

# **INT2X: Explanation for the causes of prediction**

**23-142**

Srinidee Methmal H.M

IT18161298

B.Sc. Honors Degree in Information Technology specializing in  
Software Engineering

Department of Information Technology

Sri Lanka Institute of Information Technology

Sri Lanka

September 2023

**INT2X: Explanation for the causes of prediction**

**23-142**

Srinidee Methmal H.M

IT18161298

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

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
Srinidee Methmal H.M.	IT1816128	

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

.....

.....

Signature of the Supervisor:

Date

## ABSTRACT

In recent years, the rapid advancement of machine learning techniques has driven the field of artificial intelligence into the spotlight. With these advancements and automation, the communities question the reliability of decision-making processes that use AI. Among the major drawbacks of this decision-making process are the lack of transparency and interpretability when it comes to real-world critical scenarios. As a result, many machine learning models that are used in the decision-making processes are black boxes, which means the inner workings and decision processes of these models are highly complex and difficult to understand. The concept underlying this problem is called explainable AI (XAI). Nowadays, many countries come up with rules and regulations to prohibit the use of black box models where the model interpretability is hidden (e.g.: European GDPR). Therefore, model interpretability (XAI) is the next step of AI. Even though the topic is popular, the number of studies done in this regard is less. The overview presented here with the main objective of providing a novel, model agnostic XAI solution to enhance the model interpretability of black box models by developing a novel counterfactual solution specific to the text classification domain. The proposed method will specifically target the binary classification-based logistic regression (LR) model. LR is a very popular classifier that performs well with linear data in a transformed space. However, logistic regression is not a fully black box model. When there is a non-linear relationship or greater data complexity between the features and the data, it turns into a "black box". This approach explains the feature importance of individual words in a given text and how it alters the model prediction through antonym replacement.

Key words - XAI, Counterfactual explanation, feature importance, antonym replacement

## ACKNOWLEDGMENT

Regarding the final year project involving an explanation for the causes of prediction. (INT2X), Srinidee Methmal H.M 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 like 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 will be eternally thankful to our previous teachers for the consistent support they provided.

ABSTRACT.....	4
ACKNOWLEDGMENT.....	5
LIST OF FIGURES.....	7
LIST OF TABLES.....	8
LIST OF ABBREVIATIONS .....	9
Amazon Web Services.....	9
Natural Language Toolkit.....	9
1.Introduction .....	10
1. Introduction .....	13
1.1 Background .....	13
1.2 Literature survey .....	16
1.2.1 Different feature ranking methods (SHAP, SAGE, FSP and BSP).....	16
1.2.2 Feature Relevance [13] .....	17
1.2.3 Saliency map-based technique [13] .....	17
1.2.4 Explanations by example method [16] .....	17
1.2.5 Correlation coefficient method [17] .....	17
1.2.6 Explaining Machine Learning Classifiers through Diverse Counterfactual Explanations [22] ...	18
1.3. Research Gap .....	20
1.4. Research Problem .....	22
1.5 Research Objectives.....	23
1.5.1 Main Objective .....	23
1.5.2 Specific Objectives .....	23
2.2 Counterfactual explanations for Logistic regression model .....	25
2.3 Testing and Implementation.....	28
2.3.1 Implementation .....	28
2.3.2 Testing.....	38
3.Results and discussion .....	40
3.1 Results .....	40
3.1.1 Result evaluation for positive review: .....	40
3.1.2 Result evaluation for negative review: .....	44
3.2 Discussion.....	47
4.Summary.....	48
5.Conclusion.....	51

6.References .....	52
--------------------	----

## LIST OF FIGURES

Figure 2.1- 1: System Architecture diagram .....	24
Figure 2.2- 1: Remove HTML tags.....	29
Figure 2.2- 2: Remove special character.....	29
Figure 2.2- 3: Remove stop words.....	29
Figure 2.2- 4: Lemmatize text.....	30
Figure 2.2- 5: LUTLabelEncoder.....	30
Figure 2.2- 6: Required Libraries.....	30
Figure 2.2- 7: Imported dataset and LR model.....	31
Figure 2.2- 8: Novel Implementation 1.....	32
Figure 2.2- 9: Novel Implementation 2.....	33
Figure 2.2-10: Novel implementation 3.....	33
Figure 2.2-11: Novel implementation 4.....	34
Figure 2.2-12: Novel implementation 5.....	34
Figure 2.2-13: Novel implementation 6.....	35
Figure 2.2-14: Novel implementation 7.....	36
Figure 2.2-15: Novel implementation 8.....	36
Figure 2.2-16: Novel implementation 9.....	37
Figure 2.2-17: Novel implementation 10.....	37
Figure 2.2-18: Novel implementation 11.....	38
Figure 2.2-19: Novel implementation 12.....	38
Figure 2.2-20: Novel implementation 13.....	39
Figure 3.1-1: Class changes for positive review 1 LR explanation Vs SEDC.....	40
Figure 3.1-2: Class changes for positive review 2 LR explanation Vs SEDC.....	41
Figure 3.1-3: Class changes for positive review 3 LR explanation Vs SEDC.....	43
Figure 3.2-1: Class changes for negative review 1 LR explanation Vs SEDC.....	44
Figure 3.2-2: Class changes for positive review 2 LR explanation Vs SEDC.....	45
Figure 3.2-3: Class changes for positive review 3 LR explanation Vs SEDC.....	46

## LIST OF TABLES



## 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

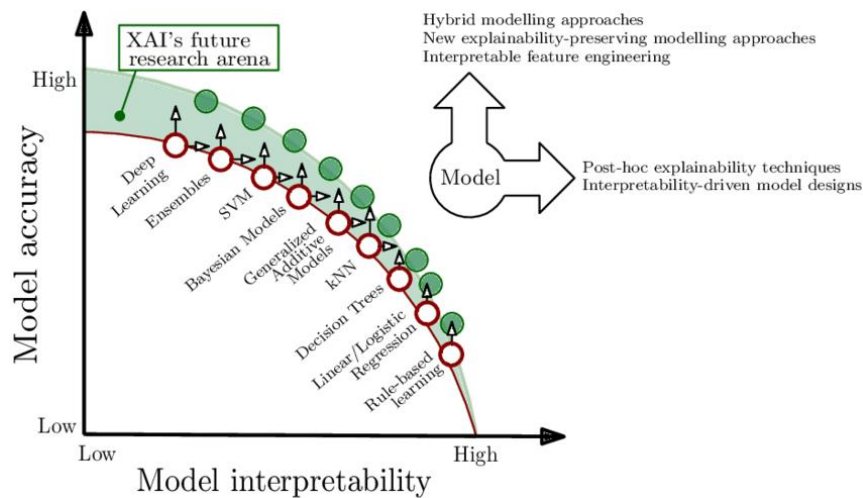
## 1.Introduction

With the fourth industrial revolution, one remarkable trend that gets immense attraction is the widespread integration of Artificial Intelligence (AI) in the decision-making processes of various machine learning domains. Notably, sectors such as health care, legal, military, transportation, and finance have witnessed rapid and extensive adoption of AI models to make critical decisions. So, to address the Uninterpretability of these black box AI models is more important with valid reasoning. As a solution, explainable AI provides clear and understandable insights into the decision-making process of AI models.

When the concept of AI became popular among a large number of domains, they were encouraged to use it in critical decision-making processes. A medical diagnosis of a disease is one such critical situation where AI models provide explanations that reveal the factors and data points in reaching a particular conclusion. It may highlight the relative importance of each input feature such as patient symptoms, laboratory results, and medical history factors that have the most significant impact on the diagnosis. XAI can provide insights into how diagnosis outcome changes if input features are different. Moreover, it shows how altering specific symptoms or test results might lead to a different diagnosis. However, there are situations where models can be biased and give inaccurate results. In such situations, model interpretability is needed to identify and avoid bias in decision making.

The model explainability and interpretability are closely related concepts in the fields of machine learning and artificial intelligence. They refer to the extent to which machine learning predictions or decisions can be understood and justified by humans. Here we introduce two taxonomies to divide machine learning models called “White Box Models” and “Black Box Models”. There are two different levels of interpretability and transparency in machine learning.

White box models are machine learning models that provide clear explanations of how they make predictions or decisions, allowing humans to understand the reasoning behind outcomes. These models further provide insights into the inner processes by making it easier to understand the decision-making process. Some examples of white box models are regression models, decision trees, and the naïve Bayes model. When considering Black box models, they provide predictions or decisions without explaining how they arrived at such decisions. These black box models can handle complex, high-dimensional data and have outstanding accuracies compared to white box models. But those models are less interpretable. Since the General Data Protection Regulation (GDPR) required justifications to comply with global regulations, less interpretability is a drawback of black box models. Some examples of black box models are boosted trees, deep neural networks, random forests, and support vector machines.



There are numerous definitions for model explainability/interpretability. One of the most popular widely used definitions is “Interpretability is the degree to which a human can understand the causes of decision” and “Interpretability is the degree to which a human can consistently predict the model’s results”. The main objective of XAI is to provide answers to the “wh” questions related to AI-based decision making. In [1] proposed four reasons why explainability is needed. Those are,

#### 1. Explain to justify

Simply this provides clear and comprehensible explanations of the AI system’s decision-making process. This further involves explaining why a particular outcome was produced with valid reasons.

#### 2. Explain to control

Control refers to the ability of humans to have a level of control or influence over AI systems. It allows humans to understand AI reasoning and potentially intervene when necessary.

#### 3. Explain to improve

Explainability helps to understand model behaviour and identify areas that can be improved further.

