IST_718_Project_Team_Four

August 27, 2022

[1]:
```
# Import packages to use in IST 718 Final Project
import pandas as pd
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt
import re
import copy
import sys
import random
import nltk
from functools import reduce
from scipy.stats import uniform
from plotly.subplots import make_subplots
import datetime
import prophet
from prophet.diagnostics import performance_metrics
from prophet.diagnostics import cross_validation
from prophet.plot import add_changepoints_to_plot, plot_cross_validation_metric
import csv
import io
import textwrap
from sklearn.metrics import r2_score
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from math import sqrt
from scipy.stats import boxcox
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline
from sklearn.linear_model import Lasso
from sklearn.ensemble import RandomForestRegressor
from joblib import dump, load
import skforecast
from skforecast.ForecasterAutoreg import ForecasterAutoreg
from skforecast.ForecasterAutoregCustom import ForecasterAutoregCustom
from skforecast.ForecasterAutoregMultiOutput import ForecasterAutoregMultiOutput
from skforecast.model_selection import grid_search_forecaster
```

```
from skforecast.model_selection import backtesting_forecaster
```

[2]:
```
# load Texas Weather dataset to Pandas Dataframe
df = pd.read_csv('C:/Users/klein/Desktop/Quarter 5 - Current Linked to Backup/
↪IST 718/Project/Project Working Folder/TexasWeatherProjectData1.csv')
```

[3]:
```
# examine dataframe to ensure loaded correctly
pd.options.display.float_format = '{:20,.2f}'.format
df
```

[3]:
```
         Date        ElPasoAvgTemp ElPasoPrecipSum ElPasoSnowSum  \
0    2000-01                49.70               0             T
1    2000-02                53.90            0.03             T
2    2000-03                57.10            0.06             0
3    2000-04                68.30            0.28             0
4    2000-05                79.00               T             0
..       …                    …              …            …
259  2021-08                81.30            2.46             0
260  2021-09                78.10            0.47             0
261  2021-10                68.30               T             0
262  2021-11                57.30            0.34             0
263  2021-12                52.70            0.59             0

          AmarilloAvgTemp AmarilloPrecipSum AmarilloSnowSum  \
0                   39.40              0.24              13
1                   47.10              0.04             1.4
2                   49.20              4.14             1.8
3                   57.80              0.43               T
4                   69.70              1.14               T
..                    …                 …              …
259                 79.40              0.88               0
260                 74.50              0.76               0
261                 62.60              0.65               0
262                 51.40                 0               0
263                 48.20                 0               0

            DallasAvgTemp DallasPrecipSum DallasSnowSum  …  HoustonSnowSum  \
0                   50.50            1.82             M  …               0
1                   57.40            1.72             M  …               0
2                   60.60            3.55             M  …               0
3                   64.90            3.13             M  …               0
4                   77.10             2.9             M  …               0
..                    …               …            …  …             …
259                 85.40            4.06             0  …               0
260                 81.90            0.32             0  …               0
261                 72.10            4.14             0  …               0
262                 57.00            3.36             0  …               0
```

```
263                60.90          0.53           0  …             0

          AustinAvgTemp AustinPrecipSum  AustinSnowSum   BrownsvilleAvgTemp  \
0                 55.30            2.85              M                66.00
1                 62.10            1.75              M                70.30
2                 66.30            1.14              M                74.40
3                 70.30             2.4              M                75.70
4                 79.10            3.25              M                82.50
..                  …               …            …                  …
259               86.00             3.6              0                87.10
260               82.90            1.79              0                84.30
261               74.30             5.3              0                80.10
262               61.60             2.4              0                71.00
263               65.10            1.69              0                73.20

    BrownsvillePrecipSum  BrownsvilleSnowSum       LaredoAvgTemp  \
0                   0.85                   0               62.00
1                   0.19                   0               69.30
2                   2.89                   0               74.50
3                   0.39                   0               77.70
4                   1.87                   0               84.80
..                  …                 …                  …
259                  0.5                   0               88.20
260                 4.64                   0               86.00
261                 9.17                   0               80.00
262                 3.84                   0               68.10
263                 1.32                   0               70.10

    LaredoPrecipSum  LaredoSnowSum
0              0.04              0
1              1.62              0
2              2.26              0
3              1.29              0
4              2.54              0
..              …             …
259            1.93              0
260            1.52              0
261            0.64              0
262            0.62              0
263            0.02              0

[264 rows x 22 columns]
```

[4]:
```python
# Create function to replace T with 0 throughout the data frame (T is for data␣
↪not collected properly) we checked the dataframe and all T values are in␣
↪either precipitation or snow
def ReplaceT(text):
```

```
    text = text.replace('T', 0)
    return text
```

[5]:
```
# Use function on all data frame variables
df = df.apply(ReplaceT)
```

[6]:
```
# Create function to replace M with 0 throughout the data frame (M is for␣
 ↪missing data) we checked the dataframe and all M values are in either␣
 ↪precipitation or snow
def ReplaceM(text):
    text = text.replace('M', 0)
    return text
```

[7]:
```
# Use function on all data frame variables
df = df.apply(ReplaceM)
```

[8]:
```
# check column data types
df.dtypes
```

[8]:
```
Date                        object
ElPasoAvgTemp              float64
ElPasoPrecipSum             object
ElPasoSnowSum               object
AmarilloAvgTemp            float64
AmarilloPrecipSum           object
AmarilloSnowSum             object
DallasAvgTemp              float64
DallasPrecipSum             object
DallasSnowSum               object
HoustonAvgTemp             float64
HoustonPrecipSum           float64
HoustonSnowSum              object
AustinAvgTemp              float64
AustinPrecipSum             object
AustinSnowSum               object
BrownsvilleAvgTemp         float64
BrownsvillePrecipSum        object
BrownsvilleSnowSum          object
LaredoAvgTemp              float64
LaredoPrecipSum             object
LaredoSnowSum               object
dtype: object
```

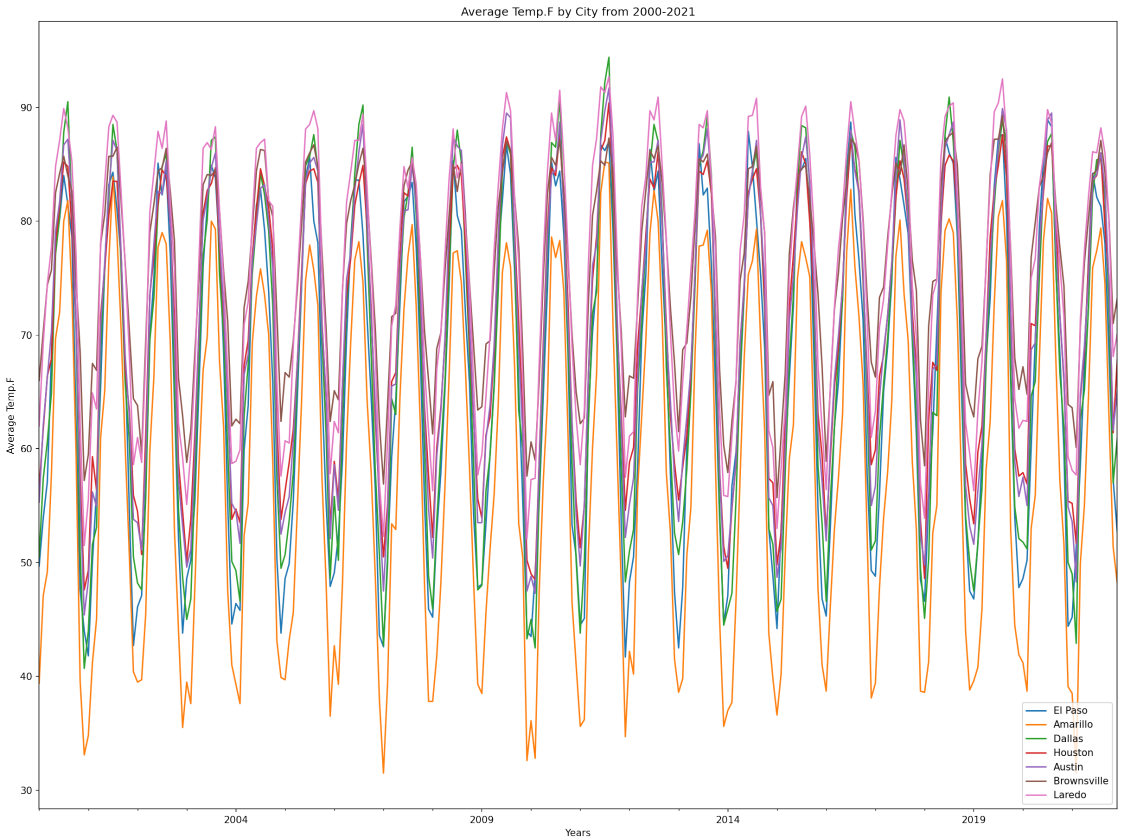
[9]:
```
# Convert Date variable to Datetime to format the data as time series data for␣
 ↪forecasting models
df ['Date'] = pd.to_datetime(df ['Date'], format='%Y-%m', errors='coerce')
```

[10]:
```
# Set Date to index and Date frequency to months
df = df.set_index('Date')
df = df.asfreq('MS')
```

[11]:
```
# create City Level Datasets for city level forecasting
dfEP = df[['ElPasoAvgTemp', 'ElPasoPrecipSum', 'ElPasoSnowSum']]
dfAM = df[['AmarilloAvgTemp', 'AmarilloPrecipSum', 'AmarilloSnowSum']]
dfD = df[['DallasAvgTemp', 'DallasPrecipSum', 'DallasSnowSum']]
dfH = df[['HoustonAvgTemp', 'HoustonPrecipSum', 'HoustonSnowSum']]
dfAU = df[['AustinAvgTemp', 'AustinPrecipSum', 'AustinSnowSum']]
dfB = df[['BrownsvilleAvgTemp', 'BrownsvillePrecipSum', 'BrownsvilleSnowSum']]
dfL = df[['LaredoAvgTemp', 'LaredoPrecipSum', 'LaredoSnowSum']]
```

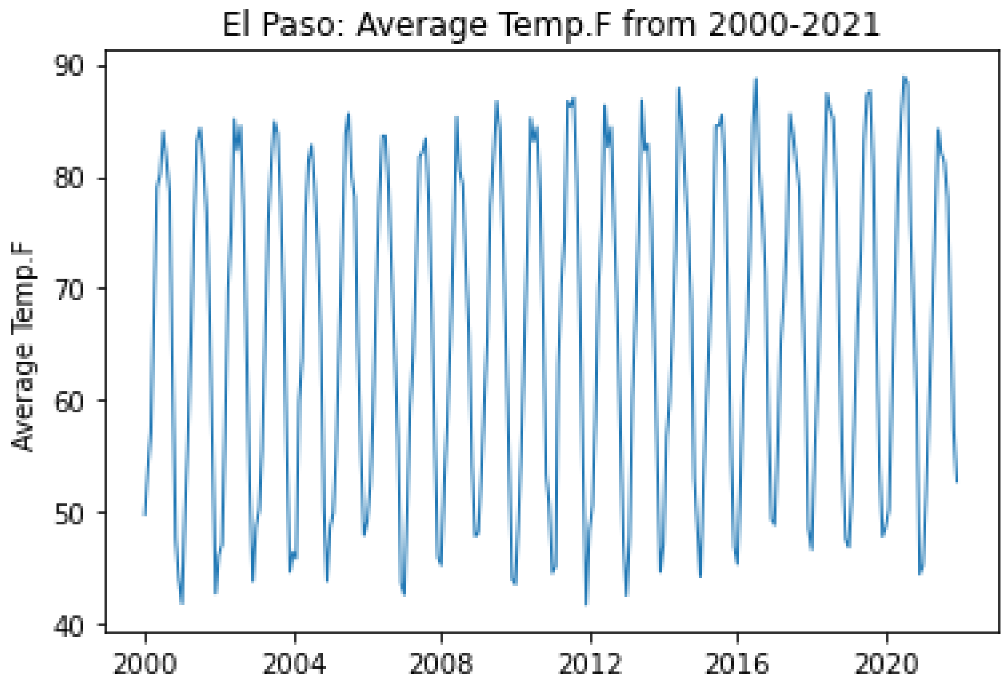
[12]:
```
# Explore what the data looks like for all seven cities
plt.figure(figsize=(20, 15), dpi=150)
df['ElPasoAvgTemp'].plot(label='El Paso')
df['AmarilloAvgTemp'].plot(label='Amarillo')
df['DallasAvgTemp'].plot(label='Dallas')
df['HoustonAvgTemp'].plot(label='Houston')
df['AustinAvgTemp'].plot(label='Austin')
df['BrownsvilleAvgTemp'].plot(label='Brownsville')
df['LaredoAvgTemp'].plot(label='Laredo')
plt.title('Average Temp.F by City from 2000-2021')
plt.xlabel('Years')
plt.ylabel('Average Temp.F')
plt.legend()
```

