# LLM - Detect AI Generated Text

Mohammadreza Seidgar

## Abstract

In this project, we aimed to develop and compare machine learning models for the classification of two categories: those written by students and those generated by large language models (LLMs). Leveraging state-of-the-art natural language processing and deep learning techniques, our approach involved two types of methods:one based on deep learning methods, and another based on non-neural network methods.
Our models were trained and evaluated based on a dataset that included essays written by students and a limited number of generated essays. Results from our experiments demonstrating the effectiveness of our proposed approach. This study makes a significant contribution to the field of natural language processing,machine learning and provides valuable insights for future developments in related areas.

## Introduction

The dataset for this project consists of four CSV files: `sample_submission.csv`, `test_essays.csv`, `train_essays.csv`, and `train_prompts.csv`.
For our analysis, we specifically utilize the `train_essays.csv` dataset, which includes essential columns such as `id`, `prompt_id`, `text`, and `generated`.

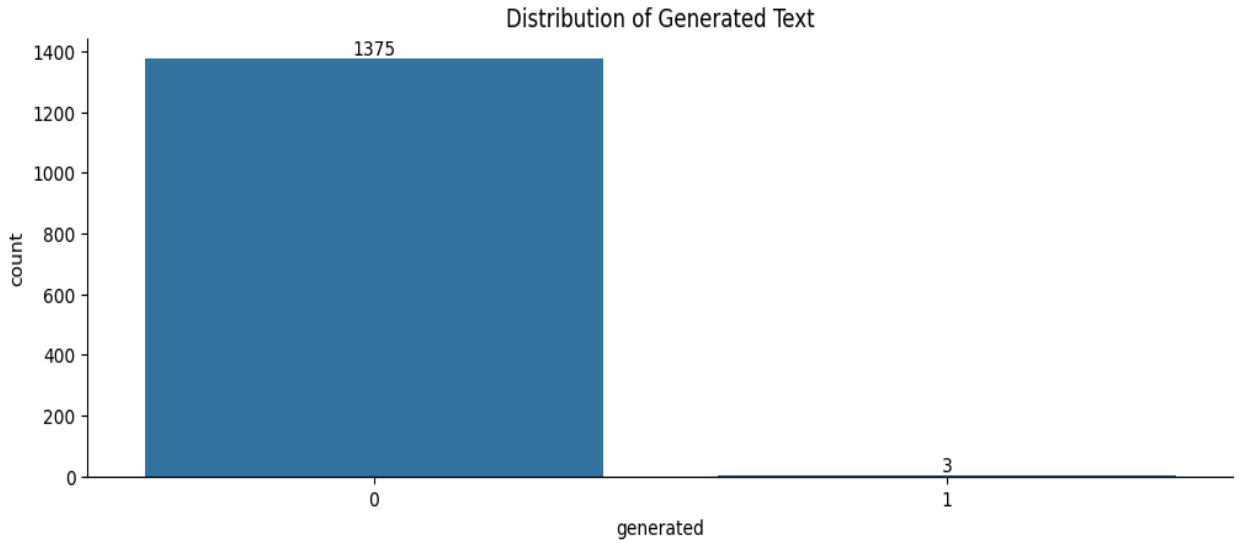
The `text` and `generated` columns are of particular significance to us, as our primary objective is to predict whether a given text, based on its content (`text`), is generated by a Large Language Model (LLM) or not.

## Dataset Overview:

Total Number of Essays: 1378

Distribution of Categories:

- Category 0 (Student-Written Essays): 1375 essays
- Category 1 (LLM-Generated Essays): 3 essays