#### 4. Explain to discover

Explainability leads to the discovery of valuable information or knowledge in large datasets. It helps to identify trends and unexpected relationships within data. Explainable Artificial Intelligence (XAI) is gaining importance in the modern world due to its various applications and the need for transparency and accountability of AI systems. In healthcare, XAI helps doctors and clinicians understand the reasoning behind AI-driven diagnoses, which makes it easier to trust and validate predictions. In finance, XAI provides explanations for risk assessment tasks such as credit scoring, loan approvals, and enabling fair lending practices. It allows financial institutions to understand trading strategies and investment decisions. In the legal sector, XAI assists lawyers in legal research by providing transparency in case law. It also facilitates contract analysis and complex legal contracts. XAI is used to explain decisions made by autonomous vehicles ensuring safety-critical decisions are more understandable and accountable. When considering this research mainly focuses to enhance the model interpretability of black box models using text classification domain. It is also beneficial in sentiment analysis which involves determining how sentimental analysis models arrive at their sentiment predictions. XAI also can reveal which words and phrases in a text influence sentimental prediction. This also assists in identifying biases in sentimental analysis models. It also can reveal if a model is consistently biased in sentiment predictions by helping to address fairness issues.

## 1. Introduction

### 1.1 Background

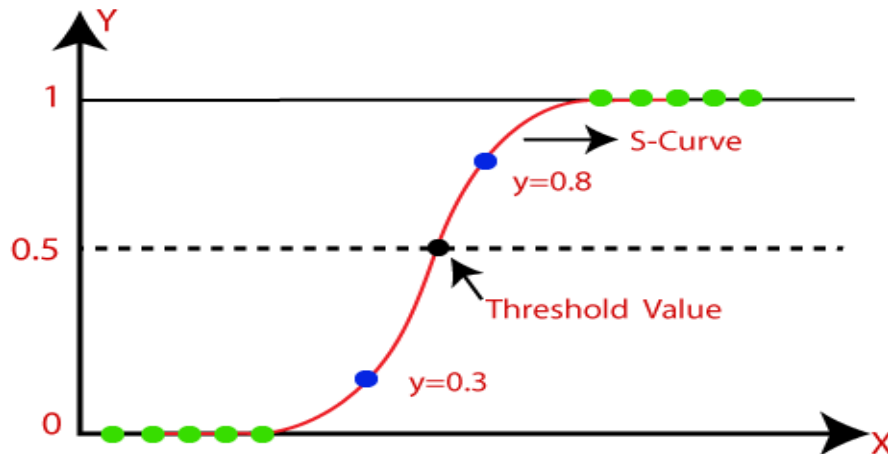
With the rapid development of AI models and applications the research field of Explainable Artificial intelligence (XAI) has gathered significant attention from researchers and the broader community. However, the decisions made by those AI models are critical. Here Explainable AI(XAI) provides clear and understandable insights into how an AI model arrives at its conclusions. So many researchers are proposing different model explainable methods for the research community. Among them, the local-global method, model specific-model agnostic method, and post hoc-intrinsic methods play an important role. Among them, the local-global approach provides interpretable explanations for the predictions made by machine learning models.

Local explanations focus on explaining the behavior of machine learning model for a specified instance or prediction. Among local explanation methods LIME (Local Interpretable Model-agnostic Explanations), SHAP (SHapley Additive exPlanations), and partial dependence plots play a major role. However global explanations provide a broader understanding of the model's behavior by analyzing overall performances across the entire dataset. Some examples of global explanation methods include feature importance analysis, feature importance heat maps, and Shapley values.

In the context of Explainable Artificial Intelligence (XAI) model-specific and model-agnostic approaches are two strategies for explaining the behavior of machine learning models. The model-specific approach is designed to provide explanations that are specific to the internal mechanisms and characteristics of those models. Apart from that, the model-agnostic approach provides explanations that are not tied to the inner workings of the specific model. Furthermore, the explanation methods can also be post hoc and intrinsic. Post-hoc methods aim to provide explanations for the model's predictions without altering the model's internal behavior. Some of the examples are LIME (Local Interpretable Model-agnostic Explanations), SHAP (SHapley Additive exPlanations), and feature importance analysis. Comparatively intrinsic methods aim to create inherently interpretable models and provide

explanations as a part of their internal structure. Models with intrinsic interpretability are decision trees, and rule-based models.

Among them, logistic regression is one of the most popular machine learning models under supervised learning. It is used to model the dichotomous dependent variable and multiple independent variables which are either continuous or categorical. When considering the behavior, instead of fitting to a regression line it fits to an 'S' shaped logistic function which predicts two possibilities 0 or 1. Furthermore, the curve of the logistic function indicates the likelihood of something happening such as cancerous or not.



Logistic regression is an essential classification algorithm. When using it under supervised classification problems, the goal of the algorithm is to find the decision boundaries among the classes. However, depending on the problem, decision boundaries can be complex and non-linear in geometric shape. With this nonlinearity in high-dimensional feature space, the dimensions of the feature space are simply determined by several elements in the feature vectors. With the increase of the feature vectors in high dimensional space logistic regression becomes a black box. Although it is not a fully black box model, it becomes a black box when increasing the number of dimensions.

There are some assumptions under which logistic regression provides valid results [1].

- Explanatory variables should have a linear relationship with the logit of the response

variable.

- Errors should not be correlated.
- Explanatory variables should not be highly correlated with each other (Multicollinearity).
- There should be no outliers, high-leverage values, or highly influential points.

Without satisfying the above assumptions, the model may have problems like unnecessarily inflated standard errors, spuriously low or high t-statistics [2], parameter estimates with illogical signs, and lead to invalid statistical inferences [3]. These are due to the multi-collinearity in explanatory variables. Here multi-collinearity means the occurrence of high correlations among two or more independent variables. Belsley [4] noted that "... in nonexperimental sciences, ..., collinearity is a natural law in the data set resulting from the uncontrollable operations of the data-generating mechanism and is simply a painful and unavoidable fact of life." In many surveys, correlated variables are collected for analysis. Shen and Gao [5] suggested a double penalized maximum likelihood estimator combining Firth's penalized likelihood equation to stabilize the estimates in cases of multicollinearity. Azar [6] proposed a method to estimate the shrinkage in parameters of a Liu-type logistic estimator. Apart from that Schaefer, Roi & Wolfe [7] proposed a ridge-type estimator that has a smaller total mean squared error than the maximum likelihood estimator under certain conditions.

Multi-collinearity can be detected with the help of tolerance and its reciprocal, which is known as the variance inflation factor [8] (VIF). The tolerance of any specific explanatory variable is  $1-R^2$  where  $R^2$  is the coefficient of determination  $n$  for the regression of that explanatory variable on all remaining independent variables. Tolerance close to 1 is little multicollinearity and close to 0 suggests that multicollinearity becomes a threat. So, a tolerance of 0.1 or less is a cause for concern. VIF means the reciprocal of tolerance. It shows how much the variance of the coefficient estimate is being inflated by multicollinearity. The values of VIF exceeding 10 are multicollinearity. But for weaker models like logistic regression values above 2.5 are a concern.

Apart from that eigen values for the scaled, uncentered cross-product matrix, condition indices, and variance proportions for each explanatory variable are also used to identify multicollinearity. If the eigen value is large, regression parameters are greatly affected by small changes in the explanatory variables or outcome. If the eigen values are similar, then the fitted model is unchanged by small changes in the measured variables [9]. The condition indices are also computed as the square root of the ratio of the largest eigen value to the eigen value of interest. When there is no collinearity the eigen values and condition indices are equal to unity. An informal rule of thumb is that if the condition index is 15, multicollinearity is a concern; if it is greater than 30, multicollinearity is a very serious concern [10].

Many explainable methods are used to provide explainable solutions to the black-box nature of logistic regression. Among them, counterfactual explanation defines why a particular model made a specific prediction by considering what changes in input data lead to a different outcome. In other words, counterfactual explanation represents the minimum change required to alter the model's prediction or decision. Here minimal changes highlight the most influential features or factors in decision making. It further involves creating hypothetical scenarios where some aspects of input data are altered while keeping the others constant. Apart from that counterfactual analysis assists in detecting and mitigating biases in text classification models. Other than that counterfactual analysis is used to understand why a text classification is making errors.

Furthermore, the counterfactual analysis ensures transparency, accountability, and fairness.

However, the key goal of this research is to overcome explainability issues by providing model agnostic, counterfactual XAI solution related to the text classification domain.

## 1.2 Literature survey

To overcome less model interpretability, researchers propose sophisticated techniques to explain black box models. Since the research area is new, most of the techniques are in research state and they build communities around the development of the technique to come up with better solutions.

The authors of [12], introduce a couple of methods as explainability strategies. They are “Scooped Related Methods” and “Model Related Methods”. Scooped related methods are globally interpretable (focus on the whole mechanism of the model) and locally interpretable (focus on explaining a specific decision or prediction). The model-related methods are also classified as Model-Specific Interpretability (methods are limited to a selected model) and Model-Agnostic Interpretability (methods are not limited to the selected model and it considers the prediction and explain separately).

Among proposed techniques some have taken the attention on researchers and new contributions have been done on top of these techniques. Those are LIME (Local Interpretable Model-Agnostic Explanation) [11] and SHAP (Shapley Additive explanations) [12].

SHAP is a unified approach that has developed based on coalitional game theory. For each feature, SHAP assigns a feature importance value for a given prediction to approach the explainability of the model. When considering LIME, it uses local surrogate models to explain individual predictions or decisions.

### 1.2.1 Different feature ranking methods (SHAP, SAGE, FSP and BSP)

Though SHAP and SAGE scores are based on Shapley theory, SHAP has better performance in terms of model performance. But SAGE scores have the least correspondence in terms of model performance. SHAP does not assume feature independence and leverages feature clustering based on correlations which explains why it performs well better than SAGE. When comparing with other feature ranking methods, forward single pass (FSP) has a lower correspondence between total performance and model performance. In here forward permutation starts with all features permuted and it breaks the relationship between features. Neglecting the important features and isolating individual importance decreases faithfulness of FSP. Other than feature correlations, backward single-pass (BSP) method is the most faithful method. In this method impact of the



correlated features on permutation importance score is heavily dependent on correlations between feature and target variable. If the two features are correlated, but there's a stronger predictor of the target variable permutation importance scores may be negligibly impacted by the feature correlations. To determine the dataset has sufficient correlations with the target variable compared to correlations between features, we compare average feature correlation and the average correlation of a feature with the target variable.

