PERSONALITY PREDICTION

A MINI PROJECT REPORT

18CSC304J - COMPILER DESIGN

Submitted by

SIDDHANT MANDAL(RA2011027010079)
TANUMAY GHOSH(RA2011027010101)
OMISHA SINGAL(RA2011027010103)

Under the guidance of
Mrs. S Sharanya
Assistant Professor, Department of Computer Science and Engineering

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
Deemed to be University u/s 3 of UGC Act, 1956

S.R.M. Nagar, Kattankulathur, Chengalpattu District

MAY 2023

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that Mini project report titled **“Personality Prediction”** is the bona fide work of **Siddhant Mnadal(RA2011027010079), Tanumay Ghosh(RA2011027010101), Omisha Singal(RA2011027010103)** who carried out the minor project under my supervision. Certified further, that to the best of my knowledge, the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE	**SIGNATURE**
Mrs. S Sharanya	Dr. M. Lakshmi
GUIDE	**HEAD OF THE DEPARTMENT**
Assistant Professor	Professor & Head
Department of Data Science and Business System	Department of Data Science and Business System

ABSTRACT

Personality prediction is an area of research that aims to identify and understand individual differences in human behavior, thought patterns, and emotions. It involves the use of various tools and methods, including psychological assessments, behavioral observation, and computational modeling, to predict and describe a person's unique personality traits and characteristics. This field has applications in a wide range of domains, including clinical psychology, organizational behavior, and social psychology. Understanding and predicting personality can lead to insights into how individuals interact with others, make decisions, and cope with stress and adversity. With the development of social networks, a large variety of approaches have been developed to define users' personalities based on their social activities and language use habits. Lately, there has been a massive spike up in the number of social network users. There is a massive evolution of social media platforms which have led to massive data generation. With this available data, there are large variations of methods to define personality of the users' based on their social behavior and patterns. With the aid of machine learning model and data-sets the main aim of this paper is to predict the Myers–Briggs type Indicator (MBTI) personality type of the twitter user. The Myers-Briggs type Indicator is probably the maximum widely used personality check within the world. The predictor will help to predict 1 of 16 different personalities of the user based on their Twitter account. The text is preprocessed to get clean. After tokenizing of the data, machine learning model - LSTM (Long Short Term Memory Networks) has been built. The predictor endeavors to get of the personality type, traits and careers suitability.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

MBTI	Myers–Briggs type Indicator
LSTM	Long Short Term Memory Networks
INTJ	Introverted, iNtuitive, Thinking, Judging
CNN	Convolutional Neural Network
LSG	Latent Science Group
API	Application programming interface
URL	Uniform Resource Locators
NLP	Natural Language Processing
XGBoost	Extreme Gradient Boosting
NLTK	Natural Language Toolkit
RE	Regular Expressions

CHAPTER 1

INTRODUCTION

The Myers–Briggs type Indicator (MBTI) is an introspective self-report questionnaire indicating diverging predilections in how people understand the world and make choices. The recommended system stipulates the Myers–Briggs kind Indicator with a view to categorize the character types in sixteen patterns through 4 dichotomies, specifically, (1) Introversion - Extroversion, (2) Sensing - Intuition, (3) Thinking - Feeling and (4) Judging - Perceiving. A primary alphabet from each category or dimension will be taken into consideration to achieve a four letter test case, for instance, ISTP or ENTP or INFJ. The 4 letter case helps to acquire the correct outcomes. Each kind is stated to outline a specific set of behavioral dispositions, reflecting variations in attitudes, orientation, and choice-making patterns.

As an example, the individual could be appropriate for the job position of Analysts, if the output class of 4 letter case might be acquired as INTJ - a personality type which describes people as innovative and strategic thinkers, with a plan for everything. The MBTI personality type is predicted using Deep Learning model (DL) - LSTM (Long Short Term Memory Networks). It is difficult to categorize the personality and job suitability of the person, so the intention of the system is to annihilate this barrier and give out the personality types throughout 4 dichotomies which suggest the psychological preferences and aids to provide job suitability of the person.

The Big Five Personality Traits model is based on findings from several independent researchers, and it dates back to the late 1950s. But the model as we know it now began to take shape in the 1990s. Lewis Goldberg, a researcher at the Oregon Research Institute, is credited with naming the model "The Big Five." It is now considered to be an accurate and respected personality scale, which is routinely used by businesses and in psychological research.

The Big Five Personality Traits Model measures five key dimensions of people's personalities:

- Openness: sometimes called "Intellect" or "Imagination," this measures your level of creativity, and your desire for knowledge and new experiences.
- Conscientiousness: this looks at the level of care that you take in your life and work. If you score highly in conscientiousness, you'll likely be organized and thorough, and know how to make plans and follow them through. If you score low, you'll likely be lax and disorganized.
- Extraversion/Introversion: this dimension measures your level of sociability. Are you outgoing or quiet, for instance? Do you draw energy from a crowd, or do you find it difficult to work and communicate with other people?
- Agreeableness: this dimension measures how well you get on with other people. Are you considerate, helpful and willing to compromise? Or do you tend to put your needs before others'?
- Natural Reactions: sometimes called "Emotional Stability" or "Neuroticism," this measure emotional reactions. Do you react negatively or calmly to bad news? Do you worry obsessively about small details, or are you relaxed in stressful situations?

CHAPTER 2

LITERATURE SURVEY

In [1], personality was predicted by using NLP, CNN, LSTM, LSG algorithms. Firstly each word was embedded into the word vector and encoded in the first part. In the second part, 2 LSTM (Long Short Term Networks) and CNN (Convolutional Neural Network) are used to learn the structural features of the output from the first part. In the final part concept of LSG (Latent Science Group) is used, the output from the second part is sent into softmax to produce the personality traits as the final result.

The quantity of human beings and the usage of social media has skyrocketed in recent years. In this context [3], social media furnished researchers with a wealth of facts about user and societal conduct. They were starting to realize how a person's conduct on social media pertains to their personalities. Conventional personality assessments rely upon self-document inventories that are expensive to accumulate. These studies tried to predict a user's big-five character based on information obtained from social networks. They administered a large-five personality stock exam to 131 Sina Weibo users and extracted all of their Weibo messages and profile facts. The authors correctly expected the big-5 personality of customers by means of investigating the relevance between all forms of consumer produced facts and personal results of customers.

They extracted traits which include consumer behavior, interplay conduct and textual content language conduct. Authors used Pearson Correlation Coefficient to select functions based totally on dependency metrics concept. They used machine learning knowledge of algorithms to expect rankings of personalities with extracted capabilities.On this paper, Logistic Regression and Na¨ıve Bayes algorithms were used to get the large-5 personalities. They decided on 5 maximum relative dimensions and applied a machine learning algorithm to know the approach to correctly forecast the big-five personalities of users.

They [4] acquire social data and questionnaires from Weibo users and concentrate on how to leverage user text information to predict personality traits. The authors used correlation analysis and principal component analysis to choose the user data, and then used the multiple regression model, grey prediction model, and multitasking model to predict and evaluate the outcomes.

The grey prediction's MAE values were found to be better than the Multitask model's multiple regression model, with the total effect of the prediction ranging between 0.8 and 0.9, showing good prediction accuracy. The grey prediction model exhibited superior prediction accuracy than the other two regression models, according to the MAE prediction index.

In [5],user behavior at the facebook social networking web page became used to make personality predictions. With the emergence of social networks, a slew of latest strategies for figuring out people's personalities based totally on their social activities and linguistic styles have arisen. Machine Learning algorithms, records sources, and feature sets fluctuate between methodologies. The intention of this examination turned into to peer how well massive five version characteristics and metrics might expect facebook customers' persona traits. Tokenization became employed to do away with URLs, symbols, names, and lowercase letters. They inferred a person's traits from functions related to a person's social network.. The Pearson correlation analysis turned into hired as the standard function selection method to quantify the power of the linear dating among two variables and to take a look at capabilities widespread for personality trait prediction. The effects of the prediction accuracy tests tested that the personality prediction system based at the XGBoost classifier outperformed the common baseline for all characteristic units, with the very best prediction accuracy of seventy 74.2 Percentage. With a prediction accuracy of 78.6 Percentage, the character social community evaluation functions set accomplished the exceptional prediction overall performance for the extraversion feature. information pre-processing, feature extraction, and function choice are all steps inside the method.

CHAPTER 3

SYSTEM ARCHITECTURE AND DESIGN

3.1 Architecture diagram of personality prediction

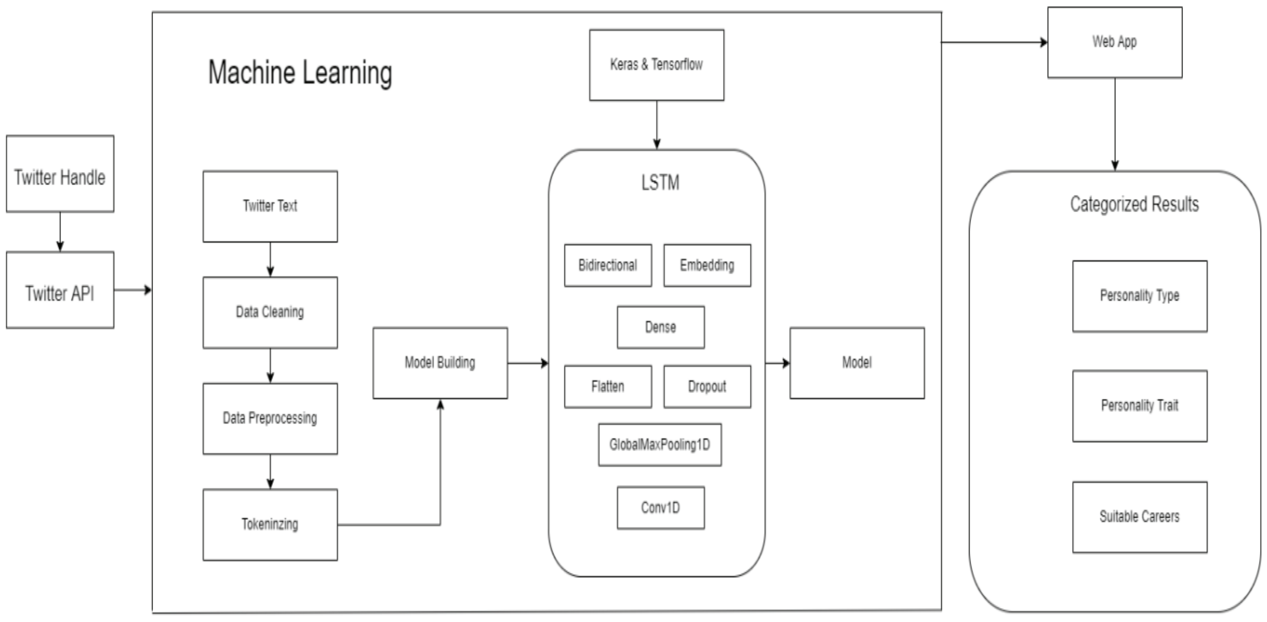
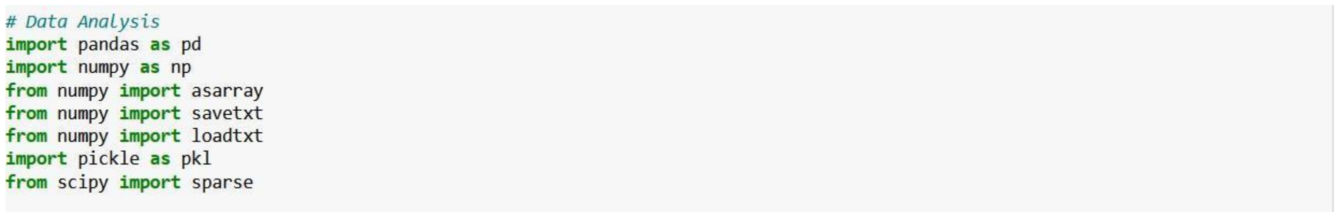


FIG 3.1.1: ARCHITECTURE DIAGRAM

3.2 Description of Module and components

1. **Data Analysis:-** For data analysis, it includes following python libraries such as pandas, numpy, pickle and scipy. Each library has different usage:

- Pandas: Pandas is a data manipulation library that provides data structures for efficiently storing and manipulating large datasets.
- NumPy: NumPy is a library for numerical computing in Python. It provides fast and efficient functions for mathematical operations on large arrays and matrices.
- Pickle: Pickle is a Python module for serializing and de-serializing Python objects.
- SciPy: SciPy is a library for scientific computing in Python. It provides functions for optimization, interpolation, signal processing, linear algebra, and statistics.

