

INT2X: EXPLANATION FOR THE CAUSES OF A PREDICTION

23-142

Britto T.A

IT20100698

B.Sc. (Hons) Degree in Information Technology Specialized in Data
Science

Department of Information Technology

Sri Lanka Institute of Information Technology
Sri Lanka

September 2023

INT2X: EXPLANATION FOR THE CAUSES OF A PREDICTION

Britto T.A

IT20100698

Dissertation submitted in partial fulfillment of the requirements for the Special
Honors Degree of Bachelor of Science in Information Technology Specializing in
Data Science


Department of Information Technology

Sri Lanka Institute of Information Technology
Sri Lanka

September 2023

DECLARATION

I declare that this is my own work, and this proposal does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any other university or Institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Name	Student ID	Signature
Britto T. A	IT20100698	

The above candidate is carrying out research for the undergraduate Dissertation under my supervision.

.....

.....

Signature of the Supervisor:

Date

ABSTRACT

In recent years, the use of AI-driven decision-making applications has increased significantly, and many of these applications involve making critical decisions that have a significant impact on human lives. Even though AI-driven systems are automated and easier to use, their lack of transparency and interpretability has become a major drawback. As a result, many machine learning models that are used for decision-making processes are black box, which means the internal behaviors of the models are more complex and difficult for humans to understand. Explainable Artificial Intelligence is an emerging area of research that aims to develop AI systems that can provide explanations for their decision-making processes. The use of ensemble models, especially Random Forest, in text classification is common because they are effective. However, understanding how they make decisions can be challenging. This research focuses on making these models easier to understand. This study introduced a novel model-specific, local XAI solution to enhance the model's explainability by developing a counterfactual explanation-based XAI solution. That uses the feature importance in the Random Forest model and provides an instance-specific counterfactual explanation in the text classification domain. Initial results show that this approach helps to understand the model's decisions better. This method looks at the feature importance of individual words in the text to see how they impact the model's decision and provides examples to show what changes could lead the model to make a different decision. This research can help those using text classification models get more insight into their results.

Keywords – XAI, Counterfactual explanation, Instance-specific, Feature importance

ACKNOWLEDGEMENT

Regarding the final year project involving an explanation for the causes of a prediction (INT2X), Britto T.A. crafted this particular paper. This project is the result of the hard work put forth by each member of the team as well as the encouragement, support, and direction provided by numerous others, in particular our supervisor Mr. Prasanna Sumathipala, and our co-supervisor Mr. Jeewaka Perera, other SLIIT institute lecturers who provided advice and guidance in turning this creative solution into a reality, and our dearest batchmates who assisted by lending a helping hand when it was necessary. A heartfelt expression of gratitude is due to all of our family members, friends, and colleagues for the unwavering support, care, and invaluable assistance that they have provided. I'd also want to use this opportunity to convey my appreciation to everyone else who has helped me or motivated me to make this a success but whose name isn't featured here. Throughout my whole academic career, I shall be eternally thankful to our previous teachers for the consistent support they provided.

Table of Contents

DECLARATION	3
ABSTRACT	4
ACKNOWLEDGEMENT	5
List OF Figure	7
List Of Table	8
LIST OF ABBREVIATIONS	9
Amazon Web Services.....	9
Natural Language Toolkit	9
1. INTRODUCTION	10
1.1 Background	10
1.2 Literature Survey	13
1.2.1 Nearest Instance Counterfactual Explanations [9]	15
1.2.2 Diverse Counterfactual Explanations [10]	16
1.2.3 Sequentially Eliminating Discrediting Counterfactuals (SEDC) & SHAP-C [11]	16
1.2.4 LIME-C [11]	17
1.3 Research Gap	18
1.4 Research Problem	21
1.5 Research Objectives	22
1.5.1 Main Objective of the Research Components	22
1.5.2 Specific Objectives	23
2. METHODOLOGY	24
2.1 Overall System Architecture	24
2.2 Counterfactual Explanation for Random Forest Model.	25
2.3 Testing & Implementation	30
2.3.1 Implementation	30
2.3.2 Testing	41
3. RESULTS & DISCUSSION	43
3.1 Results	43
3.1.1 Result Evaluation for the Negative reviews:	43
.....	46
3.1.2 Result Evaluation for the Positive reviews:	46
3.2 Discussion	50
4. SUMMARY	52
5. CONCLUSION	55
6. RERERENCES	56

List OF Figure

Figure 2.1- 1: System Architecture diagram	24
Figure 2.2- 1: Working process of Random Forest.....	26
Figure 2.2- 2: Equation of Gini Index.....	27
Figure 2.2- 3:Equation of Gini importance	28
Figure 2.2- 4:Equation of Average importance	28
Figure 2.3.1- 1:Model training.....	31
Figure 2.3.1- 2:Remove HTML tags.....	33
Figure 2.3.1- 3:Remove special characters.	33
Figure 2.3.1- 4:Remove stopwords.....	34
Figure 2.3.1- 5: LUTLabelEncoder.....	35
Figure 2.3.1- 6:TextVectorizer.....	35
Figure 2.3.1- 7:Required libraries.	36
Figure 2.3.1- 8: Import dataset and RF model.	36
Figure 2.3.1- 9:Novel Method implementation 1	37
Figure 2.3.1- 10:Novel method implementation 2	38
Figure 2.3.1- 12:Novel method implementation of class change process 2.....	39
Figure 2.3.1- 11: Novel method implementation of class change process 1	39
Figure 2.3.1- 13:Final output	40
Figure 3.1.1- 1: Class change for negative review 1 RF explanation Vs SEDC.....	44
Figure 3.1.1- 2:Class change for negative review 2 RF explanation Vs SEDC.....	45
Figure 3.1.1- 3: Class change of negative review 3 RF explanation Vs SEDC	46
Figure 3.1.1- 4:Class change of negative review 4 RF explanation Vs SEDC	46
Figure 3.1.2- 1: Class change of positive review 1 RF explanation Vs SEDC.....	47
Figure 3.1.2- 2: Class change of positive review 2 RF explanation Vs SEDC.....	48
Figure 3.1.2- 3: Class change for positive review 3 RF explanation Vs SEDC.....	49

List Of Table

Table 2.3.2- 1:Test case to check the performance of the novel XAI method.....	42
Table 4- 1: Summary of each member's contribution	54

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
GDPR	General Data Protection Regulation
XAI	Explainable Artificial Intelligence
RF	Random Forest
SHAP	Shapley Additive Explanations
LIME	Local Interpretable Model-Agnostic Explanations
AIX360	AI Explainability 360
NICE	Nearest Instance Counterfactual Explanations
DICE	Diverse Counterfactual Explanations
SEDC	Sequentially Eliminating Discrediting Counterfactuals
TF	Term Frequency
IDF	Inverse Document Frequency
AWS	Amazon Web Services
NLTK	Natural Language Toolkit

1. INTRODUCTION

1.1 Background

At the dawn of the fourth industrial revolution, different domains are witnessing a fast and widespread adoption of artificial intelligence (AI) in the decision-making process under domain experts. Healthcare, Legal, military, transportation, and finance are some domains where AI models are used for the decision-making process. Decisions made by the AI model are extremely critical in a situation like a medical diagnosis of a disease. It's crucial that both clinicians and patients can understand and validate the reasoning behind AI-driven diagnoses. Here explainable AI (XAI) provides clear, understandable insights into how an AI model arrives at its conclusions. After an AI model makes a prediction (e.g., the likelihood of a patient having a certain disease), it can present the relative importance of each input feature (e.g., patient symptoms, test results) that contributed to the prediction. Also, through the explanation, doctors can get an idea of how the outcome would be different when input features slightly changed. There are some situations in which the models can be biased and provide inaccurate results. In such situations model explainability is well needed to identify and avoid errors in decision making.

There is no exact definition for Model Explain ability/Interpretability and different authors prefer to use the most sensible definitions. The most popular and widely used definitions are “Interpretability is the degree to which a human can understand the cause of a decision” [1] and “Interpretability is the degree to which a human can consistently predict the model's result”[2]. The main objective of XAI is to answer the "wh" questions related to AI-based decision-making. For example, XAI should be able to answer, "why a particular answer was obtained? ", "how a particular answer was obtained" and "when a particular AI-based system can fail?"[3]. Explainability can be applied to a variety of tasks in a variety of fields, including justifying, controlling, improving, and discovering [3].

a. Explain to justify.

Some situations in AI/ML-based systems provide biased or discriminatory results. Therefore, the explanation of AI-based results is essential. The XAI systems provide the required information to justify the result when unexpected decisions are made.

b. Explain to control.

During the explainability process, users gain insights into the system's behavior. Greater visibility of undiscovered vulnerabilities and flaws is provided, which makes mistakes easier to spot and quicker to fix.

c. Explain to improve.

A model that can be explained can be improved easily. This is because users comprehend the relationship between input variables and the resulting output, and how to make the output smarter.

d. Explain to discover.

Explanation about the process is helpful to learn new facts, collect information, and gain knowledge. XAI models are useful to find new and hidden laws in various scenarios.

This research mainly focuses on enhancing the model explainability of Ensemble based classification models focusing on Random Forest. It provides a novel explainable method that is related to a text classification. The black box model refers to a complex computational algorithm or system that processes inputs to produce outputs, but the internal workings or mechanisms that are responsible for the transformation are not readily understandable or explainable from an external perspective.

Random forest is a supervised learning technique, that is used for classification, and regression tasks. It is an ensemble method that consists of multiple decision trees, each built on different subsets of the given dataset. By averaging their individual predictions, the Random Forest model improves the overall predictive accuracy of the dataset. Instead of relying solely on a single decision tree, the Random Forest model considers the predictions from each tree and selects the final output based on the majority votes of those predictions, thereby ensuring a more robust and accurate decision-making process. Increasing the number of trees in a forest contributes to improved accuracy and helps avoid overfitting issues. In this research provides an XAI solution for the text classification domain using Random Forests, enhancing interpretability, and understanding of the model's decisions.

The random forest model can become a black box when it has a large number of decision trees with complex interactions between the features. It is difficult to understand the overall decision-making process because the final decision is based on the output of many decision trees.

There are many explainable techniques that can be used to provide explainable solutions for Black-Box models. This is where the idea of "counterfactual explanation" comes in. A counterfactual explanation provides insight into a machine learning model's prediction by illustrating a hypothetical scenario wherein specific feature values are altered to achieve a different, desired outcome. In essence, it answers the "what if" questions by demonstrating the minimal changes needed to reverse a prediction. Counterfactual analysis assists to detect and mitigate bias in text classification models by generating hypothetical scenarios where a particular protected attribute (such as race or gender) is changed. Here we can observe whether the model's output is affected by that attribute. If the model's output changes significantly when the protected attribute is changed, it may be a sign that the model is biased. Also, this Counterfactual analysis can be used to understand why a particular text classification model is making certain errors. As the previous bias detection solution, it generates hypothetical scenarios to identify where the input is slightly changed to cause model misclassification. By observing what aspects of the inputs are causing the error we can identify areas for improvement in the model. Furthermore, this counterfactual analysis can be used to evaluate the fairness of text classification models. The rules can identify where certain groups are overrepresented or underrepresented and if the model is affected by the group differences.

