VICTORIA UNIVERSITY OF WELLINGTON
Te Whare Wānanga o te Ūpoko o te Ika a Māui

School of Engineering and Computer Science
Te Kura Mātai Pūkaha, Pūrorohiko

PO Box 600
Wellington
New Zealand

Tel: +64 4 463 5341
Fax: +64 4 463 5045
Internet: office@ecs.vuw.ac.nz

A Counterfactual Visualisation System for eXplainable Machine Learning

Caitlin Fisher

Supervisors: Dr Andrew Lensen, Dr Stuart Marshall and Hayden Andersen

Submitted in partial fulfilment of the requirements for Bachelor of Engineering with Honours.

Abstract

Machine Learning (ML) is a subset of Artificial Intelligence (AI) that can be used to solve large and complex tasks. A big issue within ML is the lack of interpretability. This issue affects people working with these ML models, as it can be unclear how the models are producing the decisions/outputs. Therefore, eXplainable Machine Learning (XML) has been developed to help improve the interpretability and understandability of ML models. A type of XML is an algorithm called counterfactuals, which analyses a specific instance in the data to explain. Counterfactual explanations improve interpretability by identifying feature values that need to be changed to allow for an altered prediction. This project proposes and prototypes a website that implements counterfactual algorithms and information visualisation techniques. Via the website, you can manipulate and filter the data to explore the counterfactual algorithm and use different graphs such as tables and bar charts to display the counterfactual results. We performed a usability study on our website and a matrices-based evaluation of our algorithm to identify issues for future work.

Contents

Chapter 1

Introduction

1.1 Problem Statement

Artificial Intelligence (AI) has now become a big part of our everyday lives and allows systems to mimic human behaviour. A sub-set within AI is Machine Learning (ML), which uses a range of different methods to improve the models by learning through its experiences and is used in systems to achieve complex tasks and make decisions. Due to ML becoming more advanced and playing bigger roles, we need to ensure that ML models are designed, constructed, and used in ethical and moral ways. Alongside model accuracy, ethical and moral obligations are important to consider in ML as the model needs to be; fair, to ensure the decisions being made contain no basis; robust, to make sure systems are stable and can withstand/overcome unfavourable conditions; and, to ensure that the model and/or decision are easy to perceive [1]. However, the problem we face is that ML systems can be extremely complex, making it difficult to interpret the decisions made [2].

1.2 Motivations

Due to these issues, eXplainable Machine Learning (XML) is currently being researched and implemented to achieve a better interpretation of the system to ensure ethical and moral obligations are met. While there are many different XML methods, an excellent method for solving the interpretability issue when using machine learning models is counterfactuals. Counterfactuals are used to identify changeable features in a dataset that could be altered to achieve a different outcome, they produce specific interpretable outputs for individual data points [1]. It can be described as "If input X had changed, output Y would have changed" or "If input X had not occurred, output Y would have changed" [3]. However, there is still a lack of options available for people to use and visualise XML and counterfactual outputs.

Therefore, this project will develop a proof-of-concept visualisation system to explore how visualisation techniques can support counterfactual algorithms and help meet ethical and moral obligations, such as the existence of biases in data, around the use of ML. Combining the counterfactual algorithm with data visualisation will improve the interpretability of ML decisions and counterfactual explanations.

1.3 Goals

The primary goal of this project is to develop a platform to generate interpretable counterfactual explanations. This goal can be achieved by completing these sub-goals:

- To implement a counterfactual algorithm that can be used on a wide range of datasets.
- To implement a visualisation system that works with datasets and displays counterfactual explanations in an interactive way. This includes: allowing users to explore counterfactuals through filtering aspects of the dataset; providing useful visualisations by displaying counterfactuals of each instance; allowing users to analyse both textual and graphical outputs of the counterfactuals. Through the visualisation system, users will achieve a better understanding of complex machine learning systems and how counterfactuals are used. The proof-of-concept prototype will use a single dataset but should be designed for a general case.
- To evaluate the system using two methods. Firstly, to test the counterfactual algorithm, we will use different metrics such as Euclidean and Manhattan distance alongside evaluating how many features are changed and how much the features have been modified. This is to check the performance and fidelity of the algorithm. Secondly, usability testing will review the visualisation system and counterfactual explanations by evaluating the usability and intuitiveness of the visualisation system. It will also determine if the counterfactuals are informative and interpretable, by ensuring the tester can understand the counterfactual.

1.4 Target Audience

This website is aimed at people who have basic knowledge of AI and/or ML. This is because the primary goal of this system is to help people explore counterfactual explanations to have a better interpretation of the dataset which may be difficult for those who do not already have a background in AI and/or ML. XML and counterfactuals may also be more complex for those who do not have basic knowledge in this area. For people who do have domain knowledge and are interested in exploring their dataset through counterfactuals, it will be an easy and quick tool to learn about the basic functionality of these models and will show the effectiveness of these systems

1.5 Contributions

The project offers the following contributions:

1. A design for a tool where counterfactual explanations can be explored and displayed through; filtering and extracting information on a dataset; producing counterfactual explanations using the built-in counterfactual algorithm; exploring counterfactual explanations using different graphs and tools.
2. A proof-of-concept prototype that demonstrates the functionality of the design, with discussions around the implementation of the prototype and future improvements.
3. An evaluation of the current prototype, that measures the reliability and robustness of the algorithm, alongside a usability test to evaluate usefulness, user-friendliness and functionality.

Chapter 2

Background and Related Work

2.1 The Problem

A big challenge that ML faces is the ability to create a model that is both interpretable and complete (complete meaning as accurate as possible) [4]. It is often found that ML models have either one or the other, as "it is difficult to achieve interpretability and completeness simultaneously" [4]. Due to this, ML models often lack interpretability and do not provide information on how the system reached a specific decision. For those people who would like to check the robustness (robust meaning is the algorithm's performance consistent and stable), no information is provided to explain how the model reached its decision. Often ML models lack interpretability even for people who understand the internal logic of the system. We can see this when comparing a decision tree and a deep neural network model (DNN).

A decision tree is a machine learning model that is considered to have a simple interpretation when the size of the decision tree is not extensive. By starting at the root node, you cascade down the tree through the nodes and edges, which define subsets within the dataset [3]. The final prediction comes when the leaf node is reached. When this tree is short, with only a few nodes, it provides a straightforward and human-friendly explanation. However, the bigger the tree gets the more nodes there are making it harder to interpret and explain [3]. For example, a tree with 10 nodes is interpretable as seen on the left-hand side of Figure 2.1 where links and nodes are clear. However, when you start increasing the number of features and variables, the nodes and links will start to increase. Therefore, the tree will become less interpretable, seen on the right-hand side of Figure 2.1. Consequently, large decision trees that support complexity have the side-effect of being less interpretable. [3].

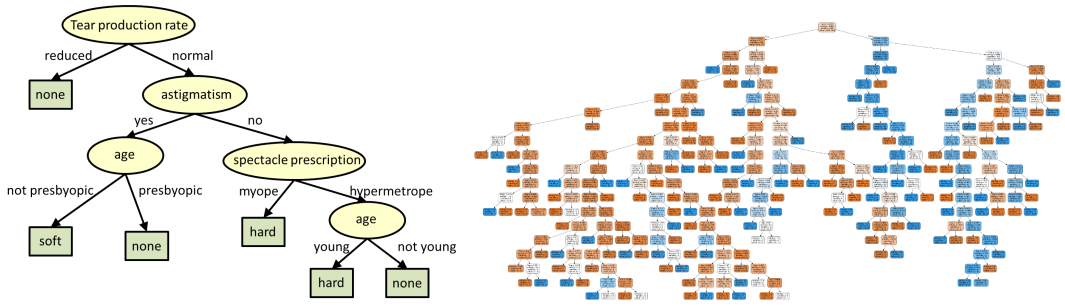

Figure 2.1: Left hand side: Small decision tree [5]. Right hand side: Big decision tree [6].

Whereas a DNN can be used for many different machine learning tasks with great completeness, it lacks interpretability even with a small model [7]. DNN contains many layers

of nodes and weights in between the input and output layers [8]. Due to the many layers and numerical weights, it does not provide reasonable interpretation and contains no information about how it reaches its predictions.

The objective of the project is to implement a counterfactual algorithm that can provide this interpretability for machine learning models while keeping its complexity and completeness. In cases such as small decision trees where it produces an interpretable model, counterfactuals may not provide more interpretability to the model as it is already easy to understand the outputs and does not need extra explanation. Therefore, counterfactuals are more needed for complex, complete and non-interpretable models such as DNN and large decision trees.

Another problem is how to create a visualisation tool that can present counterfactuals in a useful way. Therefore, the project will also aim to demonstrate a visualisation tool that is intuitive to use, and that can display useful counterfactual information.

2.2 Interpretation vs Explanation

Interpretation and explanation are commonly used when discussing XML and have very similar definitions. However, the differences between the two can be seen as interpretability focuses on the system over the outcome and measures the level of understanding of how the outcome is predicted. Explainability focuses on providing an explanation for the outcome over how the system came to this decision.

2.3 Counterfactual Algorithms

There are many different counterfactual algorithms each having its own strengths and aims for the algorithm. The counterfactual algorithm needed for this project is one that can work with multiple different datasets, not an algorithm that is fitted to a specific dataset. Due to different datasets being used, the counterfactual algorithm also needs to work with multiple different black-box models, as different types of datasets work better on certain models. Also, we want to minimise the number of features that are changed over how big the change is within a feature. Due to the scope and time limit for this project, the algorithm will need to be available to the public and easy to understand/implement within the scope.

2.3.1 Taxonomy

Taxonomy is a "scheme of classification in which things are organized into groups or types." [3]. This is useful to consider as it helps design, develop and evaluate systems [3]. There are four aspects to consider with counterfactual taxonomy: type, model, scope and result type.

Type Type distinguishes between intrinsic and post hoc. Counterfactuals are post hoc which is an XML technique that applies to a method that analyses and explains a model after training. This project will be using a black-box model (random forest) to first train on the data then the counterfactual method will be applied.

