Strathmore
UNIVERSITY

MACHINE LEARNING MODEL FOR PREDICTIVE MAINTENANCE ON LINEAR ACCELERATORS

Tonui Allan Kibet Koech

151385

Submitted in partial fulfilment of the requirements for the Degree of

Master of Science in Data Science and Analytics at Strathmore University

Strathmore Institute of Mathematical Sciences

Strathmore University

Nairobi, Kenya

June 28, 2024

ACKNOWLEDGEMENT

I would like to express my gratitude to all those who have contributed to the success of this dissertation. To my colleagues who gave their time and expertise, and to my classmates whose unending stream of advice, correction and encouragement was invaluable. To my teacher, Emmanuel Bundi, for his insights, to Dr Senagi for pushing me to do my best and to my supervisor, Dr Omondi for his guidance.

DECLARATION

I declare that this work has not been previously submitted and approved for the award of a degree by this or any other University. To the best of my knowledge and belief, the thesis contains no material previously published or written by another person except where due reference is made in the thesis itself. No part of this thesis may be reproduced without the permission of the author and Strathmore University

Candidate's Signature

Tonui Allan Kibet Koech Allan Koech **Date:** 01/ 04/ 2024

Approval

This dissertation was reviewed and approved for examination by the following:

Supervisor's Signature

Dr. Allan Omondi Digitally signed by Allan Omondi Date: 2024.04.02 05:38:54 -04'00' **Date:** 02// 04/ 2024

Research Coordinator

School of Computing and Engineering Sciences

DEDICATION

I would like to express my sincere gratitude to my family, my wife and my children for journeying with me in this academic endeavour. To my late father and grandfather for pointing the way and to my mother for her faith in me. They have been a constant source of inspiration and encouragement through it all. I pray this work will serve as an example to my children as they make their way through the world.

ABSTRACT

Predicting machine failures is the next frontier in industrial machine maintenance. However, the ideal implementation of such a program will require the fitting of industrial machines with sensors that can constantly monitor a machine's vital parameters while feeding them to a supervisory module for analysis and possible action. However, such an undertaking will require massive capital and time investments to achieve. This is where log file mining and analysis come in. By analysing the already existing machine log files of medical linear accelerators, a prediction model was developed to anticipate motor problems and notify engineers without investing the capital outlay of fitting new sensors on a machine. Mining of the log files yielded an imbalanced dataset containing 3.367% anomalies. This study tested three algorithms for their predictive power with the Random Forest classifier coming out on top with 99% precision, recall and accuracy. It was closely followed by Logistic Regression and an anomaly detector, Isolation Forest, with a precision of 59%. These strong results indicate the potential of machine learning for predicting machine breakdowns to anticipate machine failures and enable engineers to take proactive maintenance action.

Keywords: TrueBeam, Predictive maintenance, log files.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS/ACRONYMS

VMS Varian Medical Systems

OEM Original Equipment Manufacturers

PdM Predictive Maintenance

RUL Remaining Useful Life

LSTM Long Short-Term Memory

CRISP-DM Cross Industry Standard Process for Data Mining

DC Direct Current

ARIMA Autoregressive Integrated Moving Average

QA Quality Assurance

regex Regular Expressions

API Application Programming Interface

REST Representational State Transfer

Linac Linear accelerator

XGBoost Extreme Gradient Boosting Machine

SVM Support Vector Machine

MIL Multi-Instance Learning

DT Decision Tree

SPC Statistical Process Control

AUC Area Under Curve

EDA Exploratory Data Analysis

SMOTE Synthetic Minority Oversampling Technique

CSV Comma Separated Values

IF Isolation Forest

TP True Positive

TN True Negative

FN False Negative

FP False Positive

ROC Receiver Operating Characteristics

AUC Area Under the Curve

SHAP SHapley Additive exPlanations

ETL Extract, Transform and Load

ELT Extract, Load and Transform

JSON Javascript Object Notation

Chapter 1

Introduction

Original Equipment Manufacturers (OEM) owe much of their success to the reliability of their products. Scheduled preventive maintenance is the chief method they employ to ensure equipment up-time and avoid unexpected breakdowns (Sipoš et al., 2014). Preventive maintenance is calendar-based, involves some data collection and is focused on diagnostics (Achouch et al., 2022). This approach however is labour-intensive, time-consuming and poor at identifying issues that may occur between maintenance periods thus engineers fall back to a reactive maintenance approach where they let the equipment run to failure before diagnosis and repair or replacement of a component (Achouch et al., 2022).

A third modern approach is Predictive Maintenance (PdM). The European Standard defines PdM as "condition-based maintenance carried out following a forecast derived from the analysis and evaluation of significant parameters of the degradation of the item" (Standard, 2001).

PdM problems can be viewed in the following two ways (Lei et al., 2016):

1. Detecting that a monitored machine has entered a faulty state and predicting an imminent failure (diagnostic).

2. Predicting the Remaining Useful Life of the machine or a part of the machine (prognostic)

PdM has been shown to significantly improve equipment performance by allowing

engineers to conduct proactive rather than reactive maintenance on equipment (Calabrese et al., 2020). This usually involves collecting some data and measurements, calibrating some values or systems and even proactive replacement of some components whose useful lifetime can be deduced from past breakdowns.

PdM methods can be deployed by incorporating sensors, to monitor vital equipment parameters like voltage, vibration etc. then using this data with intelligent decision-making techniques like machine learning to draw insights, make predictions and support decision-making (Fernandes et al., 2019). This approach, however, is unsuitable for already installed equipment since it will require both hardware and software upgrades that will have additional unforeseen costs and there may be regulatory compliance issues to grapple with (Sipoš et al., 2014).

Modern equipment is often run by software applications. Software is designed to generate operation logs that capture valuable events from calibrations to internal states and error messages (Sipoš et al., 2014). A more realistic and expedient way to implement PdM on already installed equipment would be to data-mine these log files, identify important equipment parameters and build machine learning models that can find operating patterns and thereby predict the state of the equipment (Calabrese et al., 2020).

Linear accelerator (Linac) are versatile devices used in various fields, including medicine and industry (Hanna, 2012). They are capable of accelerating charged particles to high energies, making them suitable for a range of applications. In the medical field, Linacs are used for radiation therapy (Greene and Williams, 2017).

Medical Linacs, are devices that produce and direct high energy x-rays or electron beams that are used to target and destroy cancer cells while sparing surrounding normal tissue, this treatment is known as radiotherapy
(of North America (RSNA) and of Radiology (ACR), n.d.). Modern Linacs consist of electrical, electronic, mechanical, robotic and software components constantly interacting with each other while controlling and monitoring the output of the accelerator and its

auxiliary systems. Despite many advancements in the manufacture of linear accelerators, machine failure still occurs regularly (Hoisak et al., 2021).

Varian Medical Systems (VMS) is an American radiation oncology treatments and software maker based in Palo Alto, California, (Data, n.d.). One of their most popular linear accelerators is the TrueBeam. This machine collects and compiles events, debug information and errors into a daily combined log and sends them to a central VMS server daily. The logs contain performance data on the various electrical, electronic and mechanical parts that make up the machine. Currently, this data is not exploited and this research aims to use them to develop a model that can predict machine performance and enable engineers to undertake proactive maintenance on the machine.

For the patient, hospital, and vendor, the uptime of this machine is vital to deliver treatments, keep up with the hospital patient load with minimal to no interruptions and minimise service interventions and costs associated with the machine (Hoisak et al., 2021). Currently, service engineers are flying blind on the maintenance of this machine or follow a rigid service schedule that is not optimised for each machine and thus increases cost in terms of labour and parts used and increases the chances of machine failure leading to unplanned downtime.

One solution is to develop a predictive maintenance model that can anticipate and allow for proactive correction to reduce equipment downtime for the machine. This is quite ambitious seeing as the machine has over 800 interlocks that can prevent treatment. This study will concentrate on understanding physical motion axes. The machine has several axes; collimator rotation, jaw motions, gantry rotation, couch motions and robotic arms. The log files from TrueBeam version 4.0 include motion data like motor current, distance moved and the time it took to move. This data can be mined from the log files and used to create a model of normal operations for each axis which can then be deployed and compared against a running machine to predict any issues like greasing needs or motor/ brush changes.

1.1 Problem Statement

The Linac, just like every other manufacturer's equipment, faces wear and tear of parts due to age and use (Able et al., 2016, Kawahara et al., 2020). This could result in major equipment breakdowns due to component failure leading to negative biological effects on the patients who miss treatment (Kawahara et al., 2020) as well as financial loss (Kamat and Sugandhi, 2020). Early detection and prediction of such a failure can enable proactive correction measures to be taken in time to avoid unnecessary downtime (Lei et al., 2016).

1.2 Research Objectives

1. To identify what data the machine logs contain.
2. To find the feasibility of mining machine debug log files for features that can accurately predict machine breakdowns.
3. To generate a training dataset by using machine log files and service records to identify machines with motor problems.
4. To develop a prediction model to anticipate motor problems.
5. To present the results of the predictive model for action by service engineers.

1.3 Research Questions

1. What information do the machine log records hold?
2. Can the log records be mined to yield features that can accurately predict machine breakdowns?
3. How can the large volume of noise in the logs be systematically filtered out?

4. How can a predictive maintenance model be developed using data mined from the machine debug log files?

5. How can the developed model be deployed?

1.4 Justification of the Research

Unpredictable downtime and subsequent emergency service on machines are inevitable. This research aims to reduce the mean time to repair and the cost associated with servicing by developing a service prediction model to anticipate machine failures to enable engineers to apply preventative measures on the machine.

1.5 Research Scope and Limitations

The overall objective is to develop a predictive maintenance model by mining machine logs, that can anticipate and allow for proactive correction to reduce equipment downtime for the over 5,000 TrueBeam machines in the world. This is quite ambitious, as the machine has over 800 interlocks that can prevent treatment. This study will instead focus on developing a working model for the collimator rotation motor that can then be used to detect anomalies and inform decisions like motor changes or greasing needs.

This study is limited to the motor motion axes of the TrueBeam Linac running software version 4.0.

Chapter 2

Literature Review

(Ma et al., 2022), looked at this problem from the medical physics (application of physics to medicine (Brown et al., 2017)) perspective. They gathered daily quality check records for the linear accelerator for 3 years and then developed a stacked Long Short-Term Memory (LSTM) model. It is common practice to test linear accelerators for Quality Assurance (QA) daily to guarantee the accurate and safe delivery of radiotherapy treatments. These checks are usually performed by the site physicist before beginning the day's operations. The daily quality records were collected and used to predict performance for the next 5 days. An Autoregressive Integrated Moving Average (ARIMA) was developed on the same data set to compare against the stacked LSTM. this approach showed a lot of promise, however, they say nothing about predicting machine failure, they are just observing the performance trend.

(Hamaide et al., 2022), looked at this problem in two parts: failure detection and timing of the maintenance decision. This approach is interesting since many other methods so far have focused on detecting anomalies and reporting them leaving the decision of when to intervene entirely to the service engineers. This is probably the best approach as the service engineer has the training and experience to make intuitive decisions that will work out best. However, the goal is to use data to drive such decisions. This paper also introduces the concept of the remaining useful life Remaining Useful Life (RUL) of a spare part or the machine. This is a useful concept as it seeks to utilise a part or machine

right to the end of life rather than replacing it too early and thereby not fully utilising it or replacing it too late and encountering a debilitating breakdown. This introduces an important concept of optimisation to our research which can be used to predict or at least estimate cost savings in terms of spare parts used.

(Calabrese et al., 2020) framed the problem as a time series issue and applied an ensemble of classification algorithms like random forest and gradient boosting to log data to predict the remaining useful time of the main bearing of woodworking machines. This method achieved accuracy, recall, and precision of 98.9%, 99.6%, and 99.1%.

(Prytz et al., 2015) predicted the need for air compressor repairs and replacement in commercial lorries and buses by applying a random forest classifier and a bespoke feature selection method to a data set built from onboard vehicle logs and service records from certified workshops. This data was then fed to a binary classification model for failure prediction. Despite these data sources not having been designed for data mining, this method outperformed human experts in predicting the RUL of the air compressor.

2.1 Theoretical Framework

Machine learning is the ability of computers to learn without being explicitly programmed (SamuelA, 1959). In other words, machine learning is teaching a computer to identify patterns or inferences either implicit or explicit from data (Dahiphale et al., 2023). The patterns discerned can then be used to predict the output for new data. This is then called a machine learning model.

Traditionally computers have always been given literal step-by-step instructions on how to carry out a task however, our computing needs are increasing and we now want them to mimic human intelligence. For example, it is easy for human beings to recognise faces, photographs, sounds etc but giving a computer step-by-step instructions on how to perform these tasks is almost impossible as we do not yet know how we can perform these

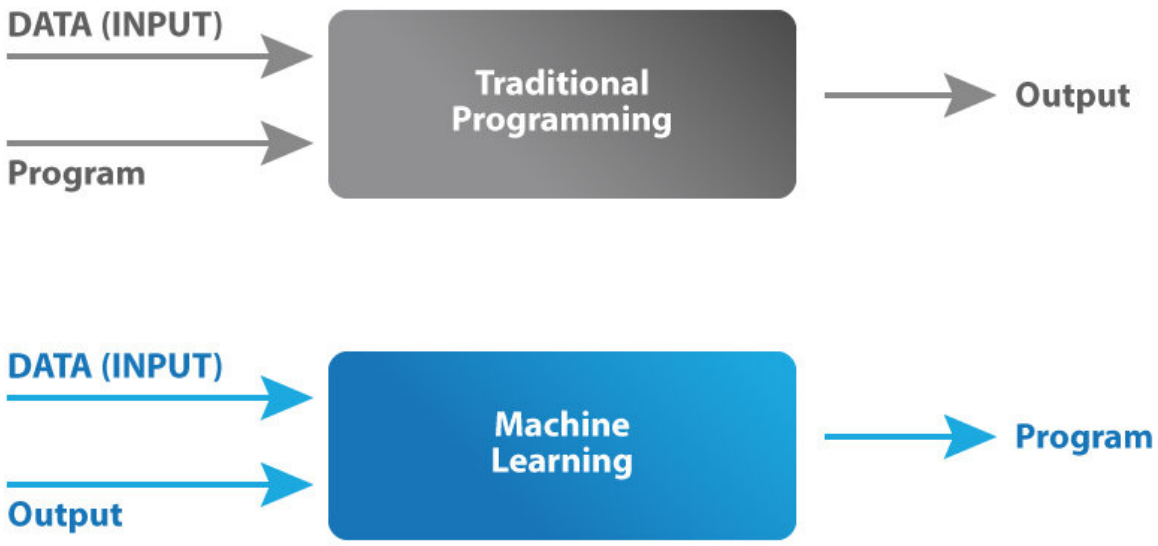


Figure 2.1: Traditional Programming vs. Machine Learning

tasks ourselves. Therefore we have resorted to feeding a machine learning algorithm with the relevant data and let it fend for itself. This strategy has gotten us surprisingly good results with machine learning algorithms being deployed to solve several human problems.

Figure 2.1 illustrates the difference between traditional programming and machine learning. Traditional programming and machine learning.

In the case of designing a predictive maintenance system, a common approach is to design an expert-based system. Here, domain experts can program a computer model or develop mathematical equations based on specific parameters that they understand i.e. use the traditional programming approach and give step-by-step instructions to the computer on how to approach the problem just like they would (Sipoš et al., 2014). This is usually done on a case-by-case basis with a predictor for each specific failure or warning/ alarm. While such systems are highly successful, they are also expensive, slow and time-consuming to design, test and deploy and thus not practical for industrial-scale systems.

According to (Ayankoso and Olejnik, 2023), there are three ways to model a dynamic

system to enable predictive maintenance:

1. **Physics based:** This approach relies on the laws of physics to come up with equations that describe the system's behaviour. These equations can then be used to predict future states. A big advantage of this method is that it forms explainable models which engineers generally prefer and may be critical in some industries. However, it may be expensive, slow and time-consuming to implement.

2. **Data driven:** This method uses real-world data and powerful algorithms to discern patterns and make predictions without explicitly building a physical model. Deep learning methods are especially popular in science and engineering due to their capability to model non-linear complex systems. This method is becoming more popular with the advent of artificial intelligence and increased computing power. However, the final model will probably be a black box which may be a problem for some systems that due to their nature or regulation have to have an explainable model.

3. **Hybrid:** This approach combines the best of both physics and data-driven models to create more accurate and nuanced models.

2.1.1 Decision Theory

Building a predictive model is one thing, we also need to decide how to decide when to perform predictive maintenance or trigger an alarm. A simple method is to establish an operating baseline and to trigger an alarm whenever this baseline is exceeded like (Able et al., 2016) did using an I/MR chart. This is a basic method as it cannot point to any specific issue, it will just inform the users that a threshold has been exceeded and investigation into the issue is required.

(Calabrese et al., 2020) designed a target variable expressed as Remaining Useful Life RUL and estimated it by predicting the remaining number of days to the failure event as

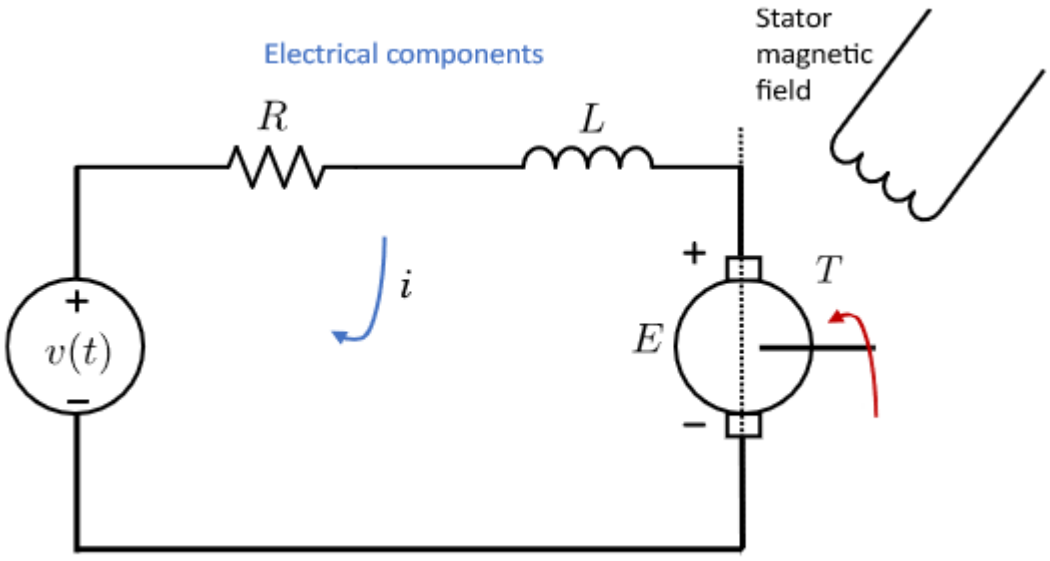


Figure 2.2: DC Motor Schematic

provided by the internal service records. The RUL was set up as a binary variable and tested for several window periods (30, 20, and 10 days) before a machine-down event.

(Hamaide et al., 2022) designed a two-level system for PdM: the first system is a health indicator built by aggregating features using a learning algorithm. The second system then decides if and when to trigger an alarm based on the health indicator.

2.1.2 DC Motor Theory

A DC motor is a device that converts electrical energy to mechanical power (Corporation, 1977). DC motors are widely employed in numerous industrial applications due to their efficient and controllable operation. Understanding the fundamental principles and characteristics of DC motors is essential for this research as it will involve developing a predictive model for some of the DC motors in the TrueBeam. Figure 2.2 shows a DC motor electrical model.

- **R:** Armature winding resistance
- **L:** Armature winding inductance

- **T:** Torque produced
- **i:** Armature current
- **E:** Back electromotive force
- $v(t)$**:** Applied DC voltage

The working principle of a DC motor is based on the fact that when a current-carrying conductor is placed in a magnetic field, the conductor experiences a mechanical force proportional to the current (Fleming, 1892). This force produces the torque (T) which rotates the motor.

DC motors come in various types, each designed for specific applications. Permanent Magnet DC (PMDC) motors utilize fixed magnets to create a magnetic field, while Brushless DC (BLDC) motors eliminate the need for brushes, enhancing reliability (Mohan et al., 2003). Separately Excited DC motors and Series DC motors represent other categories, each offering unique advantages depending on the application (Hughes and Drury, 2013).

DC Motor Features

To understand and control DC motors three characteristic curves need to be understood.

1. **Speed Vs. Torque Curve**

   This feature is illustrated through a torque-speed curve as shown in figure 2.3, depicting a downward slope to the right with torque plotted on the horizontal axis and speed on the vertical axis. The speed is at its maximum under no load conditions but decreases progressively to the right until it reaches maximum torque at zero speed ("How are DC motors controlled? - Speed control of DC motors — ASPINA", 2021).

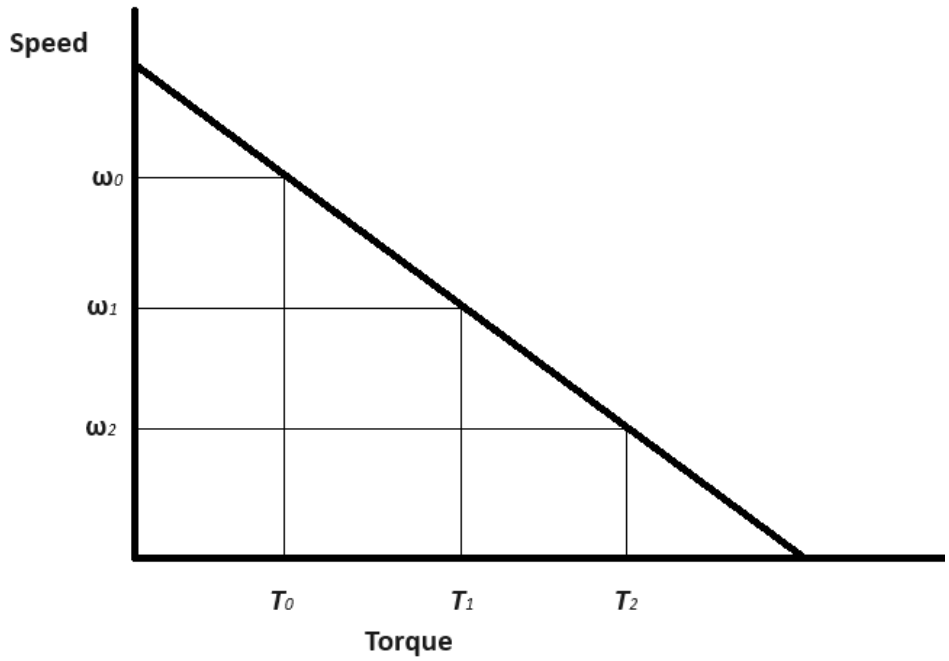


Figure 2.3: Speed Vs. Torque

The torque-speed curve indicates how torque and speed change with varying loads. Consider a motor initially rotating at speed ω_0 with torque T_0, as shown in the graph below. If the load torque increases to T_1, the motor speed will consequently decrease to the new speed ω_1. Similarly, if the load torque further increases to T_2, the speed will fall to ω_2.

