PERSONALITY PREDICTION ANALYSIS USING MACHINE LEARNING

A MINI PROJECT REPORT

Submitted by

SWATHY SREE C S (211422104509)

SRUTHI K (211422104484)

in partial fulfilment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

October 2024

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

BONAFIDE CERTIFICATE

Certified that this project report **"PERSONALITY PREDICTION ANALYSIS"** is the bonafide work of **SWATHY SREE C S (211422104509) and SRUTHI K (211422104484)** who carried out the project work under my supervision.

SIGNATURE OF THE HOD

Dr. L. JABASHEELA, M.E., Ph.D.,
HE AD OF THE DEPARTMENT
PROFEESOR
DEPARTMENT OF CSE,
PANIMALAR ENGINEERING COLLEGE ,
NASARATHPETTAI,
POONAMALLEE,
CHENNAI-600 123.

SIGNATURE OF THE SUPERVISOR

Mrs.M. ABIRAMI, M. Tech.,
SUPERVISOR
ASSISTANT PROFESSOR
DEPARTMENT OF CSE,
PANIMALAR ENGINEERING COLLEGE,
NASARATHPETTAI,
POONAMALLEE,
CHENNAI-600 123.

Certified that the above candidate(s) were examined in the Mini Project Viva-Voce Examination held on.....

INTERNAL EXAMINAR

EXTERNAL EXAMINAR

ACKNOWLEDGEMENT

We express our deep gratitude to our respected Secretary and Correspondent **Dr. P. CHINNADURAI, M.A., Ph.D.** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We would like to extend our heartfelt and sincere thanks to our Directors **Tmt. C. VIJAYARAJESWARI, Dr. C. SAKTHIKUMAR, M.E., Ph.D., and Tmt. SARANYASREE SAKTHIKUMAR B.E., M.B.A.,** for providing us with the necessary facilities for completion of this project.

We also express our gratitude to our Principal **Dr. K. Mani, M.E., Ph.D.** for his timely concern and encouragement provided to us throughout the course.

We thank the HOD of CSE Department**, Dr. L. JABASHEELA, M.E., Ph.D.,** for the support extended throughout the project.

We would like to thank my Project Guide **Mrs M. ABIRAMI, M. Tech.** and all the faculty members of the Department of CSE for their advice and suggestions for the successful completion of the project.

NAME OF THE STUDENTS

SWATHY SREE C S (211422104509)

SRUTHI K (211422104484)

ABSTRACT

In this project, we explore the application of various machine learning algorithms to predict personality types based on textual data using the Myers-Briggs Type Indicator (MBTI) dataset. The dataset comprises user-generated posts, which are preprocessed through a series of natural language processing (NLP) techniques, including text normalization, stopword removal, and lemmatization. We employ the TF-IDF (Term Frequency-Inverse Document Frequency) method to convert the textual data into numerical features. Several classifiers—Gaussian Naive Bayes, Multinomial Naive Bayes, Random Forest, XGBoost, LightGBM, Support Vector Machine (SVM), and Logistic Regression—are trained and evaluated to predict the MBTI personality types. The models are compared based on accuracy and detailed classification reports. Among the models tested, the XGBoost classifier outperforms others with an accuracy of 67.55%, demonstrating its effectiveness for this multi-class text classification problem. This project highlights the potential of machine learning in personality prediction from textual data and provides a comparative analysis of various classification algorithms for this purpose.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Personality prediction has garnered significant attention in both psychology and machine learning communities due to its potential applications in marketing, recruitment, social media analysis, and human-computer interaction. The Myers-Briggs Type Indicator (MBTI), one of the most widely recognized personality classification systems, categorizes individuals into 16 distinct personality types based on four dichotomies: Introversion (I) vs. Extraversion (E), Intuition (N) vs. Sensing (S), Thinking (T) vs. Feeling (F), and Judging (J) vs. Perceiving (P). With the growing amount of user-generated content on social media, there is an increasing opportunity to leverage machine learning models for automatically classifying individuals into personality types based on their textual data.

This project aims to explore the use of natural language processing (NLP) and machine learning techniques to predict MBTI personality types from textual posts. We utilize the MBTI dataset, which consists of posts collected from various social media platforms, each labeled with a specific MBTI personality type. The task presents several challenges, including the high dimensionality of text data and the complexity of predicting 16 distinct personality types from relatively short text samples.

To address these challenges, we preprocess the textual data by removing noise, applying lemmatization, and transforming the text into numerical representations using the Term Frequency-Inverse Document Frequency (TF-IDF) technique. A variety of machine learning models—including Naive Bayes, Random Forest, XGBoost, LightGBM, Support Vector Machine (SVM), and Logistic Regression—are employed to classify the personality types. The performance of each model is evaluated using accuracy and classification metrics, providing a comprehensive comparison of model efficacy for personality prediction tasks.

In this report, we outline the data preprocessing steps, feature extraction techniques, and the models used, followed by a detailed discussion of the results. Our goal is to identify the most effective machine learning model for this task and highlight the potential of machine learning in predicting personality traits from text.

1.1 OVERVIEW

1.**Data Collection and Preprocessing**: The project uses a publicly available dataset containing posts from users along with their corresponding MBTI personality types. To transform the raw text data into a format suitable for machine learning models, preprocessing steps such as converting text to lowercase, removing URLs, digits, punctuation, and stopwords were applied. Lemmatization was also performed to reduce words to their base forms.

2.**Feature Extraction**: A key part of the project involves converting the cleaned text into numerical features using the Term Frequency-Inverse Document Frequency (TF-IDF) method. This method captures the importance of words in relation to the entire dataset, helping to identify which words are more relevant for classification.

3. **Machine Learning Models**: Multiple machine learning models were trained and evaluated for personality prediction, including:

- **Naive Bayes Classifiers (GaussianNB, MultinomialNB)**: Simple probabilistic classifiers commonly used for text classification tasks.
- **Random Forest Classifier**: An ensemble learning method that combines multiple decision trees to improve prediction accuracy.
- **Support Vector Machine (SVM)**: A powerful classification model that works well in high-dimensional spaces like text data.
- **Logistic Regression**: A linear model that is effective for multi-class classification problems.
- **XGBoost and LightGBM**: Gradient boosting algorithms that improve performance by combining weak classifiers into a strong classifier.

4.**Model Evaluation**: The performance of each model was evaluated using metrics such as accuracy, precision, recall, and F1-score. The accuracy scores of each model were compared to identify the best-performing approach.

5.**Key Findings**: The project revealed that ensemble methods like XGBoost and LightGBM performed the best, achieving accuracy scores of around 67.5% and 67.3%, respectively. These methods outperformed simpler classifiers like Naive Bayes and Logistic Regression due to their ability to handle high-dimensional text data effectively.

1.2 Problem Definition

Personality prediction plays a vital role in various fields, including psychology, human resources, marketing, and social networking. The Myers-Briggs Type Indicator (MBTI) is one of the most widely used models for understanding human personality. However, accurately predicting an individual's MBTI personality type based on behavioral data is a challenging task, particularly when relying solely on textual content.