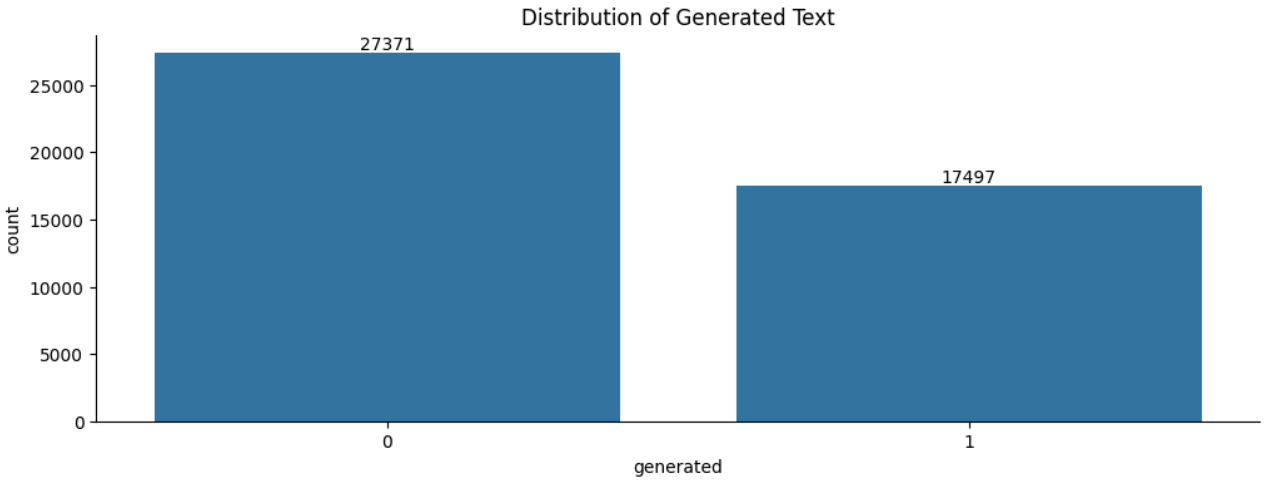


**Number of Categories:**

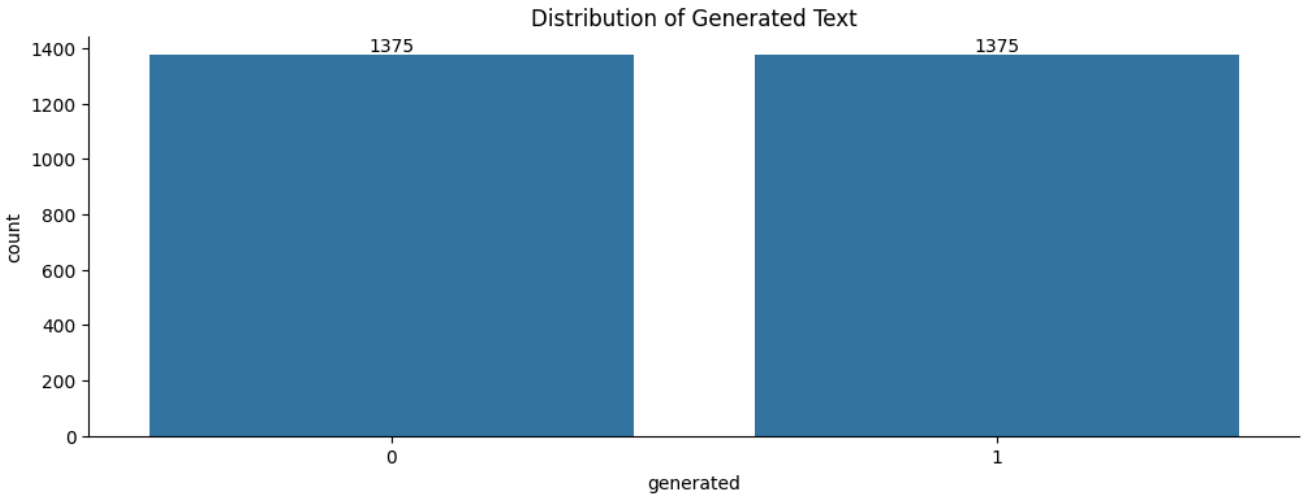
The dataset is binary, consisting of two categories: 0 (Student-Written) and 1 (LLM-Generated).

The above distribution illustrates an imbalance in the dataset, as only 3 essays are identified as LLM-generated.
To address this imbalance, we augment our dataset with additional LLM-generated data from the `daigt-v2-train-dataset`.

Distribution of daigt-v2-train-dataset:




By incorporating 1372 LLM-generated essays from this dataset into our main dataset, we successfully balance our data.




This project's focus lies in leveraging machine learning techniques to classify essays based on their origin—whether written by students or generated by LLMs. The challenges posed by the dataset's class imbalance necessitate careful consideration in model development and evaluation. Through this research, we aim to contribute insights to the field of natural language processing and enhance our understanding of the distinctions between student and LLM-generated content.

## Preprocessing Data

In this section, we discuss the essential preprocessing steps that prepare our text data for the implementation of pre-trained BERT models. Given the task's objective—distinguishing between student-written and LLM-generated essays—we opt to leverage pre-trained BERT models for enhanced performance.

Tokenization with **BERT**:

To align our text with BERT's expectations, we employ the `DistilBertTokenizer` from the Hugging Face library. This tokenizer facilitates the conversion of input text into a format suitable for BERT by introducing special tokens such as \[CLS\] (classification), \[SEP\] (separator), and \[PAD\] (padding).
The key parameters of the tokenizer play a pivotal role in shaping the tokenized output:

***- padding***: Appends padding tokens to ensure sequences are of uniform length.

***- max_length***: Specifies the maximum length of each sequence. For our actual dataset, we set this to 512, the maximum length allowed for BERT sequences.

