is standardised and scaled to unit variance in order to compensate for different units in the descriptors. Principal components analysis is applied as a dimensionality reduction technique, and it is found that the first three components account for 99.6% of the variance of the dataset.

In order for the system to have predictive ability, two modelling approaches are considered, namely Response Surface Methodology and supervised machine learning. Supervised machine learning is preferred as it offers more flexibility than a polynomial approximator, and it is more accurate. Four classifiers are built, namely logistic regression, support vector machine, multi-layer perceptron and a radial basis function network. The hyperparameters are tuned using 10-fold repeated cross-validation. The multi-layer perceptron offers the best performance in all cases. For determining the quality outcome of a cast (passed or failed), all the models perform according to business targets for accuracy, precision, sensitivity and specificity. For the inclusion type classification, the multi-layer perceptron performs within 5% of the target metrics. In order to optimise the model, a grid search is performed for optimal parameter tuning. The results offer negligible improvement, which indicates that the model has reached a global maximum in the parameter optimisation in the hyperspace. It is noted that the source of variance in the inclusion type data respondent is attributed to operator error during labelling of the dataset, among several other sources of variance. It is therefore recommended that a Gage R&R be performed in order to identify sources of variation, among other improvement recommendations. From a research perspective, a vision system is recommended for assessing metal colour, texture and other visual properties in order to provide more insights. Another possible research extension recommended is the use of Fourier Transform Infrared Spectroscopy in determining signatures of the clean metal and different inclusions for detection. The project is regarded as a success, as the business metrics are met by the solution.

Acknowledgements

I would like to extend my most sincere gratitude to my supervisor, Professor **Amit Kumar Mishra**, for his invaluable guidance towards the realisation of this project.

I would like to thank **Thirushan Govender**, **Sakhile Kubeka**, **Macdonald Valoyi** and **Sello Matlala**. They are the team at Hulamin Rolled Products that funded the project, availed the dataset used in this project, stitched the dataset with the lab analysis results and assisted with domain expertise for various stages of the casthouse, metallurgical concepts and business requirements.

This report is dedicated to my grandparents, **Seleke** and **Motimedi**, my parents, **Mmoti** and **Thabo**, my wife, **Gontse**, and my beautiful daughters, **Taiwa** and **Kanego**. I love you all.

Contents

List of Figures

List of Tables

List of Algorithms

Nomenclature

Acronyms

AUC	Area under curve
CFF	Ceramic Foam Filter
DDA	Dynamic decay adjustment
RSM	Response Surface Methodology
EM	Electromagnetic
FeO	Iron Oxide
GLM	Generalised Linear Model
LPS	Largest Particle Size
MgO	Magnesium Oxide
MLP	Multi-layer perceptron
NDT	Non-destructive testing
PCA	Principal Components Analysis
PoDFA	Porous disk filtration apparatus
RBF	Radial basis function
ReLU	Rectified linear unit
ROC	Receiver operating characteristic
SPC	Statistical process control
SVM	Support vector machine
Tanh	Hyperbolic tangent

Symbols

κ	Cohen's kappa. This is a metric used to determine a classification model's agreement with the categories when the classes are imbalanced.
K	Kernel function. This function is used in logistic regression, support vector machines and kernel-based neural networks.
γ	Gamma. This is the exponential kernel factor.
β	Beta. This is the biasing factor for the kernel function.
α	Alpha. This is the input weight for the kernel function.
ϕ	Phi. This is the radial basis function.
σ	Sigma. This is the standard deviation of a population.
θ	Theta. This is the threshold for the radial basis function with dynamic delay adjustment.
ρ	Rho. This is the gradient descent term for a neural network.
ϵ	Epsilon. This is the selection randomness probability for the learning gradient of a neural network.

Terminology

Multi-layer Perceptron a feedforward artificial neural network consisting of multiple layers of neurons. The neurons are connected by weights and activation functions typically as an input layer, one or more hidden layers and an output layer.

Inclusion the undesired particles present in molten Aluminium that are either introduced during casting as impurities, or that form as films of oxides during settling.

FeO a solid particle that forms as a result of the oxidisation of Iron.

Casthouse an industrial facility where recycled metal is treated by melting, removing impurities and mixing with different elements to achieve the desired properties.

MgO a solid particle that forms as a result of the oxidisation of Magnesium.

PoDFA a technique used to extract a small (~1kg) sample from a metal cast, use a filter to trap inclusions and analyse the inclusions for classification and quality purposes.

Support Vector Machine a machine learning algorithm used for regression and classification. SVMs can perform non-linear regression/classification using a kernel function.

Principal Components Analysis a machine learning algorithm used for regression and classification. SVMs can perform non-linear regression/classification using a kernel function.

Receiver Operating Characteristic the curve showing the specificity as a function of sensitivity for a given decision threshold in binomial classification.

Confusion Matrix a matrix used in classification problems showing how the model predicted values against the actual values.

Accuracy the number of true predictions as a ratio of all predictions made.

Precision the ratio of correctly predicted values to all predicted values.

Sensitivity the ratio of positively predicted values to true positives.

Specificity the ratio of negatively predicted values to true negatives.

Chapter 1

Introduction

The advent of real-time, objective measurement techniques has contributed largely to the improvement in efficiency, reliability and quality of many industrial processes and products [1]. In many of the processes, an operator/user has been required to formulate non-empirical results based on observations and experience, which subjects the measurements to errors and human bias. Lately, many of the measurements have been automated by means of sensing devices capable of continuous, reliable measurements.

This document discusses the application of exploratory data analytics and supervised learning algorithms to data produced by an autonomous measurement system within the context of quality control for Aluminium cleanliness in a casthouse. This field of research has been made possible by the advent of non-destructive testing methods [2], which allow for the collection of empirical datasets providing a more in-depth view of product quality.

The measurement system uses a pulsed ultrasound technique to transmit pressure waves and measure the return signal. The attenuation and distortion of the signal are analysed by the system to detect unwanted particles. The system returns metrics which have been classified using metallographic analyses. These classifications are used to train the models based on business targets for quality benchmarking and the reduction of quality related customer complaints.

1.1 Background

1.1.1 Aluminium Processing

Aluminium is one of the most widely used industrial products to date [3]. This is because it can be formed into multiple alloys with different properties, it is non-corrosive and most importantly it is 100% recyclable. Based on the wide range of applications including food packaging, it is critical to ensure that the Aluminium produced meets stringent quality standards in order to ensure that it is safe for use, does not have contaminants and it does not have leaks [4, 5]. To achieve this, many organisations extract samples for testing at each processing stage of the Aluminium product.

A typical layout of an Aluminium casthouse is illustrated in Figure 1.1:

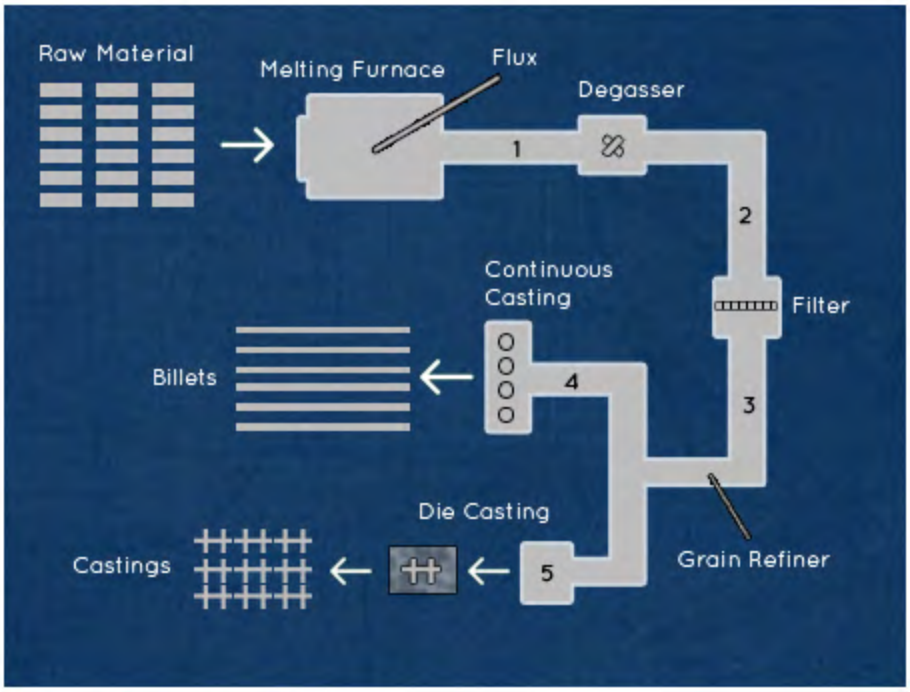

Figure 1.1: Aluminium Casthouse [6]

The process of casting Aluminium is summarised in the following steps:

Collection of raw material (top left on Figure 1.1). Since Aluminium is 100% recyclable, all scrap generated in-house is used as raw material. Recycled beverage cans are also used as input raw material. In addition, pig, which is the term used for pure Aluminium, is used in the process of alloying together with small quantities of Magnesium, Iron and Copper.

Melting. The raw material is poured into a melting furnace (component before producing stage 1 of metal on Figure 1.1), where it is heated up to ~700^0C. Once molten, a chemical flux is added in order to trap large inclusions ($>200\mu m$) and elevate them to the top of the molten solution. The inclusions are then removed using a fork lift with a specialised extraction tool.

Holding. This is part of the melting furnace and involves allowing the metal to settle for ~45 minutes. Once settled, most of the inclusions still present in the metal will have risen to the surface, where they are extracted by operators. The metal is held for ~15 minutes more before being poured onto a launder.

Degassing. This process involves the usage of rotary device to stir the metal and pass it through a chamber where gasses are trapped (produces stage 2 of the metal on Figure 1.1). This improves the metallurgical properties of the metal by reducing porosity. Porosity reduces the material strength of the metal, which could cause tearing in downstream processing.

Filtering. The filtration stage involves the use of specialised ceramic foam (CFF) filters, which are designed to trap inclusions ranging from $5\mu m$ (producing stage 3 of the metal on Figure 1.1).

Refinery. This stage involves the addition of a metal rod to the melt, which improves the metallurgical properties of the metal (produces stages 4 and 5 on Figure 1.1).

Casting. The metal is cast into solid billets by pouring it into five brick-shaped containers. As it is slowly poured into the containers and solidifies, water is sprayed onto the walls of the containers in order to control the casting temperature.

Once the metal is cast into billets, they are stored temporarily for cooling before being transported for downstream processing.

1.1.2 Quality Control

Quality control is a critical function of the casthouse. It involves the measurement of inclusions to assess whether the metal is fit for use [7]. If deemed unfit, the metal is scrapped.

During the holding and casting stages, 1kg samples are extracted for quality analyses in the lab. The samples are analysed using metallography. This involves analysis of the sample under a microscope, and usage of reference figures to identify the inclusions present in the metal and estimate their properties. Based on this analysis, a result of the sample is given based on quality guidelines within the business. Typical metallographic analysis results are shown in Table 1.1:

Table 1.1: Metallographic analysis results

Sample_Number	Inclusion_Abundance	Inclusion_Type	Magnification	Sample_Result
1	High	FeO	2x	FAILED
2	Low	SPINEL	2x	PASSED
3	Critical	FeO	2x	FAILED
4	Moderate	FeO	3x	PASSED
5	Low	MgO	2x	PASSED

Table 1.1 shows an illustration of the sample results following a laboratory metallographic analysis. For each cast of the metal, a sample is extracted and sent to the lab for processing. Processing involves preparing the sample and analysing it under a microscope to estimate structure, abundance of inclusions and the types present.

The method used in determining the success or failure of a sample is based mainly on the experience and judgement of the operator. A poster exists with visual depictions of the density formations of inclusions and their abundance from low to critical. The determination of the abundance is made by the operator, and they refer to more senior technicians for boundary cases.

The key respondents from Table 1.1 are inclusion type and sample result, which are the categorical respondents applied to the dataset for supervised machine learning.

1.2 Important Terminology

Confusion Matrix A table having the same row and column labels showing the performance of a classifier. The columns of the table are the actual classes, whereas the rows are the predicted classes. The diagonal of the table indicates the space where the predicted and actual results are matched.

Hyperparameter Hyperparameters are parameters that determine the learning efficiency of an algorithm. The hyperparameters affect the algorithm's learning ability, loss function and learning rate.

Inclusion A particle, usually a product of oxidisation with metals, that is found in an Aluminium cast. Inclusions are harder than Aluminium and have higher melting temperatures. They manifest in solid form in a cast and cause surface scratches, leaks and tears on the Aluminium metal during forming stages.

Metallography The visual analysis of physical properties of materials using a microscope. The analysis includes determining the grain structure, inter-metallics, shapes and types of particles present in the metal under evaluation.

Neural Network A machine learning algorithm used for regression and classification problems. Neural networks are very successful and have been shown to approximate many real-world problems. Neural networks are made of an input layer, hidden layer/s and an output layer. The layers are each made up of neurons.

Supervised Learning The branch of machine learning that deals with predictive analytics. The predictive component is enabled by retraining the model on labelled data. This allows the model to determine decision boundaries for the classification/regression predictions.

Waste In the context of a manufacturing plant, waste typically refers to the product that is not fit for consumption. This includes the effort, time and energy put into processing the defective product from when the defect was introduced to when the product was scrapped. The energy spent processing the defective products is deemed as wasted.

1.3 Objectives

The casthouse expressed interest in improving the quality control aspect of the casthouse production process. The main objectives are:

1. Reduce customer complaints and downstream quality issues caused by inclusions.
2. Improve time-to-reaction for anomalous situations, when the metal quality is substantially low.
3. Improve the capability for root cause analysis by identifying the inclusions responsible for low quality.

The positive outcomes for improved quality control include increased customer satisfaction, reduced downtime which improves the likelihood of meeting and exceeding production targets, and a reduced carbon footprint as a result of waste reduction.

1.4 Problem Statement

The availability of the MV20/20 system presents the opportunity to determine an optimal configuration of algorithms to achieve the quality objectives. The problem can therefore be described as the need to:

P1 Develop an algorithm to determine whether a cast is a pass or fail. This would reduce downstream processing waste and possible customer complaints.

P2 Develop a per-cast algorithm to determine the responsible inclusion type. This would minimise the time to determining the root cause, as the inclusion origins are well understood by the business.

1.5 Rationale

The Hulamin business has embarked on a plant-wide program to improve its impact to society and reduce its carbon footprint. This program is aimed more specifically at reducing waste, consumed energy, runaway greenhouse gas production and the rate of customer complaints (information obtained during a meeting conducted in October, 2019, by Mr. Rodney Green-Thompson, Manufacturing Director, Hulamin Rolled Products). As a result, the business has made investments in technologies that improve the measurement of critical plant variables.

The procurement of the MV20/20 system for real-time cleanliness measurement at the casthouse, which is the first process of the production line, has allowed for the collection of data in an objective, consistent, reliable and repeatable manner.

The collection of data therefore presents the opportunity for the application of exploratory data analytics and supervised learning. These techniques can therefore provide the business with the needed insights for the improvement of quality control as a step towards achieving the business program.

1.6 Requirements and Functions

The requirements from the business have been elicited through verbal communication during formal and informal meetings with stakeholders. The main requirements are based on the business objectives and are outlined below.

R1 Exploratory Data Analytics. This requirement is essential as a first step in understanding the nature of the data. It is also critical in determining whether the behaviour of the plant is normal or if it is anomalous.

R2 Event predictions. This requirement is based on the prediction for whether a cast is a pass or a failure, and is critical for ensuring that defective products do not get processed further, thus increasing waste. A two-stage model is required and is to perform as follows:

- R2a A supervised learning model is to be built in order to classify each cast as a pass or fail.
- R2b For the failed casts, a supervised learning model is to be built in order to classify the inclusion responsible.

The corresponding functions that need to be performed in order to achieve the given requirements are:

F1 Descriptive analysis of dataset. This function involves ingesting the data, analysing it and producing the statistical analyses of the dataset.

F2 Development of supervised learning models to perform predictions on:

- F2a The condition of the cast metal.
- F2b The responsible inclusion for the failed cast.

For acceptance testing, the following table summarises the test set based on the requirements and functions:

Table 1.2: Tests to be performed during acceptance testing. The solution is deemed a success if it can pass these test cases

Test	Description	Requirement	Function
T1	Descriptive statistics of the dataset are presented. The descriptive statistics should contain basic statistical analyses, correlations and distributions of the dataset descriptors.	R1	F1
T2	Classifier performance is to be presented by means of a confusion matrix, and a table showing the metric scores for each target metric.	R2	F2

1.7 Acceptance Test Procedure

The solution will be considered a success, according to a meeting held with Thirushan Govender, Process Engineering Manager, Hulamin, Sakhile Kubeka, Process Specialist, Hulamin, and Mani Ramdeen, Remelt Manager, Hulamin, when the following metrics are achieved. During the meeting, baseline performance metrics were determined by the business based on the nature of the business, domain knowledge, and business performance targets that are set yearly. The business targets can be summarised in the following table:

Table 1.3: Business targets for solution acceptance

Business metric	Business target
Customer complaint rate	66% reduction
Inclusion detection rate	$\geq 90\%$
False alarm rate	$\leq 10\%$
Availability	$\geq 95\%$
Reliability	$\geq 95\%$
Error rate	$\leq 5\%$

These targets constitute the minimum performance that is to be achieved by the solution, as it would ensure that he business realises the projected benefits. In terms of supervised learning, the requirements translate to classification metrics that need to be met by a machine learning model. The key metrics of classification models are discussed below.

1.7.1 Accuracy

The accuracy of a classification model is the ratio of correct predictions to the total number of predictions:

$$Accuracy = \frac{n_{correct\ predictions}}{n_{total\ predictions}} \tag{1.1}$$

Accuracy is a good measure of how well the model makes correct predictions, and does not account for the imbalance in the class distribution. The accuracy is 1 minus the error rate, which is one the business metrics. The target accuracy is therefore 95%.

1.7.2 Precision

Precision accounts for imbalance in the sizes of classes, as opposed to accuracy, which only accounts for the number of predictions regardless of classes. This ensures that all classes can be evaluated individually, as it represents the reliability of the model per class. The precision is given by:

$$Precision = \frac{n_{true\ positive}}{n_{true\ positive} + n_{false\ positive}} \tag{1.2}$$

where each class's true positive and false positive rates can be used. Precision is equivalent to reliability in terms of the business metrics, and is targeted at 95%.

1.7.3 Sensitivity/Recall

The sensitivity of a model is the ratio of correctly predicted positives from a class to the total samples from that class.

$$Sensitivity = \frac{n_{true\ positive}}{n_{true\ positive} + n_{false\ negative}} \tag{1.3}$$

The sensitivity, in terms of business metrics, is represented by inclusion detection rate. The sensitivity of the model ensures that samples with low quality can be identified and appropriate action taken, which could be re-work for severe cases and concessions for mild cases. Sensitivity is important in ensuring that the model can detect low quality conditions of the metal, thus preventing potential customer complaints or wasted effort of processing defective material. It is therefore set at a target of 90%.

1.7.4 Specificity

The specificity is the ratio of correctly predicted negatives from a class to the total number of negatives from the class:

$$Specificity = \frac{n_{true\ negative}}{n_{true\ negative} + n_{false\ positive}} \tag{1.4}$$

The specificity of a model determined how well the model can reject disturbances. In terms of business metrics, it is 1 minus the false alarm rate. The target is therefore 90%.

1.7.5 F1-Score

The F1-score is the harmonic mean of the precision and sensitivity:

$$F1 - Score = 2\frac{Precision \times Recall}{Precision + Recall} \tag{1.5}$$

The F1-score is not necessary as an independent metric as it is used when the precision and sensitivity are equally important. Since the precision is a more important metric in this case, it will gain higher preference than the sensitivity. The F1-score metric is therefore not used.

1.7.6 Kappa

Cohen's kappa is a metric for estimating the agreement of the classifier with the classes based on their respective proportions:

$$\kappa = 1 - \frac{1 - p_0}{1 - p_e} \tag{1.6}$$

, where p_0 is the observed agreement and p_e is the expected agreement. According to Landis and Koch , values greater than 0.4 indicate moderately increasing agreements with a perfect agreement at a value of $\kappa = 1$ [8]. For this work, a value of > 0.5 is regarded as sufficient for indicating agreement in the presence of imbalance.

1.7.7 Summary of Target Metrics

A summary of the target metrics for the primary classifier is given in the following table:

Table 1.4: Target metrics for sample result target respondent

Performance Metric	Target	95% CI
Accuracy	0.95	0.9 - 1
Precision	0.95	0.86 - 0.95
Sensitivity	0.9	0.86 - 0.95
Specificity	0.9	0.86 - 0.95
Kappa	> 0.5	N/A

For the secondary classifier, which classifies the responsible inclusion type in the event of a failed sample, the following metrics are to be met:

Table 1.5: Target metrics for inclusion type target respondent

Performance Metric	Target	95% CI
Accuracy	0.95	0.9 - 1
Precision	0.95	0.9 - 1
Sensitivity	0.8	0.76 - 0.84
Specificity	0.8	0.76 - 0.84
Kappa	> 0.5	N/A

The sensitivity and specificity are lower than for the primary classifier. This is because it would be more difficult to identify a single inclusion type in cases where there is more than one inclusion type present in the metal. Also, the classification of inclusions provides a benefit of faster root cause analysis, and is not directly linked to client-facing metrics.

1.8 Hypotheses

The following hypotheses are aimed at addressing each of the requirements and functions in the previous subsection:

H1 The calculation and plotting of the mean, standard deviation, min, max and variance will provide basic statistical analysis. The plotting of univariate distributions and a multivariate correlation plot will provide a comprehensive understanding on the nature of the dataset.

H2 The development of a machine learning model like a logistic regressor, support vector machine, or neural network with optimised hyperparameter tuning using 10-fold repeated cross-validation can achieve the business target metrics for a classifier.

1.9 Scope and Limitations

The work is subject to the following limitations:

1. MV20/20 ultrasonic sensor was used on a batch of samples representing an Aluminium alloy called 5182, which is among the lowest in quality regarding the presence of inclusions.
2. The datasets are collected from one furnace and localised to the end of the launder where quality measurement is of paramount importance.
3. The critical numerical features are:
   (a) Cleanliness,
   (b) MV Grade and

(c) Inclusion count.

4. The critical categorical features are:
   (a) Inclusion type, and
   (b) Result (passed or failed),
5. The inclusion types are limited to MgO, FeO and Spinel.
6. The development of all the algorithms, generation of images and chart and all source code is done using open source technologies including R and Python.

1.10 Dissemination Plan

1.10.1 Research Findings

This research is based on operations at Hulamin Rolled Products. As such, the technical staff at Hulamin Rolled Products are allowed to disseminate the findings, tables, graphs and data contained in this report as they see fit.

1.10.2 Intended Audience

This document is primarily intended for the stakeholders at Hulamin Rolled Products. The stakeholders particularly include engineering management, engineers and directors who have interest, power and influence in the realisation of the deployment of this project.

1.10.2.1 Research Application

In the event that Hulamin Rolled Products decides to implement a part or all of the findings of the research, they are free to do so without the involvement of the author. In the event that the author is required, the author undertakes to take reasonable steps to avail themselves. This is because the outcomes of a successful deployment of this research have a positive impact on the environment, and have economic benefits to the business and its employees.

1.11 Document Outline

The rest of the document is presented as follows:

Chapter 2 contains the background of the work.

Chapter 3 details the design and modelling of the system.

Chapter 4 discusses the results and presents analyses.

Chapter 5 concludes the research with major findings.

Chapter 2

Background

2.1 Literature Review

2.1.1 Review of the Technologies Used in Measuring Metal Quality

2.1.1.1 Metallography

Figure 2.1: PoDFA system [10]

PoDFA (porous disk filtration apparatus) is a technique for collecting inclusions inside a fine porosity filter disk. The molten Aluminium is extracted from the cast and poured into a heated crucible (in figure 2.1 on the left). The crucible is then placed inside the PoDFA device (in figure 2.1 on the right), where the metal is passed through the filter disc, which traps the inclusions. Typically, about 1kg of metal is used. Once ready, the crucible is removed from the device and the now solid metal is extracted from the crucible. The bottom part of the metal, which is connected to the filter, is sawn off and taken for metallographic analysis for the identification and quantification of inclusions [10]. A user analyses the inclusions to determine the types and estimate the concentration.

An illustration of the metallographic observation is shown in the Figure 2.2:

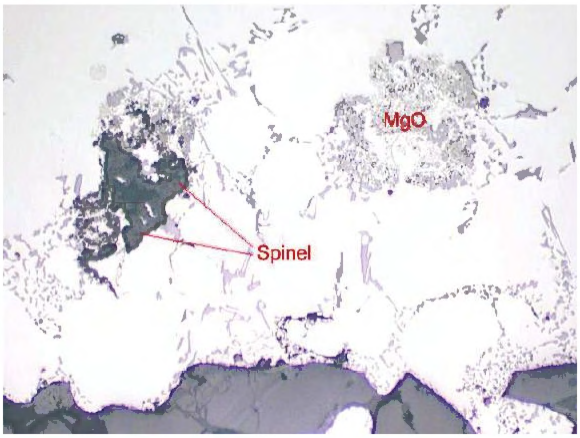

Figure 2.2: Metallographic image showing a PoDFA sample [11]

The PoDFA technique has its strength in its ability to identify the inclusions. Popular inclusions are Magnesium Oxides, Spinel and salts as can be seen from figure 2.2 [12]. It falls short as an empirical quality analysis tool, as it depends on user experience and a carefully followed process of analysis to correctly derive the quality of the metal. It is also not capable of real-time analysis as the metal needs to be extracted, allowed to cool and finally processed in the lab in order to get results.

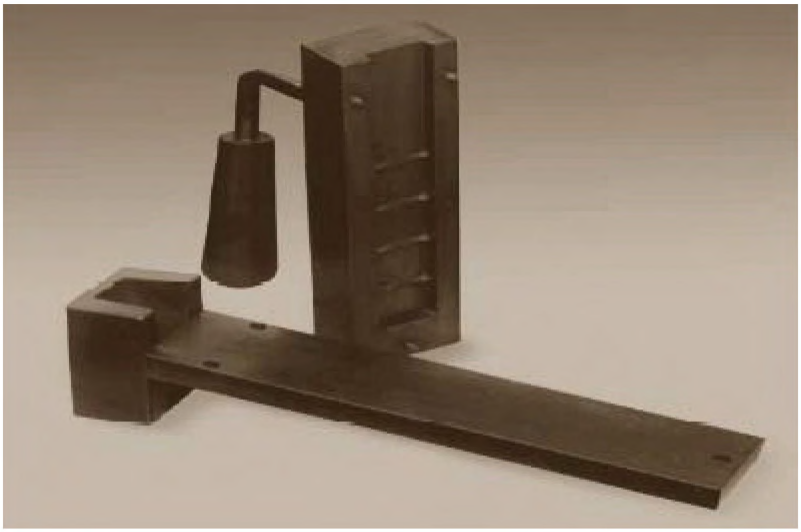

Figure 2.3: K-Mold system [13]

The **K-Mold** technique is a visual technique and provides a macro analysis of inclusion types and concentration [13]. It has proven to be relatively easy and quick to implement and receive results as it uses a fracture test [12, 13]. The method involves pouring about 400g of molten metal into a preheated notched bar mold (shown in figure 2.3). Once the metal is cast into solid cubes, the cubes are emptied onto the floor or a solid platform. They are then broken into several pieces. The fractured pieces are then analysed for macroscopic inclusions, either visually or with a magnifying glass. Based on the visual results, the cleanliness of the metal is estimated using a K value s follows:

$$K = \frac{S}{n} \tag{2.1}$$

, where

K is the number of inclusions found in one piece of a fractured sample,

S is the total number of inclusions found in all the samples, and

n is the total number of samples examined.

While K-Mold is a quick and easy method for assessing metal cleanliness on the shop floor, it is limited by the fact that only inclusions that can be resolved by the eye can be observed. Also, the counting of the inclusions is a subjective process. The fracture samples are generally not the same size, so an accurate estimation of the number of inclusions on the surface is difficult to achieve. Also, similarly to the PoDFA technique, it is not real-time due to the cooling process.

2.1.1.2 Electrical

Figure 2.4: Prefil system [14]

The **Prefil** (pressure filtration) technique offers a higher sample rate as it trends the weight of the metal as it flows through a filter. A scoop of molten metal is collected and poured into a reusable crucible which then flows the metal through a filter. The filter traps inclusions, and the flow rate drops as the inclusions build up on the filter. The slope and overall shape of the weight is used to estimate the cleanliness of the metal [14]. If the drop is too steep, it indicates the presence of more inclusions than if it is not.

In addition to trending the cleanliness, the Prefil technique provides the filter sample for metallographic analysis. Hence the inclusion types and concentrations can be determined together with the trend.

An illustration of the Prefil trend is shown in Figure 2.5:

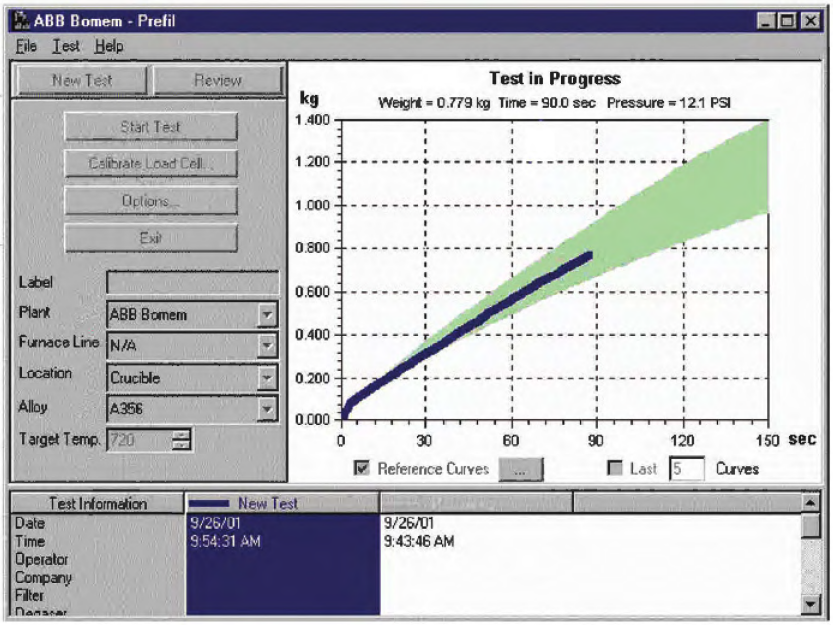

Figure 2.5: Prefil trend [14]

Prefil is a more quantitative approach to measuring metal cleanliness. The data is also digitally retained for offline analysis. However, the slope is subject to interpretation and is subject to a large variance. The system is also manual and requires that scoops of metal be taken at intervals in order to have continuous measurement. The system is therefore more suited to indicating when there is a significant volume of inclusions, as the curve might trace outside the tolerance region (indicated in Figure 2.5 by the green shading). It is, however, still a manual process as metal needs to be continuously fed into the crucible, and filters changed regularly to ensure good performance.