Therefore, what occurs to the torque-speed curve when the voltage powering the DC motor is adjusted? Figure 2.4 illustrates torque-speed curves for various voltages. When the drive voltage is doubled, both the no-load motor speed and the starting torque (torque when the motor is held in position) also double. In simple terms, elevating the voltage causes the torque-speed curve to shift upward in a parallel manner. It is possible to modify the torque-speed curve for a DC motor as desired by altering the applied voltage (P. S. Bimbhra, n.d.).

2. **Current Vs. Torque** When we examine the connection between torque and current, we find that they have a proportional relationship with each other. For a

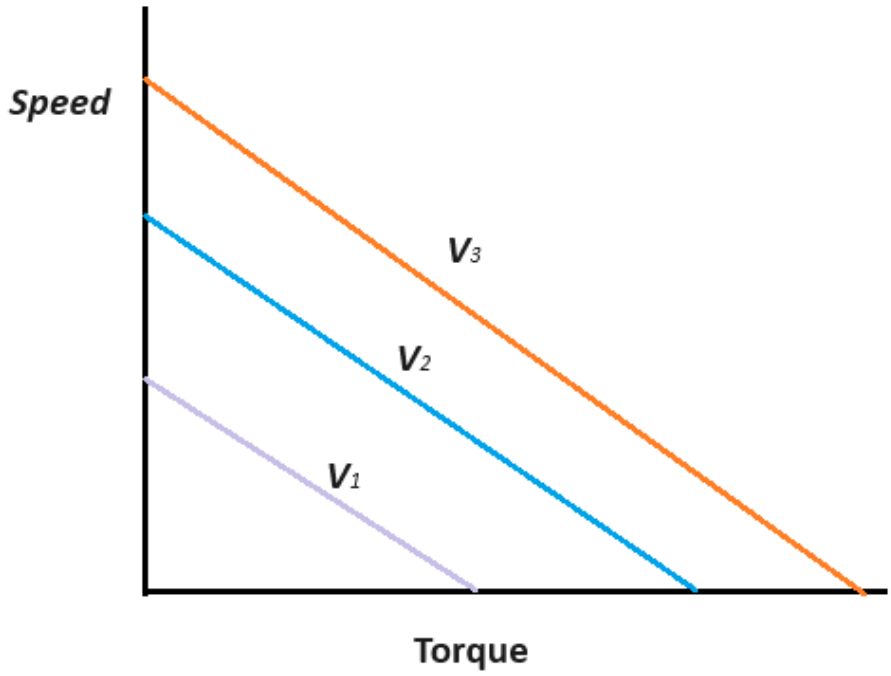


Figure 2.4: Motor Voltage and Torque

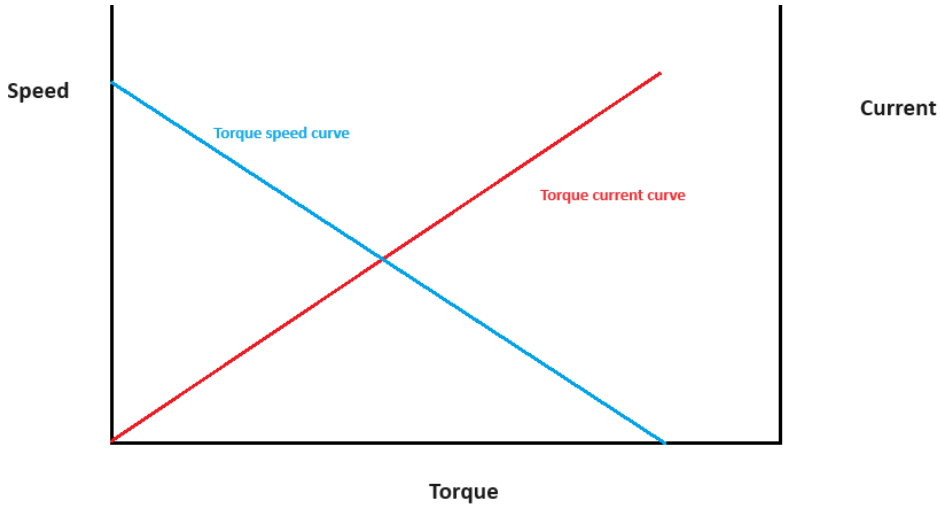


Figure 2.5: Current Vs. Torque

motor, the ratio between these two remains constant, even when the motor speed or drive voltage changes. Therefore, measuring the motor current alone can help you determine the motor torque (Richardson, 1982).

The DC motor has several unique characteristics that make it well-suited for several applications:

1. It has a high starting torque.
2. It is easy to control i.e. has a fast response to starting, stopping and acceleration commands.
3. Simple installation and maintenance.

2.2 Existing Algorithms

(Calabrese et al., 2020) used the H2O.ai ("H2O.ai — The fastest, most accurate AI Cloud Platform", n.d.) python package to train and test nine different machine learning models. The models included: Extreme Gradient Boosting Machine (XGBoost) ("XGBoost Documentation — xgboost 2.0.3 documentation", n.d.), Distributed Random Forest ("Distributed Random Forest (DRF) — H2O 3.44.0.3 documentation", n.d.), gradient Boosting Machine (GBM) (Natekin and Knoll, 2013), Support Vector Machine (SVM) with linear and Gaussian Kernel ("1.4. Support vector machines", n.d.), nearest-neighbour [NN] classifier ("1.6. Nearest neighbors", n.d.) and Decision Tree (DT). They found that XGBoost, DRF and GBM achieved the highest performance (as measured by accuracy, precision and recall) than SVM and DT ("1.10. Decision Trees", n.d.). They did not consider neural-based models because they may have limited explainability, unlike tree-based models that operate on rules that may be useful for the users in the diagnosis and prognosis while performing PdM.

(Sipoš et al., 2014) used Multi-Instance Learning (MIL) (Babenko, 2008) to design a PdM model. In MIL, the learning process differs from standard classification. Instead of receiving a set of independently labelled instances, the learner is provided with bags, each labelled as positive or negative. These bags may encompass multiple instances, such as equipment daily logs. A bag is considered negative if all instances within it are negative (indicating no failures based on service notifications) and positive if it contains at least one positive instance. The objective, given bags from various equipment and dates, is to construct a classifier that accurately labels unseen bags or instances. In this context, model accuracy is assessed at the bag level. This then is a classification problem that seeks to identify bags that indicate a problem may be in the offing.

(Cédola et al., 2021) developed their PdM model using LightGBM ("Welcome to LightGBM's documentation! — LightGBM 4.0.0 documentation", n.d.) as a binary classifier to predict the RUL of an industrial bleaching machine. The model had an accuracy of 0.962, Area Under the Curve (AUC) of 0.987, precision of 0.948 and recall of 0.918.

2.3 Machine Learning Development Frameworks

Machine learning (ML) frameworks are critical tools that facilitate the development, deployment, and scaling of machine learning models. The evolution of these frameworks has played a pivotal role in advancing the field of machine learning. The right framework is crucial for efficient model development, training, and deployment. This review delves into popular frameworks, analyzing their strengths, weaknesses, and suitability for various tasks.

An ML framework provides a comprehensive environment for building, training, and deploying ML models. It offers tools for:

- Data preprocessing and manipulation
- Model training and evaluation

- Hyperparameter tuning
- Deployment and serving

2.3.1 TensorFlow

TensorFlow is an open-source, flexible and scalable software library for numerical computations using data-flow graphs (Pang et al., 2019). It comprises a software library and associated tools that can be used to program, train and deploy machine learning models (Abadi et al., 2016). It was released in November 2015 by Google. TensorFlow can be used for the entire machine learning project, from data preparation, model development, deployment and even operations monitoring (TensorFlow, n.d.), but its speciality is deep neural networks. From the literature review, no predictive maintenance model has been developed using TensorFlow despite its popularity and rich ecosystem. Since this research will involve data mining patterns from massive log files, TensorFlow may prove to be invaluable.

2.3.2 PyTorch

PyTorch, developed by Meta AI, is a free and open-source library of Python programs for building deep learning tools and systems (Pietro Giovanni Antiga et al., 2020). It has gained popularity for its dynamic computation graph and user-friendly interface and its dynamic computational graph allows for easy debugging and experimentation. Researchers appreciate PyTorch's intuitive design for prototyping complex models, making it a preferred choice for academic and research-oriented tasks (Paszke et al., 2019). It is generally used in computer vision and natural language processing applications.

2.3.3 Scikit-Learn

Scikit-Learn, a versatile and easy-to-use ML library in Python, is known for its simple and consistent API. It is focused on traditional ML algorithms like linear regression, decision trees, and clustering. It provides a wide array of tools for data preprocessing, model selection, and evaluation (**Pedregosa-et-all-2012**). Scikit-Learn is often the framework of choice for practitioners entering the field due to its straightforward implementation of various algorithms, however, it may not be ideal for complex deep-learning tasks.

2.3.4 Keras

Initially developed as a high-level neural network API, Keras has evolved into a comprehensive ML framework. It is now integrated with TensorFlow and PyTorch providing a user-friendly interface for building and training deep learning models (Team, n.d.). Keras simplifies the development of complex neural network architectures by offering a simple, expressive, and modular API that lowers the barrier to entry for developers and researchers alike. Before Keras, deep learning frameworks were complex and challenging to learn and use, often requiring expertise in low-level code and intricate mathematical concepts. Keras is often applied in computer vision problems, natural language processing, generative models and scientific research.

2.3.5 Apache Spark

Is a multi-language engine for executing data engineering, data science, and machine learning on single-node machines or clusters (“ Apache Spark™ - Unified Engine for large-scale data analytics”, n.d.). Conventional data processing tools often struggle to keep pace with the exponential data volume and complexity growth. Spark addresses this challenge through its distributed processing framework, leveraging clusters of commodity hardware to parallelize compute tasks across multiple nodes. This enables efficient analysis of

massive data sets while maintaining high performance and fault tolerance. Apache Spark has been deployed in a wide range of solutions including business intelligence, healthcare, finance, social media and scientific research. It is scalable, fault-tolerant, has diverse capabilities and has an active community. However, it is resource-intensive for large clusters and can be intimidating for beginners.

Each framework has its pros and cons with suitable applications for each. Table 2.1 summarises the strengths and weaknesses of the reviewed frameworks:

Framework	**Strengths**	**Weaknesses**
TensorFlow	Powerful, flexible, large community	Steep learning curve, complex for simple tasks
PyTorch	Dynamic, easy to experiment, good for research	Can be slower than TensorFlow
Scikit-learn	Easy to use, well-documented, diverse algorithms	Not ideal for deep learning or complex tasks
Keras	Simple API for deep learning, built on TensorFlow	Limited low-level control

Table 2.1: Summary of Machine Learning Frameworks

2.4 Empirical Framework

(Hoisak et al., 2021) conducted a long-term longitudinal analysis of data from a custom-built machine data collector. The authors of this study developed an external logging system that would enable the machine operators to record parameters like beam energy, machine up-time, and fault events. This user-generated data combined with service report data from field service engineers working on the machine was used to gain operational insights into the machines.

Their key findings include:

- **Identifying frequent fault events:** The analysis revealed the most common types of faults encountered during operation, allowing for targeted maintenance

and preventative measures.

- **Understanding factors affecting beam energy:** The study identified correlations between specific system parameters and variations in beam energy, enabling optimization strategies for consistent output.

- **Investigating machine downtime:** By analyzing downtime trends and associated fault events, the authors identified opportunities for improving system reliability and reducing downtime.

This research was able to identify several trends and operational insights into machine performance despite utilising secondary self-reported data. This is a basis for taking this concept a step further by utilising the native machine log data that will be richer and far more accurate than human self-reported data.

(Able et al., 2016), proposed delivering a custom daily quality assurance (QA) treatment and automatically transferring the resulting log files to an analysis system. The system then subjects daily operating and performance values to statistical process control (SPC) analysis. Specifically, they use an Individuals and Moving Range (I/MR) (Tupkar et al., 2020) chart to monitor and detect any deviations. Once baselines are established, all that remains is determining whether any drifts from the established baseline have occurred and if they are enough to trigger alarms and alert service engineers. This is the simplest method to implement as all the tools needed to implement it are available. However, this study does not result in a predictive maintenance model and the chart deviations just tell you that something may be off but not what the cause could be or how to mitigate against it. Such a system may be very useful for the clinical operators but for the service engineer, not so much.

Expert knowledge has long guided predictive maintenance, with specialists manually identifying event code patterns signalling component failures. This approach, while time-intensive, showcases a breakthrough: log data holds predictive power for anticipating

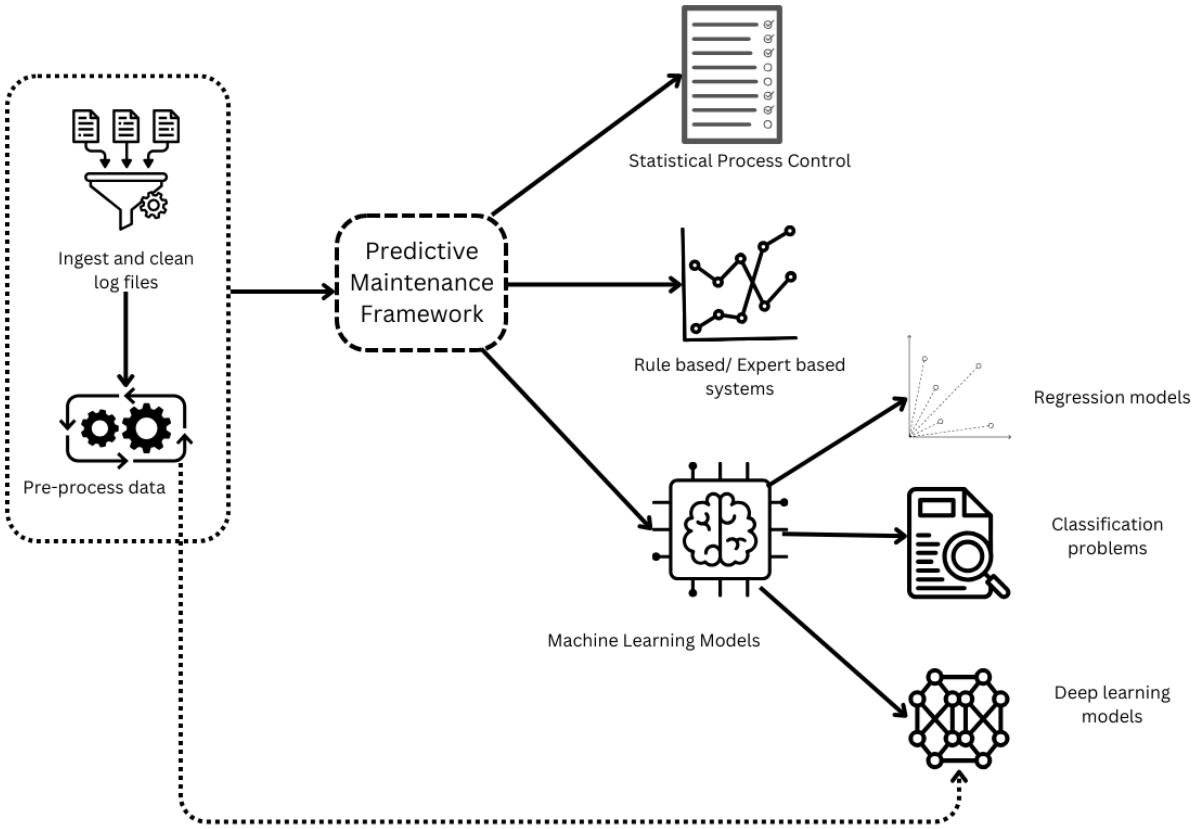


Figure 2.6: Predictive Maintenance Conceptual Framework

equipment issues. Embracing this principle, this study aims to automate and streamline this process for enhanced efficiency.

2.5 Conceptual Framework

From the literature review, a conceptual framework of PdM is illustrated in figure 2.6.

To conduct PdM using generated machine log files, one has to obtain the machine logs, clean then preprocess them. PdM fall into three broad categories:

1. Statistical Process Control (SPC) (Able et al., 2016)

2. Rule-based/Expert-based models (Kamat and Sugandhi, 2020, Ayankoso and Olejnik, 2023, Sipoš et al., 2014, Fernandes et al., 2019)

3. Machine learning models

Machine learning models can be further divided into three categories:

1. Regression models (Hamaide et al., 2022).

2. Classification problems (Calabrese et al., 2020, Sipoš et al., 2014)

3. Deep learning models. Some deep-learning models will perform the feature extraction as well as modelling (illustrated by the dashed arrow) (Ayankoso and Olejnik, 2023).

Chapter 3

Methodology

3.1 Research Design

This research study was conducted using the CRISP-DM methodology as illustrated in figure 3.1 (Chapman, 2000). This is the industry standard for conducting data science projects and studies.

The six steps of this methodology are:

1. Business Understanding - This involves defining the problem and objectives of the project which in this case is to develop a PdM model that can detect DC motor issues before they cause a machine-down situation.

2. Data Understanding - This involves gathering, examining and exploring the data required to answer the objectives. In this case, the machine log files available from the VMS servers were used.

3. Data Preparation - This involves cleaning the data, feature extraction and preparation of the dataset to ensure the quality and suitability for modelling. This is a vital stage as the quality of data directly impacts model performance.

4. Modelling - This is the core of the study. Using the data from the previous step, the right model is picked depending on how the problem is framed i.e. a cluster-

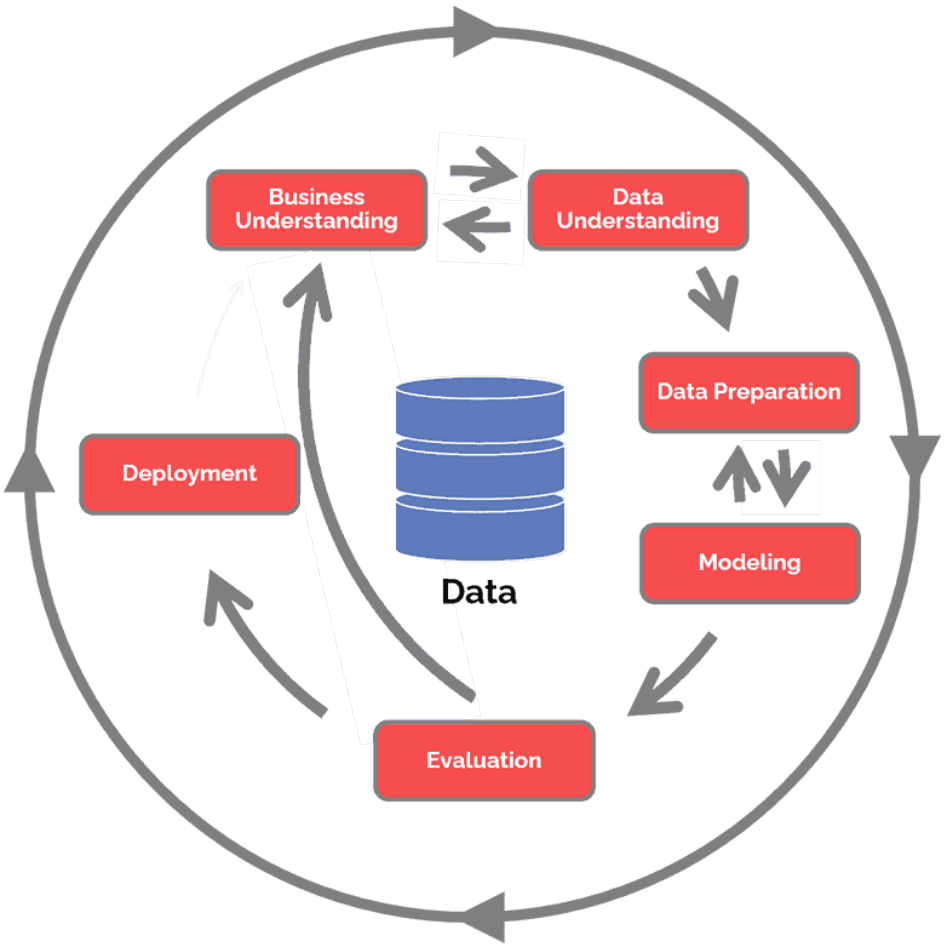


Figure 3.1: CRISP-DM Methodology

ing problem, classification problem, regression problem or a deep learning problem (Achouch et al., 2022). Three models were used in this study.

5. Evaluation - This involves measuring the model performance against a test dataset as well as in the real world and checking how well it works and whether it meets the objectives laid out. This step is also to ensure that the model generalises well in the real world with no biases. This study used ()3.5.4 precision, recall, accuracy and F1 Score to compare model performance.

6. Deployment - This is releasing the developed model for use in the real world by integrating it with the existing business processes for utilisation by users. This study utilised Streamlit, FastAPI and SQLite to deploy and monitor the final model.

This is an iterative process as shown in figure 3.1. The study went through several cycles of this framework to develop the final model.

3.2 Data Sources and Collection

The TrueBeam machine logs all relevant events on a local log file. This file usually contains thousands of lines and may be up to 10MB of text data. Every day, the machine sends the daily log to the VMS Splunk ("Splunk — The key to enterprise resilience", n.d.) server over the internet. This server aggregates and indexes all these log files. This is the source data for this research study.

