Weather-induced Asset Failure Prediction

MSc Data Science

School of Science, Engineering and Environment

The University of Salford, Manchester

MSc Project Dissertation

Author: Ambareesh Jonnavittula

Supervisor: Professor Judita Preiss

May 2022

ABSTRACT

National Highways is a government-owned, publicly funded company that are responsible for managing and improving England's Strategic Road Network (SRN) of motorways and major A roads. They plan, design, build, operate and maintain the 4,300 miles of roads, helping their customers have safer, smoother and more reliable journeys. Their business imperatives are Safety, Customers and Delivery. The objective of the project is to be able to predict the impact of rainfall and temperature on the strategic road network on a per asset basis. The increasing trend in flood incidents and road accidents is concerning, partially due to the apparent association with the quality of gully maintenance, but also the pressure on Highways England group. Having predictive abilities to classify flood risk hotspots is highly valuable especially considering the public safety and wellness in which various machine learning classification methods can be applied to for pattern detection and obtaining knowledge. The proposed models will aim to support asset managers in data-driven decision-making resulting in well-timed strategic interventions, optimized maintenance planning, resource allocation, and decreased maintenance costs. This research focuses on the implementation of models such as Logistic Regression, Decision Tree, and Isolation Forest to predict high-risk assets that have recurring maintenance jobs performed. The dataset was highly imbalanced and three balancing methods were applied; Oversampling, Undersampling, and Combined over- and under- sampling. Evaluation of the model to determine the best-performing algorithm was centered on the confusion matrix, F1 score, and AUC as selected metrics. The logistic regression algorithm delivered a high performance, and the combined method was the best way to solve the imbalanced data with an accuracy of 85%, F1 score of 0.716, and AUC of 0.785. The recurrent trend of maintenance jobs was observed over a span of 2 years on each month using time series graphs such as interactive line charts and bar charts.

Keywords: Machine Learning, Data Mining, Predictive Modelling, Decision Tree, Logistic Regression, Isolation Forest, Anomaly detection, National Highways, Time Series Analysis

ACKNOWLEDGMENTS

My deepest gratitude to my loving parents for their continual prayers and love which has got me to where I am today. Special thanks to my sister Roopini, who has been very supportive during this time.

Secondly, I would like to express my profound appreciation to my research supervisor, Professor Judita Preiss for her guidance and encouragement and Ms. Lucy Lynch from Eden Smith who gave me the opportunity to do a project with the National Highways - England. I would like to take this opportunity to express my gratitude to all the participants from the Highways' authorities and operators who so generously offered their time and insights to this project.

Finally, I thank my friends Surekha, Helen, Martin and Vadims who have helped and motivated me throughout my journey at the university.

TABLE OF CONTENTS

Contents

LIST OF FIGURES

LIST OF TABLES

LIST OF ABRRIEVIATIONS

AADT	Annual Average Daily Traffic (on a highway)
ADMM	Asset Data Management Manual
ADT	Average Daily Traffic
CDM	Conceptual Data model
CRISP-DM	Cross-Industry Standard Process for Data Mining
DaaS	Data-as-a-Service
DDaT	Digital, data and technology
DDMS	Drainage Data Management System
DfT	Department for Transport
DLE	Drainage Liaison Engineer
DPIA	Data Privacy Impact Assessment
ECHO	Every Customer Has an Opinion
HE	Highways England
HGV	Heavy Goods Vehicle
ILM	Information Lifecycle Management
ILPM	Intelligence-led Predictive Maintenance
KSI	Kills and Serious Injuries
LDM	Logical Data Model
LGV	Light Goods Vehicle
MIDAS	Met Office Integrated Data Archive System (MIDAS)
MTBF	Mean Time between Failures
NH	National Highways
NTIS	NH Transport Information System
ODBC	Open Database Connectivity
OWW	Optimized Working Windows
PDM	Physical Data model

RIS	Road Investment Strategy
RTA	Road Traffic Accident
SBP	Strategic Business Plan
SRN	Strategic Road Network
SWIS	Severe Weather Information Services
VM	Virtual Machine

Chapter 1 INTRODUCTION

1.1 Introduction

National Highways (NH) are the highway, traffic and street authority for England's motorways and certain major A-roads known as the strategic road network (SRN). They are a government-owned, arm's-length company delivering and contributing to the government's long-term plan for the SRN. To enable road safety on the Strategic Road Network (SRN), the National Highways group conducts various surveys, campaigns and conduct research that provides environmental, social, and economic benefits to the people, communities and businesses who live and work alongside our strategic road network. The Infrastructure Act 2015 sets out the overall governance framework for NH, including the Road Investment Strategy (RIS) and the Department for Transport's (DfT) Framework Document sets out the NH roles and accountabilities. There are more than 21,500 lane-miles of road and more than 21,000 structures as per the Asset management policy (2020). Assets worth £132.5 billion were managed as of FY 2020-21, for which £ 0.8 billion was allocated to renew the assets including maintenance in the same fiscal year period. NH risk assessment aims towards imperatives like customer, delivery, and safety. The project of identifying gully failures based on the weather is to focus on the safety as an imperative and the expected strategic outcome would be a well-maintained resilient network. There is a safety risk involved here since a significant asset failure could result in a major incident, leading to death, injury, or vehicle/property damage. The incident reporting process, maintenance and inspection regime are maintained by Asset data management team and the Drainage Liaison Engineer (DLE) acts as the focal point to maintain and ensure compliance with all Drainage / Flood data management specifications relevant to managing the drainage asset in accordance with CD 535 and CS 531.

Weather conditions can vary considerably throughout the year and can sometimes change at short notice. Rainfall can affect your ability to see ahead or be seen by others. It can also affect your vehicle's performance and responsiveness. High winds, either on their own or

accompanying heavy rain as part of a storm, can be hazardous to road users. One may even encounter debris from fallen trees, branches or other items blown by winds. Gusts of wind can cause your vehicle to shake. A red claim can be made where the public can allege that the injury or damage resulted from the failure to maintain the SRN properly. To defend such a claim, NH needs to prove that all reasonable measures were taken to ensure the highway was not dangerous to traffic or pedestrians.

Many industries have adopted predictive modelling by advancing technologies, allowing businesses to predict the future and successfully solve problems. The exponential growth of data and rapid technological change is causing corporations to adapt machine learning algorithms and data mining methods to identify correlation in forecasts based on previous occurrences (Mukherjee, 2019). The potential benefits of machine learning must be balanced against any risk to National Highways information or losing the trust of the public. Any machine learning done using National Highways information, either by National Highways or a National Highways Supplier, must reflect National Highways corporate values and meet National Highways information ethics requirement. Predicting / forecasting avoidable maintenances with the use of supervised learning algorithms will optimize operations and minimize potential risks. In addition, an unsupervised learning algorithm would enable the outlier / anomaly detection. This ensures overall cost benefit, value-added safety and enables data-driven decision-making rather than relying on intuition / observation. Additional benefits include resource efficiency, and re-investment of retained funds to help the public or handle the red claims. Haversine formula is applied to solve the distance between two locations in England. Time series analysis is another statistical technique in which significant data is explored to discover trends and seasonality patterns over a timeframe. Line graphs (static and interactive) would be applied to summarize the time series data which will unveil patterns of the most frequent times of the week, month, and year when gullies were maintained.

The goal of this project with National Highways Asset management team is to address the

issues with ensuring better safety in terms of flooding prevention and overall maintenance handling based on weather changes and forecast. Currently, the NH work with a flooding hotspot tool and arbitrary weather warnings to determine what preventative measures should be taken. By applying this model on gully failure data, the NH asset management team would develop an understanding if the data within their reach could indeed be used as a resource to improve the public safety and use a variety of indicators to incorporate systematic and efficient maintenance strategies.

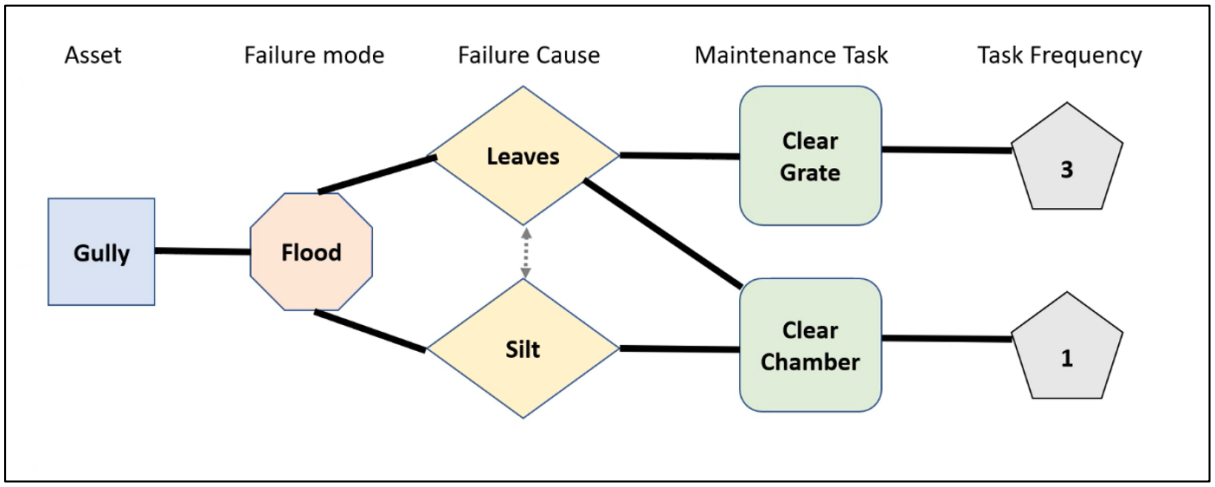
1.2 Problem Statement

The NH face concerns in relation to flooding, clogging, recurring maintenance on gullies causing financial loss due to redundant jobs. Gullies are small chambers that form the top part of the surface water drainage system. Rainwater from the road surface flows through the gratings into the gullies and then into the underground drainage system. A gully is essentially designed to discharge surface water from the SRN. Anyone who notices a gully could report the hazard to the local authorities. There are 2 types of gully maintenances – planned and unplanned. A planned maintenance happens periodically based on an agreed schedule and job costs are allocated accordingly, whereas an unplanned maintenance is commonly the result of equipment failure that was not anticipated. It happens when there is no formal strategy in place to address a repair, replacement, or inspection before it's needed. Some examples to this could include heavy rainfall and floods, human errors, etc.

Gully emptying and clearing of drainage systems includes grips, ditches, gullies, and other systems. The quick and effective drainage of surface water contributes significantly to the safety of a road. To prevent, as far as possible, standing water or flooding of a road, the grips and highway ditches are cleaned of vegetation and dug out when required. A blocked road gully is cleared using either powerful suction equipment, or a drain jet. If the problem is a buildup of leaves, silt and water-borne debris, the metal grid is removed, and the blockage is sucked out of the pipe. A gully could get clogged almost 99% of the time either by accumulated silt or by dried leaves, and both can equally pose an imminent threat to the safety of the

highway users and passengers. Different strategies are applied to handle these two problems based on a varied frequency.

Figure 1.1 – Gully Failure task frequency & cause (National Highways)



To address this problem, scientifically focused, dynamic, and widely approved approaches are dependent on cooperation between asset management group, drainage liaison engineering team, and weather data maintenance teams. The National Highways are aiming to decrease the rate of maintenance within the asset maintenance regime for gullies due to an increasing number of jobs on gullies being performed in the last few years. A model, based on predicting gully failure which were expected to eventually have a maintenance, was effectively built, and expected to be used in the areas of elicitation and vulnerabilities. The model will promptly collect and analyze the dataset to identify gully jobs and failures, outliers based on flood severity, cyclical frequency necessary to perform jobs, patterns, and factors that are the underlying cause of maintenance.

Further questions have also been posed regarding the increase in maintenance jobs in the last few years across the SRN. The concern that emerges is whether there is a common category of failure modes that are recurring, all with related weather impact. This will clarify why there is a significant spike in maintenance deficit. The data collected contains information on asset conditions during a maintenance job, failure mode, asset location carried out by asset

management team, flood hotspots, status and flood severity indices provided by drainage liaison engineers, weather parameters like rainfall intensity, wind pressure, wind direction, air temperature, wind gust and direction provided by the SWIS meteorologists, and an open-sourced elevation based on the post code. A model is to be built to uncover a specific category of failures which are recurring that represent substantial volumes or identify a pattern that ascertains the threshold of weather parameters that could potentially cause an asset failure. The results produced would offer forecasts in the probability that the category of failures would be more likely to occur again therefore, minimize the red claims, the wastage of resources like time and energy, and the financial penalties.

The data mining techniques chosen will enable asset managers for better decision-making and define a threshold-based prediction. An effective algorithm is required to generate the optimal predictions with minimal variance. This research will focus on finding a solution to the stated problems by utilizing various advanced machine learning algorithms to select a high and accurate performance model and allow improvements for the output to be reliable. The resulting model will result in a significant decline in recurring maintenance activities.

1.3 Aim of Study

The aim of this study is to construct and build a model that will predict and highlight possible high-risk gully failures that have recurring clogging and thus, promote safety by preventing flooding and reduce overall costs by optimizing future maintenance.

The historical data will be collected from NH Azure DaaS and SWIS systems. The research will include machine learning methods to discover patterns in a specific group of gully failure modes that have a higher probability of failure and thus recurring upkeep, produce accurate predictions to identify what kind of impact would weather have on the type of gully failure, also check if we could classify frequently maintained gullies, and analyze trends that can arise in a collection of failures that are classed in a similar failure type. The research also aims to decrease rates of repetitive maintenance by synthesizing forecasts for Drainage Liaison

Engineers to provide timely care and minimize excessive costs.

1.4 Objectives of Study

This research underlined the following objectives to be accomplished:

1. To explore and analyze the given data for pre-processing and modelling.
2. To conduct a literature review that will enhance the interpretation of the results.
3. To develop and build a model which highlights gully failures with the relevant floods.
4. To discover anomalies within the floods in relation to the asset failures.
5. To identify and determine times when maintenances arise using time-series visualizations.
6. To classify the data into multiple stages of gully clogging based on the weather data.

1.5 Tools

Python is an open source, interpreted, high level programming language providing great functionality to deal with mathematics, statistics, and scientific function. One of the main reasons why Python is widely used in the scientific and research communities is because of its ease of use and simple syntax. It is a widely known language for statistical computing techniques such as classification, clustering, regression, time-series analysis, linear and nonlinear modelling. The language's high extensibility and versatility allows users to develop their own software, form numerous functions, and extend via libraries, hence increasing its popularity within the community. The worldwide network of users can access Python as a free software, and allows the developer to run the code anywhere, including Windows, Mac OS X, UNIX, and Linux. In addition, it is also simple to extend the code by appending new modules that are implemented in other compiled language like C++ or C. The data analysis and modelling will be implemented using Python for this project. This programming language has been chosen for this project as it had numerous advantages that set it apart from other data analytics language and is user-friendly.

SQL is another programming language used in Microsoft SQL Server, and we use Azure SQL – Data-as-a-service (DaaS) which is the data warehouse, to collect and retrieve all the assets, jobs, and floods data for predictive modelling. Data-as-a-service (DaaS) is a data management strategy and/or deployment model that focuses on the cloud (public or private) to deliver a variety of data-related services such as storage, processing, and analytics.

Azure Data Studio is a cross-platform database tool for data professionals using on-premises and cloud data platforms on Windows, macOS, and Linux. Azure Data Studio offers a modern editor experience with IntelliSense, code snippets, source control integration, and an integrated terminal. In addition, an interactive Chart Viewer called SandDance which is downloadable from extensions, import and export options, schema comparison, Hadoop & PostgreSQL connectivity, and numerous other features are offered.

In addition, Microsoft Power BI was also utilized to work with spatial analysis, since the notebooks were not in trusted format from within the Highways systems, so the maps couldn't be generated using python script and modules like folium, etc.

Lastly, Azure Virtual Machine allows you to flexibly run virtualized environments with the ability to scale on-demand. Since we are going to deal with big data like weather reports, it is essential to have performance scaling to deal with large datasets.

1.6 Outline of Thesis

Chapter 1 provides an outline of the study, while Chapter 2 discusses the principles of other previous research as a literature review in relation to this NH based project with a detailed summary of the approaches, methods, algorithms, and strategies implemented in their work. Chapter 3 describes the types of research methodology and the reasoning of the selected method for applying to this research. Chapter 4 addresses the rationale of the business perspective and explores data understanding of the jobs and floods datasets provided by NH. Additionally, this chapter explains data pre-processing, its steps on transforming the acquired data to the correct format and visualizations for better understanding. Chapter 5 defines the

application of the selected algorithms and how the approaches produce solutions to satisfy the problem statement. Chapter 6 analyses the evaluation of the algorithms applied and facilitates findings. Furthermore, this chapter discusses any further research that could be conducted to minimize the rate of recurring maintenances and concludes this study.

Figure 1.2 – Project Overview

1.7 Summary

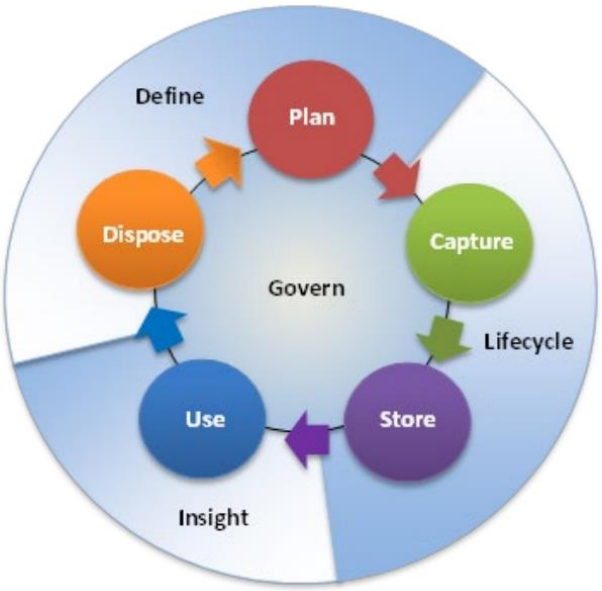
This chapter unveiled the context and purpose to be explored by this study. The aims and objective and the problem overview were defined to provide an outlook of the research. The tools utilized for developing a model has been stated and, finally, an outline of the thesis has been illustrated to be applied in this research. The next chapter will focus on the statistical and theoretical background and a review of existing literature.

Chapter 2 BACKGROUND

2.1 Introduction

The Asset Data Management Manual (ADMM) sets out Highways England's asset data requirements to achieve both its corporate objectives as well as its asset management objectives. The ADMM contains the company's asset data requirements to ensure the company collects and maintains the asset data it needs to operate safely and efficiently. It is for use by anyone creating, maintaining, or using data on behalf of or within Highways England. Good asset data management can be achieved when there is a clear strategy for the control of each stage of the life cycle.

Figure 2.1 – The Asset Data Lifecycle (ADMM)



The five stages can be briefly explained as below:

1. Plan – prior to the capture of data, it is important to look into aspects like resources required to manage data through its life cycle which is inclusive of funding as well. In addition, what systems are required for data storage and manipulation, contingency, and redundancy plan.

2. Capture – It involves collection and augmentation of asset datasets by collecting new data through a survey or records, converting, or transforming existing asset data, sharing, or exchanging data with partners, and finally purchasing data from a third-party supplier.

3. Store - Ensures that the data is stored in the right place, appropriate measures are taken to

minimize the risk of data loss and there is a clear policy for its disposal.

4. Use - The ability to prepare and issue, or disseminate, quality data to stakeholders is an important part of the life cycle process. Data needs to be used but with controls to protect against organization, legislative and commercial risk and to preserve the integrity of the data itself. Asset data should not be used unless it is accompanied by appropriate and relevant metadata.

5. Dispose - Only asset data that is generating value needs to be retained. Disposal includes archiving of historical information or deleting of redundant information. Appropriate disposal procedures reduce the organization risk by eliminating superseded, obsolete, or redundant data.

2.2 Data Mining in the Highway Industry

The potential benefits of machine learning must be balanced against any risk to National Highways information or losing the trust of the public. Any machine learning done using National Highways information, either by National Highways or a National Highways Supplier, must reflect National Highways corporate values and meet National Highways information ethics requirement. Applying data mining in the highway industry means this will ensure aspects like ethics, legal framework, auditability, interoperability and portability, privacy, and data risk management to improve the public safety and wellbeing. However, by collecting and analyzing public and private sector data, government data mining can identify potential terrorists or other dangerous activities by unknown individuals. However, this capability continues to raise concerns for private citizens when it comes to the government's access to personal data.

Road traffic accident (RTA) is a major but neglected public health challenge that requires concerted efforts for effective and sustainable prevention. According to a survey, it is estimated that there are 1.2 million deaths and 50 million injuries in road crashes worldwide each year [1]. In UK, it is reported that from 2017 to 2018, there were 1770 road deaths and 26,610 people killed or seriously injured in traffic accident, resulting in totally 165,100 casualties [2]. So new technologies and ways need to devise to uncover accident-relate factors and to

identify time and space that are risky, thus supporting traffic accident data analysis in decision-making processes. In this paper, we proposed a novel big data analytics platform for UK traffic accident analysis using data mining and deep learning techniques. We first displayed the data on an interactive map, which will effectively highlight accident hotspot in accordance with time and space. Then we visualized some attributes in the data to find key factors that related to traffic accident. Finally, we utilized state-of-the-art time series and deep learning algorithms to forecast accident in the future.

Various studies have addressed the different aspects of RTA, while most of which focus on using machine learning [3] and data mining [4] techniques to analyze traffic accidents. Kumar et al. [5] used cluster methods to identify high-frequently accident areas and then applied association rules to uncover factors that influence road accidents at those locations. Shanthi et al. [6] carried out gender-based classification, which RndTree and C4.5 using AdaBoost Meta classifier were utilized to obtain road accident patterns. Chen et al. [7] proposed a procedure which evaluates clusters of traffic accident and organized them according to their significance, thus identified high-risk locations (hotspot). Kumar et al. [8] proposed a data mining framework to analyze road accident, k-modes clustering and association rules are used to discover potential patterns and trends. In Xi et al. [9], association rules are used to categorize accidental factors and analyzed the degree of an accident or the level of influence. Park et al. [10] utilized Hadoop framework to handle large data set and built a predictive model to settle data imbalance problem. Shahrokh et al. [11] assessed trends of traffic accidents and then explored time series models to forecast them in the next years. He et al. [12] applied BP neural network to analyze the relationship between accident evaluating index and forecast the number of accidents in the future. Chee et al. [13] improved hybrid artificial neural network to visualize multidimensional data without increasing the number of neurons. Bakir et al. [14] provided a robust forecasting model to predict phone prices in European markets using Long Short-Term Memory (LSTM) neural network and Support Vector Regression (SVR).

Data mining holds great potential and has contributed to developments in highway industry and various other government-led projects, for instance dynamic energy management, smart grid security and theft detection, preventive equipment maintenance, demand response management, etc. As novel problems like climate change rise, further technologies within data mining will have a significant effect to discovering ways of improving operational efficiency, asset performance, and enhance the customer / public experience.

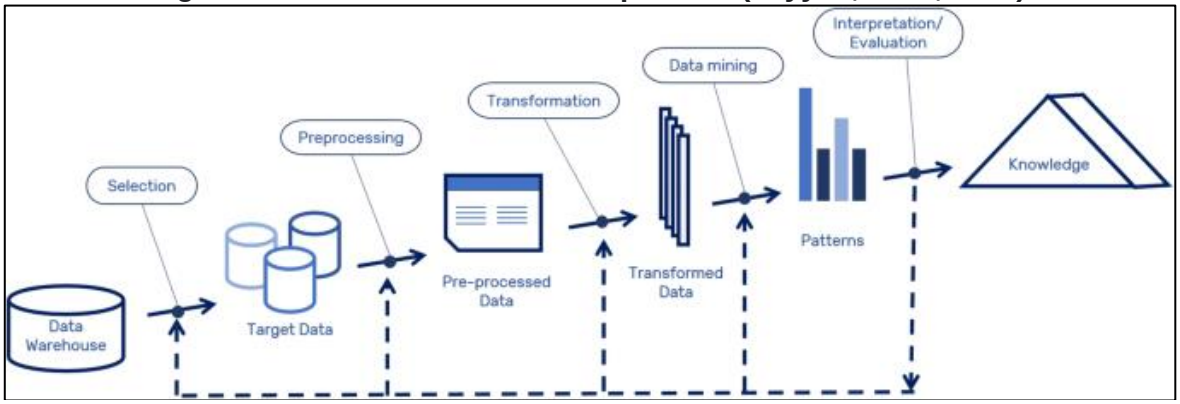
2.3 Knowledge Discovery in Databases

Knowledge Discovery in Databases (KDD) is a widely applied method to find and extract valuable insights from databases (Jackson, 2002). The method requires pre-existing knowledge and an overview of the program context. The KDD model is illustrated in Figure 2.3. There are nine stages of the method that are listed below:

1. **Application Domain Understanding:** The first stage comprises of developing knowledge of the application domain, pre-existing information, and objectives of user.
2. **Data Selection:** The targeted data stored in the data warehouse must be selected based on the intended goal of knowledge discovery. Extracting all variables relevant for the analysis to create a data set.
3. **Data Cleansing and Pre-processing:** The target data is cleaned by eliminating all missing values, noise, and outliers.
4. **Data Transformation:** Transforming the target data in the correct format for effective algorithm implementation. Transformation involves reducing irrelevant variables that do not apply in the task and normalization of the dataset.

