An Interactive Visualisation of Credit Card Fraud Detection

Submitted September 2023, in partial fulfillment of
the conditions for the award of the degree **MSc Data Science.**

Abigail Kinnaird

Supervised by Professor Robert S Laramee

School of Computer Science
University of Nottingham

I hereby declare that this dissertation is all my own work, except as indicated in the text:

Signature: Abigail Kinnaird

Date 8 / 09 / 2023

Word count: 13,557

Abstract

This project proposes an interactive dashboard as the solution to understanding a machine learning model for credit card fraud detection. The model trained for this project is a Random Forest model, which is a popular 'Black box' model. The dashboard aims to explain with imagery how the model makes its classifications, and therefore remove this idea of it being a 'Black box' model. This dashboard offers a novel approach as there is a currently a lack of research into the overlap of visualisation and credit card fraud detection. With this dashboard fraud experts will be able to fine tune models, the details of which they may not have previously understood. This allows for effective and safer models of detection.

Acknowledgements

I would like to thank Professor Robert Laramee for his advice and knowledge throughout this project, his well structured meetings and answering any of my questions made this project what it is. Thank you to the whole visualisation MSc group and particularly the Wednesday FinVis group for showing their projects each week and listening to updates on mine.
I would also like to thank my family, for their continued support, as always with my whole MSc course. Thank you to them and Saskia Verkaik for proof reading my work.

Contents

Chapter 1

Introduction and Motivation

Credit card fraud is defined as unlawful use or criminal deception for financial gain [32]. This term covers fraud involving both debit and credit cards. The issue of credit card fraud continues to grow and is projected to continue to increase. With £783.8 million [36] lost in the United Kingdom in 2020 alone. It reached an all-time high in 2022, [12] and is predicted to continue to further increase in the coming years. To account for the continuing changes of consumers and fraud perpetrators the systems for fraud detection must be highly adaptable and advanced.

Machine Learning (ML) has been used to fit this problem of detection of credit card fraud. Machine learning is also an area which is growing with new industries adopting its applications all the time. It is a powerful tool, but one that for certain algorithms is challenging to interpret. We can use visualisation to help interpretation, by removing the idea of 'Black box' ML algorithms. 'Black box' is defined by Norbert Wiener, who first introduced the concept to a wider context, as 'a set of concrete systems into which stimuli S impinge and out of which reactions R emerge' [8]. In our context this is the input of fraud data and the output of a classification, of which the steps of the classification are not interpretable to us. This project will focus on interpretability and explainability of a Random Forest which is one of the most used 'Black box' algorithms.

This dissertation proposes the development of an interactive dashboard view. Which describes with imagery the structure of a Random Forest model trained to detect fraud, with the aim to aid interpretability and remove the idea of a 'Black box'. The details of training of the model will also be documented in this dissertation.

1.1 Motivation

An interpretable system of a Random Forest model would allow input from industry experts who previously may not have understood the details of the ML model. These visuals are important for cost saving and safety. A PWC survey found 67% of business leaders 'believe that AI and automation will impact negatively on stakeholder trust levels' [31] while also estimating that AI could make $15 trillion in GDP capital gains by 2030, to keep making these gains we need explainable systems that businesses understand.

1.2 Aims and Objectives

The aim of this dissertation is the development of a novel approach to the visualisation of Random Forest algorithms on credit card fraud detection. This will be achieved by the construction of an interactive Python coded dashboard. The objectives of this project are:

- Produce a dashboard following the approach of 'Overview first, filtering and selection, details on demand' (Shneiderman, 1996) [34] and therefore include interactive features.

- Provide users with multiple connected views explaining the structure of a Random Forest model including details of individual Decision Trees, enabling them to better understand the model.

- Contribute an improvement to other software developed to visualise and explain a Random Forest model on other datasets.

1.3 Challenges

Challenges for credit card fraud detection:

- Very unbalanced datasets [3]. Fraud makes up a very small percent of all transactions. Therefore, the performance metrics need to be chosen carefully.
- Privacy of the data means there are very few datasets and ones that are available are anonymised which means interpretation of the models is much more difficult.
- Speed of the algorithm. Banks need models which allow for quick detection so they can allow customers to quickly freeze their cards to stop further fraudulent transactions [2].

Challenges for visualisation of machine learning algorithms:

- The need for interaction in the visuals to develop human-in-the-loop systems [24].
- There is a lack of guidelines and best practises for machine learning visualisation so makes it very different case by case [9].
- The nature of a black box algorithm is that we do not know easily know how they work.

1.4 Structure of the dissertation

This dissertation follows the structure outlined in Bob's project guidelines [19]. The introduction is followed by a section on background work, which includes a literature review and review of current visualisation systems, similar to the one proposed in this project. Section 3 covers the project specification including an explanation of all the features and the technology choices. Section 4 describes the project plan, which includes how the project took place on a weekly scale. Section 5 goes into detail about the design of the project that also includes documentation of all the code. Section 6 describes the implementation of the system, which includes detail of the basic features and enhancements.

Testing and evaluation are covered in Section 7, in which the case study questions are evaluated. The dissertation then ends with a conclusion, and a future works section. References and source code can also be found at the end of the document.

Chapter 2

Background

This section includes a literature review of relevant work on the topics of credit card fraud detection and machine learning visualization. The survey scope of papers and the search methodology are also included as part of this literature review. The literature review is then followed by a section on existing systems.

2.1 Literature Review

A systematic literature review will be covered in this section. With the survey scope and search methodology covered first, explaining which papers were chosen and how they were found. The review is split into the two topics of Machine learning for credit card fraud and visualisation of Machine Learning.

2.1.1 Survey Scope

This project covers the intersection of two broader topics. The first being the detection of credit card fraud using machine learning. More specifically, Random Forest (RF). The other topic is visualisation of machine learning algorithms. Both of these are large topics, so the niche of a focus on visualisation of Random Forests was chosen, along with a focus on the credit card fraud dataset from the collaboration of Machine Learning Group, Université Libre de Bruxelles, Belgium and Worldline in 2014 [3]. There will not be a focus on the explainability of any other algorithms other than RF and, by proxy, therefore

Decision Trees (DT).

The focus of the credit card fraud papers is only from 2019 onward, apart from the papers published with the dataset around 2013. As fraud has changed so much in recent years, a focus is placed on the most recent research. There is no limit placed on the time of publication for machine learning visuals as this is a more modern area.

2.1.2 Search methodology

For papers on credit card fraud detection, the search began from the source of the dataset. The dataset is listed on Kaggle [37], with citations of papers published with the dataset. These papers were the basis for the search. The choice was made to focus on a Random Forest as the Machine learning model, due to previous experience with this model, along with evidence discussed in the later section on model choice. The 'cited by' feature on Google Scholar was then used to find more specific papers from 2019 onwards were chosen. Specific papers on these were also then found using the 'cited by' feature, including approaches to dealing wih the imbalanced nature of credit card fraud datasets.

For papers on visualisation of machine learning, survey papers were used initially to understand the current technologies. The most relevant and cited papers related to the construction of good visuals for machine learning were then reviewed, using Google Scholar and survey papers to find them. Specific examples for Random Forests were then identified using the browser developed by A. Chatzimparmpas et al [1].

2.1.3 Classification of Literature

To explain how this project fits with the existing literature, see Figure 2.1. This visually shows the overlap of the discussed topics.

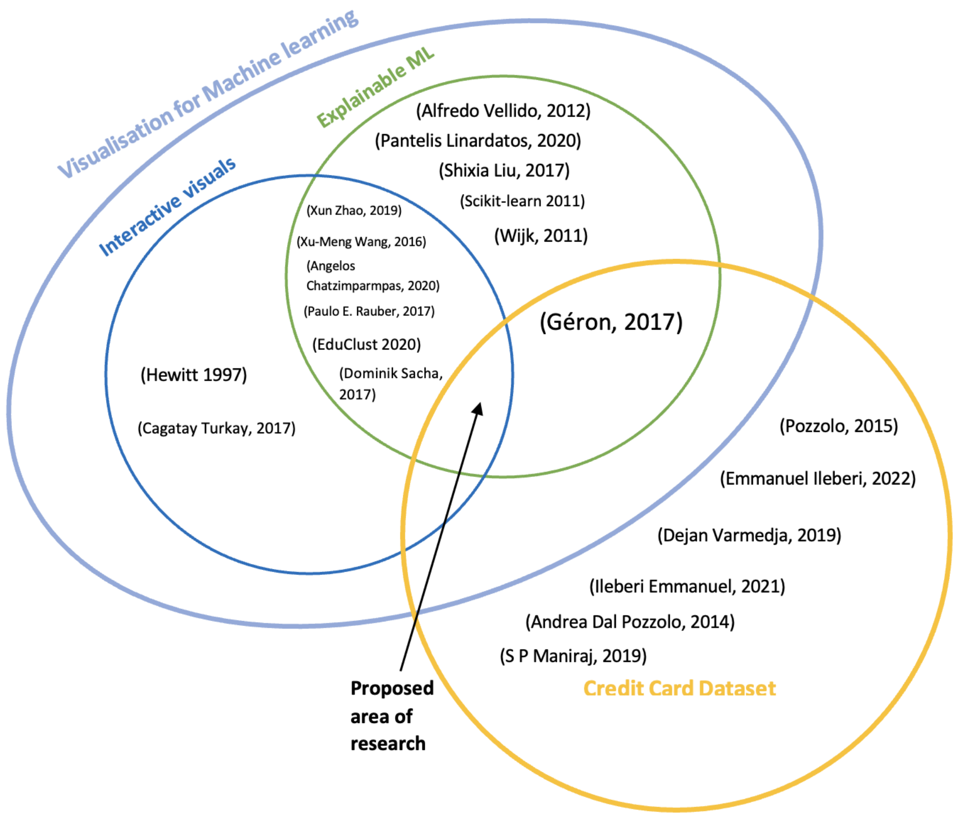


Figure 2.1: A Venn diagram created to show the topics discussed in the literature review and where this project fits.

2.1.4 Literature on Machine Learning for Fraud Detection

European bank dataset

The collaboration of Machine Learning Group, Université Libre de Bruxelles, Belgium and Worldline in 2013 kick-started a huge amount of new research [20]; in 2021 they published a handbook with 10 years' worth of research along with a dataset. This dataset details transactions over 2 days in September 2013 by European cardholders, it contains 284,807 points [37]. The variables are all from PCA and therefore numerical, they are not labelled (due to confidentiality) apart from the time and amount variables. The dataset classifies fraud and not fraud for each data point, with fraud occurring only 492 times. This dataset has been used by many as it serves as one of the only publicly available datasets on credit card fraud.

Approaches to the key challenges

Andrea Dal Pozzolo [30] proposed how to approach the problem of credit card fraud detection and aimed to address the key challenges: imbalanced dataset, concept drift and selection of performance metrics. It is proposed there are broadly two ways to approach the problem of credit card fraud detection, through a data driven approach or expert driven. A machine learning approach is a data driven approach and is the one that seems most successful, however what if we could develop a visual of a data driven approach? This would allow the input of a fraud expert who may not understand machine learning techniques, to enhance our data driven approach by applying expert driven features. Therefore giving us a model that is a combination of the two techniques.

Furthermore, for addressing the challenge of the imbalanced dataset, it is argued that under sampling is not always the best approach and it depends on the machine learning algorithm. Therefore, different balancing methods should also be investigated. The idea of a concept drift is also introduced, which is described as when we get new data that invalidates our current model. In this context it is a new technique for conducting fraud, which would then be undetected. We therefore need an algorithm that can still work but forget certain old behaviours and develop new ones based on new data.

Andrea Dal Pozzolo et al. [30] in a different publication, investigated and summarised the key challenges. They proposed a pre-processing method called Easy Ensemble to make the dataset balanced. The performance of the machine learning algorithms must also be evaluated differently as a score of over 90 can be achieved easily by over predicting not fraud. Some better metrics rather than just accuracy is proposed including F-measure, precision score and total detection cost.

Emmanuel Ileberi et al. [11] took the approach of using SMOTE to solve the issue of the imbalanced dataset. They tested this on lots of different ML algorithms including Random Forests which is most relevant to this project. For evaluation they chose Area Under the Curve (AUC) and Mathews Correlation Coefficient (MCC). AdaBoosting was also used to improve the scores. A Random Forest model with AdaBoositng produced the highest score of 99.97% accuracy and an AUC value of 1.

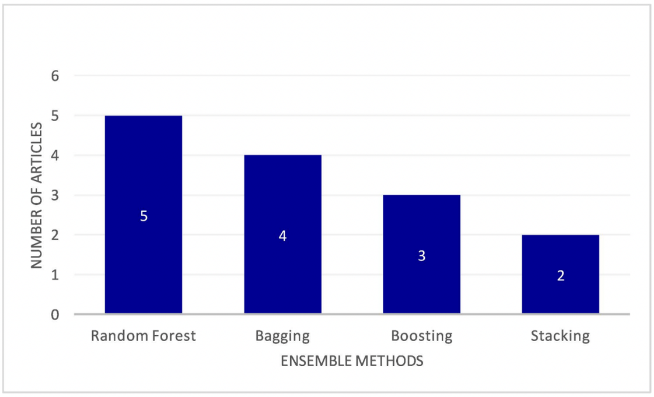


Figure 2.2: Count plot showing number of occurrences of common ensemble methods for detection of credit card fraud, from the 2021 literature review by Ashtiani2022 et al [5]

Model choice

Ashtiania and Raahemi [5] conducted a systematic literature view of intelligent fraud detection models used on corporate financial statements. The review summarised 47 articles. They found that Random Forest was the most popular choice for an ensemble method, see Figure 2.2. Decision Tree methods are the second most popular classification method after Support Vector Machine. This shows that there is a use of these techniques on real world data and therefore a need for good visualisations of them. More specifically the choice of a Random Forest and Decision Trees as the models to make interpretable with a dashboard would be one that would be beneficial.

Looking at papers on comparison of ML algorithms. Dejan Varmedja et al [10] compared the performance of Logistic Regression, Random Forest, Naïve Bayes and Multilayer Perceptron. They chose SMOTE to deal with the imbalance. They used Accuracy, Precision and Recall to evaluate performance. Random Forest was shown to have the best overall score. This is evidence that a Random Forest is a well performing model to focus the visualisation on.

Emmanuel Ileberi et al. [11] compared the performance of Decision Tree, Random Forest, Logistic regression, Artificial Neural Network and Naïve Bayes. A Genetic Algorithm (GA) was also implemented for feature selection. This is a model that is inspired by evolution which helps to solve the problem of high feature dimension space. It is again shown RF to be most effective with an accuracy score 99.98%. ANN also showed good results particularly on the synthetic data.

Understanding Random Forests

Géron 2017 [16] provides a detailed explaination of what a Random Forest model is and how to implement one, these details can then be applied to a model for fraud detection. It describes a Random forest as, an ensemble method that combines lots of Decision Trees classifiers. It therefore usually an improvement on a Decision Tree classifier. It works by averaging out predictions of DT classifiers trained on random variables i.e., it chooses the most popular result. The decision trees chosen are uncorrelated and the bagging method is used for construction of RF. Bagging which comes from bootstrap aggregating is the process of sampling with replacement, it is used to get lots of subsets for training. A Random Forest model being a ensemble of Decision Trees means, that the imagery of Decision Trees also needs to be considered.

2.1.5 Literature on visualisation of Machine Learning

To get an idea of the current state of visualisation Chatzimparmpas et al
[4] survey of surveys paper provided a summary. It also highlighted areas for potential research. See Figure 2.3 which is a table showing research opportunities. Online training processes and Enhancing trust are highlight as two big areas for research. To enhance trust we need more explainable systems and to have interactivity to allow the user to be more sure of the system as well as metrics that explain how the classifier works.
A distinction that kept coming up in the literature and important definitions to understand for this project where explainability vs interpretability. This distinction is mentioned in Chatzimparmpas et al [4], with interpretability being described as how the internals of a ML algorithm works. In contrast explainability is about why certain decision were made. In our context interpretability will be how the ML algorithm classifies fraud and explainability will be the input of fraud experts to tell the model why something is fraud.

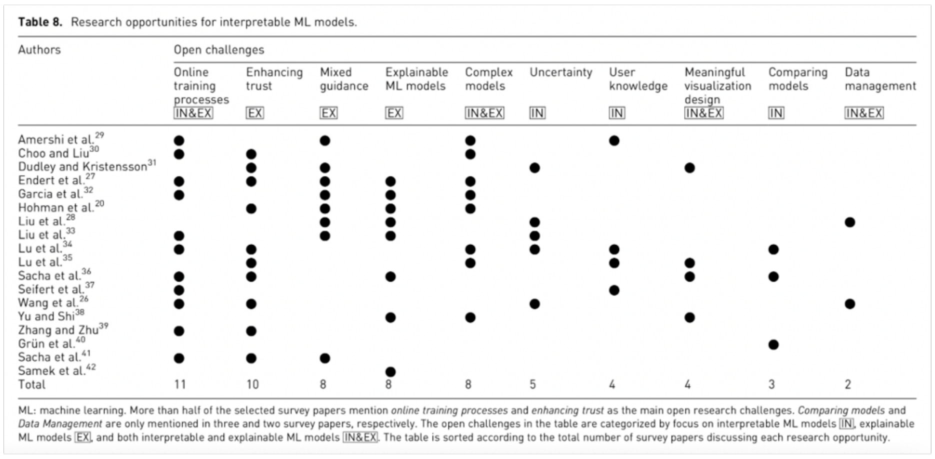


Table 8. Research opportunities for interpretable ML models.

Authors	Open challenges									
	Online training processes IN&EX	Enhancing trust EX	Mixed guidance EX	Explainable ML models EX	Complex models IN&EX	Uncertainty IN	User knowledge IN	Meaningful visualization design IN&EX	Comparing models IN	Data management IN&EX
Amershi et al.[29]	●		●		●		●			
Choo and Liu[30]	●	●			●					
Dudley and Kristensson[31]		●	●			●		●		
Endert et al.[27]	●	●	●	●	●					
Garcia et al.[32]	●		●	●	●					
Hohman et al.[20]		●	●	●	●					
Liu et al.[28]			●	●		●				●
Liu et al.[33]	●		●	●		●				
Lu et al.[34]	●	●			●	●	●		●	
Lu et al.[35]		●			●		●	●		
Sacha et al.[36]	●	●		●				●	●	
Seifert et al.[37]	●						●			
Wang et al.[26]	●	●				●				●
Yu and Shi[38]				●	●			●		
Zhang and Zhu[39]	●	●								
Grün et al.[40]									●	
Sacha et al.[41]	●	●	●							
Samek et al.[42]				●						
Total	11	10	8	8	8	5	4	4	3	2

ML: machine learning. More than half of the selected survey papers mention *online training processes* and *enhancing trust* as the main open research challenges. *Comparing models* and *Data Management* are only mentioned in three and two survey papers, respectively. The open challenges in the table are categorized by focus on interpretable ML models IN, explainable ML models EX, and both interpretable and explainable ML models IN&EX. The table is sorted according to the total number of survey papers discussing each research opportunity.