In this research, we focused on a special way to explain decisions made by the Random Forest model. This means our method is "model-specific" because it's made just for this type of model and might not work the same way for other models. When we say our method is "local", we mean it looks at individual pieces of data (like a single piece of text or one decision) rather than trying to explain everything at once. Think of it like looking closely at one tree in a forest instead of the whole forest. This way, we can give clear reasons for a particular decision the model made for that specific piece of data. By using our method, people can get clearer answers about why the model made a certain choice for a specific case. This makes it easier to trust and use the model.

1.2 Literature Survey

XAI is dedicated to demystifying the black boxes by properly explaining the internal process of AI/ML models. It improves the trustworthiness, transparency, and responsibility of AI-based decision-making processes. Since the XAI research area is still in the starting era most of the techniques are in the research phase and researchers have built communities for contributions to the XAI domain.

Some of the proposed techniques have gained popularity among the research community, and new contributions are being made on top of these techniques. LIME (Local Interpretable Model Agnostic Explanation) [4], SHAP (SHapely Additive exPlanations) [5], XAI360 [6] and Google XAI [7] are examples of that. When we consider LIME, it interprets the model and explains the classification of the model in a faithful manner. It provides local optimum explanations which compute the important features by generating samples of the feature vector. Those samples follow a normal distribution. After getting the predictions from the samples it assigns weights to each of the rows to get an idea of how close they are from the original sample. Then LIME uses a feature selection technique to identify the most significant features.

SHAP is a unified approach that has been developed based on coalition game theory [5]. In that theory, they assign a reward to the game players according to their contributions to the game. SHAP assigns a feature importance value for each feature that affects a particular output result. It maps the input features to the output results based on that Shapley value. The key difference between LIME and SHAP is in the way that they assign weights to the input features. LIME uses a cosine measurement, while SHAP uses the Shapley formula. Another significant advantage of SHAP over other methods is its adaptability. SHAP can be used for any machine learning model. Though initially more popular with tree-based models due to ease of calculation, kernel and deep learning versions of SHAP have also been developed. This adaptability has contributed to its widespread acceptance and usage in the data science community. Furthermore, SHAP also has an additive feature. The sum of SHAP values over all features is equal to the difference between the model's prediction and the mean prediction for all data points. This additive nature ensures that each feature's contribution can be directly quantified and compared, providing clear interpretability.

The AI Explainability 360 (AIX360) toolkit is an open-source library developed by IBM Research [6]. It provides a comprehensive suite of algorithms and resources to promote explainable AI (XAI) in machine learning models. By offering a variety of techniques, the toolkit caters to different users, ranging from data scientists to domain experts and end-users. The AIX360 toolkit includes several explainability algorithms, each with its unique approach to generating explanations. These algorithms can be broadly categorized into local and global explanation techniques, as well as model-specific and model-agnostic methods. Local explanation techniques focus on providing insights into individual predictions, whereas global techniques aim to offer a broader understanding of the entire model. The toolkit also provides a range of resources, including tutorials, example code, and detailed documentation to help users understand and apply these explainability techniques to their machine-learning models. By making these resources readily available, the AIX360 toolkit aims to facilitate the adoption of explainable AI practices and to encourage the development of transparent, accountable, and trustworthy AI systems. Overall, the AIX360 toolkit is a valuable resource for researchers, practitioners, and stakeholders interested in implementing explainable AI in their machine-learning models. The toolkit's diverse set of techniques and resources makes it a versatile solution for addressing the growing need for transparency and accountability in AI systems.

In this research paper, they have introduced a novel model-specific methodology that explains an approach to enhance the explainability and interpretability of tree-based models, such as decision trees and Random Forests. The authors address the growing need for improved interpretability and transparency in tree-based machine-learning models. Tree-based models, such as Decision Tree and Random Forest, have gained popularity due to their robust performance, but often suffer from limited explainability. To tackle this challenge, Lundberg and his colleagues introduced a model-agnostic method called SHAP (Shapley Additive Explanations) that provides local explanations for individual predictions made by tree-based models [8]. The paper discusses the importance of explainable AI (XAI), especially in the context of critical decision-making processes that utilize machine learning models. The authors demonstrate the application of SHAP values to decision trees, Random Forests, and gradient-boosted trees, offering a valuable contribution to the field of XAI. SHAP values help users comprehend the logic underlying the model's decisions by quantifying the contribution of each feature to the prediction for a particular instance. In addition

to local explanations, the paper extends the SHAP framework to enable a global understanding of tree-based models. By aggregating SHAP values across multiple instances, the authors present a method to identify the most important features and understand the overall structure of the model's decision-making process. Various visualization techniques are also discussed for interpreting the model behavior at a global level. This paper by Lundberg et al. represents a significant advancement in the field of XAI, specifically in providing explanations and interpretability for tree-based models. Their work has the potential to improve transparency, trust, and fairness in machine-learning applications that use decision trees, Random Forests, and other tree-based models.

An existing method called Google XAI (Explainable AI) [7] aims to create methods and tools that will make AI more understandable and transparent. This approach offers local and global explanations and is also a model-agnostic method. Testing with Concept Activation Vectors (TCAV), integrated gradients, and attention mechanisms are some of the explainable techniques developed by Google XAI. By analyzing the activations of various neurons in the deep learning architecture of the model, TCAV provides a method for understanding how models make decisions. Integrated Gradients calculate the importance of input features by integrating the gradients of the model's output concerning the input features. Furthermore, attention mechanisms are used in deep learning architectures to identify which parts of an input are most important for the model's output.

1.2.1 Nearest Instance Counterfactual Explanations [9].

A counterfactual explanation method called NICE (Nearest Instance Counterfactual Explanations) aims to identify the smallest and most significant changes to a given instance that could change the outcome predicted by the model. Compared to other counterfactual explanation techniques, this method aims to make explanations more understandable and useful. The algorithm considers the feasibility of the counterfactuals generated, ensuring that they are practically possible and not merely hypothetical. Additionally, NICE creates explanations that are simpler and easier for human users to understand. Users can understand why the model made a particular prediction and what would need to be done to change that prediction.

1.2.2 Diverse Counterfactual Explanations [10].

DICE generates a diverse set of counterfactual explanations, by providing insights into how slight changes in the input can alter the model's prediction. The concept of Counterfactuals requires imagination of hypothetical realities that can happen other than the existing situation. In this research, to understand the concept of Counterfactuals it has given a real-time scenario that can happen in day-to-day life. It will be discussed below. Consider the case of a student applying for a prestigious university scholarship and being denied. Using interpretable models, the university can explain the rejection. For instance, the reason could be due to a “lower GPA” than required. From the student's perspective, simply knowing that the GPA was too low doesn't provide a concrete path forward. The student might be left wondering, “What GPA did I need to be considered? What other factors contributed to the decision? “In cases where some features, like the number of extracurricular activities, might not be the main determinant for scholarship selection, it's crucial to provide alternative, actionable recommendations. Here's a counterfactual explanation:

“You would have been eligible for the scholarship if your GPA was 0.5 points higher, or if you had participated in two more academic competitions.”

With this information, the student can identify not only the areas of improvement but also measure the weight of each factor. They can then choose to either work on boosting their GPA or participate in more academic activities in the future.

1.2.3 Sequentially Eliminating Discrediting Counterfactuals (SEDC) & SHAP-C [11].

The SEDC method identifies counterfactual explanations in textual data. By treating each word as a distinct feature, the method iteratively removes features to shift in the model's predictions. Features that notably alter the predictions upon removal are considered highly influential for the initial classification. Expanding on this, Ramon et al. (2020) developed the SHAP-C method, blending the strengths of Shapley Additive Explanations (SHAP) and SEDC. Using SHAP's game theory foundation, it determines feature importance and subsequently, the SEDC method crafts counterfactual explanations by sequentially omitting crucial features.

1.2.4 LIME-C [11].

Another intriguing approach in the realm of counterfactual explanations is LIME-C, an extension of the widely recognized LIME methodology just for text. LIME, known for elucidating individual predictions by approximating complex models with simpler interpretable ones, finds its counterfactual counterpart in LIME-C. This method utilizes the core principles of LIME to comprehend a model's decision boundaries and then pinpoints the alterations or counterfactuals needed to modify the model's original prediction.

1.2.5 P-type Contrastive explanations [12].

Shubham Rathi has attempted to generate Partial Post hoc P-type Contrastive explanations [9] and corresponding counterfactual data points by illustrating the specific changes required in data to attain a desired output. This methodology addresses classifier's prediction for a given datapoint which serves as reference. These P-contrast questions take the form 'Why [predicted-class] not [desired-class]?' It allows for a focused exploration of a single alternative by specifying the desired class. Moreover, Shapley values are used to provide contribution of individual features for a classification of target class. The negative Shapley values indicate the features that have negatively contributed to the specific class classification and vice versa. According to the definition of counterfactuals by [Molnar, 2018], we change only the features that work against the classification of the desired category and generate counterfactual datapoints. So those points are the answers to user's contrastive query. This is the only unique and globally consistent method for generating contrastive and counterfactual explanations.

1.3 Research Gap

The main target of this research is to make the Random Forest model more interpretable related to the text classification domain. We can use a counterfactual rules generation mechanism, which proposes the changes that need to be done in the input features to flip the final predictions to achieve the Random Forest model explainable task.

When we consider existing tools like SHAP, their computations are more complex, especially for models with a large number of features or complex interactions [5]. This can make it unsuitable for real-time or large-scale applications. Furthermore, SHAP assumes that features are independent, which is often not the case in real-world datasets. SHAP is a game theory-based method for explaining individual predictions of machine learning models. It does not directly provide counterfactual explanations, but it does provide a way to understand how each feature contributes to a specific prediction, either positively or negatively, in comparison to a baseline. Although SHAP values provide detailed insights into feature contributions for individual predictions, they do not directly answer the "what if" questions posed by counterfactual explanations. There are some weaknesses in LIME as well [4]. Because of the random sampling process used in the local surrogate model fitting procedure, LIME explanations can be unstable, which means that small changes to the instance can result in different explanations. Diverse Counterfactual Explanations are designed to generate counterfactual explanations for any machine learning classifier, but the process of generating diverse counterfactuals can be computationally demanding, and it also assumes that the features are independent, which may not be the case in many real-world datasets [10]. TreeSHAP is a SHAP variant designed specifically for tree-based models such as random forests and gradient-boosting machines [8]. TreeSHAP provides feature importance and contributions but does not inherently provide counterfactual rules by default. Furthermore, the TreeSHAP method is mathematically complex and may be difficult for non-technical stakeholders to understand. The NICE method provides a counterfactual explanation as well, but it is model agnostic, and finding the nearest instance and ensuring feasibility can be computationally demanding [9].

