Decoding Hidden Patterns: A Comprehensive Data Mining and Visualisation Software Solution

By Joshua de Bruxelles

Student ID: F015011

Department of Computer Science

Loughborough University

COC251: Computer Science Project

Supervisor: Dr. Lars Nagel

Abstract

This dissertation project focuses on the development of a data analysis tool suite that utilises a variety of data mining algorithms for complex data analysis. The software solution is built using the Python programming language and its relevant libraries, with a front-end user interface developed using React. The aim of the project is to provide a tool suite that can facilitate efficient data mining and visualisation for businesses and researchers respectively, regardless of their technical literacy. The project was successful in its aim to provide an effective tool suite for data analysis and visualisation, with the included data mining algorithms being optimised to achieve high levels of accuracy in a consistent manner. The project also facilitated personal development in various areas of computer science, including front-end development, server development, machine learning, and software engineering.

Acknowledgements

I would like to express my sincere gratitude to Dr. Lars Nagel for his invaluable guidance and supervision throughout the course of this project. I would also like to express my appreciation to my family and friends for their unwavering support and encouragement throughout my university education.

Table of Contents

Table of Figures

Table of Tables

1 Introduction

1.1 Problem Statement

The abundance of data in today's world has made it increasingly difficult to extract valuable insights and make informed decisions [1]. The goal of this project is to use data mining techniques to extract useful information from a given dataset, and then use visualisation methods to effectively communicate the insights gained. This project will implement a tool suite to support the application of a selection of independent data mining techniques in a varied landscape of datasets in order to uncover patterns and relationships that would be difficult to discern through manual analysis. The project will use a variety of data mining techniques such as regression, classification, clustering, and association rule mining to extract meaningful insights.

The project will then use visualisation methods to effectively communicate the insights gained and make them easily interpretable to users of the system; the results obtained from each technique will be evaluated independently, comparing the performance with each other to select the optimal technique for a given task. The project will aim to provide a solution that enables more accurate and efficient decision making for the stakeholders involved; these will likely be individuals within a commercial environment, but may also be those using the tool for a more personal exploratory means due to the limited experience required for its operation.

The goal of this project is to package a selection of independent data mining techniques into a single software solution. A flexible and adaptable software solution will be developed that can be applied to a wide range of datasets chosen and uploaded by the users of the system. The solution will be designed with a user-friendly interface, making it accessible to users with minimal technical background, and will be easy to use. The objective of the solution is to bridge the gap between technical and non-technical users and provide organisations or individuals alike with the necessary tools to effectively analyse and extract insights from their data.

1.2 Project Objectives

1. To develop a software solution that incorporates a comprehensive suite of independent data mining techniques, including regression, classification, clustering, and association rule mining, for the purpose of uncovering previously unobserved patterns and relationships within large and complex datasets.
2. To design and implement a user interface that is intuitive and accessible to individuals with limited technical experience, while also providing advanced functionality for more experienced users, in order to facilitate widespread utilisation of the software solution.
3. To package the data mining techniques into a single, flexible, and adaptable software solution that can be applied to a wide range of datasets.
4. To utilise visualisation techniques, tailored to specifically where required to the given analysis methods, to present the results in a clear and interpretable manner, and to facilitate the communication of the insights gained from the analysis.
5. To provide users of the application with the necessary tools to make data-driven decisions that are more accurate and efficient, as a result of the insights gained from the analysis.
6. To rigorously evaluate the performance of the software solution across a diverse set of datasets, in order to establish its robustness and generalisability.
7. To comprehensively document the methods used, the results obtained, and the lessons learned, in order to serve as a reference for future work and to facilitate replication and validation of the results.
8. To identify potential avenues for future research and development, in order to continue to improve the software solution and expand its capabilities.

2 Literature Review

2.1 Data Analytics

Data analytics is the process of collecting, organising, and analysing large sets of data in order to extract meaningful insights [2]. By leveraging the power of computer systems to analyse large datasets, we are able to uncover patterns, trends, and relationships that may not be immediately apparent, using this information to make more informed and confident decisions [3]. In short, data analytics allows us to turn raw data into actionable insights, enabling us to better understand and navigate the ever-shifting, and exponentially evolving, landscape of data around us. In essence, data analytics transforms raw data into useful information. Once this information is understood, it becomes knowledge that can be used to guide actions and inform decision-making [4].

The process of data analysis can be explored and achieved in many diverse ways due to its interdisciplinary nature; each implementation can be approached based on the requirements of the system's users, consequently utilising a chosen multiplicity of analysis techniques best suited to the given system. It is therefore important to consider a range of analysis methodologies to understand the rationale for their usage and the ideal scenarios in which they should be employed.

2.2 Data Pre-processing

The fundamental importance of data pre-processing in data analysis cannot be overstated, as it is crucial for ensuring the validity and reliability of the data being analysed. Without proper pre-processing, the data may be fraught with errors and irregularities, rendering any subsequent analysis inaccurate and potentially inoperable [5]. Suitable pre-processing is consequently essential for ensuring that the data is accurate, consistent, and ready for further analysis, allowing analysts to draw more reliable insights. Data pre-processing can be decomposed into several distinct phases, each designed to further prepare the data prior to analysis [6].

Data Cleaning

Data cleaning is the systematic process of identifying and rectifying inaccuracies, inconsistencies and missing values within a given dataset [7]. While the specific methods employed during the data cleaning process may vary depending on the nature and complexity of the dataset, there are a few common steps that are generally followed in order to ensure the data is as clean and accurate as possible.

Firstly, we can begin by fixing any inherent structural errors within our dataset. This will involve performing de-duplication, as well as identifying any erroneous values. Erroneous values in this case refer to data in the wrong format (e.g., a string where an integer is expected) that has been miss input in some capacity. This stage is usually a simple process that can typically be automated either programmatically, or within data management software such as Excel [8]. Additionally, we must ensure the data we are analysing is particularly relevant to our use case. In essence, we must define the intentions of our analysis and make sure the data being employed will help us achieve this. As a result, data selection can be employed to omit data if it is not deemed relevant to the desired objectives of the analysis [9].

To continue, the next stage required to clean a given dataset involves dealing with any missing values. Once identified, a decision can be made as to how to deal with the missing values. A common approach may be to simply delete the complete case where missing values exist, known as *litewise deletion*. However, this causes a direct loss of information which may unintendingly alter the outcome of the analysis by effecting the standard deviation or distribution of the dataset. Alternatively, the missing values can be estimated; this estimation may be performed through *mean substitution*, where the mean of the dataset is used in place of the missing value, or though the *regression imputation* where a regression model is constructed with the other variables and used to create an estimation. This can provide an accurate prediction that fits within the dataset but has the inherent disadvantage of making the dataset less organic [7].

To continue, we must then analyse the dataset in order to deal with unwanted outliers. It is important to note that outliers should be dealt with heuristically and not removed by default

as in some cases they may provide value to the data being analysed [2]. Outlier detection can be accomplished through the use of the mean and standard deviation of the feature distribution; by defining a threshold k standard deviations away from the mean [2], we therefore define an outlier as any value that exceed this threshold:

$$threshold = \bar{x} \pm k\sigma$$

where,

- $\boldsymbol{threshold}$ is the range within which values will be classified as outliers,
- $\boldsymbol{\bar{x}}$ is the mean of the feature distribution,
- $\boldsymbol{k}$ is a positive constant representing the threshold size,
- $\boldsymbol{\sigma}$ is the standard deviation of the feature distribution.

As mentioned previously, only remove the identified outliers if they are irrelevant to the results of the analysis.

Finally, datasets may require formatting to ensure they are compatible with their analysis tool. The process of *data formatting* involves converting data into a specific, standardised format that makes it easier to compare and analyse. This may include converting dates into a uniform format or converting text data into numerical data. This process is crucial in facilitating effective data analysis, as it allows for the comparison and interpretation of data in a consistent manner.

Data Transformation (Scaling, and Normalisation)

Data transformation involves applying techniques such as scaling, and normalisation to the data to make it more suitable for analysis [10].

Scaling data adjusts the scales of the features to be more comparable, improving the performance and interpretability of the model. If the data ranges for different features are disparate, it can lead to certain features having a disproportionate influence on the analysis. For example, if one feature ranges from 1 to 1000, and another feature ranges from 0 to 1, the first feature will likely have a much greater impact on the data analysis model used. Scaling the data helps to adjust the scales of the features so they are more uniform, and the

model can better utilise all of the available information. Standardising the scales of the features can enhance our ability to comprehend and interpret the data, as it is easier to relate values that are on equivalent scales [11].

Normalisation, on the other hand, is a method used to adjust the actual shape of the dataset as opposed to just altering its range. The objective of normalisation is to change the dataset to fit within a normal distribution, also known as the "bell curve". It is worth noting that the necessity for a dataset to be normally distributed is dependent upon the data analysis model in use presuming that it is. This means that some models may not require normal data in order to produce accurate results, while others may be more sensitive to deviations from normality. Examples of data analysis techniques that assume normalised data include: regression analysis in the form of linear regression, as well as statistical tests: t-test, ANOVA, Chi-squared test, and z-tests [12].

Data Reduction

Data reduction is a process of reducing the amount of data that needs to be analysed in order to extract useful insights, typically by removing redundant or irrelevant data. This can be done by selecting a subset of the data to analyse, aggregating data into larger groups, or by removing unnecessary data. This process is an important consideration for data analysis because it can greatly improve the efficiency of the analysis. With large datasets, it can be computationally expensive and time-consuming to analyse all of the data [13]. By reducing the amount of data that needs to be analysed, the analysis can be performed faster and with fewer resources. Data reduction is also useful because it can help to reduce the complexity of the analysis, making it easier to understand and interpret the results. Thus, the data becomes easier to understand and trends and patterns may be more easily identified.

2.3 Data Visualisation

Data visualisation is an integral part of data analysis, as it allows analysts to effectively communicate and understand complex data sets. This involves creating visual representations of data to allow analysts to quickly identify trends, patterns, and

relationships within the data, and communicate these findings to others in a clear and succinct manner [14]. Predominantly, this can be accomplished through the use of charts, graphs, plots, as well as other types of visual elements. The selection of visual representation will depend on the specific goals of the data analysis, as different visualisations may be better suited for different types of analysis. Accordingly, a range of representations will be evaluated in order to understand their individually appropriate use cases.

Scatter Plots

Scatter plots are used to display the relationship between two quantitative variables [14]. Each observation in the data is represented as a point on the plot, with the x-axis representing one variable and the y-axis representing the other. The position of the point on the plot reflects the values of the variables for that particular observation. Scatter plots can be useful for identifying patterns and trends in the data and can also be used to visualise the relationship between variables. They are commonly used for the visualisation of a line of best fit as generated by multiple and linear regression models [15].

Line Graphs

A *line graph* is a type of chart that uses lines to represent the changes in data over a given period of time. The line graph plots points on the graph for each data point, and then connects these points with a line to show the overall trend or pattern in the data [14]. Line graphs are useful for visualising patterns in data over time and can be used to compare multiple data sets or to show the relationship between two variables.

Bar Graph

Bar graphs are a type of graphical representation that utilise horizontal or vertical bars to compare data between different category items. The size of the bars corresponds to their associated quantitative value. They are commonly utilised to compare the values of various groups within a given dataset [14]. Therefore, they can be considered as a useful means of visualising data mining performance metrics due to emphasizing the relative sizes of each category's bar, instantly presenting a clear comparison for the reader [16].

Histograms

A *histogram* is a graph that shows the distribution of quantitative data. It is similar to a bar chart, but instead of comparing categories, it uses bins to show the frequency of data within certain ranges. The size or area of the bars in the histogram represents the number of measurements within each bin [14]. Histograms are useful for understanding the distribution of data and identifying patterns or trends.

Heatmaps

Heatmaps are graphical representations that use colours to depict the relative magnitude of data points within a two-dimensional matrix. The cells within the matrix correspond to individual data points, and the colours used to represent these points are usually a gradient, with warmer colours indicating higher values and cooler colours indicating lower values [14]. Within the context of data mining, they can be used as a means of analysing classifier performance through the construction of a confusion matrix [17].

2.4 Data Mining

Data mining is the process of extracting useful information from large datasets through the application of advanced algorithms and statistical techniques. It involves the creation of descriptive and predictive models that can be used to aid decision making by discovering patterns and relationships within a given dataset [4]. There are two main categorisations that can be used to group data mining techniques: supervised learning and unsupervised learning. Supervised learning refers to the use of labelled data (i.e., data that has been associated with a designated attribute) to train a given analytical model. Correspondingly, unsupervised learning refers to the use of unlabelled data (i.e., data not associated with supplementary attributes) to extract information from the available data [18].

Supervised Learning

Classification and *Regression* are the two main methods that will be considered to further understand and apply the principles of supervised learning.

Classification

Classification is a supervised learning method that involves the predictive assignment of a class or category to a given sample or data point that is being analysed based on its attributes [18]. This can be accomplished using a variety of classification models, of which a selection will be considered to further understand their application.

- *K-Nearest Neighbours (KNN):*
  Nearest neighbour matching relies on matching an unclassified data sample with its nearest k neighbours, where k is a positive constant, and using their existing classification labels to predict the class of unlabelled data [18].
- *Logistic Regression:*
  Logistic regression makes use of correlated independent variables (which must be carefully selected) within a dataset to predict the probability of a given classification. This predicted probability can then be transformed to a binary classification [19].
- *Decision Trees and Random Forests:*
  Decision trees are a made up of classifications in a tree-like structure, traversing from the root node to a given leaf node when classifying new objects. The path followed is dependent on comparisons made between the data sample being analysed and association rules within the tree [4]. By generating a large number of decision trees in a random fashion, evaluating their performance, and selecting the most effective trees to be integrated, we can create an ensemble model that is better performing than a singular, rudimentary decision tree [18].

Regression

Regression models are a major class of data mining and supervised learning, involving the prediction of a continuous outcome, such as a numerical value or probability, based on a series of input variables. Primarily, the goal of regression is to model the relationship between a number of independent variables and a given dependent variable [4]. Regression

can be classified as either linear or non-linear. Typically, a linear regression model is used to represent the relationship between a set of input variables and a given output variable as a linear equation. This means the variation of the input variables has a direct impact on the output variable, i.e., the training data and our prediction.

We can begin by exploring the case where we have a single independent variable to perform linear regression. Thrane introduces the example of predicting house sale prices, beginning with a prediction made exclusively on their square meterage; our aim in this case is to use the single house size variable and model its effect on the house sale price. We can accomplish this by finding a line of best fit between the two in the form:

$$Y = \beta_0 + \beta_1 X$$

Where,

- $\boldsymbol{Y}$ is the dependent variable (the one we predict),
- $\boldsymbol{\beta_0}$ is the intercept of the line,
- $\boldsymbol{\beta_1}$ is the gradient of the line (the coefficient we adjust),
- $\boldsymbol{X}$ is our independent variable (the one we use to predict). [4]

This model allows us, in a rudimentary way, to represent and make predictions from an estimated equation for our linear regression [20]. We adjust the coefficient of the independent variables in order to fit the equation to make predictions with the least number of errors. In terms of predicting house sales prices, we can consequently make a prediction using only the line intercept and gradient. Proceeding this, we can form a multiple regression analysis; this is where several independent variables are used to model the relationship. Intuitively, the use of more than one input variable requires increased consideration for accuracy and an adept understanding of the variables involved. The equation for a multiple variable linear regression model is:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n$$

Where (similarly),

- $\boldsymbol{Y}$ is the dependent variable (the one we predict),
- $\boldsymbol{\beta_0}$ is the intercept of the line,

- $\boldsymbol{\beta_1, \beta_2, \ldots, \beta_n}$ is the gradient of the line (the coefficients we adjust),
- $\boldsymbol{X_1, X_2, \ldots, X_n}$ are our independent variables (the one we use to predict). [4]

Again, relating back to house price predictions, we can use the number of bedrooms, number of bathrooms, condition of the house, age of the house in combination with the size of the house as our independent variables to form a multiple regression model [20]. If we are unable to model the relationship with a linear equation, we can then make use of a non-linear regression analysis techniques which can be more flexible but more problematic to interpret [20].

Unsupervised Learning

Clustering and *Association Rule Mining* are the two main methods that will be considered to further understand and apply the principles of unsupervised learning.

Clustering

In general, clustering seeks to analyse unlabelled data, forming groups (clusters) based on the data points that are considered to be similar [18]. For this reason, clustering can be used to identify important patterns and trends within a dataset that may otherwise not present themselves. This is especially useful when analysing unlabelled data as we can reduce complexity through categorisation, providing us with a means of revealing common characteristics among a given clusters contained data points. For instance, clustering has been used to identify characteristics within the behaviour Bitcoin pricing of which little information is known, and any information than can be extracted could prove incredibly profitable [21]. K-means clustering will be considered as a means of implementing clustering.

k-means clustering works by defining a set of k clusters, where k is a positive constant, and then assigning each considered data point to the cluster with the closest mean to it. The mean of each cluster is then recalculated iteratively as data points are added until the clusters no longer change, or the maximum number of iterations is reached [18]. When implemented, this algorithm will always terminate once the clusters have been formed, however the clusters produced may not best represent the underlying structure of our data.

Therefore, we run the clustering algorithm multiple times so as to reduce our cost function to the lowest possible value, in turn leaving us with the highest possible groupings for our analysis [18].

