Assessment Report

on

"MBTI Personality Type Prediction using NLP"

submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY DEGREE

SESSION 2024-25

in

CSE(AI)-B

By

Group No. 17

Ashish Maurya

Harsh Srivastava

Kumar Aryan

Kushal Verma

Under the supervision of

"Shivansh Prasad"

KIET Group of Institutions, Ghaziabad

1. Introduction

With the increasing availability of user-generated content on social media, analyzing personality traits using written text has become a fascinating research area in Natural Language Processing (NLP). This project focuses on predicting the MBTI (Myers-Briggs Type Indicator) personality types of individuals based on their writing style. Using a dataset of user posts and machine learning models, the project aims to automatically classify a user's MBTI type using NLP techniques.

---

2. Problem Statement

To classify a user's MBTI personality type (e.g., INTP, ENFJ, ISTJ) based on their written text using NLP and supervised machine learning models. This can be helpful for psychological analysis, personalized recommendations, and HR assessments.

---

3. Objectives

● Preprocess raw text data from social media posts for model training.

● Convert textual data into numerical format using TF-IDF.

● Train a Logistic Regression classifier to predict personality types.

● Evaluate model performance using standard classification metrics.

● Visualize prediction results using a confusion matrix heatmap.

---

4. Methodology

● Data Collection:

- The dataset was obtained from Kaggle, containing MBTI types and users' posts.

● Data Preprocessing:

- Removing URLs, punctuation, and stopwords.
- Converting text to lowercase and applying stemming.
- Combining multiple posts into one per user.

● Feature Extraction:

- Applying TF-IDF vectorization to convert text into feature vectors.

● Model Building:

- Encoding MBTI labels numerically (0–15).
- Splitting data into training and test sets.
- Training Logistic Regression on vectorized data.

● Model Evaluation:

- Calculating accuracy, precision, recall, and F1-score.
- Creating and visualizing a confusion matrix with Seaborn heatmap.

---

5. Data Preprocessing

The text data was cleaned and prepared using the following steps:

● Converted text to lowercase.

● Removed hyperlinks, special characters, and punctuation.

● Removed common English stopwords (like "the", "and", etc.).

● Applied stemming to reduce words to their root forms.

● Transformed cleaned text into numerical vectors using TF-IDF.

---

6. Model Implementation

Logistic Regression was chosen for its effectiveness in multi-class classification and interpretability. The model was trained on TF-IDF features extracted from the processed user posts and used to predict MBTI types on the test set.

---

7. Evaluation Metrics

The performance of the model was evaluated using:

● **Accuracy** – Correct predictions over total predictions.

● **Precision** – Proportion of correct positive predictions.

● **Recall** – Proportion of actual positives correctly predicted.

● **F1 Score** – Harmonic mean of precision and recall.

● **Confusion Matrix** – Used to visualize classification results using a heatmap.

---

8. Results and Analysis

● The Logistic Regression model showed reasonable accuracy in predicting MBTI types.

● The confusion matrix showed how the model performed across 16 classes.

● Certain personality types were more frequently misclassified due to textual similarity.

● Precision and recall scores highlighted how well the model detected specific MBTI types.

