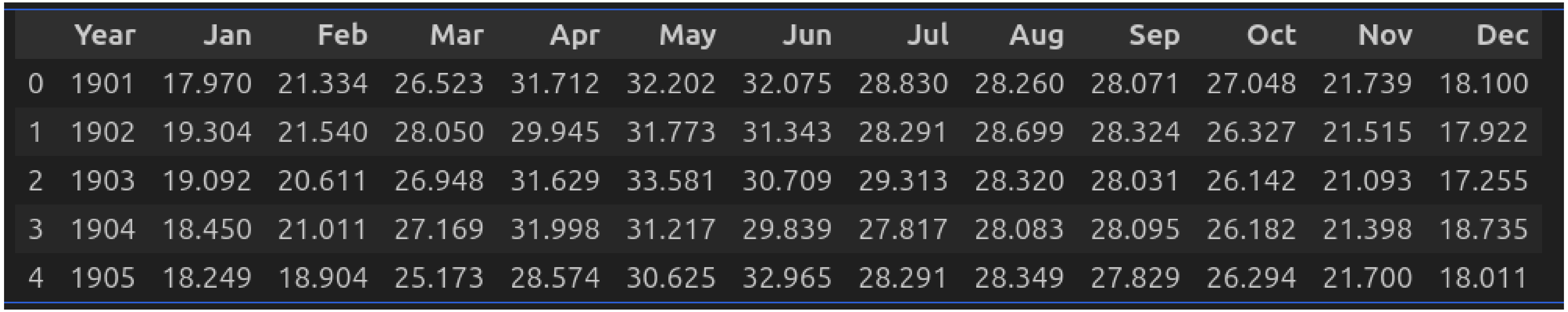
Dataset

Given data set(18 Distircts):

it contain 1901 to 2002 year (row) and Jan to Dec (column)

ex: Bankura.csv



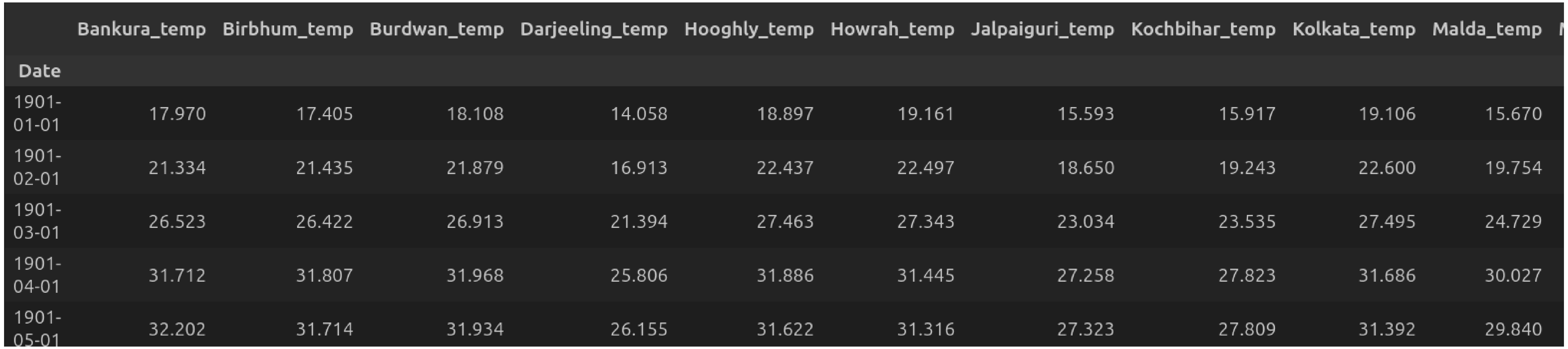
	Year	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
0	1901	17.970	21.334	26.523	31.712	32.202	32.075	28.830	28.260	28.071	27.048	21.739	18.100
1	1902	19.304	21.540	28.050	29.945	31.773	31.343	28.291	28.699	28.324	26.327	21.515	17.922
2	1903	19.092	20.611	26.948	31.629	33.581	30.709	29.313	28.320	28.031	26.142	21.093	17.255
3	1904	18.450	21.011	27.169	31.998	31.217	29.839	27.817	28.083	28.095	26.182	21.398	18.735
4	1905	18.249	18.904	25.173	28.574	30.625	32.965	28.291	28.349	27.829	26.294	21.700	18.011

↓

Modified dataset (18 District Marge with single tmp variable):

it contain 1901-1-1 to 2002-12-1 year (row) and Bankura_temp ...to all 18 district (column)