Figure 2.3: A table showing areas covered and research opportunities from the Angelos Chatzimparmpas, 2020 survey [4]

Types of visualisation for Machine Learning

There are many ways in which machine learning has been visualised in the past, as part of the survey paper, Chatzimparmpas et al [1] produced TrustViz a browser which showcases 420 different techniques for Machine Learning visualisation, a link to this site can be found here: https://trustmlvis.lnu.se/ See figure 2.4 for a screenshot of the homepage. Some of the visuals highlighted in this browser are discussed later in the review.

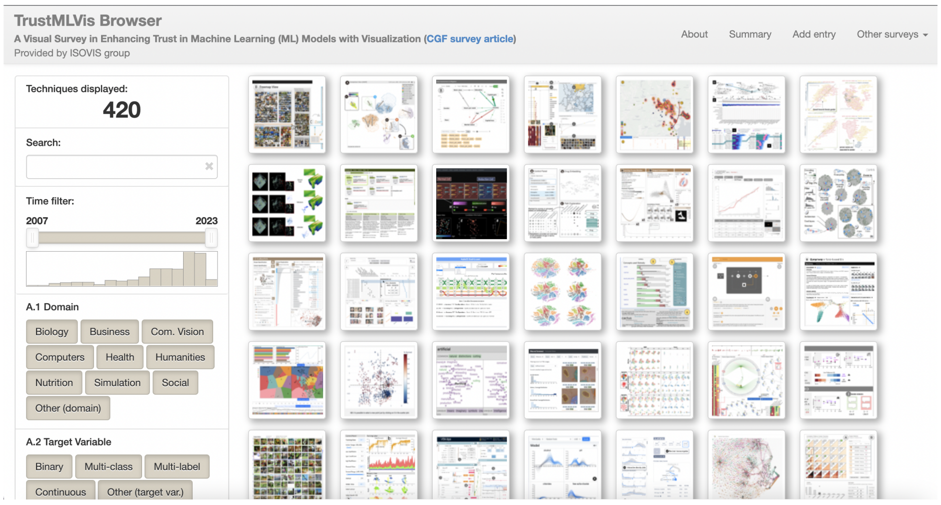


Figure 2.4: A screenshot of the trustvis browser [1]

A key challenge to producing explainable and interpretable ML models, is to explain the link between the model and the data, to see how the model actually works in the context. Having effect visuals of the data itself is therefore important. Hewitt et al [39] reviewed techniques for visualising multidimensional data. Important examples covered are the use of Glyphs first introduced by Edgar Anderson [38] which adds dimensions by the use of radius and lengths of rays of the glyph. A problem with glyphs is there a set limit to how many extra dimensions past 2D can be added. Data used in sophisticated Random Forest models will usually have many dimensions. Another technique reviewed is a parallel coordinate plot. This shows each extra axis next to each other with all the points linked up over the whole plot, Figure 2.5 shows how the structure is built up. This plot was evaluated with concerns about how cluttered it can look being mentioned. A potential solution to this is the addition of interactivity to only show certain selected points.

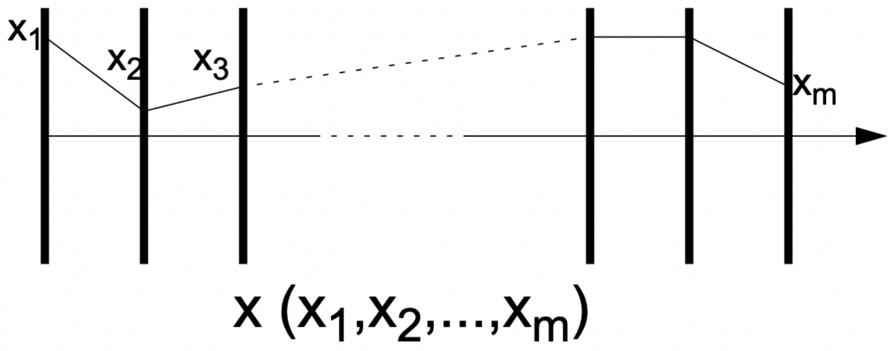


Figure 2.5: Structure of a parallel coordinate plot as shown in Hewitt et al [39]

Hierarchical visualisation will play an important part in this project, as we try to understand the structure of the Decision Trees that make the Random Forest. Schulz et al [17] surveyed different hierarchy structures, included in this was a gallery of implicit hierarchy visualisation techniques, see Figure 2.6 for a screenshot of the gallery. This showed a few examples of treemaps and sunburst plots. These could both be effective visuals in showing a Decision Tree, they are very compact and can include different mapping options to encode all the information from a Decision tree.

Clustering is an important technique that can be used to help make good visuals for machine learning. EduClust [13] is an online education platform for teaching clustering algorithms. It is an invaluable source in education of how these clustering algorithms work. It has animations that easily show how a variety of algorithms work. It also allows

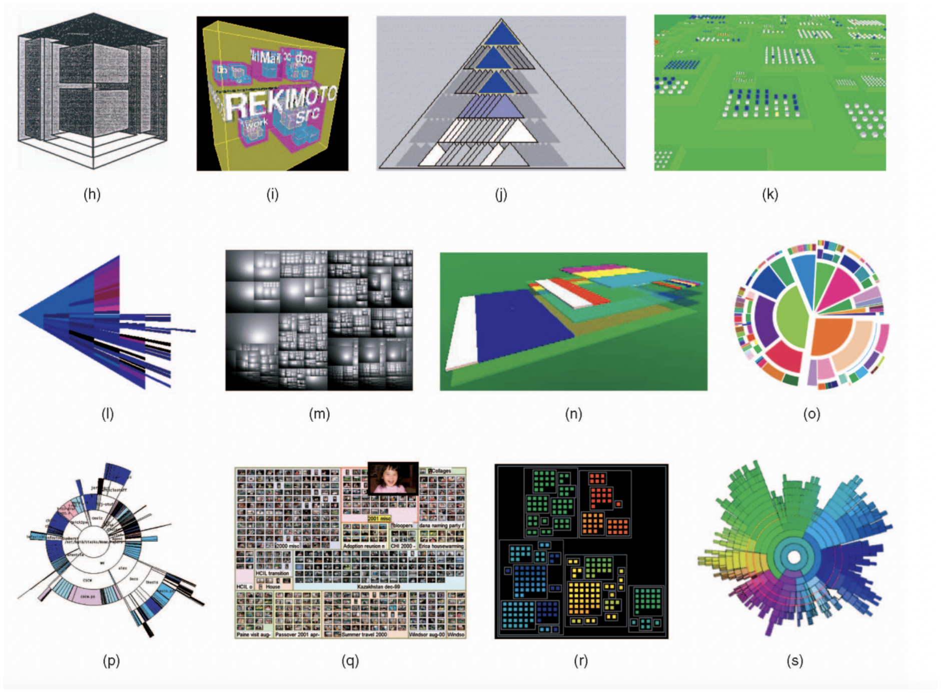


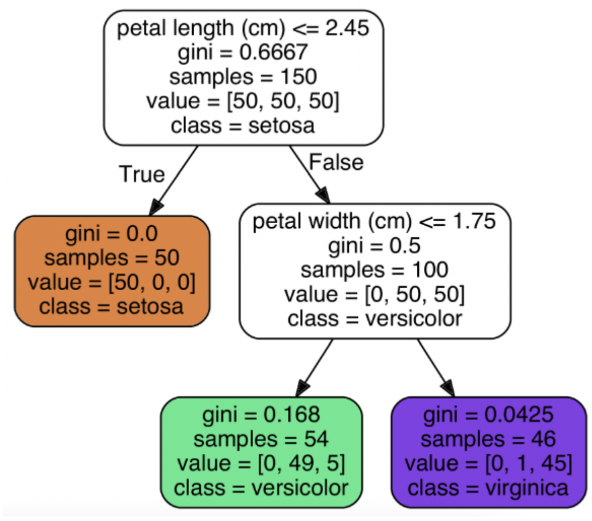
Figure 2.6: Screenshot of part of the gallery of hierarchy plots presented in Schulz et al [17]

upload of your own data. This site enables us to learn these algorithms to then apply in this project.

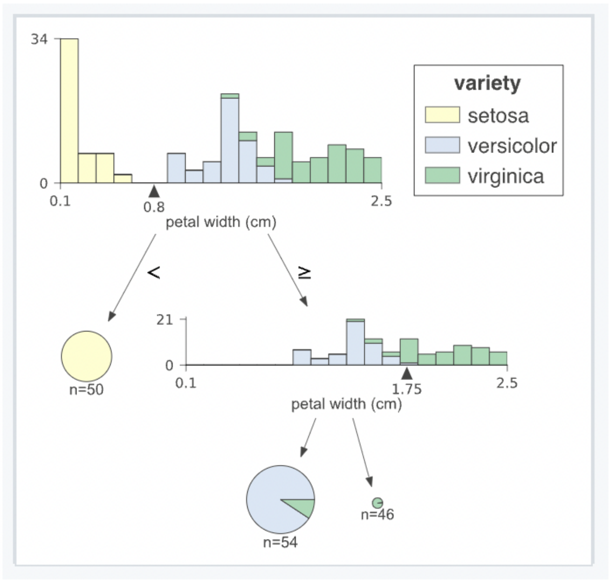
Techniques and tools

There are many pre built tools for construction of visuals of Decision Trees. Which make up a Random Forest. In the scikit-learn library [16] there is a built in Decision tree visualisation tool. This produces a tree with all the basic information that is contained within it after training. This includes the hierarchy order and each node has information about classification, threshold for splitting and number of samples. Figure 2.7 is an example of the scikit graphviz visual. There is some issues with this visual, it is not very compact. When a maximum depth of more than 3 is chosen, it starts to get hard to read as the plot gets too wide with all the child nodes.
Another Decision tree visualisation package is dtreeviz [35]. This package encodes more information than the scikit visual. It includes a histogram of the data which is being split, showing where the split occurs and the classification of the values in the histogram.

See Figure 2.7 for an example. This package still has the issue of not being very scalable, when we get to large maximum depths it will get very hard to interpret the information. In our visuals we therefore need to try an encode all the information that is present in these visual, while making it scalable for large maximum depths.



(a) Scikit graphviz tool [28]



(b) dtreeviz package [35]

Figure 2.7: Examples of packages for Decision Tree visualisation

Construction of the visuals

When creating visual analytics such as explainable and interpretable machine learning visuals there are some fundamentals to consider. Schneiderman's mantra tells us 'Overview first, filtering and selection, details on demand' [34]. This is further explored in the survey by Xu-Meng et al [42] which proposed four important features of any pipeline for the construction of visual analytics: Enabling of Induction and deduction, knowledge externalisation, data provenance and uncertainty-aware knowledge. This survey also explains many different pipelines, the one which most fits with our problem is the knowledge generation pipeline. This is about having visuals that link to a model and data which allow a human to get insights, form hypothesises and therefore knowledge from the visuals in a continuous process.

Looking more at what would be required for a good visual of an ML algorithm, we must not only create visuals for interpretability and explainability of a ML algorithm but also the performance of ML algorithms. Liu et al [21] proposed that the problem should been broken down into three steps: Understanding, Diagnosis, and Refinement. Understanding includes visuals that explain how the algorithm works, Diagnosis covers the evaluation of the performance of the model as well as selection of which algorithm performs the best, and Refinement covers allowing experts to fine-tune a model by seeing the specifics.

Current state

Turkay et al [9] identified areas that need improvement in the creation of ML visuals, noting that there is a lack of guidelines and practices for machine learning visuals. The paper describes the importance in having interaction in visuals to develop human-in-the-loop systems. This is important to consider adding in the development of new models. However they also note that there is a balance to the amount the computer does and how much the human does. The user should be able to use the system to understand the ML model and add input but not be left with computational tasks that there a clear answer to.

An example of an interesting Decision Tree view that has been developed is PyBaobab [41]. Decision Trees will be an important part of any Random Forest visualisation, therefore this package is relevant. BaobabView builds on the common Graphviz package and is easily installed in Python. It works well by being scalable for larger decision trees, it also uses colour effectively and allows for interactions. The design was inspired by the Baobab tree and their real tree like structure makes it an easy connection to understanding a Decision Tree. They proposed this package to be used by domain experts, to help them fine tune their systems. See Figure 2.8 for an example.

Another interesting Decision tree view is TimberTrek [44]. The main aim of it is the development of Decision Trees that align with human knowledge. This links to our aim of trying to make a dashboard that is easy to have industry expert input. TimberTrek offers lots of small views including an overview of all the trees, single tree paths and a search option. Figure 2.9 shows a screenshot of the system. It makes it easy to compare different trees to understand how they work and if they are classifying in an effective and

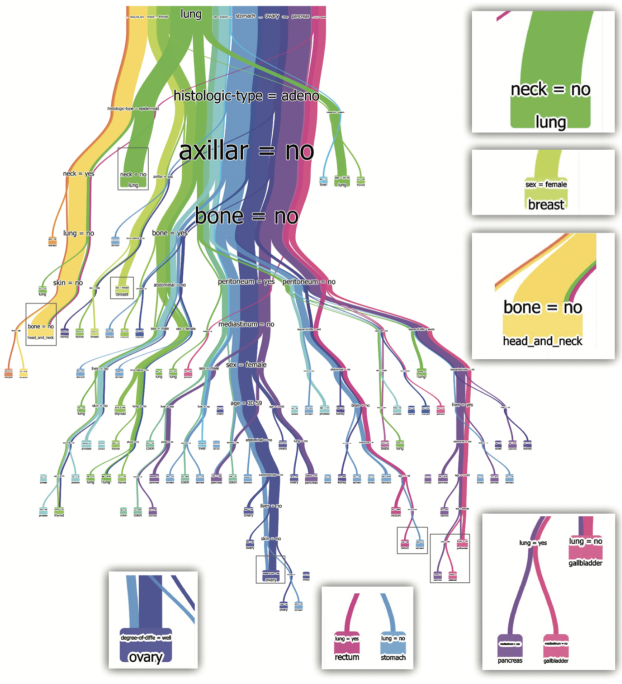


Figure 2.8: Screenshot from the Baobab paper [41] showing an example of a PyBaobab Tree. Which comes from there library for Decision Tree visualisation.

ethical way.

2.2 Previous Systems

In this section three different existing dashboards for Random Forest Visualisation are covered.

iForest (Link to iForest paper) iForest is a visual analytics tool developed specifically for

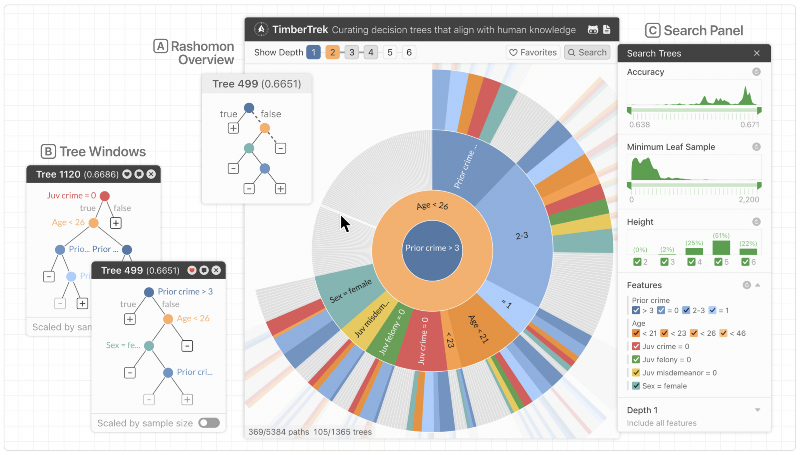


Figure 2.9: Screenshot from Timber Trek [44] showing there system for visualisation of Decision Trees

a Random Forest [43]. It is developed using scikit-learn for machine learning and Flask for the implementation of the visual. The dashboard summarises how a multi-classification model works, see Figure 2.10. This dashboard view has the following features: Data Overview, a Feature View, and a Decision Path View.

Data overview helps the user understand what data the ML model is being applied to, this includes multi dimensional visualisations. They also include a table to explore the actual values. The feature view is used to explain how the classifications are made. Finally the decision path is provided to help the user understand how the decisions are made and compare different decision trees. They devised an algorithm to compare decision path distance to allow for comparison of similar paths.

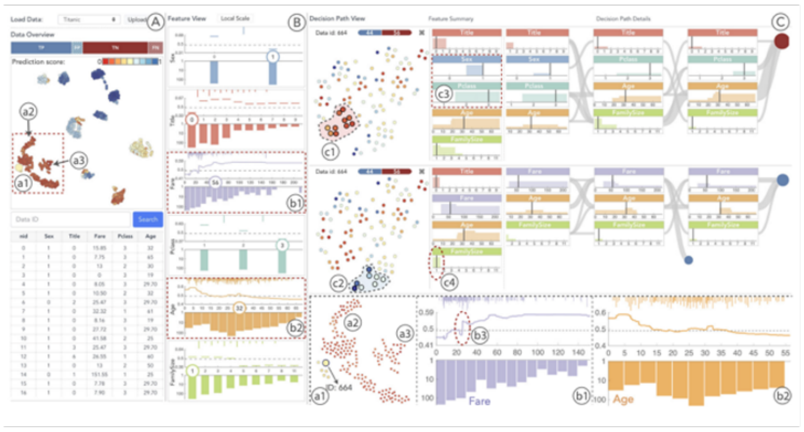


Figure 2.10: A screenshot of the iForest dashboard [43]

Rfx (Link to Rfx paper)

Another tool Rfx [18] has been developed to explain RF in the context of detection of faulty electrical vehicle parts. This dashboard view includes the individual tree view upon selection, an icicle plot to summarise the trees, a cluster view of similar trees and a histogram of the class distribution for a selected DT. See Figure 2.11 the tree structure visual is a more typical DT structure with additional confusion matrices at the nodes which works well for quick evaluation. Both of these dashboard views used clustering of similar DTs, this approach will be considered for the ML fraud visuals.

