Project Final Report: Stance Detection System for combating Fake News

Group member: LuMingrui (Group 24)

Student ID: A0307208L

1. Introduction:

In today’s digital age, the spread of misinformation, commonly known as fake news, has become a critical challenge. Social media platforms and news outlets often face the challenge of manually filtering out false information. To combat this, there is an increasing need for intelligent systems that can automatically detect the stance of news articles relative to their headlines and classify them as truthful or false, which is a critical step in identifying fake news. The goal of my project is to design an automated intelligent system that helps news platforms and fact-checking agencies identify fake news efficiently using advanced AI models and natural language processing techniques.

2. Business Case / Market Research:

Market Landscape: The fake news detection industry has been growing quickly in recent years, as the spread of false information has become a serious problem. Many companies, such as Factmata and FactCheck, are working on tools to help identify and prevent the spread of fake news. Holver, most of these tools still depend heavily on people to check and verify the information. This process can be slow and prone to human error.

With the rise of AI, there is an opportunity to use machine learning to speed up the detection of fake news. AI can process large amounts of data quickly, make decisions consistently, and reduce the chance of bias that human checkers might introduce. My project focuses on using advanced AI techniques, like Gradient Boosting, Support Vector Machines, and BERT ,to automatically analyse the relationship between a headline and the body of a news article. By doing this, I can classify whether the article agrees, disagrees, discusses, or is unrelated to the headline, helping to identify potential misinformation.

Market Needs and Opportunities: Based on my search there is a strong and growing need for tools that can:

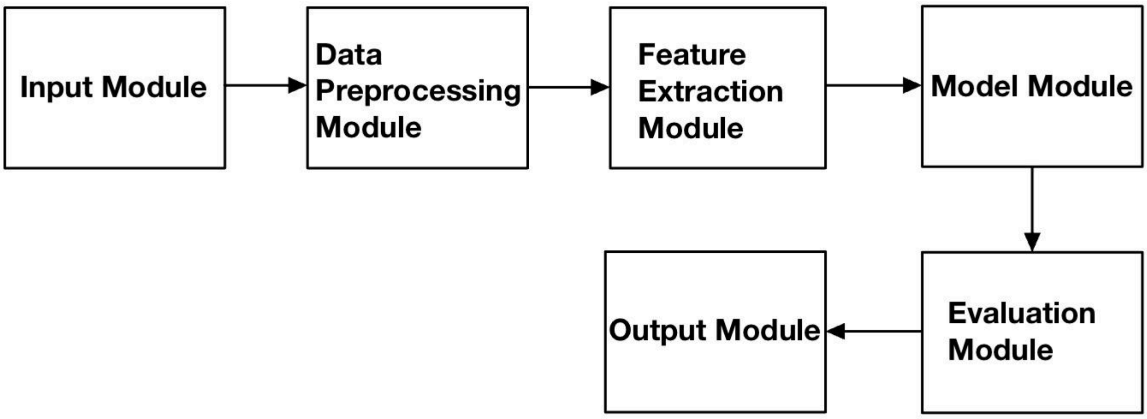
- Automatically detect fake news with high accuracy, reducing the time and effort required by human fact-checkers.
- Provide real-time insights to media platforms, allowing them to stop the spread of misinformation before it reaches a large audience.
- Handle large volumes of data, making them suitable for use by media companies, social media platforms, and government agencies.

The proposed system is designed to meet these needs by offering a fast, reliable, and scalable solution that can be easily integrated into the workflow of media companies and other organizations involved in fact-checking. By automating the process of detecting misinformation, my system will help these organisations save time, reduce costs, and ensure that false information is caught early.

3. System Design And Development:

System Overview: The intelligent reasoning system for fake news detection comprises several components, including data collection, preprocessing, machine/deep learning models, and user interfaces. The system is designed to process news articles and classify their stance relative to the headline.