In this project, the problem is framed as a **multi-class classification task** where the goal is to predict one of 16 MBTI personality types (e.g., INFP, ENTJ) based on user-generated posts. The primary challenges involved in this problem include:

1. **Textual Nature of Data**: The data consists of free-form text, which introduces variability in language, vocabulary, and expression. Text processing is crucial to convert unstructured text into numerical features suitable for machine learning models.
2. **Multi-class Classification**: The MBTI model has 16 personality types, making this a multi-class classification problem. Each type is defined by four distinct personality dimensions, and predicting the right combination of these dimensions adds complexity to the classification task.
3. **Imbalanced Data**: Personality distribution in the dataset may be imbalanced, with certain personality types being underrepresented. This imbalance could lead to biased models that favor more common personality types.
4. **Feature Engineering**: Textual data must be transformed into numerical features that machine learning algorithms can process. Efficient feature extraction techniques are required to capture meaningful patterns in the text.
5. **Model Selection and Performance**: Identifying the most effective machine learning model for this task is critical. Different models may handle text features and classification complexities differently, and the performance of each model must be rigorously evaluated.

Objectives:

- To develop a machine learning model that can predict MBTI personality types from user-generated textual posts.
- To compare the performance of multiple machine learning algorithms, including Naive Bayes, Random Forest, Support Vector Machines (SVM), XGBoost, LightGBM, and Logistic Regression.
- To achieve high accuracy while addressing challenges such as text variability and class imbalance.

This problem focuses on how well machine learning models can interpret the content and structure of text data to predict human personality traits as defined by the MBTI framework.

CHAPTER 2

LITERATURE SURVEY

The burgeoning field of personality prediction through machine learning (ML) has witnessed substantial growth, with researchers exploring diverse methodologies and data sources.

1.**"Smart-Hire Personality Prediction Using ML" (May 2023) by Isha Gupta and Manasvi Jain:**

This study underscores the practical implications of personality classification types through ML predictions, can actively engage in self-improvement efforts. The paper emphasizes the potential impact of such insights on personal and professional development. It suggests that individuals, upon discovering their personality.

2."A Study on Personality Prediction & Classification Using Data Mining Algorithms" (August 2022) by Pavitha N., Somesh Kamnapure, and Ayush Gundawa:

Highlighting the importance of personality in personal and professional contexts, this work explores data mining algorithms to rapidly predict and categorize an individual's personality. The researchers advocate for integrating ML techniques, particularly through intuitive input methods like questionnaires, to enhance prediction efficiency. This type of personality prediction can be used efficiently using data mining algorithm.

3."Language Style Matters: Personality Prediction from Textual Styles Learning" (November 2023) by Meiling Li and Hezi Liu:

This research delves into psycholinguistic literature, emphasizing the role of language styles in unveiling personality aspects. The paper contends that language styles offer insights into users' personalities, including social networks and mental health. Textual styles learning is presented as a valuable approach for personality prediction.

4."Personality Prediction using Machine Learning" (June 2022) by Hima Vijay and Neenu Sebastian:

Acknowledging the importance of sorting individuals based on personality types, this work emphasizes the applications of ML algorithms in achieving this goal. The paper contributes to the literature by exploring the potential benefits and implications of personality prediction using ML.

5. "Personality Prediction from Textual Data" by Plank and Hovy (2015):

Early research has focused on using textual data from social media, blogs, and forums to predict personality types. Techniques like Bag-of-Words (BoW) and TF-IDF have been commonly applied to represent text numerically. For example, he used TF-IDF and word embeddings to predict Big Five personality traits from Twitter posts, showing that personality prediction from short text is feasible.

6. "Machine Learning Algorithms for Personality Classification" by Verhoeven et al. (2016) and Gjurković and Šnajder (2018):

Classical machine learning models such as Naive Bayes, SVM, and Random Forest have been widely applied for personality classification. Studies by them demonstrated moderate success in using these models for personality type prediction, with Random Forest and SVM often outperforming simpler models like Naive Bayes.

7. "Deep Learning Approaches" by Kim et al. (2020):

Recent research has turned to deep learning, particularly RNNs and transformer models like BERT. LSTM networks have been used to capture text sequence patterns, improving personality classification, while transformers such as BERT offer state-of-the-art results by capturing complex contextual relationships. These models, however, require large datasets and substantial computational power.

8. "Ensemble Learning and Boosting Methods" by Li et al. (2019):

Boosting algorithms like XGBoost and LightGBM have become popular in personality prediction due to their capacity to handle high-dimensional data and reduce both bias and variance. He have shown that these models outperform traditional classifiers, particularly when combined with strong feature extraction techniques like TF-IDF.

9. "Personality prediction from multi model data" by Vinciarelli and Mohammaadi (2014):

Multi model personality prediction is gaining interest, where data from multiple sources such as video, audio, text are combined. For instance, facial expressions and voice tone may be integrated with linguistic analysis to improve prediction accuracy.

2.1 TABULATION:

S.NO	TITLE	DESCRIPTION	AUTHOR	YEAR
1	**Smart-Hire Personality Prediction Using ML**	This study underscores the practical implications of personality classification types through ML predictions, can actively engage in self-improvement efforts. The paper emphasizes the potential impact of such insights on personal and professional development. It suggests that individuals, upon discovering their personality.	**Isha Gupta and Manasvi Jain**	2023
2	**A Study on Personality Prediction & Classification Using Data Mining Algorithms**	Highlighting the importance of personality in personal and professional contexts, this work explores data mining algorithms to rapidly predict and categorize an individual's personality. The researchers advocate for integrating ML techniques, particularly through intuitive input methods like questionnaires, to enhance prediction efficiency.	**Pavitha N., Somesh Kamnapure, and Ayush Gundawa:**	2022
3	**Language Style Matters: Personality Prediction from Textual Styles Learning**	This research delves into psycholinguistic literature, emphasizing the role of language styles in unveiling personality aspects. The paper contends that language styles offer insights into users' personalities, including social networks and mental health. It is presented as a valuable approach for personality prediction.	**Meiling Li and Hezi Liu**	2023

S.NO	TITLE	DESCRIPTION	AUTHOR	YEAR
4	**Personality Prediction using Machine Learning**	Acknowledging the importance of sorting individuals based on personality types, this work emphasizes the applications of ML algorithms in achieving this goal. The paper contributes to the literature by exploring the potential benefits and implications of personality prediction using ML.	**Hima Vijay and Neenu Sebastian**	2022
5	**Personality Prediction from Textual Data**	Early research has focused on using textual data from social media, blogs, and forums to predict personality types. Techniques like Bag-of-Words (BoW) and TF-IDF have been commonly applied to represent text numerically. For example, he used TF-IDF and word embeddings to predict Big Five personality traits from Twitter posts, showing that personality prediction from short text is feasible	**Plank and Hovy**	2015
6	**Machine Learning Algorithms for Personality Classification"**	Classical machine learning models such as Naive Bayes, SVM, and Random Forest have been widely applied for personality classification. Studies by them demonstrated moderate success in using these models for personality type prediction, with Random Forest and SVM often outperforming simpler models like Naive Bayes.	**Verhoeven et al. and Gjurković and Šnajder**	2018