SYLVIA (Link to SYLVIA paper) [25] dashboard includes Icicle plots, a feature importance bar chart, and a scatterplot of Prediction vs Residual. The Icicle plot for summary

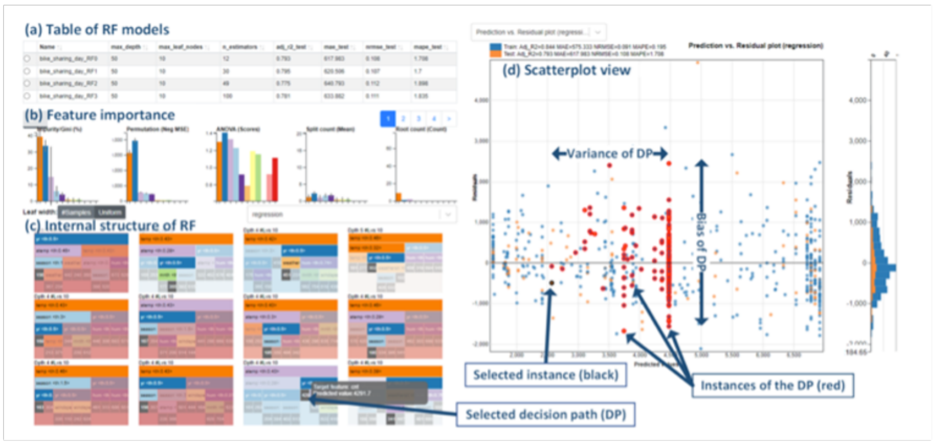

(a) Table of RF models
(b) Feature importance
(c) Internal structure of RF
(d) Scatterplot view
Variance of DP
Bias of DP
Selected instance (black)
Instances of the DP (red)
Selected decision path (DP)

Figure 2.11: A screenshot of the Rfx dashboard [18]

of the model is less easy to understand this may be due to the loss of the tree like structure. The inclusion of feature importance is a very helpful graphic, as it allows the user to quickly identify what features are most important for classification. See Figure 2.12

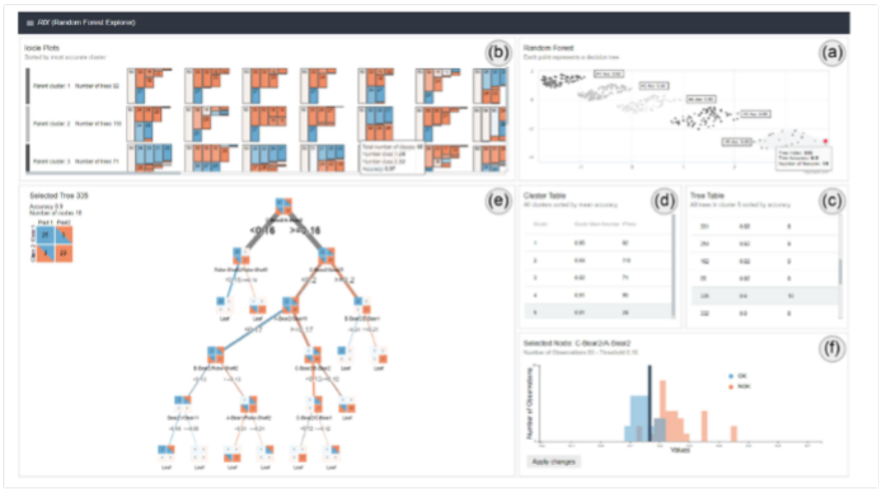


Figure 2.12: A screenshot of the SYLVIA dashboard [25]

Chapter 3

Data Characteristics

The dataset of interest is presented as a CSV file of size 150.8 MB, it covers all transactions over 2 days in September 2013 by a set of European cardholders, it contains 284,807 transactions. This dataset was published through a collaboration of Worldline and Univsersité Libre de Bruxelles [20]. A link to the dataset is provide here: kaggle.comThe variables are all from PCA, making them all numerical, they are all not labelled due to confidentiality concerns except from the 'Time' and 'Amount' variables. There is 31 variables in total, therefore the structure of the dataset is 31 volumns and 284,807 rows. Figure 3.1 shows a histogram plot of the 'Time' variable, to give an idea of the spread of the data..

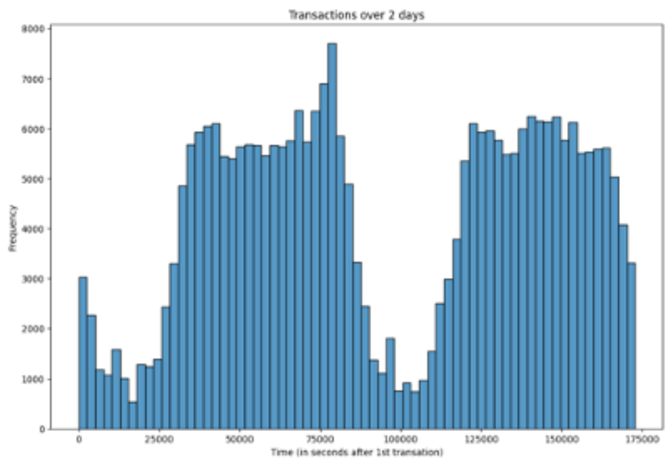


Figure 3.1: A histogram of the 'Time' variable (produced using matplotlib) for the whole credit card fraud dataset. The time is measure in seconds after the first transactions. The data ranges over 2 days.

The dataset classifies each data point as either fraud or not fraud, with fraud occurring only 492 times. This dataset has been used by many as it serves as one of the only publicly available datasets on credit card fraud. It was published to help research of machine learning for credit card fraud.

Chapter 4

Project Specification

This chapter covers the feature specification which includes basic features and enhancements followed by technology choices.

4.1 Feature Specification

This section outlines stage one of software development as outlined in Bob's project guidelines [19] which covers the generation of a requirements specification. The primary aim of this project is the development of a dashboard which helps with interpretability and explainability of the processes of a specific Random Forest model, trained on credit card fraud data. The construction of the dashboard will follow the mantra by Shneiderman, Ben 'Overview first, Zoom and filter, details on demand' [34].

4.1.1 Basic Features

The dashboard should include the following must-have features to be able to meet the aims:

1. **Fitting of a Random Forest Model** A model of good accuracy should be fitted to the data to then be visualised.

2. **Parallel Coordinate Plot** A parallel coordinate plot of all the transaction data should be displayed with different colouring for Fraud and Not Fraud.

3. **Sunburst Plot** A sunburst plot of an individual Decision Trees.
4. **Slider** A slider to allow the user to select an individual Decision Tree.
5. **Scatter Plot** A scatter plot of all the Decision Trees in the Random Forest, plotted on axis Accuracy vs Dissimilarity.
6. **A Small Multiples View** A plot of all the Decision Tree sunburst plots that make up the Random Forest.
7. **Interaction with Small Multiples Plots** Allow the user to select a Decision Tree not just by the slider but also by clicking one of the small multiple plots.
8. **Interaction of Sunburst and PCP** The ability for the user to select a certain node of a tree on the sunburst plot and see on the PCP plot how the splitting of the data occurs at the selected node.
9. **Showing Selection** The current displayed decision tree should be highlighted in grey to show it is currently selected.
10. **Reset Button** Allow the user to return the sunburst plot to original view (back to showing all the nodes) after selections have been made.

4.1.2 Enhancements

The following features build on the work done by the must-have features or add new features that enhance the visualisation.

1. **Treemap** An additional view that can be toggled instead of the sunburst plot (with same functionality) if the user prefers interpretation of a treemap over a sunburst plot.
2. **PyBaobab View** A second view of the selected tree which shows a more interpretable 'tree' structure but without interaction.
3. **Clustering** On the scatter plot colouring will be added to show k-means clustering.

4. **Ordering of the small multiple plots** An option to allow the user to sort the Decision Tree plots by accuracy or dissimilarity.

5. **Brushing in Scatter plots** Allow the user to select multiple Decision Trees by using a rectangular selection box.

6. **Summary Treemap** A treemap which summarises all the variables of the whole forest, with a mapping to size to help easily identify the most important feature

4.2 Technology Choices

Bob's project guidelines [19] outlines the three main choices of technologies which will be outlined in this section (Programming language, GUI and libraries). Xiaoxia Liu et al.'s [21] survey paper formed the basis of the search for technologies, this built on other technologies also used in work from the literature view. For the type of visuals that could be made Treevis.net [33], Trustmlvis.net [?] and Financevis.net [22] all provided good examples.

4.2.1 Programming Language

The choice of a programming language is an important decision. A language needs to be chosen for both the construction of the ML model as well as for the implementation of visuals, they do not necessarily need to be the same. Python is the most widely used language for ML [23] and looking at the previous work on credit card fraud it is a popular choice, R is another very good language for ML model implementation.
For creation of imagery, Python was a very popular choice [43] along with Java. These languages are both free, cross platform and well documented. However, it was decided that Python would be used, due to the time constraint, and familiarity with this language. Python also works well for both sub-sections so will be easy to integrate.

4.2.2 Graphical User Interface

The use of Python and the choice that the project is to be hosted locally instead of web based helps to narrow down choices of GUI frameworks. There are a few well documented, free to use and easy to set up GUI frameworks that work specifically for dashboards: Flask and Dash are two very popular choices. Flask [27] is the choice for the production of the iForest dashboard [43] an example of an very thorough documentation is shown by this detailed book by Migeul Grinberg [15] this gives an idea of what can be achieved. Another choice is Dash [29] which is made as part of Plotly, a popular Python package for production of interactive and detailed plots; it is a more modern framework. It also has the benefit of simplifying lots of the CSS and HTML so an extensive knowledge is not required.

A comparison of the two frameworks was discussed in [7] which showed benefits and disadvantages to both overall concluding that Dash is potentially easier for beginners to dashboard development but offered less front-end capabilities with Flask offering more but requiring more code to be written for the same outputs. Due to the time constraint on this project and previous knowledge of Plotly, Dash was chosen as the GUI framework.

Chapter 5

Project Plan

This section covers the plan for this project. Each section has a description with the date it was completed. The basic features and enhancements are separated into two tables.

5.0.1 Basic features

	Description	**Completion Date**
1	Implemented a Random Forest model	28-July-2023
2	Parallel Coordinate plot	5-July-2023
3	Hierarchy list function	12-July-2023
4	Treemap of Decision tree	12-July-2023
5	Sunburst of Decision tree	12-July-2023
6	Dissimilarity of trees algorithm created	19-July-2023
7	Scatter plot	19-July-2023
8	Slider	02-Aug-2023
9	Update Sunburst function	02-Aug-2023
10	Update Scatter function	02-Aug-2023
11	Update PCP function	09-Aug-2023
12	Reset button	09-Aug-2023
13	Small multiple view	09-Aug-2023
14	Update Selection function	16-Aug-2023
16	Brushing in scatter plot to show selection	16-Aug-2023
17	Sorting by Accuracy and Dissimilarity added	23-Aug-2023
18	Summary Treemap view	23-Aug-2023
19	Accuracy for each tree	23-Aug-2023

Table 5.1: Project plan for implementation of basic features, providing a description and completion date for each feature

5.0.2 Enhancements

	Description	**Completion Date**
1	PyBaobab View of Decision tree	12-July-2023
2	K-means clustering applied	19-July-2023
3	Brushing in scatter plot to show selection	16-Aug-2023
4	Sorting by Accuracy and Dissimilarity added	23-Aug-2023
5	Summary Treemap view	23-Aug-2023
6	Accuracy for each tree	23-Aug-2023

Table 5.2: Project plan for implementation of the enhancements providing a description and completion date for each feature

Chapter 6

Project Design

This stage covers the software design section of the project, as documented in [19]. The main design process for this project followed Shneiderman's, mantra 'Overview first, Zoom and filter, details on demand' [34]. The initial design of the dashboard was adapted as different features were implemented. The final dashboard was a result of an iterative process of design and evaluation.

For the visualisation of Machine Learning models, the visual analytics knowledge generation model proposed by Wang et al [42] is used. This model shows that the purpose of the imagery for machine learning is to allow the user to draw conclusions, specifically to identify the variable that is best at predicting fraud. It demonstrates that this knowledge generation process involves two human loops of exploration and verification. Therefore, the constructed visual are designed to support these processes.

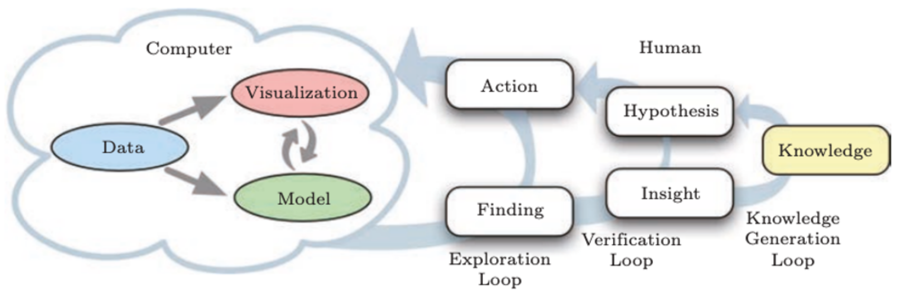


Figure 6.1: Knowledge generation model as proposed in Wang et al [42]

The following subsections go into specific detail of this design. Starting with the process design, followed by the system design, a list of all classes and functions and layout of the dashboard.

6.1 Process Design

Figure 6.2 shows the processes for this project. The four stages of processing from Ware 2004 [40] are listed below with how they are relevant to this project.

- **Collection and storage of data itself** In our case this is the Fraud labelled Credit card transaction data.
- **The pre-processing designed to transform the data into something we can understand** This is the implementation of our Random Forest ML model after pre-processing has been applied.
- **The display hardware and the graphics algorithms that produce an image on the screen** This describes all the plots and imagery that make up the dashboard view.
- **The human perceptual and cognitive system** This describes the interactions and how the user is able to interpret the ML model from the use of imagery and interactions to allow exploration.

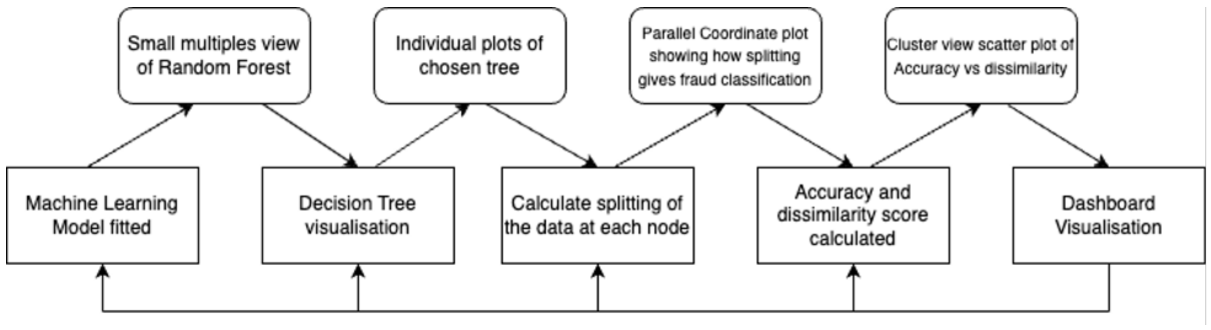

Small multiples view of Random Forest
Individual plots of chosen tree
Parallel Coordinate plot showing how splitting gives fraud classification
Cluster view scatter plot of Accuracy vs dissimilarity
Machine Learning Model fitted
Decision Tree visualisation
Calculate splitting of the data at each node
Accuracy and dissimilarity score calculated
Dashboard Visualisation

Figure 6.2: Process design diagram showing all the process with outputs, the outputs in this project are the visuals. Produced using draw.io.

6.2 Systems Diagram

There are three systems diagrams constructed for this project. The first (Figure 6.3) shows the design for section 1 of the code. Which includes all the pre-processing of the data needed to make any of the plots. Section 2 is the system used to construct the dashboard itself. For a break down of exactly the functions See section 5.3 Classes and Functions. The final diagram shows the interactions all the two sections.