[12]:
```
<matplotlib.legend.Legend at 0x1d1545ac670>
```



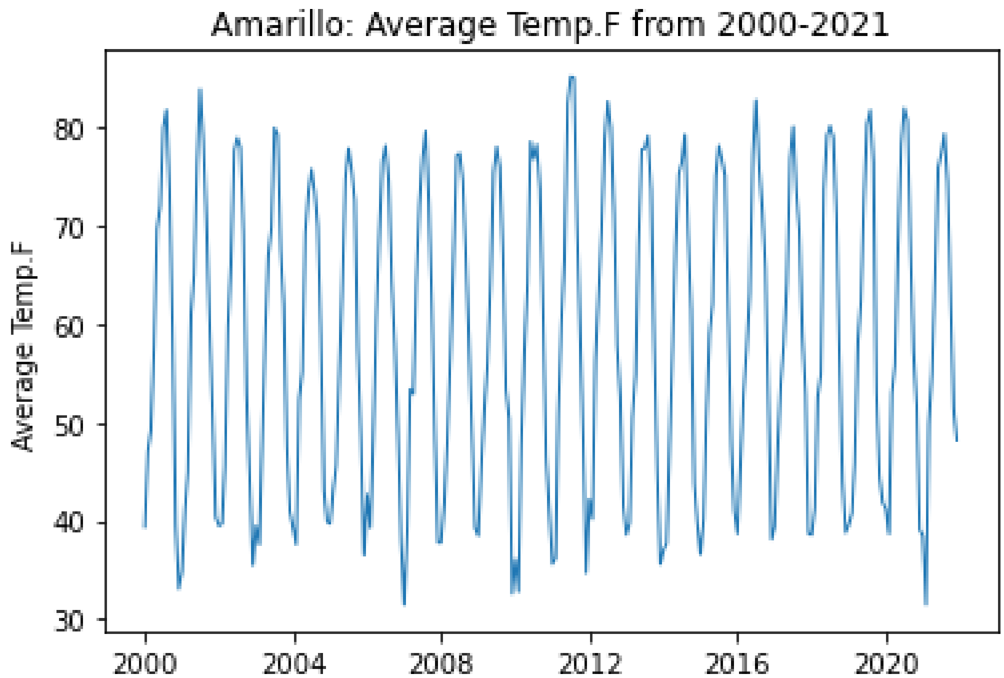
[13]:
```
# Explore what the data looks like for each city
fig, ax = plt.subplots()
ax.plot(df['ElPasoAvgTemp'], linewidth=1)
ax.set_ylabel('Average Temp.F')
ax.set_title('El Paso: Average Temp.F from 2000-2021')
```

[13]:
```
Text(0.5, 1.0, 'El Paso: Average Temp.F from 2000-2021')
```



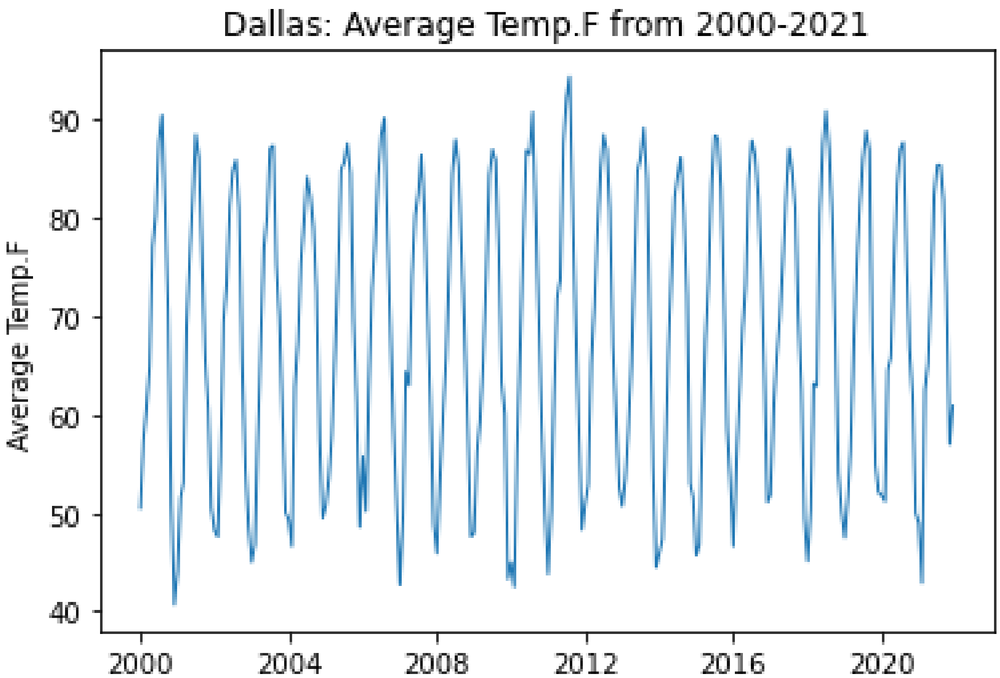
[14]:
```
# Explore what the data looks like for each city
fig, ax = plt.subplots()
ax.plot(df['AmarilloAvgTemp'], linewidth=1)
ax.set_ylabel('Average Temp.F')
ax.set_title('Amarillo: Average Temp.F from 2000-2021')
```

[14]: Text(0.5, 1.0, 'Amarillo: Average Temp.F from 2000-2021')



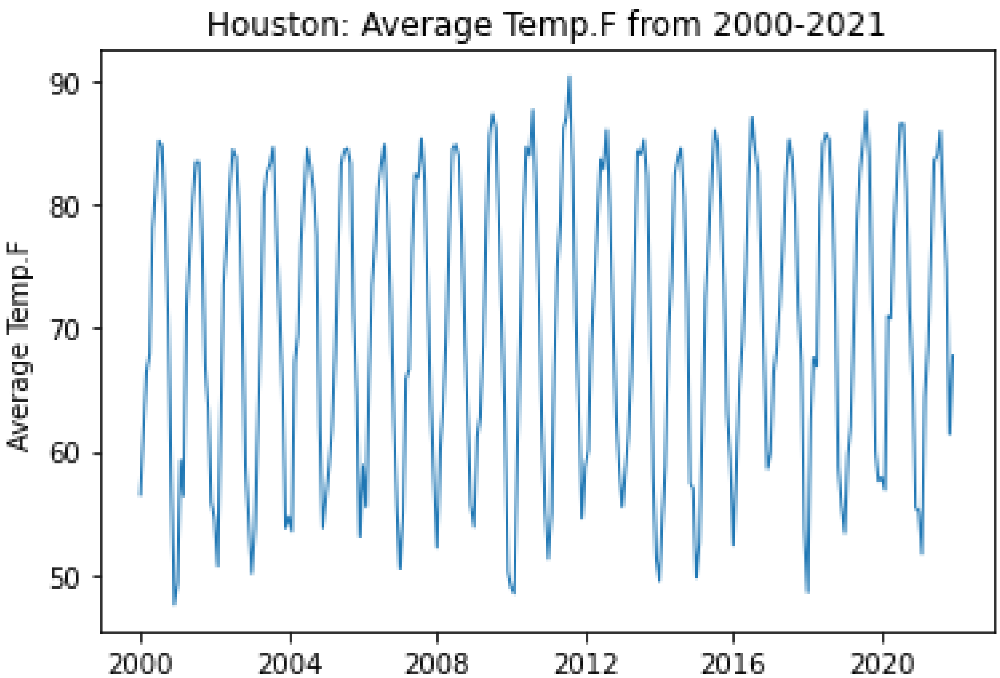

```
[15]: # Explore what the data looks like for each city
      fig, ax = plt.subplots()
      ax.plot(df['DallasAvgTemp'], linewidth=1)
      ax.set_ylabel('Average Temp.F')
      ax.set_title('Dallas: Average Temp.F from 2000-2021')
```

```
[15]: Text(0.5, 1.0, 'Dallas: Average Temp.F from 2000-2021')
```



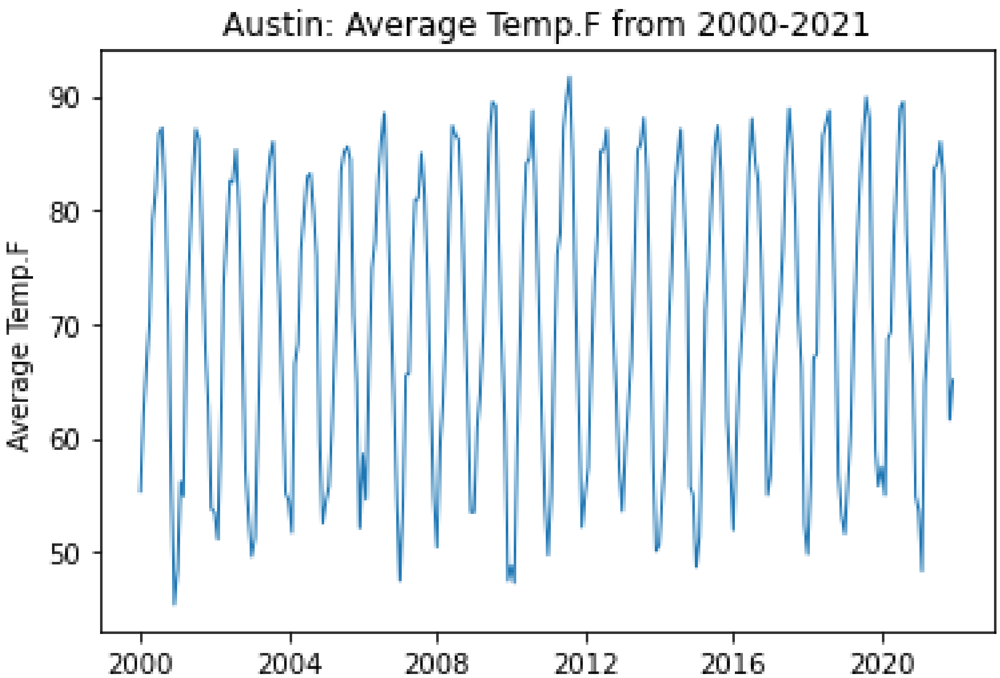
[16]:
```python
# Explore what the data looks like for each city
fig, ax = plt.subplots()
ax.plot(df['HoustonAvgTemp'], linewidth=1)
ax.set_ylabel('Average Temp.F')
ax.set_title('Houston: Average Temp.F from 2000-2021')
```

[16]: Text(0.5, 1.0, 'Houston: Average Temp.F from 2000-2021')



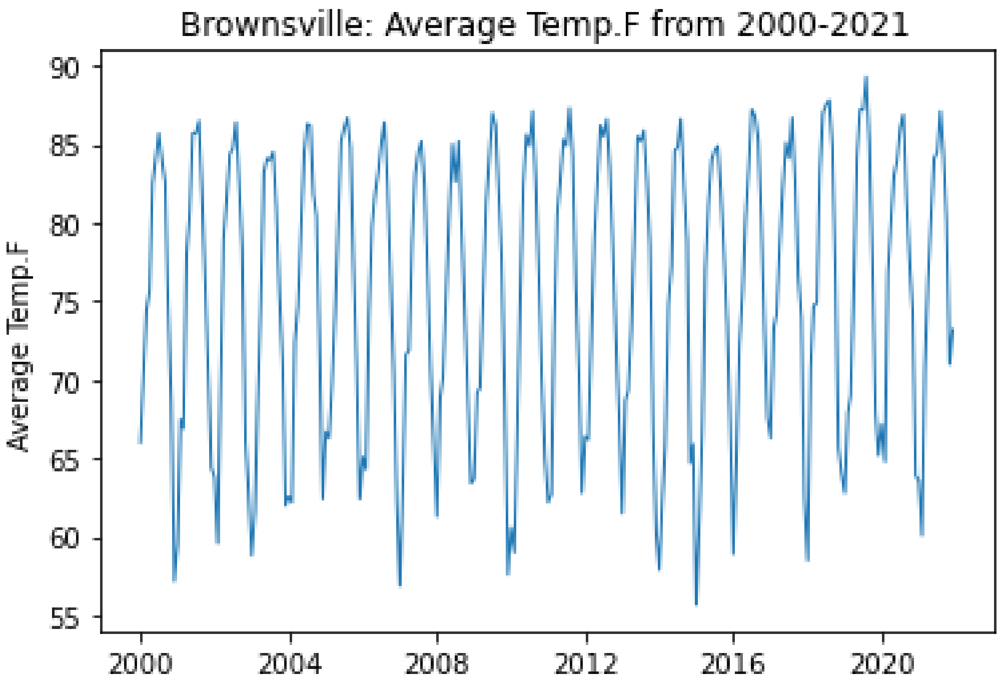
[17]:
```
# Explore what the data looks like for each city
fig, ax = plt.subplots()
ax.plot(df['AustinAvgTemp'], linewidth=1)
ax.set_ylabel('Average Temp.F')
ax.set_title('Austin: Average Temp.F from 2000-2021')
```

[17]:
```
Text(0.5, 1.0, 'Austin: Average Temp.F from 2000-2021')
```



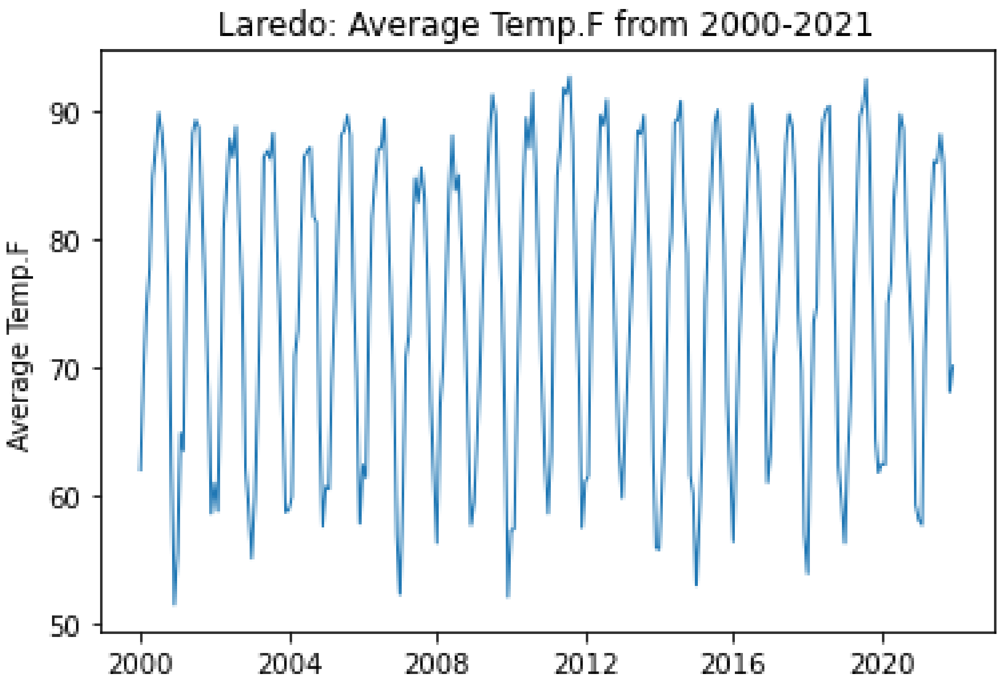
[18]:
```python
# Explore what the data looks like for each city
fig, ax = plt.subplots()
ax.plot(df['BrownsvilleAvgTemp'], linewidth=1)
ax.set_ylabel('Average Temp.F')
ax.set_title('Brownsville: Average Temp.F from 2000-2021')
```