5. **Data Mining Method Selection:** The data mining method is dependent on what is required for the goal and discovery. Example methods include classification, regression, clustering and so on.
6. **Data Mining Algorithm Selection:** Suitable algorithms and models are to be selected to discover patterns in the data.
7. **Data Mining Algorithm Employment:** The selected algorithm is implemented on the target dataset.
8. **Data Mined Patterns Evaluation:** Interpreting and evaluate results of the algorithm. Identifying and visualizing patterns of interest.
9. **Knowledge Discovery Analysis:** The extracted knowledge from the data is visualized for users can be utilized for various purposes such as to uncover the information of interest, meet the goals and objectives stated by user and decision-making.

Figure 2.2 – Overview of the KDD process (Fayyad, et al., 1996)

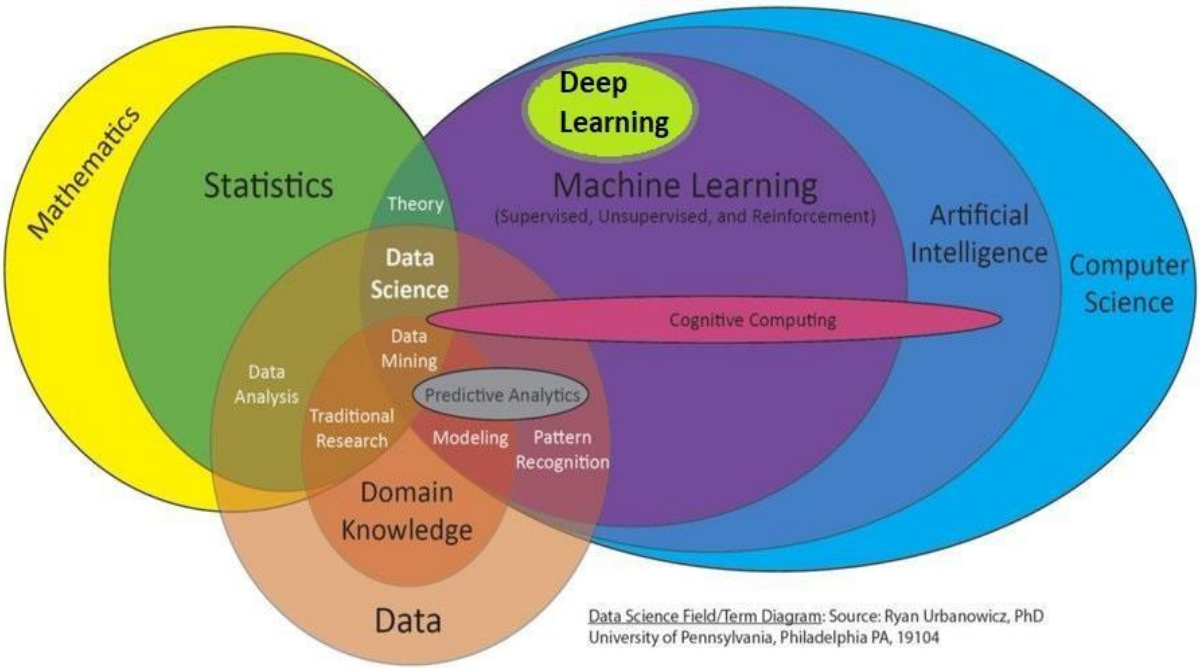


2.4 Machine Learning

Machine learning is the programming of machines to refine the success criteria using historical data to make predictions. The algorithms are programmed into the computer to build a predefined mathematical model based on recorded data, called the training data, to create

forecasts. The study of machine learning also discovers systems ability to learn, adapt and improve their performance without assistance or programming (Alpaydin, 2020). Application of the approaches are constructed upon the foundations of three research objectives: task-oriented, cognitive simulation and theoretical analysis. Task-oriented is based on creating and evaluating learning models to enhance efficiency in a specified series of tasks. Cognitive simulation is studying and machine modelling of human modelling systems. Theoretical analysis is the scientific overview of the array of potential learning algorithms beyond the application domain (Carbonell, Michalski, and Mitchell, 2014). One indicator of success in state-of-the-art machine learning is its major implementations in the modern world. The exponential advancements in big data and application of machine learning methods are the rising areas of science and technology having significant influences in various fields such as finance, bioinformatics, fraud detection, energy load forecasting, image processing, spam e-mail recognition and more (Alpaydin, 2014).

Figure 2.3 – Relationship between Machine Learning and Components of Data Science



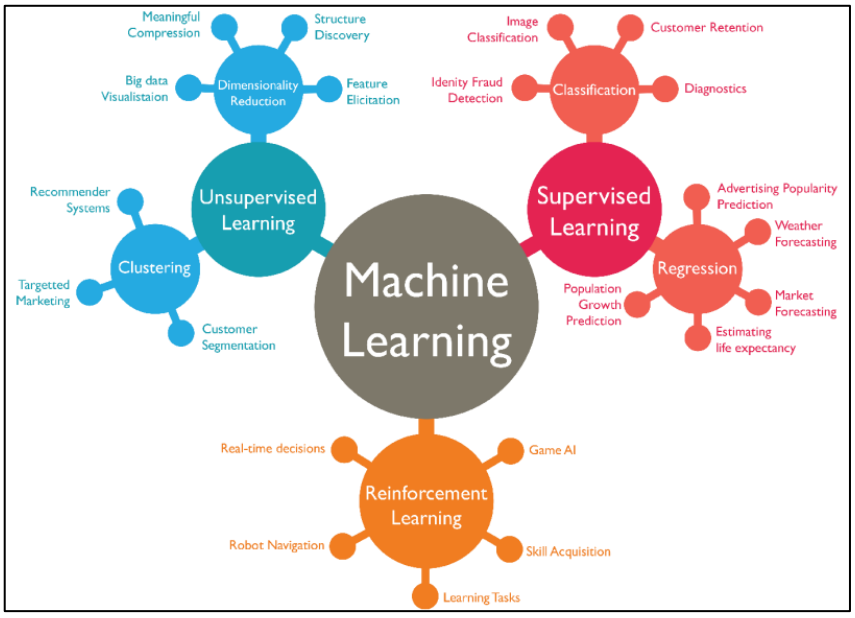
Furthermore, machine learning approaches serve a vital position in the field of computer science in a significant and increasing niche (Mitchell, 2006). It is a major domain of research that overlaps and derives concepts from a variety of related disciplines, such as artificial intelligence, data mining, and statistics. Artificial intelligence is the theory of machines programmed to execute processes that usually involve human intellect, such as sensory awareness and decision-making that require probabilistic thinking to achieve the targets effectively (Bini, 2018).

Implementing machine learning techniques on large databases is called data mining. Figure 2.4 illustrates the relationship between machine learning and the main components and fields of science and technology. Machine learning is the subfield of artificial intelligence, and deep learning is the subfield of machine learning. This study will focus on supervised machine learning methods to apply on real life data.

2.4.1 Types of Machine Learning Algorithms

Machine learning techniques are categorized into a taxonomy dependent on the intended result of the technique (Ayodele, 2010). This section will discuss the four types of machine learning techniques.

Figure 2.4 - Types of Machine Learning Algorithms



2.4.1.1 Supervised Learning

Supervised learning enables to obtain data or generate a data output from previous labelled results. The labelled training data comprises of training examples and each example is a set which has a vector and a target response (Mohri, Rostamizadeh and Talwalkar, 2012). The algorithm collects the independent attributes, which have predetermined outputs, and identifies the relationship between the independent and dependent attributes (Han, Kamber and Pei, 2012). Analyzing the data from the training set, the algorithm can label and evaluate trends to produce accurate performance forecasts and map new examples (Haney, Bowman and Chakravarty, 2015). A measure of progress for an algorithm is to observe an increase of accuracy and precision of its target output and forecasts, therefore the algorithm has learnt to execute the inputted function (Mitchell, 1997). Supervising learning methods include 2 main models: classification and regression. When the independent covariate is categorical, the classification algorithm is to be applied whereas the regression algorithm is implemented on real-valued covariates (Murphy, 2012). This study will apply classification and regression algorithms to highlight high-risk gullies which have recurring maintenances therefore prevent future jobs.

2.4.1.2 Unsupervised Learning

Unsupervised learning algorithms take a collection of data that does not have previously labelled/classified output to discover interesting structures and patterns within the data; this is known as knowledge discovery. The algorithm will identify similar characteristics and undetected patterns in the dataset with minimal human supervision. This type of machine learning is considered to be the nearest to both human and animal intellectual ability. It is more widely applied than supervised learning as there is no requirement for human oversight to label the data (Murphy, 2012). Cluster analysis is one of the key examples of unsupervised learning. Clustering is a machine learning subset that groups data points, known as clusters, by similar characteristics that does not have pre-existing labels (Roman, 2019). Other methods include association rules mining, anomaly detection, and neural networks.

2.4.1.3 Semi-supervised Learning

This type of learning paradigm falls in between supervised learning and unsupervised learning. Semi-supervised learning uses unlabeled data in conjunction with minimal labelled data to construct a classifier (Han, Kamber and Pei, 2012). The retrieval of classified data is difficult and requires human skills as opposed to unlabeled data which is accessible. Defining labels for data points have high costs and require time and energy (Chapelle, Schoellkopf and Zien, 2006).

2.4.1.4 Reinforcement Learning

The focus of reinforcement learning is to seek a compromise between discovery and exploitation of existing knowledge. It is associated with Figure 2.4 - Types of Machine Learning Algorithms whether learning algorithm, known as agents, can learn and act in their surroundings to optimize the concept of incremental reward. (Sutton and Barto, 2018).

2.5 Classification Algorithms

This research focuses on exploring various classification algorithms that provides the strongest predictions for recurring maintenance rates. The models deemed beneficial for this analysis are Decision Tree, Isolation Forest, and Logistic Regression which will be detailed in this segment.

2.5.1 Decision Tress

Decision Tree is a classification algorithm, applied in predictive modelling for machine learning. The aim is to construct a model that determines the value of the output variable dependent on multiple input variables. The decision tree is visualized as a tree-like arrangement, where the root node first and top node, branching out to internal nodes which represent a test on a feature, each branch depicts a test outcome, and each terminal/leaf node has a class label. Classification trees are implemented to identify an outcome in predefined classes based on value importance whereas regression trees are when the predicted result is regarded to be a

real number (Rokach and Maimon, 2014). Notable algorithms for decision tree learning are ID3 (Iterative Dichotomiser), C4.5 and the Classification and Regression Tree, known as CART (Han, Kamber and Pei, 2012).

The type of attribute must be considered before a decision tree is implemented. Type of attribute in a dataset can be categorical or continuous. Splitting within a decision tree is based on the type of attribute; this includes binary split or multi-way split. The strongest partition is defined when the nodes have a balanced class distribution with low impurity. Attribute selection for decision trees does not simply select random nodes as this will produce poor results with low accuracy; rather, goodness functions are used to prevent this issue. Goodness functions include:

2.5.1.1 Entropy

Entropy is used to measure the homogeneity of a node. It is more difficult to gain information from a dataset if the entropy is high. Entropy is mathematically represented as:

$$E(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

Where

S is the current state

p_i is the probability of a class i in a node

2.5.1.2 Information Gain

A decrease in entropy is information gain. It measures the discrepancy between the entropy before splitting and the mean entropy after splitting the dataset depending on the values of the specified variable. To maximise information gain, the split that has the most reduction must be selected. Information gain is mathematically represented as:

$$Information\ Gain = E(p) - \left(\sum_{i=1}^{k} \frac{n_i}{n} E(i)\right)$$

Where

p is before splitting

k is the partitions

n_i is the number of records in subset i

2.5.1.3 Gini Index

Gini index measures the probability of incorrect labelling of an attribute during random selec tion. Gini index is employed by CART to produce points of partition. Gini index is mathematically defined as:

$$Gini = 1 - \sum_{i=1}^{n} (p_i)^2$$

Where p_i is the likelihood of categorising an attribute to a specific class

Advantages of decision tree learning include individuals are able to comprehend and perceive the model easily, minimal data preparation and normalisation, and reflects human decision-making more effectively compared to other techniques. However, decision trees can be unpredictable due to a small change carried out on the training data affecting the whole structure. Additionally, generalisation of the data can cause over-fitting leading to leading to inaccurate results (James, Witten, Hastie and Tibshirani, 2015). To overcome this problem, a technique called pruning eliminates branches of the decision tree that display noise in the data and have no influence to distinguish classes; this, in effect increases precision and accuracy and minimises overfitting (Han, Kamber and Pei, 2012).

2.5.2 Isolation Forest

Any data point/observation that deviates significantly from the other observations is called an Anomaly/Outlier. Anomaly detection is important and finds its application in various domains like detection of fraudulent bank transactions, network intrusion detection, sudden rise/drop in sales, change in customer behavior, etc.

Isolation forest is an ensemble-based decision tree for anomaly detection. One of the studies tries to explain the anomalous marine engine behavior with Shapley Additive exPlanations

(SHAP) values. As a base anomaly detector, the isolation forest developed by Liu et al. [33] is adopted. As with other ensemble-based methods like the random forest, isolation forest utilized no distance or density measures, and thus methodology was performed satisfactorily in high-dimensional data that contains many irrelevant attributes. After the anomaly score is obtained, SHAP was combined with the isolation forest to calculate the feature importance of individual prediction. Thus, given an anomalous data point, anyone can easily explain which sensor value is responsible for the anomaly.

Algorithm of Isolation Forest –

1 When given a dataset, a random sub-sample of the data is selected and assigned to a binary tree.
2 Branching of the tree starts by selecting a random feature (from the set of all N features) first. And then branching is done on a random threshold (any value in the range of minimum and maximum values of the selected feature).
3 If the value of a data point is less than the selected threshold, it goes to the left branch else to the right. And thus, a node is split into left and right branches.
4 This process from step 2 is continued recursively till each data point is completely isolated or till max depth (if defined) is reached.
5 The above steps are repeated to construct random binary trees.

Limitations -

The final anomaly score depends on the contamination parameter, provided while training the model. This implies that we should have an idea of what percentage of the data is anomalous beforehand to get a better prediction. Also, the model suffers from a bias due to the way the branching takes place.

2.5.3 Logistic Regression

Logistic Regression is a mathematical model applied in regression analysis. This method is implemented to predict the target feature by modelling the probability of predictors into classes

(Witten and Frank, 2017). The target (or outcome) is the dependent variable, and the predictors are the set of independent variables which influence the dependent variable. The outcome variable has two likely values; this is called binary or dichotomous, parameterized as 0 or 1 (Hilbe, 2017). The algorithm can highlight the variables that has the greatest influence and association on the outcome variable. Logistic regression is mathematically expressed as:

$$ln \frac{(P(\mathrm{Y}))}{1 - P(Y)} = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots \beta_K X$$

Where

Y is a dichotomous outcome
$\boldsymbol{X}_{1,2,k}$ is predictor variables
$\boldsymbol{\beta}_{0,1,2,K}$ is the intercept ($\boldsymbol{\beta}$) and model coefficients ($0, 1, 2, \boldsymbol{K}$)
$\frac{ln\,(P(\mathrm{Y}))}{1 - P(Y)}$ is the log(odds) of the outcome

Advantages of logistic regression is it is highly interpretable and easy to train. Limitations of the logistic algorithm the number and efficacy of the predictor variables have an influence on the feasibility of the model. When collinearity occurs within the variables, the prediction of outcomes will be less reliable due to an increase in errors (Tolles and Meurer, 2016). This research will implement logistic regression to predict the probabilistic occurrence of whether a gully will be maintained multiple times utilizing knowledge or attributes that are presumed to affect these occurrences.

2.6 Time Series Analysis Visualisation

Time series is a collection of data that are listed in time order. The order of time is often exhibited in days, weeks, months, or years. Time series analysis provides techniques for evaluating time series data with a view to obtaining valuable knowledge (Palma, 2016). The most common approach of visualizing time series data is a basic line graph; this research will also adopt a calendar heat map to detect temporal trends. Calendar heat maps are an alternative method of analyzing time series data. The data in this research would be represented in a calendar format in which the number of reattendances are reflected over the past three years in months and weeks. The color mapping is displayed using the values of the

target variable that differs by day and the color variance is attributed to the hue or intensity which provides clear visual focus on the hidden underlying trends over the period of time. Heat maps provide insightful information for visual interpretation of patterns and seasonality (Kriebel and Murray, 2018).

2.7 Haversine Formula

There are more than just 3 trigonometric functions that all of us are familiar with – sine, cosine, tangent. The additional trigonometric functions are versine, haversine, coversine, hacoversine, exsecant, and excosecant. All of these can be expressed simply in terms of the more familiar trigonometric functions too. The haversine formula is a very accurate way of computing distances between two points on the surface of a sphere using the latitude and longitude of the two points. The haversine formula is a re-formulation of the spherical law of cosines, but the formulation in terms of haversines is more useful for small angles and distances. One of the primary applications of trigonometry was navigation, and certain commonly used navigational formulas are stated most simply in terms of these archaic function names. The Haversine formula is perhaps the first equation to consider when understanding how to calculate distances on a sphere. The word "Haversine" comes from the function:

$$\text{haversine}(\theta) = \sin^2(\theta/2)$$

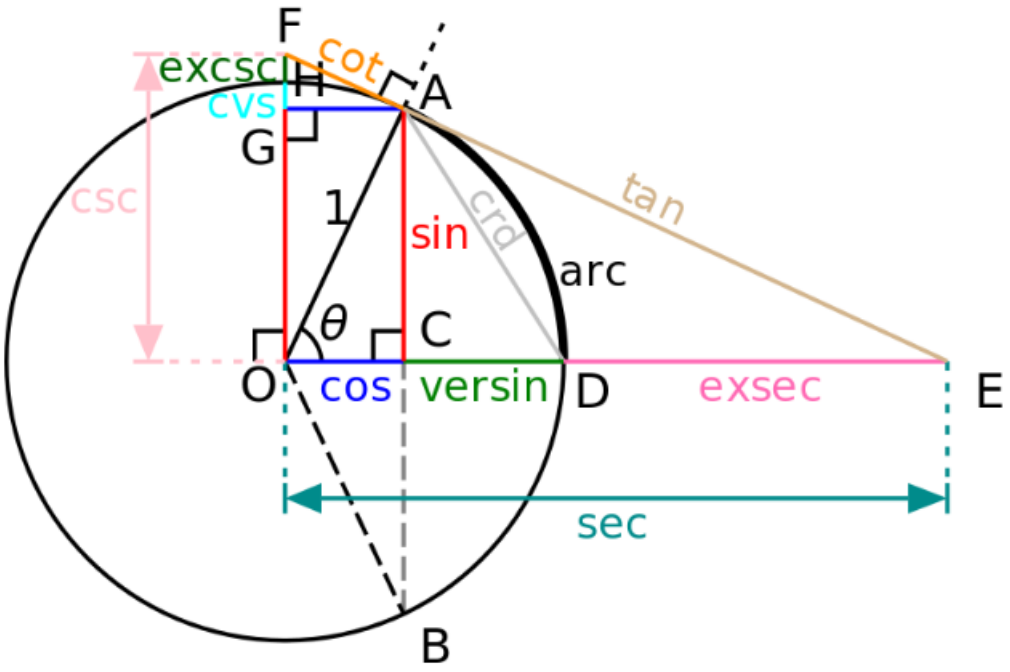
The following equation where φ is latitude, λ is longitude, R is earth's radius (mean radius = 6,371km) is how we translate the above formula to include latitude and longitude coordinates. Note that angles need to be in radians to pass to trig functions:

$$a = \sin^2(\varphi B - \varphi A/2) + \cos \varphi A * \cos \varphi B * \sin^2(\lambda B - \lambda A/2)$$

$$c = 2 * \text{atan2}(\sqrt{a}, \sqrt{(1-a)})$$

$$d = R \cdot c$$

Figure 2.5 – Overview of the Haversine formulation



2.8 Related Works

Adverse weather conditions can have a significant impact on electricity network infrastructure and will subsequently compromise the quality of power delivered to consumers. A study on the effects of climate change on the US electrical network concluded that 80% of all large-scale power outages between 2003 and 2012 were caused by weather and the average number of weather-related outages per year doubled during those years [1].

Although some results refer to weather conditions specific to the US climate, they are indicative of how changing weather conditions can affect the electricity network. In the UK, the distribution network operators have published climate adaptation reports, outlining the current risks and the anticipated impacts because of a changing climate. Among others, ref. [2] discusses the main results of a study conducted with the UK Met Office regarding the impacts on the electricity network, which identified the major causes of weather-related outages and estimated how their frequency might change in the future. Using the Met Office climate projections [3], the study showed that there is an uncertainty regarding the future occurrence of wind related faults, as there is uncertainty in the wind gust projections as well. However, the

number of lightning related faults is more likely to increase and the faults due to snow, sleet and blizzard are estimated to be fewer but with the same or increased intensity.

Pipe failure prediction has become a crucial demand of operators in daily operation and asset management due to the increase in operation risks of water distribution networks. In this paper, two machine learning algorithms, namely, random forest (RF) and logistic regression (LR) algorithms are employed for pipe failure prediction. RF algorithm consists of a group of decision trees that predicts pipe failure independently and makes the final decision by voting together. For the LR algorithm, the mapping relationship between existing data and decision variables is expressed by the logistic function. Then, the prediction is made by comparing the conditional probability with the fixed threshold value. The proposed algorithms are illustrated using an actual water distribution network in China. Results indicate that the RF algorithm performs better than the LR algorithm in terms of accuracy, recall, and area under the receiver operating characteristic curve. The effects of seven characteristics on pipe failures are analyzed, and diameter and length are identified as the top two influential factors.

The purpose of this reviewed literature was to find and relate various areas of opportunity to minimize the flood impact and the recurring maintenance rates with the use of statistical machine learning techniques. All studies have shown that there are various causes that typically lead to recurring maintenance rather than a specific, distinct factor. There has been much research and discussion conducted on minimising rates of recurring maintenance which enables the NH to assess areas of concern and recognise successful incentives for change. However, more exploration should be conducted to improve statistical models for the advancement of processes that will contribute to the preventive equipment maintenance, anomaly detection and prediction or even failure probability modeling.

2.9 Summary

This chapter presented numerous literature that discussed the basic principles and structures

in the field of data mining that can be applied for predictions of gullies which are likely to be clogged or flooded. Techniques that will be implemented in this analysis have been shown in the review of published literature. The next chapter describes the methods used to apply the analysis.

Chapter 3 METHODOLOGY

3.1 Introduction

This chapter focuses on two widely known data mining methodologies and which will be the most appropriate for this project to accomplish the target. The two data mining methods include SEMMA (Sample, Explore, Modify, Model, Assess) and Cross-Industry Standard Process for Data Mining (CRISP-DM). Stages of each methodology will be described in detail.

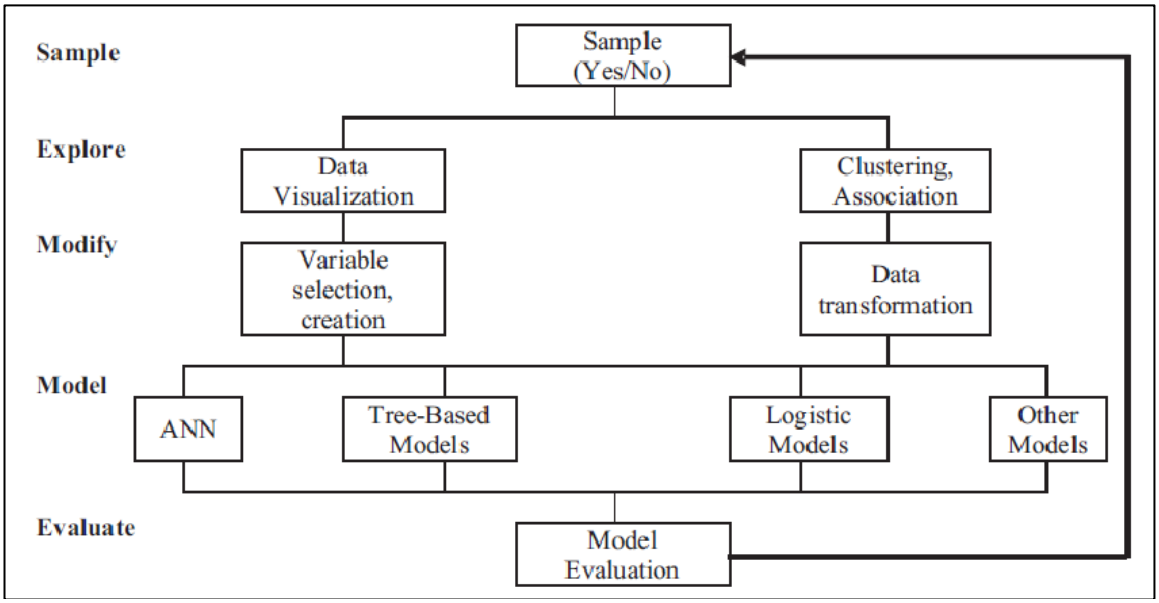
3.2 Research Methodology

3.2.1 SEMMA

The SEMMA methodology was developed by the SAS Enterprise Miner Institute for data mining. The acronym signifies five stages of the analysis:

1. **Sample:** The initial stage is optional which includes acquiring a sample from a large dataset with significant knowledge to collect and small enough to manipulate easily.
2. **Explore:** The second stage focuses on exploring the acquired data to build insights and hypotheses, as well as to improve the discovery process by observing interesting patterns, trends and outliers.
3. **Modify:** This stage involves data alterations by generating, selecting and transforming attributes to the correct format. Reduction of variables and anomalies are within this stage.
4. **Model:** A suitable algorithm is selected for data modelling to forecast results depending on the analysis objective.
5. **Assess:** The fifth and final stage involves evaluating the model and performance to gain undiscovered knowledge and relevance of results for the research.