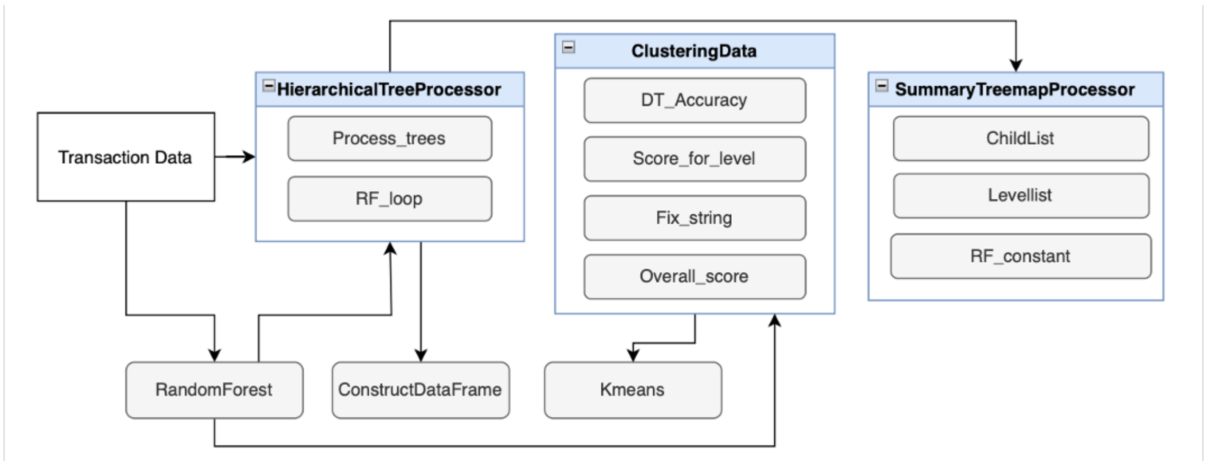


Figure 6.3: Class and function interactions for section 1 of the code (Pre-processing for the visuals), produced using draw.io

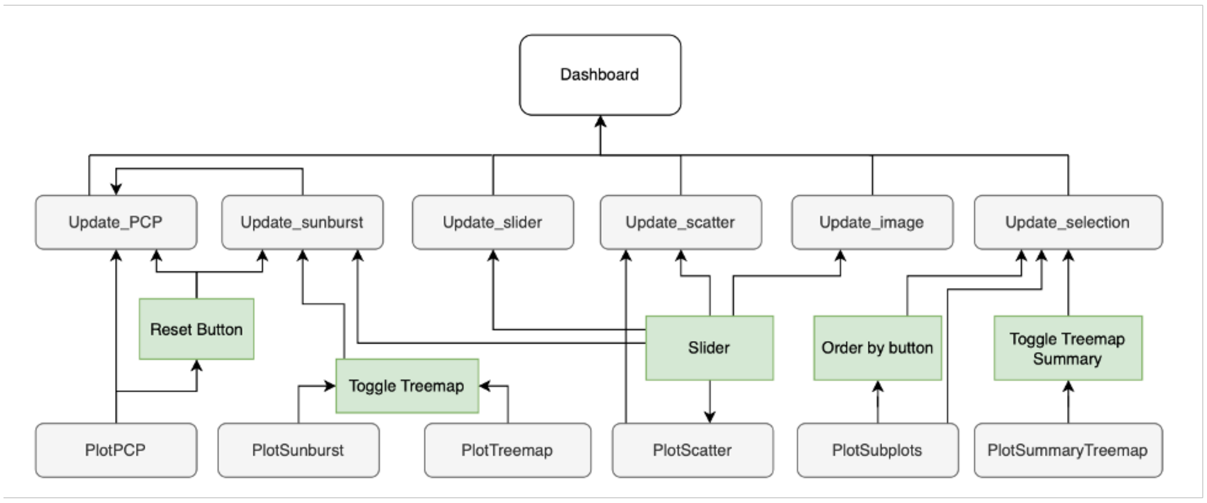


Figure 6.4: Summary of the section 2 functions of the project which includes all the interactions of the Dash app functions. Produced using draw.io

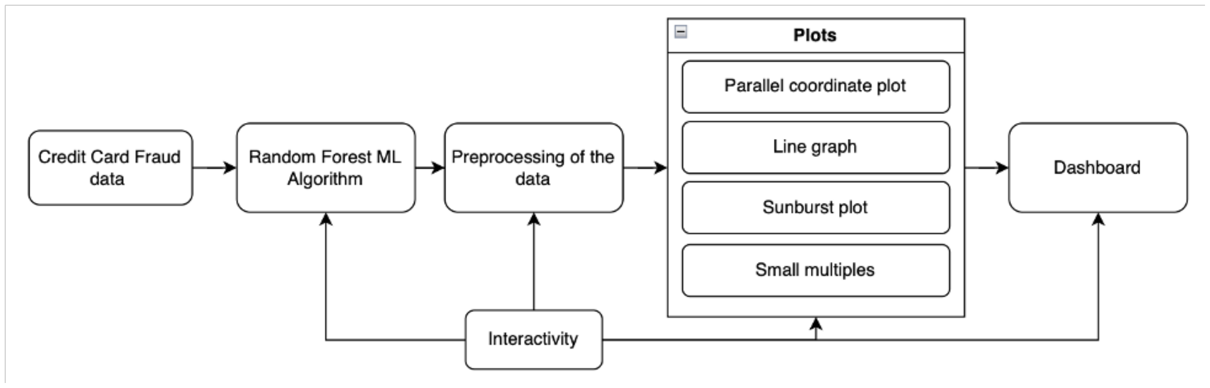


Figure 6.5: Summary systems diagram showing the main components of the system and how they interact. Produced using draw.io

6.3 Classes and functions

The structure of this code is in two sections, firstly the implementation of the Random Forest and all the pre-processing of the data before any imagery can be made. Below is tables for the classes and functions in each section along with summary diagrams of the two sections.

6.3.1 Section 1: Processing of the data

Classes

This shows the structure of the three classes: HierarchicalTreeProcessor Table 6.1, ClusteringData Table 6.2, SummaryTreemapProcessor 6.3 and other functions summarised in Table 6.4.

HierarchicalTreeProcessor:

Function name	**Description**
Process_tree	Extracts the structure of each decision tree at each node by the construction of lists for child, parent, value and threshold
RF_loop	Run the process for all the trees in the Random Forest

Table 6.1: Hierarchical Tree processor class with function description

ClusteringData:

Function name	Description
DT_Accuracy	Calculates the accuracy of each tree by applying it to the test data.
Score_for_level	Creates a summed score of how different the list for each level of the tree is from each other, using the Levenshtein score
Fix_string	Removes the spacing that was needed to make the hierarchy structure
Overall_score	Calculates the overall dissimilarity score by added the weighted scores for each level

Table 6.2: Clustering data class with function descriptions

SummaryTreemapProcessor

Function name	Description
RoundingString	Pre-processing to revert the strings back to having no 0.1 additions
ChildList	Constructs a list of only the unique variables at each level of the tree
LevelList	Creates a list of a string of the level, based on how many unique values there are
RF_Constant	A list of 'Random Forest' string for the construction of the treemap

Table 6.3: Summary Treemap class, with function explanations

Functions

Function name	Description
RandomForest	Implements the Machine learning algorithm
Kmeans	Fits a Kmeans clustering for 3 clusters on the Accuracy vs Dissimilarity data
ConstructDataFrame	Constructs a dataframe that describes the structure of each tree. Includes columns: Children, Parent, Level, Classification, Threshold, Colour_class

Table 6.4: Functions constructed in the first section of the code

6.3.2 Section 2: Visualisation

This section covers all the functions used to construct the dashboard. Table 6.5 documents each function with a description of its use.

Further documentation of the code can be seen in the HTML output of the code which can be found at this link: github.com/HTML

6.4 Dashboard Design

For the design of the dashboard, Figure 6.6 shows the chosen placement of the different plots. This design has been chosen to place plots that interact the most next to each other. The most important plots (PCP and sunburst) have been placed at the top. The two largest plots were placed diagonally to visually balance the plots.

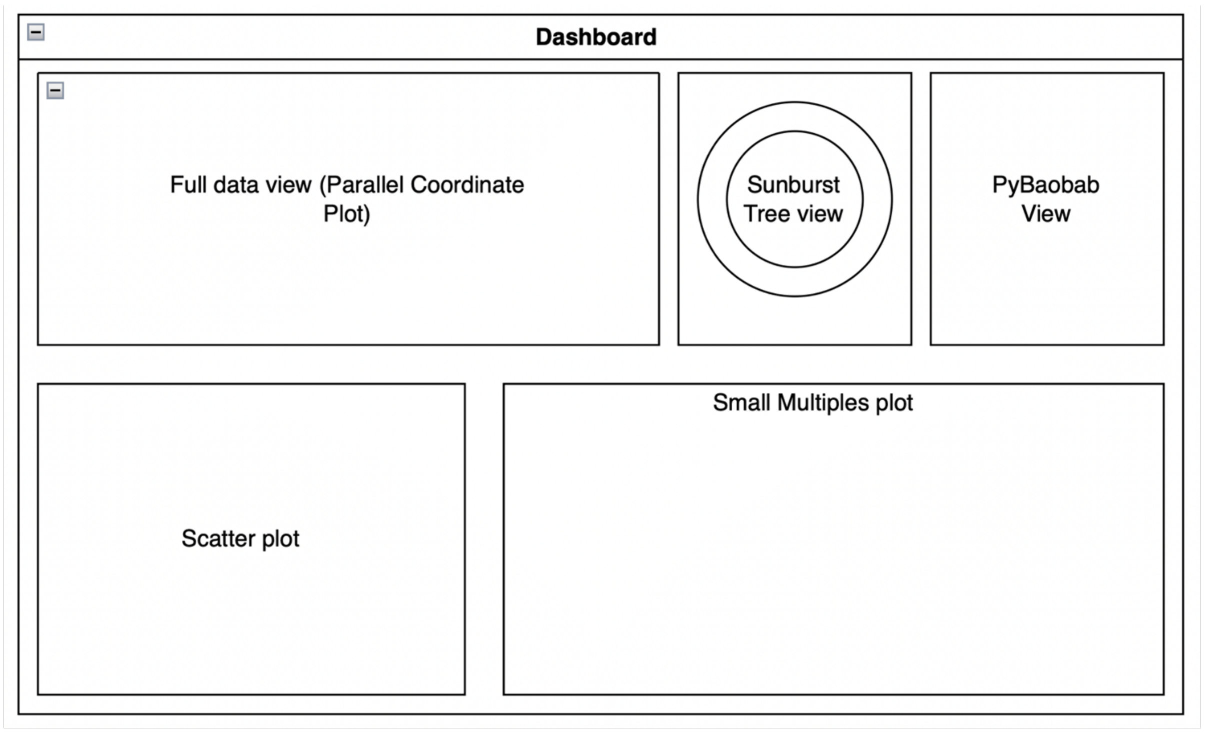


Figure 6.6: A basic outline of the design of the figures in the dashboard. Produced using draw.io

Function name	**Description**
PlotPCP	Produces the Parallel Coordinate plot, with brushing, selection and colours mapping to Fraud and Not Fraud
PlotSunburst	Plots a single sunburst which represents one Decision tree
PlotTreemap	Plots a single treemap which represents one Decision tree
PlotScatter	Plots a scatter plot of Accuracy vs Dissimilarity for all the Decision trees
PlotSubplots	Plots small multiple views
PlotSummaryTreemap	Plots the treemap which only shows unique values at each level, to give an easy overview
Update_slider	Takes in inputs from the small multiples view, updates when the user either moves the slider or clicks a new plot
Update_selection	Takes in inputs from toggle switch 2, order by accuracy button, order by dissimilarity button, slider and scatter. It checks for the triggering of any of these. It then either replots the subplots in a new order if accuracy/dissimilarity buttons are triggered or changes which plot is now a grey colour. A new mapping is used to make sure the correct plot is recoloured when the ordering has occurred.
Update_image	This takes in input from the slider, based on the value it will display a new image of the correct chosen decision tree
Update_scatter	This takes in inputs from the slider and checks for input of the button for ordering accuracy or dissimilarity. It then replots the scatter plot with the new selected tree point coloured in grey
Update_sunburst	Takes in inputs from the accuracy and dissimilarity buttons, the slider, reset button, toggle switch and the small multiples plot. This then updates the current shown Decision tree based on the selection by the user. If the toggle button is pressed it will display a treemap version of the current tree (same data as the sunburst). The reset button takes the user back to the unselected sunburst plot. i.e. all nodes showing.
Update_PCP	Takes in inputs from the sunburst plot, slider, and the reset button. It displays a constrained range for certain variables of the data based on which nodes of the selected decision tree are chosen.

Table 6.5: Functions used in the second section for production of the imagery for the dashboard.

Chapter 7

Implementation

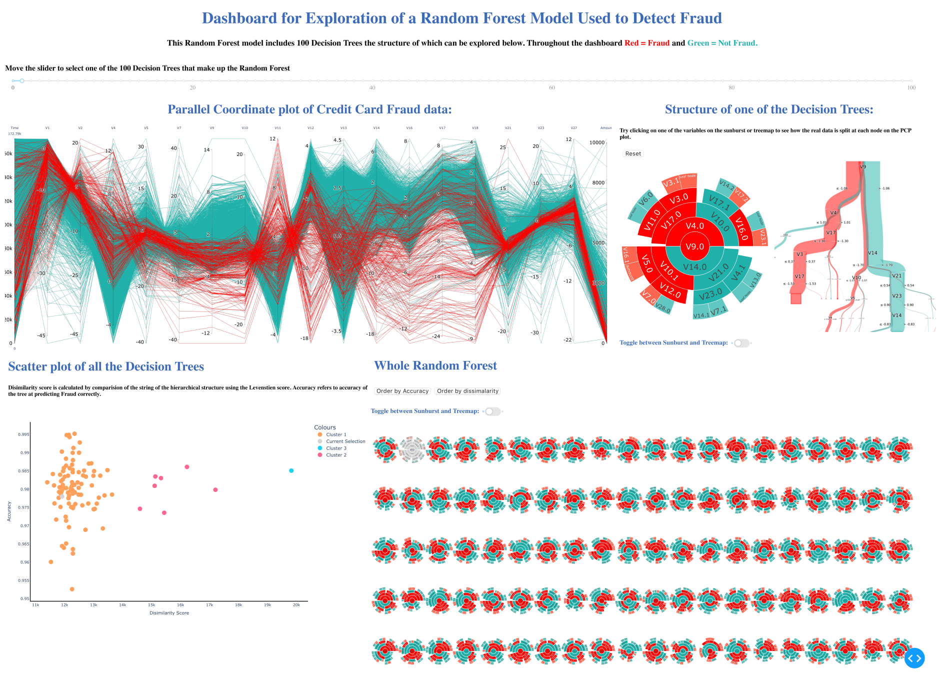
This section covers the implementation of the project, following the structure of Bob's project guidelines [19]. The section is split into basic features and enhancements. Figure 7.1 shows the finished dashboard to provide a sense of where each feature fits as they are discussed, a full scale verions of this can be seen by following this link: github.com/full_image. For each feature an explanation of what it is, how it works, and any difficulties in implementing it are outlined. There are some code snippets along with images of the features and other figures used to explain how these features work. This dashboard is hosted locally and requires only the installation of packages listed in the setup section of the code, which can be seen in the HTML output of the code. This can be seen by following the following link: github.com/HTML

7.1 Basic Features

This section covers all the basic features as outlined in the project specification.

7.1.1 Overall design choices

Firstly, consideration was put into the overall look of the dashboard. The choices in layout and colour scheme have a large impact on the effectiveness of the dashboard. A simple colour scheme was chosen, with black and blue for text. The same sea green and red were used for all the plots. This is very important to visually link all the plots. The red


Dashboard for Exploration of a Random Forest Model Used to Detect Fraud

This Random Forest model includes 100 Decision Trees the structure of which can be explored below. Throughout the dashboard Red = Fraud and Green = Not Fraud.

Move the slider to select one of the 100 Decision Trees that make up the Random Forest

Parallel Coordinate plot of Credit Card Fraud data:

Structure of one of the Decision Trees:

Try clicking on one of the variables on the sunburst or treemap to see how the real data is split at each node on the PCP plot.

Toggle between Sunburst and Treemap:

Scatter plot of all the Decision Trees

Disimilarity score is calculated by comparision of the string of the hierarchical structure using the Levenstien score. Accuracy refers to accuracy of the tree at predicting Fraud correctly.

Whole Random Forest

Toggle between Sunburst and Treemap:

Figure 7.1: The full interactive dashboard view for visualizing a Random Forest model trained to detect credit card fraud.

was chosen as the most logical colour for fraud. With seagreen offering a clear contrast. To avoid over complicating the view toggles were added to see the extra plots. Buttons used a simple style and were placed logically close to relevant plots. The dashboard was kept simple with limited text, to not visually overload. Where possible design choices were made to make the exploration process as intuitive as possible i.e., anything that you might want to select could be chosen. This meant that less explanation is provided on the page thus keeping the visual simple. The placement of the small details (text, buttons) were all carefully considered and tweaked throughout the development.

7.1.2 Random Forest Implementation

A Random Forest model is the combination of many Decision Trees; therefore, to understand how it works we have to look at how a Decision Tree classifies.

A Decision Tree works by splitting the data at different variables, e.g. taking only values for amounts less than £10. The split is decided as such that it gets the best separation of Fraud and Not Fraud in separate groups. This process happens at every node; therefore, by progressively having more splits we can end up with leaf nodes (nodes with no children) that are very pure with only Fraud or Not Fraud values. The new data is then put through these splits and based on the leaf node it reaches it will be assigned a probability that it is Fraud or Not Fraud.

The implementation of this model can be broken down into pre-processing, training of the model and evaluation.

- **Pre-processing**: The data were first imported as a dataframe and the classification was split into a separate column from the variables. The dataset is highly imbalanced and to achieve a good score this needed addressing. There are many techniques to do this. Due to the scale of this project and with the emphasis being more on explainability and interpretability using imagery, rather than achieving the best accuracy, SMOTE was chosen. SMOTE (Synesthetic Minority Oversampling technique) which was first described by N.V Chawla 2002 [26] is one of the most popular techniques to address imbalance. It works by oversampling the minority class, in this case Fraud. See Figure 7.2 for the counts of Fraud and Not Fraud before and after SMOTE is applied.