Association Rule Mining

Association Rule Mining is a method of discovering relationships between variables in a dataset. This technique involves the identification of frequently occurring co-occurrences of items within a dataset, called association rules, which can reveal relationships between different elements. To find association rules, data miners employ various algorithms that search the dataset for recurrent patterns, and then use these patterns to generate rules. These rules can then be used to make predictions about future purchases, or to identify relationships that might not have been obvious before [18].

Dimensionality Reduction

Dimensionality reduction is a method of reducing the number of variables in a dataset while preserving as much of the information as possible. This can be useful for visualising and analysing large datasets, as it can help to reduce the complexity of the data and make it easier to understand [22]. The two main approaches for dimensionality reduction considered in this case are as follows:

- *Principal component analysis (PCA):* PCA is a technique that projects the data onto a lower-dimensional space, performed by finding an optimal position for the best information variance and vector dimensional features reduction. This is often used to reduce the features of large data sets into smaller features that contain the most information, reducing computational expenses when performing analysis [23].
- *Multidimensional scaling (MDS):* MDS is a technique that projects the data onto a lower-dimensional space, performed by preserving the distances between points in the data. In addition to dimensionality reduction, MDS is commonly used to provide a graphical representation of the dissimilarities between data points [24].

3 Market Leader Analysis

Within the data intelligence commerce, there are a range of software solutions that provide their users with a means of completing adept data analysis through an efficient and automated methods. In order to understand the optimum collection of attributes required to construct a proficient data analysis and visualisation tool, it can be advantageous to review each of the market leaders in this space; by understanding the functionality of their tools that works best, and the features of their products that create friction, it will become more apparent as to the best set of requirements to define for a successful project.

3.1 Microsoft Power BI:

Microsoft Power BI is a collection of software tools concentrated on providing business intelligence and analytics [25]; the platform can be used for data visualisation, analysis, and evaluation through the creation of dashboards, reports, and entire bespoke applications specific to the analytical requirements of a given business. Power BI is an incredibly proficient tool for extracting valuable insights from a selection of data sources, combining statistical analysis, visualisations, and Artificial Intelligence to provide organisations with meaningful results and therefore a basis upon which to make confident decisions when faced with ambiguity. Power BI synchronises with different Microsoft applications, making the transfer and sharing of data simple.

Power BI is able to apply a range of statistical analysis methods to users data. Ranging from basic statistical analysis techniques, to utilising pre-trained machine learning models to provide users with artificially intelligent insights, Power BI is a highly adept software package to provide users with intelligible visualisations and examination of their data [25]. The complexity of the analysis can be enhanced using custom Data Analysis Expressions (DAX) where users can employ a collection of functions, operators, and constants to create a formula or expression to manipulate their data [26]. Moreover, Power BI is capable of implementing custom scripts using Python and R, which are both extensive programming languages with endless prospect to facilitate the advanced analysis of data beyond the predefined functionality present in Power BI – this makes Power BI one of the most

powerful data analysis software packages currently available as the opportunities are seemingly endless.

3.2 Tableau

Tableau is a data analytics and visualisation tool, aimed at giving businesses an accessible means to analyse and extract more information from their data [27]; the fact there is no requirement for prior programming knowledge, paired with the natural drag and drop functionality means the software can be readily invited into all businesses with a minimal learning curve, thus giving users access to clear visualisations and intuitive analysis techniques faster. The focus on exploratory data analysis through the creation of dashboards, and comprehensive visualisations, in addition to accepting data from wide variety of sources such as SQL databases, spreadsheets, Google Analytics and naturally Excel.

3.3 Qlik Sense

Qlik Sense allows users to execute advanced analytical techniques on their data, producing valuable and clear visualisations, predictions and calculations [28]; through the exploration of this information, integrated with machine learning models to produce artificially intelligent insights, Qlik Sense can be seen as a high-performance data analytics tool for all types of users – although it appears their primary demographic are commercial clients looking to apply a data-driven methodology when faced with critical inflection points, as well as day to day analytical needs. Users are also able to create and share interactive dashboards, enable automated analytical processes and data alerting to provide real-time insights, as well as being able to take advantage of the cloud-based capabilities of Qlik Sense to provide security and adaptability to data analytics.

3.4 Google Looker

Looker is a cloud-based data analysis platform acquired by Google in response to competing data analysis tools offered by other leading organisations such as Microsoft and Amazon. Included within the Looker software is “LookML” - a simple, accessible data modelling language based on SQL - that can be used to define reusable SQL queries, perform

calculations, and outline data relationships. Moreover, Looker has the capability to develop customized applications and dashboards, as well as to integrate analytics into external applications, automate report creation, and seamlessly integrate with Google Cloud, thereby offering businesses a robust infrastructure for data management and security [29].

3.5 Plotly Dash

Dash is an open-source python framework created by Plotly as a means of developing adept web user interfaces for data analysis dashboards without a prerequisite for understanding of web technologies such as JavaScript, HTML and CSS [30]. Dash allows users build an interface to display the visualisations provided by Plotly, allowing users to quickly bind customised Python, Julia, R and F# code to a full-stack web application; this application can then be shared by its URL.

A Dash application is comprised of two core components. The first component is the "layout" of the application, defining its aesthetic attributes, interface design and overall usability of the application. Proceeding this, in order to make the application interactive, developers must implement the second component, namely the "callbacks". Callbacks are used within Dash applications to provide some functionality through taking an input, triggering the callback function, and defining a given returned output that is used to update the application – i.e., updating a given component's data properties.

3.6 Evaluation

Upon evaluation, it appears that the majority of leading data analysis software tools each employ a comparable set of functionality to provide their users with an optimal experience. The primary differentiating factor between each of the considered systems appears to be the usability and technical prerequisites needed to make full use of the system; some of the software tools clearly have a target demographic of individuals with a more technical background with proficiency in programming languages as opposed to some of the solutions which do not require any prior technical knowledge, thus drastically reducing the system's learning curve. For example, Tableau provides users with drag and drop functionality which provides more accessibility to their application. Furthermore, the supported languages users

can employ to construct more advanced data analysis queries can provide users with knowledge of programming languages the ability to create advanced analytics tailored to their requirements.

4 Approach

4.1 Progress Reflection

Although it has not been without its challenges, as I embark on the implementation stage of my project, I am pleased with the progress that has been made thus far. It's important to note that the implementation at this stage is primarily exploratory prototyping, which means that it's not fully developed, but it's a good representation of the algorithms that will be used within the final product that can be tested, evaluated, and refined to improve its performance.

Despite encountering challenges in data acquisition and validation, I have successfully sourced and utilised cleaned datasets, such as the Iris dataset, for the tool's implementation. The initial prototype has been developed as individual rudimentary Python implementations for a select few data mining algorithms in order to understand their use further; each individual implementation performs a specific task, such as clustering, classification, or association rule mining, which when combined, will provide a comprehensive data analysis tool. Additionally, I have tried aligning various visualisation techniques with a respective analysis method, which serves as a useful representation of where they are most proficient. This prototype, however, does not yet include the complete integration of all data analysis algorithms, and does not yet have a front-end user interface to operate them with. Nonetheless, it still provides the opportunity for further testing, evaluation, and refinement. Consequently, the start of the implementation stage has proven to be a challenging task, with further developments to be made as the project continues proceeding a formal design process.

Upon completing the project's literature review, one major obstacle was the breadth of the topic itself. Data mining and visualisation encompasses a wide range of sub-topics and techniques, from machine learning algorithms to data visualisation tools. I found it difficult to focus my research and identify the specific areas that were most relevant to my project. An additional challenge encountered in the literature review was the disparate terminology used to describe similar concepts, which inhibited comprehension of the context and relevance of certain publications. This was particularly frustrating when trying to compare

and contrast different studies, as I had to spend extra time decoding the language used. Despite these challenges, I was able to gather a comprehensive collection of literature on data mining and visualisation that was relevant to my project. The process of reviewing the literature helped me to gain a deeper understanding of the research in this field and to identify gaps in the literature that my project could address. In particular, providing a user-friendly means of accessing and operating data mining algorithms with an abstracted view of their implementation.

4.2 Revised Project Plan (Gantt Chart)

The project plan, illustrated as a Gantt chart (Figure 1), is an essential tool for monitoring the progress of the project [31]. As the project progresses, the plan will be amended to reflect any fluctuations in progress, specifically task overruns, which allow for early identification of delays and adjustments to the schedule. This is essential for ensuring that the project stays on track and any potential risks can be addressed in a timely manner.

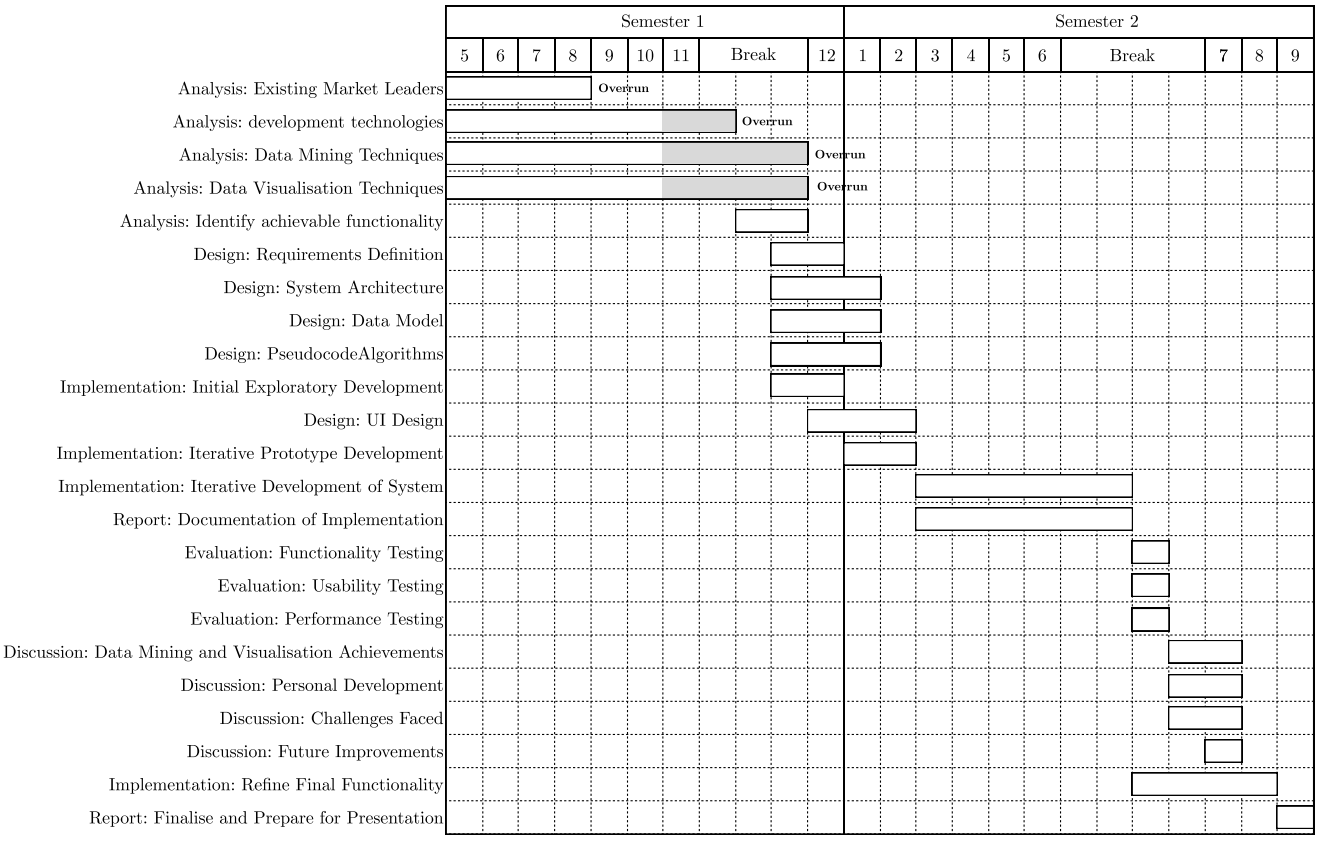


Figure 1 - Project Plan Gantt Chart

Explanation and next steps

As illustrated by the Gantt Chart (Figure 1), the next stage will involve the design phase and iterative development of the software solution. This phase will involve taking the findings from the literature review, market leader analysis and technology stack research and turning them into a detailed design for the tool. During this phase, I will be focusing on ensuring that the design meets the given objectives of the project, specifying sufficiently detailed requirements to ensure an in depth understanding of the full system prior to development.

Once the design is complete, I will begin iterative development. This means that the tool will be developed in stages, with each stage building upon the previous one – i.e., each given data analysis solution will be implemented independently, with incremental changes. Each analysis technique will be tested and evaluated, and any identified issues will be addressed before moving on to the next. This approach will allow me to make adjustments and improvements as needed, ensuring that the final product is robust and of a high standard.

Throughout the design and iterative development phase, I will also be working on designing and later creating the tool's user interface. I will also be creating and executing test cases to ensure that the tool is working correctly, and that it is free of any bugs. As the project moves forward, I will be keeping track of my progress and any notes that might help me in the future.

4.3 System Requirements

The following list of system requirements will be expressed using the MoSCoW prioritisation technique to categorise them based on their level of importance and urgency to the project [32].

1. **Data Input:** The tool should support various types of data inputs, such as CSV, Excel, JSON, and SQL. The ability to import data from external sources, such as APIs, and to connect to databases is required. Users should be able to upload files for their desired dataset, where each column represents a feature or variable, and each row represents a data point. *(MUST)*

2. **Data Cleaning:** The tool should have built-in data cleaning capabilities, including the ability to handle missing values, outliers, and duplicate data. *(COULD)*
3. **Error Management:** The tool suite should have an error validation feature that ensures the data inputted is valid and appropriate for the intended algorithm. The system front-end should notify the user if the data is not compatible with the chosen data mining algorithm. *(MUST)*
4. **Data Mining:** The tool should have a wide range of data mining algorithms that each operate to a high level of accuracy and consistency when presented with a suitable dataset, including but not limited to:
    a. Supervised Learning:
        i. Linear Regression Predictor *(MUST)*
        ii. Logistic Regression Classifier *(MUST)*
        iii. Decision Tree Classifier *(MUST)*
        iv. Random Forest Classifier *(SHOULD)*
    b. Unsupervised Learning:
        i. K-Means Clustering *(MUST)*
        ii. Principal Component Analysis *(COULD)*
        iii. Multidimensional Scaling *(COULD)*
        iv. Association Rule Mining *(COULD)*
5. **Data Visualisation:** The tool should have a variety of visualisation options, including but not limited to line charts, bar charts, scatter plots, and confusion matrixes. The visualisation should be interactive and customisable to meet the needs of different users. *(MUST)*
6. **Analysis Export:** The tool should allow the user to export the generated visualisations and analysis in a downloadable format. *(SHOULD)*
7. **User-Friendly Interface:** The tool should have a user-friendly interface that is easy to navigate, with clear and concise instructions for each step. *(MUST)*
8. **Flexibility:** The tool should be flexible, allowing users to switch between different data mining algorithms and visualisation options easily. *(MUST)*
9. **Scalability:** The tool should be able to handle large and complex data sets with ease. *(SHOULD)*

10. **Security:** The tool should include robust security features to protect data privacy and confidentiality. *(COULD)*
11. **Technical Support:** The tool should come with adequate technical support, including documentation and training resources, to help users understand and utilise the tool effectively. *(COULD)*

4.4 Project Risks

1. **Data Quality Risk:** The tool's performance and output are heavily reliant on the quality of the data input. Ensuring the data input is of high quality and valid may pose a challenge, particularly when dealing with complex data sets from various sources.
2. **Technical Complexity Risk:** Developing a data analysis tool is inherently technical, with different modules such as data input, data mining, and data visualisation. The technical complexity of the tool may pose a risk to the development process, and it is essential to have extensive research and understanding of the data mining algorithms.
3. **Scalability Risk:** The tool should be able to handle large and complex data sets with ease. The ability to scale the tool as data sets grow may pose a challenge.
4. **Security Risk:** The tool should include security features to protect data privacy and confidentiality. Data breaches may occur if adequate security measures are not implemented, putting the tool's users at risk.
5. **User Experience Risk:** The tool should have a user-friendly interface that is easy to navigate, with clear and concise instructions for each step. The user experience of the tool may pose a risk if it is not well-designed, leading to low adoption rates.
6. **Maintenance Risk:** The tool should come with adequate technical support, including documentation and training resources, to help users understand and utilise the tool effectively. Failing to provide adequate maintenance and support may lead to the tool becoming outdated and obsolete.
7. **Performance Risk:** The tool's performance may be affected by several factors such as server hardware, database management, and user traffic. Ensuring the tool performs well under different scenarios may pose a challenge.
8. **Testing Risk:** Testing the tool's various components such as data input, data mining, and data visualisation, and ensuring their functionality, may pose a challenge.

9. **Resource Constraints Risk:** Developing a data analysis tool requires resources such as time, personnel, and finances. It is crucial to ensure that adequate resources are available throughout the development process to avoid delays and cost overruns.

5 Design

The following section provides an overview of the technology and methods used to develop the project, with justifications for their selection. This section serves as a blueprint for the project's implementation and ensures that the development choices made are appropriate for the project's objectives.

