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(AIML)

By

Name	Roll No.
Dhruv Kumar	202401100400082
Harsh Jonwal	202401100400092
Harshit Kr Kushwaha	202401100400096
Mahadevan Pillai	202401100400115
Nikhil Chandra	202401100400126

Under the supervision of

"Abishek Shukla Sir"

KIET Group of Institutions, Ghaziabad

May, 2025

1. Introduction

With the growing usage of online platforms, users leave behind large volumes of written text, which can be analyzed to infer psychological traits. This project focuses on predicting users' Myers-Briggs Type Indicator (MBTI) personality types based on their social media text. Natural Language Processing (NLP) techniques are used to preprocess the data and classify user text into one of the 16 MBTI types.

2. Problem Statement

To build a classification model that predicts the MBTI personality type of a user based on their written text using NLP and machine learning techniques.

3. Objectives

- Preprocess user-written text to prepare it for analysis.
- Implement NLP techniques for tokenization, stopword removal, stemming, and vectorization.
- Train and evaluate machine learning models for personality classification.
- Use evaluation metrics to assess model performance.

4. Methodology

Data Collection:
The MBTI dataset was downloaded from Kaggle, consisting of 8600+ user posts and their respective MBTI personality types.

Data Preprocessing:

- Lowercasing all text.
- Removing URLs, special characters, and punctuations.
- Tokenization and stopword removal using NLTK.
- Lemmatization/Stemming to normalize words.
- TF-IDF vectorization to convert text into numerical form.

Model Building:

- Splitting the dataset into training and testing sets (80/20).
- Models used: Logistic Regression, Random Forest, and Naive Bayes.

Model Evaluation:

- Accuracy, precision, recall, and F1-score were calculated.
- Confusion matrix generated for model interpretability.

5. Data Preprocessing

Preprocessing included:

- Removing noise (e.g., hyperlinks and emojis).
- Applying lemmatization using SpaCy.
- Vectorizing with TF-IDF to reduce word frequency bias.
- Creating balanced datasets for each personality dimension (I/E, N/S, T/F, J/P).

6. Model Implementation

The text data was transformed using TF-IDF and fed into classification models. Logistic Regression and Random Forest yielded the best performance. Each MBTI dimension (I/E, N/S, T/F, J/P) was predicted separately using binary classification models.

7. Evaluation Metrics

- **Accuracy**: Overall classification correctness.
- **Precision**: Fraction of correct positive predictions.
- **Recall**: Fraction of actual positives correctly identified.
- **F1 Score**: Harmonic mean of precision and recall.
- **Confusion Matrix**: Visualized using Seaborn heatmap.

8. Results and Analysis

- Logistic Regression achieved ~70% accuracy on the I/E dimension.
- Random Forest performed better on N/S and T/F dimensions.
- Imbalanced class distribution impacted prediction performance.
- Visualization of confusion matrices showed clear classification patterns.

9. Conclusion

This project demonstrates how natural language can reveal psychological traits. The models developed can reasonably predict MBTI types using written text. While results are promising, further improvements can be made using deep learning models such as BERT and LSTM, and by handling class imbalance more effectively.

10. References

- MBTI Dataset on Kaggle
- scikit-learn documentation
- NLTK and SpaCy libraries
- Research articles on NLP-based personality prediction
- Seaborn visualization library

```
# Step 1: Install required libraries
!pip install -q scikit-learn pandas matplotlib seaborn nltk

# Step 2: Import libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix,
ConfusionMatrixDisplay
from sklearn.feature_extraction.text import TfidfVectorizer
import nltk
import re
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from google.colab import files
import zipfile
import io

# Step 3: Download stopwords
nltk.download('stopwords')
stop_words = set(stopwords.words('english'))

# Step 4: Upload dataset
print("Upload your zipped MBTI dataset (e.g., mbti_1.csv.zip)")
uploaded = files.upload()
```