System Architecture Diagram: The system architecture consists of:



- **Input Module**: Receives news articles, including both headlines and body texts.
- **Preprocessing Module**: Cleans, tokenizes, and normalises the text data.
- **Feature Extraction Module**: Extracts features such as word embeddings, TF-IDF, and sentiment scores.

- **Model Module**: Applies machine learning models, including SVM, Gradient Boosting, and BERT, for classification.
- **Evaluation Module**: Evaluates the performance of different models using metrics like accuracy and F1-score to determine the best-performing model for final predictions.
- **Output Module**: Provides the stance classification along with a confidence score for each prediction.

a. Dataset Analysis and Pre-processing:

Input: The dataset contains pairs of a headline and a body text, either from the same or different news articles. The task is to classify the stance of the body text in relation to the headline into one of four categories:

- **Agrees**: Body text agrees with the headline.
- **Disagrees**: Body text disagrees with the headline.
- **Discusses**: Body text discusses the same topic but without taking a clear position.
- **Unrelated**: Body text discusses an unrelated topic.

FNC-1 Dataset: The dataset is split into two CSV files:

- **train_bodies.csv**: Contains the body text of articles and their corresponding IDs.
- **train_stances.csv**: Contains labelled stances (agree, disagree, discuss, or unrelated) paired with the corresponding body IDs. The data distribution is imbalanced, with the majority class being "unrelated."

Data Pre-processing: To prepare the text data for modelling, several steps are taken:

- **Normalisation**: Words are reduced to their base form using lemmatization to minimise vocabulary diversity, ensuring words like "running" and "ran" are normalised to "run."
- **Tokenization**: Each text is broken down into individual tokens, which is essential for text processing tasks.
- **Cleaning**: Non-alphanumeric characters are removed, and text is converted to lowercase to reduce noise.
- **Stopword Removal**: Common words like "the" and "is" are removed as they don’t contribute meaningful information for classification.

Class Imbalance: During the experiment I found that the dataset has an uneven distribution of stance categories, so I applied techniques such as oversampling to add more examples from the minority classes “agree”, “disagree”, and “discuss” to balance the dataset. And I have also tried undersampling to reduce the number of examples in the majority class “unrelated” to avoid model bias. After testing both methods, oversampling proved to be more effective in improving the overall model accuracy.

Feature Engineering

- **TF-IDF Cosine Similarity**: I used TF-IDF (Term Frequency-Inverse Document Frequency) to measure the importance of words within the text and calculated the cosine similarity between the headline and the body text to assess their relevance.
- **Word2Vec**: Word embeddings from Word2Vec Ire used to represent words in a dense vector space. The average similarity between the headline and body vectors was computed to capture semantic meaning.
- **Sentiment Analysis**: Sentiment scores (positive, negative, neutral) for both headlines and bodies are extracted to understand the emotional tone of the articles.
- **Dimensionality Reduction (SVD)**: Singular Value Decomposition was applied to the TF-IDF features to reduce dimensionality and capture key patterns in the text.

BERT Tokenization: For the BERT model, I used the pre-trained **BERT tokenizer** from the transformers library to prepare the headlines and body texts Ire combined into a single sequence, with the special token "[SEP]" inserted between them to help the model differentiate between the two parts. The tokenizer also handled padding and truncation to ensure a consistent input length. After tokenization, the text was encoded into input IDs and attention masks, which Ire then converted into PyTorch tensors for use in training the model. This preprocessing ensures that the BERT model can effectively capture the relationship between the headline and the body text.

b. Model Training Description:

Gradient Boosting and Boosting Series (XGBoost, CatBoost, LightGBM):
For the Boosting models, I aimed to establish a solid baseline and then progressively optimise the performance of the system. Gradient Boosting was first used as a baseline for classification tasks. Afterward, I experimented with advanced boosting models like XGBoost, CatBoost, and LightGBM, all of which are designed to handle complex datasets with high accuracy and efficiency.

