Project 3 Code

June 1, 2024

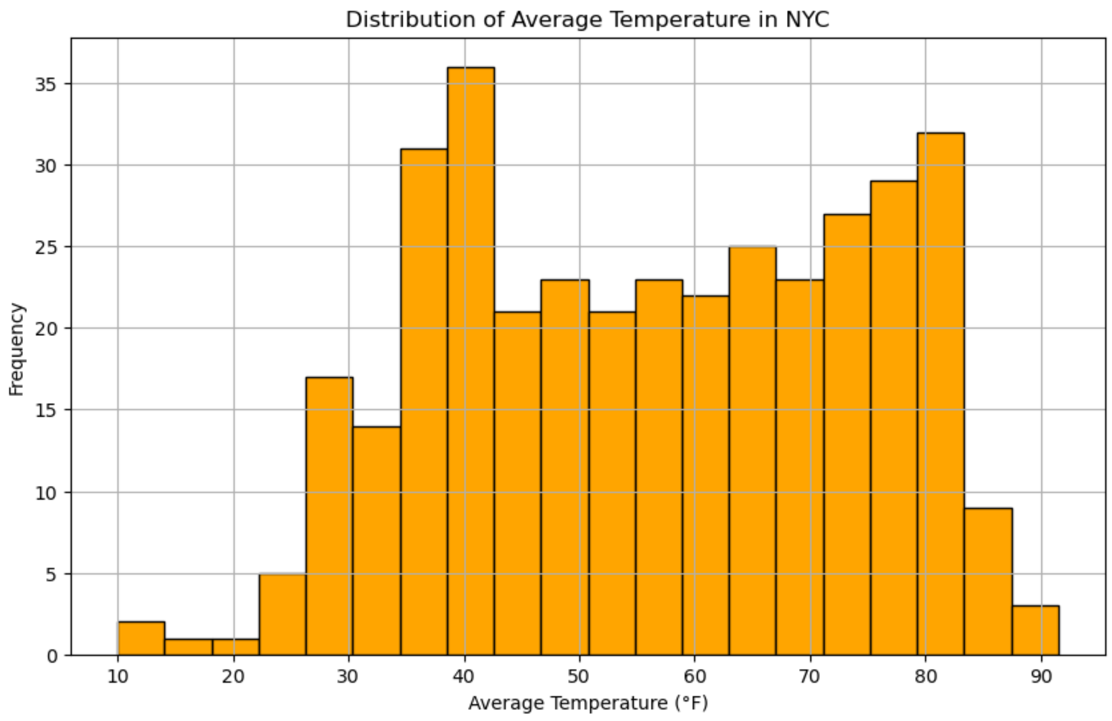
[19]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.tsa.statespace.sarimax import SARIMAX
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
```

[20]:
```
# Load the datasets
ev_charging_stations = pd.read_csv("C:/Users/mattl/Downloads/
 ↪Electric_Vehicle_Charging_Stations_in_New_York.csv")
nyc_temperature = pd.read_csv("C:/Users/mattl/Downloads/nyc_temperature.csv")