Machine debug log files collected were from January 2024 to the end of February 2024. The log files were specifically from TrueBeam machines running version 4.0, the latest version. Machines representing a diverse geographical region and both in good working condition and those that experienced failures were selected. Machines used in clinical settings to deliver treatment and those used exclusively for research purposes were also included.

The logs were selected by running the relevant Splunk query on the Splunk search bar. By cross-referencing with machine service records as pioneered by (Sipoš et al., 2014), two datasets were sourced:

1. The first dataset was of machines in good working condition.

2. The second dataset was of machines that had exhibited motor failures.

The data was then downloaded as a Comma Separated Values (CSV) file.

3.3 Data Pre-processing

The next stage was to pre-process the data to ensure its accuracy and suitability for machine learning.

3.3.1 Data Cleaning and Inspection

The machine log file is a collection of events recorded by various applications and modules running the equipment. The structure of the raw log files consists of single line entries with features separated by commas, colons and semi-colons. Each line is delimited by time and there are no column headers.

The first step therefore was to import the log files into a Jupyter notebook and separate this data into meaningful columns by splitting the log_text column along colons(:) and semi-colons(;) by using Python's native string splitting ("Built-in types", n.d.) techniques and giving relevant column names. Next, the datasets were checked for null or missing values of which there were none.

The machine log files were found to contain the following parameters shown in table 3.1 below:

The first five columns of the log entry mostly contain identifying features. The log text entry contains the salient features useful for the model development.

Feature	Description
Timestamp	Subsystem time. The machine has 10 subsystems that each control different parts of the machine and each one logs the time it writes a log entry.
Serial_No	Serial number of the machine sending the log entry.
Node	Hardware component that controls the subsystem generating the log entry.
System	Name of the subsystem hardware component.
Log_Text	It may be structured or unstructured. It describes the event and may include an event code and severity level.

Table 3.1: Log File Features

A class column of 'Label' was then added to each dataset labeling them as normal (0) or anomaly (1). Finally, the datasets were merged to form one CSV file for further processing. The final dataset has the format shown in table 3.2. Columns *'minCurrent', 'maxCurrent', 'meanCurrent', 'stdCurrent', 'Time'* and *'Distance'* are the results of splitting the *log_text* column.

The data was then examined to understand its contents and underlying structure. This involved viewing the summary statistics (mean, mode, minimum and maximum values and quartile ranges) to detect any data quality issues that may be present.

3.3.2 Outlier Detection

Using boxplots plotted for each numerical feature, outliers were detected. However, since this study aims to detect anomalies, these were left in as they may be important anomaly data points.

3.4 Feature Engineering

This step usually entails the creation of new features from the current data to improve the performance of the machine-learning model. This can be achieved by combining some

Column	Description	Datatype
Serial_No	Serial number of the machine sending the log entry.	Integer
Timestamp	Subsystem time at log entry.	Pandas Datetime object
minCurrent	Minimum current drawn by the motor	Float
maxCurrent	Maximum current drawn by the motor	Float
meanCurrent	Mean current drawn by the motor	Float
stdCurrent	Standard deviation of current drawn by the motor	Float
Time	Time taken by the motor to move	Float
Distance	Distance moved by the motor	Float
Speed	Speed of the motor. Equation (3.1) as described in section 3.4	Float
Label	1 (Anomaly) or 0 (Normal)	Integer

Table 3.2: Summary of the Dataset

relevant features into one feature. Feature engineering can also be used to handle missing data, improve model robustness and allow for the incorporation of domain knowledge into the model.

According to figure 2.5, when a DC motor draws more current than is usual for the same or similar distance, it means it is developing more torque. Now, since the load on the motor is not dynamic (meaning it does not change) this signals either an obstruction in the motor path or the motor has developed an internal fault. Using this concept, a target variable of 'Speed' was developed using the variables 'Distance' and 'Time'.

$$Speed = \frac{Distance}{Time} \tag{3.1}$$

Operation 3.1 resulted in 184 records throwing a division by zero error and thus populating the data frame with null values that were subsequently dropped.

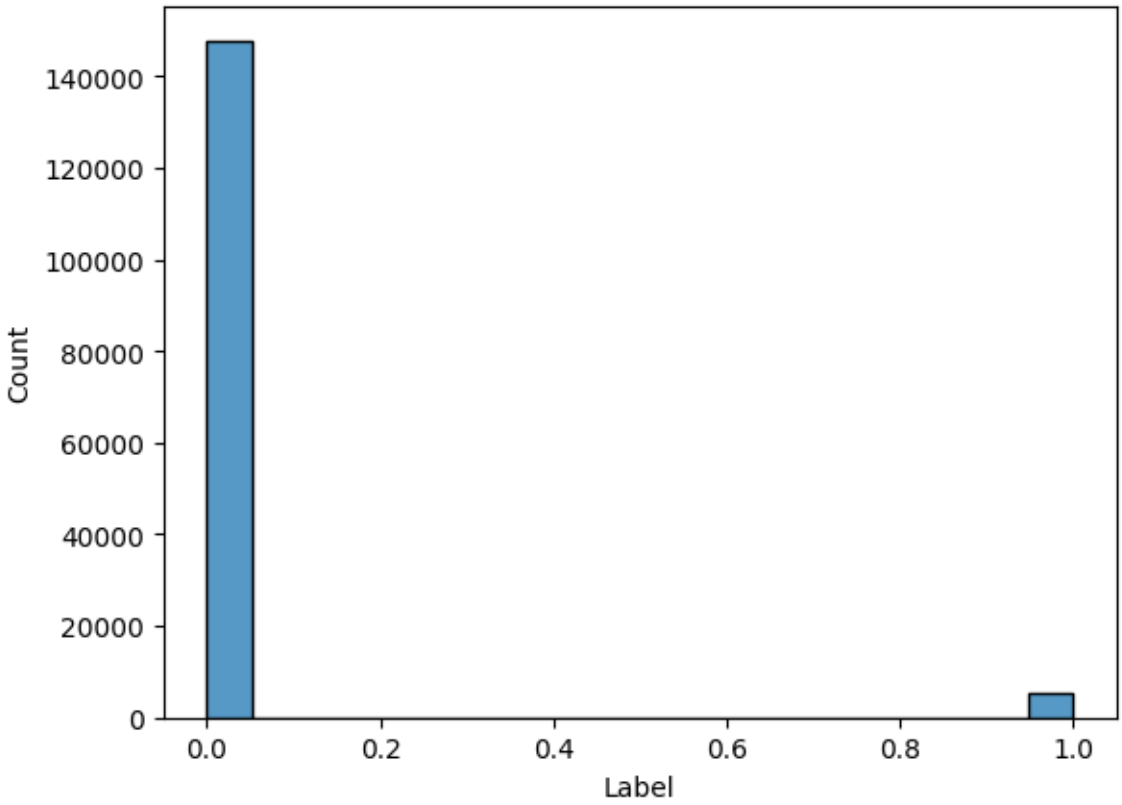


Figure 3.2: Class Imbalance

3.4.1 Data Imbalance

The dataset is highly imbalanced as shown in figure 3.2 with anomalies accounting for only 3.367% of the dataset. An imbalanced dataset is one where the distribution of target classes is not roughly equal (Chawla et al., 2002). This is quite common in real-world data where abnormality is rare as compared with normality e.g. fraud detection, disease prediction etc. This imbalance usually leads to models biased towards the majority class which is a big problem when trying to predict anomalies.

This imbalance was treated using SMOTE (Chawla et al., 2002). SMOTE addresses this by specifically creating synthetic samples for the minority class. Here is how it works:

1. **Identifying Minority Class Instances:** SMOTE first identifies the minority class instances in the dataset. These are the instances belonging to the class that has

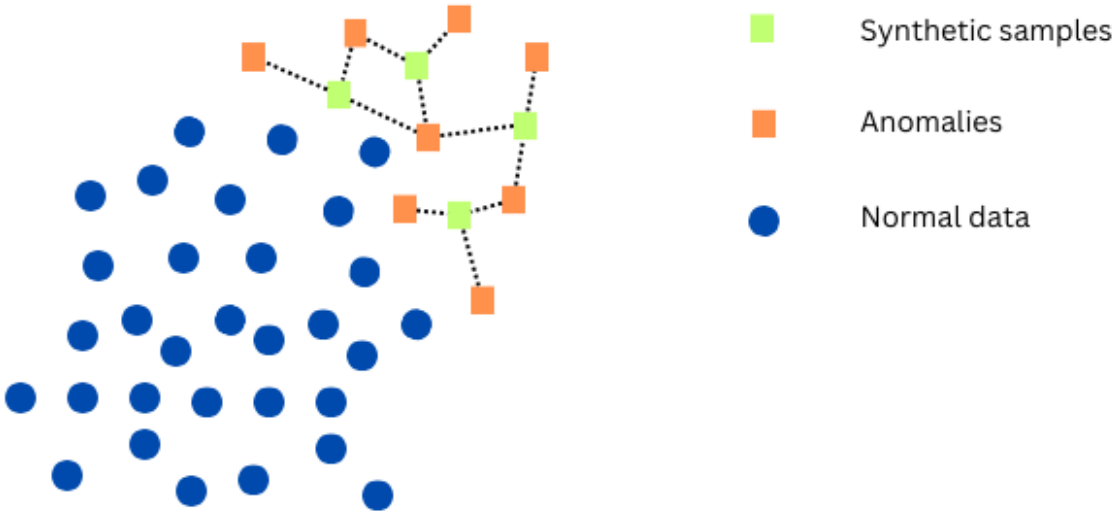


Figure 3.3: SMOTE Process

fewer samples compared to the majority class (Chawla et al., 2002).

2. **Neighbor Selection:** For each minority class instance, SMOTE selects its k-nearest neighbours. The number of neighbours to be selected can be specified by the user (Chawla et al., 2002).

3. **Synthetic Sample Generation:** For each minority class instance, SMOTE generates synthetic samples by creating new instances along the line segments joining the instance to its selected neighbours. It does this by randomly selecting a neighbour and then randomly selecting a point along the line segment connecting the minority instance and the neighbour. This synthetic sample is added to the dataset (Chawla et al., 2002) as shown in figure 3.3.

4. **Balancing the Dataset:** By repeating this process for all minority class instances, SMOTE effectively increases the number of minority class samples in the dataset, thus balancing the class distribution (Chawla et al., 2002).

SMOTE helps in mitigating the class imbalance problem by generating synthetic examples of the minority class, which can help in training a classifier that performs better

on the minority class. However, it's important to note that SMOTE does not address issues related to overlapping or noisy classes, and it may lead to over-fitting if not used judiciously. Additionally, the choice of the number of neighbours in SMOTE can have an impact on its performance and should be carefully tuned based on the specific dataset and problem at hand. In this case, this was varied from 2 to 8 with no discernible impact on the model.

3.5 Machine Learning Modeling

From the literature review, several machine learning algorithms and statistical methods were deployed in several versions of this problem. Each method has its merits and demerits depending on the overarching intention of the researcher. The model developed by this study is to predict anomalous motor data and flag them for further investigation by a service engineer.

This problem can be viewed as a classic anomaly detection problem or a traditional classification problem. Anomalies are data patterns that exhibit different characteristics compared to normal data instances (Liu et al., 2008). This study employed a semi-supervised approach using the Isolation Forest (IF) algorithm (Liu et al., 2008), a Random Forest classifier as used by (Calabrese et al., 2020 and Prytz et al., 2015) and a Logistic Regression algorithm.

3.5.1 Isolation Forest

This is an ensemble of decision trees used for anomaly detection. It works by assuming anomalies (outliers) are fundamentally different from the bulk of the data and therefore easier to isolate (Liu et al., 2008). Here is how it works:

1. **Random Partitioning:** The algorithm randomly selects a feature and a random value within the range of that feature to create a partition. This partition splits

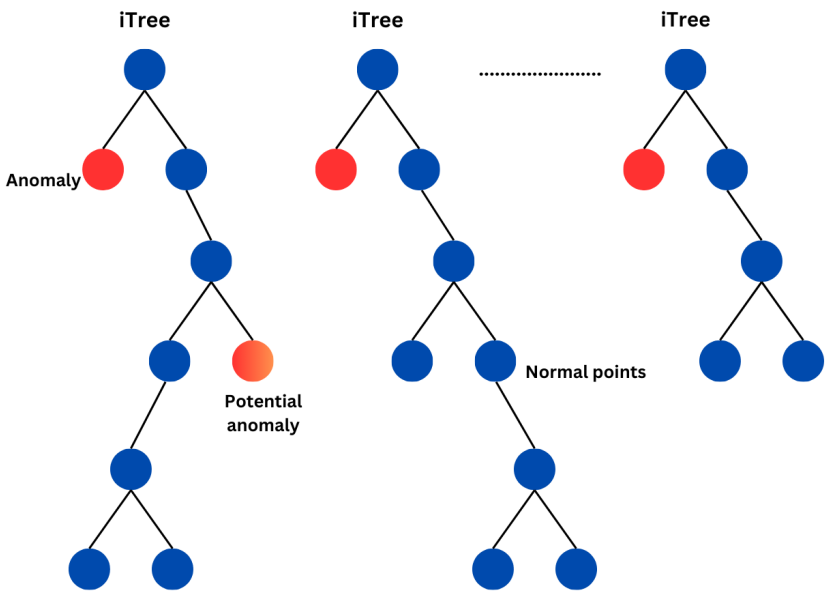


Figure 3.4: Isolation Forest

the dataset into two subsets: one subset containing data points with values below the selected value, and the other subset containing data points with values above it (Liu et al., 2008) as shown in figure 3.4.

2. **Recursive Partitioning:** The process of random partitioning is repeated recursively on each subset until all data points are isolated. This means that each partition creates a 'tree' branch until all data points are either isolated or grouped with similar points (Liu et al., 2008).

3. **Isolation:** Anomalies are expected to be isolated much faster than normal instances. Since anomalies are typically located in sparser regions of the feature space, they require fewer partitions to be isolated from the rest of the data (Liu et al., 2008).

4. **Detection:** By measuring the number of partitions required to isolate each data point, the Isolation Forest algorithm assigns an anomaly score to each point. Points requiring fewer partitions are considered more likely to be anomalies, while points that require more partitions are considered more likely to be normal instances (Liu et al., 2008).

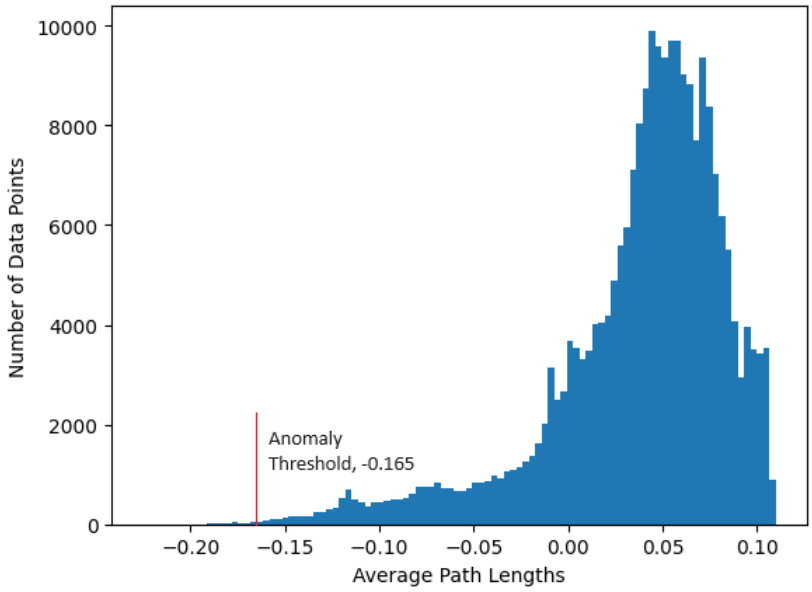


Figure 3.5: Isolation Forest Anomaly Scores

5. **Thresholding:** Finally, a threshold is applied to the anomaly scores to classify data points as either anomalies or normal instances (Liu et al., 2008) as seen in figure 3.5.

   The key idea behind Isolation Forest is that anomalies are generally fewer in number and more easily separable from normal instances in the feature space. By isolating anomalies more quickly than normal instances, the algorithm efficiently identifies them as outliers. Additionally, since it randomly selects features and thresholds for partitioning, it is less sensitive to high-dimensional data compared to other outlier detection methods. IF has a linear time complexity and is thus memory efficient. This, combined with its speed makes it suitable for large datasets (Liu et al., 2008).

   Overall, Isolation Forest is a powerful tool for anomaly detection due to its speed, efficiency, and effectiveness in identifying outliers in various types of data.

3.5.2 Random Forest

The second model used was a Random Forest Classification (Breiman, 2001) algorithm. Random forest is also an ensemble model made up of decision trees, hence the forest, used to create a robust and accurate model. It can be used on both classification and regression problems. It works as follows:

1. **Data Subsets:** Instead of using the entire training dataset, a random subset of data points (with replacement) is chosen for each tree. This technique, called bagging, helps prevent overfitting by training each tree on a slightly different view of the data.

2. **Feature Subsets:** At each split point in a tree, instead of considering all features, a random subset of features is chosen as candidates for the split. This forces the trees to learn different patterns from the data, reducing the correlation between the trees.

3. **Voting or Averaging:** Once all the trees are grown, when a new data point needs prediction, for **classification** problems, each tree "votes" for the class it thinks the data point belongs to as illustrated in 3.6. The final prediction is the most voted class (majority vote). For **regression** problems, each tree predicts a value, and the final prediction is the average of the individual tree predictions.

By combining the predictions of multiple trees with some randomness, a random forest aims to reduce variance and improve the overall accuracy of the model compared to a single decision tree.

3.5.3 Logistic Regression

Logistic Regression is a statistical method used for binary classification tasks, where the output variable (dependent variable) takes on only two possible outcomes (e.g., 0 or 1,

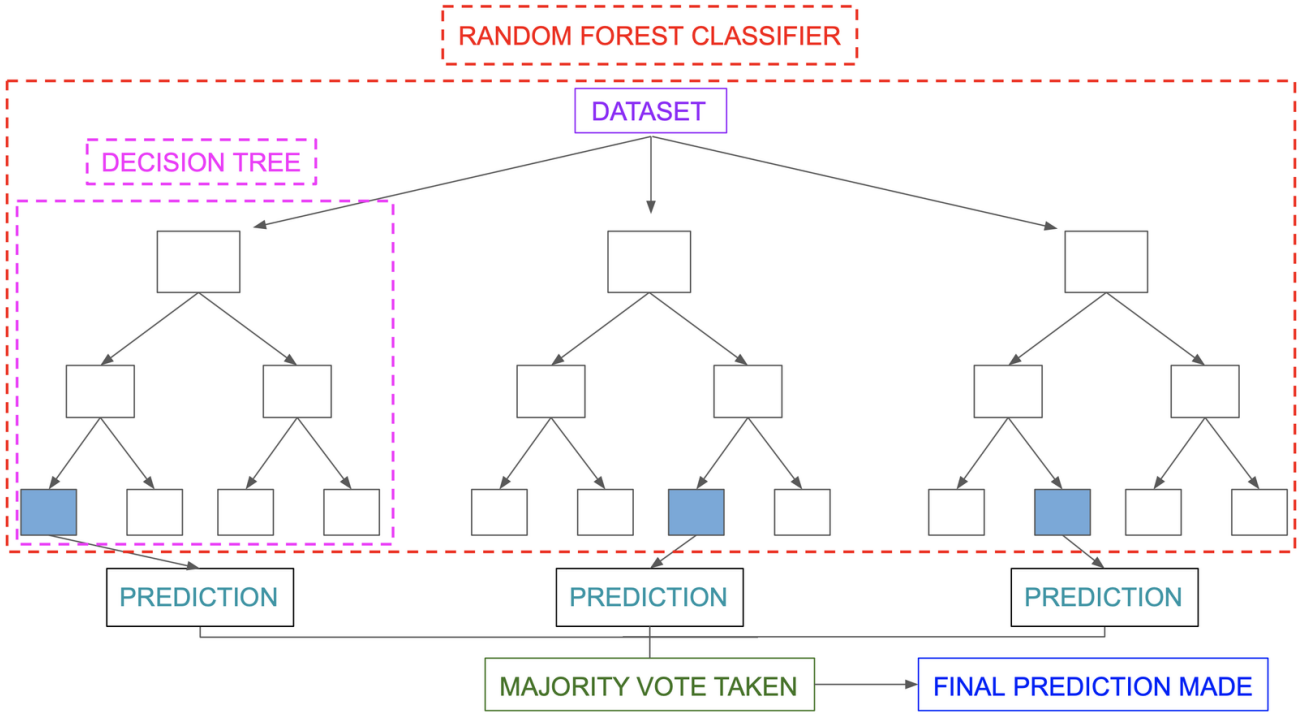


Figure 3.6: Random Forest Classifier, Source: Medium

yes or no, true or false) (Peng et al., 2002). Here's how logistic regression works:

1. **Model Construction:** Logistic regression models the probability that a given input belongs to a particular class. It applies a logistic function (also known as the sigmoid function) to a linear combination of the input features. The logistic function ensures that the output of the model lies between 0 and 1, which can be interpreted as probabilities.

   The logistic function shown in figure (3.7) is defined as:

$$logistic(z) = \frac{1}{1 + e^{-z}} \tag{3.2}$$

   where:

   $z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n$ - is the linear combination of input features.

   $\beta_0, \beta_1, \ldots, \beta_n$ - are the coefficients (parameters) learned during model training.