[18]: Text(0.5, 1.0, 'Brownsville: Average Temp.F from 2000-2021')




```
[19]: # Explore what the data looks like for each city
      fig, ax = plt.subplots()
      ax.plot(df['LaredoAvgTemp'], linewidth=1)
      ax.set_ylabel('Average Temp.F')
      ax.set_title('Laredo: Average Temp.F from 2000-2021')
```

```
[19]: Text(0.5, 1.0, 'Laredo: Average Temp.F from 2000-2021')
```



[20]:
```
# El Paso City Weather Forecasting
```

[21]:
```
# create test and train datasets (using the last 48 months as the test dataset)
Months = 48
TrainEP = dfEP[:-Months]
TestEP  = dfEP[-Months:]
```

[22]:
```
# Setup the forecasting model
ForecastModelEP = ForecasterAutoreg(regressor =␣
 ↪RandomForestRegressor(random_state=1), lags = 12)
ForecastModelEP.fit(y = TrainEP['ElPasoAvgTemp'], exog =␣
 ↪TrainEP[['ElPasoPrecipSum', 'ElPasoSnowSum']])
ForecastModelEP
```

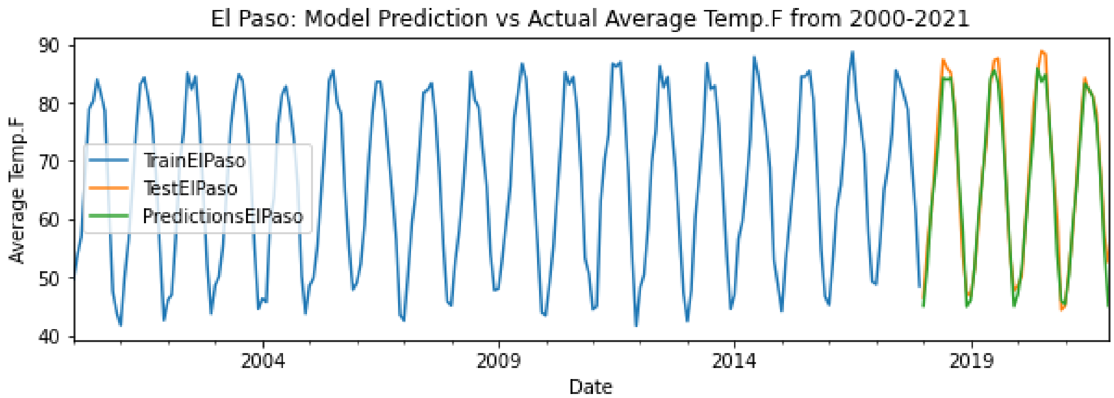
[22]:
```
=================
ForecasterAutoreg
=================
Regressor: RandomForestRegressor(random_state=1)
Lags: [ 1  2  3  4  5  6  7  8  9 10 11 12]
Window size: 12
Included exogenous: True
Type of exogenous variable: <class 'pandas.core.frame.DataFrame'>
Exogenous variables names: ['ElPasoPrecipSum', 'ElPasoSnowSum']
Training range: [Timestamp('2000-01-01 00:00:00'), Timestamp('2017-12-01
```

```
00:00:00')]
Training index type: DatetimeIndex
Training index frequency: MS
Regressor parameters: {'bootstrap': True, 'ccp_alpha': 0.0, 'criterion':
'squared_error', 'max_depth': None, 'max_features': 1.0, 'max_leaf_nodes': None,
'max_samples': None, 'min_impurity_decrease': 0.0, 'min_samples_leaf': 1,
'min_samples_split': 2, 'min_weight_fraction_leaf': 0.0, 'n_estimators': 100,
'n_jobs': None, 'oob_score': False, 'random_state': 1, 'verbose': 0,
'warm_start': False}
Creation date: 2022-08-27 12:02:19
Last fit date: 2022-08-27 12:02:19
Skforecast version: 0.4.3
```

[23]:
```
# use the trained model to predict the next 48 months (same amount in test␣
↪dataset to allow us to test the forecasts accuracy)
Months = 48
PredictionsEP = ForecastModelEP.predict(steps=Months, exog =␣
↪TestEP[['ElPasoPrecipSum', 'ElPasoSnowSum']])
PredictionsEP.head(10)
```

[23]:
```
2018-01-01                  45.14
2018-02-01                  51.86
2018-03-01                  62.09
2018-04-01                  67.69
2018-05-01                  74.61
2018-06-01                  84.21
2018-07-01                  83.96
2018-08-01                  84.24
2018-09-01                  78.10
2018-10-01                  66.96
Freq: MS, Name: pred, dtype: float64
```

[24]:
```
# Plot the training data, test data, and the predictions to visually see the␣
↪prediction accuracy
fig, ax = plt.subplots(figsize=(10, 3))
TrainEP['ElPasoAvgTemp'].plot(ax=ax, label='TrainElPaso')
TestEP['ElPasoAvgTemp'].plot(ax=ax, label='TestElPaso')
PredictionsEP.plot(ax=ax, label='PredictionsElPaso')
ax.set_ylabel('Average Temp.F')
ax.set_title('El Paso: Model Prediction vs Actual Average Temp.F from␣
↪2000-2021')
ax.legend();
```



[25]:
```
# Calculate the MSE (mean squared error)
MseEP = mean_squared_error(y_true = TestEP['ElPasoAvgTemp'], y_pred =␣
 ↪PredictionsEP)
print(f"El Paso MSE Value: {MseEP}")
```

```
El Paso MSE Value: 6.927929874999948
```

[26]:
```
# Calculate the RMSE (root mean squared error)
RmseEP = np.sqrt(MseEP)
print(f"El Paso RMSE Value: {RmseEP}")
```

```
El Paso RMSE Value: 2.6320960991194733
```

[27]:
```
# Setup the forecasting model to forecast future
ForecastModelEPF = ForecasterAutoreg(regressor =␣
 ↪RandomForestRegressor(random_state=1), lags = 12)
ForecastModelEPF.fit(y = dfEP['ElPasoAvgTemp'], exog = dfEP[['ElPasoPrecipSum',␣
 ↪'ElPasoSnowSum']])
ForecastModelEPF
```

[27]:
```
=================
ForecasterAutoreg
=================
Regressor: RandomForestRegressor(random_state=1)
Lags: [ 1  2  3  4  5  6  7  8  9 10 11 12]
Window size: 12
Included exogenous: True
Type of exogenous variable: <class 'pandas.core.frame.DataFrame'>
Exogenous variables names: ['ElPasoPrecipSum', 'ElPasoSnowSum']
Training range: [Timestamp('2000-01-01 00:00:00'), Timestamp('2021-12-01
00:00:00')]
Training index type: DatetimeIndex
Training index frequency: MS
```

```
Regressor parameters: {'bootstrap': True, 'ccp_alpha': 0.0, 'criterion':
'squared_error', 'max_depth': None, 'max_features': 1.0, 'max_leaf_nodes': None,
'max_samples': None, 'min_impurity_decrease': 0.0, 'min_samples_leaf': 1,
'min_samples_split': 2, 'min_weight_fraction_leaf': 0.0, 'n_estimators': 100,
'n_jobs': None, 'oob_score': False, 'random_state': 1, 'verbose': 0,
'warm_start': False}
Creation date: 2022-08-27 12:02:20
Last fit date: 2022-08-27 12:02:20
Skforecast version: 0.4.3
```

[28]:
```
# use the trained model to predict the next 60 months (same amount in test␣
↪dataset to allow us to test the forecasts accuracy)
MonthsF = 60
PredictionsEPF = ForecastModelEPF.predict(steps=MonthsF, exog =␣
↪dfEP[['ElPasoPrecipSum', 'ElPasoSnowSum']])
PredictionsEPF
```

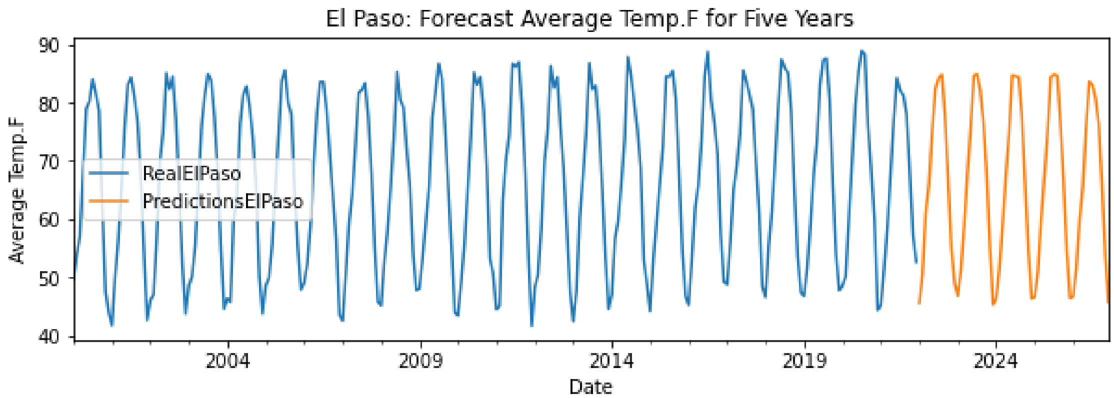
[28]:
```
2022-01-01                   45.63
2022-02-01                   50.64
2022-03-01                   61.17
2022-04-01                   66.09
2022-05-01                   75.13
2022-06-01                   82.41
2022-07-01                   84.19
2022-08-01                   84.81
2022-09-01                   77.55
2022-10-01                   66.54
2022-11-01                   55.15
2022-12-01                   49.03
2023-01-01                   46.85
2023-02-01                   51.42
2023-03-01                   58.90
2023-04-01                   67.99
2023-05-01                   75.91
2023-06-01                   84.64
2023-07-01                   84.92
2023-08-01                   82.05
2023-09-01                   76.83
2023-10-01                   68.07
2023-11-01                   55.39
2023-12-01                   45.36
2024-01-01                   46.34
2024-02-01                   51.52
2024-03-01                   59.25
2024-04-01                   67.62
2024-05-01                   74.72
2024-06-01                   84.72
```

```
2024-07-01                84.55
2024-08-01                84.37
2024-09-01                77.62
2024-10-01                66.23
2024-11-01                54.77
2024-12-01                46.41
2025-01-01                46.64
2025-02-01                50.98
2025-03-01                59.87
2025-04-01                67.33
2025-05-01                74.95
2025-06-01                84.37
2025-07-01                84.86
2025-08-01                84.65
2025-09-01                76.67
2025-10-01                66.66
2025-11-01                54.63
2025-12-01                46.47
2026-01-01                46.77
2026-02-01                51.64
2026-03-01                59.80
2026-04-01                65.64
2026-05-01                75.71
2026-06-01                83.66
2026-07-01                83.06
2026-08-01                80.69
2026-09-01                76.33
2026-10-01                66.14
2026-11-01                53.97
2026-12-01                45.78
Freq: MS, Name: pred, dtype: float64
```

[29]:

```
# Plot the training data and the predictions to visually see the predictions
fig, ax = plt.subplots(figsize=(10, 3))
dfEP['ElPasoAvgTemp'].plot(ax=ax, label='RealElPaso')
PredictionsEPF.plot(ax=ax, label='PredictionsElPaso')
ax.set_ylabel('Average Temp.F')
ax.set_title('El Paso: Forecast Average Temp.F for Five Years')

ax.legend();
```



[30]:
```
# Amarillo City Weather Forecasting
```

[31]:
```
# create test and train datasets (using the last 48 months as the test dataset)
Months = 48
TrainAM = dfAM[:-Months]
TestAM  = dfAM[-Months:]
```

[32]:
```
# Setup the forecasting model
ForecastModelAM = ForecasterAutoreg(regressor = RandomForestRegressor(random_state=1), lags = 12)
ForecastModelAM.fit(y = TrainAM['AmarilloAvgTemp'], exog = TrainAM[['AmarilloPrecipSum', 'AmarilloSnowSum']])
ForecastModelAM
```

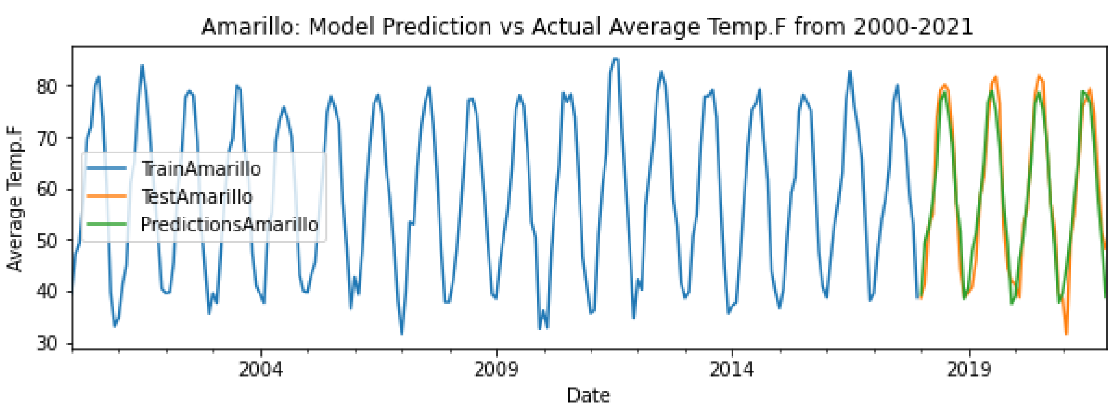
[32]:
```
=================
ForecasterAutoreg
=================
Regressor: RandomForestRegressor(random_state=1)
Lags: [ 1  2  3  4  5  6  7  8  9 10 11 12]
Window size: 12
Included exogenous: True
Type of exogenous variable: <class 'pandas.core.frame.DataFrame'>
Exogenous variables names: ['AmarilloPrecipSum', 'AmarilloSnowSum']
Training range: [Timestamp('2000-01-01 00:00:00'), Timestamp('2017-12-01
00:00:00')]
Training index type: DatetimeIndex
Training index frequency: MS
Regressor parameters: {'bootstrap': True, 'ccp_alpha': 0.0, 'criterion':
'squared_error', 'max_depth': None, 'max_features': 1.0, 'max_leaf_nodes': None,
'max_samples': None, 'min_impurity_decrease': 0.0, 'min_samples_leaf': 1,
'min_samples_split': 2, 'min_weight_fraction_leaf': 0.0, 'n_estimators': 100,
'n_jobs': None, 'oob_score': False, 'random_state': 1, 'verbose': 0,
```

```
'warm_start': False}
Creation date: 2022-08-27 12:02:21
Last fit date: 2022-08-27 12:02:21
Skforecast version: 0.4.3
```

