TITLE OF PROJECT REPORT

PERSONALITY PREDICTION

A PROJECT REPORT

Submitted by:

Rahul Kumar Gupta(202401100300191)

Mohammad Dilshad(202401100300153)

Sachin Kumar(202401100300208)

Sachin kumar(202401100300207)

Nikhil Kumar(202401100300160)

CSEAI-C

KIET GROUP OF INSTITUTION

Introduction: Personality Prediction Using NLP

In the era of digital communication, people frequently express their thoughts, feelings, and preferences through written text on social media, blogs, forums, and chat platforms. These textual footprints carry significant clues about an individual's personality traits. Understanding personality from text has numerous applications, including personalized marketing, recruitment, mental health assessment, and recommender systems.

This project leverages **Natural Language Processing (NLP)** techniques to classify a user's MBTI personality type from their written text. By analyzing language patterns, word choices, and semantic structures, it becomes possible to make accurate predictions about a person's psychological characteristics.

The system involves several key steps, including:

- **Text preprocessing** (cleaning and normalizing data),
- **Feature extraction** (using TF-IDF or transformer-based embeddings),
- **Model training** (with machine learning or deep learning algorithms), and
- **Prediction** of personality labels.

Methodology

The methodology for predicting personality types from text involves a pipeline of processes that transform raw textual data into personality predictions using Natural Language Processing (NLP) and Machine Learning techniques.

Data Collection

The dataset used contains written text samples from individuals labeled with their MBTI (Myers-Briggs Type Indicator) personality types. A commonly used dataset is the **MBTI Kaggle dataset**, which includes thousands of text posts with corresponding MBTI labels

Each data entry includes:

- **Text**: A collection of user-generated posts.
- **Label**: One of the 16 MBTI personality types

Text Preprocessing

Raw text data is often noisy and unstructured. Preprocessing ensures the input is clean, normalized, and ready for feature extraction.

Steps include:

- **Lowercasing**: Convert all text to lowercase.
- **Removing URLs, mentions, and special characters**: Strip out irrelevant elements like hyperlinks and usernames.

Feature Extraction

The processed text is converted into a numerical format suitable for machine learning algorithms.

Two approaches are commonly used:

- **TF-IDF Vectorization**: Measures the importance of words in a document relative to a corpus.
- **Transformer Embeddings**: Uses pre-trained models like BERT

- **Label Encoding**
- The MBTI personality types are categorical labels. These are encoded into numerical values using **Label Encoding** or **One-Hot Encoding** to make them compatible with classification models.
- Optionally, the prediction task can be broken down into **four binary classification tasks**, one for each personality dimension

Model Training

The extracted features and encoded labels are used to train machine learning models. Several algorithms can be employed:

- **Baseline Models**: Logistic Regression, Support Vector Machines (SVM), Random Forest.
- **Deep Learning Models**: LSTM, GRU, or Transformer-based models for capturing sequential patterns in text.