S.NO	TITLE	DESCRIPTION	AUTHOR	YEAR
7	**Deep Learning Approaches**	Recent research has turned to deep learning, particularly RNNs and transformer models like BERT. LSTM networks have been used to capture text sequence patterns, improving personality classification, while transformers such as BERT offer state-of-the-art results by capturing complex contextual relationships. These models, however, require large datasets and substantial computational power.	**Kim et al**	2020
8	**Ensemble Learning and Boosting Methods"**	Boosting algorithms like XGBoost and LightGBM have become popular in personality prediction due to their capacity to handle high-dimensional data and reduce both bias and variance. He have shown that these models outperform traditional classifiers, particularly when combined with strong feature extraction techniques like TF-IDF.	**Li et al**	2019
9	**Personality prediction from multi model data**	Multi model personality prediction is gaining interest, where data from multiple sources such as video, audio, text are combined. For instance, facial expressions and voice tone may be integrated with linguistic analysis to improve prediction accuracy.	**Vinciarelli and Mohammaadi**	2014

CHAPTER 3

SOFTWARE REQUIREMENTS

3.1 MATERIALS

The materials used in this project consist of various software tools, libraries, and datasets necessary for conducting machine learning tasks and evaluating model performance. Below is a list of the key materials:

1. Dataset:

- **MBTI Personality Dataset**: The main dataset used in this project contains textual posts from users, along with their respective MBTI (Myers-Briggs Type Indicator) personality type. This dataset is composed of two main features:
    - **Posts**: A collection of user-generated textual content (e.g., blog posts, forum comments).
    - **Personality Types**: One of 16 possible MBTI personality types assigned to each user (e.g., INFP, ENTP, etc.).
- The dataset is publicly available and was accessed via a CSV file.

2. Software and Libraries:

- **Python**: The programming language used to implement the machine learning pipeline and data processing.
- **Pandas**: For data manipulation and analysis, particularly for loading and transforming the dataset.
- **NumPy**: For numerical operations and handling array-based data.
- **Matplotlib and Seaborn**: For data visualization, including the creation of bar charts, pie charts, and distribution plots.
- **Scikit-learn**: For implementing machine learning models, performing data preprocessing, and evaluating model performance. Specific functionalities included:
    - **TF-IDF Vectorizer**: For converting textual data into numerical features.
    - **Naive Bayes Classifiers (GaussianNB and MultinomialNB)**: For personality type classification.
    - **Random Forest Classifier**: For ensemble learning using decision trees.
    - **Support Vector Machine (SVM)**: For classification tasks with high-dimensional feature spaces.

    - **Logistic Regression**: For multi-class classification.
    - **LabelEncoder**: For encoding personality types into numerical labels.
- **XGBoost and LightGBM**: Advanced gradient boosting frameworks for building highly accurate classification models.
- **NLTK (Natural Language Toolkit)**: For text preprocessing, including:
    - **Stopwords Removal**: To eliminate common words that do not contribute to the classification task.
    - **WordNet Lemmatizer**: For reducing words to their base form (lemmatization).
- **XGBoost and LightGBM**: For ensemble learning through boosting, widely used in text classification tasks.

3. Computational Resources:

- The project was executed in a local development environment and Google Colab, utilizing cloud-based resources.
- **Google Colab**: A free cloud-based platform that provides GPU acceleration, which was used for training machine learning models and handling large datasets.

4. Performance Metrics:

- Several metrics were used to evaluate the performance of the models, including:
    - **Accuracy**: The percentage of correct predictions made by the model.
    - **Precision, Recall, and F1-score**: Detailed performance metrics generated using the classification_report function in Scikit-learn to assess how well the model performs for each personality type.

These materials enabled the successful development and evaluation of multiple machine learning models for predicting MBTI personality types from textual data.

CHAPTER 4

METHODOLOGY

The methodology for this project involves several steps to preprocess the data, implement machine learning models, and evaluate their performance. Below is a detailed breakdown of the process:

1. Data Collection:

- The dataset consists of textual posts from various users, each labeled with one of the 16 MBTI personality types (e.g., INFP, ESTJ, ENFJ).
- The dataset contains two columns: one for the user's posts and another for the corresponding personality type.

2. Data Preprocessing:

To prepare the raw text for model training, several data cleaning and transformation steps were applied:

- **Lowercasing**: All text was converted to lowercase to avoid case-sensitivity issues in the model.
- **URL Removal**: URLs in the posts were removed using regular expressions to prevent noise.
- **Removing Numbers and Special Characters**: Any numbers and special characters (e.g., punctuation) were eliminated to clean the text.
- **Whitespace Normalization**: Consecutive spaces were reduced to single spaces to ensure uniformity.
- **Stopword Removal**: Common stopwords (e.g., "the," "and") were removed using NLTK's list of English stopwords, as these words do not contribute to distinguishing between personality types.
- **Lemmatization**: Using NLTK's WordNetLemmatizer, words were lemmatized to reduce them to their root forms, reducing the dimensionality of the text data.

3. Feature Engineering:

- **TF-IDF (Term Frequency-Inverse Document Frequency)**: The cleaned text data was transformed into numerical features using the TF-IDF vectorizer. This method assigns weights to words based on their frequency in a document relative to how often they appear across the entire dataset.

4. Train-Test Split:

- The dataset was divided into training and testing sets using **stratified sampling** to ensure the same distribution of personality types in both sets.
- 80% of the data was used for training, while 20% was reserved for testing the models.

5. Model Selection:

- Multiple machine learning models were trained on the training data to predict the personality types from the text. These models include:

    - **Gaussian Naive Bayes (GNB)**: Assumes that features are conditionally independent and uses Bayes' theorem to predict the class.
    - **Multinomial Naive Bayes (MNB)**: A variant of Naive Bayes used for multi-class classification problems.
    - **Random Forest Classifier (RFC)**: An ensemble learning method that constructs multiple decision trees and outputs the majority class prediction.
    - **XGBoost**: A high-performance implementation of gradient boosting, optimized for accuracy and speed.
    - **LightGBM**: Another gradient boosting model that is particularly efficient with large datasets and high-dimensional features.
    - **Support Vector Machine (SVM)**: A classifier that separates classes by finding the optimal hyperplane in a high-dimensional space.
    - **Logistic Regression (LR)**: A linear model commonly used for multi-class classification problems.

6. Model Training:

- Each machine learning model was trained using the TF-IDF transformed features of the training data. The models learned to map the text features to one of the 16 personality types.
- **Hyperparameter tuning** was applied in some models, such as XGBoost and Random Forest, to optimize their performance. However, in this methodology, we primarily used the default configurations of the models.

7. Model Evaluation:

- After training, each model was evaluated on the testing set using various performance metrics:
    - **Accuracy**: The proportion of correctly predicted instances out of the total number of instances.
    - **Classification Report**: The classification report was used to generate precision, recall, F1-score, and support metrics for each personality type. These metrics helped assess the model's performance across all personality classes.
- The models were compared based on their accuracy and classification reports to determine the best-performing classifier.

8. Model Comparison:

- The results from each model were recorded and compared in a summary table showing their test accuracy scores.
- **XGBoost** and **LightGBM** emerged as the best-performing models, achieving the highest accuracy, followed closely by **Support Vector Machine (SVM)**.