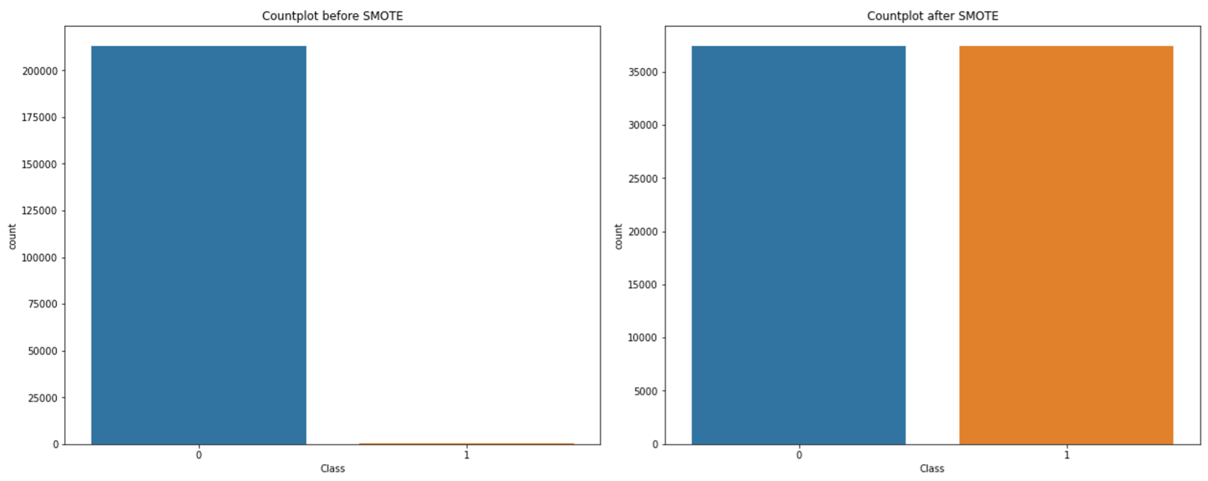


Figure 7.2: Count plot of the number of each Class (Fraud or Not Fraud) for before SMOTE is applied and after SMOTE is applied

- **Training of the model** A 75/25 split of training a test data was applied and with a random state of 99 applied for consistency of development of the dashboard. All features were included in the training. For both the number of estimators (Trees) and the max depth of these trees the optimum values were decided by running many different trails and plotting Accuracy, AUC, and Precision to see which values gave the best results. See figure 7.3

- **Evaluation** A maximum depth of 5 was chosen as this gave a good score for both accuracy and AUC and it is not too large so we avoid overfitting the data. Over fitting is a process in which the model fits too well to the training data that it is too specific to be applied to any unseen data [16]. For estimators 100 was found to be the optimum number, as after this there is very little increase in performance. The use of SMOTE gave an increase in Accuracy from 99.95% to 99.996%

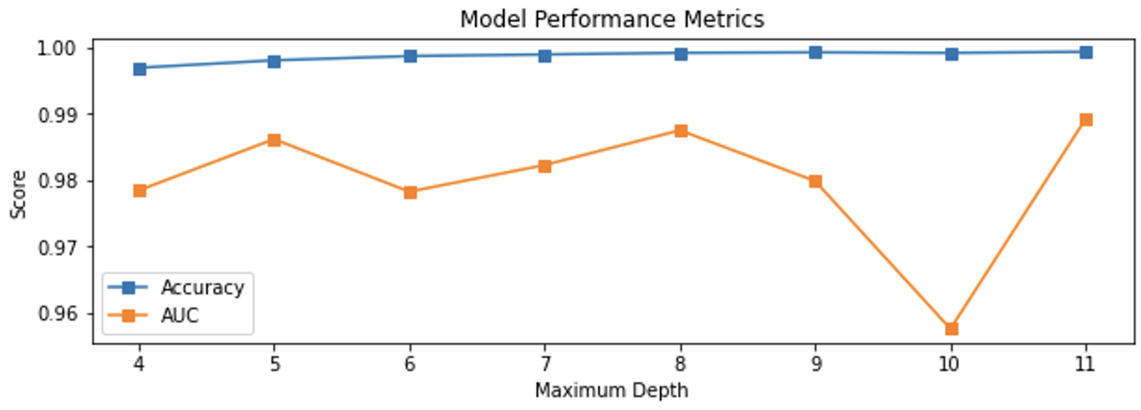
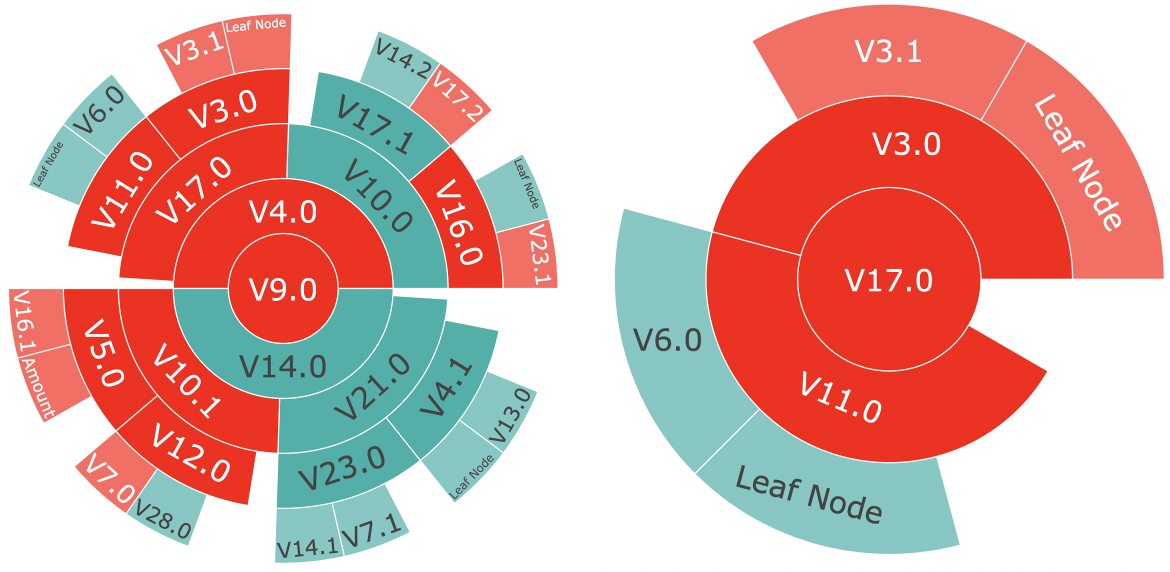


Figure 7.3: Accuracy and AUC values for a Random Forest model with different maximum depths

7.1.3 Sunburst and Treemap plots

A compact hierarchy figure is required to visualise a decision tree. Both a sunburst plot Figure 7.4 and treemap Figure 7.5 were chosen following the options present in [17]. A user option was also added to toggle between them, for personal choice on which the user finds easier to interpret. Interactivity to explore lower level nodes is also added with an animation when the change occurs.

The mapping in the sunburst plot includes colour with red to fraud and lightseagreen to Not Fraud. The radius represents the level, with an increase in radius corresponding to being further down the plot. The size represents the number of child nodes, with a larger segment having more child nodes. The labels in each segment correspond to one of the



(a) Full Sunburst (b) Selected node view

Figure 7.4: Sunburst plot showing the structure of a single Decision Tree. The labels in each section correspond to a variable in the data. Size maps to the number of children, and color represents the most likely classification. (a) Shows the full structure, and (b) shows when the V17.0 Node is selected, giving the zoomed in view.

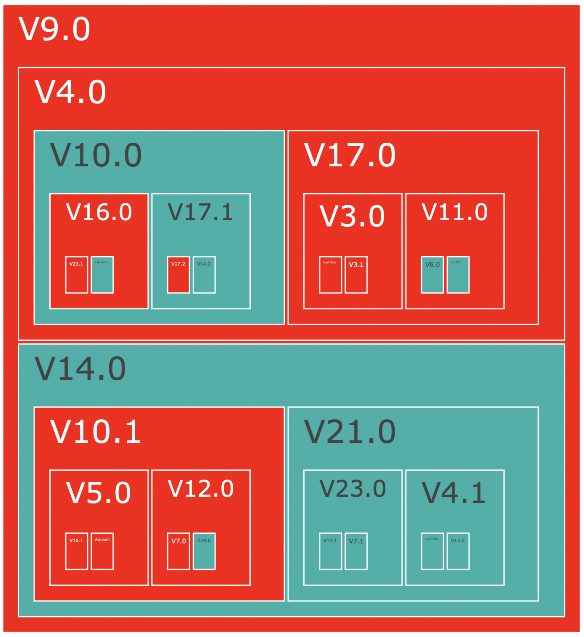
variables in the transaction data used for splitting the data. The treemap has the same colour mapping and labels. Size represents the level in the tree, with a smaller section representing a node further down the tree.

For these plots a dataframe was created for each tree, which contained columns for the child list, parent list, the level, and classification. Therefore, it was easy to access individual points to determine their respective levels, what their parent was and whether they have a classification of fraud or not fraud. The structure of the dataframe is illustrated in Table 7.1 which provides an example of the first three rows for the tree with index one.

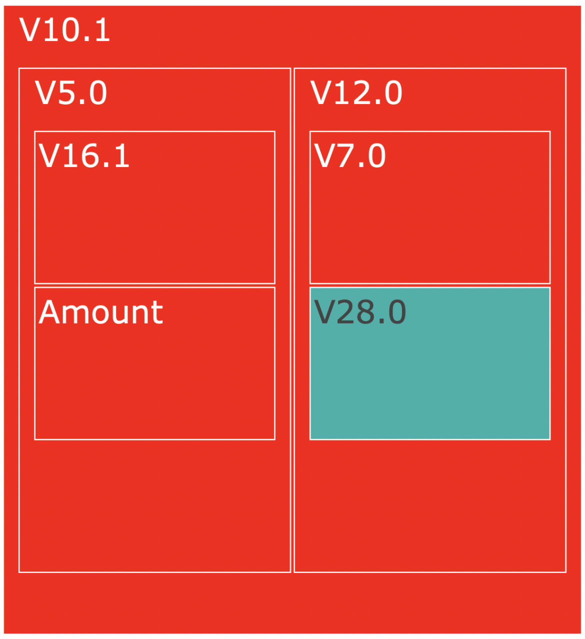
Children	Parent	Level	Classification	Threshold	Colour class
'V9.0'	' '	0	Not Fraud	-0.951	Colour Class
'V4.0'	'V9.0'	1	Fraud	1.041	Red
'V14.0'	'V9.0'	1	Not Fraud	-1.618	lightseagreen

Table 7.1: Structure of dataframe produced for each Decision Tree

To find the values that are appended in the dataframe the functions, tree.feature, tree.value and tree.threshold were used. The order of appending to these lists was very important



(a) Full Treemap



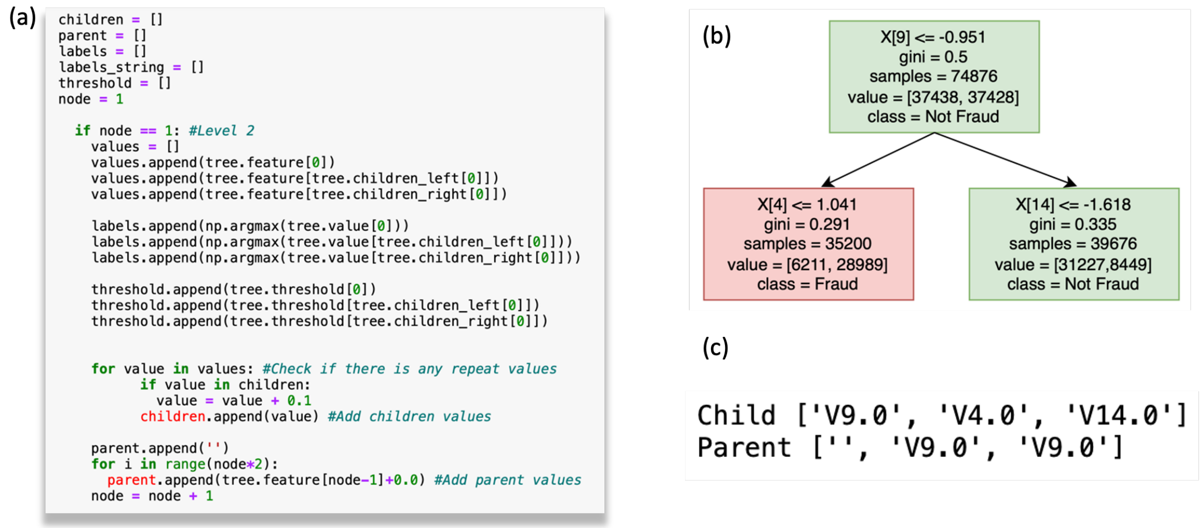
(b) Selected node view

Figure 7.5: Treemap plot of an individual Decision Tree. Size is mapped to level of the tree and colour to Fraud classification. Interactivity in the system allows the user to explore the lower levels, which can be seen less easily in the full view. (b) shows the view when V10.1 is chosen.

as this is what links the parent and child lists. Children_left and Children_right were used to transverse the tree structure and include all nodes. See Figure 7.6 for a snippet of the code and some outputs that explains how the lists where created.

There were some challenges in producing these hierarchical figures, firstly for plotting a sunburst or treemap in Plotly they require each value in the child list to be unique. In Decision Trees however the same variable can be used many times throughout the tree. This therefore meant a system had to be devised to make values identical values unique. This was achieved by adding 0.1 to each new appearance of the variable. This required an additional checking step at each level to determine if it was already present. See Figure 7.6 (a) which shows this step for level 2 of the first Decision Tree. The same issue arose for the string of Leaf node, however a space was chosen to be added to this instead of a 0.1 as the addition of an integer to this string did not make sense.

Another issue with the sunburst plot is due to a Plotly bug. Which means that the final level nodes appear a slightly different shade. This might be confusing to some users in trying to understand the classification and in the future other libraries should be explored



```
children = []
parent = []
labels = []
labels_string = []
threshold = []
node = 1

  if node == 1: #Level 2
    values = []
    values.append(tree.feature[0])
    values.append(tree.feature[tree.children_left[0]])
    values.append(tree.feature[tree.children_right[0]])

    labels.append(np.argmax(tree.value[0]))
    labels.append(np.argmax(tree.value[tree.children_left[0]]))
    labels.append(np.argmax(tree.value[tree.children_right[0]]))

    threshold.append(tree.threshold[0])
    threshold.append(tree.threshold[tree.children_left[0]])
    threshold.append(tree.threshold[tree.children_right[0]])


    for value in values: #Check if there is any repeat values
          if value in children:
            value = value + 0.1
          children.append(value) #Add children values

    parent.append('')
    for i in range(node*2):
      parent.append(tree.feature[node-1]+0.0) #Add parent values
    node = node + 1
```

```
Child ['V9.0', 'V4.0', 'V14.0']
Parent ['', 'V9.0', 'V9.0']
```

Figure 7.6: (a) shows a snippet of the code that is used to produce the child and parent lists shown in (c). The structure of this tree for just the first two layers is shown in (b) to give context for where the values in the lists come from. The parent list always starts with a blank value to indicate that the first value in the child list is the parent node and thus has no parents. Then for each layer of the parent list each value will always appear twice, to indicate it is the parent of two child nodes.

for sunburst plot visualisation.

7.1.4 Parallel Coordinate Plot

Viewing all the data is an important addition to the dashboard, it enables users to gain context of the a Random Forest model to increase understanding. As the data has many dimensions a multidimensional plot had to be chosen. A Parallel coordinate plot was chosen as it displays all the variables on one plot, and easily allows for brushing to constrain the range. Figure 7.7 shows the plot.

7.1.5 Slider

This slider allows the user to select which Decision tree is displayed. It updates the individual tree visuals along with the grey selection colour change on the scatter plot and small multiples plot. The update works by using a dash callback function. See Figure 7.8, for the code snippet. The slider index of the selected tree is outputted, which is then inputted into all the other plots to update them. It also takes in input from the small

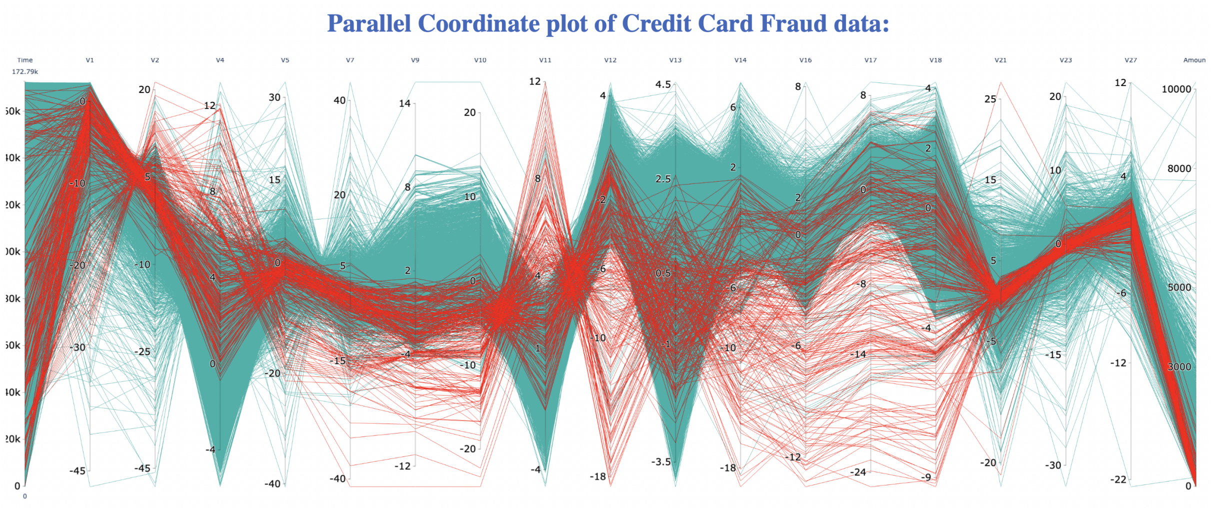


Figure 7.7: Parallel Coordinate Plot showing all the transaction data. Red maps to the fraud data and green to the Not Fraud data

multiples plot to update when a selection is made by clicking.

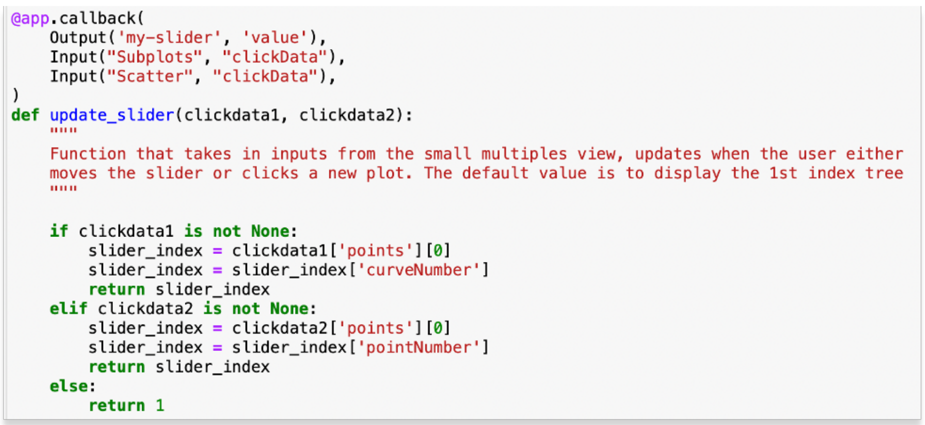

```
@app.callback(
    Output('my-slider', 'value'),
    Input("Subplots", "clickData"),
    Input("Scatter", "clickData"),
)
def update_slider(clickdata1, clickdata2):
    """
    Function that takes in inputs from the small multiples view, updates when the user either
    moves the slider or clicks a new plot. The default value is to display the 1st index tree
    """

    if clickdata1 is not None:
        slider_index = clickdata1['points'][0]
        slider_index = slider_index['curveNumber']
        return slider_index
    elif clickdata2 is not None:
        slider_index = clickdata2['points'][0]
        slider_index = slider_index['pointNumber']
        return slider_index
    else:
        return 1
```

Figure 7.8: Callback function for slider, which outputs the selected index of the tree that has been chosen as well as taking in input from the small multiples plot when a click selection is made.

