Start coding or generate with AI.

```
!pip install pandas numpy matplotlib scikit-learn statsmodels tensorflow keras
```

Show hidden output

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.arima.model import ARIMA
from sklearn.preprocessing import StandardScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
from tensorflow.keras.optimizers import Adam
```

```
temp_data = pd.read_csv("/content/US_City_Temp_Data.csv")
```

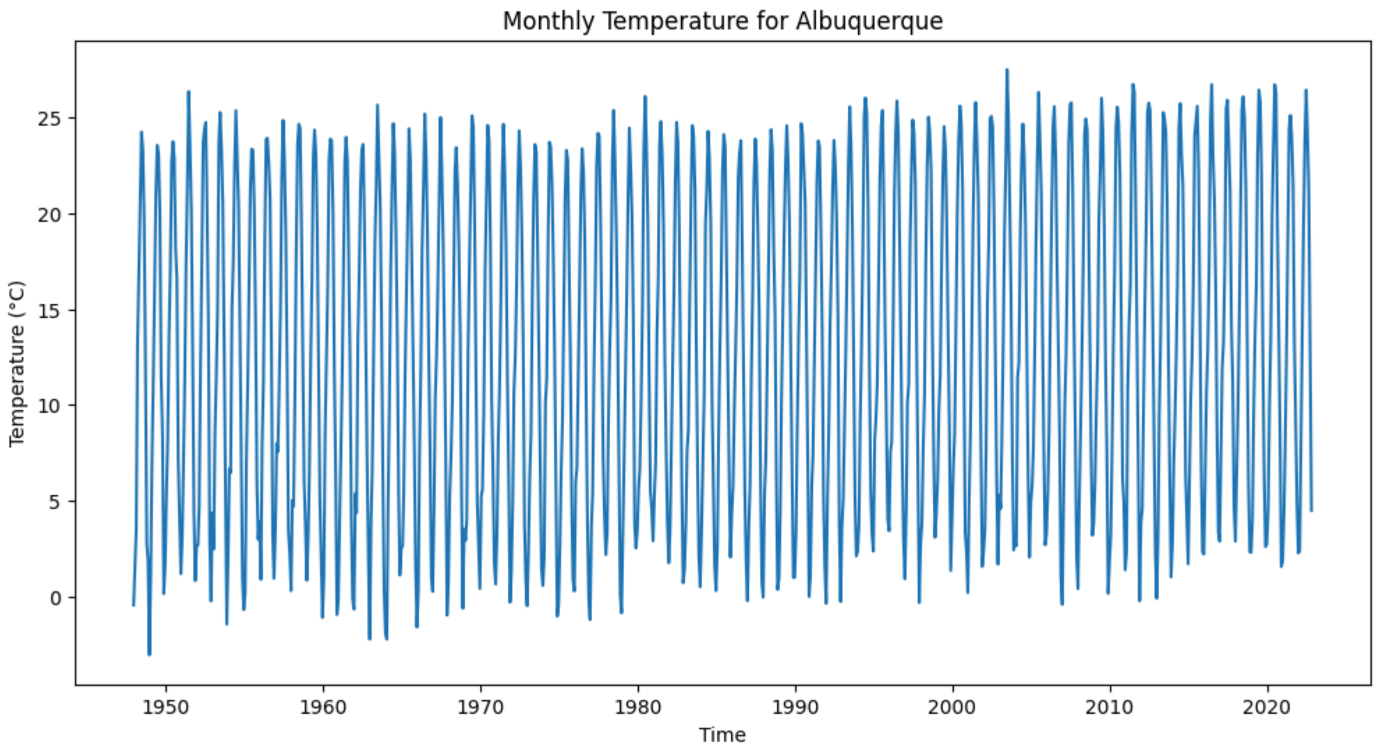
```
temp_data = pd.read_csv("US_City_Temp_Data.csv")
temp_data['time'] = pd.to_datetime(temp_data['time'])
```