ex: All_district.csv

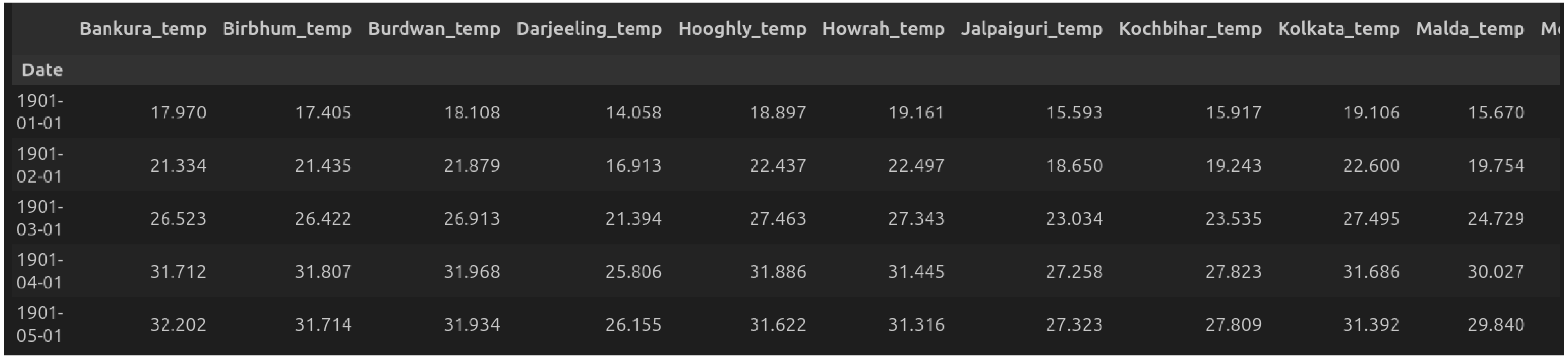


	Bankura_temp	Birbhum_temp	Burdwan_temp	Darjeeling_temp	Hooghly_temp	Howrah_temp	Jalpaiguri_temp	Kochbihar_temp	Kolkata_temp	Malda_temp
Date										
1901-01-01	17.970	17.405	18.108	14.058	18.897	19.161	15.593	15.917	19.106	15.670
1901-02-01	21.334	21.435	21.879	16.913	22.437	22.497	18.650	19.243	22.600	19.754
1901-03-01	26.523	26.422	26.913	21.394	27.463	27.343	23.034	23.535	27.495	24.729
1901-04-01	31.712	31.807	31.968	25.806	31.886	31.445	27.258	27.823	31.686	30.027
1901-05-01	32.202	31.714	31.934	26.155	31.622	31.316	27.323	27.809	31.392	29.840

Train - Test

Train(1104, 18):

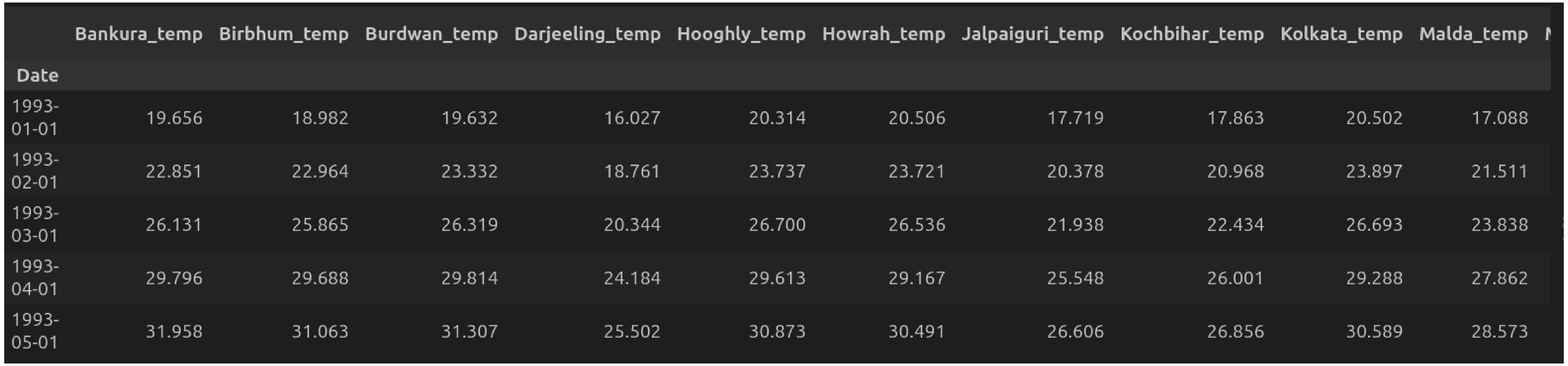
Train dataset [1901 - 1 - 1] → [1992 - 12- 1] Total(92 years)



	Bankura_temp	Birbhum_temp	Burdwan_temp	Darjeeling_temp	Hooghly_temp	Howrah_temp	Jalpaiguri_temp	Kochbihar_temp	Kolkata_temp	Malda_temp
Date										
1901-01-01	17.970	17.405	18.108	14.058	18.897	19.161	15.593	15.917	19.106	15.670
1901-02-01	21.334	21.435	21.879	16.913	22.437	22.497	18.650	19.243	22.600	19.754
1901-03-01	26.523	26.422	26.913	21.394	27.463	27.343	23.034	23.535	27.495	24.729
1901-04-01	31.712	31.807	31.968	25.806	31.886	31.445	27.258	27.823	31.686	30.027
1901-05-01	32.202	31.714	31.934	26.155	31.622	31.316	27.323	27.809	31.392	29.840

Test(120, 18):

Test dataset [1993 - 1 - 1] → [2002 - 12 - 1] Total(10 years)



	Bankura_temp	Birbhum_temp	Burdwan_temp	Darjeeling_temp	Hooghly_temp	Howrah_temp	Jalpaiguri_temp	Kochbihar_temp	Kolkata_temp	Malda_temp
Date										
1993-01-01	19.656	18.982	19.632	16.027	20.314	20.506	17.719	17.863	20.502	17.088
1993-02-01	22.851	22.964	23.332	18.761	23.737	23.721	20.378	20.968	23.897	21.511
1993-03-01	26.131	25.865	26.319	20.344	26.700	26.536	21.938	22.434	26.693	23.838
1993-04-01	29.796	29.688	29.814	24.184	29.613	29.167	25.548	26.001	29.288	27.862
1993-05-01	31.958	31.063	31.307	25.502	30.873	30.491	26.606	26.856	30.589	28.573

Sample code

```
import datetime
from dateutil.relativedelta import relativedelta # Import relativedelta


train_end_date = datetime.datetime(1992, 12, 1)

test_start = train_end_date + relativedelta(months=1)

train_data=df[:train_end_date]
test_data=df[test_start:]
```

