SASTRA

ENGINEERING · MANAGEMENT · LAW · SCIENCES · HUMANITIES · EDUCATION

DEEMED TO BE UNIVERSITY

(U/S 3 of the UGC Act, 1956)

THINK MERIT | THINK TRANSPARENCY | THINK SASTRA

THANJAVUR | KUMBAKONAM | CHENNAI

Project Title
Detect LLM generated Text

Submitted by:
B.Jaganath
125018031
B.Tech Computer Science and business systems
Sastra Deemed University
Thanjavur

Table of Contents

Project Abstract :

The aim of the project is to build a model to identify which essay was written by middle and high school students, and which was written using a large language model. With the spread of LLMs, many people fear they will replace or alter work that would usually be done by humans. Educators are especially concerned about their impact on students' skill development, though many remain optimistic that LLMs will ultimately be a useful tool to help students improve their writing Skills.

This project aims to develop an automated system capable of accurately detecting whether an essay is generated by an LLM or written by a human. The approach leverages natural language processing (NLP) techniques and machine learning algorithms to analyze linguistic patterns, coherence, structure, and stylistic nuances unique to both sources. By training on diverse datasets of essays from LLMs and human authors, the model identifies distinguishing features, offering a reliable tool for educational institutions and content verification platforms. The system aims to maintain high accuracy while adapting to evolving LLM capabilities.

Introduction :

Project Objectives :The primary objective of this project is to design and implement a robust, scalable, and efficient system that can accurately distinguish between essays generated by large language models (LLMs) and those written by humans. We implement ML algorithms like clustering to group essays which are written by humans and those generated by LLM's.
To achieve this, the project will focus on:

1. Data Collection: Curate a diverse dataset of LLM and human-authored essays, preprocess the text, and extract relevant features for analysis.
2. Feature Engineering: Identify linguistic patterns such as sentence complexity, coherence, and stylistic elements that differentiate human and LLM-generated essays.

3. Model Development: Train machine learning models, including supervised classifiers and deep learning techniques, to classify essays based on extracted features.

4. Performance Optimization: Ensure the model achieves high accuracy, precision, recall, and robustness across various LLMs and human writing styles.

Problem formulation :

With the rapid advancement of large language models (LLMs), such as GPT, AI-generated text has become increasingly indistinguishable from human-written content. This presents a significant challenge, particularly in contexts like education, content verification, and media, where the ability to distinguishing between human and AI-generated writing is crucial.

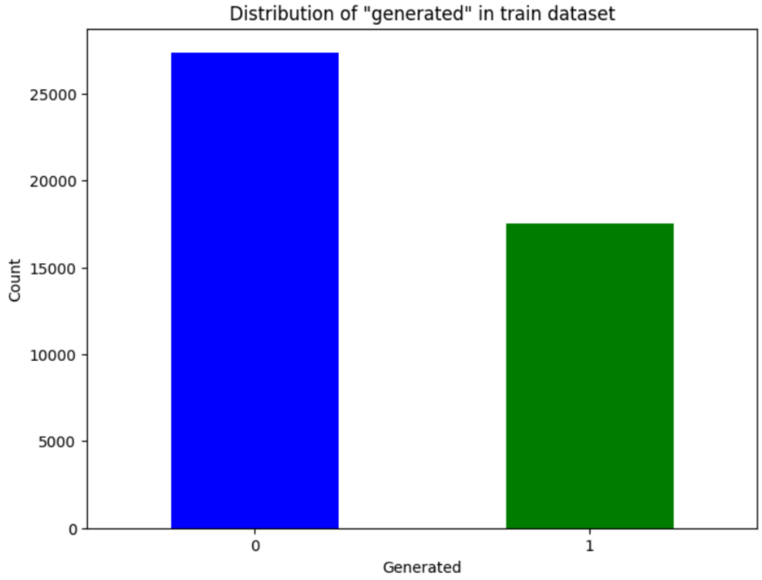
The core problem is the lack of reliable methods to automatically identify whether an essay was written by a human or generated by an LLM. Human writing tends to exhibit unique characteristics such as creativity, personal voice, and variability in structure, while LLMs often produce highly coherent but sometimes overly formulaic or generic text. The challenge is heightened by the continuous improvement in LLMs, making them better at mimicking human-like writing.

Methodology :

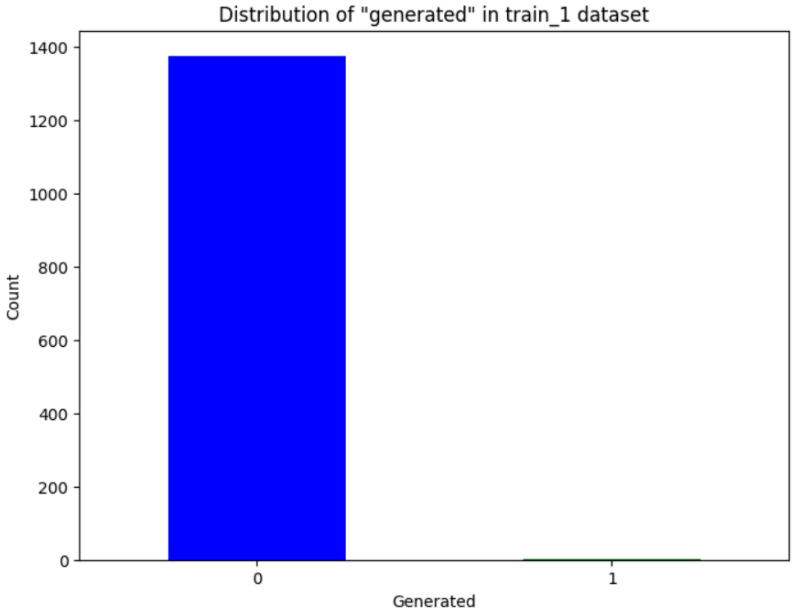
Dataset :