Figure 2.6: LiMCA system [15]

The **LiMCA** (Liquid Metal Cleanliness Analyzer) method is another method that uses an electrical signal in the measurement of metal cleanliness. The rods at the end of the arm (figure 2.6) are lowered into the launder where the metal is flowing. Every minute, about 40g of metal is sucked into an electrically insulated tube with an orifice. There are four rods dipped into the metal, which have a constant current applied to them as well as to the orifice inside the tube. The current flows through the conductive metal both outside and inside the tube, and if there is an inclusion, it causes a spike in the resistivity of the metal inside the tube. This resistivity is detected as a differential voltage between the metal inside the tube and the metal outside. Therefore, the size of the spikes and the number of spikes is used to determine the inclusion size and concentration [16].

LiMCA has been by far the industry standard for real-time measurement of metal cleanliness. It provides fully objective, real-time measurements of the metal cleanliness, which means decisions can be made faster. A screenshot of the measurements done by a LiMCA system is shown in the Figure 2.7:

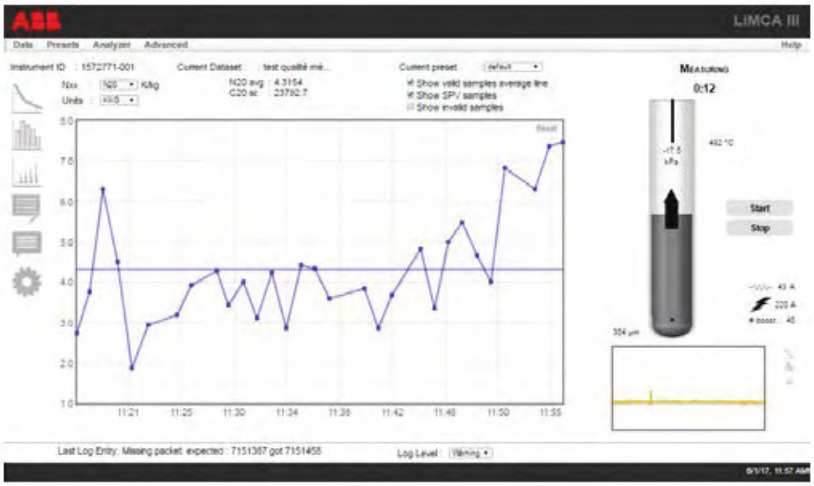

Figure 2.7: LiMCA operator display of measurement data [16]

The screenshot shown in figure 2.7 illustrates measurements taken from a cast which ran for about 40 minutes. The system took measurements once every one and half minutes on average, and plotted the trend in real-time. It provides a huge advantage as the measurements are not susceptible to human bias, and the relative effort applied to obtaining the measurements is less than that for the previously mentioned methods. This is what makes the LiMCA system one of the leading industry solutions. The system, however, falls short in terms of reliability. The glass tube has been reported to be fragile and prone to breaking during operation, thus causing delays in production (during a meeting held in February, 2020 with Mr. Nelson Dlamini, Process Technician, Remelt, Hulamin Rolled Products). This has a substantial impact on the availability of the solution.

Figure 2.8: MV20/20 system

The **MV20/20** system uses pulsed ultrasound waveforms in the detection of inclusions. A pair of piezoelectric rods are suspended into the molten metal as it flows through the launder. There is also a reflective steel plate placed at the bottom of the launder, directly underneath the rods. Pulses of ultrasonic waveforms are transmitted by one rod into the metal and reflected off the steel plate into the other rod [17]. The return signal is attenuated by the Aluminium and the inclusions. As the system is calibrated to account for the attenuation by Aluminium, the additional attenuation is attributed to inclusions. This technique allows the system to measure the cleanliness index as a fraction of the baseline cleanliness for pure Aluminium, and the count and size distribution of the inclusions [18].

A screenshot of the MV20/20 system is shown in the Figure 2.9:

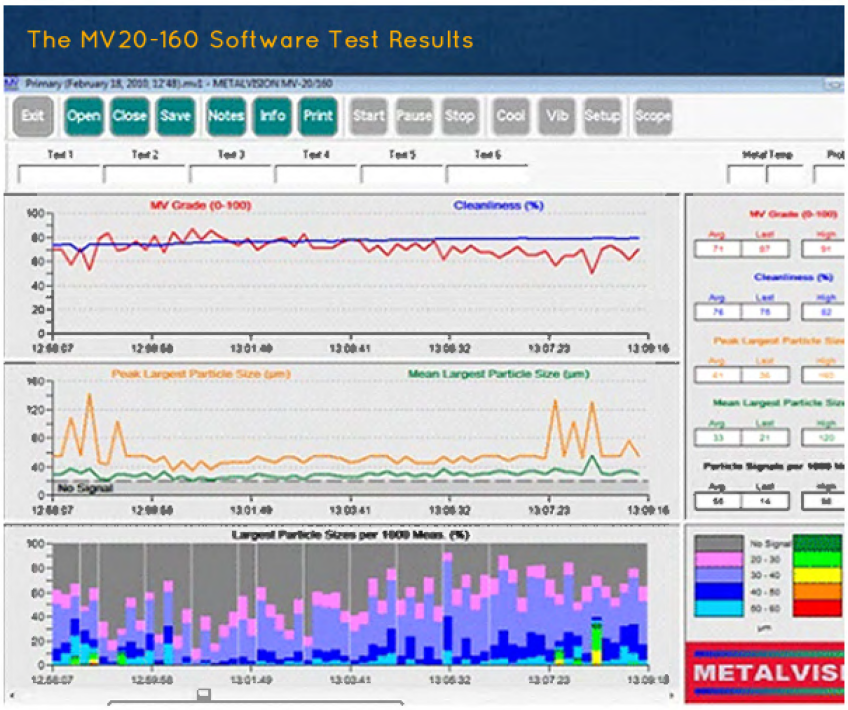

Figure 2.9: MV20/20 operator console [17]

The system is capable of sampling a large amount of the cast as the reflective plate is placed at the bottom of the launder,

thus making the entire height of metal visible to the system. This is a considerable advantage over other systems; the MV20/20 is capable of sampling well over 10% of any cast [17, 18]. This makes the sample size statistically representative of the cast. In addition, the system needs virtually no human intervention during operation. The trace of data shown in 2.9 shows the data collected during operation without human intervention. This level of autonomy is another substantial advantage as data collection happens autonomously.

2.1.1.3 Summary

Based on the different manual methods presented, the PoDFA solution is preferred over the K-Mold method. This is because the PoDFA method involves the classification, count and distribution of inclusions through metallographic analyses, whereas the K-Mold technique only provides a visual analysis of macro inclusions though results are available in a shorter period of time.

The electrical methods offer real-time analysis, but it can be seen that the MV20/20 system provides the most information relating to the quality of the metal. It also samples much more metal due to its ultrasonic sensor.

2.1.2 Review of the MV20/20 System Used in Non-Destructive Testing

Sound waves propagate through a medium and are thus dependent on the elastic modulus and density of the medium. As a consequence, sound waves are absorbed and scattered differently according to the changes in air pressure in the presence of wind [19]. In non-destructive testing, frequencies of ~2MHz have been commonly used to characterise the quality of different media using ultrasound [20]. These applications form part of nondestructive testing and have become an increasingly popular method of testing structures for integrity and liquids for foreign particles. EM waves do not possess these advantages as they are used for long range transmissions at these frequencies. This is because this frequency range is considered low in the EM spectrum.

An illustration of a basic ultrasound system used in flaw detection is shown in the Figure 2.10:

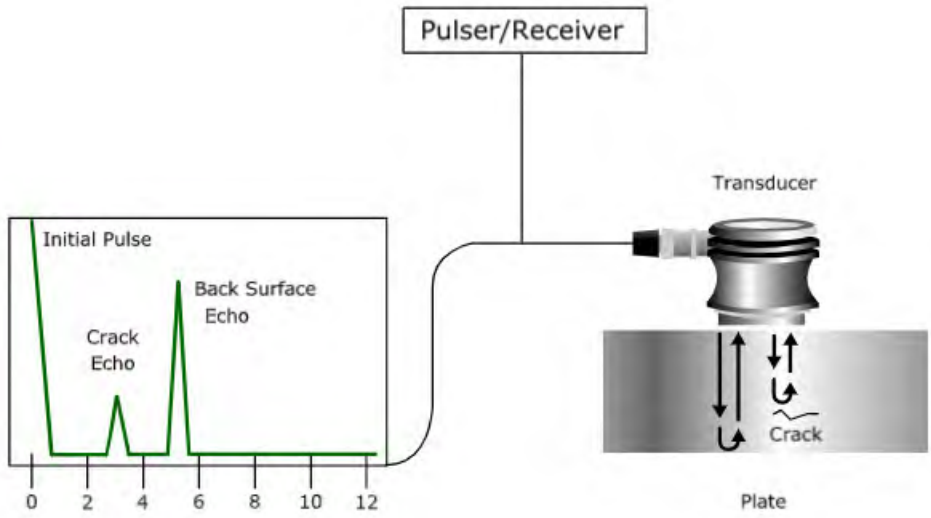

Figure 2.10: Basic operating principle of ultrasound

Shown in 2.10 is a pulsed ultrasound system. An ultrasonic waveform pulse is generated and transmitted into the homogeneous medium. Prior knowledge of the medium's elastic modulus and density allow for the determination of the speed of the waveform. Using this speed, the unambiguous range of the transmitted waveforms can be determined from the Pulse Repetition Frequency (PRF) of the ultrasound system:

$$Range < \frac{v}{2 \times PRF} \tag{2.2}$$

, where

$\boldsymbol{v}$ is the velocity of sound in the medium, and

$\boldsymbol{PRF}$ is the pulse repetition frequency of the ultrasound waveform.

Echoes from the crack are received within the pulse repetitive period, and are therefore resolved in order to determine the presence of either foreign material or defects in the medium. The summation of the returns gives the number of unwanted deformities/particles in the medium. In addition, a homogeneous medium can be used to calibrate the system to give a clarity of 100%, where the presence of foreign particles and cracks will cause absorption and destructive interference that will result in an attenuation factor. This reduced clarity can therefore be used as an indicator of the homogeneity of the medium [18, 20].

2.1.2.1 Front-End

The MV20/20, developed through some years of research at the University of Toronto, is a specialised application of non-destructive testing (NDT) of liquid Aluminium. It uses an oscillator to generate electric pulses that are sent to a piezoelectric transducer. The pulses are generated at 2.25MHz, which has been selected as the optimal frequency for the resolution of particles ranging from 20 to 200μm [18]. A second transducer is used to convert return signals to electric form, so that they can be sampled and processed by the digital signal processor for detection. A diagram illustrating the transceiver is shown below:

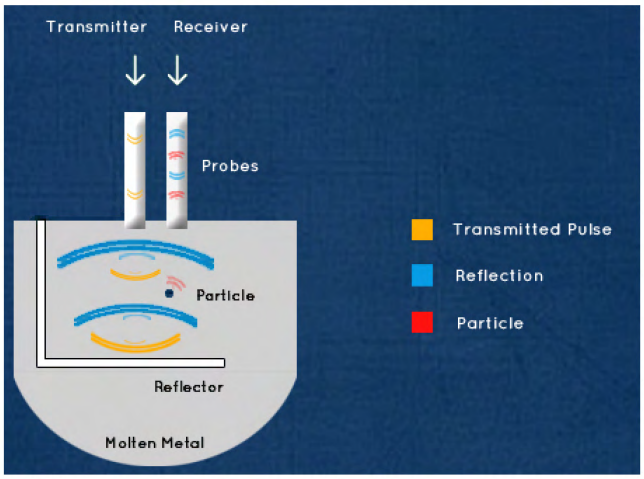

Figure 2.11: MV20/20 transceiver diagram [18]

2.1.2.2 Signal Processing

The return signal is passed through an attenuator, low-noise amplifier and filter stage before being converted to digital format.

The architecture of a typical ultrasound system is shown below:

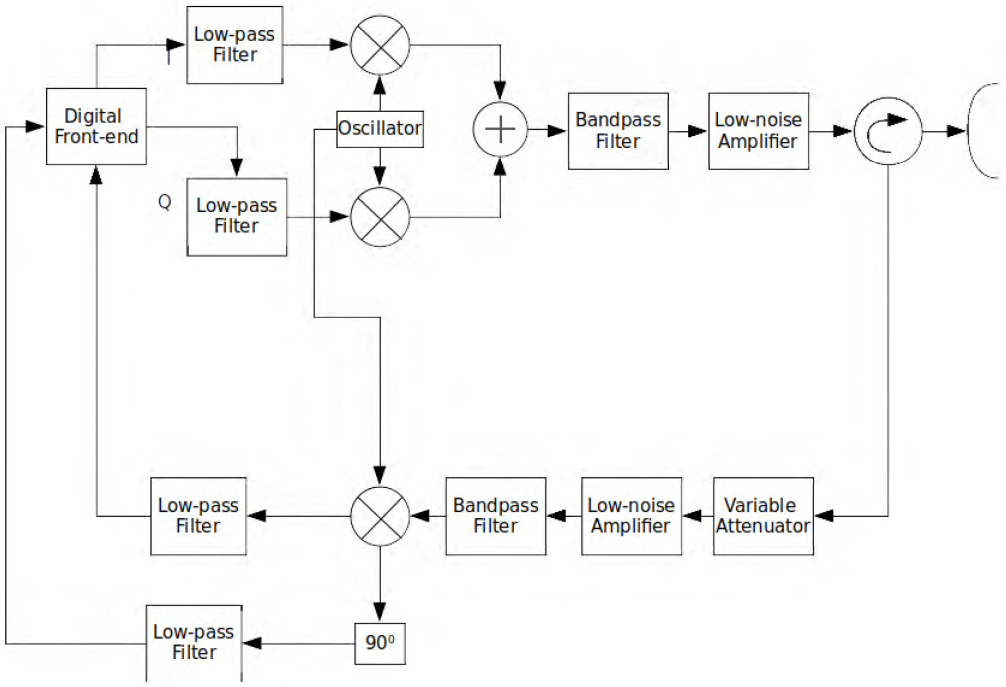

Figure 2.12: Ultrasonic system architecture. The source signal originates from the oscillator

The signal is typically made coherent in order to improve the signal-to-noise ratio. This also ensures that the phase, which is important for finding range, is kept constant. Once the signal is available in memory, it is processed with a comparator to perform particle detection, sizing and counting. The summation of the return signal within a full period, aggregated over 10 periods is used to measure the cleanliness of the metal. This is because any present inclusions will have absorbed and scattered the transmitted pulse, even if they are outside the detection range of the system.

2.1.3 Review of Data Analysis Best Practices

2.1.3.1 Response Surface Methodology

The response surface methodology technique aims to find an approximation of the response function to multivariate inputs using Ordinary Least Squares [21]. It approximates a statistical function as a response/s to the multiple input variables. While it has proven successful over the years, it has the main shortcoming that is still an approximation, so may not necessarily be optimal. The error margin could imply lost potential in large manufacturing industries. For this reason, a machine learning approximation solution is pursued for the potential benefits of quality optimisation.

2.1.3.2 Business Analytics

A proven process for analysing data in order to gain insights and inference capabilities is business analytics [23, 24]. Business analytics is a set of processes that involve extracting useful insights from data so as to optimise business performance using an empirical approach [25, 26]. The business analytics process is divided into four components:

1. Descriptive analytics. This entails analysis of historical data to understand the nature of the business process [27]. Typical outputs are statistical explanations of the data, trend analyses and other descriptive plots.

2. Diagnostic analytics. This entails analysis of historical data to understand the relationships between events (cause and effect) [28]. Typical outputs include correlation plots.

3. Predictive analytics. This includes the use of historical data to predict future events [29]. Typical outputs include future points with associated mean squared errors for regression, and a confusion matrix for classification .

4. Prescriptive analytics. This is the determination of the best future scenario based on historical and current trends [30]. Typical outputs include prescriptions of the best configuration of the business process, or specific actions in order to improve current performance or prevent predicted losses.

For this work, the applicable components used are descriptive, diagnostic and predictive analytics, as the data is available for exploration and predictive modelling. The prescriptive component requires domain expertise and reinforcement learning to maximise the value of the prescriptions. It therefore falls outside the scope of this work.

2.1.3.3 Machine Learning

Machine learning is a specialised field of data science where an algorithm is taught to learn on data [31]. This is done in contrast to traditional learning methods, where logic is manually programmed. A machine learning algorithm is trained on historical data, and it recognises trends and patterns in the data, either for exploration, diagnostics or autonomous predictions. One of the biggest advantages of machine learning is its ability to automate workflows.

Some of the most popular classification algorithms include logistic regression, naive Bayes, decision trees, support vector machines and neural networks including deep learning networks. For small datasets ($<$ 1000 observations), relatively high complexity algorithms tend to overfit and hence produce large scaled errors [32]. It therefore makes sense to consider relatively low complexity models. Naive Bayes algorithms assume independent inputs, which can be a disadvantage within industrial processes as faults symptoms tend to be related to each other [33]. Decision trees are robust against outliers and handle missing observations well. Their main disadvantage, however, is that they are prone to overfitting on small datasets [34].

In this study, four algorithms are selected and compared in order to achieve target classification metrics:

1. Logistic regression. The logistic regression model is considered as it is one of the most popular and simplest to implement. The computational complexity of the algorithm is also low .
2. Support vector machines SVM). SVMs are also popular and are known to generalise over many different types of datasets. One of the advantages of SVMs is the kernel function, of which there are several variations. This makes them adaptable to a wide range of datasets.
3. Multi-layer perceptron (MLP). MLPs are neural networks in their basal form. They are highly flexible and have the ability to extract patterns and trends that are not obvious to humans or other computer algorithms.
4. Radial basis function network (RBF). This is a specialised neural network with radial basis functions as activation functions. These radial basis functions give the RBF network the ability to generalise over many complex, nonlinear datasets.

2.1.3.4 Machine Learning in Aluminium Casting

Based on existing literature, most applications of machine learning in the context of Aluminium casting pertain to the solidification process [35, 36, 37].

M. Torabi Rad, A. Viardin, G. J. Schmitz, and M. Apel presented the modelling of the alloy solidification process using a theory-trained deep neural network [36]. The solidification process is important in ensuring that that as the metal solidifies, it does not form cracks that negatively affect quality. The model is trained on simulated data points generated by simulated points based on theoretical mathematical models. Trained models can then predict solidification temperature, for example, based on input points. The novelty of the solution is in it being the first of its kind. While the solution can identify quality defects during casting, it is limited to only considering the macro-scale quality problem, and not defects trapped deep in the alloy.

In [37], a non-destructive testing method using X-ray is used to collect training data. Ellipsoidal synthetic defects are modelled and added into the training data, and a deep convolutional neural network is trained to detect and classify them. The solution works well, but would require substantial capital investment in industrial X-ray systems.

Within the context of metal cleanliness/alloy purity, the author was not aware of any existing machine learning work during the writing of this report. This forms part of the novelty of the work. The cleanliness of the metal as a quality metric is critical based on the following:

- Filter degradation can be quickly identified as a sharp decrease in cleanliness as more inclusions pass through the filter. This means that the process can be halted and the filter replaced before the entire cast is wasted. This is a potential benefit for root cause analysis.
- Melting and holding practices can be optimised by varying the settling times for more inclusions to settle to the bottom of the launder (stage 1 in Figure 1.1).

2.2 Methodology

2.2.1 Plan of Action

As the problem is a design problem, the solution is pursued using the traditional engineering design process [38]. From a systems engineering point of view, the approach follows a waterfall model as the requirements are fixed during the life of the project, and the main tasks are connected in sequence. The workflow used in realising this project is illustrated in the following flowchart:

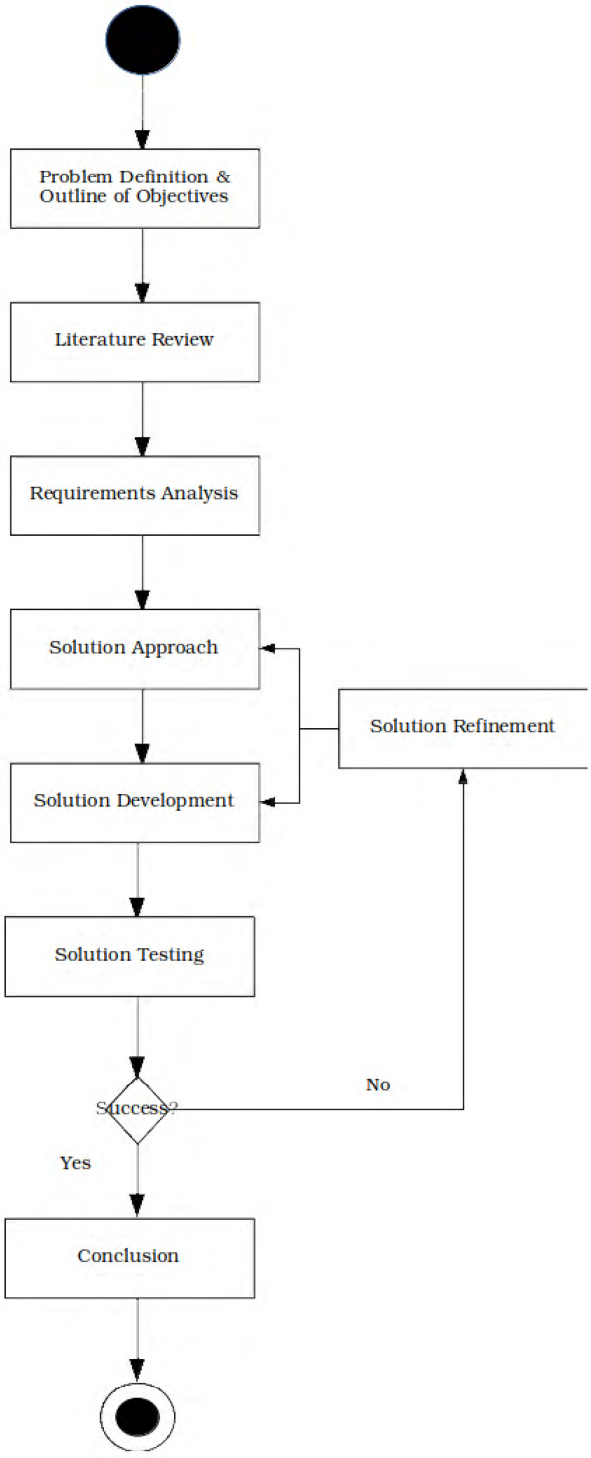

Figure 2.13: Methodology workflow.

An important thing to note from the flowchart is the refinement loop, which is critical for reviewing the algorithms used, and reviewing the parameters used in the algorithms as a function of target performance metrics.

2.2.2 Metrics Used

The performance metrics used, which are discussed under the acceptance test procedure subsection, are based on classification modelling results. These include accuracy, precision, sensitivity and specificity. In order to achieve these metrics, a confusion matrix is needed for each of the classifiers. A confusion matrix is a table that compares the predicted outcomes to actual results. It is an $n \times n$ matrix, where n is the number of classification categories. The rows represent the predicted classes while the columns represent the actual classes. In an ideal case, where the model obtained a perfect score, the confusion matrix would only have values in the diagonal, while all the other cells are zero [39].

2.2.3 Experiment Design

The experiment design in this project considers two approaches to determining the causality of the respondent variable.

2.2.3.1 Response Surface Methodology

The response surface methodology (RSM) approach uses the boundary values of the most influential independent variables to determine coefficients of a polynomial function [40]. The polynomial function represents the relationship between the respondent variable and all the descriptors. In an ideal case, the independent variable are varied one at a time, while the others are held constant and incremented in repeated experiments. For multivariate independent variables, the experiment design becomes more intensive to perform.

2.2.3.2 Machine Learning

Machine learning within a supervised context is an approach to establishing the relationship between the respondent and descriptors. This is done by projecting the multivariate input vector onto a hyperspace, and determining a point on the hyperspace that minimises the cost function. In order to achieve this, different algorithmic techniques are available, including regressors, decision trees, support vector machines and neural networks.

2.2.3.3 Summary

The following observations are made when comparing the RSM and machine learning approaches to experiment design:

1. The RSM approach has been effective in defining systems, but has the limitation of having to independently vary multiple variables in the case of a multivariate dataset. This can be the source of error. The machine learning approach overcomes this by creating a hyperspace of the entire input dataset, and therefore can account for all variables for increased accuracy.
2. The RSM approach produces a polynomial relationship between the dependent and independent variables. Machine learning algorithms provide better accuracy as they can use the kernel trick (support vector machines) for improved approximation. Neural networks have also been termed as universal approximators.

Based on this analysis, it is therefore decided that a machine learning approach to the design be used in order to offer more flexibility in approaches, and more accuracy for the multivariate dataset.

2.2.4 Data Collection Method

The dataset has been obtained from the Hulamin Rolled Products casthouse after a trial was conducted using the MV20/20 system. The MV20/20 system produces spreadsheets for the measurements. The data was saved in Excel (.xlsx) format, and contained in multiple files. In order to ingest the data into any meaningful platform, the widely supported format is .csv as it is more lightweight and relatively easier to interpret when applied to structured data.

In order to bulk convert the files, the *ssconvert* utility command from the Gnumeric project was used:

Algorithm 2.1 Bulk conversion using Bash scripting

```
for f in *.xlsx;
        do ssconvert "$f" "${f%.xlsx}.csv";
done
```

In order to tag the data with the respondents, the sample results from the metallurgy laboratory were obtained and analysed

2.3 System Requirements

2.3.1 User Requirements Analysis

This section makes considerations for the systems engineering approach to requirements analysis. This ensures that the solution complies with the different aspects of the business, legislature and sustainability. The following functional requirements are based on a meeting held with Mr. Jeremy Chetty, technical manager, Hulamin Rolled Products in February 2020.

2.3.1.1 Functional Requirements

The solution is composed of the following functionality:

FR1

Requirement Name	Descriptive Analytics of Dataset
Objectives	1. To understand the history, patterns and relationships of the process. 2. To serve as a foundation for diagnostic, prognostic and predictive analytics.
Rationale	Descriptive analytics provides an understanding of the nature of the data so that appropriate statistical techniques can be applied to achieve subsequent analytics.
Steps	1. Data ingestion. This involves producing a table showing the head of the dataset and basic statistics. 2. Trends. A plot showing the trends of the descriptors is to be developed. 3. Distributions. The distributions of the dataset are to be shown by means of boxplots.

Requirement Name	Development of Supervised Learning Classifier
Objectives	1. To predict whether the quality of a cast is acceptable or not. 2. For casts which failed, to predict the inclusion type responsible.
Rationale	Predictive analytics allows for an objective and autonomous approach to determining events as a function of the descriptors. This means the business can reduce the downtime due to investigations, and prevent defective metal from being processed downstream. It also provides autonomous determination of the problem source, which further improves the downtime due to investigation.
Steps	1. Identify classifiers based on descriptive analytics results. 2. Tune classifier hyperparameters using repeated cross-validation. 3. Assess model performance and recommend best model.

FR2

2.3.1.2 Performance Requirements

The following performance requirement was elicited from the business during a meeting with Mr. Thirushan Govender, Manager, Process Engineering, and Mr. Jeremy Chetty, Technical Manager, casthouse, held in February 2020.

PR1 Real-time classification. The classifiers shall take no more than 30s to perform classifications on input data.

2.3.1.3 Usability Requirements

It was agreed with the business that the implementation of a user interface should not form part of the requirements, and may be developed based on the outcome of this research. A user interface is therefore not considered as part of the scope of the project.

2.3.1.4 Environmental Considerations

From the initial meeting with Mr. Rodney Green-Thompson, Director, Hulamin Rolled products, held in October 2019, the need for the business to improve its carbon footprint was outlined. The business outlined the following types of waste that are associated with inclusions:

Table 2.1: Types of waste and their associated sources

Waste type	Description
Process - Cast	During production, inclusions detected in metal can cause the entire cast (~80 tons) to be scrapped for quality.
Process - Billet	During production, inclusions can cause a single billet to be scrapped for quality.
Process - Downstream	During production, undetected inclusions can cause surface cracks, scratches and tears on the metal in downstream processes.
Process - Logistics	Processed metal can fail formability tests while en-route to customers.
Customer Complaints	During processing at customer facilities, inclusions can cause metal to tear, fail formability tests, or they can cause leaks in the material.

The real-time analysis and prescription of metal quality conditions provides an opportunity to minimise the different types of waste listed in the table. This is because once defective material is detected, the rest of the cast can potentially be cured, and the loss will therefore be reduced to a single billet. In extreme cases where the cast cannot be cured, downstream waste, logistics waste and customer complaints can be prevented. The following environmental requirements are therefore presented by the business:

ER1 Reduction of cast waste by 80%. This implies the flagging of defective metal with an availability of $\geq 95\%$ (15% availability tolerance). An 80% reduction in cast waste will reduce energy and gas consumption in downstream production for defective metal by ~30% and ~20%, respectively.

ER2 Reduction of customer complaints by 66%. This reduction will ensure the business meets its continued improvement targets regarding customer satisfaction. This requirement is linked to the functional requirement FR2, which involves the implementation of a two-stage classifier. The classification of sample results is the direct requirement linked to this requirement.

These environmental requirements are outside the direct scope of this project. However, a successful realisation of the project would provide a basis for a study into the positive environmental impact of the solution. The outcomes of the study can then be used to assess whether the business has realised the environmental requirements listed. It is therefore a recommendation that an environmental impact study be performed once the solution is in deployment.

Chapter 3

System Model and Design

3.1 Suggested Approach

Business Analytics (BA) has become an increasingly important tool for data-based insights into business processes. It allows for deeper insights that might otherwise not be apparent at a superficial level. This allows for faster reaction times, more proactive approaches to business problems and business process optimisation. Based on the business analytics process, the following steps are suggested:

1. Data ingestion and descriptive analytics - data is ingested and optionally reshaped and pre-processed to ensure readiness for visualisation and modelling. Hypotheses can be developed based on the initial statistical analyses, including tests for normality, determination of distributions. In cases where the data has specific target setpoints, capability and stability can be determined using statistical process control techniques.
2. Diagnostic Analytics - correlation plots are generated in order to establish the relationships between descriptors.
3. Predictive Analytics - data is transformed and modelled in order to assess if predictions can be successfully made. This typically involves the use of machine learning models for regression or classification.
4. Once model training results are above set targets, experiments are designed in order to determine the best model and the best model parameters. The optimal model is then tested, integrated and deployed for benefit realisation.

The suggested approach for this project involves the application of BA as listed above. This approach is deemed as a sufficient approach for achieving the requirements of the project.