5.1 Architecture

The proposed system will be developed with a Python backend server, as facilitated by Flask, with a React front-end for the user-interface. The architecture of the tool suite will comprise of the following high-level components:

- **Flask server:** This component serves as the backend of the tool suite, providing APIs for the front-end to communicate with the various data mining modules. Flask is a flexible and lightweight Python web framework, enabling communication between the front-end user interface, implemented in React, and the various data mining modules running on the server [33]. The Flask backend receives requests from the React component, processes the requests, and returns the appropriate response to the user interface. These requests typically include input data from the user, which is passed to the appropriate data mining module for processing. The integration of Flask with React enables the creation of a flexible user interface, while taking advantage of the proficient Python data analysis libraries, should allow for efficient data mining capabilities within the tool suite.
- **React front-end:** This component provides the user interface for the tool suite, allowing users to interact with the backend modules and visualises the results. React facilitates communication with the Python backend modules via the Flask server and enables the development of the visualisation capabilities that are essential for displaying the results of our data analysis in a user-friendly manner. The use of React in the tool suite thus enables the creation of a flexible and modular user interface that seamlessly integrates with the Python backend, providing powerful data analysis functionality to users [34]. The large number of libraries (in particular Material UI and Plotly.js) available within React make it an ideal choice for building responsive user interfaces.

- **Python:** Python is a particularly suitable choice for the project due to its extensive ecosystem of existing packages, which provide efficient data manipulation capabilities in order to competently implement the given data mining algorithms that are being explored [35]. The use of Python within the tool suite enables users to easily input, transform, and aggregate large datasets, applying our predictive models in order to extract meaningful predictions based on the given trends in a variety of datasets. Additionally, Python's ability to be easily integrated with other technologies enables the tool suite to seamlessly communicate with the front-end React application via the Flask server API, forming a versatile backend language for data analysis tasks. The considered alternatives to Python are as follows:
    - Java – less flexible that Python with less support in terms of industry standard projects.
    - R - while popular for statistical computing and data visualization, it can be more limited in its package dependency range compared to Python.
    - Scala – much steeper learning curve that may result in unfulfilled requirement within the timeframe.

System Architecture Diagram

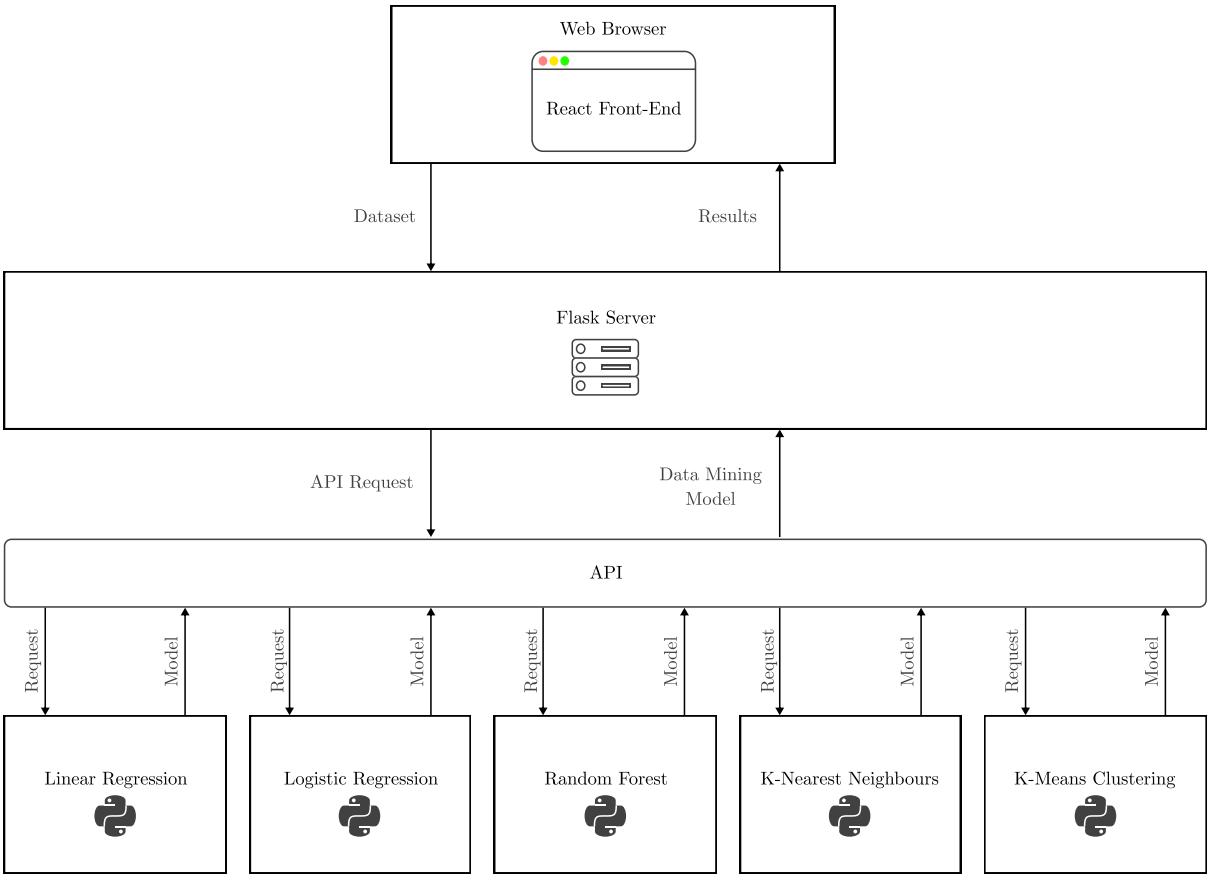


Figure 2 - System Architecture Diagram

5.2 Data

The system itself will operate as a stateless application, which ensures that no user data is stored within the system. As a result, stateless applications provide various advantages, including simplified scaling through the parallelisation of asynchronous requests, improved security, and increased reliability due to a reduced need for main memory on the server [36]. In this case, each user request is processed independently without any awareness of previous requests or user states. This design approach enables the tool suite to handle a large number of concurrent requests while simultaneously mitigating the risks of data loss or corruption. Data is uploaded via a web form, returned to the server, and processed without permanent storage; this ensures that user privacy is protected while also allowing for efficient and effective data mining and visualisation. By not storing data permanently, the system reduces the risk of data breaches and ensures that users are able to regulate the use of their data.

The data that is input to the tool suite will be structured in Comma-Separated Values (CSV) format - a universal file format employed for storing tabular data [37]; this format is a well-established standard that is commonplace for most data analysis tools as a result of its compatibility. Moreover, the input data is subjected to specific validation rules to ensure that it is properly formatted, and any errors are identified prior to processing. Accordingly, data that is entered in an incorrect format for its given data mining algorithm (e.g., unsupervised data for a supervised learning algorithm) will not be able to take advantage of the system's functionality.

In order to support efficient analysis and ensure programmatic compatibility, the data is converted into Pandas 'dataframes'. Pandas is a popular Python library that provides powerful data manipulation and analysis tools, making it an ideal choice for handling data in the tool suite [38]. Once we have our data stored as a pandas dataframe, we are able to easily split our data into training and testing datasets to ensure optimal training of our models. NumPy is a critical library for handling large-scale numerical computations, providing efficient support for multi-dimensional arrays and matrices [38]. In particular, NumPy will be utilised for processing large datasets, allowing for advanced mathematical operations with minimal computational overhead. This results in a reliable and efficient data mining service that can extract valuable insights from complex datasets.

5.3 RESTful APIs:

A RESTful API, or Representational State Transfer API, is a popular architectural style for building web services. It is based on a set of constraints that make it possible for different systems to communicate and exchange data in a standardised way using HTTP requests [39]. RESTful APIs are typically used to provide access to data or functionality in a way that is stateless, platform-independent and language impartial. The use of HTTP requests for communication means that RESTful APIs are simple to use and can be integrated easily into any application that supports HTTP requests. Thus, their flexibility renders them an ideal choice for implementing our data mining tool suite, facilitating future capacity for adaptations and independent testing.

In this case, APIs will be created for the various data analysis algorithms. These APIs will be designed to accept data in specific formats or perform pre-processing steps before running the algorithm. The Flask server can expose different endpoints, each corresponding to a different data mining algorithm that the user can choose to apply to their data. When the user sends a POST request to one of these endpoints, the Flask server can receive the data and use it to run the appropriate algorithm. Once the algorithm has been run, the server can send the results back to the front-end in a standardised format, such as JSON that will then be used to build the visualisations. This approach enables the development of a user-friendly and efficient software solution that can perform complex data analysis tasks in a flexible manner.

5.4 Algorithm Design

In order to provide a more complete understanding prior to their implementation, we will provide a comprehensive overview of the chosen data mining algorithms to be included within our tool suite. We will present pseudocode solutions for the implementation of these algorithms, which can serve as a useful reference to facilitate the translation of these algorithms to any programming language (Python in our case).

Linear Regression Algorithm

The algorithm below is an implementation of linear regression which, as mentioned in Section 2.4, is used to determine a line of best fit between two variables. The input includes two lists of values, one for the independent variable (X) and the other for the dependent variable (Y), along with the total number of observations (n). The algorithm begins by calculating the mean of both the independent and dependent variables. Next, it calculates the variance and covariance of the independent and dependent variables, respectively. The slope of the line is then obtained by dividing the covariance by the variance, and the y-intercept is obtained by subtracting the product of the slope and the mean of the independent variable from the mean of the dependent variable [40]. The resulting slope and y-intercept values can be used to plot the best-fit line, which represents the relationship between the two variables. It is important to note that this algorithm assumes a linear

relationship between the two variables, and does not account for outliers or nonlinear relationships.

Algorithm 1 Simple Linear Regression

Require: X: a list of input values, Y: a list of output values, n: the number of observations

Ensure: *slope*: the slope of the linear regression line, *intercept*: the y-intercept of the linear regression line

Initialize $mean_x$ and $mean_y$ to zero
for $i = 1$ **to** n **do**
  $mean_x \leftarrow mean_x + \frac{X_i}{n}$
  $mean_y \leftarrow mean_y + \frac{Y_i}{n}$
end for
Initialize var_x and cov_xy to zero
for $i = 1$ **to** n **do**
  $var_x \leftarrow var_x + (X_i - mean_x)^2$
  $cov_xy \leftarrow cov_xy + (X_i - mean_x) \times (Y_i - mean_y)$
end for
$slope \leftarrow \frac{cov_xy}{var_x}$
$intercept \leftarrow mean_y - slope \times mean_x$
return *slope*, *intercept*

Figure 3 - Linear Regression Pseudocode

Decision Tree Algorithm

The ID3 algorithm is a classification algorithm used to build a decision tree that can predict the class label of a new instance based on its features. In theory, the implementation of this algorithm should produce the shortest and most efficient tree [41]. The algorithm starts by creating a new node and checking if all labels in y are the same. If they are, the node is initialised as a leaf node with the class label equal to the common label, then the algorithm returns the node. If not, the algorithm recursively splits the data based on the feature that provides the highest information gain, which measures how much a feature reduces the uncertainty about the class labels. This process continues until all stopping criteria are met, such as reaching the maximum depth of the tree or having no more features to split on. At each step, the algorithm chooses the feature that provides the highest information gain and sets the current node's feature to that feature. The data is then split into subsets based on the values of the chosen feature, and a child node is created for each subset. The algorithm recursively calls itself with each child node and continues to build the decision tree until completion.

Algorithm 2 ID3 Decision Tree Algorithm

Require: X: a matrix of input features, y: a vector of output labels, $features$: a list of features, max_depth: the maximum depth of the tree
Ensure: *root*: the root node of the decision tree
  Create a new node *node*
  if all labels in y are the same **then**
    Set *node* as a leaf node with class label equal to the common label
    return *node*
  end if
  if there are no more features to split on or max_depth has been reached **then**
    Set *node* as a leaf node with class label equal to the most common label in y
    return *node*
  end if
  Choose the feature f that provides the highest information gain
  Set *node*'s feature to f
  Split the data into subsets $S_1, S_2, ..., S_k$ based on the values of f
  For each subset S_i, create a child node c_i
  Set *node*'s children to $c_1, c_2, ..., c_k$
  for each child node c_i **do**
    Recursively call the ID3 algorithm with $X = S_i$, $y =$ the labels corresponding to S_i, $features =$ all features except f, $max\_depth - 1$
    Add c_i as a child of *node*
  end for
  return *node*

Figure 4 - ID3 Decision Tree Pseudocode

Random Forest Algorithm

The random forest algorithm is an ensemble method that builds multiple decision trees and combines their predictions to improve the accuracy and stability of the model [42]. Given a matrix of input features (X) and corresponding labels (y), the algorithm bootstraps samples by training a specified number of decision trees that are randomly sampled with replacement from the data. Each tree is built with a limited maximum depth and requires a minimum number of samples to split an internal node. Additionally, the algorithm only considers a subset of features when looking for the best split. The resulting trees are then used to predict the labels for X, and the predictions from all trees are combined by taking the most common label for each input. The random forest algorithm's main advantage is its ability to reduce overfitting by inducing randomness in the sampling and feature selection.

Algorithm 3 Random Forest

Require: X: a matrix of input features, y: a vector of output labels, n_trees: the number of trees to train, max_depth: the maximum depth of each tree, $min\_samples\_split$: the minimum number of samples required to split an internal node, $n\_features$: the number of features to consider when looking for the best split

Ensure: y_pred: a vector of predicted labels for X

```
Initialize an empty list for trees
for i = 1 to n_trees do
    Initialize a new decision tree tree with parameters max_depth,
    min_samples_split, and n_features
    Sample X_sample and y_sample with replacement from X and y to create
    a bootstrap sample
    Fit tree to X_sample and y_sample
    Append tree to trees
end for
Initialize an empty list for tree_preds
for i = 1 to n_trees do
    Predict the labels for X using tree_i
    Append the predicted labels to tree_preds
end for
Transpose tree_preds to get a matrix with predicted labels as columns
Initialize an empty vector y_pred
for each column preds_i in tree_preds do
    Compute the most common label in preds_i and append it to y_pred
end for
return y_pred
```

Figure 5 - Random Forest Pseudocode

Logistic Regression Algorithm

Logistic regression is a supervised learning algorithm used for binary classification tasks. Given a dataset of input features (X) and their corresponding binary classification labels (y), the goal of logistic regression is to train a model that can predict the binary label for new input features based on calculated probability estimates for that class. The algorithm starts by initialising the model parameters to some values (often 0 or small random values), and then iteratively updates these parameters using gradient descent. In each iteration, the algorithm calculates the model's linear predictions on the input features, which are then passed through a sigmoid function to produce a probability estimate for the positive class. After that, the algorithm calculates the gradient of a cost function with respect to the model parameters and updates the parameters by moving them in the opposite direction of the gradient in a process known as gradient descent. The learning rate determines the size of the step taken in the direction of the gradient during each parameter update, and the number of iterations determines the maximum number of times the algorithm will update the parameters before stopping [43]. Once the model parameters have been learned, they can be used to predict the binary label for new input features by computing the linear predictions of the model on the input features, applying a sigmoid function to obtain a probability estimate for the positive class, and thresholding this estimate at 0.5 to obtain a binary label.

Algorithm 4 Logistic Regression

Require: Training dataset (X, y) where X is a matrix of n samples and m features, and y is a vector of labels
Require: Learning rate α, number of iterations n_{iters}
Ensure: Fitted model parameters (θ, b)
  Initialize θ and b to 0
  for $iter = 1$ to n_{iters} **do**
    Compute linear predictions: $z = X\theta + b$
    Compute sigmoid activations: $a = \frac{1}{1+e^{-z}}$
    Compute gradient: $\frac{\partial J}{\partial \theta} = \frac{1}{n} X^T (a - y)$ and $\frac{\partial J}{\partial b} = \frac{1}{n} \sum_{i=1}^{n} (a^{(i)} - y^{(i)})$
    Update θ and b: $\theta = \theta - \alpha \frac{\partial J}{\partial \theta}$ and $b = b - \alpha \frac{\partial J}{\partial b}$
  end for
  return (θ, b)

  Function predict(X, θ, b):
    Compute linear predictions: $z = X\theta + b$
    Compute sigmoid activations: $a = \frac{1}{1+e^{-z}}$
    Predict labels: $\hat{y} = \{1, if\ a \geq 0.5\ 0, otherwise$
    return $\hat{y}$

Figure 6 - Logistic Regression Pseudocode

K-Nearest Neighbour Algorithm

The K-Nearest Neighbours (KNN) Classifier is a non-parametric algorithm used for classification. It operates by classifying new instances based on the class of the K-nearest neighbours in the training set. The algorithm takes a training dataset (X, y) consisting of n training samples with m features and a n-dimensional vector of training labels y_train, as well as a testing dataset X_test with k samples to classify. The parameter K specifies the number of nearest neighbours to consider in the classification process. For each sample in the testing dataset, the KNN algorithm calculates the Euclidean distance between that sample and every instance in the training dataset [44]. The Euclidean distance equation can be defined as follows:

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

Where:

- $\boldsymbol{x}$ and $\boldsymbol{y}$ are two n-dimensional data points.

The distances are stored in a distance matrix, and the K-nearest neighbours of the test instance are identified based on the shortest distances in the matrix. The corresponding labels of these neighbours are then retrieved from the training set. To classify the test instance, the majority class among the K-nearest neighbours is determined, and the test instance is assigned that class. In case of a tie, the class with the highest frequency is chosen. Finally, the algorithm returns the predicted labels for the entire testing dataset.