```
print(temp_data.info())
print(temp_data.describe())
```

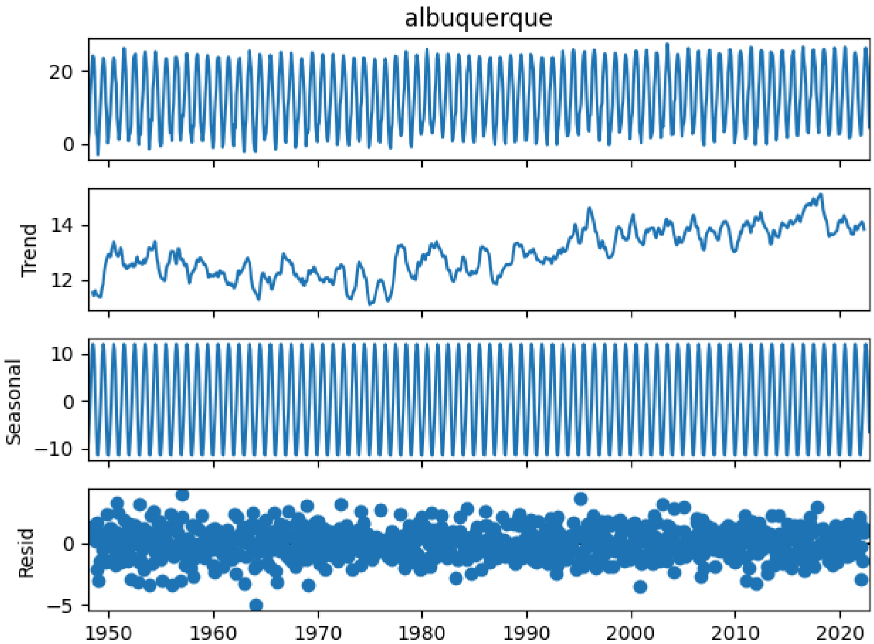
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 899 entries, 0 to 898
Data columns (total 36 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   time            899 non-null    datetime64[ns]
 1   albuquerque     899 non-null    float64
 2   anchorage       899 non-null    float64
 3   atlanta         899 non-null    float64
 4   boise           899 non-null    float64
 5   boston          899 non-null    float64
 6   buffalo         899 non-null    float64
 7   charlotte       899 non-null    float64
 8   chicago         899 non-null    float64
 9   dallas          899 non-null    float64
 10  denver          899 non-null    float64
 11  detroit         899 non-null    float64
 12  helena          899 non-null    float64
 13  honolulu        899 non-null    float64
 14  indianapolis    899 non-null    float64
 15  jacksonville    899 non-null    float64
 16  kansas_city     899 non-null    float64
 17  las_vegas       899 non-null    float64
 18  los_angeles     899 non-null    float64
 19  memphis         899 non-null    float64
 20  miami           899 non-null    float64
 21  minneapolis     899 non-null    float64
 22  new_orleans     899 non-null    float64
 23  new_york        899 non-null    float64
 24  oklahoma_city   899 non-null    float64
 25  phoenix         899 non-null    float64
 26  portland        899 non-null    float64
 27  rapid_city      899 non-null    float64
 28  reno            899 non-null    float64
 29  richmond        899 non-null    float64
 30  sacramento      899 non-null    float64
 31  salt_lake_city  899 non-null    float64
 32  san_antonio     899 non-null    float64
 33  san_francisco   899 non-null    float64
 34  seattle         899 non-null    float64
 35  tampa           899 non-null    float64
dtypes: datetime64[ns](1), float64(35)
memory usage: 253.0 KB
None
                                time  albuquerque   anchorage     atlanta  \
count                            899   899.000000  899.000000  899.000000
```

```
mean   1985-06-01 02:51:23.426028928    12.963612    2.351416   16.677050
min              1948-01-01 00:00:00    -3.040008  -16.790009   -0.489990
25%              1966-09-16 00:00:00     5.310013   -4.578400   10.255005
50%              1985-06-01 00:00:00    12.820007    2.360016   16.855652
75%              2004-02-15 12:00:00    21.180008    9.985016   23.745224
max              2022-11-01 00:00:00    27.490051   16.734528   28.950012
std                              NaN     8.402601    8.110592    7.293325

            boise      boston     buffalo   charlotte     chicago      dallas  \
count  899.000000  899.000000  899.000000  899.000000  899.000000  899.000000
mean     9.497929   10.017592    8.472634   16.049994   10.326989   19.610269
min    -13.589996   -8.088379  -11.307129   -0.919983  -12.729980    2.020020
```

```
# Select data for Albuquerque
city_data = temp_data[['time', 'albuquerque']].dropna()
city_data.set_index('time', inplace=True)

# Plot the time series
plt.figure(figsize=(12, 6))
plt.plot(city_data.index, city_data['albuquerque'])
plt.title("Monthly Temperature for Albuquerque")
plt.ylabel("Temperature (°C)")
plt.xlabel("Time")
plt.show()
```




```
# Decompose the time series
decomposed = seasonal_decompose(city_data['albuquerque'], model='additive', period=12)
decomposed.plot()
plt.tight_layout()
plt.show()
```

