A Project on

Multigenerator and Multidomain Machine-Generated Text Detection

Submitted in partial fulfilment of the requirement for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE & ENGINEERING (ML & AI)

Submitted by:

Abhishek Singh	**2017447**
Siddhant Rawat	**2017558**
Aniket Raj	**2017463**

Under the Guidance of

Dr. Anita Saroj / Mr. Rohan Verma

(Assistant Professor)

Project Team ID: MP23CST010

Department of Computer Science and Engineering
Graphic Era (Deemed to be University)
Dehradun, Uttarakhand

June-2024

CANDIDATE'S DECLARATION

I/We hereby certify that the work which is being presented in the project progress report entitled **"Multigenerator and Multidomain Machine-Generated Text Detection"** in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science and Engineering **(ML & AI)** in the Department of Computer Science and Engineering of the Graphic Era (Deemed to be University), Dehradun is an authentic record of my work carried out during a period from **August-2023 to May-2024** under the supervision of **Dr. Anita Saroj and Mr. Rohan Verma, Assistant Professor**, Department of Computer Science and Engineering, Graphic Era (Deemed to be University), Dehradun.

The matter presented in this dissertation has not been submitted by me/us for the award of any other degree of this or any other Institute/University.

Abhishek Singh	2017447
Siddhant Rawat	2017558
Aniket Raj	2017463

This is to certify that the above statement made by the candidate is correct to the best of our knowledge.

Signature

Supervisor

Signature

Head of the Department

External Viva

Name of the Examiners: **Signature with Date**

1.

2.

Abstract

The widespread use of AI language models presents several issues, with one of the most pressing being the potential compromise of intellectual and academic integrity. As AI-generated content becomes more prevalent, it is increasingly important to distinguish between material written by AI and that authored by humans. The legitimacy and authority of intellectual works could be unintentionally undermined by the spread of AI-generated information. Therefore, it is crucial to develop techniques to differentiate between AI-produced content and human-authored works.

This research evaluates the efficacy and accuracy of various machine learning techniques in distinguishing text generated by ChatGPT from human-written material. The machine learning algorithms are tasked with classification, and a comprehensive performance comparison is conducted. This study addresses the immediate challenge of differentiating ChatGPT-generated text from human-written content through the development and evaluation of a machine learning system. Additionally, it enhances our understanding of potential future scenarios and mitigation strategies for AI-generated content. In essence, in an era of rapidly evolving natural language generation models, this research has significant implications for maintaining the ethical use of AI technology and the reliability of digital communication.

The ease of using large language models (LLMs) is making machine-generated content more common across various contexts, including education, social media, news, forums, and academic settings. LLMs like ChatGPT and GPT-4, which can respond to a wide range of user queries, appear highly realistic. Given that this content is well-written, LLMs are a compelling alternative to human labor in many situations. However, this raises concerns about their potential misuse to spread misinformation and hinder the teaching and learning process. To mitigate the risk of such content being misused, automatic mechanisms for identifying machine-generated text are essential, as human ability to do so is only slightly better than chance.

Acknowledgement

Any achievement, be it scholastic or otherwise does not depend solely on individual effort but on the guidance, encouragement, and cooperation of intellectuals, elders, and friends. Several personalities in their capacity have helped me in carrying out this project work.

Our sincere thanks to project guide **Dr. Anita Saroj and Mr. Rohan Verma, Assistant Professor**, Department of Computer Science and Engineering, Graphic Era (Deemed to be University), for his valuable guidance and support throughout the course of project work and for being a constant source of inspiration.

We extend our thanks to **Prof. (Dr.) Guru Prasad M.S.**, Project Coordinator, Department of Computer Science and Engineering, Graphic Era (Deemed to be University), for his valuable suggestions throughout all the phases of the Project Work.

We are extremely grateful to **Prof. (Dr.) D. P. Singh**, HOD of the Computer Science and Engineering Department, Graphic Era (Deemed to be University), for his moral support and encouragement.

We thank the **management of Graphic Era (Deemed to be University)** for the support throughout the course of our Bachelor's Degree and for all the facilities they have provided.

Last, but certainly not least we thank all teaching and non-teaching staff of Graphic Era (Deemed to be University) for guiding us on the right path. Most importantly we wish to thank our parents for their support and encouragement.

Abhishek Singh	2017447
Siddhant Rawat	2017558
Aniket Raj	2017463

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

In the following sections, a brief introduction and the problem statement for the work has been included.

1.1 Project Introduction

The widespread application of AI language models raises several concerns, chief among them the potential erosion of academic and intellectual integrity. As AI-generated content becomes increasingly comparable to human-authored works, the ability to distinguish between text created by AI and human writers becomes increasingly critical. Developing methods to distinguish material written by AI from that written by human authors is essential because the proliferation of information created by AI could inadvertently undermine the authority and genuineness of intellectual works. Machine learning has been discussed as a potential method for identifying text generators. In this study, we evaluate the effectiveness and accuracy of 11 machine learning methods for detecting ChatGPT-generated text from human generated text. The classification task is provided to the machine learning algorithms, and a comprehensive performance comparison is performed. Through the development and assessment of our machine learning system, this research addresses the immediate challenge of distinguishing between text generated by ChatGPT and human-written content, while also expanding our understanding of how AI-produced material might be found and mitigated in the future. In the conclusion, our research has significant implications for maintaining the moral application of AI technologies and the dependability of digital communication in the era of rapidly advancing natural language generation models [10].