#### 1.2.2 Feature Relevance [13]

It is beneficial to figure out most impactful features which are crucial in decision making. For that purpose, feature importance is introduced. It shows the impact factor of each feature for decisions [14]. Along with feature importance correlation among features is also important. In AI based medical diagnosis, feature correlation in training data is one of the main forces for diagnosis.

#### 1.2.3 Saliency map-based technique [13]

There is a strong correlation between feature score-based justifications and saliency-based Visual representations. More research-based demonstrations also choose it. Saliency-based visualizations are popular because they give visually perceptive explanations that can be easily understood by the end-users [15].

#### 1.2.4 Explanations by example method [16]

This method considers extraction of data samples that relate to the result generated by a certain model. Similar, to how human behaves when explaining a given process, explanations by example are mainly focused on extracting representative samples that show inner relationships and correlations which are found by analysing the model.

#### 1.2.5 Correlation coefficient method [17]

This is a kind of feature selection method that effectively choose the most appropriate feature and reduce data dimensions. There are several supervised correlation coefficient methods. But few can detect both linear correlation and non-linear correlations. In here we further discussed supervised correlation coefficient method called consistency detection. The results obtained from above method can better retain the dimensions of classifications and detect correlation than Pearson correlation coefficient method.

Dimension reduction methods can be divided into two categories feature extraction and feature selection [18]. When considering the generality, the correlation coefficient methods can be divided into two categories called supervised correlation coefficient method and unsupervised correlation coefficient method.

Liliana Forzani et.al. [19] had considered important information on the relation between independent variable and dependent variable may be lost when using principal component

analysis. Wang et al. [20] had considered a feature selection method which can detect statistical significance between discrete dimensions and continuous index. Duan et al. [20] considered the dimensionality reduction methods which do feature selection before feature extraction. Korn F et al. [21] introduces dimensionality reduction method for real data sets with non-uniform distributions and incompletely independent dimensions.

#### 1.2.6 Explaining Machine Learning Classifiers through Diverse Counterfactual Explanations [22]

The concept of counterfactual needs imagination of hypothetical realities that happen other than the existing situation. In this research to understand the concept of counterfactuals it has given a real-time scenario. It will be discussed below.

In a situation of rejecting a loan, using interpretable models the organization may give explanation for that action. As an example, it may be due to “poor credit” history. When we consider in person point of view who has been applied for the loan, the explanation does not help him to decide “what should he/she do next?” If the system can suggest some solutions to improve the chance of getting the loan in future, that would be more effective for both sides.

However, there must be situations where the most important feature or features like gender, race may not be sufficient to flip or change the correlation of prediction features. Therefore, it is pretty much important to provide alternative rules which are actionable.

As the counterfactual explanation for above loan example, the paper has provided below information [22].

“You would have received the load if your income was higher by \$10000”

When the person gets that kind of explanation, he/she would be able to identify which features need to be improved to be eligible for the loan in future.

Ramon et al. (2020) introduced the innovative SEDC (Sequentially Eliminating Discrediting Counterfactuals) [23] method to identify counterfactual explanations in textual data. By treating each word as a distinct feature, the method iteratively removes features to gauge shifts 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. 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.

Shubham Rathi has attempted to generate Partial Post hoc P-type Contrastive explanations [ 24] 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.

The primary objective of generating counterfactual explanations for text-based classifiers is to make a set of minimal modifications to the text that changed the label assigned by the classifier. In addition to achieving this transformative effect, the researchers aim to create counterfactuals that adhere to grammatical correctness and semantic coherence. Many works have addressed this issue of generating natural-sounding counterfactual explanations in text using language representation models. Wu et al. (2021) used a language model called fine-tuned GPT-2 model to generate general-purpose counterfactuals that are model-agnostic. Yang et al. (2020) directed their attention to generating counterfactual explanations that alter the classification model's prediction within the context of financial texts.

### 1.3. Research Gap

The prominent objective of this research study is to make logistic regression more interpretable related to the text classification domain. For that, we use a counterfactual explanation mechanism that illuminates the impact of input features on the model's output. Through this process we can identify how specific words or features influence the classification, enabling a clear understanding of why a prediction was made. Moreover, the process highlights crucial features that affect predictions and the specific changes in the input that would lead to a different outcome.

SHAP computes the shapely values from cooperative game theory to assign a unique contribution score for each feature. This approach also provides local interpretations by attributing the contribution of each feature to a particular prediction. It does not directly provide counterfactual explanations but provides a way to understand how each feature contributes to a particular prediction either positively or negatively. SHAP offers detailed feature-level explanations while counterfactuals deliver practical insights by showcasing minimal changes necessary for a specific outcome. Both LIME and the novel counterfactual method aim to address the challenge of model interpretability in machine learning. LIME provides localized, understandable insights by creating simplified models around specific predictions, while the counterfactual method offers a minimal-change perspective for achieving a particular prediction outcome.

The initiated method associated with my proposed method is the SEDC method. 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 SHAP-C method develops by blending the strength 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. Another 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 and finds its counterfactual counterpart in LIME-C.

In existing methodologies, we use removing active features to change the initial prediction which is not efficient. But in the proposed methodology we use replacement instead of removing to get the word combination that changes the initial prediction. Moreover, the proposed opposite word replacement methodology gives faster results as it pushes the result toward the reverse class. Here to compute feature importance we use prediction score change and we assign a threshold value to generate counterfactuals that are minimally altered to generate a particular prediction. Additionally for generating counterfactuals through opposite word replacement, there are multiple studies. They use another language model to generate counterfactuals which is also another black box model. In the proposed approach, we use a more explainable NLTK library than the previously mentioned black box models to generate opposite words. To conclude by Recognizing the existing method challenges, our research aims to develop an opposite-word replacement mechanism accompanied by counterfactual explanations.

#### 1.4. Research Problem

When considering machine learning models, most of them are used in critical decision-making processes. So, multiple approaches have been taken to provide a proper explainable mechanism to explain these models. With that reality, my research problem focuses on generating counterfactual explanations for logistic regression classifiers when they become black boxes under high dimensional data. This problem is more pronounced with the fact that text data is exponentially growing and complex, which makes it challenging to understand the contribution of each feature to the model's decision-making process.

However, counterfactual explanations provide insights into why a particular decision was made and changes in input data or conditions that lead to a different outcome. So, this allows us to understand alternative scenarios where model prediction would change, and also a clear understanding of the factors that influence the specific output. With that, the primary goal of this research is to develop a model-agnostic method for generating a counterfactual rule-based explanation for the logistic regression classifier under the text classification domain.

The above-discussed research problem would lead to significant outcomes in the future Explainable Artificial Intelligence (XAI) domain for logistic regression classifiers, specifically when addressing high dimensional data such as text data. It could enhance interpretability, and accountability that rely on logistic regression.

## 1.5 Research Objectives

### 1.5.1 Main Objective

Provide a novel, model-agnostic, local XAI solution to enhance the model interpretability of binary classification-based logistic regression model by developing a novel counterfactual rule generation mechanism related to the text classification domain.

### 1.5.2 Specific Objectives

To achieve the main objective, there is a set of sub-objectives we should focus on.

- Prepare the dataset and implement logistic regression classifier.
- Develop the novel counterfactual rule generation mechanism related to the text classification task.
- Test the output with existing explainable methods.
- Do experiments to improve the XAI solution more.
- Do the visualization using the most appropriate Graphical User Interface (GUI) techniques.

## 2. Methodology

Several key milestones must be accomplished to implement the proposed solution. This section will discuss the steps that need to be followed to carry out the study and outline tools and technologies for each sub-task in a more detailed manner.