Method The method can either be model-specific or model-agnostic (can work on any model). Counterfactuals are model-agnostic, as many different models can be used. However, in the project, we will use a specific model.

Scope The scope can either be local (explains specific or groups of predictions) or global (explains the model as a whole). Counterfactual's scope is local as they can only explain one prediction at a time. This can be seen as a limitation of the counterfactual method, as it restricts the use of the method and could take a long time to analyse large amounts of predictions. Its advantage is that it is very specific, and can provide in-depth explanations.

Result Type The result type of a counterfactual method is a data point. The method returns existing or new data to make a model more interpretable, making it a data point type. Counterfactual explanations do this by taking a data point and returning a similar data point that contains the smallest changes to it so that the prediction changes.

2.3.2 Existing Counterfactual Algorithms

There are many counterfactual algorithms that could be used for this project. We will be using at least one of these algorithms to create the visualisation system. Here, we will discuss a few different counterfactual algorithms.

Growing Sphere

Growing Sphere is a counterfactual algorithm that uses a "generative approach that locally explores the input space of a classifier to find its decision boundary" [9]. The growing sphere uses a greedy approach, selecting the best, most obvious or most convenient option currently available [10]. This greedy approach is used to find the best counterfactual explanation by looking at all instances in all different directions. The two main steps in this method are generation and feature selection. The generation step will generate the decision boundary of the classifier. Feature selection is then used to minimise the number of features when making vectors of the explanation.

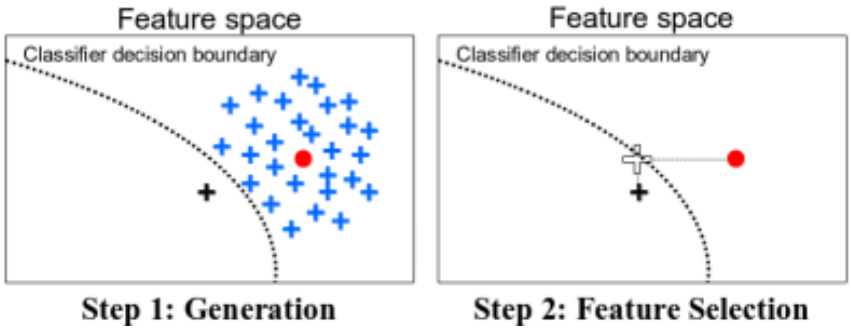

Figure 2.2: Growing Sphere Steps

DiCE

DiCE is a counterfactual algorithm that stands for Diverse Counterfactual Explanations [11]. This algorithm was created as a counterfactual explanation that aims to satisfy two properties: feasibility across users' context; diversity of the counterfactuals. Therefore, DiCE was created for "generating and evaluating a diverse set of counterfactual explanations based on determinantal point processes." [11].

The main advantage of this method over other counterfactual methods is that it provides a solution that generates counterfactuals with "substantially higher diversity for these datasets." [11]. Also generating a high number of unique and valid counterfactuals.

MACE

MACE stands for Model-Agnostic Counterfactual Explanation which aims to generate the nearest counterfactual explanations using any given distance function while supporting any plausible constraints [12]. This is done by mapping the nearest counterfactual problem into a sequence of satisfiability (SAT) problems, by expressing both the predictive model and the distance function as logic formulae.

The testing of this algorithm showed that it is able to generate plausible and diverse counterfactuals at "provably optimal distances" for any dataset [12].

2.3.3 Limitations

A limitation of counterfactual explanations is their ability to analyse groups of instances, as counterfactual algorithms are specific to each instance. Little research has gone into reviewing and visualising groups of counterfactual explanations.

Another limitation with XML and counterfactuals is the need to still have some domain knowledge in both XML and ML to understand the explanations. For those who want to learn or cannot implement an XML algorithm, it may be hard to explore these algorithms. Also without prior knowledge, the output may be difficult to understand. While XML has improved the interpretability of ML, it can still be improved upon using data visualisation methods. This can also help those with little domain knowledge to understand the explanations.

2.4 Visualisation

The aim of visualisation is to "aid our understanding of data by leveraging the human visual system's highly tuned ability to see patterns, spot trends, and identify outliers" [13][14]. Visualisation tools that are well-designed can provide an alternate way to present information other than calculations and data. Using visualisation is an effective way of making XML (counterfactual explanations) more comprehensive. It was stated in [13], that the prefered graphical representations are spatial positions such as bar charts and scatter plots, as it "leads to the most accurate decoding of numerical data and is generally preferable to visual variables such as angle, one-dimensional length, two-dimensional area, three-dimensional volume, and colour saturation" [13]. They are simple, accurate and able to display variables/values clearly.

The most important step for developing an effective visualisation is knowing the purpose and audience of your visualisation. What do they want to see? and/or what might they want to focus on? [13]. By narrowing down the purpose, we can decide how to transform the data into graph visualisation tools. The data we want to present is the counterfactual explanation, and to show the changes the counterfactual has recommended.

A few key data visualisation tasks that will be important to develop visualisations for counterfactual explanations are [15]:

- Filter: Filtering out uninteresting/irrelevant data. This will ensure the visualisation only displays the information needed to aid our understanding and to find important patterns.
- Overview: Gain an overview of a collection of data. This will provide quick information and display simplified results.
- Details-on-demand: Select an item or a group of indexes to target specific details when needed.

2.5 Proposed Solution

This project will aim to provide a tool to better understand decisions made by ML models. The project will develop a visualisation system for displaying counterfactual explanations.

XML aims to provide "fairness, robustness, transparency, and interpretability" [1] to inform and help people understand a decision made by an ML system. It also aims to provide information on what needs to be changed to reach the desired result [16].

This project will use counterfactuals, which are used to identify features in a dataset, that could be altered to achieve a different outcome [1][16]. They provide explanations about ML models' decisions to help people understand their outcomes. Counterfactuals are ideal for people who at least have basic theoretical knowledge of AI and/or ML as they do not have to understand the internal logic of the system to understand the counterfactual. When a counterfactual algorithm is applied to a model, it will produce a counterfactual for each instance in the dataset.

The visualisation system will be the focus of the project. This is an important step for this project as it will provide the resources for users to review a dataset and a black-box model. Also, pinpoint features, classes and instances within the dataset to analyse specific information. It will do this through a filtering system that will allow a user to alter the dataset so they can analyse different areas of the data. After the counterfactual is found, the system will display the counterfactual both textually and graphically. This will allow the user to read and analyse the produced counterfactual in two different ways to make it more interpretable.

2.5.1 Requirements

The following requirements have been produced after reviewing the proposed solution:

1. The system needs to be able to show an overview of each counterfactual explanation.
2. The system must allow the user to manipulate and filter the dataset.
3. The system must allow the user to analyse both a single counterfactual explanation and multiple counterfactual explanations.
4. The system needs to be able to highlight the changes within the explanation.
5. The system must allow a user to import their own data.
6. The system needs to be accessible.

Chapter 3

Design

The chapter will introduce the primary persona, use case scenarios, designs and software architecture for the system. Justification on design choices will be made to determine which design best suits the system requirements.

3.1 User Model Planning

In the background, it was shown that a visualisation system can be used to improve the interpretation of XML/counterfactual explanations. Now we need to determine useful use cases of this system. The user model plan will present the persona and use cases for this system so that designs can be made that fill these requirements.

3.1.1 Persona

Hannah Richardson is a third-year university student studying computer science, majoring in Artificial Intelligence. She has always had an interest in different ML areas and wants to explore new topics in her spare time. While she is excellent with computers and learning new things, Hannah does not have a lot of time due to university. Hannah learns best through images, graphs or short texts. Therefore, she looks for intuitive and effective visualisation tools to help her explore different ML topics.

3.1.2 Potential Use Case

This section will present possible use cases for the counterfactual visualisation system to help understand the design's purpose and functionalities. Only two use cases are displayed but they can all be seen in Appendix A or accessed through GitLab [17].

Scenario 1: Select one instance and find a counterfactual.

Hannah wants to view the counterfactual explanation for a specific instance. Therefore, she inputs the index of the counterfactual into the filter instance section and clicks the 'Find Counterfactual' button. The counterfactual will appear below the table.

User	*System*
Selects one instance. Clicks the' Find Counterfactual' button.	 Runs the counterfactual algorithm on the one instance. Display results in textual form.

Scenario 2: Select a specific class to analyse. Hannah wants to see if a specific class in the datasets have similar counterfactual outputs.

User	*System*
Selects a class in the filter section. Clicks the' Find Counterfactual' button.	 Runs the counterfactual algorithm instances. Display results in textual form and in a graph.