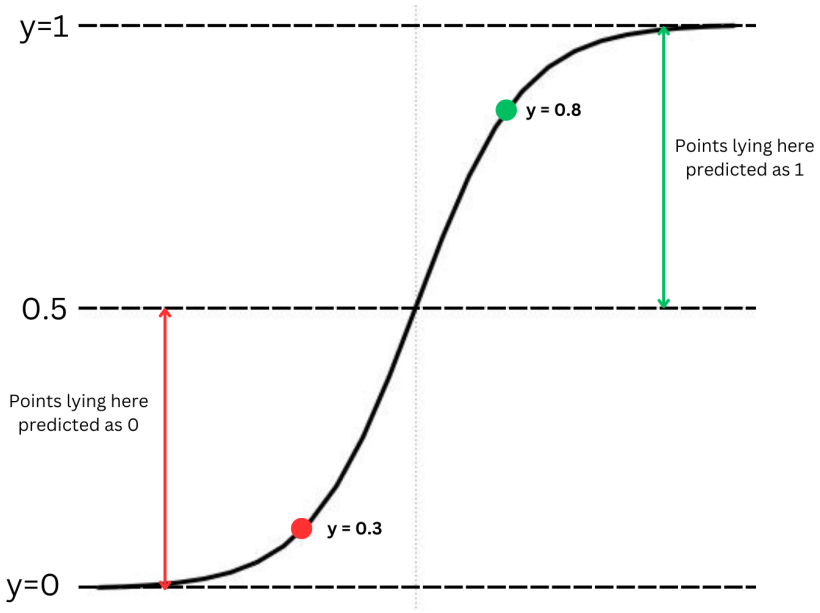


Figure 3.7: Logistic Regression

2. **Model Training:** During training, the logistic regression model learns the optimal values of the coefficients β that minimize the difference between the predicted probabilities and the actual class labels in the training data. This is typically achieved using optimization algorithms like gradient descent.

3. **Prediction:** Once the model is trained, it can be used to predict the probability that a new input belongs to a particular class. If the predicted probability is greater than a chosen threshold (usually 0.5), the input is classified as belonging to the positive class; otherwise, it is classified as belonging to the negative class.

4. **Model Evaluation:** The performance of the logistic regression model like other classification algorithms, is evaluated using metrics such as accuracy, precision, recall, F1-score, and the receiver operating characteristic ROC curve.

Logistic regression is widely used because it is simple, interpretable, and effective in many real-world applications, especially when the relationship between the independent variables and the outcome variable is assumed to be linear.

3.5.4 Evaluation and Testing

Classification algorithms are usually evaluated based on the following metrics:

1. **Precision:** This is the accuracy of positive predictions i.e. how many positive predictions are actually correct. In this case, it tells us what percentage of data points predicted as anomalies are actually anomalies.

$$Presision = \frac{TP}{TP + FP} \tag{3.3}$$

   where:

   - **True Positive (TP):** correctly classified normal cases
   - **False Positive (FP):** incorrectly classified normal cases

2. **Recall:** This measures the completeness of positive predictions. It indicates what proportion of the actual positive cases were correctly identified by the model.

$$Recall = \frac{TP}{TP + FN} \tag{3.4}$$

   where:

   - **False Negative (FN):** incorrectly classified anomalies

3. **F1-score:** This metric is the harmonic mean of precision and recall.

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{3.5}$$

   An F1 score of 1 represents perfect performance, where the model precisely identifies all positive cases (precision) and misses none of them (Recall). Conversely, a score of 0 indicates the model is failing to identify any positive cases correctly.

For imbalanced datasets, like the one used in this study, the F1 score is usually more informative than accuracy which can be misleading as the majority class can disproportionately contribute towards the accuracy.

4. **Accuracy:** This is just the proportion of predictions that a model makes that are correct. A higher accuracy indicates a better-performing model. It is simple and intuitive to understand.

$$Accuracy = \frac{Correct\ Predictions}{Total\ Predictions} \tag{3.6}$$

As hinted at before, accuracy can be misleading for imbalanced datasets. It also treats all misclassifications equally which in some cases may be a critical error. For instance, in a medical diagnosis system, a false negative (missing a disease) could be much more consequential than a false positive (mistakenly identifying a disease).

5. **ROC-AUC:** This is a tool used to evaluate the performance of classification models, especially when dealing with binary classification. It shows how well the model distinguishes positive and negative classes and it ranges from 0 to 1. A higher AUC indicates better performance. A perfect model would have an AUC of 1, while a score of 0.5 indicates that the model is just guessing. ROC analysis relies on the four metrics derived from a confusion matrix; TP, FP, True Negative (TN) and FN. From these metrics, we can calculate and plot the recall (y-axis) and Specificity $= \frac{\text{TP}}{\text{TN+FP}}$, which measures how well the model avoids identifying negative cases as positive on the x-axis. The ROC curve is then drawn by plotting the sensitivity for various classification thresholds across the data. This important curve will show us the trade-off between correctly classifying positive cases (high sensitivity) and avoiding false positives (high specificity) at different points in the model's decision-making process.

3.5.5 Model Deployment

VMS machine log data lives in a Splunk cloud and its system notifications are delivered through email and or Microsoft Teams messaging service. The notifications from this model will follow the same path, however, pending integration into the messaging system, a Streamlit ("Streamlit • A faster way to build and share data apps", n.d.) application was developed to consume an API and display the model output thus providing validation. Streamlit is a free and open-source Python library built to enable the creation of web applications for machine learning, data visualisation and data analysis with ease ("Streamlit • A faster way to build and share data apps", n.d.).

The resulting model was deployed via an API. This was done because an API will ensure a modular, easily maintainable and extensible solution that can be deployed in a variety of platforms and environments. An API will also enable separation of concerns e.g. model development from usage, as well as support scaling of the system to handle multiple users and requests.

FastAPI ("FastAPI", n.d.) was used to develop the API because:

1. It is fast. It is one of the fastest API development frameworks, about 45% faster than Flask as demonstrated by (Bansal and Ouda, 2022).

2. It automatically documents the API based on type hints and docstrings in your code.

3. It has an easy validation feature that automatically verifies input data, reducing errors and speeding up the code-writing process.

One key difference between software deployment and machine learning model deployment is that machine models degrade quicker than traditional software (Huyen, 2022). This means that deployment is not the final stage in the machine learning development life cycle, monitoring and retraining are an integral part of the deployment process. Because

of this, a database to hold features and the model prediction was developed for later use in monitoring and evaluation of the model and even retraining.

This database was built in SQLite. SQLite is a self-contained, high-reliability, embedded, full-featured, public-domain, SQL database engine ("SQLite Documentation", n.d.). It is the most used database engine on the World Wide Web. Python has a library to access SQLite databases, called SQLite3, intended for working with this database which has been included with the Python package since version 2.5. SQLite is serverless, self-contained and requires no configuration to set up.

Chapter 4

System Design and Architecture

This chapter describes the architecture of the data pipeline that performs the ELT process from the Splunk server or a CSV file containing raw log files, cleans the data, interacts with the machine learning model and outputs an alert via a Streamlit application.

This will be a batch-processing application (Huyen, 2022). Since the log files are uploaded to the central server every 24 hours, the data pipeline will be run once a day per machine and the predictions served up daily.

4.1 System Components

The prediction system comprises 6 components as shown in figure 4.1.

4.1.1 Splunk Database/ CSV Source

The data is sourced from the central Splunk server using a custom search term to fetch the required log files for the last 24 hours. For this study, a CSV file with these log files will be used but the API can still be integrated into Splunk to pull the required data.

4.1.2 Model API

An API to interact with the various system components was developed using FastAPI framework. FastAPI is a built in python and is 2 to 3 times faster than comparable frameworks like Flask (Voron, 2021). It also has an automatic documentation feature that utilises docstrings and endpoints to quickly summarise your API. It sits at the

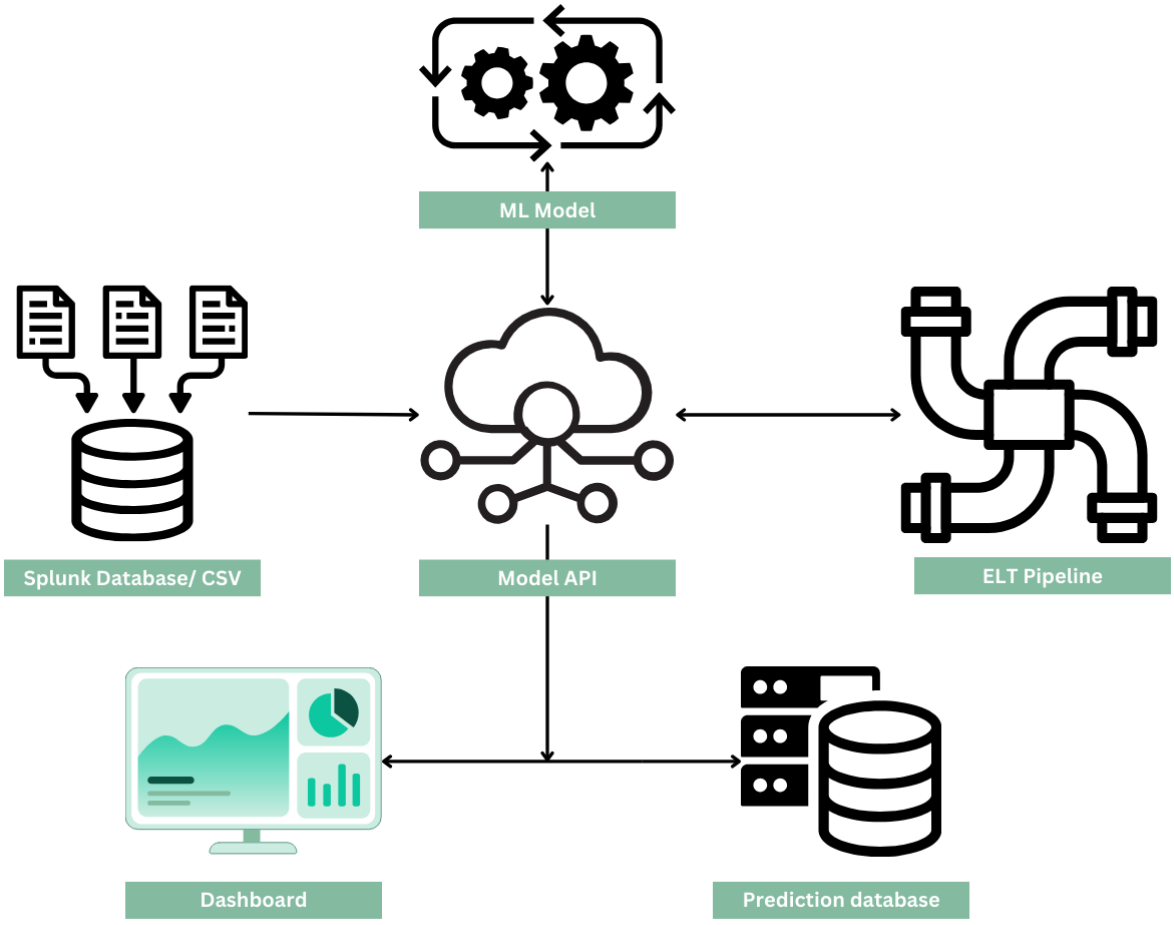


Figure 4.1: System Architecture

core of the system moving data and results between the various components. It has the following endpoints:

1. **/raw_data:** This endpoint will fetch the cleaned data as a CSV file and return it as a Javascript Object Notation (JSON) file.

2. **/prediction:** This endpoint accepts a JSON file, runs the prediction model and saves the features along with the model prediction to the SQLite database.

3. **/db_data:** This endpoint retrieves all the data in the database and serves it up as a JSON file.

4. **/clean:** This endpoint accepts the raw log file, cleans them through the ELT pipeline and returns the cleaned logs ready for the model.

The documentation can be found in 6.3

4.1.3 ELT Pipeline

This module accepts input from the /clean API endpoint. This is a Python script that extracts and loads the data and then cleans and transforms it in readiness for the model.

4.1.4 ML Model

This is the model that was developed in this study. It interacts with the API to accept feature inputs and serve up the predictions to the API for display and storage in the database.

4.1.5 Dashboard

This is the user interface that displays the data and the predictions. It was developed using Streamlit. Figure 4.2 and 4.3 show screenshots of loading the data for prediction and the prediction results.

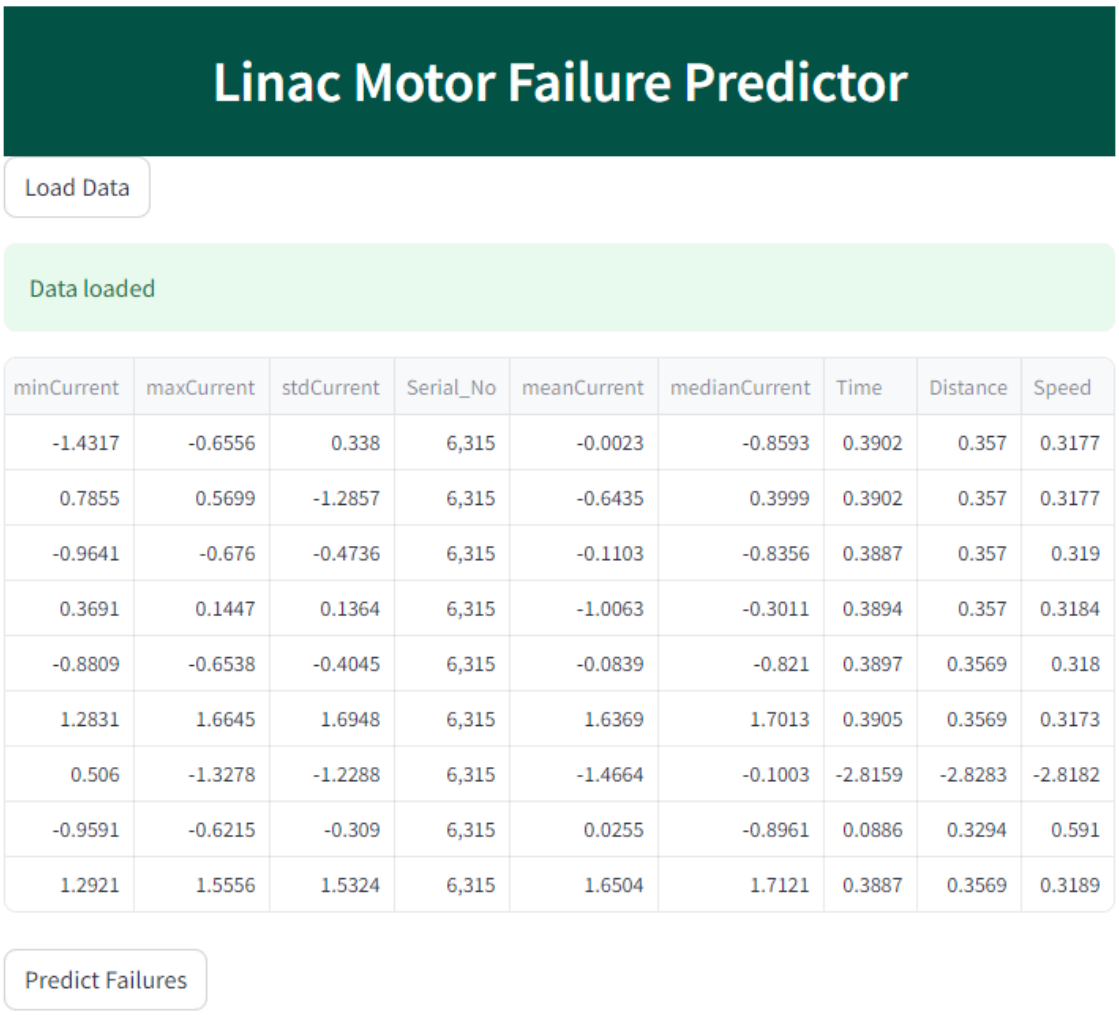
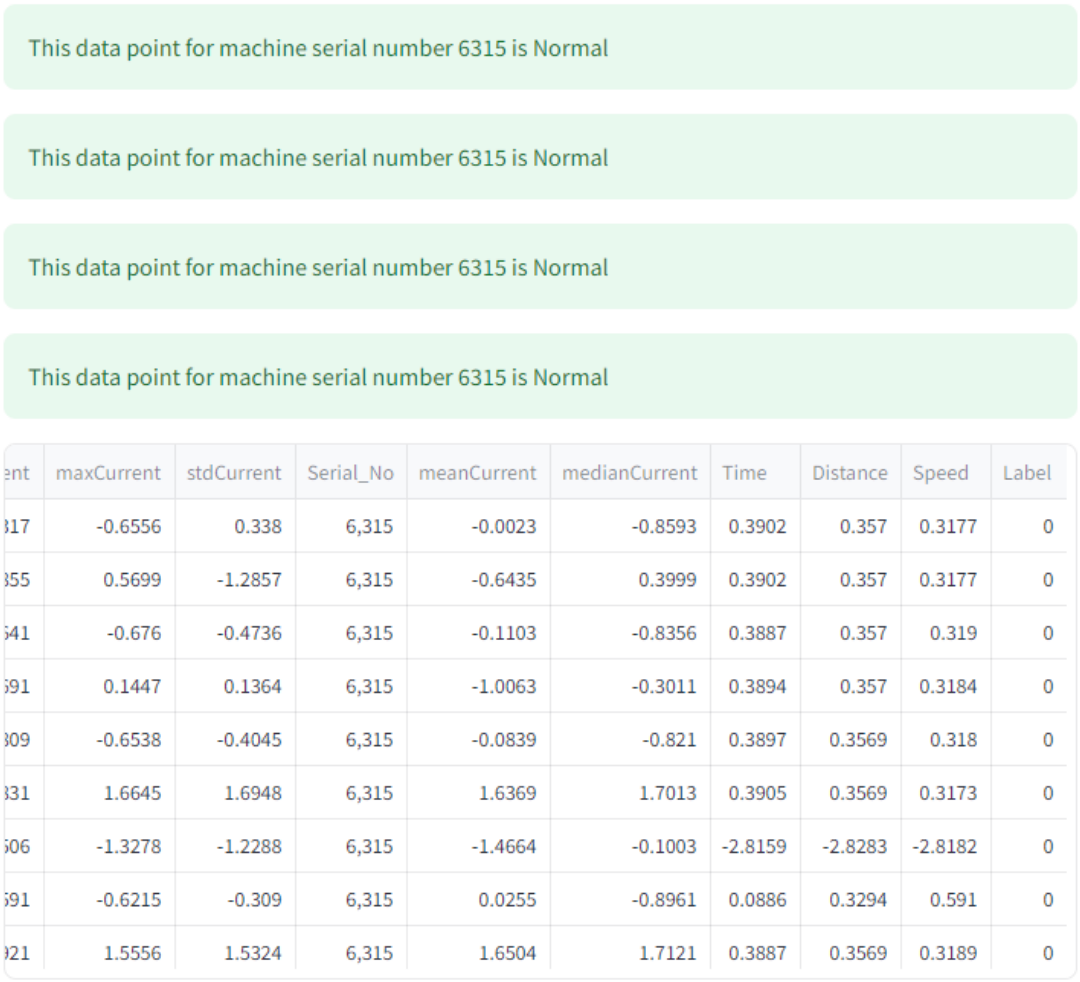


Figure 4.2: Streamlit UI I



ent	maxCurrent	stdCurrent	Serial_No	meanCurrent	medianCurrent	Time	Distance	Speed	Label
317	-0.6556	0.338	6,315	-0.0023	-0.8593	0.3902	0.357	0.3177	0
355	0.5699	-1.2857	6,315	-0.6435	0.3999	0.3902	0.357	0.3177	0
641	-0.676	-0.4736	6,315	-0.1103	-0.8356	0.3887	0.357	0.319	0
691	0.1447	0.1364	6,315	-1.0063	-0.3011	0.3894	0.357	0.3184	0
309	-0.6538	-0.4045	6,315	-0.0839	-0.821	0.3897	0.3569	0.318	0
331	1.6645	1.6948	6,315	1.6369	1.7013	0.3905	0.3569	0.3173	0
606	-1.3278	-1.2288	6,315	-1.4664	-0.1003	-2.8159	-2.8283	-2.8182	0
691	-0.6215	-0.309	6,315	0.0255	-0.8961	0.0886	0.3294	0.591	0
921	1.5556	1.5324	6,315	1.6504	1.7121	0.3887	0.3569	0.3189	0

Figure 4.3: Streamlit UI II

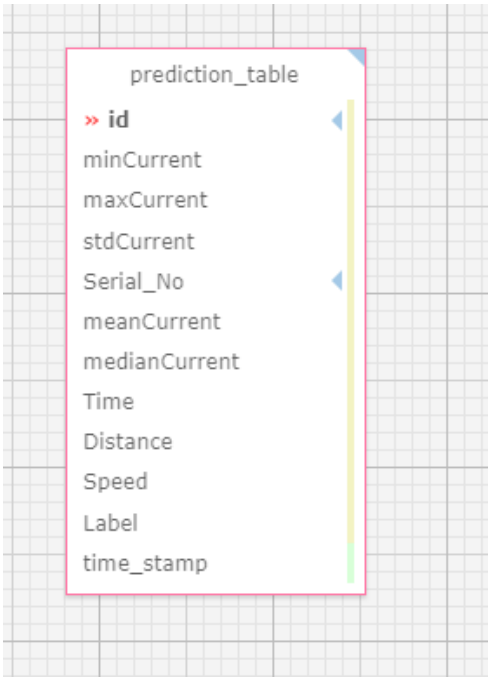


Figure 4.4: Database Design

4.1.6 Prediction Database

The performance of machine learning models usually degrades over time while in production. This is because of the fluid nature of patterns in the real world, concept drift and other reasons, unlike traditional software which will work as written with no deviations. For this reason, the life cycle of a model does not end in deployment but extends into monitoring and retraining. For this reason, a monitoring database was designed. This database is built using the lightweight SQLite. Its purpose is to store the features and predictions of the model while in production so that they can be used to monitor the model's performance and provide data for monitoring and evaluation. The schema is shown in figure 4.4 below.

Chapter 5

Results and Discussion

5.1 Exploratory Data Analysis

Exploratory Data Analysis (EDA) was conducted on the dataset to understand its distribution and relationship between the features before any machine learning modelling was carried out. The dataset consists of 153,185 rows, 8 features and a target row, (Label), of normal (0) and anomaly (1). The data was collected between January and February 2024 from the VMS Splunk repository.

5.1.1 Univariate Analysis

From the individual feature plots (5.6, 5.2, 5.3, 5.4, 5.1, 5.5, 5.6, 5.7 and 5.8), we can observe the following:

- minCurrent, maxCurrent and medianCurrent have a uniform distribution.
- stdCurrent, meanCurrent, Time, and Distance have an exponential distribution.
- Time is spread out from 0 to about 100 with a concentration between 0 and 15.
- Distance is spread out from 0 to about 700 with a concentration between 0 and 100.

