FastAPI Walmart Sales Prediction

Overview

This project is a Fast API application for predicting weekly sales using a pre-trained Linear Regression model. The dataset used is walmart_sales.csv, and the model is trained using features like store number, temperature, fuel price, CPI, and unemployment rate.

Setup Instructions

1. Prerequisites

Ensure you have the following prerequisites installed:

- Python 3.7+
- Required Python libraries: fastapi, pandas, numpy, sklearn, uvicorn, and pydantic.
- A dataset named walmart_sales.csv.

2. Installation

Install the necessary libraries using pip:
bash
Copy code
pip install fastapi pandas numpy scikit-learn uvicorn pydantic requests

3. Dataset

The dataset, walmart_sales.csv, should contain the following columns:

- **Store** (int): Store number.
- **Holiday_Flag** (int): Whether it is a holiday week (1 for holiday, 0 otherwise).
- **Temperature** (float): Average temperature.
- **Fuel_Price** (float): Cost of fuel.
- **CPI** (float): Consumer Price Index.
- **Unemployment** (float): Unemployment rate.
- **Weekly_Sales** (float): Weekly sales (target variable).

4. Running the Application

To start the API server, execute the following command:

uvicorn fast_api:app --reload

Code Sections Explained

1. Model Training and Saving

Before running the API, ensure the machine learning model is trained using the dataset. If a pre-trained model does not exist, the application will automatically train a new Linear Regression model and save it.

```python
def train_and_save_model():
    # Load dataset
    data = pd.read_csv(DATASET_PATH)

    # Select features and target
    X = data[['Store', 'Holiday_Flag', 'Temperature', 'Fuel_Price', 'CPI', 'Unemployment']]
    y = data['Weekly_Sales']

    # Train/test split
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=

    # Train the model
    model = LinearRegression()
    model.fit(X_train, y_train)

    # Save model as pickle
    with open(MODEL_PATH, 'wb') as model_file:
        pickle.dump(model, model_file)
```

2. Fast API Endpoint

The main endpoint /predict accepts a POST request with a JSON payload. It predicts the weekly sales for the input data using the pre-trained model.

Endpoint: /predict

- **Method**: POST
- **Input**: List of store-related data (in JSON format).
- **Output**: Predicted weekly sales.

```
[
    {
        "Store": 1,
        "Holiday_Flag": 0,
        "Temperature": 75.0,
        "Fuel_Price": 2.50,
        "CPI": 211.0,
        "Unemployment": 7.5
    },
    {
        "Store": 2,
        "Holiday_Flag": 1,
        "Temperature": 80.0,
        "Fuel_Price": 2.70,
        "CPI": 220.0,
        "Unemployment": 8.0
    }
]
```


3. Input Schema

The input data is validated using the Prediction Request Pedantic model, ensuring that the fields have the correct types.

```python
class PredictionRequest(BaseModel):
    Store: int
    Holiday_Flag: int
    Temperature: float
    Fuel_Price: float
    CPI: float
    Unemployment: float
```

4. Error Handling

If an error occurs during prediction, the API will return a 500 HTTP status code with an error message.

```python
@app.post("/predict")
def predict(data: list[PredictionRequest]):
    try:
        # Data processing and prediction
        input_df = pd.DataFrame([item.dict() for item in data])
        predictions = model.predict(input_df)
        return {"predictions": predictions.tolist()}
    except Exception as e:
        raise HTTPException(status_code=500, detail=str(e))
```

Dynamic Payload Creation

The script includes a function to generate the payload dynamically for different stores. This allows flexibility in modifying the store numbers without changing the payload structure.

```
def create_payload(store_numbers):
    payload = []
    for store in store_numbers:
        payload.append({
            "Store": store,
            "Holiday_Flag": 0,
            "Temperature": 75.0,
            "Fuel_Price": 2.50,
            "CPI": 211.0,
            "Unemployment": 7.5
        })
    return payload
```

Conclusion

This API provides a scalable and efficient solution to predict Walmart weekly sales using a simple Linear Regression model. The modular code structure allows for easy adaptation to other datasets or models if needed. The endpoint /predict can handle batch predictions, and the input validation ensures that only correctly formatted data is processed.

Feel free to extend this API with additional features like logging, data preprocessing, or integrating other machine learning models.