Sarima Model

Sarima Model with no Exog

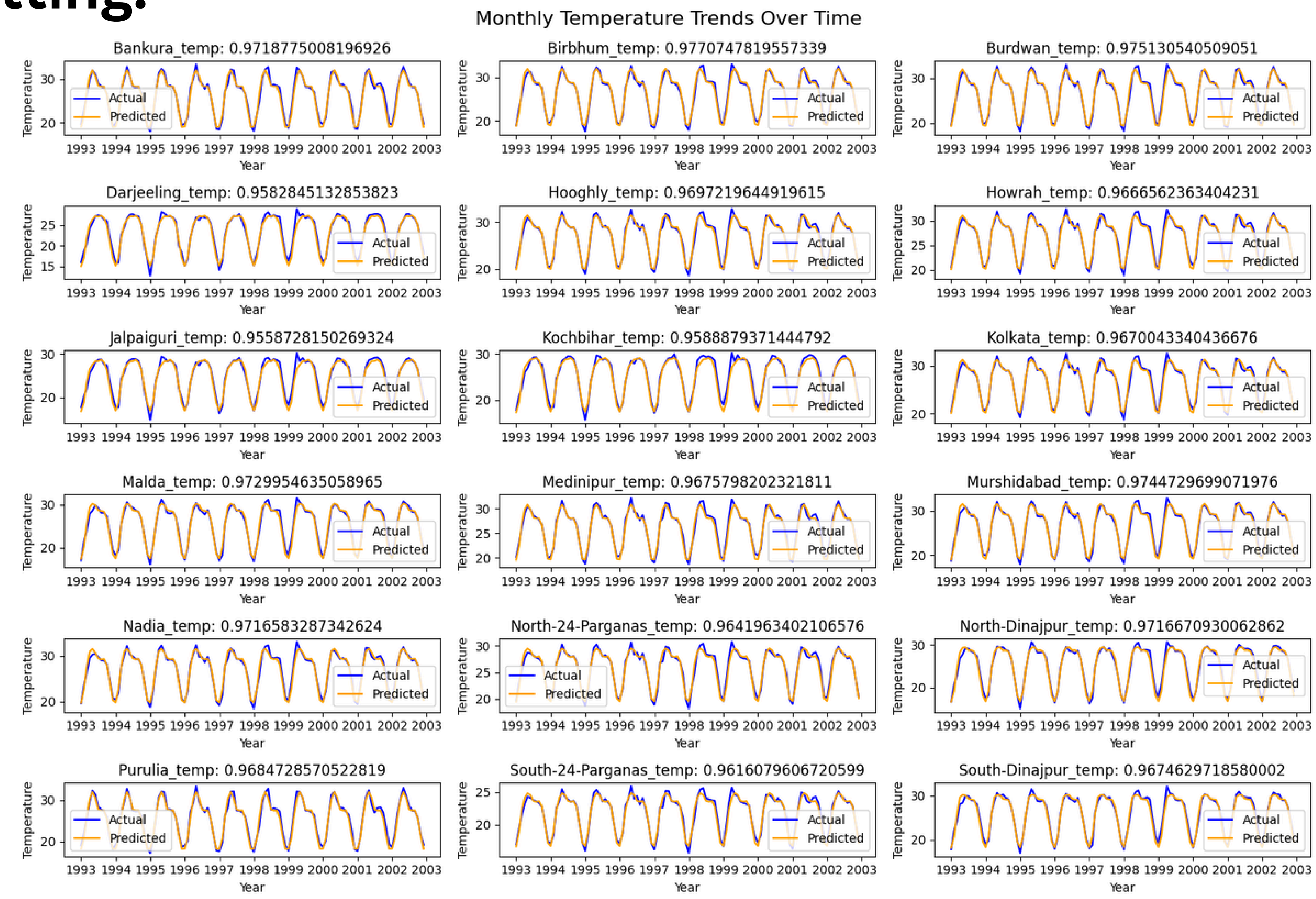
# SARIMA order
my_order = (2,1,1)
my_seasonal = (1, 1, 1, 12)

Train Data → Train Data(from actual dataset)
Test Data → Test Data (from actual dataset)
Frocast -> For each districts it forcast len(Test data) in one time

R2 value:

Bankura_temp: 0.9718775008196926
Birbhum_temp: 0.9770747819557339
Burdwan_temp: 0.975130540509051
Darjeeling_temp: 0.9582845132853823
Hooghly_temp: 0.9697219644919615
Howrah_temp: 0.9666562363404231
Jalpaiguri_temp: 0.9558728150269324
Kochbihar_temp: 0.9588879371444792
Kolkata_temp: 0.9670043340436676
Malda_temp: 0.9729954635058965
Medinipur_temp: 0.9675798202321811
Murshidabad_temp: 0.9744729699071976
Nadia_temp: 0.9716583287342624
North-24-Parganas_temp: 0.9641963402106576
North-Dinajpur_temp: 0.9716670930062862
Purulia_temp: 0.9684728570522819
South-24-Parganas_temp: 0.9616079606720599
South-Dinajpur_temp: 0.9674629718580002

Plotting:



Sample Code

```
import pandas as pd
import numpy as np
from statsmodels.tsa.statespace.sarimax import SARIMAX
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import r2_score

# Create a copy to store predictions
predict_result_no_mod = test_data.copy()

# Define SARIMA order
my_order = (2,1,1)
my_seasonal = (1, 1, 1, 12)

# Loop over all time series columns
for col in train_data.columns:
 print(f"Processing column: {col}")

 # Initialize SARIMAX Model
 try:
 model = SARIMAX(train_data[col], order=my_order, seasonal_order=my_seasonal)
 model_fit = model.fit(disp=False)\

 # Generate predictions
 pred = model_fit.predict(start=test_data.index[0], end=test_data.index[-1])

 # Store predictions
 predict_result_no_mod[col] = pred

 # Calculate R² Score
 r2 = r2_score(test_data[col], pred)
 print(f"R² Score for {col}: {r2:.4f}")

 except np.linalg.LinAlgError:
 print(f"LU Decomposition failed for {col}. Skipping...")
 continue
```