Algorithm 5 K-Nearest Neighbors Classifier

Require: Training dataset (X_{train}, y_{train}) where X_{train} is a matrix of n training samples and m features, and y_{train} is a n-dimensional vector of training labels
Require: Testing dataset X_{test} with k samples to classify
Require: Parameter K, the number of nearest neighbors to consider
Ensure: Predicted labels for X_{test}
  for $i = 1$ to k **do**
    Compute Euclidean distances between $X_{test}[i]$ and X_{train}: $distances = \sqrt{\sum_{j=1}^{m}(X_{test}[i][j] - X_{train}[:, j])^2}$
    Find the K nearest neighbors in X_{train} to $X_{test}[i]$ based on the computed distances
    Get the corresponding labels of the K nearest neighbors from y_{train}
    Predict the class of $X_{test}[i]$ based on the majority class among the K nearest neighbors
  end for
  return Predicted labels for X_{test}

Figure 7 - K-Nearest Neighbours Pseudocode

K-Means Clustering Algorithm

The K-Means Clustering algorithm is a widely used unsupervised learning method that aims to group similar data points into clusters. Given a list of data points (X) and the desired number of clusters (k), the algorithm begins by randomly assigning each data point to a cluster. It then initialises the centroids of each cluster as the coordinates of k randomly selected data points. The algorithm iteratively updates the assignments of each data point to the closest (in terms of Euclidean distance) cluster and recomputes the centroid of each cluster until convergence [45].

During each iteration, the algorithm computes the distance between each data point and the centroids of all clusters. It then assigns each data point to the cluster with the closest centroid. Next, the algorithm recomputes the centroid of each cluster as the mean of all data points assigned to it. This process repeats until convergence, which is achieved when the assignments of data points to clusters no longer change. The algorithm returns two outputs: a list of cluster assignments for each data point (C) and a list of centroid coordinates for each cluster (centroids).

Algorithm 6 K-Means Clustering

Require: X: a list of data points, k: the number of clusters to form
Ensure: C: a list of cluster assignments for each data point, *centroids*: a list of centroid coordinates for each cluster
Initialize C randomly such that each data point belongs to a cluster
Initialize *centroids* as the coordinates of k randomly selected data points
while not converged **do**
  for each data point x_i **do**
    Assign x_i to the cluster c with the closest centroid
  end for
  for each cluster c **do**
    Recompute the centroid as the mean of all data points assigned to it
  end for
end while
return C, *centroids*

Figure 8 - K-Means Clustering Pseudocode

5.5 User Interface Design

The user interface plays a pivotal role in any software system, but in particular for data mining and AI-infused systems [46], as it serves as the primary means of interaction between the user and the system. A well-designed user interface can significantly improve the user experience, providing an intuitive and user-friendly way for users to access the system's full range of features without any difficulty. In this section, we will present and analyse the user interface designs for the system that will form the basis of its implementation.

Main Dashboard

Upon opening the application, the main dashboard will be the first screen displayed, providing immediate access to each of the data mining algorithms available. The main dashboard has been designed to be clear and minimalistic, allowing users to easily select the desired data mining algorithm they wish to utilise. As a result, the main dashboard UI will serve as an effective means for users to quickly access the full range of capabilities offered by the tool suite. The sidebar navigation component will serve as a fast means of navigation around the system.

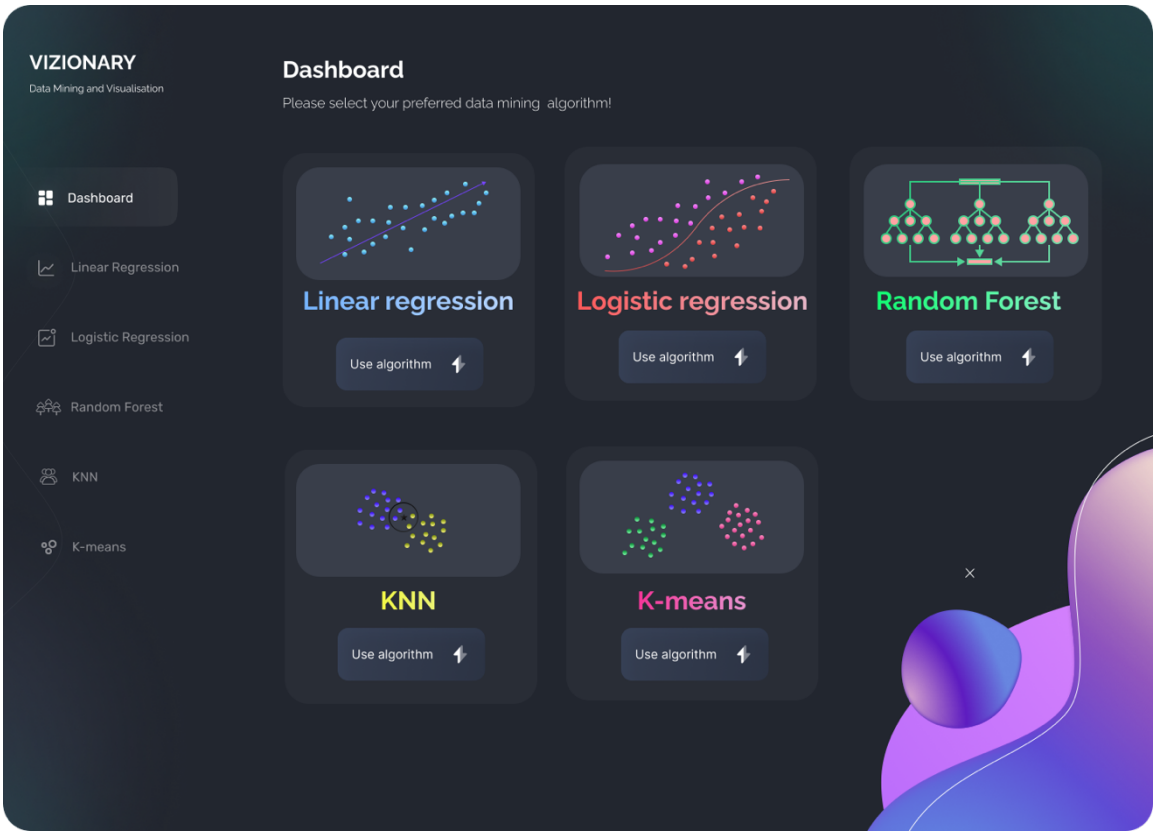


Figure 9 - Dashboard User Interface Design

Analysis Screen

Upon selecting their desired data mining algorithm, users will be presented with a user-friendly data analysis screen that allows them to upload their datasets seamlessly and visualise the results with ease. Users will be able to upload their datasets directly from their computer's file directory, provided they are in the correct format. They will then be able to select a target feature, which will be used to train the model and produce predictions. Visualisations of the results will be provided, such as scatterplots and regression lines for linear regression. In addition, users will be able to export their results for further analysis. The UI will also provide essential statistics, including the accuracy of the model. It is important to note that the UI for the other data mining algorithms will be very similar to the linear regression interface, with varying visualisations depending on the specific algorithm used. The user interface will provide clear and concise descriptions of each algorithm, as well as any necessary inputs required for each algorithm to run.

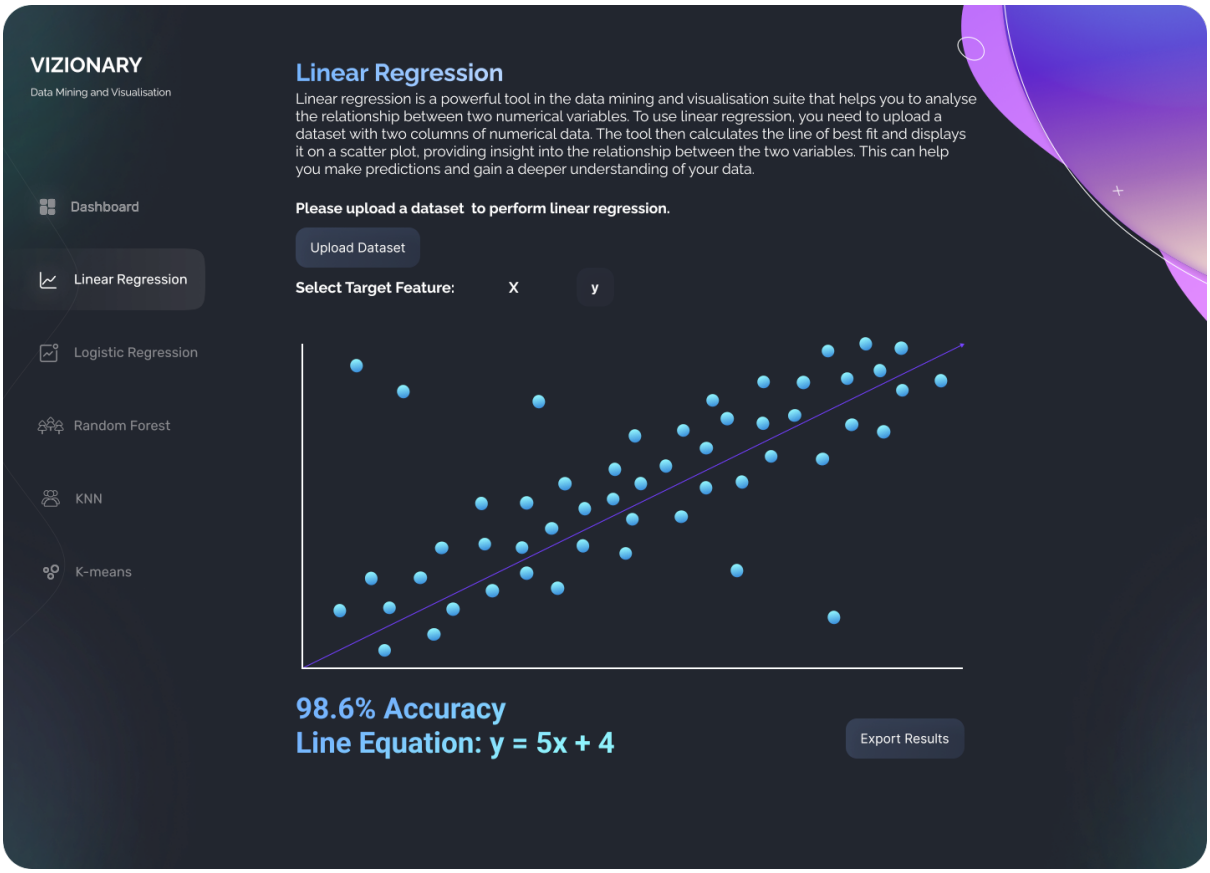


Figure 10 - Analysis Screen User Interface Design

Design Considerations

- **Vibrant Colours** – the use of vibrant colours has been shown to increase user engagement by make the interface more visually stimulating [47]; this is due to the vibrant colours grabbing the user's attention and fostering a sense of enthusiasm towards the application as opposed to the use of monotonous colour schemes. In addition, the use of contrasting colours can help to differentiate between parts of our data analysis visualisations, making it easier for the user to navigate and interact with the software.
- **Clear Typography** - comprehensible text descriptions are essential for ensuring that the user can read and understand the information presented on the interface. This is particularly important for the proposed data mining software, as the user needs to be able to understand the instructions for its use as well as interpreting the results. Therefore, the use of clear a simple font such as Raleway is preferred in this case.
- **Dark Theme** - the choice of a dark theme for software interfaces has become increasingly popular in recent years; this is likely due to a reduction in eye strain, especially when working for extended periods of time [48]. In addition, a dark theme can make the software interface more visually appealing, signifying a more sophisticated aesthetic.
- **Minimalism** – the principle of minimalism is the use of simple design elements to achieve an efficient and clear interface. This is an important design consideration for data mining software, as the user needs to focus on the data analysis without being distracted by unnecessary design elements [49].

5.6 Testing Design

Testing is an essential aspect of any software development project, as it helps to ensure that the software functions as expected to meet the requirements outset within the project's design based on the needs of its stakeholders [50]. In this case, testing is particularly crucial to ensure that the algorithms used to perform data mining are accurate and produce reliable results capable of visualising. The types of tests that will be performed include unit tests, integration tests, and acceptance tests.

Code-based testing is a fundamental aspect of software testing that involves the use of test cases designed to ensure that the program code operates as expected [50]. Consequently, code-based testing will be performed during the software development process to identify and correct defects in the code. Unit tests will be used to test individual functions and methods of the various data analysis algorithms implemented in the tool suite [50]. These tests will ensure that each function works as intended and produces the expected output. Proceeding this, integration tests will test how different parts of the system interact with each other, such as the Flask server, the different API endpoints, and the React front end. These tests will ensure that the system is functioning as a cohesive software solution.

Acceptance tests will be employed to test the overall functionality of the tool suite; these tests will be designed to ensure that the tool suite meets the requirements set forth previously based on the objectives of the project [51]. The test coverage goals will be to ensure that each function and method in the data analysis algorithms is tested thoroughly, that all API endpoints are tested with various inputs, and that the integration between the Flask server and the React front end is tested to ensure seamless communication between the two. Achieving these goals will help to ensure the overall quality and reliability of the tool suite.

6 Implementation

The following section presents the practical realisation of the proposed tool suite based on its design. The aim of this section is to provide a detailed description of the development of the tool suite, highlighting the key achievements, challenges faced, and solutions implemented. The aim is to provide a more complete comprehension of the development process and technologies used for implementation.

6.1 GitHub

Throughout the project's implementation, GitHub has been used as a version control system. This has allowed for the concurrent tracking of changes to the codebase, managing individual branches for independent testing before merging modifications into the main codebase. The use of Git grants seamless rollback to previous versions of the code, enabling any bugs or issues that arose during development to be removed if necessary. Furthermore, GitHub has facilitated transparent documentation of the project's progress, as each code change has been commented on within the platform per a given 'commit'. As a result, this implementation process has assisted my personal proficiency in utilising GitHub within software engineering.

6.2 Data Preparation

Data preparation is a crucial aspect of implementing a data mining tool suite, as it affects the accuracy and efficiency of the resulting algorithms. In this project, a range of datasets were pre-selected based on their structure and characteristics, including their cleanliness, labelling, and data format. This ensured that the datasets were appropriate for use with the selected algorithms and could be effectively processed within the system. The use of pre-selected datasets allowed for a more streamlined and focused development process, which increased the efficiency of the project as a whole. This also eliminated the need for manual data collection and cleaning, which can be a time-consuming and error-prone process.

The *Iris dataset* is a well-known dataset in the field of data mining and so was used in this project as one of the pre-selected datasets during development of our multi-classifier-based algorithms. The dataset contains 50 samples of Iris flowers, with measurements of sepal

length, sepal width, petal length, and petal width, as well as their respective species classification (Setosa, Versicolour, and Virginica) [52]. The dataset was chosen because of its clean and structured format, making it easy to work with during development. Additionally, its small size made it an ideal dataset for testing the tool suite's functionality and evaluating the performance of the implemented algorithms. The Iris dataset has been widely used in literature as a benchmark for classification algorithms, and as such, its use in this project provided a basis for comparison with other studies in the field.

The *Breast Cancer Wisconsin (Diagnostic) dataset* is another well-known dataset in the field of data mining, with the difference being it is used for binary classification tasks. This is required for our project as our logistic regression model is a binary classifier and so requires a dataset with two classifications. The dataset contains samples of breast cancer tumours, with measurements of various characteristics such as mean radius, mean texture, mean perimeter, mean area, and mean compactness, as well as a classification of whether the tumour is malignant or benign [53]. Again, this dataset has been widely used in literature as a benchmark for binary classification algorithms, and as such, its use in this project provides a basis for comparison with other studies in the field.

In addition to the use of pre-selected datasets, a custom dataset was also created for regression analysis, which had correlated X and y columns. The purpose of this dataset was to simulate a real-world scenario where the input features are interdependent. The dataset consists of 500 samples, each with only a single feature and corresponding target variable. The input feature was generated using a multivariate normal distribution with a specified covariance matrix, ensuring that the X column was correlated to the target column, but not too much. The target variable was generated by taking a linear combination of the input features, with added Gaussian noise. The use of this custom dataset allowed for the testing of the regression algorithms under conditions that were more representative of real-world

scenarios, in a format compatible with our proposed algorithm for regression.

```
import pandas as pd
import numpy as np

# Generate the data
np.random.seed(0)
cov_matrix = np.array([[1.0, 0.8], [0.8, 1.0]])
mean = np.array([0, 0])
X = np.random.multivariate_normal(mean, cov_matrix, size=500)
y = 2*X[:,0] + 3*X[:,1] + np.random.normal(size=500)

# Create DataFrame and write to CSV
df = pd.DataFrame({'X': X[:,0], 'y': y})
df.to_csv('custom_regression_dataset.csv', index=False)
```

Figure 11 - Custom Regression Dataset Generation Python Code

6.3 Flask Server Setup

The following section will describe the configuration of our Flask server, including the installation of any necessary dependencies and packages.

Server Configuration

The Flask server is an integral part of the system architecture and is responsible for managing incoming requests from the user interface and executing the data mining algorithms. In this project, the Flask server has been installed on a local machine running MacOS 13.1 (22C65), and was configured using Python 3.8. The server dependencies were managed using pip, a package manager for Python, and were installed in a virtual environment. This ensured that the dependencies were isolated from other Python projects running on the same machine, and made it simpler to manage package versions.

API Endpoints

Initially, API endpoints needed to be set up for each data mining algorithm, which were responsible for handling incoming requests, processing the input data, executing the algorithm, and returning the results to the user. The API endpoints were implemented using Flask's built-in '@app.route' decorator, defining convenient URL routes and HTTP methods for each endpoint. The API endpoints use the HTTP POST method to receive data from the user interface, and ensure it is passed to the relevant algorithm function for execution. After execution, the algorithm returns the output in a JSON format, which was sent back to the

user interface for display. The Flask server was set up to listen for requests on a specific port on the local host, ensuring secure communication between the user interface and server.