9. Results Interpretation:

- The accuracy scores and classification reports from all models were analyzed to determine which models were most effective in predicting MBTI personality types.
- The project identified **XGBoost** as the top-performing model, with the highest accuracy of **67.55%**, followed closely by **LightGBM**.

10. Visualization:

- Several visualizations were created to better understand the data and model performance:
    - A bar chart was used to display the distribution of MBTI personality types in the dataset.
    - A pie chart further illustrated the relative proportions of each personality type.

This methodology ensured a systematic approach, from data preprocessing to model evaluation, using both classical and modern machine learning techniques to predict MBTI personality types from textual data.

CHAPTER 5

SYSTEM MODEL

The system model for this personality prediction project consists of several interconnected components that work together to process data, extract features, build machine learning models, and make predictions. The system can be divided into the following key stages:

1. Data Collection and Loading

- **Input**: The system begins with the input of a dataset containing user-generated posts and their corresponding MBTI personality types.
- **Data Source**: The dataset is in CSV format, loaded into the system using Python's Pandas library for manipulation and analysis.

2. Data Preprocessing

- **Text Cleaning**: The raw text data contains noise, such as URLs, punctuation, digits, and special characters. To improve the quality of the data for model training, the system performs the following steps:
    - Lowercasing all text to ensure uniformity.
    - Removing URLs, numbers, and special characters using regular expressions.
    - Removing stopwords (e.g., common words like "the," "is") using NLTK's stopword list.
    - Lemmatizing words to convert them to their base forms (e.g., "running" becomes "run").
- **Output**: Cleaned and preprocessed text data, ready for feature extraction.

3. Feature Extraction

- **TF-IDF Vectorization**: The system uses Term Frequency-Inverse Document Frequency (TF-IDF) to transform the cleaned text into numerical features that can be used for machine learning. TF-IDF measures the importance of words in the context of the entire dataset.
    - **Vectorizer**: The TF-IDF vectorizer converts each post into a fixed-length vector of features (e.g., word importance scores).
- **Output**: A numerical matrix representing the textual data.

4. Splitting the Data

- **Training and Testing Split**: The dataset is split into a training set (80%) and a testing set (20%) using the train_test_split function from Scikit-learn. This ensures that models are trained on a portion of the data and evaluated on unseen data.
- **Stratified Sampling**: Stratified sampling is used to ensure that the distribution of MBTI personality types is preserved in both the training and testing sets.

5. Model Building

- **Machine Learning Algorithms**: The system implements several machine learning algorithms, each of which is trained to classify users into one of the 16 MBTI personality types based on their posts.
    - **Naive Bayes (Gaussian and Multinomial)**: Simple probabilistic models that work well with text data.
    - **Random Forest Classifier**: An ensemble learning model that creates multiple decision trees and averages their results.
    - **Support Vector Machine (SVM)**: A classifier that separates classes with a hyperplane, particularly useful for high-dimensional text data.
    - **XGBoost and LightGBM**: Gradient boosting models that combine weak learners to form a strong classifier.
    - **Logistic Regression**: A linear classifier effective in multi-class classification problems.

6. Model Training

- Each machine learning model is trained on the numerical feature representations from the training data. The system fits the models using Scikit-learn and other specialized libraries (e.g., XGBoost and LightGBM).
- **Label Encoding**: The MBTI personality types are encoded as numerical labels for the training process using LabelEncoder.

7. Model Evaluation

- After training, each model is evaluated on the test dataset to assess its performance. Key metrics include:
    - **Accuracy**: The percentage of correct predictions.
    - **Classification Report**: Detailed performance metrics (precision, recall, F1-score) for each MBTI personality type.

- **Comparison**: The system compares the accuracy of different models to identify the best-performing approach.

8. Model Selection

- Based on the accuracy and other performance metrics, the system identifies the best model. The comparison results show that gradient boosting methods like XGBoost and LightGBM perform better than other models.
- **Result**: The system outputs the model with the highest accuracy and generates a summary of the performance of all models.

9. Prediction Output

- The final output is a trained model capable of predicting MBTI personality types from new textual data. The system can be applied to classify the personality types of users based on their posts in real-world applications.

FLOW CHART:

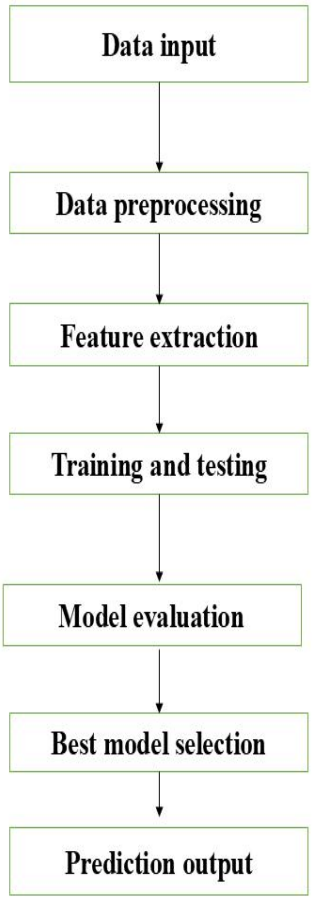

Fig.1.0. Flowchart diagram for personality prediction system analysis.

CHAPTER 6

PROPOSED SYSTEM

The aim of the code is to classify personality types based on text data (posts) from the Myers-Briggs Type Indicator (MBTI) dataset using various machine learning algorithms. The key goals are:

1. **Text Preprocessing**:
    - Clean and process the text data by removing URLs, numbers, special characters, and stopwords. The text is also lemmatized to normalize words.
2. **Feature Extraction**:
    - Convert the processed text data into numerical features using the TF-IDF (Term Frequency-Inverse Document Frequency) vectorizer to represent the importance of words in the posts.
3. **Classification of Personality Types**:
    - Build machine learning models using the processed text data to classify posts into one of the 16 personality types from the MBTI.
4. **Model Comparison**:
    - Train and evaluate several machine learning classifiers, including:
        - Gaussian Naive Bayes
        - Multinomial Naive Bayes
        - Random Forest
        - XGBoost
        - LightGBM
        - Support Vector Machine (SVM)
        - Logistic Regression

5.Performance Evaluation:

- Evaluate each model's performance based on accuracy for both training and test datasets.
- Generate classification reports showing precision, recall, F1-score, and support for each personality type.
- Compare models based on test accuracy to identify the most effective model for personality prediction.

CHAPTER 7

SYSTEM IMPLEMENTATION

PYTHON CODE:

1. LOADING LIBRARIES AND MOUNTING GOOGLE DRIVE

```
from google.colab import drive
drive.mount('/content/drive')
import pandas as pd
import numpy as np
import nltk
import re
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve, roc_auc_score, accuracy_score
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer, TfidfTransformer
from sklearn.model_selection import GridSearchCV, train_test_split
```

- **Google Drive Mounting**: Accesses files in your Google Drive.
- **Libraries**:
    - pandas, numpy for data handling.
    - nltk for NLP processing.
    - re for regular expressions.
    - seaborn and matplotlib for visualization.
    - sklearn for machine learning and metrics.

