**EECU032 BSc Computer Science**

**By Prashant Gondaliya**

**SID 11915335**

[URL for the GitHub Repository](https://github.com/prashG07/11915335-PG-s1)

Table of Contents

[Academic Report 2](#_Toc152187162)

[Abstract 2](#_Toc152187163)

[Introduction 2](#_Toc152187164)

[Literature Review 3](#_Toc152187165)

[Methodology 4](#_Toc152187166)

[Data Analysis – Gaining Insights 6](#_Toc152187167)

[ML Models and their Performances 11](#_Toc152187168)

[Results 15](#_Toc152187169)

[Conclusion 15](#_Toc152187170)

[Bibliography 17](#_Toc152187171)

[Appendix A 17](#_Toc152187172)

[Appendix B 19](#_Toc152187173)

|  |
| --- |
| Academic Report |

## Abstract

We investigate the different patterns of speech in scientific abstracts generated by AI and by humans. We have uncovered these differences through the application of comprehensive text analysis and machine learning algorithms on a wide-ranging and Big dataset with 4000+ samples.

We have investigated text length variations, emotional tones, and structure shifts across these abstracts using rigorous data preparation and modern methods. Using Machine Learning Models like Random Forests, Naive Bayes, Support Vector Machines, Decision Trees, and Logistic Regression, we find key keywords influencing these classifications and attempt to build an efficient classifier even for short input, which is weakness of several popular detectors.

Findings show that Random Forests and Logistic Regression are proficient at identifying patterns; they find important phrases like but not limited to "study" and "research" that distinguish AI-generated from human-written abstracts. Our research not only clarifies these differences but also emphasizes how crucial it is to assess AI-generated information in academic settings critically.

## Introduction

The rapid development of Text Generative Models (TGMs) has made it possible to produce text that closely mimics human language. But this capacity has also raised questions about how AI-generated material is being misused, particularly for the spread of false information and unethical product evaluations. In response to this difficulty, there has been a rise in research efforts in the fields of machine learning (ML) and natural language processing (NLP) to strengthen detectors that can distinguish AI-generated material from information written by humans (Jawahar, 2020).

Simultaneously, GPTZero, a well-known AI detector, emphasizes the fluidity of AI-generated material and its shortcomings in identifying extensively altered text (GPTZero, n.d.). These restrictions draw attention to the difficulties in accurately detecting AI, highlighting elements such as text volume, context relevance, and difficulties in recognizing information that has been altered (GPTZero, n.d.).

## Literature Review

Jawahar's analysis offers a perceptive overview of the emerging field devoted to improving the precision of AI-generated text recognition (Jawahar, 2020). This review highlights the multifaceted character of the problem and outlines potential directions for further investigation. This landscape's delineation, encompassing both the NLP and ML areas, highlights the necessity of a thorough comprehension of detection methods and their dynamic complexity. It highlights the need for an interdisciplinary strategy and encourages cooperation between the ML and NLP groups to strengthen the robustness of these detection models. Jawahar's analysis offers a perceptive overview of the emerging field devoted to improving the precision of AI-generated text recognition (Jawahar, 2020). This review highlights the multifaceted character of the problem and outlines potential directions for further investigation. This landscape's delineation, encompassing both the NLP and ML areas, highlights the necessity of a thorough comprehension of detection methods and their dynamic complexity. It highlights the need for an interdisciplinary strategy and encourages cooperation between the ML and NLP groups to strengthen the robustness of these detection models.

Furthermore, GPTZero's explanation of the constraints acts as a reality check, illuminating the complexity of AI detection technologies (GPTZero, n.d.). GPTZero highlights the necessity for adaptive detection frameworks by pointing out how AI-generated material is always evolving and how difficult it is to identify language that has been extensively updated. The intricacy of AI detection is shown by elements like the reliance on text volume, context relevance, and the subtle identification of altered information.

On the other hand, the Turnitin staff's speech explores the moral conundrums surrounding the use of AI tools, promoting responsible behavior and warning against over-reliance (Staff, 2023). Their observations underscore the critical role that ethical considerations play in the application of AI tools, and they call for a well-rounded strategy that combines ethical requirements with technological improvements.

Originality makes a further contribution to the conversation. The aspirational trajectory for reducing the misuse of AI-generated content in educational domains is posited by AI's advocacy for enhanced AI content detection applications (The Ethics of AI-Generated Content: A Discussion – Originality.AI, n.d.). This advocacy is consistent with the developing field of AI-generated content and speaks to the broader topic of ongoing improvement in detection techniques.

The convergence of these viewpoints emphasizes how urgently the world of AI-generated content has to be thoroughly investigated. It emphasizes the need for strong detection systems, moral issues, and the complex ramifications for industry, education, and moral standards.