[33]:
```
# use the trained model to predict the next 48 months (same amount in test␣
↪dataset to allow us to test the forecasts accuracy)
Months = 48
PredictionsAM = ForecastModelAM.predict(steps=Months, exog =␣
↪TestAM[['AmarilloPrecipSum', 'AmarilloSnowSum']])
PredictionsAM.head(10)
```

[33]:
```
2018-01-01    39.25
2018-02-01    49.41
2018-03-01    52.30
2018-04-01    57.71
2018-05-01    64.18
2018-06-01    77.09
2018-07-01    78.75
2018-08-01    74.39
2018-09-01    68.60
2018-10-01    57.02
Freq: MS, Name: pred, dtype: float64
```

[34]:
```
# Plot the training data, test data, and the predictions to visually see the␣
↪prediction accuracy
fig, ax = plt.subplots(figsize=(10, 3))
TrainAM['AmarilloAvgTemp'].plot(ax=ax, label='TrainAmarillo')
TestAM['AmarilloAvgTemp'].plot(ax=ax, label='TestAmarillo')
PredictionsAM.plot(ax=ax, label='PredictionsAmarillo')
ax.set_ylabel('Average Temp.F')
ax.set_title('Amarillo: Model Prediction vs Actual Average Temp.F from␣
↪2000-2021')
ax.legend();
```



[35]:
```
# Calculate the MSE (mean squared error)
MseAM = mean_squared_error(y_true = TestAM['AmarilloAvgTemp'], y_pred =␣
↪PredictionsAM)
print(f"Amarillo MSE Value: {MseAM}")
```

```
Amarillo MSE Value: 21.018619416666596
```

[36]:
```
# Calculate the RMSE (root mean squared error)
RmseAM = np.sqrt(MseAM)
print(f"Amarillo RMSE Value: {RmseAM}")
```

```
Amarillo RMSE Value: 4.5846067897548854
```

[37]:
```
# Setup the forecasting model to forecast future
ForecastModelAMF = ForecasterAutoreg(regressor =␣
↪RandomForestRegressor(random_state=1), lags = 12)
ForecastModelAMF.fit(y = dfAM['AmarilloAvgTemp'], exog =␣
↪dfAM[['AmarilloPrecipSum', 'AmarilloSnowSum']])
ForecastModelAMF
```

[37]:
```
=================
ForecasterAutoreg
=================
Regressor: RandomForestRegressor(random_state=1)
Lags: [ 1  2  3  4  5  6  7  8  9 10 11 12]
Window size: 12
Included exogenous: True
Type of exogenous variable: <class 'pandas.core.frame.DataFrame'>
Exogenous variables names: ['AmarilloPrecipSum', 'AmarilloSnowSum']
Training range: [Timestamp('2000-01-01 00:00:00'), Timestamp('2021-12-01
00:00:00')]
Training index type: DatetimeIndex
Training index frequency: MS
Regressor parameters: {'bootstrap': True, 'ccp_alpha': 0.0, 'criterion':
'squared_error', 'max_depth': None, 'max_features': 1.0, 'max_leaf_nodes': None,
'max_samples': None, 'min_impurity_decrease': 0.0, 'min_samples_leaf': 1,
'min_samples_split': 2, 'min_weight_fraction_leaf': 0.0, 'n_estimators': 100,
'n_jobs': None, 'oob_score': False, 'random_state': 1, 'verbose': 0,
'warm_start': False}
Creation date: 2022-08-27 12:02:21
Last fit date: 2022-08-27 12:02:22
Skforecast version: 0.4.3
```

[38]:
```
# use the trained model to predict the next 60 months (same amount in test␣
↪dataset to allow us to test the forecasts accuracy)
```

```python
MonthsF = 60
PredictionsAMF = ForecastModelAMF.predict(steps=MonthsF, exog =
    dfAM[['AmarilloPrecipSum', 'AmarilloSnowSum']])
PredictionsAMF
```

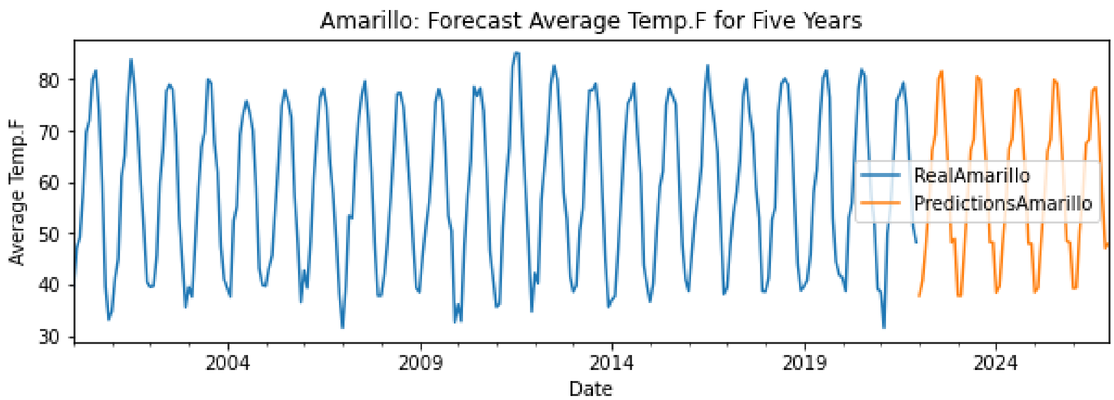
[38]:

```
2022-01-01    37.80
2022-02-01    40.77
2022-03-01    47.20
2022-04-01    55.48
2022-05-01    66.17
2022-06-01    69.31
2022-07-01    80.20
2022-08-01    81.62
2022-09-01    72.78
2022-10-01    62.38
2022-11-01    48.16
2022-12-01    48.93
2023-01-01    37.79
2023-02-01    37.79
2023-03-01    46.41
2023-04-01    56.19
2023-05-01    66.13
2023-06-01    68.39
2023-07-01    80.62
2023-08-01    79.90
2023-09-01    71.13
2023-10-01    62.52
2023-11-01    48.18
2023-12-01    48.20
2024-01-01    38.39
2024-02-01    39.68
2024-03-01    48.03
2024-04-01    55.29
2024-05-01    66.43
2024-06-01    68.46
2024-07-01    77.76
2024-08-01    78.17
2024-09-01    71.63
2024-10-01    62.12
2024-11-01    47.94
2024-12-01    47.97
2025-01-01    38.48
2025-02-01    39.30
2025-03-01    48.21
2025-04-01    56.51
2025-05-01    65.98
2025-06-01    68.35
```

```
2025-07-01                    79.96
2025-08-01                    79.28
2025-09-01                    70.93
2025-10-01                    60.64
2025-11-01                    48.46
2025-12-01                    48.17
2026-01-01                    39.19
2026-02-01                    39.34
2026-03-01                    50.19
2026-04-01                    57.13
2026-05-01                    67.48
2026-06-01                    68.28
2026-07-01                    77.74
2026-08-01                    78.45
2026-09-01                    71.30
2026-10-01                    57.98
2026-11-01                    47.10
2026-12-01                    47.98
Freq: MS, Name: pred, dtype: float64
```

[39]:

```
# Plot the training data and the predictions to visually see the predictions
fig, ax = plt.subplots(figsize=(10, 3))
dfAM['AmarilloAvgTemp'].plot(ax=ax, label='RealAmarillo')
PredictionsAMF.plot(ax=ax, label='PredictionsAmarillo')
ax.set_ylabel('Average Temp.F')
ax.set_title('Amarillo: Forecast Average Temp.F for Five Years')
ax.legend();
```



[40]:

```
# Dallas City Weather Forecasting
```

[41]:

```
# create test and train datasets (using the last 48 months as the test dataset)
Months = 48
```

```
TrainD = dfD[:-Months]
TestD  = dfD[-Months:]
```

[42]:
```
# Setup the forecasting model
ForecastModelD = ForecasterAutoreg(regressor =␣
↪RandomForestRegressor(random_state=1), lags = 12)
ForecastModelD.fit(y = TrainD['DallasAvgTemp'], exog =␣
↪TrainD[['DallasPrecipSum', 'DallasSnowSum']])
ForecastModelD
```

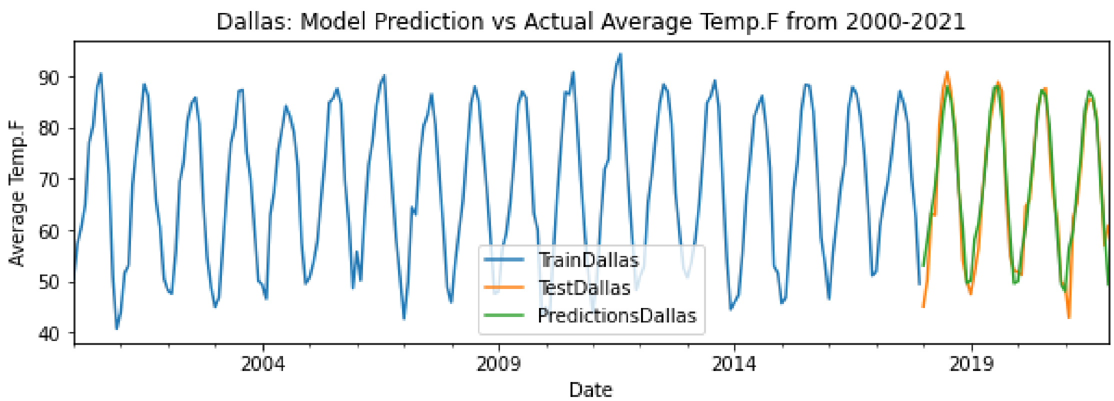
[42]:
```
=================
ForecasterAutoreg
=================
Regressor: RandomForestRegressor(random_state=1)
Lags: [ 1  2  3  4  5  6  7  8  9 10 11 12]
Window size: 12
Included exogenous: True
Type of exogenous variable: <class 'pandas.core.frame.DataFrame'>
Exogenous variables names: ['DallasPrecipSum', 'DallasSnowSum']
Training range: [Timestamp('2000-01-01 00:00:00'), Timestamp('2017-12-01
00:00:00')]
Training index type: DatetimeIndex
Training index frequency: MS
Regressor parameters: {'bootstrap': True, 'ccp_alpha': 0.0, 'criterion':
'squared_error', 'max_depth': None, 'max_features': 1.0, 'max_leaf_nodes': None,
'max_samples': None, 'min_impurity_decrease': 0.0, 'min_samples_leaf': 1,
'min_samples_split': 2, 'min_weight_fraction_leaf': 0.0, 'n_estimators': 100,
'n_jobs': None, 'oob_score': False, 'random_state': 1, 'verbose': 0,
'warm_start': False}
Creation date: 2022-08-27 12:02:22
Last fit date: 2022-08-27 12:02:22
Skforecast version: 0.4.3
```

[43]:
```
# use the trained model to predict the next 48 months (same amount in test␣
↪dataset to allow us to test the forecasts accuracy)
Months = 48
PredictionsD = ForecastModelD.predict(steps=Months, exog =␣
↪TestD[['DallasPrecipSum', 'DallasSnowSum']])
PredictionsD.head(10)
```

[43]:
```
2018-01-01    53.11
2018-02-01    58.07
2018-03-01    63.63
2018-04-01    68.63
2018-05-01    75.61
2018-06-01    83.30
2018-07-01    88.05
```

```
2018-08-01                    85.41
2018-09-01                    79.39
2018-10-01                    67.53
Freq: MS, Name: pred, dtype: float64
```

[44]:
```python
# Plot the training data, test data, and the predictions to visually see the
 ↪prediction accuracy
fig, ax = plt.subplots(figsize=(10, 3))
TrainD['DallasAvgTemp'].plot(ax=ax, label='TrainDallas')
TestD['DallasAvgTemp'].plot(ax=ax, label='TestDallas')
PredictionsD.plot(ax=ax, label='PredictionsDallas')
ax.set_ylabel('Average Temp.F')
ax.set_title('Dallas: Model Prediction vs Actual Average Temp.F from 2000-2021')
ax.legend();
```



[45]:
```python
# Calculate the MSE (mean squared error)
MseD = mean_squared_error(y_true = TestD['DallasAvgTemp'], y_pred =
 ↪PredictionsD)
print(f"Dallas MSE Value: {MseD}")
```

```
Dallas MSE Value: 17.26436185416662
```

[46]:
```python
# Calculate the RMSE (root mean squared error)
RmseD = np.sqrt(MseD)
print(f"Dallas RMSE Value: {RmseD}")
```

```
Dallas RMSE Value: 4.155040535803065
```

[47]:
```python
# Setup the forecasting model to forecast future
ForecastModelDF = ForecasterAutoreg(regressor =
 ↪RandomForestRegressor(random_state=1), lags = 12)
ForecastModelDF.fit(y = dfD['DallasAvgTemp'], exog = dfD[['DallasPrecipSum',
 ↪'DallasSnowSum']])
```

```
ForecastModelDF
```

