Kịch bản giải thích cho code Linear Regression

**1. Nhập các thư viện cần thiết**

- `import pandas as pd`: Thư viện `pandas` giúp xử lý dữ liệu dạng bảng. Chúng ta sẽ sử dụng nó để quản lý và xử lý dữ liệu dạng bảng thông qua cấu trúc `DataFrame`.

- `import matplotlib.pyplot as plt`: Thư viện `matplotlib` giúp tạo đồ thị để trực quan hóa dữ liệu.

- `from sklearn.linear\_model import LinearRegression`: Thư viện `scikit-learn` cung cấp công cụ xây dựng các mô hình học máy. Ở đây, ta sử dụng mô hình hồi quy tuyến tính (`LinearRegression`).

- `from sklearn.model\_selection import train\_test\_split`: Công cụ để chia dữ liệu thành tập huấn luyện và kiểm tra, giúp đánh giá độ chính xác của mô hình.

**2. Tạo dữ liệu giả lập**

- Đoạn mã này tạo một tập dữ liệu mẫu giả lập (có thể thay thế bằng dữ liệu thực tế trong các dự án thực). Dữ liệu bao gồm:

- `Temperature` (Nhiệt độ): Đại diện cho nhiệt độ môi trường.

- `Humidity` (Độ ẩm): Độ ẩm trong không khí.

- `Wind\_Speed` (Tốc độ gió): Tốc độ gió tại thời điểm đo.

- `Feels\_Like` (Cảm giác): Cảm giác nhiệt độ thực tế mà người dùng cảm nhận.

Tập dữ liệu này được chuyển thành cấu trúc `DataFrame` của `pandas` để dễ dàng thao tác.

**3. Chia dữ liệu thành đặc trưng (X) và nhãn (y)**

- `X`: Tập đặc trưng, chứa các biến đầu vào cần sử dụng để dự đoán (Temperature, Humidity, Wind\_Speed).

- `y`: Nhãn, tức là biến đầu ra mà mô hình cần dự đoán (`Feels\_Like`).

**4. Chia tập dữ liệu thành tập huấn luyện và kiểm tra**

- Sử dụng hàm `train\_test\_split`:

- `test\_size=0.3`: Chia 30% dữ liệu để kiểm tra mô hình, 70% để huấn luyện.

- `random\_state=42`: Đặt giá trị cố định để mỗi lần chạy lại, kết quả chia dữ liệu vẫn như cũ.

**5. Huấn luyện mô hình hồi quy tuyến tính**

- `LinearRegression()`: Khởi tạo một mô hình hồi quy tuyến tính.

- `model.fit(X\_train, y\_train)`: Huấn luyện mô hình trên tập dữ liệu huấn luyện (X\_train, y\_train).

**6. Dự đoán trên tập kiểm tra**

- `y\_pred = model.predict(X\_test)`: Mô hình dự đoán các giá trị của `Feels\_Like` trên tập kiểm tra dựa vào dữ liệu đầu vào `X\_test`.

**7. Đánh giá mô hình**

- `model.score(X\_test, y\_test)`: Tính toán hệ số R-squared:

- R-squared là thước đo mức độ tốt của mô hình hồi quy, dao động từ 0 đến 1. Giá trị càng gần 1 thì mô hình càng tốt.

**8. Vẽ đồ thị**

- Biểu đồ scatter:

- Trục X: Giá trị thực tế của cảm giác nhiệt độ (`y\_test`).

- Trục Y: Giá trị dự đoán của cảm giác nhiệt độ (`y\_pred`).

- Biểu đồ giúp kiểm tra mức độ tương đồng giữa giá trị dự đoán và giá trị thực tế.

Chạy thử và giải thích kết quả

- Sau khi chạy, bạn sẽ thấy:

1. Kết quả thực tế (`y\_test`) và dự đoán (`y\_pred`): So sánh để đánh giá sự chính xác của mô hình.

2. R-squared: Đo lường mức độ giải thích của mô hình đối với dữ liệu.

3. Biểu đồ scatter: Nếu các điểm nằm gần đường chéo (45 độ), mô hình có độ chính xác cao.

**Cách cải thiện mô hình**

- Nếu R-squared thấp:

- Thêm nhiều dữ liệu hơn.

- Thử nghiệm các mô hình khác như hồi quy phi tuyến tính.

- Kiểm tra xem có dữ liệu bị nhiễu hay không.

Kịch bản diễn giải đoạn code Logistics Regression

**1. Giới thiệu thư viện và mục đích:**

- Đầu tiên, chúng ta nhập các thư viện cần thiết để xử lý dữ liệu, chuẩn hóa, xây dựng mô hình và đánh giá kết quả.

- pandas được sử dụng để xử lý dữ liệu dạng bảng (DataFrame).

- scikit-learn cung cấp công cụ cho các bước:

- Chia dữ liệu thành tập huấn luyện và kiểm tra (`train\_test\_split`).

- Chuẩn hóa dữ liệu (`StandardScaler`).