# Display the first few rows of each dataset to understand their structure
print(ev_charging_stations.head())
print(nyc_temperature.head())
```

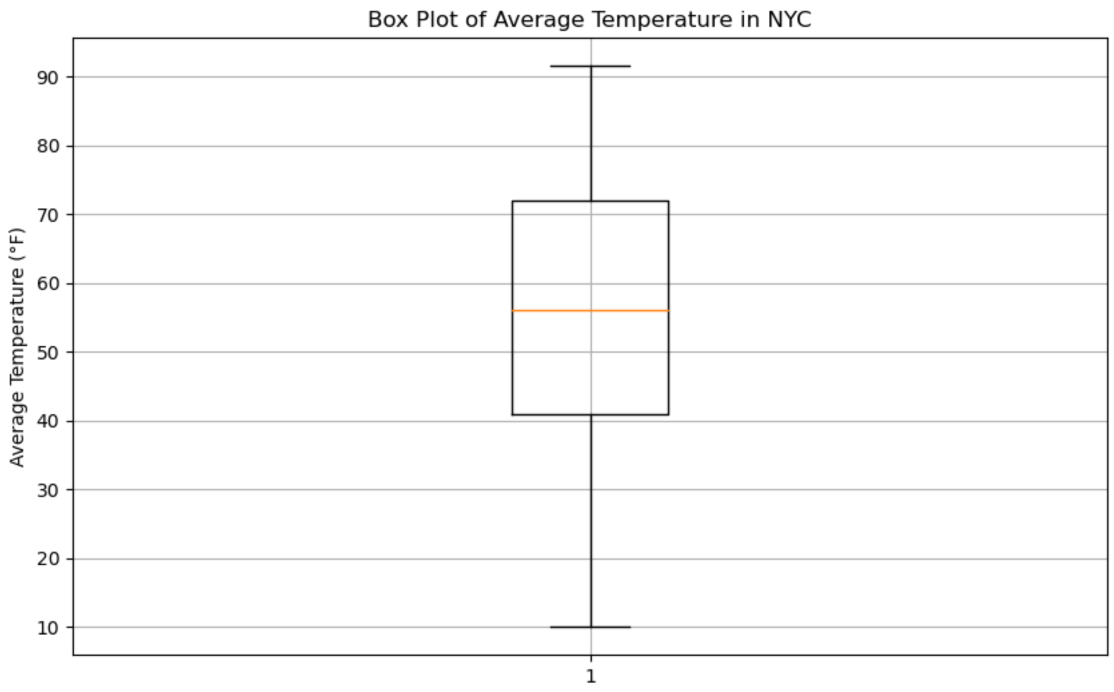
```
  Fuel Type Code            Station Name    Street Address  \
0           ELEC               Town Hall  159 Pantigo Road
1           ELEC  SUNY BUFFALO JACOBS 8  50 Augspurger Rd
2           ELEC     MUNI LOT STATION 2       2578 Main St
3           ELEC     SAREMBAS STATION 2     86 Congress St
4           ELEC       CORNELL STATION3         176 Hoy Rd

  Intersection Directions               City State    ZIP  Plus4 Station Phone  \
0                     NaN       East Hampton    NY  11937    NaN  866-816-7584
1                     NaN            Buffalo    NY  14228    NaN  888-758-4389
2                     NaN        Lake Placid    NY  12946    NaN  888-758-4389
3                     NaN  Saratoga Springs    NY  12866    NaN  888-758-4389
4                     NaN             Ithaca    NY  14853    NaN  888-758-4389

  Status Code  …   Latitude  Longitude Date Last Confirmed      ID  \
0           E  …  40.969547 -72.172070          05/18/2024  163947
1           E  …  42.998860 -78.786981          05/18/2024  182262
2           E  …  44.285366 -73.984300          05/18/2024  193882
3           E  …  43.078663 -73.790306          05/18/2024  227531
4           E  …  42.443450 -76.479460          05/18/2024  229400
```

```
            Updated At  Owner Type Code  Federal Agency ID  \
0  2024-05-18 02:59:40              NaN                NaN
1  2024-05-18 01:21:34              NaN                NaN
2  2024-05-18 01:19:21              NaN                NaN
3  2024-05-18 01:44:17              NaN                NaN
4  2024-05-18 01:00:31              NaN                NaN

  Federal Agency Name   Open Date EV Connector Types
0                 NaN  06/26/2020              J1772
1                 NaN  01/27/2021              J1772
2                 NaN  08/14/2021              J1772
3                 NaN  09/14/2022              J1772
4                 NaN  09/28/2022              J1772

[5 rows x 31 columns]
     date  tmax  tmin  tavg  departure  HDD  CDD precipitation new_snow  \
0  1/1/19    60    40  50.0       13.9   15    0          0.08        0
1  2/1/19    41    35  38.0        2.1   27    0             0        0
2  3/1/19    45    39  42.0        6.3   23    0             T        0
3  4/1/19    47    37  42.0        6.5   23    0             0        0
4  5/1/19    47    42  44.5        9.1   20    0          0.45        0