***- truncation***: Truncates tokens exceeding the maximum length.

***- return_tensors***: Dictates the type of tensors returned; we opt for PyTorch tensors.

The resulting outputs from the tokenizer, particularly the `input_ids`, `token_type_ids`, and `attention_mask`, are integral for subsequent model input:

- ***input_ids*:** Representing the id of each token, the tokenizer automatically adds special tokens like CLS, SEP, and PAD. The maximum length setting results in PAD tokens at the end.
- ***token_type_ids*:** This binary mask identifies the sequence to which a token belongs. For a single sequence, all token type ids are 0. In our text classification task, this parameter is optional.

- ***attention_mask*:** This binary mask distinguishes real words from padding. Tokens with CLS, SEP, or any real word have a mask of 1, while padding or PAD tokens have a mask of 0. These outputs are crucial for inputting data into our BERT model effectively.

We utilize the pre-trained `DistilBertTokenizer` from the 'distilbert-base-uncased' model, which is effective for English text. For multilingual datasets, such as German, Dutch, Chinese, Japanese, or Finnish, it is advisable to use 'bert-base-multilingual-cased' or a language-specific pre-trained tokenizer.

**Preprocessing Overview:**

The preprocessing code, summarized below, outlines the steps taken to prepare our dataset for training and evaluation:

***1. Tokenization***: Utilizing the `DistilBertTokenizer`, we tokenize the text, considering parameters such as padding, maximum length, and truncation.

***2. Data Splitting***: The dataset is split into training and test sets using the `train_test_split` function.

***3. Dataset and Dataloader Construction***: We construct a custom dataset, `EssaysDataset`, and embed the datasets into dataloaders using PyTorch's `DataLoader`.

These preprocessing steps ensure that our data is appropriately formatted for training BERT-based models, paving the way for effective model learning and evaluation.

## Model Building

To construct our neural network model, we employ the DistilBERT architecture using the `DistilBertModel` from the Hugging Face Transformers library. The model is customized for binary classification by adding a dropout layer and a linear layer. The parameters of the DistilBERT model are frozen to retain pre-trained knowledge.

The training loop consists of five epochs. We utilize the CrossEntropyLoss as the loss function and Adam optimizer for backpropagation. During training, we monitor both training and validation losses along with accuracy.

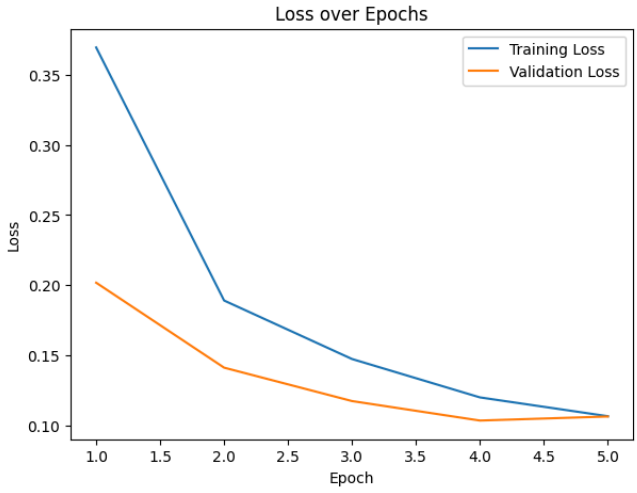
In the Path failed section of the code, we exactly performed the same preprocessing steps and built the model. The key difference was the use of the `bert-base-cased` model in this section, which apparently is a much heavier and more complex model. Training this model required significantly more time. Therefore, we decided not to remove this section from the code but did not execute it.

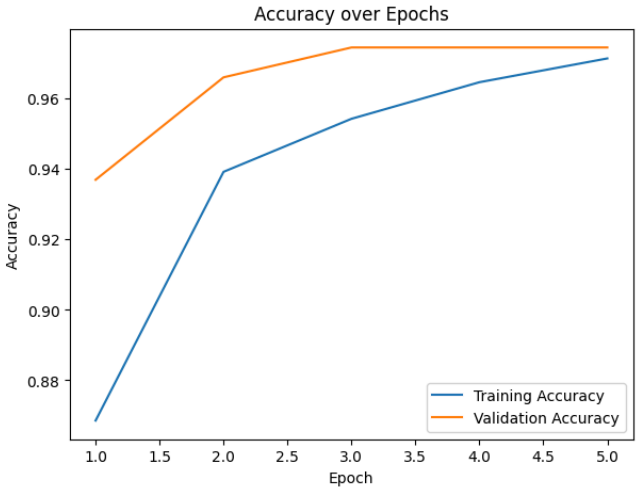
## Training Loop

The training loop iterates through each epoch, optimizing the model's weights based on the calculated loss. The accuracy and loss metrics are recorded for both the training and validation sets. The process is executed for five epochs.
The final training time for the model is approximately 229 minutes.