6.4 Uploading Datasets

One of the core features of the react interface is the ability for users to upload their personal datasets for processing. By allowing users to upload their own datasets, we can achieve our objective of allowing users to apply our algorithms to their specific use cases, gaining insights into their own data. In order to achieve this functionality, we have implemented a React form that allows users to select their desired file and then submit it to our server. The implementation of the React form for dataset uploading is composed of several components. The first is a file input component that allows users to select their desired file from their local machine. When doing so, the file type is limited to .csv files only to prevent erroneous file types being uploaded to the system. The second component is a button that triggers the submission of the selected file to our server via a POST request. The third is a confirmation message confirming the selected file name which will then change to a confirmation message if the upload is successful. The implementation of the React form leverages the Axios library to handle the POST request to the server. Axios is a popular JavaScript library for making HTTP requests and has proven to be a reliable choice for handling file uploads in our system. Furthermore, CORS (Cross-Origin Resource Sharing) was also required to be configured by using the Flask-CORS extension to set the necessary headers to allow requests from the React frontend. Initially, the POST request containing the uploaded data is sent to the API endpoint responsible for feature extraction, and then to its corresponding algorithm endpoint on our flask-server with the selected file passed as a parameter; this makes the file ready for processing.

6.5 Converting Files to Dataframes

After users have uploaded a given dataset for processing, it is crucial to convert the uploaded CSV file into a format that can be readily used by the system. Thus, the implementation of the file_to_dataframe function plays a pivotal role in this process as it takes an uploaded file as a parameter, returning a pandas dataframe object capable of being processed by the system. The implementation of this function involves several key steps. First, the function

checks if the uploaded file is allowed based on its file type; if the file is allowed, the function reads the contents of the file and converts it into a pandas dataframe using the read_csv method provided by the pandas library. Next, the function identifies any categorical columns within the dataframe and converts them to numeric values using the cat.codes method. This is an important step as the data mining algorithms will require numeric data inputs. The function then returns the resulting dataframe object. The implementation of this function is critical as it enables users to upload and analyse their data files in a seamless manner. The implementation of the function also ensures that the uploaded data is processed in a consistent manner, which enhances the reliability and reproducibility of the system's outputs during optimisation and testing on a variety of datasets.

```
def file_to_dataframe(file):
    if file and allowed_file(file.filename):
        file = file.read()
        df = pd.read_csv(io.StringIO(file.decode()), dtype='category')
        cat_columns = df.select_dtypes(['category']).columns
        df[cat_columns] = df[cat_columns].apply(lambda x: x.cat.codes)
        return df
    return None
```

Figure 12 - Converting CSV to Pandas Dataframe Code

6.6 Feature Extraction for Selection

The next stage required is the process of extracting feature headers from a given uploaded dataset is crucial because these headers are returned to the front end where users can select the target variable, i.e., the class labels or predictors, before applying their chosen data mining algorithm. This selection is important for training purposes, as it allows our algorithms to tailor its predictions based on the user's specific needs and dataset. As a result, the server-side algorithms will be compatible with a wider variety of datasets, thereby preventing potential errors that may arise from incorrect or assumed column header names. The function begins by attempting to read the file as a CSV using Pandas' "read_csv" function. If successful, it extracts the column headers from the resulting dataframe and returns them as a list.

6.7 Data Mining Class Structures

The following section will offer a more comprehensive insight to the implementation of the Python classes that encapsulate the data mining algorithm functionality required by our tool suite. As a general design for each of the algorithms has been given in Section 5.5, this section will concentrate more on the specific implementation details, and specific challenges faced for each of these classes. That being said, some reiteration of their workings will be offered for completeness when explaining certain code snippets.

Linear Regression Class

As depicted within Section 5.4, the implementation of linear regression in this project follows a standard formula, which involves calculating the means, variances, and covariances of the input variables, which are then used to derive the slope and y-intercept for the line of best fit. After calculating our line of best fit, we then use the slope and intercept variables to produce a set of predictions correlating to this line for plotting; the dot product of X and the slope is added to the y-intercept to generate the predicted target variable. The implementation makes use of built-in methods from the *pandas* and *numpy* libraries respectively to calculate these values efficiently. The fit and predict method is shown as follows:

```
def fit_and_predict(self, X, Y):
    # calculate means of X and Y
    mean_x = X.mean()
    mean_y = Y.mean()

    # calculate variance of X and covariance of X and Y
    var_x = X.var(ddof=1)
    cov_xy = np.cov(X.squeeze(), Y.squeeze(), ddof=1)[0][1]

    # calculate slope and y-intercept
    slope = cov_xy / var_x
    y_intercept = mean_y - slope * mean_x

    # calculate predicted target variable
    y_pred = X.dot(slope) + y_intercept

    return y_pred.flatten().tolist()
```

Figure 13 - Linear Regression Fit and Predict Code

Linear Regression Plot

In order to visualise the results of the linear regression analysis, a React component was implemented that renders a scatter graph with the line of best fit. This component receives the dataset and predictions produced by the linear regression algorithm, using the Plotly.js

scatter component to generate the graph. The scatter graph displays the input variables on the x-axis and the target variable on the y-axis. The predicted values are then plotted on the graph as a line component, forming our line of best fit for the given dataset. The bright contrasting colour scheme allows the user to quickly identify which points have the highest and lowest residuals. Thus, the scatter graph component provides a clear visualisation of the results of the linear regression analysis, making it easier for users to draw insights from the data in an efficient manner. An example plot exportation generated by the linear regression class is shown below:

Figure 14 - Linear Regression Plotly.js Visualisation

Furthermore, a residual plot component was also constructed in order to further illustrate the predictive capabilities of our model. This component plots the differences between the observed values of the dependent variable and the predicted values by the model which should help identify patterns within the errors of our model. As a consequence, the implemented residual plot component takes in regression data that contains the observed values of the dependent variable and the predicted values by the model. It then calculates the residuals by subtracting the predicted values from the observed values. The resulting residuals are then plotted on the y-axis, while the independent variable is plotted on the x-axis of a Plotly.js scatter component.

Decision Tree Class

Before implementing random forests, it is necessary to have a decision tree class as it serves as the fundamental building block for their construction. As a result, we require an efficient decision tree class to perform binary classification tasks. As depicted within Section 5.4, The Decision tree is built using the ID3 algorithm recursively starting from the root node. The algorithm finds the optimal split for the decision tree based on the feature and threshold that provides the highest information gain.

Node Class

The Node class serves as a foundational component for the construction of our decision trees Nodes in decision trees are used to represent features, thresholds, and branches in the decision-making process. The Node class provides a generic representation of a node in a decision tree, which can be specialised for different tasks and data sets. The class encapsulates the essential attributes of a node, including the feature index, threshold value, left and right child nodes, and class label. The class also includes a class label attribute that is assigned to a node if it is a leaf node. There is also method can be used to check if a node is a leaf node; if this is the case, then it means that the node cannot be further partitioned, and the decision tree has reached the end of a branch.

The output value or class associated with that leaf node is then used to make a prediction for the input that led to that particular path down the tree. The Node class will provide us with a modular way to implement decision trees, allowing for easy manipulation and traversal of the tree structure.

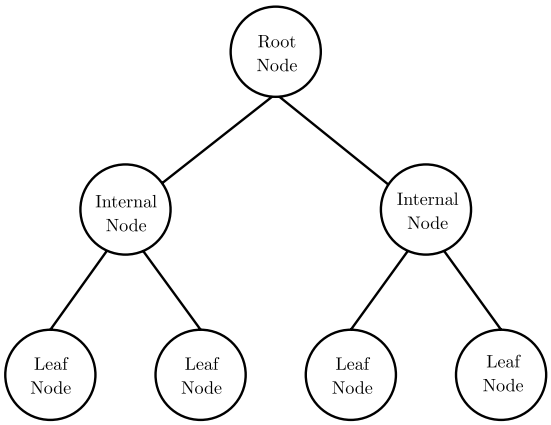


Figure 15 - Decision Tree Node Types Diagram

Decision Tree Class

The decision tree class predicts the class label of a given input sample based on a series of binary decisions.

The implementation follows a recursive structure where each node represents a split on a particular feature, with the left and right branches representing the outcomes of the split. In order to determine the optimal split for the decision tree, a random set of features is selected, and each feature's unique values are evaluated to calculate the information gain for each split. The information gain is calculated by subtracting the entropy of the child nodes from the entropy of the parent current node, hence measuring how much the entropy decreases after a split. In decision trees, entropy is a measure of the impurity of a set of labels in a dataset, i.e., the confusion of the labels. Therefore, it can be used to quantify the uncertainty associated with predicting the class of a randomly chosen sample from the set. Information gain and entropy are calculated respectively is as follows:

$$InformationGain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Where:

- $\boldsymbol{S}$ is the set of samples being split,
- $\boldsymbol{A}$ is the feature being split on,
- $\boldsymbol{S_v}$ is the subset of samples where feature A as value v.

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

Where:

- $\boldsymbol{S}$ is the set of samples being split,
- $\boldsymbol{p_i}$ is the proportion of samples that belong to class i in the current node.

After selecting the maximum information gain values, Boolean masks are constructed representing the splits for both the left and right nodes and are applied to identify our correct indices values. The feature index and threshold with the highest information gain are returned, along with the indices of samples that belong in each split node. Samples with

values less than or equal to the threshold are assigned to the left node, while those with values greater than the threshold are assigned to the right node. For reference, the method used to find the optimal split is as follows:

```
def _find_optimal_split(self, X, y, n_feats):
    feat_idxs = np.random.choice(n_feats, self.n_features, replace=False)
    gains = np.zeros((len(feat_idxs),))
    thresholds = np.zeros((len(feat_idxs),))

    for i, feat_idx in enumerate(feat_idxs):
        X_column = X[:, feat_idx]
        unique_thresholds = np.unique(X_column)
        thresholds[i] = unique_thresholds[np.random.choice(
            len(unique_thresholds))]
        gains[i] = self._information_gain(y, X_column, thresholds[i])

    best_gain_idx = np.argmax(gains)
    split_idx = feat_idxs[best_gain_idx]
    split_threshold = thresholds[best_gain_idx]

    # Get the split column from the feature matrix
    split_column = X[:, split_idx]

    # Create a boolean mask for samples that belong to the left node
    left_mask = split_column <= split_threshold

    # Create a boolean mask for samples that belong to the right node
    right_mask = split_column > split_threshold

    # Apply the masks to get the indices of the samples that belong to the left and right nodes
    left_idxs = np.where(left_mask)[0]
    right_idxs = np.where(right_mask)[0]

    return split_idx, split_threshold, left_idxs, right_idxs
```

Figure 16 - Optimal Split Determination Code

At each split, a new node is created, and the data is divided into two parts based on the selected feature and threshold. The process is repeated for each of these parts, and new nodes are created until no further splitting is possible. The stopping criterion can be met if the maximum depth of the tree is reached, if all the remaining samples belong to the same class, or if the number of samples is less than twice the minimum number of samples required to split. When the stopping criterion is met, a leaf node is created and assigned the most common label in the remaining data.

To make a prediction for a new input sample, the algorithm follows the path down the decision tree based on the feature values of the input, until it reaches a leaf node. The class label associated with that leaf node is then returned as the predicted class label for the input sample. The implementation includes methods for fitting the decision tree on a training dataset, making predictions on a test dataset, and evaluating the accuracy of the predictions.

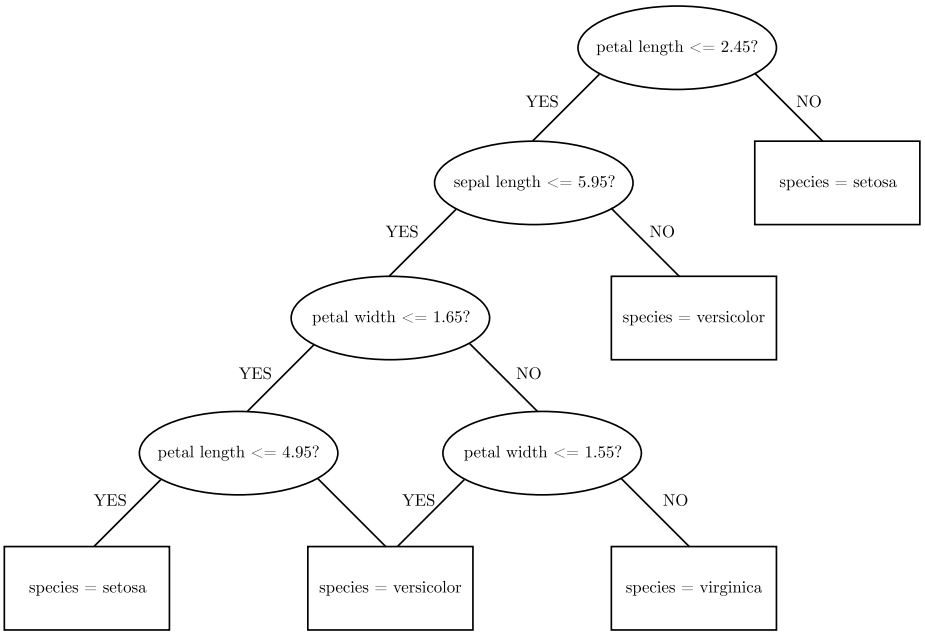


Figure 17 - Example Decision Tree Process Diagram

Random Forest Class

The random forest class combines multiple decision tree objects as previously defined to create a more robust model. In order to fit the forest to a given dataset, the random forest creates new trees by bootstrapping the data and fitting a decision tree on each bootstrap sample. As mentioned in Section 5.4, bootstrapping refers to randomly sampling the data with replacement, creating a new sample of the same size as the original data. The following function generates a bootstrapped sample of the input data by utilising the numpy library random shuffling function on the indexes of the input data, selecting the first n sample indices, and returning the corresponding subset of the input data.

```
def create_bootstrap_samples(self, X, y):
    # Create a bootstrapped sample of the data
    n_samples = X.shape[0]
    indices = np.arange(n_samples)
    np.random.shuffle(indices)
    indices = indices[:n_samples]
    return X[indices], y[indices]
```

Figure 18 - Bootstrap Sampling Code

By using bootstrapping, each decision tree is trained on a slightly different dataset, leading to diversity among the trees in the random forest. The diversity among the trees ensures that different parts of the feature space are explored and reduces the risk of overfitting. Each decision tree created during fitting is then appended to the tree list in order to 'grow' the forest. A list has been used in this case due to the dynamic nature of this growth. The

random forest's final prediction is based on a majority vote of the respective decision trees within our forest list. We therefore utilise a list comprehension to generate a set of predictions from each decision tree object, before calculating the majority vote by iterating through each of these predictions. This aggregation process results in improved accuracy and reduced variance for the model.

Logistic Regression Class

The logistic regression class implements a classification algorithm that predicts the probability of an input belonging to a particular class, and outputs a binary classification based on a decision boundary. The sigmoid function is used as the activation function for the logistic regression model; this function maps any real value to a value between 0 and 1, representing the probability of the input belonging to the positive class. The sigmoid function is applied element-wise to the output of the linear model, which is computed as the dot product of the input features and the model weights, plus the bias. The sigmoid function is defined as follows:

$$sigmoid(x) = \frac{1}{1+e^{-x}}$$

The logistic regression model works by optimising the weights and biases of the sigmoid function through iterative gradient descent, which adjusts the parameters to minimise the difference between the predicted output and the actual output. Consequently, the gradient of the loss function is used to compute the direction and magnitude of the updates to the weights and biases, and the learning rate controls the size of the updates at each iteration.

Similarly, once the model is trained, it can be used to predict the target variable for new input data by computing the linear prediction and applying the sigmoid function to get the predicted probability. Finally, the predicted probability is rounded to the nearest integer to get the predicted class label.

A key consideration required for this implementation was the process of scaling the input feature data; this is because of consistent overflow errors appearing within the application of

the sigmoid function, indicating the exponent calculation created values that were either too small or too large. To fix this, the SkLearn StandardScaler package was used to normalise the input features within the logistic regression training function. This not only fixed the overflow error encountered, but also greatly increased the predictive accuracy of our logistic regression model. That being said, within the hyperparameter optimisation process a combination of scaled and unscaled inputs will be tried to ensure this is consistently the case.

K-Means Clustering Class

The K-Means clustering class implementation follows the pseudocode discussed in Section 5.4, with a few adjustments to handle edge cases and improve performance. The K-Means algorithm starts by initialising K centroids, which represent the centre of each cluster. The 'k-means++' initialisation method is used to select the initial centroids in a way that ensures they are well spread out and reduces the risk of getting stuck in a suboptimal solution [54]. This method randomly selects the first centroid from the input matrix, and then selects subsequent centroids by sampling from the distance-based probability distribution; this aims to maximise the Euclidean distance between centroids, leading to more distinct and well-separated clusters.

```
def _kmeans_plus_plus(self):
    # initialize the centroids with KMeans++ initialization
    centroids = [self.X[np.random.randint(self.n_samples)]]

    for i in range(1, self.K):
        # calculate euclidean distances to the nearest centroid for each sample
        distances = np.array(
            [min([euclidean_distance(x, c) for c in centroids]) for x in self.X])

        # choose the next centroid based on the distances
        probabilities = distances / distances.sum()
        centroid_idx = np.random.choice(self.n_samples, p=probabilities)
        centroids.append(self.X[centroid_idx])

    return np.array(centroids)
```

Figure 19 - K-Means++ Centroid Initialisation Code

Once the initial centroids are determined, the k-means algorithm begins cluster optimisation. This is responsible for optimising the clusters by repeatedly assigning the samples to their closest centroids (again, in terms of their Euclidean distance) and then calculating new centroids from the resulting clusters. The new centroids are calculated by assigning the mean value of the samples in each cluster to the corresponding centroid. The optimisation process continues until the algorithm converges or reaches the maximum number of iterations.

