Task Documentation: Building a Machine Learning Application

Objective: The goal is to build a Python based application with a simple and user-friendly interface. The application should use a pretrained machine learning model to predict outcomes for new data uploaded by the user.

Task Description: Your task is to design an application with the following functionality:

1. PreTrained Model: Use the provided training dataset (`training_data.csv`) to train a machine learning model (e.g., Logistic Regression or Random Forest).

- The model should be pretrained and saved (e.g., using `joblib` or `pickle`).
- This step is to be done offline and is not part of the application interface.

2. User Interface (UI): Create a user-friendly interface (as shown in UI.png) using a Python library such as Streamlit or Tkinter. Include the following elements:

- File Upload: Allow the user to upload a new dataset (e.g., `testing_data.csv`) without labels.
- Submit Button: Trigger the prediction process using the pretrained model when clicked.

3. Prediction Results: The application should process the uploaded file and predict whether each row is "good" or "bad" based on the pretrained model. Display the results in the UI, either:

- As a table showing each row and its predicted label.
- Or, as an image file with the original data and the predicted labels appended.

Datasets Provided:

1. Training Dataset:

File Name: `training_data.csv`

Rows: 1000

Columns:

- Feature1, Feature2, ..., Feature8: Numerical features.
- **Label:** Categorical label ("good" or "bad").

Description: This dataset should be used to train and save your machine learning model offline.

2. Testing Dataset:

File Name: `testing_data.csv`

Rows: 10

Columns:

- Feature1, Feature2, ..., Feature8: Numerical features (without labels).

Description: This dataset should be used to test the application's prediction functionality.

Steps to Complete the Task:

1. Train the Model (Offline):

- Load the `training_data.csv` dataset.
- Preprocess the data (e.g., handle missing values, encode categorical variables).
- Train a machine learning model, such as Logistic Regression or Random Forest etc.,
- Save the trained model using a library like `joblib` or `pickle`.

2. Build the Application:

- Create a Pythonbased UI using Streamlit or Tkinter.
- Add the following components:
- A "Choose File" button for uploading the testing dataset.
- A "Submit" button to process the uploaded file and make predictions using the pretrained model.

3. Implement Prediction Logic:

- Load the pretrained model when the application starts.
- Process the uploaded file (e.g., `testing_data.csv`) to predict labels for each row.
- Display the results in a user-friendly format, such as:
    - A table with rows and their predicted labels OR an image with appended labels.

4. Enhancements (Optional):

- Add visualizations (e.g., pie charts or bar charts) to summarize the predictions.
- Style the UI for better user experience.

Deliverables:

1. Python Application:

The Python code implementing the UI and prediction functionality.

2. Trained Model:

Include the pretrained model file (e.g., `.pkl` or `.joblib`).

3. README File:

Instructions to set up and run the application.

4. Screenshots or Video:

Visual demonstration of the application.

Evaluation Criteria:

1. Functionality:

- Does the application load a pretrained model and process uploaded files correctly?
- Are predictions accurate and displayed effectively?

2. Code Quality:

- Is the code clean, modular, and well documented?

3. UI Design:

- Is the user interface intuitive and visually appealing?

4. Presentation:

- Is the output format clear and user-friendly?

Example Workflow:

1. Offline Model Training:

- Train and save the model using `training_data.csv`.

2. Using the Application:

- Step 1: Open the application.
- Step 2: Click on the "Choose File" button and upload the `testing_data.csv` file.
- Step 3: Click the "Submit" button to make predictions.
- Step 4: View the prediction results displayed in the application or download the file with predictions.