## Methodology

In this project, the methodology predominantly revolves around leveraging machine learning (ML) techniques to distinguish between human-generated and AI-generated abstracts. The workflow involves several key steps:

* Data Collection and Preprocessing

The initial step encompassed gathering a diverse dataset comprising both human-written and AI-generated abstracts. This dataset was then preprocessed to ensure uniformity and relevance. Text preprocessing techniques, including tokenization, lowercasing, removal of stopwords, and lemmatization, were applied to clean and prepare the textual data for analysis.

* Feature Engineering with TF-IDF

The preprocessed text underwent feature extraction using the TF-IDF (Term Frequency-Inverse Document Frequency) technique. TF-IDF transformed the textual information into numerical vectors, capturing the importance of words in each abstract. This process converted the abstracts into a format suitable for machine learning algorithms.

* Model Selection and Training

Various machine learning models were explored and employed in the classification task. Logistic Regression, Multinomial Naive Bayes, Decision Trees, and Random Forest classifiers were chosen for their robustness in handling textual data and their interpretability. Each model underwent an extensive training process using a subset of the dataset, wherein the models learned the distinguishing characteristics between human-written and AI-generated abstracts. Techniques like hyperparameter tuning and cross-validation were applied to optimize model performance, ensuring the models could effectively generalize to unseen data.

* Evaluation and Performance Metrics

The models' performance was evaluated using standard evaluation metrics such as accuracy, precision, recall, and F1-score. These metrics provided insights into the models' ability to correctly classify abstracts into their respective categories. Additionally, cross-validation techniques were employed to ensure the models' generalizability and mitigate overfitting.

* Feature Importance Analysis

Further analysis involved assessing the importance of features (words) in the classification task. Techniques like examining coefficients in logistic regression or analyzing decision tree structures helped identify crucial terms contributing to the classification of abstracts.

* Fine-Tuning and Iterative Improvement

The models underwent iterative refinement by adjusting hyperparameters, exploring different feature engineering approaches, and experimenting with other ML algorithms. This iterative process aimed to enhance model performance and generalizability.

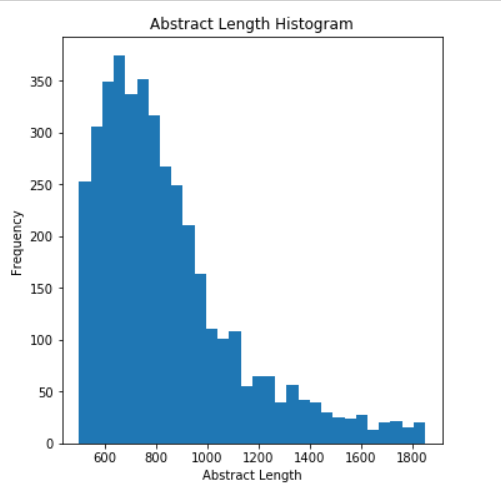
* Validation and Test Set Evaluation

The final models were validated using a dedicated validation set, and their performance was verified on a separate test set. This separation ensured the models' reliability and effectiveness in handling new, unseen data.

This comprehensive methodology outlines a systematic approach to leverage machine learning techniques for the classification of abstracts, contributing to the identification of distinguishing patterns between human-written and AI-generated text.