3.2 System Design

3.2.1 System Architecture

The following diagram depicts the architecture of the supervised learning classifier:

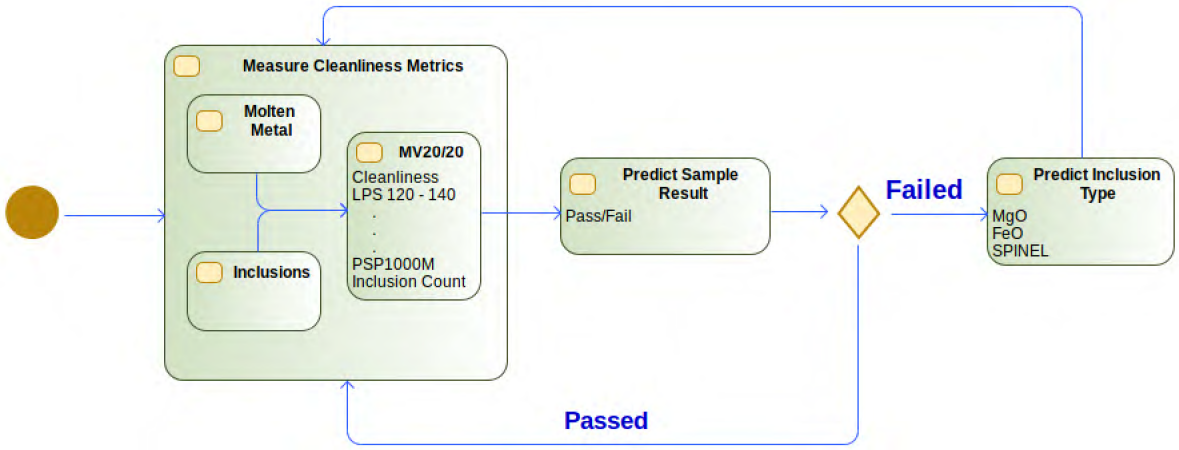

Figure 3.1: Supervised learning system architecture

The diagram shows the system flow from the measurement to the prediction of the inclusion type.

3.2.2 State Machine Diagram

The following state machine diagram shows the different states of the system:

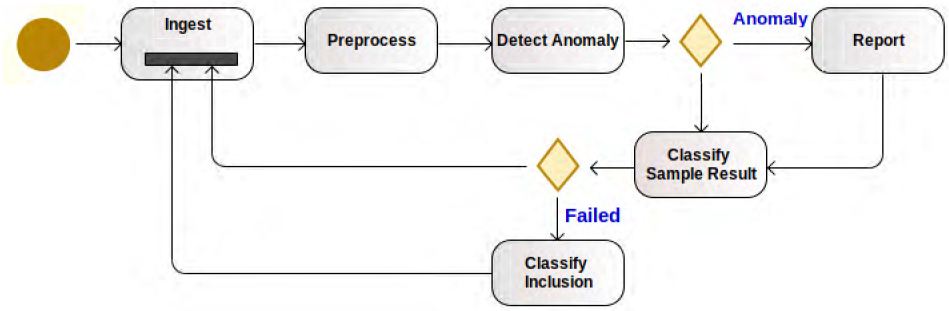

Figure 3.2: System state machine diagram

The states of the system are:

1. Ingest. This state involves ingesting an observation of data from the MV20/20 system.
2. Preprocess. This state involves standardising and centering the data. This ensures that the different ranges do not
3. Classify Sample Result. The sample result is classified by the first stage classifier.
4. Report. If an anomaly is detected, the system sends an email to a list of addresses provided by the business. The email is sent through a simple SNMP service.
5. Classify inclusion. If the outcome of the first stage classification is failed, the inclusion is classified.

3.3 Data Ingestion & Statistical Analysis

Data ingestion is the first stage to analytics. The dataset is parsed into a data frame from a file. The following algorithm shows how the data is ingested into program memory. The R programming language is used in this case.

Algorithm 3.1 Data ingestion

```
rm(list = ls())

d <-
  read.csv(file = "../Data/5182.csv",
           header = TRUE,
           stringsAsFactors = TRUE)
datalength <- length(d)
xdata = d[, 1:(datalength - 2)]

ydata_incl_type = d[, (datalength - 1)]
ydata_samp_res = d[, datalength]

test_data <-
  read.csv(file = "../Data/test.csv",
           header = TRUE,
           stringsAsFactors = TRUE)
xtest <- test_data[, 1:(datalength - 2)]
ytest_incl_type <- test_data[, (datalength - 1)]
ytest_samp_res <- test_data[, datalength]
```

A summary of the ingested data is shown in the following table:

Table 3.1: Input data summary. Trimmed represents the trimmed mean and mad is the median absolute deviation. se is the standard error

vars	mean	sd	median	trimmed	mad	min	max	range	skew	kurtosis	se
Cleanliness	55.17	2.13	55.00	55.20	2.97	50	62.00	12.00	-0.04	-0.22	0.11
LPS.120_140.__m	1.45	1.74	1.00	1.15	1.48	0	8.00	8.00	1.24	1.02	0.09
LPS.140_160.__m	0.47	0.70	0.00	0.33	0.00	0	3.00	3.00	1.34	1.03	0.04
LPS.20_30.__m	8.71	3.17	9.00	8.73	2.97	0	17.00	17.00	-0.06	-0.18	0.16
LPS.30_40.__m	17.34	10.21	17.00	16.77	10.38	0	50.00	50.00	0.47	-0.17	0.53
LPS.40_50.__m	9.16	4.03	8.00	8.80	2.97	0	22.00	22.00	0.75	0.53	0.21
LPS.50_60.__m	4.81	2.56	5.00	4.66	2.97	0	16.00	16.00	0.79	1.51	0.13
LPS.60_90.__m	4.19	2.07	4.00	3.98	1.48	1	13.00	12.00	0.91	0.70	0.11
LPS.90_120.__m	4.88	3.37	4.00	4.47	2.97	1	18.00	17.00	1.18	1.65	0.17
MV_Grade	59.75	4.31	60.00	59.63	4.45	49	72.00	23.00	0.23	-0.12	0.22
No.Signal	66.75	21.99	67.00	67.09	20.76	1	135.00	134.00	-0.17	0.35	1.13
PSP1000M	31.65	18.89	30.00	30.80	22.24	0	86.00	86.00	0.34	-0.58	0.97
INCLUSION_COUNT	24.69	19.63	20.50	22.42	20.02	1	73.00	72.00	0.78	-0.32	1.01
FILTERED_MASS	1.15	0.09	1.15	1.15	0.10	1	1.31	0.31	-0.03	-1.10	0.00
INCLUSION_TYPE*	1.99	0.82	2.00	1.99	1.48	1	3.00	2.00	0.01	-1.50	0.04
SAMPLE_RESULT*	1.62	0.49	2.00	1.65	0.00	1	2.00	1.00	-0.50	-1.75	0.02

The dataset has 16 features. Of the features, 14 are numeric and 2 are categorical (inclusion type and sample result). They are represented statistically by replacing the categories with 1, 2, and 3 for the inclusion type and 1 and 2 for the sample result. The variables LPS* represent the largest particle size. the sizes of the particles are indicated by the suffixes in microns. About half of the variables have a moderate to high level of skewness, indicating the asymmetry that exists in the distributions.

3.3.1 Numerical Features

Some of the important measurements from the MV2020 are:

- Cleanliness. This feature represents how close the metal is to pure Aluminium as a percentage. As all the products from Hulamin are Aluminium alloys, the cleanliness percentage cannot achieve 100%, and realistic targets can be set based on dedicated studies beyond the scope of this project.
- MV Grade. This is a MetalVision feature indicating the how potentially clean the metal could be if the signal attenuation due to inclusions was not present. Is it also strongly correlated to the cleanliness, although it is always higher.
- No signal. This feature maintains a count of the number of times a signal was not returned during a measurement sprint.
- Inclusion count. The inclusion count gives the number of inclusions detected during a measurement sprint.
- LPS_50_60 __m. This represents the number of particles ranging from $50 - 60\mu m$ in size. These are of particular interest as they tend to cluster and form bigger particles that affect metal quality.

Figure 3.3 shows a correlation plot of the numeric features of the dataset:

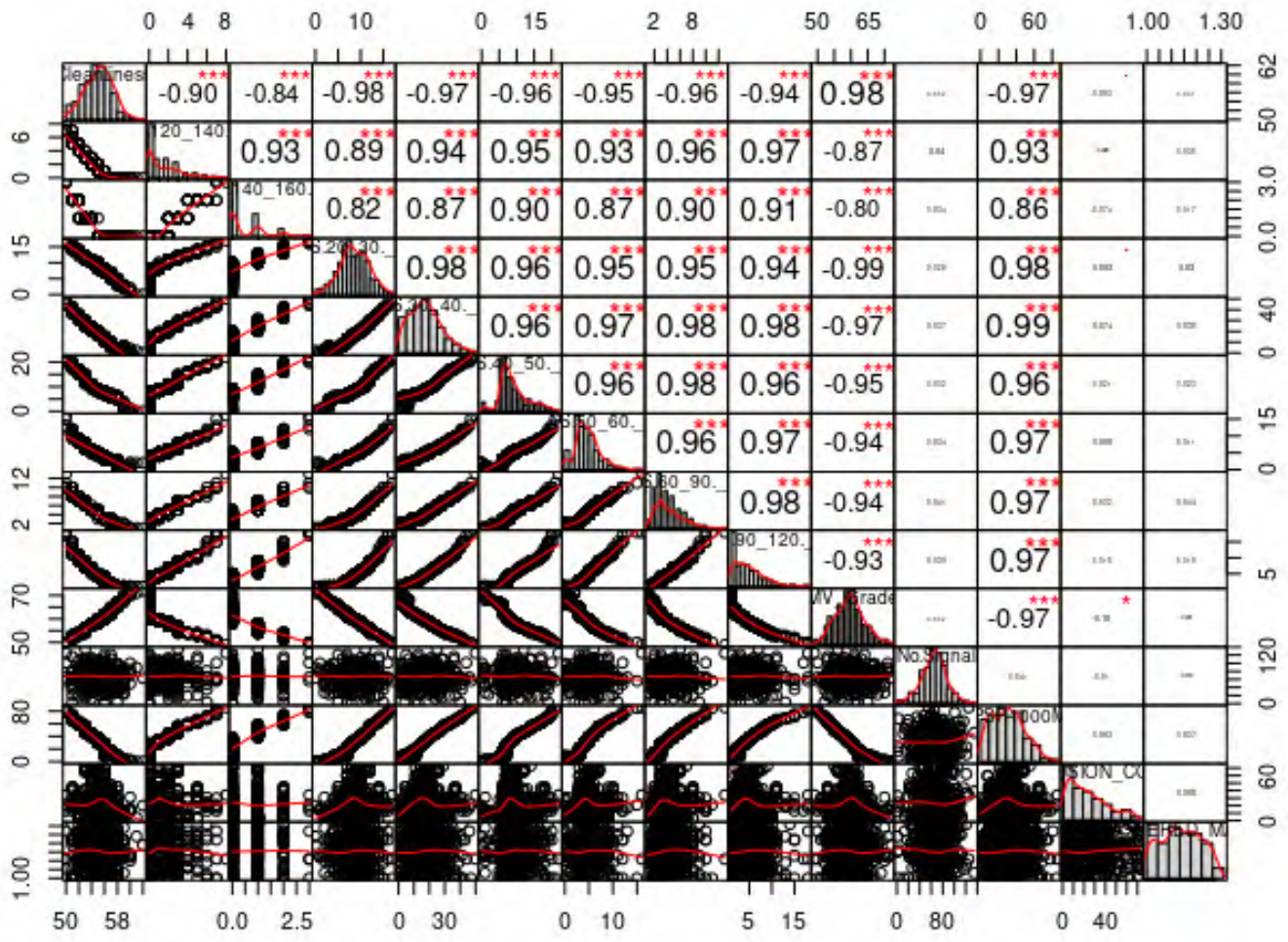

Figure 3.3: Correlation plot of the numeric features of the dataset

The dataset indicates many strong correlations, particularly among the LPS features. It is also noted that the Cleanliness feature has a strong negative correlation with the LPS features. This is an indication that the cleanliness of the metal is inversely proportional to the presence of inclusions, as expected. The inclusion count has very little correlation with any of the features. This indicates that the inclusion count cannot be used in isolation as an indicator for any quality condition of the metal.

3.3.2 Categorical Respondents

3.3.2.1 Inclusion Type

The inclusion type feature describes the inclusions mainly responsible for contaminating the metal. These inclusions have been found to originate from different upstream sources, including:

- Alloying - when creating the desired alloy, Magnesium is added as a constituent of the alloy. As the Magnesium oxidises when exposed to oxygen during processing, thin layers of MgO form on the surface and get mixed into the cast. The same is true for Iron.
- Refractory degradation - the refractory walls are subject to natural degradation due to wear and tear. During this wear, Spinel particles can mix with the metal.

The relative inclusion quantities for the inclusions are shown in Figure 3.4:

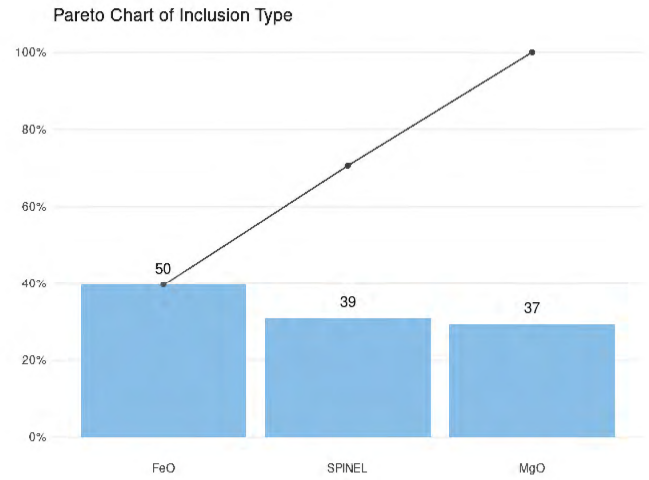

Figure 3.4: Inclusion type ratios

It is essential to keep the inclusion counts as low as possible so as to maximise the quality of the product. A scatterplot is shown in Figure 3.5, with the colouring based on the inclusion type:

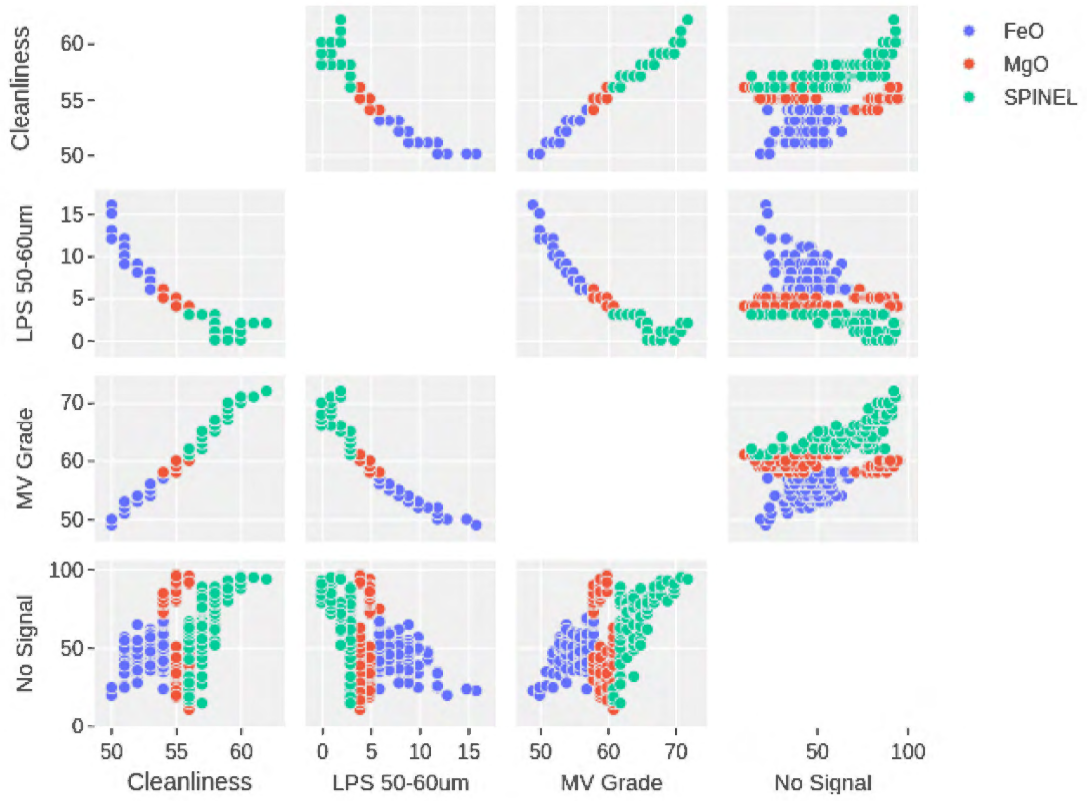

Figure 3.5: Scatterplot of numerical features coloured by inclusion type

The separability among the inclusion types is not as separable as the sample result.

3.3.2.2 Sample Result

The sample result represents the metal that failed either based on a downstream process or as a result of an inclusion-related customer complaint. This feature is critical in training the model as a target parameter, as the model is to predict whether the metal will pass or fail. The ratios of the results are shown in Figure 3.6:

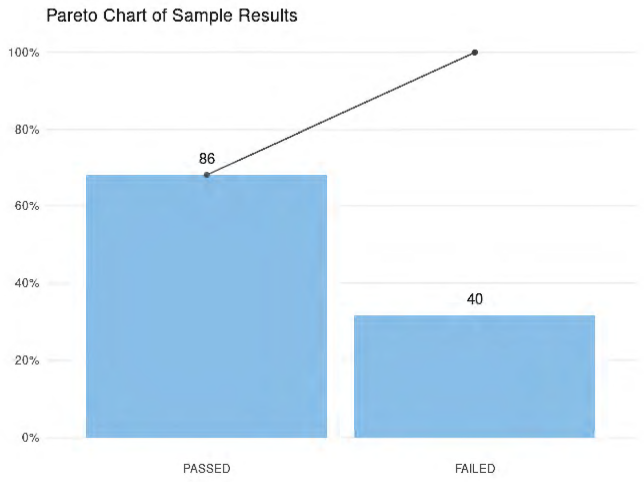

Figure 3.6: Sample result ratios

It is evident that the data for this category is unbalanced due to most of the samples passing. Ideally, the samples that failed must be kept as low as possible.

The scatterplot in Figure 3.7 shows the correlations between the features, coloured by the sample result categorical respondent:

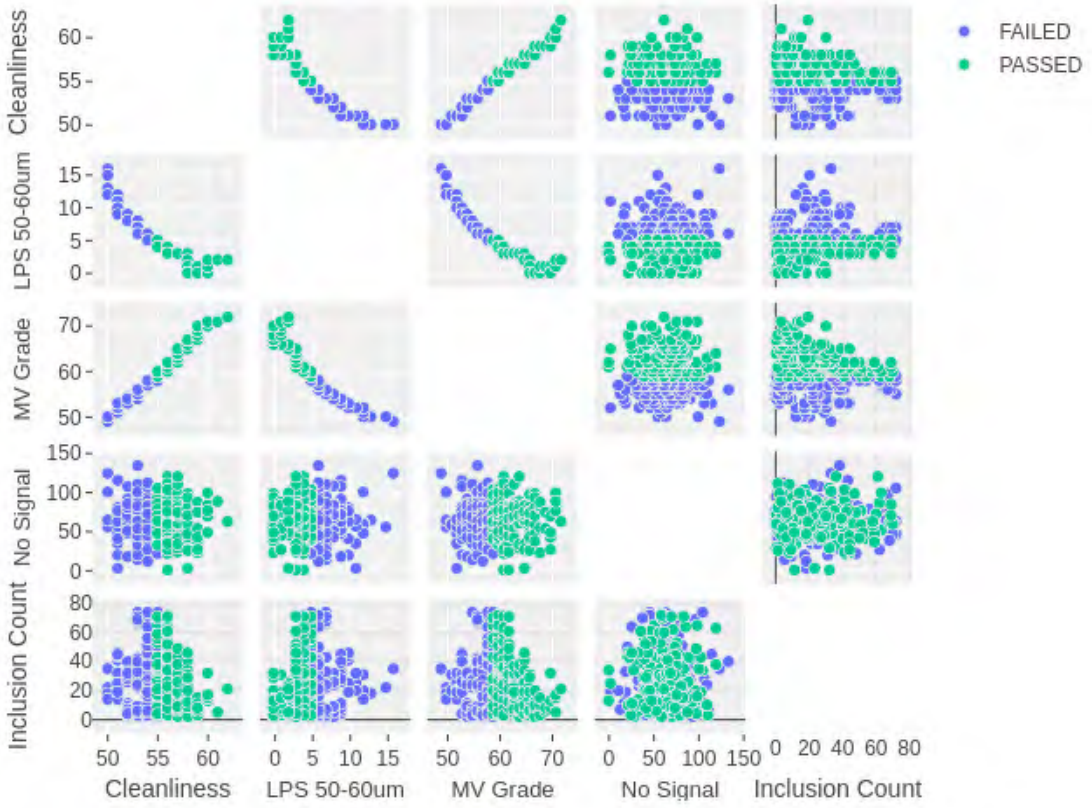

Figure 3.7: Scatterplot of numerical features coloured by sample result

The scatterplots show linear relationships between the cleanliness, MV grade and the LPS 50 - 60μm features. This is consistent with the fact that the MV grade is an estimate of the cleanliness without attenuation, and that the number of particles in the metal is inversely proportional to the cleanliness of the metal. The inclusion count and no signal features show no strong correlations to the other features. The "passed" category of the sample result shows a linear separation with all the features, except for some overlaps with the "failed" result around the centers. This is an indication that the cleanliness of the metal might have a strong influence on the result of the sample.

Another important note is the visually linear relationship between the sample result (passed or failed) and the features Cleanliness, LPS 50-60um, and MV Grade. Based on this, a solution based on the response surface methodology could potentially provide the required output prediction while keeping the solution relatively simple and computationally inexpensive. This, however, would only account for the features that are linearly correlated. The rest of the features as shown in Figure 3.3 are more sparsely correlated. In the interest of optimising the response of the solution, machine learning is the preferred solution.

3.3.3 Distribution

The distribution of the dataset is shown by the following boxplot diagrams:

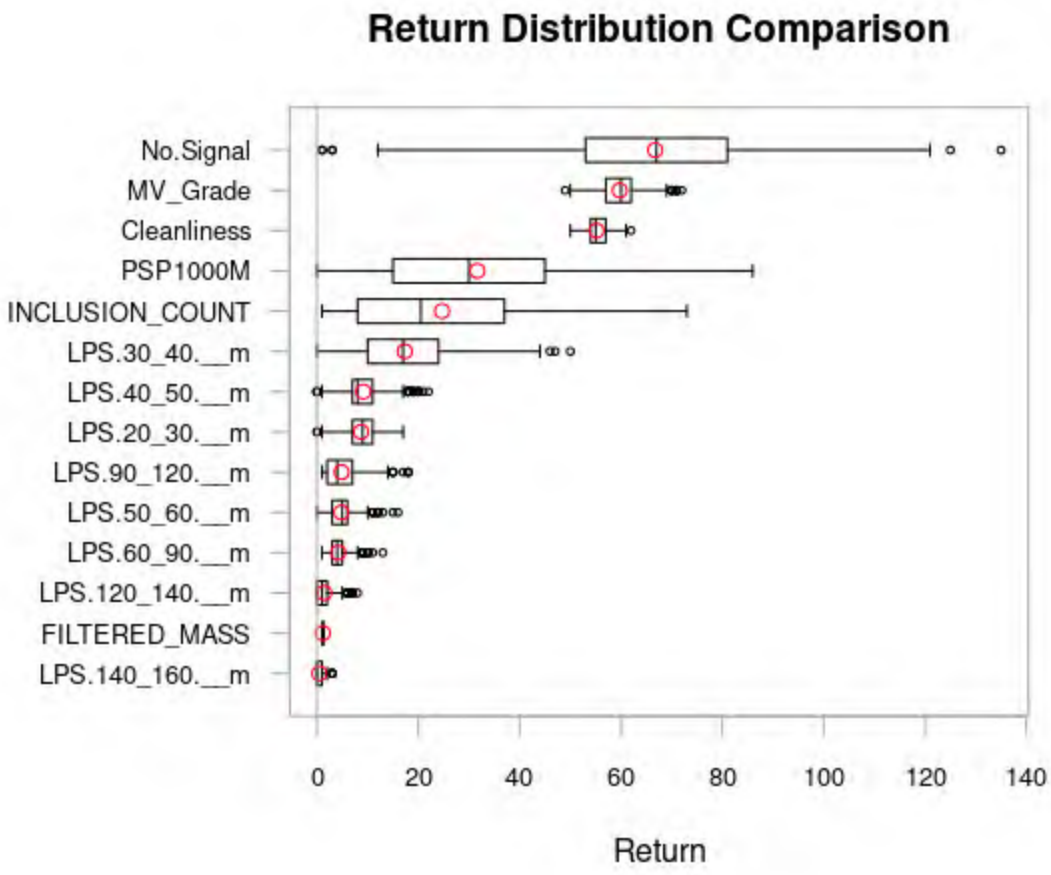

Figure 3.8: Boxplots showing the distributions of the dataset. The return axis indicates the relative ratios of the normalised variables as percentages.

The boxplots indicate that most of the variables have a normal distribution, although some appear to have skewness. There are also points outside of the whiskers for most of the LPS variables, indicating the presence of outliers.

3.3.4 Shapiro-Wilk Normality Test

The Shapiro-Wilk normality test is one of the popular methods for testing for normality on a given dataset [41]. The test makes a null hypothesis that the data is normally distributed, and the p-value provides the outcome. For a p-value > 0.05, which is a 5% error, the null hypothesis has passed and the data can be considered normally distributed. For values below 0.05, the null hypothesis fails. The test statistic, $W \in \{0, 1\}$, also indicates how well the data fits in a normal distribution. Therefore, higher values of W indicate a close fit to a normal distribution. The following table shows the normality test for all the numeric features:

Table 3.2: Shapiro-Wilk normality test on numeric features

variable	statistic	p-value
Cleanliness	0.98	1.589e-05
LPS.120_140._ _m	0.81	6.22e-21
LPS.140_160._ _m	0.68	4.104e-26
LPS.20_30._ _m	0.99	0.007
LPS.30_40._ _m	0.98	7.291e-06
LPS.40_50._ _m	0.98	1.452e-12
LPS.50_60._ _m	0.94	1.146e-10
LPS.60_90._ _m	0.92	3.461e-13
LPS.90_120._ _m	0.9	3.137e-15
MV_Grade	0.99	0.005
No.Signal	0.99	0.132
PSP1000M	0.97	1.789e-06
INCLUSION_COUNT	0.91	4.44e-14
FILTERED_MASS	0.96	8.189e-09

The normality test reveals that three out of the 14 variables have a normal distribution, namely LPS 20 30, MV Grade and No Signal. The rest of the variables failed the normality test. This, however, does not indicate that the variables cannot be processed using normal statistical methods, as the test statistic is high for most of them, indicating that a lot of the data still fits within a normal distribution. In order to ascertain whether the normal statistical methods can be applied (e.g. PCA), the results for those techniques will be assessed. It should also be noted that the nature of the casthouse from a cleanliness point of view provides a natural bias. This is because there are processes that actively try to improve the cleanliness of the metal, for example the ceramic foam filter. These processes create a bias in the dataset, causing it to be skewed.

3.3.5 Standardisation

Standardisation is a common method of ensuring the data contains some uniformity. This is particularly important when there exist different units in the input data. From the input dataset, most of the features are percentages, as they are features derived from the calibrated machine. There is also a "count" feature, which represents the number of inclusions detected. The only feature with units is the filtered mass, which is in grams. It is therefore necessary to standardise so as to ensure that the difference in units does not introduce any bias to the data. The input dataset is standardised by applying two common methods, namely scaling and centering. The data is scaled to unit variance, and the variables are centered about the mean.

3.3.6 Feature Extraction

For most of the machine learning algorithms available, the input data is best represented as a transformed dataset from the multivariate space to a space with several dimensions. The transformations aim to minimise information loss by identifying and extracting features from the input dataset and mapping them onto a latent space with fewer components than the original number of features of the input data. This also makes it efficient for post-processing.

3.3.6.1 Principal Component Analysis

Principal components analysis (PCA) is a leading technique for dimensionality reduction in multivariate data analysis problems [42]. It casts data from n dimensions to 2 or 3 dimensions which can be visualised and more easily processed. The main advantage of PCA is in the fact that it can reduce multi-dimensional data into 2 or several components while preserving most of the variance of the data. The PCA method derives its components from eigenvectors of the covariance matrix of the data matrix. In many cases, the data is scaled to unity and centered at the origin.

For a data matrix $\boldsymbol{X}$, the weight vector $\boldsymbol{w}_{(1)}$ for the first component is given as:

$$\boldsymbol{w}_{(1)} = arg\,max \left\{ \frac{\boldsymbol{w}^T \boldsymbol{X}^T \boldsymbol{X} w}{\boldsymbol{w}^T \boldsymbol{w}} \right\} \tag{3.1}$$

in order to maximise variance. To get the next components,

$$\hat{\boldsymbol{X}}_k = \boldsymbol{X} - \sum_{s=1}^{k-1} \boldsymbol{X}\boldsymbol{w}_{(s)}\boldsymbol{w}_{(s)}^T \tag{3.2}$$

and the corresponding weight vector which maximises the variance of the data matrix is:

$$\boldsymbol{w}_k = arg\,max \left\{ \frac{\boldsymbol{w}^T \hat{\boldsymbol{X}}^T \hat{\boldsymbol{X}} w}{\boldsymbol{w}^T \boldsymbol{w}} \right\} \tag{3.3}$$

In order to ensure that the number of principal components considered provides a balance between complexity (number of components) and the cumulative variance explained, a plot of the variance as a function of the number of components is shown:

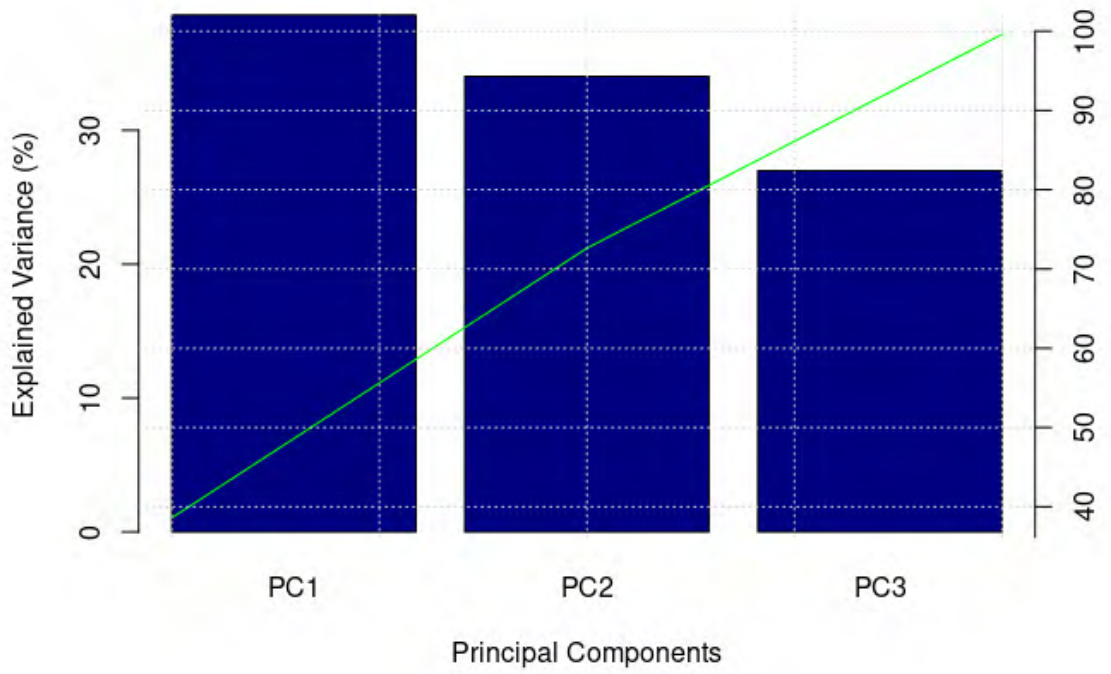

Figure 3.9: Explained variance as a function of the number of principal components

From the figure, it can be seen that 99.6% of the variance of the input dataset is explained by the first three principal components. This means that using the first three components, a sufficient variance of the dataset is explained.