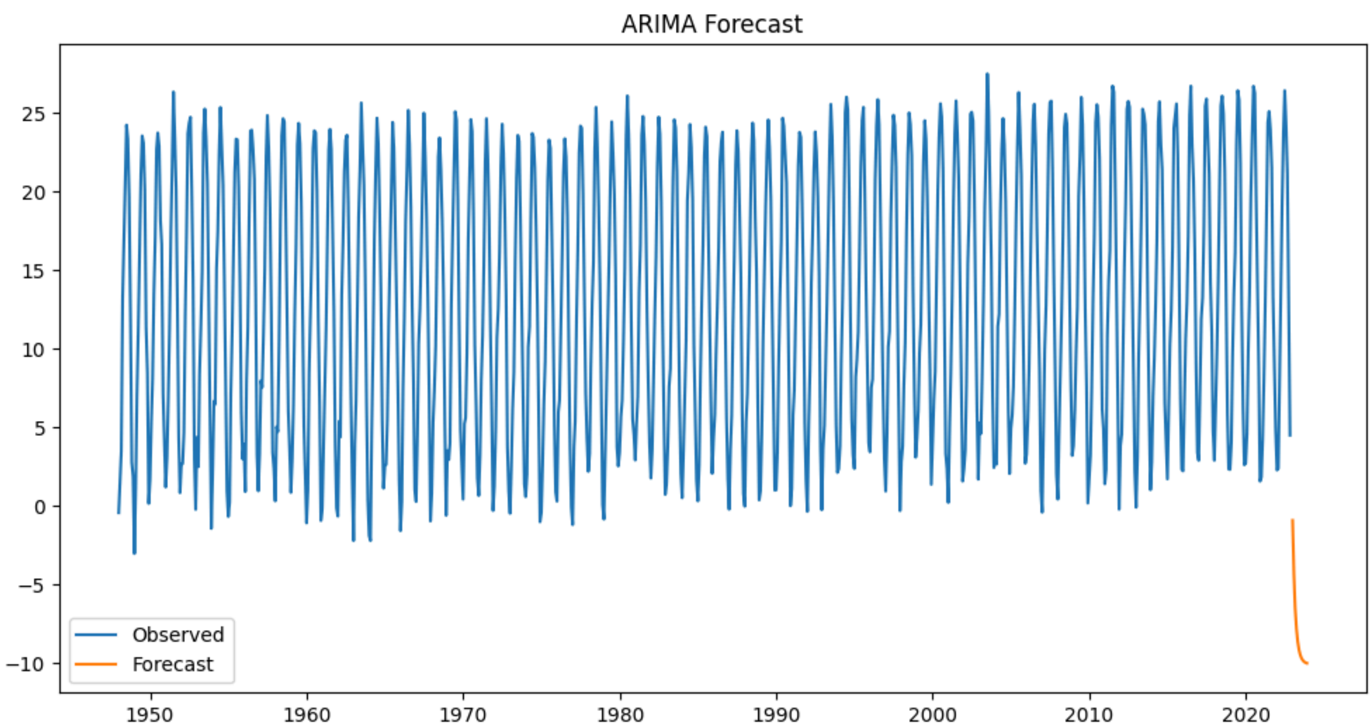



```
def adf_test(series):
    result = adfuller(series)
    print(f'ADF Statistic: {result[0]}')
    print(f'p-value: {result[1]}')
    print('Critical Values:')
    for key, value in result[4].items():
        print(f'\t{key}: {value}')

# Perform ADF test
adf_test(city_data['albuquerque'])
```

```
ADF Statistic: -3.2513043809942173
p-value: 0.01720053095520928
Critical Values:
        1%: -3.4378283848659277
        5%: -2.864841231335243
        10%: -2.5685278140988053
```

```
# Fit ARIMA model
model = ARIMA(city_data['albuquerque'], order=(1,1,1))
results = model.fit()
print(results.summary())

# Forecast
forecast = results.forecast(steps=12)
plt.figure(figsize=(12, 6))
plt.plot(city_data.index, city_data['albuquerque'], label='Observed')
plt.plot(pd.date_range(start=city_data.index[-1], periods=13, freq='M')[1:], forecast, label='Forecast')
plt.title('ARIMA Forecast')
plt.legend()
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: No frequency informatio
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: No frequency informatio
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: No frequency informatio
  self._init_dates(dates, freq)
                               SARIMAX Results
==============================================================================
Dep. Variable:            albuquerque   No. Observations:                  899
Model:                 ARIMA(1, 1, 1)   Log Likelihood               -2364.740
Date:                Sun, 08 Dec 2024   AIC                           4735.480
Time:                        16:37:18   BIC                           4749.881
Sample:                    01-01-1948   HQIC                          4740.982
                         - 11-01-2022
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.6279      0.039     16.092      0.000       0.551       0.704
ma.L1          0.1177      0.050      2.370      0.018       0.020       0.215
sigma2        11.3390      0.627     18.083      0.000      10.110      12.568
===================================================================================
Ljung-Box (L1) (Q):                   0.62   Jarque-Bera (JB):                18.43
Prob(Q):                              0.43   Prob(JB):                         0.00
Heteroskedasticity (H):               0.82   Skew:                             0.26
Prob(H) (two-sided):                  0.08   Kurtosis:                         2.54
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
<ipython-input-31-63e311eac5fd>:10: FutureWarning: 'M' is deprecated and will be removed in a future version, please
  plt.plot(pd.date_range(start=city_data.index[-1], periods=13, freq='M')[1:], forecast, label='Forecast')
```




```
# Scale the data
scaler = StandardScaler()
scaled_data = scaler.fit_transform(city_data[['albuquerque']])

# Reshape data for LSTM
X_train = scaled_data.reshape((len(scaled_data), 1, 1))
Y_train = scaled_data

# Define LSTM model
model = Sequential([
    LSTM(50, return_sequences=True, input_shape=(1, 1)),
    LSTM(50, return_sequences=False),
```

```
    Dense(1)
])

# Compile model
model.compile(optimizer='adam', loss='mean_squared_error')

# Train model
history = model.fit(X_train, Y_train, epochs=50, batch_size=32, validation_split=0.2, verbose=1)

# Make predictions
lstm_predictions = model.predict(X_train)
```

```
Epoch 1/50
/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204: UserWarning: Do not pass an `input_shape`/`:
  super().__init__(**kwargs)
23/23 ━━━━━━━━━━━━━━━━━━━━ 3s 21ms/step - loss: 0.9491 - val_loss: 0.8561
Epoch 2/50
23/23 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.7555 - val_loss: 0.5778
Epoch 3/50
23/23 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.4319 - val_loss: 0.2098
Epoch 4/50
23/23 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.1039 - val_loss: 0.0124
Epoch 5/50
23/23 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0054 - val_loss: 0.0028
Epoch 6/50
23/23 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0021 - val_loss: 0.0012
Epoch 7/50
23/23 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 6.0380e-04 - val_loss: 6.3664e-04
Epoch 8/50
23/23 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 4.0304e-04 - val_loss: 5.0845e-04
Epoch 9/50
23/23 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 4.1004e-04 - val_loss: 4.7584e-04
Epoch 10/50
23/23 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 3.6415e-04 - val_loss: 4.5294e-04
Epoch 11/50
23/23 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 3.6056e-04 - val_loss: 4.6997e-04
Epoch 12/50
23/23 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 3.6561e-04 - val_loss: 4.4277e-04
Epoch 13/50
23/23 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 3.4006e-04 - val_loss: 3.9620e-04
Epoch 14/50
23/23 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 3.2919e-04 - val_loss: 4.3896e-04
Epoch 15/50
23/23 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 3.4303e-04 - val_loss: 4.0898e-04
Epoch 16/50
23/23 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 3.1621e-04 - val_loss: 3.8431e-04
Epoch 17/50
23/23 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 3.1283e-04 - val_loss: 3.7693e-04
Epoch 18/50
23/23 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 2.9587e-04 - val_loss: 3.7971e-04
Epoch 19/50
23/23 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 3.1383e-04 - val_loss: 3.3888e-04
Epoch 20/50
23/23 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 2.8911e-04 - val_loss: 3.1861e-04
Epoch 21/50
23/23 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 3.1471e-04 - val_loss: 3.3189e-04
Epoch 22/50
23/23 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 2.6161e-04 - val_loss: 3.0550e-04
Epoch 23/50
23/23 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 2.3597e-04 - val_loss: 2.8785e-04
Epoch 24/50
23/23 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 2.2970e-04 - val_loss: 3.0792e-04
Epoch 25/50
23/23 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 2.3534e-04 - val_loss: 2.7287e-04
Epoch 26/50
23/23 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 2.3326e-04 - val_loss: 2.6934e-04
Epoch 27/50
23/23 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 2.1163e-04 - val_loss: 2.6770e-04
Epoch 28/50
23/23 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 1.9400e-04 - val_loss: 2.5310e-04
```