There are two initiating methods associated with my proposed method. They are SHAP-C and SEDC methods. The SEDC method is what we focus on first. The SEDC method provides an effective method to identify counterfactual explanations in textual data. The SEDC method for finding counterfactual explanations in text starts by treating each word in a sentence as a separate feature. It removes each feature one by one and checks how the model's prediction changes. If a single feature removal flips the prediction (e.g., from positive sentiment to negative sentiment), that feature is the counterfactual, and the search ends. If not, the method selects the feature that, when removed, most significantly alters the prediction. It then creates new subsets by removing the selected feature along with one other feature, checking how the prediction changes for each new subset. This process iterates, continually expanding and pruning the search space to avoid re-expansion of the same subsets until it finds a counterfactual or reaches a predefined limit. The features that flip the prediction upon removal are identified as the most influential for the initial classification.

The SEDC method is great for providing a detailed understanding of how individual words affect the model's output by treating each word as a separate feature. When looking into particular words or phrases, this can be especially helpful. On the other hand, SEDC considers each feature individually and removes them one by one to evaluate the model's prediction, the SEDC method can incur a significant computational cost. This approach is especially computationally intensive for long texts or datasets with numerous features.

By focusing on the most influential features first (based on the feature importance of each word), the proposed method can more quickly identify pivotal features that have a substantial impact on the model's prediction, thus potentially speeding up the process. In addition, by not examining every single feature in sequence, the proposed method avoids the redundancy of evaluating features that have minimal influence on the prediction.

SHAP-C is a hybrid explanation algorithm that combines the strengths of the Shapley Additive Explanations (SHAP) method and the Sequentially Eliminating Discrediting Counterfactuals (SEDC) method. SHAP-C works to generate counterfactual explanations using SHAP as a base. SHAP is used to compute the Shapley value of each feature in the data set [11]. The Shapley value

is a measure from cooperative game theory that indicates the contribution of each feature towards the model's prediction for a specific instance. It does this by considering all possible subsets of features and the impact of including or excluding a feature on the model's prediction. After the Shapley values are computed, the features are ranked based on these values. Features with higher Shapley values are considered more important for the model's prediction and are ranked higher. Once the features are ranked, the SEDC method is used to generate counterfactuals. This is done by removing the most important features (according to the Shapley values) from the instance one by one until a change in the model's prediction class is observed. This modified instance, which would be classified differently by the model, is considered a counterfactual instance. In the SHAP-C method, we have to calculate the shape value separately, but in my proposed method, we can get the calculated feature importance value while training the random forest, so we can save calculating time.

Further, model explainability visualizations provided by most of the XAI tools are very complex and those visualizations can be understood only by the subjects' experts. So, it is very important to provide a user-friendly more understandable visualization of the internal process of the black box models to the end users.

The proposed method in this research study provides a model-specific, local counterfactual explanation using feature importance in a random forest mechanism to enhance the explainability of the RF model related to the text classification domain.

1.4 Research Problem

The research problem focuses on generating counterfactual rule-based explanations for Random Forest classifiers when they become black boxes in text classification tasks. Text data typically consists of a vast number of unique features or words, making it challenging to comprehend the contribution of each feature to the model's decision-making process. This complexity can lead to a lack of transparency and interpretability in the model's predictions, which can hinder user trust and limit the model's practical applicability.

In this context, counterfactual explanations can provide valuable insights by identifying alternative scenarios where the model's prediction would change, thus offering users a better understanding of the factors that influence the model's decisions. The primary goal of this research is to develop a method for generating counterfactual rule-based explanations tailored for Random Forest classifiers in text classification tasks, shedding light on their decision-making process and enhancing their explainability.

Addressing this research problem could lead to significant advancements in the field of explainable artificial intelligence (XAI), particularly for ensemble models like Random Forests, which are known for their performance but often lack transparency in high-dimensional data, such as text classification. Developing a counterfactual rule generation-based explanation method could help improve trust, fairness, and accountability in AI systems that rely on Random Forest classifiers for text classification tasks.

1.5 Research Objectives

1.5.1 Main Objective of the Research Components

Provide a novel model specific, local XAI solution to enhance the model explainability of ensemble models focusing on Random Forest by developing a novel counterfactual rule generation mechanism related to the text classification domain.

In order to generate a counterfactual explanation, our original explanation method combines an iterative word-removal technique with a feature importance ranking mechanism. By systematically removing words in the order of their influence, the iterative word-removal method creates variations of the original input. Monitoring the movement of model predictions towards a predetermined threshold identifies the optimal variant that changes the prediction. The system identifies the most important words by comparing this variant to the original input and analyzing them, then provides a precise counterfactual explanation based on small changes.

In the end, the suggested solution will be represented visually using the best Graphical User Interface (GUI) techniques, which will enhance the end users' experience. The user can better understand the relationship between input features and how input features are mapped to an output through visualization.

1.5.2 Specific Objectives

To achieve the main objective, study must achieve several sub objectives within the study period.

1. Prepare the dataset and implement the RF classifier.

To perform the experiments, there should be an identified dataset that aligns with the requirements. The study used the IMDb Movie Reviews dataset, consisting of 50,000 movie reviews. Each review, which is presented as a textual statement, captures a user's viewpoint and sentiments concerning a specific movie. Reviews are categorized as "positive" or "negative," depending on the sentiment they express.

2. Develop the novel counterfactual rule generation mechanism related to the text classification task.

Used an iterative word-removal technique combined with a feature importance ranking system to create a counterfactual explanation. This word-removal process removes words iteratively based on their significance. Following that, the model's shift in predictions is closely monitored until it aligns with a predefined threshold. Finally, the system identifies the words that, when removed, resulted in a pivotal change in the prediction, providing an easily understood counterfactual explanation based on minimal modifications.

3. Test the output with existing explainable methods.

4. Do experiments to improve the XAI solution more.

5. Do the visualization using the most appropriate Graphical User Interface (GUI) techniques.

2. METHODOLOGY

Several key milestones must be met to implement the proposed solution. This section will delve into the specific steps required for this research and outline the necessary tools and technologies for each sub-task in a more detailed manner.

2.1 Overall System Architecture

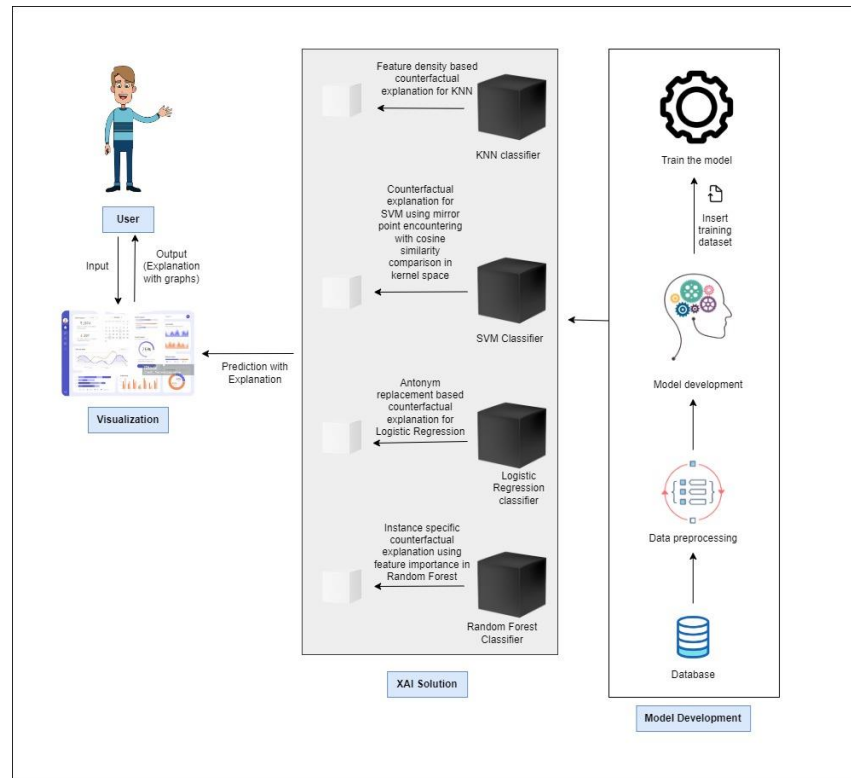


Figure 2.1- 1: System Architecture diagram

The primary objective of this project is to introduce a new XAI solution to enhance the interpretability of ensemble-based classification models, particularly focusing on the Random Forest model by developing a novel counterfactual explanation related to the text classification domain. To achieve this, the process encompasses several stages: developing the model, integrating the new XAI solution, and creating a user-friendly GUI. The system's functional workflow follows these specific steps: first, Users input the necessary training dataset and the instance to be predicted via the user interface. Then the provided data is processed using the Random Forest classifier to obtain the predictions.

To derive the counterfactual explanations, the system adopts our proposed mechanism, which is designed to generate instance-specific counterfactual explanations using feature importance that is calculated when training the Random Forest model. After extracting the counterfactual rules, it is important to evaluate the novel explanation mechanism by testing with existing explainable tools and analyzing experts' feedback. Finally, the outcomes of the process (Counterfactual Explanation) will be transferred to the GUI with appropriate visualizations to be more understandable for the users.

This study used the "IMDb Movie Reviews Dataset", which is commonly used in natural language processing (NLP) and sentiment analysis work. This dataset has 50,000 movie reviews. Every review is like a short note that tells what a person thinks about a film. Reviews are marked as 'positive' or 'negative'. If a review says good things about a movie, it is rated as 'positive'. But if it says bad things, it is rated as 'negative'.

Movie reviews often contain various textual elements like quotations, character names, or plot points. So, it is essential to follow the necessary preprocessing steps; In the model development process Initially, all characters in the text are converted to lowercase. Then remove any HTML content, any special characters, and stopwords present in the text. Finally did the words tokenization and lemmatization.

2.2 Counterfactual Explanation for Random Forest Model.

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both classification and regression tasks in ML. It is based on the concept of ensemble learning, which is the process of combining multiple classifiers to solve a complex problem and improve the performance of the model. Random Forest is a classifier that contains many decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset. Instead of relying on one decision tree, the random forest takes the predictions from each tree and based on the majority votes, predicts the final output. The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

The working process of random forest classifier (Shown in Figure 2.2-1: First, N decision trees are combined to generate the random forest, and then predictions are made for each tree that was produced in the first phase.