Figure 3.10 shows the PCA plot of the three principal components. The points are coloured by sample result.

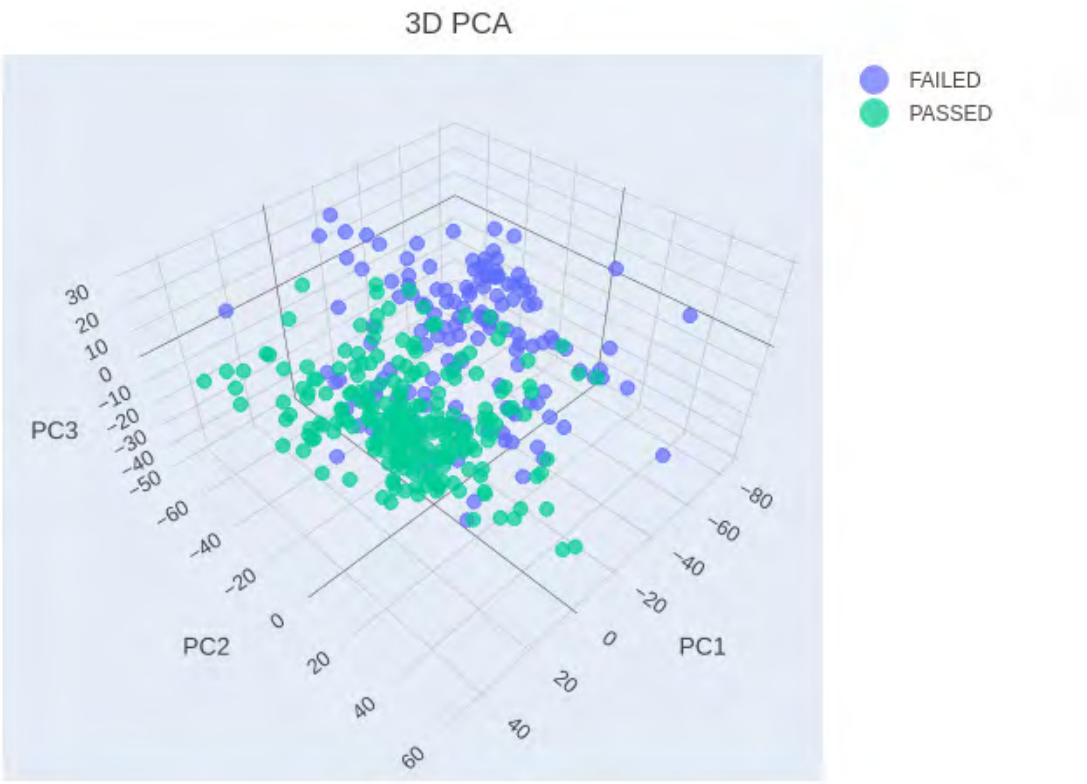

Figure 3.10: PCA plot showing the three principal components, coloured by sample result

This plot indicates that a separation exists between the two classes in the latent space, although there might be non-linear separation.

The following plot shows the principal components coloured by inclusion type:

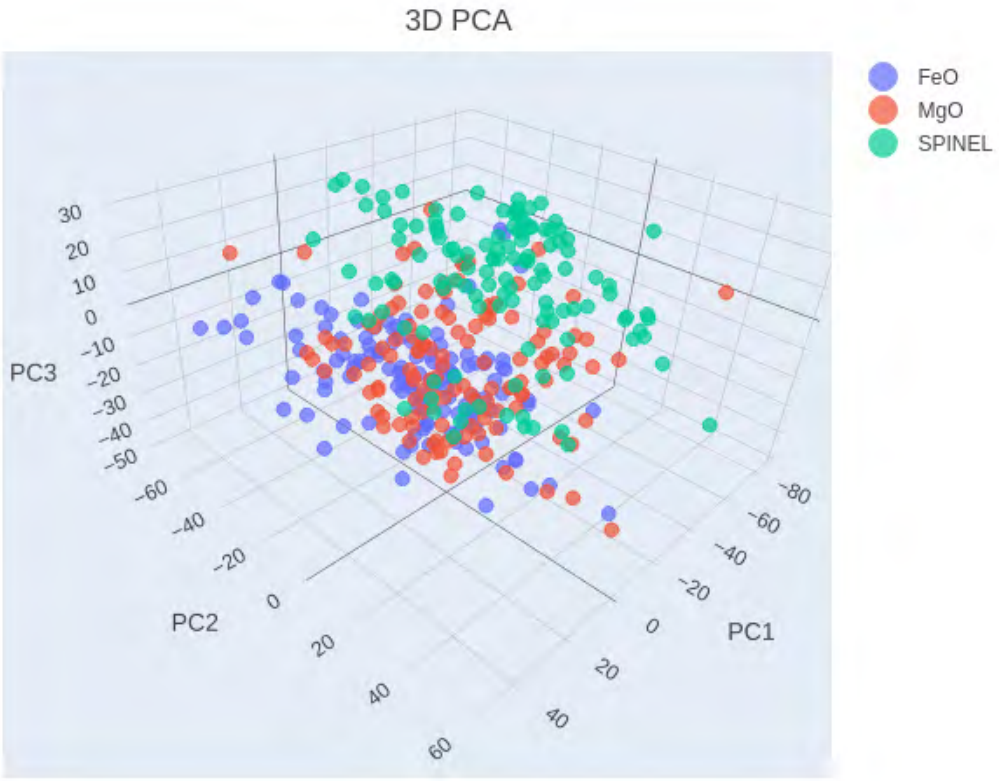

Figure 3.11: PCA plot showing the three principal components, coloured by inclusion type

The plot shows that there might be some separation between the classes in the latent space, although there appears to be overlap.

3.3.6.2 Kernel Principal Components Analysis

In order to account for non-linearities in the latent space, a kernel function can be used. The kernel function quantifies the similarity of the observations by mapping the data to a higher dimension where the classes may be more spatially separable [44]. There are three popular kernels in use, which are:

The **linear** kernel applies a linear function based on linearly separable data and is of the form:

$$K(x_i, x_{i'}) = \sum_{j=1}^{p} x_{ij} x_{i'j} \tag{3.4}$$

, where K is the kernel function of two observations $x_i, x_{i'}$. Since we are trying to account for non-linearities, this kernel will not be considered.

The **polynomial** kernel can have degrees $d > 1$ and can help separate clusters in data that is not linearly separable. The polynomial kernel function has the form:

$$K(x_i, x_{i'}) = \left(1 + \sum_{j=1}^{p} x_{ij} x_{i'j}\right)^{d} \tag{3.5}$$

This kernel offers a more flexible decision boundary than the linear kernel.

The **radial** kernel uses a radial basis function (RBF). The RBF can approximate for more non-linearities and map the data to a more linearly separable space. The RBF kernel has the form:

$$K(x_i, x_{i'}) = \exp\left(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2\right) \tag{3.6}$$

, where γ is a positive constant. The RBF kernel is a popular preprocessing method for using support vector machines. According to Scikit-Learn, a good value for γ is $\frac{1}{number\ of\ features}$ [45]. Since the dataset has 14 numerical features, a starting point for the selection of γ is $\frac{1}{14} = 0.0625$. The kernel PCA explained variance plot is shown in Figure 3.12:

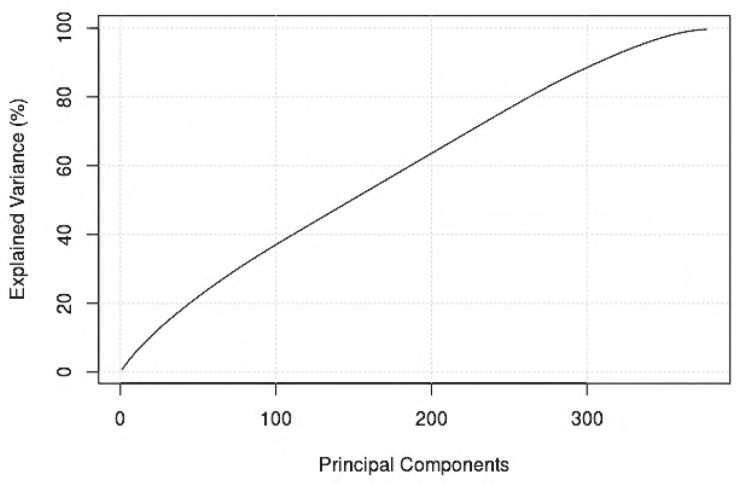

Figure 3.12: Kernel PCA explained variance plot

The kernel PCA requires too many components to explain a sufficient amount of the variance of the dataset. It is therefore deemed unfit for use within this context.

3.3.7 Summary

The data ingestion component of the study has revealed that most of the dataset does not have a normal distribution due to bias from the process. For feature extraction, the linear principal components analysis technique has proven to be an effective dimensionality reduction method. Using the linear PCA, it has been demonstrated the use of two principal components accounts for ~80% of the total variance, while using 3 principal components accounts for 99.6% of the total variance within the dataset. The data is therefore linearly separable within the PCA latent space.

3.4 Supervised Learning Classification

Supervised learning is the branch of machine learning that uses labelled datasets to develop statistical models for predicting future data. The model can be classified as a regression model or a classification model. Regression models aim to predict numerical response variables, while classification models aim to predict categorical response variables. For categorical inputs, there exist methods to perform one-hot encoding so that they are represented as numerical values for the model to ingest. They are mostly disregarded, however, in model building as they may contain biases from being user inputs. In this case, the categorical inputs are also ignored for modelling, and only the numeric features are used.

The data is split 75/25 between training and validation as this is within the widely accepted range of 70/30 to 80/20.

Classification metrics are the metrics by which the performance of trained models can be scored [39]. The assessment of success, however, is largely dependent on domain knowledge, and on specific business needs for the problem being addressed. The dataset must also satisfy the one-in-ten predictor rule, which states that you need at least one predictor for every ten observations in order to prevent overfitting [43].

It is generally a difficult task to train a machine learning model. This is because the model has hyperparameters that determine how optimally it generalises over the training data. To overcome this, the cross-validation method is used. In particular, the k-fold cross-validation method iterates through different values of the hyperparameters to determine the best combination for the specific dataset. This is achieved by holding out a validation sample each time from a different slice of the input dataset. Once validation is done, the data is shuffled and another holdout sample is drawn for validation while the model is fitted on the rest of the dataset. As the model is repeatedly resampled, the variant with the best hyperparameter performance is returned. Usually, a target metric is used to assess model performance. The value of $k = 10$ is used so as to ensure granularity in the cross-validation. The cross validation is also repeated 5 times to ensure that it is not biased.

3.4.1 Logistic Regression

The logistic regression model is an extension of the linear regression model, which is a numerical regressor, into the probability space. A logistic regression model calculates the probability that an outcome belongs to a certain category, and makes a prediction based on the highest probability [43]. It extends the linear regression model by introducing a logit function, which uses probabilities to estimate the categorical parameter from the input, instead of mapping a numerical output to the input. The logistic regression model is represented as:

$$log\left(\frac{p(X)}{1-p(X)}\right) = \beta + \alpha X \tag{3.7}$$

, where $p(X)$ is the default probability of X, β is the initial biasing factor and α is the weight applied to the input X. The logistic function ensures that for all X, the probability $p(X)$ is between 0 and 1.

The hyperparameters to optimise for the logistic regression model are shown in the following table:

Table 3.3: Logistic regression model hyperparameters

Parameter	Value
model_id	generalised Linear Model (Logistic Regression)
regularisation distribution	0 - 1
lambda regularisation	0.0625 (1 / number of observations)

The logistic regression model is part of the family of generalised linear models (GLM). For this model, the regularisation distribution is the hyperparameter to be optimised. The second parameter, lambda, is the regularisation factor and has been proven to be optimal at the inverse values of the number of features in the dataset [46].

3.4.2 Support Vector Machine

Support vector machines (SVMs) are among the most widely used and successful classifiers in use today [47]. This is because they have been known to perform satisfactorily in different applications. The support vector machine transforms the feature space into enlarged higher dimensions using kernel functions. The purpose for enlarging the feature space is to accommodate for non-linear boundaries between target classes. Considering a linear support vector classifier:

$$f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle x, x_i \rangle , \tag{3.8}$$

where the parameters $\alpha_1, \alpha_2, ...\alpha_n$ and β_0 can be determined using the inner products $\langle x_i, x_{i'} \rangle$ between all pairs of training observations. It is therefore evident that all that is needed are the inner products in order to solve for a linear support vector classifier. The support vector machine introduces a generalisation to the inner product:

$$K\left(x_i, x_{i'}\right), \tag{3.9}$$

where K is the kernel function that performs the generalisation. The kernel computes the similarity between two observations. There are three widely used kernel functions [47], namely:

- Linear: $K\left(x_i, x_{i'}\right) = \sum_{j=1}^{p} x_{ij} x_{i'j}$,

- Polynomial: $K(x_i, x_{i'}) = \left(1 + \gamma \sum_{j=1}^{p} x_{ij} x_{i'j}\right)^d$, and
- Radial basis function (RBF): $K(x_i, x_{i'}) = e^{\left(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2\right)}$.

Here, $\gamma > 0$ is a kernel weighting parameter that needs to be selected for the model (using a polynomial, RBF or sigmoid kernel function). The degree of the polynomial $d > 1$ determines the degree of flexibility of the decision boundary. The choice of a kernel function for non-linearly separable classes is not intuitive, but it is regarded as a good starting point to select the RBF kernel [47]. In order to ensure due diligence, however, the different kernel functions will be compared side-by-side to determine the optimal one for training on the dataset.

3.4.2.1 Model Parameters

The hyperparameters to optimise for the logistic regression model are shown in the following table:

Table 3.4: SVM model hyperparameters

Parameter	Value
model_id	Support Vector Machine (SVM)
cost	0 - 1
kernel	linear, polynomial, rbf
degree	1 - 5 (polynomial)
gamma	0 - 1 (polynomial, rbf)
loss function	categorical crossentropy

3.4.2.1.1 Sample Result Target Respondent The first parameter to optimise is the cost function, which is common among all the variants of the SVM model. In order to find the optimum cost coefficient, a linear variant of the activation function is used, and the cost function is incremented. The following plot shows the model training performance as a function of the cost function:

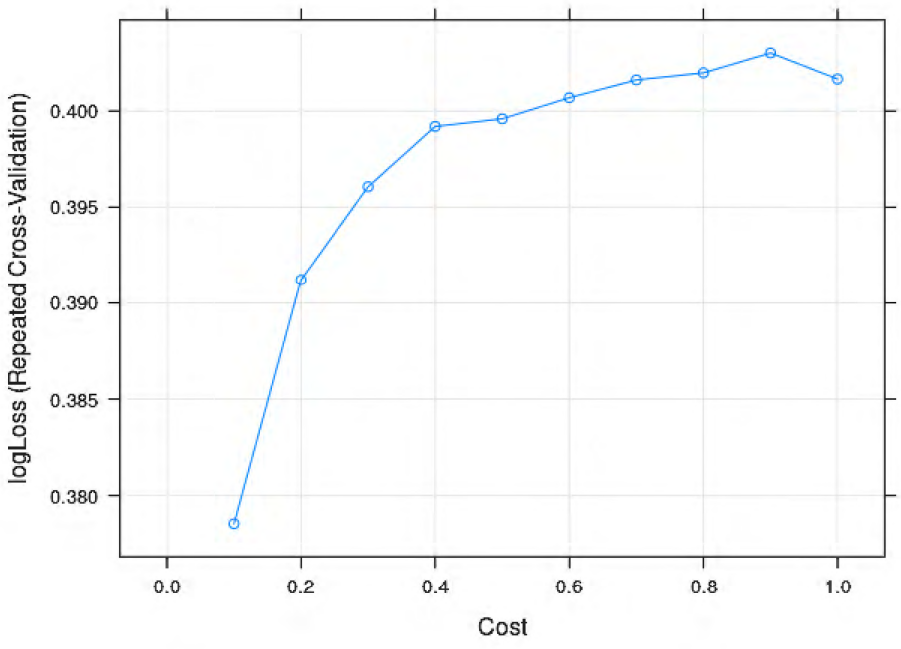

Figure 3.13: Sample result target respondent SVM cost function performance

The optimal cost function is determined to be 0.1 as the loss of the model is minimal at that value. This value is therefore used for all the variants of the SVM.

The polynomial degree is also applied as an independent variable, and the training performance is plotted as its function. The training performance is shown in Figure 3.14:

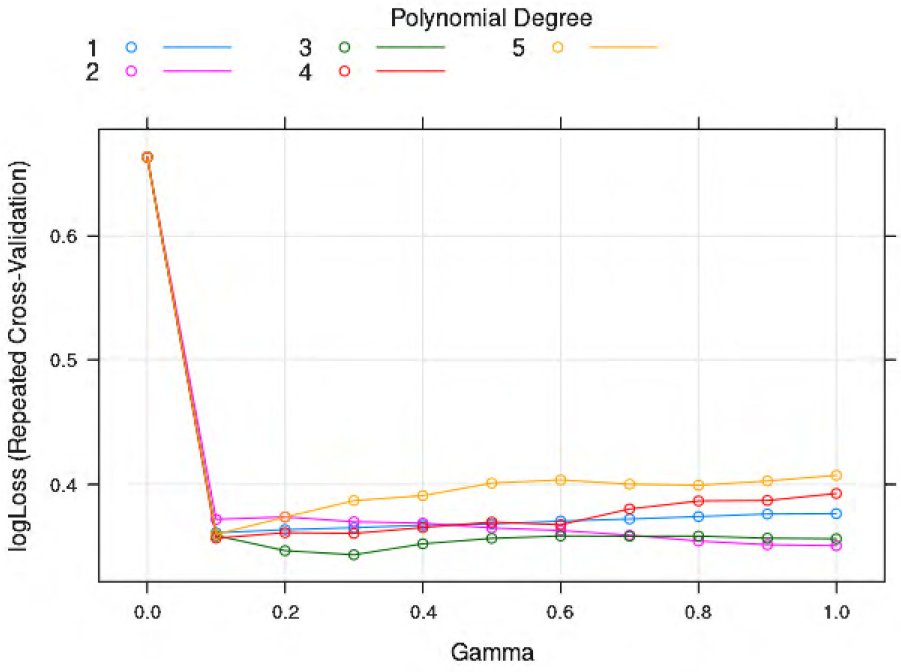

Figure 3.14: Sample result target respondent SVM polynomial degree performance

The log-loss curves show that the degree of 3 is the optimal degree for the polynomial variant of the SVM model. This is because it has the lowest loss at a corresponding gamma value of 0.3. These are therefore the selected hyperparameters for the polynomial variant.

The following plot shows the training performance of the model for different values of the gamma function:

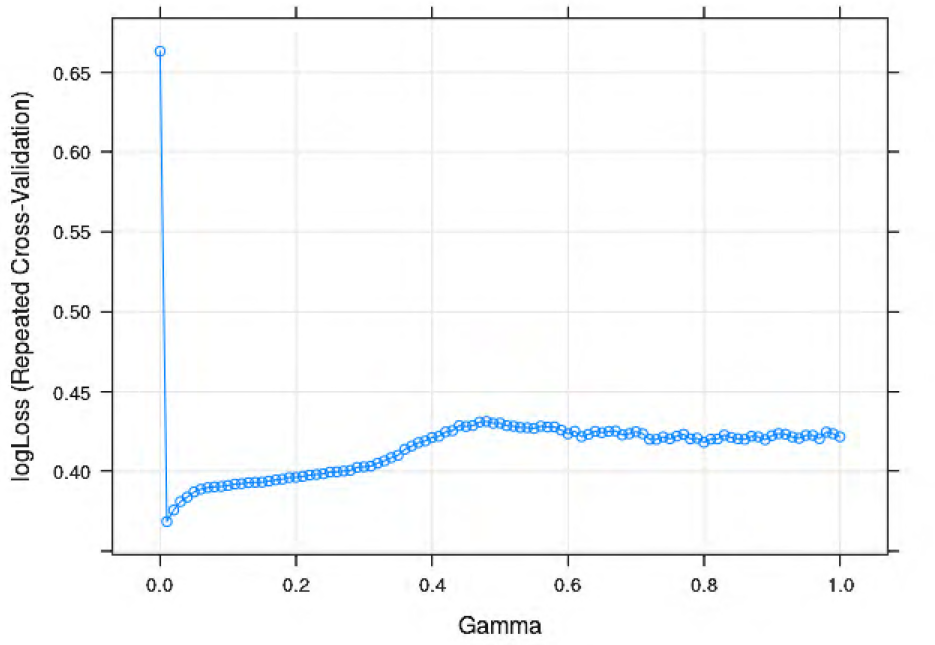

Figure 3.15: Sample results target respondent SVM RBF gamma performance

The best performance for gamma is at 0.01, where the lowest log-loss is achieved. This is therefore used to train the final model.

In order to compare the kernel functions, the optimal hyperparameters are set for each kernel function respectively, and the training performances of the kernel functions are compared. The training performances are shown in Figure 3.16:

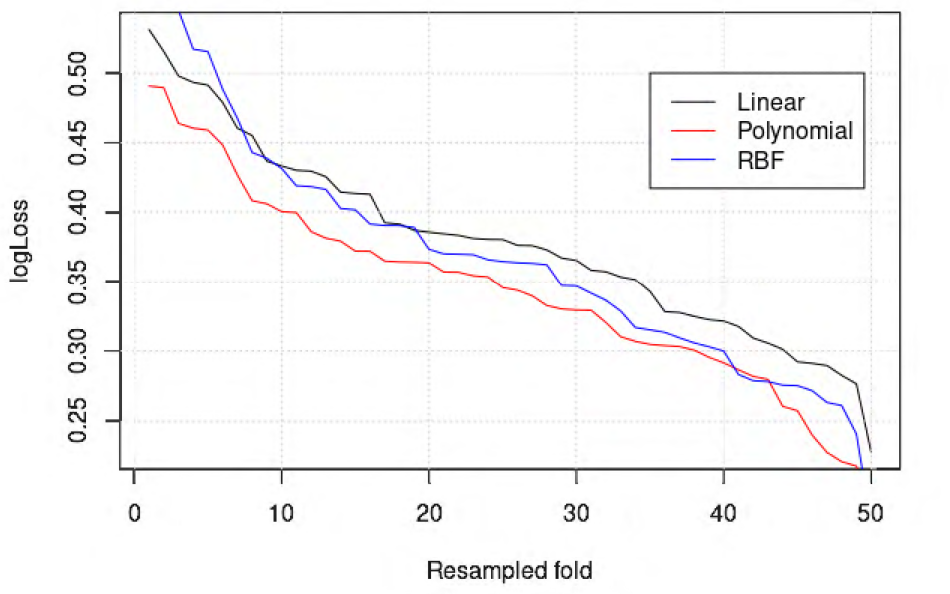

Figure 3.16: Sample result target respondent SVM kernel function comparison

The loss functions for the different kernels show little difference in performance. The polynomial kernel appears to provide

the best loss, followed by the RBF kernel. The differences are negligent, which indicates training convergence. This therefore means that the polynomial and RBF kernel functions can be used with negligible difference in performance. The RBF kernel, however, is more computationally expensive, and therefore the polynomial kernel is used in the final model.

3.4.2.1.2 Inclusion Type Target Respondent The following plot shows the model training performance as a function of the cost function:

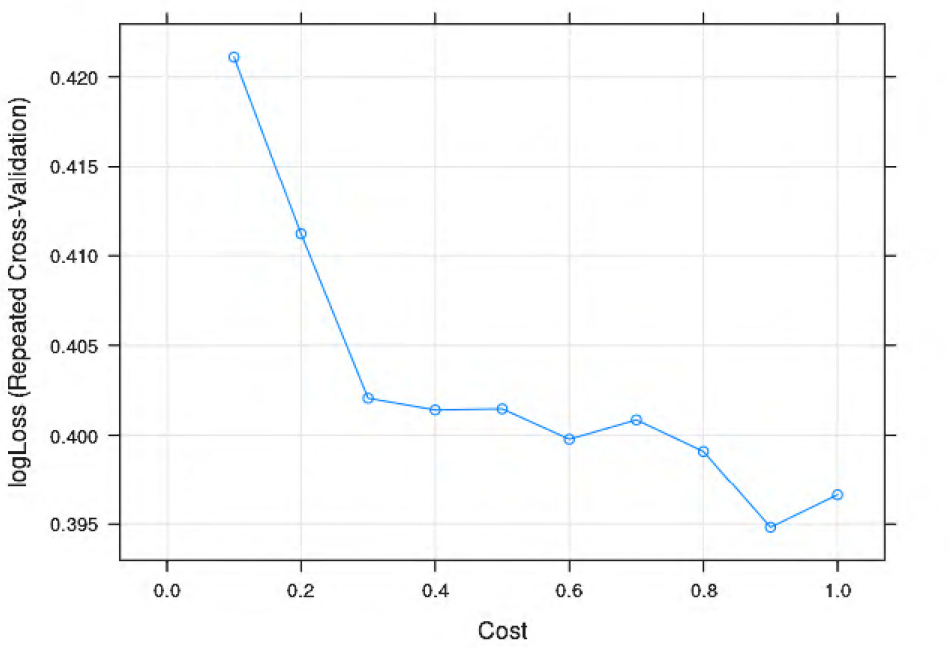

Figure 3.17: Sample result target respondent SVM cost function performance

The log-loss function sharply decreases down to a minimum of 0.395, where the cost function is 0.9. This is therefore the value used for training the SVM.

The polynomial degree is also applied as an independent variable, and the training performance is plotted as its function. The training performance is shown in Figure 3.18:

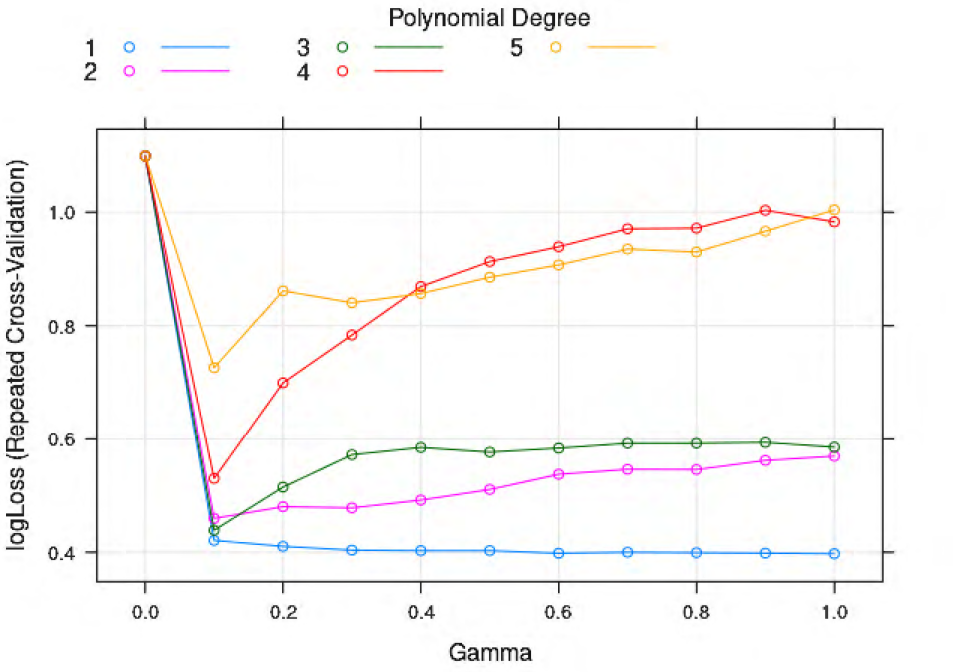

Figure 3.18: Sample result target respondent SVM polynomial degree performance

The curves show that for higher degrees of the polynomial, the loss increases after a sharp drop at $\gamma = 0.1$. The first degree is the only order to maintain a decrease in the loss function for increasing values of gamma. The lowest loss is achieved at a value of $\gamma = 1$, where the loss is 0.4.

Figure 3.19 shows the training performance of the model for different values of the gamma function:

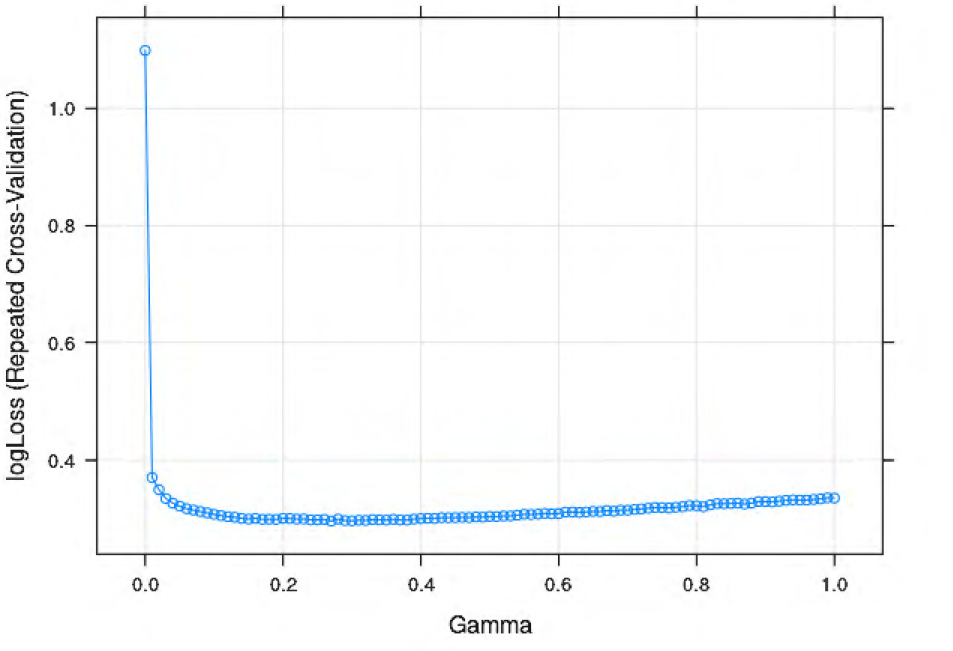

Figure 3.19: Sample results target respondent SVM RBF gamma performance

From the curve, it can be seen that the loss function takes a sharp drop before slowly increasing. the optimal value of

gamma is therefore where the loss makes a turning point, which is 0.27.