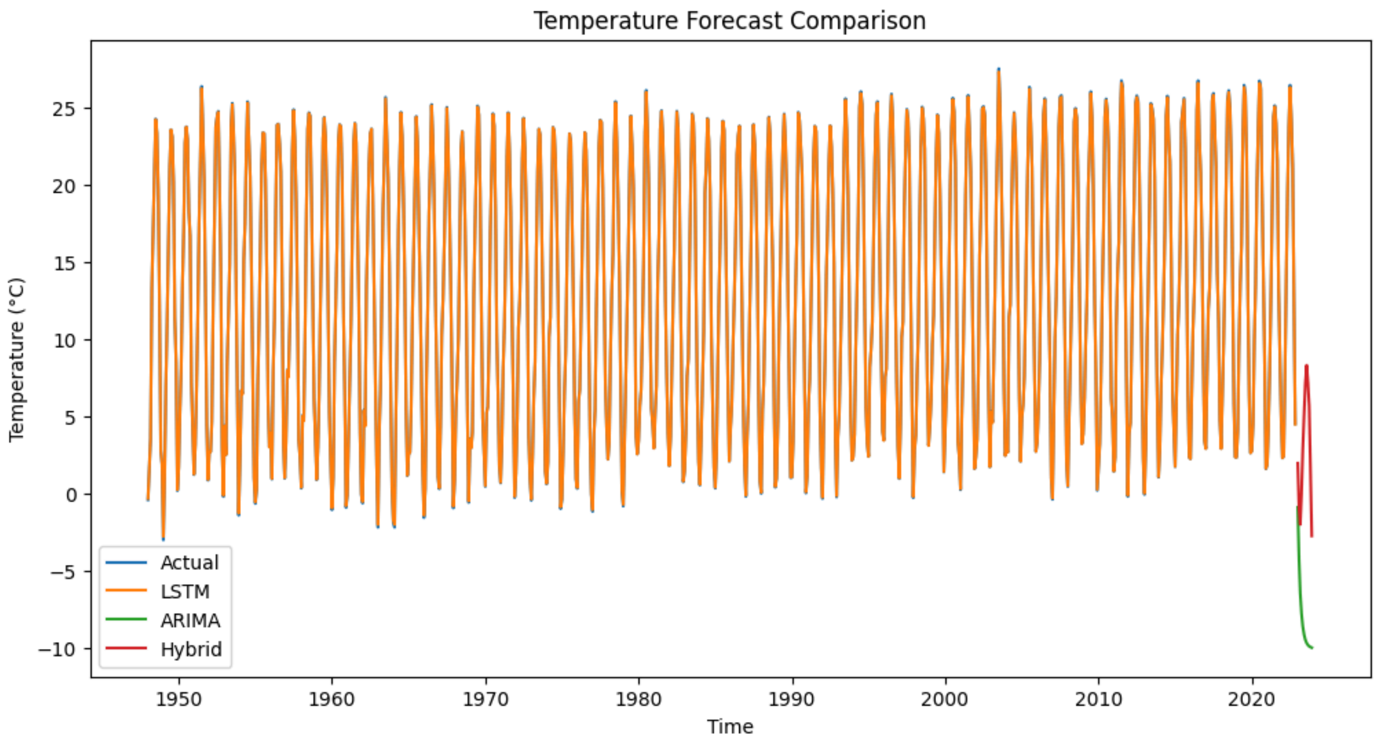
```
from sklearn.metrics import mean_squared_error
import math

# Unscale LSTM predictions
unscaled_predictions = scaler.inverse_transform(lstm_predictions).flatten()
```

```
# Calculate RMSE for ARIMA
arima_rmse = math.sqrt(mean_squared_error(city_data['albuquerque'][-12:], forecast))
print(f"ARIMA RMSE: {arima_rmse}")

# Calculate RMSE for LSTM
lstm_rmse = math.sqrt(mean_squared_error(city_data['albuquerque'], unscaled_predictions))
print(f"LSTM RMSE: {lstm_rmse}")

# Hybrid model
hybrid_predictions = (forecast.values + unscaled_predictions[-12:]) / 2
hybrid_rmse = math.sqrt(mean_squared_error(city_data['albuquerque'][-12:], hybrid_predictions))
print(f"Hybrid RMSE: {hybrid_rmse}")
```

```
ARIMA RMSE: 24.14068530672941
LSTM RMSE: 0.046944103405702244
Hybrid RMSE: 12.07835539563155
```

```
plt.figure(figsize=(12, 6))
plt.plot(city_data.index, city_data['albuquerque'], label='Actual')
plt.plot(city_data.index, unscaled_predictions, label='LSTM')
plt.plot(pd.date_range(start=city_data.index[-1], periods=13, freq='M')[1:], forecast, label='ARIMA')
plt.plot(pd.date_range(start=city_data.index[-1], periods=13, freq='M')[1:], hybrid_predictions, label='Hybrid')
plt.title('Temperature Forecast Comparison')
plt.xlabel('Time')
plt.ylabel('Temperature (°C)')
plt.legend()
plt.show()
```

```
<ipython-input-34-d7fa32016731>:4: FutureWarning: 'M' is deprecated and will be removed in a future version, please u
  plt.plot(pd.date_range(start=city_data.index[-1], periods=13, freq='M')[1:], forecast, label='ARIMA')
<ipython-input-34-d7fa32016731>:5: FutureWarning: 'M' is deprecated and will be removed in a future version, please u
  plt.plot(pd.date_range(start=city_data.index[-1], periods=13, freq='M')[1:], hybrid_predictions, label='Hybrid')
```



