The code step-by-step, explaining the libraries used, the CSV file, and the three figures generated by the code.

**Libraries Used:**

1. **os**: Used to set an environment variable to disable TensorFlow optimizations.
2. **numpy**: A library for numerical operations on arrays and matrices.
3. **pandas**: A data manipulation and analysis library, used here to read the CSV file.
4. **matplotlib**: A plotting library used to create static, animated, and interactive visualizations.
5. **mpl\_toolkits.mplot3d**: A module within matplotlib for 3D plotting.
6. **sklearn.preprocessing**: Contains tools for preprocessing data, such as scaling.
7. **sklearn.model\_selection**: Used for splitting data into training and test sets.
8. **keras**: A deep learning library used to build neural networks.

**CSV File:**

The CSV file, "L A\_Weather.csv", is expected to have weather data for Los Angeles. It likely contains columns such as average temperature, maximum temperature, and minimum temperature.

**Explanation of the Figures:**

**Figure 1: 3D Scatter Plot**

python

Copy code

fig = plt.figure(1)

ax = fig.add\_subplot(111, projection="3d")

ax.scatter(Tmax, Tmin, Tavg, marker='o')

ax.set\_xlabel('Max Temp')

ax.set\_ylabel('Min Temp')

ax.set\_zlabel('Average Temp')

plt.show(block=False)

* **Purpose**: This figure is a 3D scatter plot showing the relationship between the maximum temperature (Tmax), minimum temperature (Tmin), and average temperature (Tavg).
* **Axes**:
  + X-axis: Maximum Temperature
  + Y-axis: Minimum Temperature
  + Z-axis: Average Temperature
* **Insight**: Visualizes how these three temperature metrics are related in 3D space.

**Figure 2: Predicted vs Actual Scatter Plot**

python

Copy code

plt.figure(2)

plt.scatter(Y\_test, predict)

plt.show(block=False)

* **Purpose**: This figure is a scatter plot comparing the predicted average temperatures (predict) to the actual average temperatures (Y\_test).
* **Axes**:
  + X-axis: Actual Average Temperature
  + Y-axis: Predicted Average Temperature
* **Insight**: This plot helps visualize the performance of the neural network model. Ideally, points should lie along the line y=x, indicating perfect predictions.

**Figure 3: Predicted vs Actual Line Plot**

python

Copy code

plt.figure(3)

Test = plt.plot(Y\_test)

Predict = plt.plot(predict)

plt.legend([Predict, Test], ["Predicted Data", "Real Data"])

plt.show()

* **Purpose**: This figure plots the predicted temperatures and the actual temperatures over the test dataset.
* **Insight**: By comparing the two plots, you can see how closely the predicted values match the actual values over the range of the dataset.

**Code Breakdown:**

1. **Reading the CSV File:**

python

Copy code

df = pd.read\_csv("L A\_Weather.csv")

print(df.shape)

* + Reads the CSV file into a DataFrame and prints its shape (number of rows and columns).

1. **Extracting Temperature Data:**

python

Copy code

Tavg = np.array([df.iloc[:, 3]])

Tmax = np.array([df.iloc[:, 4]])

Tmin = np.array([df.iloc[:, 5]])

* + Extracts average temperature, maximum temperature, and minimum temperature from the DataFrame.

1. **Creating the 3D Scatter Plot:**
   * As explained above in Figure 1.
2. **Data Preprocessing:**

python

Copy code

Temp = np.concatenate([Tmax, Tmin], axis=0)

Temp = np.transpose(Temp)

Tavg = np.transpose(Tavg)

scaler = MinMaxScaler()

scaler.fit(Temp)

Temp = scaler.transform(Temp)

scaler1 = MinMaxScaler()

scaler1.fit(Tavg)

Tavg = scaler1.transform(Tavg)

* + Concatenates Tmax and Tmin, transposes them, and scales both the temperature data (Temp) and average temperature (Tavg).

1. **Splitting the Data:**

python

Copy code

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(Temp, Tavg, test\_size=0.3)

* + Splits the data into training and testing sets.

1. **Building and Training the Neural Network:**

python

Copy code

model = Sequential()

model.add(Dense(32, activation='relu', input\_dim=2))

model.add(Dense(32, activation='relu'))

model.add(Dense(1, activation='sigmoid'))

model.compile(loss='mean\_squared\_error', optimizer='rmsprop', metrics=[metrics.mean\_absolute\_error])

model.fit(X\_train, Y\_train, epochs=500, batch\_size=32, verbose=2)

predict = model.predict(X\_test, verbose=1)

* + Creates a Sequential model with two hidden layers (32 neurons each, ReLU activation) and an output layer with one neuron (sigmoid activation).
  + Compiles the model with mean squared error loss and RMSprop optimizer.
  + Trains the model for 500 epochs on the training data.
  + Predicts the average temperatures on the test set.

1. **Generating the Predicted vs Actual Plots:**
   * As explained above in Figures 2 and 3.

By following this step-by-step explanation, you should have a clear understanding of how the code processes the weather data and visualizes the results.