The dataset comprises about 1,00,000 essays, some written by students and some generated by a variety of large language models (LLMs) equally divided between those two . The goal of the competition is to determine whether or not an essay was generated by an

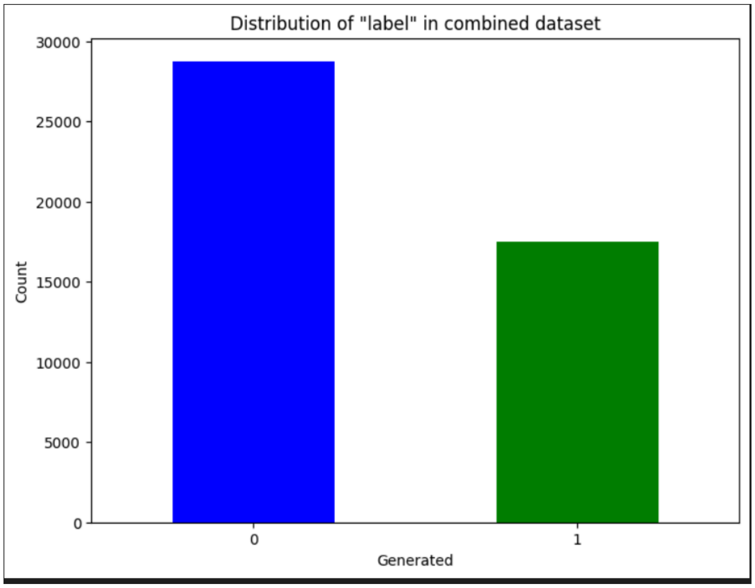
LLM. Here, I combined 3 datasets to increase my dataset size.



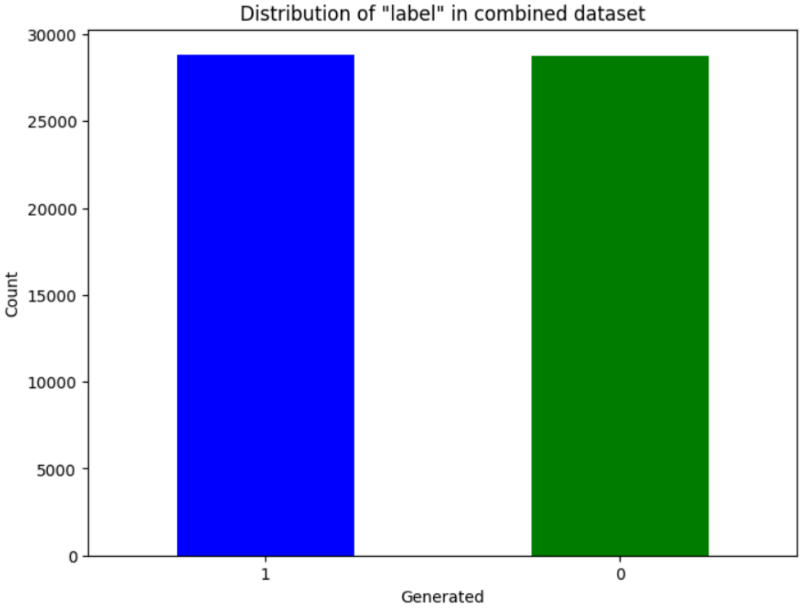
This is my dataset 1 where 1 denotes LLM generated essays and 0 denotes student written essays.



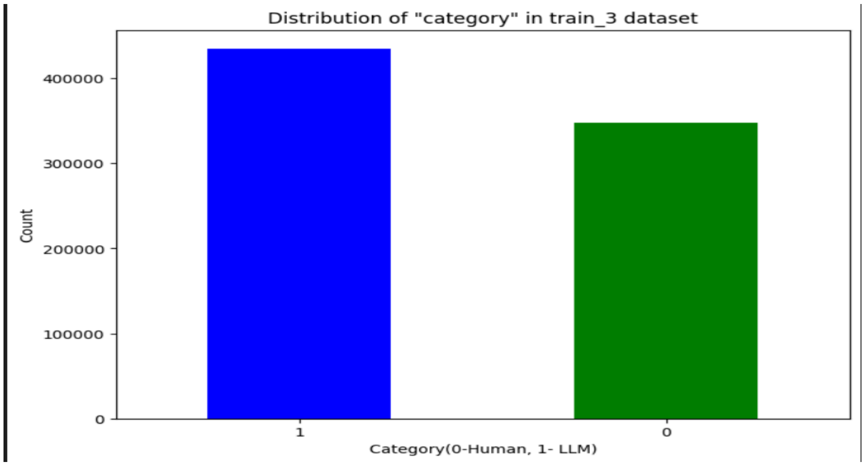
This is the dataset which comprises only Human written essays.