3.2 Software Architecture

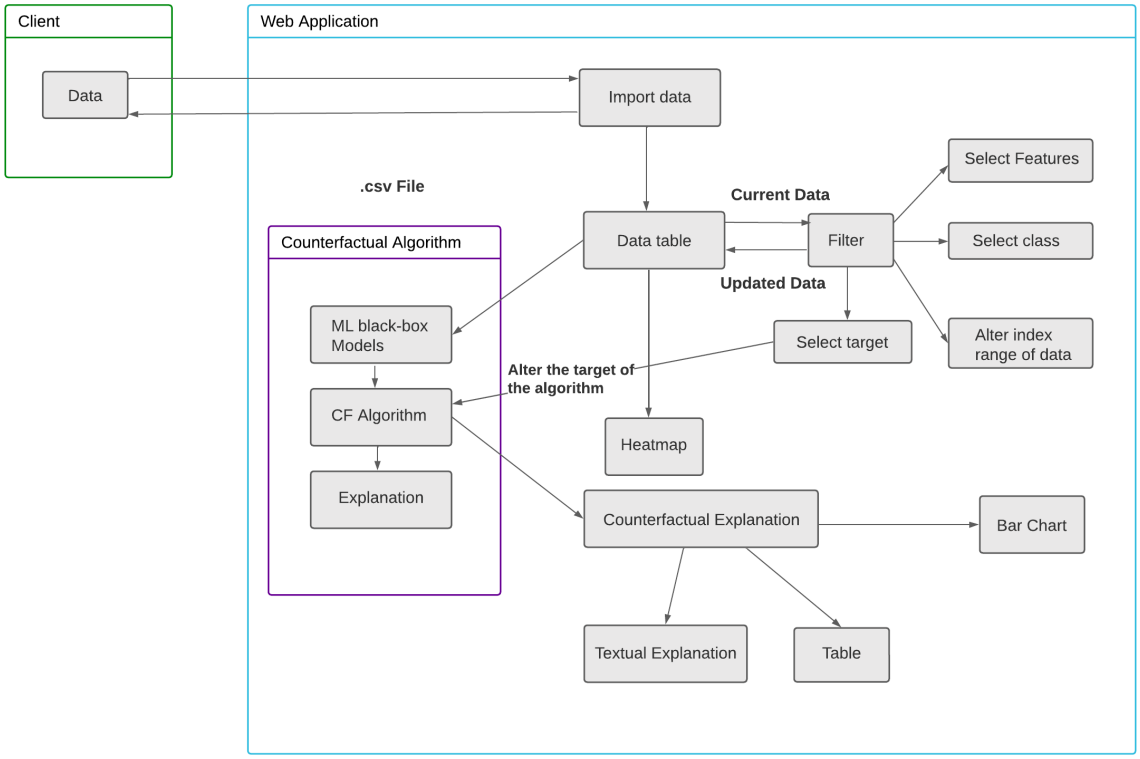

Figure 3.1: Diagram of software architecture

The diagram of the system architecture, Figure 3.1, displays the different components, the user, web applications and the counterfactual logic and demonstrates the relationship between each component. The web application component was separated into small specialised functions. Each component and function was given a name to express its function and links to other components/functions to demonstrate the flow of the system. Through this diagram, we can see the modularity and cohesiveness of the overall system.

The user component represents what an individual user needs to provide to the system. At the start of the user interaction with the web application, they will need to provide a .csv file that contains a dataset they want to review.

The web application will then take this .csv file and convert it into a readable data table that will be displayed on the website. If a user decides to filter/manipulate this data the

Requirement	Description	State
R1	Overview of counterfactual explanation	Yes
R2	Manipulate and filter data	Yes
R3	Analyse counterfactual explanations	Yes
R4	Highlight changes	Yes
R5	Import	Yes
R6	Accessible	Yes

Table 3.1: State of requirements within the software architecture

system will use the current data displayed in the table and alter the data by changing the selected feature, classes and/or the index range. Once this has been applied it will be passed back to the data table so that the updated dataset can be displayed. If the target class is changed, the counterfactual algorithm will be altered, as the target class is changing the algorithm, not the data. Using the data table a heatmap can be generated to find correlations between features and the class.

Once the data table displays what the user wants to review, the user can click a button to start the counterfactual algorithm. This algorithm will produce a new data point that can be used to form an explanation.

The counterfactual explanations will then automatically be displayed both textually and via tables to display what feature needs to be changed and display the values changes. Using the counterfactual explanation a bar chart can be displayed to graph the feature changes.

As seen in the diagram, the system architecture does not require a database as the graphs used in the design do not need counterfactual explanations to be stored. However, a database will need to be added if a graph were to be implemented that needs previous explanation data points. This would add a database component to the system architecture.

Table 3.1 shows that the system architecture meets all the requirements for the system.

3.3 Wireframes and Designs

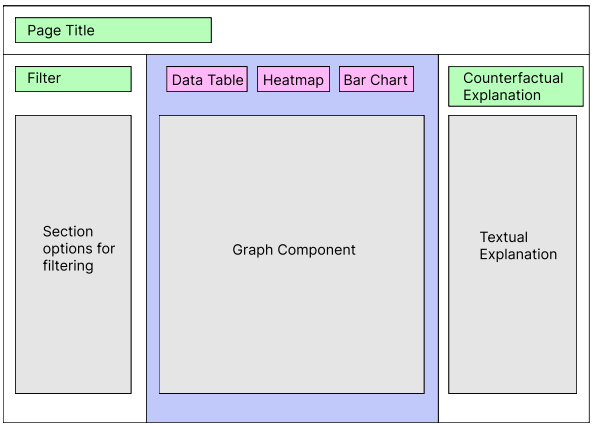

Figure 3.2: Wireframe of design 1

Design 1 seen in wireframe Figure 3.2, offers filtering, graphs and the counterfactual explanation all on one page. The pink rectangles are buttons that can be selected to swap between different graphs and tables. Provides a platform for future expansion as many more graphs could be added by just adding a button.

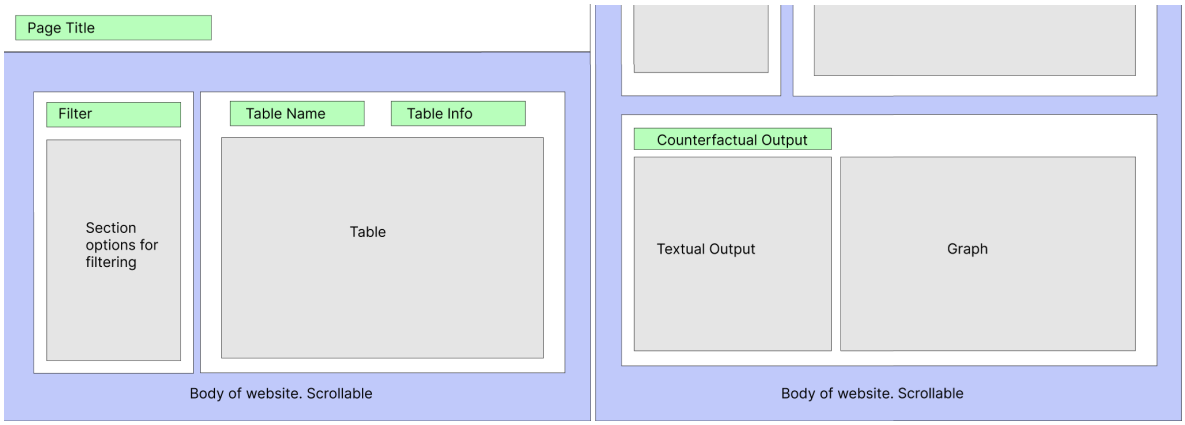

Figure 3.3: Wireframes of design 2

Design 2 seen in wireframes Figure 3.3, is more spaced out compared to design 1. However, offers all the same features.

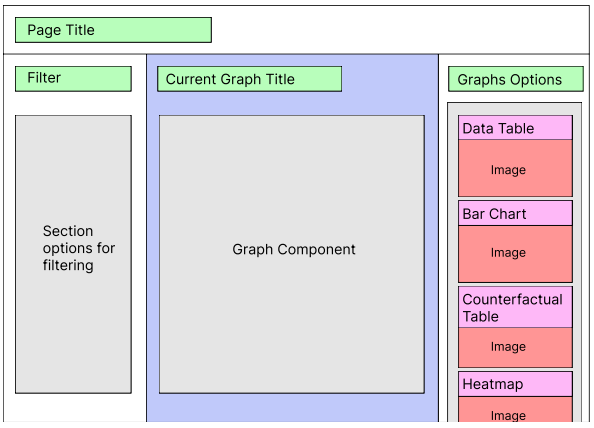

Figure 3.4: Wireframe of design 3

Design 3 seen in wireframe Figure 3.4, is similar to design 1, however, the textual counterfactual explanations are seen as a graph option that can be clicked on to view in the 'Graph Component'. Each option is also paired with an image so users can get an idea of the graph they are selecting.

3.3.1 Potential Graphs

Animated Bubble Chart

A bubble chart may help analyse multiple counterfactual explanations simultaneously, to help identify patterns. By using this chart we can analyse the effects of the changes made to each feature compared to how many other features were changed in the same explanations. More information can be added as more counterfactual explanations are found. An issue is that the bubble chart will need a lot of data to provide interesting patterns/analyse. If data is lacking the chart may not display better results than a bar chart, however, be more complex.

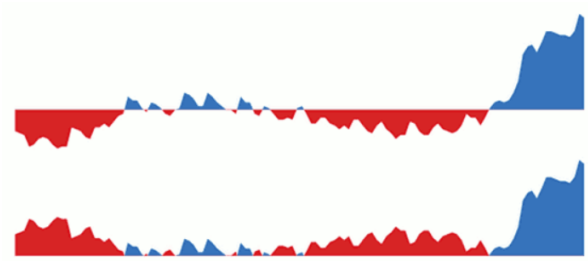

Figure 3.5: An example of a horizon graph

Time-Series Data: Horizon Graphs

A horizon graph could be used to compare different counterfactual changes in features. An example of this can be seen in Figure 3.5, the line could be used to indicate the original value to show how the explanation increases or decreases a value. Each feature could have its own graph and they could be presented underneath each other to so correlations and patterns between the features. This graph is very similar to the index chart however it is clearly showing the changes it is making rather than the differences between feature values. This would require past counterfactual explanations to be stored.

Bar chart

A bar chart could be used to compare the original feature value with the altered feature value. It would be a very simple graph that would show the differences between numeric features and the number of changed features for an instance. The bar chart would have difficulty representing boolean or non-numeric valued features. Another issue would be for groups of instances, it would only be useful for one instance at a time. However, no extra data would be required and it does not need to store any data.

Heatmap

A heatmap would be a simple graph that could show the correlations between features. While this graph would not provide any analysis for counterfactual explanations it could provide more information to help users understand changes made in the explanation.

3.4 Design choices

Design 2 (Figure 3.3) has been chosen as the design to implement, due to the design being more spaced out and separating the website into two sections: (1) the original data that can be filtered. (2) the counterfactual explanations and graphs. The other design would have struggled to display large datasets and users may get confused about which data table/graphs can and cannot be filtered.

3.4.1 Layout

The two wireframes seen in Figure 3.2 are the layout for the design. What is not clearly seen in these figures is that the body screen is scrollable. The overview of the data and the filtering section are seen first. The user can scroll down to see the counterfactual output. This is done as it follows Shneiderman's Information Seeking Mantra [18]. Which states the

overview always comes first. Also before counterfactuals can be found, filtering of the data can be done.

