\*\*Personality Project Instructions\*\*

\*\*Overview:\*\*

The Personality Project is an NLP-based project aimed at predicting personality types based on people's social media posts and search histories. It utilizes various machine learning models to classify individuals into different personality types.

\*\*Files:\*\*

1. `backend.ipynb`: This Jupyter Notebook contains the backend code for the project. It includes data preprocessing, model training, evaluation, and saving the best model. Additionally, it contains the trained model for Linear SVC, label encoder, and TF-IDF vectorizer, all of which are pre-saved.

2. `frontend.py`: This Python script serves as the frontend interface for the project. It allows users to input text data and get predictions on personality types using the trained model. It loads the pre-trained Linear SVC model, label encoder, and TF-IDF vectorizer from the designated folder.

3. Pre-trained Files (Folder):

- `linear\_svc\_model.pkl`: Pre-trained Linear SVC model saved using joblib or pickle.

- `tfidf\_vectorizer.pkl`: Pre-trained TF-IDF vectorizer for text representation.

- `label\_encoder.pkl`: Pre-trained label encoder for transforming personality type labels.

\*\*Instructions:\*\*

\*\*1. Backend (backend.ipynb):\*\*

- Ensure you have all necessary libraries installed. Common libraries used in this project include scikit-learn, pandas, numpy, and matplotlib.

- Load the dataset containing people's posts and search histories along with their labeled personality types.

- Preprocess the data by cleaning, tokenizing, and transforming it into a format suitable for model training.

- Split the dataset into training and testing sets for model evaluation.

- Train multiple machine learning models on the dataset. While Linear SVC is the best-performing model in this project, other models like CatBoost can be tried as well (though commented out in the code).

- Evaluate the performance of each model using appropriate metrics such as accuracy, precision, recall, and F1-score.

- Select the best-performing model based on evaluation metrics and save it along with the trained label encoder and TF-IDF vectorizer.

\*\*2. Frontend (frontend.py):\*\*

- Import necessary libraries, including the pre-trained model, label encoder, and TF-IDF vectorizer from the designated folder.

- Create a user-friendly interface for users to input their text data (posts or search histories).

- Preprocess the input text data similar to the preprocessing done in the backend (cleaning, tokenization, etc.) using the loaded TF-IDF vectorizer.

- Utilize the pre-trained Linear SVC model to predict the personality type based on the input text data.

- Use the loaded label encoder to transform the predicted personality type labels back to their original form.

- Display the predicted personality type to the user.

\*\*3. Running the Project:\*\*

- Ensure the pre-trained model files (`linear\_svc\_model.pkl`, `tfidf\_vectorizer.pkl`, `label\_encoder.pkl`) are available in the designated folder.

- Run `backend.ipynb` to train the model, save the trained model, label encoder, and TF-IDF vectorizer to the designated folder.

- Ensure the trained model files are accessible to `frontend.py`.

- Run `frontend.py` to start the frontend interface.

- Input text data when prompted and receive predictions on personality types.

\*\*Note:\*\*

- It's important to understand the nature of the data and the limitations of the model. NLP projects can be sensitive to biases in the data and may not always provide accurate predictions.

- Regularly update the model with new data and retrain if necessary to improve performance.

- Consider implementing additional features or refining existing ones to enhance the project's capabilities over time.