2. LOADING THE DATASET

```
mbti_df = pd.read_csv('/content/drive/MyDrive/PRML_Bonus_Project/mbti_1.csv')
mbti_df.head()
mbti_df.posts[0]
mbti_df.info()
```

- Loads the dataset into a pandas DataFrame mbti_df.
- Displays the first few rows using head().
- info() shows the structure and summary of the data.

3. VISUALIZING PERSONALITY TYPES DISTRIBUTION

- **Bar Plot of Personality Types**:

```
pd.DataFrame(mbti_df.type.value_counts()).plot.bar()

plt.ylabel('Frequency')

plt.xlabel('Types of Categories')

plt.title('Bar graph showing frequency of different types of personalities')

plt.show()
```

- **Pie Chart**:

```
mbti_df.type.value_counts().plot(kind='pie', figsize=(12,12), autopct='%1.1f%%',
explode=[0.1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0])

plt.title('Pie plot showing different types of personalities')

plt.show()
```

4. TEXT PREPROCESSING

- **Post Length Distribution**:

```
sns.distplot(mbti_df["posts"].apply(len))

plt.xlabel("Length of posts")

plt.ylabel("Density")

plt.title("Distribution of lengths of the post")
```

- **Lowercasing and Cleaning Posts**: Converts all text in the posts column to lowercase:

```
mbti_df["posts"] = mbti_df["posts"].str.lower()
```

- **Removing URLs**:

```
pattern = re.compile(r'https?://[a-zA-Z0-9./-]*/[a-zA-Z0-9?=_.]*[_0-9.a-zA-Z/-]*')

mbti_df._set_value(i, 'posts', re.sub(pattern, ' ', post_temp))
```

  - This uses regular expressions to remove URLs from the posts.

- **Removing Numbers and Special Characters**:

```
pattern = re.compile(r'[0-9]')

post_temp = re.sub(pattern, ' ', post_temp)

pattern = re.compile('\W+')

post_temp = re.sub(pattern, ' ', post_temp)
```

- **Removing Extra Whitespace**:

```
pattern = re.compile('\s+')

post_temp = re.sub(pattern, ' ', post_temp)
```

  - Replaces multiple whitespaces with a single space.

- **Stopword Removal**:

```
from nltk.corpus import stopwords

nltk.download('stopwords')

remove_words = stopwords.words("english")
```

- Removes common stopwords (e.g., "the", "is") from the posts.

- **Lemmatization**:

```
from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()
```

- Lemmatization reduces words to their base forms (e.g., "running" → "run").

5. SPLITTING DATA INTO TRAINING AND TEST SETS

```
train_data, test_data = train_test_split(mbti_df, test_size=0.2, random_state=42, stratify=mbti_df.type)
```

- Splits the dataset into 80% training and 20% test data. Stratified sampling ensures that each personality type is represented proportionally in both sets.

6. VECTORIZATION USING TF-IDF

```
vectorizer = TfidfVectorizer(max_features=5000, stop_words='english')

train_post = vectorizer.transform(train_data.posts).toarray()

test_post = vectorizer.transform(test_data.posts).toarray()
```

- Transforms the posts into TF-IDF (Term Frequency-Inverse Document Frequency) vectors. It limits the features to the top 5000 most important words, ignoring common English stopwords.

7. ENCODING PERSONALITY TYPES

```
from sklearn.preprocessing import LabelEncoder

target_encoder = LabelEncoder()

train_target = target_encoder.fit_transform(train_data.type)

test_target = target_encoder.fit_transform(test_data.type)
```

- Encodes the 16 personality types into numerical values (0–15).

8. EVALUATING THE MODEL

1.Gaussian Naive Bayes Model

```
from sklearn.naive_bayes import GaussianNB

model_gnb = GaussianNB()

model_gnb.fit(train_post, train_target)

pred_gnb = model_gnb.predict(test_post)

pred_training_gnb = model_gnb.predict(train_post)
```

- Trains a Gaussian Naive Bayes classifier using the TF-IDF vectors and personality type labels.

- Predicts personality types on both training and test data.

```
print("The train accuracy score for model trained on Gaussian Naive Bayes Classifier is:", accuracy_score(train_target, pred_training_gnb))

print("The test accuracy score for model trained on Gaussian Naive Bayes Classifier is:", accuracy_score(test_target, pred_gnb))
```

- Outputs the accuracy of the model on both the training and test sets.
- **Classification Report**:

```
from sklearn.metrics import classification_report

personality_types = target_encoder.inverse_transform([i for i in range(16)])

print('Test classification report of Gaussian Naive Bayes\n', classification_report(test_target, pred_gnb, target_names=personality_types))
```

  - Provides precision, recall, F1-score, and support for each personality type.

2.Multinomial Naive Bayes Classifier

```
from sklearn.naive_bayes import MultinomialNB

model_mnb = MultinomialNB()

model_mnb.fit(train_post, train_target)

pred_mnb = model_mnb.predict(test_post)

pred_training_mnb = model_mnb.predict(train_post)
```

- **Model**: Multinomial Naive Bayes is suited for discrete features (e.g., word counts).
- **Training**: Trains on the TF-IDF vectorized training data (train_post) and train_target.
- **Prediction**: Predictions are made on both the training and test data.

```
print("The train accuracy score for model trained on Multinomial Naive Bayes Classifier is:", accuracy_score(train_target, pred_training_mnb))

print("The test accuracy score for model trained on Multinomial Naive Bayes Classifier is:", accuracy_score(test_target, pred_mnb))
```

- **Accuracy**: Prints the accuracy scores for the training and test data.
- **Classification Report**:

```
print('Test classification report of Multinomial Naive Bayes \n', classification_report(test_target, pred_mnb, target_names=personality_types))
```

  - Outputs precision, recall, F1-score, and support for each personality type.

3.Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier

model_rfc = RandomForestClassifier()
```

```
model_rfc.fit(train_post, train_target)

pred_rfc = model_rfc.predict(test_post)

pred_training_rfc = model_rfc.predict(train_post)
```

- **Model**: Random Forest is an ensemble method that builds multiple decision trees and aggregates their results.
- **Training and Prediction**: Trains the model and predicts for both training and test data.

```
print("The train accuracy score for model trained on Random Forest Classifier is:",
accuracy_score(train_target, pred_training_rfc))

print("The test accuracy score for model trained on Random Forest Classifier is:",
accuracy_score(test_target, pred_rfc))
```

- **Accuracy**: Prints accuracy for both training and test sets.
- **Classification Report**:

```
print('Test classification report of Random Forest Classifier\n', classification_report(test_target,
pred_rfc, target_names=personality_types))
```

4. XGBoost Classifier

```
from xgboost import XGBClassifier

model_xgb = XGBClassifier()

model_xgb.fit(train_post, train_target)

pred_xgb = model_xgb.predict(test_post)

pred_training_xgb = model_xgb.predict(train_post)
```

- **Model**: XGBoost is an advanced boosting technique that builds trees sequentially to correct errors from the previous ones.
- **Training and Prediction**: Trains and predicts similarly to the Random Forest.