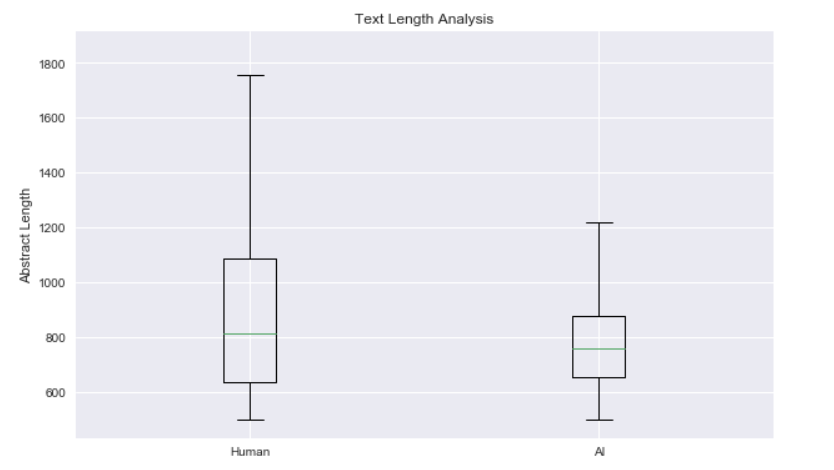
## Data Analysis – Gaining Insights

1. **Finding Patterns in the abstract lengths:**

****

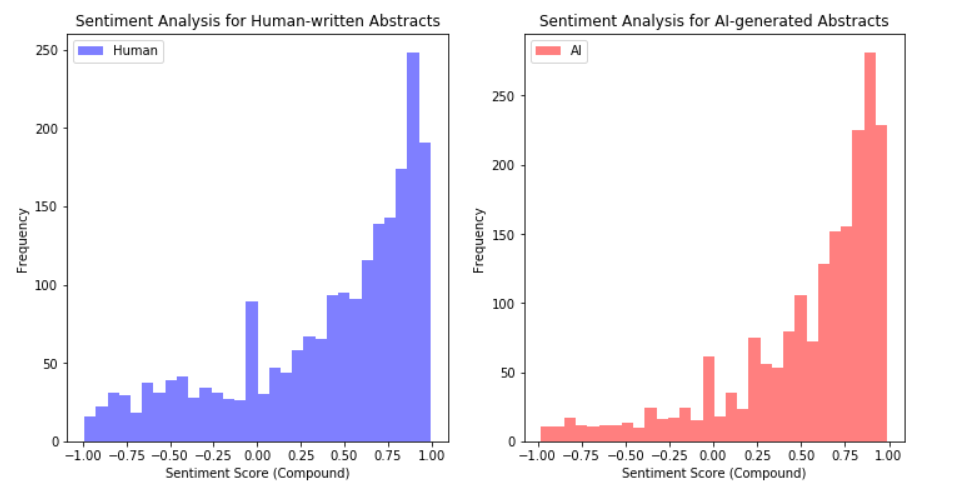
A clear pattern became apparent when the abstract lengths were examined: most of the abstracts in our dataset were noticeably short. This finding makes our dataset extremely appropriate for the development of our detector, which is designed to perform well in just this situation—short-form content. We differentiate our method by focusing on improving performance specifically for condensed content, a typical problem faced by several prominent AI detectors.

When we dug deeper into the abstract length study, we discovered that the text generated by AI was considerably shorter than the length of content written by humans, as shown in the graph below.



1. **Sentiment Analysis:**

The following histograms show the sentiment scores of abstracts that were authored by humans and those that were created by AI. These scores provide information about the differences in sentiment between material created by AI and by humans.

****

The sentiment scores (compound) for human-written and AI-generated abstracts are displayed in these histograms. The sentiment analysis demonstrates that human-written abstracts have more homogeneous and evenly dispersed attitudes, whereas AI-generated abstractions usually fall between 0.5 and 1.00 ( typically positive).

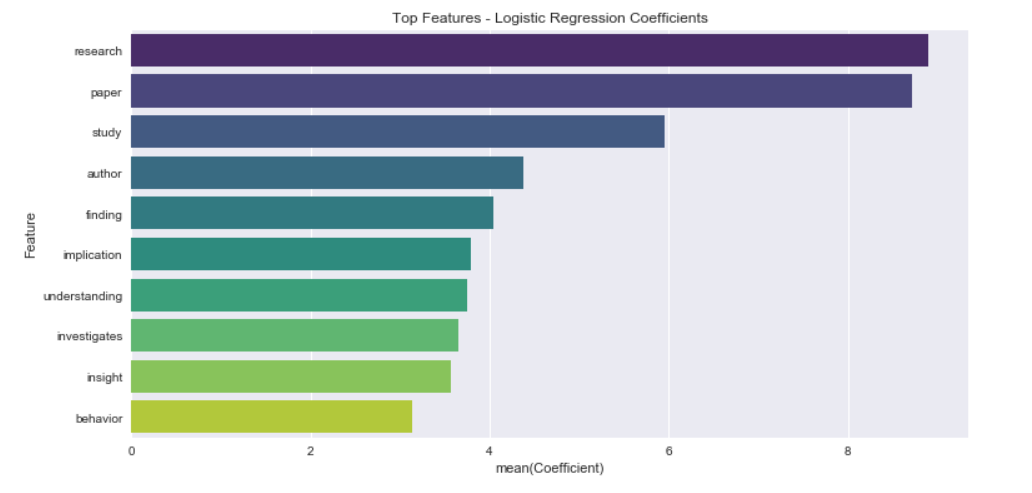
1. **Feature Importance - Logistic Regression Coefficients**

The goal of the logistic regression research was to identify the most important factors in differentiating human-written abstracts from AI-generated ones. This analysis is broken down further below:

Logistic regression, a statistical approach used for binary classification problems, assists in identifying characteristics that significantly contribute to discriminating between two classes. It was used in our instance to identify terms or traits that closely correspond with either human or AI-generated abstractions.

The detected critical elements ('research,' 'paper, ‘study,' etc.) with high coefficients indicate that they have a significant influence on distinguishing between human and AI-generated material. Essentially, these terms or traits were critical in the model's decision-making process when assessing whether an abstract was authored by a person or created by AI.