Convergence is considered to be achieved when the distances between the old centroids and the new centroids are equal to zero, i.e., when the centroids stop moving between iterations. The maximum number of interactions can be assigned when instantiating the K-Means clustering class. After grouping the samples into clusters, a label is assigned to each sample based on the cluster it belongs to.

Proceeding initial implementation, there are a few optimisations that were made to improve performance. One optimisation is the use of the NumPy library to compute the distances between the data points and centroids. This allows for efficient vectorised calculations and reduces the amount of looping over the data. Another optimisation is the use of the Elbow method to determine the optimal number of clusters. This method helps to avoid the need for manual tuning of the number of clusters and ensures that the algorithm performs optimally. During the implementation phase, one challenge that was encountered was how to handle cases where a cluster has no data points assigned to it. To solve this, the algorithm assigns the nearest data point to an empty cluster as its centroid. This ensures that every cluster has at least one data point and prevents the formation of empty clusters.

K-Means Clustering Plot

This k-means clustering plot, created using the Plotly.js library, displays data points that were clustered and the cluster centres. The data used to generate the plot is obtained from a POST request to the Flask backend, returning the data points for each centroid and points within each cluster. The plot is created by passing an array of traces to the data object. The first trace represents the centroid of each cluster, which is plotted as a white cross with the x and y coordinates from the centroids data. The remaining traces represent the data points in each cluster and are each plotted with a unique colour, allowing clear visualisation of the independent clusters.

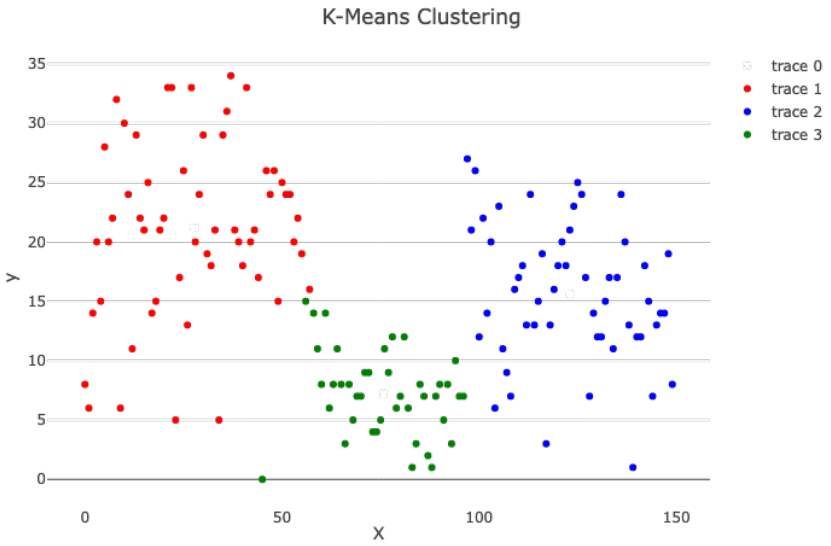


Figure 20 - K-Means Clustering Plot

K-Nearest Neighbours Classifier Class

The predict method calculates the Euclidean distance between the input instance and each instance in the training data; it then keeps track of the k nearest neighbours using a heap which stores the k smallest distances found so far. If the heap has less than k items, the current distance and label are added to it. If it has k items, the current distance and label are compared to the largest distance and label in the heap. If the current distance is smaller than the largest distance, it is added to the heap and the largest item is removed. Once the k nearest neighbours are found, the method counts the number of occurrences of each label among the neighbours using the NumPy unique function, and returns the label that appears most frequently. If multiple labels are tied for the most frequent, the first one encountered is returned.

```
def _predict(self, x):
    # compute the distance and keep track of the k nearest neighbors
    k_nearest = []
    for i, x_train in enumerate(self.X_train):
        dist = euclidean_distance(x, x_train)
        if len(k_nearest) < self.k:
            heappush(k_nearest, (-dist, self.y_train[i]))
        else:
            heappushpop(k_nearest, (-dist, self.y_train[i]))

    # count the labels of the k nearest neighbors
    labels, counts = np.unique(
        [label for (_, label) in k_nearest], return_counts=True)

    # return the most common label
    return labels[np.argmax(counts)]
```

Figure 21 - KNN Predict Method Code

During implementation, one of the challenges encountered was ensuring that the heap was sorted in ascending order of distance, while keeping track of the labels as well. The heapq library was used to maintain the heap, and the distances were negated to sort the heap in descending order, allowing only the k smallest distances to be kept. This was implemented using the heappush and heappushpop functions provided by the heapq library. Another challenge was ensuring that the algorithm was efficient enough to handle large datasets. To optimise performance, NumPy was used for vectorised computations, allowing calculations to be performed on entire arrays at once, rather than one element at a time. This significantly improved the speed of the algorithm, allowing it to handle larger datasets without becoming unreasonably slow.

Confusion Matrix Component

After creating the Python implementations of our classifiers, we require a component to visualise their accuracy for the users. As a result, we define a confusion matrix component, designed to take in two arrays of predicted and actual classification values, which are used to create a heatmap based on the given number of times a predicted value correctly correlates to the actual value it aims to predict. More specifically, to create the heatmap, the component first generates the confusion matrix within a JavaScript function before feeding the results to a Plotly.js heatmap component. The x and y properties of the data object are set to the unique classification labels, while the z property is set to the corresponding matrix values counting our correct number of predictions.

6.8 React Front End

This section will describe the implementation details of the React front-end dashboard and analysis screens, including the component implementation, key features, and functionalities of the screens, as well as integration with the Flask server. To enhance the design of the application, the Material-UI framework is utilised, which provides pre-built React components with a consistent aesthetic.

Data Mining Dashboard

The dashboard, which serves as the central navigation point for the data mining application, has been developed using a variety of React library components. In this case, the application itself consists of multiple components, each responsible for rendering a specific data mining algorithm; these components form our analysis screens for linear regression, random forest, logistic regression, K-Means, and KNN. Each of these components are imported into the main dashboard file ready for rendering upon selection.

In order to achieve a customised and responsive layout for the dashboard that matches the design within Section 5.5, several custom components were developed, such as the nested cards, side navigation, and gradient text. The nested card component was designed to display hierarchical data in a visually appealing manner, while the sidenav component was used to provide easy navigation between different sections of the dashboard. The gradient text component was developed to enhance the aesthetic appeal of the dashboard by adding a colourful and dynamic element to the typography.

The React Router library is used to create a single-page application that allows users to navigate between the different data mining algorithm components. The BrowserRouter and Routes components are used to create the routing structure, with each Route component mapping a specific URL path to a corresponding component. This allows users to access each algorithm component by simply clicking on its corresponding link, rather than having to load a new page.

The Grid component combined with useMediaQuery are utilised to create a responsive grid of cards that represent the different data mining algorithms. Each card is implemented using the NestedCard component, which renders an image, a title, and a link to the corresponding algorithm component, allowing users to easily select and navigate to their desired data mining algorithm.

Analysis Screen

The implementation of the analysis screen component in the tool suite is designed with a uniform structure, consisting of a header, navigation panel, and main content. The navigation panel presents a list of analysis options for users to select from, allowing easy navigation between analysis components. The main content is divided into sections based on the analysis selected.

The analysis screen contains several states to keep track of the selected file, the features extracted from the file, the selected target feature for classification, and finally the results of our data mining algorithm. When the component is rendered, it displays a form for uploading a CSV file, allowing the user to input their chosen dataset. When a file is selected and the "Upload" button is clicked, the component sends a POST request to the server to extract the features from the file as explained previously. If the request is successful, the extracted features are saved in the feature list state variable and the "success" state is set to true.

Once the "success" state is true, the component displays a dynamic checkboxes list to allow the user to select the feature to be used for classification, i.e., the labels within our dataset. When the user selects a target feature and clicks the "Submit" button, the component sends a POST request to the server to perform the analysis using the selected feature which will then return the results. If the request is successful, the component saves the analysis results in a results data state variable and sets the "success" state to true. The component then renders the respective plot required to display the results using the "Plot" component from the "react-plotly.js" library.

7 System Testing and Evaluation

The system testing and evaluation section of this project report aims to assess the overall performance of the developed system, while evaluating its ability to meet the outset requirements stated within Section 4.3. For our predictive models, this will cover the hyperparameter optimisation process leading to the final accuracy levels, while from a software engineering perspective this will cover a functionality and usability assessment from a prospective user's perspective.

7.1 Functionality and Usability evaluation:

The following section aims to assess the proposed functionality and ease-of-use of the developed system. It is important to note that due to the lack of human participants for testing, the results obtained from this evaluation are subjective in nature. Nonetheless, the evaluation provides valuable insights into the performance of the system and identifies areas for improvement.

Test Scenario	Test Results	User Interaction Simulation	Evaluation of Usability
Tool successfully imports data	Pass	N/A	The tool was found to be straightforward to use and provided clear instructions for importing data.
Model building and training	Pass	Trained models were able to accurately classify data.	Simulated user interactions with the model training process were found to be intuitive and easy to follow.
Visualisations of model performance and insights	Pass	Trained model predictions are illustrated with appropriate visualisations.	The visualisations are clear and visually appealing, demonstrating a vibrant representation of the given algorithm.

Evaluation metrics calculation	Pass	Metrics were calculated accurately and efficiently.	The metrics displayed to the user are intuitive and informative regarding the performance of the model on the uploaded dataset.
User interface design	Pass	N/A	The tool's user interface was found to be visually appealing and easy to navigate, with clear and concise instructions throughout.

Table 1 - Functionality and Usability Evaluation

7.2 Hyperparameter Optimisation and Performance evaluation:

Regression Analysis Metrics

In order to evaluate the performance of our linear regression model, various evaluation metrics will be used.

- **Mean Absolute Error (MAE):** the average absolute difference between the predicted values and the actual values. This allows us to quantify how far off the predictions are from the actual values.
- **Mean Squared Error (MSE):** the average difference between the predicted values and the actual values, squared. This gives us an idea of how much the predictions diverge from the actual values.
- **Root Mean Squared Error (RMSE):** the square root of the mean squared error value. This can be a more interpretable metric than mean squared error due to being in the same unit as the target variable.
- **Coefficient of Determination (R^2):** a statistic measuring the proportion of the target variable's variance that can be explained by the model's independent variables. It is a number between 0 and 1, with 0 indicating that the model explains no variance and 1 indicating that the model explains all variance.

Performance Evaluation: Linear Regression TODO – review results

To ensure that the performance of the linear regression model is tested in a number of circumstances, it is crucial to use a variety of dataset formats while evaluating it. A significantly correlated dataset will be used for evaluation of making predictions where there is a significant linear relationship between the independent variables and the dependent variable. A dataset with low correlation will also be used to evaluate the model's ability to manage noise within a dataset and reveal that there is no meaningful association between the variables.

The results for the significantly correlated dataset, as generated per Section 6.2, are as follows:

Metric	Value
MAE	75.28
MSE	9829.46
RMSE	99.14
R^2	0.53

Table 2 - Linear Regression Results

Based on the data provided, the linear regression model has an MAE of 75.28, indicating that, on average, the predicted values are 75.28 units off from the actual values. The model's RMSE is 99.14, indicating that, while the model's predictions and actual values are reasonably similar, there is still room for improvement. According to the model's R2 value of 0.53, the independent variables account for 53% of the variance in the target variable. Despite the presence of errors in the model's predictions, the plot of the data suggests that the line of best fit must be reasonable accurate in capturing the underlying trends and patterns in the data.

Classification Model Evaluation Metrics

In order to assess the performance of our classification models, various evaluation metrics will be used. In this case, we are assessing the model performance for multi-class classification problems, where there can be more than two possible classes, and so metrics such as precision, recall, F1-score, and accuracy can a useful indication of how well the model is performing [55]. They are defined as follows:

$$Precision = \frac{TP}{TP + FP}$$

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

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

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

Where:

- ***TP*** (True Positives) is the number of correctly predicted positive instances.
- ***TN*** (True Negatives) is the number of correctly predicted negative instances.
- ***FP*** (False Positives) is the number of negative instances that were incorrectly predicted as positive.
- ***FN*** (False Negatives) is the number of positive instances that were incorrectly predicted as negative.

They can be adapted to be calculated on a per-class basis or across all classes using various aggregation methods. Due to the dynamic nature of our tool suite's data inputs, an aggregated result will be used to assess each model.

Performance Evaluation: Decision Tree

The hyperparameter optimization process for the decision tree model involved exploring various combinations of two hyperparameters: the maximum depth and the minimum sample split. The highest accuracy of 0.97 was achieved with a maximum depth of 150 and a minimum sample split of 6. The results are summarised in the following table:

Max Depth	Minimum Sample Split	Accuracy
10	2	0.76
10	4	0.79
10	6	0.82
20	2	0.83
20	4	0.85
20	6	0.87
50	2	0.88

50	4	0.91
50	6	0.92
100	2	0.91
100	4	0.93
100	6	0.95
150	2	0.94
150	4	0.96
150	6	0.97

Table 3 - Decision Tree Hyperparameter Optimisation Results

The increase in the max depth hyperparameter likely lead to a more complex decision tree with more nodes and branches, resulting in the model capturing more intricate patterns in the data. However, a deeper tree may also lead to overfitting, where the model becomes too closely tailored to the training data and fails to generalise well to new, unseen data. On the other hand, increasing the minimum sample split hyperparameter helps to prevent overfitting by requiring a minimum number of samples in a leaf node before it can be split further. This encourages the model to learn more generalisable patterns in the data. As a result, there is a trade-off between the two hyperparameters, where increasing one may lead to better performance up to a certain point, but beyond that point, further increases may lead to diminishing returns or even worse performance. In this case, it appears that a max depth of 150 and a minimum sample split of 6 strike a good balance between capturing complex patterns and preventing overfitting, resulting in the highest accuracy of 0.96.

Performance Evaluation: Random Forest

The hyperparameter optimisation the random forest model was conducted by testing different configurations for number of estimators (decision trees) only; this is because our analysis of the decision trees within the forests provided an optimal configuration for the remaining hyperparameters. The results are summarised in the following table:

Number of Estimators	Accuracy
1	0.97
3	0.73
5	0.93
10 (optimal)	0.97
20	0.97
50	0.97
100	0.97

Table 4 - Random Forest Hyperparameter Optimisation Results

The results shown indicate that the accuracy of the model increases as the number of estimators increase, up to a certain point where adding more estimators does not improve performance. Specifically, the optimal number of estimators was found to be 10 with an accuracy of 0.97. Increasing the number of estimators beyond 10 resulted in no significant increase in accuracy, with the accuracy plateauing at 0.9667 for 20, 50, 100 estimators. As the number of decision trees increases, the random forest can better capture the complexity of the data and the decision boundary, leading to improved accuracy. However, as the number of decision trees increases further, the model may have started to overfit the training data, leading to diminishing returns and no significant increase in accuracy on the test data. Thus, the optimal number of estimators must balance the trade-off between model complexity and performance.

Performance Evaluation: Logistic Regression

The hyperparameter optimisation for the logistic regression model was conducted by testing different configurations for learning rate and maximum iterations. The learning rate determines how quickly the algorithm learns from the training data, while the number of iterations determines the maximum number of times the algorithm will iterate over the entire training dataset during the optimisation process. As mentioned during the implementation of our logistic regression model, there were issues with the scaling of the data and so testing has been conducted for both scaled and unscaled inputs. The results are summarised in the following table:

Learning rate	Max iterations	Feature scaling	Accuracy
0.001	1000	None	0.69
0.01	1000	None	0.77
0.1	1000	None	0.75
0.001	5000	None	0.72
0.01	5000	None	0.76
0.1	5000	None	0.71
0.001	1000	StandardScaler	0.88
0.01 (optimal)	1000 (optimal)	StandardScaler (optimal)	0.96
0.1	1000	StandardScaler	0.95
0.001	5000	StandardScaler	0.93
0.01	5000	StandardScaler	0.95
0.1	5000	StandardScaler	0.92

Table 5 - Logistic Regression Hyperparameter Optimisation Results

The results show that increasing the learning rate generally improves the accuracy of the logistic regression model up to a certain point. For instance, the model with a learning rate of 0.01 achieves the highest accuracy in most cases, while a learning rate of 0.1 may lead to reduced accuracy. However, when the learning rate is too small, such as 0.001, the model may take longer to converge, achieving a lower accuracy. Moreover, the maximum number of iterations has a significant effect on the model's accuracy; increasing the number of iterations improves performance up to a certain point, after which the model may overfit the training data and reduce accuracy. This explains why the model with a maximum of 1000 iterations generally performs better than the model with 5000 iterations. Regarding feature scaling, the results indicate that using standard scaling consistently improves the model's accuracy across all hyperparameter configurations. This can be attributed to the fact that feature scaling helps to normalise the range of predictor variables, making the algorithm less sensitive to the magnitude of each feature. The model with standard scaling and a learning rate of 0.01 and a maximum of 1000 iterations performs the best with an accuracy of 0.96.