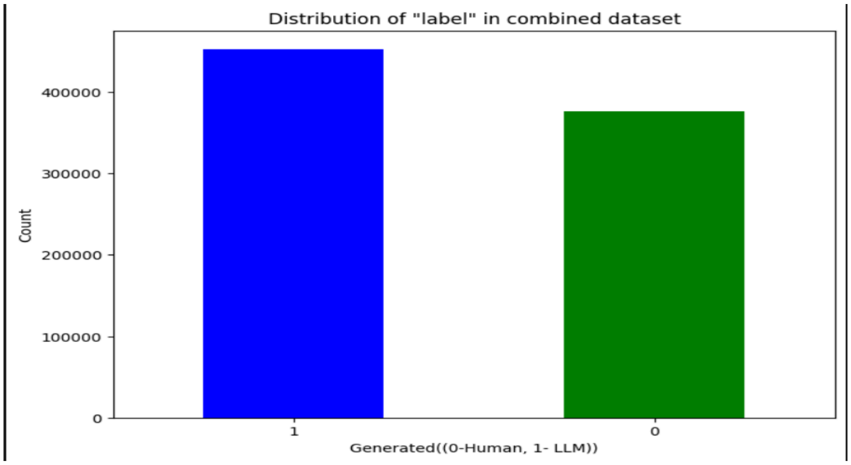
Distribution of new dataset combining dataset1 and dataset 2.



This is my third dataset where 0 denotes human written essays and 1 denotes LLM generated essays.

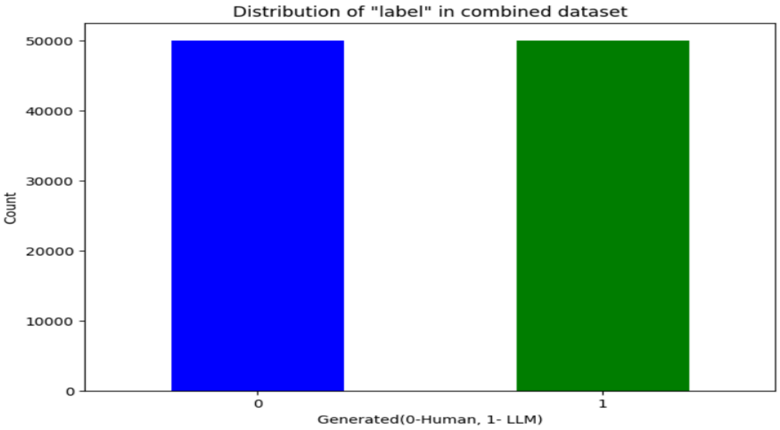


So after combining all 3 datasets , this is my final dataset



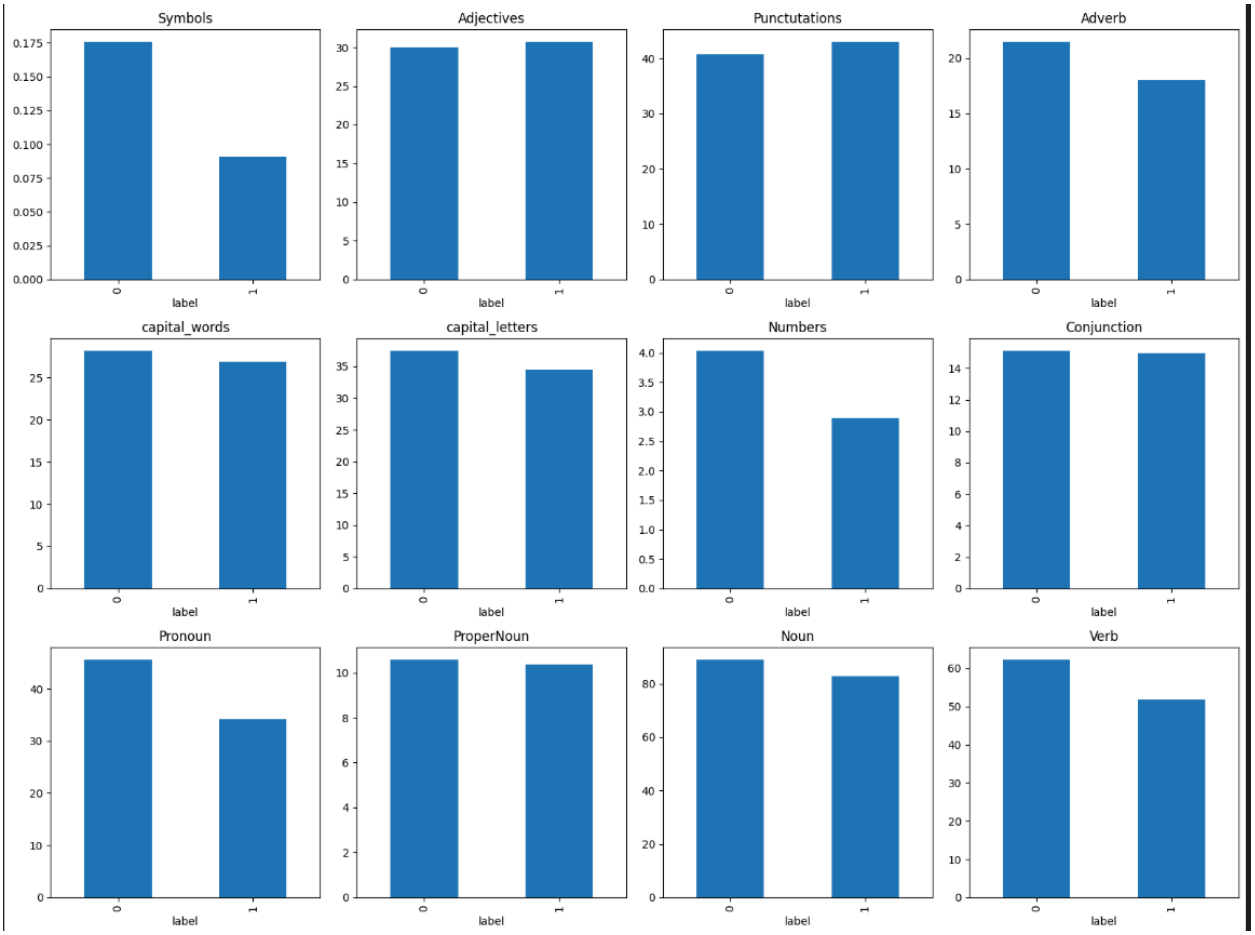
As the number of Human written and LLM Generated records and both really wide and not equal , I take a portion of this dataset consisting of 50,000 records from each of the 2 categories.

