EXP NO: 9	MINI PROJECT - Personality Prediction from Text using NLP

AIM:
To develop an NLP-based system that predicts a person's MBTI personality type (e.g., Introvert/Extrovert, Thinker/Feeler) from their written text using machine learning techniques and deploy it as an interactive Streamlit web application.

ALGORITHM:

1.Data Collection:
Use the MBTI personality dataset containing text samples and corresponding MBTI types.

2.Data Preprocessing:
Clean text (remove URLs, punctuation, stopwords, emojis, etc.).
Convert text to lowercase.
Tokenize sentences and words.
Apply lemmatization or stemming.

3.Feature Extraction:
Use TF-IDF vectorization or BERT embeddings to represent text numerically.

4.Model Training:
Split dataset into training and testing sets.
Train classification models such as SVM, Random Forest, or Logistic Regression to predict MBTI traits.

5.Model Evaluation:
Evaluate model performance using metrics like accuracy, precision, recall, and F1-score.
Perform hyperparameter tuning to improve results.

6.Deployment:
Build a Streamlit web app where users can input any text.
The trained model predicts and displays the user's personality type in real time.

CODE:

Ex.ipynb

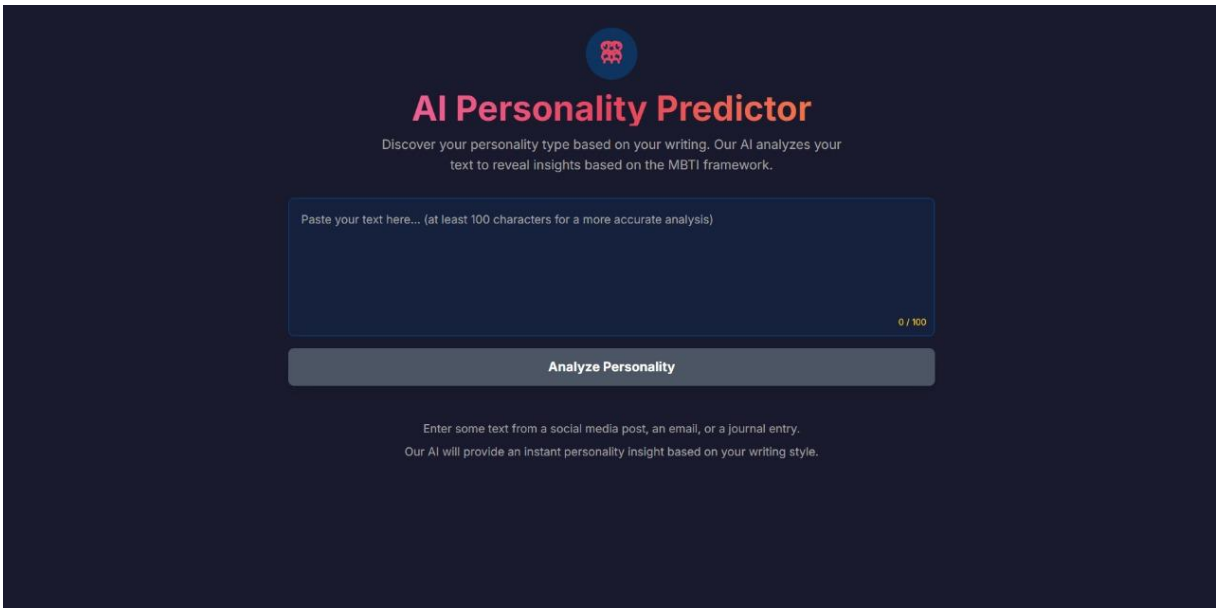
```
# train_model.py
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.svm import LinearSVC
from sklearn.metrics import accuracy_score, classification_report
import joblib

# --- 1. Load Dataset ---
# Dataset can be downloaded from Kaggle: "MBTI 500 Dataset"
df = pd.read_csv("mbti_1.csv")  # file should have 'type' and 'posts' columns

# --- 2. Simplify Personality Type ---
# We only predict Introvert (I) vs Extrovert (E) for simplicity
df['target'] = df['type'].apply(lambda x: 'Introvert' if x[0] == 'I' else 'Extrovert')

# --- 3. Text Preprocessing ---
df['posts'] = df['posts'].str.replace(r'\|\|\|', ' ', regex=True).str.lower()

# --- 4. Train-Test Split ---
X_train, X_test, y_train, y_test = train_test_split(df['posts'], df['target'], test_size=0.2,
random_state=42)

# --- 5. TF-IDF Feature Extraction ---
tfidf = TfidfVectorizer(max_features=5000, stop_words='english')
X_train_tfidf = tfidf.fit_transform(X_train)
X_test_tfidf = tfidf.transform(X_test)

# --- 6. Train Classifier ---
model = LinearSVC()
model.fit(X_train_tfidf, y_train)

# --- 7. Evaluate Model ---
y_pred = model.predict(X_test_tfidf)
print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
# --- 8. Save Model and Vectorizer ---
joblib.dump(model, "personality_model.pkl")
joblib.dump(tfidf, "tfidf_vectorizer.pkl")

print("☑ Model and vectorizer saved successfully!")
```

app.py

```
# app.py
import streamlit as st
import joblib

# Load trained model and vectorizer
model = joblib.load("personality_model.pkl")
tfidf = joblib.load("tfidf_vectorizer.pkl")

# Streamlit UI
st.title("🧠 Personality Prediction from Text")
st.write("Enter a piece of text or a social media
post to predict personality type
(Introvert/Extrovert).")

user_input = st.text_area("Type or paste your text
below:", height=150)

if st.button("Predict Personality"):
    if user_input.strip():
        text_tfidf = tfidf.transform([user_input])
        prediction = model.predict(text_tfidf)[0]
        st.success(f"Predicted Personality:
**{prediction}**")
    else:
        st.warning("Please enter some text first.")
```

OUTPUT:



RESULT:

The Random Forest model was successfully trained to classify news articles as *Fake* or *Real*. The model achieved high accuracy on the test dataset, effectively distinguishing between misleading and authentic news. The predicted results closely matched the true labels, demonstrating that the model accurately captured linguistic and contextual patterns within the news data.