  snow_depth
0          0
1          0
2          0
3          0
4          0
```

[21]:
```python
# Temperature Distribution
plt.figure(figsize=(10, 6))
plt.hist(nyc_temperature['tavg'], bins=20, color='orange', edgecolor='black')
plt.xlabel('Average Temperature (°F)')
plt.ylabel('Frequency')
plt.title('Distribution of Average Temperature in NYC')
plt.grid(True)
plt.show()
```

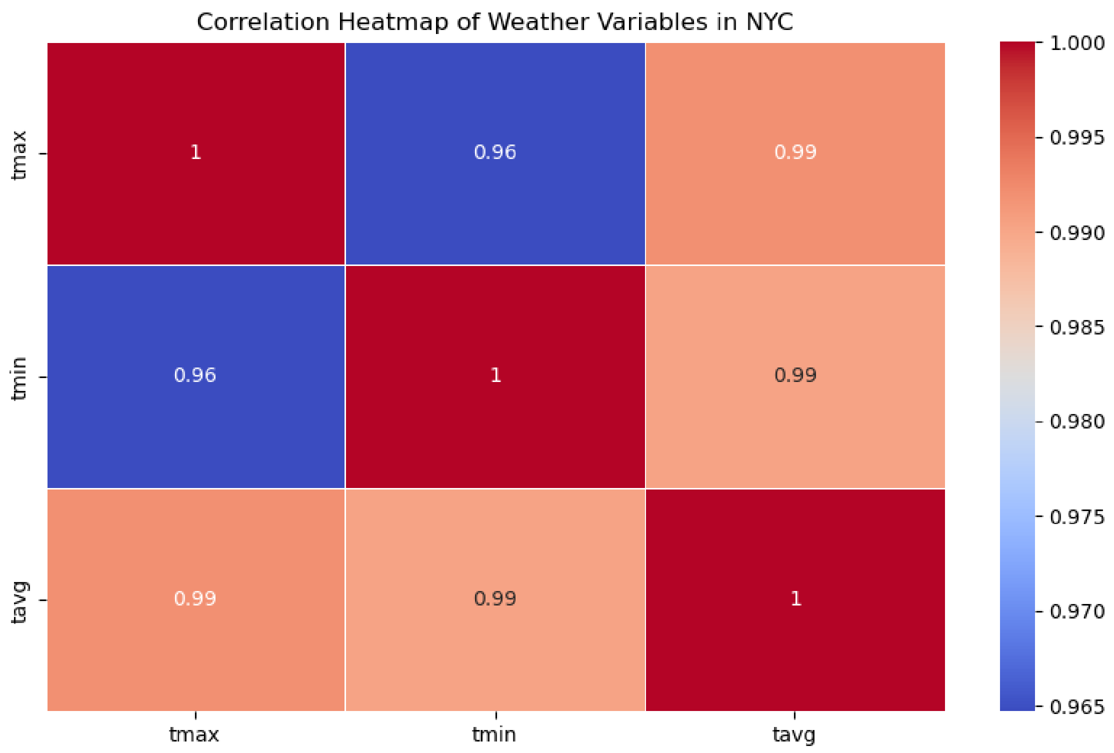


[22]:
```
# Box Plot of Temperature
plt.figure(figsize=(10, 6))
plt.boxplot(nyc_temperature['tavg'])
plt.ylabel('Average Temperature (°F)')
plt.title('Box Plot of Average Temperature in NYC')
plt.grid(True)
plt.show()
```



```
[23]: # Correlation Heatmap
      correlation_matrix = nyc_temperature[['tmax', 'tmin', 'tavg', 'precipitation']].
        ↪corr()