---

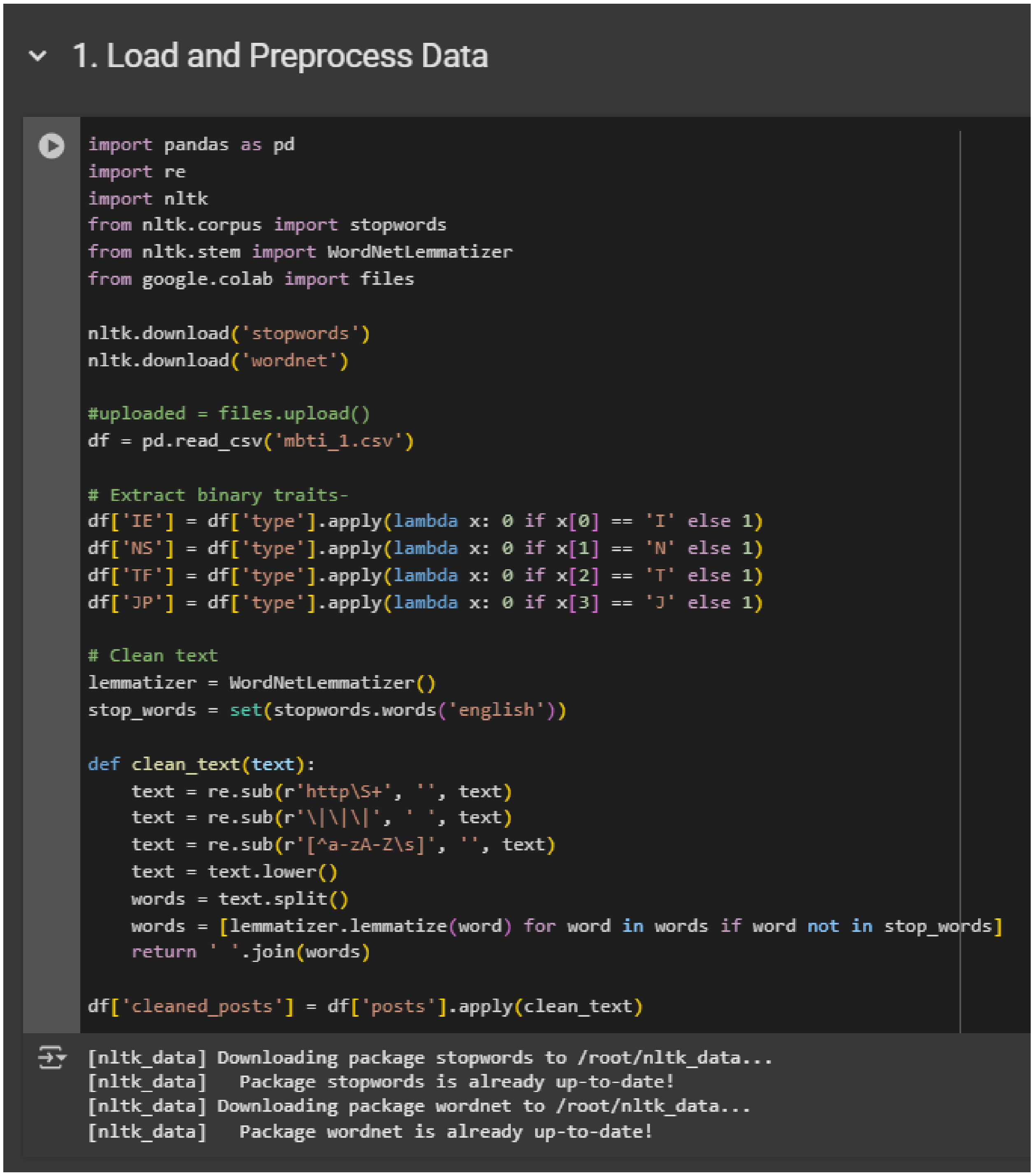
9. Conclusion

This project demonstrates the potential of using NLP and machine learning to classify personality types based on written text. Despite being a simple baseline using Logistic Regression, the results were promising. Further improvements can include deep learning models like LSTM or BERT and addressing class imbalance in the dataset for better accuracy.