Figure 3.1 – SEMMA Methodology Steps (Jackson, 2002)



3.2.2 CRISP-DM

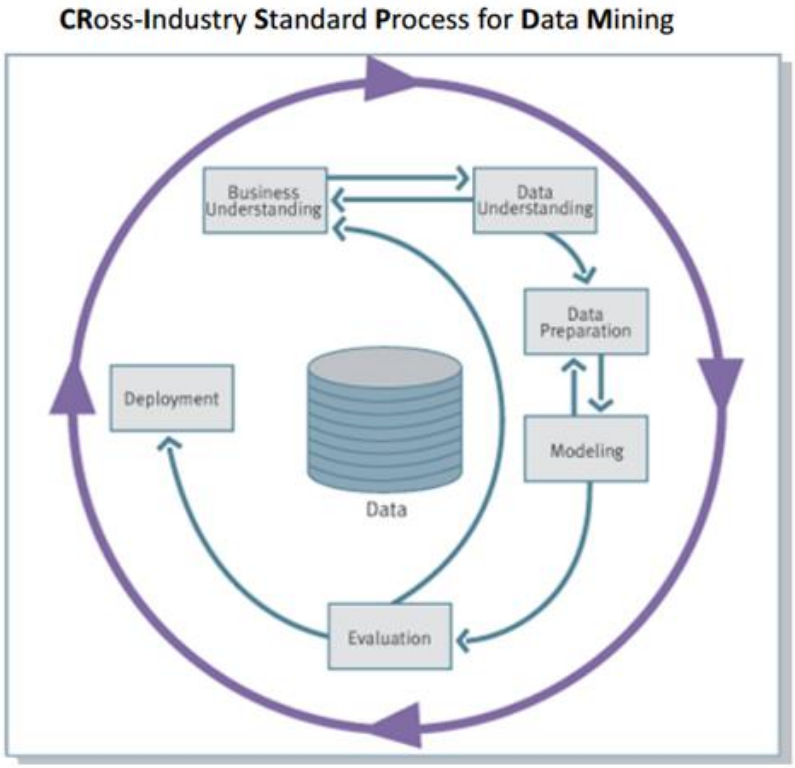
Cross-Industry Standard Process for Data Mining (CRISP-DM) was first established in late 1996 by Daimler Chrysler, ISL, NCR and OHRA. This comprehensive tool sets a standardised framework and structure for data miners to understand the data mining process and plan a project (Shearer, 2000). The CRISP-DM cycle is in six phases:

1. **Business Understanding:** The first stage focuses on identifying the objectives and criteria of the project from a business perspective, translating this information to apply onto the data mining task and creating a project plan.
2. **Data Understanding:** The data is gathered, explored and analysed to recognise data quality, uncover patterns, and develop hypotheses.
3. **Data Preparation:** The initial raw data is processed by cleansing of the data, integrating records, transforming to the correct format, and subset selection to create the final dataset.
4. **Modelling:** Various modelling methods are selected and implemented to the final dataset and their parameters are optimised to the appropriate values. There are many

approaches for similar types of problems, hence it will be necessary to go back to the data preparation phase in order to correspond to the specific prerequisites.

5. **Evaluation:** The constructed models are evaluated on their performance to review whether the business objectives of the project have been achieved. Before final deployment is reached, results of the data mining model must correlate to the goals; if they are not met, the cycle should be repeated in order to identify solutions to further develop and strengthen the model.
6. **Deployment:** The final stage of the cycle assesses the performance obtained from the built model to address the problems, report and present the acquired knowledge.

Figure 3.2 – Stages of the CRISP-DM Cycle (Shearer, 2000)



A comparison of the two methodologies demonstrated that the CRISP-DM approach is best suited for this research. While both approaches are effective methodologies for the implementation of data mining tasks, CRISP-DM was selected for this study. CRISP-DM's cyclical and versatile design offers several benefits to the user such as observing progress

between phases and having the option to go through and review each stage, make iterations, and therefore have a better understanding of the data and challenge. The knowledge gained from previous stages will then be incorporated into the subsequent cycles. The SEMMA methodology omits steps covered in CRISP-DM such as Business Understanding and Deployment (Table 3.1). Furthermore, SEMMA was developed to support SAS Enterprise Miner users hence, it is ambiguous to implement the method beyond the SAS software.

Table 3.1 – Comparison between SEMMA and CRISP-DM methodologies

Data Mining Methodologies	**SEMMA**	**CRISP-DM**
No. of Steps	5	6
Steps	----------	Business Understanding
	Sample	Data Understanding
	Explore	
	Modify	Data Preparation
	Model	Modelling
	Assessment	Evaluation
	----------	Deployment

3.3 Summary

This chapter discussed the methodologies for data mining: SEMMA and CRISP-DM. The comparison of both methods clarified the use of CRISP-DM for this analysis by reviewing stages from each process. CRISP-DM is a well-balanced methodology that is suited for this study and following chapters are corresponding to the chosen methodology.

Chapter 4 BUSINESS AND DATA UNDERSTANDING

4.1 Introduction

This chapter discusses the dataset acquired from NH to gain better understanding of the parameters utilized for modelling. It also provides insight into the business perspective as well as how the data will be preprocessed to implement in the analysis.

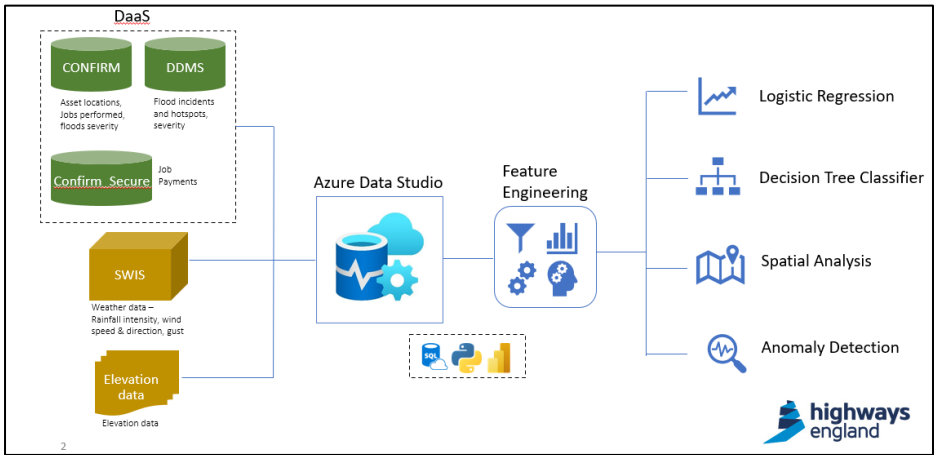
4.2 Business Understanding

The key objective of this study is to identify the best performing and high accuracy resulting classification or regression model that provides correct predictions to classify asset failures from nonfailures. The dataset gathered contains jobs performed on the gully or asset located on a section of a highway or a motorway, severity of the floods recorded, observations, and diagnostics. By applying statistical principles and integrating appropriate algorithms on the data, critical evaluation of the models will be conducted to select the best choice.

4.3 Data Understanding

The second stage of the CRISP-DM methodology is to collect, explore and understand the data. In the Azure SQL - DaaS, the data is stored in various schematic tables. The data was manipulated and transformed to meet the specifications of this research in relation to external sources like SWIS – weather data and elevation data which are maintained in .csv format. The dataset was then extracted from the database using Azure Data Studio where we explore the dataset.

Figure 4.1 – Logical Data flow diagram



4.3.1 Data Preparation

The data preparation is a two-part process. Firstly, the weather data from SWIS is imported, aggregated daily and stored in .csv format. The aggregation is necessary since the meteorological data from SWIS is recorded for every 10 seconds, and it is essential to aggregate not simply by a mean, but also relevant statistical parameters like mean, median, minimum, maximum recorded as of that date for a given radar station across England.

	RadarID	Date	Present_Weather				Air_Temp				...	Wind_Gust_Speed		Wind_Gust_Direction				Rain_Intensity			
			mean	median	min	max	mean	median	min	max	...	min	max	mean	median	min	max	mean	median	min	max
0	10000	2018-01-01	94.17	100.0	50.0	100.0	4.49	4.6	1.9	6.4	...	2.52	35.64	252.82	295.5	1.0	357.0	0.03	0.0	0.0	0.75
1	10001	2018-01-01	99.65	100.0	50.0	100.0	4.04	4.2	1.7	6.1	...	8.28	48.96	281.7	284.0	7.0	343.0	0.0	0.0	0.0	0.44
2	10002	2018-01-01					3.1	3.25	1.0	5.6	...	3.96	42.48					0.0	0.0	0.0	0.0
3	10003	2018-01-01					4.94	5.0	2.7	7.6	...	2.52	28.44	244.24	301.0	0.0	357.0	0.18	0.0	0.0	7.44
4	10004	2018-01-01	99.1	100.0	50.0	100.0	4.66	4.3	3.1	8.0	...	0.36	19.8	146.19	121.0	1.0	349.0	0.01	0.0	0.0	0.57

5 rows × 30 columns

Based on these aggregated values, the condensed information is extended to features that would be used within the prediction model. Data availability and quality would also be checked during feature selection, condensation, and aggregation processes. The extension criteria following condensation is maintained for the periods of last 48 hours, last 96 hours, last 1 week and last 2 weeks.

Secondly, the jobs, job payments, elevations and floods datasets are combined to model the data and explored from various perspectives. All these datasets are stored as pandas dataframes within the python notebook. Jobs data is extracted from CONFIRM layer of DaaS. Total number of jobs on the gullies recorded from 29/03/2018 to 2020/01/01 are 122222 against which there were 70129 assets / asset locations identified. Since the weather data is available only for 2 years 2018 and 2019, the job entry dates have been filtered out accordingly.

	JobID	EntryDateTime	Year_Month	Notes	CentralAssetID	EastingCentral	NorthingCentral	East_Nort	LatitudeCentral	LongitudeCentral	...	DueDateGroup (CalcTarget)
0	41196417	2018-03-29	2018-03	4.3.1 Clean empty gullies - M6 M Link 11\r\n (...	DRGU/040785	350233.56	530218.65	350233_530218	54.664585	-2.773067	...	Due Date Passed

Total Jobs recorded : 122222

Total number of Asset locations : 70129

A second table for all the jobs is also used to understand and have an in-depth knowledge of all the gully jobs that were performed since the beginning of time. The dimensionality for this table is 793709 records (jobs performed) and 65 columns. This table has all the data from 2018 to 2022 (present).

Floods data is also extracted from DaaS like the jobs dataset, and we store it in a pandas dataframe. Floods data is extracted from DDMS layer of DaaS. The dimensionality for this table is 472516 records (floods recorded) and 21 columns.

	FloodID	Road	Area	Latitude	Longitude	CarriagewayDirection	CarriagewayType	SectionID	FloodStatus	MaxEffect	...	AffectedLength	FloodSeverityIndex
0	18755	M60	10	53.438609	-2.130051	CW	Main Carriageway	4200M60/679	Closed	Hard shoulder only	...	NaN	4.80
1	514	A40	2	51.872160	-2.405290	WB	Main Carriageway	1600A40/678	Historic	None	...	NaN	0.67

Similarly, job payments are also extracted from CONFIRM_Secure layer, which is a strictly confidential layer. It was absolutely necessary to not let these payments data spread out of National Highways Azure tenancy as part of the data privacy and non-disclosure. The payments shown in the below screenshot are replaced by randomizing.

	JobID	Payment_Amount
0	41196417	531
1	41196418	399

4.3.2 Data Exploration

The Jobs and weather combined dataset initially consisted of 122222 observations of 60 variables. However, during dimensionality reduction, only variables relevant to the research were chosen hence the dataset was transformed to 122222 observations of 26 to 28 variables. The records cover the timeframe from March 2018 to Dec 2019. The dataset will be explored and visualized using statistical analytics to capture beneficial trends and valuable insight in

the form of tables and graphs. The selected columns for modelling are listed and described in Table 4.1.

Table 4.1 - Data Dictionary

VARIABLE NAME	VARIABLE TYPE	DESCRIPTION
JobID	Categorical	ID number for the job performed on a gully
EntryDateTime	Date	Entry date recorded for the job
Year_Month	Categorical	Year & month from Entry Date
Notes	Categorical	Notes / Description
CentralAssetID	Categorical	ID number to identify the asset / gully
EastingCentral	Integer	Easting (Geospatial purposes)
NorthingCentral	Integer	Northing (Geospatial purposes)
LatitudeCentral	Integer	Latitude (Geospatial purposes)
LongitudeCentral	Integer	Longitude (Geospatial purposes)
SiteID	Categorical	ID number to identify the site (group of gullies)
SiteName	Categorical	Site Name
PlotNo	Categorical	Plot number
RouteID	Categorical	ID number to identify the route
RouteName	Categorical	Route name
DepotID	Categorical	ID number to identify the depot
DepotName	Categorical	Depot Name
RoadName	Categorical	Road Name
ContractAreaID	Categorical	ID number to identify the contract area
ContractArea	Categorical	Contract Area name
AssetTypeID	Categorical	ID number to identify the asset type
AssetTypeName	Categorical	Asset type name
RegimeID	Categorical	ID number to identify the Regime ID

RegimeName	Categorical	Regime Name
CustomerID	Categorical	ID number to identify the customer
Customer Name	Categorical	Customer Name
CostID	Categorical	ID number to identify the cost allocated
ParameterTypeID	Categorical	ID number to identify the parameter type
ParameterValueID	Categorical	ID number to identify the parameter value
% Full	Categorical	% of clogging identified when the job was performed
Year	Categorical	Year of the entry date
Month	Categorical	Month of the entry date
Elevation(m)	Float	Elevation / Altitude (in metres)
Elevation(ft)	Float	Elevation / Altitude (in feet)
RadarID	Categorical	ID number specific to each radar station
Date	Date	Weather recorded date
ESS_Location	Categorical	Radar station location name
Rain_Intensity_Avg_Last_48hrs	Float	Average of Rain Intensity in the last 48 hours
Rain_Intensity_Avg_Last_96hrs	Float	Average of Rain Intensity in the last 96 hours
Rain_Intensity_Avg_Last_Week	Float	Average of Rain Intensity in the last 1 week
Rain_Intensity_Avg_Last_2_Weeks	Float	Average of Rain Intensity in the last 2 weeks
Rain_Intensity_Max_Last_48hrs	Float	Maximum of Rain Intensity in the last 48 hours
Rain_Intensity_Max_Last_96hrs	Float	Maximum of Rain Intensity in the last 96 hours
Rain_Intensity_Max_Last_Week	Float	Maximum of Rain Intensity in the last 1 week
Rain_Intensity_Max_Last_2_Weeks	Float	Maximum of Rain Intensity in the last 2 weeks
Wind_Speed_Mean_Last_48hrs	Float	Average of Wind Speed in the last 48 hours

Wind_Speed_Mean_Last_96hrs	Float	Average of Wind Speed in the last 96 hours
Wind_Speed_Mean_Last_Week	Float	Average of Wind Speed in the last 1 week
Wind_Speed_Mean_Last_2_Weeks	Float	Average of Wind Speed in the last 2 weeks
Wind_Direction_Mean_Last_48hrs	Float	Average of Wind Direction in the last 48 hours
Wind_Direction_Mean_Last_96hrs	Float	Average of Wind Direction in the last 96 hours
Wind_Direction_Mean_Last_Week	Float	Average of Wind Direction in the last 1 week
Wind_Direction_Mean_Last_2_Weeks	Float	Average of Wind Direction in the last 2 weeks
Wind_Gust_Mean_Last_48hrs	Float	Average of Wind Gust in the last 48 hours
Wind_Gust_Mean_Last_96hrs	Float	Average of Wind Gust in the last 96 hours
Wind_Gust_Mean_Last_Week	Float	Average of Wind Gust in the last 1 week
Wind_Gust_Mean_Last_2_Weeks	Float	Average of Wind Gust in the last 2 weeks
Wind_Gust_Direction_Mean_Last_48hrs	Float	Average of Wind Gust Direction in the last 48 hours
Wind_Gust_Direction_Mean_Last_96hrs	Float	Average of Wind Gust Direction in the last 96 hours
Wind_Gust_Direction_Mean_Last_Week	Float	Average of Wind Gust Direction in the last 1 week
Wind_Gust_Direction_Mean_Last_2_Weeks	Float	Average of Wind Gust Direction in the last 2 weeks

4.3.3 Data Transformations

Firstly, we acquire the dataset from data.gov.uk – National Statistics postcodes, which essentially has all the postcodes within the UK, or in our case, all England only (since this project is related to only Highways England). The variables that we grab from this dataset are going to be post code, county code, county name, easting, northing, latitude, longitude, Output area classification

name. In order to establish a key value pair relationship between latitude & longitude, we round off the floating-point numbers into 2 decimal places and make them work for the nearest 1.11 km radius of the looked-up value from the key, and later make it a composite key by using an underscore as a delimiter. This data is stored in a single pandas dataframe.

Secondly, we bring the elevation dataset also from data.gov.uk, which is basically the altitude of the sea-level from above the surface level. From this dataset, elevation values against each of the postcodes from UK are brought into a pandas dataframe. Later, the locations and elevations are merged using the postcodes (without any space to avoid redundancies). So now we have full location details along with the postcodes. In addition to the existing elevation values in meters (m), we also consider a new column which is a transformation of the existing elevation column in feet (ft). For this, a factor of 3.28084 is multiplied with the elevation values which are in meters (m). The dataframe is then sorted by the composite key – Lat_Long. During the merge of location, elevation and jobs datasets, elevation data is grouped and aggregated based on the mean value after removal of duplicates, and later the index is reset to perform the correct merge.

```
# PostCodes Dataset
df_loc = pd.read_csv (filepath + 'UK Locations_csv\National_Statistics_Postcode_Lookup_UK.csv')
df_loc = df_loc[['Postcode 2', 'County Code', 'County Name', 'Easting', 'Northing', 'Latitude', 'Longitude', 'Output Area Classification Name']]
df_loc['Lat_Long'] = df_loc['Latitude'].round(2).astype(str) + '_' + df_loc['Longitude'].round(2).astype(str)
df_loc['Post_code_noSpaces'] = df_loc['Postcode 2'].str.replace(' ', '')

# Altitude/Elevation Data
df_elev = pd.read_csv (filepath + 'Elevation_csv\open_postcode_elevation.csv')
df_elev.columns = ['Post_code', 'Elevation(m)']
df_elev['Post_code_noSpaces'] = df_elev['Post_code'].str.replace(' ', '')
del df_elev['Post_code']

# Merge Location & Elevation
df_loc_elev = pd.merge(df_loc, df_elev, how='left', on = 'Post_code_noSpaces')
df_loc_elev['Elevation(ft)'] = df_loc_elev['Elevation(m)'] * 3.28084
df_loc_elev.sort_values(by=['Lat_Long'], inplace=True)

# Elevation by Location
df_loc_elev2 = df_loc_elev[['Lat_Long', 'Latitude', 'Longitude', 'Elevation(m)', 'Elevation(ft)']]
df_loc_elev2['Latitude'] = df_loc_elev2['Latitude'].round(2)
df_loc_elev2['Longitude'] = df_loc_elev2['Longitude'].round(2)
df_loc_elev2 = df_loc_elev2.drop_duplicates(['Lat_Long', 'Latitude', 'Longitude', 'Elevation(m)', 'Elevation(ft)'])
df_loc_elev2 = df_loc_elev2.groupby(['Lat_Long', 'Latitude', 'Longitude'])['Elevation(m)', 'Elevation(ft)'].mean().reset_index()

# Merge Jobs, Location, Elevation
df_jobs_loc_elev = pd.merge(df_jobs, df_loc_elev2,  how='left', left_on=['Lat_Long'], right_on = ['Lat_Long'])
df_jobs_loc_elev.head(2)

```

The result of the merge is as shown below. We observe elevation against all the jobs.

NorthingCentral	East_Nort	LatitudeCentral	LongitudeCentral	...	ParameterTypeID	ParameterValueID	% Full	Year	Month	EntryDate	Latitude	Longitude	Elevation(m)	Elevation(ft)
530218.65	350233_530218	54.664585	-2.773067	...	%FUL	NK	Change Me	2018	03	2018_03_29	54.66	-2.77	150.0	492.126
529899.18	350310_529899	54.661722	-2.771825	...	%FUL	NK	Change Me	2018	03	2018_03_29	54.66	-2.77	150.0	492.126

The next step is to incorporate Haversine Triangulation method, which is essentially calculating the distance between two locations on a spherical surface. The earth's radius 'R' varies from 3949.9 miles (6356.7 km) at the poles to 3963.1 (6378.1 km) at the equator. Looking at UK, we can consider that the earth's radius would be 3958.756 miles (6371 km).

Firstly, a function is created to calculate the geographical distance between 2 locations in miles (using their latitude and longitude as arguments). Asin, cos, radians, sin, sqrt functions from python Math library are used to perform trigonometric operations on the arguments provided. Individual absolute distances between latitudes and longitudes are calculated first, and later the trigonometric operation is performed. The outcome is multiplied with the radius of the earth 'R'.