Massive language models (LLMs) are becoming widely used, and this convenience is driving the growth of machine-generated content in a variety of domains, such as education, social media, news, forums, and even academic settings. More recently, LLMs that generate responses to a wide range of user inquiries, such as ChatGPT and GPT-4, sound incredibly natural. The well-written nature of this produced text makes LLMs a compelling substitute for human labor in many circumstances. However, this has sparked concerns about how they might be misused to spread misleading information and obstruct the teaching and learning process. Since humans only marginally outperform chance in this

regard, developing automatic systems to identify machine-generated text is necessary to reduce the potential misuse of this type of content [10].

ChatGPT - ChatGPT, or Chat Generative Pre-training Transformer, is an advanced text generation tool developed by OpenAI, an AI research organization. ChatGPT is a type of language model that's able to generate human-like responses to a given input, making it ideal for use in chatbots, language learning apps, and other conversational AI applications [11].

Dolly-v2 - Databricks' dolly-v2-12b, an instruction-following large language model trained on the Databricks machine learning platform that is licensed for commercial use. Based on pythia-12b, Dolly is trained on ~15k instruction/response fine tuning records databricks-dolly-15k generated by Databricks employees in capability domains from the InstructGPT paper, including brainstorming, classification, closed QA, generation, information extraction, open QA and summarization [12].

Cohere - They provide a platform and tools for businesses to build applications and services that leverage the power of NLP and text generation. Cohere's NLP models are designed to understand and generate human-like text, making it easier for developers to create chatbots, virtual assistants, content generation tools, and more. Cohere's text generation capabilities are based on advanced machine learning techniques, and they offer developers APIs (Application Programming Interfaces) to integrate these capabilities into their applications. This allows businesses to create interactive and engaging experiences for their users through natural language understanding and generation [13].

DaVinci – It is the commercial version of OpenAI's GPT-3 language model. GPT-3, which stands for "Generative Pre-trained Transformer 3," is a state-of-the-art language model designed to generate human-like text based on the input it receives. DaVinci is one of the several models developed by OpenAI using the GPT-3 architecture. The DaVinci model is known for its text generation capabilities, which include tasks such as natural language understanding, text completion, language translation, content generation, and much more. It can generate coherent and contextually relevant text in a wide range of languages and domains [14].

1.2 Problem Statement

The problem for the project is to classify a generated text into human generated or machine generated. The objective of this report is to further classify this machine generated outcome into their respective text generation models. For example, if a text is found to be machine generated then we should also find out if it was generated by ChatGPT, Dolly, Cohere, Bloom LLM or DaVinci etc.

1.3 Objectives

1 The proposed work objectives are as follows:

2 **Subtask A.** Create a classification model to distinguish between human-generated and AI-generated texts.

3 **Subtask B.** Evaluate the model's accuracy and visualize its performance.

4 **Subtask C.** Upgrade the initial model to differentiate human-generated texts from various types of AI-generated texts.

5 **Subtask D.** Assess the updated model's accuracy and visualize its effectiveness across different AI-generated content.

6 **Subtask E.** Compare the model's performance against different algorithms for classifying human vs AI-generated texts.

7 **Subtask F.** Evaluate accuracy and visualize results to identify the most effective algorithm.

Chapter 2

Literature Survey/ Background

Recent advances in LLM can be traced back to those reported by Vaswani et al. Proposed transformer architecture. [3] introduces a self-attention mechanism that allows the model to focus on different regions of the input sequence. Later, Radford et al. [4] develop a generative pre-trained transformer (GPT) based on the transformer architecture. Because GPT is trained on a large corpus of text data, it achieves good performance on a variety of language generation tasks. GPT2 [5], a larger version of GPT that includes more parameters and is trained on a larger corpus, was developed and achieves better performance than GPT. GPT3 [6] is the third generation of GPT with over 175 billion parameters and is proven to produce consistent and context-appropriate text even in situations where minimal input or guidance is required. I am. Since November 2022, OpenAI has released his ChatGPT [7]. It is trained on his GPT 3.5 architecture and uses reinforcement learning from human feedback (RLHF) [8, 9] to improve its generative capabilities. Innovative text generation skills are displayed by ChatGPT, which can provide consistent and pertinent content for a range of applications, including chatbots, customer support, and education.
Xin lei He and colleagues' work provides a thorough examination of the approaches currently in use for identifying machine-generated text (MGT) when strong Language Models (LLMs) are at play. They examine six metric-based and four model-based detection techniques, testing their efficacy on a variety of datasets. Among all the approaches, the LM Detector is the most successful and resilient in identifying MGTs with fewer words or those produced by distinct LLMs.

The researchers next investigate how to apply MGT identification techniques to the trickier job of text attribution. In this case, model-based techniques—particularly the LM Detector—outperform metric-based techniques because they are more skilled at capturing the syntactic and semantic connections between words and sentences. Although metric-based approaches have difficulty differentiating source LLMs, all approaches show potential for improvement in precisely identifying the source of MGTs.[26]
Through the addition of adversarial perturbations to MGTs, the study explores the resilience of MGT detection techniques even more. The ChatGPT Detector exhibits notable

susceptibility to minor disturbances, underscoring the urgent necessity for the advancement of more robust MGT detection techniques.[25]

The researchers present MGTBench, a modular framework that integrates datasets and detection techniques, as a forward-thinking endeavour. This novel technology is well-positioned to act as a standard, enabling further studies to improve MGT detection techniques and enhance LLM training protocols. [14]The study emphasises how important it is to keep working to improve the reliability and effectiveness of MGT detection techniques.