---

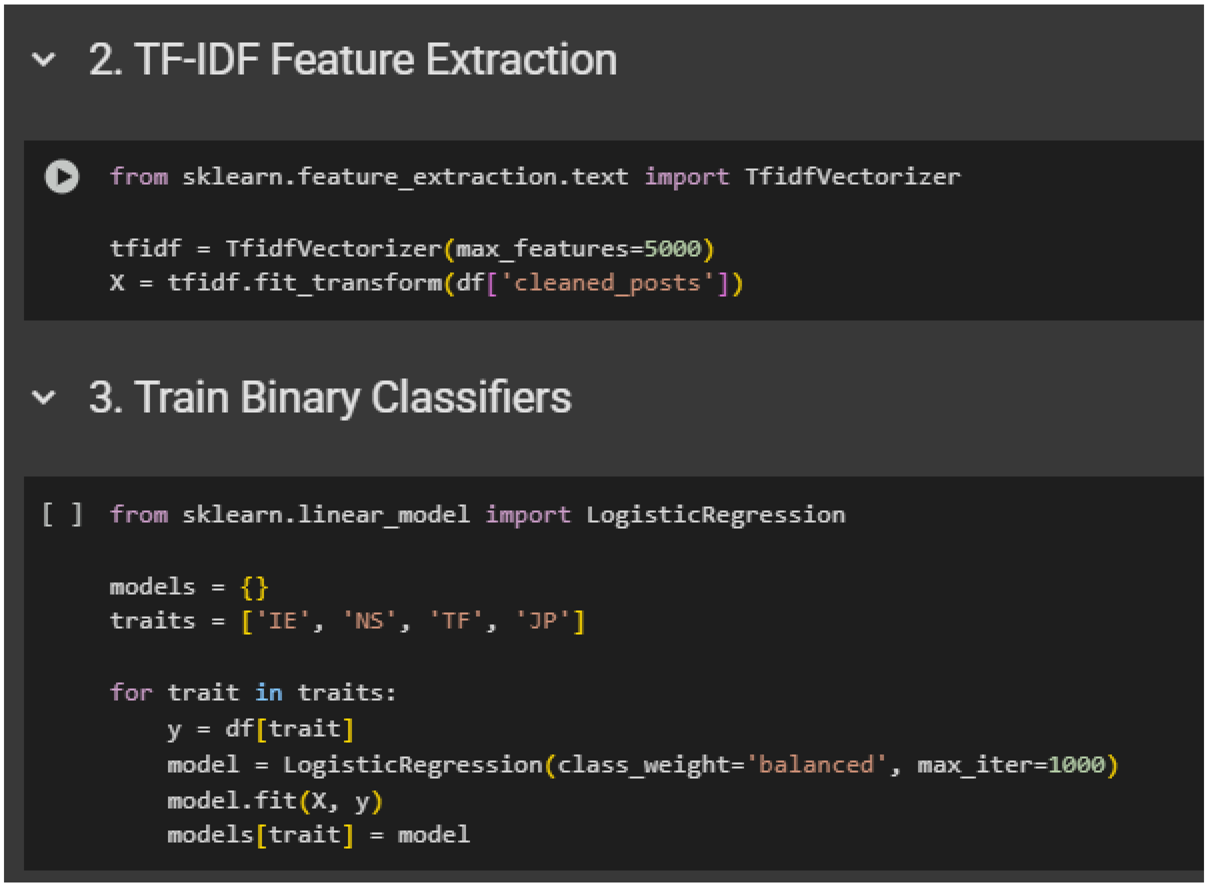
10. References

● scikit-learn documentation
● pandas documentation
● Seaborn visualization library
● Kaggle MBTI Dataset
● Research papers on MBTI and NLP


