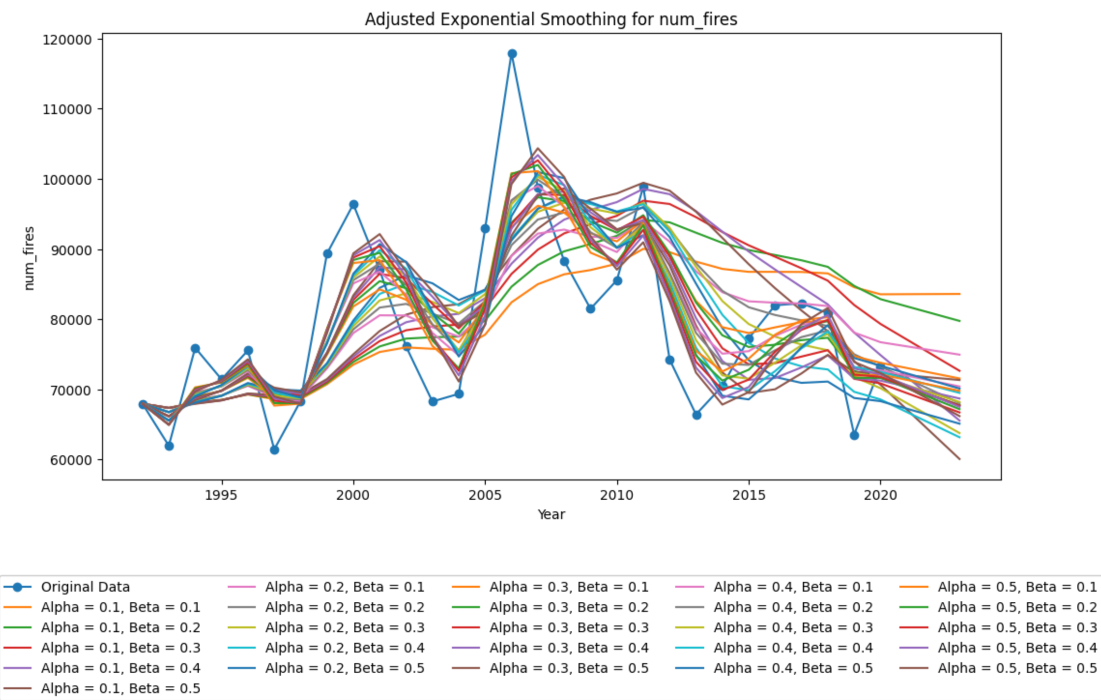
forecast

October 5, 2023

[46]:
```python
import sqlite3
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Define the Adjusted Exponential Smoothing function
def adjusted_exponential_smoothing(data, alpha, beta, forecast_years):
    forecast = [data.iloc[0]]  # Initial forecast is the first data point
    trend = [0]  # Initial trend is 0
    for i in range(1, len(data) + forecast_years):
        if i < len(data):
            forecast.append(alpha * data.iloc[i] + (1 - alpha) * (forecast[i -␣
↪1] + trend[i - 1]))
            trend.append(beta * (forecast[i] - forecast[i - 1]) + (1 - beta) *␣
↪trend[i - 1])
        else:
            forecast.append(forecast[-1] + trend[-1])
    return forecast

# Connect to the SQLite database
conn = sqlite3.connect('.database')  # Replace 'your_database.db' with the␣
↪actual database filename

# Define a list of alpha and beta values to try
alpha_values = [0.1, 0.2, 0.3, 0.4, 0.5]
beta_values = [0.1, 0.2, 0.3, 0.4, 0.5]

# Create a dictionary to store the forecasts for each column, alpha, and beta␣
↪value
forecasts_aes = {column: {alpha: {} for alpha in alpha_values} for column in␣
↪data.columns[1:]}

# Load data from the SQLite database
data = pd.read_sql_query("SELECT * FROM Data", conn)