5.1.2 Bivariate Analysis

Bivariate analysis is a technique used to investigate the relationship between two variables. This can reveal hidden patterns and insights hidden in the data that can be used to explain

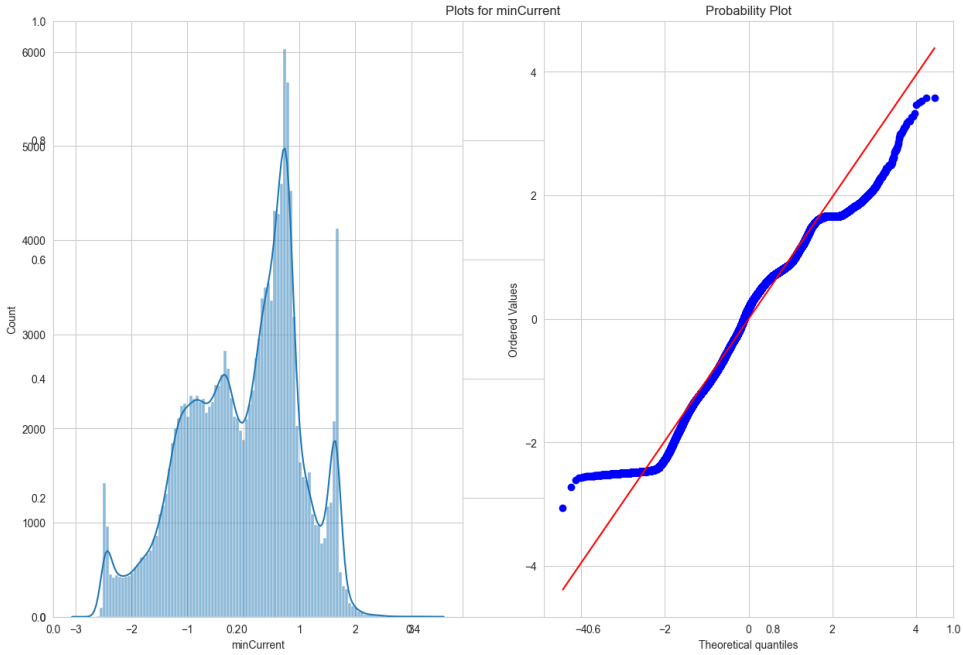


Figure 5.1: Minimum current distribution

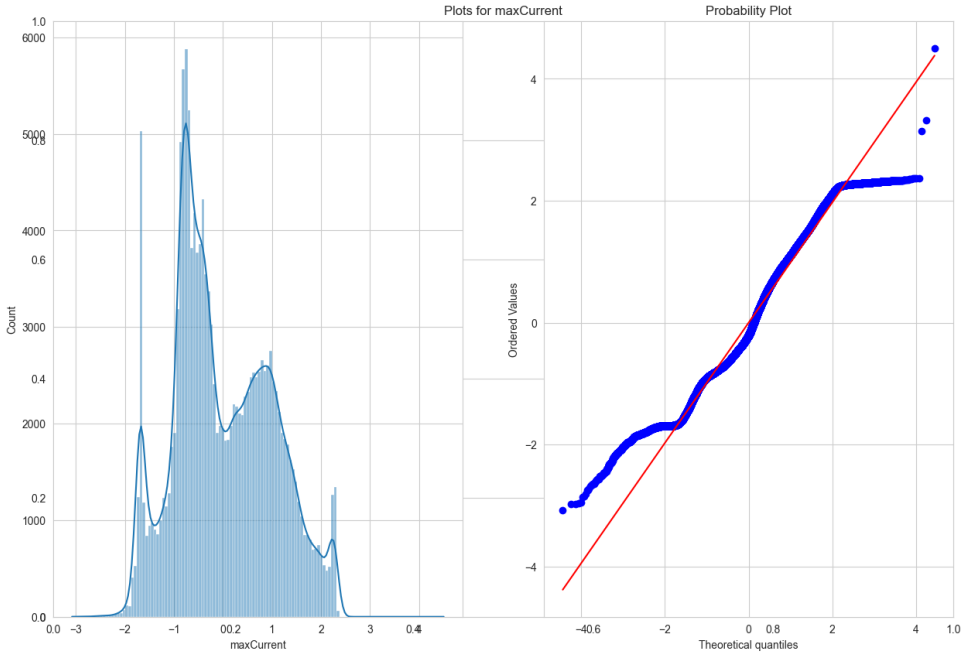


Figure 5.2: Maximum current distribution

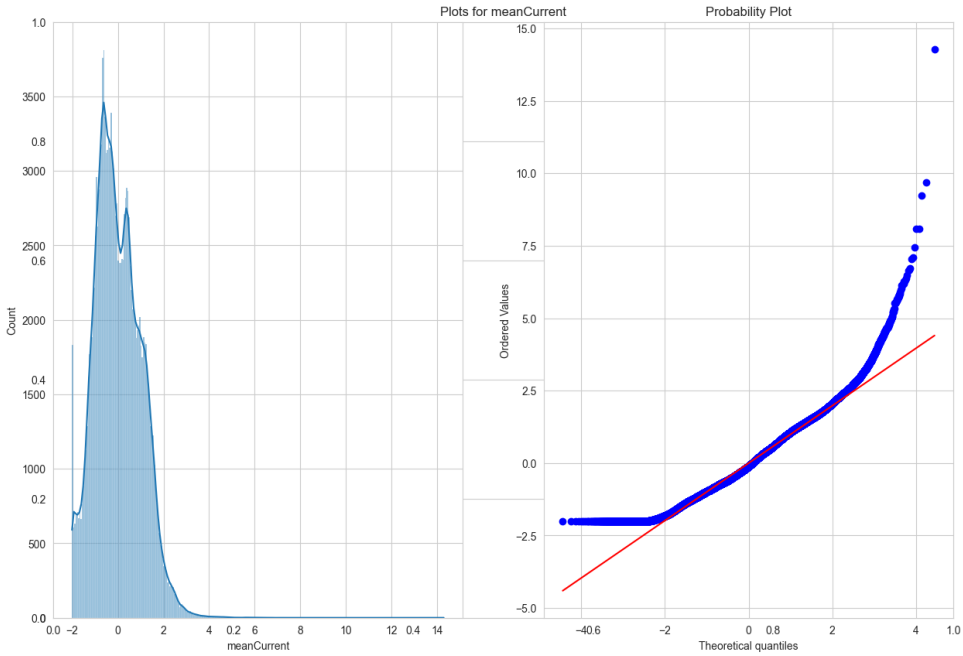


Figure 5.3: Mean current distribution

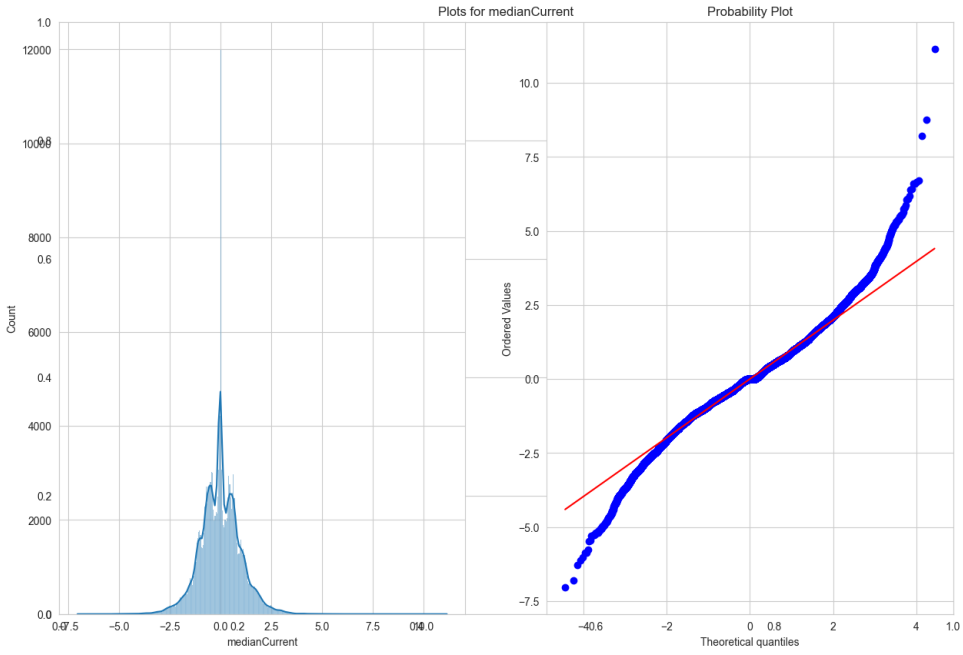


Figure 5.4: Median current distribution

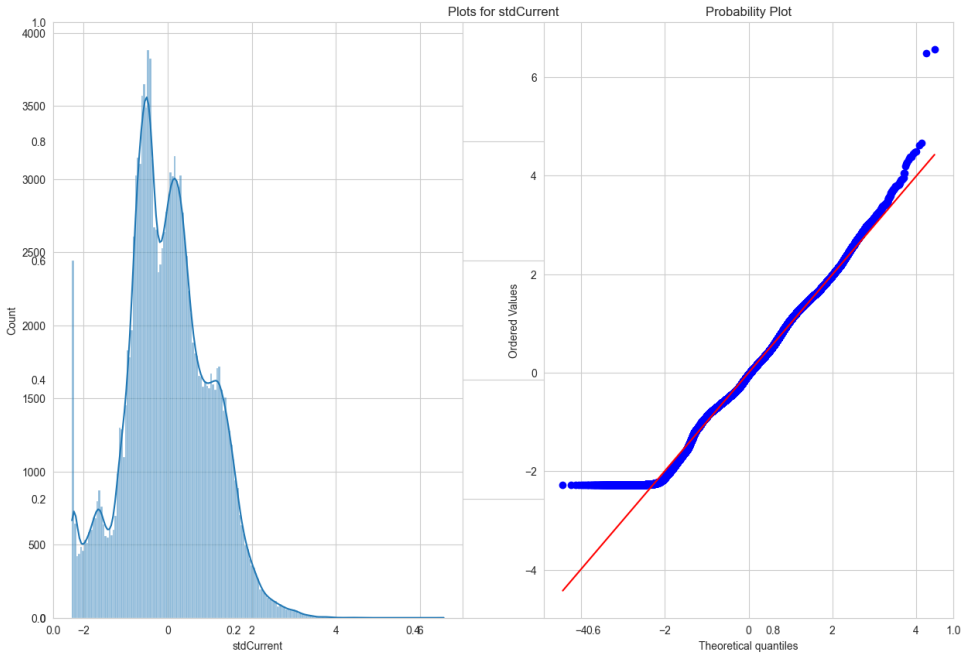


Figure 5.5: Standard deviation of current distribution

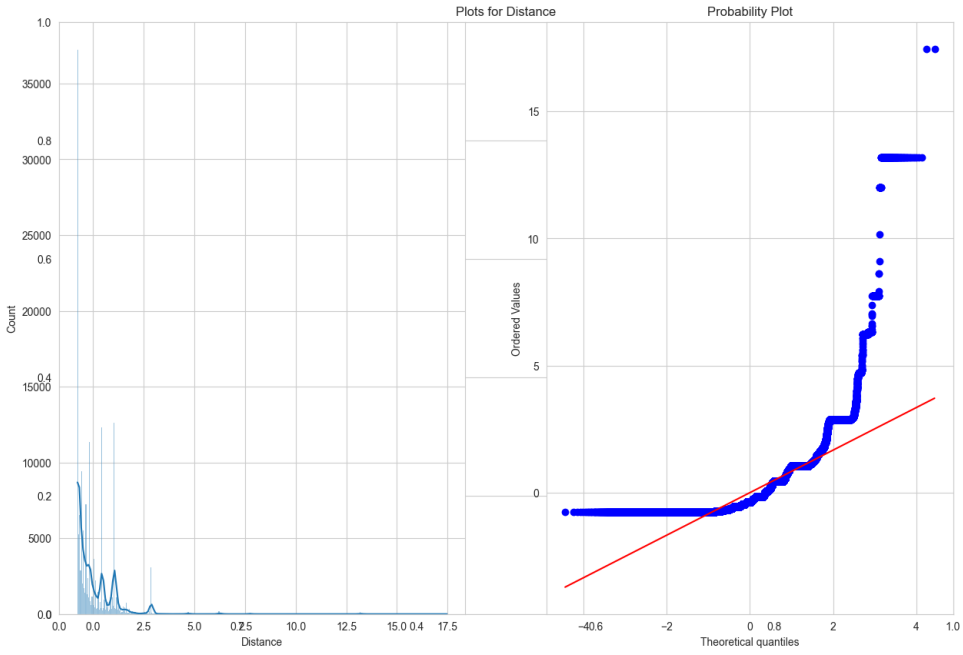


Figure 5.6: Distance distribution

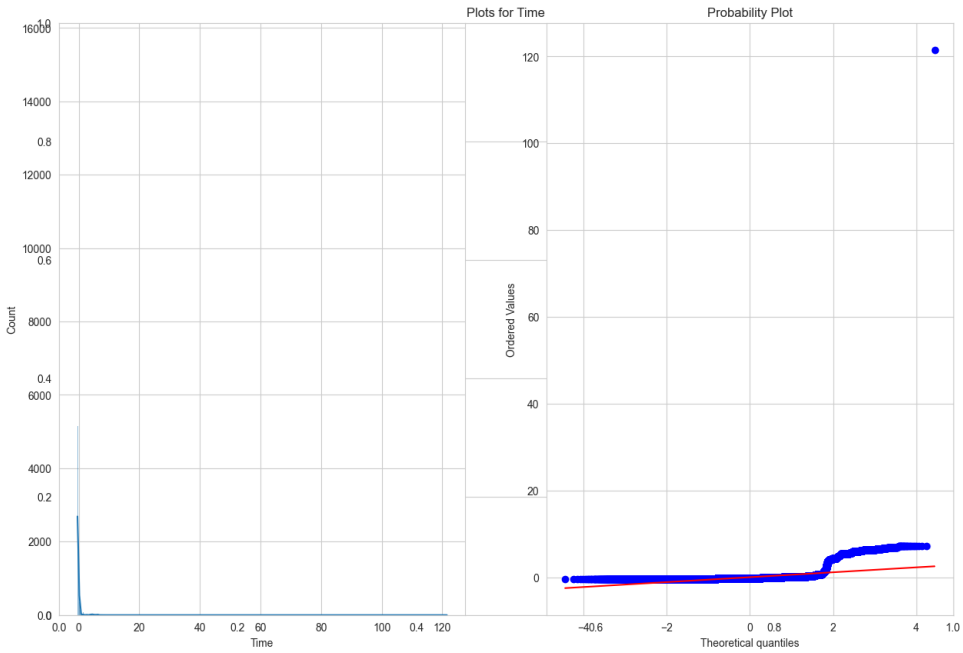


Figure 5.7: Time distribution

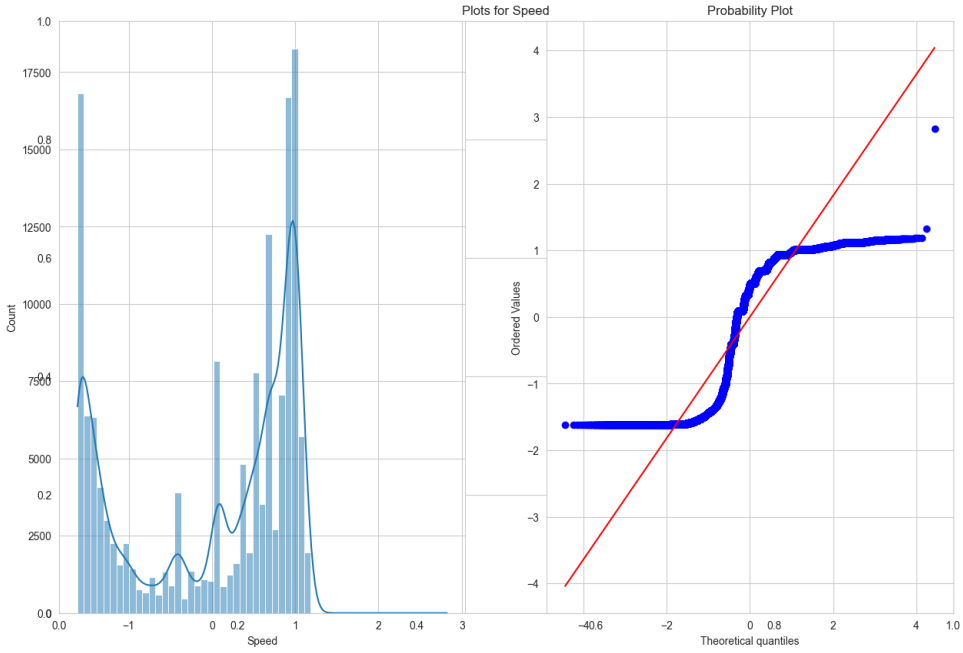


Figure 5.8: Speed distribution

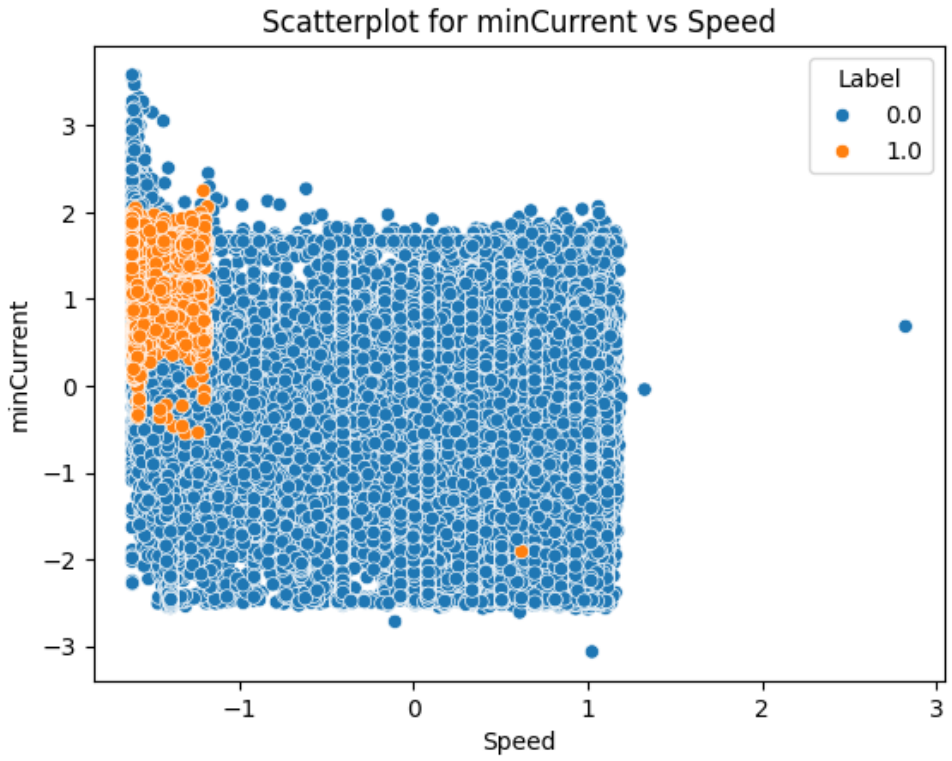


Figure 5.9: Minimum current Vs. Speed

the model. Below are some interesting plots between specific variables.

An interesting trend is observed from these graphs; almost all the anomalous data points exhibit a low speed. This is in line with our literature review that suggests that a slow motor is a motor with an issue.

5.1.3 Multivariate Analysis

The correlation heatmap in 5.16 shows the correlation between the various features.