1. Load and Preprocess Data

```
import pandas as pd
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from google.colab import files

nltk.download('stopwords')
nltk.download('wordnet')

#uploaded = files.upload()
df = pd.read_csv('mbti_1.csv')

# Extract binary traits-
df['IE'] = df['type'].apply(lambda x: 0 if x[0] == 'I' else 1)
df['NS'] = df['type'].apply(lambda x: 0 if x[1] == 'N' else 1)
df['TF'] = df['type'].apply(lambda x: 0 if x[2] == 'T' else 1)
df['JP'] = df['type'].apply(lambda x: 0 if x[3] == 'J' else 1)

# Clean text
lemmatizer = WordNetLemmatizer()
stop_words = set(stopwords.words('english'))

def clean_text(text):
    text = re.sub(r'http\S+', '', text)
    text = re.sub(r'\|\|\|', ' ', text)
    text = re.sub(r'[^a-zA-Z\s]', '', text)
    text = text.lower()
    words = text.split()
    words = [lemmatizer.lemmatize(word) for word in words if word not in stop_words]
    return ' '.join(words)

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

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

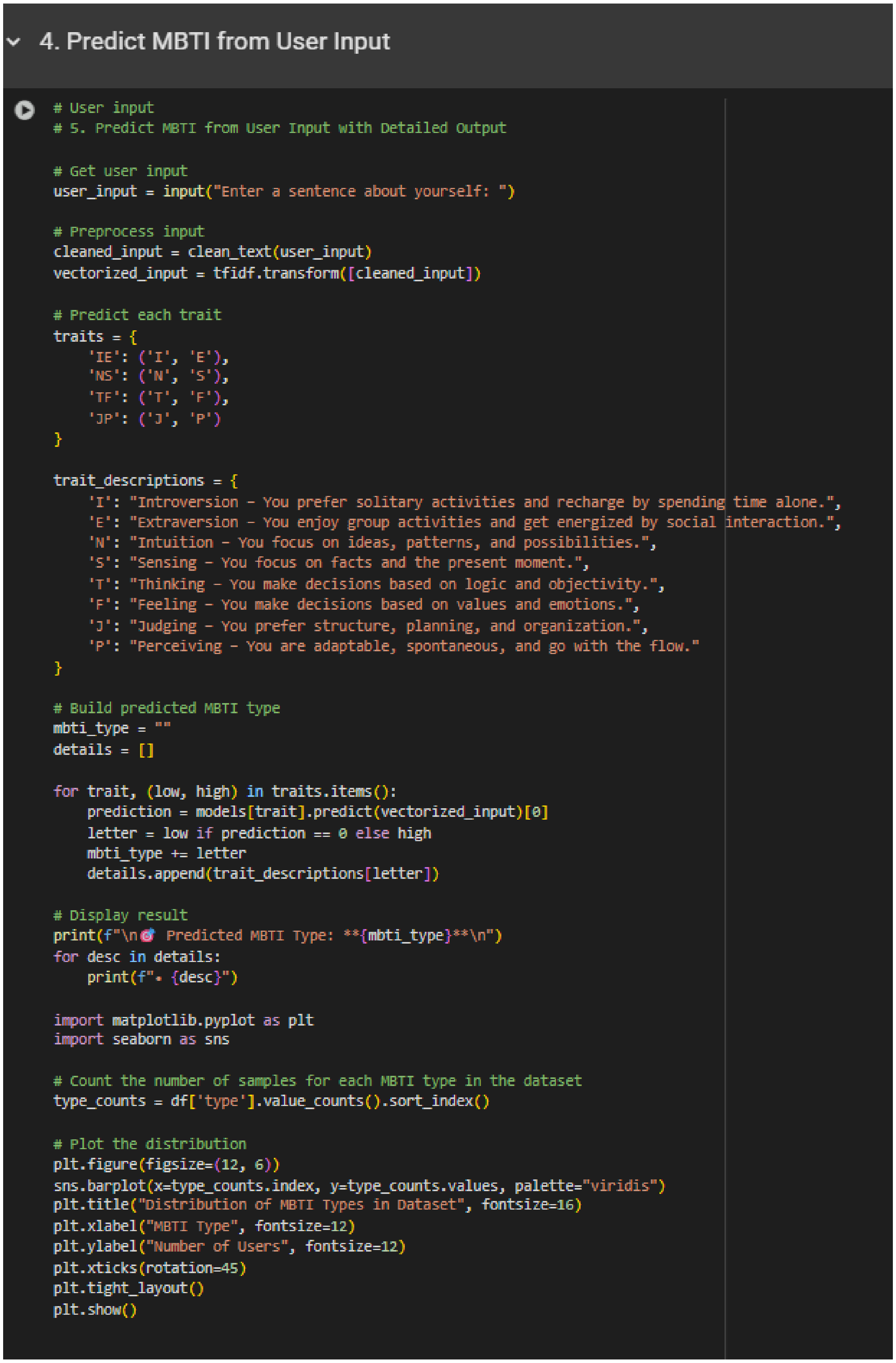


2. TF-IDF Feature Extraction