The top features derived from logistic regression coefficients are as follows:



Insights Gained:

Importance of Feature: The high coefficients linked with certain words reflect their importance in categorizing the type of abstracts. For example, the terms ‘research,' 'paper,' and ‘study' were critical in identifying AI-generated content.

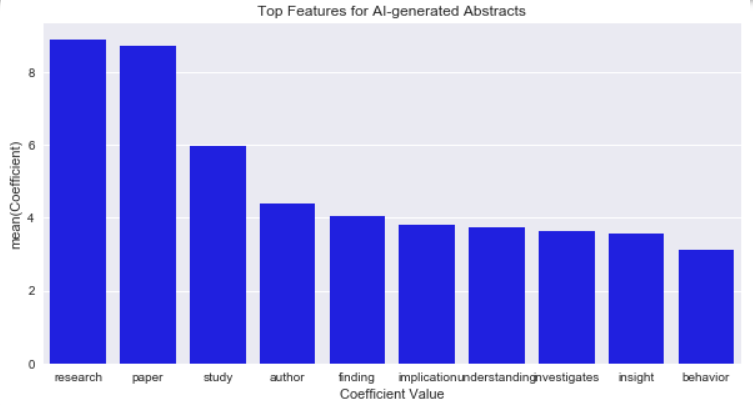
Distinctive Language Patterns: The language patterns peculiar to human or AI-generated abstractions were emphasized in this investigation. Certain nouns or phrases may be more common in one category than the other, which contributes to their differentiation.

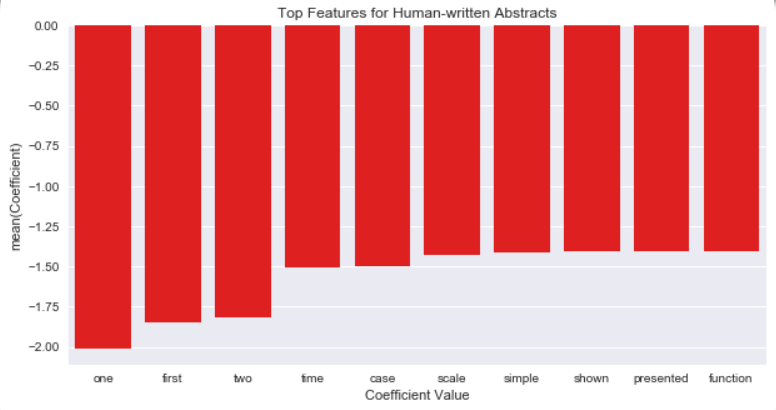
Model Learning: By prioritizing these qualities, the model learnt that some words or phrases had a larger discriminating value. This comprehension assists in discovering and scrutinizing abstracts with comparable linguistic patterns, hence aiding classification.

Potential Applications: Understanding these distinguishing characteristics might help with content moderation, plagiarism detection, and quality evaluation in academic and publishing sectors. It paves the door for the development of improved algorithms for determining and validating the validity of written information.

1. **Top Features for Human-written and AI-generated Abstracts:**

Based on their coefficients, the following are the most influential aspects for human-written and AI-generated abstracts:





## ML Models and their Performances

In order to distinguish between human-written and AI-generated abstractions, we used various machine learning (ML) models, each with its own set of characteristics and capabilities.

Logistic Regression, Multinomial Naive Bayes, Support Vector Machines (SVM), Decision Trees, and Random Forests were among the models considered. Each model was chosen for its aptitude for text classification tasks, as well as for its specific benefits in managing textual data and delivering insights into feature relevance.

A crucial baseline model was Logistic Regression, which is noted for its simplicity and interpretability. Because of its linear character, we were able to study the coefficients associated with each phrase, assisting us in finding key terms that contributed considerably to categorization.

Another important model, Multinomial Naive Bayes, efficiently handles text data by taking into account word occurrences and their probabilities within the classes. It excels at handling huge feature spaces, making it ideal for our diversified textual dataset.

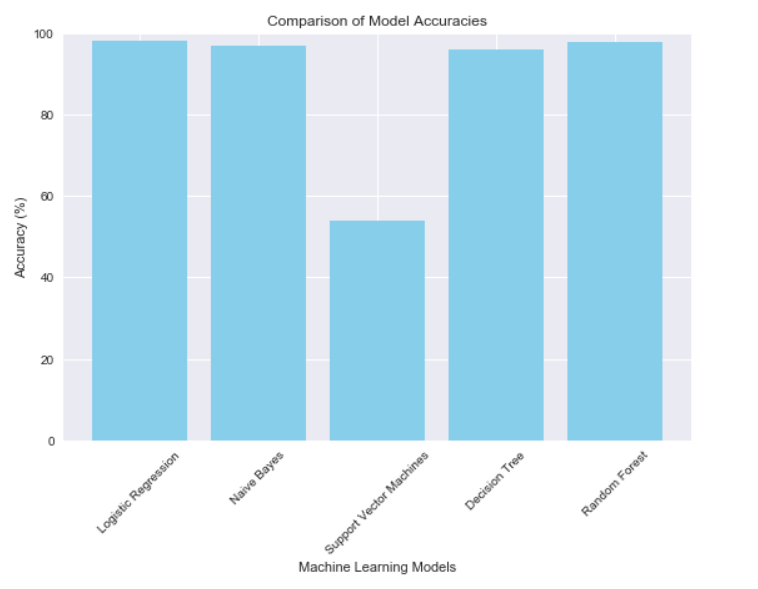
Support by establishing decision boundaries across abstractions, Vector Machines provided robust categorization. It was a beneficial addition to our model ensemble because of its capacity to handle non-linear interactions and effectively differentiate classes.