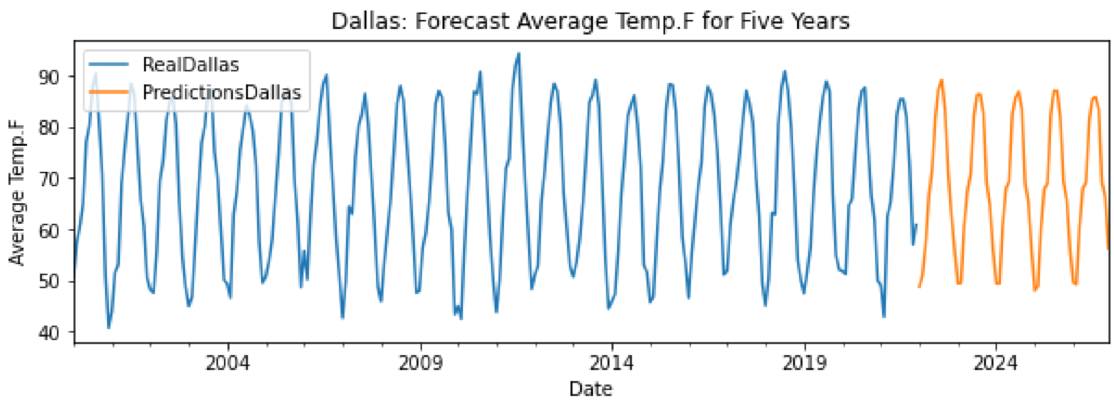
[47]:
```
=================
ForecasterAutoreg
=================
Regressor: RandomForestRegressor(random_state=1)
Lags: [ 1  2  3  4  5  6  7  8  9 10 11 12]
Window size: 12
Included exogenous: True
Type of exogenous variable: <class 'pandas.core.frame.DataFrame'>
Exogenous variables names: ['DallasPrecipSum', 'DallasSnowSum']
Training range: [Timestamp('2000-01-01 00:00:00'), Timestamp('2021-12-01
00:00:00')]
Training index type: DatetimeIndex
Training index frequency: MS
Regressor parameters: {'bootstrap': True, 'ccp_alpha': 0.0, 'criterion':
'squared_error', 'max_depth': None, 'max_features': 1.0, 'max_leaf_nodes': None,
'max_samples': None, 'min_impurity_decrease': 0.0, 'min_samples_leaf': 1,
'min_samples_split': 2, 'min_weight_fraction_leaf': 0.0, 'n_estimators': 100,
'n_jobs': None, 'oob_score': False, 'random_state': 1, 'verbose': 0,
'warm_start': False}
Creation date: 2022-08-27 12:02:23
Last fit date: 2022-08-27 12:02:23
Skforecast version: 0.4.3
```

[48]:
```
# use the trained model to predict the next 60 months (same amount in test␣
↪dataset to allow us to test the forecasts accuracy)
MonthsF = 60
PredictionsDF = ForecastModelDF.predict(steps=MonthsF, exog =␣
↪dfD[['DallasPrecipSum', 'DallasSnowSum']])
PredictionsDF
```

[48]:
```
2022-01-01    48.72
2022-02-01    51.09
2022-03-01    57.09
2022-04-01    66.56
2022-05-01    71.78
2022-06-01    81.82
2022-07-01    87.51
2022-08-01    89.18
2022-09-01    83.31
2022-10-01    71.10
2022-11-01    63.10
2022-12-01    56.00
2023-01-01    49.40
2023-02-01    49.50
2023-03-01    60.78
```

```
2023-04-01                 67.55
2023-05-01                 70.84
2023-06-01                 81.99
2023-07-01                 86.32
2023-08-01                 86.42
2023-09-01                 82.83
2023-10-01                 69.09
2023-11-01                 64.73
2023-12-01                 55.88
2024-01-01                 49.41
2024-02-01                 49.39
2024-03-01                 60.61
2024-04-01                 67.91
2024-05-01                 69.31
2024-06-01                 82.45
2024-07-01                 85.83
2024-08-01                 86.85
2024-09-01                 83.53
2024-10-01                 69.07
2024-11-01                 65.40
2024-12-01                 56.09
2025-01-01                 48.01
2025-02-01                 48.87
2025-03-01                 60.40
2025-04-01                 67.84
2025-05-01                 69.07
2025-06-01                 82.35
2025-07-01                 87.05
2025-08-01                 87.02
2025-09-01                 81.71
2025-10-01                 69.31
2025-11-01                 65.82
2025-12-01                 56.18
2026-01-01                 49.58
2026-02-01                 49.24
2026-03-01                 60.39
2026-04-01                 67.93
2026-05-01                 69.20
2026-06-01                 81.24
2026-07-01                 85.45
2026-08-01                 85.89
2026-09-01                 83.44
2026-10-01                 69.13
2026-11-01                 66.41
2026-12-01                 56.26
Freq: MS, Name: pred, dtype: float64
```

[49]:
```
# Plot the training data and the predictions to visually see the predictions
fig, ax = plt.subplots(figsize=(10, 3))
dfD['DallasAvgTemp'].plot(ax=ax, label='RealDallas')
PredictionsDF.plot(ax=ax, label='PredictionsDallas')
ax.set_ylabel('Average Temp.F')
ax.set_title('Dallas: Forecast Average Temp.F for Five Years')
ax.legend();
```



[50]:
```
# Houston City Weather Forecasting
```

[51]:
```
# create test and train datasets (using the last 48 months as the test dataset)
Months = 48
TrainH = dfH[:-Months]
TestH  = dfH[-Months:]
```

[52]:
```
# Setup the forecasting model
ForecastModelH = ForecasterAutoreg(regressor =␣
 ↪RandomForestRegressor(random_state=1), lags = 12)
ForecastModelH.fit(y = TrainH['HoustonAvgTemp'], exog =␣
 ↪TrainH[['HoustonPrecipSum', 'HoustonSnowSum']])
ForecastModelH
```

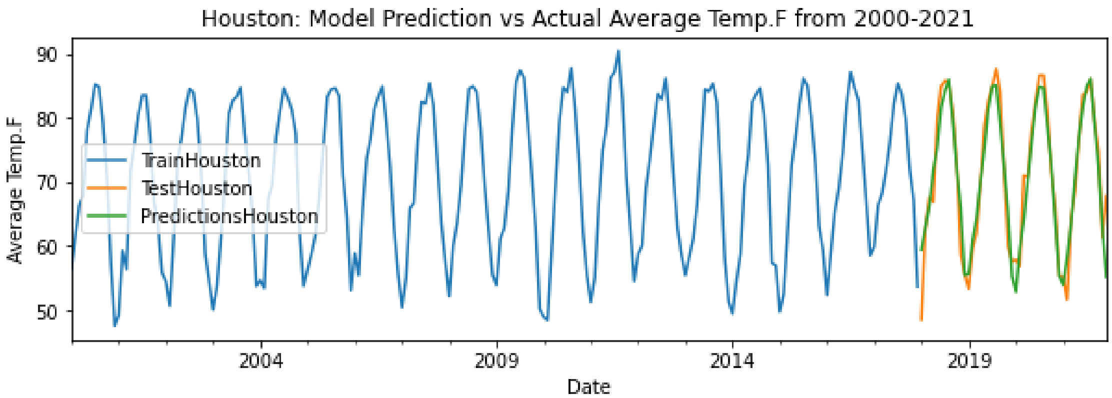
[52]:
```
=================
ForecasterAutoreg
=================
Regressor: RandomForestRegressor(random_state=1)
Lags: [ 1  2  3  4  5  6  7  8  9 10 11 12]
Window size: 12
Included exogenous: True
Type of exogenous variable: <class 'pandas.core.frame.DataFrame'>
Exogenous variables names: ['HoustonPrecipSum', 'HoustonSnowSum']
Training range: [Timestamp('2000-01-01 00:00:00'), Timestamp('2017-12-01
```

```
00:00:00')]
Training index type: DatetimeIndex
Training index frequency: MS
Regressor parameters: {'bootstrap': True, 'ccp_alpha': 0.0, 'criterion':
'squared_error', 'max_depth': None, 'max_features': 1.0, 'max_leaf_nodes': None,
'max_samples': None, 'min_impurity_decrease': 0.0, 'min_samples_leaf': 1,
'min_samples_split': 2, 'min_weight_fraction_leaf': 0.0, 'n_estimators': 100,
'n_jobs': None, 'oob_score': False, 'random_state': 1, 'verbose': 0,
'warm_start': False}
Creation date: 2022-08-27 12:02:24
Last fit date: 2022-08-27 12:02:24
Skforecast version: 0.4.3
```

[53]:
```
# use the trained model to predict the next 48 months (same amount in test␣
↪dataset to allow us to test the forecasts accuracy)
Months = 48
PredictionsH = ForecastModelH.predict(steps=Months, exog =␣
↪TestH[['HoustonPrecipSum', 'HoustonSnowSum']])
PredictionsH.head(10)
```

[53]:
```
2018-01-01                  59.47
2018-02-01                  62.87
2018-03-01                  66.27
2018-04-01                  71.77
2018-05-01                  75.17
2018-06-01                  81.64
2018-07-01                  84.22
2018-08-01                  85.95
2018-09-01                  79.43
2018-10-01                  71.70
Freq: MS, Name: pred, dtype: float64
```

[54]:
```
# Plot the training data, test data, and the predictions to visually see the␣
↪prediction accuracy
fig, ax = plt.subplots(figsize=(10, 3))
TrainH['HoustonAvgTemp'].plot(ax=ax, label='TrainHouston')
TestH['HoustonAvgTemp'].plot(ax=ax, label='TestHouston')
PredictionsH.plot(ax=ax, label='PredictionsHouston')
ax.set_ylabel('Average Temp.F')
ax.set_title('Houston: Model Prediction vs Actual Average Temp.F from␣
↪2000-2021')
ax.legend();
```



[55]:
```
# Calculate the MSE (mean squared error)
MseH = mean_squared_error(y_true = TestH['HoustonAvgTemp'], y_pred =␣
 ↪PredictionsH)
print(f"Houston MSE Value: {MseH}")
```

```
Houston MSE Value: 13.476819666666628
```

[56]:
```
# Calculate the RMSE (root mean squared error)
RmseH = np.sqrt(MseH)
print(f"Houston RMSE Value: {RmseH}")
```

```
Houston RMSE Value: 3.671078815098721
```

[57]:
```
# Setup the forecasting model to forecast future
ForecastModelHF = ForecasterAutoreg(regressor =␣
 ↪RandomForestRegressor(random_state=1), lags = 12)
ForecastModelHF.fit(y = dfH['HoustonAvgTemp'], exog = dfH[['HoustonPrecipSum',␣
 ↪'HoustonSnowSum']])
ForecastModelHF
```

[57]:
```
=================
ForecasterAutoreg
=================
Regressor: RandomForestRegressor(random_state=1)
Lags: [ 1  2  3  4  5  6  7  8  9 10 11 12]
Window size: 12
Included exogenous: True
Type of exogenous variable: <class 'pandas.core.frame.DataFrame'>
Exogenous variables names: ['HoustonPrecipSum', 'HoustonSnowSum']
Training range: [Timestamp('2000-01-01 00:00:00'), Timestamp('2021-12-01
00:00:00')]
Training index type: DatetimeIndex
Training index frequency: MS
```

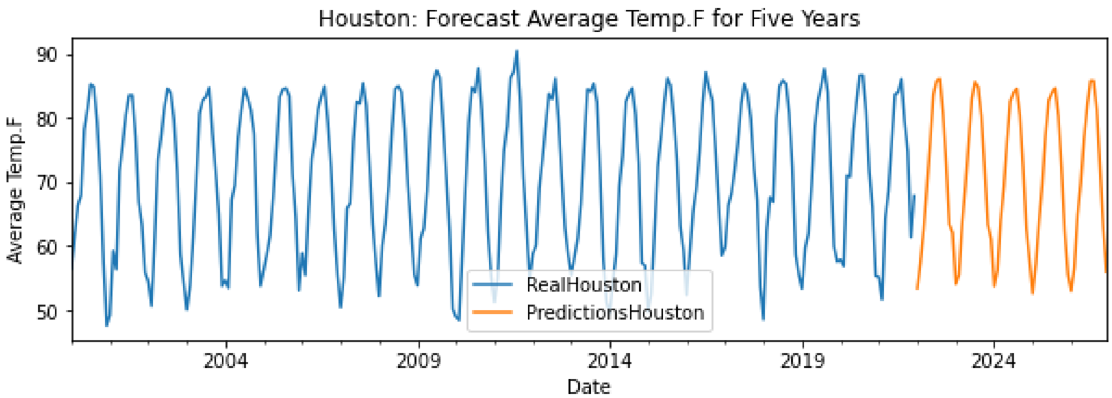
```
Regressor parameters: {'bootstrap': True, 'ccp_alpha': 0.0, 'criterion':
'squared_error', 'max_depth': None, 'max_features': 1.0, 'max_leaf_nodes': None,
'max_samples': None, 'min_impurity_decrease': 0.0, 'min_samples_leaf': 1,
'min_samples_split': 2, 'min_weight_fraction_leaf': 0.0, 'n_estimators': 100,
'n_jobs': None, 'oob_score': False, 'random_state': 1, 'verbose': 0,
'warm_start': False}
Creation date: 2022-08-27 12:02:25
Last fit date: 2022-08-27 12:02:25
Skforecast version: 0.4.3
```

[58]:
```
# use the trained model to predict the next 60 months (same amount in test␣
↪dataset to allow us to test the forecasts accuracy)
MonthsF = 60
PredictionsHF = ForecastModelHF.predict(steps=MonthsF, exog =␣
↪dfH[['HoustonPrecipSum', 'HoustonSnowSum']])
PredictionsHF
```

[58]:
```
2022-01-01                  53.45
2022-02-01                  57.61
2022-03-01                  62.60
2022-04-01                  69.25
2022-05-01                  76.12
2022-06-01                  83.60
2022-07-01                  85.74
2022-08-01                  86.04
2022-09-01                  80.69
2022-10-01                  72.72
2022-11-01                  63.43
2022-12-01                  62.06
2023-01-01                  54.10
2023-02-01                  55.55
2023-03-01                  63.45
2023-04-01                  69.33
2023-05-01                  76.30
2023-06-01                  82.95
2023-07-01                  85.59
2023-08-01                  84.74
2023-09-01                  80.31
2023-10-01                  72.21
2023-11-01                  63.38
2023-12-01                  61.11
2024-01-01                  53.75
2024-02-01                  56.23
2024-03-01                  63.99
2024-04-01                  69.26
2024-05-01                  76.41
2024-06-01                  82.55
```

```
2024-07-01               83.90
2024-08-01               84.50
2024-09-01               80.40
2024-10-01               72.43
2024-11-01               62.80
2024-12-01               57.70
2025-01-01               52.66
2025-02-01               57.19
2025-03-01               63.83
2025-04-01               69.68
2025-05-01               77.37
2025-06-01               82.70
2025-07-01               83.89
2025-08-01               84.59
2025-09-01               80.23
2025-10-01               72.20
2025-11-01               62.33
2025-12-01               55.66
2026-01-01               53.02
2026-02-01               56.68
2026-03-01               64.81
2026-04-01               69.85
2026-05-01               76.77
2026-06-01               81.88
2026-07-01               85.73
2026-08-01               85.74
2026-09-01               81.30
2026-10-01               72.43
2026-11-01               62.97
2026-12-01               56.01
Freq: MS, Name: pred, dtype: float64
```