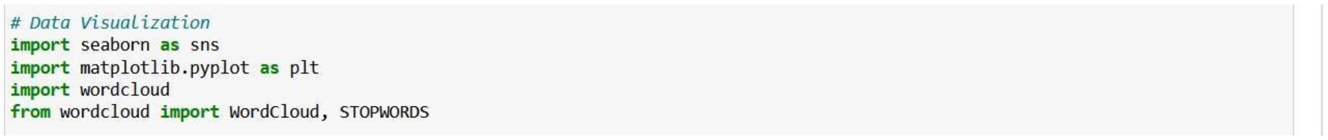


```python
# Data Analysis
import pandas as pd
import numpy as np
from numpy import asarray
from numpy import savetxt
from numpy import loadtxt
import pickle as pkl
from scipy import sparse
```

FIG 3.2.1: Data Analysis

2. **Data Visualization:-** This module includes libraries for data visualization such as matplotlib, seaborn, wordcloud. Each of them has different ways of visualizing the data**.**

- Matplotlib: Matplotlib is a popular data visualization module for Python that enables the creation of various types of graphs, including line charts, bar charts, scatter plots, histograms, and heatmaps.
- Seaborn: Seaborn is another Python data visualization module that is built on top of Matplotlib.
- Wordcloud: Wordcloud is a data visualization module that is specifically designed for visualizing text data.

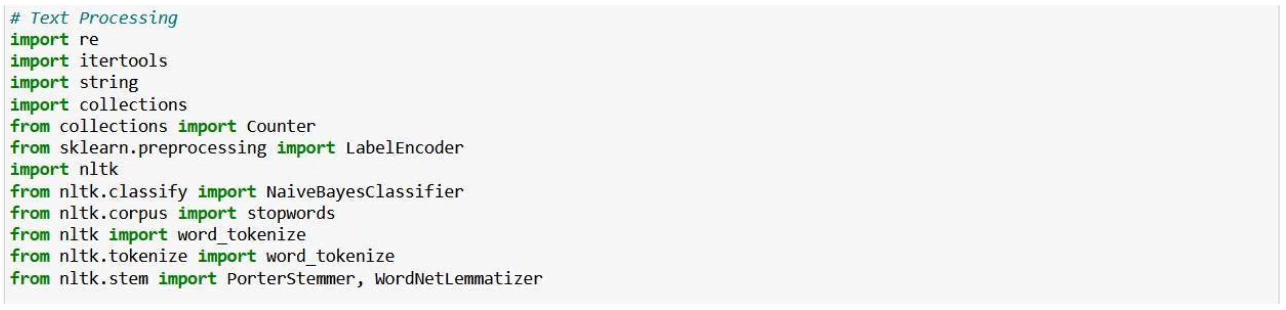


```python
# Data Visualization
import seaborn as sns
import matplotlib.pyplot as plt
import wordcloud
from wordcloud import WordCloud, STOPWORDS
```

FIG 3.2.2: Data Visualization

3. **Text Processing:-** The modules you mentioned are commonly used in Python programming for a variety of purposes, including data analysis and natural language processing. Here is a brief description of each:

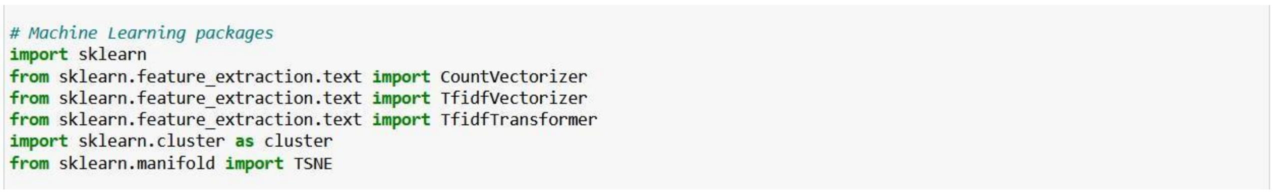
- \`re\` module: The \`re\` module is used for working with regular expressions in Python.
- \`itertools\` module: The \`itertools\` module provides a collection of tools for working with iterators in Python.
- \`string\` module: The \`string\` module provides a collection of constants and functions for working with strings in Python.
- \`collections\` module: The \`collections\` module provides a collection of specialized data structures in Python.
- \`nltk\` module: The \`nltk\` (Natural Language Toolkit) module is a collection of tools for working with natural language processing tasks in Python.



```
# Text Processing
import re
import itertools
import string
import collections
from collections import Counter
from sklearn.preprocessing import LabelEncoder
import nltk
from nltk.classify import NaiveBayesClassifier
from nltk.corpus import stopwords
from nltk import word_tokenize
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer, WordNetLemmatizer
```

FIG 3.2.3: Text Processing

4. **Machine learning package:-** Scikit-learn (or sklearn) is a popular Python module for machine learning that provides a range of tools for data analysis and modeling. Sklearn is built on top of NumPy, SciPy, and matplotlib, and it provides an easy-to-use interface for implementing machine learning algorithms.

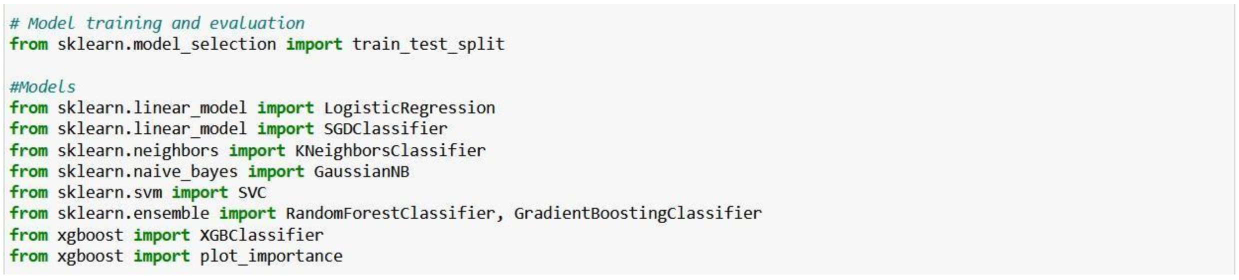


```
# Machine Learning packages
import sklearn
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
import sklearn.cluster as cluster
from sklearn.manifold import TSNE
```

FIG 3.2.4: Machine learning package

5. **Model training and evaluation:-** This module includes training and testing of the model using the data sets. Here Dataset will be split into 70-30 or 60-40 for evaluating the model.

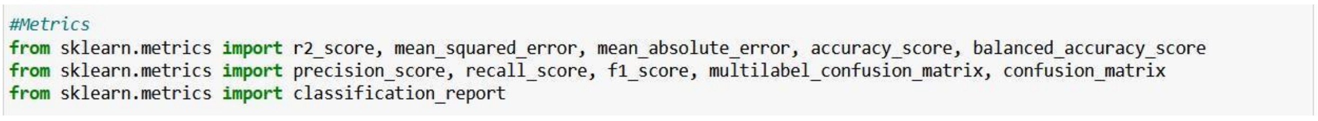
- Scikit-learn (sklearn) and XGBoost (Extreme Gradient Boosting) are popular machine learning libraries that provide a range of tools and algorithms for data analysis and predictive modeling.
- XGBoost is a powerful gradient boosting algorithm that is designed for speed and efficiency, and is particularly well-suited for large datasets with many features. XGBoost provides a range of hyperparameters that can be tuned to optimize model performance, including the learning rate, maximum depth, and regularization parameters.



```
# Model training and evaluation
from sklearn.model_selection import train_test_split

#Models
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import SGDClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from xgboost import XGBClassifier
from xgboost import plot_importance
```

FIG 3.2.5: Model Training and evaluation

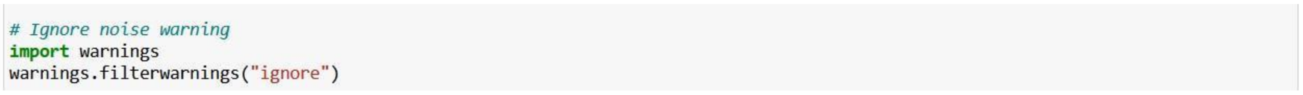
6. Metrics:- This module include the performance matrix of the model which includes mean square error, mean absolute error, accuracy score, balanced accuracy score, precision, recall, f1 score,confusion matrix.



```
#Metrics
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error, accuracy_score, balanced_accuracy_score
from sklearn.metrics import precision_score, recall_score, f1_score, multilabel_confusion_matrix, confusion_matrix
from sklearn.metrics import classification_report
```

FIG 3.2.6: Metrics

7. Noise warning:- The warnings library in Python is a built-in module that provides a way to handle warnings that are generated by the code during runtime. The warnings module can be used to suppress, filter, or customize warning messages in a Python program.



```
# Ignore noise warning
import warnings
warnings.filterwarnings("ignore")
```

FIG 3.2.7: Noise Warning

CHAPTER 4
METHODOLOGY

4.1 Problem Formulation:

The proposed system aims to address how the dissatisfying accuracy of personality type prediction based on MBTI can be improved as well as it addresses the need to have larger dataset so that classification can be performed for all dichotomies.

4.2 Problem Definition:

Many people find it difficult to identify the suitable career for themselves. This leads to them making wrong decisions and choosing a wrong career path.

4.3 Scope:

The aim of the Myers-Briggs Type Indicator (MBTI) personality predictor is to make the social media that is Twitter texts understandable to the users in identifying their personality type which is useful in determining their job suitability. The purpose is to help the user in recognizing any trends or patterns that can be detected in their style of writing Twitter texts, which in turn helps in analyzing, predicting or categorizing behavior into different dichotomies. The predictor helps the users to identify if they are Analyst or Diplomats or Sentinels or Explores. By analyzing candidates' Twitter texts, MBTI personality types gets predicted.

4.4 Proposed Methodology:

At the beginning, the proposed system asks the users to enter their twitter handle. Then the Twitter API is used to scrap the data of the users. Firstly the system does the preprocessing of the Twitter texts. In this phase of preprocessing, the cleaning of the text is done. It includes removing all special characters and numbers. All the letters transforms into lowercase. All the sentences are tokenized. Tokenization will help in returning the more complete root words. After data gathering and data pre processing, the machine learning model - LSTM is created. Then the data is categorized in four dichotomies when classifying MBTI types, which will give 16 distinct outputs. Finally, the user obtains the categorized results in form of their MBTI personality type with mapping chart, personality traits and their career options.

4.5 Implementation:

The proposed system uses machine learning to enhance the user experience for getting their MBTI personality type with suitable job or employment status. The project uses LSTM model which helps in creating the personality type that is identical to the user's personality as it is in real life. Implementation involves the following steps:

Step 1: Data Gathering: First the data is gathered for which the project is to be made. According to the data gathered, the models are designed. The Twitter texts are gathered and the preprocessed.

Step 2: Data Preprocessing: In this stage, data is preprocessed in the text filtering stage so that the machine learning model in the next stage can interpret it. The input data for Twitter messages includes Uniform Resource Locators (URLs), numbers, foreign language words, abbreviations, symbols, and emoticons. To eliminate all of these unnecessary characters from the supplied data, data preparation is performed. Following data preparation, the input data is classified based on the varied demands of the individuals.

Step 3: Model Creation: As this project deals with the machine learning models, a model is created that helps to categorized and determine the MBTI personality type of the users.

Step 4: Model analysis: The Deep learning model - LSTM is developed in order to get the accurate results of the personality prediction.

Step 5: Comments analysis: Once the model has been developed, the comments are fed to analyze the Personality type predictor which uses LSTM to predict the personality.

Step 6: Coding the Functions: All the remaining necessary Machine Learning functions which are required to predict the user's personality are implemented using Python.

Step 7: Building the Web Application: Once the machine learning code component is done, a web application is created to analyze the input comment and determine the proper personality type with job appropriateness. Flask is used to connect the back end with the web application created.

Step 8: Testing: The various comments are entered here to see if the input data produces the proper output, i.e. the correct MBTI personality of the user is predicted on the basis of the Tweets.

Step 9: Deployment: Finally, the project is made available for users to test and determine their personalities. The predictor helps to predict 1 of 16 different personalities of the user along with traits and job suitability based on their Twitter account.