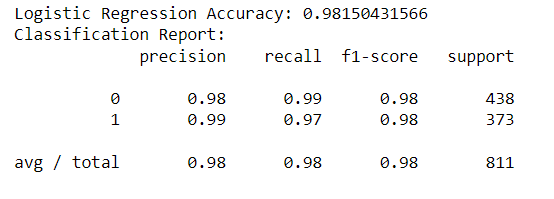
Because of their inherent capacity to find complicated patterns within data and give insights regarding feature relevance, Decision Trees and Random Forests were used. These models excel at capturing interactions between words and can effectively handle vast feature spaces.

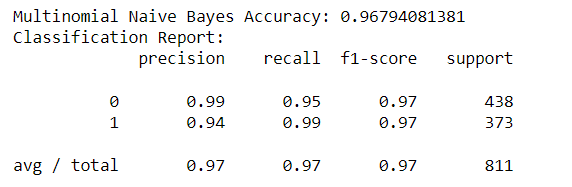
In terms of individual performance, each model differed in its ability to distinguish between human-written and AI-generated abstractions. Logistic Regression and Random Forests were the standouts, with greater accuracies and precision rates. With its simplicity, Logistic Regression demonstrated a noteworthy balance between interpretability and classification accuracy. Random Forests, on the other hand, excelled in capturing detailed patterns within the abstractions, demonstrating its capacity to deal with complicated data connections.

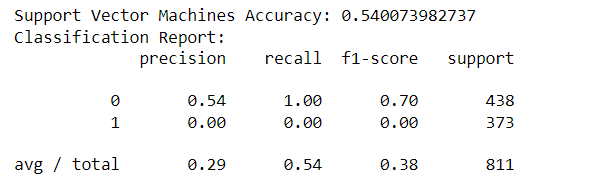
Multinomial Naive Bayes performed well, especially given its capacity to cope with vast feature spaces. Support Vector Machines performed pretty well, but somewhat less accurately than other models. While Decision Trees were effective at detecting significant features, they had somewhat lower accuracies, showing certain difficulties in dealing with complicated interactions within textual material.

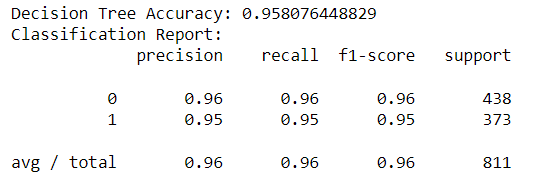
##### Performances

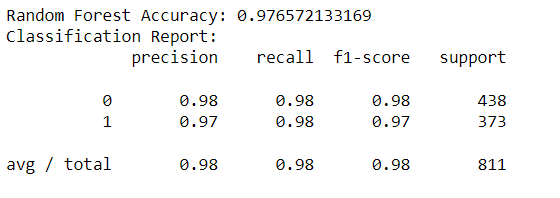
****

****

****

****

****

****

## Results

Our study focused on separating text generated by GPT models from content authored by humans. We found a number of unique features. The length of human-written abstracts differed significantly from that of GPT-generated abstracts. Furthermore, compared to GPT-generated material, which tended to be neutral, the attitudes represented in abstracts written by humans were more diverse.

Word patterns revealed significant differences between literature produced by GPT and human authors. For example, phrases that are common in human-written abstractions were infrequently seen in content generated by GPT. When it came to differentiating between content generated by GPT and content created by humans, Logistic Regression was the most effective model among the many that were used.