Dtw Result

Bankura_temp -> ['Birbhum_temp' 'Burdwan_temp' 'Hooghly_temp' 'Howrah_temp' 'Kolkata_temp' 'Malda_temp' 'Medinipur_temp' 'Murshidabad_temp' 'Nadia_temp' 'North-24-Parganas_temp' 'Purulia_temp' 'South-Dinajpur_temp']

Birbhum_temp -> ['Burdwan_temp' 'Hooghly_temp' 'Howrah_temp' 'Kolkata_temp' 'Malda_temp' 'Medinipur_temp' 'Murshidabad_temp' 'Nadia_temp' 'North-24-Parganas_temp' 'Purulia_temp' 'South-Dinajpur_temp' 'Bankura_temp']

Burdwan_temp -> ['Birbhum_temp' 'Hooghly_temp' 'Howrah_temp' 'Kolkata_temp' 'Malda_temp' 'Medinipur_temp' 'Murshidabad_temp' 'Nadia_temp' 'North-24-Parganas_temp' 'Purulia_temp' 'South-Dinajpur_temp' 'Bankura_temp']

Darjeeling_temp -> ['Jalpaiguri_temp']

Hooghly_temp -> ['Birbhum_temp' 'Burdwan_temp' 'Howrah_temp' 'Kolkata_temp' 'Medinipur_temp' 'Murshidabad_temp' 'Nadia_temp' 'North-24-Parganas_temp' 'Purulia_temp' 'South-Dinajpur_temp' 'Bankura_temp']

Howrah_temp -> ['Birbhum_temp' 'Burdwan_temp' 'Hooghly_temp' 'Kolkata_temp' 'Medinipur_temp' 'Murshidabad_temp' 'Nadia_temp' 'North-24-Parganas_temp' 'South-Dinajpur_temp' 'Bankura_temp']

Jalpaiguri_temp -> ['Malda_temp' 'North-24-Parganas_temp' 'South-Dinajpur_temp' 'Darjeeling_temp' 'Kochbihar_temp' 'North-Dinajpur_temp']

Kochbihar_temp -> ['Malda_temp' 'Medinipur_temp' 'North-24-Parganas_temp' 'South-Dinajpur_temp' 'Jalpaiguri_temp' 'North-Dinajpur_temp']

Kolkata_temp -> ['Birbhum_temp' 'Burdwan_temp' 'Hooghly_temp' 'Howrah_temp' 'Medinipur_temp' 'Murshidabad_temp' 'Nadia_temp' 'North-24-Parganas_temp' 'South-Dinajpur_temp' 'Bankura_temp']

Malda_temp -> ['Birbhum_temp' 'Burdwan_temp' 'Medinipur_temp' 'Murshidabad_temp' 'North-24-Parganas_temp' 'Purulia_temp' 'South-Dinajpur_temp' 'Bankura_temp' 'Jalpaiguri_temp' 'Kochbihar_temp' 'North-Dinajpur_temp']

Medinipur_temp -> ['Birbhum_temp' 'Burdwan_temp' 'Hooghly_temp' 'Howrah_temp' 'Kolkata_temp' 'Malda_temp' 'Murshidabad_temp' 'Nadia_temp' 'North-24-Parganas_temp' 'Purulia_temp' 'South-Dinajpur_temp' 'Bankura_temp' 'Kochbihar_temp']

Murshidabad_temp -> ['Birbhum_temp' 'Burdwan_temp' 'Hooghly_temp' 'Howrah_temp' 'Kolkata_temp' 'Malda_temp' 'Medinipur_temp' 'Nadia_temp' 'North-24-Parganas_temp' 'Purulia_temp' 'South-Dinajpur_temp' 'Bankura_temp']

Dtw Result

Nadia_temp -> ['Birbhum_temp' 'Burdwan_temp' 'Hooghly_temp' 'Howrah_temp' 'Kolkata_temp' 'Medinipur_temp' 'Murshidabad_temp' 'North-24-Parganas_temp' 'Purulia_temp' 'South-Dinajpur_temp' 'Bankura_temp']

North_24_Parganas_temp -> ['Birbhum_temp' 'Burdwan_temp' 'Hooghly_temp' 'Howrah_temp' 'Kolkata_temp' 'Malda_temp' 'Medinipur_temp' 'Murshidabad_temp' 'Nadia_temp' 'South-Dinajpur_temp' 'Bankura_temp' 'Jalpaiguri_temp' 'Kochbihar_temp' 'North-Dinajpur_temp']

North-Dinajpur_temp -> ['Malda_temp' 'North-24-Parganas_temp' 'South-Dinajpur_temp' 'Jalpaiguri_temp' 'Kochbihar_temp']

Purulia_temp -> ['Birbhum_temp' 'Burdwan_temp' 'Hooghly_temp' 'Malda_temp' 'Medinipur_temp' 'Murshidabad_temp' 'Nadia_temp' 'South-Dinajpur_temp' 'Bankura_temp']

South-24-Parganas_temp -> []