# Forecasting parameters
forecast_years = 2023 - data['year'].max()  # Forecast to the year 2023
```

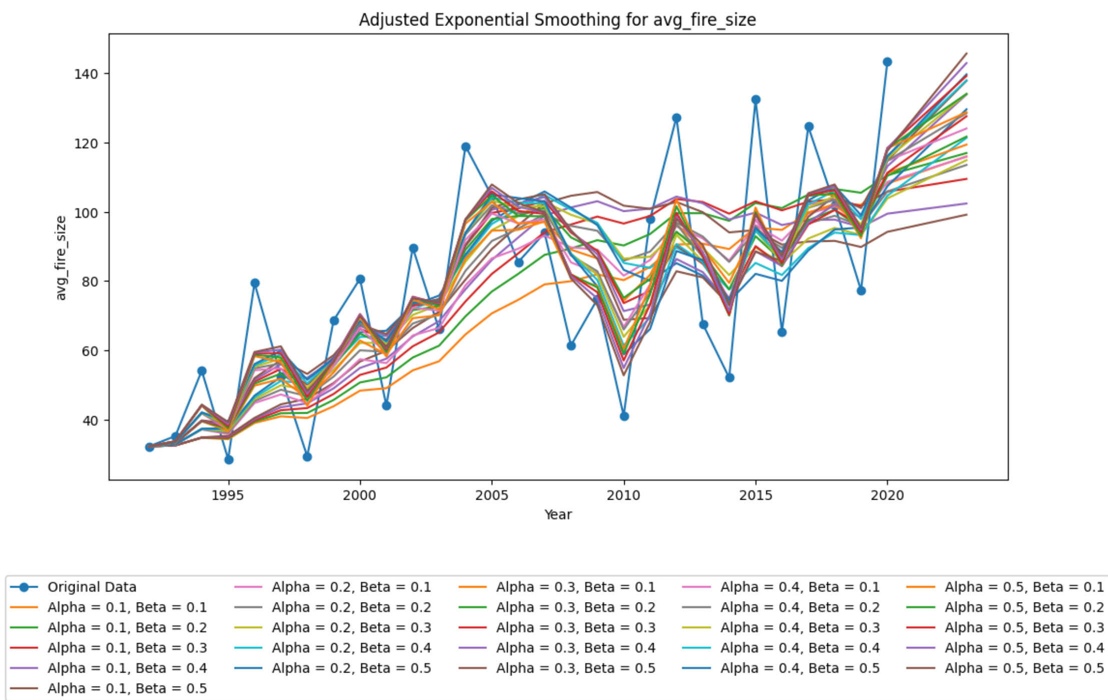
```
# Iterate through each column, alpha, and beta, and calculate forecasts
for column in data.columns[1:]:
    for alpha in alpha_values:
        for beta in beta_values:
            forecasts_aes[column][alpha][beta] =␣
 ↪adjusted_exponential_smoothing(data[column], alpha, beta, forecast_years)

# Create an array of years for plotting
years = data['year'].tolist() + list(range(data['year'].max() + 1, 2023 + 1))

# Predict the value for the year 2023 for each column, alpha, and beta, and␣
 ↪plot the graphs
year_to_predict = 2023
for column in data.columns[1:]:
    plt.figure(figsize=(12, 6))
    plt.plot(data['year'], data[column], label='Original Data', marker='o')
    for alpha, beta_dict in forecasts_aes[column].items():
        for beta, forecast in beta_dict.items():
            plt.plot(years, forecast, label=f'Alpha = {alpha}, Beta = {beta}')
    plt.title(f'Adjusted Exponential Smoothing for {column}')
    plt.xlabel('Year')
    plt.ylabel(column)

    # Place the legend underneath the plot
    plt.legend(loc='upper center', bbox_to_anchor=(0.5, -0.2), fancybox=True,␣
 ↪shadow=True, ncol=5)

    plt.show()
    print(f'Predicted values for {column} with different alphas and betas for␣
 ↪2023:')