A strong, positive correlation exists between meanCurrent and stdCurrent (0.76). medianCurrent also has a strong positive correlation with minCurrent(0.67) and maxCurrent(0.67) which is to be expected as the more current the motor draws, the higher the median, maximum and minimum values will be and vice versa. Speed and Distance have a moderate positive correlation (0.5) which is normal as they are directly proportional. The target variable has at best a weak positive correlation with minCurrent (0.21) and a

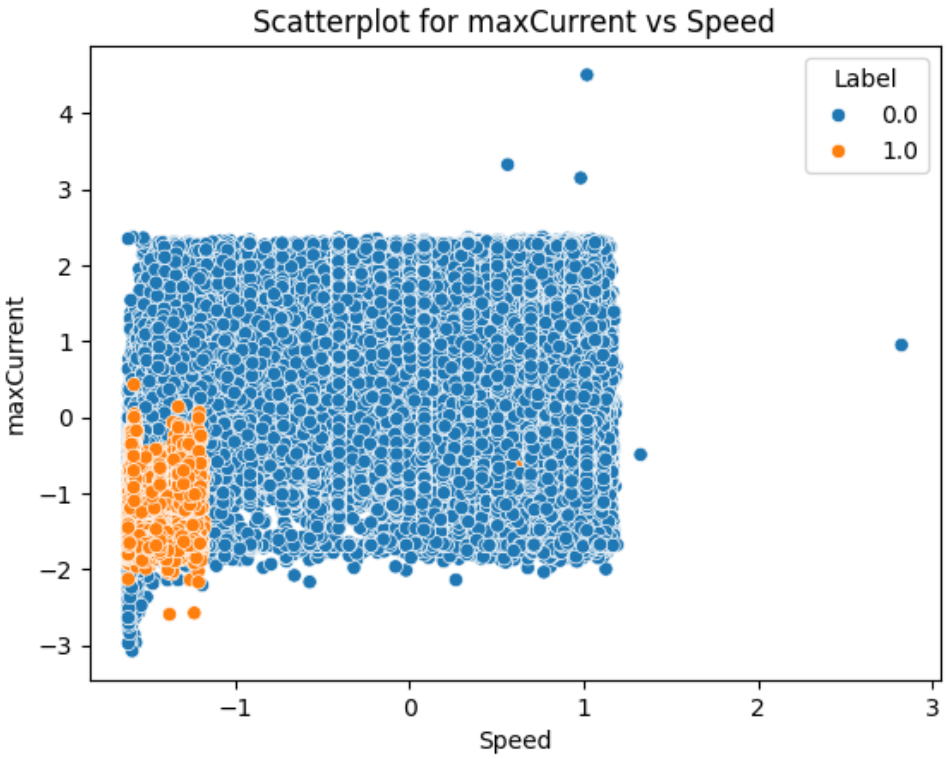


Figure 5.10: Maximum current Vs. Speed

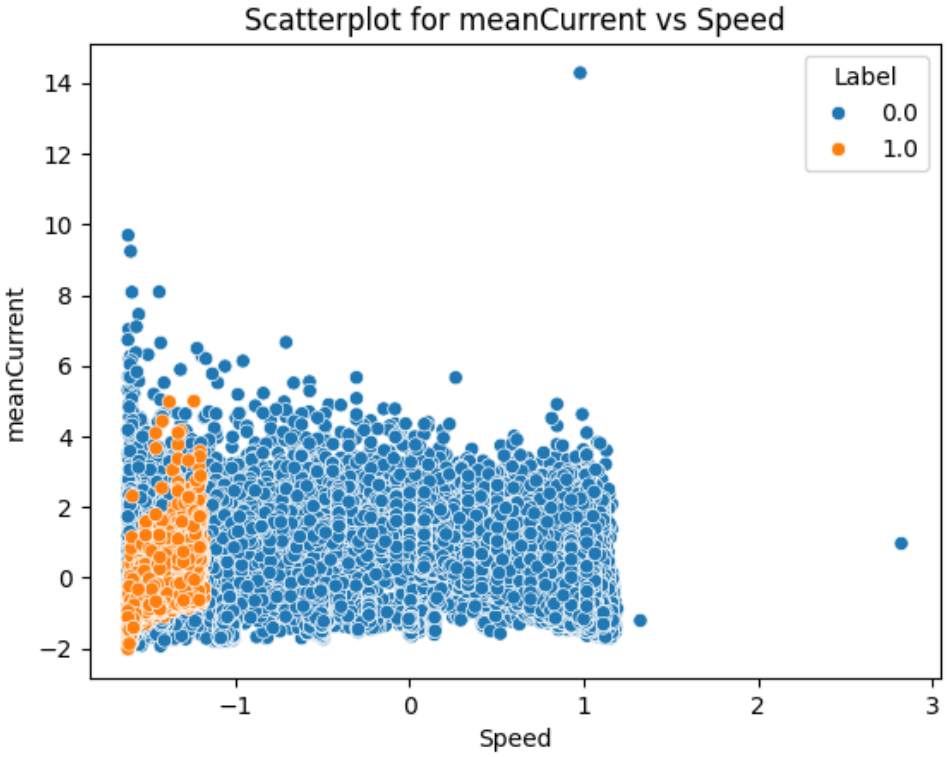


Figure 5.11: Mean current Vs. Speed

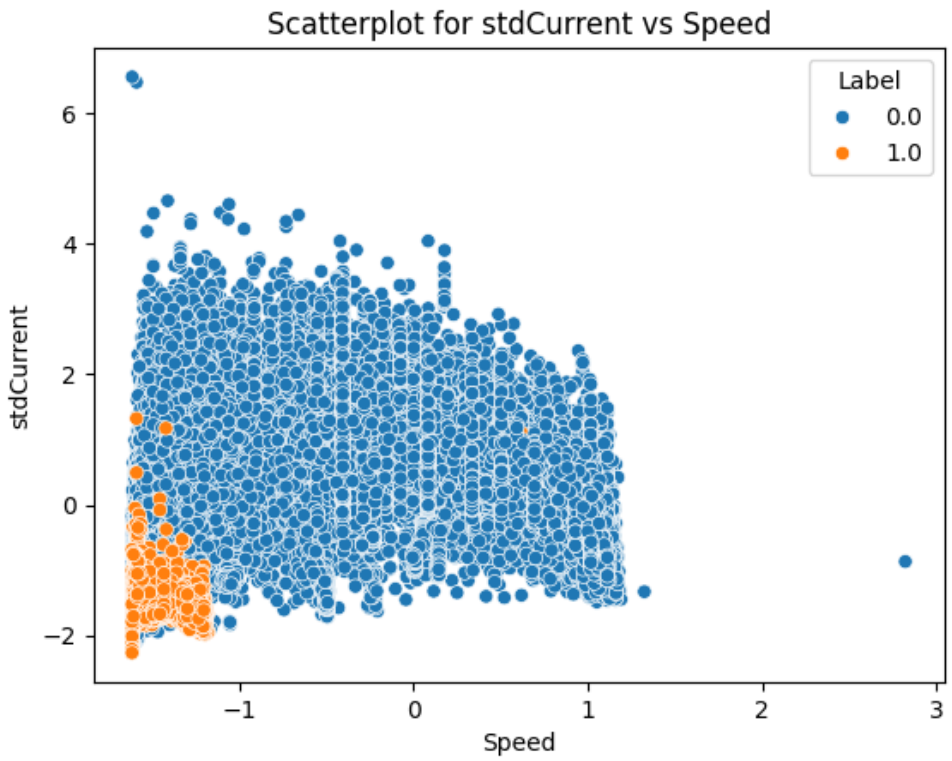


Figure 5.12: Standard deviation of current Vs. Speed

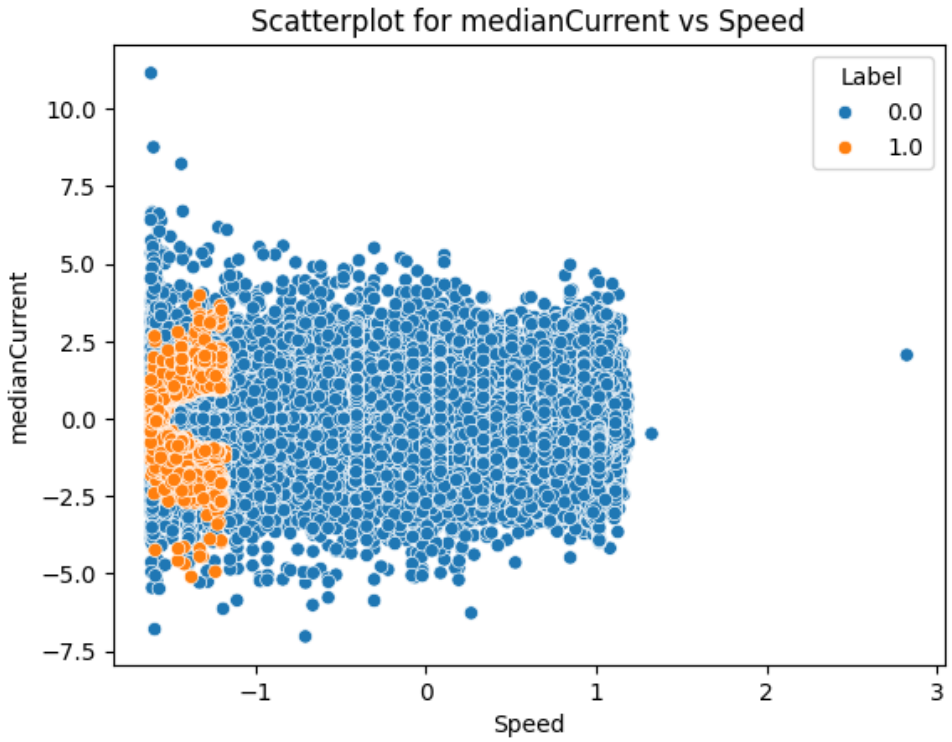


Figure 5.13: Median current Vs. Speed

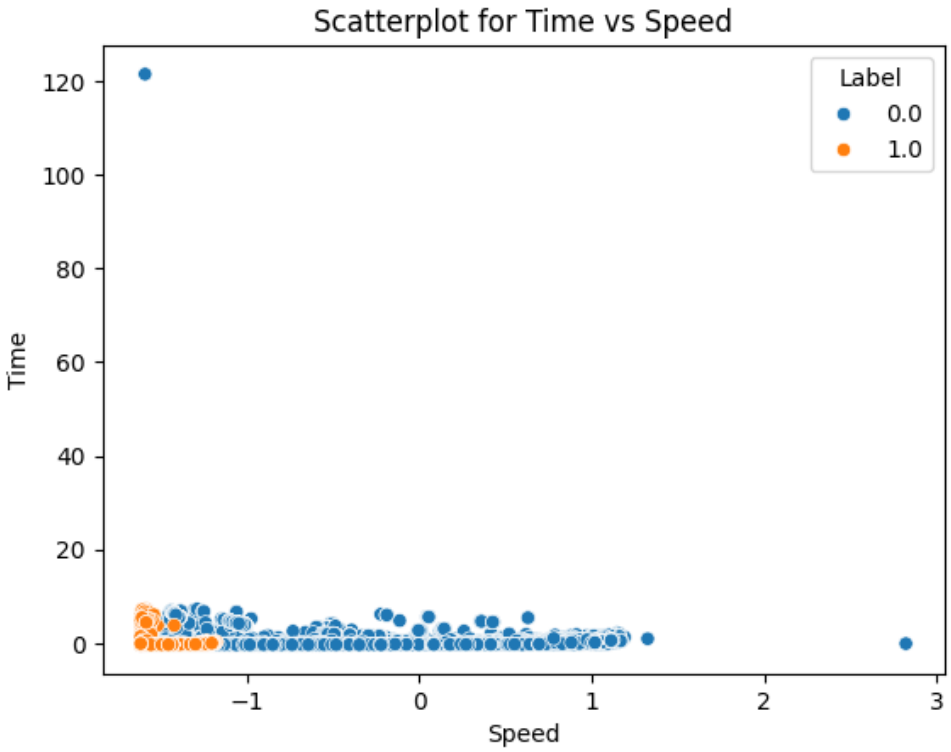


Figure 5.14: Time Vs. Speed

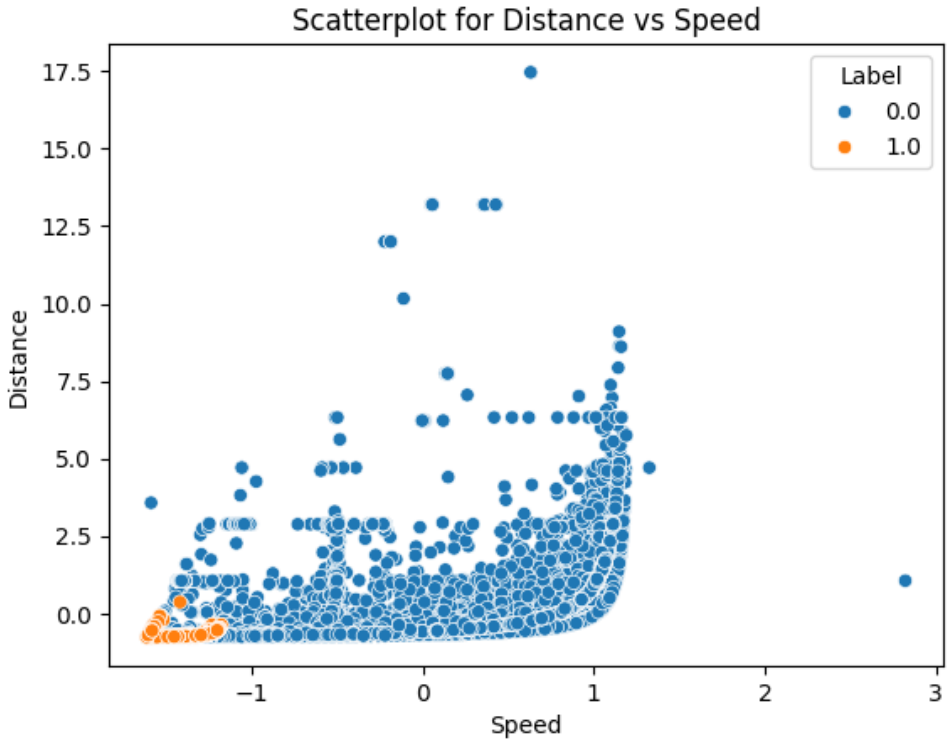


Figure 5.15: Distance Vs. Speed

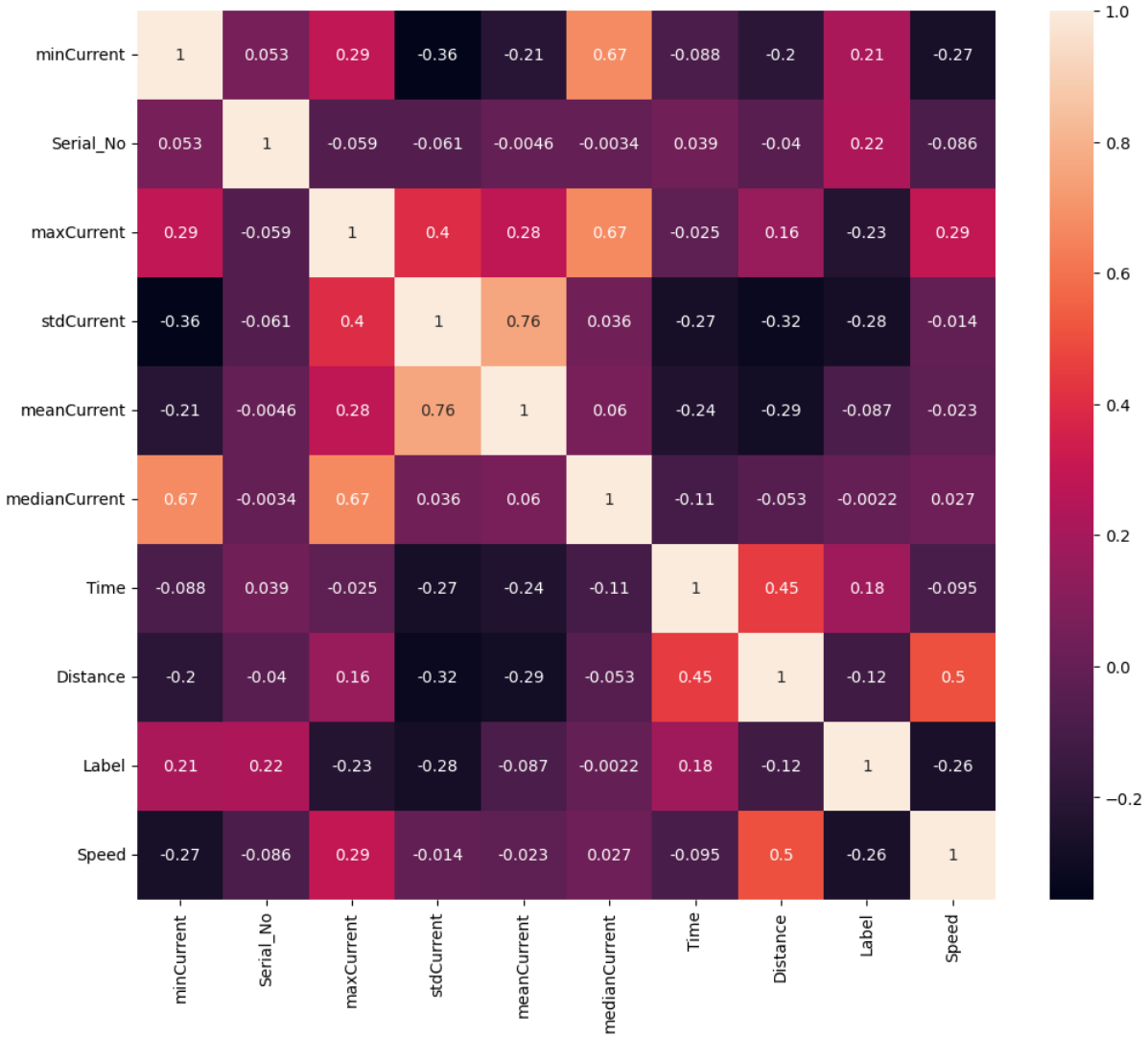


Figure 5.16: Correlation Heatmap

weak negative correlation with Speed (-0.26).

5.2 Model Performance and Evaluation

Here we compare the performance of the 3 classifiers based on the following metrics:

1. **Precision:** This is the accuracy of positive predictions i.e. how many positive predictions are actually correct as illustrated in equation 3.3. In this case, it tells us what percentage of data points predicted as anomalies are actually anomalies. From plot 5.22 we can see that the Random Forest algorithm performed the best at 99.87% closely followed by Logistic Regression at 99.14%. Isolation Forest is a distant third at 58.48% meaning it is doing slightly more than just guessing.

2. **Recall:** This measures the completeness of positive predictions as seen in equation 3.4. It indicates what proportion of the actual positive cases were correctly identified by the model. In this case, recall tells us what percentage of anomalies the model managed to catch. From the plot 5.25, we can see that Random Forest came out on top at 99.86%, Logistic Regression at 93.8% and Isolation Forest at 70.08%.

3. **F1-score:** This is the harmonic mean of precision and recall calculated using equation 3.5. For imbalanced datasets, like the one used in this study, the F1 score is usually more informative than accuracy which can be misleading as the majority class can disproportionately contribute towards the accuracy. As visualised in 5.23, in this study, Random Forest again came out on top with 99.87%, Logistic Regression followed with 96.39% and Isolation Forest had 63.76%.

4. **Accuracy:** This is just the proportion of predictions that a model makes that are correct. A higher accuracy indicates a better-performing model. It is simple and intuitive to understand as shown in equation 3.6. As hinted at before, accuracy can be misleading for imbalanced datasets. It also treats all misclassifications equally,

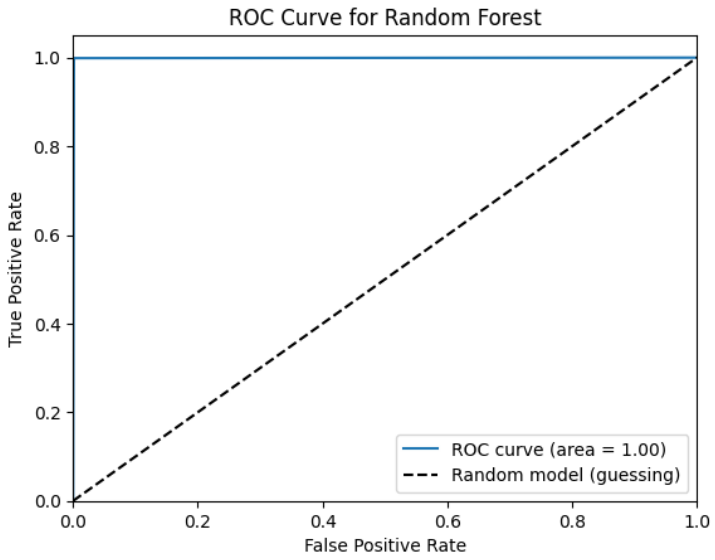


Figure 5.17: Random Forest ROC-AUC Curve

which may sometimes be a critical error. For instance, in a medical diagnosis system, a false negative (missing a disease) could be much more consequential than a false positive (mistakenly identifying a disease). As shown in 5.23, Random Forest had the highest accuracy at 99.87%, Logistic Regression 96.28% and Isolation Forest came third at 60.1%. With such good results, over-fitting was suspected and to check, the Random Forest model was tested against both the training data and test data and yielded a test score of 99.86% and training score of 100% dispelling any suspicions of over-fitting.

5. **ROC-AUC:** This curve shows how well the model distinguishes positive and negative classes and it ranges from 0 to 1. A higher AUC indicates better performance. A perfect model would have an AUC of 1, while a score of 0.5 indicates that the model is just guessing. Figures 5.17, 5.18 and 5.19 show the ROC-AUC curves for the 3 models. Random Forest is the best performer at 0.9987, Logistic Regression at 0.9644 and Isolation Forest at 0.602.

Table 5.1 summarises the model performance.