South-Dinajpur_temp -> ['Birbhum_temp' 'Burdwan_temp' 'Hooghly_temp' 'Howrah_temp' 'Kolkata_temp' 'Malda_temp' 'Medinipur_temp' 'Murshidabad_temp' 'Nadia_temp' 'North-24-Parganas_temp' 'Purulia_temp' 'Bankura_temp' 'Jalpaiguri_temp' 'Kochbihar_temp' 'North-Dinajpur_temp']

Sarima Model with Exog

# SARIMA order

my_order = (2,1,1)

my_seasonal = (1, 1, 1, 12)

Train Data → Train Data(from actual dataset)

Test Data → Test Data (from actual dataset)

Exog → previous predicted sarima with no mod use base on **Dtw** result

Frocast → For each districts it forcast len(Test data) in one time

R2 value:

Bankura_temp: 0.9716255231870972

Birbhum_temp: 0.9772328482608365

Burdwan_temp: 0.9749751719323221

Darjeeling_temp: 0.9623650275218317

Hooghly_temp: 0.9695010877530159

Howrah_temp: 0.9669450387916823

Jalpaiguri_temp: 0.9515157812747885

Kochbihar_temp: 0.9573916125713449

Kolkata_temp: 0.9674372027291505

Malda_temp: 0.9719031139606114

Medinipur_temp: 0.9646302816611658

Murshidabad_temp: 0.9745310866596398

Nadia_temp: 0.9718513966808351

North-24-Parganas_temp: 0.9632328585911818

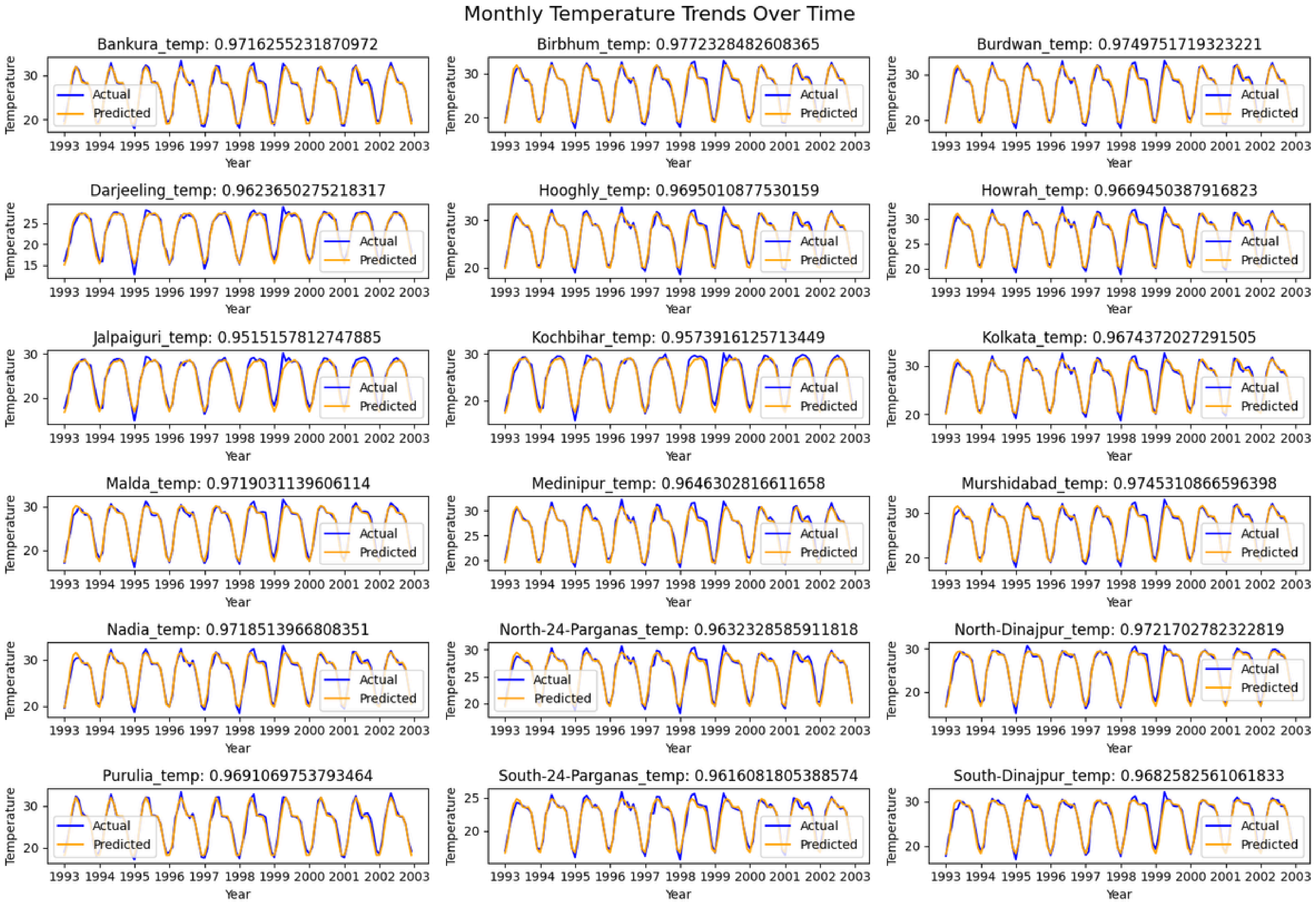
North-Dinajpur_temp: 0.9721702782322819

Purulia_temp: 0.9691069753793464

South-24-Parganas_temp: 0.9616081805388574

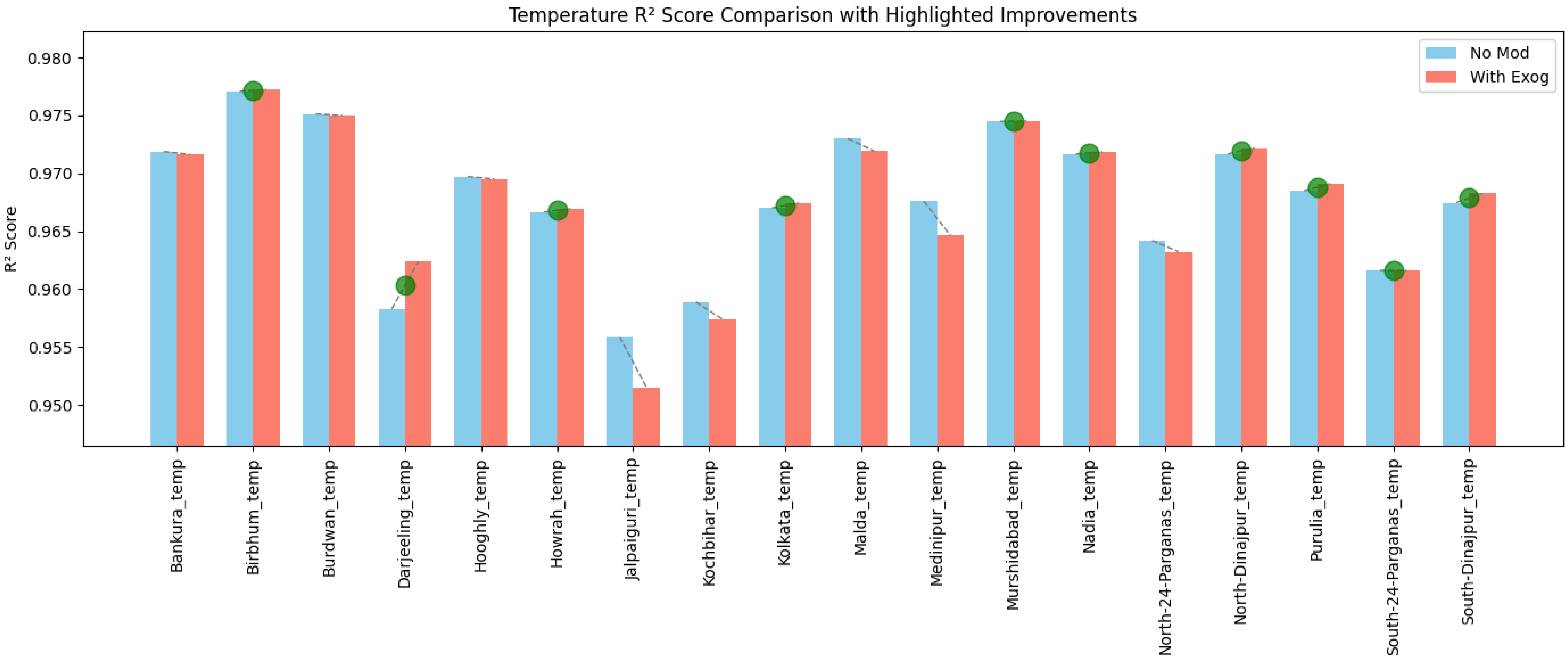
South-Dinajpur_temp: 0.9682582561061833

Plotting:



comparison

Sarima with no exog vs Sarima with exog:



Sample Code

```
import pandas as pd
import numpy as np
from statsmodels.tsa.statespace.sarimax import SARIMAX
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import r2_score

# Create a dictionary to store scalers for each column
scalers = {}

# Define SARIMA order
my_order = (2 , 1, 1)
my_seasonal = (1, 1, 1, 12)

# Copy test_data to store predictions
predict_result_exog_pred = test_data.copy()

# Loop over all columns in train_data
for col in train_data.columns:
 print(f"Processing column: {col}")

 # Initialize MinMaxScaler for target variable
 scaler = StandardScaler()