In the overview section seen on the left-hand side of figure 3.3, the data table is given more emphasis through its size on the screen over the filtering options (Requirement 2). This will guide the user's focus to the data table, as the data provides most of the information. The first column of the data table will be the indexes of the instances. This was done to identify the different instances so the users can correctly filter them. Also, the data may contain many different features and over a thousand instances, therefore the bigger the table, the easier it will be to read and understand the data.

The counterfactual section in the wireframe takes up the entire page as we expect the user's attention will only be on the counterfactual output. The layout currently has been split between an overview of the explanations and options for further analysis using graphs (R1 and R3). A textual explanation and two tables (one for the original data and another for the counterfactual explanation) will be used for an overview of both single and groups of counterfactuals (R1 and R3). The tables will be able to provide further analysis of each features changes (R4). While the heatmap and bar chart will allow for deeper analysis (R3 and R4).

3.4.2 Project Functions

The main functions of the design include:

- Import: Users will be able to import their own data to use in the system (R5).
- Dataset display: The dataset display would present the data that the system is using. It would be displayed in a table format with all the features and instances.
- Filtering: This function will allow the user to alter different aspects of the dataset (R2). This includes features, classes and instances. This will be done by creating a sidebar with all available filters that could be made.
- Counterfactual: The algorithm will use the filtered data to find the counterfactual for each instance.
- Counterfactual display and analyses: different graphs and textual outputs will be generated to review and analyse the counterfactual (R1, R3, R4). Four different methods will be used: (1) textual output, (2) two tables to compare data before and after counterfactual, (3) heatmap and (4) bar chart of the changed features in an instance.

Website

We have chosen a website for our application and will be designed for a standard-sized desktop (1440 x 1024 pixels) over a mobile app or mobile screen due to the large amounts of data that would be hard to read on a phone due to the small screens. Users are also more likely to look for these learning tools online. Therefore, making a website would be more accessible (R6). Also, the website requires no installation to start using their tool.

3.4.3 Filtering

We decided to include a filtering section that allows the user to filter the data as it helps achieve the main aim of this project which is to allow users to explore counterfactuals. By allowing users to adjust the data, they are able to look into more specific areas of the data and find more patterns. Included in the filter section are:

- Select the features, users can remove/add features from the view of the dataset that will be used for the counterfactual.
- Select a specific class to review or select all classes.
- Select a target class. This will allow the growing sphere algorithm to change features to achieve this target class.
- Select an index range, where the dataset starts and where it ends. This is an important filtering element as datasets can contain hundreds of instances and the algorithm will take hours to find all the counterfactuals for each instance. Therefore, by only targeting a range of 5 instances, the counterfactual result will be achieved faster.

3.4.4 Counterfactual Output

We have used multiple different ways to present the counterfactual output. We have a basic textual output and multiple graphs to help understand the counterfactual.

Textual Output

A simple counterfactual explanation can be described through textual output, also called a counterfactual statement. For example: "If X was changed to a specific value, Y could be changed to a specific Z" or "starting from instance C, change A and B to get a counterfactual instance" [4]. These counterfactual explanations would be interpretable for each instance. However textual outputs may be more overwhelming when evaluating a large group of instances.

Graphical Output

There are many different graphs and charts that we could use to produce this output. However, a lot of these effective and complex graphs need a lot of data from the counterfactuals. This would be difficult within the time frame of this project and would require data to be stored.

Therefore, we used three different ways that are less complex to present the counterfactual and its changes and patterns. (1) Produce two tables, one for the current data and one for the data after being put through the growing sphere algorithm, Figure 3.2. The counterfactual table shows the features that were affected and changed by the algorithm. Also highlights the feature values with green if it was increased or red if it was decreased. We used these colours as they are universal indicators for increasing and decreasing. (2) Provide a heat map graph of how each feature correlates to the others. This shows the relationships between two variables help identify patterns with the changes made using the growing sphere algorithm. (3) Lastly, is a bar chart, that takes a single instance to compare the differences between the current value and the counterfactual changes. This bar chart presents the features that have been altered in a simple format, that easily shows the changes.

Chapter 4

Implementation

4.1 Technology

Python was chosen as the language for my project over Java and react (JavaScript). Python has been heavily used throughout University courses meaning the language is familiar to engineering and COMP/AIML students at Victoria University. Python is also a popular choice for AI/ML applications due to their many advantages over other languages including [19]:

- Concise and understandable
- Helpful libraries (such as SciKit-Learn) to make coding easier
- High development speed

While react's primary use is for web development, we have less experience coding in this language and Java and Python provide better libraries and resources for machine learning models. However, Python also contains different libraries for the visualisation of both the data and websites. Plotly is a library that contains many great features which will be beneficial when creating a graphical output of the counterfactual data.

Plotly also contains a library called dash that we will be using to create the website, *Counterfactual Explainer (CFE)*. While Java contains libraries for website development, we have chosen python as it is more user-friendly as it is completely dynamic and Python programs are faster than Java programs [20]. It will also be faster to implement as we will not have to swap between languages or merge two components together.

4.2 Machine Learning Model

The ML model used is the random forest algorithm. This algorithm was chosen as it performs well with both classification and regression tasks, and can work with both categorical and continuous variables. This means that it can perform using multiple types of data which is important for the system, as users may input different datasets into the system. The model is also very resistant to noise, missing values and outliers which may be necessary if there are missing preprocessing steps. The Random Forest algorithm is very stable, which means if new data points are introduced into the dataset, there would not be much affected on the overall model. Random forest algorithms are complex when given large datasets, providing a good example for displaying the benefits of counterfactuals.

This algorithm runs the given data from the web application component before the counterfactual algorithm starts. This model is then used in the counterfactual algorithm to generate predictions for the new data points to check whether the counterfactual changes have made an impact on the outcome. These predictions help identify possible counterfactual explanations.

4.3 Growing Sphere Counterfactual

The counterfactual algorithm we have implemented is the Growing Sphere algorithm. This algorithm supports the requirements discussed in section 2.3 and was easy to implement within the scope. This method uses a "generative approach that locally explores the input space of a classifier to find its decision boundary" [9]. This section will go into more depth around the two main steps.

Generation

The generation's main aim is to make observations in the features space in l_2-spherical layers around an instance x to find the closest enemy [9]. The enemy is a term used by the author of the algorithm which is a term for a possible counterfactual, for the sake of consistency we will continue to use this term. This enemy is the closest to the instance which is found using the "pairwise_distances". Using two positive numbers a_0 and a_1, we can define (a_0, a_1)-spherical layer SL around x [9, 21]:

$$SL(x, a_0, a_1) = Z \in: a_0 \leq ||X - Z||Z \leq a_1 \tag{4.1}$$

The first step in the generation algorithm is to create uniformly n observations in the l_2-ball of radius η and center x as *SL(x, 0, η)*. To make sure that the generation step finds the closest decision boundary, the radius is updated and repeats the first step until no enemy is found in the first l_2-ball, *SL(x, 0, η)* [9, 21].

$$\eta \leftarrow \eta/2 \tag{4.2}$$

The generation algorithm will return the closest generated enemy e that was observed. This is then passed onto the feature selection to make it more interpretable [21].

Feature Selection

After the closest enemy e is found from the generation step, the next step is to minimize the l_0 component of the cost function *c(x, e)* [9]. The aim is to reduce the number of features changed in the counterfactual while still changing the output. This means we needed to maximize the sparsity of the vector *e-x* so that:

$$f(e) \neq f(x) \tag{4.3}$$

The final result of the feature selection will be e^* which is the final enemies identified in this algorithm.

Algorithm Output

```
Obs:  [7.06364698e+00 1.40612866e+02 1.79418113e+04 6.27530460e+00
 3.76046156e+02 4.02593602e+02 1.24397397e+01 2.94684281e+01 3.39586624e+00]
0 ennemies found in initial hyperball.
Expanding hypersphere...
Final number of iterations:  349
Final radius:  (6.273999999999953, 6.291999999999953)
Final number of ennemies:  1
Feature selection...
Reduced 8 coordinates
New enn:  [7.06364698e+00 1.40612866e+02 1.79418113e+04 1.09058218e+01
 3.76046156e+02 4.02593602e+02 1.24397397e+01 2.94684281e+01 3.39586624e+00]
```

Figure 4.1: Counterfactual output

The output displays the number of enemies/features that the feature selection has identified. It then displays the new feature values to show the changes that have been made. With these changes, a textual counterfactual explanation is created stating what needs to be changed for each instance.

4.4 Data

The current dataset used in the system is a water potability dataset that was found on Kaggle [22]. Due to not having implemented the import dataset function yet, we have chosen this dataset for implementing other components and evaluation. However, most of the system architecture is not hard coded to allow for easy dataset changes. The aim of the dataset is to identify if water is safe for human consumption. It contains nine features (ph, hardness, solids, chloramines, sulfate, conductivity, organic carbon, trihalomethanes, turbidity) and is a binary classification, where one means potable and zero means not potable.

Some pre-processing was done on the data, where any instance was removed if it had a missing value, as we wanted to ensure the data was accurate. After pre-processing there is a total of 400 instances. Due to the dataset being set in class order, meaning all classes were grouped together, we shuffled the data to ensure we split the data with both types of classes. Lastly, we added an index column in the data. This was done for the visualisation of the data so that users can identify individual instances in the data and filter the index range.

While the system is currently only set up to use the pre-defined dataset, a small number of changes would need to be done to the code so that users can import their own data.

4.5 Visualisation

We have been able to implement nearly all of the functions and design elements designed in Section 3.2. We have followed the layout of the design closely as seen in Figure 4.2 and 4.3, with only a few changes. While implementing we changed the layout and some design elements to improve usability and readability. The main layout change was the in the counterfactual explanation section, where we set the two data tables that displayed the original data vs the counterfactual changes to fixed (could not be changed). This was because these changes are closely linked to the textual explanations and would also be useful to look at this information while analysing the graphs. They are also displayed virtually not horizontally, thus reducing horizontal scrolling and would be easier to read.