In order to compare the kernel functions, the optimal hyperparameters are set for each kernel function respectively, and the training performances of the kernel functions are compared. The training performances are shown in Figure 3.20:

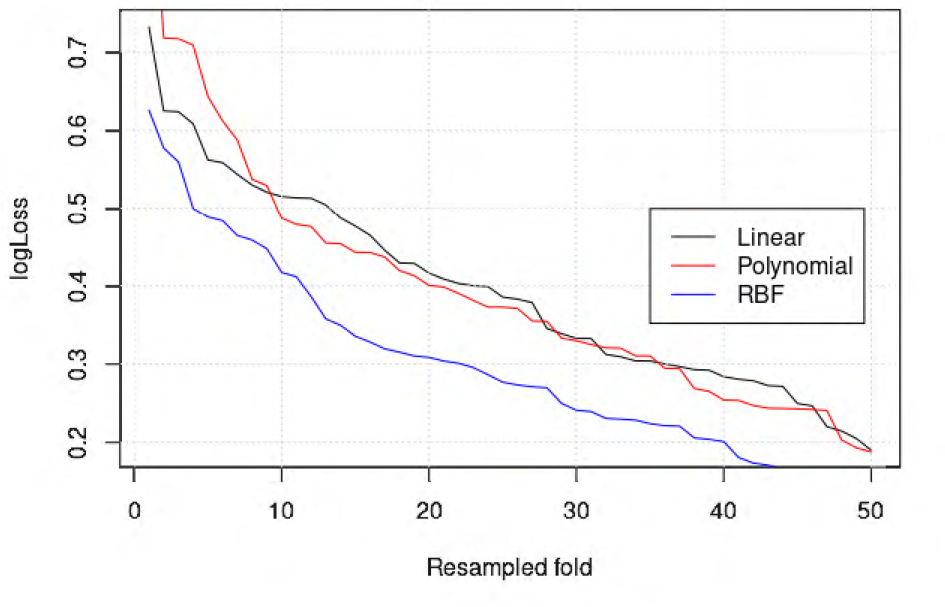

Figure 3.20: Sample result target respondent SVM kernel function comparison

The RBF has proven to be the optimal kernel function for fitting the data, as it offers the best overall performance in relation to the loss function. The linear and polynomial functions have comparable performance. The RBF is therefore the preferred kernel for building the final model.

3.4.3 Multi-Layer Perceptron

The use of artificial neural networks as brought the ability to approximate many problems that could not be well approximated using traditional methods [31]. To date, numerous networks have been developed that have been targeted at solving traditionally difficult problems, including machine vision, natural language processing, sentiment analysis and various regression and classification problems across many fields. They have also been extensively used in medical applications. They are named after the biological neural network, as they also contain neurons and synapses that interconnect them [48].

The multi-layer perceptron (MLP) is the most basic form of a neural network. An illustration of an MLP is shown in Figure 3.21:

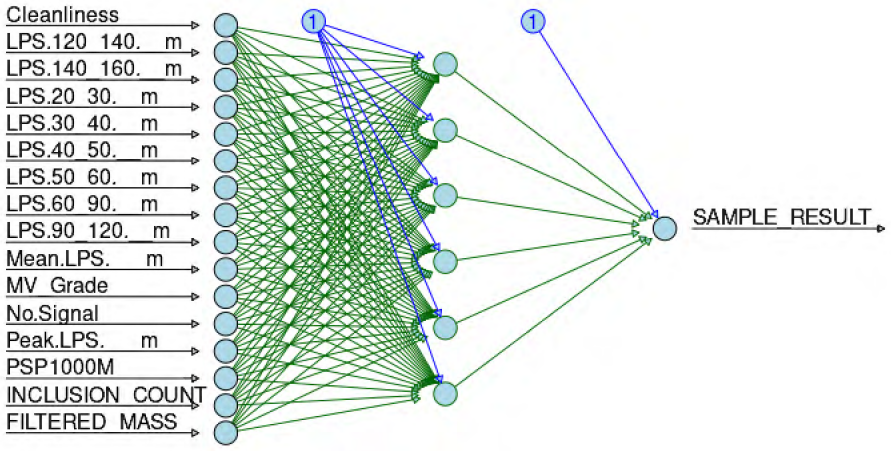

Figure 3.21: Neural network architecture

The network, as shown in the figure, consists of connected layers of neurons, including the input, hidden and output layers. The components are discussed in the following subsection.

3.4.3.1 Neurons

The neurons, which are also called nodes, are the main elements that are interconnected to form the neural network. Each neuron is represented by an activation function, which is activated when the input exceeds a given threshold [48]. The threshold and activation function form the core of the neuron. A neuron can have many inputs, but it has only one output. The activation function of a neuron is shown below:

$$f(x, w, b) = \begin{cases} \boldsymbol{w} \times \boldsymbol{x} + \boldsymbol{b} & \boldsymbol{w} \times \boldsymbol{x} + \boldsymbol{b} \geq \theta \\ 0 & otherwise \end{cases} \tag{3.10}$$

, where

$f(x, w, b)$ is the activation function,

$\boldsymbol{w}$ is the weight vector,

$\boldsymbol{x}$ is the input signal,

$\boldsymbol{b}$ is the bias, and

$\boldsymbol{\theta}$ is the activation threshold.

The neuron uses the threshold to filter out weighted signals, and the activation function transforms the signals that have exceeded the threshold according to the selected activation function. It is important to find an activation function that will not suppress the input (diminishing gradient) or boost noise (exploding gradient). As a result, there exists a body of knowledge which prescribes the best practices for selecting an activation function [9, 31, 48]. Some of the most widely used activation functions are presented in the following subsection.

3.4.3.2 Activation Functions

The ReLU (rectified linear unit) is the most popular activation function in neural networks. This is because the function is linear above the origin point and zero elsewhere:

$$f(x) = \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases} \tag{3.11}$$

The ReLU function is the preferred starting point as it retains x for all positive values of x. This gives a safe performance regarding diminishing gradients and exploding gradients as it is non-saturating and it offers an accelerated gradient descent towards a minimum value of the loss function [49]. The drawback with the ReLU function, however, is that for negative inputs the model will be poorly trained as they are turned to zero. This may result in accuracy implications on models.

There exist several versions of the ReLU function to overcome this. A popular variant is the leaky ReLU, where the function is given a small gradient in the negative space. This makes it better for negative inputs, but the predictions may not be consistent for negative inputs due to the different gradients. Another popular variant is the exponential linear units (ELU), where the negative inputs are mapped onto an exponential function. This gives better sensitivity to negative inputs, but introduces the challenge of exploding gradients.

The maxout activation function is a generalisation of the ReLU and leaky ReLU activation functions [50]. It selects the maximum feature from the inputs. The maxout function is given as:

$$f(x) = max(w_1^T x + b_1, w_2^T x + b_2, ..., w_n^T x + b_n) \tag{3.12}$$

for n inputs. The main advantage of maxout functions is that with at least two maxout units, they can approximate any function. They have also been proven to perform well for most applications.

The linear activation function maps the output to the input for all values of the input:

$$f(x) = x \tag{3.13}$$

While for positive values of x the linear function shares the advantages of the ReLU function, its major drawback is that it does not support backpropagation. This is because the derivative of the function is a constant value (1) which has no relationship to the input. In addition, the linear activation function decomposes multiple hidden layers to the equivalent of one, as the last hidden layer is a linear function of the input. This defeats the purpose of having hidden layers.

The sigmoid function is an inverse of the exponential decay function:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{3.14}$$

The sigmoid function casts any input to a value between 0 and 1. This makes it ideal for cases where inputs might be unevenly weighted, as the input contributions will not differ by much. This also means that the sigmoid can be used to predict probabilities, as probabilities only exist between 0 and 1. The function is also differentiable, but the differential function ceases to learn as the gradient approaches 0. Also, the function outputs positive values for all x, meaning that the learning f the model might be unstable [49].

The hyperbolic tangent (tanh) activation function has an S-shaped curve which bears similarity to the sigmoid function in the first quadrant. The tanh function, however, converges towards -1 for increasingly negative values of x. The function is given by:

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{3.15}$$

The tanh function gives better performance than the sigmoid function for multi-layered neural networks [51]. Also, the function is zero-centered, so the output values can be categorised as strongly negative, neutral, or strongly positive. The tanh activation function also has the diminishing gradient drawback as the derivative converges towards 0 in either direction. It is better than the sigmoid in that the gradients are not confined to a particular direction. Hence its non-linearity is preferable to that of the sigmoid function, especially for deeper networks.

The softmax activation function is a combination of multiple sigmoid functions. It is most commonly used as an activation function for the output layer of the neural network in multi-class problems [49]. It returns multiple probabilities for each class, and the class with the highest probability is the predicted class. The softmax function is given as:

$$f(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}} \tag{3.16}$$

The softmax function is ideal for multi-class classification networks, and is among the most widely used in these applications, particularly in the output layer. As a result, the output layer for deep classifiers is often called the softmax layer. The softmax function is used in this project for all output layers [48].

3.4.3.3 Layers

The **connections** between neurons are simply weights. These weights act on the signal to either amplify or compress it. The weights together with the bias are part of the learned features of the model to ensure that signals are weighted in terms of relative importance, and noise is suppressed. The weights are determined iteratively by the model during the learning stage.

The **input layer** is a column vector of nodes. These nodes represent the interface to the features of the input data and the rest of the model. For a multivariate dataset, the input layer will contain n neurons, which are the total number of features to be learned by the model. Each model has only one input layer.

The **hidden layers** of the neural network form the intermediate layers of the network where the learning takes place. The depth of a neural network is based on the number of hidden layers. Each layer is a vector of unconnected neurons, with each neuron being connected to all neurons from the preceding layer and all neurons of the subsequent layer. The neurons from different layers are connected by weights and biases. Consequently, most of the computational intensity comes from the hidden layers. The last hidden layer is connected to the output, and the first hidden layer is connected to the input.

The **output layer** consists of a vector of unconnected neurons that represent the output. Classification neural networks have the same number of neurons on the output layer as the number of target classes. As most neural networks use probabilities, the output layer usually contains probability outputs, with the target output having the highest probability.

The loss function characterises the error of the model. It provides feedback to the model by updating the gradients used to adjust the weights. The model therefore aims to minimise the loss function as it learns [31]. Some of the most widely loss functions include:

1. Mean Squared Error. This function is used in most regression problems.
2. Binary Crossentropy. This function is used in most binary classification problems.
3. Categorical Crossentropy. This function is used in most multi-class classification problems.

For this problem, categorical crossentropy is used. Based on the components listed above, the hyperparameters of the multi-layer perceptron are given in the following table:

Table 3.5: Multi-layer perceptron model hyperparameters

Parameter	Value
model_id	multi-layer perceptron
number of hidden layers	1 - 2
number of neurons	5 - 10; 1 - 10
loss function	categorical crossentropy
activation function (hidden layer)	relu, linear, sigmoid, tanh
activation function (output layer)	softmax

3.4.4 Radial Basis Function Network

Radial Basis Function (RBF) networks are a special case of the multi-layered perceptron, with all the activation functions of the hidden layers being radial basis functions [52]. Typically, an RBF network consists of an input layer, a hidden layer with radial basis activation functions, and a linear combination of the radial basis functions into the output layer. The network is a fully connected feed-forward network given as:

$$y_i(\boldsymbol{x}) = \sum_{k=1}^{J_m} w_{ki}\phi\left(\left|\left|\boldsymbol{x} - \boldsymbol{c}_k\right|\right|\right),\ i = 1, ..., J_{m'} \tag{3.17}$$

, where

$y_i(\boldsymbol{x})$ is the i^{th} output,

w_{ki} is the connection weight from the k^{th} hidden unit to the i^{th} output unit,

ϕ is the RBF,

$\boldsymbol{x}$ is the input,

$\boldsymbol{c}_k$ is the center of the k_{th}node, and

$||.||$ is the Euclidean norm.

It is common occurrence to have the same radial basis function across all the nodes in the hidden layer of the network, so that the same non-linearity is applied to all the inputs [52].

An advantage of the RBF network is that is possesses interpolation capabilities that give it the ability to universally approximate functions [53, 54]. It has also been found that the Gaussian RBF can universally approximate any function, given that it has a sufficient number of centers (nodes) [52, 55]. The Gaussian representation of the RBF is given as:

$$\phi(r) = e^{-r^2/2\sigma^2} \tag{3.18}$$

, where

$r > 0$ is the distance from a data point $\boldsymbol{x}$ to a center $\boldsymbol{c}$, and

σ is the standard deviation and is used as a smoothness factor for the interpolation function.

As can be seen from the equation, the RBF is monotonically non-increasing. This implies that for $r \to \infty$, $\phi(r) \to 0$, i.e. for greater distances to the center, the RBF contribution decays towards zero.

From [52, 53, 54], it has been shown that the application of a special form of the RBF, namely the Dynamic Decay Adjustment (RBF-DDA) algorithm is faster, constructive and has better generalisation capabilities for classification tasks. The learning process involves:

1. The algorithm is designed for the automatic selection of the parameters r and σ.
   (a) The distance r is computed as the mean of all the distances used during the training stage where the model performed correct classifications. This ensures that variable importance is also factored into the model. It should also be noted that this makes the model specific to an application, as the change in input statistics (mean, standard deviation) will render the model inefficient.
   (b) The smoothness factor σ is used for interpolation so as to create a continuous probability space.
2. Once the model has determined r and σ, thresholds need to be selected in order to optimise the learning process.

The DDA algorithm introduces two thresholds, namely θ^- and θ^+. These thresholds, when applied in the probability space, define an area of conflict, where no other prototype of a conflicting class is allowed to exist [53]. Once a correct classification has been made, the algorithm does not allow any other prototype of the model to be built (whether correct or incorrect) within the boundaries set by the threshold. This gives the DDA algorithm a few advantageous properties:

- Constructive. The number of neurons needed by the model is determined dynamically during the training stage. This ensures that the model will have an optimal number of nodes in the hidden layer for the specific problem.
- Consequently, the model is faster than a multi-layer perceptron. This is due to its constructive nature.
- The model is guaranteed to converge.
- The model is simplified to a problem of selecting the threshold parameters θ^- and θ^+.
- For $\theta^+ > \theta^-$, all wrong classifications lie below θ^- and all correct classifications lie above θ^+ [56].

A simplifying assumption of symmetry about the Gaussian center is made for the model, so that $\theta^- = -\theta^+$ [57]. This hyperparameter is therefore optimised by increments from 0 to 1 of 0.01, and the training model loss functions are presented in the following subsection.

3.4.4.1 Sample Result Target Respondent

The negative threshold tuning by means of repeated cross-validation is shown in Figure 3.22:

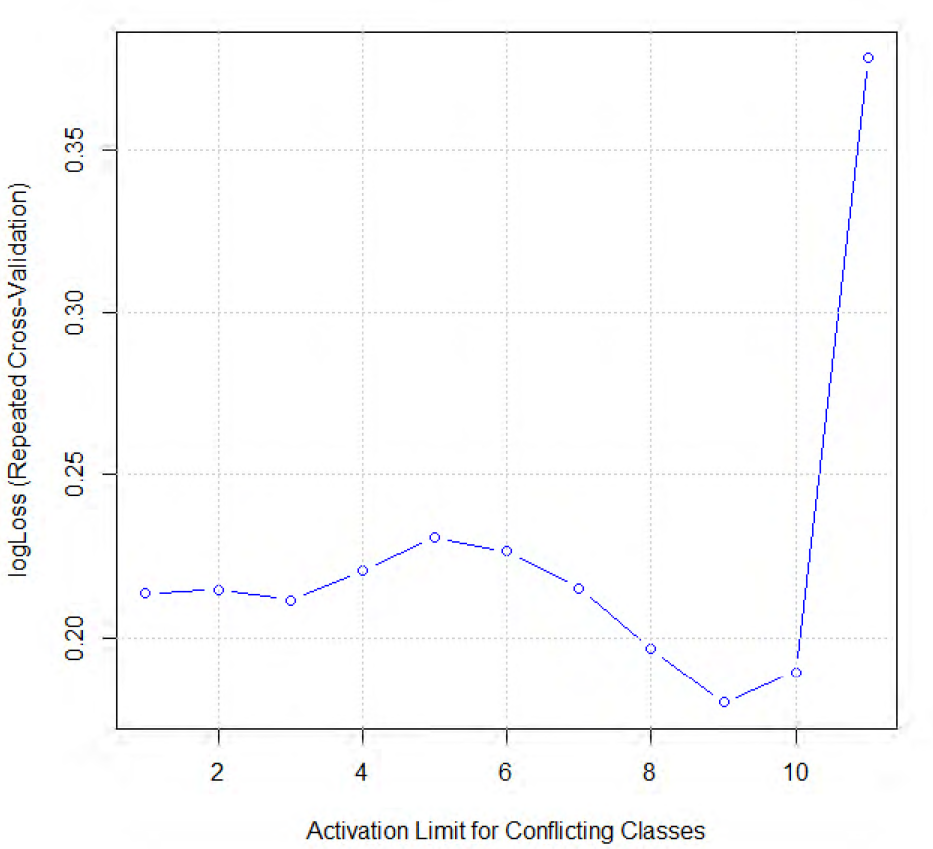

Figure 3.22: Sample result target respondent RBF training performance

The loss curve shows a dip at 0.8 and a sharp incline. The optimal threshold is therefore 0.8.

3.4.4.2 Inclusion Type Target Respondent

The negative threshold tuning by means of repeated cross-validation is shown in Figure 3.23:

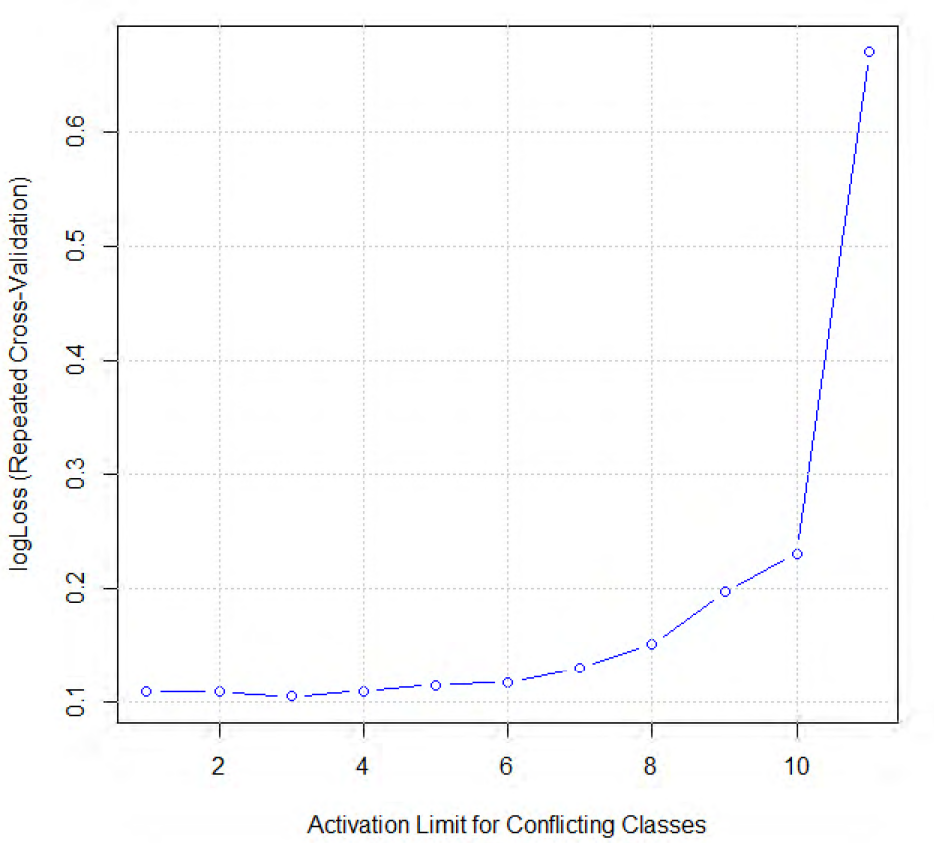

Figure 3.23: Inclusion type target respondent RBF training performance

The log-loss function has its minimum at a threshold of 0.2, before it steadily increases. The optimal threshold used is therefore 0.2.

3.5 Integration and Commissioning Considerations

3.5.1 Interface Control

The business has expressed interest in a system that can run autonomously, and require minimal input from users. This significantly simplifies the interactions that the system needs.

The input to the system is obtained from the MV20/20 system, which is specified in the following table:

Table 3.6: MV20/20 interface specification. The system also contains a configuration module that allows for the update rate, transport and port numbers to be configured.

System	MV20/20
Protocol	OPC-UA
Transport	TCP/IP
Interface	localhost:4840
Tags	13
Update Rate	1 second
Format	ASCII

By implementing a simple OPC-UA client, the data can be extracted from the source into a local dataframe.

3.5.2 System Requirements

The deployment of the system requires meeting of hardware dependencies in order to ensure the system runs smoothly. All development work has been performed on the following system specifications:

Table 3.7: System specifications for PC used in the development work for this research

Software Specifications	
Operating System	Ubuntu 20.04.3 LTS
Operating System Type	64-bit
GNOME Version	3.36.8
Windowing System	X11
Hardware Specifications	
Memory	11.7GiB
Processor	Intel® Xeon(R) CPU X5650 @ 2.67GHz × 12
Graphics	Quadro 2000D/PCIe/SSE2

These system specifications do not indicate minimum requirements, but rather demonstrate a specific use case. For most applications, the prescribed minimum specifications include a 64-bit architecture, 8GB RAM and a quad-core processor.

3.6 Experimentation

3.6.1 Logistic Regression

3.6.1.1 Model Training Loss

The optimal value for α is determined using cross-validation. The values of α are incremented from 0 to 1 by steps of 0.01 to ensure granularity.

3.6.1.1.1 Sample Result Target Respondent The repeated cross-validation loss curve for the model is given in Figure 3.24:

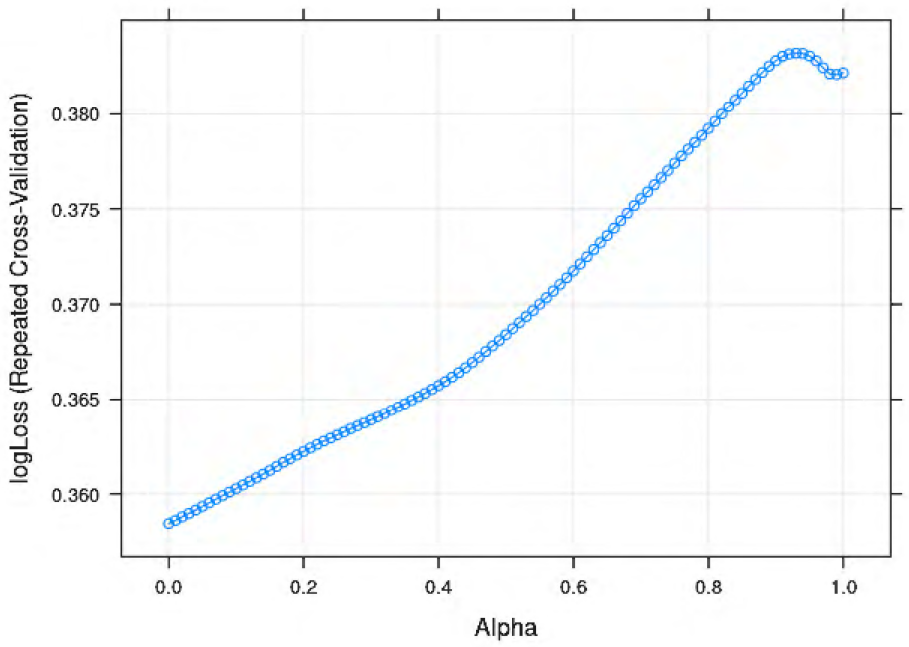

Figure 3.24: Sample Result target respondent cross-validation model training for different values of α

The curve shows a steady increase in log-loss as alpha increases, peaking around $\alpha = 0.9$. The optimal value of alpha is therefore 0, where the training loss is at its lowest.

3.6.1.1.2 Inclusion Type Target Respondent The repeated cross-validation loss curve for the model is given in Figure 3.25:

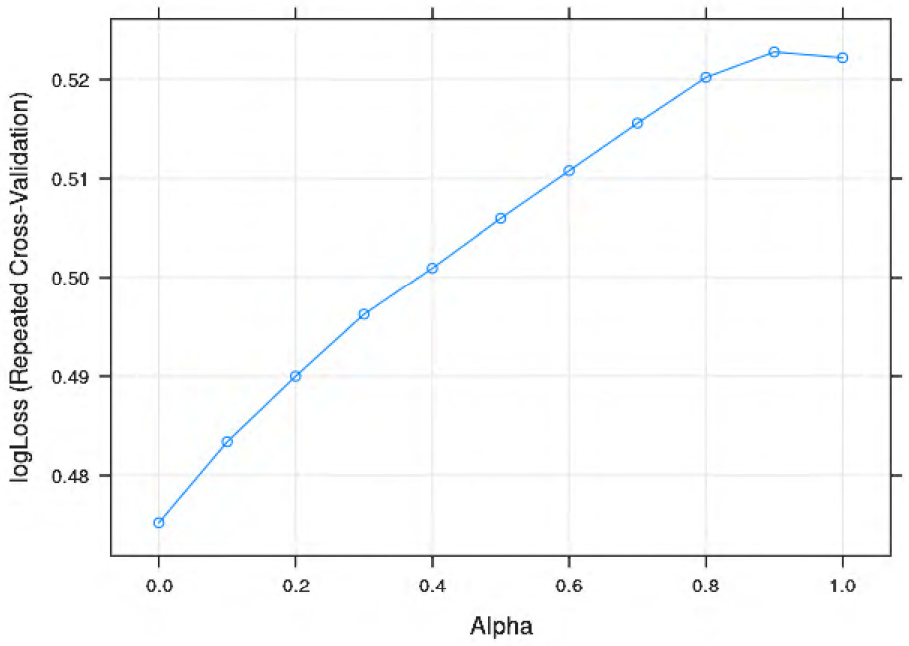

Figure 3.25: Inclusion type target respondent accuracy model training for different values of α

The curve shows that the training loss is at its minimum when $\alpha = 0$. This is therefore the optimal hyperparameter used to build the final model.

3.6.2 Support Vector Machine

3.6.2.1 Sample Result Target Respondent

3.6.2.1.1 Linear Cost Function The first parameter to optimise is the cost function, which is common among all the variants of the SVM model. In order to find the optimum cost coefficient, a linear variant of the activation function is used, and the cost function is incremented. The following plot shows the model training performance as a function of the cost function:

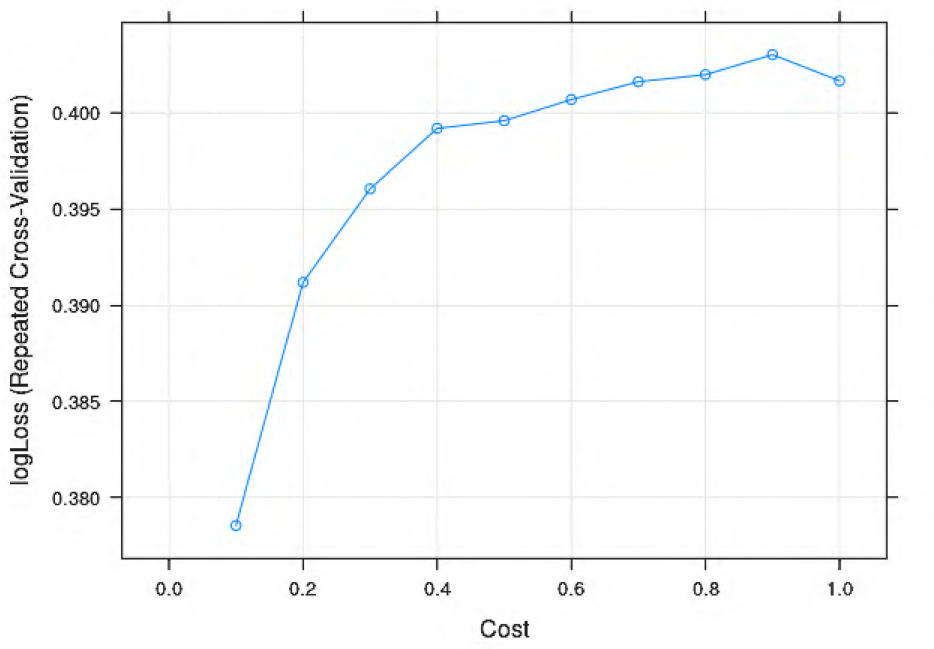

Figure 3.26: Sample result target respondent SVM cost function performance

The optimal cost function is determined to be 0.1 as the loss of the model is minimal at that value. This value is therefore used for all the variants of the SVM.

3.6.2.1.2 Polynomial Degree The polynomial degree is also applied as an independent variable, and the training performance is plotted as its function. The training performance is shown in Figure 3.27:

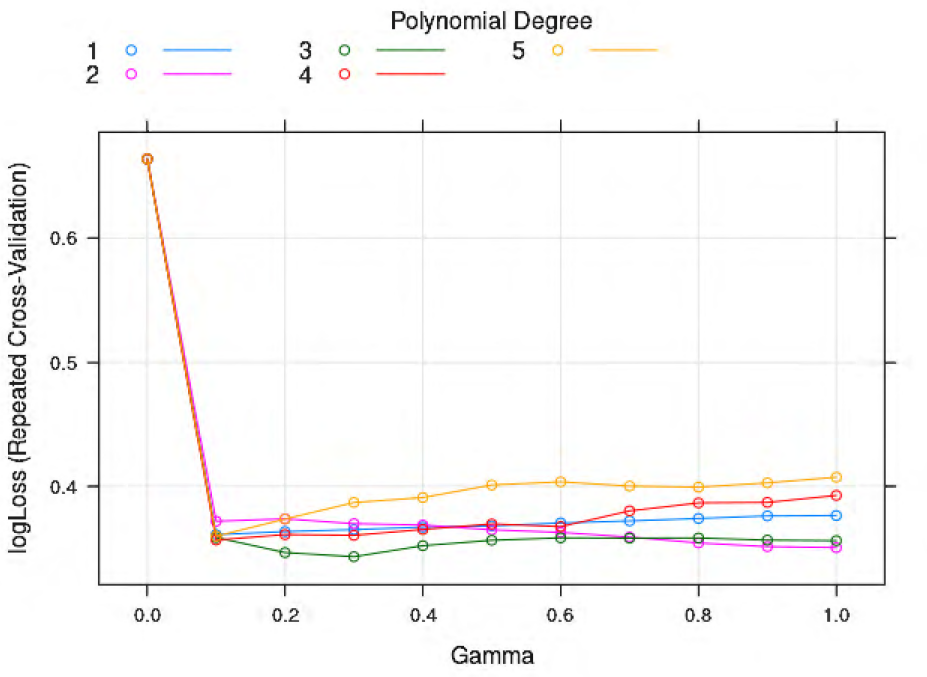

Figure 3.27: Sample result target respondent SVM polynomial degree performance

The log-loss curves show that the degree of 3 is the optimal degree for the polynomial variant of the SVM model. This is because it has the lowest loss at a corresponding gamma value of 0.3. These are therefore the selected hyperparameters for the polynomial variant.