7.1.6 Scatter plot of the Decision Trees

The scatter plot includes an x axis of dissimilarity score and y axis of accuracy. The colouring maps to the KMeans clustering that is applied with a cluster number of three (Figure 7.17), with the current selection coloured in grey. The number of clusters was chosen as three based on an elbow plot which identified this as the optimum number of clusters.

The data for this scatter plot is constructed as part of the class ClusterData. The accuracy

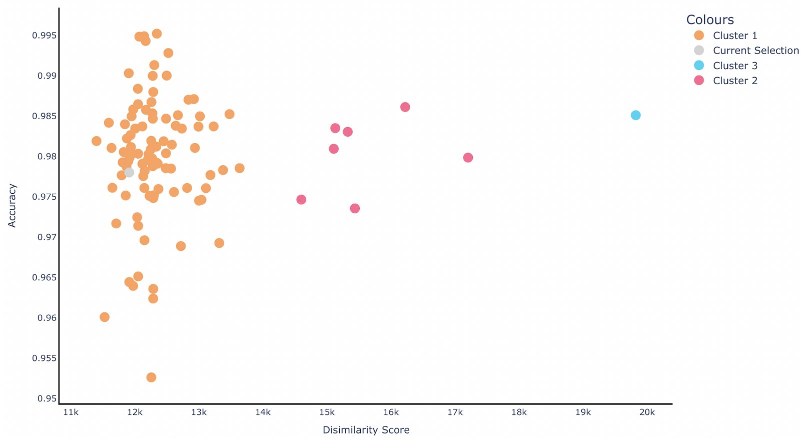
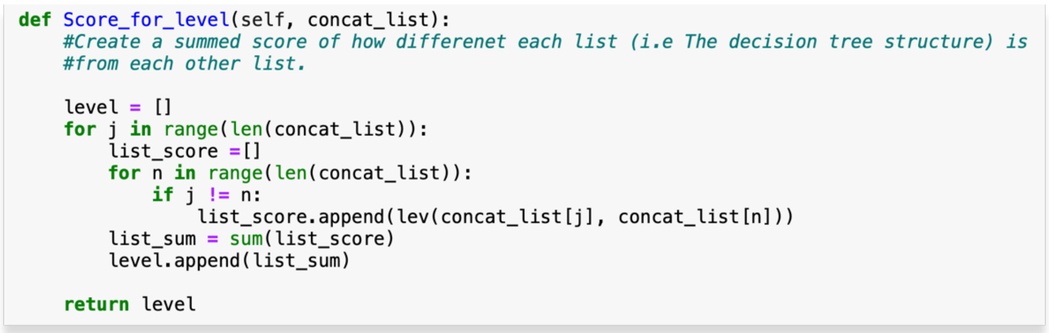


Figure 7.9: Scatter plot of all the Decision Trees in the Random Forest model, with x axis showing dissimilarity score and y axis showing accuracy.



```
def Score_for_level(self, concat_list):
    #Create a summed score of how differenet each list (i.e The decision tree structure) is
    #from each other list.

    level = []
    for j in range(len(concat_list)):
        list_score =[]
        for n in range(len(concat_list)):
            if j != n:
                list_score.append(lev(concat_list[j], concat_list[n]))
        list_sum = sum(list_score)
        level.append(list_sum)

    return level
```

Figure 7.10: Function for finding the string dissimilarity of the string that represents the nodes of the Decision tree at each level of the tree.

is calculated using the inbuilt scikit learn accuracy metric which is found for each of the 100 trees. The dissimilarity score is calculated by comparing the list of strings that represent the hierarchy (i.e the Child list). These lists are concatenated into strings for each level of the tree, which are then compared to every other string at this level using the Levenstien string similarity score. See Figure 7.10 for the code snippet that calculates this. Each level is assigned a certain weight as the first node in a Decision tree has a much larger importance than nodes in the 5th level.

7.1.7 Small multiples view of Decision Tree

This view shows all the sunburst plots of the 100 Decision Trees, it is displayed in a grid of 5 rows and 20 columns (Figure 7.11). The current selection is shown in light grey. This plot was chosen as it gives a good sense of the whole Random Forest being a collection

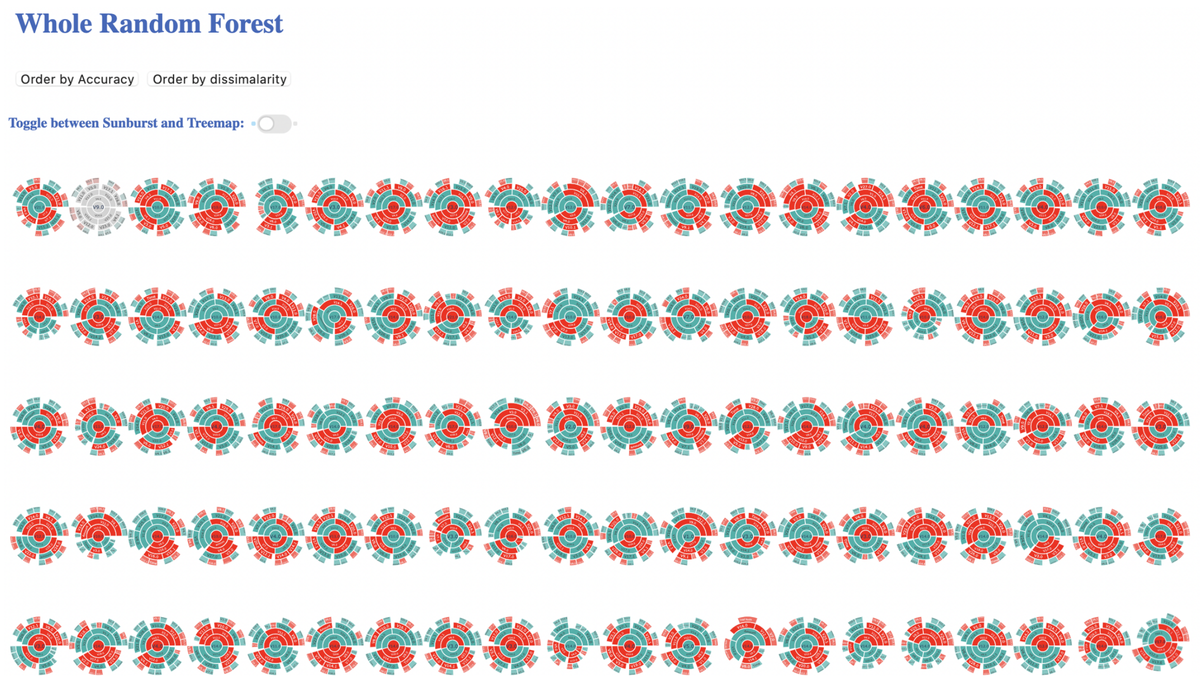
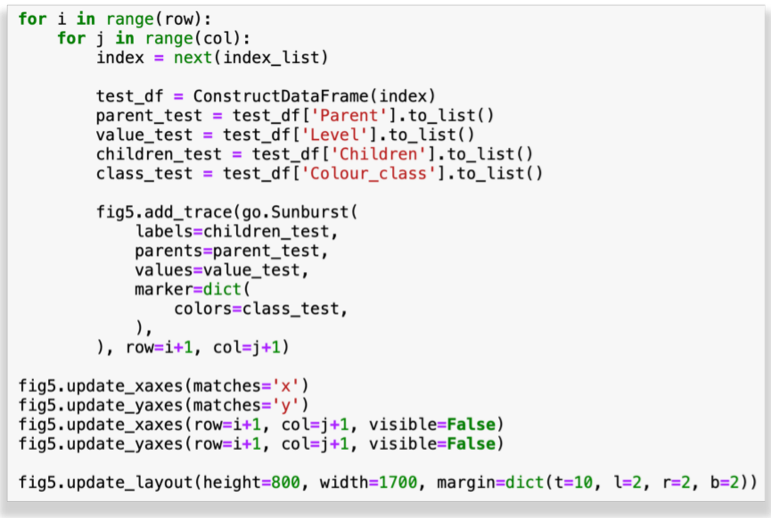


Figure 7.11: Small multiples plot, each plot is one Decision of the 100 that make up the Random Forest model

of many trees. It also easily allows the user to investigate how each tree classifies as the small version gives a good overview of the structure. This allows for further investigation through selection. The construction of this small multiples plot makes use of the traces feature in plotly, by adding each new sunburst as a trace. Figure 7.12 shows a snippet of code for displaying them all. This works by looping through the rows and columns. For the current selection the trace was added with all segments in light grey. This grey trace updates as the index of the current displayed tree is updated.

7.1.8 Selection using small multiples

A feature was added to allow the user to select a tree to be displayed by clicking the small version of it within the small multiples plot. This was achieved by adding an input into the update slider callback. This can be seen in the previous snippet of code Figure 7.8. This function extracted the index of the plot that was clicked and then inputted the new index into the other update functions, in the same way as when the slider is manually moved.

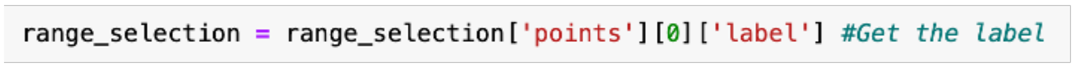


```
for i in range(row):
    for j in range(col):
        index = next(index_list)

        test_df = ConstructDataFrame(index)
        parent_test = test_df['Parent'].to_list()
        value_test = test_df['Level'].to_list()
        children_test = test_df['Children'].to_list()
        class_test = test_df['Colour_class'].to_list()

        fig5.add_trace(go.Sunburst(
            labels=children_test,
            parents=parent_test,
            values=value_test,
            marker=dict(
                colors=class_test,
            ),
        ), row=i+1, col=j+1)

fig5.update_xaxes(matches='x')
fig5.update_yaxes(matches='y')
fig5.update_xaxes(row=i+1, col=j+1, visible=False)
fig5.update_yaxes(row=i+1, col=j+1, visible=False)

fig5.update_layout(height=800, width=1700, margin=dict(t=10, l=2, r=2, b=2))
```

Figure 7.12: Snippet of code which shows how the small multiples plot is constructed.

7.1.9 Brushing of Parallel Coordinate plot

This feature means that the user can select certain variables of the sunburst or treemap by clicking a segment which then adds a constrained range onto the variable axis. The range is determined by the threshold of splitting that is used to split Fraud or Not Fraud. In this case all the values show splits that mean it is most likely fraud. Therefore, each selection with make it more likely that only fraud transactions are shown. This allows, the user to slowly build a PCP with only fraud data. To apply the range, first, the variable that was clicked had to be accessed from the plot, this is carried out by making use of the code in Figure 7.13.



```
range_selection = range_selection['points'][0]['label'] #Get the label
```

Figure 7.13: Code snippet used to get the variable that was chosen by click selection of a segment of sunburst or treemap

The threshold for this variable is then found by making use of the constructed dataframe for each tree, as described in Table 7.1. This threshold is then used to update the constraintrange section of the dictionary that makes up the dimensions which are then substituted into the PlotPCP function. To obtain only the values that are most likely Fraud a check was made if it was Fraud or Not fraud and then the range was flipped from either minimum value to threshold to threshold to maximum. See Figure 7.14 for an explanation

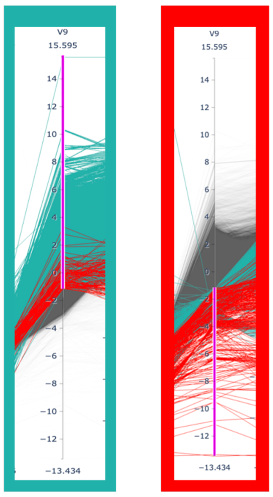


Figure 7.14: Splitting that would occur at a V9 node for if it was splitting off Not Fraud (Green) or Fraud (Red).

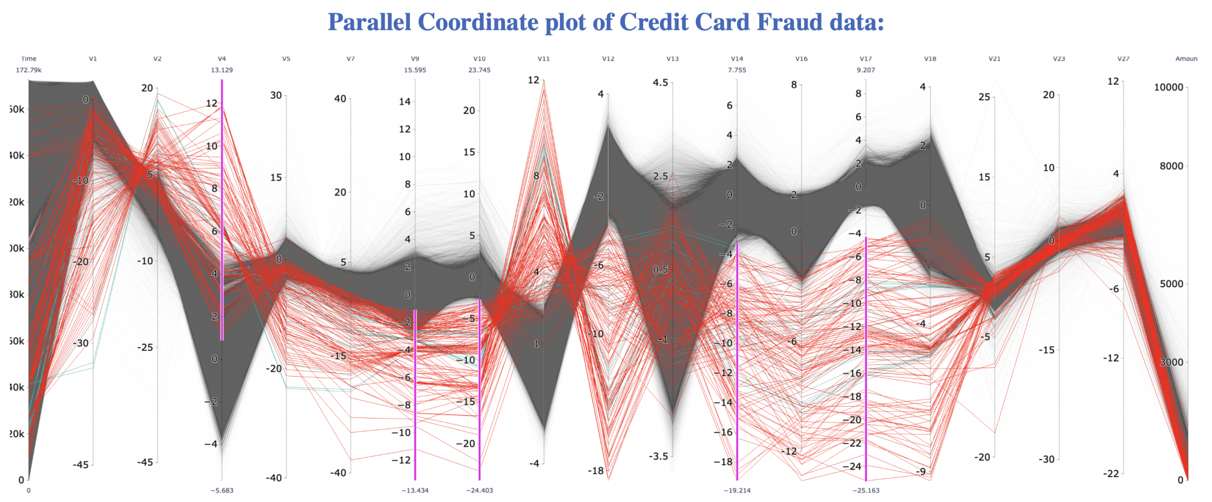
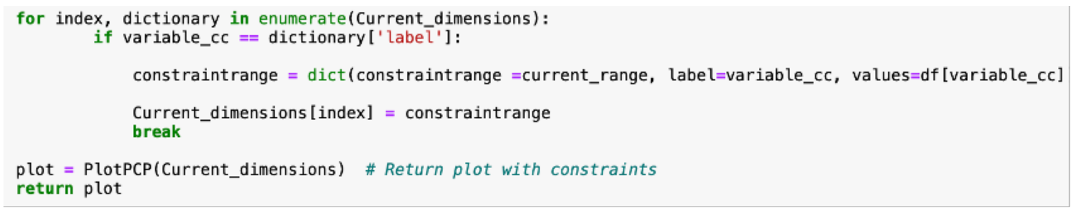


Figure 7.15: Brushing of the Parallel Coordinate plot. When selections of nodes have been made on the sunburst plot of V4, V9, V10, V14, V17.

of this. We see that when V9 is Not Fraud it would default to having all the values be -0.951 to the max of 15.595 and for Fraud it takes all the values from -0.951 to -13.433. Therefore in our system if we had a node of V9 classifying Not Fraud we would flip it to show -0.951 to -13.433, thus taking the inverse to allow the build up of only Fraud data. For Figure 7.15 shows where one branch of a tree has been navigated. To achieve this effect a list of all the current selected variables was created and to construct the dimensions list, the variables were looped through. See Figure 7.16

There were some difficulties with implementation of this feature. Some of the variables did not follow the pattern of requiring a flip of max and min which meant that before



```
for index, dictionary in enumerate(Current_dimensions):
        if variable_cc == dictionary['label']:

            constraintrange = dict(constraintrange =current_range, label=variable_cc, values=df[variable_cc]

            Current_dimensions[index] = constraintrange
            break

plot = PlotPCP(Current_dimensions)  # Return plot with constraints
return plot
```

Figure 7.16: Code snippet that loops through currently selected variables and updates their ranges to show values that are split to be most likely fraud.

extra debugging code was added for certain variables.

7.1.10 Reset Button

For the user to easily keep investigating how the tree classifies a reset button was added. This resets both the Parallel coordinate plot and the sunburst plot back to having no selections. This then easily allows for a new route of the tree to be chosen or for a completely new tree. This was achieved by resetting the list of currently selected variables to being empty and replotting the PCP with the default dimensions and replotting the sunburst/treemap plot when input of reset became true.

sectionEnhancements

7.1.11 Summary Treemap

A toggle was added to the small multiples plot to display an alternate summary of the whole forest which was the addition of a summary treemap. This displays all of the unique variables for each level. This view makes it easy to make conclusions about the use of variables for the whole model as well as identifying the most important features as these are the ones that appear most frequently, especially in the higher levels.

Labels	Values	Size
‘Level 0’	‘V11.0’	10
‘Level 0’	‘V14.0’	35
‘Level 0’	’V21.0’	4

Table 7.2: Structure of the dataframe for the summary treemap, showing an example first three rows

Implementing this involved the construction of a dataframe. See table 7.2, for the struc-

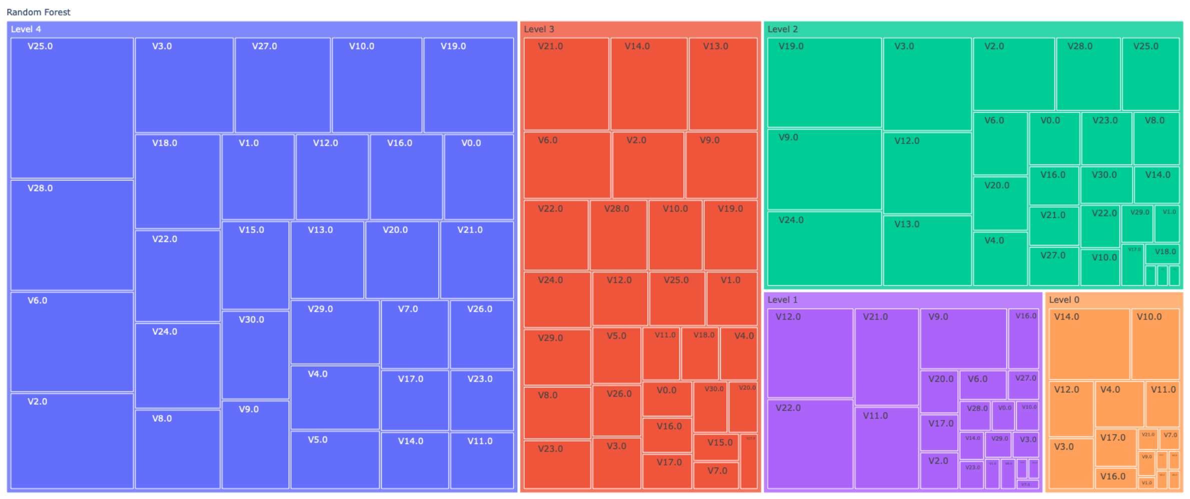


Figure 7.17: Summary treemap, containing all the unique values from each tree. With size mapping to how many times they occur

ture. To create this dataframe, the variables were obtained from each child list for each Decision Tree. They were then split into the five levels and all 0.1 additions were removed so that there were no duplicates. All of the values at the same level were then combined and the set() function was used to only obtain the unique values. The dataframe was then applied to the Plotly treemap.

