Team Number: 2

Team Members: Ameya Padwad, Risa Samanta and Balasubramaniam Renganathan

Project Report

Detecting AI generated content using Machine Learning

1. Introduction

With the rise of large language models, Artificial Intelligence models are getting better and better at generating content, and it is getting harder to distinguish between human generated and AI generated content. Our goal is to identify and compare unique language patterns in synthetic text and human written text and analyze the Machine Learning models that can perform best with those patterns. This approach will help guide the creation of reliable algorithms for detection of AI generated text. We discovered through our findings that the random forest classifier had the best overall performance for this type of problem setting.

2. Related Work

- **Debora Weber-Wulff, Alla Anohina-Naumeca, "Testing of detection tools for AI-generated text":** This research investigates the performance of detection algorithms aimed at recognizing AI-generated text. It delves into methodologies and algorithms used for detection and evaluates the performance of these tools in various scenarios. However, it might lack in providing an extensive analysis of the limitations of existing methods and the areas where improvements are needed. [1]
- **GPTZero – An AI detection model:** GPTZero is the first commercially viable AI detection tool launched in January 2023. The current GPTZero model leverages a multilayered approach and an end-to-end deep learning model retrained on, and performing with high accuracy for, GPT4 datasets. One limitation as of now is that the GPTZero classifier can sometimes flag other machine-generated or highly procedural text as AI-generated, and as such, should be used on more descriptive portions of text. [2]
- **Mahdi Dhaini, Wessel Poelman, Ege Erdogan, "Detecting ChatGPT":** This survey provides an overview of the current approaches employed to differentiate between texts generated by humans and ChatGPT. It provides general insights on the different datasets constructed for detecting ChatGPT-generated text, the various methods utilized, and what qualitative analyses into the characteristics of human versus ChatGPT-generated text have been performed. However, the scope of this study is limited to academic papers and excludes online nonacademic tools. [3]
- **LLM - Detecting AI-Generated Text (Project on Kaggle):** This Kaggle project employs the AutoTokenizer transformer to generate features from three distinct datasets, followed

by classification using the DeBERTa model. However, a key issue with this project is its reliance on a pretrained model. [4]

3. Problem Formulation

We are using a subset of the M4 dataset [5], which includes abstracts on selected topics. This subset contains 12,000 human-written responses and 30,000 responses from various AI models, including Bloomz, ChatGPT, Cohere, Davinci, Dolly, and Flan-t5.

The problem formulation has two parts:

1. Analyzing two features:
    a. Token Frequency (TF) monograms *(Max 10,000 features)*
    b. Token Frequency-Inverse Document Frequency (TF-IDF) monograms *(Max 10,000 features)*
2. Analyzing three models:
    a. Logistic Regression
    b. Random Forest
    c. Neural Network

Our objective is to identify the best feature-model combination that can solve the given problem.

4. Methodology

The dataset essentially contains two columns:

1. Abstracts – Contains the human-written or AI generated text
2. Labels – Contains '0's denoting human-written and '1's denoting AI generated

Data preprocessing included the following steps:

1. Data cleaning *(using pandas, nltk and strings)*:
    a. Dropping rows containing Null values and removing whitespace before and after the abstracts
    b. Removing words containing numbers and words having length less than 3 characters
    c. Removing stop words *("a", "the", etc.)* and punctuations *(".", ",", etc.)*
2. Tokenizing, Stemming and Lemmatizing the abstracts *(using nltk)*
3. Generating TF and TFIDF matrices *(using sklearn text feature extractor)*
4. Normalizing the features *(using standardized normalization)*
5. Splitting the dataset into *train:test:valid = 0.8:0.1:0.1*

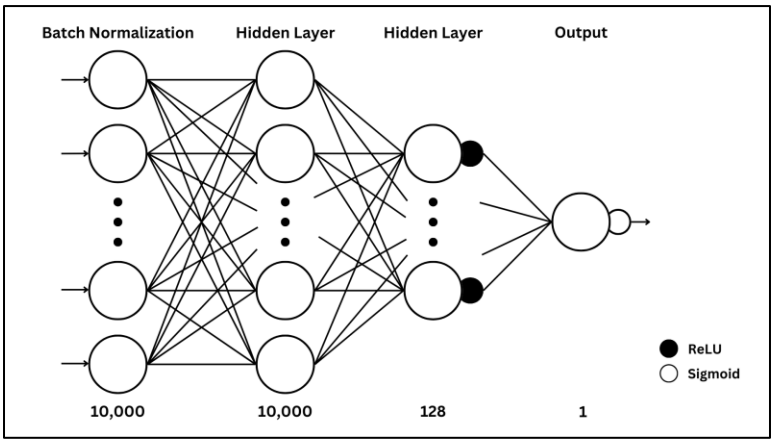
We experimented with multiple hyperparameters and validated them against the validation dataset to select the best model out of Logistic Regression, Random Forest, and Neural Network for each of the two features.

5. Experiments

5a. Experimental Setup

The three models are configured with the following parameters, and have the corresponding hyperparameter(s) to be tuned:

1. Logistic Regression:
    a. Parameters:
        i. Penalty = L2
        ii. Solver = lbfgs
    b. Hyperparameters:
        i. C (Regularization Coefficient)
2. Random Forest:
    a. Parameters:
        i. Criterion = gini impurity
        ii. Max features = sqrt(n_features)
    b. Hyperparameters:
        i. n (Number of estimators)
3. Neural Network:
    a. Architecture:



    b. Parameters:
        i. Criterion = Binary Cross Entropy Loss
        ii. Optimizer = Adam
    c. Hyperparameters:
        i. Learning Rate
        ii. Number of Epochs

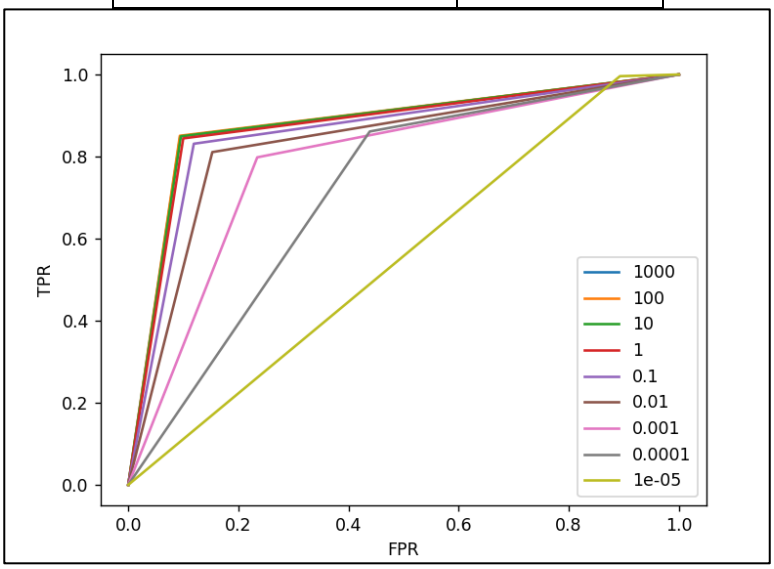
We used a combination of ROC AUC, accuracy score and F1 score to evaluate the performance of the models.

5b. Model Evaluation

5b(1) Logistic Regression:

1. TF Evaluations:

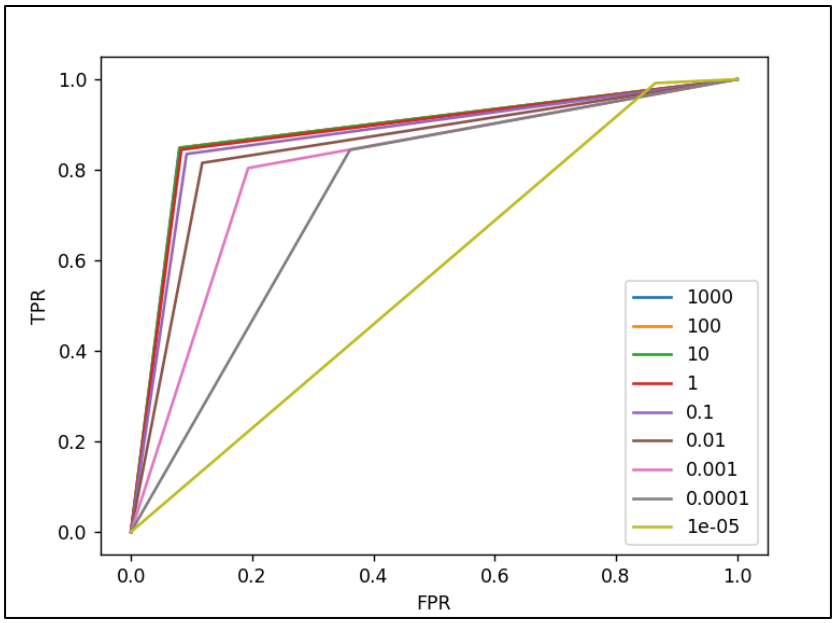
C	ROC AUC
1000	0.8774
100	0.8785
10	**0.8769**
1	0.8722
0.1	0.8561
0.01	0.8291
0.001	0.7819
0.0001	0.7113
0.00001	0.5512



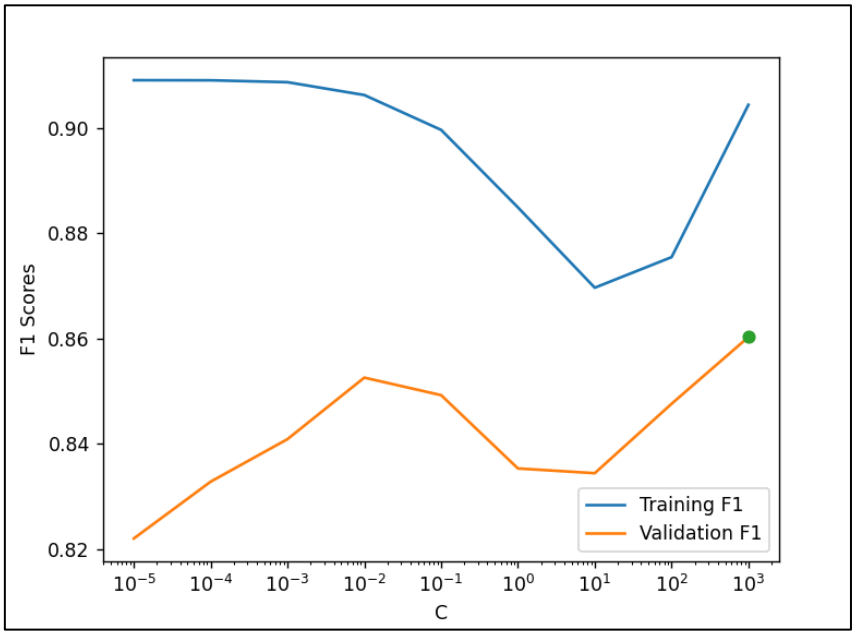
Graph 5(1). *Logistic Regression ROCs for different values of C with TF features*

2. TFIDF Evaluations:

C	ROC AUC
1000	**0.8844**
100	0.8840
10	0.8837
1	0.8807
0.1	0.8716
0.01	0.8488
0.001	0.8053
0.0001	0.7413
0.00001	0.5633



Graph 5(2). *Logistic Regression ROCs for different values of C with TFIDF features*



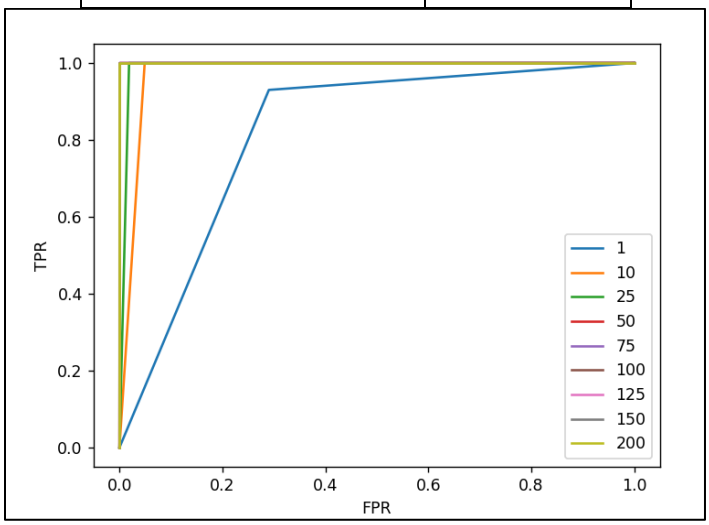
Graph 5(3). *F1 scores vs C for Logistic Regression when trained on TFIDF Feature*

The optimal value of C was **10 for TF** and **100 for TFIDF** with ROC AUCs of 0.8769 and 0.8844, respectively. The value that maximizes the validation score was selected as optimal.

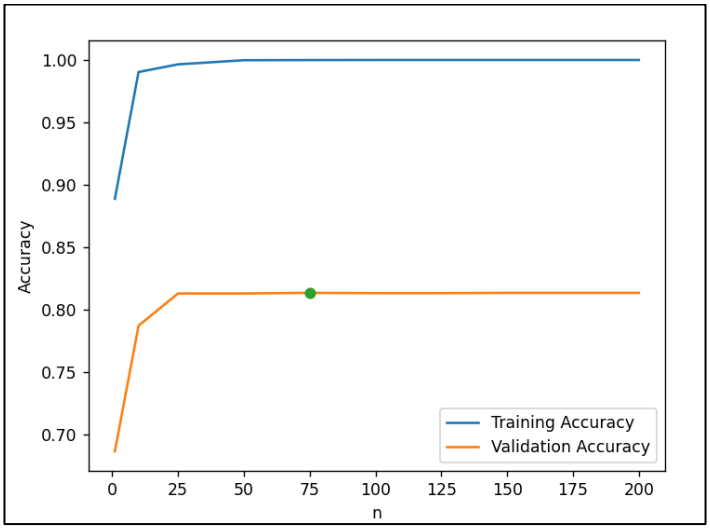
5b(2) Random Forest:

1. TF Evaluations:

n	ROC AUC
1	0.8199
10	0.9752
25	0.9906
50	0.9995
75	**0.9998**
100	1.0
125	1.0
150	1.0
200	1.0



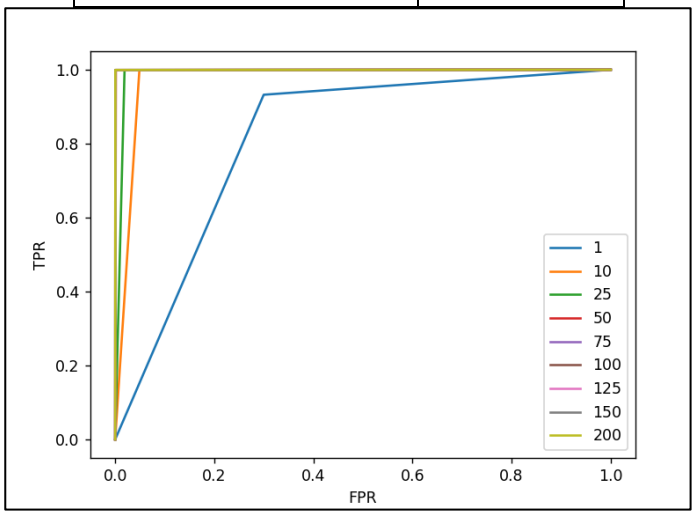
Graph 5(4). *ROC for Random Forest for different values of n when trained on TF Features*



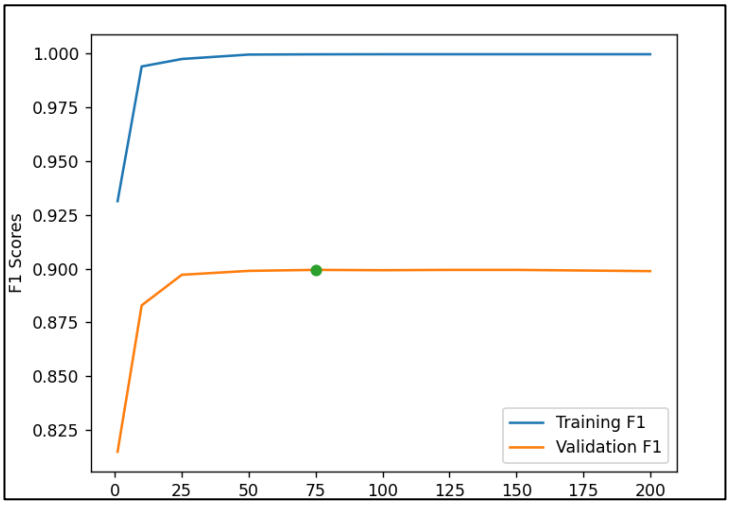
Graph 5(5). *F1 scores vs n for Random Forest when trained on TF Features*

2. TFIDF Evaluations:

n	ROC AUC
1	0.8163
10	0.9752
25	0.9902
50	0.9989
75	**0.9994**
100	0.9995
125	0.9995
150	0.9995
200	0.9995



Graph 5(6). *ROC for Random Forest for different values of n when trained on TFIDF Features*



Graph 5(7). *F1 scores vs n for Random Forest when trained on TFIDF Features*

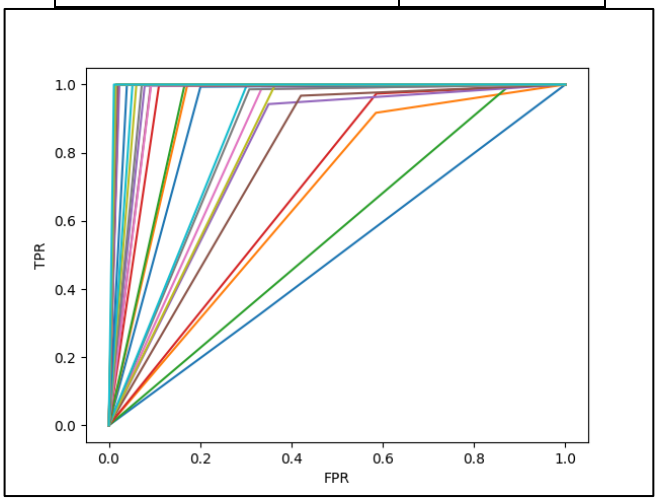
The optimal value of n was **75 for both TF and TFIDF** features with ROC AUCs of 0.9998 and 0.9994, respectively. The value that maximizes the validation score was selected as optimal.

5b(3) Neural Network:

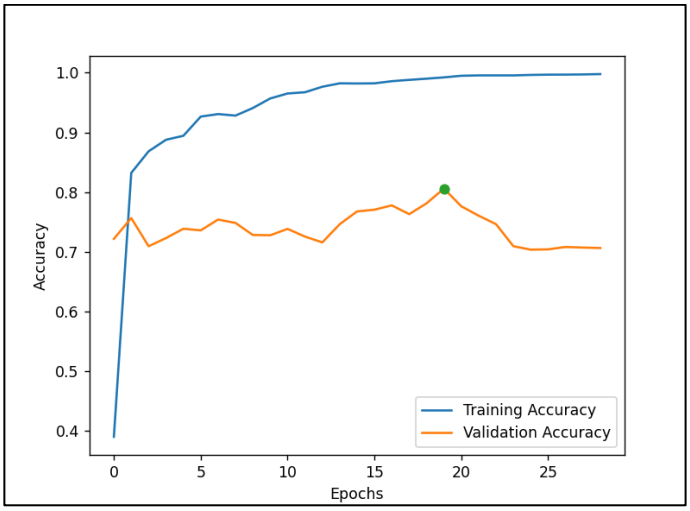
To determine the learning rate, the neural network was trained for 30 epochs on the TF and TFIDF features separately.