```
print("The train accuracy score for model trained on XGBoost Classifier is:",
accuracy_score(train_target, pred_training_xgb))

print("The test accuracy score for model trained on XGBoost classifier is:",
accuracy_score(test_target, pred_xgb))
```

- **Accuracy**: Accuracy scores for training and test sets.
- **Classification Report**:

```
print('Test classification report of XGBoost Classifier\n', classification_report(test_target, pred_xgb,
target_names=personality_types))
```

5. LightGBM Classifier

```
from lightgbm import LGBMClassifier as lgb

model_lgb = lgb()

model_lgb.fit(train_post, train_target)
```

```
pred_lgb = model_lgb.predict(test_post)

pred_training_lgb = model_lgb.predict(train_post)
```

- **Model**: LightGBM is a gradient boosting algorithm optimized for speed and performance, especially with large datasets.
- **Training and Prediction**: Trains and predicts similarly to XGBoost.

```
print("The train accuracy score for model trained on LightGBM Classifier is:",
accuracy_score(train_target, pred_training_lgb))

print("The test accuracy score for model trained on LightGBM classifier is:",
accuracy_score(test_target, pred_lgb))
```

- **Classification Report**:

```
print('Test classification report of LightGBM Classifier\n', classification_report(test_target, pred_lgb,
target_names=personality_types))
```

6. Support Vector Machine (SVM) Classifier

```
from sklearn.svm import SVC

model_svc = SVC()

model_svc.fit(train_post, train_target)

pred_svc = model_svc.predict(test_post)

pred_training_svc = model_svc.predict(train_post)
```

- **Model**: Support Vector Classifier (SVC) is a powerful model that finds a hyperplane in an n-dimensional space that best separates classes.
- **Training and Prediction**: Trains and predicts for the test data.

```
print("The train accuracy score for model trained on Support Classifier is:",
accuracy_score(train_target, pred_training_svc))

print("The test accuracy score for model trained on Support Vector classifier is:",
accuracy_score(test_target, pred_svc))
```

- **Classification Report**:

```
print('Test classification report of Support Vector Machine\n', classification_report(test_target,
pred_svc, target_names=personality_types))
```

7. Logistic Regression Classifier

```
from sklearn.linear_model import LogisticRegression

model_lr = LogisticRegression()

model_lr.fit(train_post, train_target)

pred_lr = model_lr.predict(test_post)

pred_training_lr = model_lr.predict(train_post)
```

- **Model**: Logistic Regression is a linear model often used for classification problems.

- **Training and Prediction**: Similar process of training and prediction.

```
print("The train accuracy score for model trained on Logistic Regression is:",
accuracy_score(train_target, pred_training_lr))

print("The test accuracy score for model trained on Logistic Regression is:",
accuracy_score(test_target, pred_lr))
```

- **Classification Report**:

```
print('Test classification report of Logistic Regression\n', classification_report(test_target, pred_lr,
target_names=personality_types))
```

8. Comparing Results Across Models

```
result_df = pd.DataFrame({

   'Model': ["Gaussian NB", "Multinomial NB", "Random Forest", "XGBoost", "LightGBM", "SVM",
"Logistic Regresssion"],

   'Accuracy': [accuracy_score(test_target, pred_gnb), accuracy_score(test_target, pred_mnb),

            accuracy_score(test_target, pred_rfc), accuracy_score(test_target, pred_xgb),

            accuracy_score(test_target, pred_lgb), accuracy_score(test_target, pred_svc),

            accuracy_score(test_target, pred_lr)]

})