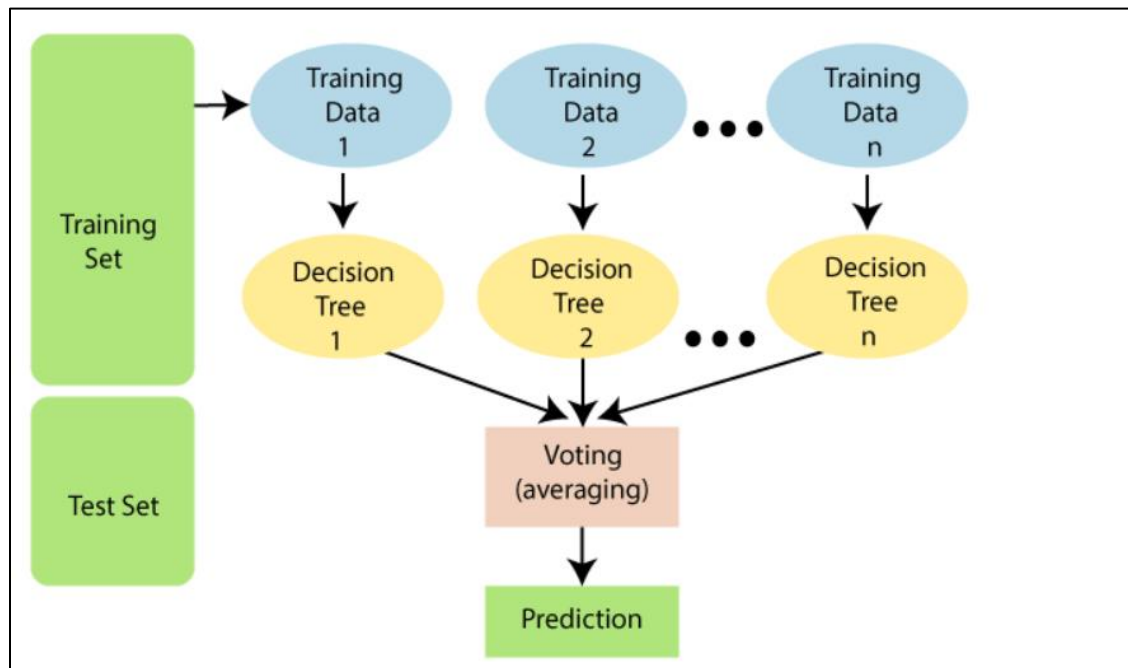


Figure 2.2- 1: Working process of Random Forest.

In the context of the Random Forest model, feature importance plays a crucial role in decision-making. It is computed during the model training phase and is relatively efficient, especially when compared to calculating SHapley value. When talking about feature importance in a random forest, we are essentially trying to calculate the significance of each feature in making accurate predictions. The algorithm offers a direct method to estimate this significance based on how frequently and how deeply each feature is used to split the data across all trees. When calculating feature importance in Random Forest, before providing the text data into the Random Forest, it needs to be converted into a numeric format. This process is called vectorization. Term Frequency-Inverse Document Frequency (TF-IDF) is a method for converting text data into a vector form. This technique assigns weight to each word in a document signifying its importance in the entire corpus. Term Frequency (TF) is the ratio of the number of times a word appears in a document compared to the total number of words in that document. Inverse Document Frequency (IDF) measures the significance of a word in the entire corpus. It is calculated by dividing the total number of documents by the number of documents containing the word and then taking the logarithm of that quotient.

Term Frequency (TF): It measures the frequency of a word in a document.

$$TF = \frac{\text{Number of times term } t \text{ appears in a document}}{\text{Total number of terms in the document}}$$

Inverse Document Frequency (IDF): It measures the importance of a term.

$$IDF(t) = \log e \left(\frac{\text{Total number of documents}}{\text{Number of documents with term } t \text{ in it}} \right)$$

TF-IDF Vector is the product of TF and IDF:

$$TFIDF(t) = TF(t) \times IDF(t)$$

Note that t represents a specific term or word in the document.

Once we have our text data in a vectorized format, we can provide it to a random forest model. Each word in the vector representation is considered a feature. When building decision trees for text classification, each word in your TF-IDF vectorized data becomes a potential feature on which the tree might decide to split. To decide the best word (feature) to split on, the tree considers the Gini impurity. For a node, the Gini impurity is calculated using the equation shown in Figure 2.2-2.

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

Figure 2.2- 2: Equation of Gini Index

Note that t is a particular node. c is the number of classes (e.g., “positive” and “negative” for binary sentiment analysis). P_i is the proportion of text samples that belong to class i at node t .

Every time a word (feature) is used to split the data, the algorithm calculates how much that split decreases the impurity. The more the impurity decreases, the more important the feature is considered to be for that split. The mean decrease impurity is calculated using the equation shown in Figure 2.2-3.

$$\text{Mean Decrease Impurity (Gini importance)} = \text{Gini}(t) - \left(\frac{n_{tL}}{n_t} \times \text{Gini}(t_L) + \frac{n_{tR}}{n_t} \times \text{Gini}(t_R) \right)$$

Figure 2.2- 3: Equation of Gini importance

Note that $\text{Gini}(t)$ is the Gini impurity of the parent node before the split. $\text{Gini}(t_L)$ and $\text{Gini}(t_R)$ are the Gini impurities of the left and right child nodes after the split. n_t is the total number of samples in the parent node. n_{tL} and n_{tR} are the number of samples in the left and right child nodes respectively.

This calculated importance is averaged over all the decision trees in the forest to get the final feature importance for each word. Figure 2.2-4 shows the equation of the average importance.

$$\text{Average Importance} = \frac{\text{Decrease in Impurity from Tree 1} + \text{Decrease in Impurity from Tree 2} + \dots}{\text{Total Number of Trees}}$$

Figure 2.2- 4: Equation of Average importance

Once you obtain the feature importance, you can rank the words based on their importance scores. Higher scores indicate words that played a more crucial role in making decisions across all the trees in the random forest.

However, a primary concern arises from the fact that these feature importance values are global, meaning they reflect the importance of features concerning the entire dataset, not for individual instances. This global nature makes it difficult to pinpoint which features are significant for a particular instance or prediction. Furthermore, they don't provide the direction of class change, which can be crucial for interpretability. Recognizing these challenges, our research aims to develop an instance-specific feature importance mechanism, accompanied by counterfactual explanations.

Here the steps involved in the proposed method are discussed further. First, Extract feature importance from the trained Random Forest model. Next, Select a specific instance for examination. Here we can give the instance (movie review) manually. Then remove feature importance not related to the chosen instance. After that process the selected instance by removing common stop words. This helps in filtering out unnecessary words which do not contribute much to the overall sentiment. As the next step transform words to their base or root form by applying stemming and making an array. Using the model, get the prediction score of the instance and classify the instance as positive or negative. After that, every word in our processed array was considered a feature. We get their feature importance for each feature. Next, Compute the direction of change each feature offers to the given instance. If a feature pushes the prediction towards the positive class, assign a "+" sign, while those pushing towards negative received a "-". Then sort the features according to their importance. If the model classifies the instance as positive, sort the features according to the descending order. Otherwise, the model classifies the instance as negative and sorts the features according to the ascending order. This helps us strategize which words to focus on for our counterfactual explanations.

The main part of this method is removing features one by one. We start removing the most impactful words and see how the model's prediction shifts. If the movie review is a negative review, we first remove the feature that pushes the most to the negative side. Then proceed by iteratively removing the most impactful words from the text based on their feature importance. For the first three iterations, the guidance toward the counterfactual is based on the algebraic sum of the feature importance of the removed features. However, starting from the fourth iteration, guiding is done through the regular scoring method (according to the SEDC method). This is because when using only the algebraic sum of feature importance, it tends to neglect the combinational effect of features, which may lead to non-convergence to a result. For instance, the words "not" and "bad" are separately negative. Therefore, they have negative importance. But "not bad" is a generally good sentiment that may have a positive effect. But if we consider only the feature importance of the algebraic sum, it will be considered negative. This is a simple example, and real-life situations may vary.

The idea is to see how the absence of these words affects the model's prediction. We continue this removal process, rechecking the prediction each time, until the score drops below/above our predefined threshold value. If the movie review is a negative review, we do the removing process until the prediction score is greater than the threshold value. This threshold is crucial because it helps us determine if the model's prediction changes from one class to another. The loop (the iterative removal process) can stop for various reasons: all explanations have been found, the maximum number of iterations has been reached, or the time has been exceeded. This iterative removal helps us pinpoint which words, when absent, lead to significant changes in model predictions. **The words you removed in this process essentially form the counterfactual explanation.** They represent the minimum changes required to the original text to change the model's decision.


In essence, this method determines which features (or words or aspects in the review, in this case) have the most significant impact on the model's prediction for a particular instance. By iteratively evaluating the model's prediction when these influential features are removed, you're finding out what changes would lead to a different prediction, thus providing a counterfactual explanation. This is especially useful when you want to know the 'why' behind a model's prediction for a specific instance.

2.3 Testing & Implementation

2.3.1 Implementation

“Instance-specific counterfactual explanation using feature importance in Random Forest” is a novel XAI solution to enhance the model explainability of Random Forest. From processing movie reviews to identifying the words with the most influence on a prediction, we aimed to provide clear or more intuitive explanations for the decisions our model made. The implementation process began by extracting feature importance from the Random Forest model. We used the feature importance of these words to guide our decisions. By doing so, we hoped to see which words, if changed, would make our model think differently about a review. Let's walk through this process and understand how each step contributes to the larger picture of obtaining reliable counterfactual explanations.