For **XGBoost**, I applied various parameter tuning techniques, including adjusting max_depth, subsample, and colsample_bytree to optimise performance. A 10-fold cross-validation strategy was used to ensure robust evaluation. The model was trained on multiple folds and validated on the remaining one, with the best-performing fold used to evaluate both the holdout and competition datasets.

CatBoost was also applied using the MultiClass loss function. Like XGBoost, the features were standardised, and a 10-fold cross-validation approach was used. Both XGBoost and CatBoost were further optimised based on their performance, with the most successful models selected for final evaluation.

LightGBM is another powerful gradient boosting framework, and was also incorporated into the model ensemble. It was trained with a multi-class objective, designed to handle the four stance detection categories. LightGBM was chosen for its ability to handle large datasets efficiently and its faster training time compared to other boosting models, making it highly suitable for real-world applications. LightGBM followed the same 10-fold cross-validation approach, and the best-performing model was evaluated against the holdout and competition datasets. Its ability to efficiently handle categorical features and offer quicker training times made it a crucial component of the final model selection process.

Support Vector Machine (SVM): A Support Vector Machine is a widely used supervised learning algorithm for both classification and regression problems. This model was implemented using the Scikit-learn library to classify the stance of news articles based on their headline and body text. A 10-fold cross-validation approach was also employed. Features for each fold were generated using a custom feature extraction process, and standardised using the StandardScaler to ensure consistent scaling across features. An SVM with an RBF kernel was chosen for its ability to handle non-linear relationships in the data. The model was trained on the training folds and validated on the remaining fold, with performance evaluated after each iteration. The best-performing fold model was selected for final evaluation on the holdout dataset and competition dataset, with accuracy scores reported. This process alloId for robust model training and testing across different subsets of the data.

BERT Model Training: For training the BERT model, I first define the processing device, which could be a GPU if available, or a CPU otherwise, to take advantage of faster computations when possible. The BERT model is instantiated for sequence classification using the “BertForSequenceClassification” class from the transformers library. The number of output labels is set to four to match the stance detection categories: unrelated, discuss, agree, and disagree. After defining the model, it is moved to the appropriate device using .to(device). Next, I configure the optimizer and learning rate scheduler. I select the AdamW optimizer with a learning rate of 2e-5, and I use a linear learning rate scheduler with a warm up phase to adjust the learning rate dynamically during the training process. The training loop is executed over a predetermined number of epochs. During each epoch, I iterate over batches from the DataLoader. For each batch, the input IDs, attention masks, and labels are moved to the processing device. Gradients are reset before performing a forward pass through the model. After computing the loss, which is based on the cross-entropy loss function, I backpropagate the gradients and update the model parameters using the optimizer. Additionally, gradient clipping is applied to prevent the gradients from exploding and ensure stable training. At the end of each epoch, the average training loss is recorded.

During my experimentation with hyper-parameter tuning for the BERT model, I tested different learning rates, including 1e-5, 2e-5, and 5e-5. I found that a learning rate of 2e-5 provided the best results, so it was chosen as the optimal rate for training. The number of epochs, which represents the full passes over the training dataset, was also carefully adjusted. After experimenting with 5, 10, 15, and 20 epochs, I found that the model converged effectively after 15 epochs, leading us to finalise the training process at 20 epochs to avoid underfitting or overfitting. Additionally, I tested various optimizers and determined that AdamW, an improved variant of the Adam optimizer that includes light decay regularisation, performed better than other alternatives like SGD. Thus, AdamW was selected as the final optimizer for my model.

c. Model evaluating and Results Discussion:

Overview of Evaluation Methodology: my evaluation methodology is designed to thoroughly assess the model's performance using a weighted, two-level scoring system that reflects the complexity of fake news detection tasks.