      plt.figure(figsize=(10, 6))
      sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
      plt.title('Correlation Heatmap of Weather Variables in NYC')
      plt.show()
```

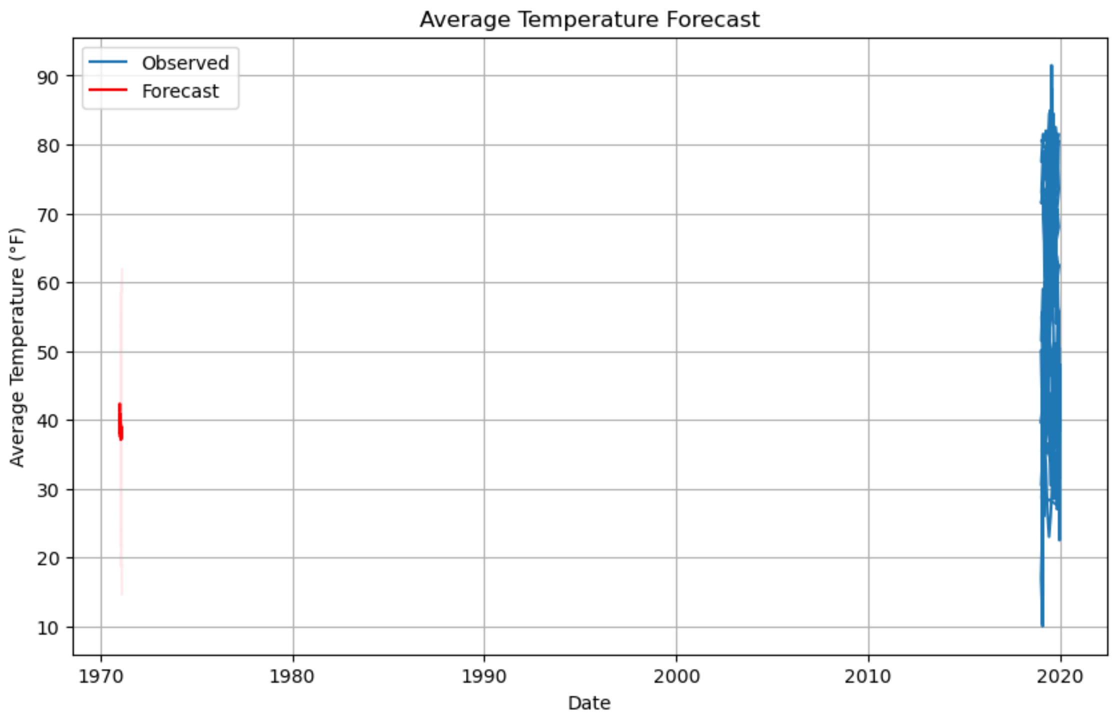
```
C:\Users\mattl\AppData\Local\Temp\ipykernel_29200\3633526517.py:2:
FutureWarning: The default value of numeric_only in DataFrame.corr is
deprecated. In a future version, it will default to False. Select only valid
columns or specify the value of numeric_only to silence this warning.
  correlation_matrix = nyc_temperature[['tmax', 'tmin', 'tavg',
'precipitation']].corr()
```



[24]:
```
# Convert 'date' column to datetime in nyc_temperature dataset
nyc_temperature['date'] = pd.to_datetime(nyc_temperature['date'],␣
↪errors='coerce')

# Drop rows with invalid dates
nyc_temperature = nyc_temperature.dropna(subset=['date'])

# Replace 'T' in precipitation with 0.001 and convert to float
nyc_temperature['precipitation'] = nyc_temperature['precipitation'].
↪replace('T', 0.001).astype(float)

# Convert 'Date Last Confirmed' to datetime in ev_charging_stations dataset
ev_charging_stations['Date Last Confirmed'] = pd.
↪to_datetime(ev_charging_stations['Date Last Confirmed'], errors='coerce')

# Merge the two datasets on date (assuming both datasets have date columns)
merged_data = pd.merge(nyc_temperature, ev_charging_stations, left_on='date',␣
↪right_on='Date Last Confirmed', how='inner')

# Select relevant columns for modeling
data = merged_data[['date', 'tavg', 'precipitation']]
```

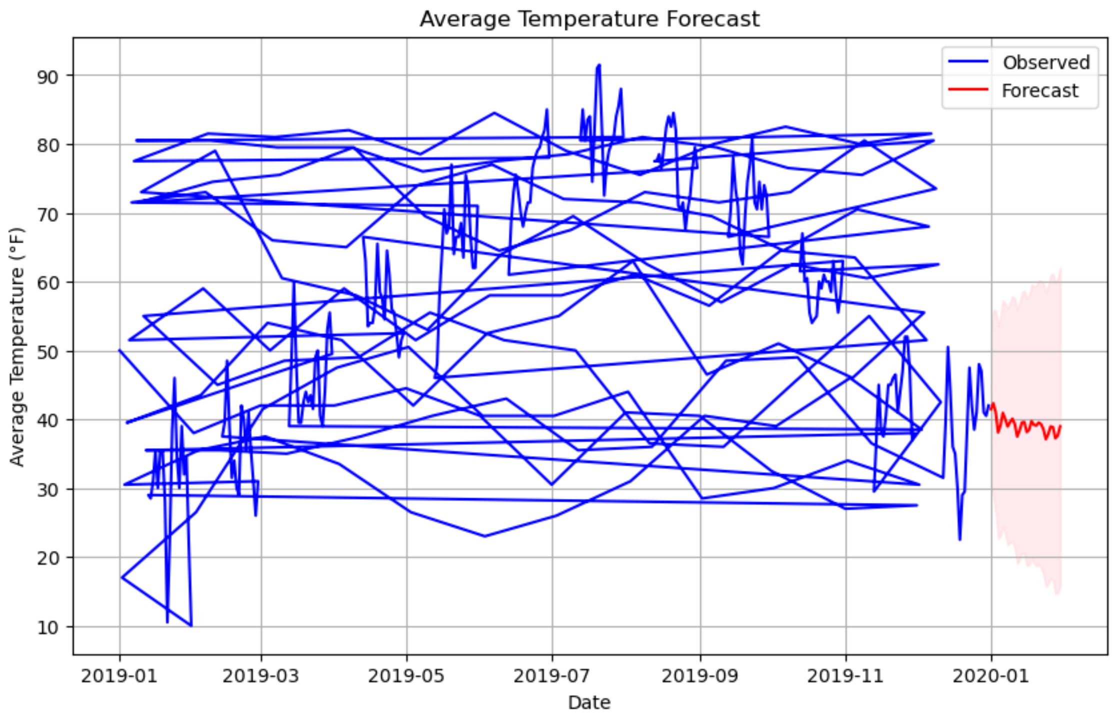
[25]:
```
# Fit SARIMA model
model = SARIMAX(nyc_temperature['tavg'], order=(1, 1, 1), seasonal_order=(1, 1,␣
 ↪1, 12))
results = model.fit()