Performance Evaluation: K-Nearest Neighbours

The hyperparameter optimisation the k-nearest neighbours model was conducted by testing different configurations for number of neighbours (k) only. This represents the number of nearest data points to a given point that are used to predict its class. The results are summarised in the following table:

k	Accuracy
1	0.86
3	0.89
5 (optimal)	0.93
7	0.91
10	0.88

Table 6 - KNN Hyperparameter Optimisation Results

These results indicates that the accuracy of our KNN classifier is heavily influenced by the choice of its single hyperparameter, k. When k is set to small values like 1, the accuracy of the classifier is found to significantly reduced, likely due to overfitting of the training data. In contrast, when k is set to larger values like 10, the model is seen to underfit the data, failing to capture the data's complexities. The results suggest that the model's accuracy

tends to improve as we increase the value of k, up to a certain point, beyond which it starts to decrease, possibly due to the inclusion of irrelevant data points. The increase in accuracy with k could be attributed to increased generalization of the model, making it less sensitive to noise and outliers in the data. However, beyond a certain point, the model becomes too general and includes more irrelevant data points, leading to a decrease in accuracy. The results demonstrate that 5 is an optimal value for the number of neighbours to be used within our model.

Clustering Model Evaluation Metric

The Silhouette Score is a widely used metric that can be used to assess the performance of clustering algorithms such as K-Means. It is a measurement of the degree of separation between clusters and the degree of similarity within clusters. A higher score indicates a better fit to its own cluster and a worse fit to other clusters, while a lower score indicates possible misclassification. The formula for calculating Silhouette score is as follows:

$$s = \frac{b - a}{max(a, b)}$$

Where:

- $\boldsymbol{a}$ is the mean distance between a sample and all other points in the same cluster
- $\boldsymbol{b}$ is the mean distance between a sample and all other points in the nearest cluster

Performance Evaluation: K-Means Clustering

The hyperparameter optimization for our K-Means clustering model involves varying the maximum number of iterations the algorithm is run for, i.e., the number of times we optimise our clusters unless they converge before this limit. The results are summarised in the following table:

Max Number of Iterations	Silhouette Score
1	0.36
2	0.44
4	0.53
5	0.56
10	0.56

50	0.56
100	0.56

Table 7 - K-Means Hyperparameter Optimisation Results

The table shows that as the maximum number of iterations is increased from 1 to 5, the Silhouette Score is increased. At this point, however, the silhouette score plateaus at 0.56 after more than 5 rounds, indicating that the quality of the clustering does not significantly increase past this point. This plateau is most likely to have been caused by the K-Means algorithm's early convergence, negating the impact of raising the maximum number of iterations further. As a result, 5 would be the optimal value for the maximum number of iterations, yielding the highest Silhouette Score of 0.56. That being said, the algorithm's point of convergence will be heavily dependent on the particular dataset being examined and so a greater value of around 300 maximum iterations will be set to ensure flexibility while balancing computational efficiency. The Silhouette score of 0.56 confirms the effectiveness of the algorithm, although there is certainly room for improvement. Consequently, in order to evaluate the model performance further we will refer to our visualisation. When plotting the clusters, there is clear distinction between each of the cluster groups, indicating that the algorithm has effectively separated the data into discrete categories. Therefore, these results suggest that the K-Means clustering algorithm is an effective method for grouping this particular dataset.

7.3 Requirements Review

The following section will review the outset requirements for the project as depicted within Section 4.2. This will be used to provide a complete picture for the success of the project's implementation, and as a point of further personal reflection to be explored within Section 8.

Requirement	**Pass Mark (/10)**	**Remarks**
1. Data Input (MUST)	6	Users are able to successfully upload their chosen .csv datasets, achieving the required base functionality.

		However, the remaining dataset types are not currently supported.
2. Data Cleaning (COULD)	2	The system does not currently have explicit measures to clean and format a dataset in terms of input, However, there is capacity to scale datasets where required such as for logistic regression.
3. Error Management (MUST)	8	Despite dataset compatibility errors being shown, further measures could be implemented to explain why the dataset is not compatible with the given algorithm.
4. Data Mining		
a. Supervised Learning		
i. Linear Regression Predictor (MUST)	10	Implemented as described, producing valid results.
ii. Logistic Regression Classifier (MUST)	10	
iii. Decision Tree Classifier (MUST)	10	
iv. Random Forest Classifier (COULD)	10	
b. Unsupervised Learning		
i. K-Means Clustering (MUST)	10	Implemented as described, producing valid results.
ii. Principal Component	N/A	Not implemented due to timeframe constraints.

Analysis (COULD)		
iii. Multidimensional Scaling (COULD)	N/A	
iv. Association Rule Mining (COULD)	N/A	
5. Data Visualisation (MUST)	8	A range of relevant data visualisations have been included, however more could be developed if desired for further analysis.
6. Analysis Export (SHOULD)	8	The Plotly.js visualisation components generated from the analysis can be downloaded as PNGs. There is potential for more comprehensive downloads as explained later in Section 8.2.
7. User-friendly Interface (MUST)	N/A – Pass	As this is subjective in nature, assigning a numerical value has been omitted but it can be considered as a 'passed' requirement due to the results from Section 7.1. Please see the appendix for screenshots of the final application user interface.
8. Flexibility (MUST)	N/A – Pass	As this is subjective in nature, assigning a numerical value has been omitted but it can be considered as a 'passed' requirement due to the results from Section 7.1.
9. Scalability (SHOULD)	10	The size of the datasets entered to the tool do not appear to impede its performance in producing results, both

		in terms of computational efficiency and accuracy.
10. Security (COULD)	N/A	As the project is currently only deployed locally, the data has not yet been encrypted but can be considered secure for its present deployment state. If the server was to be deployed externally, RSA encryption can be seen as a secure method for data transfer in addition to HTTPS for our POST requests. A numerical score has therefore been omitted for this requirement as it is technically secure but not in the intended sense for the outset requirement.
11. Technical Support (COULD)	8	The system provides sufficient descriptions of each of the algorithms and their expected use, although a more comprehensive user guide could be produced with example datasets and interpreting the outputs.
Percentage Completion	100/150 (67%)	Despite a seemingly low requirement completion percentage of 67%, the majority of the missed requirements were categorised as 'could have'. Given the timeframe for its completion amongst the additional learning required from both a conceptual and technical perspective, the project's overall completion rate can be considered highly satisfactory in terms of those absolutely necessary

		for its success. However, there is certainly room for further development moving forward as will be explored within Section 8.

Table 8 - Requirements Review

8 Reflection

The following section will be a reflective commentary on the project, explaining the personal development experienced as a result of its completion, considering its limitation and potential moving forward as well as a final conclusive analysis of the achievements made.

8.1 Personal Development and Challenges Faced

This section will explore the challenges faced during the project's completion, as well as illustrating the personal development that has been achieved as a result of overcoming these challenges.

Data Analysis Fundamentals

Learning the fundamentals of data analysis, mining and visualisation techniques has been a challenging but rewarding experience within this project. Initially, I struggled with the mathematical concepts underlying data analysis and mining, such as regression, clustering, and classification. However, through the consistent revision of academic literature, I developed a better understanding of these concepts and how they can be applied to real-world problems. Subsequently, the final year module 'Data Mining and Machine Learning' has been particularly useful in clarifying my understanding of the techniques utilised for evaluating certain data mining algorithms; as this is a second semester module, the initial research stage had already been completed prior to its commencement. Furthermore, when trying to visualise the results of the analysis in a clear and concise manner, significant experimentation with different visualisation tools and techniques was required to find the most effective way to present the data. A substantial amount of research was required in order to develop the bespoke visualisation of the analysis results. Ultimately, this project has

been an excellent opportunity for personal development, allowing me to enhance my technical skills and deepen my understanding of data analysis, mining, and visualisation techniques.

Effective researching

Initially, it was challenging to identify the most relevant and informative academic resources due to the constantly evolving nature of the field of data analysis, with new algorithms and techniques being introduced regularly. Developing a foundational knowledge of data analysis and visualisation techniques through reputable textbooks proved to be crucial before tackling more complex literature. By concentrating on reputable sources and staying up-to-date with the latest developments in the field, I was able to gain a better understanding of the most appropriate techniques to implement within the tool suite. The extensive research required to complete the project to a high personal standard significantly enhanced my ability to effectively research and assess information, which will undoubtedly prove valuable for the remainder of my studies and future professional career

Understanding of Python and Flask Server Development

The Flask framework proved to be a challenging yet rewarding aspect of the project, as it required a comprehensive understanding of both server functionality and the intricacies of the Python language. Drawing on the knowledge gained from modules such as 'Introduction to Programming' and 'AI Methods', as well as my prior experience as a data analyst during a recent summer internship, I was able to develop a solid understanding of the Python language which served as a solid foundation for Flask-specific implementation issues. Despite the steep initial learning curve, through continuous practice and problem-solving, I was able to improve my understanding of Flask development and overcome technical difficulties associated with implementing RESTful APIs. This experience has proven invaluable in enhancing my skill set as a software developer and has provided me with a solid foundation for future server-side development projects.

Developing a Front-end User Interface in React

The development of the front-end user interface using React posed a significant challenge during the project. While I had some prior experience with JavaScript, using React was entirely new to me, and I had to learn it from scratch. This involved familiarising myself

with React's component-based architecture, as well as understanding the virtual DOM and JSX syntax. Although this was challenging, I found that the React documentation as well as other online tutorials, were extremely helpful in overcoming these obstacles. As I progressed with the development, I found that my understanding of React increased, and I was able to create a more sophisticated user interface that was both visually appealing and intuitive to use. This experience has significantly enhanced my skills in front-end development and has equipped me with the necessary knowledge and expertise to tackle similar challenges in future projects.

8.2 System Limitations and Future Improvement

Throughout the development of our software solution, several future improvements have been identified that could further enhance its functionality moving forward. These improvements represent a natural evolution for the tool suite, building on its current capabilities to provide more advanced features and greater flexibility for users.

User Accounts and Model Management

A primary potential improvement of the project is the addition of user accounts within the tool suite. Currently, the suite operates in a stateless fashion with no prior data stored for a given user to ensure ease of use and readily available access to the analysis techniques. That being said, if implemented correctly, the creation of user accounts can facilitate the development of other features not yet realised by the tool suite that may be useful to many users. The first of these is the ability to save and reuse trained models; this would entail storing the trained data mining objects in an external database, allowing repeated prediction and training opportunities to users. This feature would allow users to save time and effort by avoiding the need to retrain models from scratch each time new data is added, as well as providing a basis for performance comparison among the different algorithms offered within the software. Currently, the only way to store results is to export the visualisations from the tool suite into the user's personal file directory and so this may allow the tool suite to be more all-inclusive for their data mining requirements. Moreover, after providing the ability to store a given trained model, sharing access rights can be seen as the next likely development in functionality. This would provide users with the ability to share access rights for their stored models, enabling users to collaborate on a project and efficiently share data

with team members. That being said, these enhancements present a significant challenge in terms of security and data privacy and so must be approached with due care.

Sample-based Estimations

The ability to enter a sample of data and receive an estimated categorization or prediction value is another potential improvement for the data mining tool suite. Currently, the tool suite requires a complete dataset to perform analysis and generate predictions or classifications. However, users may want to quickly estimate the outcome of a certain data sample by making use of their already trained models predictive capabilities. Implementing this feature would allow users to quickly obtain preliminary insights for new data samples. For instance, as alluded to within Section 2.4, if we trained our regression model on a housing dataset, we could allow the user to directly make use of the predicted equation within the tool suite and receive an estimate for future house prices. Furthermore, this feature would present a method of validation for users to evaluate the performance of their trained models by inputting a known dataset and comparing the estimated output to the actual output.

Enhanced Data Input Flexibility

Currently, the analysis screen data upload only accepts data in a .csv format. However, the ability to integrate with a wider variety of data sources would provide users with more flexibility within their analysis, expanding the tool suite's applications. Initially, this could be expanded to a wider variety of file types such as excel files (.xlsx, .xls), tab-separated values (.tsv), and JSON files (.json). Furthermore, users may want to analyse data stored in a relational database management system (RDBMS), such as MySQL or Oracle, without having to export the data into a .csv file. By integrating with RDBMS systems, users could connect directly to their database and query the data for analysis, providing a more efficient workflow. Additionally, integration with application programming interfaces (APIs) could enable the tool suite to access data from various sources, such as social media platforms, weather data, or financial data feeds. Therefore, this flexibility would enable the tool suite to be used in a wider range of applications and use cases.

Enhanced Export Functionality for Analysis Results

Exporting analysis results into a variety of formats is another potential improvement for the data mining tool suite. Presently, the tool suite only provides the ability to export visualisations in the PNG image file format. However, providing the ability to export results in various formats, such as .csv, would allow users to further manipulate the results and use them in other applications. For instance, exporting clustering results as a .csv file would allow users to label the data points and then use it as training data for a classification algorithm. This would enable the tool suite to be used for a wider range of applications and provide more value to users. Additionally, exporting results as a .csv file would allow users to integrate the results with other software applications and tools, allowing users to share their results with others who may not have access to the tool suite or may prefer to work with data in a different format; this would enable users to gain insights from the data in a way that best suits their needs.

ChatGPT API Integration

Based on the present GPT-3.5 architecture, ChatGPT is an artificially intelligence language model developed by OpenAI, that is capable of understanding natural language and generating intelligible responses in real-time [56]. By utilising its adept natural language processing, the ChatGPT API could provide a user-friendly interface for explaining the results of a given data mining algorithm to the user, allowing them to ask specific questions regarding their analysis, and providing helpful feedback based on their queries. To ensure that only relevant outputs are returned, parameters would need to be set on the ChatGPT API; these parameters could include the type of analysis being conducted, the specific data being analysed, and the user's level of expertise in the subject matter. This integration would not only enhance the overall usability of the tool suite but also provide a more intuitive means for users to understand their data mining results. In essence, it would hopefully decrease the relative effort required for users to analyse their data.

8.3 Data Mining and Visualisation Achievements (Concluding Remarks)

As the world becomes almost entirely reliant on digital tools to support all aspects of our lives, the quantity of data generated each day continues to grow at an exponential rate [57]. This growth has led to the formation of seemingly unending data sets, with potential insights

only data analysis processes can unlock. Data mining techniques provide the automation of the analysis required to extract useful information from these large data sets. As such, data mining has become an essential area of study that enables businesses, governments, and educational institutions to make data-driven decisions in the modern world [58]. Within our concluding remarks, a reflection will be given on how successful the implemented software solution was in addressing a resolution to the challenges presented by this abundance of data.

In reflection, the initial breadth of the project certainly fostered ambiguity in terms of system design and the intended functionality breakdown of the tool suite moving forward. There was a persistent question of the feasibility of the tool suite in being both accessible to the everyday user but also implementing a wide enough range of high performing data mining models. Given the time constraints surrounding the project, the accomplishments made in terms of balancing computational ability while maintaining usability are undoubtedly something to be proud of. Although the implementations themselves may not be revolutionary in their design, it is the manner in which they are made accessible that differentiates them within the field of data mining and software engineering.

The focal achievement of this project was bridging the gap between technically proficient and non-technical users to enable the effective use of data mining techniques behind a simple and self-explanatory user interface. Traditionally, data mining techniques have been at the sole exploit of experts in the field who possess the specialised knowledge and technical proficiency to implement the measures themselves for a bespoke use case. This has made it difficult for non-technical users to take full advantage of the abundance of data available to them beyond the rudimentary application of commonplace software packages. Consequently, this projects seeks to empower the average user to train highly capable predictive models to generate accurate insights for their own personal datasets. On the tested datasets used for optimisation purposes, the implemented classifier models consistently performed to levels of 90%+ accuracy for our classifiers, with our linear regression and clustering implementations also generating adept visualisations for their respective use case. All of our outset project objectives from Section 1.2 have been met to a high degree of satisfaction.

9 References

[1] V. Mayer-Schönberger and K. Cukier, Big Data: A revolution that will transform how we live, work, and think, Boston: Mariner Books, Houghton Mifflin Harcourt, 2014.

[2] T. A. Runkler, Data Analytics: Models and Algorithms for Intelligent Data Analysis, 3rd Edition ed., Wiesbaden: Springer Vieweg, 2020.

[3] G. Cao, Y. Duan and G. Li, "Linking Business Analytics to Decision Making Effectiveness: A Path Model Analysis," *IEEE Transactions on Engineering Management,* vol. 62, no. 3, pp. 384-395, 2015.

[4] D. L. Olson, Predictive Data Mining Models, 2nd Edition ed., Springer, 2020.

[5] B. Bilalli, A. Abelló, T. Aluja-Banet and R. Wrembel, "Intelligent Assistance for data pre-processing," *Computer Standards & Interfaces,* vol. 57, 2018.

[6] S.-A. N. Alexandropoulos, S. B. Kotsiantis and M. N. Vrahatis, "Data preprocessing in Predictive Data Mining," *The Knowledge Engineering Review,* vol. 34, 2019.

[7] J. W. Osborne, Best practices in data cleaning: A complete guide to everything you need to do before and after collecting your data, New York: Independently Published, Jason W. Osborne, 2019.

[8] X. Chu, I. F. Ilyas, S. Krishnan and J. Wang, "Data Cleaning: Overview and Emerging Challenges," 2016.