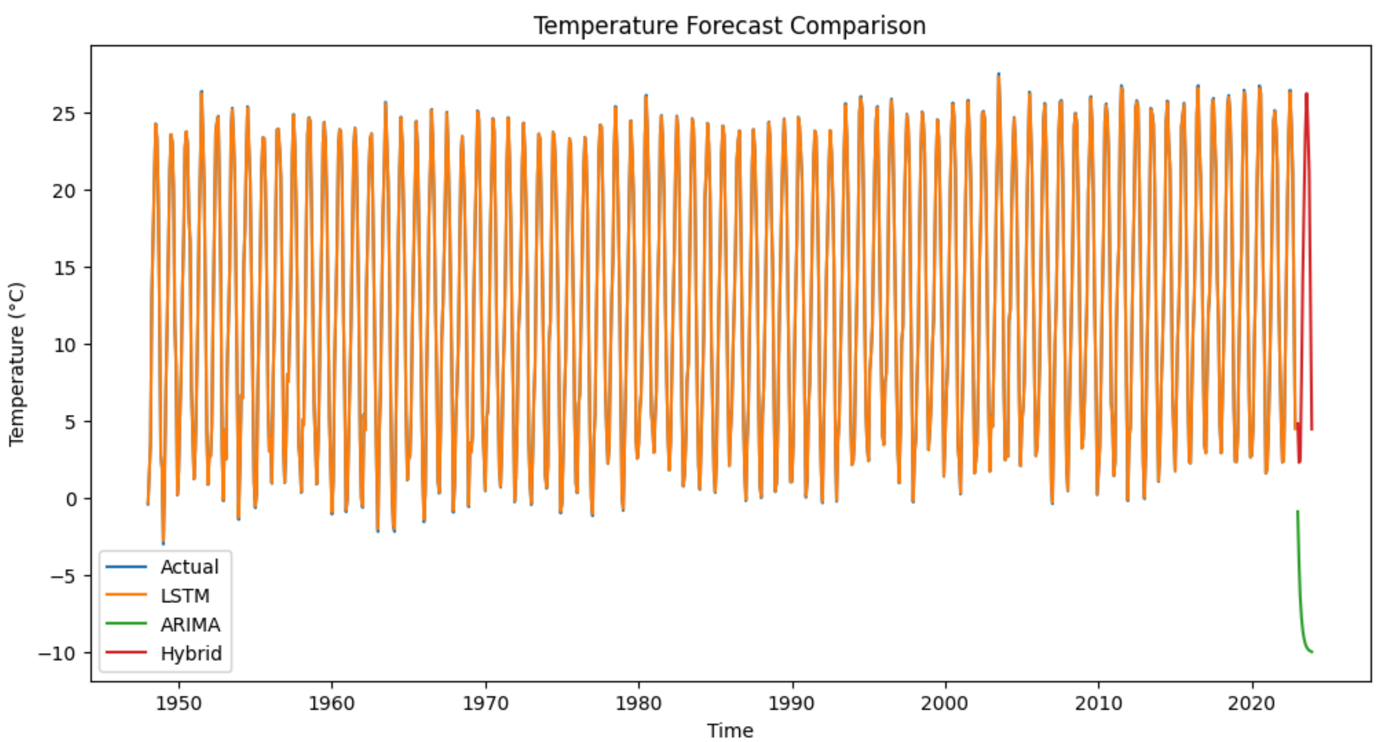

```
total_rmse = arima_rmse + lstm_rmse
lstm_weight = 1 - (lstm_rmse / total_rmse)
arima_weight = 1 - (arima_rmse / total_rmse)

# Normalize weights to sum to 1
sum_weights = lstm_weight + arima_weight
lstm_weight = lstm_weight / sum_weights
arima_weight = arima_weight / sum_weights
```

```
print(f"LSTM Weight: {lstm_weight:.4f}")
print(f"ARIMA Weight: {arima_weight:.4f}")

# Weighted hybrid model
hybrid_predictions = (arima_weight * forecast.values +
                     lstm_weight * unscaled_predictions[-12:])

# Calculate RMSE for weighted hybrid
hybrid_rmse = math.sqrt(mean_squared_error(city_data['albuquerque'][-12:],
                                           hybrid_predictions))
print(f"Weighted Hybrid RMSE: {hybrid_rmse}")
```

```
LSTM Weight: 0.9981
ARIMA Weight: 0.0019
Weighted Hybrid RMSE: 0.079295779332574
```