[59]:
```
# Plot the training data and the predictions to visually see the predictions
fig, ax = plt.subplots(figsize=(10, 3))
dfH['HoustonAvgTemp'].plot(ax=ax, label='RealHouston')
PredictionsHF.plot(ax=ax, label='PredictionsHouston')
ax.set_ylabel('Average Temp.F')
ax.set_title('Houston: Forecast Average Temp.F for Five Years')
ax.legend();
```



[60]:
```
# Austin City Weather Forecasting
```

[61]:
```
# create test and train datasets (using the last 48 months as the test dataset)
Months = 48
TrainAU = dfAU[:-Months]
TestAU  = dfAU[-Months:]
```

[62]:
```
# Setup the forecasting model
ForecastModelAU = ForecasterAutoreg(regressor =␣
 ↪RandomForestRegressor(random_state=1), lags = 12)
ForecastModelAU.fit(y = TrainAU['AustinAvgTemp'], exog =␣
 ↪TrainAU[['AustinPrecipSum', 'AustinSnowSum']])
ForecastModelAU
```

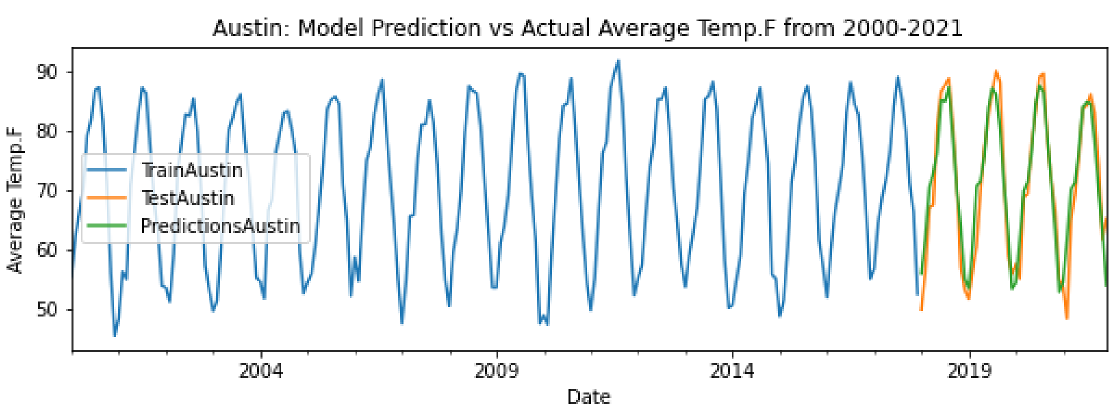
[62]:
```
=================
ForecasterAutoreg
=================
Regressor: RandomForestRegressor(random_state=1)
Lags: [ 1  2  3  4  5  6  7  8  9 10 11 12]
Window size: 12
Included exogenous: True
Type of exogenous variable: <class 'pandas.core.frame.DataFrame'>
Exogenous variables names: ['AustinPrecipSum', 'AustinSnowSum']
Training range: [Timestamp('2000-01-01 00:00:00'), Timestamp('2017-12-01
00:00:00')]
Training index type: DatetimeIndex
Training index frequency: MS
Regressor parameters: {'bootstrap': True, 'ccp_alpha': 0.0, 'criterion':
'squared_error', 'max_depth': None, 'max_features': 1.0, 'max_leaf_nodes': None,
'max_samples': None, 'min_impurity_decrease': 0.0, 'min_samples_leaf': 1,
'min_samples_split': 2, 'min_weight_fraction_leaf': 0.0, 'n_estimators': 100,
'n_jobs': None, 'oob_score': False, 'random_state': 1, 'verbose': 0,
```

```
'warm_start': False}
Creation date: 2022-08-27 12:02:25
Last fit date: 2022-08-27 12:02:25
Skforecast version: 0.4.3
```

[63]:
```python
# use the trained model to predict the next 48 months (same amount in test␣
↪dataset to allow us to test the forecasts accuracy)
Months = 48
PredictionsAU = ForecastModelAU.predict(steps=Months, exog =␣
↪TestAU[['AustinPrecipSum', 'AustinSnowSum']])
PredictionsAU.head(10)
```

[63]:
```
2018-01-01    55.88
2018-02-01    61.76
2018-03-01    70.37
2018-04-01    72.76
2018-05-01    76.12
2018-06-01    85.04
2018-07-01    84.79
2018-08-01    87.22
2018-09-01    78.75
2018-10-01    69.69
Freq: MS, Name: pred, dtype: float64
```

[64]:
```python
# Plot the training data, test data, and the predictions to visually see the␣
↪prediction accuracy
fig, ax = plt.subplots(figsize=(10, 3))
TrainAU['AustinAvgTemp'].plot(ax=ax, label='TrainAustin')
TestAU['AustinAvgTemp'].plot(ax=ax, label='TestAustin')
PredictionsAU.plot(ax=ax, label='PredictionsAustin')
ax.set_ylabel('Average Temp.F')
ax.set_title('Austin: Model Prediction vs Actual Average Temp.F from 2000-2021')
ax.legend();
```



[65]:
```
# Calculate the MSE (mean squared error)
MseAU = mean_squared_error(y_true = TestAU['AustinAvgTemp'], y_pred =␣
 ↪PredictionsAU)
print(f"Austin MSE Value: {MseAU}")
```

```
Austin MSE Value: 18.771893812499965
```

[66]:
```
# Calculate the RMSE (root mean squared error)
RmseAU = np.sqrt(MseAU)
print(f"Austin RMSE Value: {RmseAU}")
```

```
Austin RMSE Value: 4.332654361070124
```

[67]:
```
# Setup the forecasting model to forecast future
ForecastModelAUF = ForecasterAutoreg(regressor =␣
 ↪RandomForestRegressor(random_state=1), lags = 12)
ForecastModelAUF.fit(y = dfAU['AustinAvgTemp'], exog = dfAU[['AustinPrecipSum',␣
 ↪'AustinSnowSum']])
ForecastModelAUF
```

[67]:
```
=================
ForecasterAutoreg
=================
Regressor: RandomForestRegressor(random_state=1)
Lags: [ 1  2  3  4  5  6  7  8  9 10 11 12]
Window size: 12
Included exogenous: True
Type of exogenous variable: <class 'pandas.core.frame.DataFrame'>
Exogenous variables names: ['AustinPrecipSum', 'AustinSnowSum']
Training range: [Timestamp('2000-01-01 00:00:00'), Timestamp('2021-12-01
00:00:00')]
Training index type: DatetimeIndex
Training index frequency: MS
Regressor parameters: {'bootstrap': True, 'ccp_alpha': 0.0, 'criterion':
'squared_error', 'max_depth': None, 'max_features': 1.0, 'max_leaf_nodes': None,
'max_samples': None, 'min_impurity_decrease': 0.0, 'min_samples_leaf': 1,
'min_samples_split': 2, 'min_weight_fraction_leaf': 0.0, 'n_estimators': 100,
'n_jobs': None, 'oob_score': False, 'random_state': 1, 'verbose': 0,
'warm_start': False}
Creation date: 2022-08-27 12:02:26
Last fit date: 2022-08-27 12:02:26
Skforecast version: 0.4.3
```

[68]:
```
# use the trained model to predict the next 60 months (same amount in test␣
 ↪dataset to allow us to test the forecasts accuracy)
MonthsF = 60
```

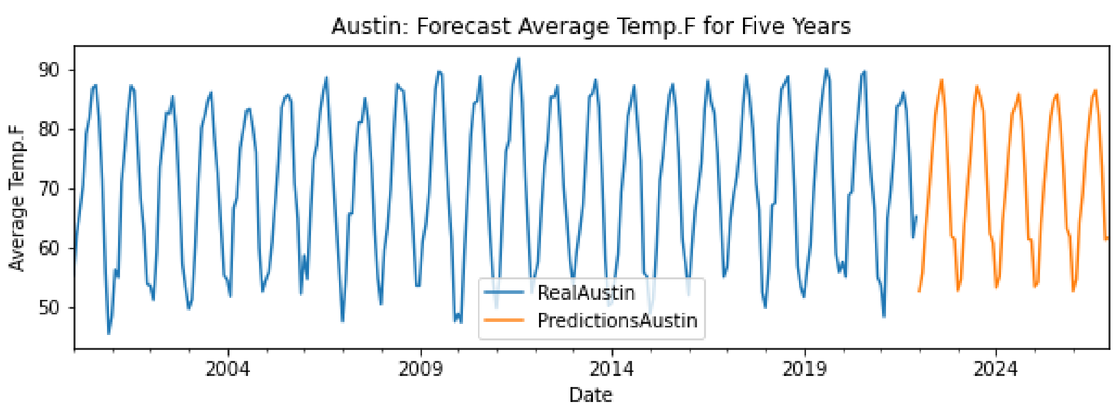
```
PredictionsAUF = ForecastModelAUF.predict(steps=MonthsF, exog =␣
↪dfAU[['AustinPrecipSum', 'AustinSnowSum']])
PredictionsAUF
```

[68]:

```
2022-01-01    52.58
2022-02-01    55.70
2022-03-01    63.30
2022-04-01    69.18
2022-05-01    75.79
2022-06-01    82.18
2022-07-01    85.38
2022-08-01    88.11
2022-09-01    83.07
2022-10-01    72.82
2022-11-01    61.82
2022-12-01    61.32
2023-01-01    52.69
2023-02-01    54.58
2023-03-01    63.46
2023-04-01    69.92
2023-05-01    75.85
2023-06-01    83.41
2023-07-01    87.04
2023-08-01    85.23
2023-09-01    82.79
2023-10-01    72.28
2023-11-01    62.33
2023-12-01    60.72
2024-01-01    53.15
2024-02-01    55.06
2024-03-01    64.32
2024-04-01    70.04
2024-05-01    76.79
2024-06-01    82.38
2024-07-01    83.73
2024-08-01    85.72
2024-09-01    81.23
2024-10-01    72.38
2024-11-01    61.35
2024-12-01    61.15
2025-01-01    53.31
2025-02-01    54.07
2025-03-01    64.10
2025-04-01    70.89
2025-05-01    76.71
2025-06-01    81.89
2025-07-01    84.74
```

```
2025-08-01                  85.67
2025-09-01                  80.77
2025-10-01                  72.88
2025-11-01                  63.17
2025-12-01                  61.69
2026-01-01                  52.58
2026-02-01                  54.55
2026-03-01                  64.19
2026-04-01                  68.95
2026-05-01                  76.27
2026-06-01                  81.13
2026-07-01                  85.19
2026-08-01                  86.37
2026-09-01                  82.09
2026-10-01                  73.37
2026-11-01                  61.22
2026-12-01                  61.57
Freq: MS, Name: pred, dtype: float64
```

[69]:
```
# Plot the training data and the predictions to visually see the predictions
fig, ax = plt.subplots(figsize=(10, 3))
dfAU['AustinAvgTemp'].plot(ax=ax, label='RealAustin')
PredictionsAUF.plot(ax=ax, label='PredictionsAustin')
ax.set_ylabel('Average Temp.F')
ax.set_title('Austin: Forecast Average Temp.F for Five Years')
ax.legend();
```



[70]:
```
# Brownsville City Weather Forecasting
```

[71]:
```
# create test and train datasets (using the last 48 months as the test dataset)
Months = 48
TrainB = dfB[:-Months]
```

```
TestB  = dfB[-Months:]
```

[72]:
```
# Setup the forecasting model
ForecastModelB = ForecasterAutoreg(regressor =␣
↪RandomForestRegressor(random_state=1), lags = 12)
ForecastModelB.fit(y = TrainB['BrownsvilleAvgTemp'], exog =␣
↪TrainB[['BrownsvillePrecipSum', 'BrownsvilleSnowSum']])
ForecastModelB
```

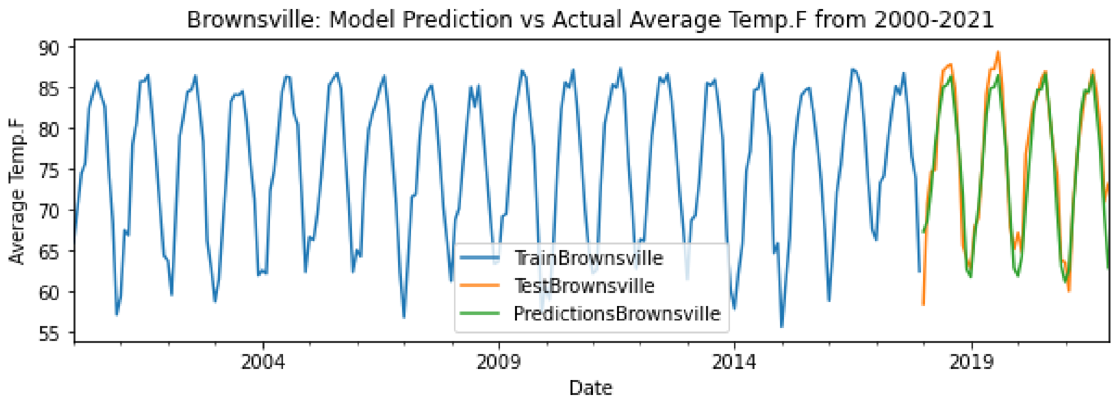
[72]:
```
=================
ForecasterAutoreg
=================
Regressor: RandomForestRegressor(random_state=1)
Lags: [ 1  2  3  4  5  6  7  8  9 10 11 12]
Window size: 12
Included exogenous: True
Type of exogenous variable: <class 'pandas.core.frame.DataFrame'>
Exogenous variables names: ['BrownsvillePrecipSum', 'BrownsvilleSnowSum']
Training range: [Timestamp('2000-01-01 00:00:00'), Timestamp('2017-12-01
00:00:00')]
Training index type: DatetimeIndex
Training index frequency: MS
Regressor parameters: {'bootstrap': True, 'ccp_alpha': 0.0, 'criterion':
'squared_error', 'max_depth': None, 'max_features': 1.0, 'max_leaf_nodes': None,
'max_samples': None, 'min_impurity_decrease': 0.0, 'min_samples_leaf': 1,
'min_samples_split': 2, 'min_weight_fraction_leaf': 0.0, 'n_estimators': 100,
'n_jobs': None, 'oob_score': False, 'random_state': 1, 'verbose': 0,
'warm_start': False}
Creation date: 2022-08-27 12:02:27
Last fit date: 2022-08-27 12:02:27
Skforecast version: 0.4.3
```