print(result_df.sort_values(by='Accuracy'))
```

- **Creating a DataFrame**: The model accuracies are compiled into a DataFrame for comparison.
- **Sorting**: The DataFrame is sorted by accuracy to see which model performs best.

CHAPTER 8

PERFORMANCE AND ANALYSIS

In this, we will evaluate and analyse the performance of different machine learning, examining classification reports, and discussing the strengths and weaknesses of each models used for personality prediction based on textual data. This includes comparing accuracy scores model.

1. Model Performance Metrics

The performance of each model is typically evaluated using the following metrics:

- **Accuracy**: The proportion of correct predictions made by the model out of all predictions.
- **Precision**: The ratio of correctly predicted positive observations to the total predicted positives. This measures the quality of positive predictions.
- **Recall (Sensitivity)**: The ratio of correctly predicted positive observations to all actual positives. This reflects the ability of the model to identify all relevant instances.
- **F1 Score**: The harmonic mean of precision and recall, providing a balance between the two. It is especially useful when dealing with imbalanced classes.
- **Confusion Matrix**: A table used to describe the performance of a classification model by comparing true values against predicted values.

2. Model Comparisons

The models evaluated in this project include:

1. **Gaussian Naive Bayes (GNB)**
    - **Accuracy**: 25.9%
    - **Analysis**: Simple and fast; however, GNB's assumption of feature independence limits its performance in complex datasets like personality prediction.

2. **Multinomial Naive Bayes (MNB)**
    - **Accuracy**: 38.0%
    - **Analysis**: Improved accuracy compared to GNB, particularly for text classification. However, it still struggles with capturing the nuances of the data.

3. **Random Forest Classifier**
    - **Accuracy**: 55.9%
    - **Analysis**: As an ensemble method, Random Forest handles overfitting well and provides decent accuracy, but it may still not be optimal for complex personality traits.

4. **Support Vector Machine (SVM)**
    - **Accuracy**: 65.4%
    - **Analysis**: SVM performs well with high-dimensional data, making it a good choice for text classification, though it may require careful tuning of parameters.

5. **Logistic Regression**
    - **Accuracy**: 65.0%
    - **Analysis**: Effective for binary classification but can also handle multi-class problems. Performs similarly to SVM in this context.

6. **XGBoost**
    - **Accuracy**: 67.5%
    - **Analysis**: A powerful gradient boosting algorithm that excels with structured data. XGBoost's handling of complex relationships in the data leads to improved performance.

7. **LightGBM**
    - **Accuracy**: 67.4%
    - **Analysis**: Similar to XGBoost, but optimized for speed and efficiency. Performs comparably well, though slightly less than XGBoost in this scenario.

3. Overall Analysis

- **Best Performing Models**: Based on accuracy, **XGBoost** emerged as the best model, closely followed by **LightGBM**. Both models demonstrated the capability to effectively capture the complexity of personality traits in textual data.

- **Model Limitations**: Despite the improvements in accuracy with ensemble methods like XGBoost and LightGBM, challenges remain:
    - **Imbalanced Classes**: Personality types are often imbalanced in datasets, affecting model performance. Addressing class imbalance through techniques like oversampling, undersampling, or synthetic data generation (e.g., SMOTE) may improve results.
    - **Feature Selection**: The quality of feature extraction directly impacts model performance. More advanced NLP techniques or additional contextual features could enhance prediction accuracy.
    - **Computational Resources**: More complex models like XGBoost and LightGBM require higher computational resources, which may be a limitation in environments with restricted resources.

4. Visual Representation

- **Confusion Matrix**: Visualizing the confusion matrix for the best-performing model (XGBoost) can provide insights into which personality types are most often confused with one another.
- **Classification Report**: Including precision, recall, and F1 scores for each personality type will provide a more comprehensive view of model performance.