3.6.2.1.3 RBF Gamma The following plot shows the training performance of the model for different values of the gamma function:

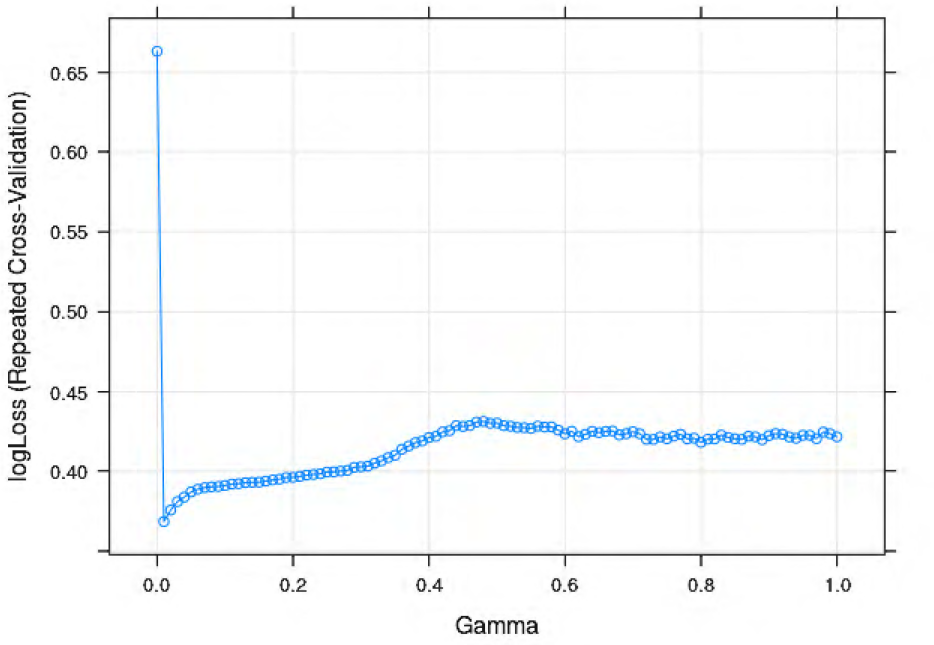

Figure 3.28: Sample results target respondent SVM RBF gamma performance

The best performance for gamma is at 0.01, where the lowest log-loss is achieved. This is therefore used to train the final model.

3.6.2.1.4 Kernel Function In order to compare the kernel functions, the optimal hyperparameters are set for each kernel function respectively, and the training performances of the kernel functions are compared. The training performances are shown in Figure 3.29:

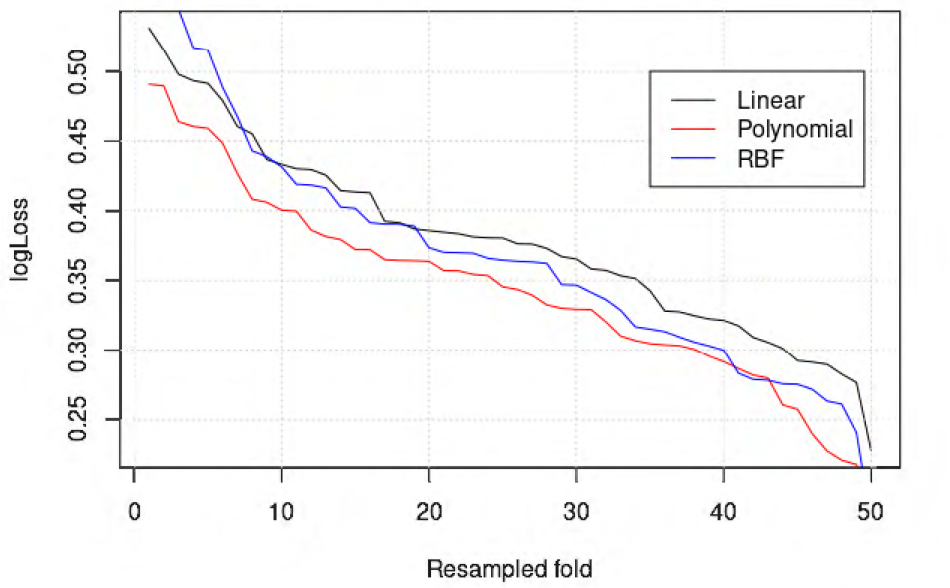

Figure 3.29: Sample result target respondent SVM kernel function comparison

The loss functions for the different kernels show little difference in performance. The polynomial kernel appears to provide the best loss, followed by the RBF kernel. The differences are negligent, which indicates training convergence. This implies that the polynomial and RBF kernel functions can be used with negligible difference in performance. The RBF kernel, however, is more computationally expensive, and therefore the polynomial kernel is used in the final model.

3.6.2.2 Inclusion Type Target Respondent

3.6.2.2.1 Linear Cost Function The following plot shows the model training performance as a function of the cost function:

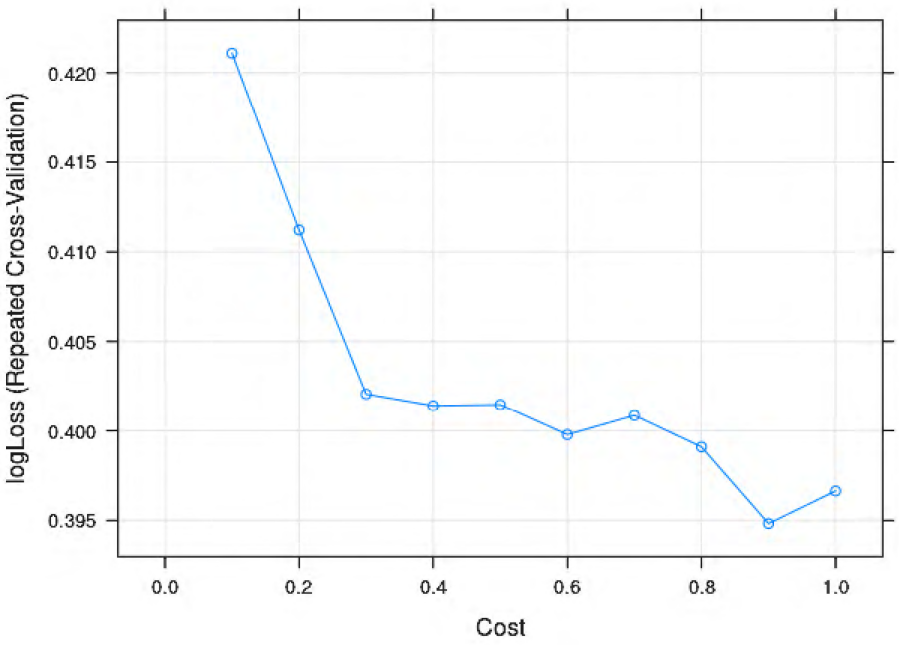

Figure 3.30: Sample result target respondent SVM cost function performance

The log-loss function sharply decreases down to a minimum of 0.395, where the cost function is 0.9. This is therefore the value used for training the SVM.

3.6.2.2.2 Polynomial Degree The polynomial degree is also applied as an independent variable, and the training performance is plotted as its function. The training performance is shown in Figure 3.31:

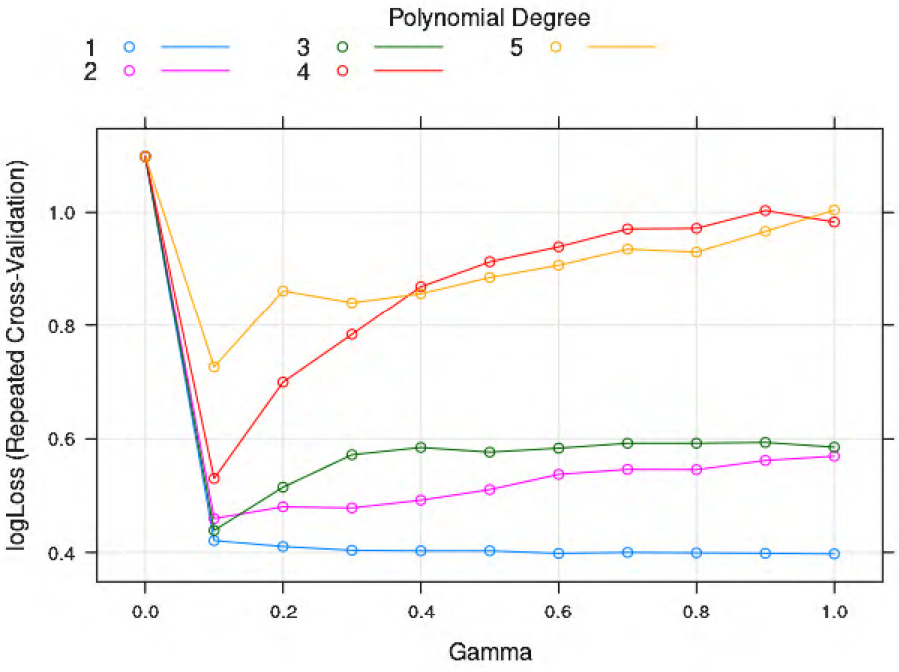

Figure 3.31: Sample result target respondent SVM polynomial degree performance

The curves show that for higher degrees of the polynomial, the loss increases after a sharp drop at $\gamma = 0.1$. The first degree is the only order to maintain a decrease in the loss function for increasing values of gamma. The lowest loss is achieved at a value of $\gamma = 1$, where the loss is 0.4.

3.6.2.2.3 RBF Gamma The following plot shows the training performance of the model for different values of the gamma function:

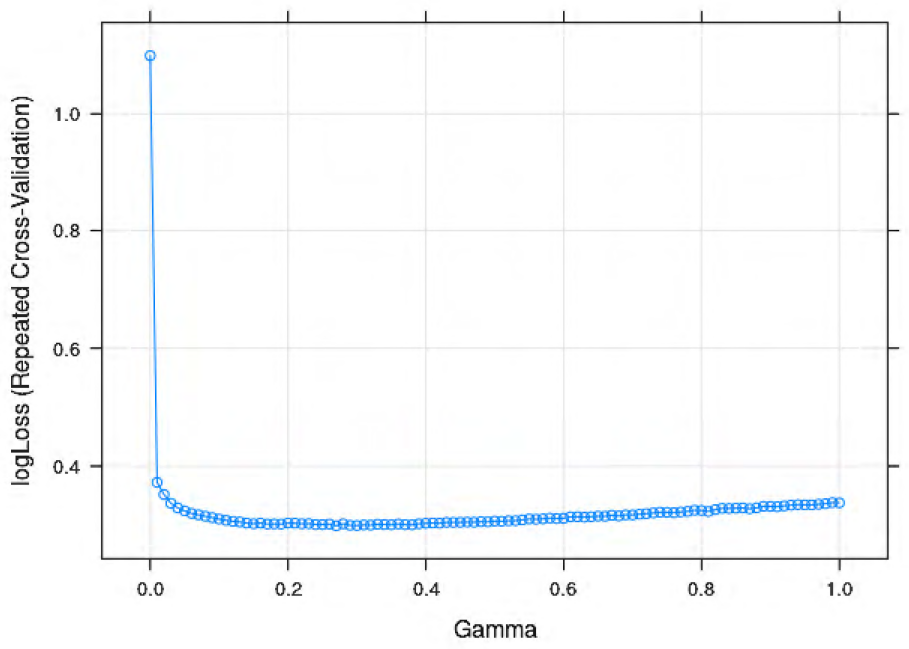

Figure 3.32: Sample results target respondent SVM RBF gamma performance

From the curve, it can be seen that the loss function takes a sharp drop before slowly increasing. The optimal value of gamma is therefore where the loss makes a turning point, which is 0.27.

3.6.2.2.4 Kernel Function In order to compare the kernel functions, the optimal hyperparameters are set for each kernel function respectively, and the training performances of the kernel functions are compared. The training performances are shown in Figure 3.33:

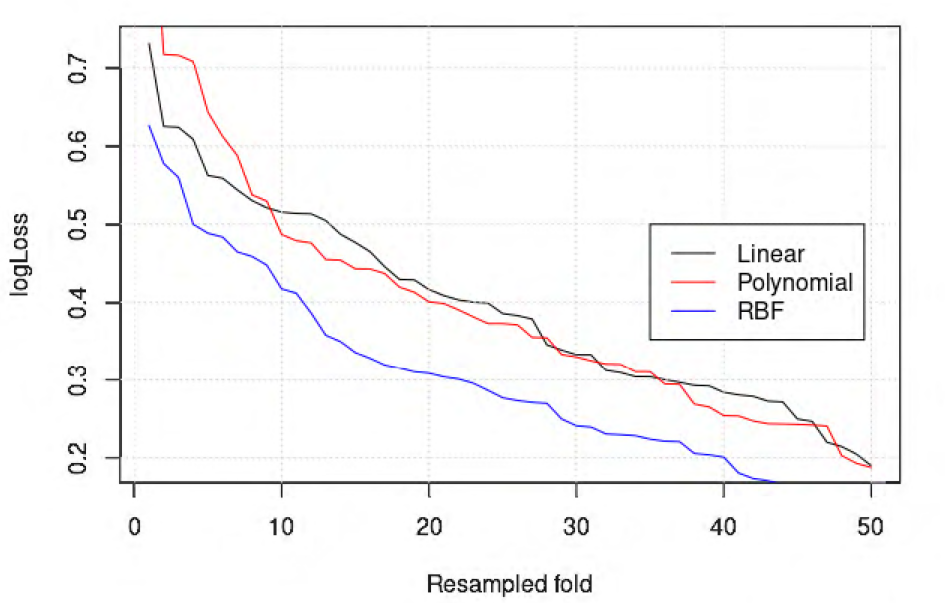

Figure 3.33: Sample result target respondent SVM kernel function comparison

The RBF has proven to be the optimal kernel function for fitting the data, as it offers the best overall performance in relation to the loss function. The linear and polynomial functions have comparable performance. The RBF is therefore the preferred kernel for building the final model.

3.6.3 Multi-Layer Perceptron

3.6.3.1 Sample Result Target Respondent

As a start, the model is trained with one hidden layer. The number of neurons and the activation function are optimised using cross-validation, and the model loss functions are presented in Figure 3.34:

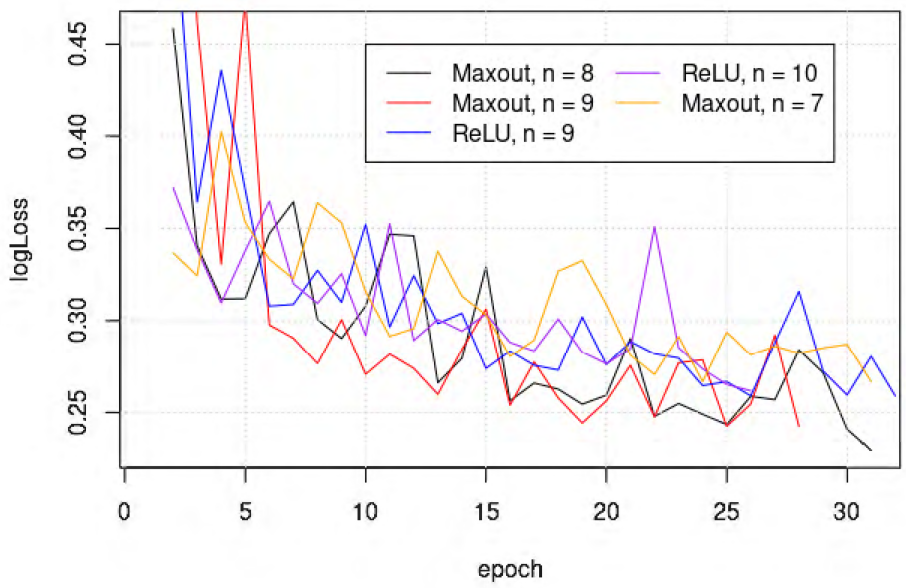

Figure 3.34: Sample result target respondent multi-layer perceptron training performance for one hidden layer. The activation function and number of neurons are the optimised parameters

It is difficult to tell from the model which of the combinations yields the best training performance. The maxout function with 8 neurons, however, appears to have the lowest training loss towards the last epoch. The table below summarises the respective model configurations in order of increasing log-loss. As there are $4 \times 6 = 24$ models built from cross-validation, only the top 5 are presented.

Table 3.8: Sample result target respondent multi-layer perceptron training performance for one hidden layer

Model	Hidden layers	Neurons	Activation function	Log-loss
Multi-layer perceptron	1	8	Maxout	0.2293
Multi-layer perceptron	1	9	Maxout	0.2425
Multi-layer perceptron	1	9	ReLU	0.259
Multi-layer perceptron	1	10	ReLU	0.2617
Multi-layer perceptron	1	7	Maxout	0.2667

It is evident that the maxout activation function is dominating the performance, followed by the ReLU function. The optimal number of neurons for the first hidden layer is 8, as it presents the lowest training loss. The model might be overfitting in cases where $n > 8$.

The addition of a second hidden layer, while keeping the units of the first hidden layer at the optimal value of 8, is presented in the following table:

Table 3.9: Sample result target respondent multi-layer perceptron training performance for two hidden layers

Model	Hidden layers	Neurons	Activation function	Log-loss
Multi-layer perceptron	2	[8, 6]	Maxout	0.1356
Multi-layer perceptron	2	[8, 8]	Maxout	0.1622
Multi-layer perceptron	2	[8, 9]	Maxout	0.1973
Multi-layer perceptron	2	[8, 5]	Maxout	0.2038
Multi-layer perceptron	2	[8, 3]	Maxout	0.2203

The table indicates that an additional hidden layer improves training performance. The best configuration involves the second hidden layer with 6 neurons. Since this is a significant improvement from the training performance of the model with one hidden layer, this configuration is the preferred one for building the final model.

3.6.3.2 Inclusion Type Target Respondent

The model training performance for one hidden unit is shown in Figure 3.35. The activation functions are compared:

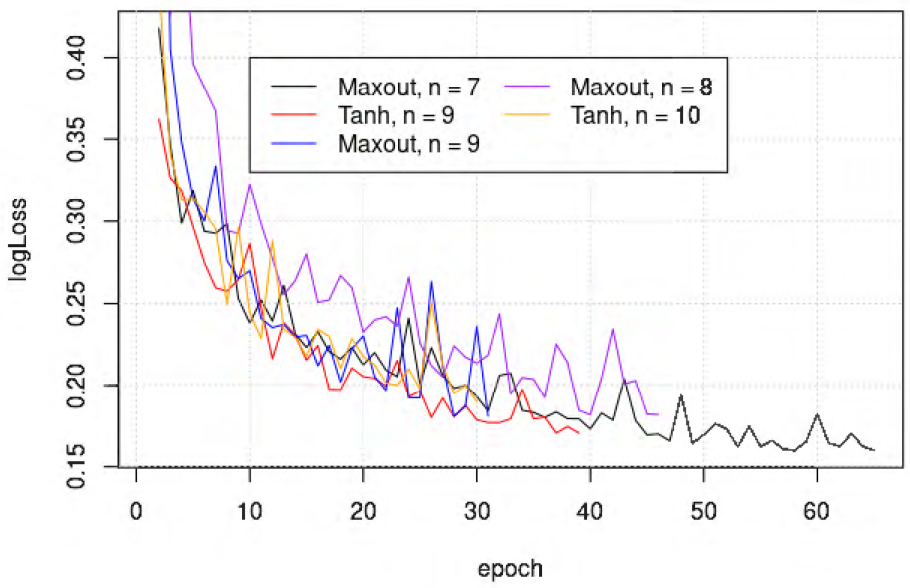

Figure 3.35: Inclusion type target respondent multi-layer perceptron training performance for one hidden layer. The activation function and number of neurons are the optimised parameters

The model configurations indicate comparable training loss performances, also indicating a convergence condition. The following table shows the model configurations ordered by increasing log-loss:

Table 3.10: Sample result target respondent multi-layer perceptron training performance for one hidden layer

Model	Hidden layers	Neurons	Activation function	Log-loss
Multi-layer perceptron	1	7	Maxout	0.16
Multi-layer perceptron	1	9	Tanh	0.1707
Multi-layer perceptron	1	9	Maxout	0.1809
Multi-layer perceptron	1	8	Maxout	0.1818
Multi-layer perceptron	1	10	Tanh	0.1898

The maxout activation function dominates the performance for the single hidden layer configuration of the model, followed by the tanh function. It is therefore the optimal activation function used in building the final model.

The second hidden layer is added to the configuration, and the training results are shown in the following table:

Table 3.11: Inclusion type target respondent multi-layer perceptron training performance for two hidden layers

Model	Hidden layers	Neurons	Activation function	Log-loss
Multi-layer perceptron	2	[8, 3]	Maxout	0.1236
Multi-layer perceptron	2	[8, 8]	Maxout	0.1666
Multi-layer perceptron	2	[8, 9]	Maxout	0.1872
Multi-layer perceptron	2	[8, 5]	Maxout	0.1894
Multi-layer perceptron	2	[8, 10]	Maxout	0.2

The performance for the configuration with the second hidden layer shows only a slight improvement from the configuration with a single hidden layer. This means that the configuration with a single hidden layer can be used without compromising too much training loss.

3.6.4 Radial Basis Function Network

3.6.4.1 Sample Result Target Respondent

The negative threshold tuning by means of repeated cross-validation is shown in Figure 3.36:

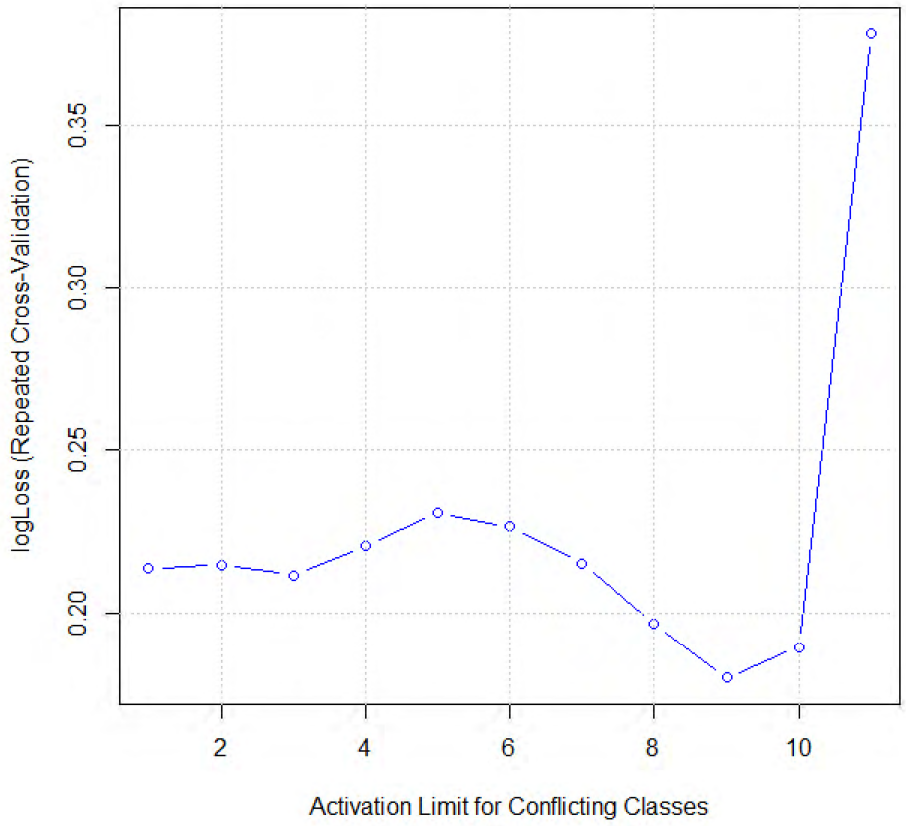

Figure 3.36: Sample result target respondent RBF training performance

The loss curve shows a dip at 0.8 and a sharp incline. The optimal threshold is therefore 0.8.

3.6.4.2 Inclusion Type Target Respondent

The negative threshold tuning by means of repeated cross-validation is shown in Figure 3.37:

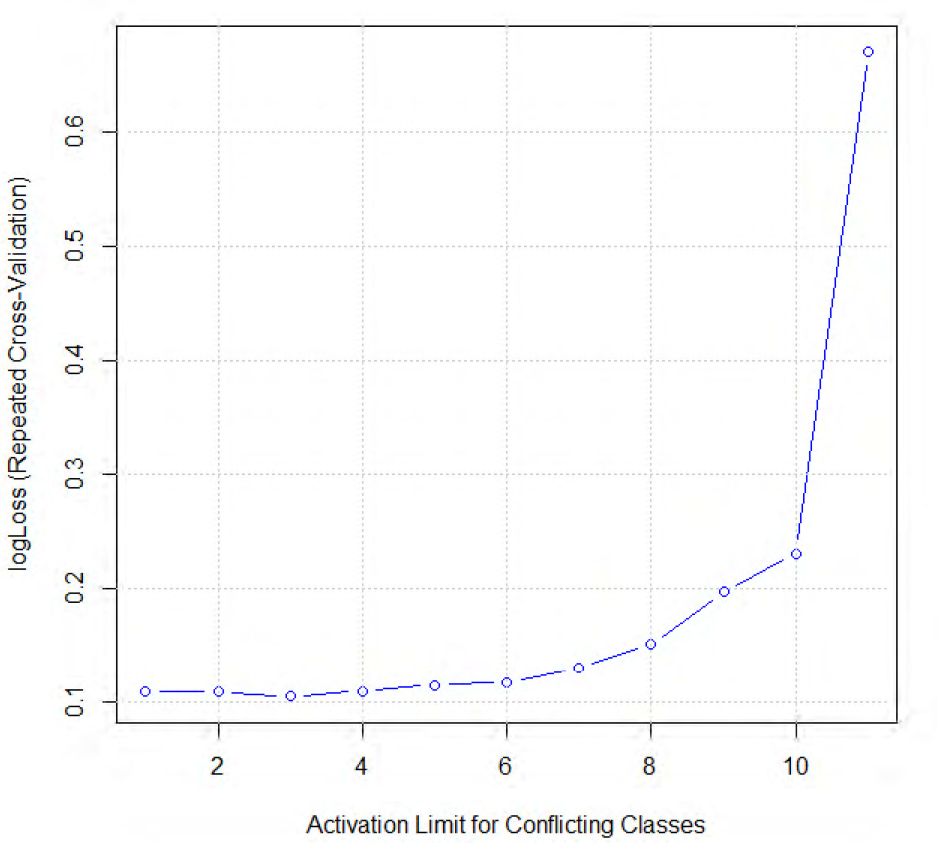

Figure 3.37: Inclusion type target respondent RBF training performance

The log-loss function has its minimum at a threshold of 0.2, before it steadily increases. The optimal threshold used is therefore 0.2.

Chapter 4

Results and Analysis

The previous sections presented the optimal training of two classical models and two neural networks using their respective hyperparameters and 10-fold repeated cross-validation. These hyperparameters serve as a basis for comparing the models in order to determine the best model for the dataset. In this section, the models are tested on the test data split from the training data. The test data consists of 126 observations and constitutes 25% of the total data.

The test results are presented in the form of a confusion matrix, which quantifies how well the model performs on unknown data.

4.1 Results

4.1.1 Sample Result Target Respondent

4.1.1.1 Logistic Regression

The performance of the logistic regression model is given in the following table:

Table 4.1: Logistic regression model performance for the sample result target respondent

(a) Confusion matrix

*Predicted**Actual*	Failed	Passed
Failed	36	7
Passed	4	79

(b) Metric scores

Metric	Target	95% CI	Score
Accuracy	0.95	0.9 - 1	0.91
Precision	0.95	0.86 - 0.95	0.95
Sensitivity	0.9	0.86 - 0.95	0.92
Specificity	0.9	0.86 - 0.95	0.9
ROC	0.9	0.86 - 0.95	0.91
Kappa	0.7	0.67 - 0.74	0.8

Th logistic regression model has performed satisfactorily as it satisfied the metrics except for accuracy, where it achieved 0.4% below the target. This is within the 95% confidence interval, so it is considered a success.

4.1.1.2 Support Vector Machine

The performance of the SVM model is given in the following table:

Table 4.2: SVM model performance for the sample result target respondent

(a) Confusion matrix

Predicted\Actual	Failed	Passed
Failed	34	0
Passed	6	86

(b) Metric scores

Metric	Target	95% CI	Score
Accuracy	0.9	0.9 - 1	0.95
Precision	0.9	0.86 - 0.95	0.93
Sensitivity	0.8	0.86 - 0.95	1
Specificity	0.8	0.86 - 0.95	0.85
ROC	0.8	0.86 - 0.95	0.88
Kappa	0.7	0.67 - 0.74	0.89

The SVM model gave a better overall performance than the logistic regression model. It achieved a higher score for each of the performance metrics, with a perfect score for sensitivity. It is therefore regarded a success.

4.1.1.3 Multi-layer Perceptron

The performance of the MLP model is given in the following table:

Table 4.3: MLP model performance for the sample result target respondent

(a) Confusion matrix

Predicted\Actual	Failed	Passed
Failed	36	2
Passed	4	84

(b) Metric scores

Metric	Target	95% CI	Score
Accuracy	0.9	0.9 - 1	0.95
Precision	0.9	0.86 - 0.95	0.96
Sensitivity	0.8	0.86 - 0.95	0.98
Specificity	0.8	0.86 - 0.95	0.9
ROC	0.8	0.86 - 0.95	0.91
Kappa	0.7	0.67 - 0.74	0.89

The MLP model has so far shown the best performance as it has exceeded all the target scores.

4.1.1.4 Radial Basis Function Network

The performance of the RBF network model is given in the following table:

Table 4.4: RBF network model performance for the sample result target respondent

(a) Confusion matrix

Predicted\Actual	Failed	Passed
Failed	36	6
Passed	4	80

(b) Metric scores

Metric	Target	95% CI	Score
Accuracy	0.95	0.9 - 1	0.92
Precision	0.9	0.86 - 0.95	0.95
Sensitivity	0.8	0.86 - 0.95	0.93
Specificity	0.8	0.86 - 0.95	0.9
ROC	0.8	0.86 - 0.95	0.91
Kappa	0.7	0.67 - 0.74	0.82

The RBF network model has also exceeded all target scores, although its performance is slightly below that of the MLP.