The proposed method's implementation phase involved several steps, including model training with Google Colab and final XAI solution implementation with Visual Studio Code. Model training was carried out using Google Colab. It is an open-source, cloud-based platform provided by Google. It enables users to write and execute code within a browser environment. Besides allowing seamless code writing and execution, Google Colab supports interactive visualizations and comes with a pre-configured data science setup, making it a preferred choice for machine learning and data analysis tasks. The IMDB movie review dataset was trained using the Random Forest model on Google Colab using Python language. The trained model efficiently distinguishes user-provided movie reviews as either positive or negative. Consequently, when provided with a movie review, our trained model can promptly and accurately classify it based on its sentiment.



```

# number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 10, stop = 100, num = 10)]
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(10, 100, num = 5)]
max_depth.append(None)
# Minimum number of samples required to split a node
min_samples_split = [2, 5, 10]
# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2, 4]
# Method of selecting samples for training each tree
bootstrap = [True, False]
# Create the grid
grid_rf = {'n_estimators': n_estimators,
          'max_depth': max_depth,
          'min_samples_split': min_samples_split,
          'min_samples_leaf': min_samples_leaf,
          'bootstrap': bootstrap}

print(grid_rf)

{'n_estimators': [10, 20, 30, 40, 50, 60, 70, 80, 90, 100], 'max_depth': [10, 32, 55, 77, 100, None], 'min_samples_split': [2, 5, 10], 'min_samples_leaf': [1, 2, 4]}

[ ] from sklearn.ensemble import RandomForestClassifier

[ ] #Therefore used randomised -> takes less time -> issue - sometimes go to local minima
grid_imdb_rf = RandomizedSearchCV(RandomForestClassifier(), param_distributions = grid_rf, n_iter = 200, cv = 3, verbose=2, random_state=42, n_jobs = -1)# Fit the r
# Fit the random search model
grid_imdb_rf.fit(x_train_imdb,y_train_imdb.ravel())
#save the trained model to a pickle file to use afterwards
pickle.dump(grid_imdb_rf, open("grid_imdb_rf.pickle", "wb"))

```

Figure 2.3.1- 1:Model training

After successfully training our model on Google Colab, the next phase was the development of the novel Explainable Artificial Intelligence (XAI) solution. For this task, we chose Visual Studio Code (VS Code), a versatile and lightweight code editor from Microsoft. VS Code enabled a streamlined coding experience, offering integrated Git support, syntax highlighting, and an intelligent code completion feature. In this environment, we implemented our novel counterfactual rule generation mechanism, tailored specifically for the text classification domain. The front-end

development related to the visualization part of the novel XAI solution was done using NestJs and MantineUI. NestJs is a versatile framework for building efficient and scalable server-side applications. It offers a modular structure, helping developers to be more organized and efficient. It's primarily used for the back end but can also assist in connecting the back and front ends of a project. On the other hand, Mantine UI is a modern set of high-quality components and hooks for React and Next.js. It provides ready-made design elements and tools that make it easier to create interactive and visually appealing web interfaces. Together, NestJs and Mantine UI provided a solid foundation for building and styling the front end of our visualization project. AWS was used to host the backend server of the web application. AWS is a comprehensive cloud services platform provided by Amazon. It offers a wide range of infrastructure services, including computing power, storage, and databases, which can be used to host web applications, among other things. We ensured that our application had a reliable and scalable environment by hosting our backend server on AWS.

Data Preprocessing:

The Natural Language Toolkit (NLTK) provides various tools and functionalities that can be applied to text data. Preprocessing the textual data was essential in the implementation of our solution to ensure the effectiveness of our model. With nltk, we can break down sentences into words, find out the main meaning of words, and even figure out how words are used in a sentence (like if a word is a noun or a verb). For our project, nltk was super helpful. It made preparing and cleaning our text data easier. So, before we tried to understand the feelings or ideas in movie reviews, we used nltk to clean up our data.

Movie reviews often include textual elements such as quotations, character names, or plot points. As a result, it is essential to complete the necessary preprocessing steps before beginning the model development process. As the preprocessing steps, initially, all characters in the text are converted to lowercase. Then remove any HTML content as shown in Figure 2.3.1-2. Sometimes, movie reviews pulled from websites come with these HTML tags. So, we use 'BeautifulSoup', a great tool for managing and cleaning up web content. When we pass a review through this method, 'BeautifulSoup' finds and takes out all those HTML tags, giving us back just the plain review text.


```

28 class Preprocessor:
29     def _strip_html(self, text):
30         soup = BeautifulSoup(text, "html.parser")
31         text = soup.get_text()
32         return text
33

```

Figure 2.3.1- 2:Remove HTML tags.

This makes sure that our analysis is based only on the actual words of the review, without any web page code mixed in.

The special character removal method helps clean up movie reviews by taking out any characters that aren't letters or numbers. Sometimes, movie reviews have symbols, punctuation, or other special characters that we might not need when analyzing the sentiment or content of the review. By doing this, we make sure our analysis focuses only on the main words and content of the review, without any potentially distracting symbols or characters. Figure 2.3.1-3 shows the implemented code used to remove special characters.

```

33
34     def _remove_special_characters(self, text, remove_digits=True):
35         pattern = r"[^a-zA-Z0-9\s]"
36         text = re.sub(pattern, "", text)
37         return text
38

```

Figure 2.3.1- 3:Remove special characters.

The ‘remove stopwords’ method cleans up movie reviews by taking out common words that don't add much meaning to the text as shown in Figure 2.3.1-4. These words are called “stop words”. Examples of stop words include “and”, “the”, and “is”. When analyzing the sentiment or content of a movie review, these words might not give us a lot of information. As a result, the cleaned-up review focuses only on the keywords that may assist us in understanding the feelings or ideas expressed in the movie review.

```

38
39     def _remove_stopwords(self, text, is_lower_case=False):
40         tokens = self.tokenizer.tokenize(text)
41         tokens = [token.strip() for token in tokens]
42         if is_lower_case:
43             filtered_tokens = [
44                 token for token in tokens if token not in self.stop_words
45             ]
46         else:
47             filtered_tokens = [
48                 token for token in tokens if token.lower() not in self.stop_words
49             ]
50         filtered_text = " ".join(filtered_tokens)
51         return filtered_text

```

Figure 2.3.1- 4: Remove stopwords

The ‘lemmatize_text’ method is used to change words in the movie reviews to their base or root form. This process is called "lemmatization". This method makes it easier to compare and analyze different reviews. For example, “run”, “runs”, “running”, and “ran” would all become “run”.

As shown in Figure 2.3.1-5, LUTLabelEncoder is a custom tool developed to simplify the process of converting categorical data, like textual labels, into numerical values that machines understand more efficiently. It functions as a dictionary where every unique label is associated with a unique number. When initializing LUTLabelEncoder, it's provided with a list of labels, and internally, it assigns a unique number to each one. This mechanism enables two primary functions: transform, which takes in textual labels and returns their corresponding numerical values, and ‘inverse transform’, which does the opposite, converting numbers back into their original textual labels. Utilizing this encoder ensures that our machine learning processes and models are fed with data in a format they can process, while still retaining the capability to interpret results back in a human-readable format.

```

15 class LUTLabelEncoder:
16     def __init__(self, labels: List[str]) -> None:
17         self.lut = labels
18
19     def transform(self, df: pd.DataFrame) -> np.array:
20         enc_lbls = df.apply(lambda st: self.lut.index(st)).to_numpy()
21         return enc_lbls
22
23     def inverse_tranform(self, labels: List[int]) -> List[str]:
24         labels = [self.lut[lbl] for lbl in labels]
25         return labels
26

```

Figure 2.3.1- 5: LUTLabelEncoder

As shown in Figure 2.3.1-6, The TextVectorizer was created in the implementation to convert movie reviews from textual format to a numerical representation suitable for machine learning models. To begin, the Preprocessor () ensures thorough text cleaning by removing unnecessary characters, stopwords, and lemmatizing words according to the above-mentioned steps. Secondly, vectorized text data using TF-IDF vectorizer to transform the cleaned text into its numerical. The result is a matrix where each row represents a movie review in numerical format. This matrix can then be used as input to machine learning models.

```

103
104 class TextVectorizer:
105     def __init__(self, tfidf_path: str) -> None:
106         self.preproc = Preprocessor()
107         self.vectorizer = joblib.load(tfidf_path)
108
109     def __call__(self, txts: Union[List[str], str]) -> csr_matrix:
110         if type(txts) == str:
111             txts = [txts]
112
113         preprocs = [self.preproc(txt) for txt in txts] # l sentences
114         vects = self.vectorizer.transform(preprocs) # matrix of shape (l, n)
115
116         return vects
117

```

Figure 2.3.1- 6:TextVectorizer

Novel XAI solution Implementation:

Before implementation, required libraries were imported to the environment as shown in Figure 2.3.1-6.

```
src > analyzers > rf.py > RFAnalyzer
import time
import numpy as np
from scipy.sparse import lil_matrix, csr_matrix
from ordered set import OrderedSet
import joblib
from itertools import compress
from ..processors import TextVectorizer
from .base import BaseAnalyzer
import json
from typing import Dict, Any
import re
import matplotlib.pyplot as plt
```

Figure 2.3.1- 7:Required libraries.

After importing the necessary libraries, the dataset and the RF model were imported as shown in Figure 2.3.1-7.

```
# Import DataSet and Models
from src.models import AnalysisModels as Models
from src.datasets import IMDBDataset

ds = IMDBDataset(
    config_path="./configs/datasets/imdb.yaml", root="datasets/imdb", download=True
)

models = Models(
    config_path="./configs/models/analysis-models.yaml",
    root="./models/analysis-models",
    download=True,
)

loaded_plain_model_rf = models.rf.model
```

Figure 2.3.1- 8: Import dataset and RF model.

As shown in Figure 2.3.1-8 and 2.3.1-9, the first step in implementing a novel XAI solution is to Extract feature importance from the trained Random Forest model. Next, Select a specific instance for examination. Here we select the instance manually. Then remove feature importance not related to the chosen instance. Using the model, get the prediction score of the selected movie review and classify the review as positive or negative. After that, every word in our processed array was considered a feature. We calculated the feature importance for each feature. Next, Compute the direction of change each feature offers to the given instance. If a feature pushes the prediction towards the positive class, assign a "+" sign, while those pushing towards negative received a "-".

```
def _get_features_importances(self, instance):
    feature_importance = self._model.feature_importances_
    initial_score = self._model.predict_proba(instance)[0][1]
    print("initial_score ", initial_score)
    indices_active_elements = np.array(np.nonzero(instance)[1]).reshape(
        len(np.nonzero(instance)[1]), 1
    )
    feature_set = [frozenset(x) for x in indices_active_elements]
    candidates_to_expand = []
    for features in indices_active_elements:
        candidates_to_expand.append(OrderedSet(features))
    explanation_candidates = candidates_to_expand.copy()
    perturbed_instances = [
        self._perturb_fn(x, inst=instance.copy()) for x in explanation_candidates
    ]
    scores_explanation_candidates = [
        self._classifier_fn(x) for x in perturbed_instances
    ]
    sign_change = [
        1 if (initial_score - x[0]) > 0 else -1
        for x in scores_explanation_candidates
    ]

    current_index = 0
    sign_changed_importances = []
    for element in indices_active_elements:
        sign_changed_importances.append(
            {
                str(self.feature_names[element[0]]): sign_change[current_index]
                * feature_importance[element[0]]
            }
        )
```

Figure 2.3.1- 9: Novel Method implementation 1

```

current_index = 0
sign_changed_importances = []
for element in indices_active_elements:
    sign_changed_importances.append(
        {
            str(self.feature_names[element[0]]): sign_change[current_index]
            * feature_importance[element[0]]
        }
    )
    print(
        element[0],
        " ",
        feature_importance[element[0]],
        sign_change[current_index],
        scores_explanation_candidates[current_index],
    )
    feature_importance[element[0]] = (
        sign_change[current_index] * feature_importance[element[0]]
    )
    print(element[0], " ", feature_importance[element[0]])
    current_index += 1
print("sign_changed_importances ", sign_changed_importances)
self._report_data["feature_importances"] = sign_changed_importances
# if sign change is 0, feature_importance value set to -value
# feature_importance = [x if x > 0 else -x for x in feature_importance]
return feature_importance