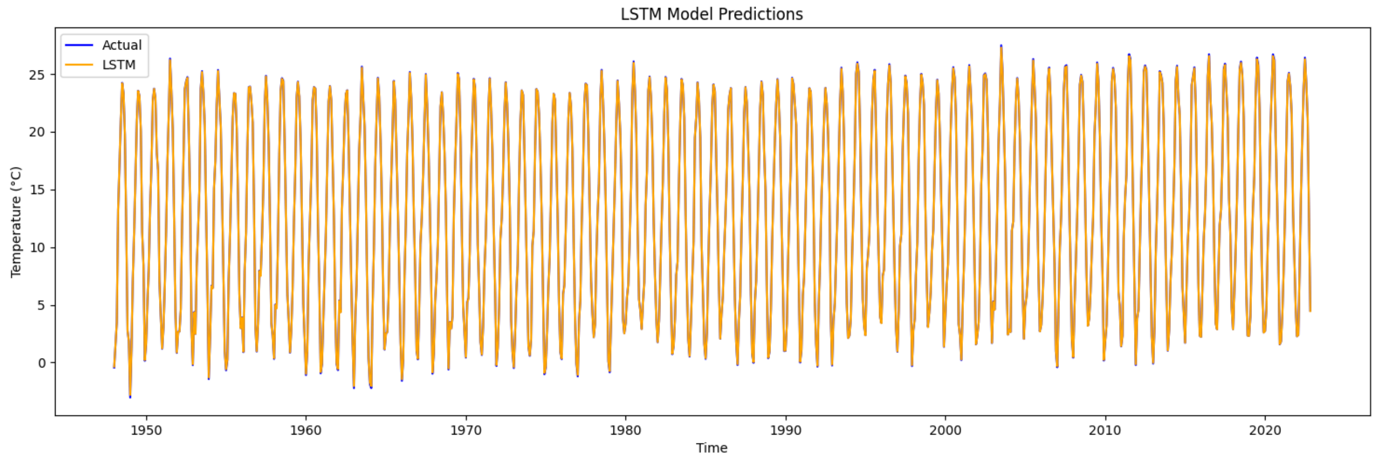
```
plt.figure(figsize=(12, 6))
plt.plot(city_data.index, city_data['albuquerque'], label='Actual')
plt.plot(city_data.index, unscaled_predictions, label='LSTM')
plt.plot(pd.date_range(start=city_data.index[-1], periods=13, freq='M')[1:], forecast, label='ARIMA')
plt.plot(pd.date_range(start=city_data.index[-1], periods=13, freq='M')[1:], hybrid_predictions, label='Hybrid')
plt.title('Temperature Forecast Comparison')
plt.xlabel('Time')
plt.ylabel('Temperature (°C)')
plt.legend()
plt.show()
```

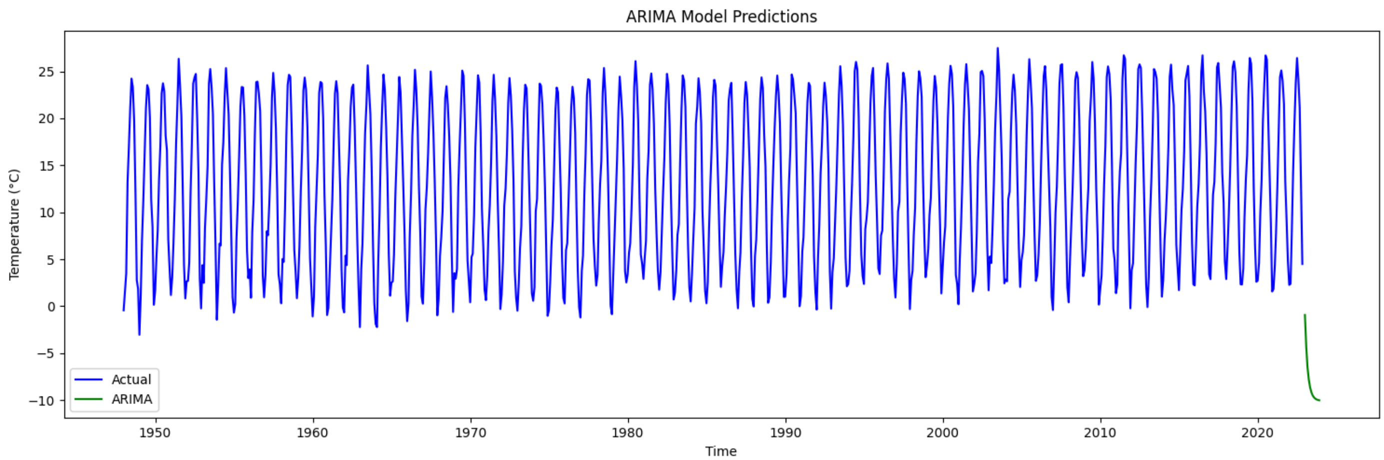
```
<ipython-input-36-d7fa32016731>:4: FutureWarning: 'M' is deprecated and will be removed in a future version, please u
  plt.plot(pd.date_range(start=city_data.index[-1], periods=13, freq='M')[1:], forecast, label='ARIMA')
<ipython-input-36-d7fa32016731>:5: FutureWarning: 'M' is deprecated and will be removed in a future version, please u
  plt.plot(pd.date_range(start=city_data.index[-1], periods=13, freq='M')[1:], hybrid_predictions, label='Hybrid')
```