4.1.1.5 Results Summary

The following table shows a side-by-side comparison of the models:

Table 4.5: Model performance comparisons for sample result target respondent

Metric	Logistic Regression	Support Vector Machine	Multi-layer Perceptron	Radial Basis Function Network
Accuracy	0.91	0.95	0.95	0.92
Precision	0.95	0.93	0.96	0.95
Sensitivity	0.92	1	0.98	0.93
Specificity	0.9	0.85	0.9	0.9
ROC	0.91	0.88	0.91	0.91
Kappa	0.8	0.89	0.89	0.82

Figure 4.1 shows the comparison between the models:

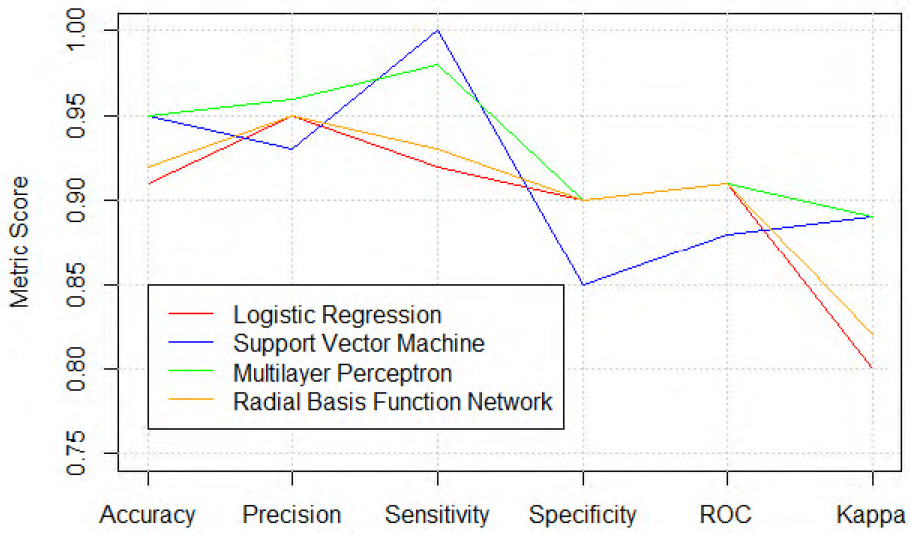

Figure 4.1: Model performance comparisons for the sample result target respondent

The models have all shown the capability to generalise well over the training data. This can be seen in the fact that the confusion matrices have shown good scores in testing performance over data that the models have not seen before. The logistic regression model, while the worst performing from the four, is still within the 95% tolerance of the target metrics. The MLP, SVM and RBF network models all performed well. The MLP gave the best performance, and is therefore recommended as the model to use. This is because the costs associated with each false alarm or miss are high within the context of an Aluminium manufacturing factory. Each loss can potentially cost the business hundreds of thousands of Rands.

4.1.2 Inclusion Type Target Respondent

For the multiclass problem, the metric scores are presented per class, so as to assess the performance of the model over individual classes in addition to the overall performance.

4.1.2.1 Logistic Regression

The performance of the logistic regression model is shown in the following table:

Table 4.6: Logistic regression model performance for the inclusion type target respondent

(a) Confusion matrix

Prd\Act	FeO	MgO	SPINEL
FeO	39	9	0
MgO	6	1	3
SPINEL	5	27	36

(b) Metric scores

Metric	Target	95% CI	FeO	MgO	SPINEL	Overall
Accuracy	0.95	0.9 - 1	0.83	0.46	0.78	0.69
Precision	0.9	0.86 - 0.95	0.82	0.1	0.53	0.48
Sensitivity	0.8	0.76 - 0.84	0.78	0.02	0.92	0.57
Specificity	0.8	0.76 - 0.84	0.88	0.9	0.63	0.8
Kappa	0.7	0.67 - 0.74	0.4			

The results show that the logistic regression model has scored below the target overall, except for specificity. Even for specificity, the per-class scores show that it achieved 0.63 for the SPINEL inclusion type, which is below target by 0.13. The best scores achieved are for FeO, which are also below target. This makes sense as the value of $\alpha = 0$ reduces the model to a constant logit function which is insensitive to the input.

4.1.2.2 Support Vector Machine

The performance of the SVM model is shown in the following table:

Table 4.7: SVM model performance for the inclusion type target respondent

(a) Confusion matrix

Prd\Act	FeO	MgO	SPINEL
FeO	32	7	0
MgO	15	24	1
SPINEL	3	6	38

(b) Metric scores

Metric	Target	95% CI	FeO	MgO	SPINEL	Overall
Accuracy	0.95	0.9 - 1	0.77	0.73	0.94	0.82
Precision	0.9	0.86 - 0.95	0.82	0.6	0.81	0.75
Sensitivity	0.8	0.76 - 0.84	0.64	0.65	0.97	0.75
Specificity	0.8	0.76 - 0.84	0.91	0.82	0.9	0.88
Kappa	0.7	0.67 - 0.74	0.62			

The SVM performance is better than the performance of the logistic regression model, with all the overall scores higher for the SVM than the logistic regression model. The model, however, did not meet all targets. The model scored above target only for the specificity class. The scores for accuracy, precision and sensitivity are not as far below target as for the logistic regression model. The value of kappa also indicates that there is substantial value in the model agreement with the dataset, as opposed to a completely random guess of the data [8]. The model, however, is considered inadequate as it does not satisfy the target metrics.

4.1.2.3 Multi-Layer Perceptron

The performance of the logistic regression model is shown in the following table:

Table 4.8: MLP model performance for the inclusion type target respondent

(a) Confusion matrix

Prd\Act	FeO	MgO	SPINEL
FeO	45	7	0
MgO	5	29	0
SPINEL	0	1	39

(b) Metric scores

Metric	Target	95% CI	FeO	MgO	SPINEL	Overall
Accuracy	0.95	0.9 - 1	0.9	0.86	0.99	0.92
Precision	0.9	0.86 - 0.95	0.87	0.85	0.98	0.9
Sensitivity	0.8	0.76 - 0.84	0.9	0.78	1	0.89
Specificity	0.8	0.76 - 0.84	0.91	0.94	0.99	0.95
Kappa	0.7	0.67 - 0.74	0.84			

The MLP is once again showing the best performance so far, with targets for precision, sensitivity and specificity met. The accuracy is slightly below target, but is still within the tolerance. The sensitivity and specificity have been well exceeded,

as the model especially gave few erroneous predictions for the SPINEL class. The MLP model is therefore considered a success.

4.1.2.4 Radial Basis Function Network

The performance of the logistic regression model is shown in the following table:

Table 4.9: RBF network model performance for the inclusion type target respondent

(a) Confusion matrix

Prd\Act	FeO	MgO	SPINEL
FeO	30	6	0
MgO	16	17	0
SPINEL	4	14	39

(b) Metric scores

Metric	Target	95% CI	FeO	MgO	SPINEL	Overall
Accuracy	0.9	0.9 - 1	0.76	0.64	0.9	0.77
Precision	0.9	0.86 - 0.95	0.83	0.52	0.68	0.68
Sensitivity	0.8	0.76 - 0.84	0.6	0.46	1	0.69
Specificity	0.8	0.76 - 0.84	0.92	0.82	0.79	0.84
Kappa	0.7	0.67 - 0.74	0.53			

The RBF network model performance is worse than that of the MLP model for all the metrics. This implies that the application of radial basis functions as activation functions for the classification of inclusions gives a worse performance than applying maxout functions, which are used in the MLP.

4.1.3 Results Summary

The following table shows a side-by-side comparison of the models:

Table 4.10: Model performance comparisons for sample result target respondent

Metric	Logistic Regression	Support Vector Machine	Multi-layer Perceptron	Radial Basis Function Network
Accuracy	0.69	0.82	0.92	0.77
Precision	0.48	0.75	0.9	0.68
Sensitivity	0.57	0.75	0.89	0.69
Specificity	0.8	0.88	0.95	0.84
Kappa	0.4	0.62	0.84	0.53

Figure 4.2 shows the comparison between the models:

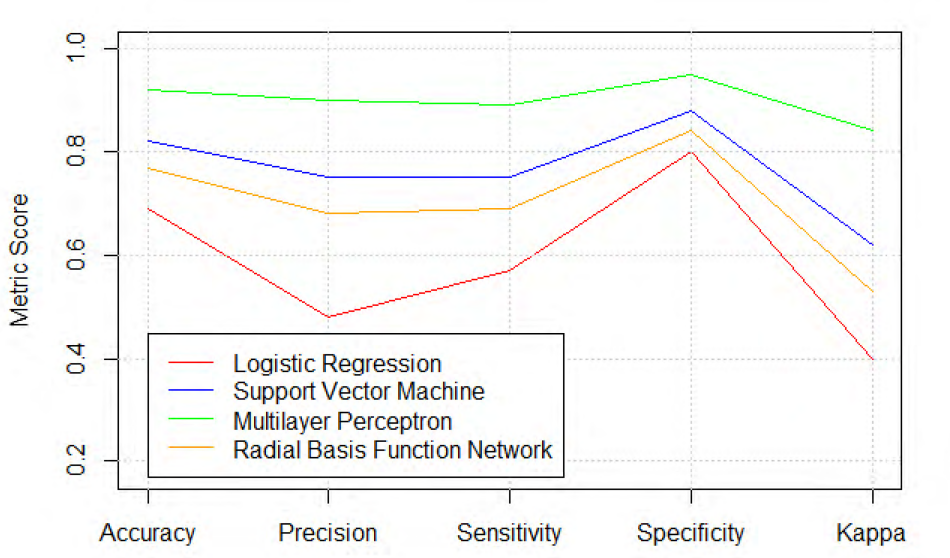

Figure 4.2: Model performance comparisons for the sample result target respondent

The problem of generalising over the inclusion types has proven to be much more difficult to solve than predicting the outcome of the metal quality. This might be attributed to the following:

- The attenuation caused by the different inclusions is similar from an ultrasonic point of view.
- The inclusion sizes and counts for the different classes are similar and not easily separable. This could be due to the filter that the metal passes through just before the casting stage.
- The results of the metallographic analysis used to classify the inclusions are not entirely reliable due to operator error.

The MLP can therefore be considered as it provides the best results, and subsequent tuning of the model can improve performance.

4.2 Performance Optimisation

The previous subsection has shown that all the models are capable of providing good predictions over the sample result target respondent. This is largely because the results of the failed class were based on samples that actually failed, and therefore the separability between the samples that passed and those that failed have enabled the models to generalise well over the dataset. The same cannot be said for the inclusion classification problem, as the prediction scores for the models were largely below target.

4.2.1 Hyperparameter Tuning

The best performing model, namely the MLP, is tuned further in this section with the intention of assessing whether an improvement in performance can be achieved. In order to achieve this, more specialised tuning parameters can be iterated over using repeated cross-validation. The parameters are given in the following table:

Table 4.11: Multi-layer perceptron model hyperparameters

Parameter	Value
model_id	multi-layer perceptron
number of hidden layers	1 (universal approximation)
number of neurons	8 - 10 (8 optimal, change for reference)
loss function	categorical crossentropy
activation function (hidden layer)	maxout
activation function (output layer)	softmax
epsilon	0 - 1 (selection randomness probability)
l1	0 - 0.2 (Lasso regularisation)
l2	0 - 0.2 (Ridge regularisation)
rho	0.9 - 1 (gradient descent term)

The additional parameters from the table include:

- epsilon, which changes the selection randomness probability for the learning gradient. A large value of ε would mean that the learning diverges, while a small value would mean the the learning converges too slowly.
- L1, which is the Lasso regularisation parameter. It ensures that the model is penalised for learning loss so as to minimise the effect of some weights. A high value of L1 would see more weights being set to zero.
- L2, which is the Ridge regularisation parameter. It also penalises the cost function, but never sets the weights to zero.
- rho, which is the learning rate decay factor. It is responsible for ensuring that the gradient descent is smooth. Higher values of ρ tend to give better smoothing results.

4.2.2 Grid Search Results

The following multivariate plot shows the training performance of the models for the given hyperparameter grid search:

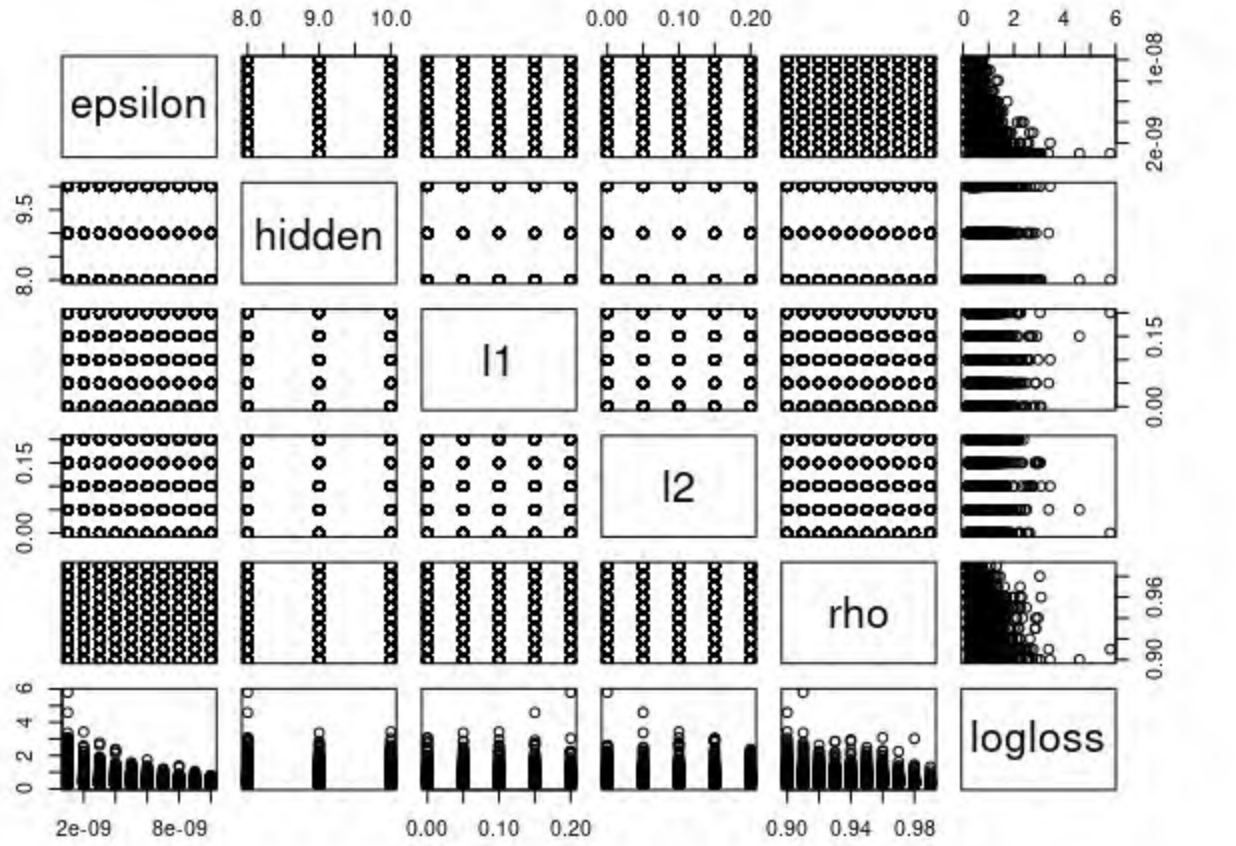

Figure 4.3: Model grid search training performance

The grid search produced 187 500 models based on the given hyperparameters. This is because a grid search runs all possible combinations as opposed to a random search, which randomly selects a combination of hyperparameters. A random search is given a stopping metric, whereas a grid search guarantees the best possible performance based on the given hyperparameter ranges. It is slower and more computationally expensive.

The grid search results revealed the following points:

- The number of hidden layers does not significantly improve the performance of the model beyond neurons. It is therefore confirmed that keeping the number of neurons at 8 and applying the law of universal approximation (one hidden layer) is sufficient for achieving an optimal model.
- The regularisation parameters $l1$ and $l2$ do not have a significant effect on the training performance of the model. This can be seen in the grid search plot, where their values are closely related with respect to the loss function of the model.
- The gradient descent term ρ has an inversely proportional relationship with the training loss of the model. It can therefore be set at its highest value in order to achieve the lowest training loss.
- The selection randomness probability ε has an inversely proportional relationship with the training loss of the model. It can therefore be set at its highest value in order to achieve the lowest training loss.

The following table shows a summary for the parameters for the top 5 models based on the lowest training log-loss:

Table 4.12: Grid search model log-loss performance

ε	hidden	$l1$	$l2$	ρ	logloss
1e-8	8	0	0.05	0.99	0.15622
1e-8	10	0	0.1	0.98	0.15838
3e-9	8	0.05	0	0.99	0.16224
4e-9	8	0	0.05	0.99	0.16750
8e-9	10	0.15	0	0.99	0.17360

Based on the table, it can be seen that the training performance of the model does not improve much as the hyperparameters are changed. It should also be noted that the training performance of the model is comparable to that of the multi-layer perceptron prior to the employment of a grid search.

4.3 Final Model Results

The model is built based on the best parameters, and tested on the test data. The following confusion matrix shows the performance of the model:

Table 4.13: MLP model performance after grid search

(a) Confusion matrix

Prd\Act	FeO	MgO	SPINEL
FeO	45	2	0
MgO	5	24	0
SPINEL	0	11	39

(b) Metric scores

Metric	Target	95% CI	FeO	MgO	SPINEL	Overall
Accuracy	0.95	0.9 - 1	0.94	0.86	0.91	0.86
Precision	0.9	0.86 - 0.95	0.96	0.83	0.78	0.86
Sensitivity	0.8	0.76 - 0.84	0.9	0.65	1	0.85
Specificity	0.8	0.76 - 0.84	0.97	0.94	0.87	0.93
Kappa	0.7	0.67 - 0.74	0.78			

Based on the confusion matrix and metric scores shown in the table, the following observations are made:

1. The model after grid search is not much better than the model before grid search. This is most likely an implication of the model having reached its learning potential.
2. The MgO inclusion has the worst performance. The metrics are below target except for specificity. This implies that the model is not able to generalise well over this inclusion type.
3. The SPINEL inclusion type is within the target limits except for the precision metric. For the other metrics, it has exceeded targets.
4. The FeO inclusion type has the best performance and has exceeded the targets for all metrics.

The model does not therefore generalise well over the inclusion types. A plot of the model's decision boundaries is shown in Figure 4.4:

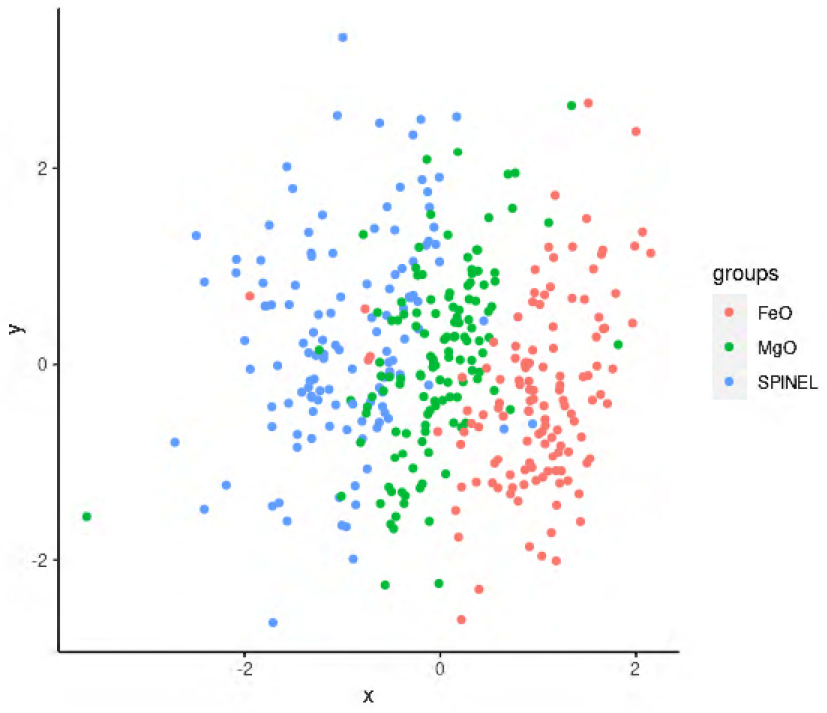

Figure 4.4: Model decision analysis grouped by inclusion type. The axes represent the principal components used to reduce data dimensionality.

As can be seen from the figure, there is substantial overlap between the SPINEL and MgO inclusion types. It is therefore unlikely that the model can separate the three classes sufficiently for it to reach all the performance metrics.

Chapter 5

Conclusions

5.1 Summary of Work Done

An opportunity has been identified in an Aluminium manufacturing plant to improve quality control by means of a pulsed ultrasound system. This system is capable of performing real-time measurements on molten metal, which reveal the cleanliness of the metal. In order to automate the process of accepting the metal as clean, supervised learning is applied.

The supervised learning component involves the development of a two-stage classifier. The first stage determines whether the metal quality is adequate for production. The second stage determines the dominant inclusion responsible for the quality deterioration. Four models are trained, namely logistic regression, support vector machine, multi-layer perceptron and a radial basis function network. For the passed/failed classification problem, all the models score within the metrics. The multi-layer perceptron offers the best performance and is therefore considered for production. For the inclusion classification problem, the models do not generalise as well over the training data, and the performance is borderline. The multi-layer perceptron proves to be the best performing model as it meets target for two out of three classes. In order to optimise the model to better generalise over the training data, a grid search involving several hyperparameters is performed. The hyperparameters are swept from zero to one with granular increments, resulting in the building of 187 500 models. The combination with the lowest training loss is selected and a final model is built. The results of the final model reveal that the performance is not significantly improved. This is an indication of the fact that the model has reached its learning potential and cannot generalise better than it already has over the training data. The possible reasons for this include operator error/bias during measurements and multiple sources of variance from the measurement technique.

A summary of the problem, method and results is shown in the following table:

Table 5.1: Summary of work done

Problem	**Requirement**	**Hypothesis**	**Method**	**Results**	
P1 Cast quality manually determined	**R2a** Implement autonomous classifier	**H2** Supervised learning classifier sufficient	MLP	**Metric**	**Score**
				Acc	0.95
				Prec	0.96
				Sens	0.98
				Spec	0.9
P2 Inclusions classified manually	**R2b** Implement autonomous classifier	**H2** Supervised learning classifier sufficient	MLP	**Metric**	**Score**
				Acc	0.92
				Prec	0.9
				Sens	0.89
				Spec	0.95

While the inclusion type classifier gave a boundary performance on accuracy and precision, the values are within the 95% tolerance range. The project is therefore considered a success.

5.2 Limitations of Current Work

The main cause for the model not generalising well over all the inclusion types can be attributed to operator errors, according to the domain experts in the plant. This is attributed to the fact that operators use a visual method, namely metallographic to identify the inclusions. The following are contributing factors to the error in measurements:

1. A deficit of adequately trained personnel. As a result, the identification of inclusions is delegated to students and in-service trainees.
2. The measurement technique itself. As PoDFA is the method used to extract and filter samples, there are several sources of variance including:
   (a) the ceramic foam filter used,
   (b) the settling time of the metal,
   (c) the amount of metal poured in the filtration system,
   (d) the time at which the metal was extracted from the furnace relative to the settling duration, and
   (e) the visual inspection of the metal by the operator.
3. The decision-making process. It is possible for an operator to identify more than one inclusion type in the metal, and in such cases the operator makes the call on the dominant inclusion. This is usually supported by the types of defects detected on the metal as well.

5.3 Recommendations for Future Work

Recommendations for future work are three-fold. Firstly, unsupervised learning is recommended. Secondly, research-based recommendations are made. The last part is based on process-related considerations.

5.3.1 Unsupervised Learning for Anomaly Detection

Unsupervised learning has not been implemented as it did not form part of the scope for this work. It is, however, recommended as an additional method for real-time condition monitoring for individual variables and the multivariable process. Techniques like the Hotelling's T^2 statistic can be computed with control limits determined by the business, so as to ensure that the process is monitored for being in control.

5.3.2 Research-Based Recommendations

The research potential for industrial processes is virtually limitless with regards to the fourth industrial revolution. This research focused on data produced by a single sensor. In order for a more holistic solution to be realised, more sensors can be integrated into the measurements. The following potential opportunities have been identified while conducting this research.

5.3.2.1 Vision System

During casting, the metal forms a thin oxidisation layer on the surface, which is an indication of the presence of some inclusions at the top of the metal. A vision system can be employed to analyse the texture, colour and other visual properties of the metal in order to provide more insights relating to the nature of inclusions, the intensity of the inclusions and the effects of different casting parameters on the texture of the metal.

5.3.2.2 Fourier Transform Infrared Spectroscopy

The attenuation levels of inclusions compared to pure Aluminium could produce different infrared signatures, which could be measured and analysed using Fourier Transforms. this is because different elements possess different reflectance and attenuation properties at different wavelengths. Classifiers can then be built to determine the types and intensities of inclusions based on the spectral properties of the measurements.

5.3.3 Process-Based Recommendations

At first, the recommendations are process-based and pertain to the improvement of current processes for better collection of data. These recommendations include:

1. The usage of a K-Mold system for more objective inclusion classifications.
2. Operator training for better equipment use.
3. Employment of dedicated resources to performing measurements.
4. Regular Gage R&R analysis for maintaining measurement integrity. This analysis helps quantify process variations and the responsible components by measuring repeatability and reproducibility of results in the specific process [58].
5. The deployment of more MV20/20 systems at different points along the cast.

More reliable data can then be collected in order to assess whether the models can generalise better and meet the target thresholds for performance.

Bibliography

[1] S. V. Buer, G.I. Fragapane and J.O. Strandhagen, *The data-driven process improvement cycle: Using digitalization for continuous improvement.* IFAC-PapersOnLine. 2018.

[2] S. K. Dwivedi, M. Vishwakarma and A. Soni, *Advances and researches on non destructive testing: A review.* Materials Today: Proceedings. 2018.

[3] M. F. Gándara, *Aluminium: the metal of choice.* Mater. Tehnol. 2013.

[4] R. Gallo, *Differentiating Inclusions in Molten Aluminium Baths and in Castings.* Pyrotek Inc. OH, USA, 2017

[5] E. Eckert and B. Cochran, *The Importance of Metal Quality in Molten Secondary Aaluminium.* The Minerals, Metals and Materials Society. 2000

[6] R. Gallo, H. Mountford and I. Sommerville, *Ultrasound for On-Line Inclusion Detection in Molten Aluminium Alloys: Technology Assessment.* First International Conference on Structural Aluminium Castings, Orlando, Florida, USA. 2003

[7] S. Poynton, M. Brandt and J. Grandfield, *A review of inclusion detection methods in molten aluminium.* Light Metals, 3. 2009.

[8] J. R. Landis and G. G. Koch, *The Measurement of Observer Agreement for Categorical Data.* International Biometric Society. 1977

[9] T. Hastie, R. Tibshirani and J. Friedman, *The Elements of Statistical Learning Second Edition.* Stanford University, Stanford, California. 2001

[10] ABB Inc., *PoDFA | The complete solution for inclusion measurement.* Revision A01, 2016.

[11] D. Veillette and D Paquin, *Metallographic Analyses.* International Aluminium Casting, Canada. 2006

[12] M B. Djurdjević, Z. Odanović and J. Pavlović-Krstić, *Melt Quality Control at Aluminium Casting Plants.* Association of Metallurgical Engineers of Serbia. 2010

[13] Dy-Kast Supply, *Evaluation of Aluminium Melt Quality With K-Mold Fracture Test.* Avon Ohio, USA. 2021

[14] ABB Inc., *Prefil ® -Footprinter | Pressure Filtration Melt Cleanliness analyser.* Revision B01. 2016

[15] ABB Inc., *Mobile liquid aluminium cleanliness analyser.* ABB Inc., Oerlikon, Zurich, Switzerland. July 2017. [Online]. Available: https://new.abb.com/products/measurement-products/analytical/metallurgical-analysers/limca-iii

[16] ABB Inc., *LiMCA III | Liquid Metal cleanliness analyser.* 2017

[17] D D. Smith, B Hixson, H Mountford and I Sommerville, *Practical Use of the MetalVision Ultrasonic Inclusion analyser.* JW Aluminium, 528 Old Mt Holly Road, Goose Creek, South Carolina, 29445, USA. 2015

[18] R. Gallo, H. Mountford and I. Sommerville, *Ultrasound for On-Line Inclusion Detection in Molten Aluminium Alloys: Technology Assessment.* First International Conference on Structural Aluminium Castings, Orlando, Florida, USA. 2003

[19] S C. Trikootam and M Hornikx, *The wind effect on sound propagation over urban areas: Experimental approach with an uncontrolled sound source.* Building and Environment Volume 149, Elsevier Ltd. 2018

[20] Iowa State University, *Nondestructive Evaluation and Nondestructive Testing Education: Ultrasonic Testing*. Center for Nondestructive Evaluation, Iowa State University, Iowa, USA. 2001

[21] P. Kayaroganam, *Response Surface Methodology in Engineering Science*. London, United Kingdom, IntechOpen, 2021 [Online]. Available: https://www.intechopen.com/books/10198 doi: 10.5772/intechopen.90965

[22] T. Bayrak, *A Review of Business Analytics: A Business Enabler or Another Passing Fad*. Western New England University, 1215 Wilbraham Rd. Springfield, MA, 01119, USA. 2015

[23] TAC-12 Migrant & Seasonal Head Start Technical Assistance Center, *Introduction to Data Analysis Handbook*. Academy for Educational Development 1875 Connecticut Avenue, NW Washington, DC 20009. Spring, 2006

[24] C. Wild and G. Seber, *CHANCE ENCOUNTERS: A First Course in Data Analysis and Inference*. John Wiley & Sons, New York. 2006

[25] I. A. Rana and Chancellor A. Rehman, *Past, Present and Future of Business Analytics – A Review*. International Journal of Management Sciences and Business Research, 2014 ISSN (2226-8235) Vol-3, Issue 9. 2014

[26] Y. Duan, G. Cao and J. S. Edwards, *Understanding the Impact of Business Analytics on Innovation*. Business School, University of Bedfordshire, Luton, LU1 3JU. 2018.

[27] G. C. Souza, *Supply chain analytics*. Business Horizons, 57(5). 2014.

[28] D. Delen and S. Ram, *Research challenges and opportunities in business analytics*. Journal of Business Analytics, 1(1). 2018.

[29] W. W. Eckerson, *Predictive analytics. Extending the Value of Your Data Warehousing Investment*. TDWI Best Practices Report. 2007.

[30] D. Frazzetto, T.D. Nielsen, T.B. Pedersen and L. Šikšnys, *Prescriptive analytics: a survey of emerging trends and technologies*. The VLDB Journal, 28(4). 2019.