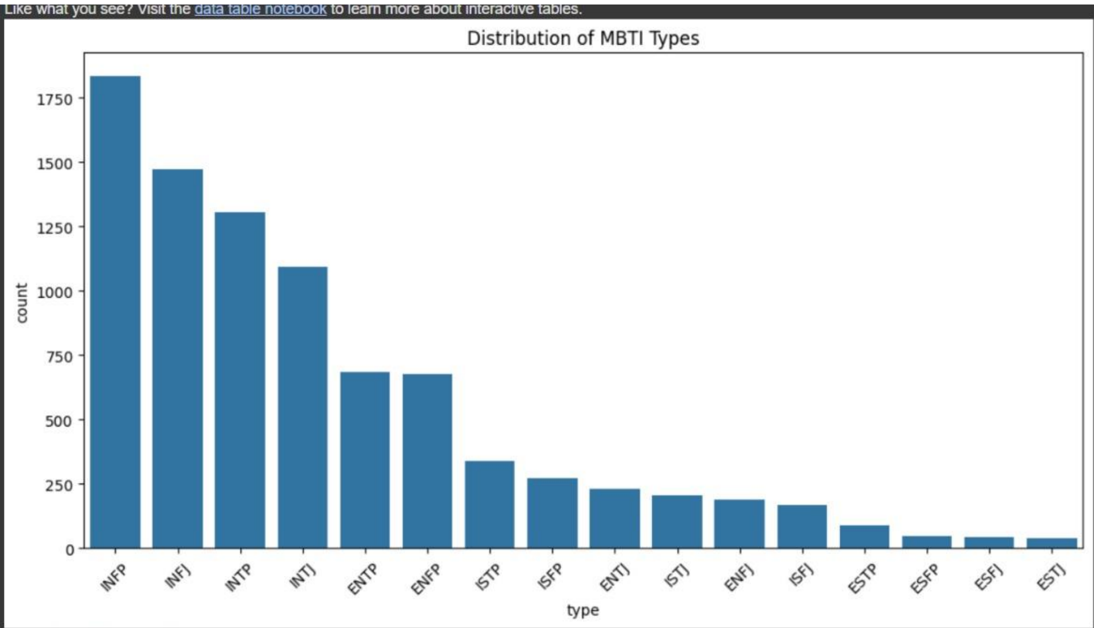
```
# Step 5: Unzip and load the CSV file
for filename in uploaded.keys():
    if filename.endswith('.zip'):
        with zipfile.ZipFile(io.BytesIO(uploaded[filename]), 'r') as zip_ref:
            zip_ref.extractall("mbti_data")

df = pd.read_csv("mbti_data/mbti_1.csv")
print("Dataset loaded. First few rows:")
display(df.head())

# Step 6: Visualize MBTI distribution
plt.figure(figsize=(12,6))
sns.countplot(data=df, x='type', order=df['type'].value_counts().index)
plt.title("Distribution of MBTI Types")
plt.xticks(rotation=45)
plt.show()

# Step 7: Preprocess text
stemmer = PorterStemmer()

def preprocess_text(text):
    text = text.lower()
    text = re.sub(r"http\S+|www\S+|https\S+", '', text)
    text = re.sub(r'[^a-z\s]', '', text)
    tokens = text.split()
    tokens = [word for word in tokens if word not in stop_words]
    tokens = [stemmer.stem(word) for word in tokens]
    return ' '.join(tokens)
```