Secondly, the closest radar location is identified from a given location using the above given geographical distance function. Exception handling is used to execute the code and avoid any interruptions. Later we import SWIS locations which has weather data locations of the radars. As part of the data transformations, we bring only the radars for which for which the ESSID is available and also the ESS ID data type is converted into integer and later into string and only the required columns ESSID, HE area, latitude and longitude are considered as necessary columns and we can drop all the other columns.

```
## Note - The Earth's radius 'R' varies from 3949.902 miles (6356.752 km) at the poles to 3963.190 (6378.137) km at the equator.

# 1. Identify the geographical distance between 2 locations in miles (using Lat & Long as args)
def geo_dist(*args):

    from math import asin, cos, radians, sin, sqrt
    lat1, lat2, long1, long2 = map(radians, args)
    R_earth = 3958.756                                  # 6371 km

    dist_lats = abs(lat2 - lat1)
    dist_longs = abs(long2 - long1)

    x = sin(dist_lats/2)**2 + cos(lat1) * cos(lat2) * sin(dist_longs/2)**2
    y = asin(sqrt(x)) * 2

    return y * R_earth

# 2. Identify the closest Radar location from a given location
def closest_Radar(radar_locs, x):
    try:
        return min(radar_locs, key=lambda p: geo_dist(x['lat'],p['lat'],x['lon'],p['lon']))
    except TypeError:
        print('Not a list or not a number.')

# 3. Import Radar Locations from SWIS
df_Radar_Loc = pd.read_csv (filepath + 'SWIS_Locations/Locations.csv')
```

The radar locations are pivoted using pandas pivot_table() method, and converted into a python dictionary, which means for each of the Radar location as a key, we have individual latitude and longitude information in dictionary values. These location values for which we have created a haversine function will be fetched against each of the latitude and longitude information for the jobs that were merged earlier.

```
# 4. Necessary Transformations
df_Radar_Loc = df_Radar_Loc[df_Radar_Loc['ESS ID'] > 0]
df_Radar_Loc['ESS ID'] = df_Radar_Loc['ESS ID'].astype(int).astype(str)
df_Radar_Loc.columns = ['ESS ID', 'HE Area', 'lat', 'lon']
df_Radar_Loc = df_Radar_Loc.drop('HE Area', 1)

# 5. Pivot Radar Locations and Convert into a dictionary
df_Radar_Loc_t = pd.pivot_table(data=df_Radar_Loc,index=['ESS ID']).T
dict_RadarLoc = df_Radar_Loc_t.to_dict()

# 6. Find the nearest Radar location for our jobs
list_AssetID = []
list_RadarLoc = []
list_lat = []
list_lon = []

for k_a, v_a in dict_AssetLoc.items():
    v_radarloc = closest_Radar(dict_RadarLoc.values() , v_a )
    list_AssetID.append(k_a)
    list_RadarLoc.append(v_radarloc)
    list_lat.append(v_radarloc['lat'])
    list_lon.append(v_radarloc['lon'])

df_Asset_Radar_Link = pd.DataFrame(list_AssetID, columns = {'AssetID'})
df_Asset_Radar_Link['Radar_Loc'] = list_RadarLoc
df_Asset_Radar_Link['Radar_Lat'] = list_lat
df_Asset_Radar_Link['Radar_Lon'] = list_lon
```

All the jobs are essentially performed on an asset, and each of the assets have individual asset

IDs for which we are fetching the radar locations.

```
# 7. Merge to Get Radar IDs
df_Asset_Radar_Link = pd.merge(df_Asset_Radar_Link , df_Radar_Loc,  \
                               how='left', left_on=['Radar_Lat', 'Radar_Lon'], right_on = ['lat', 'lon'])
df_Asset_Radar_Link = df_Asset_Radar_Link[['AssetID', 'Radar_Lat', 'Radar_Lon', 'ESS ID']]
df_Asset_Radar_Link.columns = ['AssetID', 'Radar_Lat', 'Radar_Lon', 'Radar ID']

# 8. Merge to link Radar locations with Jobs
df_jobs_lc_el_rd = pd.merge(df_jobs_loc_elev , df_Asset_Radar_Link,  \
                               how='left', left_on=['CentralAssetID'], right_on = ['AssetID'])

df_jobs_lc_el_rd.head(2)
```

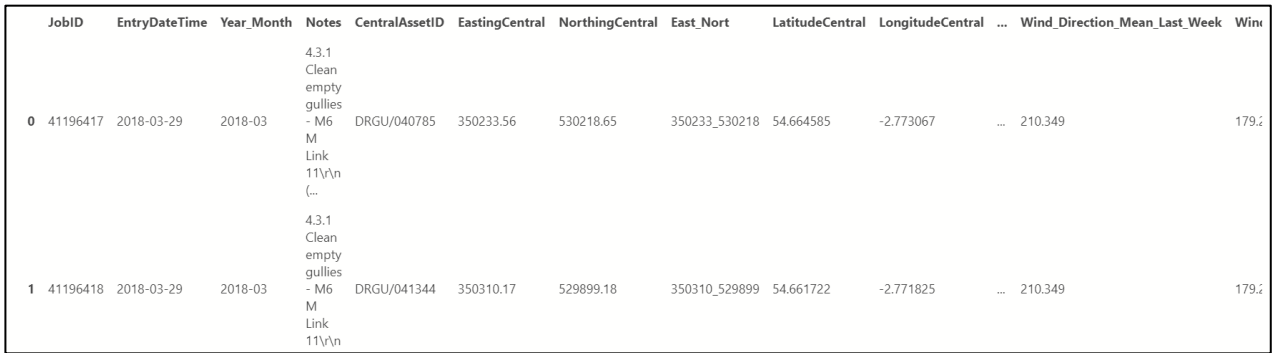
The result of the haversine triangulation is as shown below. We observe nearest radar locations against all the jobs / assets.

ntral	NorthingCentral	East_Nort	LatitudeCentral	LongitudeCentral	...	Month	EntryDate	Latitude	Longitude	Elevation(m)	Elevation(ft)	AssetID	Radar_Lat	Radar_Lon	Radar ID
	530218.65	350233_530218	54.664585	-2.773067	...	03	2018_03_29	54.66	-2.77	150.0	492.126	DRGU/040785	54.674595	-2.77809	10071
	529899.18	350310_529899	54.661722	-2.771825	...	03	2018_03_29	54.66	-2.77	150.0	492.126	DRGU/041344	54.674595	-2.77809	10071

Now that we have the radar locations against the jobs data, we import the weather data for which we have applied the data transformations initially and we merge with the combined data set to form a final dataset. All this data is moved into a pandas data frame called df_jobs_weather. As part of the transformations, we exclude the index values, trim the existing radar ids and convert them into string datatype to avoid incorrect merges. In addition to the radar ids, we also use the Date column which is the entry date on which a job was performed and thus we link the jobs, and the weather data sets.

```
df_weather = []

# Import Weather data
df_weather = pd.read_csv(filepath + 'Weather/Weather_Aggregated.csv')

# Necessary Transformations
df_weather = df_weather.replace(np.nan, '').iloc[:,1:]

# Convert Data types
df_jobs_lc_el_rd['Radar ID'] = df_jobs_lc_el_rd['Radar ID'].replace(np.nan, None).astype('str').str.strip()
df_weather.RadarID = df_weather.RadarID.astype('str').str.strip()
df_weather.Date = df_weather.Date.str.strip().str[:4] + '_' + df_weather.Date.str.strip().str[5:7] + '_' + df_weather.Date.str.strip().str[-2:]

# Merge with Jobs
df_jobs_weather = pd.merge(df_jobs_lc_el_rd, df_weather, how='left', \
                           left_on=['Radar ID', 'EntryDate'] , right_on = ['RadarID', 'Date'])

df_jobs_weather.head(2)
```

The output for the merge can be seen as below where we have the jobs data, the location data along with the weather data as of a given date, all of them merged in a single data frame.



	JobID	EntryDateTime	Year_Month	Notes	CentralAssetID	EastingCentral	NorthingCentral	East_Nort	LatitudeCentral	LongitudeCentral	...	Wind_Direction_Mean_Last_Week	Wind
0	41196417	2018-03-29	2018-03	4.3.1 Clean empty gullies - M6 M Link 11\r\n (...	DRGU/040785	350233.56	530218.65	350233_530218	54.664585	-2.773067	...	210.349	179.2
1	41196418	2018-03-29	2018-03	4.3.1 Clean empty gullies - M6 M Link 11\r\n	DRGU/041344	350310.17	529899.18	350310_529899	54.661722	-2.771825	...	210.349	179.2

A backup copy of the dataset Is exported into a .csv file.

2.5 Export the merged data as a .csv file

```
[19]  1  df_jobs_weather.to_csv(filepath_code + 'Saved/jobs_weather_merged.csv')
```

```
[20]  1  print('Total records from Weather dataset :' , len(df_weather))
      2  print('Total Length after merge :' , len(df_jobs_weather))
```

The total number of records from the weather data set are 188956, whereas the total length after the merge with the jobs dataset is 122222.

```
Total records from Weather dataset : 188956
Total Length after merge : 122222
```

Both jobs & weather merged data set along with the floods data sets are exported.

```
[21]  1  df_floods.to_csv(filepath_code + 'Saved/floods.csv')
```

```
[22]  1  print('Total records from Floods dataset :' , len(df_floods))
```

The total number of records from flood status Are 472516.

```
Total records from Floods dataset : 472516
```

In order to enable the payments related information as well to exactly measure what kind of Payments were made for each of the jobs, we merge the payments data set with the final merged jobs and weather data set.

2.6 Import and apply data transformations to the merged dataset - Jobs & weather

```
# Import Weather data
df_jobs_weather = pd.read_csv(filepath_code + 'jobs_weather_merged.csv')

# Exclude the Index
df_jobs_weather = df_jobs_weather.iloc[: , 1:]

# EntryDate2 for Floods
df_jobs_weather['EntryDate2'] = df_jobs_weather['EntryDate'].str.replace('_', '')
df_jobs_weather[['EntryDate2']] = df_jobs_weather[['EntryDate2']].applymap(str).applymap(lambda s: "{}/{}/{}".format(s[4:6],s[6:], s[0:4]))
df_jobs_weather['EntryDate2'] = pd.to_datetime(df_jobs_weather['EntryDate2'])

# Merge with Payments - Access Revoked (as of 21-April)
df_jobs_weather = pd.merge(df_jobs_weather, df_job_pmt, how='left', \
                           left_on=['JobID'] , right_on = ['JobID'])
```

Regression related transformation is performed where an asset failure is identified from the jobs dataset. All the assets which were classified as 75 to 100% full are considered as asset failures during a job, and everything else is considered as ‘not a failure’.

```
# Regression
# Define the asset failures (75 - 100% full is considered as asset failure)
df_jobs_weather['Asset_failure'] = np.where(df_jobs_weather['% Full']!= '75 - 100% Full', 0, 1)
df_jobs_weather['Asset_failure'].value_counts()
```

Classification related transformations are also performed to identify what kind of class is the asset status is in order to be able to identify whether it is a high priority class or low priority class. In order to enable this, all the values present within 75 to 100% full are categorized as L2, all the values present between 25% to 75% full is categorized as L1 and whatever values are present between 0% to 25% full is considered as L0 and everything else is not applicable or NA. Data balance would be checked using a quick value_counts() Function firstly to identify how many failure classes are present from within df_jobs_weather data set.

```
# Classification
# Define the asset failure classes (75 - 100% full - L2 ; 25 - 75% Full - L1 ; Below 25% - L0)
df_jobs_weather['Asset_failure_class'] = np.where(df_jobs_weather['% Full'] == '75 - 100% Full', 'L2',
                                          np.where(df_jobs_weather['% Full'] == '50 - 75% Full' , 'L1',
                                              np.where(df_jobs_weather['% Full'] ==  '25 - 50% Full' , 'L1',
                                                  np.where(df_jobs_weather['% Full'] ==  '0 - 25% Full' , 'L0',
                                                      np.where(df_jobs_weather['% Full'] ==  '0 % (Clear)' , 'L0', 'NA')))))
df_jobs_weather['Asset_failure_class'].value_counts()
```

Some necessary variables are created like the earliest date and the latest date from which the jobs were present. And later the dimensionality, the maximum payment amount, the minimum payment amount and the average payment amount are Identified and rounded off to 3 decimal places.

```
# Necessary variables
max_date = df_jobs_weather.EntryDate.max().replace('_', '/')
min_date = df_jobs_weather.EntryDate.min().replace('_', '/')

# Outcomes
print('Dimensionality:' , df_jobs_weather.shape)
print('Maximum Payment Amount: £', round(df_jobs_weather.Payment_Amount.max(), 3))
print('Minmum Payment Amount: £', round(df_jobs_weather.Payment_Amount.min(), 3))
print('Average Payment Amount: £', round(df_jobs_weather.Payment_Amount.mean(), 3))
```

As we can see, now there are a total of 122,473 records and 134 columns. The maximum payment amount recorded Is £ 1100 and the minimum payment amount recorded is £ 10, and the average payment amount recorded is £ 554.9

```
Dimensionality: (122473, 134)
Maximum Payment Amount: £ 1100
Minmum Payment Amount: £ 10
Average Payment Amount: £ 554.999
```

The outcome of the previous steps where we perform the necessary transformations is shown below, with asset_failure and asset_failure_class along with the payment_amount.

.ast_48hrs	Wind_Gust_Direction_Mean_Last_96hrs	Wind_Gust_Direction_Mean_Last_Week	Wind_Gust_Direction_Mean_Last_2_Weeks	EntryDate2	Payment_Amount	Asset_failure	Asset_failure_class
	153.495	161.571	152.98	2018-03-29	531	0	NA
	153.495	161.571	152.98	2018-03-29	399	0	NA

In addition, the floods data set is also utilized where the flood reported date and time can be used

along with an offset of 14 days to merge relatively with a range with the gully maintenance jobs since the flood could have occurred anytime in the last two weeks from a job was performed based on the functional / business understanding and the same has been applied while performing the merge within Python.

2.7 Import & apply data transformations - Jobs & Floods

```
# Import Floods data
df_floods = pd.read_csv(filepath_code + 'floods.csv')

# Exclude the Index
df_floods = df_floods.iloc[: , 1:]

# Convert to DateTime & setup lead time
df_floods['ReportedDateTime'] = pd.to_datetime(df_floods['ReportedDateTime'])
df_floods['LeadTime'] = pd.to_datetime(df_floods['ReportedDateTime']) + pd.DateOffset(days=14)

# Jobs & Floods merge in a different dataframe
df_jobs_floods = df_jobs_weather[['JobID', 'EntryDate2', 'Year_Month', 'CentralAssetID', 'LatitudeCentral', 'LongitudeCentral', 'Payment_Amount', 'SiteName', '% Full',

# Extract SectionID from Jobs
df_jobs_floods['SectionID2'] = df_jobs_floods.SiteName.str.split(' ').str[0].str.strip()
del df_jobs_floods['SiteName']
```

Pandasql library is used to perform the ranged merge which will allow to identify a particular section ID (a group of gullies can be present within each of the sections) and an allowed lead time from the date of the flood reported.

```
# Merge Jobs and Floods based on SectionID and Allowed lead time from the date of flood reported
sqlcode = '''
select *
from df_floods
left join df_jobs_floods on df_jobs_floods.SectionID2=df_floods.SectionID
where df_jobs_floods.EntryDate2 >= df_floods.ReportedDateTime
and df_jobs_floods.EntryDate2 <= df_floods.LeadTime
'''

df_jobs_floods = ps.sqldf(sqlcode,locals())
del df_jobs_floods['SectionID2']

print('Total number of jobs performed on gullies subsequent to the floods (with a lead time of 2 weeks):', df_jobs_floods['JobID'].nunique())
```

A total of 843 jobs were reported subsequently within 2 weeks after a flood was reported.

```
Total number of jobs performed on gullies subsequent to the floods (with a lead time of 2 weeks): 843
```

Necessary fields like flood ID, reported datetime, latitude and longitude, section ID, asset ID, asset failure, flood severity index, payment amount are captured.

```
[27]  df_jobs_floods_f = df_jobs_floods[['FloodID', 'ReportedDateTime', 'JobID', 'EntryDate2', 'Year_Month', 'SectionID', 'CentralAssetID', 'LatitudeCentral', 'LongitudeCent
                                        '% Full', 'Asset_failure', 'Asset_failure_class', 'FloodSeverityIndex', 'Payment_Amount' ]].drop_duplicates()

      df_jobs_floods_f.head(3)
```

	FloodID	ReportedDateTime	JobID	EntryDate2	Year_Month	SectionID	CentralAssetID	LatitudeCentral	LongitudeCentral	% Full	Asset_failure	Asset_failure_class	FloodS
0	22541	2019-04-03 20:06:00.000000	42637490	2019-04-04 00:00:00.000000	2019-04	4200M60/229	DR/HEA10/DRGU/031576	53.518306	-2.367400	25 - 50% Full	0	L1	3.78
1	22541	2019-04-03 20:06:00.000000	42637491	2019-04-04 00:00:00.000000	2019-04	4200M60/229	DR/HEA10/DRGU/031535	53.517184	-2.368232	25 - 50% Full	0	L1	3.78
2	22541	2019-04-03 20:06:00.000000	42637492	2019-04-04 00:00:00.000000	2019-04	4200M60/229	DR/HEA10/DRGU/031524	53.514844	-2.368688	25 - 50% Full	0	L1	3.78

4.3.4 Exploratory Data Analysis -

Exploratory data analysis is essential before coming to any assumptions. It ensures that the results produced are valid and applicable to business outcomes and goals. It could also help us figure out if the statistical procedures we are considering for data analysis are appropriate. Before modelling the data, it gives insight into all the data and the numerous interactions between the data elements. EDA makes it simple to comprehend the structure of a dataset, making data modelling easier. The primary goal of EDA is to make data 'clean' implying that it should be devoid of redundancies. Outliers or abnormal occurrences in a dataset can have an impact on the accuracy of machine learning models. The dataset might also contain some missing or duplicate values.

4.3.4.1 Time Series 1 – Number of Maintenance Jobs per month

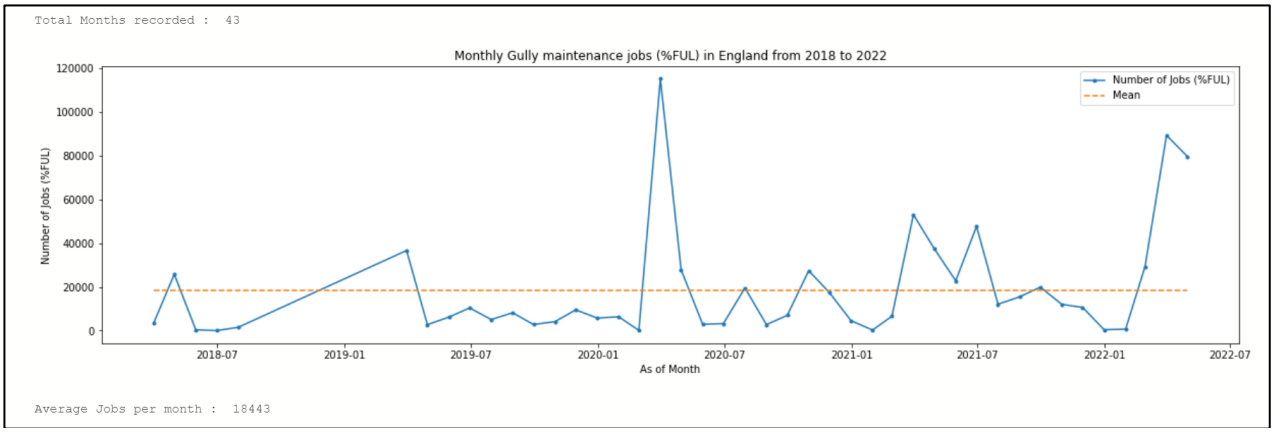
The total number of maintenance jobs per month can be extracted by using a given query 'q_jobs_ts', with the help of existing Azure SQL connection. Here we are going to assign X With the year_month column And Y value with the number of jobs in that year_month. The required aggregations have been performed within the same SQL query in order to optimize the overall performance. The total number of months are recorded along with the average number of jobs also which can be seen in a form of a line in the same chart. This chart or query essentially gives all the jobs from the beginning of time i.e., March 2018 to the present date. However, the weather data That we have is only for the two years - 2018 and 2019, and hence we only utilize the two years' worth jobs data going forward however this piece of information is crucial to understand the information and the underlying potential for the National Highways group. The required title and labels for X axis and Y axis are given, and the figure size is adjusted for the viewer's ease.

3.1 Time Series - Number of Maintenance jobs per month

```
import matplotlib.pyplot as plt
import numpy as np

# 3.1 Time Series - End of Month (SQL query)
df_jobs_ts = pd.read_sql_query( q_jobs_ts , conn)
print('Total Months recorded : ', len(df_jobs_ts))

# Assign x & y to df
x = df_jobs_ts.As_Of_YearMonth
y = df_jobs_ts.Num_Jobs

# Calculate the simple average of the data
y_mean = [np.mean(y)]*len(x)

fig,ax = plt.subplots()

# Plot the data
data_line = ax.plot(x,y, label='Number of Jobs (%FUL)', marker='.')

# Plot the average line
mean_line = ax.plot(x,y_mean, label='Mean', linestyle='--')

# Make a legend
legend = ax.legend(loc='upper right')

# Title , X & Y labels
plt.gca().set(title='Monthly Gully maintenance jobs (%FUL) in England from 2018 to 2022', xlabel='As of Month', ylabel='Number of Jobs (%FUL)')

# Adjust the size
plt.rc('figure', figsize=(20, 5))

plt.show()
print('Average Jobs per month : ', round(np.mean(y)))
```

The outcome can be seen below Where we have specific outliers especially in the month of March 2020 and March 2022, which are basically the preemptive actions that the highways group have performed as part of planned maintenances. Also, we can note that from 2018 to 2019 there were no particular outliers, and also from 2018 July to 2019 March there were no jobs performed, because it was initially assumed that the planned maintenance jobs were sufficient in order to avoid road accidents due to floods.

Figure 4.2 – Monthly gully maintenance jobs (%FUL) in England from 2018 to 2022



4.3.4.2 Time Series 2 – Maintenance Regime (Interactive plot)

This interactive plot signifies the aggregated time series maintenance regime of the jobs that were performed from 2018 to 2019 which is specific to the corresponding weather data. Plotly graph objects have been utilized to enable the interactivity of the charts, and we could clearly monitor the various aspects of the time series based on the given ranges. Here we have considered one month, six months, year-to-date, one year and all the jobs' data as individual ranges by using the arguments step and step mode.

3.2 Time Series - Maintenance Regime (Interactive plot)

```
import plotly.graph_objects as go
import pandas as pd

# 3.2 Aggregate data by EntryDateTime - Time Series (all dates)
s = pd.to_datetime(df_jobs_weather['EntryDateTime'])
df_jobs_ts_all = s.groupby(s.dt.floor('d')).size().reset_index(name='Num_Jobs')

# Create figure
fig = go.Figure()
fig.add_trace(
    go.Scatter(x=list(df_jobs_ts_all.EntryDateTime), y=list(df_jobs_ts_all.Num_Jobs)))

# Set title
fig.update_layout(
    title_text= ("Total Gully maintenance jobs (%FUL) in England " + min_date + " to " + max_date)
)

```

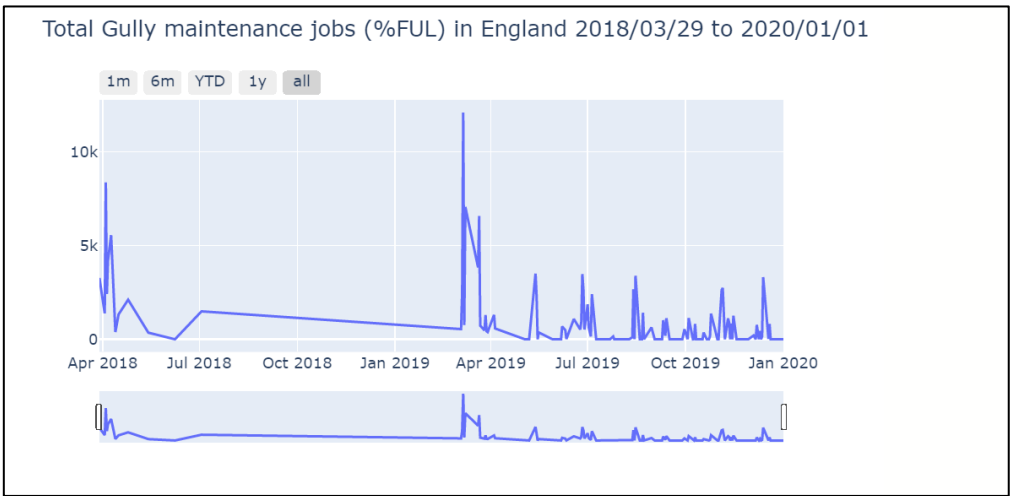
The range slider Has been enabled in this piece of script and the bottoms along with this step and step mode has been given based on the dictionary inputs.

```
# Add range slider
fig.update_layout(
    xaxis=dict(
        rangeselector=dict(
            buttons=list([
                dict(count=1,
                     label="1m",
                     step="month",
                     stepmode="backward"),
                dict(count=6,
                     label="6m",
                     step="month",
                     stepmode="backward"),
                dict(count=1,
                     label="YTD",
                     step="year",
                     stepmode="todate"),
                dict(count=1,
                     label="1y",
                     step="year",
                     stepmode="backward"),
                dict(step="all")
            ])
        ),
        rangeslider=dict(
            visible=True
```

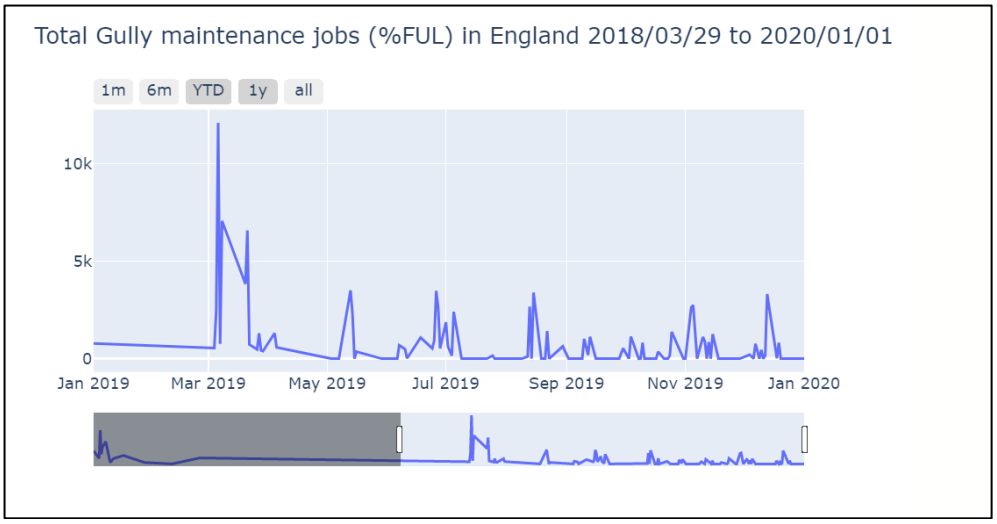
```
            ),
            type="date"
        )
    )

    fig.show()
```

The below chart shows all the gully maintenance jobs from March 29th, 2018, to Jan 1st, 2020.