The result is obtained as follows:



POS Tagging :

I have added more features in the dataset but extracting different POS in each essays for getting more insights
This the overview of the information added in our combined dataset:



Machine Learning Model :

Model 1 : XGBoost Classifier

The XGB (XGBoost) classifier is a machine learning algorithm based on the gradient boosting framework. It stands for Extreme Gradient Boosting, which is an optimized implementation of the gradient boosting technique, designed for both performance and speed.

How XGBoost works ?

Boosting is an ensemble technique that builds a strong predictive model by combining multiple weaker models, typically decision trees.

Gradient Boosting specifically builds trees sequentially, where each new tree attempts to correct the errors (residuals) of the previous trees.

The name "gradient" comes from the use of **gradient descent** to minimize a loss function by learning from the gradients of the errors.

Training process:

The training process of XGBoost involves several steps:

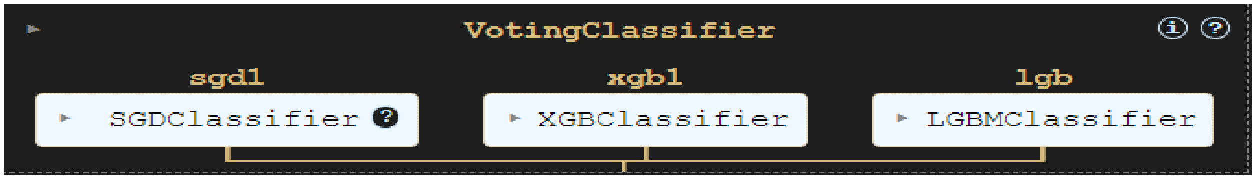
Initialization: Start with an initial prediction, which could be a simple average of the target values.

Iterative Learning: For each iteration:

- Compute the residuals (difference between the predicted and actual values).
- Fit a new decision tree (weak learner) to these residuals.
- Add the new tree's prediction to the previous predictions, weighted by a learning rate.
- Update the model to reduce the loss function by using gradient descent on the residuals.
- Repeat the process until a specified number of trees is reached or the error no longer decreases significantly.

Model 2 : VotingClassifier(Combination of SGDClassifier, XGBoost, LGBMClassifier)

The Voting Classifier is an ensemble learning method that combines multiple models to improve the overall predictive performance. It aggregates the predictions of several base classifiers and uses a majority vote (for classification) or average (for regression) to make a final prediction. This approach can often lead to better generalization on unseen data by leveraging the strengths of each individual model. Here I used Soft voting.



How does soft voting work ?

It considers the probability estimates of each classifier rather than just their hard class predictions.

It averages the predicted probabilities of each class and then selects the class with the highest average probability.

Soft voting is typically more accurate than hard voting if the individual models are well-calibrated (i.e., their probability estimates are reliable).

Optimization technique :

Bayesian Optimization for Hyperparameter tuning :

Bayesian Optimization is a strategy for optimizing objective functions that are expensive to evaluate. It is particularly useful when the function has no known analytical form and its evaluation (like training a machine learning model) is time-consuming or computationally expensive.

Using Bayesian Optimization for hyperparameter tuning is a popular approach to find the optimal hyperparameters of machine learning models, especially when training models is computationally expensive. This method helps identify the best set of hyperparameters by intelligently selecting the values to test, rather than exploring the space randomly or exhaustively.

Model Performance evaluation:

Each model was trained by splitting the dataset into 2, train and test sets and fitting the respective model on them.

Evaluation metrics:

-**Accuracy**: The percentage of correctly predicted instances.

- **Precision**: The proportion of positive predictions that were correct.