### 2.1 Overall system architecture

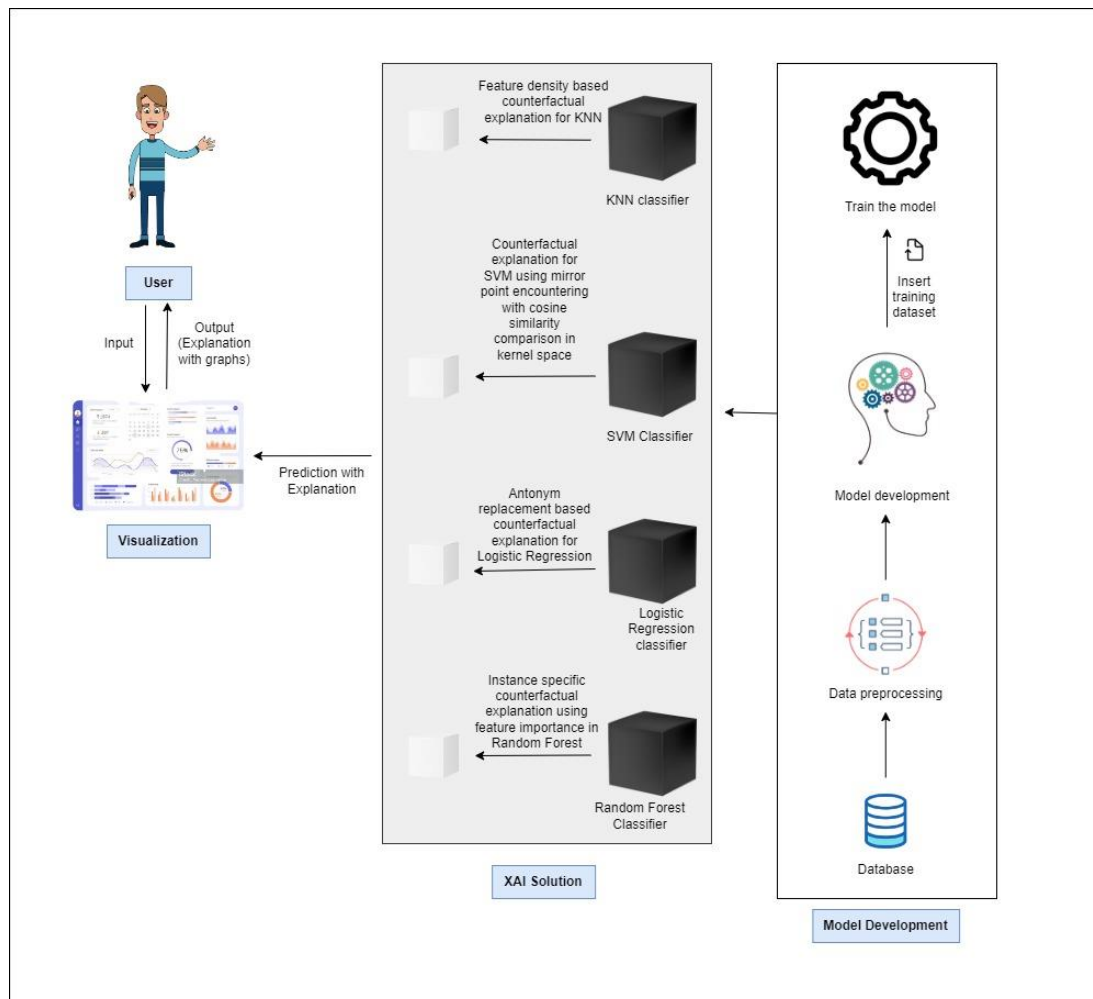


Figure 2.1- 1: System Architecture diagram

The project's primary objective is to introduce a new XAI solution to enhance the interpretability of binary classification-based models focused on Logistic Regression by developing a novel counterfactual explanation related to the text classification domain. Here the main objective is accomplished through several steps. Namely developing the model, encountering the novel XAI



solution, and developing GUI for end users. The system's functional workflow encountered several steps. First, the user has to provide the relevant training data set and the instance to be predicted via the user interface. Then the provided data should be applied to the logistic regression classifier to get predictions.

To extract counterfactual explanations the system adopts the proposed mechanism which is designed to generate antonym replacement counterfactual explanation using feature importance which is calculated in the training phase. After extracting counterfactual rules, it is important to evaluate the novel explanation mechanism by testing with existing explainable tools and analyzing expert feedback. Finally, the outcome of the process (counterfactual explanations) will be transferred to GUI with appropriate visualizations in a user-friendly way.

This research study used the “IMDb Movie Reviews Dataset” which is commonly used in natural language processing (NLP) and sentiment analysis. This dataset has 50,000 movie reviews and each review consists of the user's opinion and the sentiment regarding the particular film. These reviews are labeled as ‘positive’ or ‘negative’ signifying the overall sentiment of that particular review. If a review is about good things and admires the movie and expresses a positive sentiment, it is rated as a ‘positive’ review. But if it is about a bad critical review, 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 stop words present in the text. Finally did the words tokenization and lemmatization.

## 2.2 Counterfactual explanations for Logistic regression model

Logistic Regression is one of the most popular machine learning algorithms that comes under supervised learning techniques. It is used for both classification and regression tasks in machine learning. When talking about the feature importance in the Logistic regression model, it relates to the significance of each feature in making accurate and reliable model predictions. When considering feature importance, first the text data should be converted to numerical vectors. This is the process called vectorization. For that, here we use TF-IDF vectorization which evaluates the importance of a term (word or phrase) within a document relative to a collection of documents (corpus). So, the TF-IDF score reflects both the frequency of a term within the document (TF) and

its importance across a collection of documents (corpus). Here the TF-IDF values are calculated by multiplying the Term frequency (TF) and Inverse Document Frequency (IDF). Mathematically TFIDF is computed as,

$$\text{TF-IDF (t, d)} = \text{TF (t, d)} \times \text{IDF(t)}$$

**t = term**

**d = document**

In the above equation, **TF** refers to the calculation of number of times a term occurs in a document and normalizing it with the total number of documents. The formula for **TF** can be expressed as,

$$\text{TF (t, d)} = \frac{\text{Number of times term t appears in a document d}}{\text{Total number of documents in document d}}$$

**Total number of documents in document d**

At the same time, **IDF** refers to the importance of a word in a collection of documents.

First extract feature importance from the logistic regression model and get the indices of features as an array. Then we can initialize the instance by giving any textual movie review. After that, preprocess the selected instance by removing irrelevant text and stop words. Then we can do lemmatization or stemming. This will reduce words to their base or root form to handle variations. As the next step, we can do text vectorization, which means converting text data to numeric format using the TF-IDF technique. By using the logistic regression classifier, get the prediction score of the initial instance and classify the instance as positive or negative. Then checked antonyms for each feature in the array. If an antonym is not found, we remove the features. Otherwise, we replaced the words with their antonyms and got the probability of the predicted class using the predicted probability function. As the next step if the prediction probability is less than the predefined threshold value, we take it as a class change and add it to the list of counterfactual explanations. **(Here we get the change done to minimize the threshold value)**. This paves the path to generate the best counterfactual explanations. If no class changes, add it to the array of combinations to expand. Here we should follow this process iteratively by removing or replacing each word. Then terminate the above process if the maximum number of iterations run and the maximum time that the algorithm runs exceeds.

After that, we take the word combination to remove where changing the prediction score towards

the reverse class is maximum. Here we remove features that have maximum score change and see how model prediction shifts. If the class is changing from positive to negative, to change the class, the maximum positive score should be removed first. Then proceed by iteratively removing the maximum prediction score change words in the given text. Then we expand the above word combination without the combinations in explanation. With the above new combination, if the opposite word is not found simply remove them. If found replace the particular word with its antonyms. Then get the probability using the logistic regression classifier. If the probability is less than the threshold value, we take it as a class change and add it to the explanations. If not add it to the word combination to expand and get the probability of the combination to expand with that feature. Then we compute the Shapley values of the combination to expand with that feature. Finally, iterate the above process until the end. Further, it outputs the words removed to reverse the model towards the reverse class. This essentially forms counterfactual explanations by representing the minimum changes that should be done to change the model's prediction.

To shed light on the inner workings of the logistic regression model, we conducted an important feature analysis. In existing methodologies, shap values are used to give individual measures for each feature, but for a feature set algebraic sum of shap values is not a good measure. But in the proposed method we use score change to determine feature importance which is more efficient. Apart from that, existing methodology generates a neighborhood first and generates sentences. In contrast, the proposed methodology replaces opposite words instead of generating sentences. Here we need to choose proper antonyms as antonyms are chosen to push toward the current class. In the new method, this is prevented by using shap values to choose antonyms. The initial methods also generate new sentences using different language models that are classified as black box models. But there we propose a novel counterfactual opposite word replacement method using a more explainable NLTK library.

## 2.3 Testing and Implementation

### 2.3.1 Implementation

The proposed “Antonym replacement based counterfactual explanation for logistic regression” is a novel XAI solution that provides counterfactual explanations to enhance the model explainability of logistic regression model. From the processing stage to the final stage, it analyzes the output with replacement words and how the final text changes to provide a reverse sentiment. Here we used the prediction score change to guide our decisions. Here we explore which words are changed to give an opposite sentiment to the given movie review. By following through this approach, we generate more explainable counterfactual explanations to reverse model prediction which is efficient and productive in decision-making.

The model training phase is done using Google Colab and implementation is handled through Visual Studio code in Python. First, the logistic regression classifier is trained using the IMDB movie review dataset. After successfully training our data set using Google Colab, the next phase is the development of a novel XAI solution. For this implementation, we use Visual Studio Code which is a lightweight source code editor. Then the model can efficiently identify the input movie review as positive or negative. Apart from that, we use AWS to host the backend server of our web application. Through that, we can ensure the reliability and scalability of our web application.

The front-end development is done using NestJs and Mantine UI. NestJs is a versatile framework for building efficient and scalable server-side applications. On the other hand, Mantine UI consists of a modern set of components and hooks for ReactJs and NextJs. Mantine UI provides a solid foundation for building and styling our user interfaces. These two frameworks collectively provide ready-made design elements and tools to create interactive, user-friendly interfaces.

#### **Data Pre-processing:**

Data preprocessing is a crucial step in preparing text data for experimentation. It is essential in the implementation phase to ensure the effectiveness of our model. The Natural Language Toolkit (NLTK) also provides various tools and resources for working with human language data, that are crucial for data preprocessing. With NLTK, we can figure out how words are used in a sentence (like nouns, verbs, adjectives, etc). Here NLTK is used mainly for preparing and cleaning our text data easily. Other than that, movie reviews include textual data such as character names, quotations, or plot points. So, there is a necessity to apply data preprocessing steps before the model development phase. As the preprocessing steps, first, all the characters in the text are converted to lowercase. Then remove the HTML content as shown in the below image. As the movie reviews pulled from the website come with HTML tags, we use ‘BeautifulSoup’ which is a great tool for managing and cleaning up web content. When entering a movie review this tool takes off all the HTML tags by giving us back plain text.

```

## Support functions
def strip_html(text):
    soup = BeautifulSoup(text, "html.parser")
    return soup.get_text()

```

*Figure 2.2- 1: Remove HTML tags.*

The special characters removal process helps to clean up movie reviews by taking out any characters that are not characters or numbers. Sometimes, they may contain punctuation marks, symbols, and other special characters that are irrelevant to text analysis tasks. With this, we make sure our analysis only focuses on the main words and content of the review. The figure shows the implemented code used to remove special characters.

```

def remove_special_characters(text, remove_digits=True):
    pattern=r'^a-zA-Z0-9\s'
    text=re.sub(pattern, '',text)
    return text

```

*Figure 2.2- 2: Remove special character*

When analyzing the sentiment or content of a movie review, these stop words may not give us a lot of information. Examples of stop words are “and”, “the” and “is”. As an advantage, clean-up reviews only focus on keywords that assist in understanding feelings or ideas in a given movie review.

```

def remove_stopwords(text, is_lower_case=False):
    tokens = tokenizer.tokenize(text)
    tokens = [token.strip() for token in tokens]
    if is_lower_case:
        filtered_tokens = [token for token in tokens if token not in stop_words]
    else:
        filtered_tokens = [token for token in tokens if token.lower() not in stop_words]
    filtered_text = ' '.join(filtered_tokens)
    return filtered_text

```

*Figure 2.2- 3: Remove stop words*

The data preprocessing technique called ‘`lemmatize_text`’ is used to change the words to their base or root form. It has the advantage of comparing and analyzing movie reviews. As an example, “move”, “moves”, “moving” and “moved” become “move”.

```
def lemmatize_text(text):
    words=word_tokenize(text)
    edited_text = ''
    for word in words:
        lemma_word=lemmatizer.lemmatize(word)
        extra=" "+str(lemma_word)
        edited_text+=extra
    return edited_text
```

Figure 2.2- 4: Lemmatize text.