Chapter 3

Project Work Carried Out

3.1 Steps Performed

A) IMPORT LIBRARIES

The libraries used during this project are –

1. **NumPy**

   NumPy (Numerical Python) is an open source Python library that's used in almost every field of science and engineering. It's the universal standard for working with numerical data in Python, and it's at the core of the scientific Python and PyData ecosystems. NumPy users include everyone from beginning coders to experienced researchers doing state-of-the-art scientific and industrial research and development. The NumPy API is used extensively in Pandas, SciPy, Matplotlib, scikit-learn, scikit-image and most other data science and scientific Python packages. The NumPy library contains multidimensional array and matrix data structures (you'll find more information about this in later sections). It provides nd array, a homogeneous n-dimensional array object, with methods to efficiently operate on it. NumPy can be used to perform a wide variety of mathematical operations on arrays. It adds powerful data structures to Python that guarantee efficient calculations with arrays and matrices, and it supplies an enormous library of high-level mathematical functions that operate on these arrays and matrices [1].

2. **Pandas**

   Pandas is an open-source Python package that is most widely used for data science/data analysis and machine learning tasks. It is built on top of another package named NumPy, which provides support for multi-dimensional arrays. During this project pandas will be used to pre-process and engineer the dataset to cater to our needs to provide us with the best possible results [15].

3. Sklearn

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modelling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib [16].

B) DATA PREPROCESSING

The dataset file is read from memory. We iterate through every row of the DataFrame and extract the 'paraphrase' column from the current row. We also remove the square brackets at the beginning and end, and then split the string into a list. The first paraphrase then gets its square brackets removed and gets added to the dictionary with the category 'AI'. The original text from the 'text' column is then added to dictionary with category 'human'.

A new DataFrame is created with columns namely 'text' and 'category'. Since the size of our dataset is very vast (808028 entries) we first shuffle the rows of the DataFrame and then select only a sample of rows from it.

C) SPLIT INTO TRAIN AND TEST

With the help of 'train_test_split' function the data is split into training and testing sets. 80% of the data will be used for training while the rest of the 20% for testing.

D) TEXT DATA FREQUENCY COUNT

TF-IDF Vectorizer is a statistical method that measures the importance of a word in a document. It's based on the idea that words that appear more often in a document are more relevant to the document. In other words, it is a numerical representation of text data that considers the importance of each term in a document relative to its occurrence across all documents. The vectorizer is fitted on the training data and transforms the input data.

E) CLASSIFIERS AND THEIR SCORE

Then the process of deciding classifier for the model is decided. The model is tested through various classifiers. The classifiers used in this project are Logistic Regression, Support Vector Machine, Multinomial Naïve Bayes, KNeighborsClassifier, Random

Forest Classifier, Extra Trees Classifier, AdaBoost, Bagging Classifier, Gradient Boosting Classifier. Then the scores such as accuracy, precision, recall, f1 score, Matthew's correlation coefficient are calculated for each classification algorithm.

3.2 Algorithms Used:

3.2.1. Logistic Regression:

Two typical uses for this kind of statistical model are in classification and predictive analytics. Using a dataset of independent variables, logistic regression calculates the probability of an event like voting or not occurring. With a probability as the outcome, the dependent variable's range is 0 to 1. The odds in logistic regression are calculated by dividing the probability of success by the probability of failure. This is known as a logit transformation. The log odds or natural logarithm of odds are other names for this logistic function. The following is the formula: [2]

3.2.2. Support Vector Machine (SVM):

Support Vector Machine (SVM), a supervised machine learning algorithm, is used for both regression and classification. Regression problems are best suited for use in classification problems, nevertheless. Finding the optimal hyperplane in an N-dimensional space to divide data points into different feature space classes is the main objective of the SVM algorithm. The hyperplane ensures that the closest points of different classes are as far apart as feasible. The dimension of the hyperplane is determined by the number of features. If there are just two input features, the hyperplane is just a line. The hyperplane is just a line if there are only two input features. The hyperplane transforms into a 2-D plane if there are three input features. If there are more than three features, it gets hard to imagine [3].

3.2.3. K-Nearest Neighbors (KNN):

One of the most fundamental yet crucial machine learning classification algorithms is K-Nearest Neighbors. It is heavily used in pattern recognition, data mining, and intrusion detection and is a member of the supervised learning domain. Since it is non-parametric—that is, it does not make any underlying assumptions about the distribution of data—it is extensively applicable in real-life scenarios (in contrast to other algorithms like GMM, which assume a Gaussian distribution of the given data). An attribute-based prior data set (also known as training data) is provided to us, allowing us to classify coordinates into groups based on the similarity principle, the K-Nearest Neighbors (KNN) algorithm predicts the label or value of a new data point by taking into account the labels or values of its K nearest neighbors in the

training dataset.The algorithm determines the distance between every new data point in the test dataset and every data point in the training dataset in order to generate predictions [4].

3.2.4. Random Forest:

One popular machine learning algorithm is Random Forest, which is used in supervised learning techniques. It can be used to solve regression and classification-based machine learning problems. Its basis is the concept of ensemble learning, which is the process of combining multiple classifiers to improve the functionality of the model and solve a difficult problem. As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." The random forest predicts the result based on the majority vote of predictions from each decision tree, as opposed to relying solely on one. The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting [5].

3.2.5. Extra Trees Classifier:

An example of an ensemble learning technique is the Extremely Randomized Trees Classifier (Extra Trees Classifier), which generates a classification result by combining the output of several de-correlated decision trees gathered in a "forest." It is conceptually very similar to a Random Forest Classifier and only differs in how the forest's decision trees are built. The original training sample serves as the foundation for every Decision Tree in the Extra Trees Forest. After that, each decision tree at each test node receives a random sample of k features from the feature-set, from which it must choose the best feature to divide the data according to a set of mathematical standards (usually the Gini Index). Several de-correlated decision trees are produced as a result of this random feature selection process. During the construction of the forest, for each feature, the normalized total reduction in the mathematical criteria used in the feature of split decision (Gini Index if the Gini Index is used in the construction of the forest) is computed in order to perform feature selection using the above forest structure. The Gini Importance of the feature is the name given to this value. The process of feature selection involves sorting each feature based on its Gini Importance in descending order, and the user chooses the top k features based on personal preference [6].

where N is the total number of samples, N_t is the number of samples at the current node, N_t_L is the number of samples in the left child, and N_t_R is the number of samples in the right child [6].

3.2.6. AdaBoost:

Adaptive Boosting, or AdaBoost, is a machine learning algorithm for ensembles that can be applied to a range of classification and regression tasks. This supervised learning algorithm creates a strong learner by merging several weak or base learners (like decision trees) to classify data. In order for AdaBoost to function, it weighs each instance in the training dataset according to how well it was previously classified. This method builds a model and gives each data point the same weight. Then, it gives points that were wrongly classified larger weights. Every point in the model that follows is assigned a higher weight. Models will be trained continuously until a smaller error is received back [7].

3.2.7. Bagging Classifier:

A sort of ensemble learning known as "bagging" (also called "Bootstrap aggregating") involves training several base models concurrently and independently on various subsets of the training set. Bootstrap sampling, which selects data points at random with replacement, is used to create each subset. For the Bagging classifier, majority voting is used to aggregate the all-base model's predictions to arrive at the final prediction. Regression bagging is the process of determining the final prediction by averaging the predictions of the all-base model [8].

3.2.8. Gradient Boosting:

With gradient descent, each new model is trained to minimize the loss function, such as mean squared error or cross-entropy of the previous model. Gradient boosting is a potent boosting algorithm that turns multiple weak learners into strong learners. The algorithm calculates the gradient of the loss function in relation to the current ensemble's predictions for each iteration, and then trains a new weak model to minimize this gradient. Next, the new model's predictions are included in the ensemble, and the process is continued until a stopping requirement is satisfied. Unlike AdaBoost, each predictor is trained using the residual errors of the previous model as labels rather than adjusting the weights of the training instances. There is a technique called the Gradient Boosted Trees whose base learner is CART (Classification and Regression Trees) [9].

Chapter 4

Code /Code Templates

4.1 Subtask One – Binomial Classification (Human generated vs AI generated text):

[1]:
```
import numpy as np
import pandas as pd
import re
```

[2]:
```
#pip install -U scikit-learn
```

[3]:
```
data=pd.read_csv('chatgpt_paraphrases.csv')
data.head()
```

[5]:
```
category={}
for i in range(len(data)):
    ai=data.iloc[i]["paraphrases"][1:-1].split(', ')
    for j in ai[:1]:
        category[j[1:-1]]='ai'
    category[data.iloc[i]['text']]="human"
```

[6]:
```
data=pd.DataFrame(category.items(),columns=["text","category"])
data.info()
```

[7]:
```
data=data.sample(frac=1)
data=data[:20000]
data
```

[9]:
```
from sklearn.model_selection import train_test_split
X=data['text']
y=data['category']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

[10]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer()
X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test)
```

[11]:
```
from sklearn.ensemble import VotingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.naive_bayes import MultinomialNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
```

```
from sklearn.ensemble import␣
↪RandomForestClassifier,ExtraTreesClassifier,AdaBoostClassifier,BaggingClassifier,
```

[12]:
```
lg = LogisticRegression(penalty='l1',solver='liblinear')
sv = SVC(kernel='sigmoid',gamma=1.0)
mnb = MultinomialNB()
dtc = DecisionTreeClassifier(max_depth=5)
knn = KNeighborsClassifier()
rfc = RandomForestClassifier(n_estimators=50,random_state=2)
etc = ExtraTreesClassifier(n_estimators=50,random_state=2)
abc = AdaBoostClassifier(n_estimators=50,random_state=2)
bg = BaggingClassifier(n_estimators=50,random_state=2)
gbc = GradientBoostingClassifier(n_estimators=50,random_state=2)
```

[13]:
```
from sklearn import metrics
```

[14]:
```
def score_prediction(model,X_train,X_test,y_train,y_test):
    model.fit(X_train,y_train)
    pr = model.predict(X_test)
    acc_score = metrics.accuracy_score(y_test,pr)
    pre_score = metrics.precision_score(y_test,pr,average="binary",␣
 ↪pos_label="ai")
    recall= metrics.recall_score(y_test,pr,average="binary", pos_label="ai")
    f1= metrics.f1_score(y_test,pr,average="binary", pos_label="ai")
    mcc= metrics.matthews_corrcoef(y_test,pr)
    return acc_score,pre_score,recall,f1,mcc
```

4.2 Subtask 2 -Multi-Classification (Human generated vs AI generated text):

[1]:
```
import numpy as np
import pandas as pd
import re
```

[2]:
```
df=pd.read_csv('subtaskB_dev.csv')
df.head()
```

[7]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

[8]:
```
sns.set_style('whitegrid')
sns.countplot(x='model',hue='model',data=df)
```