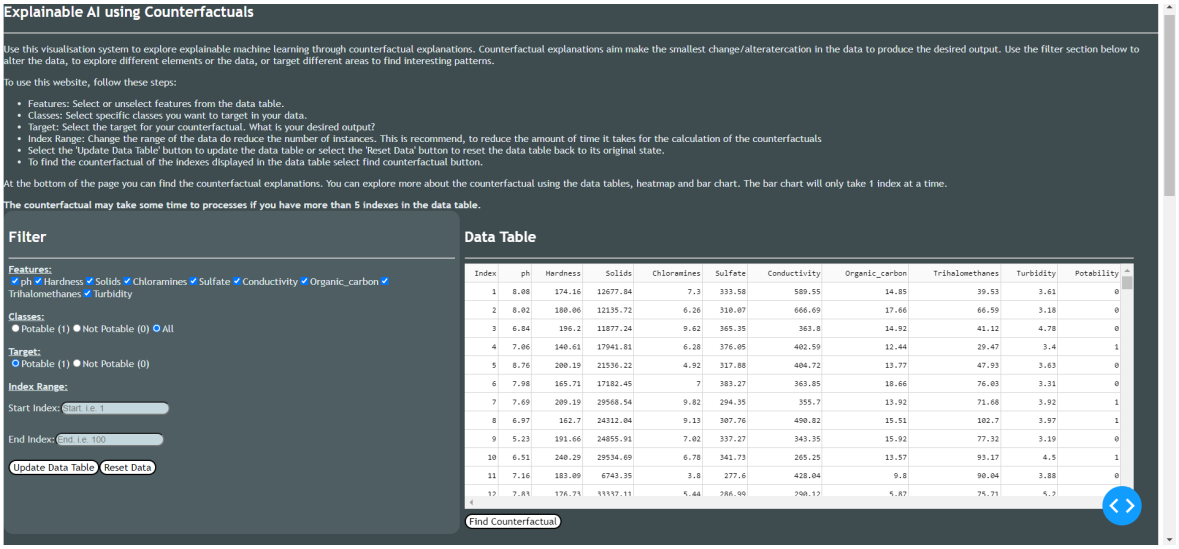

Figure 4.2: Layout of filtering section in CFE

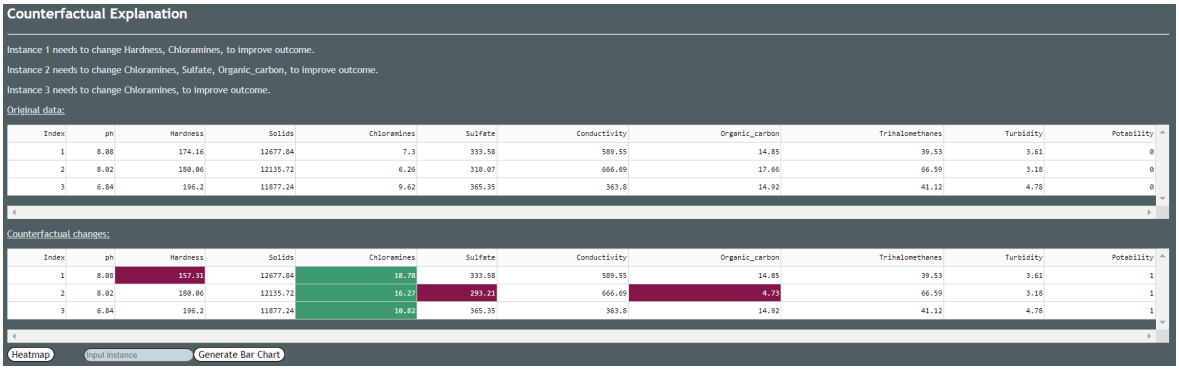

Figure 4.3: Layout of counterfactual explanations and graphs in CFE

The filter component uses a combination of different elements such as a button, radio button, checklist and text inputs. A checklist is used for the features sections to allow users to select different features to review. The checklist will allow the user to select multiple features to analyse. Whereas the classes and target filter sections use radio buttons, this will limit the user to select just one class. This will mean that the user can either select one class or all classes. The last filter section is the index range, we first used a range slider as we thought it would be a more effective way of changing the range. However, the dataset contains over 400 instances which made it difficult to accurately select the rows you want and the numbers would have to be too close together making it hard to read. Therefore, we used two text inputs, that were limited to numbers only within the dataset range. This allows users to adjust the index range and select the instances they want to review.

The data table is created on a scrollable area where only the first few rows are seen. This is so the data does not take up a lot of the page but the users can still view all the data.

4.5.1 Counterfactual Explanation

The implementation of the graphs was done using Plotly methods. Each graph has its own button that after clicking presents the graph.

The bar chart, Figure 4.5, displays the original value and the counterfactual value for

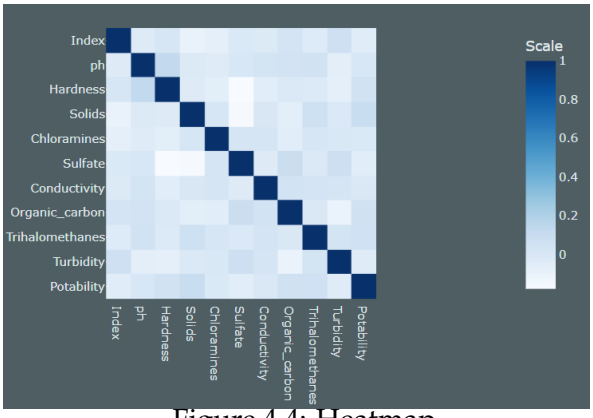

Figure 4.4: Heatmap

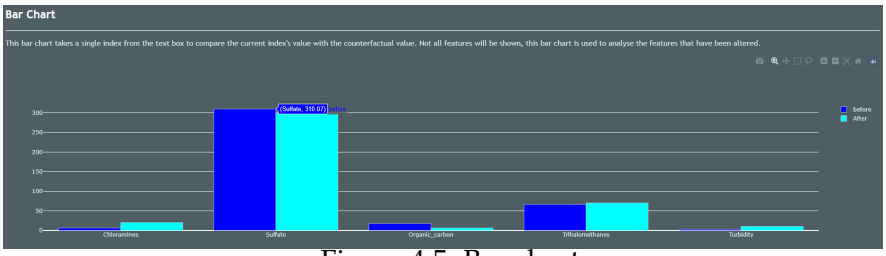

Figure 4.5: Bar chart

each of the altered features of the instance's counterfactual explanation. The features that were not changed are not displayed on the chart, as this would be redundant information. Currently, the bar chart is only developed for one instance at a time. So there is a text input where the user can submit a specific instance they want to review.

4.6 Requirements

The following Table 4.1, displays what requirements were met after completing the implementation

Requirement	**Description**	**State**
R1	Overview of counterfactual explanation	Yes
R2	Manipulate and filter data	Yes
R3	Analyse counterfactual explanations	Yes
R4	Highlight changes	Yes
R5	Import	No
R6	Accessible	Partially

Table 4.1: State of requirements after implementation

R6 is only partially implemented as we have implemented it on a website which makes it accessible (see Section 3.4), however, it is still being locally hosted which makes it unavailable to the public.

Chapter 5

Evaluation

We undertook an evaluation of the prototype, focusing on usability and algorithm robustness. By evaluating both of these aspects of the project, we can identify strengths and weaknesses that inform future work.

5.1 Usability Testing

5.1.1 Aim

User testing will identify any bugs/issues in the visualisation system with respect to gulf of execution (e.g. intuitiveness) and the gulf of evaluation (e.g. responsiveness and clarity) for the counterfactuals [23].

5.1.2 Method

Expectations of Usability Testing

The main expectation in this evaluation is for the participants to be able to filter/manipulate data and interpret the counterfactual explanations. However, due to the small amount of time they have in this testing, it is expected they will not be experts or fully able to understand the counterfactual algorithm. During the completion of the tasks, it is predicted that participants will find the filtering section comprehensive and will be able to distinguish counterfactual explanation changes. The qualitative and quantitative results from the testing are expected to display the strengths and weaknesses of the design and provide a means of showing how successful the visualisation of the explanations and data is.

Participants

We have gathered five students from level-400 VUW engineering to be testers for the projects. Only five participants were found for testing due to the scope of the project, however, it meets the standard that will provide good and effective insight into my evaluation [24]. While no major statistical analysis can be made on a group this small, key challenges and issues can be identified for evaluation and exploration. The following table presents each participant's attributes, the domain knowledge was done on a scale of 1(low)-5(high). Due to participants being 400-level students it was expected for their XML/counterfactual and water potability domain knowledge to be low. Each participant still fits into the persona because of their AI domain knowledge, however, their lack of domain knowledge does need to be considered.

Participants	Gender	AI Domain Knowledge	XML/counterfactual Domain Knowledge	Water Potability Domain Knowledge
P1	male	3	1	1
P2	male	3	1	2
P3	female	3	2	1
P4	female	3	1	1
P5	female	4	3	1

Table 5.1: The table displays the attributes of each of the participants in the evaluation.

Threat of Validity

Due to participants ranking low for domain knowledge for XML/counterfactual and water potability, there is a risk that these participants are not truly representative of the persona. This could mean that some results need to be taken at face value, as participants may express confusion when using the system due to their lack of domain knowledge. This can be seen later on with the dataset, as nearly all participants stated that their lack of understanding of the dataset did have an effect on the usability of the system.

Another threat to the validity of the usability testing were all the participants were friends and therefore may have tried to be nicer and provide more positive feedback than negative. Strangers would have been a better option for participants, however, within the scope of the project this was difficult.

Human Ethics Approval

Due to ECS gaining approval for usability testing for 400-level courses, no further approval is needed aside from a consent form, which can be found in Section A.1. All participants are 400-level engineering students which met the requirements. Also, potentially sensitive information will not be gathered from the participants. This includes not revealing the names of the participants within the report or to the supervisors, as they may be accessing one of these participants in a course.

Method

To ensure usability testing is effective, a plan was created, which includes tasks, a script and a questionnaire. Tasks were created using subsection 3.1.2. This ensured that we are testing all aspects of the system. There are nine tasks in total, that review the text, filtering functions and counterfactual explanation section. The script can be read in Appendix C.