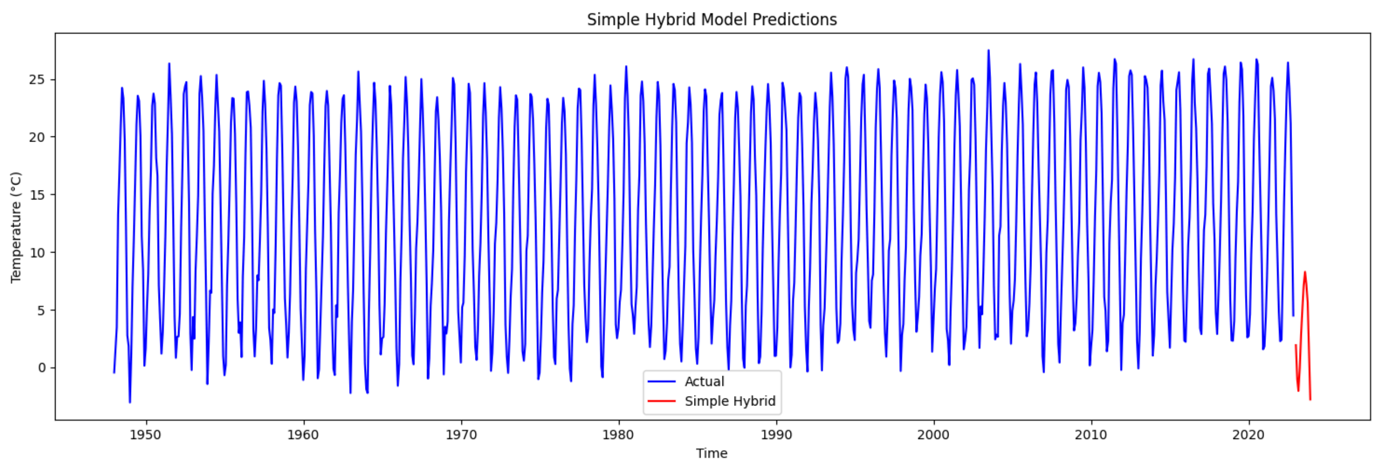


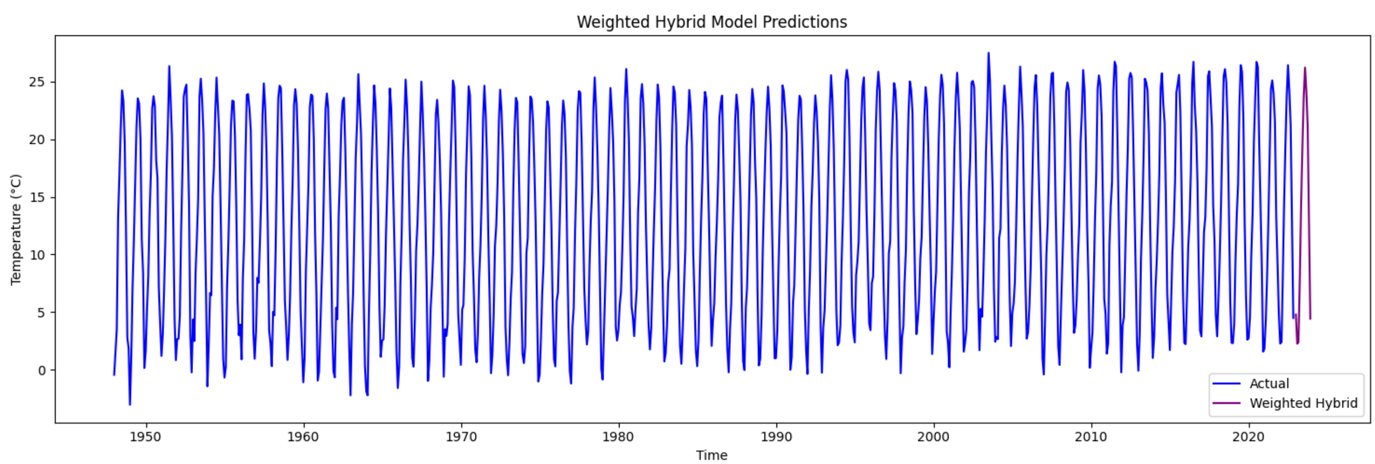

```
import matplotlib.pyplot as plt

# Create a figure with subplots
plt.figure(figsize=(15, 20))

# Plot 1: LSTM Predictions
plt.subplot(4, 1, 1)
plt.plot(city_data.index, city_data['albuquerque'], label='Actual', color='blue')
plt.plot(city_data.index, unscaled_predictions, label='LSTM', color='orange')
plt.title('LSTM Model Predictions')
```

```
plt.xlabel('Time')
plt.ylabel('Temperature (°C)')
plt.legend()

# Plot 2: ARIMA Predictions
plt.subplot(4, 1, 2)
plt.plot(city_data.index, city_data['albuquerque'], label='Actual', color='blue')
plt.plot(pd.date_range(start=city_data.index[-1], periods=13, freq='ME')[1:],
         forecast, label='ARIMA', color='green')
plt.title('ARIMA Model Predictions')
plt.xlabel('Time')
plt.ylabel('Temperature (°C)')
plt.legend()

# Plot 3: Simple Hybrid Predictions
plt.subplot(4, 1, 3)
plt.plot(city_data.index, city_data['albuquerque'], label='Actual', color='blue')
plt.plot(pd.date_range(start=city_data.index[-1], periods=13, freq='ME')[1:],
         (forecast.values + unscaled_predictions[-12:]) / 2, label='Simple Hybrid', color='red')
plt.title('Simple Hybrid Model Predictions')
plt.xlabel('Time')
plt.ylabel('Temperature (°C)')
plt.legend()

# Plot 4: Weighted Hybrid Predictions
plt.subplot(4, 1, 4)
plt.plot(city_data.index, city_data['albuquerque'], label='Actual', color='blue')
plt.plot(pd.date_range(start=city_data.index[-1], periods=13, freq='ME')[1:],
         hybrid_predictions, label='Weighted Hybrid', color='purple')
plt.title('Weighted Hybrid Model Predictions')
plt.xlabel('Time')
plt.ylabel('Temperature (°C)')
plt.legend()

# Adjust layout
plt.tight_layout()
plt.show()

# Print model performance metrics
print("\nModel Performance Metrics:")
print(f"LSTM RMSE: {lstm_rmse:.4f}")
print(f"ARIMA RMSE: {arima_rmse:.4f}")
print(f"Simple Hybrid RMSE: {math.sqrt(mean_squared_error(city_data['albuquerque'][-12:], (forecast.values + unscaled_pr
print(f"Weighted Hybrid RMSE: {hybrid_rmse:.4f}")
print(f"\nModel Weights:")
print(f"LSTM Weight: {lstm_weight:.4f}")
print(f"ARIMA Weight: {arima_weight:.4f}")
```










```
Model Performance Metrics:
LSTM RMSE: 0.0469
ARIMA RMSE: 24.1407
```

```
Simple Hybrid RMSE: 12.0784
Weighted Hybrid RMSE: 0.0793

Model Weights:
LSTM Weight: 0.9981
ARIMA Weight: 0.0019
```