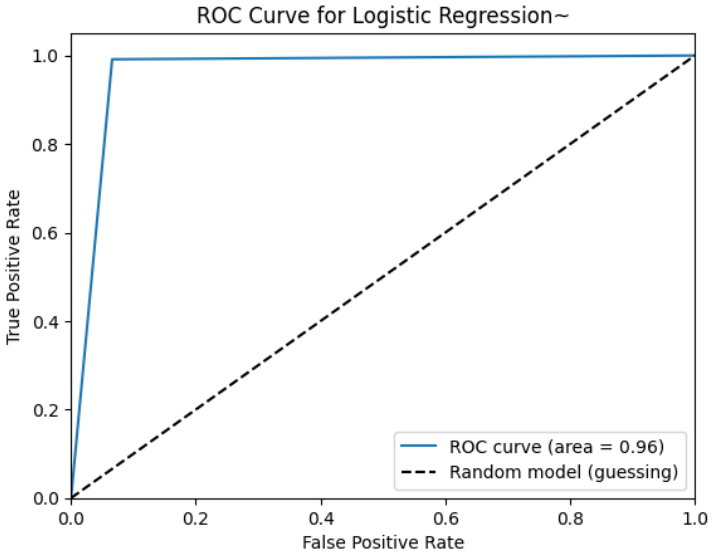


Figure 5.18: Logistic Regression ROC-AUC Curve

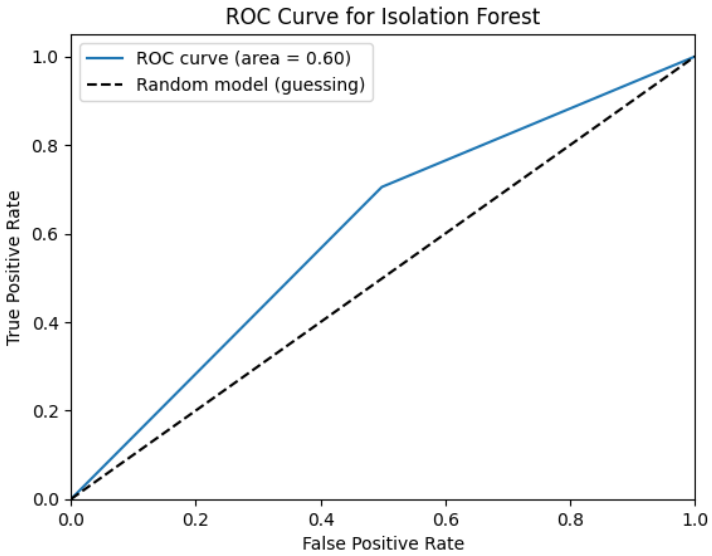


Figure 5.19: Isolation Forest ROC-AUC Curve

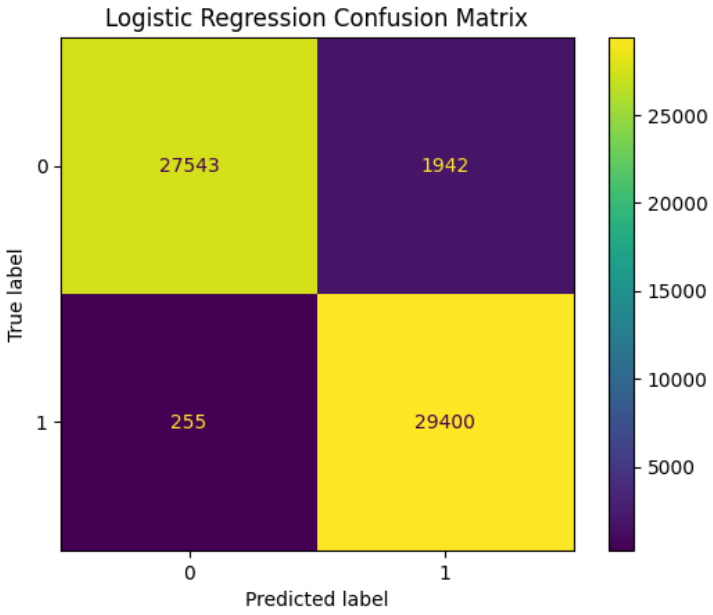


Figure 5.20: Logistic Regression Confusion Matrix

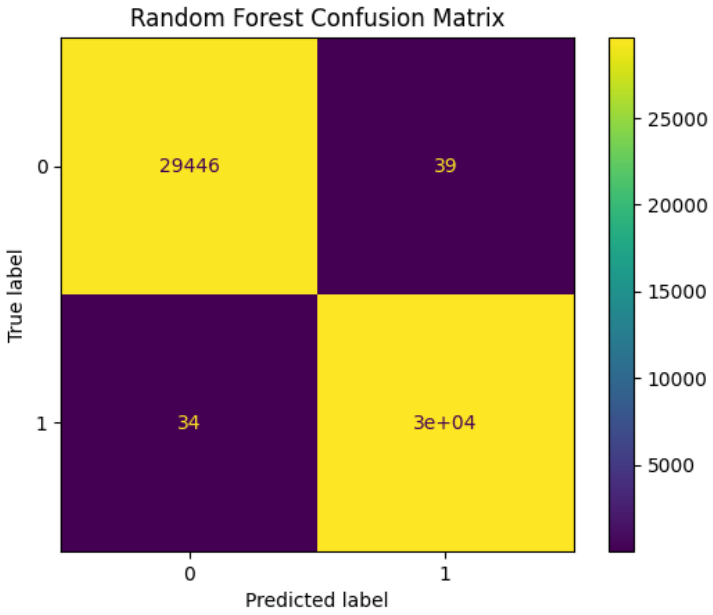


Figure 5.21: Random Forest Confusion Matrix

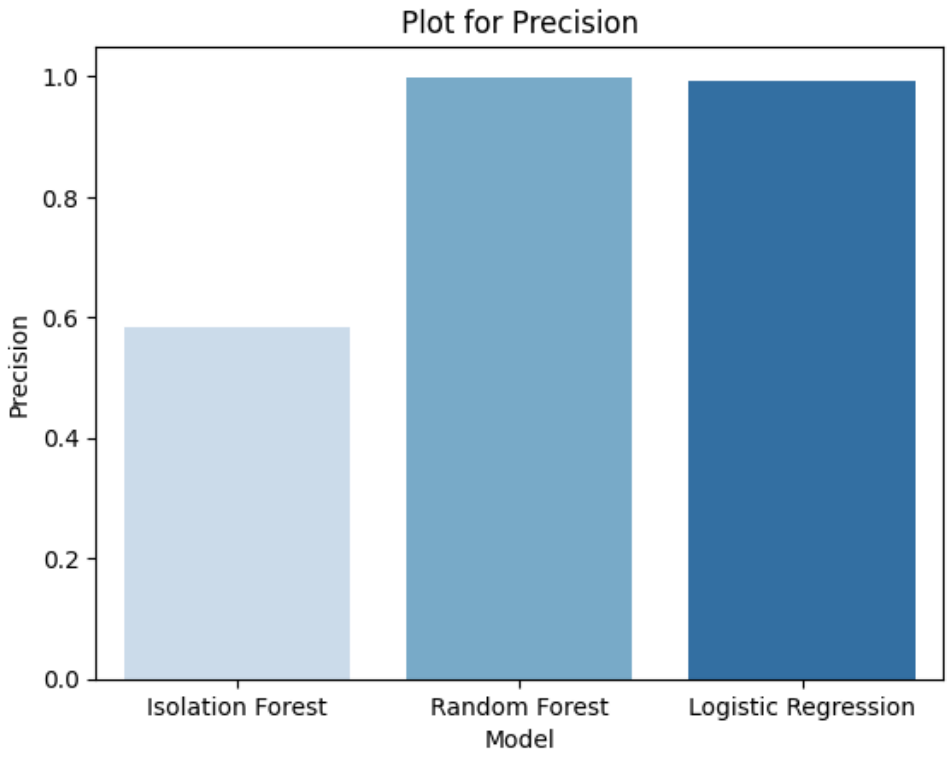


Figure 5.22: Precision Score

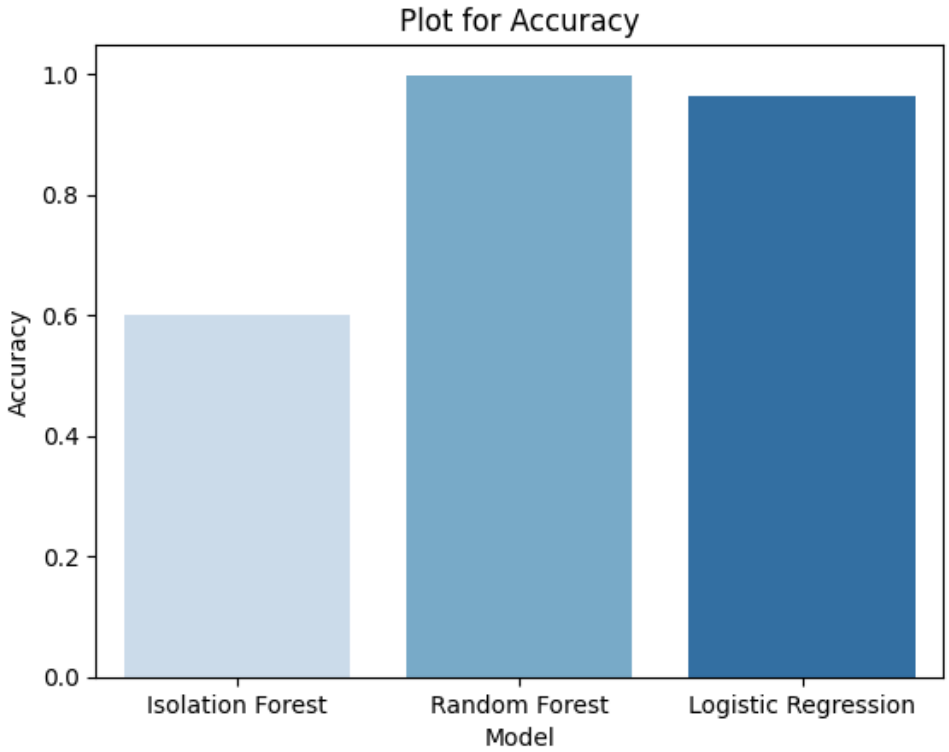


Figure 5.23: Accuracy

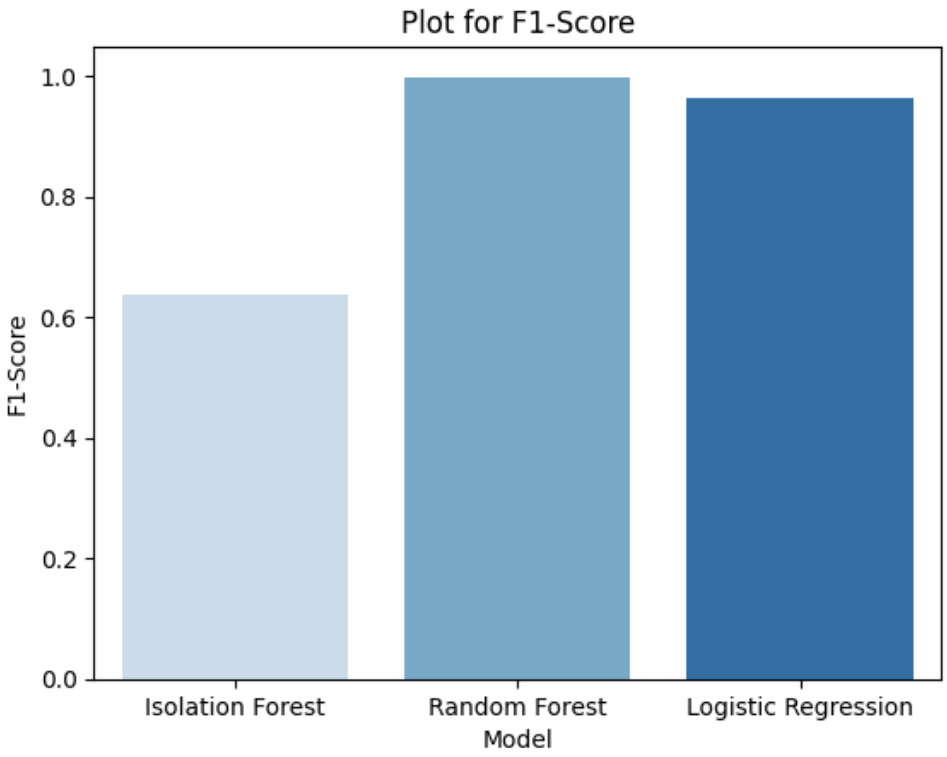


Figure 5.24: F1 Score

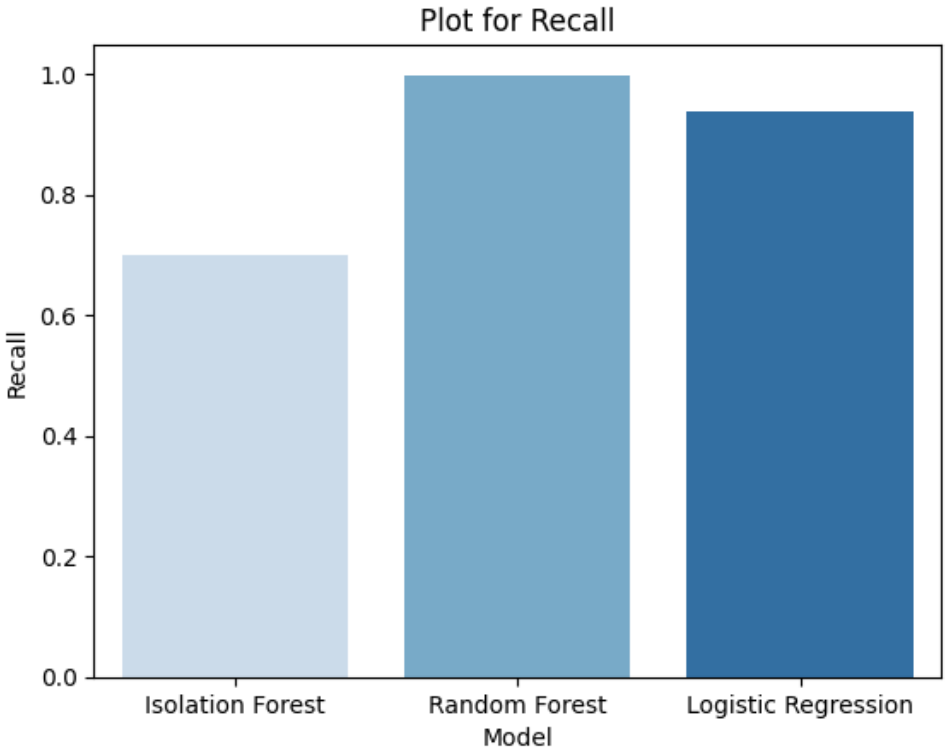


Figure 5.25: Recall

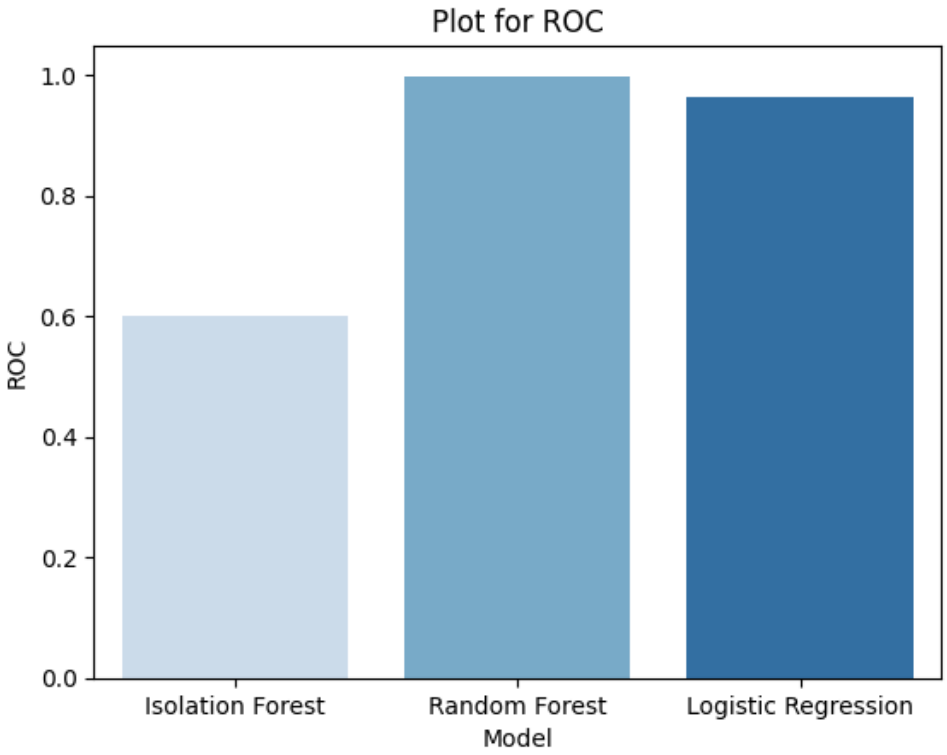


Figure 5.26: ROC

Model	F1-Score	Precision	Recall	ROC-AUC	Accuracy
Isolation Forest	0.640921	0.587421	0.705142	0.601995	0.601924
Random Forest	0.998769	0.998853	0.998685	0.998766	0.998766
Logistic Regression	0.963982	0.991401	0.938038	0.964433	0.962851

Table 5.1: Summary of Model Performance

5.3 Model Feature Importance

In machine learning, feature importance is determining the implication of the model input features to the final model output. In this case, it determines how significant an input feature or parameter is in predicting whether a data point is normal or an anomaly. This study utilised SHAP values to calculate feature importance. SHAP are useful for explaining the significance of the features behind a machine learning model (Lundberg and Lee, 2017). They are based on game theory concepts where each feature is viewed

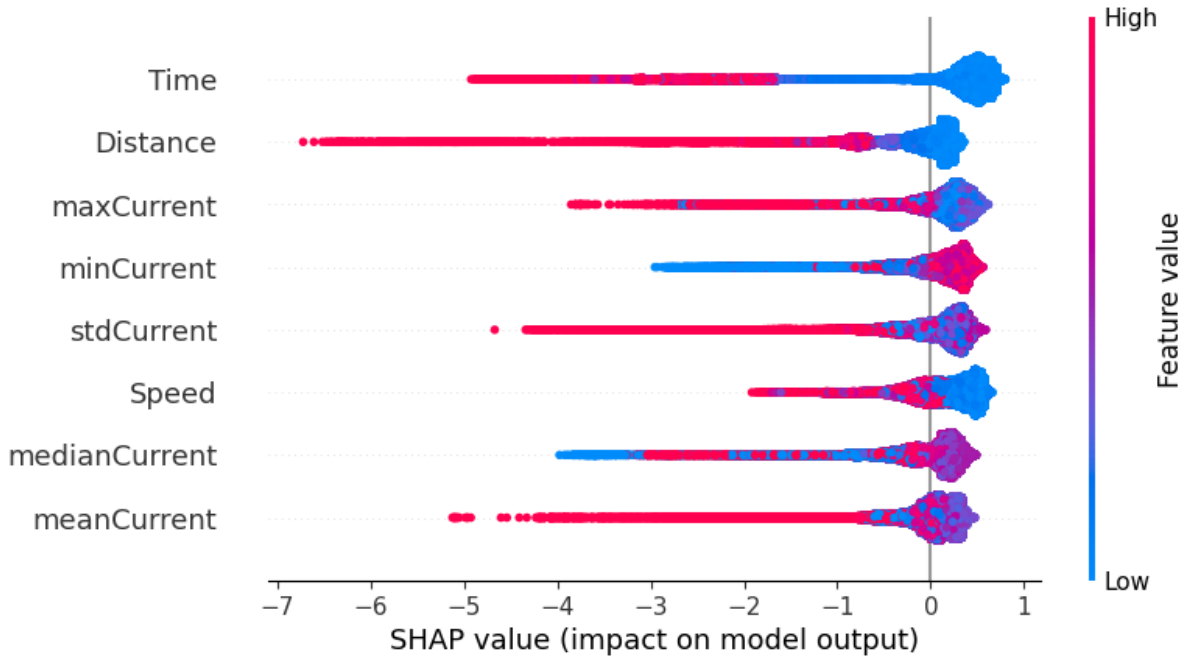


Figure 5.27: Isolation Forest SHAP Feature Importance

as a player in a collaborative game. SHAP values are calculated depending on the scores (both positive and negative) that each feature contributes to the final prediction. Features with positive SHAP values have a positive contribution to the model prediction and vice versa. The magnitude of the feature value tells us how impactful the feature is to the model i.e. a high value means a high impact and thus high importance. The results of each algorithm are discussed below.

1. **Isolation Forest**

   From the plot in 5.27, the top three important features according to Isolation Forest are Time, Distance and maxCurrent and they chiefly have a negative impact on the final prediction scores.

2. **Random Forest**

   As shown in 5.28 Speed, stdCurrent and Distance are the top three features of Random Forest. Speed has a mostly negative impact on the final prediction score

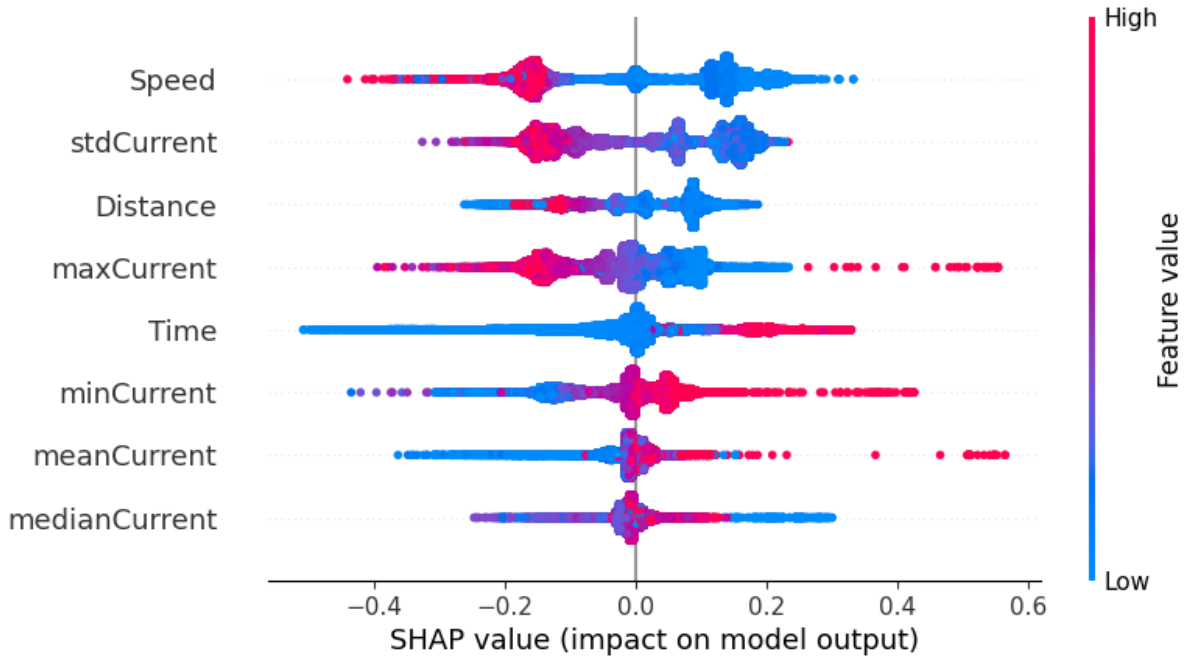


Figure 5.28: Random Forest SHAP Feature Importance

with the other two having an almost balanced influence on the prediction score. meanCurrent and median Current have the lowest influence on the prediction score.

3. **Logistic Regression**

   From the SHAP plot in 5.27 we can observe that stdCurrent, Time and Distance have the most influence on the prediction scores for Logistic Regression.

From our bivariate analysis, we observed that the anomalies were mostly clustered at the low end of speed, meaning that motors that were starting to have problems had a noticeable reduction in speed which agrees with the literature on motors. Secondly, from the feature importance we have just discussed, The top two performing algorithms, Random Forest and Logistic Regression, have stdCurrent and Distance in their top three features. It is also worth noting that Speed is the most important feature in the best-performing model, the Random Forest.