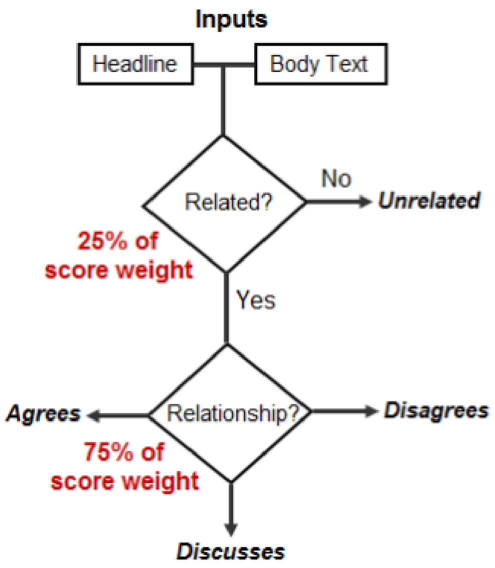
Scoring System: The scoring is divided into two levels:

- **Related or Unrelated Classification**: This task identifies whether the headline and body text are related or not. Correctly classifying this relationship contributes 25% of the total score, as it is less complex and not directly focused on detecting fake news.
- **Stance Detection**: For related pairs, the model further classifies the stance as agreeing, disagreeing, or discussing. This task, which contributes 75% to the total score, is more challenging and essential for understanding how information is presented.

Implementation Details:

- **Confusion Matrix**: This visual tool helps track model performance across the different stance classes, highlighting specific strengths and weaknesses.
- **Score Calculation**: Each correct classification of related/unrelated adds 0.25 to the score, while identifying the exact stance adds 0.75 for related pairs.
- **Performance Metrics**: Metrics such as accuracy, precision, recall, and F1-score offer a detailed view of the model's effectiveness, particularly in dealing with class imbalances.

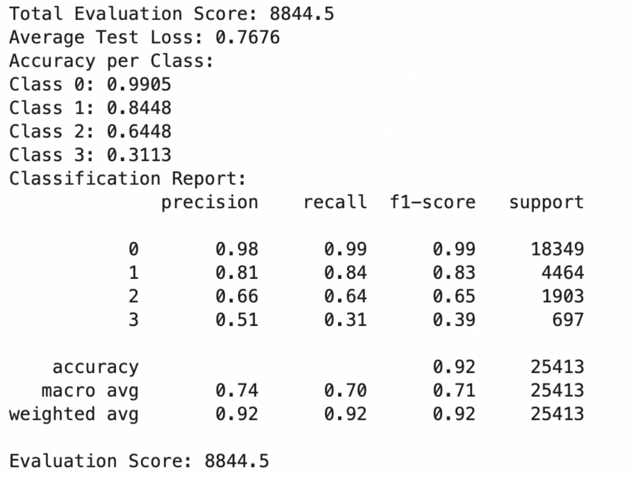
This two-tiered evaluation provides a balanced analysis of the model's ability to handle both the simpler classification of relatedness and the more nuanced stance detection. It is particularly well-suited to the complexities of fake news detection, where accuracy in both broad categorization and detailed stance identification is critical.



Bert Test Data Processing and Evaluation:

I began by loading the test data from the CSV files containing article bodies and their corresponding stances, merging them based on the 'Body ID' column. Headlines and body texts were concatenated with a "[SEP]" token and tokenized using the BERT tokenizer, ensuring sequences were padded or truncated to 128 tokens. The tokenizer’s encode_plus method generated input IDs and attention masks, which were then converted to tensors. Stance labels were mapped to integers (0 for 'unrelated', 1 for 'discuss', 2 for 'agree', 3 for 'disagree'). Finally, a PyTorch DataLoader was created to batch process the test data for evaluation.