The script was read aloud to the testers and created a structure for the testing. It contains the tasks and any questions we would like them to answer while using the system, if the participants were confused or unclear with any tasks or questions we would repeat or reword the task/question to help them complete the testing. These questions identify how they are feeling or what they may be struggling with. Lastly, there will be a questionnaire at the end of the user testing session.

These questions are Likert scale (rating), short answer and ranking questions. There are questions that ask the tester to rate different features and aspects of the system, such as the counterfactual explanations and how easy it was to understand. This is so that weaknesses in the system can be highlighted. For ranked questions, we ask the participant to elaborate on why they gave it that ranking, to understand the results more.

5.1.3 Pilot Study

A pilot study was completed at the beginning of my evaluation using one person, P1. This helped refine and test my usability method before testing on numerous participants.

P1 did not have any knowledge of water potability and the website CFE did not provide any clarity. This meant that he struggled to understand the meaning of the dataset. Therefore, in the script, we included an introduction to the dataset. This was done instead of adding an explanation of the data on my website because the website should not be hard coded to one dataset.

The pilot study tested my usability testing process, including all tasks and the survey. P1 identified some grammar issues that were fixed after the pilot study and found an issue with the filtering of the features which meant that after submitting the data to find a counterfactual explanation if you wanted to go back and change the features selected it broke the system. This was fixed shortly after the study. P1 completed the testing in around 20 minutes which was expected.

P1 did not struggle with the survey, however, after some review with my supervisors, we decided to change and improve it by rewording to make the questions more specific and adding some questions that were missed.

The following are P1's results from the survey:

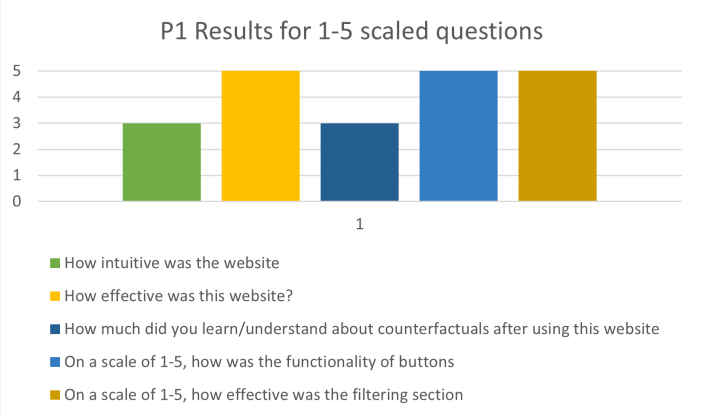

Figure 5.1: P1's results from survey

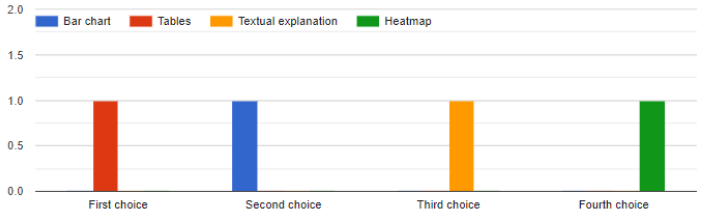

Figure 5.2: P1's ranking of counterfactual explanation display options

5.1.4 Results

While completing the tasks there was some common behaviour, 3 out of the 4 participants displayed confusion between the class and target options in the filtering section. Most were able to identify the differences after re-reading the introduction but P2 could not see the differences. Also, participants 2,3 and 4 did not input an instance before clicking the bar chart and had to be told why the graph wasn't being displayed. Even though it is stated in

the introduction, most participants did not remember or correctly read the text introduction, even though the first task in the testing was to read the text introduction.

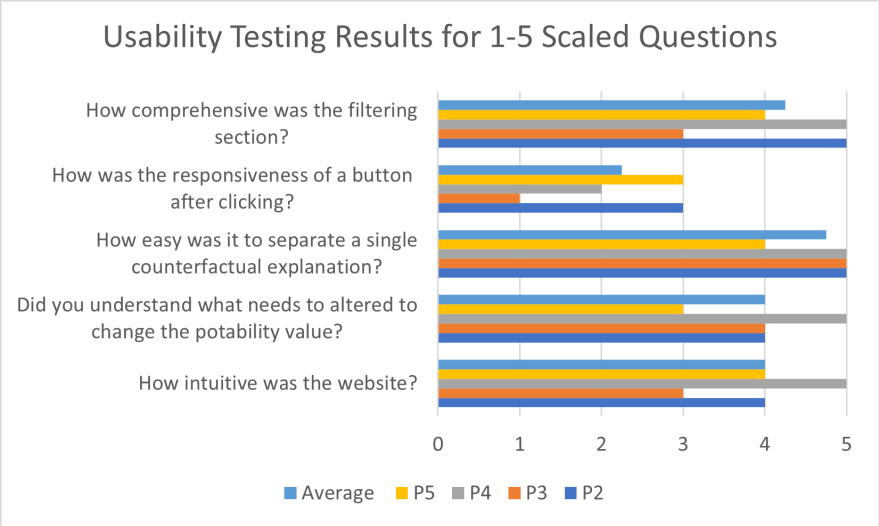

Figure 5.3: Results from survey

Figure 5.3 displays the results from the survey, for most of the questions the average result was between 4-5, which shows that the website was successful. However, the responsiveness of the buttons is highlighted as an issue within the website (scale average = 2.23). Participants (P2, P3, P4, P5) stated that this was because when they hit a button, nothing presented that the button had been pressed. Also, buttons often lagged after being selected due to the complexity of the algorithms and operations.

Most participants ranked intuitiveness (scale average = 4) highly, stating that it was easy to navigate the UI and find features and information. They also understood all the variables and changes that could be made. However, P3 ranked it at 3 stating that "the styling of it made it harder to visually interpret." Some participants also mentioned that their lack of domain knowledge about counterfactuals and water potability impacted their understanding of the system and intuitiveness.

All participants were able to identify and understand what needed to be altered to change the potability using the two tables and/or the textual explanation (scale average = 4). P3 did scale this question as a 3, however, they stated that this was due to not having a deep understanding of how these changes occurred and her lack of water potability knowledge.

Participants found that the comprehensiveness of the filtering section provided very good feature options (scale average = 4.25). A participant recommended that a filtering option in the feature selections should contain an 'All' option, so they don't have to re-select all the individual boxes. However, most participants identified an issue within the feature section, where when the data is reset, the filtering option does not change back to its original state. Meaning that if you unselected the first feature from the data and then hit the reset button, the data would reset but the first feature will remain unselected in the filtering section.

In the qualitative survey questions survey, half of the participants (P4 and P5) stated that the purpose of the website was introduced correctly and clearly, while the other half said it was a bit confusing in places (P2 and P3). This was because of the lack of knowledge on counterfactuals and water potability. P2 recommended changing the layout of the filtering section by separating the features, classes and range index, as these options filter the data,

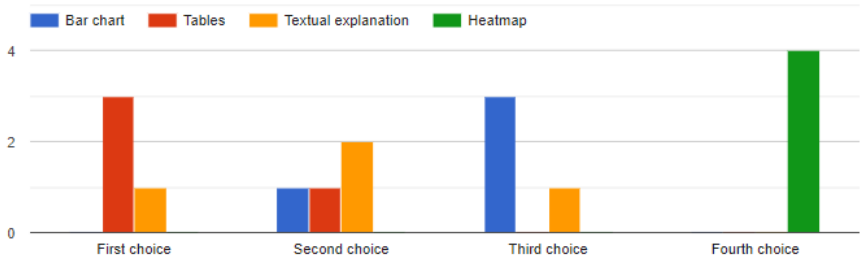

Figure 5.4: Ranking of group counterfactual explanation display options

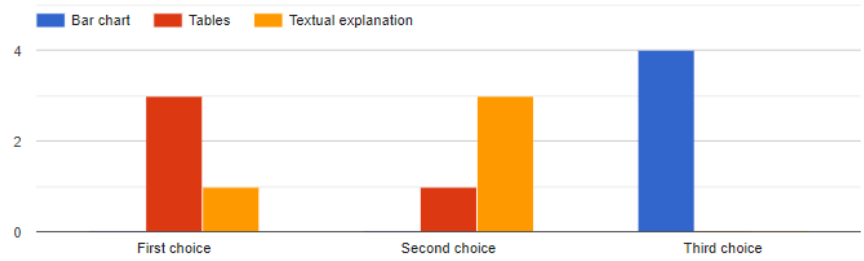

Figure 5.5: Ranking of single counterfactual explanation display options

from the target option, as this option filters the algorithm. P2, P3 and P4 advised improving the design by changing the font and text layout, changing the colours of buttons when they are clicked and changing the colour pallet. P4 and P5 highlighted that they could see the benefits of the website for ML researchers and that the idea contains a lot of merits.

Figures 5.4 and 5.5 show the results where participants were asked to rank the best display options for both displaying group and single counterfactual explanations. Both of these graphs are very similar and highlight that the tables were the best counterfactual explanation display as 75% of the time it ranked first and 25% of the time it ranked second for both single and group explanations. The second best was the textual displays which ranked second 50% of the time for group explanations and 75% for single explanations. Textual explanation ranked first 25% for both single and group explanations, and 25% for third for group explanations. Bar charts ranked third, ranking third 75% of the time for group explanations and 100% of the time for single explanations. The heatmap ranked fourth 100% of the time for group explanation and was not an option for single counterfactuals as the heatmap will likely not provide any information for single counterfactual explanations.

5.1.5 Discussion

Difficulty in parsing and retaining the information in the upfront text-based instructions is one possible reason for the confusion regarding the differences between classes and targets. This is not surprising as most people skip these kinds of big paragraphs of text. Therefore a better option for distinguishing between the target and class filtering options may be adding an information icon beside both options to explain each option. This will mean less scrolling as they will not need to go back to the top of the page to re-read the instructions, but users are more likely to read this as it is more convenient.

To improve the responsiveness of the bottom, the button can change colours when they have been clicked. This would be a very simple way to indicate that something has happened even if there is a lag in the system. We did have a look into improving this after finishing evaluation but found it difficult to implement with the current libraries and tools

we are using. This could be something to look into in the future. Also, breaking up the introduction and instructions at the top of the page into information icons could present information when the mouse if being hovered over it, this would make information more convenient. Popup dialogue could also indicate important information to grab the user's attention.