7.1.12 Pybaobab View

This view is an image that displays the structure of individual Decision trees (Figure 7.18). It was constructed using the PyBaobab packaged which was published with the 2011 paper [41]. This plot was selected as it helps to explain the sunburst and treemap plots. However, it does not have any interaction so works well in parallel with the Plotly plots. The image was produced by using the package as published in [41]. This software pulls the data straight from the trained Random Forest model, so it is very easy to implement, the only customisation was the colouring to show the Fraud and Not Fraud classifications. To display these plots, each image was created based on the current selected tree, the image was then saved. The saved image was then imported as an HTML image which displays on the dashboard.

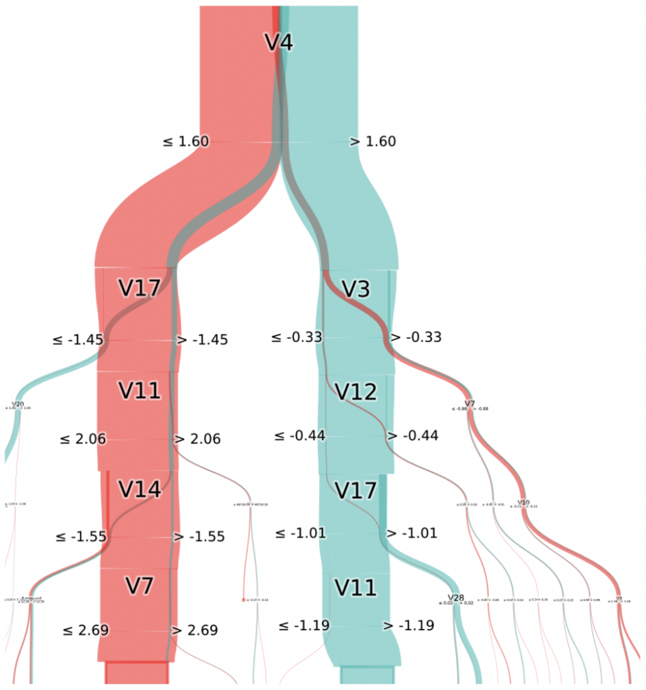


Figure 7.18: Decision Tree plot produced using the PyBaobab library [41], Fraud classification is shown in red and Not Fraud in sea green. The tree goes to a maximum depth of five splits

7.1.13 Sorting buttons

Buttons were added to the small multiples plot that allow the user to re-order the subplots by dissimilarity score or accuracy scoreThis makes it easy to make conclusions about the best-performing trees and their structures. This functionality was achieved by adding inputs to the UpdateSelection function, which were used to trigger the use of new columns in the scatter plot dataframe which were now ordered by accuracy or dissimilarity. Once ordered the index column was then used as the order of input into the ConstructDataframe() function. This meant that instead of 1,2,3 being inputted and plotted as the traces it could be for example, 84,32,12 inputted and then plotted as the 1st 2nd and 3rd traces. The difficulty of adding these buttons was being able to keep changing which it sorted by, as initially the condition of if n_clicks (which was the input variable for number of clicks of the sorting button) was just if it was bigger than zero. This was then updated to if n_clicks > accuracy_count or diss_count. These values were made global variables and an additional 1 was added to the score each time the button was pressed. This allowed the button to be pressed repeatedly Another issue was when the new indexes were used the

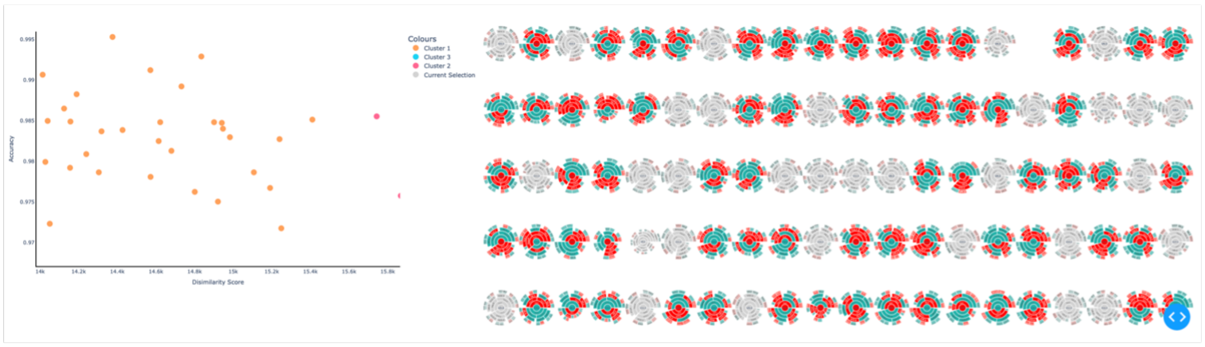


Figure 7.19: Box selection in the scatter plot, with selected plots shown in grey. This is when the default order of the plot has been used.

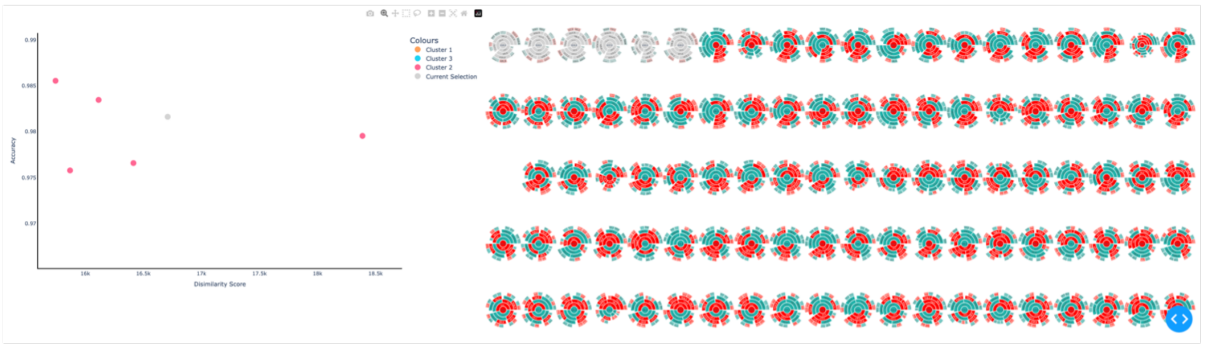


Figure 7.20: Box selection in the scatter plot, with selected plots shown in grey. The order by dissimilarity button has be pressed and in the scatter plot the box selection was placed around the high dissimilarity trees.

current selection also needed to be updated, otherwise it would still show the old position as the displayed Decision Tree. This was solved by adding a check and using a mapping to update the selected index. The new grey trace was then added with the mapped index.

7.1.14 Brushing in Scatter plot

Allow the user to select multiple Decision Trees by using a rectangular selection box. This feature allows the user to use a rectangular selection box to select multiple decision tree values on the scatter plot and have them selected on the small multiples plot. This is particularly useful in identifying the Trees which have the highest or lowest accuracy/dissimilarity scores. See Figure 7.19 and 7.20 for examples of its use.

The implementation of this involved the addition of a relayout variable of which the x and y axis ranges were extracted. The Decision Tree values were then looped through and only the values that fell within the ranges were appended to a list. The trace plots where then re-plotted with all those in the list plotted in light grey.

7.2 Code Documentation

As part of the code documentation a HTML document produced by sphinx is provided. This is also hosted on the GitHub along with the raw code, follow this link for the HTML output: github.com/HTML. This documentation breaks down the structure of the code and is easy to navigate. Due to the code structure of a Dash app, (not using many classes) a Doxygen documentation was not used. When produced this only showed good documentation of the pre-processing code. Sphinx gave a better overall documentation. The raw code can also be seen hosted on Github follow this link: github.com/raw_code

Chapter 8

Testing and evaluation

This section presents the final dashboard and an evaluation of the project. The evaluation is carried out by checking if case study questions have been answered along with an analysis of performance.

8.1 Results

For the best presentation of the software refer to the two videos

1. For the full video explaining all features and case studies follow this link: youtube.com/demo

2. For a shortened video intended for presentations follow this link: youtube.com/shortdemo

3. For a full scale image of the default dashboard view follow this link: github.com/full_image

Final screenshots are also provided in this section of the whole system. See Figure 8.1. The nature of this data meant that the software was unfortunately not able to be tested on other data sets, because no other suitable credit card fraud dataset are publicly available. Different numbers of samples were however used, refer to the performance analysis section for more detail.

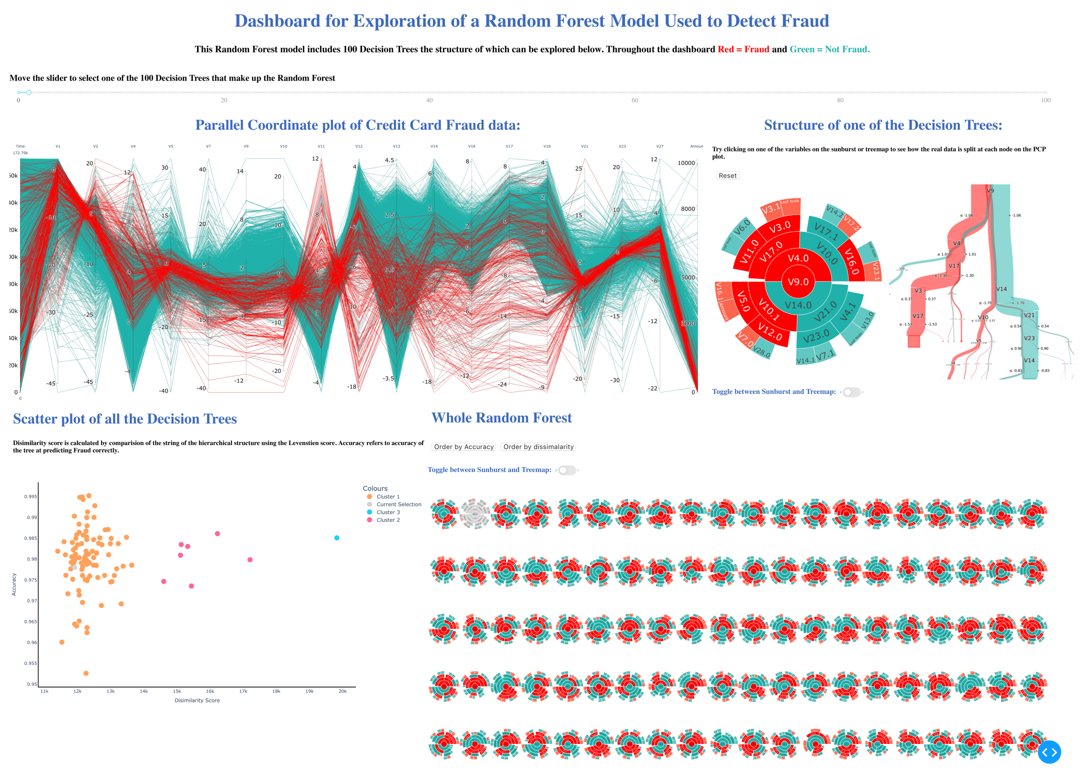

Dashboard for Exploration of a Random Forest Model Used to Detect Fraud
This Random Forest model includes 100 Decision Trees the structure of which can be explored below. Throughout the dashboard Red = Fraud and Green = Not Fraud.
Move the slider to select one of the 100 Decision Trees that make up the Random Forest
Parallel Coordinate plot of Credit Card Fraud data:
Structure of one of the Decision Trees:
Try clicking on one of the variables on the sunburst or treemap to see how the real data is split at each node on the PCP plot.
Reset
Toggle between Sunburst and Treemap:
Scatter plot of all the Decision Trees
Dissimilarity score is calculated by comparision of the string of the hierarchical structure using the Levenstien score. Accuracy refers to accuracy of the tree at predicting Fraud correctly.
Whole Random Forest
Order by Accuracy
Order by dissimalarity
Toggle between Sunburst and Treemap:
Colours
Cluster 1
Current Selection
Cluster 3
Cluster 2
Accuracy
Dissimilarity Score

Figure 8.1: Full view of the Credit Card Fraud detection dashboard. This shows the default view when initially loaded.

8.2 Case studies

To conduct testing and obtain results some case studies were generated. These case studies helped evaluate if there is a need for this software, its functionality, and whether it is an improvement on existing systems. Evidence for the answers to these case studies questions is provided below in the form of screenshots of the system. The video explanation of this dashboard also addresses these case study questions.

Case Study A

Can we understand and draw meaningful conclusions about a Random Forest model from this dashboard? There are a few ways that this dashboard achieves this case study:

1. The small multiples plot with the feature of interactivity allow us to easily see the structure of Decision tree plots. This helps us understand the structure of an individual tree as well as the overall structure of the whole Forest.

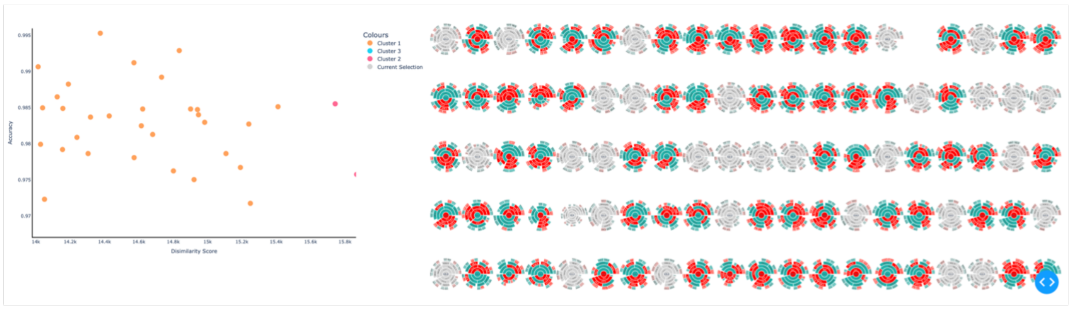


Figure 8.2: Scatter plot and small multiples plot with box selection shown. The plots chosen in the box selection are shown in grey on the small multiples plot.

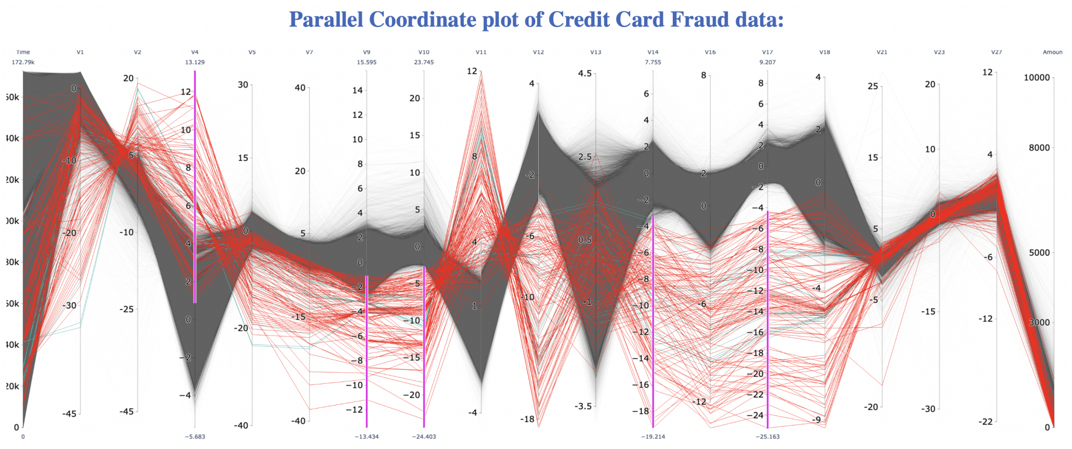


Figure 8.3: Parallel Coordinate plot showing the navigation of one path of one of the Decision Trees, which is demonstrated by the splitting of the data being applied to the parallel coordinate plot as range constraints.

2. The scatter plot with axis accuracy and dissimilarity score allows us to see similar trees and the performance of the trees. The interaction of this with the small multiples plot allows us to see exactly which tree has the properties shown in the scatter plot. This is illustrated with the brushing shown in Figure 8.2

3. Being able to see how a Decision Tree classifies by seeing it update in real time on a parallel coordinate plot is a very valuable resource in interpretability. It shows a clear link between the Decision tree and transaction data. See Figure 8.3

Case Study B

What is the most important variable for prediction of credit card fraud? To quickly identify the most important feature for the whole Random Forest, the summary treemap can be used Figure 8.4. The mapping to size allows us to see the largest section

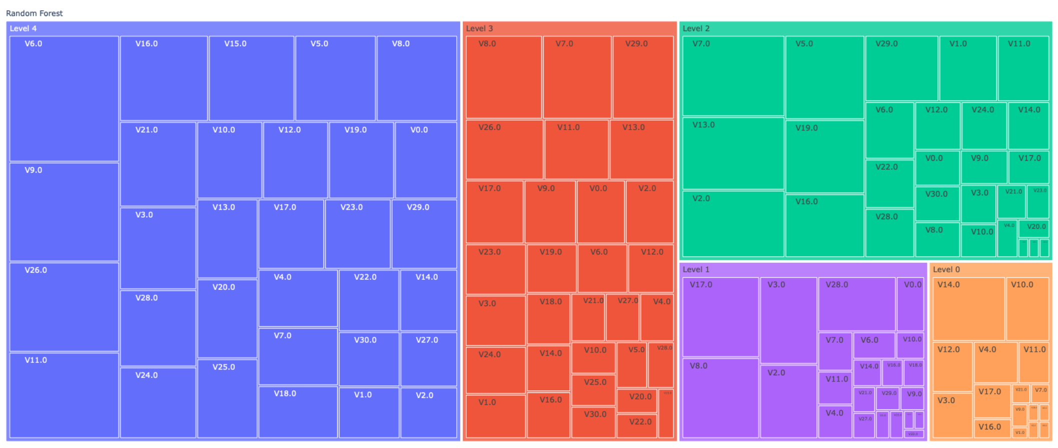


Figure 8.4: Summary treemap, which shows every value that occurs in any of the 100 trees for each level. Size maps to the number of occurrences i.e the largest section equates to the most common feature.

for the highest level of the tree (Level 0) corresponds to the most important variable. We see that in the case of our trained model V14 is the most important variable.