```

Figure 2.3.1- 10: Novel method implementation 2

Then sort the features according to their importance. The main part of this method is removing features one by one. We start removing the most impactful words and see how the model's prediction shifts. Then proceed by iteratively removing the most impactful words from the text based on their feature importance. We continue this removal process, rechecking the prediction each time, until the score drops below/above our predefined threshold value as shown in Figure 2.3.1-10 and 2.3.1-11.

```

442 |         )
443 |         sorted_data_in = sorted(features, key=lambda x: x["importance"], reverse=True)
444 |         inverse_sorted_data_in = sorted(features, key=lambda x: x["importance"])
445 |
446 |         print("sorted_data_in ", sorted_data_in)
447 |         if self.revert == 1:
448 |             sorted_data_in = inverse_sorted_data_in
449 |
450 |         indices_active_elements = np.nonzero(text)[
451 |             1
452 |         ] ## -> Gets non zero elements in the instance as an array [x, y, z]
453 |         sorted_indices = sorted(
454 |             indices_active_elements, key=lambda x: importances[x], reverse=True
455 |         )
456 |         indices_active_elements = np.array(sorted_indices)
457 |         number_active_elements = len(indices_active_elements)
458 |         indices_active_elements = indices_active_elements.reshape(
459 |             (number_active_elements, 1)
460 |         ) ## -> Reshape to get a predictable
461 |
462 |         candidates_to_expand = (
463 |             []
464 |         ) ## -> These combinations are further expanded -> These are the elements to be removed from the sentence
465 |         for features in indices_active_elements:
466 |             candidates_to_expand.append(OrderedSet(features))
467 |         print("candidates_to_expand ", candidates_to_expand)
468 |         ## > Gets an array with each element in reshaped incides as an ordered set -> [OrderedSet([430]), OrderedSet([
469 |
470 |         explanation_candidates = candidates_to_expand.copy()
471 |         print("explanation_candidates ", explanation_candidates)
472 |         ## Gets a copy of the above array -> Initially
473 |
474 |         feature_set = [
475 |             frozenset(x) for x in indices_active_elements

```

Figure 2.3.1- 12: Novel method implementation of class change process 1

```

scores_explanation_candidates = [
    self.classifier_fn(x, self.revert) for x in perturbed_instances
]
# Get predictions for each perturbed instance where one or more elements are removed from the in
# It is in form of [[x], [y], [z]]
print(
    "scores_explanation_candidates \n",
    scores_explanation_candidates,
    "\n",
)
scores_candidates_to_expand = scores_explanation_candidates.copy()

scores_perturbed_new_combinations = [
    x[0] for x in scores_explanation_candidates
]
# Therefore get it to the shape [x, y, z] by getting the [0] th element of each element array
# print(
#     "scores_perturbed_new_combinations ", scores_perturbed_new_combinations
# )

# ***CHECK IF THERE ARE EXPLANATIONS***
new_explanations = list(
    compress(
        explanation_candidates,
        scores_perturbed_new_combinations < self.threshold_classifier,
    )
)
# Get explanation candidates where their probability is less than the threshold classifier -> Positi
# print("New Explanations \n", new_explanations)
explanations += list(
    compress(
        explanation_candidates,
        scores_perturbed_new_combinations < self.threshold_classifier,
    )
)
# print("\n explanations - explanations_score_change" - explanations)

```

Figure 2.3.1- 11: Novel method implementation of class change process 2

As shown in Figure 2.3.1-12, it prints the words removed in this process essentially form the counterfactual explanation. They represent the minimum changes required to the original text to change the model's decision.

```
        final_words.append(word)

    # Join the words back into a final string
    final_string = " ".join(final_words)

    word_pattern = r"\b\w+\b"

    # Split the input string into words while preserving punctuation
    words_with_punctuation = re.findall(word_pattern, final_string)

    final_words = []
    for word in words_with_punctuation:
        if word.lower() in removed_words:
            final_words.append("-" * len(word))
        else:
            final_words.append(word)

    final_string = " ".join(final_words)

    if final_prob > self.threshold_classifier:
        final_class = [1]
    else:
        final_class = [0]

    self._report_data["output"] = {
        "Removed_words": removed_words,
        "final_text": final_string,
        "final score for positive": final_prob[0],
        "final class": final_class[0],
    }
```

Figure 2.3.1- 13:Final output

2.3.2 Testing

The novel explanation method had to be tested as the final step in the development process. The method's performance is evaluated by testing each of the explanation steps with the necessary inputs to return the specified output. Following that, the integrated system was tested separately by sending inputs through the built user interface. Furthermore, the accuracy of the novel method must be evaluated by comparing the outputs with existing methods. We can identify the drawbacks of the novel method during the testing process and improve the solution by addressing those drawbacks.

Test Plan and Test Strategy

The novel method is tested to ensure its effectiveness and accuracy. A proper test plan is essential for the testing process. A test plan includes the scope of testing, objectives for tracking project progress, and a list of the tasks to be tested. When we consider testing strategies, we get a roadmap for how testing activities will be carried out, what will be tested, how testing resources will be allocated, and the goals that need to be fulfilled through the testing process.

Steps of test strategy:

- Define the items to be tested
- Select the functions based on the importance and risk to the user
- Design test cases as identified by the use case description
- Execute the testing
- Record the results of the conducted tests
- Identifying the bugs
- Correcting the bugs
- Repeat the test case until the output results are same as the expected results.

Test Case Id	01
Test Case	Verify the novel method provides the counterfactual explanation successfully.
Test Scenario	Verify whether the novel counterfactual explanation method successfully display the affected features.
Input	Watching that film was a complete waste of time. The plot was dull from start to finish, and the performances were bad. I can't recommend it to anyone.
Expected Output	“waste”, “dull”, “bad”
Actual Result	“waste”, “dull”, “bad”
Status (Pass/Fail)	Pass

Table 2.3.2- 1:Test case to check the performance of the novel XAI method.

3. RESULTS & DISCUSSION

3.1 Results

In this research study, implemented counterfactual explanation for RF focusing on sentimental analysis of text classification tasks. RF's novel methodology provides instance-specific counterfactual explanation based on the feature importance values that are calculated from the gini impurity values. To evaluate the performance of the novel RF explanation method, we compared it to the SEDC method that initiated the implementation of the novel RF explanation. The SEDC provides a method to identify counterfactual explanations in textual data based on prediction scores of the output.

To provide novel explanation solutions we utilize the IMDB movie reviews dataset. When we consider the sentimental analysis of the above dataset, it classifies the reviews as positive and negative labels. Therefore, we evaluated the RF explanation's performance both for positive and negative reviews.

3.1.1 Result Evaluation for the Negative reviews:

We applied the same negative review for both the novel method and the SEDC method. The class change of the two methods related to the applied negative review is visualized using the multiple-line graph. The y-axis of the graph shows the prediction score, and the x-axis shows the iteration number. The blue and green lines represent the class changes of the novel method and SEDC method respectively. The Orange line indicates the threshold value, 0.49. This threshold value is a predefined value that is used in both SEDC and novel methods. In both SEDC and novel methods to occur a class change to positive the prediction score must be greater than the threshold value for the negative reviews.

Negative Review 1:

“What can I say? I ignored the reviews and went to see it myself. Damn the reviews were so right. What a waste of money considering its budget. Good thing, I went to see Kill Bill after this one. To see a really scary movie, would be Crossroads! I like "Girl in Gold Boots" better than this crap.” [class label – negative]

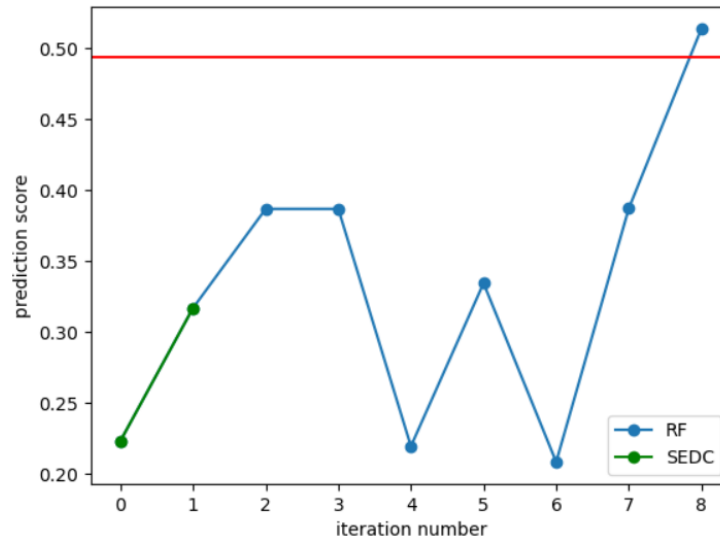


Figure 3.1.1- 1: Class change for negative review 1 RF explanation Vs SEDC

According to Figure 3.1.1-1, the initial prediction score of the novel method is 0.223 which means it is a negative review (threshold > 0.22). First start the removal process according to the “instance-specific counterfactual explanation using feature importance in RF model”. Then continue this removal process by rechecking the prediction score each time, until the score goes up to the predefined threshold value. According to the above graph, there is a class change into positive in the 8th iteration and the final score is mentioned as 0.528. However, even after the maximum number of iterations, there is no class change from negative to positive in the SEDC method. In Figure 3.1.1-1, the green line ends between the 0.47 and 0.48 prediction scores and does not exceed the threshold value.

The instance-specific counterfactual explanation for random forest returns the most affecting words to provide a counterfactual explanation. For the above-mentioned review the explanation returns "ignore", "waste", "kill" and "crap" words. Therefore, those features most affected to the label change from negative to positive.

Negative Review 2:

This film might have bad production values, but that is also what makes it so good. The special effects are gross out and well done. Robert Prichard as the leader of the gang is hilarious, as are the other members. This film is actually trying to make a point, by saying that nuclear waste plants are bad. 4/10 Fair comedy, gross out film.