Look-Up Table Label Encoder (LUTLabelEncoder) is used to encode categorical labels into numerical values. When initializing LUTLabelEncoder, it's provided with a list of labels, and internally, it assigns a unique number to each one. This LUTLabelEncoder ensures consistency in the encoding process by mapping each unique category to a specific integer across different datasets or instances of the model.

```
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_transform(self, labels: List[int]) -> List[str]:
24         labels = [self.lut[lbl] for lbl in labels]
25         return labels
26
```

Figure 2.2- 5: LUTLabelEncoder

## Counterfactual explanations for Logistic Regression

As the initial step required libraries into the environment as shown in figure.


```
# IMPORTS
import numpy as np # linear algebra

from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
```

Figure 2.2- 6: Required Libraries

Then the dataset and LR model were imported as shown in figure.

```
# Import DataSet ds 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")
print(
    ds.x_test.shape,
    ds.x_train.shape,
    ds.x_val.shape,
    ds.y_test.shape,
    ds.y_train.shape,
    ds.y_val.shape,
)
print(
    type(ds.x_test),
    type(ds.x_train),
    type(ds.x_val),
    type(ds.y_test),
    type(ds.y_train),
    type(ds.y_val),
)

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

loaded_plain_model_lr = models.lr.model
```

*Figure 2.2- 7: Imported dataset and LR model*

The first step in implementing a novel XAI solution for the Logistic Regression model is to extract feature importance from the trained Logistic Regression model. Then we can select any instance to start our analysis process. Here we provide the instance manually. Then we get antonyms for the input words. Here it checks all the antonyms found in the current vocabulary and returns the first antonym found. If the feature indices contain antonyms, they are replaced by antonyms. Otherwise, if antonyms are not found, simply remove the feature indices. These steps are clearly shown in the below figure.

```
def classifier_fn_lr(x, negative_to_positive=0):
    """Returns the prediction probability of class 1 -> Not class 0"""
    # print('loaded_plain_model_svc.decision_function(x) - ', loaded_plain_model_svc.decision_function(x))
    prediction = loaded_plain_model_lr.predict_proba(x)
    # If prediction is [1] return the probability of class 1 else return probability of class 0
    if negative_to_positive == 1:
        return prediction[:, 0]
    return prediction[:, 1]
```

*Figure 2.2- 8: Novel Implementation 1.*

```

def get_antonyms(word, model):
    """ " Get antonyms of a word and their indices in the feature vector

    Args:
        word: word to get antonyms for
        model: trained model with feature_importances_

    Returns:
        tuple of antonyms and their indices in the feature vector
    """
    antonyms = []
    antonyms_indices = []
    feature_importance = []
    temp_dict = {}
    for syn in wordnet.synsets(word):
        for i in syn.lemmas():
            if i.antonyms():
                antonyms.append(i.antonyms()[0].name())
    # Remove duplicates in antonyms
    antonyms = list(set(antonyms))

    for word in antonyms:
        if word in loaded_vocab:
            # antonyms_indices.append(ds.feature_names.tolist().index(word))
            # feature_importance.append(
            #     abs(coefficients[loaded_vocab[word]]))
            temp_dict[word] = abs(coefficients[loaded_vocab[word]])
    # Sort the antonyms and their indices based on feature importance
    # antonyms_indices = [x for _, x in sorted(
    #     zip(feature_importance, antonyms_indices), reverse=True)]
    # antonyms = [x for _, x in sorted(
    #     zip(feature_importance, antonyms), reverse=True)]
    # print(temp_dict)

```

Figure 2.2- 9: Novel Implementation 2

```

# return the key with the highest value
if len(temp_dict) > 0:
    max_importance_idx = max(temp_dict, key=temp_dict.get)
    return [loaded_vocab[max_importance_idx]]
else:
    return []

# print(antonyms)
# print(feature_importance)
# if len(feature_importance) > 0:
#     max_importance_idx = np.argmax(feature_importance)
#     return [antonyms_indices[max_importance_idx]]
# else:
#     return []

# if len(antonyms_indices) > 0:
#     return [antonyms_indices[0]]
# else:
#     return []

```

Figure 2.2- 10: Novel Implementation 3



```

def perturb_fn(x, inst, print_flag=0):
    """Function to perturb instance x -> Deform the array -> assign 0 to the x-th column"""
    """
    Returns perturbed instance inst
    """
    inst[:, x] = 0
    return inst

def replace_fn(x, y, inst, print_flag=0):
    """Function to perturb instance x -> Deform the array -> assign 0 to the x-th column"""
    """
    Returns perturbed instance inst
    """
    new_inst = inst.copy()
    try:
        temp_x = inst[:, x]
        temp_y = inst[:, y]
        new_inst[:, x] = temp_y
        new_inst[:, y] = temp_x
    except:
        new_inst[:, x] = 0
    return new_inst

```

Figure 2.2-11: Novel implementation 4.

```

def conditional_replace_fn(x, y, inst, print_flag=0):
    for i in range(len(x)):
        if isinstance(y[i], str):
            inst[:, x[i]] = 0
        else:
            temp_x = inst[:, x[i]]
            temp_y = inst[:, y[i]]
            inst[:, x[i]] = temp_y
            inst[:, y[i]] = temp_x
    return inst

def print_instance(pert_inst, ref_inst, feature_names):
    """Function to print the perturbed instance"""
    """
    Returns perturbed instance inst
    """
    indices_active_elements_ref = np.nonzero(ref_inst)[1]
    indices_active_elements_pert = np.nonzero(pert_inst)[1]
    ref_set = set(indices_active_elements_ref)
    pert_set = set(indices_active_elements_pert)
    # elements in ref_set but not in pert_set
    removed_word_indices = ref_set - pert_set
    # elements in pert_set but not in ref_set
    added_word_indices = pert_set - ref_set
    printable_array = []
    for item in indices_active_elements_ref:
        printable_array.append(".." + feature_names[item] + "..")
    # Change formatting of removed words
    for item in removed_word_indices:
        printable_array[printable_array.index(".." + feature_names[item] + "..")] = (
            "--" + feature_names[item] + "--"
        )
    # change formatting of added words
    for item in added_word_indices:
        printable_array.append("++" + feature_names[item] + "++")

```

Figure 2.2-12: Novel implementation 5

```

# change formatting of added words
for item in added_word_indices:
    printable_array.append("++" + feature_names[item] + "++")
printable_array.append(classifier_fn_lr(pert_inst))
print(printable_array)
return printable_array

def print_ref_instance(ref_inst, feaaure_names):
    printable_array = []
    indices_active_elements = np.nonzero(ref_inst)[1]
    for item in indices_active_elements:
        printable_array.append(".." + feature_names[item] + "..")
    print(printable_array)

```

*Figure 2.2-13: Novel implementation 6*

Meanwhile by using the predicted probability function, we get the probability of the initial predicted class. If the probability is less than the predefined threshold value, we take it as a class change and add it to explanations. If the probability is not less than the threshold, we add those features to a new array. Then one word by one word all the words are removed/replaced. Then we select the word combination where the prediction score towards reverse class is maximum (Simply it means if the class is changing from positive to negative, removing maximum positive value makes it closer towards the negative class). Then we expand the above word combination excluding the combination of explanations by continuing removal or replacement process. Finally iterates on the above steps until the end.

```

def expand_and_prune(
    comb,
    replacement_comb_to_expand,
    expanded_combis,
    feature_set,
    candidates_to_expand,
    candidates_to_expand_replacements,
    explanations_sets,
    explanation_replacement_sets,
    scores_candidates_to_expand,
    instance,
    cf,
    revert=0,
    replacements=[],
):
    """Function to expand "best-first" feature combination and prune explanation_candidates and candidates_to_expand"""

    comb = OrderedSet(comb)
    replacement_comb_to_expand = OrderedSet(replacement_comb_to_expand)
    print("comb: ", comb)
    print("replacement_comb_to_expand: ", replacement_comb_to_expand)
    expanded_combis.append(comb)

    old_candidates_to_expand = [frozenset(x) for x in candidates_to_expand]
    old_candidates_to_expand = set(old_candidates_to_expand)
    print("feature_set: ", feature_set)
    feature_set_new = []
    feature_set_new_replacements = []
    ## If the feature is not in the current combination -> add it to a new list
    for feature in feature_set:
        list_feature = list(feature)
        if len(comb & feature) == 0: # set operation: intersection
            replacement_feature = get_antonyms(
                feature_names[list_feature[0]], loaded_plain_model_rf
            )
            replacement_feature = frozenset(replacement_feature)

            new_string = "0" * (len(comb) + 1)
            replacement_feature = frozenset([new_string])
            # print("replacement_feature: ", replacement_feature, "feature: ", feature)
            feature_set_new.append(
                feature
            )
            # If the feature is not in the current combination to remove from the instance
            feature_set_new_replacements.append(replacement_feature)

    print("feature_set_new: ", feature_set_new)
    print("feature_set_new_replacements: ", feature_set_new_replacements)
    # Add each element in the new set -> which were initially not present -> to the accepted combination -> create new combinations -> (EXPANSION)
    new_explanation_candidates = []
    new_explanation_candidates_replacements = []
    # for element in feature_set_new:
    #     union = (comb|element) #set operation: union
    #     union_replacements = (replacement_comb_to_expand|feature_set_new_replacements[i])
    #     new_explanation_candidates.append(union) # Create new combinations to remove from the instance
    #     new_explanation_candidates_replacements.append(union_replacements)

    for i in range(len(feature_set_new)):
        union = comb | feature_set_new[i]
        union_replacements = (
            replacement_comb_to_expand | feature_set_new_replacements[i]
        )
        new_explanation_candidates.append(
            union
        )
        # Create new combinations to remove from the instance
        new_explanation_candidates_replacements.append(union_replacements)

    print("new_explanation_candidates: ", new_explanation_candidates)
    print(
        "new_explanation_candidates_replacements: ",