## Model Evaluation

The training and validation losses, as well as accuracy, are plotted over the epochs in two separate graphs. The training loss decreases, and accuracy increases, indicating the model's learning and generalization capabilities. The validation metrics show consistent performance, suggesting that the model is not overfitting to the training data.
The achieved accuracy on the validation set is around 97.45% in the final epoch, demonstrating the effectiveness of the DistilBERT-based model for the binary classification task.



## Non-Neural Network Model Evaluation

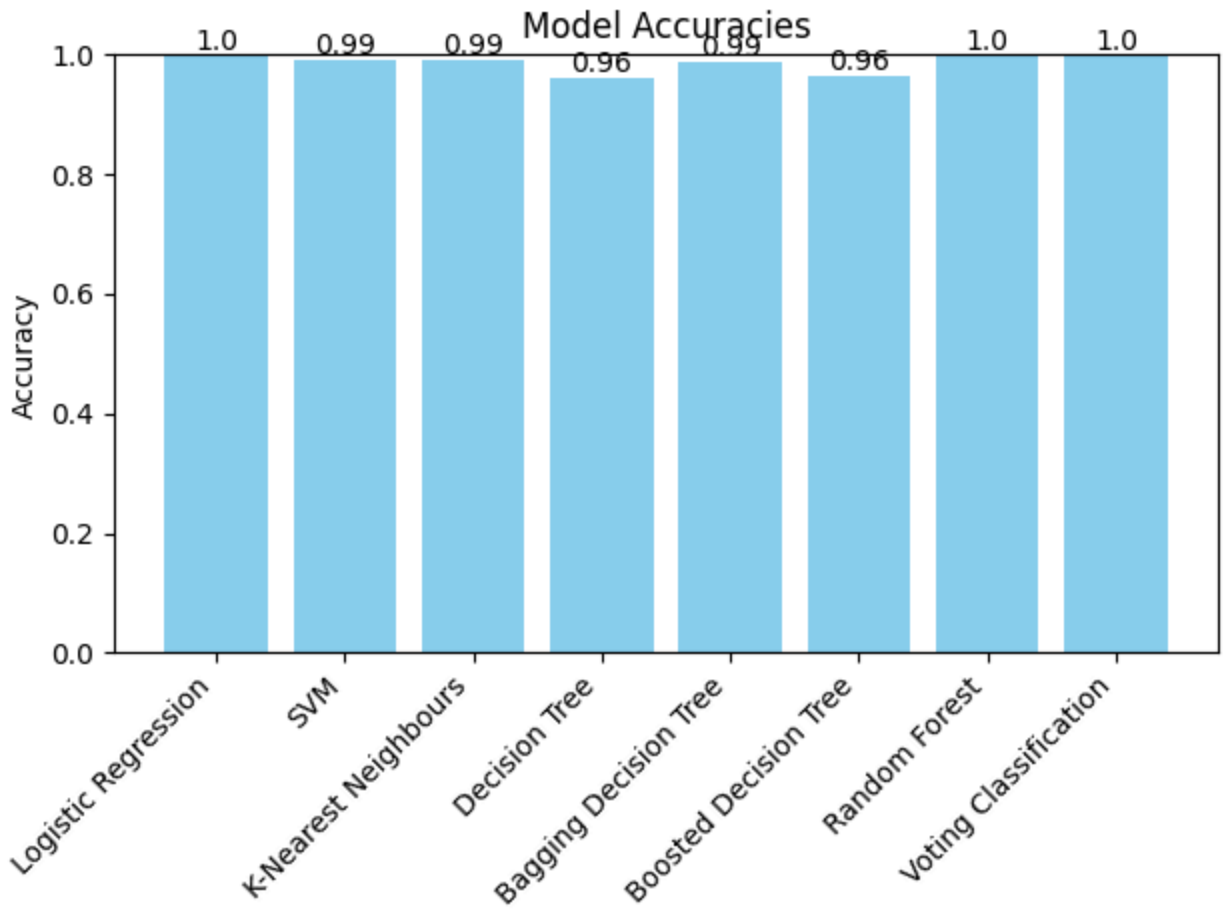
We extended our analysis beyond neural network models to explore the performance of traditional machine learning models. Utilizing the DistilBERT embeddings as features, we trained and evaluated several classifiers. Here are the results:

- Logistic Regression Accuracy: 1.00
- SVM Accuracy: 0.99
- K-Nearest Neighbors Accuracy: 0.99
- Decision Tree Accuracy: 0.96
- Bagging Decision Tree Accuracy: 0.99
- Boosted Decision Tree Accuracy: 0.96
- Random Forest Accuracy: 1.00
- Voting Classification Accuracy: 1.00

The Voting Classification model, comprises the following individual models:

- Logistic Regression (logreg_model)
- Support Vector Machine (svm_model)
- Bagging Decision Tree (bagging_tree_model)

The models are combined using a Hard Voting strategy within the VotingClassifier.