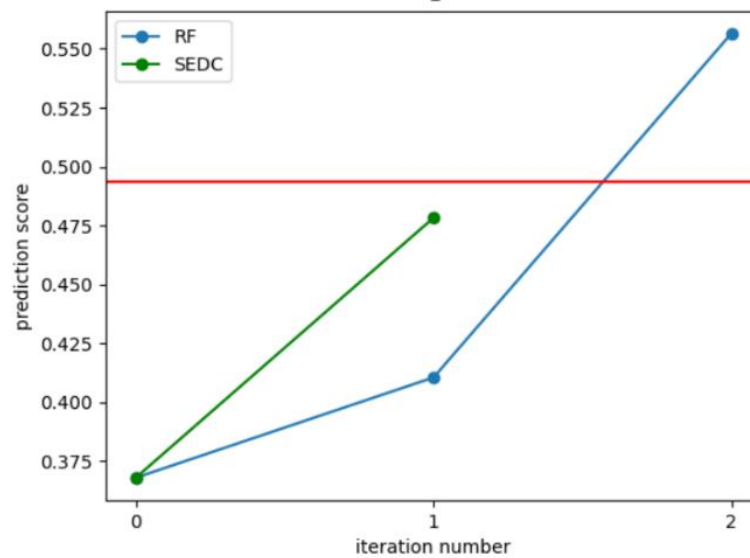


Figure 3.1.1- 2: Class change for negative review 2 RF explanation Vs SEDC

According to Figure 3.1.1-2, the initial prediction score of the novel method is 0.368 which means it is a negative review (threshold > 0.37). As shown in the above graph, there is a class change into positive in the 2nd iteration and the final score is mentioned as 0.556. In the SEDC method, like the previous negative review 1, even after the maximum number of iterations, there is no class change from negative to positive for the negative review 2. Also, the green line ends between the 0.46 and 0.48 prediction scores and does not exceed the threshold value as well. For the above-mentioned review, the explanation returns "bad" and "waste" words. Therefore, those features most affected the label change from negative to positive.

Further, we compared the novel method with the SEDC method using another set of negative reviews, and the following graphs show the class label changes that occurred in the novel method and SEDC method related to those negative reviews. According to those figures, we can realize the SEDC method does not provide any class label change from negative to positive. However,

the novel RF explanation method always provides a class change from negative to positive.

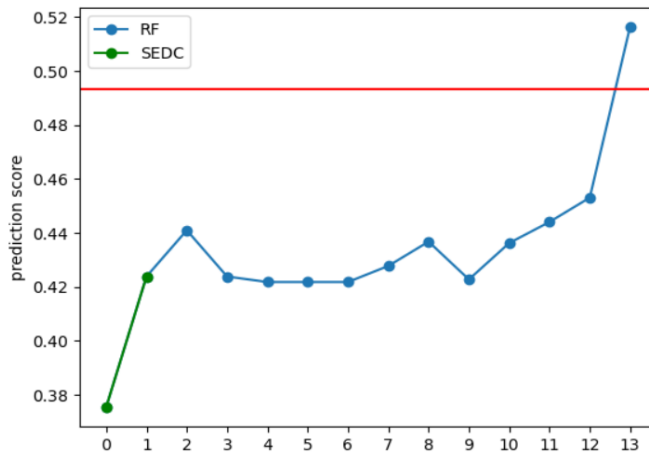


Figure 3.1.1- 3: Class change of negative review 3 RF explanation Vs SEDC

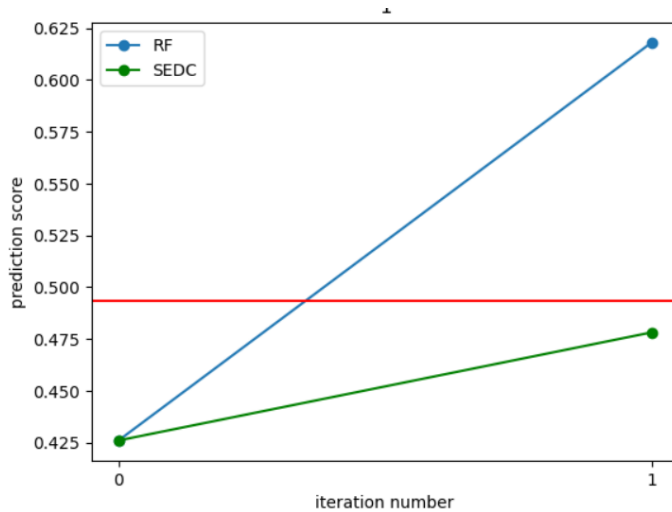


Figure 3.1.1- 4: Class change of negative review 4 RF explanation Vs SEDC

3.1.2 Result Evaluation for the Positive reviews:

Here we applied the same positive review for both the novel method and the SEDC method. The class change of the two methods related to the applied positive review is visualized using the multiple-line graph. Same as the negative reviews' evaluation, we get the threshold value as 0.493. In both SEDC and novel methods to change the class label to negative the prediction score must be less than the threshold value for the positive reviews.

Positive Review 1:

This movie is one of the funniest, saddest and most accurate portrayals of the mentality that seems to have pervaded the Balkans yet again, 45 years after the time depicted. All the usual characters and conflicts are presented with such anger, sadness and love combined that it is impossible to decide whether crying or laughing would be the more appropriate response. The accuracy of portrayal and the timelessness of the types, however, make it for a great film to watch if one wants to understand a little bit of what drove ex-Yugoslavia to its madness. In fact, no diplomat dealing with the region should attempt anything until they saw this movie, and its twin, *Maratonci trce počasni krug. * Did I mention it is one of the funniest movies I've ever seen?

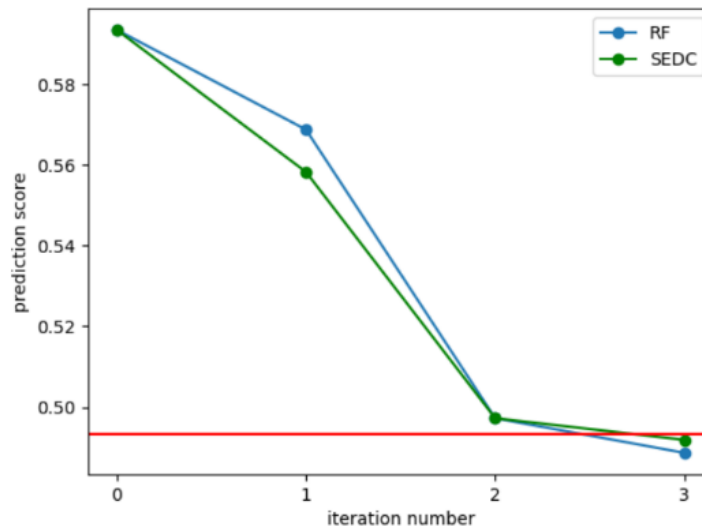


Figure 3.1.2- 1: Class change of positive review 1 RF explanation Vs SEDC

According to Figure 3.1.2-1, the initial prediction score of the novel method is 0.593 which means it is a positive review (threshold < 0.59). After input the positive review to the instance specific counterfactual explanation, it follows the feature removal process according to the feature importance. This removal process is continued by rechecking the prediction score each time until the score is less than the predefined threshold value. According to the above graph, there is a class change into negative in the 3rd iteration and the final score is mentioned as 0.489. As shown in Figure 3.1.2-1, When we apply the SEDC method for the above positive reviews it also provides

class change to negative after 3 iterations. The prediction score value is less than the threshold value, 0.493. For the above-mentioned review, the explanation returns "funny", "great", "love" and "laugh" words. Therefore, those features most affected the label change from positive to negative.

Positive Review 2:

I like Steven Seagal but I have not a CLUE what this movie was about. I am not easily lost with movies but I had no idea who was on who's side. Hopefully some of his future movies will be a little better. My wife still thinks he looks good in black jeans, however.

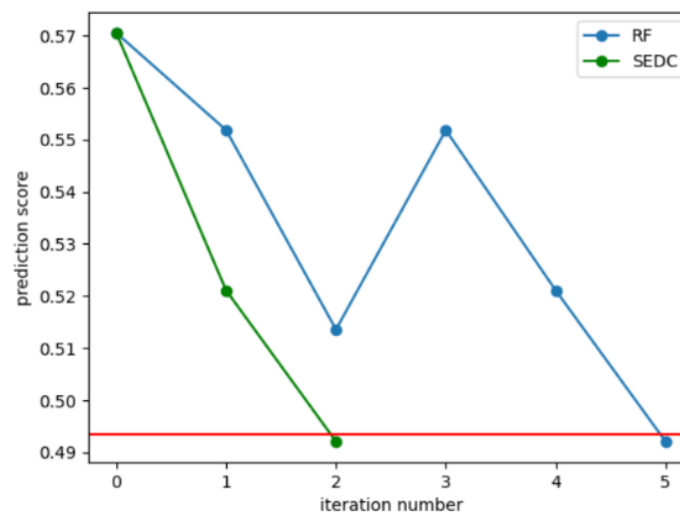


Figure 3.1.2- 2: Class change of positive review 2 RF explanation Vs SEDC

According to Figure 3.1.2-2, the initial prediction score of the novel method is 0.570 which means it is a positive review (threshold < 0.57). As shown in the above graph, there is a class change into the negative in the 5th iteration and the final score is mentioned as 0.492. When we apply the SEDC method for the above positive reviews it also provides class change to negative after 2 iterations. The prediction score value is less than the threshold value, 0.493. For the above-mentioned review, the explanation returns "better" and "good" words. Therefore, those features most affected the label change from positive to negative.

Positive Review 3:

Garde À Vue has to be seen a number of times in order to understand the sub-plots it contains. If you're not used to french wordy films, based upon conversation and battle of wits rather than on action, don't even try to watch it. You'll only obtain boredom to death, and reassured opinion that french movies are not for you.

Garde À Vue is a wordy film, essentially based upon dialogs (written by Audiard by the way)and it cruelly cuts the veil of appearances.

Why does Maître Martineau (Serrault) prefer to be unduly accused of being a child murderer rather than telling the truth ? Because at the time of the murder he was with a 18 years old girl with which he has a 8-years sexual relation. His wife knows it, she's jealous of it and he prefers to be executed (in 1980 in France, there was still death penalty) rather than unveiling the sole "pure and innocent" aspect of his pitiful life.

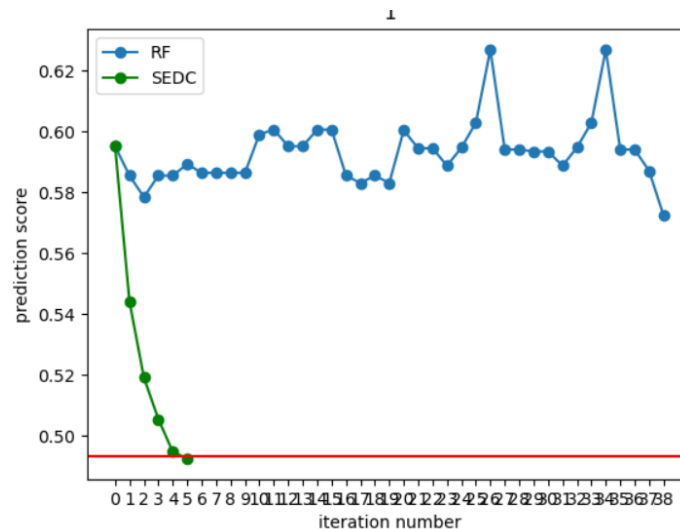


Figure 3.1.2- 3: Class change for positive review 3 RF explanation Vs SEDC

According to Figure 3.1.2-3, the initial prediction score of the novel method is 0.5951940366225161 which means it is a positive review (threshold < 0.59). After applying the counterfactual explanation for the above positive review, it does not provide any class change even 24 iterations occurred. However, the SEDC method changes the class label to negative after the 3rd iteration. In that, we can realize the novel counterfactual method of RF does not provide class change to negative in some situations.