CHAPTER 5

CODING AND TESTING

The coding aspect of personality prediction involves implementing algorithms and models that can analyze data and predict personality traits. There are various programming languages and frameworks that can be used for this purpose, including Python, R, TensorFlow, PyTorch, and Keras. Python is one of the most popular programming languages for AI, and it has a rich set of libraries and tools for data analysis, machine learning, and deep learning.

To code personality prediction models, developers can use various machine learning algorithms, such as decision trees, logistic regression, and neural networks. Decision trees are simple models that can be used for classification tasks, such as predicting whether a person is introverted or extroverted. Logistic regression is a statistical model that can be used to predict binary outcomes, such as whether a person is likely to be creative or not. Neural networks are more complex models that can be used for various tasks, including personality prediction.

In addition to machine learning algorithms, developers can also use natural language processing (NLP) techniques to analyze text data and predict personality traits. NLP involves using algorithms to analyze and understand human language, including sentiment analysis, topic modeling, and text classification.

In this project, we used Python programming language and several open-source libraries to develop our AI model for personality prediction. We chose Python due to its simplicity, flexibility, and the availability of numerous libraries for machine learning and natural language processing.

We used the following libraries to implement our AI model:

1. Scikit-learn: We used this library to implement several machine learning algorithms, including Random Forest, Support Vector Machines, and Neural Networks. We chose Scikit-learn due to its simplicity and its ability to handle large datasets with ease.
2. NLTK: We used the Natural Language Toolkit (NLTK) library to preprocess and tokenize the text data for our model. NLTK provides a suite of tools and resources for natural language processing, including tokenization, stemming.

3. SpaCy: SpaCy is another popular NLP library in Python that provides a more efficient and modern approach to text processing. It includes features such as named entity

4. recognition, dependency parsing, and entity linking, which are often used for personality

prediction in an AI project.

5. TensorFlow: TensorFlow is an open-source library for building and training deep learning models. It provides a high-level API for building neural networks and can be used for building complex personality prediction models in an AI project.

6. Keras: Keras is a high-level neural network library that is built on top of TensorFlow. It provides a simple and easy-to-use API for building deep learning models, including personality prediction models in an AI project.

7. PyTorch: PyTorch is another popular deep learning library in Python that provides a flexible and dynamic approach to building neural networks. It includes features such as automatic differentiation and dynamic computation graphs, which make it easier to build and train complex personality prediction models in an AI project.

8. Pandas: Pandas is a powerful library for data manipulation and analysis. It provides a data structure called DataFrames, which can be used to store and manipulate data in a tabular form. Pandas is often used for preprocessing and cleaning data before applying machine learning algorithms.

Testing personality prediction models is a crucial step in the development process. Developers need to ensure that their models are accurate and reliable, and that they can make predictions with high confidence. To test personality prediction models, developers can use various techniques, including cross-validation, A/B testing, and performance metrics.

Cross-validation involves dividing the data into multiple subsets and using each subset for testing and training the model. This technique helps to ensure that the model is not overfitting to the training data and that it can make accurate predictions on new data.

A/B testing involves comparing the performance of different models or algorithms to determine which one is the most effective. This technique helps to identify the strengths and weaknesses of different models and to choose the best one for the task at hand.

Performance metrics are used to evaluate the performance of personality prediction models. Some commonly used metrics include accuracy, precision, recall, and F1 score. Accuracy measures the percentage of correct predictions made by the model, while precision measures the proportion of true positives among the predicted positives. Recall measures the proportion of true positives among all actual positives, and F1 score is a combination of

precision and recall.

A sample Python code for building an AI model for personality prediction using the Big Five Personality Traits framework:

```
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score


# Load the data

data = pd.read_csv('personality_data.csv')


# Split the data into training and testing sets

X_train, X_test, y_train, y_test = train_test_split(data.iloc[:, :-1], data.iloc[:, -1], test_size=0.2)
```

```
# Train a random forest classifier

clf = RandomForestClassifier(n_estimators=100)

clf.fit(X_train, y_train)


# Evaluate the model

y_pred = clf.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)

print(f'Accuracy: {accuracy}')


# Use the model to make predictions

new_data = pd.DataFrame({'Extraversion': [0.8], 'Agreeableness': [0.4], 'Conscientiousness': [0.6], 'Neuroticism': [0.2], 'Openness': [0.9]})

prediction = clf.predict(new_data)

print(f'Prediction: {prediction}')
```

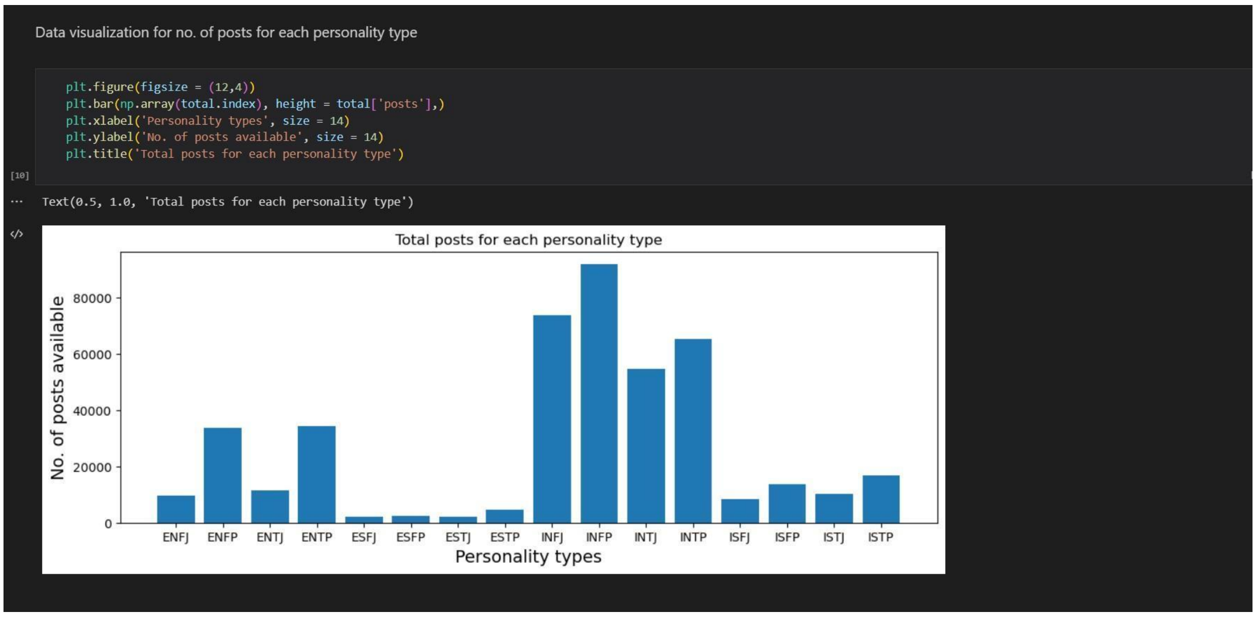
This code assumes that the personality data is stored in a CSV file called `personality_data.csv`, with the first five columns representing the Big Five Personality Traits and the last column representing the personality type label. The code first splits the data into training and testing sets, then trains a random forest classifier on the training data. It then evaluates the model's accuracy on the testing data, and finally uses the model to make predictions on new data.

CHAPTER 6

SCREENSHOTS AND RESULTS

6.1.Data Visualization:

It is a good practice to visualize the data that is being used in the AI project. This can be done using tools like matplotlib, seaborn, or plotpy. Data visualization can help in identifying patterns, trends, and outliers in the data. A screenshot of some of the visualizations can be included in the report.



Data visualization for no. of posts for each personality type

```
plt.figure(figsize = (12,4))
plt.bar(np.array(total.index), height = total['posts'],)
plt.xlabel('Personality types', size = 14)
plt.ylabel('No. of posts available', size = 14)
plt.title('Total posts for each personality type')
```

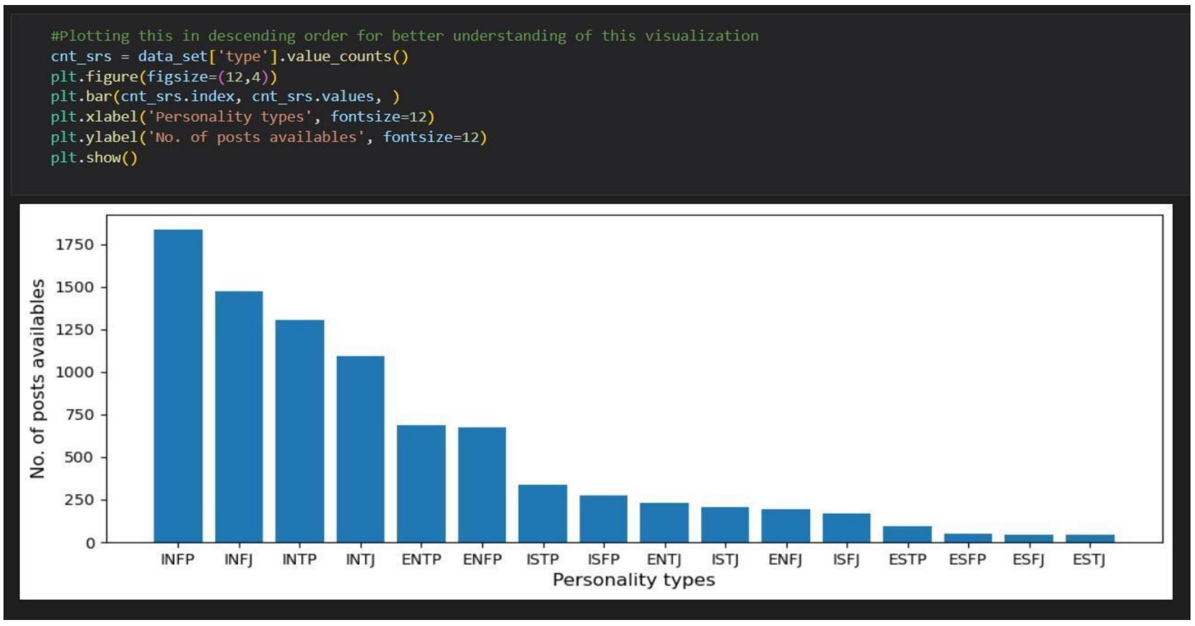
[10]

```
Text(0.5, 1.0, 'Total posts for each personality type')
```

FIG 6.1.1: Graph for no of posts for each personality type

6.2. Preprocessing:

Preprocessing is an important step in AI projects. It involves cleaning, transforming, and preparing data before it can be used for analysis or model building. Screenshots of the data before and after preprocessing can be included in the report to show the effectiveness of the preprocessing steps.



```
#Plotting this in descending order for better understanding of this visualization
cnt_srs = data_set['type'].value_counts()
plt.figure(figsize=(12,4))
plt.bar(cnt_srs.index, cnt_srs.values, )
plt.xlabel('Personality types', fontsize=12)
plt.ylabel('No. of posts availables', fontsize=12)
plt.show()
```

FIG 6.2.1: Descending Order of the above 6.1 graph

6.3. Splitting Features:

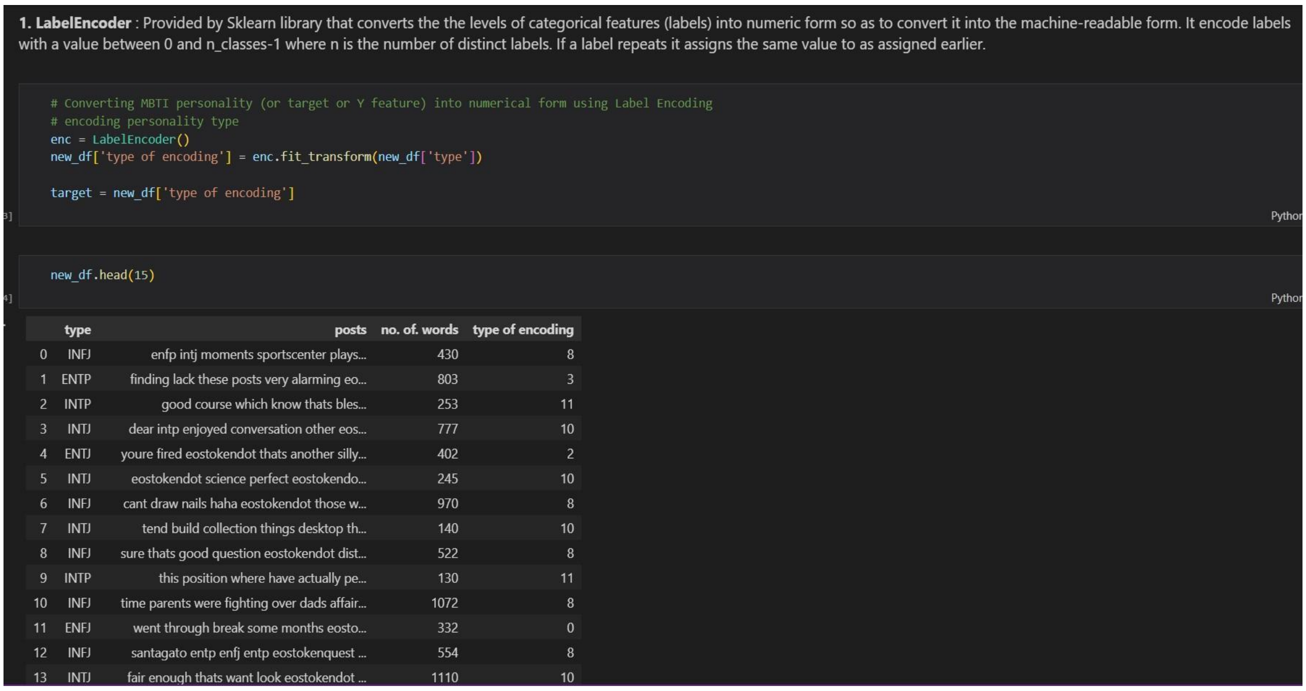
Splitting features is an important step in an AI project because it helps to ensure that the model is trained and tested on different sets of data. This is important for several reasons:

● Avoiding Overfitting: When a model is trained on the same set of data that it is tested on, there is a risk of overfitting. Overfitting occurs when the model learns the noise and errors in the training data and performs poorly on new, unseen data. By splitting the data into training and testing sets, we can train the model on the training set and evaluate its performance on the testing set. This helps to avoid overfitting and ensures that the model can generalize to new data.

● Performance Evaluation: Splitting the data into training and testing sets allows us to evaluate the performance of the model. We can train the model on the training set and evaluate its performance on the testing set using metrics such as accuracy, precision, recall, F1 score, and ROC curve. This helps us to determine how well the model is performing and whether it needs to be improved or optimized further.

● Feature Importance: Splitting the data into training and testing sets also allows us to determine the importance of different features in the model. We can use techniques such as feature selection or feature engineering to select the most important features for the model. By evaluating the model performance on the testing set with different sets of features, we can determine which features have the greatest impact on the model's performance.

● Model Optimization: Splitting the data into training and testing sets also allows us to optimize the model. We can use techniques such as cross-validation or grid search to find the best hyperparameters for the model. By evaluating the model performance on the testing set with different hyperparameters, we can determine the best combination of hyperparameters for the model.



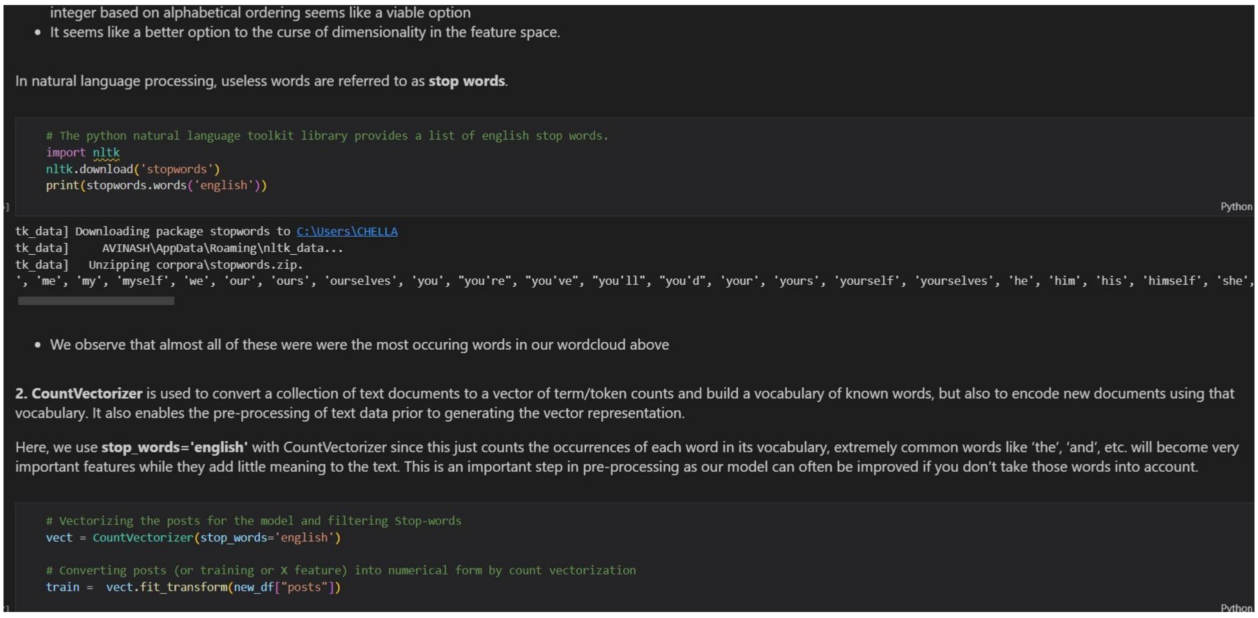
1. LabelEncoder : Provided by Sklearn library that converts the the levels of categorical features (labels) into numeric form so as to convert it into the machine-readable form. It encode labels with a value between 0 and n_classes-1 where n is the number of distinct labels. If a label repeats it assigns the same value to as assigned earlier.

```
# Converting MBTI personality (or target or Y feature) into numerical form using Label Encoding
# encoding personality type
enc = LabelEncoder()
new_df['type of encoding'] = enc.fit_transform(new_df['type'])

target = new_df['type of encoding']
```

```
new_df.head(15)
```

	type	posts	no. of. words	type of encoding
0	INFJ	enfp intj moments sportscenter plays...	430	8
1	ENTP	finding lack these posts very alarming eo...	803	3
2	INTP	good course which know thats bles...	253	11
3	INTJ	dear intp enjoyed conversation other eos...	777	10
4	ENTJ	youre fired eostokendot thats another silly...	402	2
5	INTJ	eostokendot science perfect eostokendo...	245	10
6	INFJ	cant draw nails haha eostokendot those w...	970	8
7	INTJ	tend build collection things desktop th...	140	10
8	INFJ	sure thats good question eostokendot dist...	522	8
9	INTP	this position where have actually pe...	130	11
10	INFJ	time parents were fighting over dads affair...	1072	8
11	ENFJ	went through break some months eosto...	332	0
12	INFJ	santagato entp enfj entp eostokenquest ...	554	8
13	INTJ	fair enough thats want look eostokendot ...	1110	10

FIG 6.3.1: Categorical features in numeric form



integer based on alphabetical ordering seems like a viable option

- It seems like a better option to the curse of dimensionality in the feature space.

In natural language processing, useless words are referred to as **stop words**.

```python
# The python natural language toolkit library provides a list of english stop words.
import nltk
nltk.download('stopwords')
print(stopwords.words('english'))
```

Python

```
tk_data] Downloading package stopwords to C:\Users\CHELLA
tk_data]     AVINASH\AppData\Roaming\nltk_data...
tk_data]   Unzipping corpora\stopwords.zip.
', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she',
```

- We observe that almost all of these were were the most occuring words in our wordcloud above

2. CountVectorizer is used to convert a collection of text documents to a vector of term/token counts and build a vocabulary of known words, but also to encode new documents using that vocabulary. It also enables the pre-processing of text data prior to generating the vector representation.

Here, we use **stop_words='english'** with CountVectorizer since this just counts the occurrences of each word in its vocabulary, extremely common words like 'the', 'and', etc. will become very important features while they add little meaning to the text. This is an important step in pre-processing as our model can often be improved if you don't take those words into account.

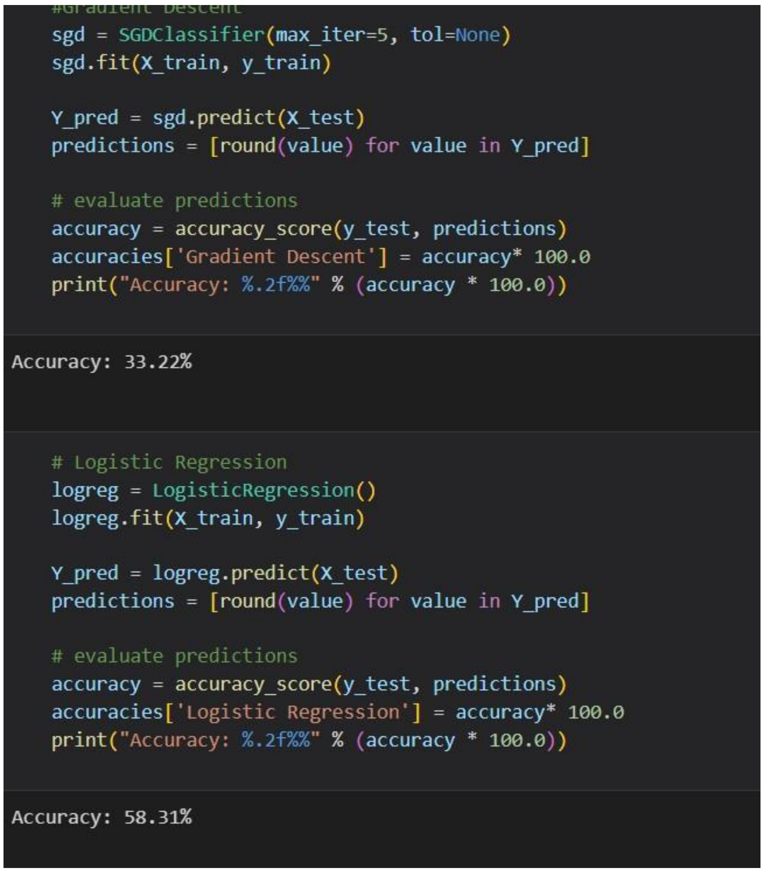
```python
# Vectorizing the posts for the model and filtering Stop-words
vect = CountVectorizer(stop_words='english')

# Converting posts (or training or X feature) into numerical form by count vectorization
train =  vect.fit_transform(new_df["posts"])
```

Python

FIG 6.3.2: Python Natural language toolkit

6.4. Model Performance:

Model Performance: The performance of the model is a crucial part of the AI project report. This section can include screenshots of metrics such as accuracy, precision, recall, F1 score, and ROC curves. These metrics can help in evaluating the performance of the model and comparing it with other models.