[73]:
```
# use the trained model to predict the next 48 months (same amount in test␣
↪dataset to allow us to test the forecasts accuracy)
Months = 48
PredictionsB = ForecastModelB.predict(steps=Months, exog =␣
↪TestB[['BrownsvillePrecipSum', 'BrownsvilleSnowSum']])
PredictionsB.head(10)
```

[73]:
```
2018-01-01              67.34
2018-02-01              68.68
2018-03-01              72.75
2018-04-01              77.88
2018-05-01              81.75
2018-06-01              84.99
2018-07-01              85.30
2018-08-01              86.36
```

```
2018-09-01                 82.33
2018-10-01                 77.64
Freq: MS, Name: pred, dtype: float64
```

[74]:
```
# Plot the training data, test data, and the predictions to visually see the␣
↪prediction accuracy
fig, ax = plt.subplots(figsize=(10, 3))
TrainB['BrownsvilleAvgTemp'].plot(ax=ax, label='TrainBrownsville')
TestB['BrownsvilleAvgTemp'].plot(ax=ax, label='TestBrownsville')
PredictionsB.plot(ax=ax, label='PredictionsBrownsville')
ax.set_ylabel('Average Temp.F')
ax.set_title('Brownsville: Model Prediction vs Actual Average Temp.F from␣
↪2000-2021')
ax.legend();
```



[75]:
```
# Calculate the MSE (mean squared error)
MseB = mean_squared_error(y_true = TestB['BrownsvilleAvgTemp'], y_pred =␣
↪PredictionsB)
print(f"Brownsville MSE Value: {MseB}")
```

```
Brownsville MSE Value: 9.249766083333373
```

[76]:
```
# Calculate the RMSE (root mean squared error)
RmseB = np.sqrt(MseB)
print(f"Brownsville RMSE Value: {RmseB}")
```

```
Brownsville RMSE Value: 3.04134280924288
```

[77]:
```
# Setup the forecasting model to forecast future
ForecastModelBF = ForecasterAutoreg(regressor =␣
↪RandomForestRegressor(random_state=1), lags = 12)
ForecastModelBF.fit(y = dfB['BrownsvilleAvgTemp'], exog =␣
↪dfB[['BrownsvillePrecipSum', 'BrownsvilleSnowSum']])
```

```
ForecastModelBF
```

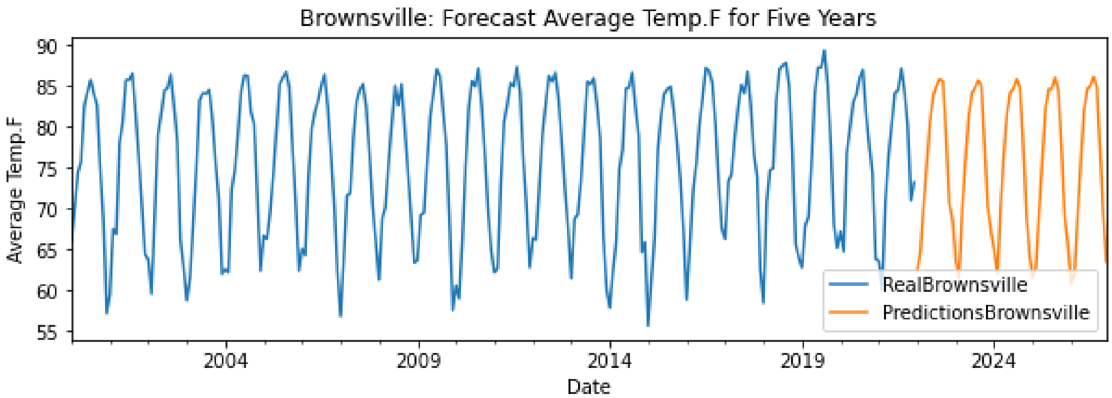
[77]:
```
=================
ForecasterAutoreg
=================
Regressor: RandomForestRegressor(random_state=1)
Lags: [ 1  2  3  4  5  6  7  8  9 10 11 12]
Window size: 12
Included exogenous: True
Type of exogenous variable: <class 'pandas.core.frame.DataFrame'>
Exogenous variables names: ['BrownsvillePrecipSum', 'BrownsvilleSnowSum']
Training range: [Timestamp('2000-01-01 00:00:00'), Timestamp('2021-12-01
00:00:00')]
Training index type: DatetimeIndex
Training index frequency: MS
Regressor parameters: {'bootstrap': True, 'ccp_alpha': 0.0, 'criterion':
'squared_error', 'max_depth': None, 'max_features': 1.0, 'max_leaf_nodes': None,
'max_samples': None, 'min_impurity_decrease': 0.0, 'min_samples_leaf': 1,
'min_samples_split': 2, 'min_weight_fraction_leaf': 0.0, 'n_estimators': 100,
'n_jobs': None, 'oob_score': False, 'random_state': 1, 'verbose': 0,
'warm_start': False}
Creation date: 2022-08-27 12:02:28
Last fit date: 2022-08-27 12:02:28
Skforecast version: 0.4.3
```

[78]:
```
# use the trained model to predict the next 60 months (same amount in test␣
↪dataset to allow us to test the forecasts accuracy)
MonthsF = 60
PredictionsBF = ForecastModelBF.predict(steps=MonthsF, exog =␣
↪dfB[['BrownsvillePrecipSum', 'BrownsvilleSnowSum']])
PredictionsBF
```

[78]:
```
2022-01-01                  62.45
2022-02-01                  64.92
2022-03-01                  70.44
2022-04-01                  75.62
2022-05-01                  80.77
2022-06-01                  83.86
2022-07-01                  84.98
2022-08-01                  85.82
2022-09-01                  85.48
2022-10-01                  78.39
2022-11-01                  70.59
2022-12-01                  68.65
2023-01-01                  63.63
2023-02-01                  61.55
2023-03-01                  69.64
```

```
2023-04-01                75.86
2023-05-01                81.30
2023-06-01                84.07
2023-07-01                84.73
2023-08-01                85.66
2023-09-01                85.05
2023-10-01                77.64
2023-11-01                70.12
2023-12-01                67.14
2024-01-01                64.87
2024-02-01                62.07
2024-03-01                70.43
2024-04-01                76.75
2024-05-01                81.63
2024-06-01                84.12
2024-07-01                84.68
2024-08-01                85.80
2024-09-01                84.78
2024-10-01                77.94
2024-11-01                69.44
2024-12-01                64.40
2025-01-01                61.58
2025-02-01                62.75
2025-03-01                69.43
2025-04-01                75.86
2025-05-01                82.32
2025-06-01                84.57
2025-07-01                84.61
2025-08-01                85.97
2025-09-01                84.47
2025-10-01                77.44
2025-11-01                69.07
2025-12-01                65.52
2026-01-01                60.84
2026-02-01                61.91
2026-03-01                68.90
2026-04-01                75.62
2026-05-01                81.97
2026-06-01                84.55
2026-07-01                85.02
2026-08-01                86.07
2026-09-01                84.88
2026-10-01                77.38
2026-11-01                69.33
2026-12-01                63.52
Freq: MS, Name: pred, dtype: float64
```

[79]:
```
# Plot the training data and the predictions to visually see the predictions
fig, ax = plt.subplots(figsize=(10, 3))
dfB['BrownsvilleAvgTemp'].plot(ax=ax, label='RealBrownsville')
PredictionsBF.plot(ax=ax, label='PredictionsBrownsville')
ax.set_ylabel('Average Temp.F')
ax.set_title('Brownsville: Forecast Average Temp.F for Five Years')
ax.legend();
```



[80]:
```
# Laredo City Weather Forecasting
```

[81]:
```
# create test and train datasets (using the last 48 months as the test dataset)
Months = 48
TrainL = dfL[:-Months]
TestL  = dfL[-Months:]
```

[82]:
```
# Setup the forecasting model
ForecastModelL = ForecasterAutoreg(regressor =␣
↪RandomForestRegressor(random_state=1), lags = 12)
ForecastModelL.fit(y = TrainL['LaredoAvgTemp'], exog =␣
↪TrainL[['LaredoPrecipSum', 'LaredoSnowSum']])
ForecastModelL
```

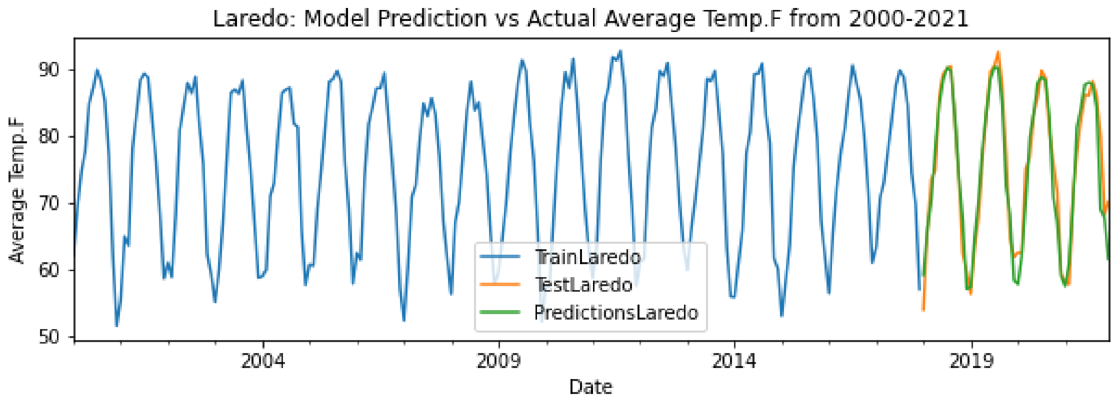
[82]:
```
=================
ForecasterAutoreg
=================
Regressor: RandomForestRegressor(random_state=1)
Lags: [ 1  2  3  4  5  6  7  8  9 10 11 12]
Window size: 12
Included exogenous: True
Type of exogenous variable: <class 'pandas.core.frame.DataFrame'>
Exogenous variables names: ['LaredoPrecipSum', 'LaredoSnowSum']
Training range: [Timestamp('2000-01-01 00:00:00'), Timestamp('2017-12-01
```

```
00:00:00')]
Training index type: DatetimeIndex
Training index frequency: MS
Regressor parameters: {'bootstrap': True, 'ccp_alpha': 0.0, 'criterion':
'squared_error', 'max_depth': None, 'max_features': 1.0, 'max_leaf_nodes': None,
'max_samples': None, 'min_impurity_decrease': 0.0, 'min_samples_leaf': 1,
'min_samples_split': 2, 'min_weight_fraction_leaf': 0.0, 'n_estimators': 100,
'n_jobs': None, 'oob_score': False, 'random_state': 1, 'verbose': 0,
'warm_start': False}
Creation date: 2022-08-27 12:02:28
Last fit date: 2022-08-27 12:02:29
Skforecast version: 0.4.3
```

[83]:
```
# use the trained model to predict the next 48 months (same amount in test␣
↪dataset to allow us to test the forecasts accuracy)
Months = 48
PredictionsL = ForecastModelL.predict(steps=Months, exog =␣
↪TestL[['LaredoPrecipSum', 'LaredoSnowSum']])
PredictionsL.head(10)
```

[83]:
```
2018-01-01                  59.14
2018-02-01                  65.94
2018-03-01                  70.44
2018-04-01                  77.87
2018-05-01                  84.47
2018-06-01                  88.42
2018-07-01                  90.20
2018-08-01                  89.66
2018-09-01                  82.85
2018-10-01                  73.86
Freq: MS, Name: pred, dtype: float64
```

[84]:
```
# Plot the training data, test data, and the predictions to visually see the␣
↪prediction accuracy
fig, ax = plt.subplots(figsize=(10, 3))
TrainL['LaredoAvgTemp'].plot(ax=ax, label='TrainLaredo')
TestL['LaredoAvgTemp'].plot(ax=ax, label='TestLaredo')
PredictionsL.plot(ax=ax, label='PredictionsLaredo')
ax.set_ylabel('Average Temp.F')
ax.set_title('Laredo: Model Prediction vs Actual Average Temp.F from 2000-2021')
ax.legend();
```



[85]:
```
# Calculate the MSE (mean squared error)
MseL = mean_squared_error(y_true = TestL['LaredoAvgTemp'], y_pred =␣
 ↪PredictionsL)
print(f"Laredo MSE Value: {MseL}")
```

```
Laredo MSE Value: 10.563244208333318
```

[86]:
```
# Calculate the RMSE (root mean squared error)
RmseL = np.sqrt(MseL)
print(f"Laredo RMSE Value: {RmseL}")
```

```
Laredo RMSE Value: 3.2501144915730764
```

[87]:
```
# Setup the forecasting model to forecast future
ForecastModelLF = ForecasterAutoreg(regressor =␣
 ↪RandomForestRegressor(random_state=1), lags = 12)
ForecastModelLF.fit(y = dfL['LaredoAvgTemp'], exog = dfL[['LaredoPrecipSum',␣
 ↪'LaredoSnowSum']])
ForecastModelLF
```