```

Figure 2.2-14: Novel implementation 7.

```

        feature_names[list_feature[0]], loaded_plain_model_rf
    )
    replacement_feature = frozenset(replacement_feature)
    if replacement_feature == frozenset():
        new_string = "0" * (len(comb) + 1)
        replacement_feature = frozenset([new_string])
    # print("replacement_feature: ", replacement_feature, "feature: ", feature)
    feature_set_new.append(
        feature
    )
    # If the feature is not in the current combination to remove from the instance
    feature_set_new_replacements.append(replacement_feature)

    print("feature_set_new: ", feature_set_new)
    print("feature_set_new_replacements: ", feature_set_new_replacements)
    # Add each element in the new set -> which were initially not present -> to the accepted combination -> create new combinations -> (EXPANSION)
    new_explanation_candidates = []
    new_explanation_candidates_replacements = []
    # for element in feature_set_new:
    #     union = (comb|element) #set operation: union
    #     union_replacements = (replacement_comb_to_expand|feature_set_new_replacements[i])
    #     new_explanation_candidates.append(union) # Create new combinations to remove from the instance
    #     new_explanation_candidates_replacements.append(union_replacements)

    for i in range(len(feature_set_new)):
        union = comb | feature_set_new[i]
        union_replacements = (
            replacement_comb_to_expand | feature_set_new_replacements[i]
        )
        new_explanation_candidates.append(
            union
        )
        # Create new combinations to remove from the instance
        new_explanation_candidates_replacements.append(union_replacements)

    print("new_explanation_candidates: ", new_explanation_candidates)
    print(
        "new_explanation_candidates_replacements: ",

```

Figure 2.2-15: Novel implementation 8.

```

candidates_to_expand_notpruned = candidates_to_expand.copy()
candidates_to_expand_replacements_notpruned = (
    candidates_to_expand_replacements.copy()
)
# for new_candidate in new_explanation_candidates:
#     candidates_to_expand_notpruned.append(new_candidate)
for i in range(len(new_explanation_candidates_replacements)):
    candidates_to_expand_notpruned.append(new_explanation_candidates[i])
    candidates_to_expand_replacements_notpruned.append(
        new_explanation_candidates_replacements[i]
    )

print("candidates_to_expand_notpruned: ", candidates_to_expand_notpruned)
print(
    "candidates_to_expand_replacements_notpruned: ",
    candidates_to_expand_replacements_notpruned,
)

# Calculate scores of new combinations and add to scores_candidates_to_expand
# perturb each new candidate and get the score for each.
# perturbed_instances = [perturb_fn(x, inst=instance.copy(), print_flag=1) for x in new_explanation_candidates]
replaced_instances = []
for i in range(len(new_explanation_candidates)):
    # print("i: ", i, "new_explanation_candidates[i]: ", new_explanation_candidates[i], "new_explanation_candidates_replacements[i]: ", new_expl.
    # if isinstance(new_explanation_candidates_replacements[i][0], int):
    #     replaced_instances.append(perturb_fn(x=new_explanation_candidates[i], inst=instance.copy()))
    # else:
    replaced_instances.append(
        conditional_replace_fn(
            x=new_explanation_candidates[i],
            y=new_explanation_candidates_replacements[i],
            inst=instance.copy(),
            print_flag=1,
        )
    )
)

```

Figure 2.2-16: Novel implementation 9

```

# -----
print("len(perturbed_instances): ", len(replaced_instances))
perturbed_instances = replaced_instances

# word_sets = []
# replacement_word_sets = []
# for item in new_explanation_candidates:
#     item_word = []
#     word_replacement = []
#     for index in item:
#         item_word.append(feature_names[index])
#         word_replacement.append(get_antonyms(feature_names[index], loaded_plain_model_rf))
#     word_sets.append(item_word)
#     replacement_word_sets.append(word_replacement)
# print(word_sets)
# print(replacement_word_sets)
# #replacements = np.array(replacement_word_sets).reshape(len(replacement_word_sets), 1)
# replacements = []
# # for features in replacement_word_sets:
# #     replacements.append(OrderedSet(features))
# # replacements = [frozenset(x) for x in replacements]

# replaced_instances = []
# perturbed_instances = [perturb_fn(x, inst=instance.copy()) for x in new_explanation_candidates]
print("Expanded sentences from the above chosen combination")
for item in perturbed_instances:
    print_instance(item, instance.copy(), feature_names)
scores_perturbed_new = [cf(x, revert) for x in perturbed_instances]
## Append the newly created score array to the passes existing array
scores_candidates_to_expand_notpruned = (
    scores_candidates_to_expand + scores_perturbed_new
)
# create a dictionary of scores dictionary where the
# keys are string representations of the candidates from candidates_to_expand_notpruned, and the

```

Figure 2.2-17: Novel implementation 10

```

# create a dictionary of scores dictionary where the
# keys are string representations of the candidates from candidates_to_expand_notpruned, and the
# values are the corresponding scores from scores_candidates_to_expand_notpruned
dictionary_scores = dict(
    zip(
        [str(x) for x in candidates_to_expand_notpruned],
        scores_candidates_to_expand_notpruned,
    )
)

# *** Pruning step: remove all candidates to expand that have an explanation as subset ***
candidates_to_expand_pruned_explanations = []
candidates_to_expand_pruned_replacements_explanations = []
# take one combination from candidates
# Rewritten using list indices
# for combi in candidates_to_expand_notpruned:
#     pruning=0
#     for explanation in explanations_sets: # if an explanation is present as a subser in combi, does not add it to the to be expanded list
#         if ((explanation.issubset(combi)) or (explanation==combi)):
#             pruning = pruning + 1
#     if (pruning == 0): # If it is not a superset of a present explanation -> add it to the list
#         candidates_to_expand_pruned_explanations.append(combi)
# Write the above function using list indices
for i in range(len(candidates_to_expand_notpruned)):
    pruning = 0
    for explanation in explanations_sets:
        if (explanation.issubset(candidates_to_expand_notpruned[i])) or (
            explanation == candidates_to_expand_notpruned[i]
        ):
            pruning = pruning + 1
    if pruning == 0:
        candidates_to_expand_pruned_explanations.append(
            candidates_to_expand_notpruned[i]
        )
        candidates_to_expand_pruned_replacements_explanations.append(

```

Figure 2.2-18: Novel implementation 11.

```

candidates_to_expand_pruned_explanations_frozen = [
    frozenset(x) for x in candidates_to_expand_pruned_explanations
]
candidates_to_expand_pruned_replacements_explanations_frozen = [
    frozenset(x) for x in candidates_to_expand_pruned_replacements_explanations
]

# But the total set of frozen sets are not frozen
candidates_to_expand_pruned_explanations_ = set(
    candidates_to_expand_pruned_explanations_frozen
)
candidates_to_expand_pruned_replacements_explanations_ = set(
    candidates_to_expand_pruned_replacements_explanations_frozen
)

expanded_combis_frozen = [frozenset(x) for x in expanded_combis]
expanded_combis_ = set(expanded_combis_frozen)

# *** Pruning step: remove all candidates to expand that are in expanded_combis *** -> Same as above
candidates_to_expand_pruned = (
    candidates_to_expand_pruned_explanations_ - expanded_combis_
)
candidates_to_expand_pruned_replacements = (
    candidates_to_expand_pruned_replacements_explanations_ - expanded_combis_
)
ind_dict = dict(
    (k, i) for i, k in enumerate(candidates_to_expand_pruned_explanations_frozen)
)
indices = [ind_dict[x] for x in candidates_to_expand_pruned]
candidates_to_expand = [
    candidates_to_expand_pruned_explanations[i] for i in indices
]
candidates_to_expand_replacements = [
    candidates_to_expand_pruned_replacements_explanations[i] for i in indices
]

```

Figure 2.2-19: Novel implementation 12

```

    candidates_to_expand_pruned_replacements_explanations[i] for i in indices
]

# The new explanation candidates are the ones that are NOT in the old list of candidates to expand
new_explanation_candidates_pruned = (
    candidates_to_expand_pruned - old_candidates_to_expand
)
candidates_to_expand_frozen = [frozenset(x) for x in candidates_to_expand]
candidates_to_expand_replacements_frozen = [
    frozenset(x) for x in candidates_to_expand_replacements
]

ind_dict2 = dict((k, i) for i, k in enumerate(candidates_to_expand_frozen))
indices2 = [ind_dict2[x] for x in new_explanation_candidates_pruned]
explanation_candidates = [candidates_to_expand[i] for i in indices2]
explanation_candidates_replacements = [
    candidates_to_expand_replacements[i] for i in indices2
]

# Get scores of the new candidates and explanations.
scores_candidates_to_expand = [
    dictionary_scores[x] for x in [str(c) for c in candidates_to_expand]
]
scores_explanation_candidates = [
    dictionary_scores[x] for x in [str(c) for c in explanation_candidates]
]

return (
    explanation_candidates,
    explanation_candidates_replacements,
    candidates_to_expand,
    candidates_to_expand_replacements,
    expanded_combis,
    scores_candidates_to_expand,
    scores_explanation_candidates,
)

```

*Figure 2.2-20: Novel implementation 13*

## 2.3.2 Testing

In the development phase, the final step is Testing. It ensures the effectiveness and accuracy of our novel explanation method. Here the performance is evaluated by testing each step with inputs to return the specific output. Furthermore, the testing strategies provide a roadmap of how testing activities are conducted, how testing resources are allocated, and the goals that should be achieved in 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	To verify whether novel method provides the best counterfactual explanations for a given instance.
Test Scenario	Verify whether proposed method provides novel counterfactual explanations by specifying the remove/replace words to give an opposite sentiment.
Input	The movie was excellent, with fantastic performances and a gripping plot
Expected Output	“excellent”, ”performance
Actual Result	“excellent”, ”performance
Status (Pass/Fail)	Pass

## 3.Results and discussion

### 3.1 Results

The counterfactual opposite word replacement process, as applied to text classification tasks, yielded significant insights and practical implications. The key objective of this research was to explore the impact of word replacements on the predictive outcomes of text classification models. To evaluate the novelty of the proposed method we compared it to the SEDC method which is initiated to the implementation of our novel counterfactual explanation method. In addition, here we evaluate both positive and negative reviews to optimize the performance of novel LR methodology.