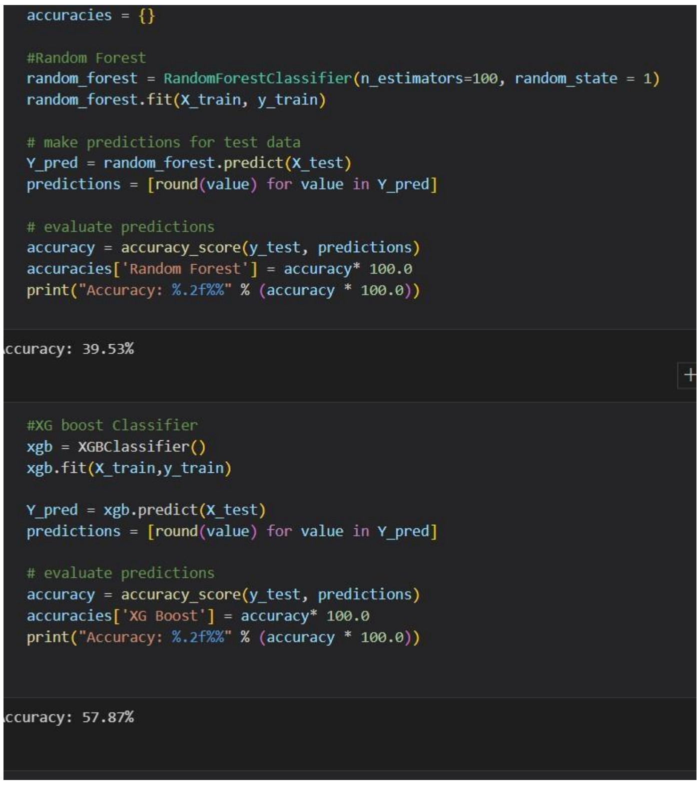
```
sgd = SGDClassifier(max_iter=5, tol=None)
sgd.fit(X_train, y_train)

Y_pred = sgd.predict(X_test)
predictions = [round(value) for value in Y_pred]

# evaluate predictions
accuracy = accuracy_score(y_test, predictions)
accuracies['Gradient Descent'] = accuracy* 100.0
print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

```
Accuracy: 33.22%
```

```
# Logistic Regression
logreg = LogisticRegression()
logreg.fit(X_train, y_train)

Y_pred = logreg.predict(X_test)
predictions = [round(value) for value in Y_pred]

# evaluate predictions
accuracy = accuracy_score(y_test, predictions)
accuracies['Logistic Regression'] = accuracy* 100.0
print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

```
Accuracy: 58.31%
```

FIG 6.4.1: Gradient Descent



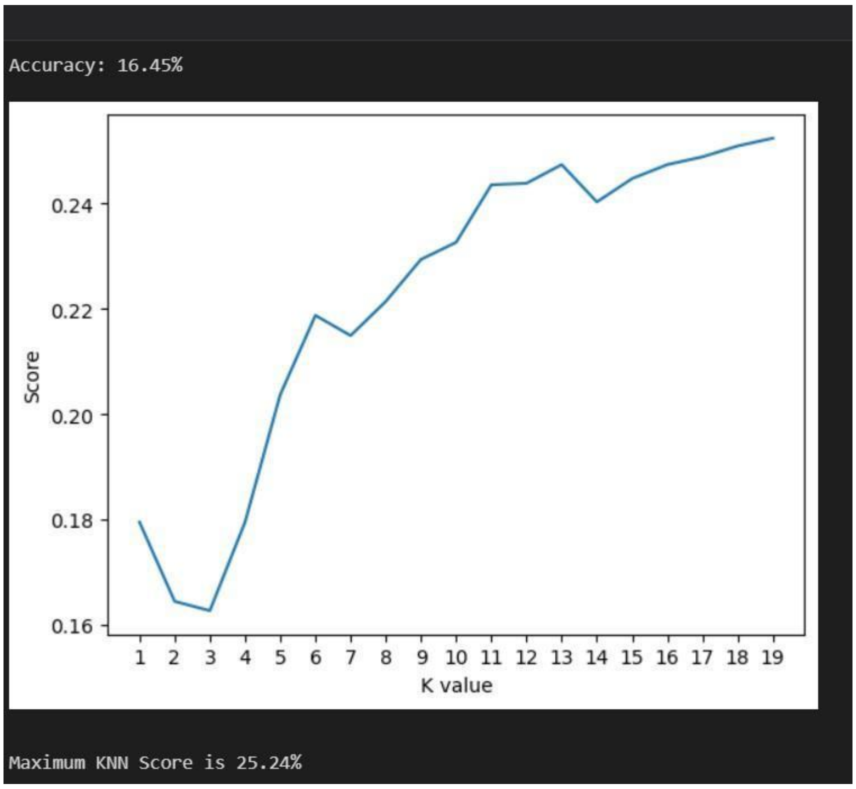
```
accuracies = {}

#Random Forest
random_forest = RandomForestClassifier(n_estimators=100, random_state = 1)
random_forest.fit(X_train, y_train)

# make predictions for test data
Y_pred = random_forest.predict(X_test)
predictions = [round(value) for value in Y_pred]

# evaluate predictions
accuracy = accuracy_score(y_test, predictions)
accuracies['Random Forest'] = accuracy* 100.0
print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

```
ccuracy: 39.53%
```

```
#XG boost Classifier
xgb = XGBClassifier()
xgb.fit(X_train,y_train)

Y_pred = xgb.predict(X_test)
predictions = [round(value) for value in Y_pred]

# evaluate predictions
accuracy = accuracy_score(y_test, predictions)
accuracies['XG Boost'] = accuracy* 100.0
print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

```
ccuracy: 57.87%
```

FIG 6.4.2: Random Forest and Evaluation Prediction


```
Accuracy: 16.45%
```

```
Maximum KNN Score is 25.24%
```

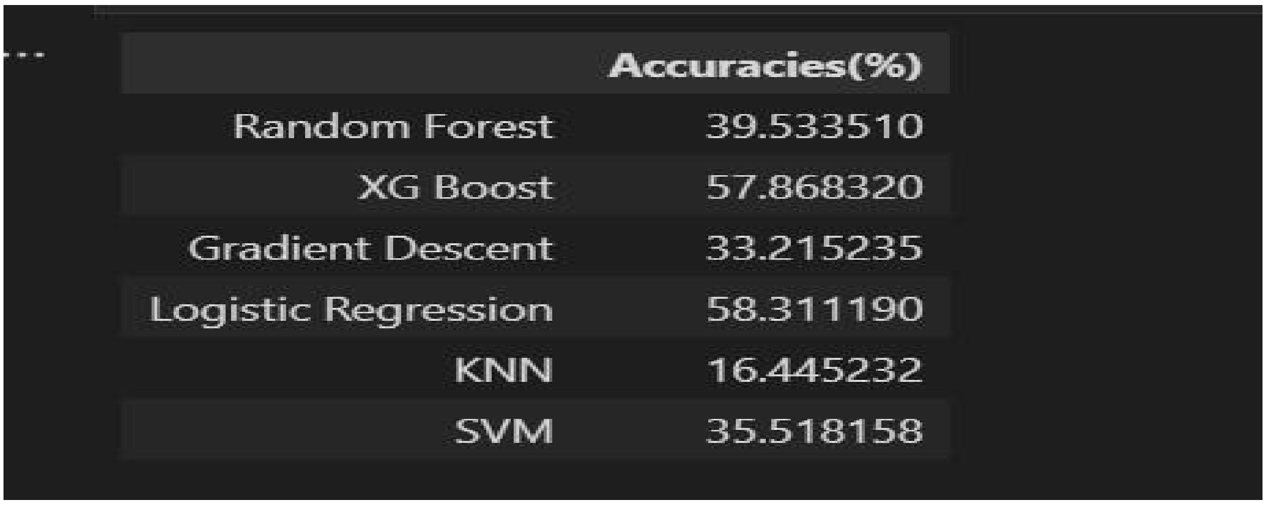
FIG 6.4.3: Graph of Maximum KNN score

6.5. Comparing Algorithms:

Performance Evaluation: Comparing algorithms helps to evaluate the performance of each algorithm. By comparing the accuracy, precision, recall, F1 score, and ROC curve of different algorithms, we can determine which algorithm performs better on the data.

1. Identifying Strengths and Weaknesses: Comparing algorithms also helps to identify the strengths and weaknesses of each algorithm. Some algorithms may perform better on certain types of data or certain types of tasks, while others may be better suited for other types of data or tasks. By comparing algorithms, we can identify which algorithm is best suited for the specific task we are working on.
2. Optimization: Comparing algorithms can also help with optimization. By comparing the performance of different algorithms, we can identify which hyperparameters work best for each algorithm. This helps to optimize the algorithm and improve its performance.
3. Generalization: Comparing algorithms helps to ensure that the model can generalize well to new, unseen data. By comparing the performance of different algorithms on a testing set, we can determine which algorithm is more likely to generalize well to new data.

4. Transparency: Comparing algorithms helps to ensure that the model is transparent and interpretable. By comparing different algorithms, we can identify which algorithm is more transparent and easier to interpret. This is important for applications where transparency is critical, such as in healthcare or finance.



	Accuracies(%)
Random Forest	39.533510
XG Boost	57.868320
Gradient Descent	33.215235
Logistic Regression	58.311190
KNN	16.445232
SVM	35.518158

FIG 6.5.1: Comparison of Algorithms

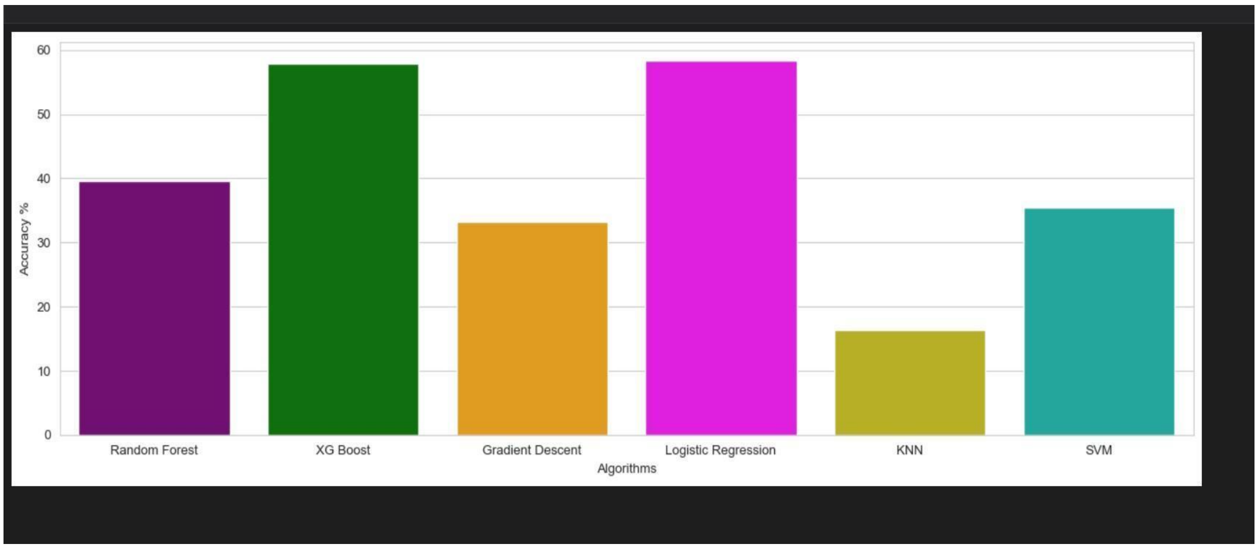
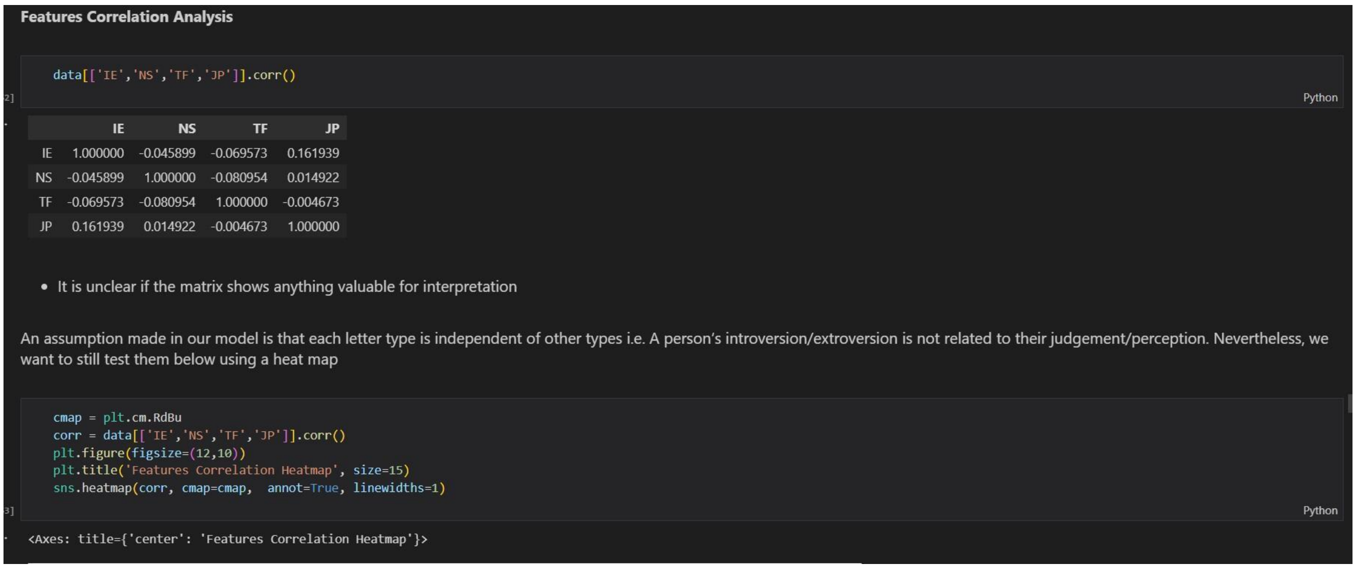


FIG 6.5.2: Graph of Algorithms accuracies

6.6. Feature Correlation Analysis:

Feature correlation analysis is an important step in an AI project because it helps to identify the relationships between the features and the target variable. Here are some reasons why feature correlation analysis is important in an AI project:

1. Feature Selection: Feature correlation analysis can help to identify highly correlated features, which can be combined or removed to reduce the dimensionality of the data. This helps to improve the model's performance and reduce the risk of overfitting.
2. Model Interpretation: Feature correlation analysis can also help to interpret the model's results. By identifying which features are highly correlated with the target variable, we can gain insights into which factors are most important for predicting the target variable.
3. Model Performance: Feature correlation analysis can also help to improve the model's performance. By identifying which features are highly correlated with the target variable, we can select the most relevant features and build a more accurate model.
4. Data Cleaning: Feature correlation analysis can also help to identify errors or anomalies in the data. For example, if two features are highly correlated but one has missing or incorrect data, this could indicate an error in the data that needs to be corrected.
5. Feature Engineering: Feature correlation analysis can also help to guide feature engineering. By identifying which features are highly correlated with the target variable, we can create new features that capture these relationships and improve the model's performance.