For identifying the most important feature for a singular Decision tree, it is easy to see the highest variable node by selecting individual trees. As well as being able to choose the most important feature based on where the variable appears in the tree, the parallel coordinate plot can also be used, by seeing which feature appears to split the data better when a variable is selected. This is described in Case Study B Figure 8.3

Case Study C

Does this system offer any improvements on existing software? In the previous systems three different systems were reviewed. A comparison of them with this system will now be carried out. Table 8.1 shows a comparison.

We see that for the features reviewed that our dashboard covers most of them. It also offers improvements on some of the features covered by the other dashboards. The parallel coordinate plot showing all the data is a better overall view than the data tables. The full forest view was built upon the Icicle version in Rfx, and therefore offers some improvements. It has been scaled so that all the trees can be seen in one view which due to sunburst being chosen instead of icicle makes it easy to view the whole forest.

	Fraud Detection Dashboard	**iForest [43]**	**Rfx [18]**	**SYLVIA [25]**
Full Data View	Yes, Parallel Coordinate plot. (Hard to pinpoint each data point however the overall trends are easy to see).	Yes, a table view of the data. Hard to navigate the data when it is a large dataset.	No	No
Splitting values shown	Yes, within the parallel coordinate plot	No	Yes, in the tree view	No
Full Forest view	Yes, Sunburst plot	No	Icicle plots	No
Individual Tree View	Yes, Sunburst and PyBaobab. (The paths of the two views are easy to navigate)	Yes, Decision Path view (Path adds extra data by adding a histogram plot for each feature however navigating the Decision path is difficult)	Yes, a tree view (Easy to follow the path)	Yes (The Decision Tree view is harder to follow than the other dashboards)
Clustering of Trees	Yes, Scatter plot using dissimilarity of string.	Yes, within the data overview.	Yes, within a scatter plot. 5 clusters are shown.	No
Interactivity	Yes	Yes	Yes	No
Feature View	Yes, Summary Treemap.	Yes, a histogram of features is included.	No, not a comparison of the features	Yes, a list of features sorted by importance
Evaluation of prediction	No, accuracy is included but no other evaluation.	Yes, (True positives, false positives etc are shown).	No	No

Table 8.1: Comparison of the three other dashboards for Random Forest visualisation with our software (Fraud Detection Dashboard)

8.3 Interpretation of the results

Firstly, looking at the overall view of the trees we see that most have very similar structures. With only a few having higher dissimilarity scores and some extreme examples. The trees which show more dissimilar structures are not the highest accuracy trees. They all have accuracy around 98%, which is around the average.
The most important values are the variables 14, 10 and 12. This information is gained from the summary tree map plot. This conclusion also makes sense when looking at the parallel coordinate plot, (Figure 8.5) the data shows clear splits of Fraud and Not Fraud. This data is all anonymised, so we do not know what these variables directly correspond to. There is also a lack of information about what they are to avoid revealing how these models work. To stop perpetrators from by passing the models detection. However, we can make some educated assumptions about what they could be. The simulated dataset published as part of a paper with the original data provides some insight into what these variables could be [20].

- Distance from previous transactions (A large distance could indicate someone else using the card)
- Number of previous transactions to an account (A low number could indicate a payment to a new account)
- Time between previous transactions (Fraud perpetrators sometimes carry out as many transactions as they can before the card gets blocked, therefore a very short time between transactions)

8.4 Performance analysis

As defined in Bobs project guidelines [19] a distinction is made between the software in presentation mode and exploration mode. The presentation mode is optimized for the best visuals but has the trade-off of lower load time. However, as this mode is just used to present the system the slower times do not matter as much. For the exploration mode it

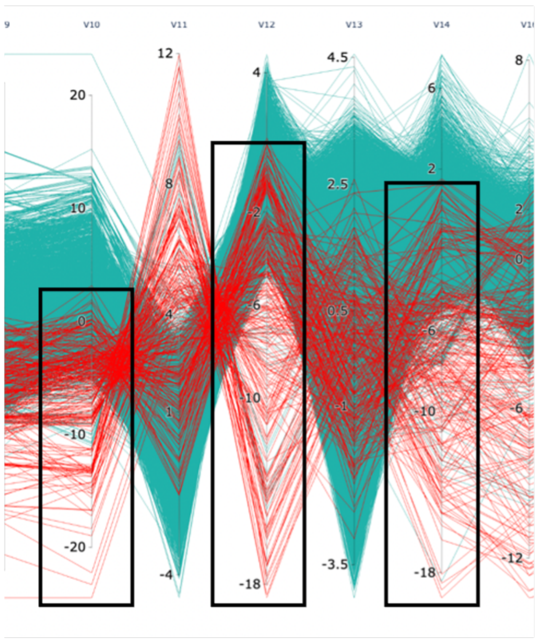


Figure 8.5: Parallel coordinate plot when V10, V12 and V14 have been selected. All these variables show a clear split in Fraud and Not Fraud.

was chosen to only use 50,000 data points. This is still a large amount of data but makes loading speeds much faster. It makes it a much better user experience, as it quickly implements the interactions that help the user make conclusions. The number of data points for each mode was chosen based on performance of the Random Forest model and the speed of loading. Firstly, looking at the performance of the Random Forest model. The accuracy, AUC and Precision were all calculated See Table 8.2. We see that 150,000 performs the best for accuracy and precision, hence it was chosen for the presentation mode, to give us the best results.

	50,000	**100,000**	**150,000**	**200,000**	**250,000**	**284,807 (All data)**
Accuracy	0.99920	0.99956	0.99971	0.99954	0.99950	0.99941
AUC	0.97617	0.99584	0.97613	0.98463	0.98262	0.97560
Precision	0.76190	0.8095	0.90196	0.83333	0.88679	0.88549

Table 8.2: Accuracy, Precision and AUC scores for differnet numbers of samples

8.4.1 Performance time evaluation

To first help us understand why it is important for quick load speeds, the 2019 article by Pavic et al [6], explains that 'High-performing sites engage and retain customers better than low-performing ones'. This applies to online websites but is relevant to this dashboard as it behaves the same as a locally hosted website. The performance of the dashboard particularly when it is in exploration mode is therefore very important. This article also gives some examples, 'the BBC found they lost an additional 10% of users for every additional second their site took to load'. It is therefore vital that if we want users to use this software as a resource, it needs to make them wait for the least amount of time possible. Table 8.3 shows the differing speeds of the software for the two different modes. This software was run locally on a MacBook Air M1 2020 (8GB RAM).

We see that the exploration mode (50,000) performs better for most features and the same for the scatter plot selection. The loading times for the exploration mode are still not optimal for the best user experience and future work should be carried out to lower these speeds.

	50,000	**150,000**
Initial full Load	1.3s	2.46s
Selecting a new Tree	2.21s	4.5s
Brushing of PCP	2.15	4.3s
Scatter plot selection	1.44	1.44

Table 8.3: Table of performance times for different features

To give an evaluation of the performance of this dashboard Google's lighthouse tool [14] was used. This is aimed at online websites however we again want a similar evaluation for our dashboard. This is because the interactions from users with a website and with this dashboard are very similar. The metrics it tested for are as followed:

- First Contentful Paint – Marks the time at which the first text or image is printed.
- Largest Contentful Paint – Time at which the largest text or image is painted.
- Speed Index – How quickly the contents of the page visibly populate.
- Cumulative layout shift – Movement of visible elements

	50,000	**150,000**
First Contentful Paint	0.4s	1.2s
Largest Contentful Paint	22.3s	47.9s
Speed Index	11.5s	25.5s
Cumulative layout shift	0.19	0.107

Table 8.4: Evaluation of performance using Google's lighthouse tool [14] for the exploration mode (50,000) and presentation mode (150,000). Colouring representing the classification of performance.

For all but the FCP score for 50,000 these values are all outside the range that is considered a good performance. This aligns with the conclusion about the load speeds of the features. The software is still usable, however to optimise user experience these load speeds should be improved. The use of a smaller dataset is also shown to increase speeds, however decreasing it more than the 50,000 used in exploration mode will decrease how well the Random Forest model performs in its classification of Fraud.

There is lots of animation which improves the visual but slows down the software, this could be removed to increase speeds. The removal of the enhancement features also improves the speed of the software however without these some of important capabilities of interpretability are lost as well as a better overall user experience. There are also inefficiencies within the code that slow down the performance. For example, to show a selection, a new graph is traced over the original. This means new graphs must be drawn all the time instead of just a change of colour. An additional function to make sure old graphs are cleared may help to improve the performance.

Chapter 9

Conclusion

This project has covered the production of a novel approach to visualisation of credit card fraud detection. This was created in the form of an interactive Python coded dashboard. There is an increasing need for explainable machine learning and visualisation is a powerful tool for this. The dashboard was built following the principles of visualisation 'Overview first, filtering and selection, details on demand', [34] while building on previous work for visualising Machine learning (specifically a Random Forest). This was combined with the training of a Random Forest model to detect Fraud. Which was developed building on previous work and using data published by Worldline [20]. The dashboard includes the following imagery:

- Parallel Coordinate Plot of all the data, with interaction to show how the tree splits the data.
- Sunburst/treemap plot which shows an individual Decision tree structure.
- PyBaobab tree view
- A Scatter plot of Accuracy vs Dissimilarity with each tree represented by one point.
- Small multiples plot which shows every tree of the forest in one large view.
- Summary treemap that includes all the unique variables at each level.

The aim of this dashboard was to help make this model interpretable and explainable. This would allow fraud experts who may not understand a Random Forest model, to begin to understand how it classifies. Fraud is a important issue and one that is predicted to cost the UK more each year. To prevent fraud, we need effective and often complex models, like the use of our Random Forest classifier. Removing the idea of a Random Forest being a 'Black box' model also helps to highlight any ethical issues with the model. Does the model make assumptions that it should not? Does it overlook anything that could be a safety concern?

The case studies used to evaluate performance of our dashboard, allow us to conclude that the system presents: An effective way of making conclusions about a Random Forest model, allows us to see the most important features and offers improvements on previous systems. There is still parts of the model that need effective visuals and some of these suggestions can be see in Section 9 (Future work).

Chapter 10

Future work

There are improvements that could be made to the existing dashboard, were there no time constrains.

- Improvement of visibility of the parallel coordinate plot. Plotly does not allow for the width of the data lines to be increased which make it hard to distinguish some of the points. A different plotting library could be explored to improve the Parallel Coordinate plot.
- Evaluation of how well the trees and forest are classifying would be a good addition. This could link to fine tuning of the Random Forest from within the dashboard, for example, if the user was able to edit the max depth of the tree within the dashboard.
- Option to change number of clusters and the clustering algorithm. To add more user options.
- Choice of string dissimilarity measure. The only option currently is the use of Levenstien score. A different measure may give new spreads of the Decision Trees.
- When and if new data that is not anonymised becomes available training of a model and displaying this in the dashboard would be a valuable update. This would allow for more conclusions based on the actual data values.
- Improve the load speeds by removing inefficiencies with the code and explore other ways to make some of the visuals. This will give a better user experience.

Bibliography

[1] A. Chatzimparmpas, R. Martins, I. J. K. K. F. R. A. K. The state of the art in enhancing trust in machine learning models with the use of visualizations. *State of the art report* (2020).

[2] A. Vellido, J. Martín-Guerrero, P. L. Making machine learning models interpretable. *European Symposium on Artificial Neural Networks Volume 12* (2020), 163–172.

[3] Andrea Dal Pozzolo, O. C. Y.-A. L. B. Learned lessons in credit card fraud detection from a practitioner perspective. *Expert Systems with Applications 41* (2014), 4915–4928.

[4] Angelos Chatzimparmpas, R. M. M. I. J. a. A survey of surveys on the use of visualization for interpreting machine learning models. *Information Visualization 19* (2020), 207–233.

[5] ASHTIANI, M. N., and RAAHEMI, B. Intelligent fraud detection in financial statements using machine learning and data mining: A systematic literature review. *IEEE Access 10* (2022), 72504–72525.

[6] Bojan Pavic, Chris Anstey, J. W., 2020. https://web.dev/why-speed-matters/: :text=Performance

[7] Braganca, W. O. a. I. E. K. Comparative analysis of python microframeworks: Flask, dash, and cherrypy. *A Guide for Newly Graduated College Students* (2023).

[8] Bunge, M. A general black box theory. *Philosophy of Science 30* (1963), 346–358.

[9] Cagatay Turkay, R. L. A. H. On the challenges and opportunities in visualization for machine learning and knowledge extraction: A research agenda. *International Cross-Domain Conference for Machine Learning and Knowledge Extraction 10410* (2017), 191–198.

[10] Dejan Varmedja, M. K. S. S. M. A. A. A. Credit card fraud detection - machine learning methods. *International Symposium INFOTEH-JAHORINA* (2019), 1–5.

[11] Emmanuel Ileberi, Y. S. . Z. W. A machine learning based credit card fraud detection using the ga algorithm for feature selection. *Journel of Big Data 9* (2022).

[12] Experian. Credit card fraud detection kaggle, Accessed 20-06-2023. https://www.experianplc.com/media/latest-news/2023/credit-card-fraud-soars-to-10-year-high/.

[13] Fuchs, J. e. a. Teaching clustering algorithms with educlust: Experience report and future directions. *IEEE computer graphics and applications.* (2020), 98–102.

[14] Google, 2016. https://developer.chrome.com/docs/lighthouse/overview/.

[15] Grinberg, M. *Flask web Development: Developing Web applications with python.* O'Reilly Media, 2018.

[16] Géron, A. *Teaching Clustering Algorithms With EduClust: Experience Report and Future Directions.* O'Reilly Media, 2017.

[17] Hans-Jörg Schulz, S. H. a. H. S. Survey, the design space of implicit hierarchy visualization. *IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS 9* (2021), 165286–16529.

[18] J.Eirich, M. M. D. J. M. S. J. B. T. S. Rfx: A design study for the interactive exploration of a randomforest to enhance testing procedures for electrical engines. *Computer graphics forum 41* (2022), 302–315.

[19] Laramee, R. S. Bob's project guidelines: Writing a dissertation for a bsc or msc in computer science. *Innovation in Teaching and Learning in Information and Computer Sciences* (2011).

[20] Le Borgne, Y.-A., Siblini, W., Lebichot, B., and Bontempi, G. *Reproducible Machine Learning for Credit Card Fraud Detection - Practical Handbook.* Université Libre de Bruxelles, 2022.

[21] Liu, X. A. M. S. C. J. D. A. R. D. F. E. E. W. Q. . L. R. S. Visualization resources: A survey. *Information Visualization 22* (2023), 3–30.

[22] Maxime Dumas, M. J. M. V. L. L. Financevis.net: A visual survey of financial data visualizations. *IEEE InfoVis 2014 2* (2014).

[23] McFarland, A. What is the best language for machine learning?, Accessed 05-09-2023. https://www.unite.ai/what-is-the-best-language-for-machine-learning/: :text=Over

[24] Mosqueira-Rey, E. H.-P. E. A.-R. D. e. a. Human-in-the-loop machine learning: a state of the art. *Artifical Intelligence Review 56* (2022), 3005–2054.

[25] N. Medoc, V. C., . F. P. . a. M. G. Visualizing prediction provenance in regression random forests. s.l.:s.n. *Artifical Intelligence Review* (2022).

[26] N. V. Chawla, K. W. B. L. O. H. W. P. K. Smote: Synthetic minority over-sampling technique. *Journal Of Artificial Intelligence Research 16* (2002), 321–357.

[27] Pallets. Pallets, 2010. flask, Accessed 21-08-2023. https://flask.palletsprojects.com/en/2.3.x/.

[28] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research 12* (2011), 2825–2830.

[29] Plotly, Accessed 20-08-2023. https://plotly.com/dash/dashboard-engine/.

[30] Pozzolo, A. D. *Adaptive Machine Learning for credit card fraud.* PhD thesis, Université Libre de Bruxelles, 2020.

[31] PWC, 2018. https://www.pwc.co.uk/audit-assurance/assets/explainable-ai.pdf.

[32] S P Maniraj, A. S. S. D. S. Credit card fraud detection using machine learning. *International Journal of Engineering Research Technology (IJERT) 8* (2019), 1–6.

[33] Schulz, H. J. Treevis.net: A tree visualization reference. *Computer Graphics and Applications 31* (2011), 11–15.

[34] Shneiderman, B. *The eyes have it: A task by data type taxonomy for information visualizations.* Boulder CO, 1996.

[35] Terence Parr, Tudor Lapusan, P. G. M. E., 2021. https://github.com/parrt/dtreeviz/blob/master/README.md.

[36] uk finance. Fraud - the facts 2021, Accessed 24-06-2023. https://www.ukfinance.org.uk/system/files/Fraud

[37] ULB, M. L. G. Credit card fraud detection kaggle, Accessed 20-06-2023. https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud.

[38] W T Hewitt, S Larkin, A. J. G. A semigraphical method for the analysis of complex problems. *Proceedings of the National Academy of Sciences* (1957), 923–927.

[39] W T Hewitt, S Larkin, A. J. G. Techniques for visualizing multidimensional data. *1997 CERN School of Computing* (1997), 189–199.

[40] Ware, C. *Information Visualization: Perception for Design: Second Edition.* Elsevier., 2004.

[41] Wijk, S. v. d. E. J. J. v. Baobabview: Interactive construction and analysis of decision trees. *Conference on Visual Analytics Science and Technology 31* (2011), 151–160.

[42] Xu-Meng Wang, T.-Y. Z. Y.-X. M. J. X. . W. C. A survey of visual analytic pipelines. journal of computer science and technology. *Conference on Visual Analytics Science and Technology 31* (2016), 787–804.

[43] Xun Zhao, Y. W. D. L. L. a. W. C. iforest: Interpreting random forests via visual analytics. *IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS 25* (2019).

[44] Zijie J. Wang, Chudi Zhong, R. X. T. T. Z. C. D. H. C. C. R. M. S. Timbertrek: Exploring and curating sparse decision trees with interactive visualization. *IEEE VIS 2022 Volume 12* (2022), 163–172.