[9] A. Bommert, X. Sun, B. Bischl, J. Rahnenführer and M. Lang, "Benchmark for filter methods for feature selection in high-dimensional classification data," *Computational Statistics & Data Analysis,* vol. 143, 2020.

[10] J. Song, G. Alter and H. V. Jagadish, "Structured data transformation algebra (SDTA) and its applications," *Distributed and Parallel Databases,* vol. 40, no. 2-3, 2022.

[11] G. Aurélien, Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition ed., Beijing: O'Reilly, 2019.

[12] R. Tatman, "Data Cleaning Challenge: Scale and normalize data," 2018. [Online]. Available: https://www.kaggle.com/code/rtatman/data-cleaning-challenge-scale-and-normalize-data. [Accessed 11 December 2022].

[13] S. Li, N. Marsaglia, C. Garth, J. Woodring, J. Clyne and H. Childs, "Data reduction techniques for simulation, visualization and data analysis," *Computer Graphics Forum,* vol. 37, no. 6, 2018.

[14] A. Kirk, Data Visualisation: A Handbook For Data Driven Design, 2nd Edition ed., Sage Publications, 2019.

[15] L. Shao, A. Mahajan, T. Schreck and D. J. Lehmann, "Interactive Regression Lens for Exploring Scatter Plots," *Computer Graphics Forum,* vol. 36, no. 3, pp. 157-166, 2017.

[16] S. B. Bavdekar, "Using Tables and Graphs for Reporting Data," *Journal of The Association of Physicians of India,* vol. 63, 2015.

[17] S. Narkhede, "Understanding Confusion Matrix," Towards Data Science, 9 5 2018. [Online]. Available: https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62.

[18] M. Bramer, Principles of Data Mining, 4th Edition ed., London: Springer, 2020.

[19] J. C. Stoltzfus, "Logistic Regression: A Brief Primer," *Academic Emergency Medicine,* vol. 18, no. 10, pp. 1099-1104, 2011.

[20] C. Thrane, Applied Regression Analysis : Doing, Interpreting and Reporting, 1st Edition ed., Abingdon, Oxon ; New York, NY: Routledge, 2020.

[21] A. Urquhart, "Price clustering in Bitcoin," *Economics Letters,* vol. 159, pp. 145-148, 2017.

[22] X. Huang, L. Wu and Y. Ye, "A review on dimensionality reduction techniques," *International Journal of Pattern Recognition and Artificial Intelligence,* vol. 33, no. 10, 2019.

[23] B. M. Salih Hasan and A. M. Abdulazeez, "A Review of Principal Component Analysis Algorithm for Dimensionality Reduction Authors," *Journal of Soft Computing and Data Mining,* vol. 2, no. 1, 2021.

[24] N. Saeed, H. Nam, T. Y. Al-Naffouri and M.-S. Alouini, "A State-of-the-Art Survey on Multidimensional Scaling-Based Localization Techniques," *IEEE Communications Surveys & Tutorials,* vol. 21, no. 4, pp. 3565-3583, 2019.

[25] L. T. Becker and E. M. Gould, "Microsoft Power BI: Extending Excel to Manipulate, Analyze, and Visualize Diverse Data," *Serials Review,* vol. 45, no. 3, pp. 184-188, 2019.

[26] M. Pearson, B. Knight, D. Knight and M. Quintana, Pro Microsoft Power Platform, Berkeley, CA: Apress , 2020.

[27] D. G. Murray, Tableau your data!: Fast and Easy Visual Analysis with Tableau Software, Indianapolis: Wiley, 2016.

[28] Qlik, "Qlik sense | modern cloud analytics," [Online]. Available: https://www.qlik.com/us/products/qlik-sense. [Accessed 28 December 2022].

[29] Google, "Looker Business Intelligence Platform & Embedded Analytics | Google Cloud," [Online]. Available: https://cloud.google.com/looker. [Accessed 28 December 2022].

[30] E. Dabbas, Interactive dashboards and data apps with plotly and dash: Harness the power of a fully fledged frontend web framework in python - no javascript required, Birmingham; Mumbai: Packt, 2021.

[31] R. Amit and C. Nagaraj G, "Application of Lean Principles in Software Development Processes," in *2021 International Conference on Disruptive Technologies for Multi-Disciplinary Research and Applications (CENTCON)*, Bengaluru, India, 2021.

[32] A. Hudaib, R. Masadeh, M. H. Qasem and A. Alzaqebah, "Requirements Prioritization Techniques Comparison," *Modern Applied Science,* vol. 12, no. 2, 2018.

[33] M. Grinberg, Flask Web Development, O'reilly Media, Incorporated, 2018.

[34] P. Rawat, "ReactJS: A Modern Web Development Framework," *International Journal of Innovative Science and Research Technology,* vol. 5, no. 11, 2020.

[35] I. Stančin and A. Jović, "An overview and comparison of free Python libraries for data mining and big data analysis," in *42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, Opatija, Croatia, 2019.

[36] S. Gruner, J. Pfrommer and F. Palm, "RESTful industrial communication with OPC UA," *IEEE Transactions on Industrial Informatics,* vol. 12, no. 5, pp. 1832-1841, 2016.

[37] C. Tapsai, "Information Processing and Retrieval from CSV File by Natural Language," in *2018 IEEE 3rd International Conference on Communication and Information Systems (ICCIS)*, Singapore, 2018.

[38] F. Nelli, Python Data Analytics, 2 ed., Berkeley, CA: Apress, 2018.

[39] S. Patni, Pro RESTful APIs, Design, Build and Integrate with REST, JSON, XML and JAX-RS, Berkeley, CA: Apress, 2017.

[40] D. C. Montgomery, E. A. Peck and G. Vining, Introduction to Linear Regression Analysis, Oxford: Wiley-Blackwell, 2013.

[41] A. Rajeshkanna and K. Arunesh, "ID3 Decision Tree Classification: An Algorithmic Perspective based on Error rate," in *2020 International Conference on Electronics and Sustainable Communication Systems (ICESC)*, Coimbatore, 2020.

[42] G. Biau and E. Scornet, "A random forest guided tour," vol. 25, no. 2, pp. 197-227, 2016.

[43] T. Hastie, R. Tibshirani and J. H. Friedman, The Elements of Statistical Learning 2nd Edition, Springer, 2009.

[44] L.-Y. Hu, M.-W. Huang, S.-W. Ke and C.-F. Tsai, "The distance function effect on K-nearest neighbor classification for medical datasets," *SpringerPlus,* vol. 5, no. 1, 2016.

[45] S. Na, L. Xumin and G. Yong, "Research on k-means Clustering Algorithm: An Improved k-means Clustering Algorithm," in *2010 Third International Symposium on Intelligent Information Technology and Security Informatics*, Jian, China, 2010.

[46] S. Amershi, D. Weld, M. Vorvoreanu, A. Fourney, B. Nushi, P. Collisson, J. Suh, S. Iqbal, P. N. Bennett, K. Inkpen, J. Teevan, R. Kikin-Gil and E. Horvitz, "Guidelines for Human-AI Interaction," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems - CHI '19*, 2019.

[47] S. Singh, "Impact of Color on Marketing," *Management Decision,* vol. 44, no. 6, pp. 783-789, 2006.

[48] X. Xie, F. Song, Y. Liu, S. Wang and D. Yu, "Study on the Effects of Display Color Mode and Luminance Contrast on Visual Fatigue," *IEEE Access,* vol. 9, p. 35915–35923, 2021.

[49] S. Krug, Don't make me think, revisited : a common sense approach to Web usability, Berkeley, Calif.: New Riders, 2014.

[50] P. C. Jorgensen, Software testing: A craftsman's approach, 4th Edition ed., Hoboken: CRC Press, 2013.

[51] L. Crispin and J. Gregory, Agile Testing: a practical guide for Testers and agile teams, Upper Saddle River: Addison-Wesley, 2014.

[52] "Iris Species," www.kaggle.com, [Online]. Available: https://www.kaggle.com/datasets/uciml/iris.

[53] "Breast Cancer Wisconsin (Diagnostic) Data Set," www.kaggle.com, [Online]. Available: https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data.

[54] A. Kapoor and A. Singhal, "A comparative study of K-means, K-means++ and fuzzy c-means clustering algorithms," in *2017 3rd IEEE International Conference on Computational Intelligence & Communication Technology (CICT)*, Ghaziabad, 2017.

[55] Ž. Đ. Vujovic, "Classification Model Evaluation Metrics," *International Journal of Advanced Computer Science and Applications,* vol. 12, no. 6, 2021.

[56] I. Cribben and Y. Zeinali, "The Benefits and Limitations of ChatGPT in Business Education and Research: A Focus on Management Science, Operations Management and Data Analytics," *SSRN Electronic Journal,* 2023.

[57] M. Naeem, T. Jamal, J. Diaz-Martinez, S. A. Butt, N. Montesano, M. I. Tariq, E. De-la-Hoz-Franco and E. De-La-Hoz-Valdiris, "Trends and Future Perspective Challenges in Big Data," *Advances in Intelligent Data Analysis and Applications,* vol. 253, pp. 309-325, 2021.

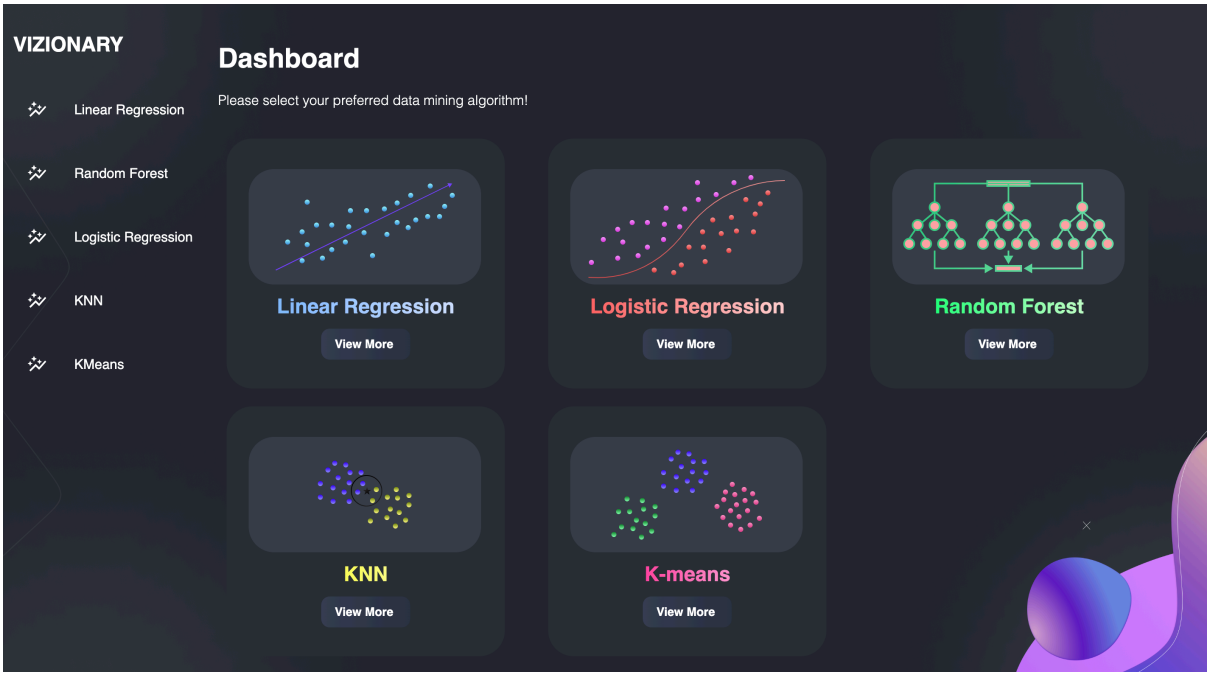
[58] D. Delen, Real-world data mining : applied business analytics and decision making, Upper Saddle River, New Jersey: Pearson Education Ltd, 2015.

[59] A. Rodriguez, "RESTful Web services: The basics," IBM, 2008.

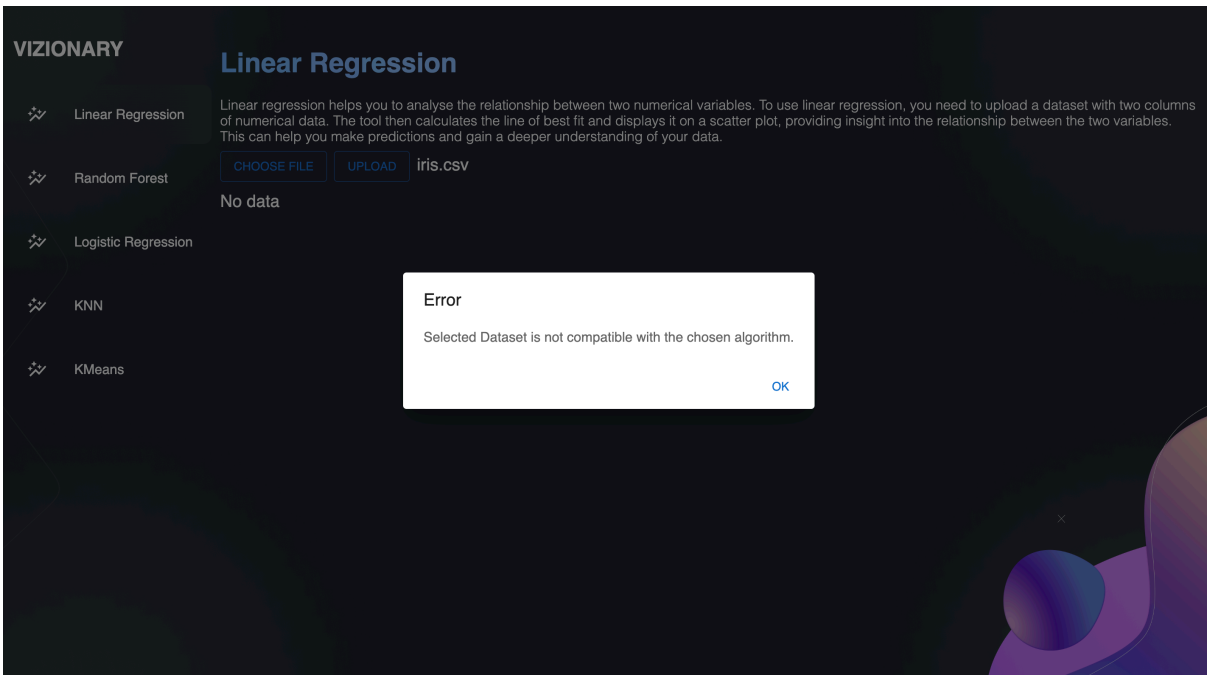
Appendix

The following appendix has been included for completeness in terms of displaying screenshots from the final software solution.

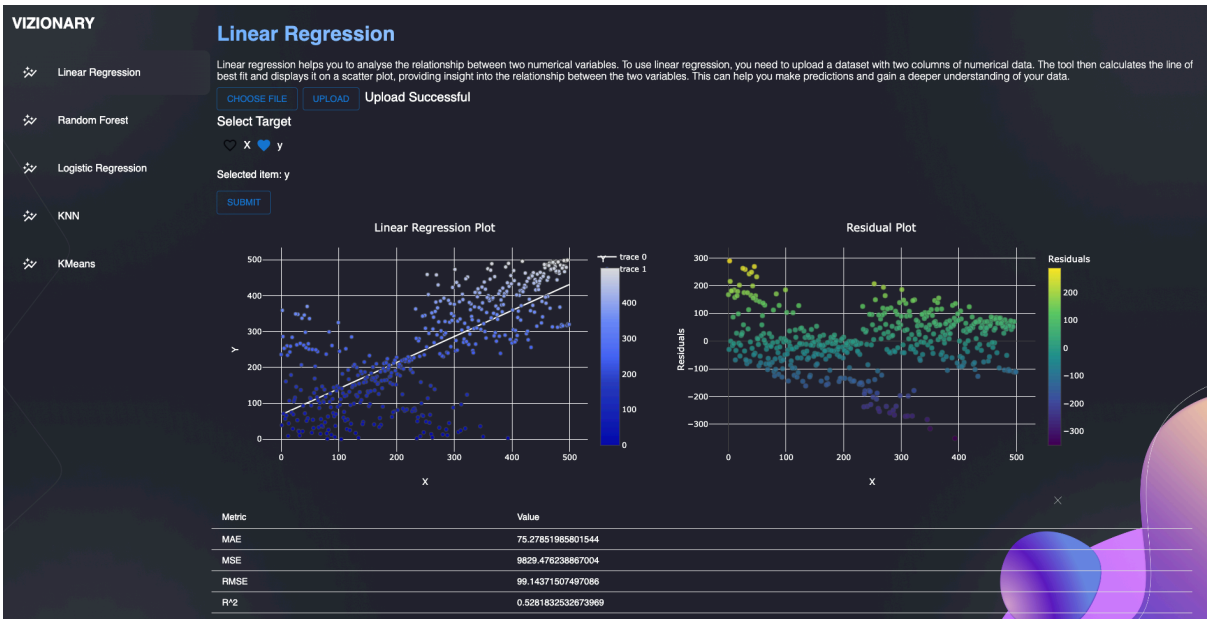
Final Dashboard:



Incorrect Algorithm Selection:



Linear Regression Analysis:



Random Forest Analysis:

Logistic Regression Analysis:

K-Nearest Neighbours Analysis:

K-Means Clustering Analysis: