Document: Overview of Flask API for Wine Quality Prediction

Introduction

The objective of this project was to create a Flask-based API that accepts a CSV file containing wine characteristics, processes the data using a trained Random Forest model, and returns predicted wine quality. This API is intended to be tested using Postman, a popular API testing tool.

Project Overview

1. Flask Framework:

- **Purpose**: Flask is a lightweight WSGI web application framework in Python. It is designed with simplicity and flexibility in mind, making it a great choice for creating web APIs.
- **Usage**: The Flask framework was used to set up the server, define routes, and handle incoming HTTP requests.

2. Pandas Library:

- **Purpose**: Pandas is a powerful data manipulation and analysis library for Python.
- **Usage**: Pandas was used to read the CSV file and perform necessary preprocessing on the data.

3. Scikit-learn Library:

- **Purpose**: Scikit-learn is a machine learning library for Python. It includes various algorithms for classification, regression, clustering, and more.
- **Usage**: Scikit-learn was used to train the Random Forest model for predicting wine quality.

4. Joblib Library:

- **Purpose**: Joblib is a library for saving and loading machine learning models.
- **Usage**: Joblib was used to serialize the trained Random Forest model so it could be loaded and used in the Flask API.

Steps Taken

Setting Up the Environment:

- Installed the necessary libraries (Flask, Pandas, Scikit-learn, Joblib) using pip.
- Created a new Flask application.

Training the Random Forest Model:

- Loaded the wine quality dataset.
- Preprocessed the data (e.g., handled missing values, normalized features).
- Split the data into training and testing sets.
- Trained a Random Forest model on the training set.
- Evaluated the model's performance on the testing set.
- Saved the trained model using Joblib.

Creating the Flask Application:

- Defined routes and endpoints for the API.
- Implemented logic to handle file uploads and process CSV data.
- Loaded the pre-trained Random Forest model.
- Defined an endpoint for predicting wine quality from new data.

Reading and Processing the CSV File:

- Read the CSV file into a Pandas DataFrame.
- Preprocessed the data as required by the model.
- Used the loaded model to make predictions on the new data.

Returning JSON Responses:

- Ensured that the predictions were converted into a JSON-compatible format.
- Sent the results back to the client as a JSON response.

Testing with Postman

Setting Up Postman:

- Created a new Postman collection for the API.
- Defined requests to test each endpoint, particularly the one for uploading the CSV file and retrieving predictions.

Testing Endpoints:

- Uploaded a sample CSV file containing wine characteristics.
- Verified the JSON response to ensure that the predictions were accurate and correctly formatted.

Learned Concepts

RESTful APIs:

- **Concept**: REST (Representational State Transfer) is an architectural style for designing networked applications. It relies on a stateless, client-server communication protocol, typically HTTP.

- **Application**: Learned how to design and implement RESTful APIs using Flask, including defining endpoints, handling HTTP methods (GET, POST), and structuring JSON responses.

File Handling in Flask:

- **Concept**: Handling file uploads in a web application.
- **Application**: Implemented file upload functionality in Flask, allowing users to upload CSV files via a POST request.

Data Processing with Pandas:

- **Concept**: Data manipulation and preprocessing using Pandas.
- **Application**: Used Pandas to read CSV data, perform necessary preprocessing, and prepare the data for prediction.

Machine Learning with Scikit-learn:

- **Concept**: Training and using machine learning models.
- **Application**: Trained a Random Forest model on the wine quality dataset, evaluated its performance, and used it to make predictions on new data.

Model Serialization with Joblib:

- **Concept**: Saving and loading machine learning models.
- **Application**: Used Joblib to serialize the trained Random Forest model for later use in the Flask API.

API Testing with Postman:

- **Concept**: Testing APIs to ensure they behave as expected.
- **Application**: Created a Postman collection to automate and validate API requests and responses.

How to Run this from Github:

1. **Install Python 3.8:**
    - **Windows:**
        1. Download the installer from the official Python website.
        2. Run the installer and follow the prompts, making sure to check "Add Python to PATH".
    - **macOS:**

      ```
      brew install python@3.8
      ```

    - **Linux:**

      ```
      sudo apt update
      sudo apt install python3.8
      ```

2. **Clone the repository:**

```
git clone https://github.com/shriyakela/apicollection.git
cd flask-project
```

3. **Create a virtual environment:**

```
python3.8 -m venv venv
venv/bin/activate
# On Windows: venv\Scripts\activate
```

4. **Install dependencies:**

```
pip install -r requirements.txt
```

5. **Run the Flask application:**

```
python app.py
```

6. **Testing the API using Postman:**
    - **Create a new POST request:**
        - **URL:** `http://127.0.0.1:5000/api/predict_wine_quality`
        - **Body tab:** Select `form-data`.
        - **Add a key:** `csv_file` with type `File` and choose your CSV file.
        - **Send the request.**

GitHub Link

- GitHub Repository


500 × 400