Features Correlation Analysis

```python
data[['IE','NS','TF','JP']].corr()
```

	IE	NS	TF	JP
IE	1.000000	-0.045899	-0.069573	0.161939
NS	-0.045899	1.000000	-0.080954	0.014922
TF	-0.069573	-0.080954	1.000000	-0.004673
JP	0.161939	0.014922	-0.004673	1.000000

- It is unclear if the matrix shows anything valuable for interpretation

An assumption made in our model is that each letter type is independent of other types i.e. A person’s introversion/extroversion is not related to their judgement/perception. Nevertheless, we want to still test them below using a heat map

```python
cmap = plt.cm.RdBu
corr = data[['IE','NS','TF','JP']].corr()
plt.figure(figsize=(12,10))
plt.title('Features Correlation Heatmap', size=15)
sns.heatmap(corr, cmap=cmap,  annot=True, linewidths=1)
```

```
<Axes: title={'center': 'Features Correlation Heatmap'}>
```

FIG 6.6.1: Feature Correlation Analysis

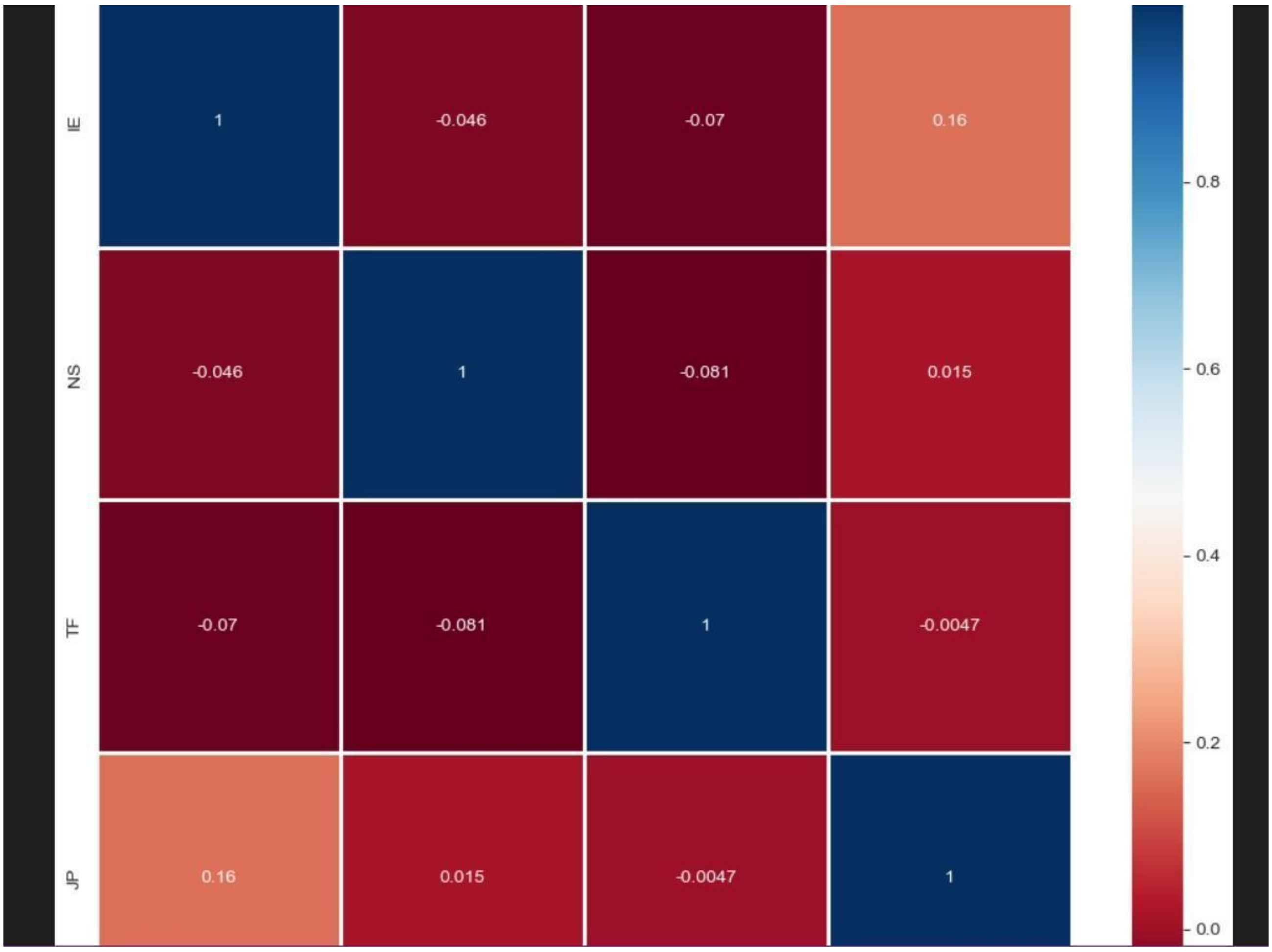


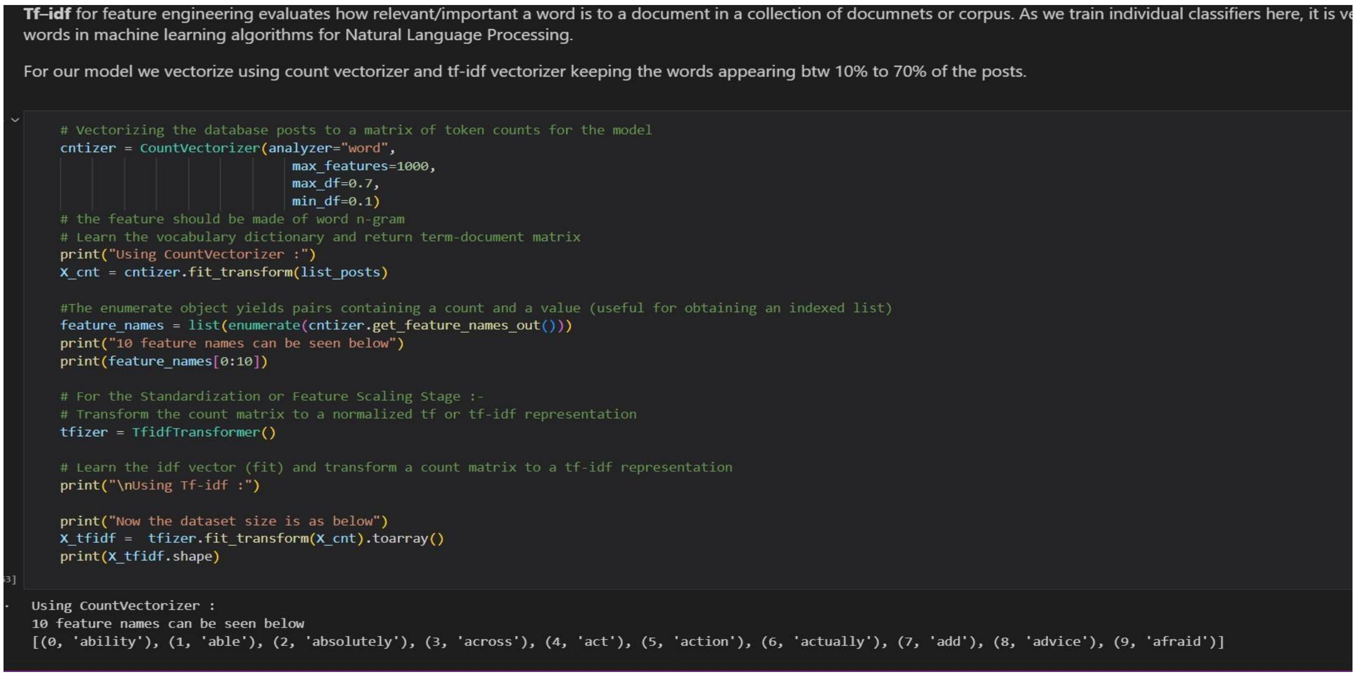
FIG 6.6.2: Graph for Feature Correlation Analysis

6.7. Feature Engineering:

Feature engineering is an important step in an AI project because it helps to extract relevant information from the raw data and create new features that can improve the performance of the model. Here are some reasons why feature engineering is important in an AI project:

1. Data Quality: Feature engineering can help to improve the quality of the data. By identifying missing values, outliers, and other anomalies, we can clean the data and create new features that are more robust to noise.
2. Model Performance: Feature engineering can help to improve the performance of the model. By creating new features that capture the underlying patterns in the data, we can improve the accuracy and reliability of the model.
3. Generalization: Feature engineering can help to improve the generalization of the model. By creating new features that capture the underlying relationships in the data, we can make the model more robust and better able to generalize to new, unseen data.

4. Model Interpretation: Feature engineering can help to interpret the model's results. By creating new features that are more interpretable, we can gain insights into which factors are most important for predicting the target variable.
5. Domain Expertise: Feature engineering can also help to incorporate domain expertise into the model. By creating new features that capture domain-specific knowledge, we can improve the performance of the model and make it more relevant to the problem at hand.



Tf–idf for feature engineering evaluates how relevant/important a word is to a document in a collection of documnets or corpus. As we train individual classifiers here, it is ve
words in machine learning algorithms for Natural Language Processing.

For our model we vectorize using count vectorizer and tf-idf vectorizer keeping the words appearing btw 10% to 70% of the posts.

```
# Vectorizing the database posts to a matrix of token counts for the model
cntizer = CountVectorizer(analyzer="word",
                             max_features=1000,
                             max_df=0.7,
                             min_df=0.1)
# the feature should be made of word n-gram
# Learn the vocabulary dictionary and return term-document matrix
print("Using CountVectorizer :")
X_cnt = cntizer.fit_transform(list_posts)

#The enumerate object yields pairs containing a count and a value (useful for obtaining an indexed list)
feature_names = list(enumerate(cntizer.get_feature_names_out()))
print("10 feature names can be seen below")
print(feature_names[0:10])

# For the Standardization or Feature Scaling Stage :-
# Transform the count matrix to a normalized tf or tf-idf representation
tfizer = TfidfTransformer()

# Learn the idf vector (fit) and transform a count matrix to a tf-idf representation
print("\nUsing Tf-idf :")

print("Now the dataset size is as below")
X_tfidf =  tfizer.fit_transform(X_cnt).toarray()
print(X_tfidf.shape)
```

```
Using CountVectorizer :
10 feature names can be seen below
[(0, 'ability'), (1, 'able'), (2, 'absolutely'), (3, 'across'), (4, 'act'), (5, 'action'), (6, 'actually'), (7, 'add'), (8, 'advice'), (9, 'afraid')]
```

FIG 6.7.1: Feature Engineering

6.8. Implementing the model to predict personalities:

Implementing AI models on real-life content is important for several reasons:

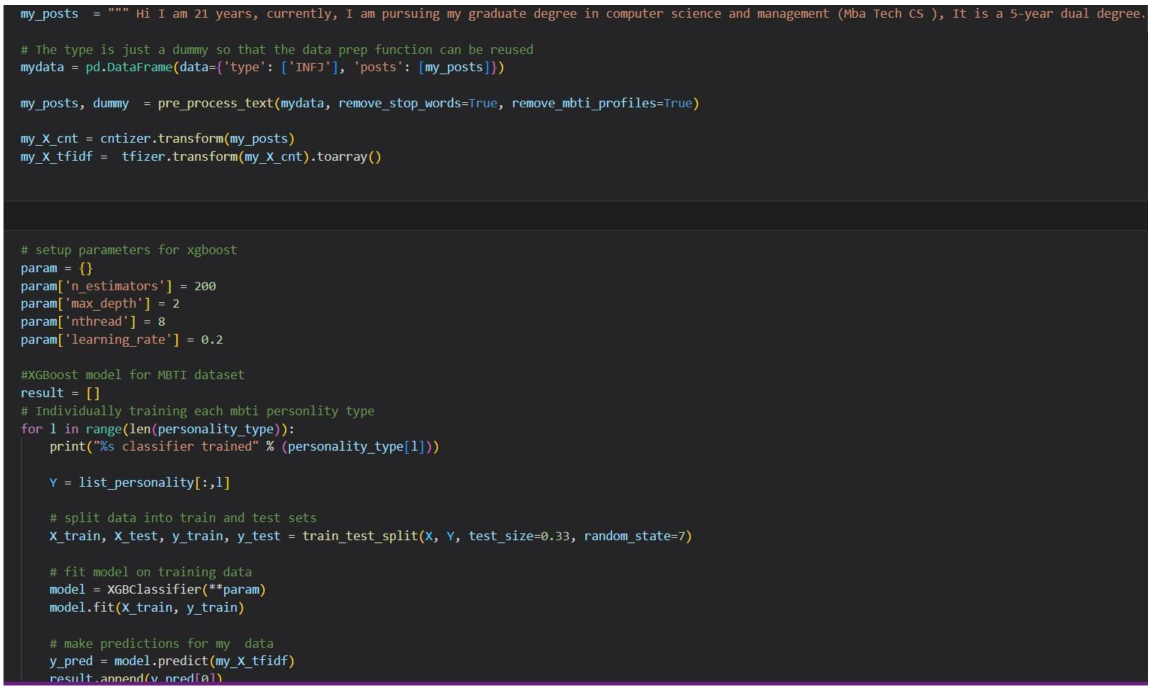
1. Validation: Real-life content allows you to validate the effectiveness of your AI model. Testing your model on real-world data helps you to identify and address any discrepancies between your model's performance and what happens in the real world.

2. Generalization: AI models that are trained on real-life content are more likely to be able to generalize to new data. This is because real-life content is typically more diverse than synthetic or simulated data, which can make your AI model more robust.

3. Impact: The ultimate goal of AI is to make a positive impact on society. Implementing AI models on real-life content allows you to create applications that solve real-world problems and improve people's lives.

4. Feedback: Real-life content provides a rich source of feedback for improving your AI model. By analyzing the results of your model's performance on real-world data, you can identify areas for improvement and refine your model to make it more effective.

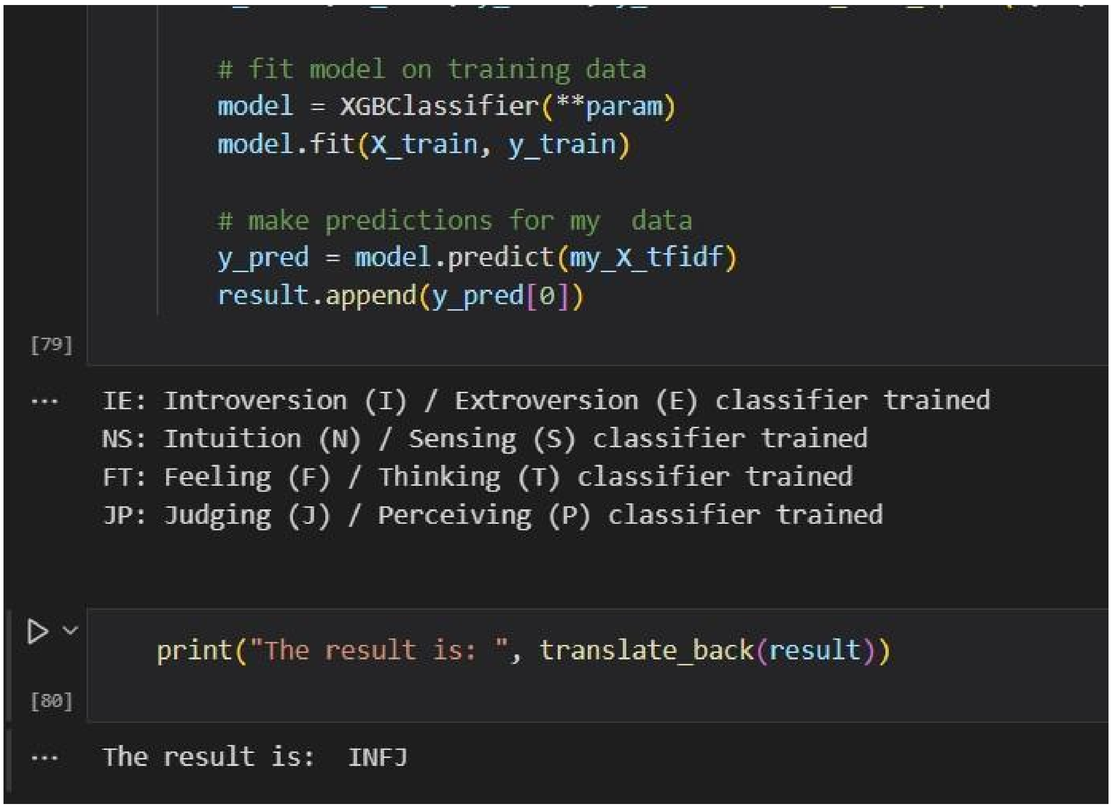
5. Ethics: AI models that are trained on real-life content are more likely to be ethically sound. This is because real-life data is more representative of the diversity of the world, and can help to mitigate the risk of bias and discrimination in AI systems.



```
my_posts  = """ Hi I am 21 years, currently, I am pursuing my graduate degree in computer science and management (Mba Tech CS ), It is a 5-year dual degree.

# The type is just a dummy so that the data prep function can be reused
mydata = pd.DataFrame(data={'type': ['INFJ'], 'posts': [my_posts]})

my_posts, dummy  = pre_process_text(mydata, remove_stop_words=True, remove_mbti_profiles=True)

my_X_cnt = cntizer.transform(my_posts)
my_X_tfidf =  tfizer.transform(my_X_cnt).toarray()
```

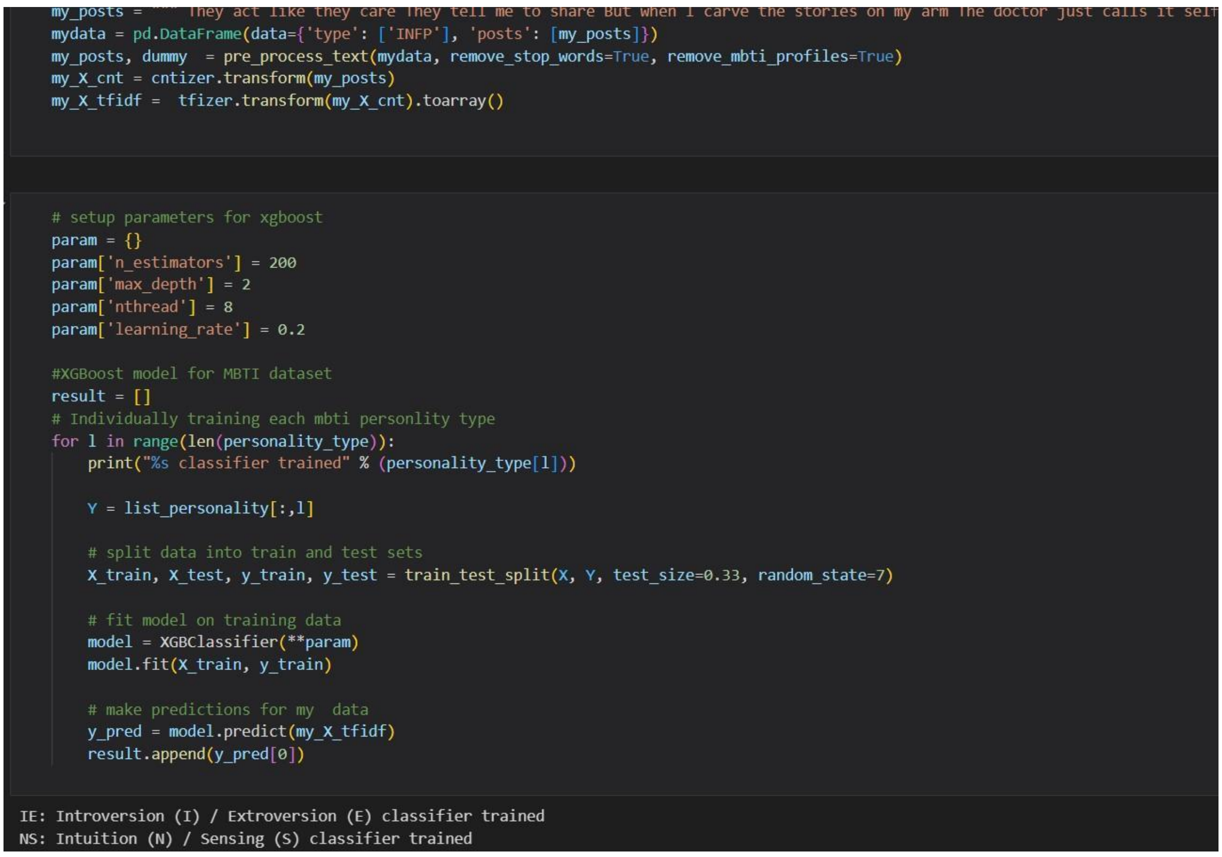
```
# setup parameters for xgboost
param = {}
param['n_estimators'] = 200
param['max_depth'] = 2
param['nthread'] = 8
param['learning_rate'] = 0.2

#XGBoost model for MBTI dataset
result = []
# Individually training each mbti personlity type
for l in range(len(personality_type)):
    print("%s classifier trained" % (personality_type[l]))

    Y = list_personality[:,l]

    # split data into train and test sets
    X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.33, random_state=7)

    # fit model on training data
    model = XGBClassifier(**param)
    model.fit(X_train, y_train)

    # make predictions for my  data
    y_pred = model.predict(my_X_tfidf)
    result.append(y_pred[0])
```

FIG 6.8.1: Prediction through cover letter



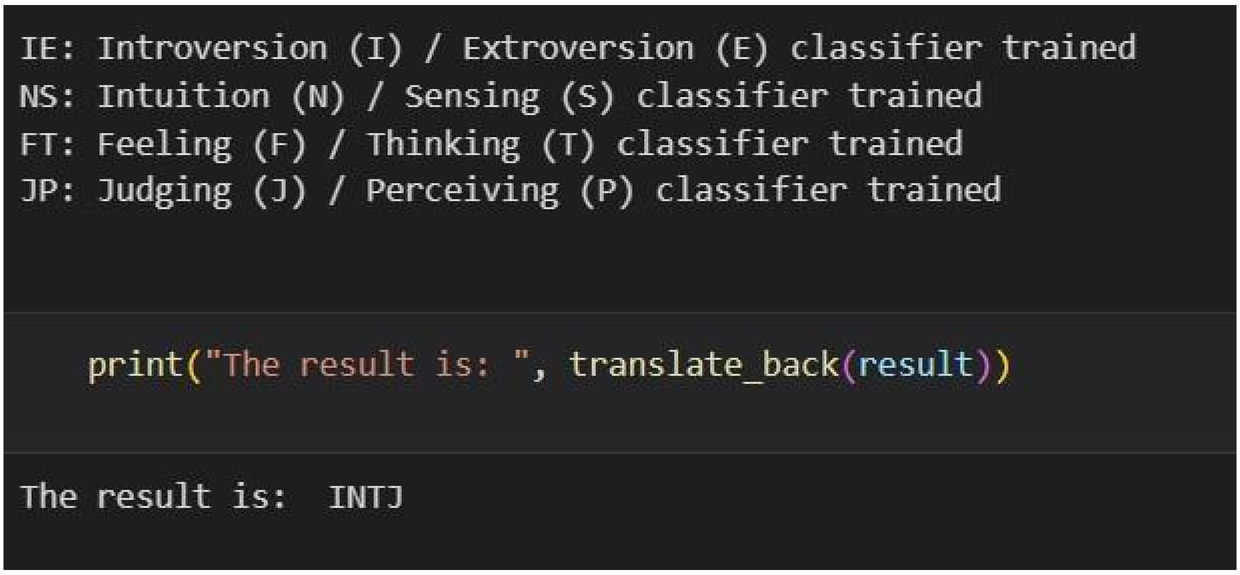
```
        # fit model on training data
        model = XGBClassifier(**param)
        model.fit(X_train, y_train)

        # make predictions for my  data
        y_pred = model.predict(my_X_tfidf)
        result.append(y_pred[0])
[79]

...  IE: Introversion (I) / Extroversion (E) classifier trained
     NS: Intuition (N) / Sensing (S) classifier trained
     FT: Feeling (F) / Thinking (T) classifier trained
     JP: Judging (J) / Perceiving (P) classifier trained

     print("The result is: ", translate_back(result))
[80]

...  The result is:  INFJ
```