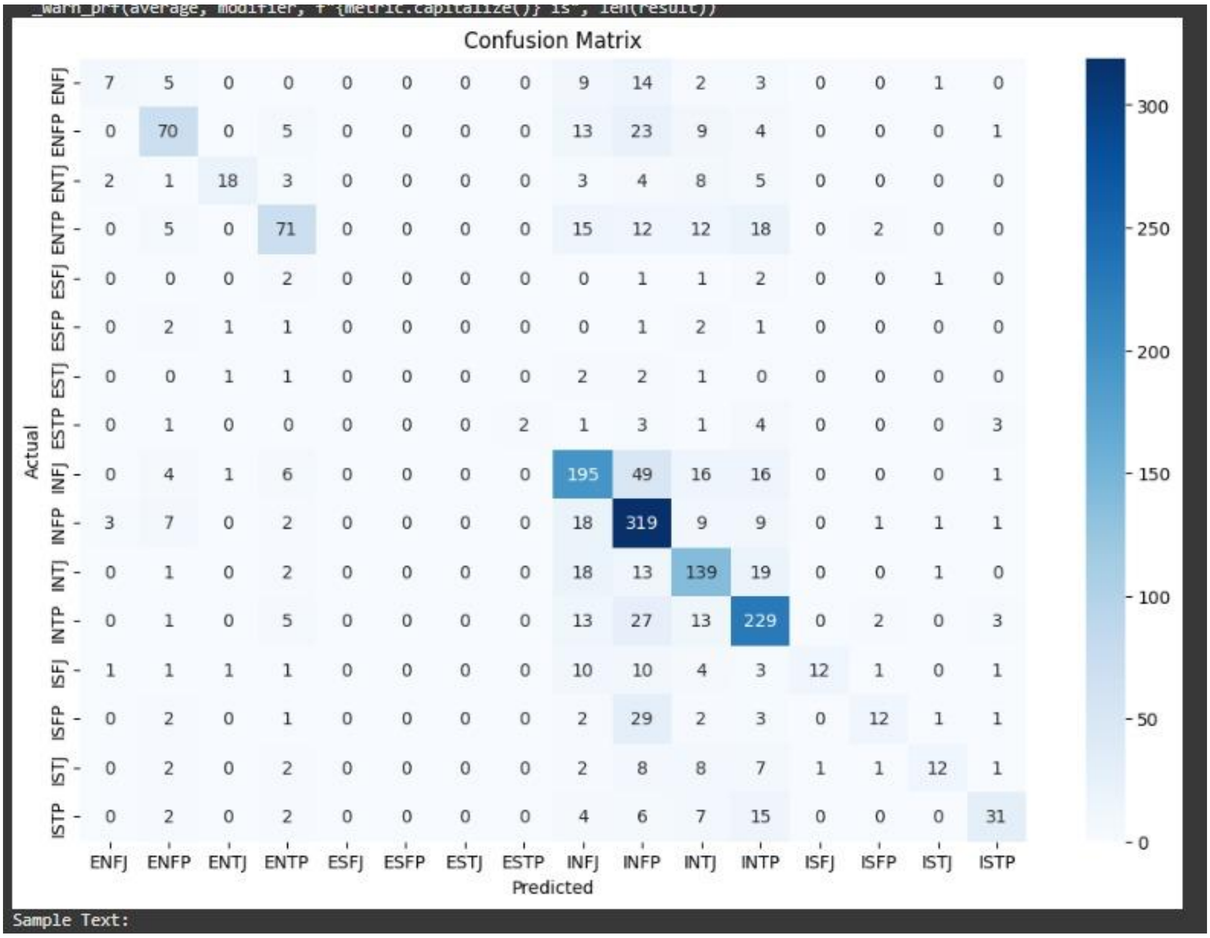
```
df['cleaned_posts'] = df['posts'].apply(preprocess_text)

# Step 8: Encode labels
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['type_code'] = le.fit_transform(df['type'])

# Step 9: Vectorize text
vectorizer = TfidfVectorizer(max_features=5000)
X = vectorizer.fit_transform(df['cleaned_posts'])
y = df['type_code']

# Step 10: Train/test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Step 11: Train model
model = LogisticRegression(max_iter=300)
model.fit(X_train, y_train)

# Step 12: Evaluate model
y_pred = model.predict(X_test)

# Classification Report
print("Classification Report:\n")
print(classification_report(y_test, y_pred, target_names=le.classes_))

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
```