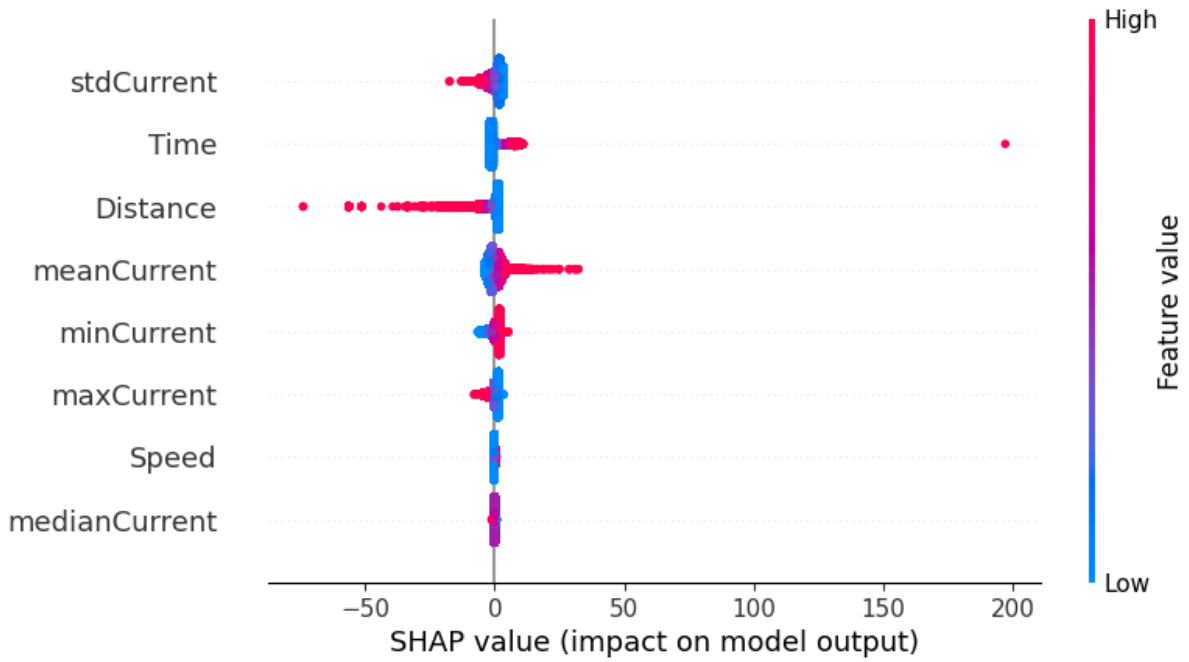


Figure 5.29: Logistic Regression SHAP Feature Importance

Chapter 6

Conclusion and Recommendation

This study demonstrated that machine debug log files of a medical linear accelerator can be mined to extract parameters that can then be used to develop an accurate motor breakdown prediction model.

The log files were found to contain several useful features as listed in table 3.2 and these features which contain data on the operation of the collimator motor were mined.

An ELT method that systematically cleans the unstructured log files and then loads them into the model for processing was also developed and implemented as a data pipeline.

By using the log files and service records a labelled training and testing dataset to train a model was successfully developed and used to train a Random Forest classifier that was then deployed for utilisation by service engineers.

Isolation Forest algorithm for anomaly detection, Random Forest and Logistic Regression were trained and tested on this data. The models were then evaluated using F1-Score, Precision, Recall and Accuracy with the pure classifiers emerging on top; Random Forest (Precision: 99.85%, Recall: 99.85%) then Logistic Regression (Precision: 99.1% and Recall: 93.83%) as illustrated in table 5.1.

6.1 Feature Importance

From the study findings, motor speed, distance moved by the motor and the standard deviation of the motor current were the most important features in predicting a motor failure as illustrated in figure 5.28.

According to the literature in 2.1.2, in a DC motor, the speed is directly proportional to the torque output. When the speed slows down, this is an indicator that there is a higher torque output 2.3 and since the load on the motor is not dynamic (unchanging) this could be an indicator of an obstruction, a lubrication issue or the beginning of some internal motor problem that ought to be investigated. This is a key finding that merits further investigation and data collection to improve the model performance and understanding further.

The bivariate analysis conducted as part of the EDA also pointed to this fact as shown in figures 5.15, 5.10, 5.11, 5.13, 5.9, 5.12 and 5.14. Here, the anomalies were clustered around where the speed was low.

The standard deviation of the current (stdCurrent) represents fluctuations in the motor current. This implies that fluctuations in the motor current, portend a motor issue. This is because the current drawn by the motor on a constant load should be fairly steady. When the fluctuations increase, it could be an indicator of a developing issue, much like flickering lights could be an issue with the electrical power in a house.

6.2 Strengths and Limitations of the Study

6.2.1 Strengths

The successful application of simple classification algorithms to this problem is a significant outcome of this study. Despite this being a complex anomaly detection problem, the Logistic Regression model, a simple binary classifier, did quite well in identifying normal data points from anomalies. The model of choice, Random Forest, did well on both training and test data sets. Both top-performing models also agreed on the most important features for predicting anomalies: motor speed, the standard deviation of current through the motor, and the distance moved by the motor. These findings also echo back to the bi-variate analysis which identified speed as an important factor in isolating anomalies.

6.2.2 Limitations

Despite the outstanding performance of the chosen model, more work can be done to improve it. More features like the age of the machine, the specific motor type, workload of the machine and many other specific features.

This study relies heavily on machine log files that must be uploaded to a central server every 24 hours. A break in this pipeline will mean that the model may not be run for the day or will give erroneous results.

Like every technology structure, designers will inevitably introduce changes to the machine log files as they improve the system. Such a change in the machine log file structure will affect the data ingestion pipeline and this will have to be planned for and the pipeline modified in case of such a change.

6.3 Recommendations and Future Works

This work can be further improved with additional features like the age of the machine, the specific motor characteristics and even a temporal aspect of the machine operations.

The model is exposed through an API thus it can be integrated within the company pipeline to consume data from the Splunk server for data ingestion. An online prediction system would then be possible, whereby log data could be streaming through the data pipeline in real-time and thus predictions can be made in real-time and alarms triggered in real time.

The model can also be deployed on the company cloud to serve all machines and engineers.

VMS should also consider designing the log file to collect more useful data in their future machine software designs to enable data scientists to develop more robust models thus leading to better outcomes for the company and customers.

References

1.10. Decision Trees. (n.d.). https://scikit-learn.org/stable/modules/tree.html

1.4. Support vector machines. (n.d.). https://scikit-learn.org/stable/modules/svm.html

1.6. Nearest neighbors. (n.d.). https://scikit-learn.org/stable/modules/neighbors.html

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., & Zhang, X. (2016). Tensorflow: A system for large-scale machine learning.

Able, C., Baydush, A. H., Nguyen, C., Gersh, J., Ndlovu, A., Rebo, I., Booth, J., Pérez, M. A., Sintay, B., & Munley, M. T. (2016). A model for preemptive maintenance of medical linear accelerators—predictive maintenance. *Radiation Oncology*, *11*(1). https://doi.org/10.1186/s13014-016-0602-1

Achouch, M., Dimitrova, M., Ziane, K., Karganroudi, S. S., Dhouib, R., Ibrahim, H., & Adda, M. (2022). On Predictive Maintenance in Industry 4.0: Overview, models, and challenges. *Applied sciences*, *12*(16), 8081. https://doi.org/10.3390/app12168081

Apache Spark™ - Unified Engine for large-scale data analytics. (n.d.). https://spark.apache.org/

Ayankoso, S. A., & Olejnik, P. (2023). Time-Series Machine Learning Techniques for Modeling and Identification of Mechatronic Systems with Friction: A Review and Real Application. *Electronics*, *12*(17), 3669. https://doi.org/10.3390/electronics12173669

Babenko, B. (2008). Multiple instance learning: Algorithms and applications.

Bansal, P., & Ouda, A. (2022). Study on integration of fastapi and machine learning for continuous authentication of behavioral biometrics. *2022 International Symposium on Networks, Computers and Communications (ISNCC)*, 1–6. https://doi.org/10.1109/ISNCC55209.2022.9851790

Breiman, L. (2001). Random Forests. *Machine Learning*, *45*(1), 5–32. https://doi.org/10.1023/a:1010933404324

Brown, B. H., Smallwood, R. H., Barber, D. C., Lawford, P., & Hose, D. (2017, September). *Medical Physics and Biomedical Engineering*. https://doi.org/10.1201/9781315275604

Built-in types. (n.d.). https://docs.python.org/3/library/stdtypes.html

Calabrese, M., Cimmino, M., Fiume, F., Manfrin, M., Romeo, L., Ceccacci, S., Paolanti, M., Toscano, G., Ciandrini, G., Carrotta, A., Mengoni, M., Frontoni, E., & Kapetis, D. (2020). SOPHIA: An Event-Based IoT and machine learning architecture for predictive maintenance in industry 4.0. *Information*, *11*(4), 202. https://doi.org/10.3390/info11040202

Cédola, A. P., Rossini, R., Bosi, I., & Conzon, D. (2021). Feature engineering and machine learning modelling for predictive maintenance based on production and stop events. https://api.semanticscholar.org/CorpusID:244727976

Chapman, P. (2000). Crisp-dm 1.0: Step-by-step data mining guide. https://api.semanticscholar.org/CorpusID:59777418

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling technique. *Journal of Artificial Intelligence Research*, *16*, 321–357. https://doi.org/10.1613/jair.953

Corporation, E.-C. (1977, January). *DC Motors, Speed Controls, Servo Systems*. Pergamon Press.

Dahiphale, D., Shinde, P. V., Patil, K., & Dahiphale, V. (2023). Smart Farming: Crop Recommendation using Machine Learning with Challenges and Future Ideas. *JOURNAL OF IEEE TRANSACTIONS ON ARTIFICIAL INTELLIGENCE*, *0*. https://doi.org/10.36227/techrxiv.23504496.v1

Data, G. (n.d.). Varian Medical Systems Inc Company Profile - Overview. https://www.globaldata.com/company-profile/varian-medical-systems-inc/

Distributed Random Forest (DRF) — H2O 3.44.0.3 documentation. (n.d.). https://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/drf.html

FastAPI. (n.d.). https://fastapi.tiangolo.com/

Fernandes, M., Canito, A., Bolón-Canedo, V., Conceição, L., Praça, I., & Marreiros, G. (2019). Data analysis and feature selection for predictive maintenance: A case-study in the metallurgic industry. *International Journal of Information Management*, *46*, 252–262. https://doi.org/10.1016/j.ijinfomgt.2018.10.006

Fleming, J. (1892). *The alternate current transformer in theory and practice*. "The Electrician" printing; publishing Company, limited. https://books.google.co.ke/books?id=17sKAAAAIAAJ

Greene, D., & Williams, P. C. (2017, August). *Linear accelerators for radiation therapy*. https://doi.org/10.1201/9780429246562

H2O.ai — The fastest, most accurate AI Cloud Platform. (n.d.). https://h2o.ai/

Hamaide, V., Joassin, D., Castin, L., & Glineur, F. (2022). A two-level machine learning framework for predictive maintenance: comparison of learning formulations. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2204.10083

Hanna, S. M. (2012). Rf linear accelerators for medical and industrial applications. https://api.semanticscholar.org/CorpusID:107055051

Hoisak, J., Kim, G.-Y. G., Atwood, T., & Pawlicki, T. (2021). Operational insights from the longitudinal analysis of a linear accelerator machine log. *Cureus*. https://doi.org/10.7759/cureus.16038

How are dc motors controlled? - speed control of dc motors — aspina. (2021, February). https://eu.aspina-group.com/en/learning-zone/columns/what-is/011/

Hughes, A., & Drury, B. (2013). *Electric motors and drives: Fundamentals, types and applications*. Elsevier Science. https://books.google.co.ke/books?id=jjuTYtKokc8C

Huyen, C. (2022, May). *Designing machine learning systems*. "O'Reilly Media, Inc.".

Kamat, P., & Sugandhi, R. (2020). Anomaly Detection for Predictive Maintenance in Industry 4.0- A survey. *E3S web of conferences*, *170*, 02007. https://doi.org/10.1051/e3sconf/202017002007

Kawahara, D., Nakano, H., Saito, A., Ochi, Y., & Nagata, Y. (2020). Formulation of objective indices to quantify machine failure risk analysis for interruptions in radiotherapy. *Journal of Applied Clinical Medical Physics*, *22*(1), 165–173. https://doi.org/10.1002/acm2.13126

Lei, Y., Li, N., Gontarz, S., Lin, J., Radkowski, S., & Dybała, J. (2016). A Model-Based method for remaining useful life prediction of machinery. *IEEE Transactions on Reliability*, *65*(3), 1314–1326. https://doi.org/10.1109/tr.2016.2570568

Liu, F., Ting, K. M., & Zhou, Z.-H. (2008). Isolation Forest. *2008 Eighth IEEE International Conference on Data Mining*. https://doi.org/10.1109/icdm.2008.17

Lundberg, S., & Lee, S.-I. (2017, May). A unified approach to interpreting model predictions. http://arxiv.org/abs/1705.07874

Ma, M., Liu, C., Wei, R., Liang, B., & Dai, J. (2022). Predicting machine's performance record using the stacked long short-term memory (LSTM) neural networks. *Journal of Applied Clinical Medical Physics*, *23*(3). https://doi.org/10.1002/acm2.13558

Mohan, N., Undeland, T., & Robbins, W. (2003). *Power electronics: Converters, applications, and design*. John Wiley & Sons. https://books.google.co.ke/books?id=kbUQgAACAAJ

Natekin, A., & Knoll, A. (2013). Gradient boosting machines, a tutorial. *Frontiers in neurorobotics*, *7*, 21. https://doi.org/10.3389/fnbot.2013.00021

of North America (RSNA), R. S., & of Radiology (ACR), A. C. (n.d.). Linear Accelerator. https://www.radiologyinfo.org/en/info/linac

P. S. Bimbhra, G. (n.d.). *Electrical machines-i*. KHANNA PUBLISHING HOUSE. https://books.google.co.ke/books?id=LmIEEAAAQBAJ

Pang, B., Nijkamp, E., & Wu, Y. (2019). Deep Learning with TensorFlow: a review. *Journal of Educational and Behavioral Statistics*, *45*(2), 227–248. https://doi.org/10.3102/1076998619872761

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., ... Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. https://api.semanticscholar.org/CorpusID:202786778

Peng, C.-Y. J., Lee, K. L., & Ingersoll, G. M. (2002). An introduction to logistic regression analysis and reporting. *The Journal of Educational Research*, *96*(1), 3–14. https://doi.org/10.1080/00220670209598786

Pietro Giovanni Antiga, L., Stevens, E., & Viehmann, T. (2020, July). *Deep Learning with PyTorch*. Simon; Schuster.

Prytz, R., Nowaczyk, S., Rögnvaldsson, T., & Byttner, S. (2015). Predicting the need for vehicle compressor repairs using maintenance records and logged vehicle data. *Engineering Applications of Artificial Intelligence*, *41*, 139–150. https://doi.org/10.1016/j.engappai.2015.02.009

Richardson, D. (1982). *Rotating electric machinery and transformer technology*. Reston Publishing Company. https://books.google.co.ke/books?id=eexSAAAAMAAJ

SamuelA, L. (1959). Some studies in machine learning using the game of checkers. *IBM journal of research and development*, *3*(3), 210–229. https://doi.org/10.1147/rd.33.0210

Sipoš, R., Fradkin, D., Moerchen, F., & Wang, Z. (2014). Log-based predictive maintenance. *KDD '14: The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. https://doi.org/10.1145/2623330.2623340

Splunk — The key to enterprise resilience. (n.d.). https://www.splunk.com/

SQLite Documentation. (n.d.). https://www.sqlite.org/docs.html

Standard, E. (2001). Maintenance Terminology (Standard EN 13306:2001). https://standards.iteh.ai/catalog/standards/cen/02bfab80-f6f0-4f69-9cad-49b3dc70923f/en-13306-2001

Streamlit • A faster way to build and share data apps. (n.d.). https://streamlit.io/

Team, K. (n.d.). *Keras: Deep Learning for humans*. https://keras.io/

TensorFlow. (n.d.). TensorFlow. https://www.tensorflow.org/

Tupkar, R. S., Lande, M., Jawalekar, S. B., & Prof, A. (2020). Implementation of Lean Six Sigma. *ResearchGate*. https://www.researchgate.net/publication/339738274_Implementation_of_Lean_Six_Sigma

Voron, F. (2021, October). *Building Data Science Applications with FastAPI*. Packt Publishing Ltd.

Welcome to LightGBM's documentation! — LightGBM 4.0.0 documentation. (n.d.). https://lightgbm.readthedocs.io/en/stable/

XGBoost Documentation — xgboost 2.0.3 documentation. (n.d.). https://xgboost.readthedocs.io/en/stable/

Appendix A

Ethical Clearance Confirmation

Strathmore UNIVERSITY

16 March 2024

Dear Mr. Allan Koech

RE: MACHINE LEARNING MODEL FOR PREDICTIVE MAINTENANCE ON LINEAR ACCELERATORS

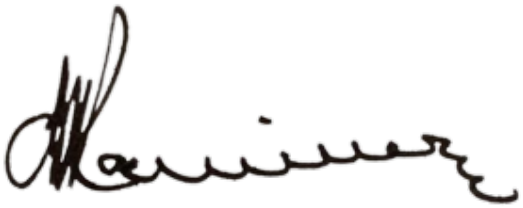
This is to inform you that SU-ISERC has reviewed and Approved your above research proposal. Your application reference number is SU-ISERC2037/24. The approval period is valid for exactly **one year** from today.

This approval is subject to compliance with the following requirements:

1. Only approved documents including (informed consents, study instruments, MTA) will be used.
2. All changes including (amendments, deviations, and violations) are submitted for review and approval by SU-ISERC.
3. Death and life-threatening problems and serious adverse events or unexpected adverse events whether related or unrelated to the study must be reported to SU-ISERC within 72 hours of notification.
4. Any changes anticipated or otherwise that may increase the risks or affected safety or welfare of study participants and others or affect the integrity of the research must be reported to SU-ISERC within 72 hours.
5. Clearance for the export of biological specimens must be obtained from relevant institutions.
6. Submission of a request for renewal of approval at least 60 days prior to the expiry of the approval period. Attach a comprehensive progress report to support the renewal.
7. Submission of an executive summary report within 90 days of completion of the study to SU-ISERC.

Before commencing your study, you will be expected to obtain a research license from the National Commission for Science, Technology, and Innovation (NACOSTI) https://research-portal.nacosti.go.ke/ and obtain other clearances needed.

Yours sincerely,



Mr Ambrose Rachier,

Chairperson; SU-ISERC

Strathmore
UNIVERSITY

Appendix B

API Documentation

```
{
    "openapi": "3.1.0",
    "info": {
        "title": "FastAPI",
        "version": "0.1.0"
    },
    "paths": {
        "/clean": {
            "post": {
                "summary": "Clean Data",
                "description": "endpoint to clean raw log files and return clean
data",
                "operationId": "clean_data_clean_post",
                "responses": {
                    "200": {
                        "description": "Successful Response",
                        "content": {
                            "application/json": {
                                "schema": {}
                            }
                        }
                    }
                }
            }
        },
        "/prediction": {
            "post": {
                "summary": "Prediction Endpoint",
                "description": "Function to read json of features, run the predict
model and save the\nfeatures and results into the db.",
                "operationId": "prediction_endpoint_prediction_post",
                "requestBody": {
                    "content": {
                        "application/json": {
                            "schema": {
                                "$ref": "#/components/schemas/FeaturesItem"
                            }
                        }
                    },
                    "required": true
                },
                "responses": {
                    "200": {
                        "description": "Successful Response",
                        "content": {
                            "application/json": {
                                "schema": {}
                            }
                        }
                    },
```

```
                    "422": {
                        "description": "Validation Error",
                        "content": {
                            "application/json": {
                                "schema": {
                                    "$ref":
"#/components/schemas/HTTPValidationError"
                                }
                            }
                        }
                    }
                }
            }
        },
        "/raw_data": {
            "post": {
                "summary": "Get Csv Data",
                "description": "Function to read csv of raw machine log file
and\nreturn it as a json file",
                "operationId": "get_csv_data_raw_data_post",
                "responses": {
                    "200": {
                        "description": "Successful Response",
                        "content": {
                            "application/json": {
                                "schema": {}
                            }
                        }
                    }
                }
            }
        },
        "/db_data": {
            "post": {
                "summary": "Get Db Data",
                "description": "Function to read and retrieve db data",
                "operationId": "get_db_data_db_data_post",
                "responses": {
                    "200": {
                        "description": "Successful Response",
                        "content": {
                            "application/json": {
                                "schema": {}
                            }
                        }
                    }
                }
            }
        }
    },
```

```
"components": {
    "schemas": {
        "FeaturesItem": {
            "properties": {
                "minCurrent": {
                    "type": "number",
                    "title": "Mincurrent"
                },
                "maxCurrent": {
                    "type": "number",
                    "title": "Maxcurrent"
                },
                "stdCurrent": {
                    "type": "number",
                    "title": "Stdcurrent"
                },
                "Serial_No": {
                    "type": "integer",
                    "title": "Serial No"
                },
                "meanCurrent": {
                    "type": "number",
                    "title": "Meancurrent"
                },
                "medianCurrent": {
                    "type": "number",
                    "title": "Mediancurrent"
                },
                "Time": {
                    "type": "number",
                    "title": "Time"
                },
                "Distance": {
                    "type": "number",
                    "title": "Distance"
                },
                "Speed": {
                    "type": "number",
                    "title": "Speed"
                }
            },
            "type": "object",
            "required": [
                "minCurrent",
                "maxCurrent",
                "stdCurrent",
                "Serial_No",
                "meanCurrent",
                "medianCurrent",
                "Time",
```

```
        "Distance",
        "Speed"
    ],
    "title": "FeaturesItem",
    "description": "Data model for the prediciton features"
},
"HTTPValidationError": {
    "properties": {
        "detail": {
            "items": {
                "$ref": "#/components/schemas/ValidationError"
            },
            "type": "array",
            "title": "Detail"
        }
    },
    "type": "object",
    "title": "HTTPValidationError"
},
"ValidationError": {
    "properties": {
        "loc": {
            "items": {
                "anyOf": [
                    {
                        "type": "string"
                    },
                    {
                        "type": "integer"
                    }
                ]
            },
            "type": "array",
            "title": "Location"
        },
        "msg": {
            "type": "string",
            "title": "Message"
        },
        "type": {
            "type": "string",
            "title": "Error Type"
        }
    },
    "type": "object",
    "required": [
        "loc",
        "msg",
        "type"
    ],
```

```
                "title": "ValidationError"
            }
        }
    }
}
```

Appendix C

Plagiarism Report

Figure C.1: Turnitin Report