[16]:
```
from sklearn.ensemble import VotingClassifier
```

[14]:
```
from sklearn.model_selection import train_test_split
X=df['text']
y=df['model']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

[15]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer()
X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test)
```

```
#mnb = MultinomialNB()
#dtc = DecisionTreeClassifier(max_depth=5)
knn = KNeighborsClassifier()
rfc = RandomForestClassifier(n_estimators=50,random_state=2)
etc = ExtraTreesClassifier(n_estimators=50,random_state=2)
abc = AdaBoostClassifier(n_estimators=50,random_state=2)
bg = BaggingClassifier(n_estimators=50,random_state=2)
gbc = GradientBoostingClassifier(n_estimators=50,random_state=2)
```

[18]:
```
from sklearn import metrics
```

[19]:
```
def score_prediction(model, X_train, X_test, y_train, y_test):
    model.fit(X_train, y_train)
    pr = model.predict(X_test)
    acc_score = metrics.accuracy_score(y_test, pr)
    pre_score = metrics.precision_score(y_test, pr, average="weighted")
    recall = metrics.recall_score(y_test, pr, average="weighted")
    f1 = metrics.f1_score(y_test, pr, average="weighted")
    mcc = metrics.matthews_corrcoef(y_test, pr)
    return acc_score, pre_score, recall, f1, mcc
```

Chapter 5

Results and Discussion

The results of the Multigenerator and Multidomain Machine-Generated Text Detection project are summarized based on various evaluation metrics for different classifiers:

Binomial Classification Scores Table

	Accuracy score	Precision score	Recall	f1_score	Matthews correlation coefficient
Logistic Regression	**0.749**	**0.7520**	**0. 7315**	**0. 7416**	**0. 4979**
SVM	**0.76375**	**0.7795**	**0. 7254**	**0. 7515**	**0. 5282**
KNN	**0.665**	**0.6656**	**0. 6426**	**0. 6539**	**0. 3297**
Random Forest	**0.77**	**0.7521**	**0. 7949**	**0. 773**	**0. 5411**
Extra Tree	**0.7845**	**0.7750**	**0. 7924**	**0. 7836**	**0. 5692**
AdaBoost	**0.718**	**0.6864**	**0. 7868**	**0. 7332**	**0. 4417**
Bagging	**0.7335**	**0.71523**	**0. 7624**	**0. 7381**	**0. 4684**
Gradient Boosting	**0.72175**	**0.7001**	**0. 7609**	**0. 7293**	**0. 4457**

Table 1

Multi-classification Score Table

	Accuracy score	Precision score	Recall	f1_score	Matthews correlation coefficient
Logistic Regression	**0.9447**	**0.944**	**0. 9447**	**0. 9447**	**0. 9335**
SVM	**0.9584**	**0.9592**	**0. 9584**	**0. 9586**	**0. 9501**
KNN	**0.3659**	**0.5176**	**0. 3659**	**0. 3887**	**0. 2576**
Random Forest	**0.9255**	**0.9251**	**0. 9255**	**0. 9247**	**0. 9106**
Extra Tree	**0.9337**	**0.9334**	**0. 9337**	**0. 9330**	**0. 9205**
AdaBoost	**0.3734**	**0.3372**	**0. 3734**	**0. 3028**	**0. 3123**
Bagging	**0.8884**	**0.8882**	**0. 8884**	**0. 8881**	**0. 8660**
Gradient Boosting	**0.9261**	**0.92606**	**0. 9261**	**0. 9259**	**0. 9113**

Table 2

5.1. Accuracy:

In this project the Extra Tree Classifier is found to have the highest accuracy of 78.45% for binomial classification and Support Vector with 95.84% accuracy for multi-classification. Accuracy score is important because it is a simple indicator of model performance. Although high accuracy is preferred but 100% accuracy is a sign of some error, such as overfitting. It could also indicate logical flaw or data leakage.

Accuracy = (number of correct predictions)/(total number of predictions) [23]