- Xây dựng mô hình hồi quy logistic (`LogisticRegression`).

- Đánh giá mô hình (`accuracy\_score`, `confusion\_matrix`, `classification\_report`).

**2. Dữ liệu và mục tiêu:**

- Dữ liệu bao gồm các đặc trưng thời tiết như: nhiệt độ, độ ẩm, áp suất, và hai nhãn để dự đoán:

- rain: Có mưa hay không (1 là mưa, 0 là không mưa).

- storm: Có giông bão hay không (1 là giông bão, 0 là không có).

- Dữ liệu được tổ chức thành một `DataFrame` sử dụng thư viện `pandas`.

**3. Chọn đặc trưng và nhãn:**

- Đặc trưng (features): `temperature`, `humidity`, và `pressure` là những yếu tố dùng để dự đoán.

- Nhãn (labels):

- `rain`: Dự đoán khả năng có mưa.

- `storm`: Dự đoán khả năng có giông bão.

**4. Chia dữ liệu:**

- Sử dụng `train\_test\_split` để chia dữ liệu thành:

- Tập huấn luyện: 80% dữ liệu, dùng để xây dựng mô hình.

- Tập kiểm tra: 20% dữ liệu, dùng để kiểm tra hiệu quả của mô hình.

- `random\_state=42` đảm bảo việc chia dữ liệu luôn cho ra kết quả giống nhau khi chạy lại.

**5. Chuẩn hóa dữ liệu:**

- Chuẩn hóa dữ liệu giúp các đặc trưng có giá trị trung bình là 0 và độ lệch chuẩn là 1. Điều này rất quan trọng với các mô hình như hồi quy logistic để tăng hiệu quả và tốc độ hội tụ.

- Quy trình:

- Huấn luyện bộ chuẩn hóa (`fit\_transform`) trên tập huấn luyện.

- Sử dụng bộ chuẩn hóa đó để chuẩn hóa tập kiểm tra (`transform`).

**6. Xây dựng mô hình dự đoán mưa:**

- Khởi tạo mô hình: Dùng `LogisticRegression` để dự đoán khả năng mưa.

- Huấn luyện mô hình: Sử dụng tập dữ liệu huấn luyện đã chuẩn hóa để huấn luyện mô hình.

- Dự đoán và đánh giá:

- Dự đoán trên tập kiểm tra.

- Đánh giá mô hình qua các chỉ số:

- Độ chính xác (Accuracy): Tỉ lệ dự đoán đúng trên tổng số mẫu.

- Ma trận nhầm lẫn (Confusion Matrix): Hiển thị số lượng dự đoán đúng và sai theo từng lớp.

- Báo cáo phân loại (Classification Report): Thống kê chi tiết về độ chính xác, độ nhạy (recall), độ đặc hiệu, và F1-score.

**7. Xây dựng mô hình dự đoán giông bão:**

- Tương tự như dự đoán mưa:

- Huấn luyện mô hình trên tập dữ liệu giông bão.

- Dự đoán trên tập kiểm tra.

- Đánh giá hiệu quả qua các chỉ số tương tự.

**8. Kết quả và ý nghĩa:**

- Kết quả cho thấy mô hình có khả năng phân loại tốt dựa trên các đặc trưng thời tiết.

- Các chỉ số trong báo cáo phân loại giúp hiểu rõ mô hình có hoạt động tốt ở lớp nào và cần cải thiện ở đâu.

**Quy trình phân tích và xây dựng mô hình LSTM để dự đoán nhiệt độ**

**1. Giới thiệu các thư viện**

Đầu tiên, chúng ta sử dụng các thư viện Python mạnh mẽ để phân tích dữ liệu và xây dựng mô hình:

- `numpy`: Hỗ trợ làm việc với các mảng và thực hiện các phép toán số học.

- `pandas`: Quản lý dữ liệu dưới dạng bảng, rất tiện lợi để xử lý tập dữ liệu lớn.

- `matplotlib`: Vẽ biểu đồ giúp trực quan hóa dữ liệu và kết quả dự đoán.

- `sklearn.preprocessing.MinMaxScaler`: Công cụ chuẩn hóa dữ liệu về một khoảng cố định (thường là [0, 1]), giúp mô hình dễ dàng xử lý hơn.

- `tensorflow.keras`: Xây dựng và huấn luyện mô hình học sâu, với `Sequential` hỗ trợ xây dựng mô hình tuyến tính từng bước, và `LSTM` giúp phân tích chuỗi thời gian.

**2. Chuẩn bị dữ liệu**

Vì đây là một ví dụ minh họa, chúng ta tạo ra dữ liệu nhiệt độ giả lập:

**data = pd.DataFrame({'temperature': 20 + 10 \* np.sin(np.linspace(0, 2 \* np.pi, 365\*24)) + np.random.normal(0, 1, 365\*24)})**

- Dữ liệu này mô phỏng chu kỳ nhiệt độ ngày-đêm (hàm sin) cộng thêm yếu tố nhiễu (nhiệt độ thay đổi ngẫu nhiên).