[31] I. Goodfellow, Y. Bengio and A. Courville, *Deep Learning*. MIT Press, 2016. Accessed on: Oct. 4, 2021 [Online]. Available: https://www.deeplearningbook.org/

[32] Y. Zhang and C. Ling, *A strategy to apply machine learning to small datasets in materials science*. npj Computational Materials 4, Atricle Number 25. 2018 (Online). Available: https://doi.org/10.1038/s41524-018-0081-z

[33] G. F. M. de Souza, A. C. Netto, A. H. de Andrade Melani, M. A. de Carvalho Michalski and R. F. da Silva, *Reliability Analysis and Asset Management of Engineering Systems*. Elsevier, 2021.

[34] Y. Song and Y. Lu, *Decision tree methods: applications for classification and prediction*. Shanghai Archives of Psychiatry, 2015

[35] S. Shahane, N. Aluru, P. Ferreira, S. G. Kapoor and S. P. Vanka, *Optimization of solidification in die casting using numerical simulations and machine learning*. Journal of Manufacturing Processes, Volume 51, 2020

[36] M. Torabi Rad , A. Viardin, G. J. Schmitz, and M. Apel, *Theory-training deep neural networks for an alloy solidification benchmark problem*. arXiv: 1912.09800v1. 2019

[37] Mery, D, *Aluminum Casting Inspection Using Deep Learning: A Method Based on Convolutional Neural Networks*. J Nondestruct Eval 39, 12 (2020). https://doi.org/10.1007/s10921-020-0655-9

[38] A. Ertas and J. Jones, *The Engineering Design Process*. 2nd ed. New York, N.Y.. John Wiley & Sons. 1996

[39] S. Minaee, "An introduction to the most important metrics for evaluating classification, regression, ranking, vision, NLP, and deep learning models", *20 Popular Machine Learning Metrics. Part 1: Classification & Regression Evaluation Metrics,* Oct. 2019. Accessed on: Sept. 26, 2021 [Online]. Available: https://towardsdatascience.com/20-popular-machine-learning-metrics-part-1-classification-regression-evaluation-metrics-1ca3e282a2ce

[40] A. I. Khuri and S. Mukhopadhyay, *Response surface methodology*. Wiley Interdisciplinary Reviews: Computational Statistics. 2010.

[41] P. Mishra, C. M. Pandey, U. Singh, A. Gupta, C. Sahu and A. Keshri, *Descriptive statistics and normality tests for statistical data*. Ann Card Anaesth. 2019.

[42] N. Salem and S. Hussein, *Data dimensional reduction and principal components analysis.* Procedia Computer Science, 163. 2019.

[43] P. Peduzzi, J. Concato, E. Kemper, T R. Holford and A R. Feinstein, *A Simulation Study of the Number of Events per Variable in Logistic Regression Analysis.* Departments of Medicine (Clinical Epidemiology Unit) and Epidemiology and Public Health, Yale University School of Medicine, New Haven, Conneticut, USA, 06510. 1996

[44] S. Kung, *Kernel Methods and Machine Learning.* Cambridge: Cambridge University Press. 2014.

[45] Scikit-Learn, *KernelPCA.* Accessed Sept. 9, 2021. [Online]. Available on: https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.KernelPCA.html

[46] A Jain, *A Complete Tutorial on Ridge and Lasso Regression in Python*, Jan. 2016. Accessed on: Oct. 2, 2021 [Online]. Available: https://www.analyticsvidhya.com/blog/2016/01/ridge-lasso-regression-python-complete-tutorial

[47] C. Hsu, , C. Chang and C. Lin, *A Practical Guide to Support Vector Classification.* Department of Computer Science, National Taiwan University, Taipei 106, Taiwan. 2016

[48] A. Abraham, *Artificial Neural Networks.* Oklahoma State University, Stillwater, OK, USA. 2005

[49] P. Baheti, *12 Types of Neural Network Activation Functions: How to Choose?.* V Labs, 2021. Accessed on Oct. 5, 2021 [Online]. Available: https://www.v7labs.com/blog/neural-networks-activation-functions#choose-activation-function

[50] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville and Y. Bengio, *Maxout Networks.* D´epartement d'Informatique et de Recherche Op´erationelle, Universit´e de Montr´eal 2920, chemin de la Tour, Montr´eal, Qu´ebec, Canada, H3T 1J8. 2013

[51] C. E. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, *Activation Functions: Comparison of Trends in Practice and Research for Deep Learning.* arXiv:1811.03378v1 [cs.LG]. 2018

[52] Y. Wu, H. Wang, B. Zhang and K. L. Du, *Using Radial Basis Function Networks for Function Approximation and Classification.* Enjoyor Laboratories, Enjoyor Inc., Hangzhou 310030, China and Department of Electrical and Computer Engineering, Concordia University, Montreal, QC, Canada H3G 1M8. 2011

[53] J. Park and I. W. Sandberg, *Universal Approximation using Radial-Basis-Function Networks.* Department of Electrical and Computer Engineering, University of Texas at Austin, Austin, Texas 78712, USA

[54] N. B. Karayiannis, *Reformulated Radial Basis Neural Networks Trained by Gradient Descent.* IEEE Transactions on Neural Networks Vol. 10, No. 3. 1999

[55] K. Kawaguchi, L. P. Kaelbing and Y. Bengio, *generalisation in Deep Learning.* Massachusetts Institute of Technology and University of Montreal, 2020

[56] A. L. I. Oliveira, B. J. M. Melo, F. B. L. Neto and S. R. L. Meira, *Combining Data Reduction and Parameter Selection for Improving RBF-DDA Performance.* IBERAMIA 2004. IBERAMIA 2004. Lecture Notes in Computer Science, vol 3315. Springer, Berlin, Heidelberg. 2004

[57] N. Mache, *The Dynamic Decay Adjustment Algorithm.* Stuttgart Neural Network Simulator, University of Stuttgart. Maintained at the University of Tübingen. Accessed on Nov. 16 2021 [Online]. Available: https://www.ra.cs.uni-tuebingen.de/SNNS/UserManual/node193.html

[58] M. A. Durivage, *Practical Attribute and Variable Measurement Systems Analysis (MSA): A Guide for Conducting Gage R&R Studies and Test Method Validations.* Asq Quality Press. 2015

Appendix

Appendix A

Source Code

This section contains the source code developed during the execution of this research. Most of the source code is developed in the R programming language.

A.1 Exploratory Data Analytics

Algorithm A.1 Data exploration

```
rm(list = ls())

library(DataExplorer)

d <-
  read.csv(file = "../Data/5182.csv",
           header = TRUE,
           stringsAsFactors = TRUE)
datalength <- length(d)
xdata = d[, 1:(datalength - 2)]
ydata = d[, datalength - 1]

create_report(data = d,
              output_file = "data_exploration.html",
              y = "SAMPLE_RESULT",
              report_title = "Data Exploration",
              config = configure_report(add_introduce = TRUE,
                                        add_plot_intro = TRUE,
                                        add_plot_str = TRUE,
                                        add_plot_histogram = TRUE,
                                        add_plot_qq = TRUE,
                                        add_plot_bar = TRUE,
                                        add_plot_correlation = TRUE,
                                        add_plot_prcomp = TRUE,
                                        add_plot_boxplot = TRUE,
                                        plot_prcomp_args = TRUE
                                                                )
                                    )
```

The output of this source code is an html report showing a summary of the data. The report contains:

- An introduction which shows the numerical and categorical features as well as missing values,

- The structure of the data,
- A histogram of the continuous variables,
- A quantile-quantile plot of the continuous variables,
- A bar plot of the categorical variables,
- A correlation plot of the continuous variables,
- Boxplots of the continuous variables, and
- Principal components.

A.2 Normality Test

The source code used for the determination of normal variables is given in the following listing:

Algorithm A.2 Normality test

```
rm(list = ls())

library(qicharts2)

d <-
  read.csv(file = "../Data/5182.csv",
           header = TRUE,
           stringsAsFactors = TRUE)
datalength <- length(d)
xdata = d[, 1:(datalength - 2)]

df = list()
for (i in 1:length(xdata)) {
  s <- shapiro.test(xdata[, i])
  df[[i]] <-  c(colnames(xdata)[i], as.numeric(s$statistic[[1]]), as.numeric(s$p.value))
}
df <- do.call(rbind, df)
colnames(df) <- c("variable", "W statistic", "p-value")
```

The dataframe, df, contains three columns, namely the variable name, the test (W) statistic and the p-value.

A.3 Principal Components Analysis

A.3.1 Linear PCA

Principal components analysis is used for decomposing the data into several principal components as a feature extraction technique. Algorithms like k-means then use the principal components to perform the clustering.

The source code developed for computing and plotting the principal components is given in the following listing:

Algorithm A.3 Principal components analysis

```
rm(list = ls())

library(plotly)
library(stats)
library(data.table)

d <- read.csv(file = "../Data/test.csv", header = TRUE, stringsAsFactors = TRUE)
d <- d[d$SAMPLE_RESULT == 'PASSED', ]
datalength <- length(d)
xdata = d[, 1:(datalength - 2)]
prin_comp <- prcomp(xdata, rank. = 3
explained_variance_ratio <- summary(prin_comp)[["importance"]]['Proportion of Variance',]
explained_variance_ratio <- 100 * explained_variance_ratio
components <- prin_comp[["x"]]
components <- data.frame(components)
components <- cbind(components, d$SAMPLE_RESULT)
components$PC3 <- -components$PC3
components$PC2 <- -components$PC2

axis = list(showline=FALSE,
            zeroline=FALSE,
            gridcolor='#ffff',
            ticklen=4,
            titlefont=list(size=13))

# Plot PCA grid
fig <- components %>%
  plot_ly()  %>%
  add_trace(type = 'splom', dimensions = list(
  list(label=paste('PC 1 (',toString(round(explained_variance_ratio[1],1)),'%)',sep = ''),
       values=~PC1),
  list(label=paste('PC 2 (',toString(round(explained_variance_ratio[2],1)),'%)',sep = ''),
       values=~PC2)
  list(label=paste('PC 3 (',toString(round(explained_variance_ratio[3],1)),'%)',sep = ''),
       values=~PC3)
    ),
    color = ~d$SAMPLE_RESULT, colors = c('#636EFA','#EF553B','#00CC96')
  ) %>%
  style(diagonal = list(visible = FALSE)) %>%
  layout(legend=list(title=list(text='color')), hovermode='closest', dragmode= 'select',
    plot_bgcolor='rgba(240,240,240, 0.95)',
    xaxis=list(domain=NULL, showline=F, zeroline=F, gridcolor='#ffff', ticklen=4),
    yaxis=list(domain=NULL, showline=F, zeroline=F, gridcolor='#ffff', ticklen=4),
    xaxis2=axis, xaxis3=axis, xaxis4=axis, yaxis2=axis, yaxis3=axis, yaxis4=axis)
fig
# Plot 3D PCA
tit = '3D PCA'
fig <- plot_ly(components, x = ~PC1, y = ~PC2, z = ~PC3, color = ~d$INCLUSION_TYPE,
colors = c('#636EFA','#EF553B','#00CC96') ) %>% add_markers(size = 12)
fig <- fig %>% layout(title = tit, scene = list(bgcolor = "#e5ecf6"))
fig
```

A.3.2 Kernel PCA

The source code developed for computing the principal components using the kernel trick is shown in the following listing:

Algorithm A.4 Kernel principal components analysis

```
rm(list = ls())

library(kernlab)
library(plotly)
library(stats)

d <-
  read.csv(file = "../Data/5182.csv",
           header = TRUE,
           stringsAsFactors = TRUE)
datalength <- length(d)
xdata = d[, 1:(datalength - 2)]

prin_comp <- kpca(~., data = xdata, kernel = 'rbfdot',
        kpar = list(sigma = 1 / length(xdata)), features = 0)
plot(cumsum(eig(prin_comp) * 100), type = 'l',
        xlab = 'Principal Components', ylab = 'Explained Variance (%)')
grid()
```

A.4 Supervised Learning

for the supervised learning algorithms, the source code shown is specifically for the sample result target feature. This is done in order to avoid duplication, as the source code is the same except for the target parameter chosen and the classification family (binomial vs. multinomial).

A.4.1 Logistic Regression

The source code developed to implement the logistic regression algorithm is shown in the following listing:

Algorithm A.5 Logistic regression

```
rm(list = ls())

library(dplyr)
library(caret)
library(RSNNS)

set.seed(49237)

d <- read.csv(file = "../Data/5182.csv", header = TRUE, stringsAsFactors = TRUE)

datalength <- length(d)
xdata = d[, 1:(datalength - 2)]
ydata = d[, datalength]

td <- read.csv(file = "../Data/test.csv", header = TRUE, stringsAsFactors = TRUE)
xtest <- td[, 1:(datalength - 2)]
ytest <- td[, datalength]


# Cross-validation variables
k = 10 # 10-fold cross-validation
num_repeats = 5 # Repeats the cross-validation 5 times

completeSummary  <- function(data, lev = NULL, model = NULL) {
  a1 <- defaultSummary(data, lev, model)
  a1
}

# Setup for cross validation
trCtrl <- trainControl(method = "repeatedcv",
                       number = k, repeats = num_repeats,
                       summaryFunction = completeSummary,
                       classProbs = TRUE, savePred = TRUE,
                       returnResamp = "final")
alpha <- seq(from = 0, to = 1, by = 0.01)
lambda <- 1 / length(d)

# Logistic regression model
model_logreg <- caret::train(xdata, ydata, method = "glmnet",
                             tuneGrid = expand.grid(.alpha = alpha,
                             .lambda = lambda), trControl = trCtrl,
                             preProc = c("center", "scale"), # Center and scale data
                             family = "binomial", metric = "logLoss"
)

pred_logreg <- predict(model_logreg, newdata = xtest)
logreg_cm <- caret::confusionMatrix(data = pred_logreg,
                             reference = ytest, positive = "PASSED")
logreg_cm

plot(model_logreg, xlab = 'Alpha')
```

A.4.2 Support Vector Machine

The source code developed to implement the support vector machine algorithm is shown in the following listing:

Algorithm A.6 Support vector machine

```
rm(list = ls())

library(caret)
library(RSNNS)
library(pROC)
library(MLeval)

set.seed(49237)

d <- read.csv(file = "../Data/5182.csv", header = TRUE, stringsAsFactors = TRUE)
datalength <- length(d)
xdata = d[, 1:(datalength - 2)]
ydata = d[, datalength]

td <-
  read.csv(file = "../Data/test.csv",
           header = TRUE,
           stringsAsFactors = TRUE)
xtest <- td[, 1:(datalength - 2)]
ytest <- td[, datalength]


# Cross-validation variables
k = 10 # 10-fold cross-validation
num_repeats = 5 # Repeats the cross-validation 5 times

completeSummary  <- function(data, lev = NULL, model = NULL) {
  a1 <- defaultSummary(data, lev, model)
  a1
}

# Setup for cross validation
trCtrl <- trainControl(method = "repeatedcv", number = k, repeats = num_repeats,
                       summaryFunction = completeSummary, classProbs = TRUE,
                       savePred = TRUE, returnResamp = "final")
C <- seq(from = 0, to = 1, by = 0.1)
degree_pol <- c(1, 2, 3, 4, 5)
scale_pol <- seq(from = 0, to = 1, by = 0.1)

svm_poly <- caret::train(x = xdata, y = ydata, method = "svmPoly",
  tuneGrid = expand.grid(.C = C, .degree = degree_pol,  .scale = scale_pol), prox = TRUE,
                allowParallel = TRUE, preProc = c("center", "scale"), metric = "logLoss",
                trControl = trCtrl, family = 'binomial')
plot(svm_poly$resample$logLoss, decreasing = TRUE), type = 'l', col = 'black',
                xlab = 'Resampled fold', ylab = 'logLoss')
```

A.4.3 Multi-Layer Perceptron

The source code developed to implement the multi-layer perceptron algorithm is shown in the following listing:

Algorithm A.7 Multi-layer perceptron

```
rm(list = ls())

library(dplyr)
library(caret)
library(h2o)
library(hyperSpec)

d <- read.csv(file = "/home/kgothatso/Documents/mv2020data/5182.csv", header = TRUE,
           stringsAsFactors = TRUE)
xdata = d[, 1:18]
y_sample_res <- d[, 18]
y_sample_res <- as.factor(y_sample_res)

h2o.init()
input_data_sample_res <- as.h2o(x = xdata[c(1:16, 18)], destination_frame = "input_data")
n = sequence(from = 8, to = 20, by = 1)
for(act in c("Rectifier", "Tanh", "Maxout")) {
  nn_sample_res <- h2o.deeplearning(x = colnames(xdata[c(1:16)]), y = colnames(xdata)[18],
      model_id = paste0("nn_model_sample_res_", act),
      training_frame = input_data_sample_res, seed = 4778, nfolds = 10,
          overwrite_with_best_model = TRUE, use_all_factor_levels = TRUE,
      standardise = TRUE, activation = act, hidden = c(n), epochs = 1000,
          adaptive_rate = FALSE, rho = 0.99, epsilon = 1e-08, rate = 0.001,
          rate_annealing = 1e-06, loss = "CrossEntropy", diagnostics = TRUE,
      fast_mode = TRUE, variable_importances = TRUE, auc_type = "AUTO",
          verbose = TRUE)
  h2o.saveModel(nn_sample_res, path = ".", force = TRUE)
}

model_rect <- h2o.loadModel("~/nn_model_sample_res_Rectifier")
model_tanh <- h2o.loadModel("~/nn_model_sample_res_Tanh")
model_maxout <- h2o.loadModel("~/nn_model_sample_res_Maxout")

plot(
  model_rect@model$training_metrics@metrics$thresholds_and_metric_scores$fpr,
  model_rect@model$training_metrics@metrics$thresholds_and_metric_scores$tpr,
  type = 'l',
  col='blue',
  main = paste0("ROC Curves for Different Activation Functions"),
  xlab = "False Positive Rate",
  ylab = "True Positive Rate",
  panel.first = grid(
    nx = 10
  )
)
abline(coef = c(0,1))
lines(
  model_tanh@model$training_metrics@metrics$thresholds_and_metric_scores$fpr,
  model_tanh@model$training_metrics@metrics$thresholds_and_metric_scores$tpr,
  type = 'l', col='red')
lines(
  model_maxout@model$training_metrics@metrics$thresholds_and_metric_scores$fpr,
  model_maxout@model$training_metrics@metrics$thresholds_and_metric_scores$tpr,
  type = 'l', col='green')
legend(0.6, 0.4, legend = c("ReLU", "Tanh", "Maxout"), col = c("blue", "red","green"),
  lty = c(1, 1, 1), ncol = 1)
```

A.4.4 Radial Basis Function Network

The source code developed to implement the radial basis function network algorithm is shown in the following listing:

Algorithm A.8 Radial basis function network

```
rm(list = ls())

library(caret)
library(RSNNS)

set.seed(49237)

d <-read.csv(file = "../Data/5182.csv",header = TRUE, stringsAsFactors = TRUE)
datalength <- length(d)
xdata = d[, 1:(datalength - 2)]
ydata = d[, datalength]

td <-read.csv(file = "../Data/test.csv", header = TRUE, stringsAsFactors = TRUE)
xtest <- td[, 1:(datalength - 2)]
ytest <- td[, datalength]

# Cross-validation variables
k = 10 # 10-fold cross-validation
num_repeats = 5 # Repeats the cross-validation 5 times

completeSummary  <- function(data, lev = NULL, model = NULL) {
  a1 <- defaultSummary(data, lev, model)
  a1
}

# Setup for cross validation
trCtrl <- trainControl(method = "repeatedcv", number = k, repeats = num_repeats,
                       summaryFunction = defaultSummary, classProbs = TRUE,
                         savePred = TRUE, returnResamp = "final")
negativeThreshold <- seq(from = 0, to = 1, by = 0.01)
rbf <- caret::train(x = xdata, y = ydata, method = "rbfDDA", tuneGrid = expand.grid(
                                .negativeThreshold = negativeThreshold),
                                  prox = TRUE, allowParallel = TRUE,
                                  preProc = c("center", "scale"),
                                  metric = "logLoss",
                                  trControl = trCtrl,
                                  family = 'binomial')
plot(rbf)
```

A.5 Grid Search Optimisation

The source code shown in the following listing implements a grid search:

Algorithm A.9 Grid search using a multi-layer perceptron

```
rm(list = ls())

library(h2o)

set.seed(49237)

d <-
  read.csv(file = "../Data/5182.csv",
           header = TRUE,
           stringsAsFactors = TRUE)
datalength <- length(d)
xdata = d[, 1:(datalength - 2)]
ydata = d[, (datalength - 1)]

td <-
  read.csv(file = "../Data/test.csv",
           header = TRUE,
           stringsAsFactors = TRUE)
xtest <- td[, 1:(datalength - 2)]
ytest <- td[, (datalength - 1)]

hyper_params <- list(
  activation = "Maxout",
  hidden = seq(from = 8, to = 10, by = 1),
  balance_classes = TRUE,
  epsilon = seq(from = 1e-9, by = 1e-9, to = 1e-4),
  l1 = seq(from = 0, to = 0.2, by = 0.01),
  l2 = seq(from = 0, to = 0.2, by = 0.01),
  rho = seq(from = 0.9, to = 0.99, by = 0.01),
  balance_classes = TRUE,
  standardise = TRUE,
  variable_importances = TRUE,
  fast_mode = TRUE
)

h2o.init()
input_data <-
  as.h2o(x = d[, c(1:(length(d) - 2), length(d) - 1)], destination_frame = "input_data")

model_grid <- h2o.grid(
  algorithm = "deeplearning",
  x = colnames(xdata),
  y = colnames(d)[length(d) - 1],
  training_frame = input_data,
  hyper_params = hyper_params
)
model_grid
```

Appendix B

Benchmark Tests

B.1 Data Exploration Algorithms

The following table shows the benchmark test results for the data exploration component of the project:

Table B.1: Benchmark results for exploration algorithms

expression	min	lq	mean	median	uq	max	neval
source("normality_test.R")	15.04504	15.59895	17.06221924	15.8122505	16.290536	102.964836	100
source("pca.R")	6.043272	6.3039375	21.54007741	6.5629395	6.910908	1472.821494	100
source("kpca.R")	66.95008	75.4342695	134.86497656	79.092618	96.7484755	835.871011	100

The number of evaluations is set at 100 so as to achieve a convergence in the metrics. The data is therefore presented as statistical values from the 100 samples per expression generated. The kernel PCA algorithm has the highest time complexity as compared to the linear PCA algorithm. This is an indication of the additional overhead from the kernel trick.

B.2 Supervised Learning Algorithms

The following table shows the benchmark results for the supervised learning algorithms implemented in this research. The times are in minutes.

Table B.2: Benchmark results for supervised learning models

expression	min	lq	mean	median	uq	max	neval
source("logreg_sample_res.R")	1.896759	1.907826	1.960281	1.913904	1.965818	2.634149	100
source("logreg_incl_type.R")	3.357058	3.365332	3.395522	3.375876	3.401593	3.677278	100
source("svm_sample_res.R")	5.180742	5.314039	5.538302	5.447466	5.635973	7.260541	100
source("svm_incl_type.R")	6.577784	6.757567	7.031634	6.876002	7.154056	9.012589	100
source("mlp_sample_res.R")	2.776238	2.82304	2.896857	2.876329	2.952716	3.709405	100
source("mlp_incl_type.R")	2.635283	2.689308	2.754709	2.740065	2.794966	2.982061	100
source("rbf_sample_res.R")	8.890423	8.981384	9.052325	9.015393	9.078025	10.30787	100
source("rbf_incl_type.R")	8.710007	8.801853	8.879227	8.830452	8.89132	9.539425	100

The radial basis function network algorithm has the highest computational load as it takes the longest to compile and return results. The support vector machine has the second highest computational complexity, while the logistic regression model is the fastest.

Appendix C

Project Management

The project management aspect of this project was accomplished using Clickup. The Clickup platform allows for brainstorming and building mind-maps, it integrates with mail clients for reminders, it has a built-in chat feature and it allows for sharing of files. The most important feature is the creation and tracking of tasks using Kanban boards and Gantt Charts, among other interfaces.

C.1 Mind-Map

The mind-map developed during the brainstorming sessions with the supervisor is shown in Figure C.1:

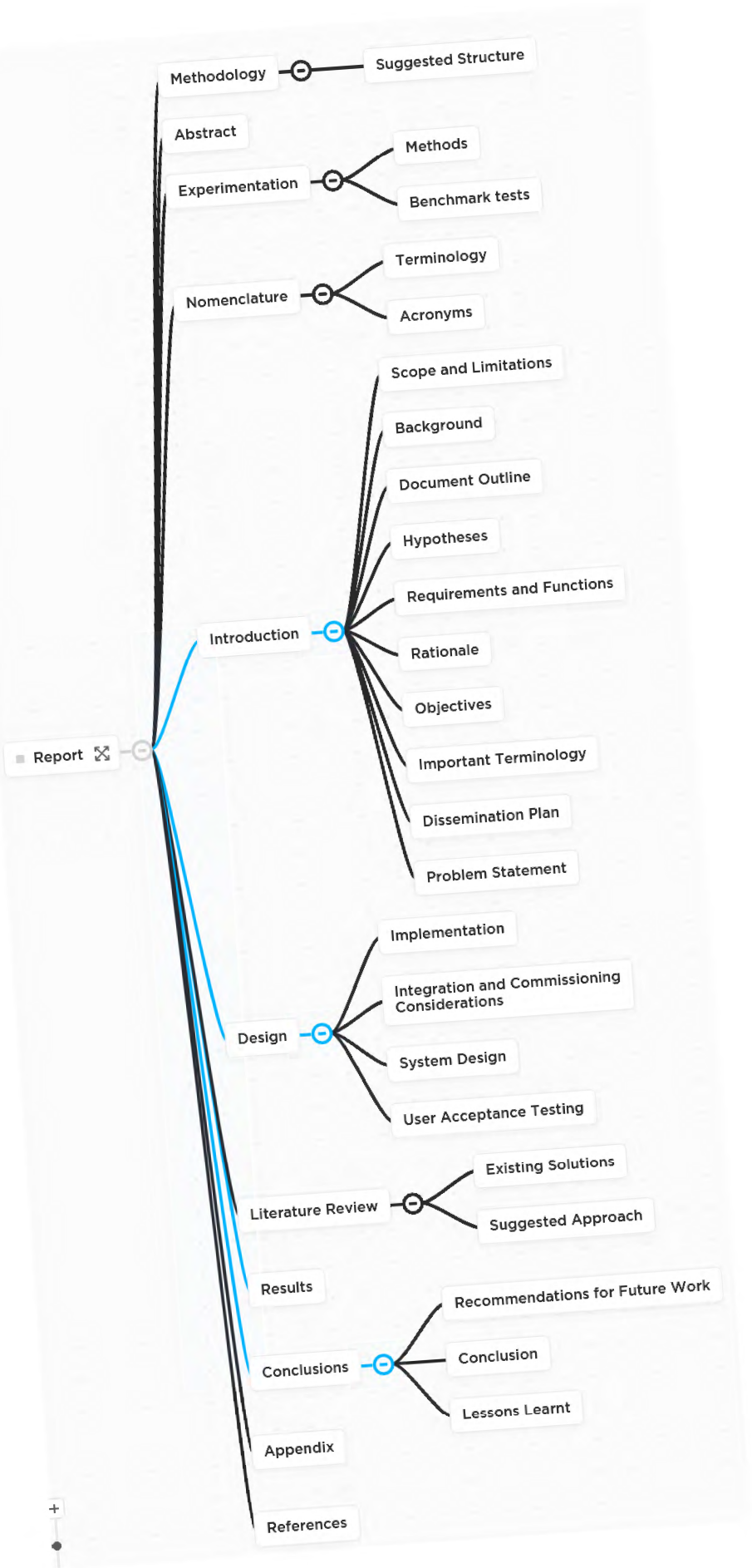

Figure C.1: Brainstorming mind-map

C.2 Time Management

The time management for the activities of the project was managed using a Gantt Chart. The Gantt Chart is shown in Figure C.2:

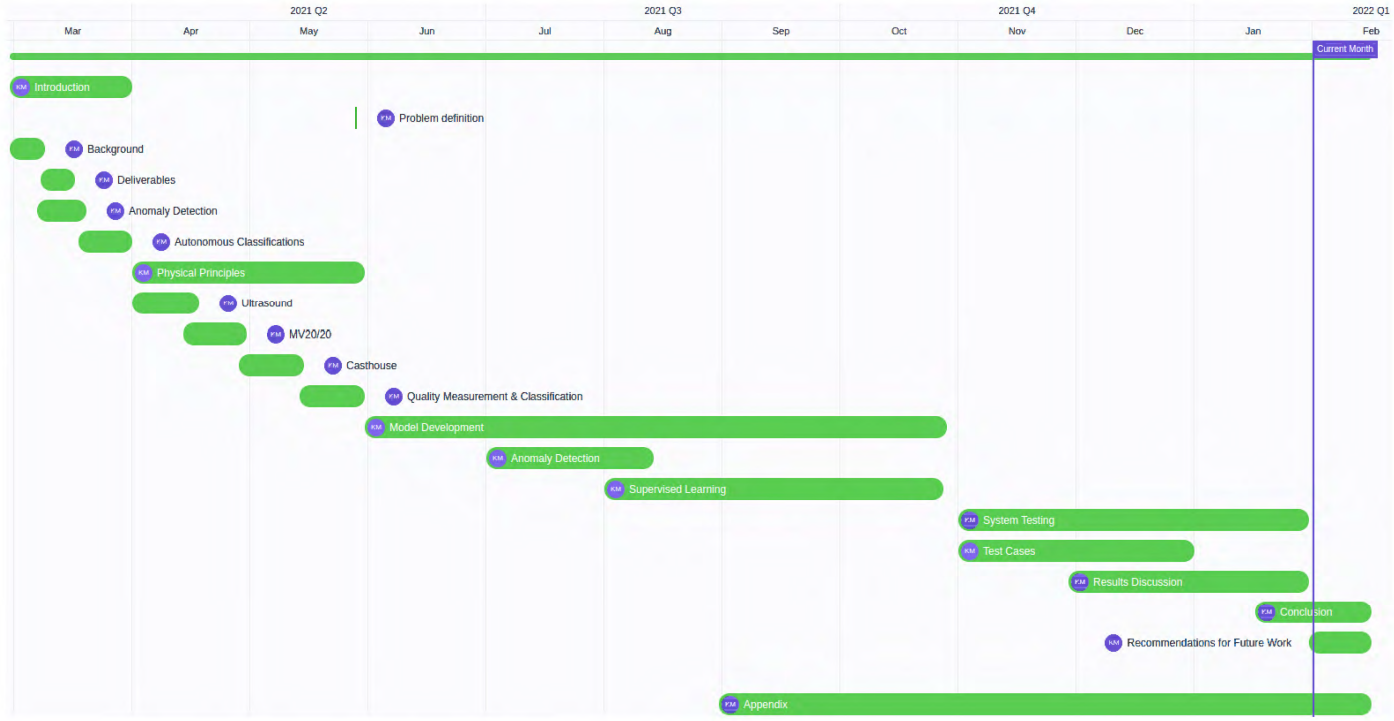

Figure C.2: Project Gantt Chart