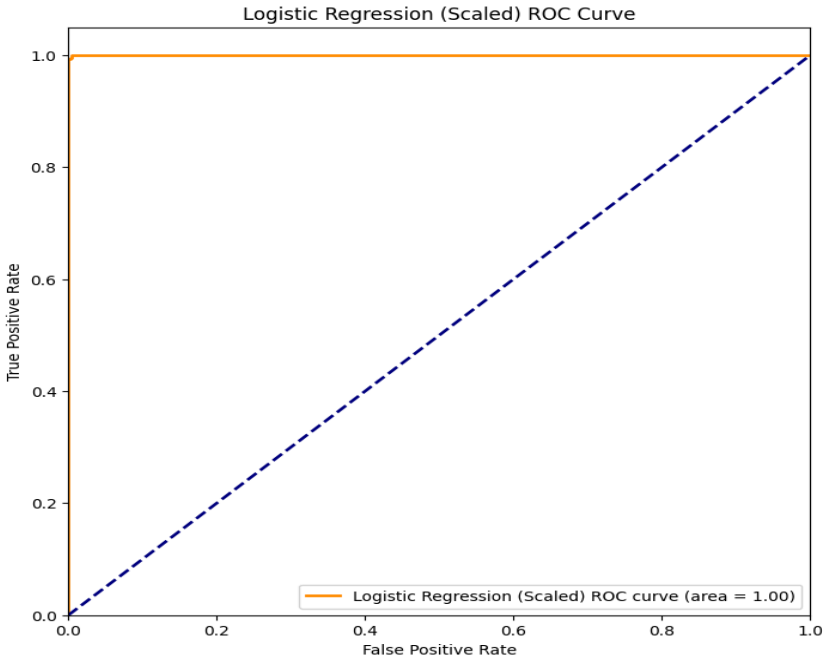



The bar chart above visually represents the accuracies of these models. The LR ,RF,VC models exhibit the highest accuracy among the evaluated models. It's crucial to note that these accuracies may vary depending on the specific dataset characteristics and preprocessing steps. Further experimentation and fine-tuning could enhance the overall effectiveness of the models.

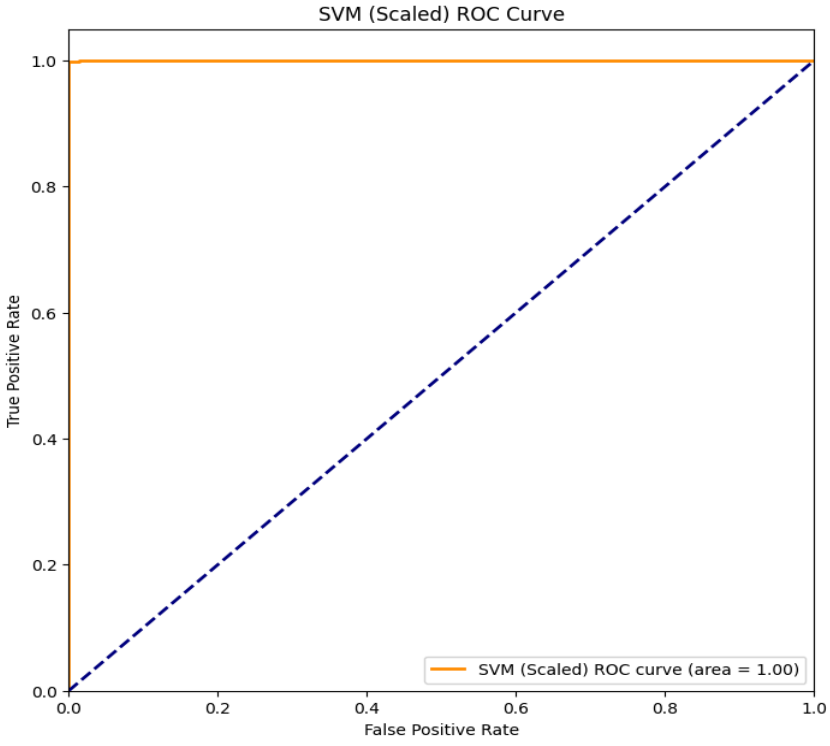
## ROC-AUC

In the end, we calculate and plot ROC-AUC scores for various models. Here is a summary:

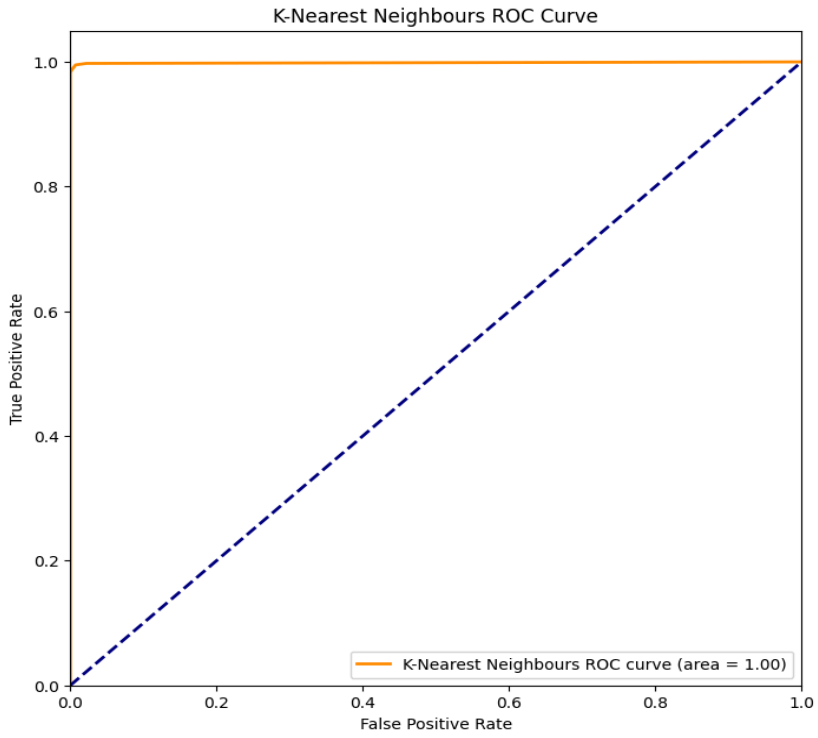
1. Logistic Regression (Scaled):