During the evaluation phase, I switched the BERT model to evaluation mode to disable certain layers like dropout layer. Using the DataLoader, I passed the test data through the model to obtain logits for each batch, and the loss was calculated using the cross-entropy function. To assess stance detection performance, I use the custom scoring system mentioned above: correctly predicted labels earned 0.25 points, with an additional 0.50 points for non-'unrelated' correct predictions. If both true and predicted labels were related but did not match exactly, 0.25 points were awarded. Finally, the evaluation score, average test loss, accuracy per class, and classification report were printed to offer a comprehensive assessment of the BERT model's stance detection performance.



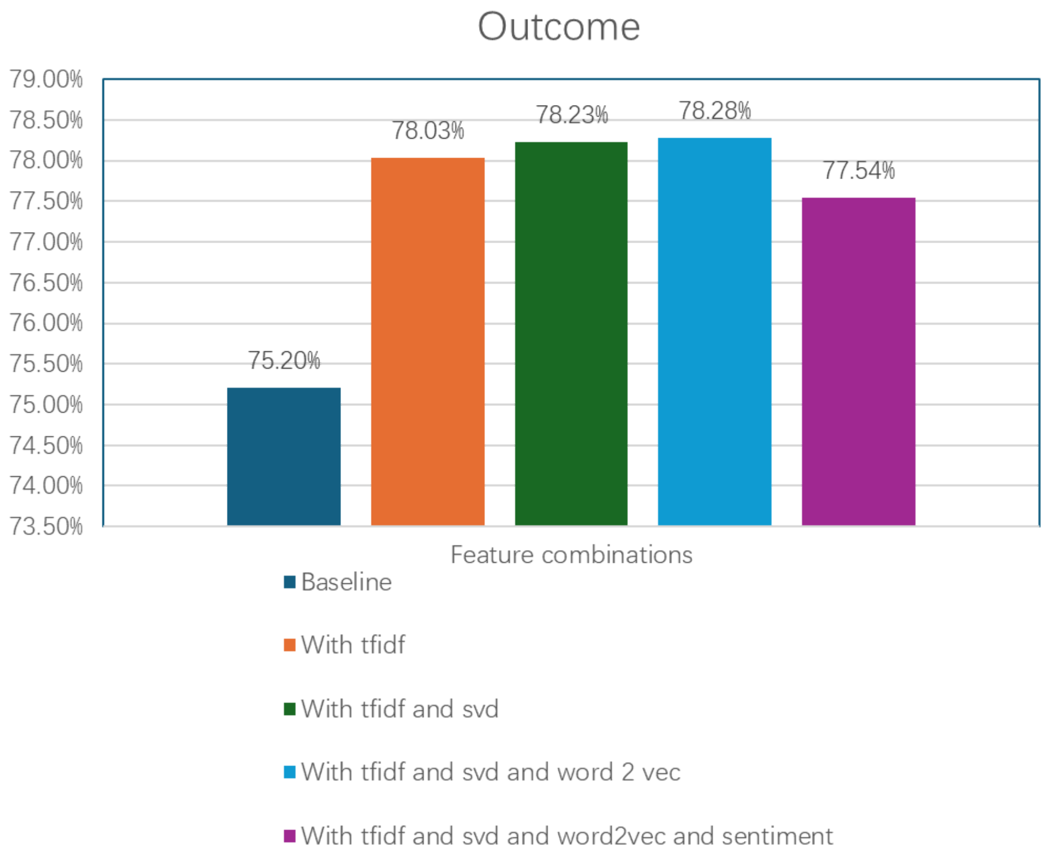
```
Total Evaluation Score: 8844.5
Average Test Loss: 0.7676
Accuracy per Class:
Class 0: 0.9905
Class 1: 0.8448
Class 2: 0.6448
Class 3: 0.3113
Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.99      0.99     18349
           1       0.81      0.84      0.83      4464
           2       0.66      0.64      0.65      1903
           3       0.51      0.31      0.39       697

    accuracy                           0.92     25413
   macro avg       0.74      0.70      0.71     25413
weighted avg       0.92      0.92      0.92     25413

Evaluation Score: 8844.5
```