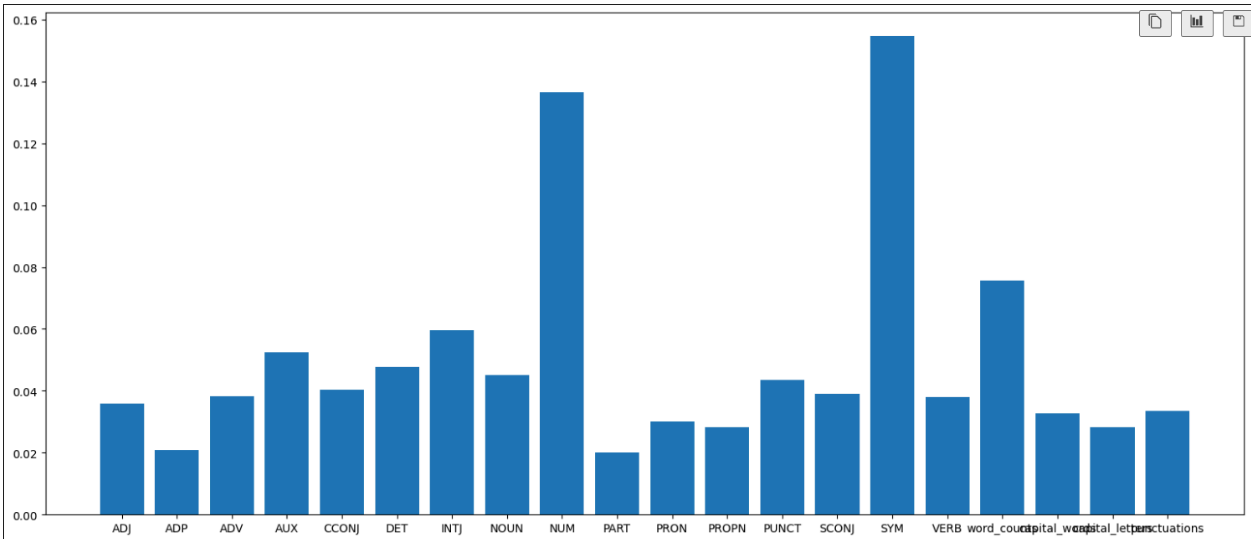
- **Recall**: The proportion of actual positives that were identified correctly.

- **F1 Score**: The harmonic mean of precision and recall.

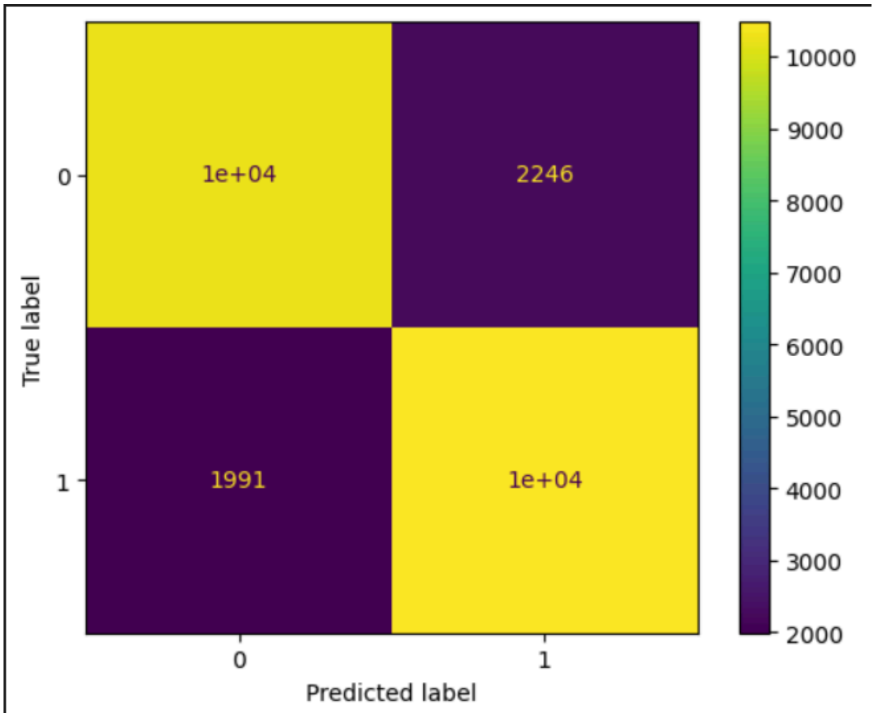
Results Obtained :

XGBoost classifier without hyper-parameter tuning :

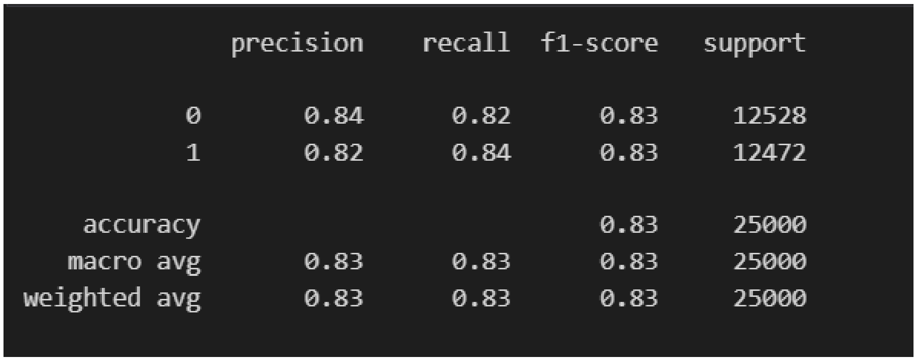
Feature Importance:



Confusion Matrix:



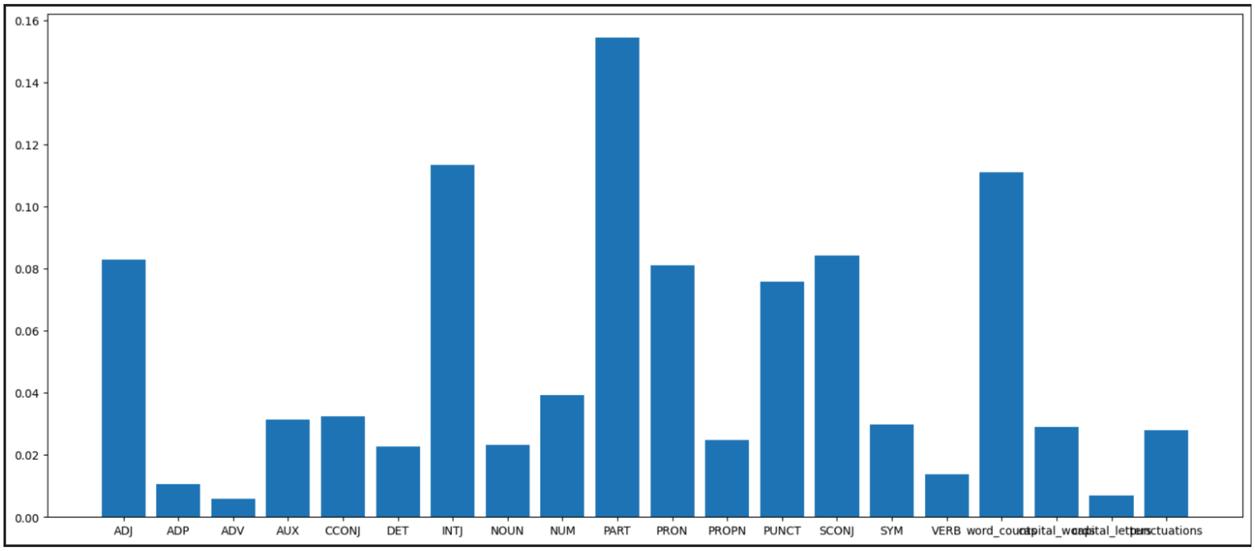
Classification Report :



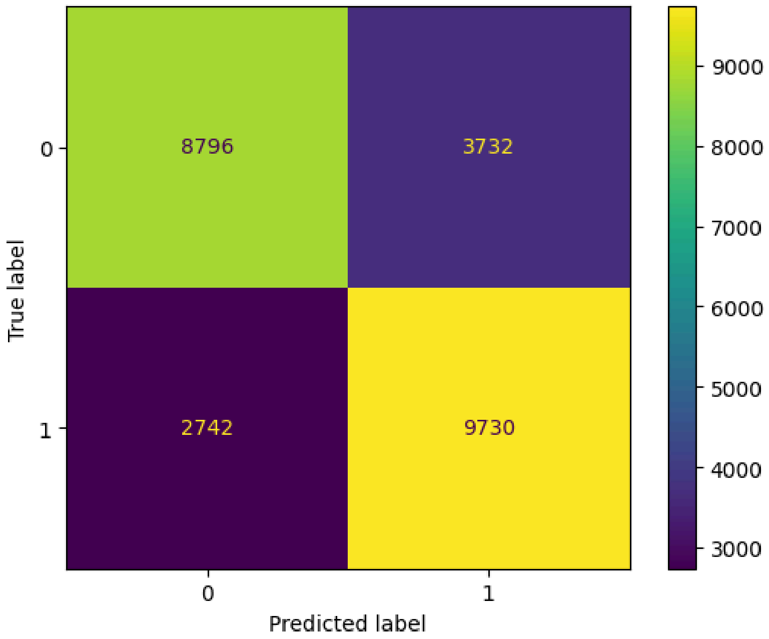
```
              precision    recall  f1-score   support

           0       0.84      0.82      0.83     12528
           1       0.82      0.84      0.83     12472

    accuracy                           0.83     25000
   macro avg       0.83      0.83      0.83     25000
weighted avg       0.83      0.83      0.83     25000
```

XGBoost classifier After hyper-parameter tuning :

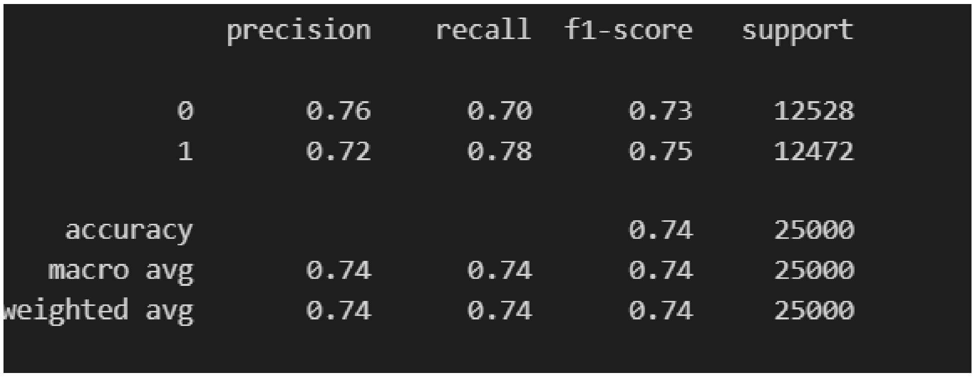
Feature Importance :