```python
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf = TfidfVectorizer(max_features=5000)
X = tfidf.fit_transform(df['cleaned_posts'])
```

3. Train Binary Classifiers

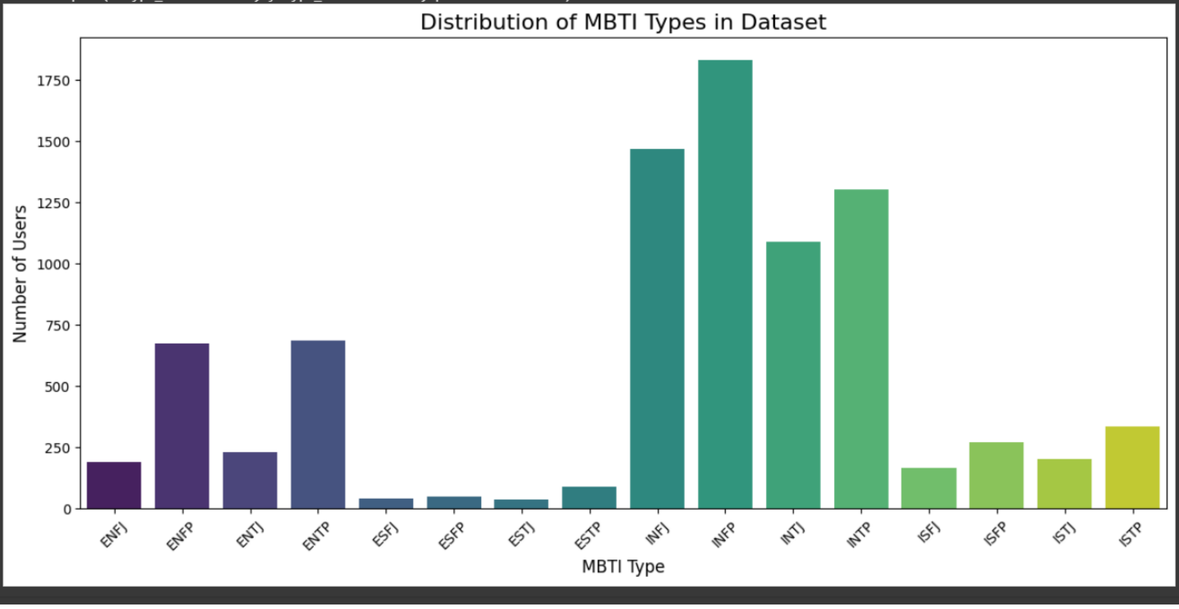
```python
from sklearn.linear_model import LogisticRegression

models = {}
traits = ['IE', 'NS', 'TF', 'JP']

for trait in traits:
    y = df[trait]
    model = LogisticRegression(class_weight='balanced', max_iter=1000)
    model.fit(X, y)
    models[trait] = model
```



4. Predict MBTI from User Input

```python
# User input
# 5. Predict MBTI from User Input with Detailed Output

# Get user input
user_input = input("Enter a sentence about yourself: ")

# Preprocess input
cleaned_input = clean_text(user_input)
vectorized_input = tfidf.transform([cleaned_input])

# Predict each trait
traits = {
    'IE': ('I', 'E'),
    'NS': ('N', 'S'),
    'TF': ('T', 'F'),
    'JP': ('J', 'P')
}

trait_descriptions = {
    'I': "Introversion – You prefer solitary activities and recharge by spending time alone.",
    'E': "Extraversion – You enjoy group activities and get energized by social interaction.",
    'N': "Intuition – You focus on ideas, patterns, and possibilities.",
    'S': "Sensing – You focus on facts and the present moment.",
    'T': "Thinking – You make decisions based on logic and objectivity.",
    'F': "Feeling – You make decisions based on values and emotions.",
    'J': "Judging – You prefer structure, planning, and organization.",
    'P': "Perceiving – You are adaptable, spontaneous, and go with the flow."
}

# Build predicted MBTI type
mbti_type = ""
details = []

for trait, (low, high) in traits.items():
    prediction = models[trait].predict(vectorized_input)[0]
    letter = low if prediction == 0 else high
    mbti_type += letter
    details.append(trait_descriptions[letter])

# Display result
print(f"\n🎯 Predicted MBTI Type: **{mbti_type}**\n")
for desc in details:
    print(f"• {desc}")

import matplotlib.pyplot as plt
import seaborn as sns

# Count the number of samples for each MBTI type in the dataset
type_counts = df['type'].value_counts().sort_index()

# Plot the distribution
plt.figure(figsize=(12, 6))
sns.barplot(x=type_counts.index, y=type_counts.values, palette="viridis")
plt.title("Distribution of MBTI Types in Dataset", fontsize=16)
plt.xlabel("MBTI Type", fontsize=12)
plt.ylabel("Number of Users", fontsize=12)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

```
Enter a sentence about yourself: I love party

🎯 Predicted MBTI Type: **ESFJ**

• Extraversion – You enjoy group activities and get energized by social interaction.
• Sensing – You focus on facts and the present moment.
• Feeling – You make decisions based on values and emotions.
• Judging – You prefer structure, planning, and organization.
<ipython-input-31-519ed3b1a604>:53: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same ef

  sns.barplot(x=type_counts.index, y=type_counts.values, palette="viridis")
```



SS