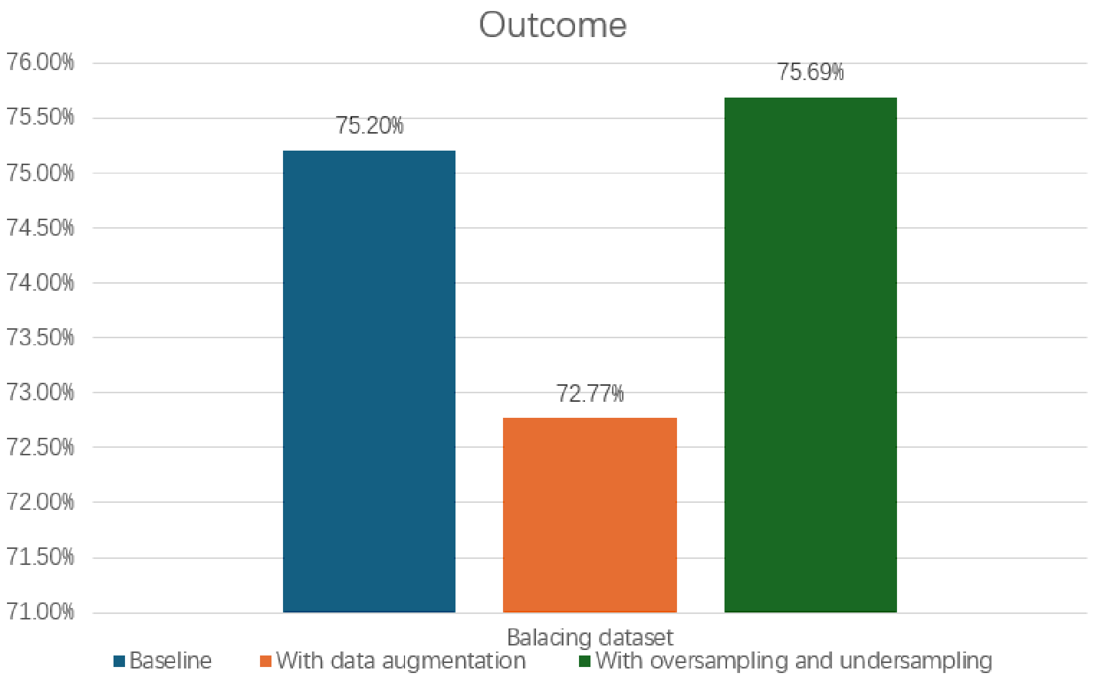
Results Discussion:

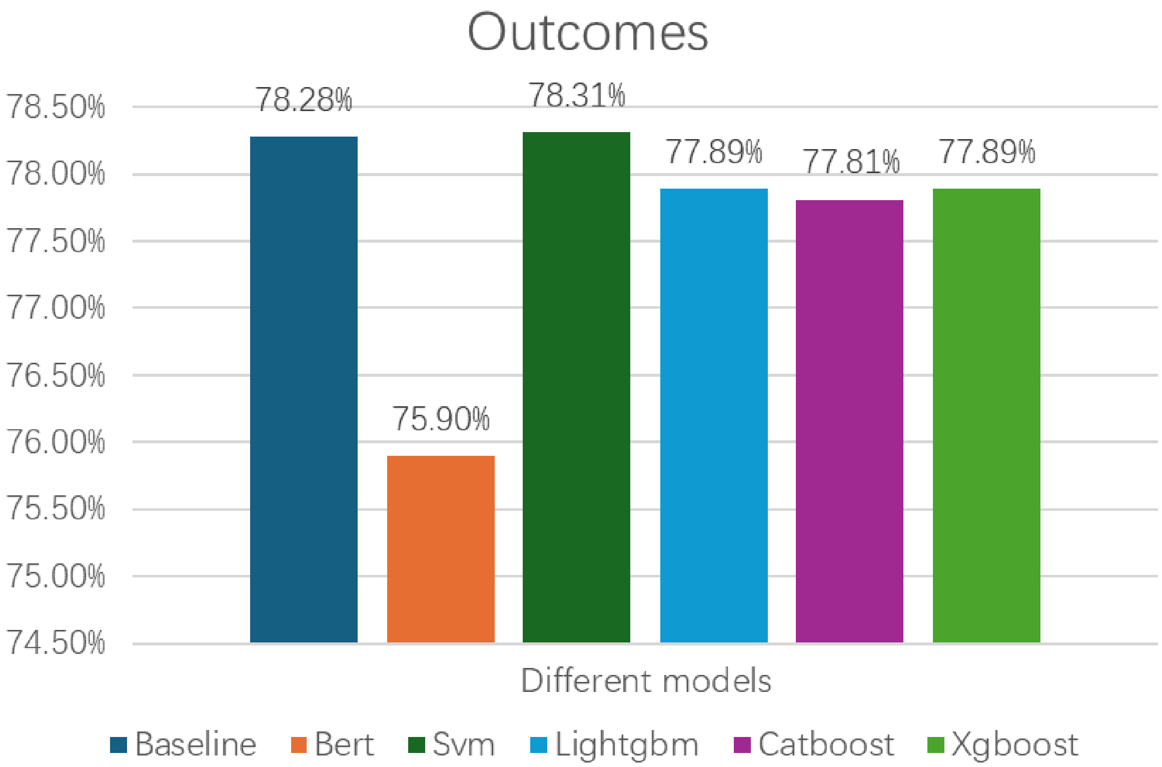
In implementing the model, I utilised key techniques from the baseline model, such as data loading, k-fold cross-validation, and various feature generation methods. Building on this foundation, I made strategic modifications and ran experiments to enhance the model's performance.

A key focus of my efforts was balancing the dataset to reduce any biases caused by uneven class distributions. I also expanded the feature set by including a wider range of data characteristics, aiming to capture more subtle patterns in the text. In addition to traditional machine learning algorithms, I explored deep learning transformer based models to take advantage of their strength in recognizing patterns and feature interactions. My experiments include parameter tuning and testing different loss functions to improve the performance of my classifiers. However, I learned that adding more features doesn’t always lead to better accuracy. In fact, too many features can create noise, with less useful or repeated information confusing the model. Therefore, while balancing the data is important, I also need to focus on the quality of the data.

The charts below illustrate the comparative outcomes of my experiments. They show that while some strategies yielded significant improvements, others offered only minor gains, and some even had a negative impact. These highlight the importance of careful feature selection and a balanced approach to building models. Additionally, they provide a clear comparison of the overall accuracy across all the models I experimented with, helping to identify the most effective approach for this task.