 # Normalize train & test data
 train_scaled = scaler.fit_transform(train_data[[col]]) # Fit & transform train data
 test_scaled = scaler.transform(test_data[[col]]) # Transform test data

 # Convert to Pandas Series (SARIMAX requires time-indexed series)
 train_scaled_series = pd.Series(train_scaled.flatten(), index=train_data.index)

 # Store scaler for inverse transformation
 scalers[col] = scaler

 # Get selected districts based on DTW results
 selected_districts = dtw_result_df[col].dropna().index if col in dtw_result_df else []

 # Create exogenous variables for training
```

```
exog_train = train_data[selected_districts] if len(selected_districts) > 0 else None

# Normalize exogenous variables if they exist
if exog_train is not None:
scaler_exog = StandardScaler()
exog_train = pd.DataFrame(scaler_exog.fit_transform(exog_train),
columns=exog_train.columns, index=exog_train.index)

# Initialize exog_test using predicted values from `predict_result2`
exog_test = predict_result_no_mod[selected_districts] if len(selected_districts) > 0 else None

# Apply the same scaling transformation to exog_test
if exog_test is not None:
exog_test = pd.DataFrame(scaler_exog.transform(exog_test),
columns=exog_test.columns, index=exog_test.index)
else:
exog_test = None # Ensure proper handling in SARIMAX

# Initialize SARIMAX model
model = SARIMAX(train_scaled_series, exog=exog_train, order=my_order, seasonal_order=my_seasonal)

# Fit the model
model_fit = model.fit(disp=False)

# Generate predictions
pred_scaled = model_fit.predict(start=test_data.index[0], end=test_data.index[-1], exog=exog_test)

# Inverse transform predictions to original scale
pred_original = scaler.inverse_transform(pred_scaled.values.reshape(-1, 1)).flatten()

# Store predictions
predict_result_exog_pred[col] = pred_original

# Calculate R² score
r2 = r2_score(test_data[col], pred_original)
print(f"R² score for {col}: {r2:.4f}")
```