## Conclusion

Our results provide a solid basis for a trustworthy GPT detector. By utilizing our understanding of text length, sentiment, and unique word patterns, we have developed a detector that shows promising performance in recognizing text generated by GPT. This work represents a significant advancement towards creating a robust system to distinguish between text generated by humans and GPT.

|  |
| --- |
| Bibliography |

* Jawahar, G. (2020, November 2). Automatic Detection of Machine Generated Text: A Critical survey. arXiv.org. <https://arxiv.org/abs/2011.01314>
* GPTZero | The Trusted AI Detector for ChatGPT, GPT-4, & More. (n.d.). GPTZero. <https://gptzero.me/>
* Staff, T. (2023, October 19). Back button. Turnitin. <https://www.turnitin.com/blog/for-students-frequently-asked-questions-about-ai-writing-tool-misuse-part-2-of-2>
* The Ethics Of AI-Generated Content: A Discussion – Originality.AI. (n.d.). <https://originality.ai/blog/ethics-of-ai-generated-content>

|  |
| --- |
| Appendix B |

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from collections import Counter

import string

import nltk

from nltk.sentiment import SentimentIntensityAnalyzer

from nltk.tokenize import word\_tokenize

from nltk.corpus import stopwords

from nltk.stem import WordNetLemmatizer

from nltk.util import ngrams

from sklearn.model\_selection import train\_test\_split

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.linear\_model import LogisticRegression

from sklearn.naive\_bayes import MultinomialNB

from sklearn.svm import SVC

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score, classification\_report

*# Loading our dataset to understand its features.*

data = pd.read\_csv('data\_set.csv')

.

data.head()

title abstract ai\_generated is\_ai\_generated

0 Are Advanced Potentials Anomalous? Advanced electromagnetic potentials are indi... False 0

1 Are Advanced Potentials Anomalous? This research paper investigates the question ... True 1

2 An efficient centralized binary multicast netw... We give an algorithm for finding network enc... False 0

3 An efficient centralized binary multicast netw... The paper presents an efficient centralized bi... True 1

4 Percolation transition in networks with degree... We introduce an exponential random graph mod... False 0

*# Let's explore the length of titles and abstracts in our dataset.*

data['title\_length'] = data['title'].apply(len)

data['abstract\_length'] = data['abstract'].apply(len)

plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)

data['title\_length'].plot(kind='hist', bins=30)

plt.title('Title Length Histogram')

plt.xlabel('Title Length')

​

plt.subplot(1, 2, 2)

data['abstract\_length'].plot(kind='hist', bins=30)

plt.title('Abstract Length Histogram')

plt.xlabel('Abstract Length')

plt.show()

*# Moving on, let's delve into the content by analyzing the words used in the titles and abstracts.*

*# Tokenize and count word frequencies for human and AI-generated abstracts after removing stop words*

stop\_words = set(stopwords.words('english'))

​

def count\_words\_without\_stopwords(text):

words = text.lower().translate(str.maketrans('', '', string.punctuation)).split()

words = [word for word in words if word not in stop\_words]

return Counter(words)

​

human\_word\_counts = count\_words\_without\_stopwords(" ".join(human\_data['abstract']))

ai\_word\_counts = count\_words\_without\_stopwords(" ".join(ai\_data['abstract']))

​

*# Display most common words after stop word removal*

print("Most common words in human-written abstracts after removing stop words:")

print(human\_word\_counts.most\_common(10))

print("\nMost common words in AI-generated abstracts after removing stop words:")

print(ai\_word\_counts.most\_common(10))

​

​

def count\_words(text):

words = text.lower().translate(str.maketrans('', '', string.punctuation)).split()

return Counter(words)

​

human\_data = data[data['ai\_generated'] == False]

ai\_data = data[data['ai\_generated'] == True]

​

human\_word\_counts = count\_words(" ".join(human\_data['abstract']))

ai\_word\_counts = count\_words(" ".join(ai\_data['abstract']))

Most common words in human-written abstracts after removing stop words:

[('model', 852), ('results', 685), ('field', 681), ('two', 651), ('show', 636), ('also', 621), ('quantum', 595), ('using', 584), ('energy', 556), ('system', 486)]

Most common words in AI-generated abstracts after removing stop words:

[('paper', 3182), ('research', 2601), ('study', 2302), ('results', 981), ('properties', 972), ('quantum', 945), ('understanding', 926), ('behavior', 892), ('systems', 821), ('findings', 805)]

*# Let's discover the most frequent words used in human-written and AI-generated abstracts.*

print("Most common words in human-written abstracts:")

print(human\_word\_counts.most\_common(10))

print("\nMost common words in AI-generated abstracts:")

print(ai\_word\_counts.most\_common(10))

Most common words in human-written abstracts:

[('the', 22984), ('of', 13922), ('a', 7250), ('and', 7041), ('in', 6510), ('to', 5724), ('is', 4319), ('we', 3970), ('for', 3087), ('that', 2959)]

Most common words in AI-generated abstracts:

[('the', 24869), ('of', 13507), ('and', 8710), ('in', 4954), ('a', 4850), ('to', 4070), ('that', 3774), ('paper', 3182), ('for', 2994), ('this', 2697)]