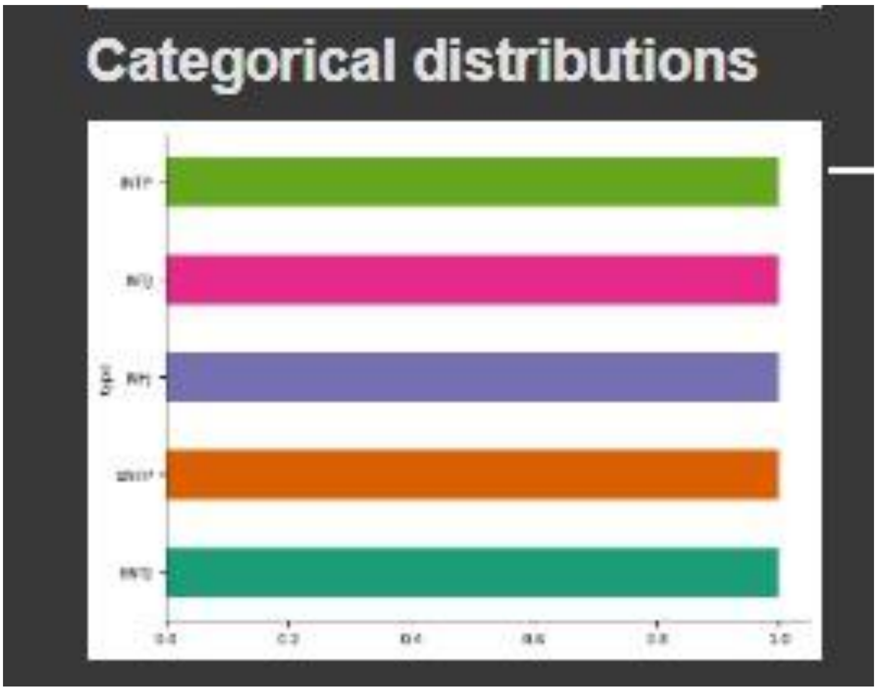
```
plt.figure(figsize=(12, 8))
sns.heatmap(cm, annot=True, fmt='d', xticklabels=le.classes_,
yticklabels=le.classes_, cmap="Blues")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()

# Step 13: Predict a new sample
def predict_mbti(text):
    text = preprocess_text(text)
    vector = vectorizer.transform([text])
    pred = model.predict(vector)
    return le.inverse_transform(pred)[0]

# Example prediction
sample_text = "I enjoy helping people and discussing theories about the
universe."
print("Sample Text:\n", sample_text)
print("Predicted MBTI type:", predict_mbti(sample_text))
```

```
Classification Report:

              precision    recall  f1-score   support

        ENFJ       0.54      0.17      0.26        41
        ENFP       0.67      0.56      0.61       125
        ENTJ       0.82      0.41      0.55        44
        ENTP       0.68      0.53      0.59       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       0.61      0.86      0.72       370
        INTJ       0.59      0.72      0.65       193
        INTP       0.68      0.78      0.73       293
        ISFJ       0.92      0.27      0.41        45
        ISFP       0.63      0.23      0.33        53
        ISTJ       0.71      0.27      0.39        44
        ISTP       0.72      0.46      0.56        67

    accuracy                           0.64      1735
   macro avg       0.58      0.38      0.42      1735
weighted avg       0.65      0.64      0.62      1735
```

Like what you see? Visit the data table notebook to learn more about interactive tables.




Confusion Matrix
Actual
Predicted
ENFJ
ENFP
ENTJ
ENTP
ESFJ
ESFP
ESTJ
ESTP
INFJ
INFP
INTJ
INTP
ISFJ
ISFP
ISTJ
ISTP
Sample Text:


Categorical distributions