#### 3.1.1 Result evaluation for positive review:

Here we evaluate the same positive review for both the novel method and the SEDC method. The class change of the two methods is also clearly visualized in the multiple-line graph. The predefined threshold value is taken as 0.493. Moreover, in both SEDC and the novel method, to change the class label to negative, the prediction score must be less than the threshold value.

#### **Positive Review 1**



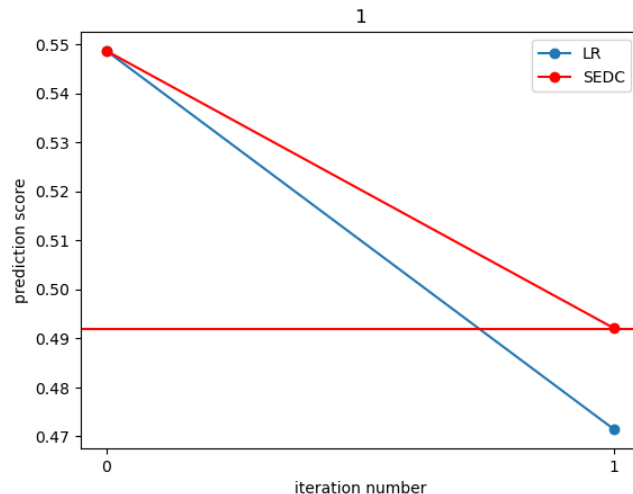


Figure 3.1-1: Class changes for positive review 1 RF explanation Vs SEDC

### Review:

**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"**

### Explanation

As depicted in the above figure, the initial prediction score of the novel LR methodology is 0.547 which means it is a positive review (threshold < 0.547). The process initiated with the removal process using the counterfactual opposite word replacement method. Then continue the removal process by rechecking the prediction score each time until the score goes below the predefined threshold value. There is a notable class change in the novel method in the first iteration by showing the final prediction score as 0.47. In the SEDC method also shows a class change in its first iteration. For the above-mentioned review, the explanation returns “funny,” “great”, “love” and “laugh” as the most affected words to make the label change from positive to negative.

## Positive Review 2

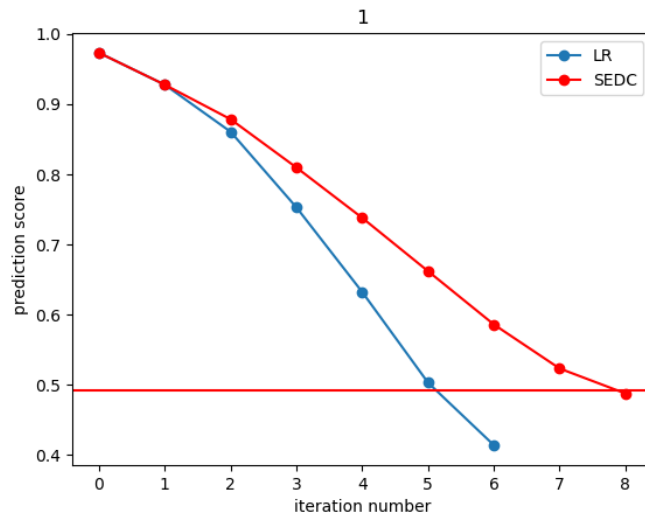


Figure 3.1-2: Class changes for positive review 2 RF explanation Vs SEDC

### Review:

**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.**

### Explanation:

As depicted in the above figure, the initial prediction score of the novel LR methodology is 0.97 which means it is a positive review (threshold < 0.97). The process initiated with the removal process using the counterfactual opposite word replacement method. Then continue the removal process by rechecking the prediction score each time until the score goes below the predefined threshold value. There is a notable class change in the novel method in the 6th iteration by showing the final prediction score as 0.43. In the SEDC method also shows a class change in its 8th iteration. For the above-mentioned review, the explanation returns “better”, “good” as the most affected words to make the label change from positive to negative.

## Positive Review 3

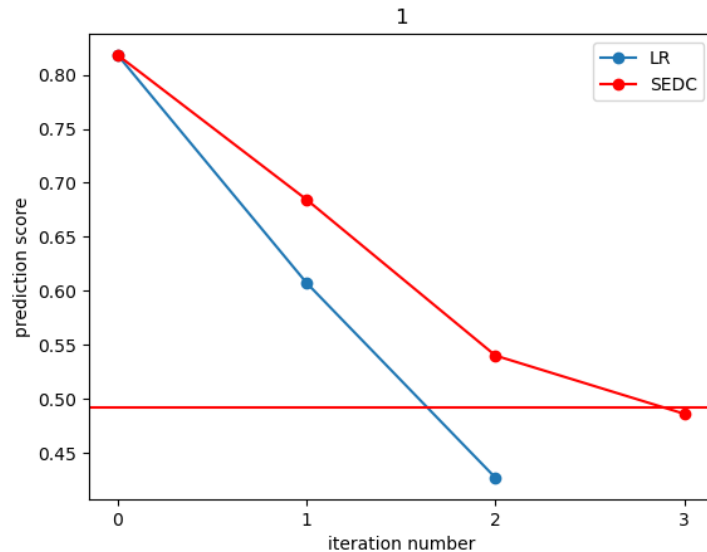


Figure 3.1-3: Class changes for positive review 3 RF explanation Vs SEDC

### Review:

**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.

### Explanation:

As depicted in the above figure, the initial prediction score of the novel LR methodology is 0.82, which means it is a positive review (threshold < 0.82). The process initiated with the removal process using the counterfactual opposite word replacement method. Then continue the removal process by rechecking the prediction score each time until the score goes below the predefined threshold value. There is a notable class change in the novel method in the 2nd iteration by showing the final prediction score as 0.22. In the

SEDC method also shows a class change in its 3rd iteration. For the above-mentioned review, the explanation returns “better”, “good” as the most affected words to make the label change from positive to negative. Here we can conclude that the novel method always provides a class change to negative.

### 3.1.2 Result evaluation for negative review:

Here we evaluate the same negative review for both the novel method and the SEDC method. The class change of the two methods is also clearly visualized in the multiple-line graph. The predefined threshold value is taken as 0.493. Moreover, in the SEDC method, there is not any class change for negative reviews. However, in our novel method, there is a clear class change for negative reviews when the prediction score is greater than the threshold value.

#### Negative Review 1

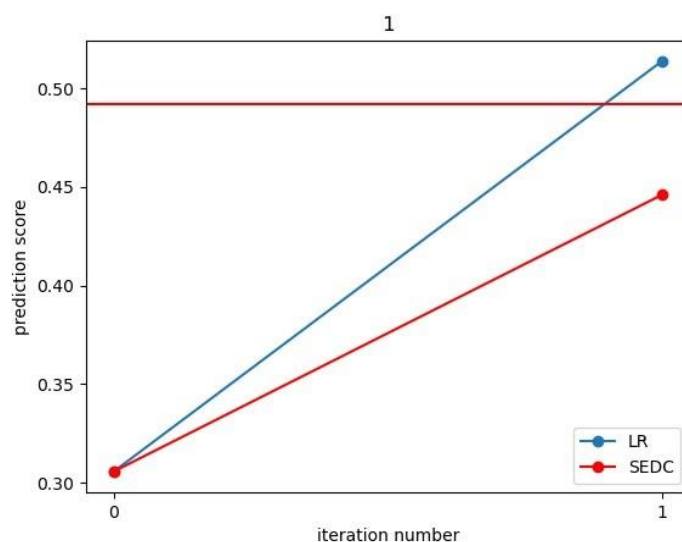


Figure 3.2-1: Class changes for negative review 1 LR explanation Vs SEDC.

#### Review:

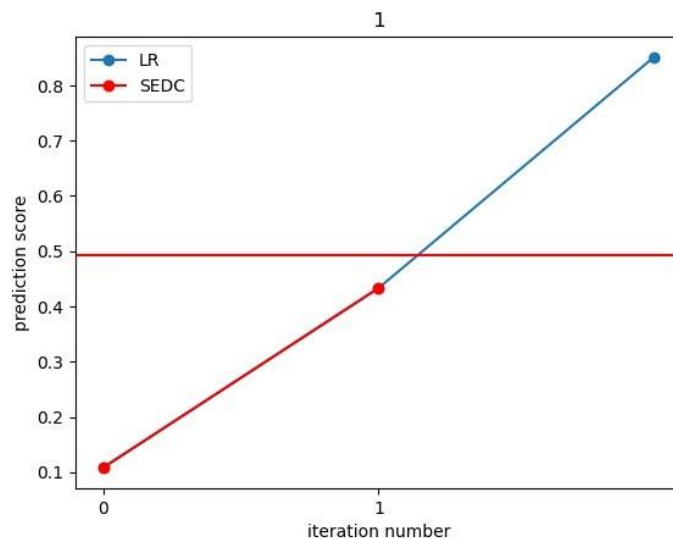
**“What can I say? I ignored the reviews and went to see it myself. Damn the reviews were so right. What a waste of money considering its budget. Good thing, I went to see Kill Bill after this one. To see a really scary movie, would be Crossroads! I like "Girl in Gold Boots" better**

than this crap.” [class label – negative].