1. TF Evaluations:

Learning Rate	ROC AUC
1	0.5
0.1	0.5
0.01	**0.9943**
0.001	0.8943
0.0001	0.6704
0.00001	0.5712
0.000001	0.5051



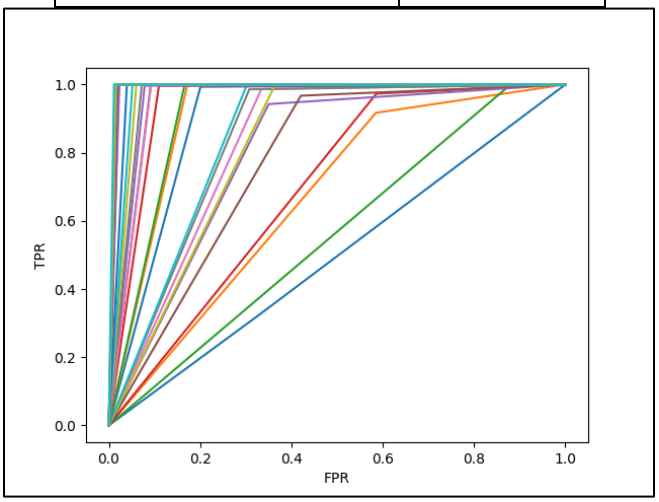
Graph 5(8). *ROC for NN with LR = 0.01 and Epochs = 30 when trained on TF Features*



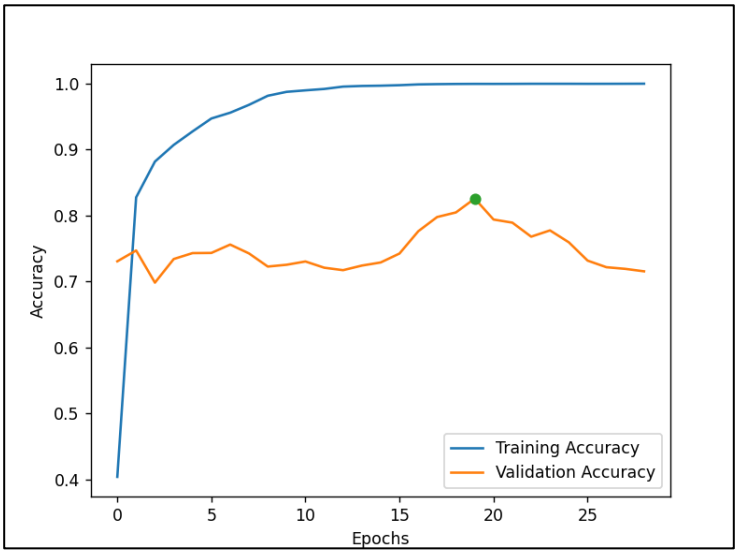
Graph 5(9). *Accuracy vs Epochs for NN with LR = 0.01 when trained on TF Features*

2. TFIDF Evaluations:

Learning Rate	ROC AUC
1	0.5
0.1	0.5
<u>0.01</u>	**<u>0.9988</u>**
0.001	0.9423
0.0001	0.7
0.00001	0.5775
0.000001	0.5045



Graph 5(10). *ROC for NN with LR = 0.01 and Epochs = 30 when trained on TFIDF Features*

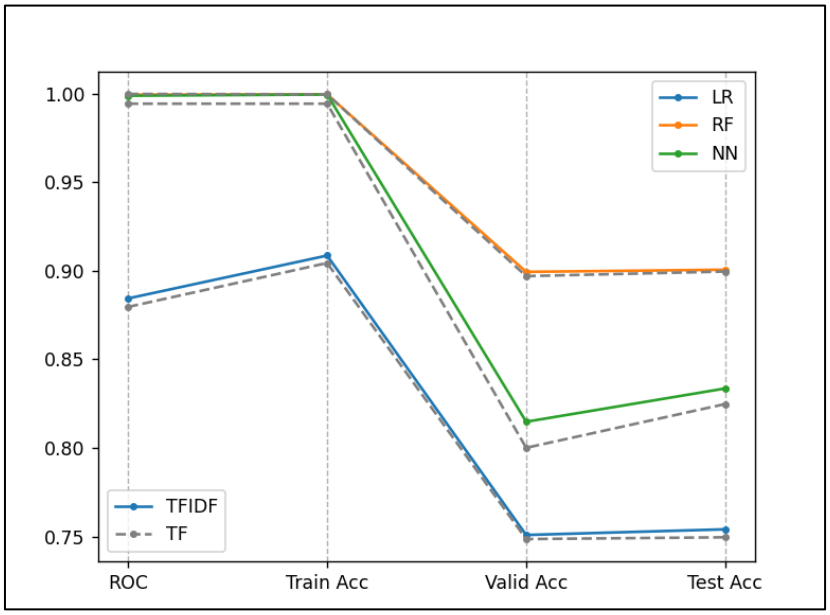


Graph 5(11). *Accuracy vs Epochs for NN with LR = 0.01 when trained on TFIDF Features*

The optimal learning rate was **0.01 for both TF and TFIDF** features with ROC AUCs of 0.9943 and 0.9988, respectively. The value that maximizes the validation score was selected as optimal.