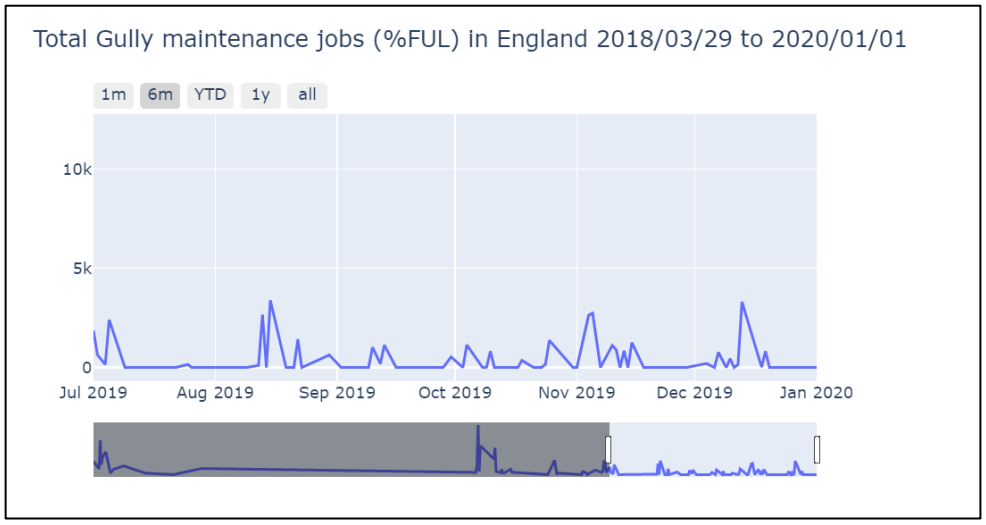
Figure 4.3 – Total gully maintenance jobs (%FUL) in England from 2018 to 2019



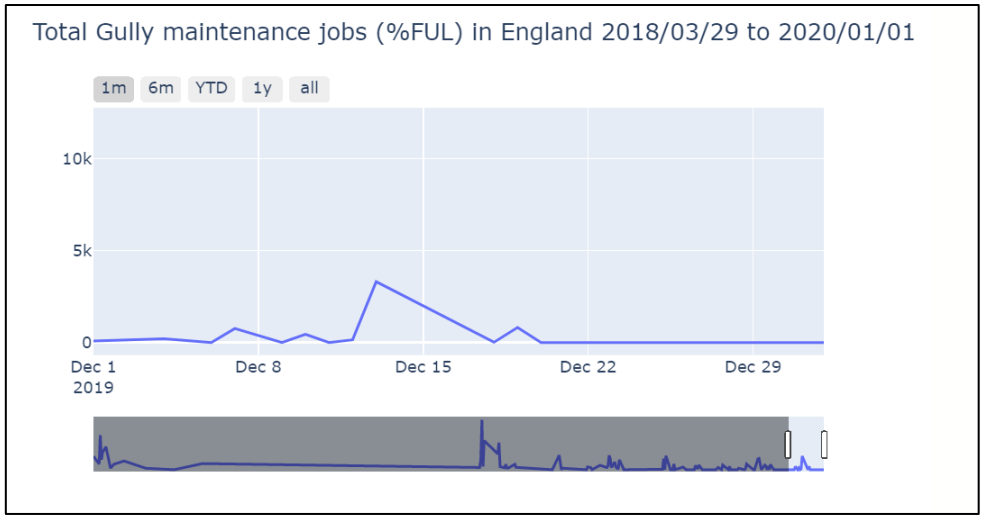
Upon selection of 1 YTD or 1 year, we could observe that the range has changed and now we see the total number of jobs from Jan 2019 to Jan 2020.



Similarly, six months selection shows the step value for the last six months From July 2019 to January 2020.



Lastly, 1 month selection shows the step value for the last 1 month From December 2019 to January 2020.



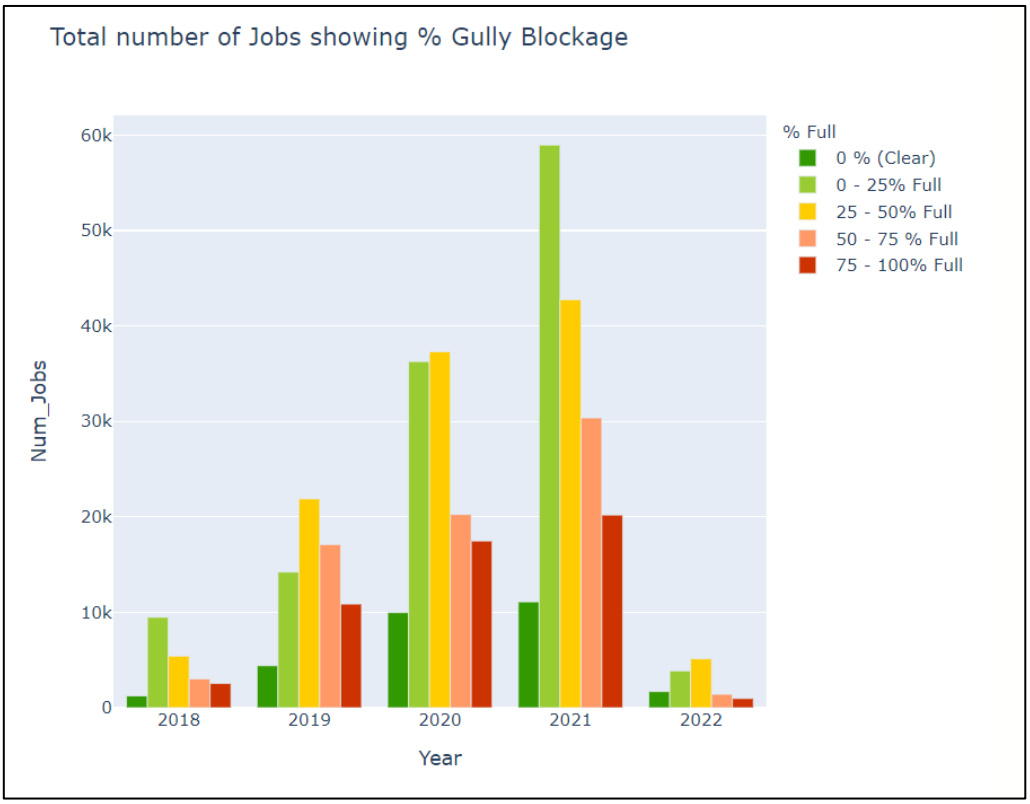
4.3.4.3 Gully clogging % over the years (Bar plots)

Plotly express charts have been used to visualize the total number of jobs broken down by the amount of congestion over the time. Initially the data has been filtered for ‘NA’ & ‘Not Known’ jobs, and then grouping has been performed on Year, % Full columns and later the index has been reset. The bar mode is going to be ‘group’.

```
3.3 Gully Clogging % over the years & months

import plotly.express as px

# 3.3 Grouping by Year & % Full
df_jobs_status_all = df_jobs2[(df_jobs2['% Full'] != 'Non Applicable') & (df_jobs2['% Full'] != 'Change Me')] \
                              .groupby(['Year', '% Full']) \
                              .size() \
                              .reset_index(name='Num_Jobs')

# Define figure
fig = px.bar( df_jobs_status_all, x='Year', y='Num_Jobs',
              color='% Full',
              color_discrete_sequence=['#339900', "#99cc33", "#ffcc00", "#ff9966", "#cc3300", 'black'],
              title="Total number of Jobs showing % Gully Blockage",
             #  text_auto='.2s',
              barmode='group',
              height=600
             )

fig.show()
```

The color coding has been given such that there is a range difference between a clean gully vs. a fully clogged gully. A fully clogged gully can be seen in the color ‘red’ whereas a clean gully can be seen in the color ‘green’. It could be observed here that there is huge potential in analyzing the forthcoming data from 2020 onwards rather than simply looking at 2018 and 2019 alone.

Figure 4.4 – Total number of jobs showing % Gully congestion (Grouped bar)

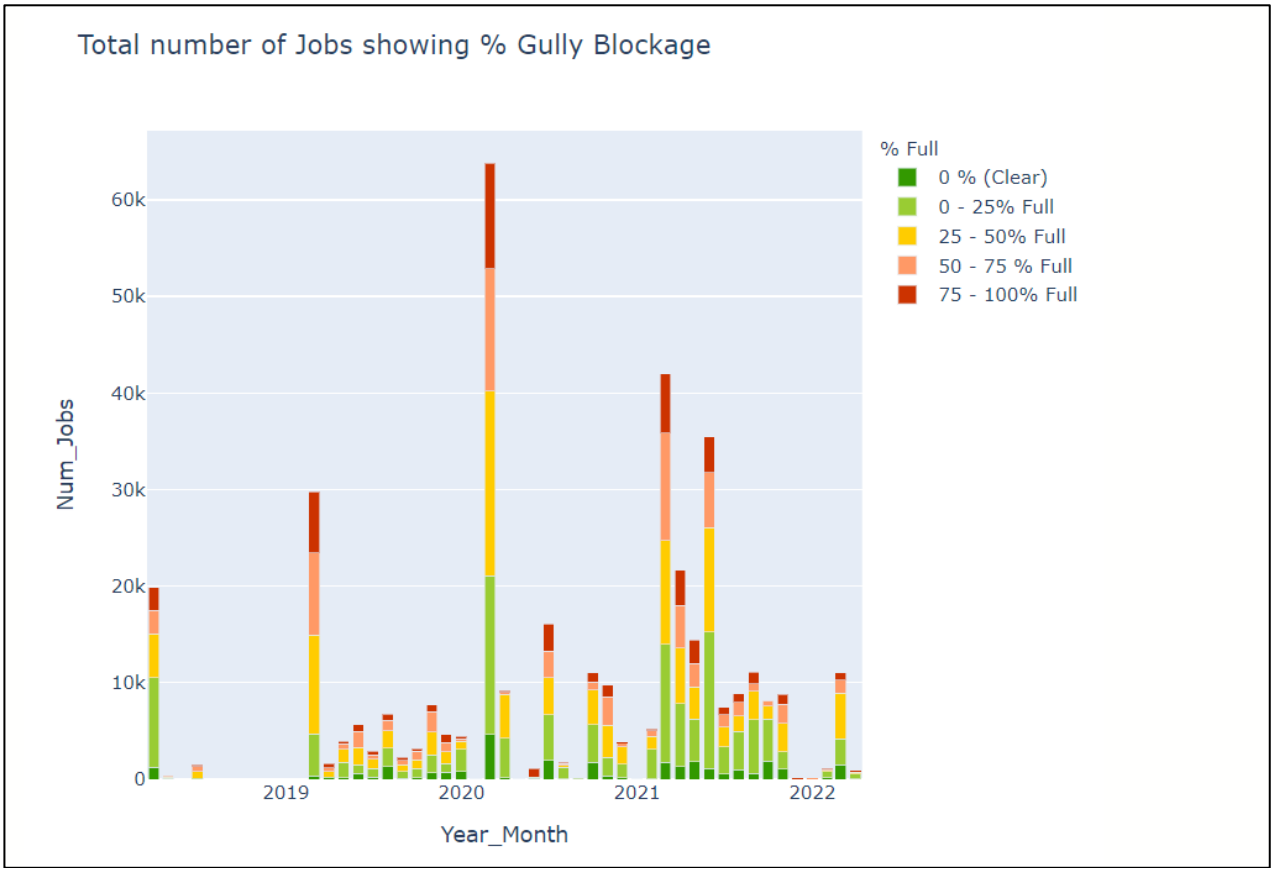


Similarly, we use a stacked bar chart to observe the gully congestion over the years and month as well just to observe if there are any outliers or huge number of jobs that were performed in a single month. The bar mode is going to be ‘stack’.

```
import plotly.express as px

# 3.3 Grouping by Year_Month & % Full
df_jobs_status_all = df_jobs2[(df_jobs2['% Full'] != 'Non Applicable') & (df_jobs2['% Full'] != 'Change Me')] \
                             .groupby(['Year_Month', '% Full']) \
                             .size() \
                             .reset_index(name='Num_Jobs')

df_jobs_status_all.head()

# Define figure
fig = px.bar( df_jobs_status_all, x='Year_Month', y='Num_Jobs',
              color='% Full',
              color_discrete_sequence=['#339900', "#99cc33", "#ffcc00", "#ff9966", "#cc3300"],   # 'black',
              title="Total number of Jobs showing % Gully Blockage",
             #  text_auto='.2s',
              barmode='stack',
              height=600
             )

fig.show()
```

Some of the notable outliers are observed in the months of April 2018, March 2019 and March 2020, which are essentially planned maintenances. The total number of jobs do not exceed 30,000 In a given month from 2018 to 2019, however more than 60,000 jobs were performed in the month of March 2020. We could note that there are considerable amount of asset failures in almost all the months, which should enable us to proceed with data modelling.

Figure 4.5 – Total number of jobs showing % Gully congestion (Stacked bar)



4.3.4.4 Number of assets maintained vs. the frequency of Maintenance (Histogram)

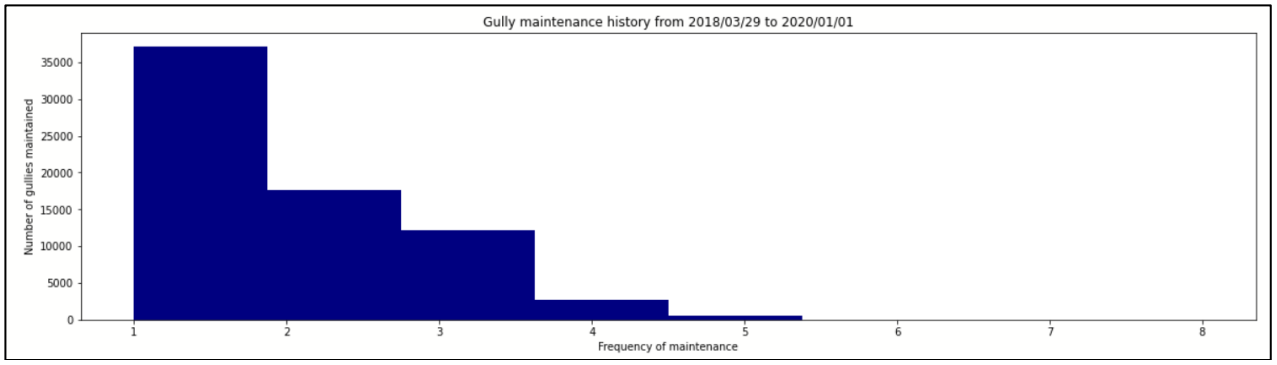
The total number of assets are investigated using the central asset ID and we could check the total number of bins of the histogram as the maximum number of maintenance jobs performed on a single gully. As we could see, there are a few gullies on which the maintenance was performed 6, 7 or even 8 times from 2018 to 2019.

3.4 Number of assets maintained against the frequency of maintenance

```
[33]  1  # Asset Maintenance History
      2  df_mt_rg = df_jobs_weather.groupby('CentralAssetID').size().reset_index(name='Count')
      3
      4  # Calculate the bins and dates
      5  bins = df_mt_rg.Count.max()
      6
      7  # Plot the histogram
      8  plt.xlabel("Frequency of maintenance")
      9  plt.ylabel("Number of gullies maintained")
     10  plt.title("Gully maintenance history from " + min_date + ' to ' + max_date)
     11  plt.hist(df_mt_rg.Count, bins=bins, color='navy')
     12
     13  plt.show()
```

Plotly histogram is shown below, which is essentially the gully maintenance history from March 2018 to Jan 2020.

Figure 4.6 – Histogram showing gully maintenance history (Frequency of maintenance)



4.3.4.5 Floods Status & Cyclical Frequency determination

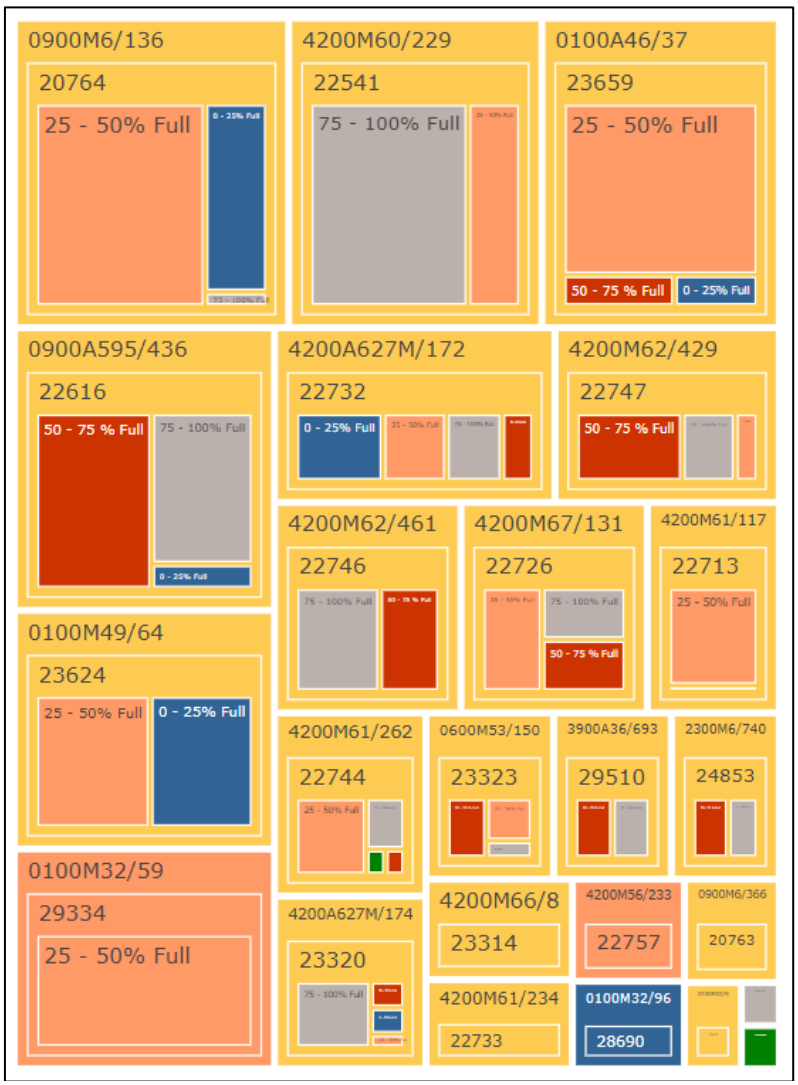
The flood status can be investigated for a gully only through a section ID instead of a single asset. A group of assets / gullies can be considered as a section, and the floods are recorded against gullies alone. A discrete color sequence is defined specially to identify asset failures (75 – 100% full) and potential asset failures (50 – 75 % full).

3.5 Floods Status & Cyclical Frequency determination

```
[35]  1  # 3.5 Grouping by Year_Month & Floods
      2  df_jobs_floods_grp = df_jobs_floods_f[(df_jobs_floods_f['% Full'] != 'Non Applicable') & (df_jobs_floods_f['% Full'] != 'Change Me')] \
      3                              .groupby(['Year_Month', 'FloodID', 'SectionID', '% Full']) \
      4                              .size() \
      5                              .reset_index(name='Num_Jobs')
      6
      7  fig3 = px.treemap(df_jobs_floods_grp,
      8                    path=[ 'SectionID', 'FloodID', '% Full'],
      9                    values='Num_Jobs',
     10                    color='% Full',
     11                    title="Jobs performed within 2 weeks after the floods were reported",
     12                    color_discrete_sequence=[ "#ff9966", "#bab0ac",  "green", "#cc3300", '#316395' , "#fecb52" , "#e2e2e2"],
     13                    height=800,
     14                    width=600)
     15
     16  fig3.show()
```

In order to enable the cyclical frequency, the potential asset failures are highlighted in ‘red’. The highways group uses this chart to prioritize the maintenance activities in the forthcoming years. The section ID is at the highest level of the tree map, for example – ‘0900M6/136’ is a section ID with the highest number of jobs performed 2 weeks after the flood has been reported there. The flood reported over there was flood ID – ‘20764’.

Figure 4.7 – Tree Map showing jobs performed within 2 weeks of floods being reported



4.3.4.6 Maintenance Costs

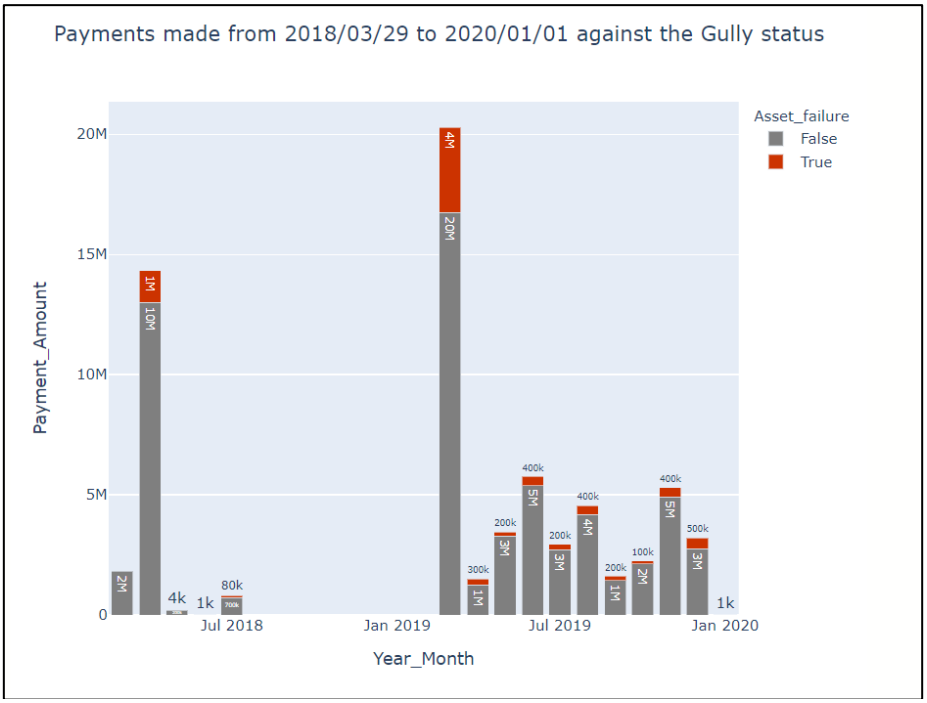
The maintenance costs can be identified in relation to the overall asset failures. By optimizing the planned and unplanned maintenance activity, the overhead cost of maintaining redundant gullies could be excluded, thus enabling the costs to cut back and utilize the savings elsewhere like red claims, etc.

3.6 Cost of Maintenance

```
[36]
# Payments made - Confidential Data
# Number of asset failures & other jobs against the cost of maintenance per month in a given month

# Asset Maintenance History
df_job_cost = df_jobs_weather[['Year_Month', 'Asset_failure', 'Payment_Amount', 'JobID']]

df_job_cost['Asset_failure'] = np.where(df_job_cost['Asset_failure'] == 0, False, True)

df_job_cost = pd.pivot_table(df_job_cost,
                             index=['Year_Month', 'Asset_failure'],
                             aggfunc={'Payment_Amount': np.sum, 'JobID': len}
                             ).rename(columns={'JobID': 'Num_Jobs'}).reset_index()

# Define figure
fig = px.bar( df_job_cost, x='Year_Month', y='Payment_Amount',
              color='Asset_failure',
              color_discrete_sequence=["#7f7f7f" , "#cc3300"],
              title=("Payments made from " + min_date + ' to ' + max_date + " against the Gully status"),
              text_auto='.1s',
              barmode='stack',
              height=600
             )

fig.show()
```

During the planned maintenances of April 2018 and March 2019, the cost component of around £ 5 million has been identified as useful maintenance, but around £ 30 million has been used for investigative purposes only of which 80% is absolutely redundant. All the months during which the jobs were performed were similarly proportional in terms of the costs and asset failures.

Figure 4.8 – Stacked bar chart showing jobs payments vs. asset failures



4.3.4.7 Correlation Analysis

The required weather-based aggregated features like average rain intensity, maximum rain intensity, wind speed, wind direction, wind gust, wind gust direction in addition to elevation. In order to visualize the correlation between asset failures and input features as parameters.