- Kết quả là nhiệt độ biến động quanh giá trị trung bình 20°C, tạo ra một tập dữ liệu có 8760 (365×24) mẫu.

**3. Tiền xử lý dữ liệu**

- Chuẩn hóa dữ liệu:

Chúng ta sử dụng `MinMaxScaler` để đưa giá trị nhiệt độ về khoảng [0, 1]:

**scaler = MinMaxScaler(feature\_range=(0, 1))**

**data['temperature\_scaled'] = scaler.fit\_transform(data[['temperature']])**

- Tạo chuỗi dữ liệu:

LSTM cần dữ liệu quá khứ để dự đoán tương lai, vì vậy, chúng ta xây dựng một hàm:

**def create\_sequences(data, seq\_length):**

**sequences = []**

**labels = []**

**for i in range(len(data) - seq\_length):**

**sequences.append(data[i:i + seq\_length])**

**labels.append(data[i + seq\_length])**

**return np.array(sequences), np.array(labels)**

- `seq\_length=24`: Sử dụng 24 giờ gần nhất để dự đoán giờ tiếp theo.

- Hàm trả về `sequences` (dữ liệu đầu vào) và `labels` (giá trị cần dự đoán).

- Chia dữ liệu:

Dữ liệu được chia thành tập huấn luyện (80%) và tập kiểm tra (20%):

**split = int(0.8 \* len(X))**

**X\_train, X\_test = X[:split], X[split:]**

**y\_train, y\_test = y[:split], y[split:]**

**4. Xây dựng mô hình LSTM**

Mô hình dự đoán nhiệt độ được thiết kế như sau:

- LSTM layer: Sử dụng 50 đơn vị LSTM để ghi nhớ thông tin quá khứ.

- Dense layer: Lớp đầu ra dự đoán một giá trị duy nhất (nhiệt độ giờ tiếp theo).

**model = Sequential([**

**LSTM(50, input\_shape=(seq\_length, 1)),**

**Dense(1)**

**])**

Mô hình được biên dịch với:

- `adam`: Bộ tối ưu hiệu quả cao.

- `mean\_squared\_error` (MSE): Hàm mất mát đo lường sai số giữa dự đoán và thực tế.

**5. Huấn luyện mô hình**

Mô hình được huấn luyện trên tập dữ liệu trong 10 epoch:

python

**history = model.fit(X\_train, y\_train, epochs=10, batch\_size=64, validation\_split=0.2)**

- `batch\_size=64`: Kích thước của mỗi lần cập nhật trọng số.

- `validation\_split=0.2`: Sử dụng 20% dữ liệu huấn luyện để đánh giá mô hình trong quá trình học.

**6. Dự đoán và đánh giá**

- Dự đoán:

Mô hình dự đoán nhiệt độ trong tập kiểm tra:

**y\_pred = model.predict(X\_test)**

- Chuyển đổi về thang đo ban đầu:

Vì dữ liệu đã được chuẩn hóa, chúng ta cần chuyển đổi lại để dễ so sánh:

python

**y\_pred\_rescaled = scaler.inverse\_transform(y\_pred)**

**y\_test\_rescaled = scaler.inverse\_transform(y\_test.reshape(-1, 1))**

- Vẽ biểu đồ:

Biểu đồ so sánh giá trị thực và giá trị dự đoán:

**plt.figure(figsize=(10, 6))**

**plt.plot(y\_test\_rescaled, label='Giá trị thực')**

**plt.plot(y\_pred\_rescaled, label='Dự đoán', linestyle='--')**

**plt.xlabel('Thời gian')**

**plt.ylabel('Nhiệt độ')**

**plt.legend()**

**plt.show()**

- Đánh giá sai số:

Chúng ta tính toán mức độ mất mát (loss) của mô hình trên tập kiểm tra:

**loss = model.evaluate(X\_test, y\_test, verbose=0)**

**print(f'Loss: {loss}')**

Tóm tắt quy trình

1. Nhập các thư viện và chuẩn bị dữ liệu: Tạo ra một tập dữ liệu nhiệt độ giả lập.

2. Tiền xử lý dữ liệu: Chuẩn hóa dữ liệu và tạo các chuỗi đầu vào cho mô hình.

3. Xây dựng mô hình LSTM: Thiết kế một mô hình với các lớp LSTM và Dense.

4. Huấn luyện mô hình: Sử dụng tập huấn luyện để học từ dữ liệu quá khứ.

5. Dự đoán và đánh giá: Dự đoán nhiệt độ, chuyển đổi kết quả về thang đo ban đầu và so sánh với giá trị thực tế bằng biểu đồ.

Kết quả

Quy trình này minh họa cách sử dụng LSTM để dự đoán chuỗi thời gian, từ chuẩn bị dữ liệu đến đánh giá mô hình. Việc so sánh kết quả dự đoán với dữ liệu thực tế qua biểu đồ giúp đánh giá hiệu quả của mô hình trong bối cảnh cụ thể.