[87]:
```
=================
ForecasterAutoreg
=================
Regressor: RandomForestRegressor(random_state=1)
Lags: [ 1  2  3  4  5  6  7  8  9 10 11 12]
Window size: 12
Included exogenous: True
Type of exogenous variable: <class 'pandas.core.frame.DataFrame'>
Exogenous variables names: ['LaredoPrecipSum', 'LaredoSnowSum']
Training range: [Timestamp('2000-01-01 00:00:00'), Timestamp('2021-12-01
00:00:00')]
Training index type: DatetimeIndex
Training index frequency: MS
```

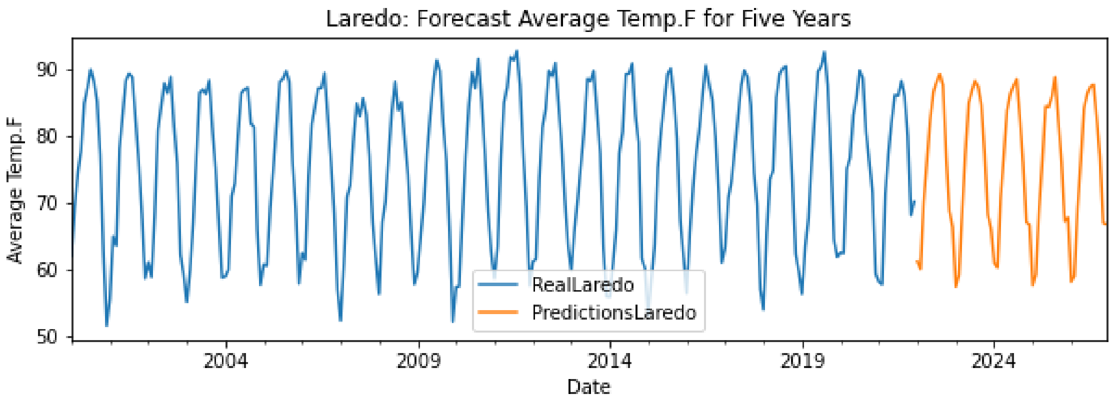
```
Regressor parameters: {'bootstrap': True, 'ccp_alpha': 0.0, 'criterion':
'squared_error', 'max_depth': None, 'max_features': 1.0, 'max_leaf_nodes': None,
'max_samples': None, 'min_impurity_decrease': 0.0, 'min_samples_leaf': 1,
'min_samples_split': 2, 'min_weight_fraction_leaf': 0.0, 'n_estimators': 100,
'n_jobs': None, 'oob_score': False, 'random_state': 1, 'verbose': 0,
'warm_start': False}
Creation date: 2022-08-27 12:02:29
Last fit date: 2022-08-27 12:02:29
Skforecast version: 0.4.3
```

[88]:
```
# use the trained model to predict the next 60 months (same amount in test␣
 ↪dataset to allow us to test the forecasts accuracy)
MonthsF = 60
PredictionsLF = ForecastModelLF.predict(steps=MonthsF, exog =␣
 ↪dfL[['LaredoPrecipSum', 'LaredoSnowSum']])
PredictionsLF
```

[88]:
```
2022-01-01    61.15
2022-02-01    60.01
2022-03-01    69.98
2022-04-01    76.18
2022-05-01    82.39
2022-06-01    86.49
2022-07-01    88.03
2022-08-01    89.21
2022-09-01    87.53
2022-10-01    76.86
2022-11-01    68.80
2022-12-01    66.32
2023-01-01    57.36
2023-02-01    59.13
2023-03-01    68.53
2023-04-01    77.47
2023-05-01    84.86
2023-06-01    86.71
2023-07-01    88.22
2023-08-01    87.34
2023-09-01    84.61
2023-10-01    77.34
2023-11-01    68.21
2023-12-01    66.01
2024-01-01    60.91
2024-02-01    60.26
2024-03-01    70.72
2024-04-01    77.10
2024-05-01    83.78
2024-06-01    86.25
```

```
2024-07-01                   87.50
2024-08-01                   88.45
2024-09-01                   82.64
2024-10-01                   75.88
2024-11-01                   67.02
2024-12-01                   66.67
2025-01-01                   57.55
2025-02-01                   59.23
2025-03-01                   68.80
2025-04-01                   76.08
2025-05-01                   84.39
2025-06-01                   84.29
2025-07-01                   85.84
2025-08-01                   88.75
2025-09-01                   82.17
2025-10-01                   76.05
2025-11-01                   67.17
2025-12-01                   67.77
2026-01-01                   58.11
2026-02-01                   58.99
2026-03-01                   69.01
2026-04-01                   76.17
2026-05-01                   84.14
2026-06-01                   86.29
2026-07-01                   87.34
2026-08-01                   87.64
2026-09-01                   82.54
2026-10-01                   76.74
2026-11-01                   66.91
2026-12-01                   66.75
Freq: MS, Name: pred, dtype: float64
```

[89]:
```
# Plot the training data and the predictions to visually see the predictions
fig, ax = plt.subplots(figsize=(10, 3))
dfL['LaredoAvgTemp'].plot(ax=ax, label='RealLaredo')
PredictionsLF.plot(ax=ax, label='PredictionsLaredo')
ax.set_ylabel('Average Temp.F')
ax.set_title('Laredo: Forecast Average Temp.F for Five Years')
ax.legend();
```



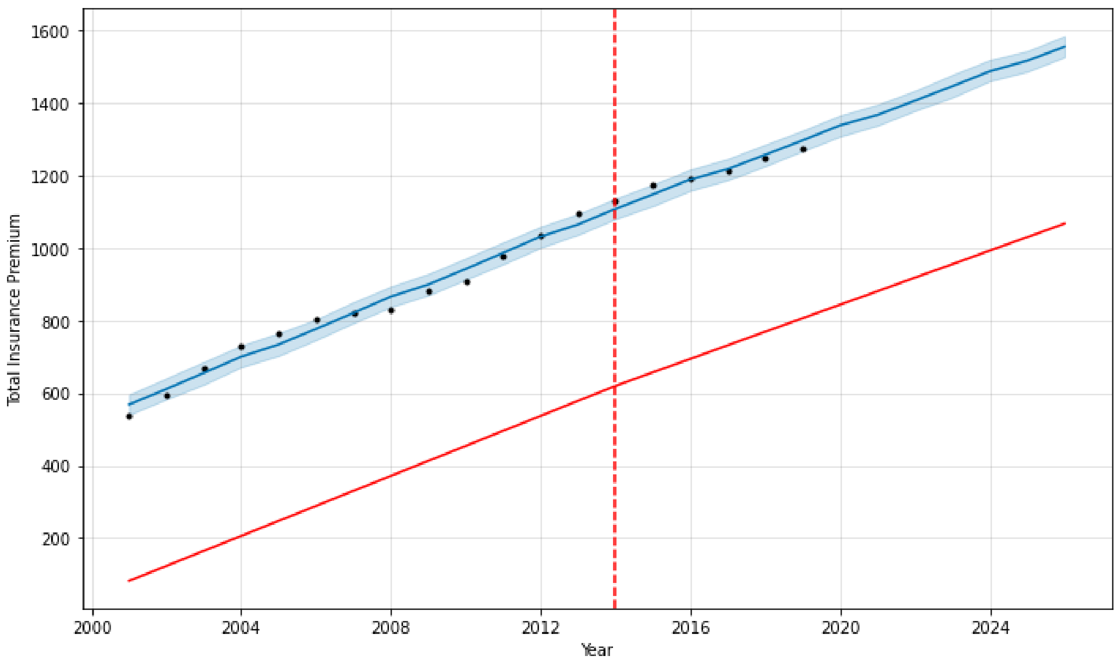
[90]:
```
# load Texas HomeOwners Insurance dataset to Pandas Dataframe
dfI = pd.read_csv('C:/Users/klein/Desktop/Quarter 5 - Current Linked to Backup/
↪IST 718/Project/Project Working Folder/HomeOwnersInsuranceData1.csv')
```

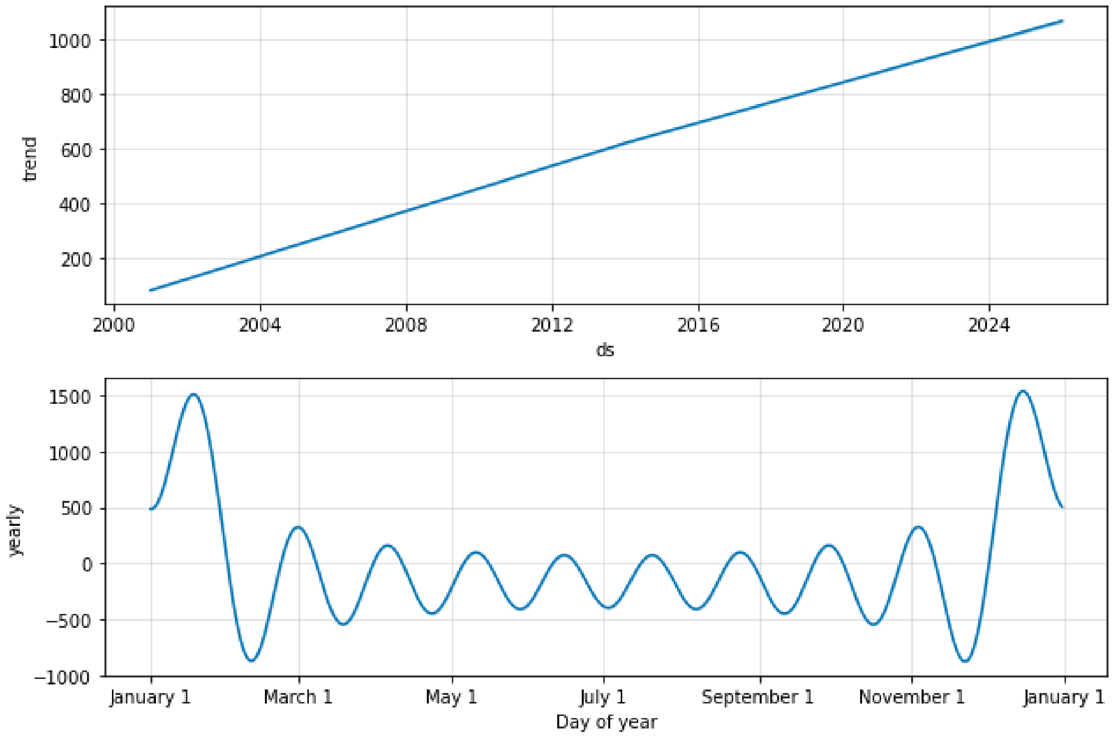
[91]:
```
# rename fields to prophet naming conventions and change Year to datetime format
dfI.rename(columns = {'Year':'ds'}, inplace = True)
dfI.rename(columns = {'Total':'y'}, inplace = True)
dfI['ds'] = pd.to_datetime(dfI.ds, format='%Y')
```

[92]:
```
# Use prophet for Texas HomeOwners Insurnance Forecasting
prophet = prophet.Prophet(yearly_seasonality = True)
prophet.fit(dfI)
future = prophet.make_future_dataframe(periods=7, freq='YS')
forecast = prophet.predict(future)
fig = prophet.plot(forecast, xlabel = 'Year', ylabel = 'Total Insurance␣
↪Premium')
a = add_changepoints_to_plot(fig.gca(), prophet, forecast)
fig = prophet.plot_components(forecast)
```

```
cmdstanpy  DEBUG cmd: where.exe tbb.dll
cwd: None
cmdstanpy  DEBUG Adding TBB (C:\Users\klein\anaconda3\lib\site-
packages\prophet\stan_model\cmdstan-2.26.1\stan\lib\stan_math\lib\tbb) to PATH
prophet    INFO  Disabling weekly seasonality. Run prophet with
weekly_seasonality=True to override this.
prophet    INFO  Disabling daily seasonality. Run prophet with
daily_seasonality=True to override this.
prophet    INFO  n_changepoints greater than number of observations. Using 14.
cmdstanpy  DEBUG input tempfile:
C:\Users\klein\AppData\Local\Temp\tmpvvblk02h\07kf2s3g.json
cmdstanpy  DEBUG input tempfile:
C:\Users\klein\AppData\Local\Temp\tmpvvblk02h\18zwsewd.json
cmdstanpy  DEBUG idx 0
```

```
cmdstanpy  DEBUG running CmdStan, num_threads: None
cmdstanpy  DEBUG CmdStan args: ['C:\\Users\\klein\\anaconda3\\Lib\\site-
packages\\prophet\\stan_model\\prophet_model.bin', 'random', 'seed=89482',
'data',
'file=C:\\Users\\klein\\AppData\\Local\\Temp\\tmpvvblk02h\\07kf2s3g.json',
'init=C:\\Users\\klein\\AppData\\Local\\Temp\\tmpvvblk02h\\18zwsewd.json',
'output', 'file=C:\\Users\\klein\\AppData\\Local\\Temp\\tmpt1hnn31a\\prophet_mod
el-20220827120231.csv', 'method=optimize', 'algorithm=newton', 'iter=10000']
12:02:31 - cmdstanpy - INFO - Chain [1] start processing
cmdstanpy  INFO  Chain [1] start processing
12:02:31 - cmdstanpy - INFO - Chain [1] done processing
cmdstanpy  INFO  Chain [1] done processing
```





[93]:
```python
# examine forecast
forecast
forecast2 = forecast[['ds', 'yhat']]
forecast2['yhat'].round(2)
forecast2
```

[93]:

	ds	yhat
0	2001-01-01	568.11
1	2002-01-01	611.17
2	2003-01-01	655.28
3	2004-01-01	700.45
4	2005-01-01	733.74
5	2006-01-01	776.78
6	2007-01-01	820.87
7	2008-01-01	866.01
8	2009-01-01	899.27
9	2010-01-01	942.30
10	2011-01-01	986.38
11	2012-01-01	1,031.52
12	2013-01-01	1,064.79
13	2014-01-01	1,107.77
14	2015-01-01	1,147.80
15	2016-01-01	1,188.85

16	2017-01-01	1,218.02
17	2018-01-01	1,256.95
18	2019-01-01	1,296.95
19	2020-01-01	1,337.99
20	2021-01-01	1,367.16
21	2022-01-01	1,406.10
22	2023-01-01	1,446.09
23	2024-01-01	1,487.14
24	2025-01-01	1,516.30
25	2026-01-01	1,555.24