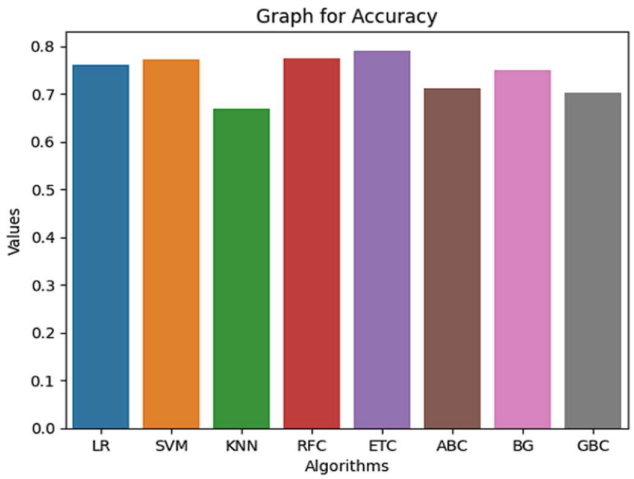

Fig. 1.1

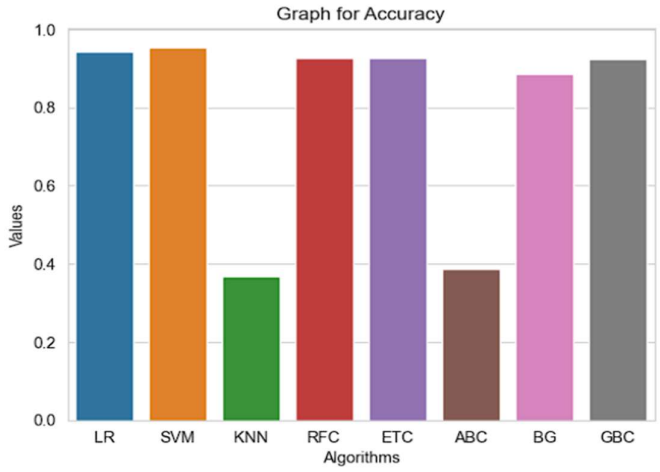

Fig. 1.2

5.2. Precision Score:

Looking at the table 'Scores Table' above we find that SVM gives the highest precision score of 77.95% and Extra Tree Classifier falls just behind giving 77.50% for binomial classification and Support Vector with 95.92% precision for multi-classification.

The precision score is calculated using the following formula:

Precision = (True Positives) / (True Positives + False Positives) [23]

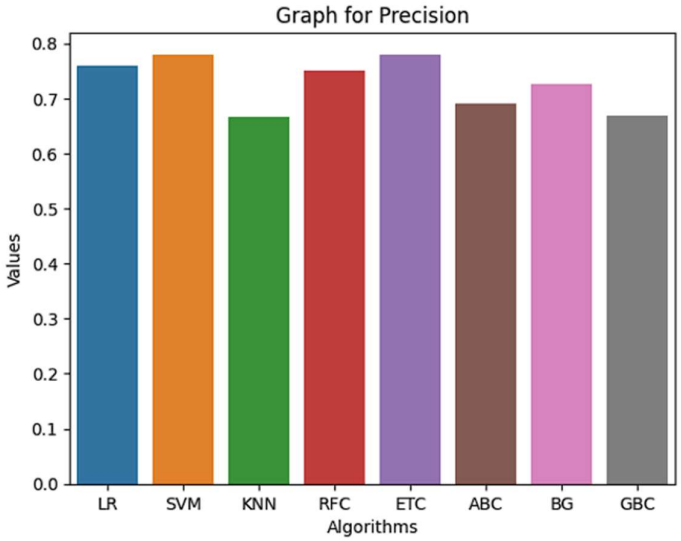

Fig. 2.1

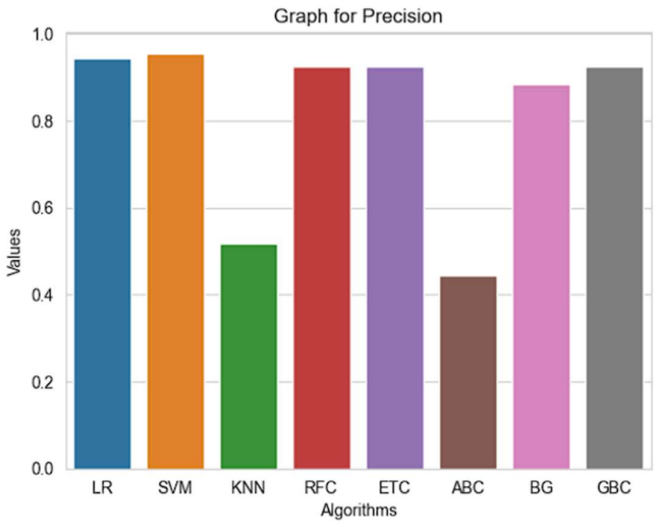

Fig. 2.2

5.3. Recall Scores:

Looking at the table ‘Table 1’ above we find that Random Forrest gives the highest precision score of 79.49% and Extra Tree Classifier falls just behind giving 79.23% . ‘Table 2’ shows that SVM has the highest recall score of 95.84%.

The higher the recall score, the better the model is at identifying both positive and negative examples. Similarly, a low recall score indicates that the model is not good at identifying positive examples.

Recall = True Positives / (False Negatives + True Positives) [23]