    for alpha, beta_dict in forecasts_aes[column].items():
        for beta, forecast in beta_dict.items():
            predicted_value = forecast[-1]
            print(f'Alpha = {alpha}, Beta = {beta}: {predicted_value}')

# Calculate accuracy metrics (MAPE) for each column and alpha-beta combination
for column in data.columns[1:]:
    print(f'Accuracy metrics for {column} for 2023:')
    actual_values = data[column].tail(forecast_years).values
    for alpha, beta_dict in forecasts_aes[column].items():
        for beta, forecast in beta_dict.items():
            forecasted_values = forecast[-forecast_years:]
            mape = np.mean(np.abs((actual_values - forecasted_values) /␣
 ↪actual_values)) * 100
            mse = np.mean((actual_values - forecasted_values) ** 2)
            rmse = np.sqrt(mse)
```

```python
            print(f'Alpha = {alpha}, Beta = {beta}:')
            print(f'MAPE: {mape:.2f}%')
            print(f'MSE: {mse:.2f}')
            print(f'RMSE: {rmse:.2f}')
            print('-' * 40)

# Close the database connection
conn.close()
```



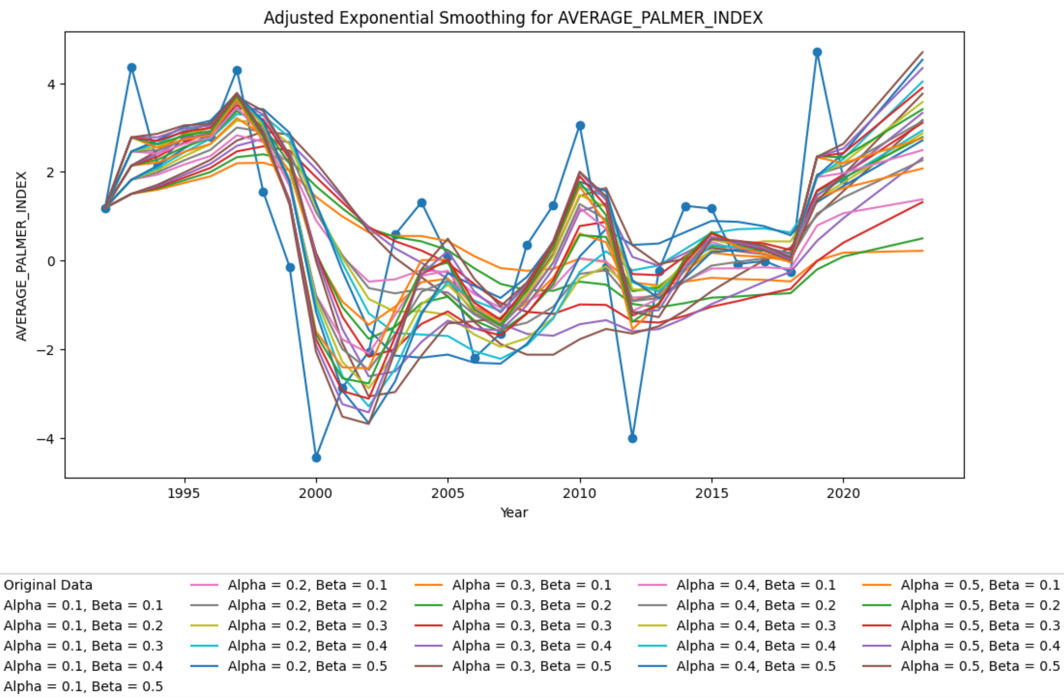

```
Predicted values for num_fires with different alphas and betas for 2023:
Alpha = 0.1, Beta = 0.1: 83607.41026334147
Alpha = 0.1, Beta = 0.2: 79762.15904736413
Alpha = 0.1, Beta = 0.3: 72651.71157663976