Sarima with exog and rolling window 1

# SARIMA order

my_order = (2,1,1)

my_seasonal = (1, 1, 1, 12)

Train Data → Train Data(from actual dataset)

Test Data → Test Data (from actual dataset)

exog: it use exog value for train it use full train data + form test(1 month) and test use only one month form previous year same month value

ex: if it forcast 1998 - 2- 1 then train_exog= start to 1998 - 1 - 1
 and test_exog = 1997 - 2 - 1

Frocast -> For each districts it forcast one month at a time

R2 value:

Bankura_temp: 0.9723916165899635

Birbhum_temp: 0.9771054003011164

Burdwan_temp: 0.9749345730504514

Darjeeling_temp: 0.9654659534018565

Hooghly_temp: 0.9693619227544201

Howrah_temp: 0.96661334992619

Jalpaiguri_temp: 0.9654011819611849

Kochbihar_temp: 0.96961140558803

Kolkata_temp: 0.966496792732888

Malda_temp: 0.9759132145655857

Medinipur_temp: 0.9695542244875318

Murshidabad_temp: 0.9756812367887967

Nadia_temp: 0.9725139990879994

North-24-Parganas_temp: 0.9644262173113769

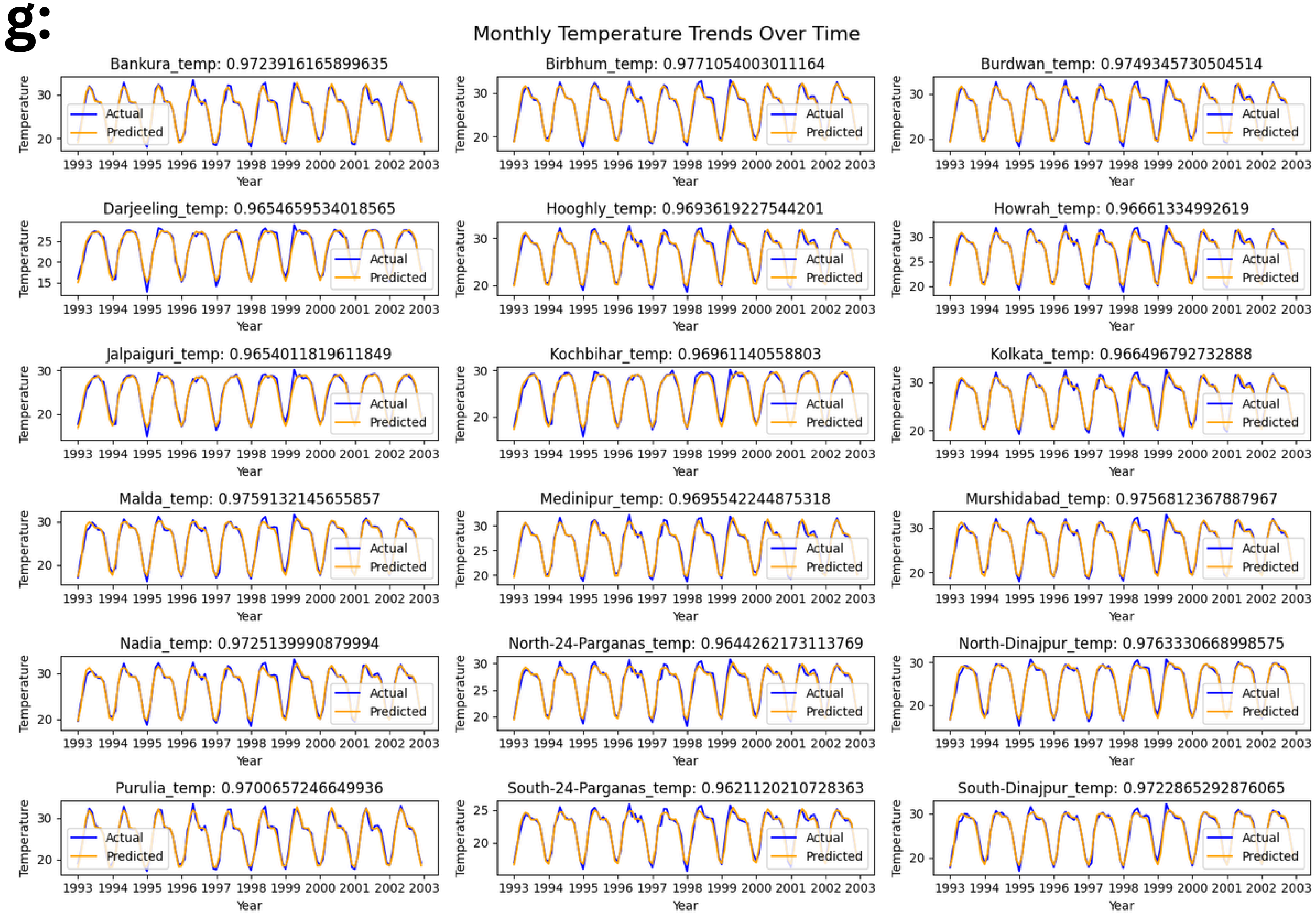
North-Dinajpur_temp: 0.9763330668998575

Purulia_temp: 0.9700657246649936

South-24-Parganas_temp: 0.9621120210728363

South-Dinajpur_temp: 0.9722865292876065

Plotting:



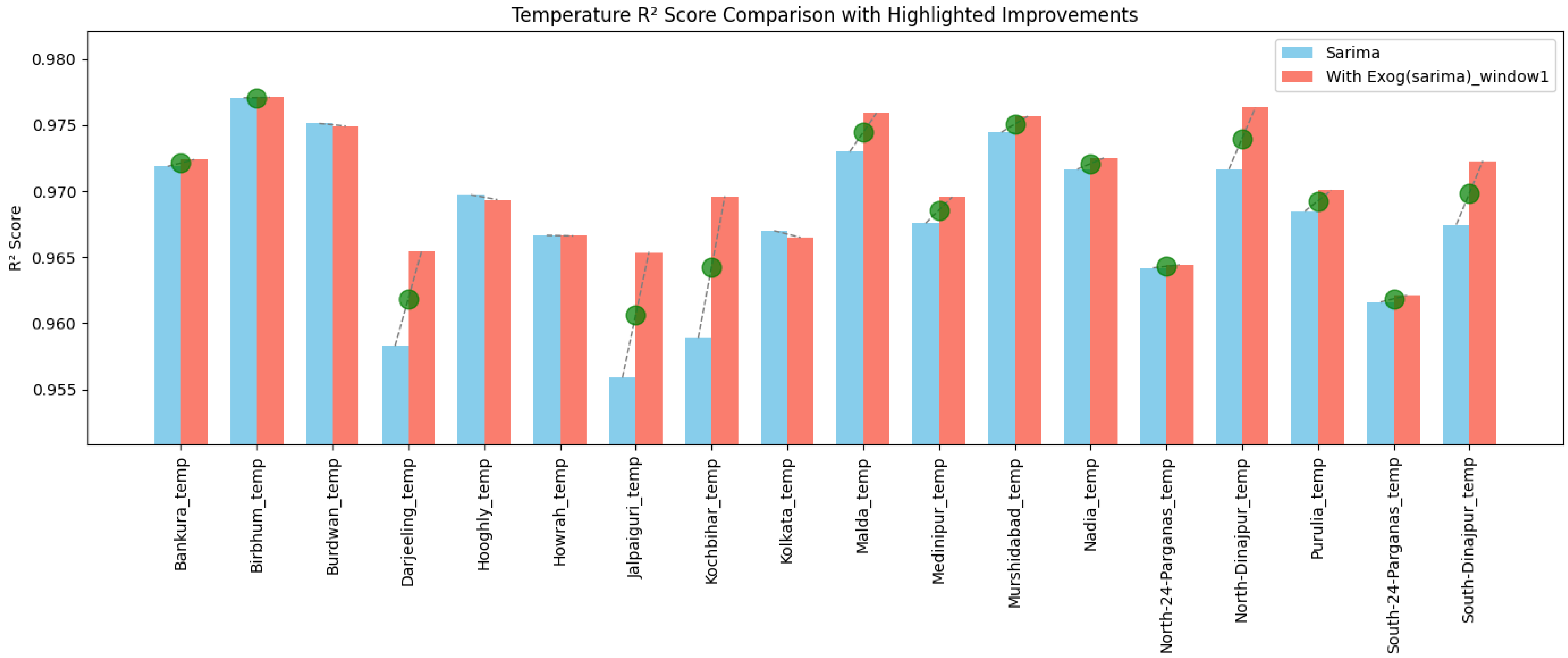
Sample Code

```
from datetime import timedelta
from dateutil.relativedelta import relativedelta
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.statespace.sarimax import SARIMAX
from sklearn.metrics import r2_score


import pandas as pd

# Assuming df_long is your long-format time series with a DateTime index
# Ensure the index has a frequency (e.g., 'MS' for monthly data)
rolling_predictions = test_data.copy()

for col in df.columns:
 df_long = df[col].asfreq('MS')

 # Create a copy of the test data to store rolling predictions
 selected_districts = dtw_result_df[col].dropna().index if col in dtw_result_df else []

 # Iterate over the test data index
 for train_end in test_data.index:
 # Extract training data up to the current train_end (excluding the last observation)
 train_data_roll = df_long[:train_end][:-1]

 exog_train = df[selected_districts][:train_end-relativedelta(months=1)] if len(selected_districts) > 0
else None
      # exog_test = predict_result_no_mod[selected_districts][train_end:train_end] if
len(selected_districts) > 0 else None
          exog_test = df[selected_districts][train_end-relativedelta(months=12):train_end-
relativedelta(months=12)] if len(selected_districts) > 0 else None

 # print(exog_test)

 # Fit the ARIMA model
 my_order = (1,1,1)
 my_seasonal_order = (1, 1, 1, 12)
```

```
model = SARIMAX(train_data_roll, order=my_order, seasonal_order=my_seasonal_order)
model_fit = model.fit() # Use a valid optimization method

# Forecast the next value
pred = model_fit.forecast()

# Store the prediction in the rolling_predictions DataFrame
rolling_predictions.loc[train_end,col] = pred.iloc[0] # Use .iloc[0] to avoid FutureWarning

print(f"Prediction for {train_end}: {pred.iloc[0]}")

# Print the final rolling predictions

# print(rolling_predictions)
r2 = r2_score(test_data[col], rolling_predictions[col])
print(f"R² score for {col}: {r2}")
```

comparison

Sarima with no exog vs Sarima with exog rolling window 1:


Temperature R² Score Comparison with Highlighted Improvements
R² Score
Sarima
With Exog(sarima)_window1
Bankura_temp
Birbhum_temp
Burdwan_temp
Darjeeling_temp
Hooghly_temp
Howrah_temp
Jalpaiguri_temp
Kochbihar_temp
Kolkata_temp
Malda_temp
Medinipur_temp
Murshidabad_temp
Nadia_temp
North-24-Parganas_temp
North-Dinajpur_temp
Purulia_temp
South-24-Parganas_temp
South-Dinajpur_temp