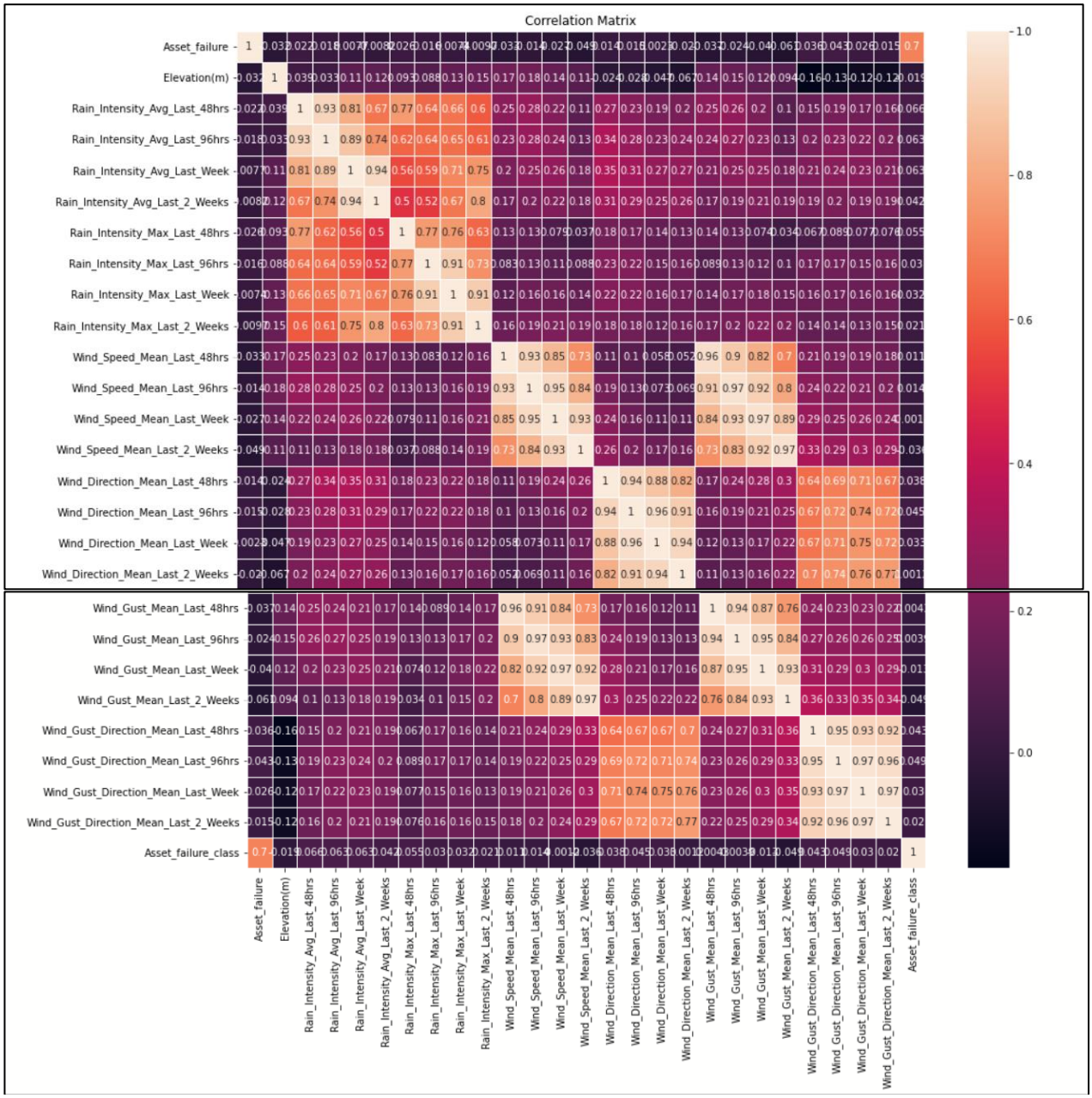
3.7 Correlation Analysis

[37]

```
# Subset the required columns into a dataframe
df_corr = df_jobs_weather[['% Full', 'Asset_failure', 'Elevation(m)', 'Rain_Intensity_Avg_Last_48hrs', 'Rain_Intensity_Avg_Last_96hrs', 'Rain_Intensity_Avg_Last_Week', \
'Rain_Intensity_Avg_Last_2_Weeks', 'Rain_Intensity_Max_Last_48hrs', 'Rain_Intensity_Max_Last_96hrs', \
'Rain_Intensity_Max_Last_Week', 'Rain_Intensity_Max_Last_2_Weeks', 'Wind_Speed_Mean_Last_48hrs', \
'Wind_Speed_Mean_Last_96hrs', 'Wind_Speed_Mean_Last_Week', 'Wind_Speed_Mean_Last_2_Weeks', \
'Wind_Direction_Mean_Last_48hrs', 'Wind_Direction_Mean_Last_96hrs', 'Wind_Direction_Mean_Last_Week', \
 'Wind_Direction_Mean_Last_2_Weeks', 'Wind_Gust_Mean_Last_48hrs', 'Wind_Gust_Mean_Last_96hrs', \
 'Wind_Gust_Mean_Last_Week', 'Wind_Gust_Mean_Last_2_Weeks', 'Wind_Gust_Direction_Mean_Last_48hrs', \
 'Wind_Gust_Direction_Mean_Last_96hrs', 'Wind_Gust_Direction_Mean_Last_Week', 'Wind_Gust_Direction_Mean_Last_2_Weeks' ]]

# Converting categorical variable (% Full) into an ordinal & quanititative variable
df_corr['Asset_failure_class'] = np.where(df_corr['% Full'] == '75 - 100% Full', 4,
                                          np.where(df_corr['% Full'] == '50 - 75% Full' , 3,
                                              np.where(df_corr['% Full'] ==  '25 - 50% Full' , 2,
                                                  np.where(df_corr['% Full'] ==  '0 - 25% Full' , 1,
                                                      np.where(df_corr['% Full'] ==  '0 % (Clear)' , 0, -1)))))

# Plot the correlation matrix
corrMatrix = df_corr.corr()
fig, ax = plt.subplots(figsize=(15,15))         # Sample figsize in inches
sn.heatmap(corrMatrix, annot=True, linewidths=.5, ax=ax).set(title='Correlation Matrix')
plt.show()
```

The correlation matrix can be visualized as follows. It seems that there is no major correlation between the individual features and the asset failure alone. Perhaps, the multivariate analysis or the usage of multiple variables within the supervised learning models could be used to establish a relationship between asset failures and weather-based data.

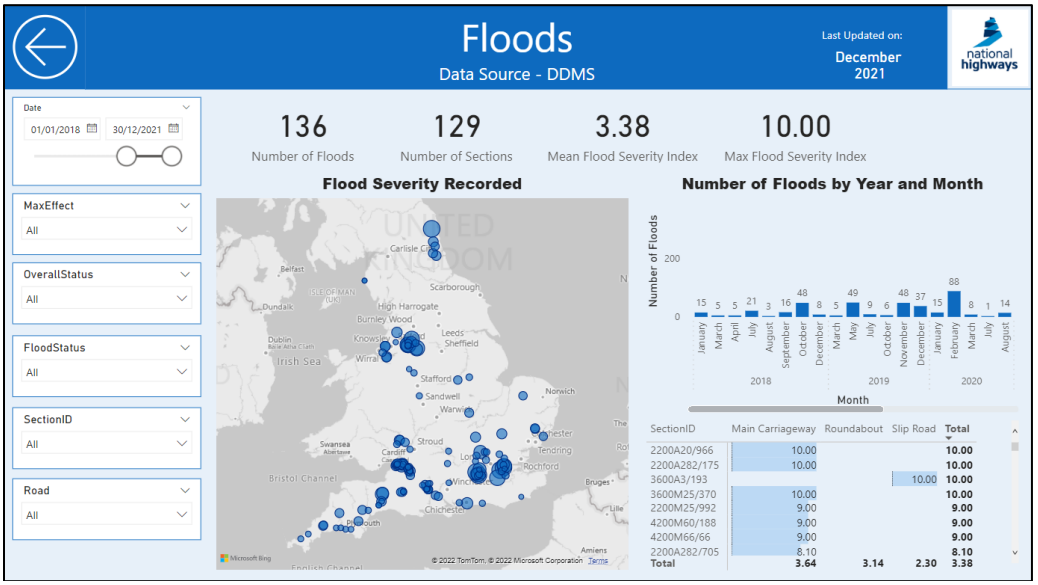
Figure 4.9 – Pearson Correlation Analysis



4.3.4.8 Spatial Analysis

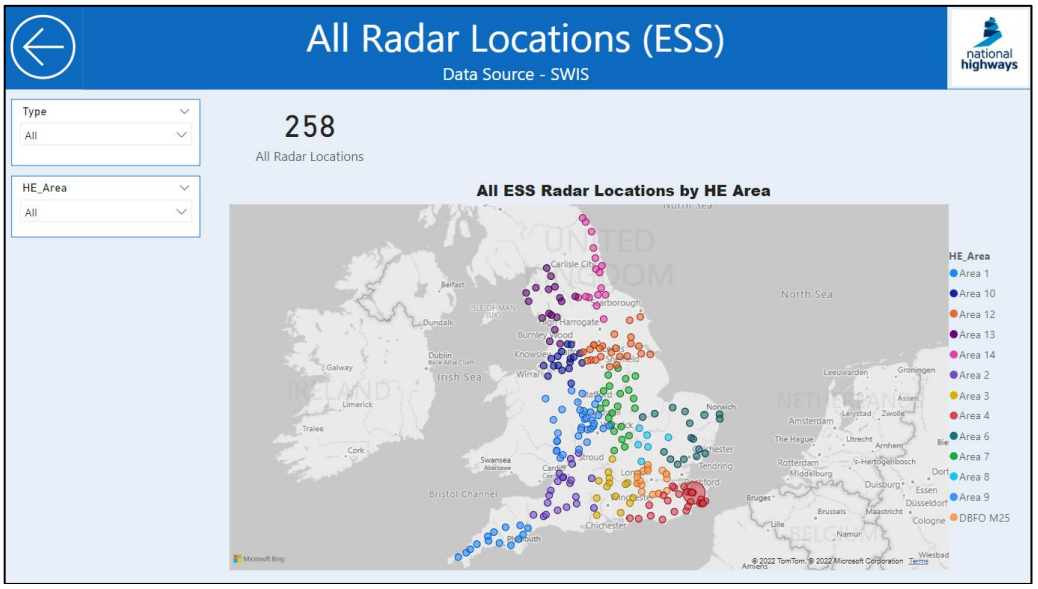
Microsoft Power BI was used to identify the geospatial locations using latitude and longitude values from within the cleaned dataset. Both floods and jobs were visualized to identify the flood incidents, flood hotspots, and also the asset locations of the 15 HE areas present across England. Using the drainage data based on the inputs available, we could note that the mean flood severity index recorded was 3.38 for a total of 129 sections with 136 floods overall. Flood severity has been highlighted in the map chart on the left-hand side of the report where we can observe the impact in terms of size of the bubble.

Figure 4.10 – Spatial Analysis – Floods data



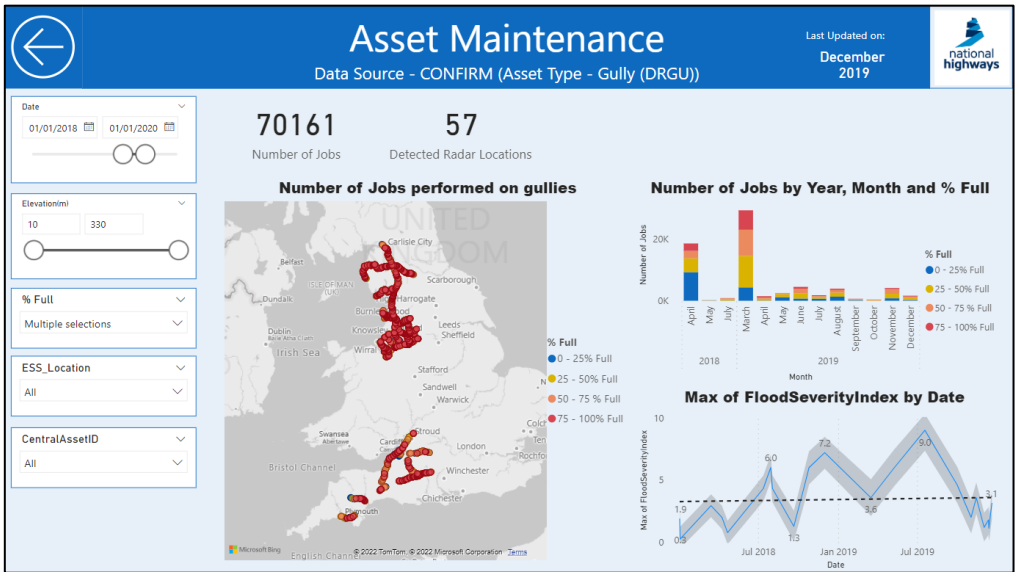
Of all the given 15 HE areas, we managed to identify the 258 Radar locations across England.

Figure 4.11 – Spatial Analysis – All the SWIS radar regions broken down by HE Areas



Asset maintenance tasks are also shown in a different sheet – the total number of jobs shown from 2018 to 2019 are 70161 for which the radar locations detected were only 57. This, albeit, a drawback, could be a game changer if the process of identifying the gully congestion could be implemented from the remaining HE areas as well. Since, this is a limitation, we could only reproduce the results based on data availability. Only 4 of 15 HE areas were associated with the process implemented and thus we have huge underlying potential for this data model.

Figure 4.12 – Spatial Analysis – Asset Maintenance



4.3.5 Data Preprocessing and cleansing

Data preprocessing and cleansing are critical components of EDA. Understanding the variables and the structure of the dataset is the initial stage in this process. The data must then be cleaned. The dataset may contain redundancy such as irregular data, missing values or outliers that may cause the model to overfit or underfit during training. Removing or resolving these redundancies is known as data cleaning. The last part is analyzing the relationship between the variables.

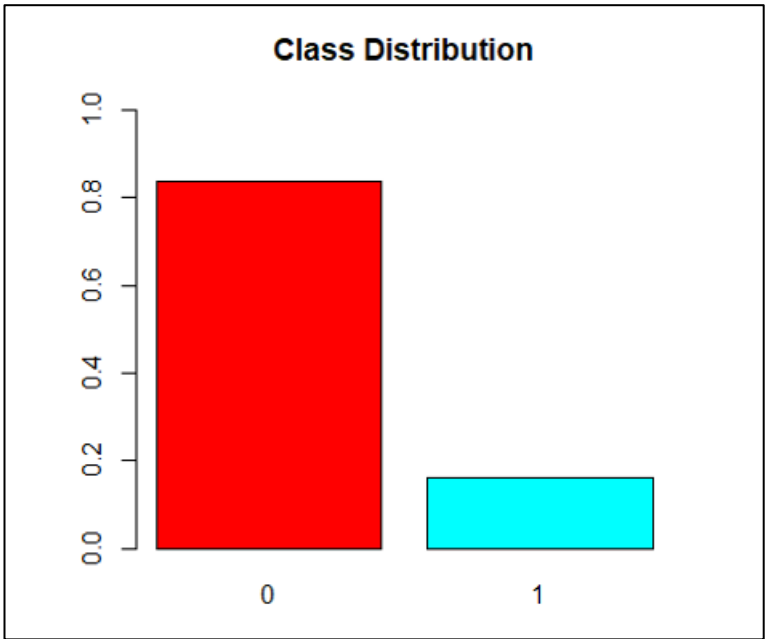
This stage is to examine the raw data for pre-processing through cleaning and transformation. Missing values, irrelevant columns, imbalanced data were all determined and resolved. This stage has been repeated and re-examined to increase the model's performance and accuracy.

4.4 Data Imbalance

Imbalanced data is predominant within highways datasets. There is a percentage of anomalous data points; this can contribute to complications when building a model. When one out of the two classes have drastically higher instances than the other causing the model to

have a bias towards the majority class (Li, Liu and Hu, 2010). In this dataset, there is imbalance within the instances as 83.6% of assets did not fail and 17.4% were failures. Figure 4.19 illustrates the imbalance within the asset failure variable. This can be rectified by applying the imblearn module that provides methods such as Oversampling, Undersampling, Combined Oversampling and Undersampling, and by selecting which method has a higher accuracy to form a balanced dataset.

Figure 4.13 - Bar Graph displaying the imbalance within asset failure class



Imbalanced-Learn is a bootstrap-based approach in which new instances are created for the minority class. The module employs conventional solutions to create a perfect balance between the two classes such as oversampling, undersampling, and combined over- and under- sampling. Oversampling increases the instances in the minority class by random selecting for duplication. On the contrary, undersampling decreases instances from the majority class by rejection hence reducing the imbalance ratio. Combination of over- and under- sampling allowing both processes to perform together or consecutively (Vluymans, 2018). The ***RandomOverSampler()*** function was applied to conduct balancing methods. Table 4.2 demonstrates the distribution of data within the original dataset compared to the three methods of reducing imbalance ratio.

Table 4.2 Imbalance within the asset failure class variable

Method	No failure (0)	Failure exists (1)
Original Dataset	76886	13436
Oversampling	76886	76886
Undersampling	13436	13436
Combined Over and Under	31212	30657

4.5 Feature Selection

Feature selection focuses on designing and selecting attributes of a dataset that are valuable for creating a successful predictor. Large data sets are mostly represented with a lot of features, hence are not practical for developing a machine learning model (Guyon and Elisseeff, 2003). Preparation of big datasets slows down operating machines, causes resource waste, and troublesome. The most significant limiting factor of implementing algorithms across a broad set of features is the model performance accuracy declines. Thus, minimal collection of variables within a dataset must be chosen an optimal outcome (Kursa and Rudnicki, 2010). Feature selection is a pre-processing filter step which explores the noisy data and removes any irrelevant and redundant information. Despite the dimensionality of the data reducing, the algorithm can process and perform better. Algorithms such as decision trees tend to have overfitting of their untrained data creating inaccurate predictions (Hall, 1999).

4.6 Summary

This chapter focused on exploring and analyzing relationships within the dataset using various visualizations to have a deeper understanding for the research. Understanding of the data enabled the recognition of problems and the development of solutions which is prepared for modelling. Preparation of the data was addressed with the use of transformation, cleansing and handling imbalanced data. Finally, feature selection was explained and how it improved

the selection of attributes by assessing its relevance for this research. Processing data techniques were discussed, and the final dataset is available for model application. However, successful variable selection enables the elimination of irrelevant and redundant details ensuring improved forecasts.

Chapter 5 MODELLING

5.1 Introduction

This chapter will discuss the machine learning methods implemented on the jobs_weather dataset to analyze predictions on the rate of asset failures. This is the fourth phase of the CRISP-DM methodology after exploring and pre-processing of the data. Furthermore, a comprehensive analysis of the development of models will be included and the stages employed to determine the optimal approach for accurate results.

5.2 Model Development

This section describes the classification and regression algorithms implemented to build a prediction model. Before building the model, the dataset was divided into training and test sets with 75:25 ratio. For classification using decision tree, the dimension of the training set consisted of 2,427 observations and the test set contained 809 observations. For logistic regression, the dimension of the training set consisted of 67,741 observations and the test set contained 22,581 observations. The response variables 'asset_failure_class' and "asset_failure" were converted to factor variables. Balancing methods were implemented on the training data to overcome imbalance and achieve better accuracy.

5.2.1 Decision Tree Model

Decision Tree is a supervised classification algorithm, applied in predictive modelling to display every potential outcome of a decision. The training model is built to predict the target class by learning decision-making rules extracted from historical data (training data). The sklearn.tree library has been selected for the ***DecisionTreeClassifier()*** function because of its unbiased recursive partitioning nature.

The decision tree modelling was implemented using the function ***DecisionTreeClassifier()*** to predict the response variable against all the predictors. The formula and data are the key elements of this function, which pertains to the model where 'y' or the asset failure is the

response variable, and ‘x’ refers to all the predictors which are the decision nodes. The data specifies the dataset to retrieve the predictor variables listed in the formula. A random state is defined to ensure that the random sample generated is repeated on different machines.

```
[38]
# Classification
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, confusion_matrix

# Subset the required columns into a dataframe
df_class = df_jobs_weather[['Asset_failure_class', 'Elevation(m)', 'Rain_Intensity_Avg_Last_48hrs', 'Rain_Intensity_Avg_Last_96hrs', 'Rain_Intensity_Avg_Last_Week', \
'Rain_Intensity_Avg_Last_2_Weeks', 'Rain_Intensity_Max_Last_48hrs', 'Rain_Intensity_Max_Last_96hrs', \
'Rain_Intensity_Max_Last_Week', 'Rain_Intensity_Max_Last_2_Weeks', 'Wind_Speed_Mean_Last_48hrs', \
'Wind_Speed_Mean_Last_96hrs', 'Wind_Speed_Mean_Last_Week', 'Wind_Speed_Mean_Last_2_Weeks', \
'Wind_Direction_Mean_Last_48hrs', 'Wind_Direction_Mean_Last_96hrs', 'Wind_Direction_Mean_Last_Week', \
 'Wind_Direction_Mean_Last_2_Weeks', 'Wind_Gust_Mean_Last_48hrs', 'Wind_Gust_Mean_Last_96hrs', \
 'Wind_Gust_Mean_Last_Week', 'Wind_Gust_Mean_Last_2_Weeks', 'Wind_Gust_Direction_Mean_Last_48hrs', \
 'Wind_Gust_Direction_Mean_Last_96hrs', 'Wind_Gust_Direction_Mean_Last_Week', 'Wind_Gust_Direction_Mean_Last_2_Weeks' ]]

# Data Transformations - Replace & Impute blank values with resp. medians and drop the duplicates
df_class = df_class[df_class['Asset_failure_class'] != 'NA'].replace('', np.nan).fillna(df_class.median()).drop_duplicates()

# Data Preprocessing
x = df_class.drop('Asset_failure_class', axis=1)       # dep
y = df_class['Asset_failure_class']                    # indep
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25, random_state=28)      # Model Preparation

print('################ Classification ################')

print('----> Value Counts:', '\n', df_class['Asset_failure_class'].value_counts())
print('----> Number of records in Training dataset:', len(x_train), 'and Number of records in Testing dataset:', len(x_test))

# Model training
classifier = DecisionTreeClassifier()
classifier.fit(x_train, y_train)
```

For evaluation, we use the following script. Firstly, we evaluate the training dataset, and later testing dataset and compare the results.

```

# Evaluation using Training data
y_pred = classifier.predict(x_train)
print('************************', '\n', 'Training Dataset Evaluation: ' , '\n', '************************')
print('---> Confusion Matrix: ', '\n' , confusion_matrix(y_train, y_pred))
print('---> Classification Report: ', '\n' , classification_report(y_train, y_pred))

# Evaluation using Testing data
y_pred = classifier.predict(x_test)
print('************************', '\n', 'Testing Dataset Evaluation: ' , '\n', '************************')
print('---> Confusion Matrix: ', '\n' , confusion_matrix(y_test, y_pred))
print('---> Classification Report: ', '\n' , classification_report(y_test, y_pred))
```

Using the ***confusion_matrix()*** function from the sklearn.metrics module, the model was evaluated using the test set. A plot of the resulting tree cannot be displayed since the results were substandard. The accuracy of the model for the testing set is 11% with a misclassification error of 89%. Due to the high misclassification error rate, feature selection was applied to boost the model’s accuracy, but in vain.

Another factor for the poor accuracy of the model is having insignificant features in the data hence the model's learning is based on these features. Alternatively, the Boruta() function can be utilized to implement feature selection to select most important variables relevant to the prediction variable. There were a series of trial-and-error tests performed using this model, but all the results were pointing towards inadequacy in terms of accuracy and precision.

```
################## Classification ##################
----> Value Counts:
 L1    1131
L0    1112
L2     993
Name: Asset_failure_class, dtype: int64
----> Number of records in Training dataset: 2427 and Number of records in Testing dataset: 809
************************
 Training Dataset Evaluation:
 ************************
---> Confusion Matrix:
 [[832   0   0]
 [507 341   0]
 [398 181 168]]
---> Classification Report:
              precision    recall  f1-score   support

          L0       0.48      1.00      0.65       832
          L1       0.65      0.40      0.50       848
          L2       1.00      0.22      0.37       747

    accuracy                           0.55      2427
   macro avg       0.71      0.54      0.50      2427
weighted avg       0.70      0.55      0.51      2427
```

```
************************
 Testing Dataset Evaluation:
 ************************
---> Confusion Matrix:
 [[ 45 173  62]
 [196  23  64]
 [178  57  11]]
---> Classification Report:
              precision    recall  f1-score   support

          L0       0.11      0.16      0.13       280
          L1       0.09      0.08      0.09       283
          L2       0.08      0.04      0.06       246

    accuracy                           0.10       809
   macro avg       0.09      0.10      0.09       809
weighted avg       0.09      0.10      0.09       809
```

5.2.2 Logistic Regression Model

Multiple logistic regression is the prediction of observations that are classified into one type of category from a binary variable based on other predictor variables. The model would be structured to include insights on whether gullies are likely to get clogged due to changes in weather and ultimately fail.

The model is built using the function ***LogisticRegression()*** which fits generalized linear models. As shown in previous functions, the formula refers to the model where asset failure is the outcome variable, and terms applies to all the linear predictors. The parameters also set the data frame as the training set and the family type is stated as binomial. This specifies the distribution to be binomial. The summary of the model is shown in the below figure.

The nulls and nan constitute of around 10% of the data, and all of them are replaced by the corresponding feature medians in order to impute the data which is skewed.

5.1 Model building and evaluation

```
# Logistic Regression
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix

# Subset the required columns into a dataframe
df_logr = df_jobs_weather[df_jobs_weather['% Full'] != 'Change Me']
df_logr = df_logr[['Asset_failure', 'Elevation(m)', 'Rain_Intensity_Avg_Last_48hrs', 'Rain_Intensity_Avg_Last_96hrs', 'Rain_Intensity_Avg_Last_Week', \
'Rain_Intensity_Avg_Last_2_Weeks', 'Rain_Intensity_Max_Last_48hrs', 'Rain_Intensity_Max_Last_96hrs', \
'Rain_Intensity_Max_Last_Week', 'Rain_Intensity_Max_Last_2_Weeks', 'Wind_Speed_Mean_Last_48hrs', \
'Wind_Speed_Mean_Last_96hrs', 'Wind_Speed_Mean_Last_Week', 'Wind_Speed_Mean_Last_2_Weeks', \
'Wind_Direction_Mean_Last_48hrs', 'Wind_Direction_Mean_Last_96hrs', 'Wind_Direction_Mean_Last_Week', \
 'Wind_Direction_Mean_Last_2_Weeks', 'Wind_Gust_Mean_Last_48hrs', 'Wind_Gust_Mean_Last_96hrs', \
 'Wind_Gust_Mean_Last_Week', 'Wind_Gust_Mean_Last_2_Weeks', 'Wind_Gust_Direction_Mean_Last_48hrs', \
 'Wind_Gust_Direction_Mean_Last_96hrs', 'Wind_Gust_Direction_Mean_Last_Week', 'Wind_Gust_Direction_Mean_Last_2_Weeks' ]]

# Data Transformations - Replace & Impute blank values with resp. medians and drop the duplicates
df_logr = df_logr.replace('', np.nan).fillna(df_logr.median())

# Data Preprocessing
x = df_logr.drop('Asset_failure', axis=1)       # dep
y = df_logr['Asset_failure']                    # indep
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25, random_state=28)      # Model Preparation

print('################# Logistic Regression #################')

print('----> Value Counts:', '\n', df_logr['Asset_failure'].value_counts())
print('----> Number of records in Training dataset:', len(x_train), 'and Number of records in Testing dataset:', len(x_test))

```

Predictor variables and response variables are defined, and later the training and test data are split into 75:25 ratio again.