# Generate forecast
forecast_steps = 30
forecast = results.get_forecast(steps=forecast_steps)
forecast_ci = forecast.conf_int()

# Align forecast dates with the observed data frequency
last_date = nyc_temperature['date'].max()
forecast_dates = pd.date_range(start=last_date, periods=forecast_steps + 1,␣
 ↪closed='right')


# Plot the results
plt.figure(figsize=(10, 6))
plt.plot(nyc_temperature['date'], nyc_temperature['tavg'], label='Observed')
plt.plot(forecast.predicted_mean.index, forecast.predicted_mean, color='r',␣
 ↪label='Forecast')
plt.fill_between(forecast_ci.index, forecast_ci.iloc[:, 0], forecast_ci.iloc[:,␣
 ↪1], color='pink', alpha=0.3)
plt.xlabel('Date')
plt.ylabel('Average Temperature (°F)')
plt.title('Average Temperature Forecast')
plt.legend()
plt.grid(True)
plt.show()
```

```
C:\Users\mattl\anaconda3\lib\site-
packages\statsmodels\tsa\statespace\sarimax.py:978: UserWarning: Non-invertible
starting MA parameters found. Using zeros as starting parameters.
  warn('Non-invertible starting MA parameters found.'
C:\Users\mattl\AppData\Local\Temp\ipykernel_29200\1336704412.py:12:
FutureWarning: Argument `closed` is deprecated in favor of `inclusive`.
  forecast_dates = pd.date_range(start=last_date, periods=forecast_steps + 1,
closed='right')
```



[26]:
```
# Fit SARIMA model
model = SARIMAX(nyc_temperature['tavg'], order=(1, 1, 1), seasonal_order=(1, 1,␣
↪1, 12))
results = model.fit()

# Generate forecast
forecast_steps = 30
forecast = results.get_forecast(steps=forecast_steps)
forecast_ci = forecast.conf_int()

# Align forecast dates with the observed data frequency
last_date = nyc_temperature['date'].max()
forecast_dates = pd.date_range(start=last_date, periods=forecast_steps + 1,␣
↪closed='right')

# Plot the results
plt.figure(figsize=(10, 6))
plt.plot(nyc_temperature['date'], nyc_temperature['tavg'], label='Observed',␣
↪color='blue')
plt.plot(forecast_dates, forecast.predicted_mean, color='r', label='Forecast')
plt.fill_between(forecast_dates, forecast_ci.iloc[:, 0], forecast_ci.iloc[:,␣
↪1], color='pink', alpha=0.3)
plt.xlabel('Date')
plt.ylabel('Average Temperature (°F)')
plt.title('Average Temperature Forecast')
plt.legend()
plt.grid(True)
plt.show()
```

```
C:\Users\mattl\anaconda3\lib\site-
packages\statsmodels\tsa\statespace\sarimax.py:978: UserWarning: Non-invertible
starting MA parameters found. Using zeros as starting parameters.
  warn('Non-invertible starting MA parameters found.'
C:\Users\mattl\AppData\Local\Temp\ipykernel_29200\3187447840.py:12:
FutureWarning: Argument `closed` is deprecated in favor of `inclusive`.
  forecast_dates = pd.date_range(start=last_date, periods=forecast_steps + 1,
closed='right')
```


Average Temperature Forecast
Observed
Forecast
Average Temperature (°F)
Date