```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
```

```
# Step-1 load the dataset
# Parse_dates : Used to automatoically converted a string object in to a datetime
df = pd.read_csv("/content/daily_minimum_temps.csv", parse_dates = ["Date"],index_
```

```
/tmp/ipython-input-1048835125.py:3: UserWarning: Could not infer format, so each el
  df = pd.read_csv("/content/daily_minimum_temps.csv", parse_dates = ["Date"],index
```

```
df.head()
```

	Temp
Date	
1981-01-01	20.7
1981-01-02	17.9
1981-01-03	18.8
1981-01-04	14.6
1981-01-05	15.8

Next steps: Generate code with df View recommended plots New interactive sheet

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 3650 entries, 1981-01-01 to 1990-12-31
Data columns (total 1 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Temp    3650 non-null   object
dtypes: object(1)
memory usage: 57.0+ KB
```

```
# Clean the data
df["Temp"] = pd.to_numeric(df["Temp"],errors = 'coerce') # coerce = ignore missing
df = df.dropna()
```

```
# Step - 3 Normalise the temperature values
scaler = MinMaxScaler()
data_scaled = scaler.fit_transform(df["Temp"].values.reshape(-1,1))
data_scaled
```

```
array([[0.78707224],
       [0.68060837],
       [0.7148289 ],
       ...,
       [0.51330798],
       [0.59695817],
       [0.49429658]])
```

```
# Creating sequence is the most important step in RNN
# Step -4 :- Create input sequences for LSTM
def create_sequences(data, seq_length):
  X, y = [],[]
  for i in range(len(data) - seq_length):
    X.append(data[i:i+seq_length])
    y.append(data[i+seq_length])


  return np.array(X),np.array(y)
```

```
seq_length = 30
X,y = create_sequences(data_scaled, seq_length)
```

```
# Step -5 Train-test-split (no splitting for time series)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, shuffl
```

```
# step - 6 : Build the LSTM model
model = Sequential([
    LSTM(64, activation = 'relu', input_shape = (seq_length,1)),
    Dense(1) # Output is a single value
])
```

```
/usr/local/lib/python3.12/dist-packages/keras/src/layers/rnn/rnn.py:199: UserWarnin
  super().__init__(**kwargs)
```

```
model.compile(optimizer='adam', loss = 'mse')
```

```
model.summary()
```

```
Model: "sequential"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ Layer (type)                     ┃ Output Shape           ┃       Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
│ lstm (LSTM)                      │ (None, 64)             │        16,896 │
├──────────────────────────────────┼────────────────────────┼───────────────┤
│ dense (Dense)                    │ (None, 1)              │            65 │
└──────────────────────────────────┴────────────────────────┴───────────────┘
 Total params: 16,961 (66.25 KB)
 Trainable params: 16,961 (66.25 KB)
 Non-trainable params: 0 (0.00 B)
```

```
# Step - 7: Train the model
model.fit(X_train,y_train, epochs = 20, batch_size = 32)
```

```
Epoch 1/20
91/91 ━━━━━━━━━━━━━━━━━━━━ 3s 14ms/step - loss: 0.0564
Epoch 2/20
91/91 ━━━━━━━━━━━━━━━━━━━━ 2s 14ms/step - loss: 0.0112
Epoch 3/20
91/91 ━━━━━━━━━━━━━━━━━━━━ 3s 14ms/step - loss: 0.0106
Epoch 4/20
91/91 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - loss: 0.0107
Epoch 5/20
91/91 ━━━━━━━━━━━━━━━━━━━━ 2s 22ms/step - loss: 0.0107
Epoch 6/20
91/91 ━━━━━━━━━━━━━━━━━━━━ 2s 17ms/step - loss: 0.0102
Epoch 7/20
91/91 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - loss: 0.0102
Epoch 8/20
91/91 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - loss: 0.0098
Epoch 9/20
91/91 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - loss: 0.0103
Epoch 10/20
91/91 ━━━━━━━━━━━━━━━━━━━━ 3s 14ms/step - loss: 0.0098
Epoch 11/20
91/91 ━━━━━━━━━━━━━━━━━━━━ 3s 14ms/step - loss: 0.0092
Epoch 12/20
91/91 ━━━━━━━━━━━━━━━━━━━━ 2s 22ms/step - loss: 0.0084
Epoch 13/20
91/91 ━━━━━━━━━━━━━━━━━━━━ 2s 21ms/step - loss: 0.0087
Epoch 14/20
91/91 ━━━━━━━━━━━━━━━━━━━━ 2s 14ms/step - loss: 0.0092
Epoch 15/20
91/91 ━━━━━━━━━━━━━━━━━━━━ 3s 15ms/step - loss: 0.0094
Epoch 16/20
91/91 ━━━━━━━━━━━━━━━━━━━━ 2s 14ms/step - loss: 0.0090
Epoch 17/20
91/91 ━━━━━━━━━━━━━━━━━━━━ 3s 14ms/step - loss: 0.0088
Epoch 18/20
91/91 ━━━━━━━━━━━━━━━━━━━━ 3s 20ms/step - loss: 0.0092
Epoch 19/20
91/91 ━━━━━━━━━━━━━━━━━━━━ 2s 14ms/step - loss: 0.0085
Epoch 20/20
91/91 ━━━━━━━━━━━━━━━━━━━━ 3s 14ms/step - loss: 0.0086
```