### **Explanation:**

As depicted in the figure, the initial prediction score of the novel method is 0.30 which means it is a negative review (threshold > 0.30). The process initiated with the removal process using the counterfactual opposite word replacement method. Then continue this removal process by rechecking the prediction score each time, until the score goes up to the predefined threshold value. There is a notable class change into positive in the 1st iteration and the final prediction score is mentioned as 0.57. However, even after the maximum number of iterations, there is no class change from negative to positive in the SEDC method. The red line ends between 0.40 and 0.45 and doesn't exceed the threshold. For the above-mentioned review, the novel method returns “ignore”, waste”, “kill” and “crap” as the most affected words to the label change from negative to positive.

### **Negative Review 2**



*Figure 3.2-2: Class changes for negative review 2 LR explanation Vs SEDC.*

### **Review:**

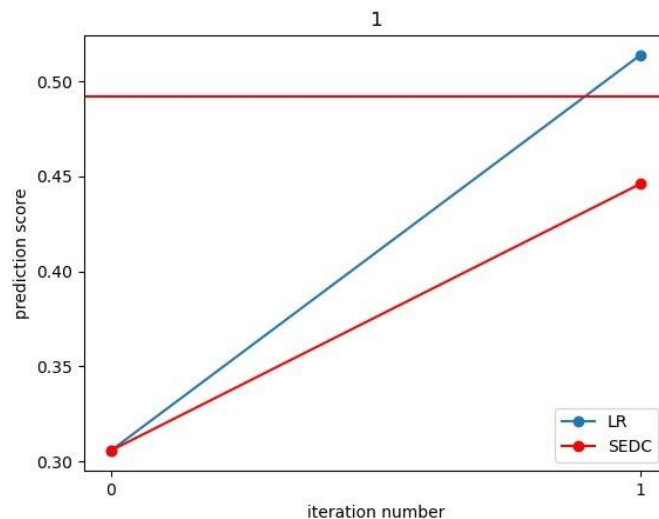
**This film might have bad production values, but that is also what makes it so good. The special effects**

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.

### **Explanation:**

As depicted in the figure, the initial prediction score of the novel method is 0.1 which means it is a negative review (threshold  $> 0.1$ ). The process initiated with the removal process using the counterfactual opposite word replacement method. Then continue this removal process by rechecking the prediction score each time, until the score goes up to the predefined threshold value. There is a notable class change into positive in the 2<sup>nd</sup> iteration and the final prediction score is mentioned as 0.86. However, even after the maximum number of iterations, there is no class change from negative to positive in the SEDC method. The red line ends between 0.4 and 0.5 and doesn't exceed the threshold. For the above-mentioned review, the novel method returns “bad” and “waste” as the most affected words to the label change from negative to positive.

### **Negative Review 3**



*Figure 3.2-3: Class changes for negative review 3 LR explanation Vs SEDC.*

### **Explanation:**

Further, we compared the SEDC method and the novel method using another set of negative reviews, and the following graph depicts the class label change that occurred in the novel method and the SEDC method for negative reviews. Here we can realize that the SEDC method does not

provide any class label change from negative to positive. However, the novel LR method always provides a class change from negative to positive.

### 3.2 Discussion

Through this research, our main goal is to implement a novel, model agnostic XAI solution to enhance the model explainability of the Logistic Regression model under the text classification domain. Here the logistic regression model becomes a black box under high dimensional data. In high-dimensional spaces, it is challenging to find the interaction effects of two or more features that are influenced for a specific prediction. So, interpreting these interactions becomes increasingly difficult as the number of potential interactions grows exponentially with the number of features. So, to overcome these drawbacks many explainable techniques can be used. This is where the term ‘counterfactual explanations’ becomes more pronounced. Counterfactual explanations illustrate hypothetical scenarios where specific feature values are changed to get a desired outcome.

Here in our proposed method, we use counterfactual opposite word replacement methodology which utilizes feature importance extracted from the LR model to identify which words are influential in making predictions. Moreover, the iterative removing process in the proposed method uncovers which specific words, when removed, lead to a change in model prediction. This helps to provide a contextual and comprehensive way of why the model classified a text the way it did.

However, this research study does not conclude here. It can be modified in numerous ways to provide remarkable innovative research ideas. One such modification is to extend the method to check for multiple numbers of opposite words in a given text. Furthermore, it can be used to identify the context of words (noun, adjective,) and assign the most suitable opposite words considering word type (The same word has different meanings in different word contexts).

These results hold the promise of delivering increasingly accurate and user-centric explainable Artificial Intelligence (XAI) solutions. In a world driven by AI technologies, these advancements are primed to meet the burgeoning needs for transparency, interpretability, and trustworthiness.

## 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"><li>• Prepare the data set and implement Support Vector Machine classifier.</li><li>• Generate a novel counterfactual rule-based mechanism using text classification domain.</li><li>• Implement the custom word flipping generator.</li><li>• Vectorize all the prompts using a TFIDF vectorizer and map into the kernel space.</li><li>• Calculate the mirror point of the original input review.</li><li>• Calculate the cosine similarity between each contradiction with the mirror point and return the closest contradiction to the mirror point.</li><li>• Results evaluation conduct for the SVM's novel explanation and RF's novel explanation. Here</li></ul>



		<p>RF's explanation compares with the SEDC method.</p> <ul style="list-style-type: none"> <li>• Frontend implementation related to the SVM's explanation.</li> </ul>
--	--	--

<p>Srinidee Methmal H.M (IT18161298)</p>	<p>Provide a novel counterfactual explanation for Logistic Regression related to the text classification domain. (Antonym replacement based counterfactual explanation)</p>	<ul style="list-style-type: none"> <li>• Prepare the data set and implement Logistic Regression classifier.</li> <li>• Develop a novel counterfactual rule-based mechanism using text classification domain.</li> <li>• Vectorize all the text using TFIDF vectorizer.</li> <li>• Antonym selection and iteration replacement and removal</li> <li>• Get the prediction score of instances and classify the class label.</li> <li>• Get the feature importance of each feature and sort.</li> <li>• Results evaluation conduct for the LR's novel explanation and RF's novel explanation. Here RF's explanation compares with the SEDC method Frontend.</li> </ul>
--	---	--

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

	(Feature density comparison based counterfactual explanation)	<ul style="list-style-type: none"> <li>• Generate a novel counterfactual rule-based mechanism using text classification domain.</li> <li>• Implement the custom word flipping generator.</li> <li>• Vectorize all the prompts using a TFIDF vectorizer and map into the kernel space.</li> <li>• Calculate the probability scores.</li> <li>• Calculate the neighbour statistics and return the best counterfactual.</li> <li>• Results evaluation conduct for the KNN's novel explanation. Here RF's explanation compares with the SEDC method Frontend.</li> <li>• implementation related to the KNN's explanation.</li> </ul>
--	---	--

## 5.Conclusion

In this study, our main focus is to examine the crucial concerns of enhancing model explainability

by using K Nearest Neighbor, Logistic Regression, Random Forest, and Support Vector Machine models. To achieve this common goal, we use a counterfactual generation mechanism with novel explainable techniques. Here our four proposed methods 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.

Here the implementation phase is accomplished through several milestones. They are model development, encountering the novel XAI solution, and developing the GUI for end users. The backend implementation of the solution is done using Google Colab and Visual Studio code using Python language. The front-end part is implemented using NestJs with Mantine UI while the backend is hosted through the AWS cloud.

The result evaluation phase is done by comparing the novel method with existing methods to improve the novel XAI solution. To accomplish this, we selected an opposite word replacement counterfactual methodology for the LR model. In the evaluation process, we did it by comparing it with the SEDC method that initiated the novel LR methodology. In the result evaluation phase for positive reviews, both the proposed method and the SEDC method show a class change.

## 6. References

- [1] 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.
- [2] A. Field, *Discovering statistics using IBM SPSS statistics.*, sage, 2013 Feb 20.
- [3] D. Liao and R. Valliant, "Condition indexes and variance decompositions for diagnosing collinearity in linear model analysis of survey data," *Survey Methodology* 38, no. 2, pp. 189-202, 2012.
- [4] D. Belsley, "A guide to using the collinearity diagnostics," *Computer Science in Economics and Management* 4, no. 1, pp. 33-50, 1991.
- [5] J. Shen and S. Gao, "A solution to separation and multicollinearity in multiple logistic regression," *Journal of data science: JDS* 6, no. 4, p. 515, 2008 Oct 10.
- [6] Y. Asar, "Some new methods to solve multicollinearity in logistic regression," *Communications in Statistics-Simulation and Computation* 46, no. 4, pp. 2576-2586, 2017.
- [7] R. Schaefer, L. Roi and R. Wolfe, "A ridge logistic estimator," *Communications in Statistics-Theory and Methods* 13, no. 1, pp. 99-113, 1984 Jan 1.
- [8] N. Senaviratna and T. Cooray, "Diagnosing multicollinearity of logistic regression model," *Asian Journal of Probability and Statistics* 5, no. 2, pp. 1-9, 2019 Oct 1.
- [9] S. Rana, H. Midi and S. Sarkar, "Validation and performance analysis of binary logistic regression model.," in WSEAS Press, 2010.
- [10] H. Midi, S. Sarkar and S. Rana, "Collinearity diagnostics of binary logistic regression model," *Journal of interdisciplinary mathematics* 13, no. 3, pp. 253-267, 2010.
- [11] Molnar, Christoph. "Interpretable machine learning. A Guide for Making Black Box Models Explainable", 2019. <https://christophm.github.io/interpretable-ml-book/>
- [13] 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.