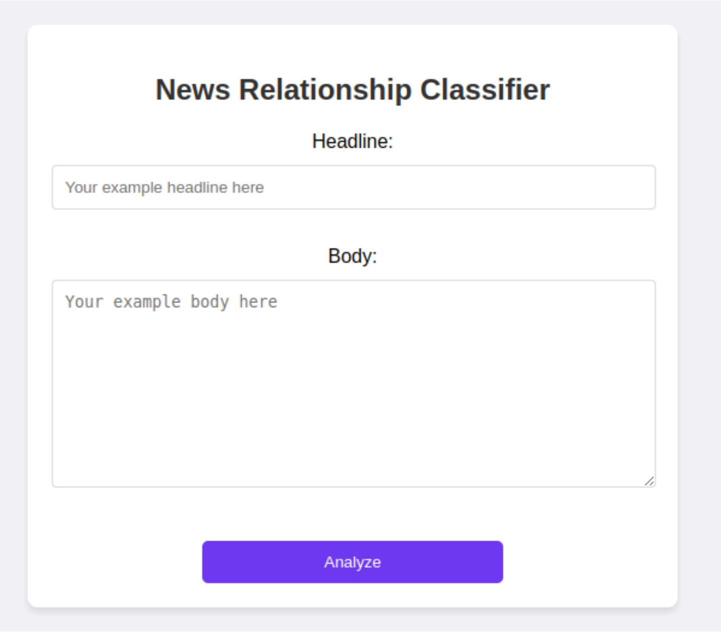



Outcome
76.00%
75.50%
75.00%
74.50%
74.00%
73.50%
73.00%
72.50%
72.00%
71.50%
71.00%
75.20%
72.77%
75.69%
Balacing dataset
Baseline
With data augmentation
With oversampling and undersampling

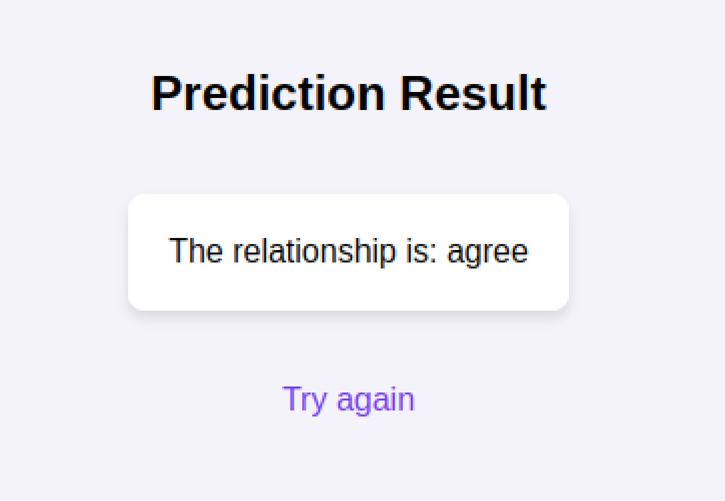

Outcomes
78.50%
78.00%
77.50%
77.00%
76.50%
76.00%
75.50%
75.00%
74.50%
78.28%
75.90%
78.31%
77.89%
77.81%
77.89%
Different models
Baseline
Bert
Svm
Lightgbm
Catboost
Xgboost

4. User Interface Demonstration:

This image is shown below the user interface of the "Stance Detection System" a web-based tool designed to analyse the connection between a news headline and its body text. The tool is integrated with the best-performing model from my previous experiments, ensuring accurate and reliable results based on the tested models. The interface is simple and easy to use, with two input fields for users to enter the headline and the body of the article. The 'Headline' field is for the title of the news, while the 'Body' field is for the full text of the article. Placeholder texts like "Your example headline here" and "Your example body here" help guide users on where to input their data. After clicking the 'Analyze' button, the application processes the information using a model to determine the relationship, such as whether the body agrees with, disagrees with, or is unrelated to the headline.



The second image shows the results page of the "Stance Detection System" after the analysis is finished. The result is displayed clearly, with the message "The relationship is: agree," indicating that the news body agrees with the headline based on the model's analysis. There is also a 'Try again' link, allowing users to return to the main page and input a new headline and body for analysis.



Prediction Result

The relationship is: agree

5. Conclusion:

In summary, my project successfully developed a stance detection system that utilise the advanced AI models such as SVM, XGBoost, and BERT. Through extensive experimentation with data preprocessing, feature engineering, model training, evaluation, and deployment in a real-world application. I achieved a reliable system capable of analysing the relationship between news headlines and their corresponding bodies.

This project has the potential to make a significant impact in the fake news detection market. By automating the process of stance detection, the system offers a scalable, real-time solution that can save time and resources for media platforms and fact-checking organisations. Furthermore, its ability to process large volumes of data with high accuracy makes it suitable for various real-world applications in the digital media landscape. The models and techniques explored in my project provide a foundation for real-world systems aimed at combating misinformation. By integrating these AI models into practical tools, my project showcases the value of academic research in addressing pressing challenges in today's information-driven world.