The textual explanation and the tables were shown as the best representations of the counterfactuals for both single and group counterfactuals. These results were not surprising for the group of counterfactuals display as the bar chart can only display one instance at a time (Figure 5.4). However, we expected the bar chart graph to rank high for display options for a single counterfactual (Figure 5.5), P5 stated that they ranked the textual explanation and tables high because they complemented each other well, and were easy to move between the two display options. The heatmap was ranked poorly, P2, P4 and P5 were confused with what this graph displayed and/or the correlations it had to counterfactuals. This may be due to the features not containing high and different correlation values therefore not fully representing the benefits of the heatmap.

5.2 Counterfactual Algorithm Testing

5.2.1 Aim

To evaluate the counterfactual algorithm different metrics and methods will be used to identify the effectiveness and accuracy of the system. We will use Euclidean distance and Manhattan distance alongside evaluating how many features are changed and how much the features have been modified, to check the performance and fidelity of the algorithm.

5.2.2 Method

The first metric is the Manhattan distance, which is used to calculate the shortest path between two points. This is done through the sum of the absolute differences of data point p_1 with co-ordinates at (x_1, y_1) and the nearest neighbor data point p_2, with co-ordinates at (x_2, y_2) [25].

$$Manhattan = |x_1 - x_2| + |y_1 - y_2| \tag{5.1}$$

Secondly, the Euclidean distance, which finds the distance of a segment that connects two points using equation 5.2 [26]. The two points used, are the same as the Manhattan distances

$$Euclidean = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{5.2}$$

Both the Manhattan and Euclidean distances are very similar equations. The main difference is that Manhattan distance is only permitted to move in straight lines, either vertical or horizontal lines. Whereas, Euclidean distance can move in diagonal lines. These matrices will be able to calculate the difference between the counterfactual explanation data and its original data point values based on the counterfactual output. This will identify how similar or different the two points are and the results can highlight issues with the algorithm. These issues can be identified through large distance values, as this can mean that there have been large changes within the feature values and/or many features have been changed. While this is not always an issue it can mean that the counterfactual algorithm may not be reducing the explanation correctly, as the feature selection step (Section 4.3) in the algorithm aims

Test ID	Index 1	Index 2	Index 3	Index 4	Index 5	Index 6	Index 7	Index 8	Index 9	Index 10
Test 1	5.014	2.391	0.7433	4.319	5.009	2.046	0.3842	0.8282	10.28	1.874
Test 2	27.71	18.7	2.674	5.664	11.42	2.33	0.8078	1.832	11.85	2.255
Test 3	8.213	16.81	0.401	2.726	5.253	1.835	0.6465	1.813	4.954	5.71
Test 4	6.648	13.04	2.133	1.108	15.6	2.137	0.942	3.518	7.452	14.97
Test 5	12.65	3.231	0.97	4.211	3.7	1.045	0.1884	2.79	1.48	4.9
Average	12.047	10.8344	1.38426	3.6056	8.1964	1.8786	0.59378	2.15624	7.2032	5.9418

Table 5.2: Euclidean distance per feature

Test ID	Index 1	Index 2	Index 3	Index 4	Index 5	Index 6	Index 7	Index 8	Index 9	Index 10
Test 1	9.215	4.924	0.933	4.319	8.719	2.763	0.656	1.12	13.45	2.767
Test 2	30.4	23.38	4.474	5.664	14.31	4.372	1.112	4.061	16.28	3.29
Test 3	10.38	22.7	0.519	3.726	12.12	3.102	0.647	2.179	7.775	5.71
Test 4	12.15	15.35	2.133	1.816	18.47	3.486	1.308	4.31	15.15	19.49
Test 5	15.57	5.025	0.97	4.211	4.435	1.684	0.256	2.79	2.492	4.9
Average	15.543	14.2758	1.8058	3.9472	11.6108	3.0814	0.7958	2.892	11.0294	7.2314

Table 5.3: Manhattan distance per feature

to reduce the changes. Another issue can also be identified through a fluctuation of the distances when repeating the counterfactual algorithm when using the same data point. This can show inconsistencies within the algorithm.

Lastly, by reviewing the number of features changed and how they were modified, we will be able to identify any unusual changes made to the features. Another way to achieve this is to apply the modified feature values to the model to check if the changes affected the outcome of the instance.

5.2.3 Results

Each table contains the values for either the Euclidean or Manhattan distance. We have tested the algorithm five times using the same ten instances/indexes, details on the ten instances are displayed in Appendix D. This is to check the consistency of the algorithm and identify any big changes that are being made. The small distance values identify that not a lot of changes have been made to the counterfactual, while bigger distance values show that there have been more changes i.e. a feature may have been increased/decreased by a large amount. Each distance has been divided by the number of features that were changed as some counterfactual explanations may have only one feature change while others have four.

Figure 5.6 displays the results from the Euclidean distance table above, where the bars indicate each index's test results and the green line shows the averages for each index. We have used a green line for the averages to easily compare the pattern the results are showing. Through this graph, we can see that there is fluctuation in some of the counterfactual explanations such as index 1, 2, 5, 9, 10. These same indexes showed the highest distance averages. The Manhattan distances display nearly identical patterns, with the same fluctuations. However, the results and averages are higher likely due to the equation using absolute values.

Figure 5.8 displays how many features were changed in each counterfactual explanation. These results are important to look at when reviewing this counterfactual algorithm as a part of the algorithm is to reduce the number of features changed. Therefore, we want the

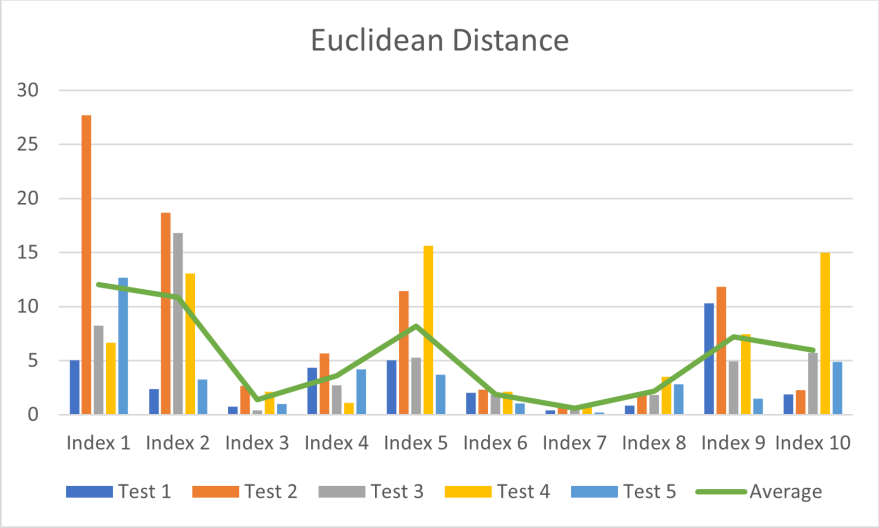

Figure 5.6: Euclidean distance

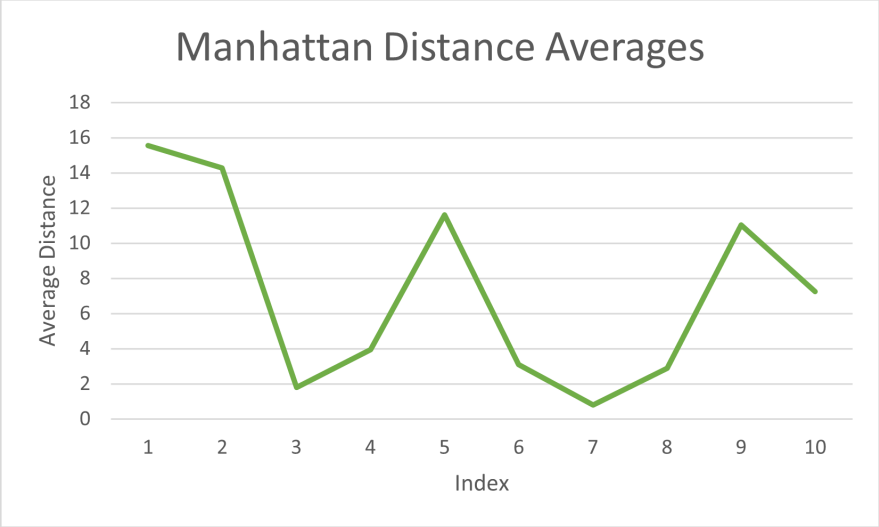

Figure 5.7: Manhattan distance averages

number of feature changes to be low. In the graph, the bars indicate each index's results and the green line shows the averages of all the tests for each index. From this we can see indexes like 1, 3, 4, 7 and 10 have pretty consistent feature changes as they either switch between two values and/or all values are close together. However, indexes like 5, 6, 7, 8 and 9 fluctuate.

5.2.4 Discussion

The results from the counterfactual algorithm testing (5.2.2) displayed some successful outcomes. This is evident through the averages displayed in both the distance and number of features changed graphs. When the tests were run, the target class was set to 1. This means that instances that had a class of 0, should have more changes that would need to be made. This can be seen in the results, for instance, a 'Potable' class of 0 has a mostly higher average for both distance and number of features changed.