The optimal number of epochs was **20 and 19 for TF and TFIDF respectively**.

5c. Results



Graph 5(12). *Metrics of all the models for TF and TFIDF features*

1. Logistic Regression:
    a. TF:
        i. ROC: 0.8796
        ii. Training F1 Score: 0.9044
        iii. Validation F1 Score: 0.7486
        iv. Test F1 Score: 0.7496
        v. Confusion Matrix: [2282 1007]

        [216 517]

    b. TFIDF:
        i. ROC: 0.8844
        ii. Training F1 Score: 0.9086
        iii. Validation F1 Score: 0.7509
        iv. Test F1 Score: 0.7541
        v. Confusion Matrix: [2309 989]

        [207 517]

2. Random Forest:
    a. TF:
        i. ROC: 0.9998
        ii. Training F1 Score: 0.9996
        iii. Validation F1 Score: 0.8970
        iv. Test F1 Score: 0.8996
        v. Confusion Matrix: [2991 401]

        [103 630]

    b. TFIDF:
        i. ROC: 0.9994
        ii. Training F1 Score: 0.9996
        iii. Validation F1 Score: 0.8994
        iv. Test F1 Score: 0.9006
        v. Confusion Matrix: [3002 390]

        [97 624]

3. Neural Network:
    a. TF:
        i. ROC: 0.9943
        ii. Training Accuracy: 0.9977
        iii. Average Validation Accuracy: 0.8001
        iv. Average Test Accuracy: 0.8248
        v. Confusion Matrix: [2787 605]

        [128 502]

    b. TFIDF:
        i. ROC: 0.9988
        ii. Training Accuracy: 0.9995
        iii. Average Validation Accuracy: 0.8148
        iv. Average Test Accuracy: 0.8336
        v. Confusion Matrix: [2798 631]

        [93 491]

6. Conclusion

It can be concluded from the results that the Logistic Regression and Neural Network models performed slightly better when trained on TFIDF features as compared to TF features, whereas the Random Forest classifier had similar performance for both the features. It can also be seen from *Graph 5(12)* that Random Forest classifier had the best overall performance of the three models.

Future Scope:

1. Ensemble methods with the aforementioned models like bagging and boosting have the potential to significantly enhance accuracy.
2. Due to computational constraints, we did not utilize the entire M4 dataset. However, using the full dataset should yield superior results as more data helps in reducing bias and increasing variance without overfitting.
3. When generating the TF and TFIDF matrices, we constrained to a maximum of 10,000 features and focused solely on monograms. Employing TF and TFIDF without feature limitations and exploring bigrams and trigrams could reveal more prominent patterns.
4. The feature scope can also be broadened by incorporating features such as Bag of Words, Part-of-Speech (POS) tagging, Sentiment analysis, and Named Entity Recognition.

7. Task Allocation

1. Ameya Padwad: Feature Extraction and Neural Network
2. Balasubramaniam Renganathan: Random Forest
3. Risa Samanta: Logistic Regression

8. Project Files

All the project files can be found on this github repository: link.

9. References

[1] Debora Weber-Wulff, Alla Anohina-Naumeca, “Testing of detection tools for AI-generated text”, International Journal for Educational Integrity – link to paper

[2] GPTZero – An AI detection model - link

[3] Mahdi Dhaini, Wessel Poelman, Ege Erdogan, “Detecting ChatGPT: A Survey of the State of Detecting ChatGPT-Generated Text” - link to paper

[4] LLM - Detecting AI Generated Text – project on kaggle

[5] M4 Dataset – link to dataset