The analysis of performance highlights that while traditional models like Naive Bayes provide a baseline, advanced ensemble methods such as XGBoost and LightGBM significantly enhance predictive accuracy in personality classification tasks. Continuous improvements in feature extraction and addressing class imbalance can further enhance model performance in future iterations of this project.

CHAPTER 9

CONCLUSION

This project aimed to classify Myers-Briggs Type Indicator (MBTI) personality types using textual data from online posts through the application of various machine learning models. After preprocessing the data, including text cleaning, stopword removal, and feature extraction using TF-IDF, multiple models were trained and evaluated for performance.

The models explored include Gaussian Naive Bayes, Multinomial Naive Bayes, Random Forest, Support Vector Machine (SVM), Logistic Regression, LightGBM, and XGBoost. Among these models, **XGBoost** achieved the highest test accuracy of **67.55%**, followed closely by LightGBM with **67.38%**.

On the other hand, simpler models like Gaussian Naive Bayes and Multinomial Naive Bayes performed poorly, highlighting that more complex algorithms are needed for this multi-class, text-based classification problem.

9.1 KEY TAKEAWAYS:

- **XGBoost** and **LightGBM** performed the best, demonstrating that ensemble and boosting methods are suitable for personality classification.
- **SVM** and **Logistic Regression** also provided relatively strong performance but lagged behind boosting methods.
- Text preprocessing and feature extraction, such as using **TF-IDF**, significantly contributed to the overall performance of the models.

9.2 FUTURE WORK:

- Incorporating **deep learning models**, such as recurrent neural networks (RNNs) or transformer-based models (e.g., BERT), could further enhance prediction accuracy by better capturing the sequential and contextual relationships within the text.
- Addressing the imbalanced nature of the dataset could improve performance for underrepresented personality types, possibly through techniques like data augmentation or synthetic data generation.
- Further exploration of advanced ensemble techniques and hyperparameter tuning could push the models toward higher accuracy and generalization.

SAMPLE SCREENSHOTS:

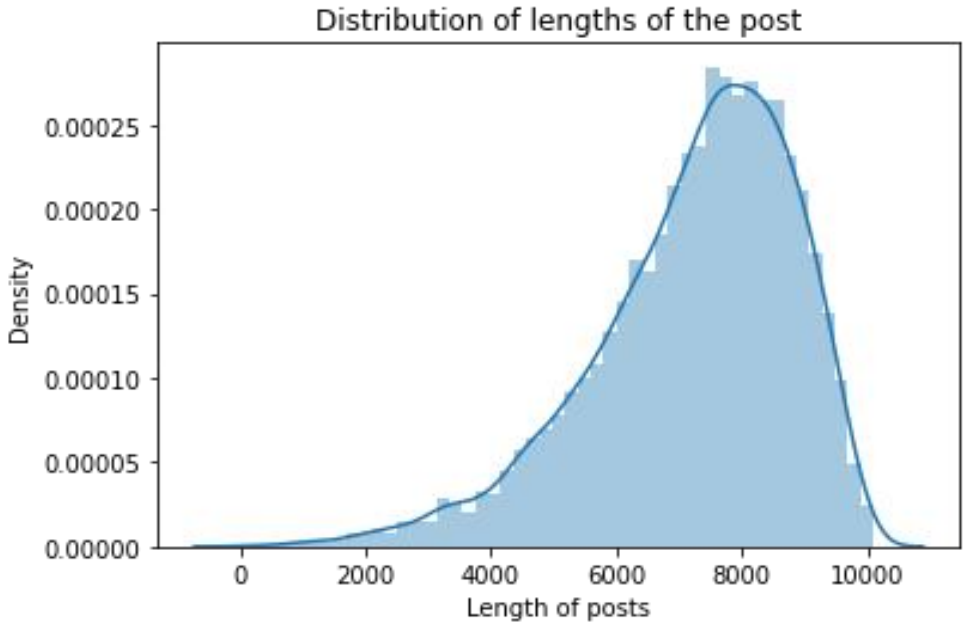

Fig.1.1 Distribution of length of the post

BAR GRAPH:

Frequency of different types of personaities

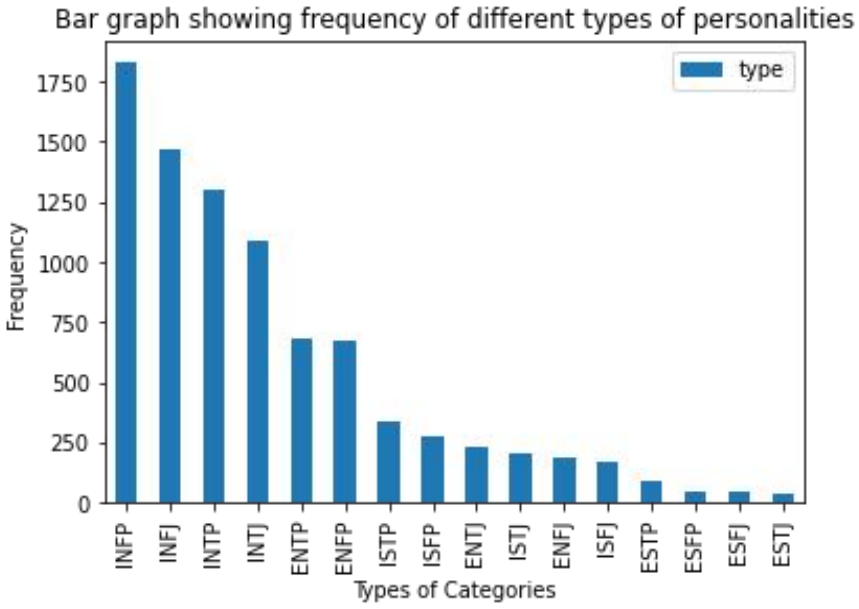

Fig.1.2 Bar graph showing different types of personalities

PIE CHART:

Ploting different types of personalities.

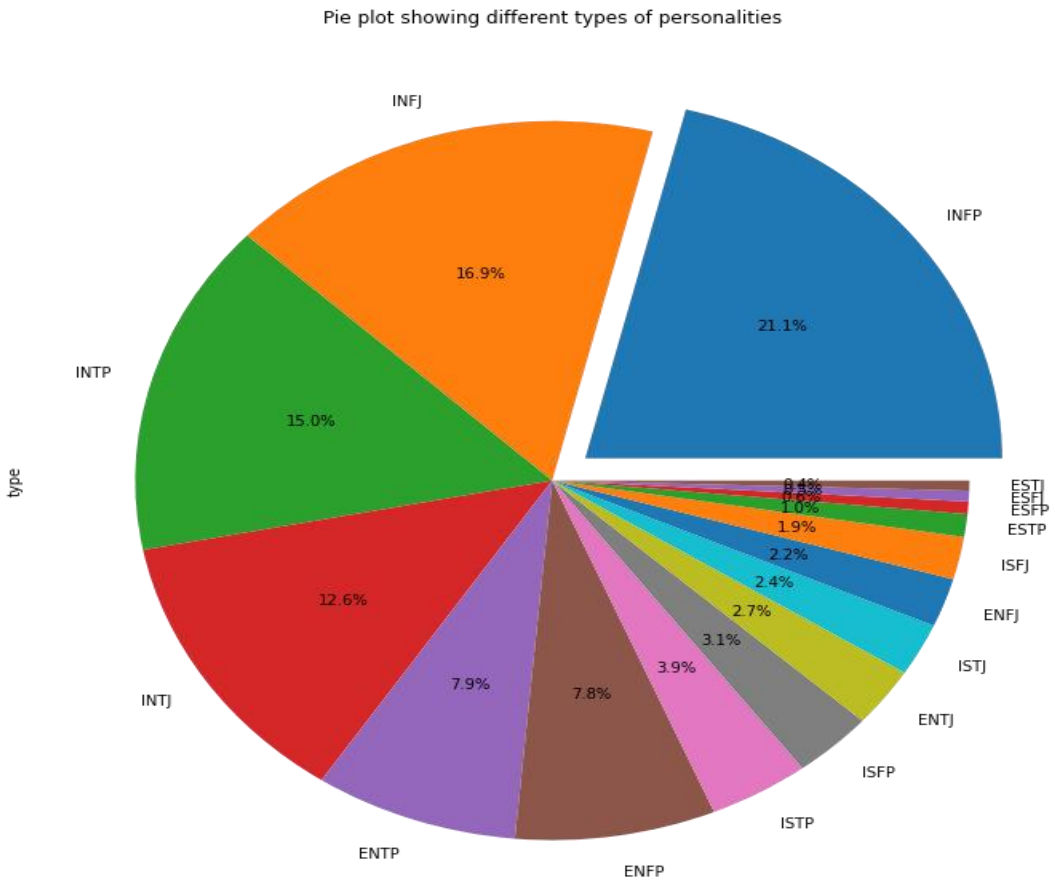

Fig.1.3 Pie chart showing different types of personalities

BAR GRAPH:

Comparison of different models used in the project for prediction and their accuracy.

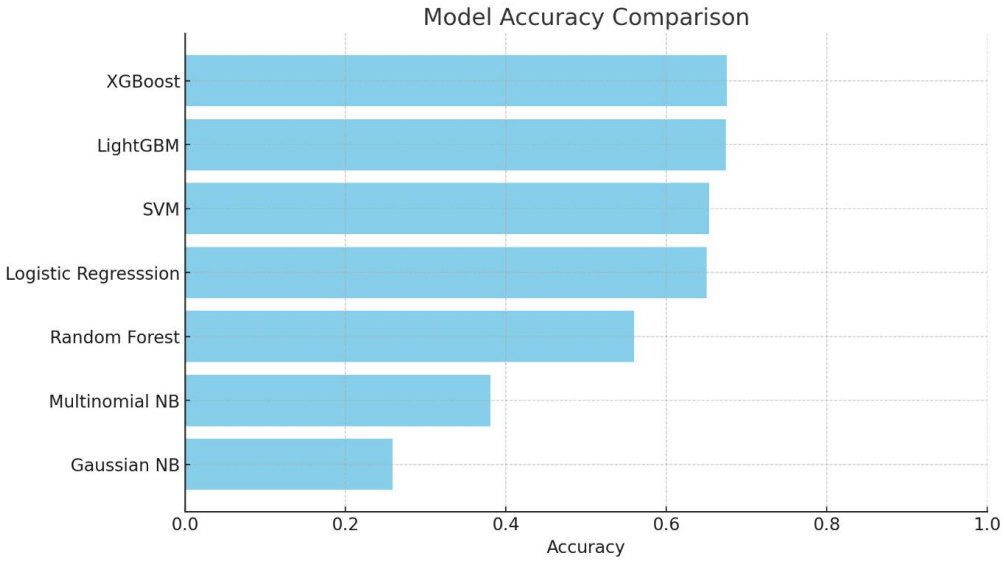

Fig.1.4 Bar graph for model accuracy comparison

OUTPUT:

Output of the given data by using the XGBoost algorithm.

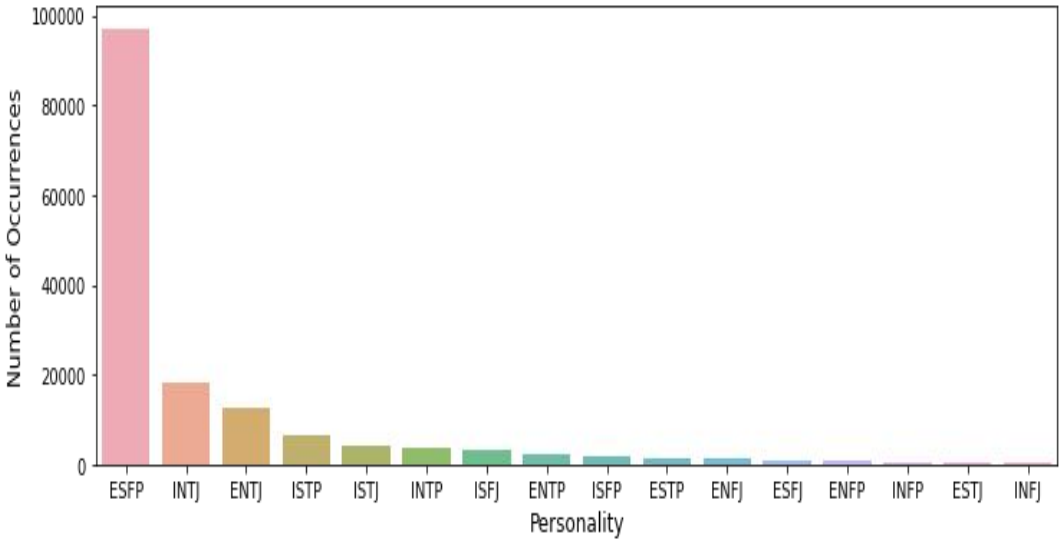

Fig.1.5 Final output of the analysis using XGBoost algorithm

REFERENCES

[1] **Faisal, L. (2024)** “Everyone is an Alien Somewhere: Investigating the Skills Needed for Effective Learning Advising” 21(9),3074.

[2] **Beckley, J. (2024)** “Personality prediction using medias” Do emotion motives constrain the selections of visual, auditory, and taste stimuli to up- and down- regulate emotions.[IEEE].

[3] **Meiling Li and Hezi Liu (2023)** “Language Style Matters: Personality Prediction from Textual Styles Learning” pp.49-67.

[4] **H. N. Desai and R. Patel (2020)** "A Study of Data Mining Methods for Prediction of Personality Traits," 2020 International Conference on Smart Electronics and Communication (ICOSEC), 2020, pp. 58- 64.

[5] **S. Wang, L. Cui, L. Liu, X. Lu and Q. Li (2020)**, “Personality Traits Prediction Based on Users: Digital Footprints in Social Networks via Attention RNN," pp. 78-92.

[6] **Meiling Li Hezi Liu** (2023) “Language Style Matters: Personality Prediction from Textual Styles Learning” [IEEE]

[7]. **Hima Vijay Neenu Sebastian (2020)** “Personality Prediction using Machine Learning” [IEEE]

[8]. **Huang Tao and Lin Zheng-Cha (2022)** “Non-operative Personality Prediction Based on Knowledge Driven” [IEEE]