Comparing the graphs, Figures 5.6, 5.7 and 5.8, we can see that there is a high correlation

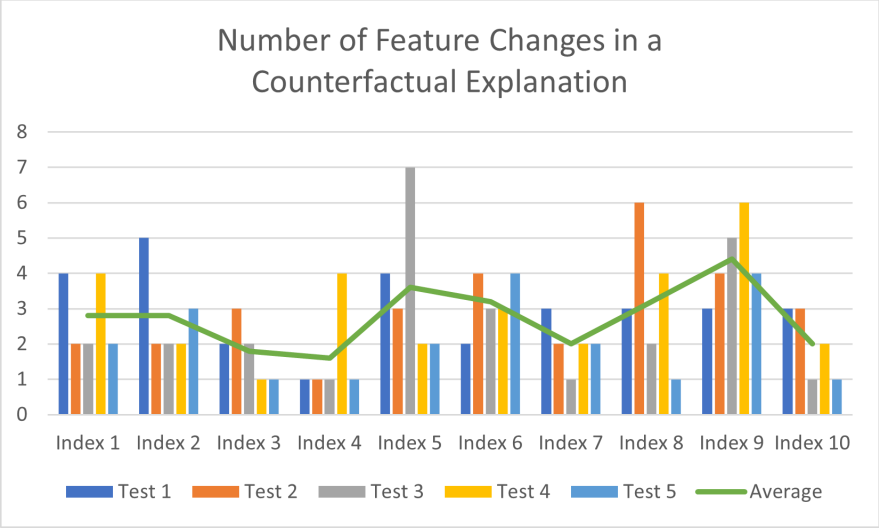

Figure 5.8: Number of feature changes

Index	ph	Hardness	Solids	Chloramines	Sulfate
5	6.89	210.43	21536.22	17.56	317.88
6	7.59	165.71	17182.45	11.81	388.07

Figure 5.9: Heatmap from the website

between the distance averages and how many features were changed. Highlighting that the more features that need to be changed in an instance, the greater the distance between the counterfactual explanation and the original distance. This is not surprising and was expected.

However, the evaluation did highlight some inconsistencies with the algorithm. Half of the indexes display fluctuation in Figures 5.6 and 5.8. In Figure 5.6, index 1 has the greatest distance difference within an instance with its lowest value of 5.014 and its highest value of 27.71, these values have a difference of 22.696.

Another interesting observation found in testing was that the feature 'Chloramines' was always changed in the counterfactual explanation. No investigation has been done into this so it is unclear if this change is always necessary or if there is an issue within the code.

An interesting finding found using the heatmap on the website Figure 4.4, was the lack of correlation between the features. The highest correlation identified, by looking at the shade and the scale value provided by hovering over the square is the diagonal line that has a scale of 1. However, we should ignore this line as it is just comparing the feature against itself, therefore, is redundant and irrelevant information. Excluding the diagonal line, the highest correlation is between ph and hardness with a value of 0.128. The lowest relationship is between hardness and sulphate. This relationship can be seen in the counterfactual in figure 5.9, because with index 5, ph and hardness have been changed together, however in index 6 when sulfate is altered hardness remains the same.

Chapter 6

Conclusion

6.1 Summary of Key Takeaways

We successfully designed a tool that has fulfilled all requirements (Table 3.1) and implemented a website, based on our design, that fulfilled most of the requirements (Table 4.1) The tools is able to provide visualising and exploring counterfactuals available to people.

The most important aspects that were considered when designing this tool were data filtering and manipulation and the display of the counterfactuals. Many design choices were made around what techniques should be used for the data visualisation as this was one of the most important and difficult parts of displaying the counterfactual explanations. Many ideas for graphs and techniques were thought/researched, however, we narrowed it down to a combination of tables, bar charts and a heatmap. Through the evaluation, we can see that this was able to provide good explanations, however, the decision to use these graphs partially centred around data collection and the scope of the project.

We were able to implement most of the requirements within the proof-of-concept prototype, including filtering/manipulation of the data (R2), the different counterfactual displays (R1, R3, R4) and partially R6 for accessibility. However, we were unable to implement the import function. The implementation of this function could have been easy to implement if we did not start this project by focusing on one dataset and implemented this function earlier on. Focusing on only one dataset, meant that some areas were unintentionally hard code. Even though we spent some time changing this, so that the code was not fixed to one dataset, we ran out of time to implement this function.

The usability testing did highlight an issue with unappealing aesthetics such as bad choices in background colour and style. This was due to the scope of the project. When designing the tool not a lot of focus was put on the colour palette and text (font and size) as they were not considered important aspects at the time. However, this ended up taking away from the UI and usability of the website. This could have been improved upon by focusing on providing good aesthetics in the design stages, rather than just looking into the layout and functionalities of the tool. By improving the balance of the design stage, it would have been quicker to implement better UI and usability through these improved aesthetics. These issues could have also been identified and improved upon if usability testing was done easier, even though the prototype would not have been completed, we could have provided a mock-up of the prototype through Figma to get feedback on the style, colour and UI.

The results of the counterfactual algorithm evaluation showed fluctuations in the distance matrix. However, overall the evaluation showed the algorithm worked correctly.

Next time, something that should have been taken into big consideration is the scope of the project. However, overall the project showed positive and successful outcomes. The

prototype presented how a counterfactual algorithm can be used to interpret data and alter the outcomes of an individual data point and how graphs can be used to display and analyse these counterfactual explanations.

6.2 Future Work

Through the evaluation of the tool, issues and ideas were highlighted that could be completed in the future.

The current design of the data visualisation provides an effective analysis of single counterfactual explanations but the usability testing showed a lack of proper tools for exploring groups of counterfactuals. The heatmap was supposed to be able to provide more information to explain the explanations, however, it ranked the lowest. There were other graphs that could provide better and more interesting visualisations for exploring groups of counterfactual explanations but we would have struggled to complete a prototype of the tool with the scope (Section 3.3.1). In the future, more research could be done into graphing groups of counterfactuals.

Other recommendations that could improve the tool include: improving the aesthetics of the design (layout, style, colour etc), this could include, adding information icons beside each filter option, brightening up the colour palette, and changing text font to make the text more readable; implementing the import function (R5) to improve the accessibility (R6) and usefulness of the system; creating a more stable and robust counterfactual algorithm to improve the counterfactuals performance.

In the CFE website, it uses a fixed black-box model (random forest), however, counterfactuals are model-agnostic (Section 2.3.1), which allows the algorithm to work on any black-box model. In the future, we could implement a function that allows users to select different black-box models as this would allow for better analysis of the data and counterfactual algorithm. Also later extending the website to offer more counterfactual algorithms, to allow for analysis between the different algorithms.

The evaluation did present an issue around the responsiveness of the buttons in the system. This is greatly due to the tools and libraries used to implement this system. While python was a good language to use for ML and XML algorithms and methods. It did not provide the best libraries and tools for implementing a web application as the Plotly dash library did not contain an easy way to implement UI feature such as the button responsiveness. Therefore, an analysis of the tools used in the project would be an interesting and enlightening task to complete in the future.

6.3 Key Achievements

We have implemented a proof-of-concept prototype that uses counterfactuals to explain why a Machine Learning system reached a particular decision The system uses information visualisation techniques and targets users without a deep understanding of machine learning

We evaluated my prototype using two different techniques: usability testing to find strengths and weaknesses in the user interface from the perspective of the target users; distance matrices to ensure the core counterfactual algorithm was correct. The evaluations demonstrated users could successfully explore appropriate counterfactuals and highlighted some areas of the user interface that can be improved in subsequent versions of the system

Bibliography

[1] S. Sharma, J. Henderson, and J. Ghosh, "Certifai: Counterfactual explanations for robustness, transparency, interpretability, and fairness of artificial intelligence models," 2019.

[2] F. Xu, H. Uszkoreit, Y. Du, W. Fan, D. Zhao, and J.Zhu, *Explainable AI: A Brief Survey on History, Research Areas, Approaches and Challenges*. Springer and Cham, 2019.

[3] C. Molnar, *Interpretable Machine Learning*. 2 ed., 2022.

[4] L. Gilpin, D. Bau, B. Yuan, A. Bajwa, M. Specter, and L. Kagal, "Explaining explanations: An overview of interpretability of machine learning," 2019.

[5] "Decision trees,"

[6] J. Wong, "Decision trees," 2021.

[7] Y. Dong, H. Su, J. Zhu, and B. Zhang, "Improving interpretability of deep neural networks with semantic information," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 975–983, 2017.

[8] D. Zheng, W. Zhang, S. N. Alemu, P. Wang, G. T. Bitew, D. Wei, and J. Yue, *Short-term renewable generation and load forecasting in microgrids*, ch. 4.4.3.4. Science Direct, 2021.

[9] T. Laugel, M.-J. Lesot, C. Marsala, X. Renard, and M. Detyniecki, "Inverse classification for comparison-based interpretability in machine learning," 2017.

[10] R. Greenlaw, H. J. Hoover, and W. L. Ruzzo, "Greedy algorithms," 2020.

[11] R. K. Mothilal, A. Sharma, and C. Tan, "Explaining machine learning classifiers through diverse counterfactual explanations," 2019.

[12] A.-H. Karimi, G. Barthe, B. Balle, and I. Valera, "Model-agnostic counterfactual explanations for consequential decisions," 2020.

[13] J. Heer, M. Bostock, and V. Ogievetsky, "A tour through the visualization zoo," *acm queue*, vol. 8, no. 5, 2010.

[14] M. S. T. Carpendale, "Considering visual variables as a basis for information visualisation," *University of Calcary*, 2003.

[15] B. Shneiderman, "The eyes have it: A task by data type taxonomy for information visualizations," *Proceedings 1996 IEEE Symposium on Visual Languages*, pp. 336–343, 1996.

[16] R. M. Byrne, "Counterfactuals in explainable artificial intelligence (xai): Evidence from human reasoning," *International Joint Conference on Artificial Intelligence (IJCAI-19)*, pp. 6276–6282, 1970.

[17] "Use cases," *GitLab*.

[18] B. Shneiderman, "The eyes have it: a task by data type taxonomy for information visualizations," *Proceedings 1996 IEEE Symposium on Visual Languages*, p. 336–, 1996.

[19] E. S. Chukwuemeka, "Best programming languages for artificial intelligence 2022: Top 10," 2021.

[20] A. Smith, "Python vs java — which is more compatible for web development?," *medium*.

[21] T. Laugel, "Growing spheres," 2022.

[22] A. Kadiwa, "Water potability," 2021.

[23] *Gulf of Evaluation and Gulf of Execution*. The Glossary of Human Computer Interaction.

[24] J. Nielsen, *Usability Engineering*, ch. 6. Elsevier Science & Technology, 1994.

[25] G. R. Sahani, "Euclidean and manhattan distance metrics in machine learning.," *Medium*, 2020.

[26] "Euclidean distance formula," *CueMath*.