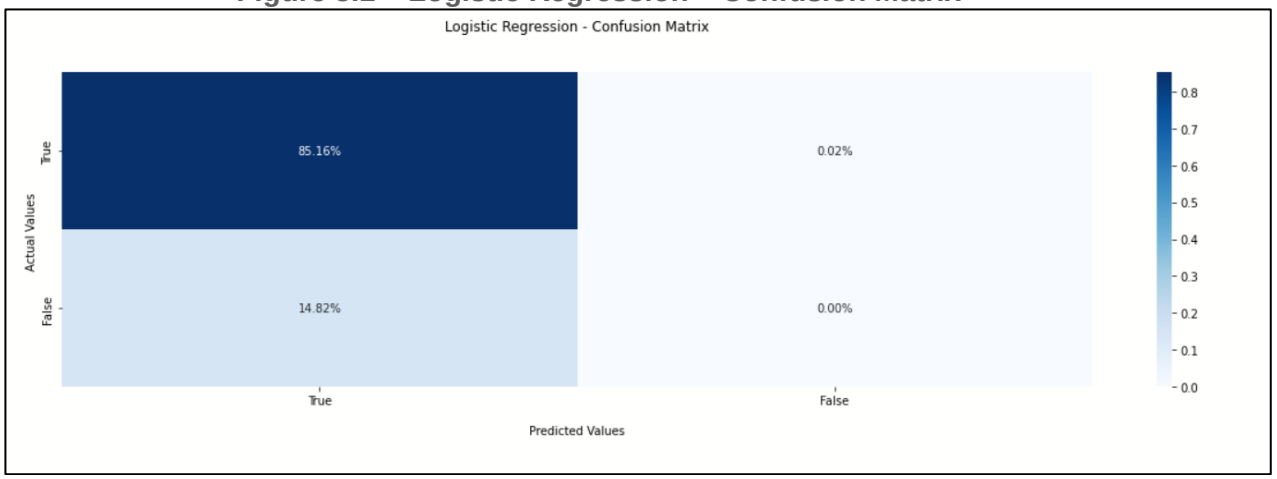
```
# Model training
logisticRegr = LogisticRegression()
logisticRegr.fit(x_train, y_train)

# Determine Accuracy of Training data
score = logisticRegr.score(x_train, y_train)
print('----> Model Accuracy (Training data) :', round(score, 4)*100, '%' )

# Determine Accuracy of Testing data
score = logisticRegr.score(x_test, y_test)
print('----> Model Accuracy (Testing data) :', round(score, 4)*100, '%' )

# Predictive Analysis (for testing data)
y_pred = logisticRegr.predict(x_test)
score = logisticRegr.score(x_test, y_pred)
print('----> Model Prediction :', round(score, 4)*100, '%' )

# Confusion matrix
confusion_matrix = confusion_matrix(y_test, y_pred)
print('----> Confusion Matrix : ', '\n', confusion_matrix)

# Plot the confusion matrix
ax = sn.heatmap(confusion_matrix/np.sum(confusion_matrix), annot=True, fmt='.2%', cmap='Blues')
ax.set_title('Logistic Regression - Confusion Matrix\n\n');
ax.set_xlabel('\nPredicted Values')
ax.set_ylabel('Actual Values ')
plt.figure(1, figsize=(2, 2))
ax.xaxis.set_ticklabels(['True','False'])
ax.yaxis.set_ticklabels(['True','False'])
plt.show()
```

The overall model accuracy is around 85% for both training and testing data, and upon oversampling and undersampling the range changes from 84% to 92%.

```
################## Logistic Regression ##################
----> Value Counts:
 0    76886
1    13436
Name: Asset_failure, dtype: int64
----> Number of records in Training dataset: 67741 and Number of records in Testing dataset: 22581

----> Model Accuracy (Training data) : 85.09 %
----> Model Accuracy (Testing data) : 85.16 %
----> Model Prediction : 100.0 %
----> Confusion Matrix :
 [[19230     5]
 [ 3346     0]]
```

A visual confusion matrix has been created to highlight the accuracy of the model. The model overall is still unable to interpret false negatives as well, which means that additional data needs to be fed to the model (data from 2020) onwards and then utilize the weather parameters to predict the asset failure class.

Figure 5.1 – Logistic Regression – Confusion Matrix



A second level prediction for uncategorized asset failure classes – 'NA' or 'Not Known'. Consequentially, for both these categories, the weather-based prediction has proved to return 100% accuracy, emphasizing that the weather didn't impact these gullies much and they need not have been part of the maintenance regime.

5.2 Predictive Analysis

```
[41]
# Subset the required columns into a dataframe
df_logr_u = df_jobs_weather[(df_jobs_weather['% Full'] == 'Change Me') | (df_jobs_weather['% Full'] == 'Not Applicable')]
df_logr_u = df_logr_u[['Elevation(m)', 'Rain_Intensity_Avg_Last_48hrs', 'Rain_Intensity_Avg_Last_96hrs', 'Rain_Intensity_Avg_Last_Week', \
'Rain_Intensity_Avg_Last_2_Weeks', 'Rain_Intensity_Max_Last_48hrs', 'Rain_Intensity_Max_Last_96hrs', \
'Rain_Intensity_Max_Last_Week', 'Rain_Intensity_Max_Last_2_Weeks', 'Wind_Speed_Mean_Last_48hrs', \
'Wind_Speed_Mean_Last_96hrs', 'Wind_Speed_Mean_Last_Week', 'Wind_Speed_Mean_Last_2_Weeks', \
'Wind_Direction_Mean_Last_48hrs', 'Wind_Direction_Mean_Last_96hrs', 'Wind_Direction_Mean_Last_Week', \
 'Wind_Direction_Mean_Last_2_Weeks', 'Wind_Gust_Mean_Last_48hrs', 'Wind_Gust_Mean_Last_96hrs', \
 'Wind_Gust_Mean_Last_Week', 'Wind_Gust_Mean_Last_2_Weeks', 'Wind_Gust_Direction_Mean_Last_48hrs', \
 'Wind_Gust_Direction_Mean_Last_96hrs', 'Wind_Gust_Direction_Mean_Last_Week', 'Wind_Gust_Direction_Mean_Last_2_Weeks' ]]

# Data Transformations - Replace & Impute blank values with resp. medians and drop the duplicates
df_logr_u = df_logr_u.replace('', np.nan).fillna(df_logr_u.median())

# Predictive Analysis (for uncategorized data)
y_pred = logisticRegr.predict(df_logr_u)
score = logisticRegr.score(df_logr_u, y_pred)
print('----> Model Prediction for Uncategorized data:', round(score, 4)*100, '%' )
```

```
----> Model Prediction for Uncategorized data: 100.0 %
```

5.2.3 Anomaly Detection

Isolation Forests (IF), like Random Forests, are built based on decision trees. And since there are no pre-defined labels here, it is an unsupervised model. After an ensemble of iTrees(Isolation Forest) is created, model training is complete. During scoring, a data point is traversed through all the trees which were trained earlier. Now, an 'anomaly score' is assigned to each of the data

points based on the depth of the tree required to arrive at that point. This score is an aggregation of the depth obtained from each of the iTrees. An anomaly score of -1 is assigned to anomalies and 1 to normal points based on the contamination (percentage of anomalies present in the data) parameter provided.

From sklearn.ensemble module, we import **IsolationForest()** function which is used to identify the flood severity that could be considered as an outlier, in relation to the asset failures. Model fitting and model predictions are used to evaluate the outcomes, and to enable the color coding for asset failures (in terms of sizes), we use the value of 0.1 as asset not failed and 0.9 as asset failed which would become quantifiable and thus could be applied to size as a parameter.

```
from sklearn.ensemble import IsolationForest

# Subset the requried data for anomaly detection
df_anom_det = df_jobs_floods_f[['FloodID', 'ReportedDateTime', 'FloodSeverityIndex', 'Asset_failure']]

# Necessary Transformations
df_anom_det['ReportedDateTime'] = pd.to_datetime(df_anom_det['ReportedDateTime'])
df_anom_det['Year'] = df_anom_det["ReportedDateTime"].dt.year
df_anom_det = df_anom_det.set_index('ReportedDateTime')

# Filter the data from before the Year 2013
df_anom_det = df_anom_det[df_anom_det['Year'] >= 2013 ]
del df_anom_det['Year']

# Create the model and fit the data
model = IsolationForest()
model.fit(df_anom_det)
predictions = model.predict(df_anom_det)
df_anom_det['predictions'] = predictions

# Renaming for identification
df_anom_det['Predicted'] = np.where(df_anom_det['predictions'] == -1, 'Outliers', 'Inliers')
df_anom_det['Asset_failure_type'] = np.where(df_anom_det['Asset_failure'] == 0, 'Not failed', 'Failed')
df_anom_det['Asset_failure'] = np.where(df_anom_det['Asset_failure'] == 0, 0.1, 0.9)

print('################## Anomaly Detection ##################')

# How many Asset failures / Gullies fully clogged ?
print('-------> Asset Failures:' , '\n' , df_anom_det.Asset_failure_type.value_counts())

# How many outliers found?  1 - Inliers and -1 - Outliers
print('-------> Outliers vs. Inliers:' , '\n' , df_anom_det['predictions'].value_counts())
```

A quick summary statistic is observed for the total number of jobs related to floods where an asset has failed or not. A scatter plot has been created to view the outliers vs. inliers (which are distinctly color-coded) and the larger size of the bubble implies the asset failure.

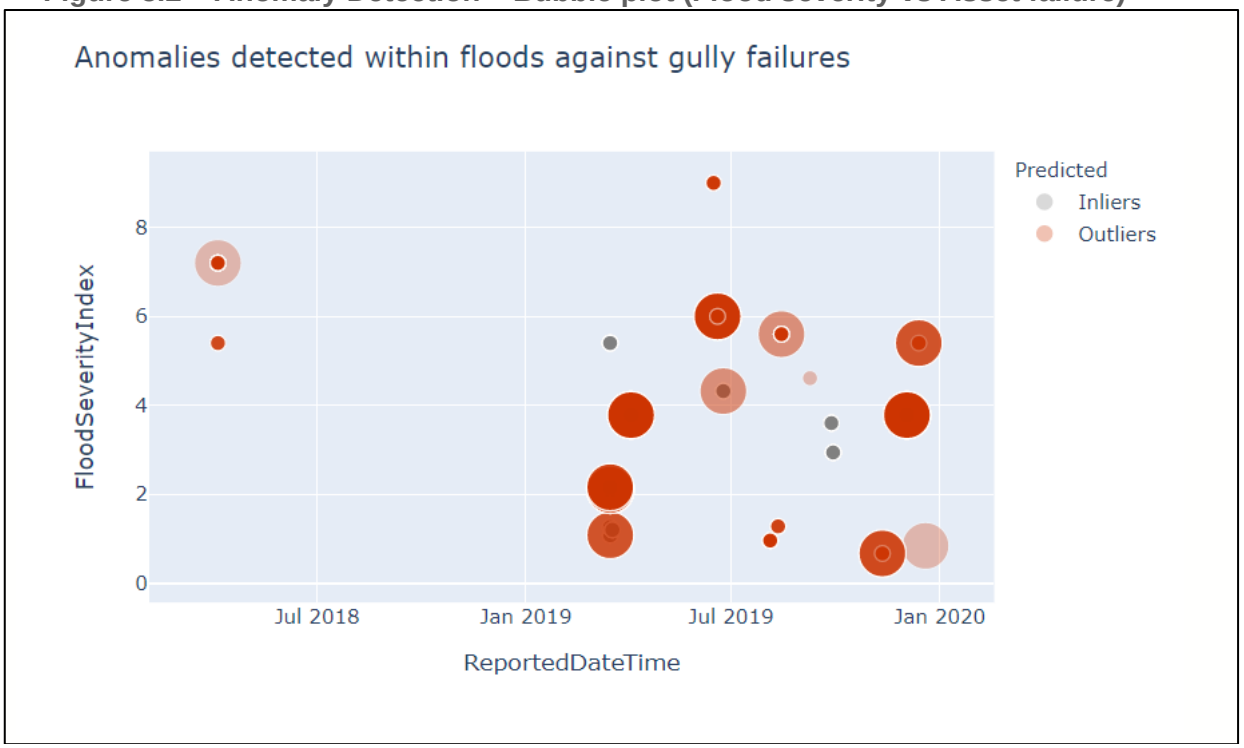
```
# #summary statistics
df_anom_det['FloodSeverityIndex'].describe()

# Scatter Plot
fig = px.scatter(df_anom_det,
                         x=df_anom_det.index,
                         y="FloodSeverityIndex",
                         color="Predicted",
                         size="Asset_failure",
                         opacity = 0.3,
                         color_discrete_sequence=[ "#7f7f7f", "#cc3300"],
                         title=("Anomalies detected within floods against gully failures"))

fig.show()
```

The outcome of the summary statistics is shown as below.

```
################## Anomaly Detection ##################
-------> Asset Failures:
 Not failed    721
Failed         122
Name: Asset_failure_type, dtype: int64
-------> Outliers vs. Inliers:
  1    565
-1    278
Name: predictions, dtype: int64
```

The outcome chart for the anomalies detected within floods against gully failures is shown below.

Figure 5.2 – Anomaly Detection – Bubble plot (Flood severity vs Asset failure)



5.3 Summary

This chapter detailed the implementation of classification models including all the stages and reasoning for the development of this project. Furthermore, visualizations of the analysis were presented, prepared for evaluation in the next chapter.

Chapter 6 TESTING AND EVALUATION

6.1 Introduction

Following the development of the classification modelling, the accuracy and predictive capability are evaluated to determine whether the constructed models can adapt to new data without overfitting the training data and if they are successful in solving the defined problem. Using metrics for performance evaluation, such as the confusion matrix along with each model's accuracy, will be examined in this chapter. The Appendix will include all model's results and informative visualizations of the outputs.

6.2 Evaluation Metrics

Performance measurement for the classification models is evaluated by using a confusion matrix. A confusion matrix is a table summarizing the performance of a classifier to identify true values from a test dataset. The terms used for the prediction values are displayed in Figure 6.1 and as follows:

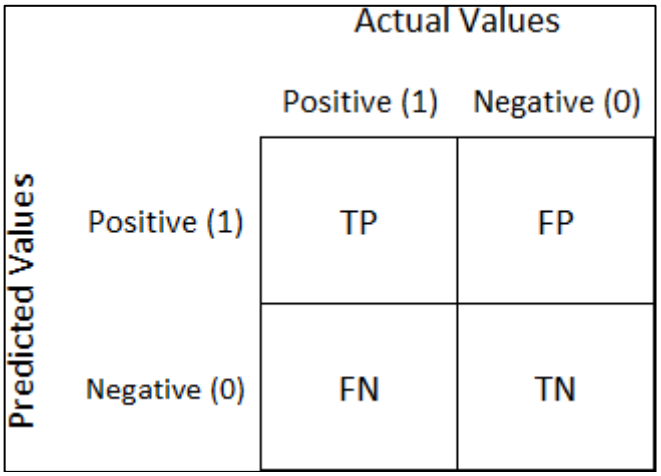
1. True Positive (TP) – This refers to the positive tuples that have been correctly classified by the model.

2. True Negative (TN) – This is the negative tuples that have been correctly classified by the model.

3. False Positive (FP) – These values have been incorrectly classified as positive but are negative tuples.

4. False Negative (FN) – These values have been mislabeled as negative but are positive tuples.

Figure 6.1 Confusion Matrix (Towards Data Science, 2018)



The accuracy of the model is how many classes were correctly predicted. It is calculated using this equation:

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

Precision is the percentage of the findings that are significant. An effective classifier will have a high level of precision. It is expressed as:

$$Precision\ (P) = \frac{TP}{TP + FP}$$

Recall corresponds to the percentage of total relevant labels correctly classified by the classifier. Also termed as Sensitivity, the calculation is shown below:

$$Recall\ (R) = \frac{TP}{TP + FN}$$

F1 score is determined based on both precision and recall values to compute a score. A high F1 score is achieved by having high precision and recall values. The F1 value hits its highest value at 1 and its lowest value at 0. The formula for F1 is as follows:

$$F_1 = 2 \times \frac{P \times R}{P + R}$$

A receiving operating characteristic curve (ROC) is a schematic graph that represents the analytical performance of the classifier method. The curve is created by plotting the true positive rate (TPR) and false positive rate (FPR) at various classification settings. The area under the curve (AUC) is determined by the cumulative area under the ROC curve which shows how much the model can predicts the classes successfully. A strong model with AUC closer to 1 implies that it has the better separability metric.

6.3 Model Evaluation

The sklearn package was used to obtain the ***confusionMatrix()*** function. The cross- tabulation of the actual and predicted classes allows to analyse the performance of each model in order to determine the best algorithm.

Table 6.1 Model Evaluation Results using various Balancing Methods

Model	*Method*	*Accuracy*	*F_1 score*	*AUC*
Classification - Decision Tree	Oversampling	25.21%	0.156	0.189
	Undersampling	6.72%	0.052	0.231
	Combined	11.2%	0.127	0.207
Logistic Regression	Oversampling	92.1%	0.709	0.767
	Undersampling	84.47%	0.698	0.783
	Combined	85.14%	0.716	0.785

The accuracy of each model was extracted from the confusion matrix for comparison. The F_1 score was calculated using the precision and recall values, and the AUC was computed using the ROC package.

Figure 6.2 ROC Curve for Logistic Regression with Test dataset

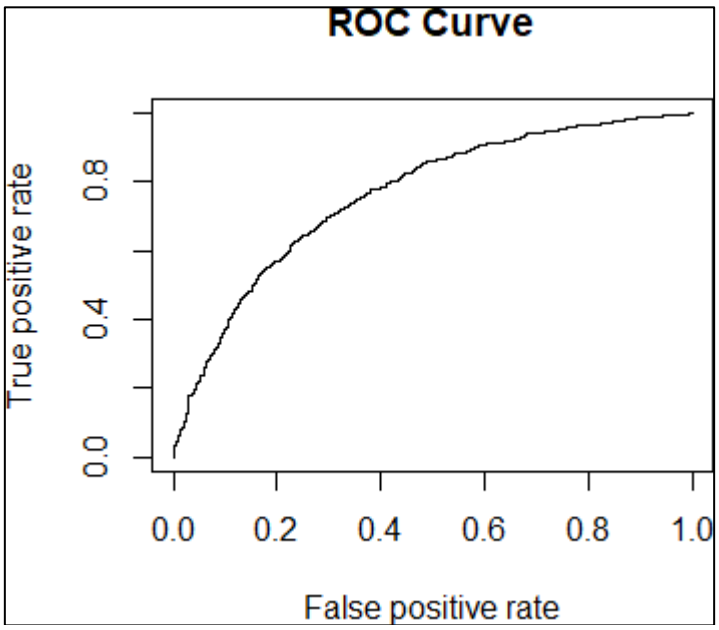


Figure 6.2 is a plot of the ROC curve for logistic regression on the test set. The graph presents the TPR against the FPR. The TPR value at the operating threshold is approximately in between 0.84 to 0.92.

6.4 Model Assessment

Understanding the features of binary classification metrics is essential when selecting the best model. The best model is chosen based on the output of each balancing method with new data. The accuracy of a good model will be closer to 100% while F_1 score and AUC of a model with good predictive capability should be closer to 1. Taking into consideration the dataset, the main issue was that it was highly imbalanced. This causes our model to be successful at estimating true negatives by default, and thus the accuracy will be high. The focus is to correctly predict an asset failure on which the sensitivity parameter is based. According to this, the oversampling method had the lowest sensitivity across all models which contradicts the application of the algorithm and research aims. The undersampling method of the logistic regression model had the highest accuracy. Overall, the logistic regression model has the best performance in all balancing approaches than the other two models.

6.5 Summary

This chapter presented classification algorithms and balancing methods implemented on imbalanced datasets. Various evaluation metrics were discussed to be applied that are considered to perform well for the issue described in this research. Based on the findings, the combined over- and under- sampling method produces the best result for TPR and FPR and the logistic regression model performed best.

REFERENCES

Explainable Anomaly Detection Framework for Maritime Main Engine Sensor Data - Donghyun Kim 1, Gian Antariksa 2, Melia Putri Handayani 2, Sangbong Lee 3 and Jihwan Lee

Alpaydin, E., 2014. *Introduction To Machine Learning*. 3rd ed. pp.1-14. Alpaydin, E., 2020. *Introduction To Machine Learning*. 4th ed. pp.3-15.
Ayodele, T.O., 2010. Types of machine learning algorithms. *New advances in machine learning*, pp.19-48.

https://www.usna.edu/Users/oceano/pguth/md_help/html/approx_equivalents.htm

https://www.data.gov.uk/ – National Statistics

https://community.esri.com/t5/coordinate-reference-systems-blog/distance-on-a-sphere-the-haversine-formula/ba-p/902128

Buuren, S., 2020. *Multivariate Imputation By Chained Equations*.

Carbonell, J., Michalski, R. and Mitchell, T., 2014. *Machine Learning: An Artificial Intelligence Approach*. 1st ed. Morgan Kaufmann Publishers, Inc., pp.3-5.

Chapelle, O., Schȍlkopf, B. and Zien, A., 2006. *Semi-Supervised Learning*. Cambridge, MA.: MIT Press, p.11.

Cutler, A., Cutler, R. and Stevens, J., 2012. Chapter 6 Isolation Forests. *Ensemble Machine Learning*, pp.186-188.

Dai, D. & Hua, S. 2016, *Random Under-Sampling Ensemble Methods for Highly Imbalanced Rare Disease Classification*, The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), Athens.

Delen, D., 2009. Analysis of cancer data: a data mining approach. *Expert Systems*, 26(1), pp.100-112.

Fayyad, U., Piatetsky-Shapiro, G. and Smyth, P., 1996. From Data Mining to Knowledge Discovery in Databases. *AI Magazine*, 17(3), p. 37-54.

Guyon, I. and Elisseeff, A., 2003. An introduction to variable and feature selection. *Journal of machine learning research*, *3*(Mar), pp.1157-1182.

Hall, M., 1999. *Correlation-Based Feature Selection For Machine Learning*. Ph.D. The University of Waikato.

Hastie, T., Tibshirani, R., Friedman, J., Tibshirani, R. and Friedman, J., 2009. *The Elements Of Statistical Learning,*. 2nd ed. Springer, p.592.

Hilbe, J., 2017. *Logistic Regression Models*. CRC Press, pp.1-2.

Jackson, J., 2002. Data Mining; A Conceptual Overview. *Communications of the Association for Information Systems*, [online] 8(1), pp.267-290. Available at: <: https://aisel.aisnet.org/cais/vol8/iss1/19>.

James, G., Witten, D., Hastie, T. and Tibshirani, R., 2015. *An Introduction To Statistical Learning*. New York: Springer, p.315.

Kriebel, A. and Murray, E., 2018. *#Makeovermonday: Improving How We Visualize And Analyze Data, One Chart At A Time*. John Wiley & Sons, pp.131-132.
Kursa, M.B. and Rudnicki, W.R., 2010. Feature Selection with the Boruta Package. *Journal of Statistical Software*, *36*(11), pp.1-13.

Mitchell, T., 1997. *Machine Learning*. New York: McGraw Hill.

Mitchell, T.M., 2006. *The discipline of machine learning* (Vol. 9). Pittsburgh: Carnegie Mellon University, School of Computer Science, Machine Learning Department.

Mohri, M., Rostamizadeh, A. and Talwalkar, A., 2012. *Foundations Of Machine Learning*. Cambridge: MIT Press, pp.1-4.

Murphy, K., 2012. *Machine Learning : A Probabilistic Perspective*. Cambridge: MIT PRESS, pp.1-2, 9-10.

Rokach, L. and Maimon, O., 2014. *Data Mining With Decision Trees*. 2nd ed. Singapore: World Scientific, pp.10-16.

Roman, V., 2019. *Unsupervised Machine Learning: Clustering Analysis*. [online] Medium. Available at: <https://towardsdatascience.com/unsupervised-machine-learning-clustering-analysis-d40f2b34ae7e>.

Shearer, C., 2000. The CRISP-DM Model: The New Blueprint for Data Mining. *Journal of Data Warehousing*, [online] 5(4), pp.13-22. Available at: <https://mineracaodedados.files.wordpress.com/2012/04/the-crisp-dm-model-the-new-blueprint-for-data-mining-shearer-colin.pdf>.

Sutton, R. and Barto, A., 2018. *Reinforcement Learning*. 2nd ed. Cambridge: MIT Press, pp.3-7.

Tolles, J. and Meurer, W., 2016. Logistic Regression. *JAMA*, 316(5), p.533.

Vluymans, S., 2018. *Dealing With Imbalanced And Weakly Labelled Data In Machine Learning Using Fuzzy And Rough Set Methods*. New York: Springer, pp.83-84.

Witten, I. and Frank, E., 2017. *Data Mining*. 4th ed. San Francisco, Calif.: Morgan Kaufmann, pp.129-130.