3.2 Discussion

In this study, our main goal was to implement a novel model specific, local XAI solution to enhance the model explainability of the random forest model. The random forest model can become a black box when it has a large number of decision trees with complex interactions between the features. It is difficult to understand the overall decision-making process because the final decision is based on the output of many decision trees. There are many explainable techniques that can be used to provide explainable solutions for black box models.

This is where the idea of "counterfactual explanation" comes in. A counterfactual explanation provides insight into a machine learning model's prediction by illustrating a hypothetical scenario wherein specific feature values are altered to achieve a different, desired outcome. In essence, it answers the "what if" questions by demonstrating the minimal changes needed to reverse a prediction. Here we used a counterfactual rule-based mechanism to explain the black box behavior of the RF model.

Our method goes beyond just knowing how important factors are. This means we can find out what needs to change to get a different prediction. We do this by removing the process considering feature importance and seeing how it affects the model's prediction. Until the first three iterations, we used feature importance and from the fourth iteration, guiding is done through the regular scoring method. This helps us make sure our results are strong and trustworthy.

We tested our method on real-life situations, like understanding why a movie review got a negative rating or figuring out the most important words in the movie reviews. This shows that our method is useful for explaining how models work in different situations. It can help us understand why a machine made a particular decision.

We tested our method on real-life situations, like understanding why a movie review got a negative rating or figuring out the most essential words in the movie reviews. This shows that our method helps explain how models work in different situations. It can help us understand why a machine model made a particular decision.

However, this journey does not conclude here. There remains a host of possibilities for future enhancements and directions for this research: While our novel counterfactual method for Random Forest models is a significant stride towards transparency, it does not provide a class change in some situations to positive reviews. This isn't a consistent issue, but addressing it is a priority for future work. In future research, the focus will be on exploring ways to fine-tune the novel method to ensure a more reliable counterfactual method. Enhancing the approach aims to provide users with even more accurate and insightful explanations.

Advanced Visualization Techniques: An area ripe for future development lies in the realm of visualization. Presently, our XAI solutions lack comprehensive visualizations to facilitate user interaction. Future research can concentrate on delving into visualization techniques derived from the fields of User Interface (UI) and User Experience (UX) design. By incorporating effective visualization strategies, we can enable more accessible and user-friendly interactions with our XAI solutions, thereby further augmenting their practicality and user adoption.

This research has laid a solid foundation for enhancing the transparency and trustworthiness of AI-driven decision-making systems. Through innovative methodologies tailored to diverse black-box models, we have demonstrated the potential to unravel key model features influencing predictions and elucidate the ramifications of data alterations. The promising results achieved thus far serve as a steppingstone for future refinements, promising more precise and user-friendly XAI solutions to meet the growing demands of an AI-driven world.

4. SUMMARY

Member	Component	Task
Warnasooriya S.D (IT20097660)	Provide a novel counterfactual explanation for Support Vector Machine classifier related to the text classification domain. (Custom word flipping generator implementation and cosine similarity comparison based counterfactual explanation)	<ul style="list-style-type: none"> • Prepare the data set and implement Support Vector Machine classifier. • Generate a novel counterfactual rule-based mechanism using text classification domain. <ul style="list-style-type: none"> ➤ Implement the custom word flipping generator. ➤ Vectorize all the prompts using a TFIDF vectorizer and map into the kernel space. ➤ Calculate the mirror point of the original input review. ➤ Calculate the cosine similarity between each contradiction with the mirror point and return the closest contradiction to the mirror point. • Results evaluation conduct for the SVM's novel explanation and RF's novel explanation. Here RF's explanation compares with the SEDC method. • Frontend implementation related to the SVM's explanation.
Srinidee Methmal H.M (IT18161298)	Provide a novel counterfactual explanation for Logistic Regression related to the text classification domain. (Antonym replacement based counterfactual explanation)	<ul style="list-style-type: none"> • Prepare the data set and implement Logistic Regression classifier. • Develop a novel counterfactual rule-based mechanism using text classification domain. <ul style="list-style-type: none"> ➤ Vectorize all the text using TFIDF vectorizer. ➤ Antonym selection and iteration replacement and removal ➤ Get the prediction score of instances and classify the class label.

		<ul style="list-style-type: none"> ➤ Get the feature importance of each feature and sort. • Results evaluation conduct for the LR's novel explanation and RF's novel explanation. Here RF's explanation compares with the SEDC method Frontend. • Frontend implementation related to the LR's explanation.
Britto T.A (IT201009698)	Provide a novel counterfactual explanation for Random Forest model related to the text classification domain. (Instance specific counterfactual explanation)	<ul style="list-style-type: none"> • Prepare the data set and implement Random Forest model. • Generate a novel counterfactual rule-based mechanism using text classification domain. <ul style="list-style-type: none"> ➤ Vectorize all the text using TFIDF vectorizer. ➤ Extract the feature importance and remove the features that are not related to the given instance. ➤ Get the prediction score and classify the class labels. ➤ Get the feature importance and sort. ➤ Remove the most impact features iteratively until get a class change. • Results evaluation conduct for the RF's novel explanation. Here RF's explanation compares with the SEDC method Frontend. • implementation related to the RF's explanation.
Lakshani N.V.M (IT20013950)	Provide a novel counterfactual explanation for K nearest neighbour model related to the text classification domain. (Feature density comparison based counterfactual explanation)	<ul style="list-style-type: none"> • Prepare the data set and implement K-nearest Neighbour classifier. • Generate a novel counterfactual rule-based mechanism using text classification domain. <ul style="list-style-type: none"> ➤ Implement the custom word flipping generator.

		<ul style="list-style-type: none"> ➤ Vectorize all the prompts using a TFIDF vectorizer and map into the kernel space. ➤ Calculate the probability scores. ➤ Calculate the neighbour statistics and return the best counterfactual. • Results evaluation conduct for the KNN's novel explanation. Here RF's explanation compares with the SEDC method Frontend. • implementation related to the KNN's explanation.
--	--	---

Table 4- 1: Summary of each member's contribution

5. CONCLUSION

This study analyzed the critical issues of improving model explainability in AI-driven decision-making systems, with a focus on K Nearest Neighbor, Logistic Regression, Random Forest, and Support Vector Machine models in the context of text classification. The impetus for this study has arisen from the rapidly growing utilization of AI in industries where transparency and explainability of AI-based decisions are very important. As a solution for the discussed explainability problem, this research study provides novel explainable methods for the above-mentioned ML models using a counterfactual rule generation mechanism. When we consider the proposed methods, they include instance-specific counterfactuals for RF, custom word flipping generator implementation and cosine similarity comparison-based mechanism for SVM, feature density comparison-based mechanism for KNN, and antonym replacement-based explanations for LR. In the end, all of these explanations return the features that are most affected to provide the counterfactual explanation. The implementation of the counterfactual explanations for each model is accomplished through the process which contains model development, encountering the novel XAI solution, and developing the GUI for end users. Backend implementations are done in Google Colab using Python and the frontend is implemented using NestJs with Mantine UI. Further backend is hosted using an AWS cloud. When evaluating the results, novel proposed explainable methods must be compared to existing methods to improve the novel XAI solutions. Here we selected an instance-specific counterfactual explanation of the RF method to conduct the result evaluation process from the implemented novel methods. To evaluate the performance of the novel RF explanation method, we compared it to the SEDC method that initiated to the implementation of the novel RF explanation. Through the process of evaluating the results, it was identified that the SEDC method does not provide any class change to positive, but the proposed method provides that. Furthermore, the SEDC method offers a class change to negative in some situations while the novel proposed method does not. Future work related to this research study includes exploring ways to fine-tune the novel method of RF to ensure a more reliable counterfactual method and the development of advanced visualizations to provide counterfactual explanations for end users in a more understandable way.

6. RERERENCES

- [1] Miller, Tim. "Explanation in artificial intelligence: Insights from the social sciences." arXiv Preprint arXiv:1706.07269. (2017).
- [2] Kim, Been, Rajiv Khanna, and Oluwasanmi O. Koyejo. "Examples are not enough, learn to criticize! Criticism for interpretability." Advances in Neural Information Processing Systems (2016).
- [3] A.Adadi and M. Berrada, "Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI)," IEEE Access, 2018, doi: 10.1109/ACCESS.2018.2870052.
- [4] M. T. Ribeiro, S. Singh, and C. Guestrin, "'Why should i trust you?' Explaining the predictions of any classifier," in Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Aug. 2016, vol. 13-17-Aug, pp. 1135–1144, doi: 10.1145/2939672.2939778.
- [5] S. M. Lundberg and S. I. Lee, "A unified approach to interpreting model predictions," Adv. Neural Inf. Process. Syst., vol. 2017-Decem, no. Section 2, pp. 4766–4775, 2017.
- [6] [AI Explainability 360 — aix360 0.1 documentation](#)
- [7] Arrieta, A. B., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., ... & Herrera, F. (2020). Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. Information fusion, 58, 82-115
- [8] Lundberg, S. M., Erion, G., Chen, H., DeGrave, A., Prutkin, J. M., Nair, B., ... & Lee, S. I. (2019). Explainable AI for trees: From local explanations to global understanding. arXiv preprint arXiv:1905.04610.
- [9] Brughmans, D., Leyman, P., & Martens, D. (2023). Nice: an algorithm for nearest instance

counterfactual explanations. *Data Mining and Knowledge Discovery*, 1-39.

[10] R. K. Mothilal and C. Tan, “Explaining Machine Learning Classifiers through Diverse Counterfactual Explanations.”, 2019.

[11] Ramon, Y., Martens, D., Provost, F., & Evgeniou, T. (2020). A comparison of instance-level counterfactual explanation algorithms for behavioral and textual data: SEDC, LIME-C and SHAP-C. *Advances in Data Analysis and Classification*, 14, 801-819.

[12] Rathi, S. (2019). Generating counterfactual and contrastive explanations using SHAP. arXiv preprint arXiv:1906.09293.

[13] Martens, D., & Provost, F. (2014). Explaining data-driven document classifications. *MIS quarterly*, 38(1), 73-100.