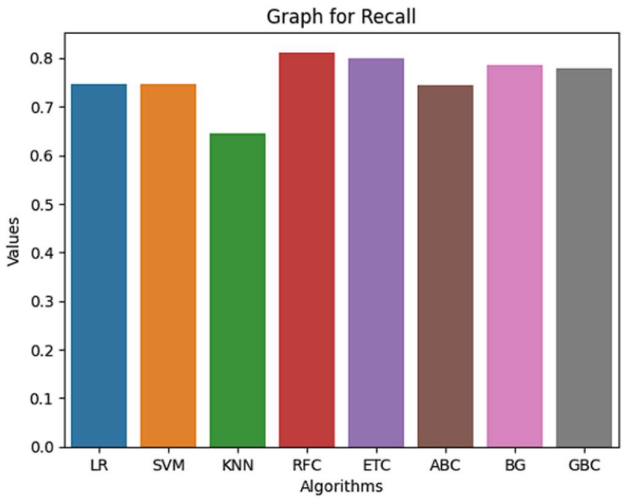

Fig. 3.1

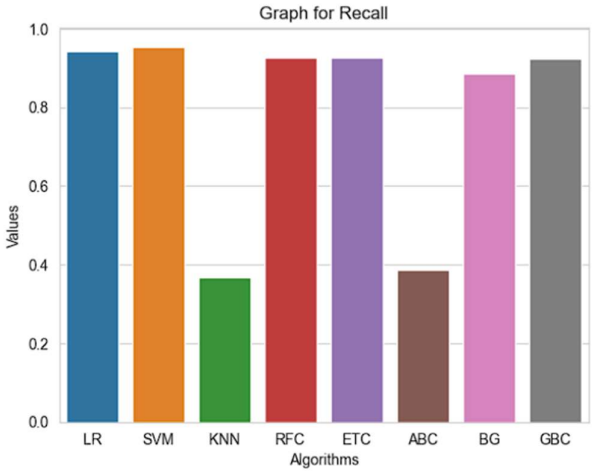

Fig. 3.2

5.4. F1 Scores:

Looking at the table ‘Table 1’ above we find that Random Forest gives the highest precision score of 78.36% and Extra Tree Classifier falls just behind giving 77.50% and ‘Table 2’ shows that SVM has the highest f1 score of 95.86%.

F1 = 2 * (precision * recall) / (precision + recall) [23]

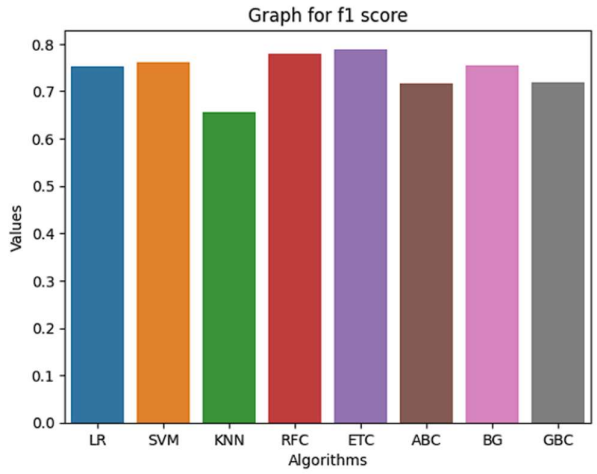

Fig. 4.1

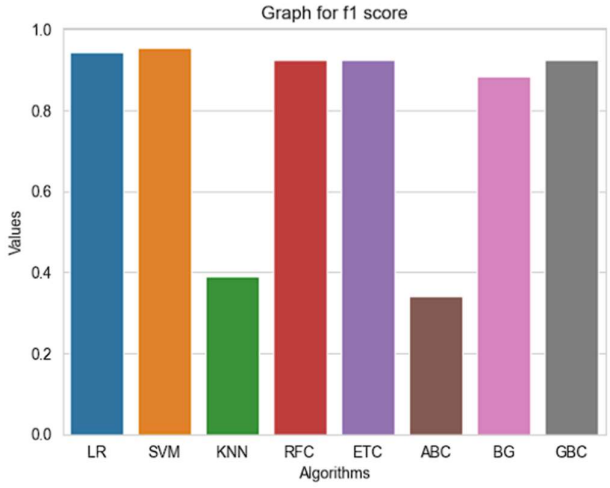

Fig. 4.2

5.5. Matthews Correlation Coefficient (MCC) Scores:

The MCC ranges from -1 to +1, where +1 indicates perfect prediction, 0 indicates no better than random prediction, and -1 indicates total disagreement between prediction and observation[19].

'Table 1' shows that ETC is the best algorithm for binary classification with a mcc score of 56.92% while for multi-classification SVM is the better choice with a mcc score of 95.01%.

The Matthews Correlation Coefficient is calculated using the following formula:

Mcc = (TP * TN – FP * FN) / √ ((TP + FP) * (TP + FN) * (TN + FP) * (TN + FN)) [24]