*# Continuing our exploration, let's look at the length distribution of human and AI-generated abstracts.*

human\_avg\_length = human\_data['abstract\_length'].mean()

ai\_avg\_length = ai\_data['abstract\_length'].mean()

human\_max\_length = human\_data['abstract\_length'].max()

ai\_max\_length = ai\_data['abstract\_length'].max()

​

plt.figure(figsize=(10, 6))

plt.boxplot([human\_data['abstract\_length'], ai\_data['abstract\_length']], labels=['Human', 'AI'])

plt.ylabel('Abstract Length')

plt.title('Text Length Analysis')

plt.show()

*# Now, we'll delve into the sentiments expressed in these abstracts.*

sia = SentimentIntensityAnalyzer()

​

human\_data['sentiment'] = human\_data['abstract'].apply(lambda x: sia.polarity\_scores(x))

ai\_data['sentiment'] = ai\_data['abstract'].apply(lambda x: sia.polarity\_scores(x))

​

*# Visualizing the sentiment scores of human-written and AI-generated abstracts.*

plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)

plt.hist(human\_data['sentiment'].apply(lambda x: x['compound']), bins=30, color='blue', alpha=0.5, label='Human')

plt.xlabel('Sentiment Score (Compound)')

plt.ylabel('Frequency')

plt.title('Sentiment Analysis for Human-written Abstracts')

plt.legend()

​

plt.subplot(1, 2, 2)

plt.hist(ai\_data['sentiment'].apply(lambda x: x['compound']), bins=30, color='red', alpha=0.5, label='AI')

plt.xlabel('Sentiment Score (Compound)')

plt.ylabel('Frequency')

plt.title('Sentiment Analysis for AI-generated Abstracts')

plt.legend()

plt.show()

C:\Anaconda3\lib\site-packages\ipykernel\_launcher.py:4: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.

Try using .loc[row\_indexer,col\_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html*#indexing-view-versus-copy*

after removing the cwd from sys.path.

C:\Anaconda3\lib\site-packages\ipykernel\_launcher.py:5: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.

Try using .loc[row\_indexer,col\_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html*#indexing-view-versus-copy*

*# N-gram Analysis to understand word combinations in abstracts.*

def extract\_ngrams(text, n):

tokens = nltk.word\_tokenize(text)

n\_grams = list(ngrams(tokens, n))

return n\_grams

​

human\_bigrams = human\_data['abstract'].apply(lambda x: extract\_ngrams(x, 2)).sum()

human\_bigram\_counts = Counter(human\_bigrams)

​

ai\_bigrams = ai\_data['abstract'].apply(lambda x: extract\_ngrams(x, 2)).sum()

ai\_bigram\_counts = Counter(ai\_bigrams)

*# Now, let's prepare our data for machine learning models.*

train\_data, test\_data, train\_labels, test\_labels = train\_test\_split(data['abstract'], data['is\_ai\_generated'], test\_size=0.2, random\_state=42)

*# Text Preprocessing to prepare text data for model training.*

def preprocess\_text(text):

tokens = word\_tokenize(text)

tokens = [token.lower() for token in tokens]

stop\_words = set(stopwords.words('english'))

tokens = [token for token in tokens if token not in stop\_words]

lemmatizer = WordNetLemmatizer()

tokens = [lemmatizer.lemmatize(token) for token in tokens]

preprocessed\_text = ' '.join(tokens)

return preprocessed\_text

​

train\_data = train\_data.apply(preprocess\_text)

test\_data = test\_data.apply(preprocess\_text)

*# Text Vectorization using TF-IDF to convert text data into numerical format.*

tfidf\_vectorizer = TfidfVectorizer(max\_features=4500)

tfidf\_vectorizer.fit(train\_data)

train\_vectors = tfidf\_vectorizer.transform(train\_data)

test\_vectors = tfidf\_vectorizer.transform(test\_data)

*# Now, let's train various machine learning models and evaluate their performances.*

​

*# Logistic Regression*

logistic\_model = LogisticRegression(random\_state=42)

logistic\_model.fit(train\_vectors, train\_labels)

logistic\_predictions = logistic\_model.predict(test\_vectors)

*# Evaluate Logistic Regression model*

accuracy\_logistic = accuracy\_score(test\_labels, logistic\_predictions)

print("Logistic Regression Accuracy:", accuracy\_logistic)

print("Classification Report:")

print(classification\_report(test\_labels, logistic\_predictions))

Logistic Regression Accuracy: 0.98150431566

Classification Report:

precision recall f1-score support

0 0.98 0.99 0.98 438

1 0.99 0.97 0.98 373

avg / total 0.98 0.98 0.98 811

*# Multinomial Naive Bayes*

nb\_model = MultinomialNB()

nb\_model.fit(train\_vectors, train\_labels)

nb\_predictions = nb\_model.predict(test\_vectors)