APPENDIX

A-1.SQL Queries to extract jobs and floods

- q_jobs = 'SELECT a.[JobID] ,a.[EntryDateTime] ,LEFT(a.[EntryDateTime],4) + \'-
\' + SUBSTRING(CONVERT(varchar(25), a.[EntryDateTime]), 6, 2) as Year_Month ,a.[Notes] ,a.[CentralAssetID] ,e.[EastingCentral] ,e.[NorthingCentral] ,CAST(CAST(e.[EastingCentral] as INT) as VARCHAR) + \'_\' + CAST(CAST(e.[NorthingCentral] as INT) as VARCHAR) as East_Nort ,e.[LatitudeCentral] ,e.[LongitudeCentral] ,CAST(FORMAT(e.[LatitudeCentral] , \'##.##\') as VARCHAR) + \'_\' + CAST(FORMAT(e.[LongitudeCentral] , \'##.##\') as VARCHAR) as Lat_Long ,a.[SiteID] ,a.[SiteName] ,a.[PlotNo] ,a.[RouteID] ,a.[RouteName] ,a.[DepotID] ,a.[DepotName] ,a.[RoadName] ,a.[ContractAreaID] ,a.[ContractArea] ,a.[AssetTypeID] ,a.[AssetTypeName] ,a.[LogNo] ,a.[JobTypeName] ,a.[JobStatusFlag] ,a.[JobStatusName] ,a.[FollowUpName] ,a.[JobAssignedOfficerID] ,a.[JobCreatedByOfficerID] ,a.[ContractID] ,a.[DimJobScheduleID] ,a.[PriorityName] ,a.[RegimeID] ,a.[RegimeName] ,a.[StartDateTime] ,a.[ActualStartingDateTime] ,a.[TargetCompletionDateTime] ,a.[CalcTargetCompletionDateTime] ,a.[CompleteDateTime] ,a.[ActualCompletionDateTime] ,a.[ActualCompletionDate] ,a.[ActualCompletionYearMonth] ,a.[ActualCompletionWeekEnd] ,a.[CommitDateTime] ,a.[CommitDate] ,a.[CommitYearMonth] ,a.[CommitMonth] ,a.[CommitWeekStart] ,a.[CommitWeekEnd] ,a.[CommitDayOfWeek] ,a.[CommitDayType] ,a.[CommitDayNight] ,a.[CompletionStatus] ,a.[CompletionStatus (CalcTarget)] ,a.[DeliveredLessThan2Hours] ,a.[DueDateGroup] ,a.[DueDateGroup (CalcTarget)] ,a.[CustomerID] ,d.[Name] as [Customer Name] ,a.[CostID] ,b.ParameterTypeID ,b.ParameterValueID ,c.Name as [% Full] FROM [CONFIRM_Consumption].[FactJob] as a LEFT JOIN CONFIRM.JobParameterValue as b ON a.JobID = b.JobID LEFT JOIN CONFIRM.ParameterValue as c ON b.ParameterValueID = c.ParameterValueID AND b.ParameterTypeID = c.ParameterTypeID LEFT JOIN CONFIRM.Customer as d ON a.[CustomerID] = d.[CustomerID] LEFT JOIN CONFIRM.Asset as e ON a.[CentralAssetID] = e.[CentralAssetID] WHERE a.[AssetTypeID] = \'DRGU\' AND b.ParameterTypeID = \'%FUL\' AND CAST(EntryDateTime as DATE) <= \'2020-01-
01\' AND b.IsCurrentFlag = 1 AND c.IsCurrentFlag = 1 AND e.IsCurrentFlag = 1 ORDER BY a.[EntryDateTime], a.JobID ASC'

- q_jobs2 = 'SELECT a.[JobID] ,a.[EntryDateTime] ,LEFT(a.[EntryDateTime],4) + \'-
\' + SUBSTRING(CONVERT(varchar(25), a.[EntryDateTime]), 6, 2) as Year_Month ,a.[Notes] ,a.[CentralAssetID] ,e.[EastingCentral] ,e.[NorthingCentral] ,CAST(CAST(e.[EastingCentral] as INT) as VARCHAR) + \'_\' + CAST(CAST(e.[NorthingCentral] as INT) as VARCHAR) as East_Nort ,e.[LatitudeCentral] ,e.[LongitudeCentral] ,CAST(FORMAT(e.[LatitudeCentral] , \'##.##\') as VARCHAR) + \'_\' + CAST(FORMAT(e.[LongitudeCentral] , \'##.##\') as VARCHAR) as Lat_Long ,a.[SiteID] ,a.[SiteName] ,a.[PlotNo] ,a.[RouteID] ,a.[RouteName] ,a.[DepotID] ,a.[DepotName] ,a.[RoadName] ,a.[ContractAreaID] ,a.[ContractArea] ,a.[AssetTypeID] ,a.[AssetTypeName] ,a.[LogNo] ,a.[JobTypeName] ,a.[JobStatusFlag] ,a.[JobStatusName] ,a.[FollowUpName] ,a.[JobAssignedOfficerID] ,a.[JobCreatedByOfficerID] ,a.[ContractID] ,a.[DimJobScheduleID] ,a.[PriorityName] ,a.[RegimeID] ,a.[RegimeName] ,a.[StartDateTime] ,a.[ActualStartingDateTime] ,a.[TargetCompletionDateTime] ,a.[CalcTargetCompletionDateTime] ,a.[CompleteDateTime] ,a.[ActualCompletionDateTime] ,a.[ActualCompletionDate] ,a.[ActualCompletionYearMonth] ,a.[ActualCompletionWeekEnd] ,a.[CommitDateTime] ,a.[CommitDate] ,a.[CommitYearMonth] ,a.[CommitMonth] ,a.[CommitWeekStart] ,a.[CommitWeekEnd] ,a.[CommitDayOfWeek] ,a.[CommitDayType] ,a.[CommitDayNight] ,a.[CompletionStatus] ,a.[CompletionStatus (CalcTarget)] ,a.[DeliveredLessThan2Hours] ,a.[DueDateGroup] ,a.[DueDateGroup (CalcTarget)] ,a.[CustomerID] ,d.[Name] as [Customer Name] ,a.[CostID] ,b.ParameterTypeID ,b.ParameterValueID ,c.Name as [% Full] FROM [CONFIRM_Consumption].[FactJob] as a LEFT JOIN CONFIRM.JobParameterValue as b ON a.JobID = b.JobID LEFT JOIN CONFIRM.ParameterValue as c ON b.ParameterValueID = c.ParameterValueID AND b.ParameterTypeID = c.ParameterTypeID LEFT JOIN CONFIRM.Customer as d ON a.[CustomerID] = d.[CustomerID] LEFT JOIN CONFIRM.Asset as e ON a.[CentralAssetID] = e.[CentralAssetID] WHERE a.[AssetTypeID] = \'DRGU\' AND b.ParameterTypeID = \'%FUL\' AND b.IsCurrentFlag = 1 AND c.IsCurrentFlag = 1 AND e.IsCurrentFlag = 1 ORDER BY a.[EntryDateTime], a.JobID ASC'

- q_jobs_ts = 'SELECT EOMONTH([EntryDateTime]) as As_Of_YearMonth, COUNT(DISTINCT [JobID]) as Num_Jobs FROM (SELECT a.[JobID] ,a.[EntryDateTime] ,a.[Notes] ,a.[CentralAssetID] ,a.[SiteID] ,a.[SiteName] ,a.[PlotNo] ,a.[RouteID] ,a.[RouteName] ,a.[DepotID] ,a.[DepotName] ,a.[RoadName] ,a.[ContractAreaID] ,a.[ContractArea] ,a.[AssetTypeID] ,a.[AssetTypeName] ,a.[LogNo] ,a.[JobTypeName] ,a.[JobStatusFlag] ,a.[JobStatusName] ,a.[FollowUpName] ,a.[JobAssignedOfficerID] ,a.[JobCreatedByOfficerID] ,a.[ContractID] ,a.[DimJobScheduleID] ,a.[Easting] ,a.[Northing] ,a.[Latitude] ,a.[Longitude] ,a.[PriorityName] ,a.[RegimeID] ,a.[RegimeName] ,a.[StartDateTime] ,a.[ActualStartingDateTime] ,a.[TargetCompletionDateTime] ,a.[CalcTargetCompletionDateTime] ,a.[CompleteDateTime] ,a.[ActualCompletionDateTime] ,a.[ActualCompletionDate] ,a.[ActualCompletionYearMonth] ,a.[ActualCompletionWeekEnd] ,a.[CommitDateTime] ,a.[CommitDate] ,a.[CommitYearMonth] ,a.[

CommitMonth] ,a.[CommitWeekStart] ,a.[CommitWeekEnd] ,a.[CommitDayOfWeek] ,a.[CommitDayType] ,a.[CommitDayNight] ,a.[CompletionStatus] ,a.[CompletionStatus (CalcTarget)] ,a.[DeliveredLessThan2Hours] ,a.[DueDateGroup] ,a.[DueDateGroup (CalcTarget)] ,a.[CustomerID] ,d.[Name] as [Customer Name] ,a.[CostID] ,b.ParameterTypeID ,b.ParameterValueID ,c.Name as [% Full] FROM [CONFIRM_Consumption].[FactJob] as a LEFT JOIN CONFIRM.JobParameterValue as b ON a.JobID = b.JobID LEFT JOIN CONFIRM.ParameterValue as c ON b.ParameterValueID = c.ParameterValueID AND b.ParameterTypeID = c.ParameterTypeID LEFT JOIN CONFIRM.Customer as d ON a.[CustomerID] = d.[CustomerID] WHERE a.[AssetTypeID] = \'DRGU\' AND b.ParameterTypeID = \'%FUL\' AND b.IsCurrentFlag = 1 AND c.IsCurrentFlag = 1) as t1 GROUP BY EOMONTH([EntryDateTime]) ORDER BY EOMONTH([EntryDateTime]) ASC'

- q_floods = 'SELECT a.[FloodID] ,a.[Road] ,a.[Area] ,a.[Latitude] ,a.[Longitude] ,a.[CarriagewayDirection] ,a.[CarriagewayType] ,a.[SectionID] ,a.[FloodStatus] ,a.[MaxEffect] ,a.[ReportedDateTime] ,a.[AffectedLength] ,a.[FloodSeverityIndex] ,a.[IsDeletedFlag] ,a.[IsCurrentFlag] ,b.[FloodHotspotID] ,b.[BaselineID] ,b.[BaselineRiskLevel] ,b.[IsFloodRiskAreaID] ,b.[ActionStatus] ,b.[OverallStatus] FROM [DDMS].[FloodIncident] as a LEFT JOIN [DDMS].[FloodHotspot] as b ON a.Road = b.Road AND a.Area = b.Area AND a.CarriagewayDirection = b.Direction'

- q_job_pmt = 'SELECT OrderJobID, SUM(PaymentValue) + SUM(FeeValue) as [Payment_Amount] FROM CONFIRM_Secure.JobPaymentItem WHERE IsCurrentFlag = 1 GROUP BY OrderJobID'

A-2. Decision Tree classifier (with Data Preprocessing and Feature selection)

```
# Classification
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, confusion_matrix

# Subset the required columns into a dataframe
df_class = df_jobs_weather[['Asset_failure_class', 'Elevation(m)', 'Rain_Intensity_Avg_Last_48hrs', 'Rain_Intensity_Avg_Last_96hrs',
'Rain_Intensity_Avg_Last_Week', \
'Rain_Intensity_Avg_Last_2_Weeks', 'Rain_Intensity_Max_Last_48hrs', 'Rain_Intensity_Max_Last_96hrs', \
'Rain_Intensity_Max_Last_Week', 'Rain_Intensity_Max_Last_2_Weeks', 'Wind_Speed_Mean_Last_48hrs', \
'Wind_Speed_Mean_Last_96hrs', 'Wind_Speed_Mean_Last_Week', 'Wind_Speed_Mean_Last_2_Weeks', \
'Wind_Direction_Mean_Last_48hrs', 'Wind_Direction_Mean_Last_96hrs', 'Wind_Direction_Mean_Last_Week', \
 'Wind_Direction_Mean_Last_2_Weeks', 'Wind_Gust_Mean_Last_48hrs', 'Wind_Gust_Mean_Last_96hrs', \
 'Wind_Gust_Mean_Last_Week', 'Wind_Gust_Mean_Last_2_Weeks', 'Wind_Gust_Direction_Mean_Last_48hrs', \
 'Wind_Gust_Direction_Mean_Last_96hrs', 'Wind_Gust_Direction_Mean_Last_Week', 'Wind_Gust_Direction_Mean_Last_2_Weeks' ]]

# Data Transformations - Replace & Impute blank values with resp. medians and drop the duplicates
df_class = df_class[df_class['Asset_failure_class'] != 'NA'].replace('', np.nan).fillna(df_class.median()).drop_duplicates()

# Data Preprocessing
x = df_class.drop('Asset_failure_class', axis=1)    # dep
y = df_class['Asset_failure_class']          # indep
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25, random_state=28)   # Model Preparation

print('################## Classification ##################')

print('----> Value Counts:', '\n', df_class['Asset_failure_class'].value_counts())
print('----> Number of records in Training dataset:', len(x_train), 'and Number of records in Testing dataset:', len(x_test))

# Model training
classifier = DecisionTreeClassifier()
classifier.fit(x_train, y_train)

# Evaluation using Training data
y_pred = classifier.predict(x_train)
print('************************', '\n', 'Training Dataset Evaluation: ' , '\n', '************************')
print('---> Confusion Matrix: ', '\n' , confusion_matrix(y_train, y_pred))
print('---> Classification Report: ', '\n' , classification_report(y_train, y_pred))

# Evaluation using Testing data
y_pred = classifier.predict(x_test)
print('************************', '\n', 'Testing Dataset Evaluation: ' , '\n', '************************')
print('---> Confusion Matrix: ', '\n' , confusion_matrix(y_test, y_pred))
print('---> Classification Report: ', '\n' , classification_report(y_test, y_pred))
```

1. Output for Decision tree classifier –

```
################## Classification ##################
----> Value Counts:
 L1  1131
L0  1112
```

```
L2   993
Name: Asset_failure_class, dtype: int64
----> Number of records in Training dataset: 2427 and Number of records in Testing dataset: 809
************************
 Training Dataset Evaluation:
 ************************
---> Confusion Matrix:
 [[832  0  0]
 [507 341  0]
 [398 181 168]]
---> Classification Report:
       precision  recall f1-score  support

    L0    0.48   1.00   0.65    832
    L1    0.65   0.40   0.50    848
    L2    1.00   0.22   0.37    747

  accuracy             0.55   2427
  macro avg    0.71   0.54   0.50   2427
weighted avg    0.70   0.55   0.51   2427

************************
 Testing Dataset Evaluation:
 ************************
---> Confusion Matrix:
 [[ 45 173 62]
 [196 23 64]
 [178 57 11]]
---> Classification Report:
       precision  recall f1-score  support

    L0    0.11   0.16   0.13    280
    L1    0.09   0.08   0.09    283
    L2    0.08   0.04   0.06    246

  accuracy             0.10    809
  macro avg    0.09   0.10   0.09    809
weighted avg    0.09   0.10   0.09    809
```

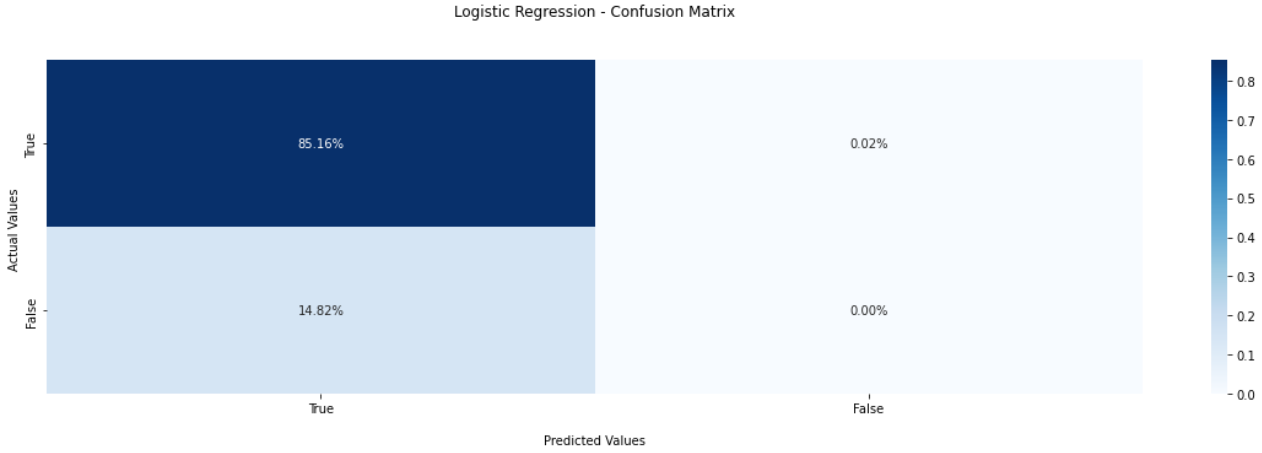
A-3.Logistic Regression (with Data Preprocessing and Feature selection)

```
# Logistic Regression
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix

# Subset the required columns into a dataframe
df_logr = df_jobs_weather[df_jobs_weather['% Full'] != 'Change Me']
df_logr = df_logr[['Asset_failure', 'Elevation(m)', 'Rain_Intensity_Avg_Last_48hrs', 'Rain_Intensity_Avg_Last_96hrs', 'Rain_Intensity_Avg_Last_Week',
\
'Rain_Intensity_Avg_Last_2_Weeks', 'Rain_Intensity_Max_Last_48hrs', 'Rain_Intensity_Max_Last_96hrs', \
'Rain_Intensity_Max_Last_Week', 'Rain_Intensity_Max_Last_2_Weeks', 'Wind_Speed_Mean_Last_48hrs', \
'Wind_Speed_Mean_Last_96hrs', 'Wind_Speed_Mean_Last_Week', 'Wind_Speed_Mean_Last_2_Weeks', \
'Wind_Direction_Mean_Last_48hrs', 'Wind_Direction_Mean_Last_96hrs', 'Wind_Direction_Mean_Last_Week', \
 'Wind_Direction_Mean_Last_2_Weeks', 'Wind_Gust_Mean_Last_48hrs', 'Wind_Gust_Mean_Last_96hrs', \
 'Wind_Gust_Mean_Last_Week', 'Wind_Gust_Mean_Last_2_Weeks', 'Wind_Gust_Direction_Mean_Last_48hrs', \
 'Wind_Gust_Direction_Mean_Last_96hrs', 'Wind_Gust_Direction_Mean_Last_Week', 'Wind_Gust_Direction_Mean_Last_2_Weeks' ]]

# Data Transformations - Replace & Impute blank values with resp. medians and drop the duplicates
df_logr = df_logr.replace('', np.nan).fillna(df_logr.median())

# Data Preprocessing
x = df_logr.drop('Asset_failure', axis=1)    # dep
y = df_logr['Asset_failure']          # indep
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25, random_state=28)   # Model Preparation

print('################## Logistic Regression ##################')

print('----> Value Counts:', '\n', df_logr['Asset_failure'].value_counts())
print('----> Number of records in Training dataset:', len(x_train), 'and Number of records in Testing dataset:', len(x_test))

# Model training
logisticRegr = LogisticRegression()
logisticRegr.fit(x_train, y_train)

# Determine Accuracy of Training data
score = logisticRegr.score(x_train, y_train)
print('----> Model Accuracy (Training data) :', round(score, 4)*100, '%' )
```

```
# Determine Accuracy of Testing data
score = logisticRegr.score(x_test, y_test)
print('----> Model Accuracy (Testing data) :', round(score, 4)*100, '%' )

# Predictive Analysis (for testing data)
y_pred = logisticRegr.predict(x_test)
score = logisticRegr.score(x_test, y_pred)
print('----> Model Prediction :', round(score, 4)*100, '%' )

# Confusion matrix
confusion_matrix = confusion_matrix(y_test, y_pred)
print('----> Confusion Matrix : ', '\n', confusion_matrix)

# Plot the confusion matrix
ax = sn.heatmap(confusion_matrix/np.sum(confusion_matrix), annot=True, fmt='.2%', cmap='Blues')
ax.set_title('Logistic Regression - Confusion Matrix\n\n');
ax.set_xlabel('\nPredicted Values')
ax.set_ylabel('Actual Values ')
plt.figure(1, figsize=(2, 2))
ax.xaxis.set_ticklabels(['True','False'])
ax.yaxis.set_ticklabels(['True','False'])
plt.show()

# Uncategorized gullies
# Subset the required columns into a dataframe
df_logr_u = df_jobs_weather[(df_jobs_weather['% Full'] == 'Change Me') | (df_jobs_weather['% Full'] == 'Not Applicable')]
df_logr_u = df_logr_u[['Elevation(m)', 'Rain_Intensity_Avg_Last_48hrs', 'Rain_Intensity_Avg_Last_96hrs', 'Rain_Intensity_Avg_Last_Week', \
'Rain_Intensity_Avg_Last_2_Weeks', 'Rain_Intensity_Max_Last_48hrs', 'Rain_Intensity_Max_Last_96hrs', \
'Rain_Intensity_Max_Last_Week', 'Rain_Intensity_Max_Last_2_Weeks', 'Wind_Speed_Mean_Last_48hrs', \
'Wind_Speed_Mean_Last_96hrs', 'Wind_Speed_Mean_Last_Week', 'Wind_Speed_Mean_Last_2_Weeks', \
'Wind_Direction_Mean_Last_48hrs', 'Wind_Direction_Mean_Last_96hrs', 'Wind_Direction_Mean_Last_Week', \
 'Wind_Direction_Mean_Last_2_Weeks', 'Wind_Gust_Mean_Last_48hrs', 'Wind_Gust_Mean_Last_96hrs', \
 'Wind_Gust_Mean_Last_Week', 'Wind_Gust_Mean_Last_2_Weeks', 'Wind_Gust_Direction_Mean_Last_48hrs', \
 'Wind_Gust_Direction_Mean_Last_96hrs', 'Wind_Gust_Direction_Mean_Last_Week', 'Wind_Gust_Direction_Mean_Last_2_Weeks' ]]

# Data Transformations - Replace & Impute blank values with resp. medians and drop the duplicates
df_logr_u = df_logr_u.replace('', np.nan).fillna(df_logr_u.median())

# Predictive Analysis (for uncategorized data)
y_pred = logisticRegr.predict(df_logr_u)
score = logisticRegr.score(df_logr_u, y_pred)
print('----> Model Prediction for Uncategorized data:', round(score, 4)*100, '%' )
```

1. Prediction for Categorized data

```
################## Logistic Regression ##################
----> Value Counts:
0  76886
1  13436
Name: Asset_failure, dtype: int64
----> Number of records in Training dataset: 67741 and Number of records in Testing dataset: 22581
----> Model Accuracy (Training data) : 85.09 %
----> Model Accuracy (Testing data) : 85.16 %
----> Model Prediction : 100.0 %
----> Confusion Matrix :
 [[19230   5]
 [ 3346   0]]
```



2. Prediction for Uncategorized data

----> Model Prediction for Uncategorized data: 100.0 %

A-4.Anomaly Detection (using IsolationForest)

```
from sklearn.ensemble import IsolationForest

# Subset the requried data for anomaly detection
df_anom_det = df_jobs_floods_f[['FloodID', 'ReportedDateTime', 'FloodSeverityIndex', 'Asset_failure']]

# Necessary Transformations
df_anom_det['ReportedDateTime'] = pd.to_datetime(df_anom_det['ReportedDateTime'])
df_anom_det['Year'] = df_anom_det["ReportedDateTime"].dt.year
df_anom_det = df_anom_det.set_index('ReportedDateTime')

# Filter the data from before the Year 2013
df_anom_det = df_anom_det[df_anom_det['Year'] >= 2013 ]
del df_anom_det['Year']

# Create the model and fit the data
model = IsolationForest()
model.fit(df_anom_det)
predictions = model.predict(df_anom_det)
df_anom_det['predictions'] = predictions

# Renaming for identification
df_anom_det['Predicted'] = np.where(df_anom_det['predictions'] == -1, 'Outliers', 'Inliers')
df_anom_det['Asset_failure_type'] = np.where(df_anom_det['Asset_failure'] == 0, 'Not failed', 'Failed')
df_anom_det['Asset_failure'] = np.where(df_anom_det['Asset_failure'] == 0, 0.1, 0.9)

print('################## Anomaly Detection ##################')

# How many Asset failures / Gullies fully clogged ?
print('-------> Asset Failures:' , '\n' , df_anom_det.Asset_failure_type.value_counts())

# How many outliers found? 1 - Inliers and -1 - Outliers
print('-------> Outliers vs. Inliers:' , '\n' , df_anom_det['predictions'].value_counts())

# #summary statistics
df_anom_det['FloodSeverityIndex'].describe()

# Scatter Plot
fig = px.scatter(df_anom_det,
        x=df_anom_det.index,
        y="FloodSeverityIndex",
        color="Predicted",
        size="Asset_failure",
        opacity = 0.3,
        color_discrete_sequence=[ "#7f7f7f", "#cc3300"],
        title=("Anomalies detected within floods against gully failures"))

fig.show()
```

1. Output for Anomaly Detection –

```
################## Anomaly Detection ##################
-------> Asset Failures:
 Not failed  721
Failed    122
Name: Asset_failure_type, dtype: int64
-------> Outliers vs. Inliers:
 1  565
-1  278
Name: predictions, dtype: int64
```