Assessment Report

on

"Personality Prediction using MBTI Dataset"

submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY DEGREE

SESSION 2024-25

in

CSE(AI&ML)

By

Name : Shivanshi Dhyani(202401100400178)

Shreya Gupta(202401100400180)

Shreya Yadav(202401100400182)

Siddhant Tiwari(202401100400185)

Siddhi Gupta(202401100400186)

Section: C

Group:07

Under the supervision of

Abhishek Shukla Sir

KIET Group of Institutions, Ghaziabad

May, 2025

---

1. Introduction

As social media and online platforms generate vast amounts of user-generated content, analyzing written text to infer psychological traits has become an emerging research area. This project aims to classify personality types based on the Myers-Briggs Type Indicator (MBTI) using Natural Language Processing (NLP). Using a dataset of user posts, we explore how textual patterns reflect individual personality traits and train machine learning models for personality classification.

---

2. Problem Statement

To predict the MBTI personality type of users based solely on their written text. This involves building a multi-class classification model that understands and extracts meaningful linguistic features from text using NLP techniques.

---

3. Objectives

- Preprocess raw text data to remove noise and standardize format.
- Apply NLP techniques to extract features such as TF-IDF and embeddings.
- Train a classification model (e.g., Logistic Regression or Random Forest) to predict one of the 16 MBTI personality types.
- Evaluate the model's performance using accuracy and other classification metrics.
- Visualize key insights such as class distribution and word importance.

---

4. Methodology

- **Data Collection:**

  The dataset is obtained from Kaggle, containing over 8,000 user posts labeled with MBTI personality types.

- **Data Preprocessing:**

  Convert all text to lowercase.

  Remove URLs, stop words, punctuations, and special characters.

  Tokenize text and apply lemmatization.

- **Feature Extraction:**

  Use TF-IDF Vectorizer to convert textual data into numerical form.

  Alternative experiments include Word2Vec or BERT embeddings.

- **Model Building:**

  Split the dataset into training and testing sets (typically 80/20).

  Train multiple classifiers (e.g., Logistic Regression, Random Forest, SVM).

  Use one-vs-rest or one-vs-one strategy for multi-class classification.

- **Model Evaluation:**

  Measure performance using accuracy, precision, recall, F1-score.

  Visualize confusion matrix and top features contributing to each personality class.

---

5. Data Preprocessing

The preprocessing pipeline includes:

- Lowercasing and removing URLs, HTML tags, and punctuations.
- Tokenization and lemmatization using spaCy or NLTK.
- Stopword removal to retain meaningful terms.
- TF-IDF applied to generate feature vectors from cleaned text.
- Dataset is split: 80% for training, 20% for testing.

---

6. Model Implementation

Multiple models are explored for performance:

Logistic Regression: A baseline model for multi-class classification.

Random Forest Classifier: Captures non-linear relations in data.

Support Vector Machine (SVM): Suitable for high-dimensional feature space.

Experiments also conducted with deep learning models like LSTM or BERT for better semantic understanding.

---

7. Evaluation Metrics

The following metrics are used to evaluate the model:

- **Accuracy:** Percentage of correctly classified instances.
- **Precision, Recall, F1-score:** For class-wise performance analysis.
- **Confusion Matrix:** Helps identify misclassifications across the 16 MBTI types.
- **Macro and Weighted Averages:** To handle class imbalance.

---

8. Results and Analysis

- The Logistic Regression model achieved reasonable baseline performance.
- Advanced models (SVM, Random Forest) slightly improved classification scores.
- BERT-based models significantly outperformed traditional ML on semantic understanding.
- Class imbalance observed, especially in types like INFP and ISTP, affecting model generalization.
- Confusion matrix and word cloud analysis provided insights into type-specific language patterns.

---

9. Conclusion

This project demonstrates the feasibility of predicting MBTI personality types using written text and NLP. While traditional models offer a good starting point, incorporating advanced language models significantly boosts performance. Future improvements include data augmentation, addressing class imbalance, and using ensemble learning or fine-tuned transformer models.

---

10. References

- scikit-learn documentation
- pandas documentation

- NLTK documentation
- spaCy documentation
- Kaggle MBTI Dataset
- Research papers on personality prediction and linguistic analysis

---

Code:

```
# Step 1: Install required packages
!pip install nltk scikit-learn


# Step 2: Import libraries
import pandas as pd
import numpy as np
import re
import nltk
import matplotlib.pyplot as plt
import seaborn as sns


from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, accuracy_score
```