```
<keras.src.callbacks.history.History at 0x7f212099f800>
```

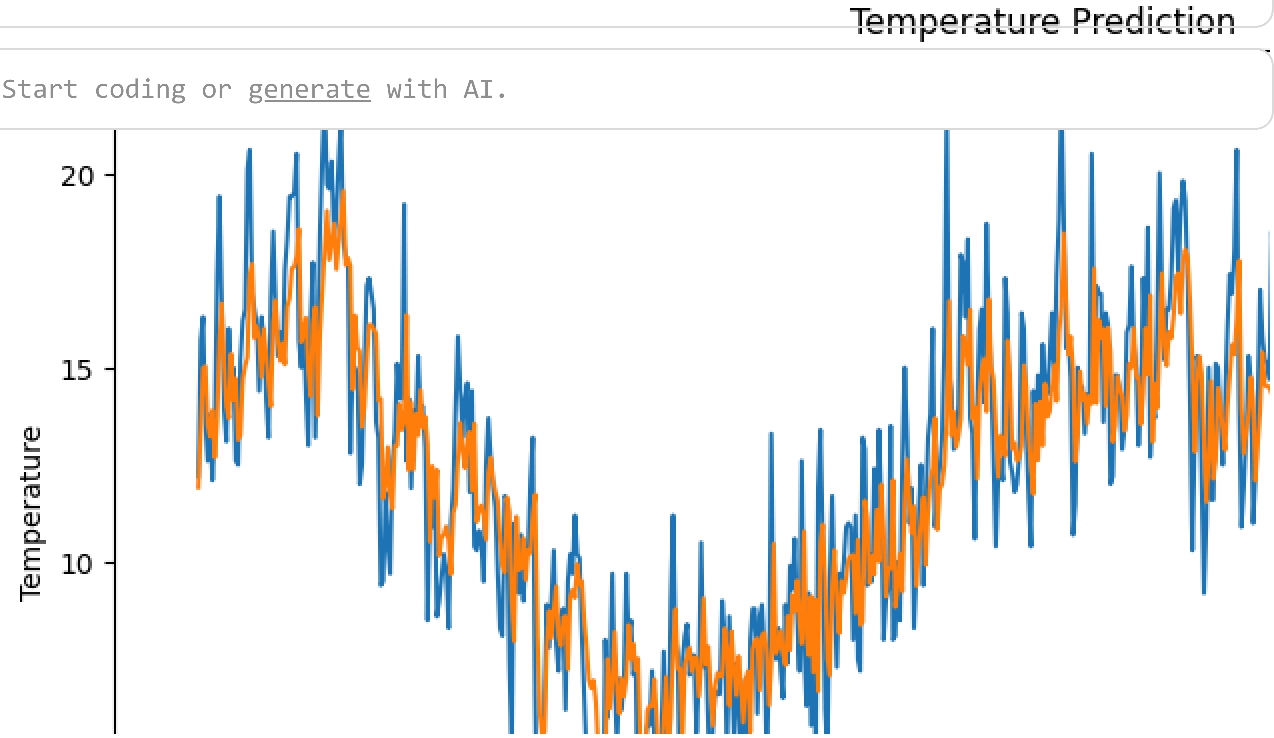
```
# Step - 8 :- Make predictions on the test set
y_pred_scaled = model.predict(X_test)
```

```
23/23 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step
```

```
# Clip predictions to [0,1] before inverse transform
y_pred_scaled = np.clip(y_pred_scaled, 0, 1)
y_pred = scaler.inverse_transform(y_pred_scaled)
y_test_actual = scaler.inverse_transform(y_test)
```

```
# Step - 9 plot predictions
plt.figure(figsize = (12,6))
plt.plot(y_test_actual,label = "Actual Temperatures")
plt.plot(y_pred, label = "Predicted Temperatures")
plt.title("Daily Min Temperature Forecasting (LSTM)")
plt.xlabel("Time")
plt.ylabel("Temperature")
plt.legend()
plt.title("Temperature Prediction")
```

```
Text(0.5, 1.0, 'Temperature Prediction')
```



```
Start coding or generate with AI.
```