Confusion Matrix:



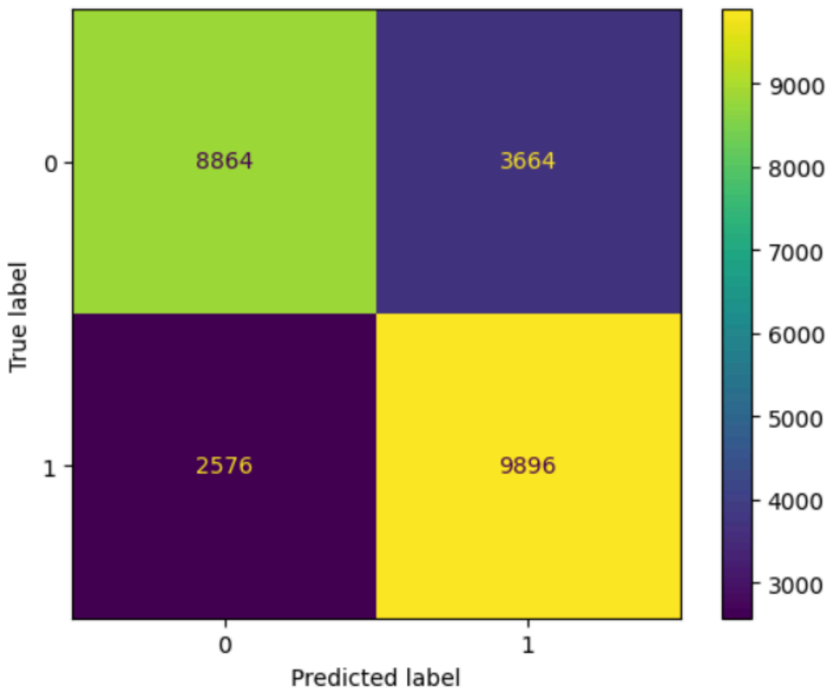
Classification Report :



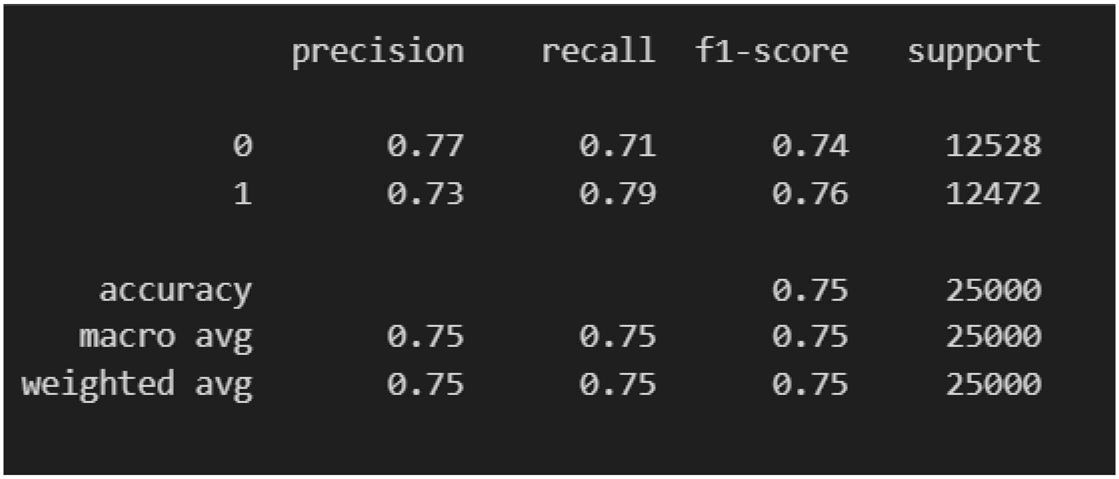
```
              precision    recall  f1-score   support

           0       0.76      0.70      0.73     12528
           1       0.72      0.78      0.75     12472

    accuracy                           0.74     25000
   macro avg       0.74      0.74      0.74     25000
weighted avg       0.74      0.74      0.74     25000
```

Voting classifier without hyper-parameter tuning :

Confusion matrix:



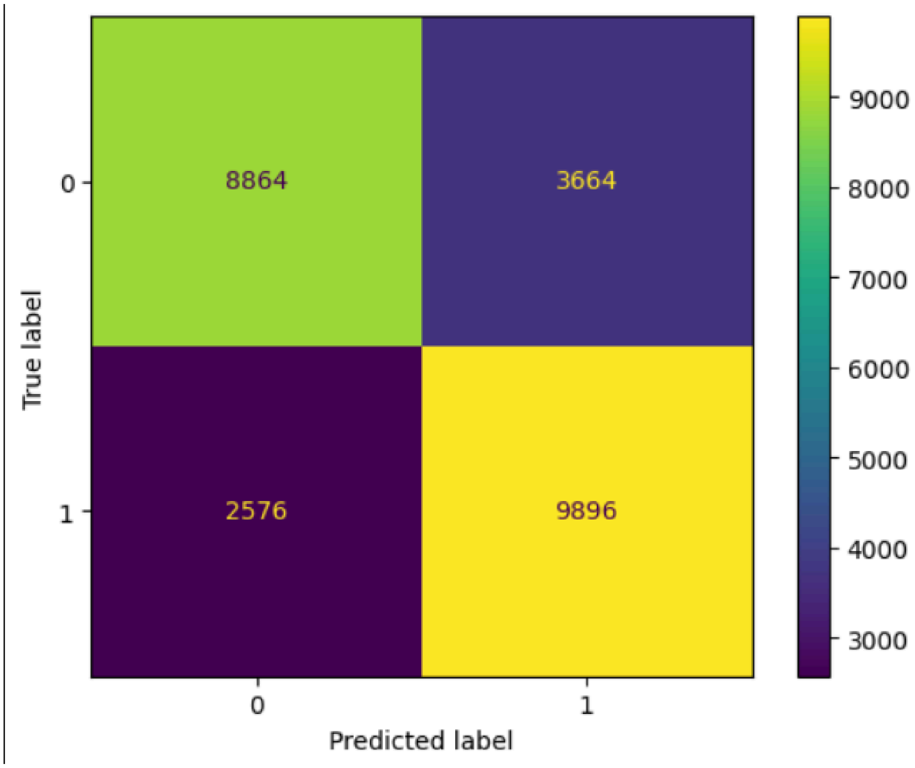
Classification Report:



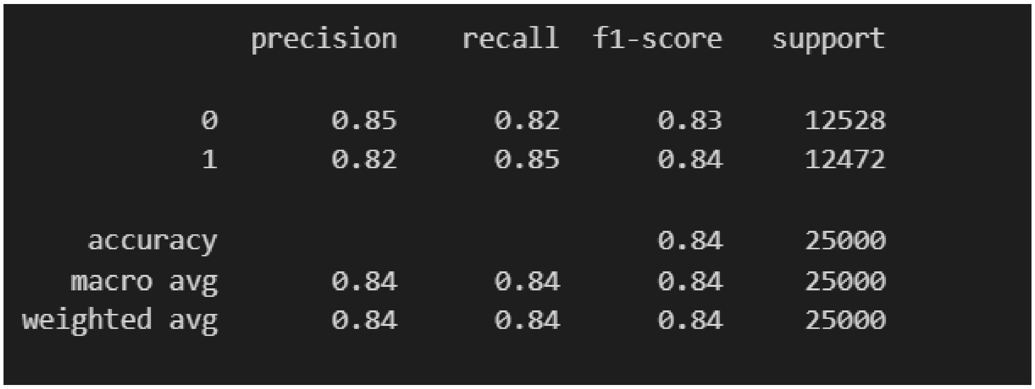
```
              precision    recall  f1-score   support

           0       0.77      0.71      0.74     12528
           1       0.73      0.79      0.76     12472

    accuracy                           0.75     25000
   macro avg       0.75      0.75      0.75     25000
weighted avg       0.75      0.75      0.75     25000
```

Voting classifier After hyper-parameter tuning :

Confusion matrix:



Classification report:



```
              precision    recall  f1-score   support

           0       0.85      0.82      0.83     12528
           1       0.82      0.85      0.84     12472

    accuracy                           0.84     25000
   macro avg       0.84      0.84      0.84     25000
weighted avg       0.84      0.84      0.84     25000
```

Scope for Improvement:

Need to fit the data using other models like ANN , to see how it improves our result and improve it further in the future .

Learning Outcomes:

Skills used:

Various skills used in this project are as follows :

- Data analysis
- Data augmentation
- Increasing features but extracting POS
- Python Data structures (hash maps for POS identification)
- Data visualization

Tools Used:

Various tools and libraries used in this project is as follows :

- SciKit learn
- Pandas
- Matplotlib
- Seaborn
- Visual Studio Code
- Kaagle for reference and data set collection

Dataset Used :

I have used a combination of 3 datasets which consists of both LLM Generated and human written essays. The first data set consists of around 27,000 LLM Generated essays and 16,000 Human written essays. The second dataset consists of around 1400 LLM Generated essays and the third dataset consists of only student essays. So a total of around 57000 essays are present in my dataset.

Further I extracted the POS in each essay and added them to increase the feature set of my dataset.

Topic Learnt :

These are the topics I have learnt during the course of my project :

- Data Augmentation
- Different Classification techniques
- POS tagging

- Ensemble learning techniques
- Voting classifier(Combination of different classifiers)
- Hyper-Parameter Tuning

Conclusion:

The system classifies essays as either human-written or LLM-generated.Using a dataset of 1,00,000 essays, various machine learning models were trained and evaluated. Out of which VotingClassifier after hyperparameter tuning gave better accuracy of around 84 .Key linguistic features like sentence structure,coherence, and lexical variety were found to be significant in distinguishing between human and LLM-generated essays . We increased our dataset size by augmenting it.

Link To code:

https://github.com/jagan1508/detect_LLM_generated_essay