Code Typed

```
#Import necessary libraries
import pandas as pd  # For reading and handling dataset
import re  # For regular expression (text cleaning)
import matplotlib.pyplot as plt  # For plotting graphs
import seaborn as sns  # For improved graph styling

# Machine Learning libraries
from sklearn.model_selection import train_test_split  # To split dataset
from sklearn.feature_extraction.text import TfidfVectorizer  # To convert text to numbers
from sklearn.linear_model import LogisticRegression  # Classification model
from sklearn.metrics import classification_report, accuracy_score  # For evaluating the model

#Step 1: Load the MBTI dataset
# Ensure 'mbti_1.csv' is in the same directory as your Python file
data = pd.read_csv("/content/mbti_1.csv")

# Display the first 5 rows of the dataset
print("First 5 entries in the dataset:")
print(data.head())

#Plot 1: Show how many users belong to each MBTI personality type
plt.figure(figsize=(10, 6))  # Set figure size
sns.countplot(data['type'], order=data['type'].value_counts().index, palette='Set2')
plt.title("MBTI Personality Type Distribution")
```

```
plt.xlabel("Personality Type")

plt.ylabel("Number of Users")

plt.xticks(rotation=45)

plt.tight_layout()

plt.show()


#Step 2: Clean the text using a function

def clean_text(text):

    text = text.lower()  # Convert all characters to lowercase

    text = re.sub(r"http\S+", "", text)  # Remove URLs

    text = re.sub(r"[^a-z\s]", "", text)  # Remove special characters and numbers

    return text


# Apply the cleaning function to all posts

data['cleaned'] = data['posts'].apply(clean_text)


#Step 3: Create binary labels for each MBTI letter

# Example: Type "INTJ" → ['I', 'N', 'T', 'J']

# We classify each letter separately


# If first letter is I (Introvert), label 0; else E (Extrovert), label 1

data['IE'] = data['type'].apply(lambda x: 0 if x[0] == 'I' else 1)

data['NS'] = data['type'].apply(lambda x: 0 if x[1] == 'N' else 1)

data['TF'] = data['type'].apply(lambda x: 0 if x[2] == 'T' else 1)

data['JP'] = data['type'].apply(lambda x: 0 if x[3] == 'J' else 1)
```

```
#Step 4: Convert cleaned text into numeric features using TF-IDF
# TF-IDF (Term Frequency–Inverse Document Frequency) gives importance to words
vectorizer = TfidfVectorizer(max_features=3000)  # Limit to top 3000 words
X = vectorizer.fit_transform(data['cleaned'])  # Fit and transform text

# Create output labels (target variables)
y = data[['IE', 'NS', 'TF', 'JP']]

#Step 5: Split data into training and testing sets
# 80% for training, 20% for testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

#Step 6: Train separate Logistic Regression models for each MBTI trait
models = {}  # Dictionary to hold models
accuracy_scores = {}  # To store accuracy of each trait model

# Loop over each MBTI trait
for trait in ['IE', 'NS', 'TF', 'JP']:
    print(f"\n� Training model for trait: {trait}")
    model = LogisticRegression(max_iter=1000)  # Create model
    model.fit(X_train, y_train[trait])  # Train on respective trait
    models[trait] = model  # Save the model

    # Predict on test data
```

```
    y_pred = model.predict(X_test)
    # Calculate and save accuracy
    acc = accuracy_score(y_test[trait], y_pred)
    accuracy_scores[trait] = acc

    # Show detailed classification report
    print("Classification Report:")
    print(classification_report(y_test[trait], y_pred))

#Plot 2: Visualize accuracy of each trait prediction
plt.figure(figsize=(6, 4))
sns.barplot(x=list(accuracy_scores.keys()), y=list(accuracy_scores.values()), palette='pastel')
plt.title("Accuracy for Each MBTI Trait Model")
plt.xlabel("MBTI Trait")
plt.ylabel("Accuracy")
plt.ylim(0, 1)
plt.tight_layout()
plt.show()

#Step 7: Create a function to predict MBTI type from new text
def predict_mbti(text):
    cleaned = clean_text(text)  # Clean the input
    features = vectorizer.transform([cleaned])  # Convert to TF-IDF vector
    result = ""
    result += "I" if models['IE'].predict(features)[0] == 0 else "E"
```

```
    result += "N" if models['NS'].predict(features)[0] == 0 else "S"

    result += "T" if models['TF'].predict(features)[0] == 0 else "F"

    result += "J" if models['JP'].predict(features)[0] == 0 else "P"

    return result


#Step 8: Try predicting on a new sample text

sample_text = "I enjoy thinking deeply about abstract ideas and prefer calm environments."

predicted_type = predict_mbti(sample_text)

print("\nPredicted MBTI Type for Sample Text:")

print(predicted_type)
```

Output:

```
First 5 rows of data:
   type                                              posts
0  INFJ  'http://www.youtube.com/watch?v=qsXHcwe3krw|||...
1  ENTP  'I'm finding the lack of me in these posts ver...
2  INTP  'Good one  _____   https://www.youtube.com/wat...
3  INTJ  'Dear INTP,   I enjoyed our conversation the o...
4  ENTJ  'You're fired.|||That's another silly misconce...

Training model for IE
Results for IE
              precision    recall  f1-score   support

           0       0.81      0.98      0.89      1335
           1       0.79      0.25      0.38       400

    accuracy                           0.81      1735
   macro avg       0.80      0.61      0.63      1735
weighted avg       0.81      0.81      0.77      1735


Training model for NS
Results for NS
              precision    recall  f1-score   support

           0       0.89      0.99      0.94      1522
           1       0.71      0.09      0.17       213

    accuracy                           0.88      1735
   macro avg       0.80      0.54      0.55      1735
weighted avg       0.87      0.88      0.84      1735


Training model for TF
Results for TF
              precision    recall  f1-score   support

           0       0.82      0.83      0.82       799
           1       0.85      0.85      0.85       936

    accuracy                           0.84      1735
   macro avg       0.84      0.84      0.84      1735
weighted avg       0.84      0.84      0.84      1735


Training model for JP
Results for JP
              precision    recall  f1-score   support

           0       0.82      0.59      0.69       692
           1       0.77      0.91      0.84      1043

    accuracy                           0.78      1735
   macro avg       0.79      0.75      0.76      1735
weighted avg       0.79      0.78      0.78      1735


Predicted MBTI type: INFP
```

REFERENCE:

Plank, B., Hovy, D., & Søgaard, A. (2014).
Personality traits on Twitter — Or how to get 1,500 personality tests in a week.
Proceedings of the 6th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis.
https://aclanthology.org/W14-2603

Yarkoni, T. (2010).
Personality in 100,000 Words: A large-scale analysis of personality and word use among bloggers.
Journal of Research in Personality, 44(3), 363–373.
https://doi.org/10.1016/j.jrp.2010.04.001

 Golbeck, J., Robles, C., & Turner, K. (2011).
Predicting Personality with Social Media.
CHI '11 Extended Abstracts on Human Factors in Computing Systems.
https://doi.org/10.1145/1979742.1979614

Kaggle Dataset — MBTI Personality Type Dataset.
Datalab - MBTI Personality Prediction.
https://www.kaggle.com/datasnaek/mbti-type