​

*# Evaluate Naive Bayes model*

accuracy\_nb = accuracy\_score(test\_labels, nb\_predictions)

print("Multinomial Naive Bayes Accuracy:", accuracy\_nb)

print("Classification Report:")

print(classification\_report(test\_labels, nb\_predictions))

Multinomial Naive Bayes Accuracy: 0.96794081381

Classification Report:

precision recall f1-score support

0 0.99 0.95 0.97 438

1 0.94 0.99 0.97 373

avg / total 0.97 0.97 0.97 811

*# Support Vector Machines*

svm\_model = SVC(random\_state=42)

svm\_model.fit(train\_vectors, train\_labels)

svm\_predictions = svm\_model.predict(test\_vectors)

​

*# Evaluate SVM model*

accuracy\_svm = accuracy\_score(test\_labels, svm\_predictions)

print("Support Vector Machines Accuracy:", accuracy\_svm)

print("Classification Report:")

print(classification\_report(test\_labels, svm\_predictions))

Support Vector Machines Accuracy: 0.540073982737

Classification Report:

precision recall f1-score support

0 0.54 1.00 0.70 438

1 0.00 0.00 0.00 373

avg / total 0.29 0.54 0.38 811

C:\Anaconda3\lib\site-packages\sklearn\metrics\classification.py:1113: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples.

'precision', 'predicted', average, warn\_for)

*# Decision Tree*

tree\_model = DecisionTreeClassifier(random\_state=42)

tree\_model.fit(train\_vectors, train\_labels)

tree\_predictions = tree\_model.predict(test\_vectors)

​

*# Evaluate Decision Tree model*

accuracy\_tree = accuracy\_score(test\_labels, tree\_predictions)

print("Decision Tree Accuracy:", accuracy\_tree)

print("Classification Report:")

print(classification\_report(test\_labels, tree\_predictions))

Decision Tree Accuracy: 0.958076448829

Classification Report:

precision recall f1-score support

0 0.96 0.96 0.96 438

1 0.95 0.95 0.95 373

avg / total 0.96 0.96 0.96 811

*# Random Forest*

rf\_model = RandomForestClassifier(random\_state=42)

rf\_model.fit(train\_vectors, train\_labels)

rf\_predictions = rf\_model.predict(test\_vectors)

​

*# Evaluate Random Forest model*

accuracy\_rf = accuracy\_score(test\_labels, rf\_predictions)

print("Random Forest Accuracy:", accuracy\_rf)

print("Classification Report:")

print(classification\_report(test\_labels, rf\_predictions))

Random Forest Accuracy: 0.976572133169

Classification Report:

precision recall f1-score support

0 0.98 0.98 0.98 438

1 0.97 0.98 0.97 373

avg / total 0.98 0.98 0.98 811

*# Feature Importance Analysis to identify the most influential features for our models.*

feature\_names = tfidf\_vectorizer.get\_feature\_names()

coefficients\_df = pd.DataFrame({'Feature': feature\_names, 'Coefficient': logistic\_model.coef\_[0]})

sorted\_coefficients = coefficients\_df.sort\_values(by='Coefficient', ascending=False)

top\_features = sorted\_coefficients.head(10)

p

*# Visualizing the top features for AI-generated and human-written abstracts.*

ai\_features = coefficients\_df[coefficients\_df['Coefficient'] > 0].nlargest(10, 'Coefficient')

human\_features = coefficients\_df[coefficients\_df['Coefficient'] < 0].nsmallest(10, 'Coefficient')

​

plt.figure(figsize=(10, 5))

sns.barplot(ai\_features['Feature'], ai\_features['Coefficient'], color='blue')

plt.xlabel('Coefficient Value')

plt.title('Top Features for AI-generated Abstracts')

plt.show()

​

plt.figure(figsize=(10, 5))

sns.barplot(human\_features['Feature'], human\_features['Coefficient'], color='red')

plt.xlabel('Coefficient Value')

plt.title('Top Features for Human-written Abstracts')

plt.show()

*# List of model names and their accuracies*

model\_names = ['Logistic Regression', 'Naive Bayes', 'Support Vector Machines', 'Decision Tree', 'Random Forest']

accuracies = [98.1, 96.7, 54.0, 95.8, 97.6] *# Accuracy values in percentage*

​

*# Create numerical x-axis positions for the bars*

x\_pos = range(len(model\_names))

​

*# Plotting the accuracies*

plt.figure(figsize=(12, 10))

plt.bar(x\_pos, accuracies, color='skyblue')

plt.xlabel('Machine Learning Models')

plt.ylabel('Accuracy (%)')

plt.title('Comparison of Model Accuracies')

plt.xticks(x\_pos, model\_names, rotation=45) *# Set x-axis tick positions and labels*

plt.ylim(0, 100) *# Set y-axis limits to percentage scale*

plt.show()