FIG 6.8.2: Result from Cover letter



```
my_posts = """ They act like they care They tell me to share But when I carve the stories on my arm The doctor just calls it self
mydata = pd.DataFrame(data={'type': ['INFP'], 'posts': [my_posts]})
my_posts, dummy  = pre_process_text(mydata, remove_stop_words=True, remove_mbti_profiles=True)
my_X_cnt = cntizer.transform(my_posts)
my_X_tfidf =  tfizer.transform(my_X_cnt).toarray()


# setup parameters for xgboost
param = {}
param['n_estimators'] = 200
param['max_depth'] = 2
param['nthread'] = 8
param['learning_rate'] = 0.2

#XGBoost model for MBTI dataset
result = []
# Individually training each mbti personlity type
for l in range(len(personality_type)):
    print("%s classifier trained" % (personality_type[l]))

    Y = list_personality[:,l]

    # split data into train and test sets
    X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.33, random_state=7)

    # fit model on training data
    model = XGBClassifier(**param)
    model.fit(X_train, y_train)

    # make predictions for my  data
    y_pred = model.predict(my_X_tfidf)
    result.append(y_pred[0])

IE: Introversion (I) / Extroversion (E) classifier trained
NS: Intuition (N) / Sensing (S) classifier trained
```

FIG 6.8.3: Prediction through a poem

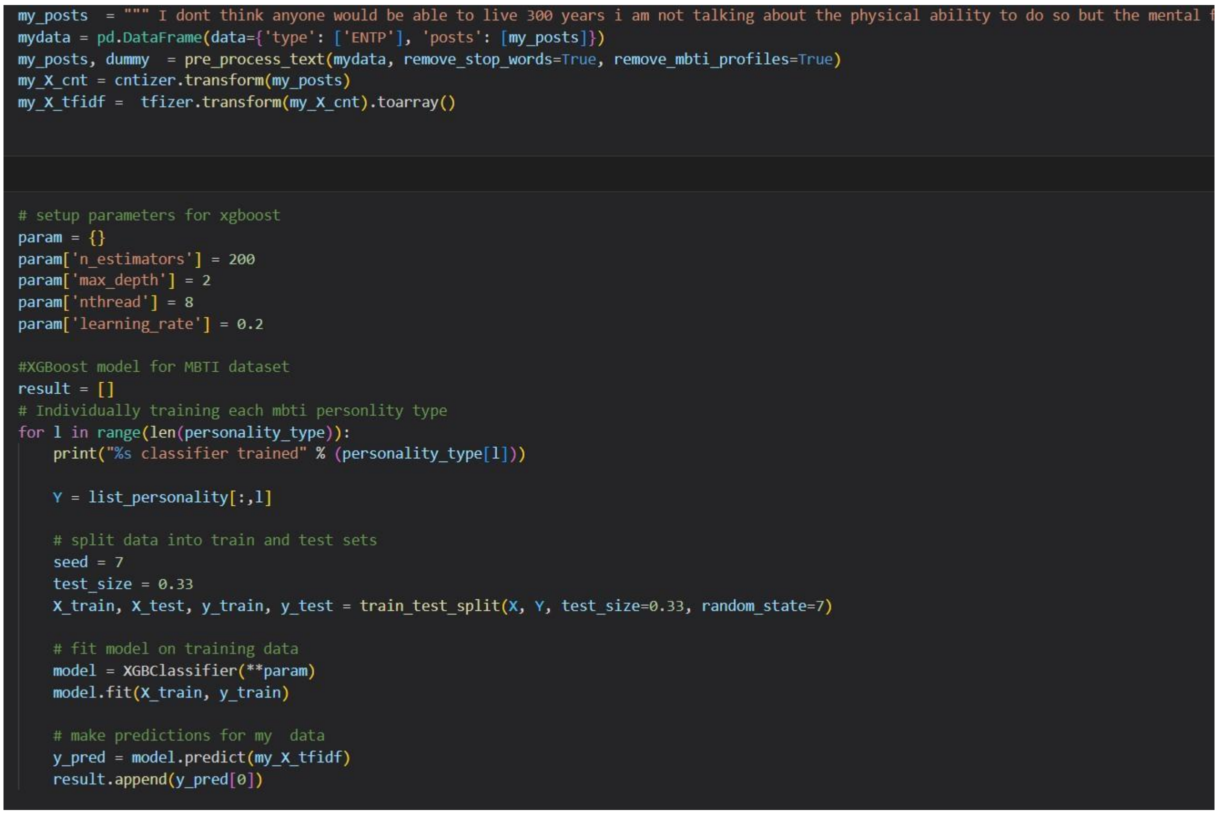


```
IE: Introversion (I) / Extroversion (E) classifier trained
NS: Intuition (N) / Sensing (S) classifier trained
FT: Feeling (F) / Thinking (T) classifier trained
JP: Judging (J) / Perceiving (P) classifier trained
```

```python
print("The result is: ", translate_back(result))
```

```
The result is:  INTJ
```

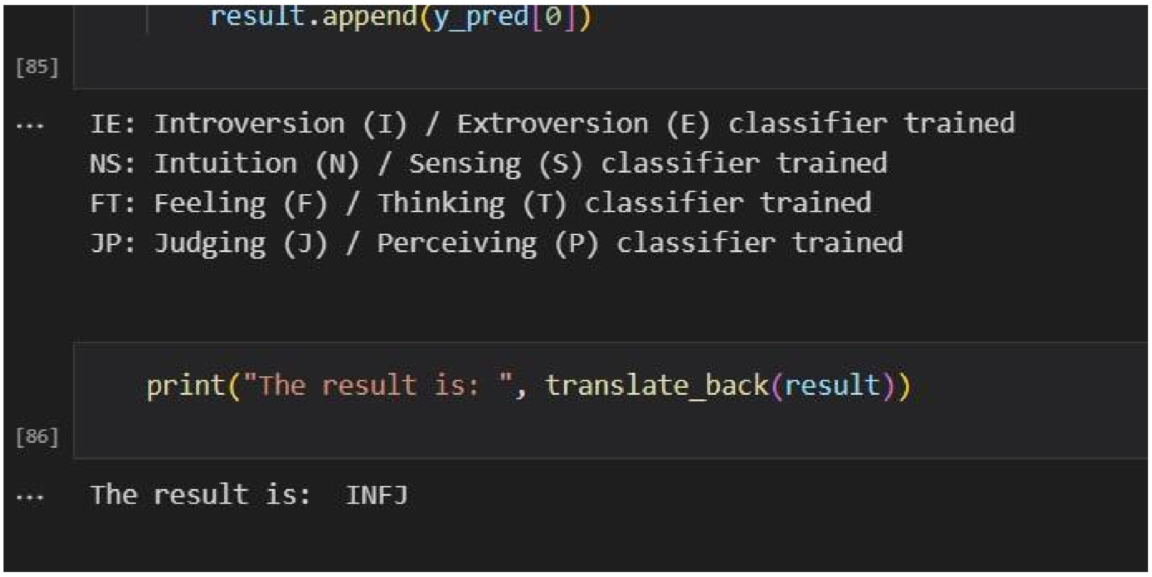
FIG 6.8.4: Result from a Poem



```python
my_posts  = """ I dont think anyone would be able to live 300 years i am not talking about the physical ability to do so but the mental 
mydata = pd.DataFrame(data={'type': ['ENTP'], 'posts': [my_posts]})
my_posts, dummy  = pre_process_text(mydata, remove_stop_words=True, remove_mbti_profiles=True)
my_X_cnt = cntizer.transform(my_posts)
my_X_tfidf =  tfizer.transform(my_X_cnt).toarray()
```

```python
# setup parameters for xgboost
param = {}
param['n_estimators'] = 200
param['max_depth'] = 2
param['nthread'] = 8
param['learning_rate'] = 0.2

#XGBoost model for MBTI dataset
result = []
# Individually training each mbti personlity type
for l in range(len(personality_type)):
    print("%s classifier trained" % (personality_type[l]))

    Y = list_personality[:,l]

    # split data into train and test sets
    seed = 7
    test_size = 0.33
    X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.33, random_state=7)

    # fit model on training data
    model = XGBClassifier(**param)
    model.fit(X_train, y_train)

    # make predictions for my  data
    y_pred = model.predict(my_X_tfidf)
    result.append(y_pred[0])
```

FIG 6.8.5: Prediction through a short essay



```
        result.append(y_pred[0])
[85]
...  IE: Introversion (I) / Extroversion (E) classifier trained
     NS: Intuition (N) / Sensing (S) classifier trained
     FT: Feeling (F) / Thinking (T) classifier trained
     JP: Judging (J) / Perceiving (P) classifier trained

    print("The result is: ", translate_back(result))
[86]
...  The result is:  INFJ
```

FIG 6.8.6: Result from a short essay

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENTS

7.1 Conclusion

In this project, we developed an model that can accurately predict personality traits based on user-generated content.The recommended system stipulates the Myers–Briggs kind Indicator with a view to categorize the character types in sixteen patterns through 4 dichotomies, specifically, (1) Introversion - Extroversion, (2) Sensing - Intuition, (3) Thinking - Feeling and (4) Judging - Perceiving.

Overall, our project has significant implications for the fields of personality psychology and AI. Our model has the potential to help organizations make better hiring decisions, improve mental health interventions, and enhance personalized user experiences across various industries.

7.2 Future Enhancements

While our project has achieved promising results, there are several ways in which it could be enhanced in the future. Here are a few suggestions:

- Expand the dataset: Our model was trained on a relatively small dataset, which may limit its generalizability. Future research could collect larger datasets from more diverse populations to improve the accuracy and robustness of the model.
- Incorporate additional features: In this project, we focused on language style and content as the main predictors of personality. However, there may be other features, such as social media data or facial expressions, that could be incorporated into the model to improve its performance.
- Evaluate the model's fairness: As with any AI model, there is a risk of bias or unfairness in the predictions. Future research could explore ways to ensure that the model is fair and unbiased, particularly with respect to factors such as gender, race, and age.
- Extend the model to other personality traits: In this project, we focused on predicting the Big Five personality traits. However, there are other personality dimensions that could be incorporated into the model, such as grit, emotional intelligence, or creativity.

- Develop user-friendly interfaces: Our model currently requires users to input text data to generate personality predictions. However, future research could explore ways to make the model more accessible and user-friendly, such as through voice recognition or natural language processing.

REFERENCES

[1] X. Sun, B. Liu, J. Cao, J. Luo and X. Shen, “Who Am I? Personality Detection Based on Deep Learning for Texts,” 2018 IEEE International Conference on Communications (ICC), Kansas City, MO, USA, 2018.

[2] I. Liu, S. Ni and K. Peng, “Predicting Personality with Smartphone Cameras: A Pilot Study,” 2020 IEEE International Conference on Human-Machine Systems (ICHMS), Rome, Italy, 2020.

[3] Wan D., Zhang C., Wu M., An Z. (2014) “Personality Prediction Based on All Characters of User Social Media Information”. In: Huang H., Liu T., Zhang HP., Tang J. (eds) Social Media Processing. SMP 2014. Communications in Computer and Information Science, vol 489. Springer, Berlin, Heidelberg, 2014.

[4] C. Li, J. Wan and B. Wang, “Personality Prediction of Social Network Users,” 2017 16th International Symposium on Distributed Computing and Applications to Business, Engineering and Science (DCABES), Anyang, China, 2017.

[5] M. M. Tadesse, H. Lin, B. Xu and L. Yang, “Personality Predictions Based on User Behavior on the Facebook Social Media Platform,” in IEEE Access, vol. 6, pp. 61959-61969, 2018.

[6] T. Tandera, D. Suhartono, R. Wongso, Y. L. Prasetio et al., “Personality prediction system from facebook users,” Procedia Computer Science, vol. 116, pp. 604–611, 2017.