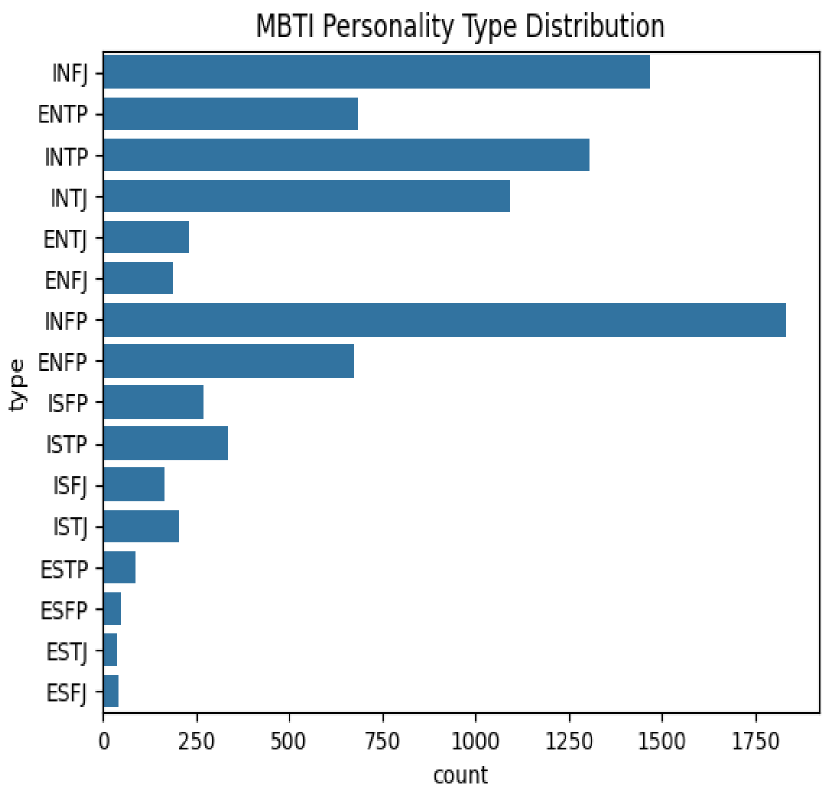
```python
nltk.download('stopwords')

from nltk.corpus import stopwords


# Step 3: Upload and unzip the dataset

from google.colab import files

uploaded = files.upload()


import zipfile

import os


# Unzip the uploaded file

with zipfile.ZipFile("mbti_1.csv.zip", 'r') as zip_ref:

    zip_ref.extractall()


# Check extracted files

print("Extracted files:", os.listdir())


# Step 4: Load the CSV file

df = pd.read_csv("mbti_1.csv")

df.head()


# Step 5: Basic preprocessing

print("Unique MBTI Types:", df['type'].unique())


# Check class distribution
```

```python
sns.countplot(y='type', data=df)

plt.title("MBTI Personality Type Distribution")

plt.show()


# Step 6: Clean the text

stop_words = set(stopwords.words('english'))


def clean_text(text):

    text = text.lower()

    text = re.sub(r'https?://\S+', '', text)  # remove URLs

    text = re.sub(r'[^a-z\s]', '', text)      # remove non-alphabetic

    text = re.sub(r'\s+', ' ', text)          # remove extra spaces

    words = text.split()

    words = [word for word in words if word not in stop_words]

    return ' '.join(words)


df['clean_text'] = df['posts'].apply(clean_text)


# Step 7: Vectorization using TF-IDF

vectorizer = TfidfVectorizer(max_features=5000)

X = vectorizer.fit_transform(df['clean_text'])


# Step 8: Encode target labels

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
```

```python
y = le.fit_transform(df['type'])


# Step 9: Train-test split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)


# Step 10: Train a classifier

model = LogisticRegression(max_iter=1000)

model.fit(X_train, y_train)


# Step 11: Evaluate

y_pred = model.predict(X_test)

print("Accuracy:", accuracy_score(y_test, y_pred))

print("Classification Report:")

print(classification_report(y_test, y_pred, target_names=le.classes_))
```

Output:

```
Requirement already satisfied: nltk in /usr/local/lib/python3.11/dist-packages (3.9.1)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.11/dist-packages (1.6.1)
Requirement already satisfied: click in /usr/local/lib/python3.11/dist-packages (from nltk) (8.2.0)
Requirement already satisfied: joblib in /usr/local/lib/python3.11/dist-packages (from nltk) (1.5.0)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.11/dist-packages (from nltk) (2024.11.6)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages (from nltk) (4.67.1)
Requirement already satisfied: numpy>=1.19.5 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (2.0.2)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.15.3)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (3.6.0)
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
Choose Files mbti_1.csv.zip
• mbti_1.csv.zip(application/x-zip-compressed) - 25621004 bytes, last modified: 5/27/2025 - 100% done
Saving mbti_1.csv.zip to mbti_1.csv.zip
Extracted files: ['.config', 'mbti_1.csv.zip', 'mbti_1.csv', 'sample_data']
Unique MBTI Types: ['INFJ' 'ENTP' 'INTP' 'INTJ' 'ENTJ' 'ENFJ' 'INFP' 'ENFP' 'ISFP' 'ISTP'
 'ISFJ' 'ISTJ' 'ESTP' 'ESFP' 'ESTJ' 'ESFJ']
```



```
Accuracy: 0.6334293948126801
Classification Report:
              precision    recall  f1-score   support

        ENFJ       0.56      0.12      0.20        41
        ENFP       0.71      0.56      0.62       125
        ENTJ       0.79      0.34      0.48        44
        ENTP       0.67      0.52      0.58       135
        ESFJ       0.00      0.00      0.00         7
        ESFP       0.00      0.00      0.00         8
        ESTJ       0.00      0.00      0.00         7
        ESTP       1.00      0.13      0.24        15
        INFJ       0.64      0.68      0.66       288
        INFP       0.60      0.86      0.71       370
        INTJ       0.62      0.70      0.66       193
        INTP       0.63      0.78      0.70       293
        ISFJ       0.91      0.22      0.36        45
        ISFP       0.69      0.21      0.32        53
        ISTJ       0.71      0.23      0.34        44
        ISTP       0.71      0.37      0.49        67

    accuracy                           0.63      1735
   macro avg       0.58      0.36      0.40      1735
weighted avg       0.64      0.63      0.61      1735
```