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from statsmodels.tsa.arima.model import ARIMA
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
from sklearn.metrics import mean_squared_error
import math

def create_lstm_model():
    model = Sequential([
        LSTM(50, return_sequences=True, input_shape=(1, 1)),
        LSTM(50, return_sequences=False),
        Dense(1)
    ])
    model.compile(optimizer='adam', loss='mean_squared_error')
    return model

def analyze_city(city_data, city_name):
    # Prepare data
    scaler = StandardScaler()
    scaled_data = scaler.fit_transform(city_data[[city_name]])

    # LSTM
    X_train = scaled_data.reshape((len(scaled_data), 1, 1))
    model = create_lstm_model()
    model.fit(X_train, scaled_data, epochs=50, batch_size=32, verbose=0)
    lstm_predictions = model.predict(X_train)
    unscaled_predictions = scaler.inverse_transform(lstm_predictions).flatten()

    # ARIMA
    arima_model = ARIMA(city_data[city_name], order=(1,1,1))
    arima_results = arima_model.fit()
    forecast = arima_results.forecast(steps=12)

    # Calculate metrics
    lstm_rmse = math.sqrt(mean_squared_error(city_data[city_name], unscaled_predictions))
    arima_rmse = math.sqrt(mean_squared_error(city_data[city_name][-12:], forecast))

    # Weighted hybrid
    total_rmse = arima_rmse + lstm_rmse
    lstm_weight = 1 - (lstm_rmse / total_rmse)
    arima_weight = 1 - (arima_rmse / total_rmse)

    # Normalize weights
    sum_weights = lstm_weight + arima_weight
    lstm_weight = lstm_weight / sum_weights
    arima_weight = arima_weight / sum_weights

    hybrid_predictions = (arima_weight * forecast.values +
                          lstm_weight * unscaled_predictions[-12:])
    hybrid_rmse = math.sqrt(mean_squared_error(city_data[city_name][-12:],
                                               hybrid_predictions))

    return {
        'city': city_name,
        'lstm_rmse': lstm_rmse,
        'arima_rmse': arima_rmse,
        'hybrid_rmse': hybrid_rmse,
        'lstm_weight': lstm_weight,
        'arima_weight': arima_weight
```

```
    }

# Main execution
def run_multi_city_analysis(temp_data):
    results = []
    cities = temp_data.columns[1:]  # Exclude 'time' column

    for city in cities:
        city_data = temp_data[['time', city]].dropna()
        city_data.set_index('time', inplace=True)
        result = analyze_city(city_data, city)
        results.append(result)

    return pd.DataFrame(results)

# Run analysis
results_df = run_multi_city_analysis(temp_data)
print(results_df.sort_values('hybrid_rmse'))
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204: UserWarning: Do not pass an `input_shape`/`:
  super().__init__(**kwargs)
29/29 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: No frequency informatio
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: No frequency informatio
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: No frequency informatio
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204: UserWarning: Do not pass an `input_shape`/`:
  super().__init__(**kwargs)
29/29 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: No frequency informatio
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: No frequency informatio
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: No frequency informatio
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204: UserWarning: Do not pass an `input_shape`/`:
  super().__init__(**kwargs)
29/29 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: No frequency informatio
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: No frequency informatio
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: No frequency informatio
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204: UserWarning: Do not pass an `input_shape`/`:
  super().__init__(**kwargs)
29/29 ━━━━━━━━━━━━━━━━━━━━ 1s 18ms/step
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: No frequency informatio
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: No frequency informatio
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: No frequency informatio
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204: UserWarning: Do not pass an `input_shape`/`:
  super().__init__(**kwargs)
29/29 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: No frequency informatio
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: No frequency informatio
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: No frequency informatio
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204: UserWarning: Do not pass an `input_shape`/`:
  super().__init__(**kwargs)
29/29 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: No frequency informatio
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: No frequency informatio
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: No frequency informatio
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204: UserWarning: Do not pass an `input_shape`/`:
  super().__init__(**kwargs)
29/29 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: No frequency informatio
```