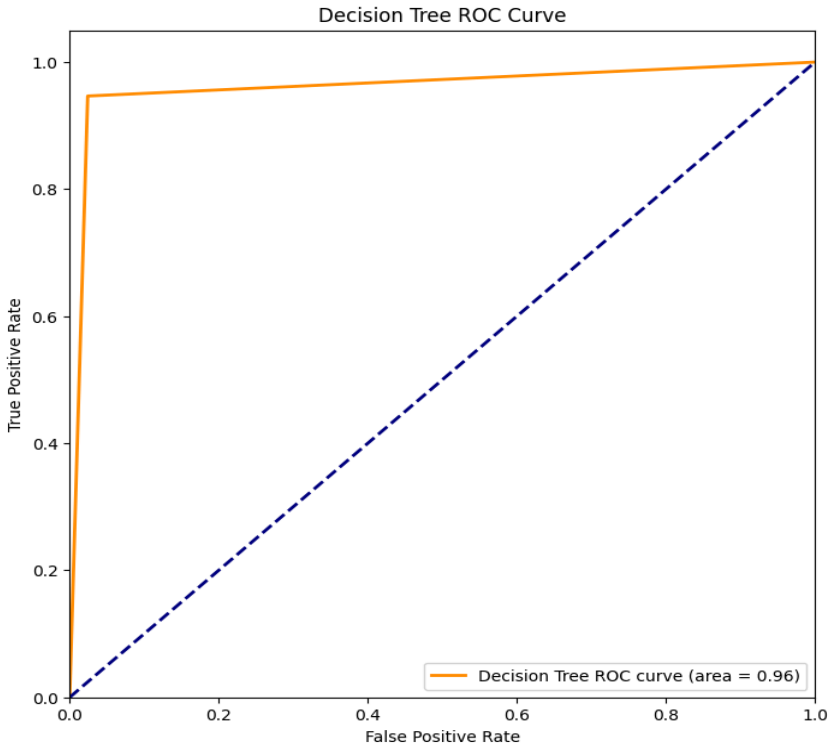

2. SVM (Scaled):



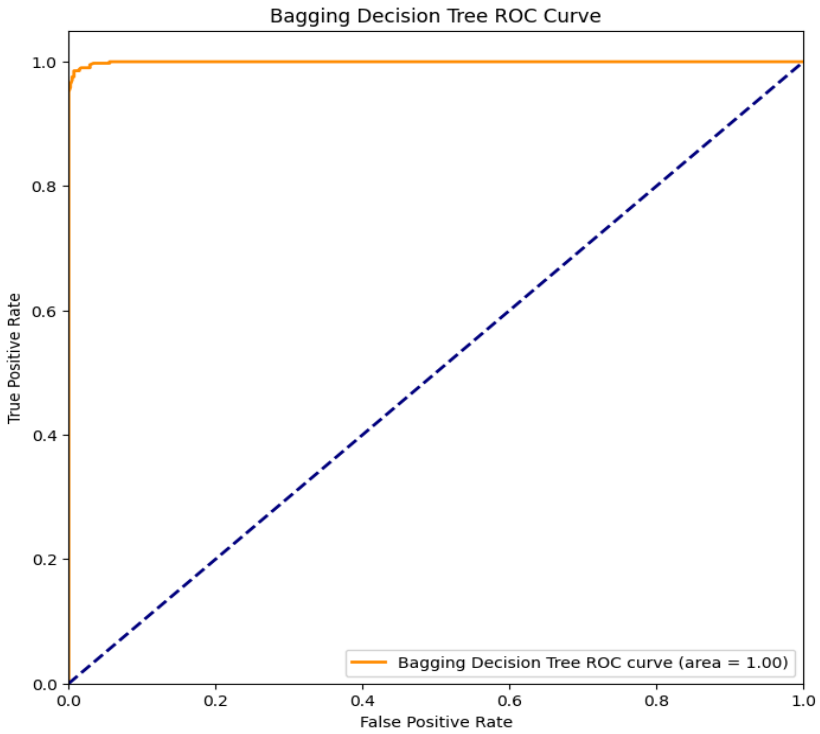
3. K-Nearest Neighbours:



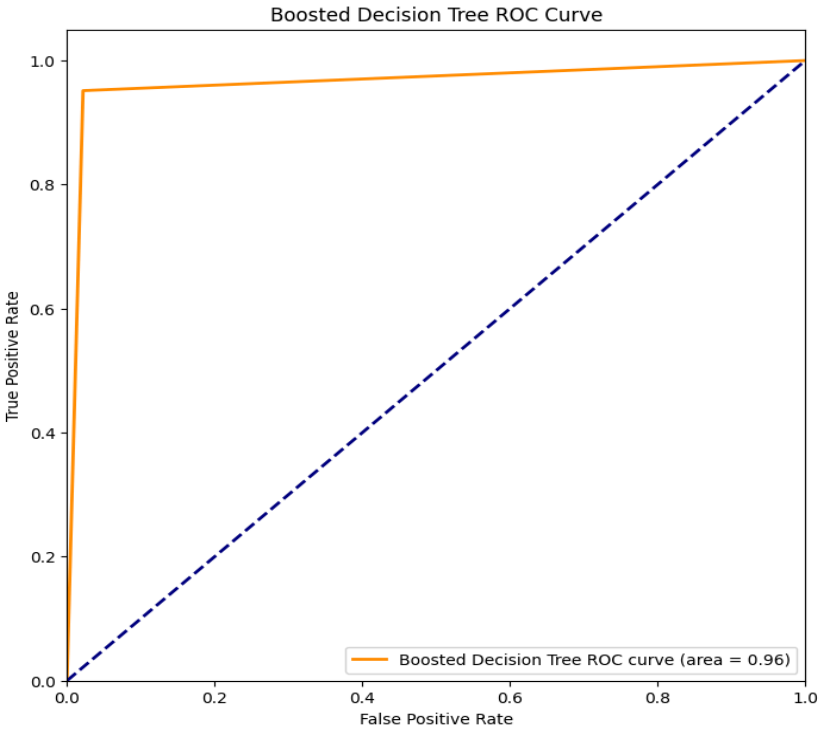

4. Decision Tree:



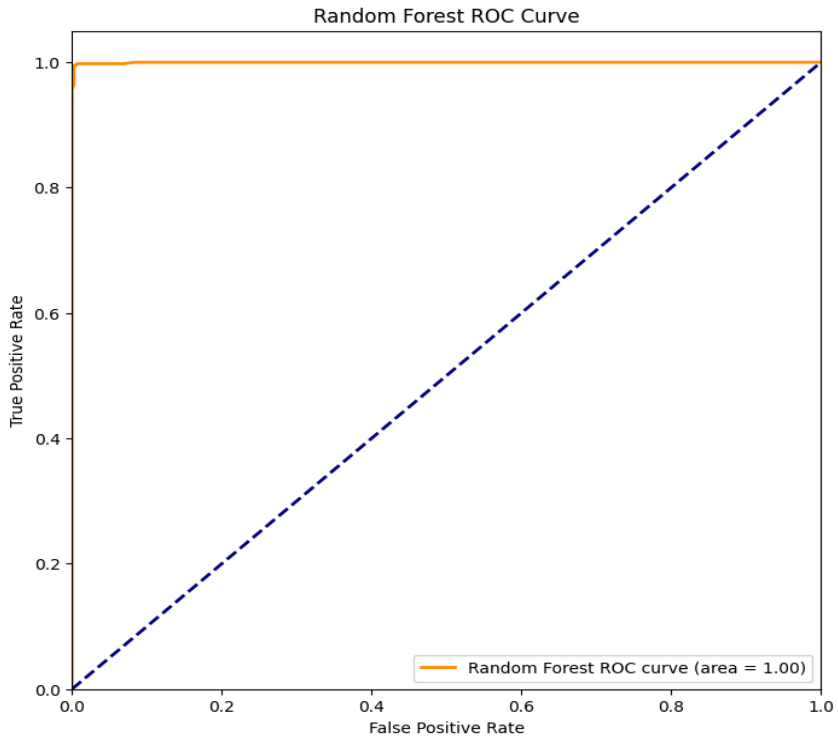
## 5. Bagging Decision Tree:



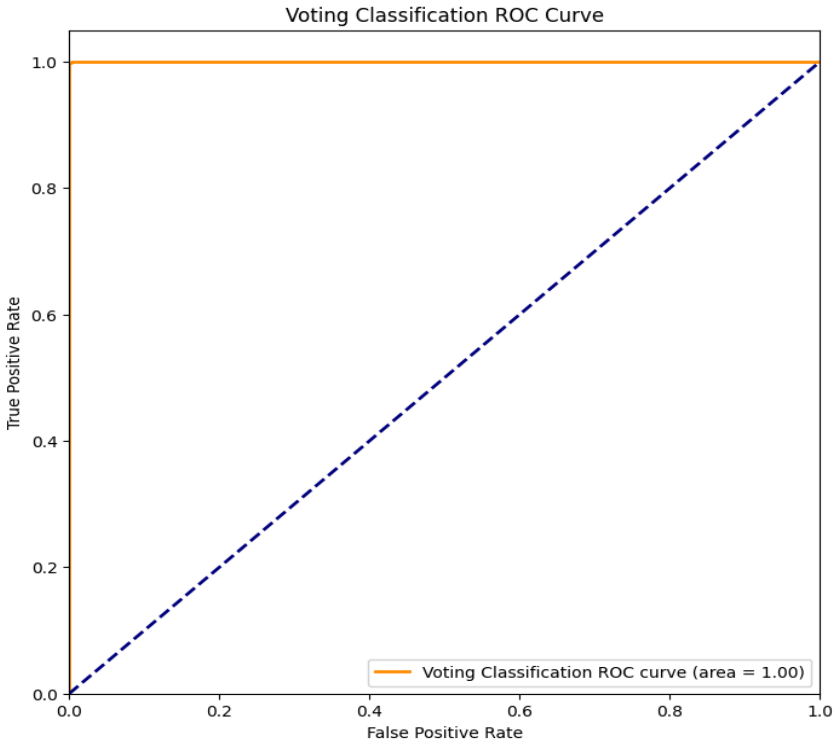

## 6. Boosted Decision Tree:



7. Random Forest:




10. Voting Classification:



Each section of the code follows a similar structure: fitting the model, predicting probabilities, calculating ROC-AUC, and finally, plotting the ROC curve. The models include standard and ensemble techniques, providing a comprehensive evaluation of their performance.

## Conclusion

In this study, we explored the task of distinguishing between student-written essays and those generated by Large Language Models (LLMs). Leveraging advanced techniques such as pre-trained DistilBERT embeddings and traditional machine learning models, we conducted a comprehensive analysis.

Our investigation involved the development and evaluation of neural network models, including a DistilBERT-based architecture. Additionally, we explored non-neural network models such as Logistic Regression, SVM, K-Nearest Neighbours, Decision Tree, Bagging Decision Tree, Boosted Decision Tree, Random Forest, and a Voting Classification ensemble.

Among the models, the  Logistic Regression ,Random Forest ,Voting Classification models demonstrated the highest accuracy, highlighting its effectiveness in discerning between student-generated and LLM-generated essays. However, the performance of models can be influenced by various factors, including dataset characteristics, model hyperparameters, and the complexity of the task.

This study serves as a foundation for further research and model refinement. Fine-tuning models, experimenting with different embeddings, and exploring additional features may contribute to improved performance. Moreover, ongoing

advancements in natural language processing and machine learning techniques could offer new avenues for enhancing the accuracy and robustness of models in similar tasks.