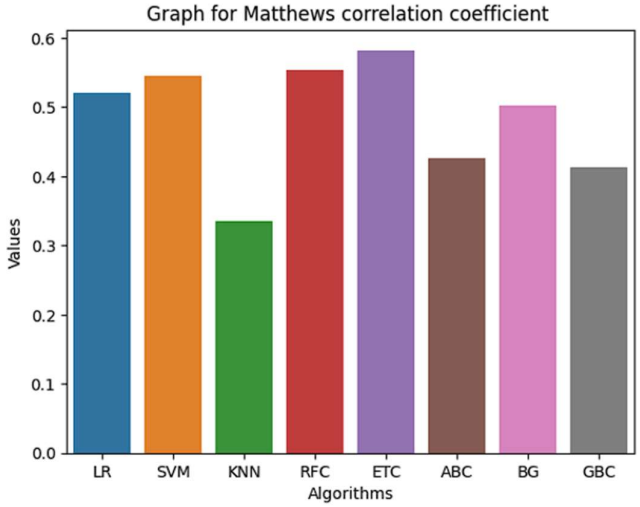

Fig. 5.1

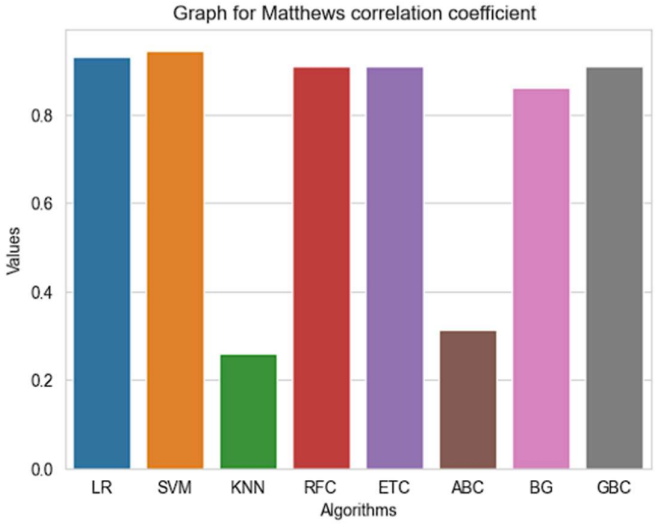

Fig. 5.2

5.6. Heatmap Visualization of the Confusion Matrix:

Fig. 6 Confusion Matrix Visualization

Chapter 6

Conclusion and Future Work

6.1 Conclusion:

The Extra Trees Classifier (ETC) stands out as the top-performing model (for binary classification of text between human vs AI) and SVM as a better model for multi-classification of text generated by different text generating models, demonstrating their robustness in distinguishing between machine-generated and human-written text. It excels in achieving a high balance between precision and recall, making it a suitable choice for the specified task. However, it's crucial to consider the specific requirements of the project and choose a model based on the importance of precision, recall, or a trade-off between the two.

6.2 Future Work

Building on the promising results of this study, future work will focus on several key areas to enhance the detection and classification of AI-generated text. First, we will explore advanced deep learning techniques, such as transformer-based models like BERT and its variants, to improve the classification accuracy further. Additionally, we plan to expand the dataset to include a more diverse range of AI text generators and human-written texts from various domains and styles, thereby increasing the robustness and generalizability of our models.

We also intend to implement real-time detection systems that can operate efficiently on live data streams, which will be crucial for applications in content moderation and academic integrity. Another vital aspect of our future work involves developing adversarial training techniques to make our models more resilient to sophisticated attempts at evading detection, such as adversarial attacks that slightly modify AI-generated text to mimic human writing styles more closely.

Lastly, we aim to collaborate with educational institutions and online platforms to integrate our detection systems into their workflows, providing practical tools to uphold academic integrity and prevent the spread of misinformation. By continuously refining our models and expanding their application scope, we hope to contribute significantly to the ethical and reliable use of AI technologies in digital communication.

Chapter 7

Details of Research Publication

The details of our research publication are as follows:

1. Abhishek Singh, Aniket Raj, Siddhant Rawat and Rohan Verma, “Multigenerator and Multidomain Machine-Generated Text Detection” (Communicated to Supervisor).

Chapter 8

References

[1] https://numpy.org/doc/stable/user/absolute_beginners.html

[2] https://www.ibm.com/topics/logistic-regression

[3] https://www.geeksforgeeks.org/support-vector-machine-algorithm

[4] https://www.geeksforgeeks.org/k-nearest-neighbours/

[5] https://www.javatpoint.com/machine-learning-random-forest-algorithm

[6] https://www.geeksforgeeks.org/ml-extra-tree-classifier-for-feature-selection/

[7] https://www.almabetter.com/bytes/tutorials/data-science/adaboost-algorithm

[8] https://www.geeksforgeeks.org/ml-bagging-classifier/

[9] https://www.geeksforgeeks.org/ml-gradient-boosting/

[10] M4: Multi-generator, Multi-domain, and Multi-lingual Black-Box Machine-Generated Text Detection Yuxia Wang, Jonibek Mansurov,∗Petar Ivanov,∗ Jinyan Su,∗ Artem Shelmanov,∗ Akim Tsvigun, Chenxi Whitehouse, Osama Mohammed Afzal, Tarek Mahmoud, Alham Fikri Aji, Preslav Nakov

[11] A. Harada, D. Bollegala and N. P. Chandrasiri, "Discrimination of human-written and human and machine written sentences using text consistency," 2021 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS), Greater Noida, India, 2021, pp. 41-47, doi: 10.1109/ICCCIS51004.2021.9397237.

[12] H. Alamleh, A. A. S. AlQahtani and A. ElSaid, "Distinguishing Human-Written and ChatGPT-Generated Text Using Machine Learning," 2023 Systems and Information Engineering Design Symposium (SIEDS), Charlottesville, VA, USA, 2023, pp. 154-158, doi: 10.1109/SIEDS58326.2023.10137767.

[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In Annual Conference on Neural Information Processing Systems (NIPS), pages 5998–6008. NIPS, 2017.

[14] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel HerbertVoss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam

McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language Models are Few-Shot Learners. In Annual Conference on Neural Information Processing Systems (NeurIPS). NeurIPS, 2020.

[15] Paul F. Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep Reinforcement Learning from Human Preferences. In Annual Conference on Neural Information Processing Systems (NIPS), pages 4299–4307. NIPS, 2017.

[16] https://platform.openai.com/docs/guides/text-generation

[17] https://huggingface.co/databricks/dolly-v2-12b

[18] https://docs.cohere.com/docs

[19]https://codecanyon.net/item/openai-davinci-ai-writing-assistant-and-content-creator-as-saas/43564164

[20] https://pandas.pydata.org/docs/

[21] https://scikit-learn.org/0.18/_downloads/scikit-learn-docs.pdf

[22]https://scikitlearn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

[23] https://scikit-learn.org/0.21/_downloads/scikit-learn-docs.pdf

[24] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.matthews_corrcoef.html

[25] Nisan Stiennon, Long Ouyang, Jeff Wu, Daniel M. Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F. Christiano. Learning to summarize from human feedback. CoRR abs/2009.01325, 2020.

[26] Das, B., & Chakraborty, S. (2018). An Improved Text Sentiment Classification Model Using TF-IDF and Next Word Negation. *ArXiv, abs/1806.06407*