Alpha = 0.1, Beta = 0.4: 65563.93978563353
Alpha = 0.1, Beta = 0.5: 60080.80506165465
Alpha = 0.2, Beta = 0.1: 74955.81441968396
Alpha = 0.2, Beta = 0.2: 68049.61163208084
Alpha = 0.2, Beta = 0.3: 63775.75468514865
Alpha = 0.2, Beta = 0.4: 63181.701672043535
Alpha = 0.2, Beta = 0.5: 65114.48917954008
Alpha = 0.3, Beta = 0.1: 71565.8111119523
Alpha = 0.3, Beta = 0.2: 67187.04519996174
Alpha = 0.3, Beta = 0.3: 66718.12340914301
Alpha = 0.3, Beta = 0.4: 68702.4299625298
Alpha = 0.3, Beta = 0.5: 71339.67373975222
```

```
Alpha = 0.4, Beta = 0.1: 70425.57332643589
Alpha = 0.4, Beta = 0.2: 67772.56168423429
Alpha = 0.4, Beta = 0.3: 68200.64358517397
Alpha = 0.4, Beta = 0.4: 69505.2372099851
Alpha = 0.4, Beta = 0.5: 70125.99421491066
Alpha = 0.5, Beta = 0.1: 69795.4402592008
Alpha = 0.5, Beta = 0.2: 67730.40278113387
Alpha = 0.5, Beta = 0.3: 67690.65523631648
Alpha = 0.5, Beta = 0.4: 67523.1845495206
Alpha = 0.5, Beta = 0.5: 66198.27701644872
```



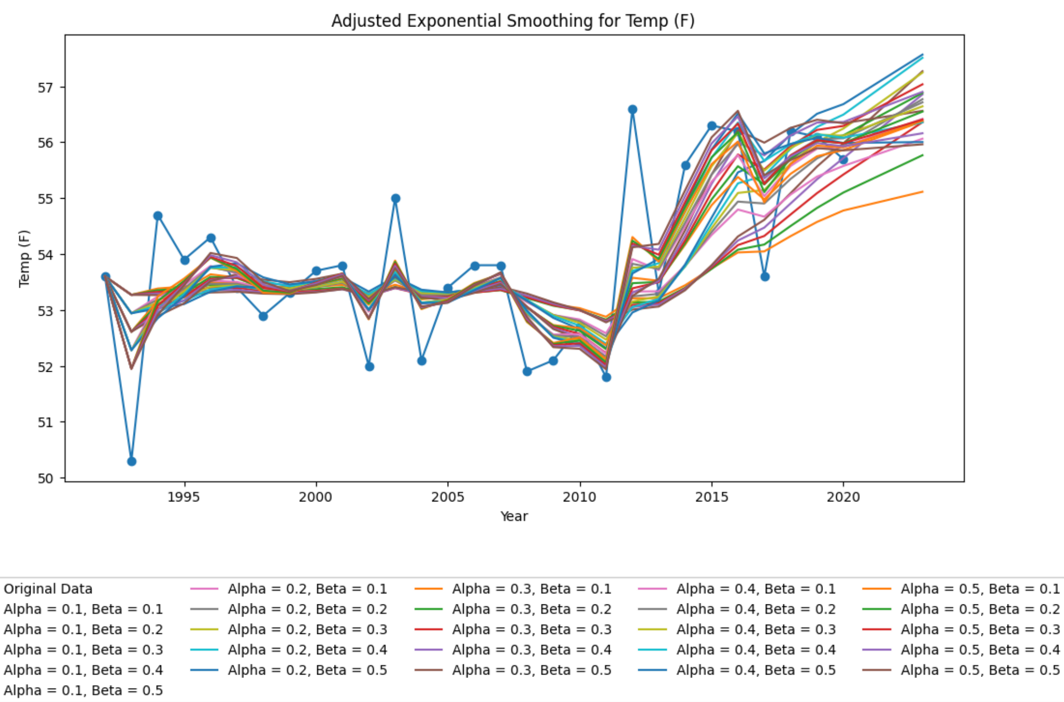

```
Predicted values for avg_fire_size with different alphas and betas for 2023:
Alpha = 0.1, Beta = 0.1: 115.95286354065642
Alpha = 0.1, Beta = 0.2: 116.90398285175304
Alpha = 0.1, Beta = 0.3: 109.44770954711821
Alpha = 0.1, Beta = 0.4: 102.36556572714268
Alpha = 0.1, Beta = 0.5: 99.14068042896395
Alpha = 0.2, Beta = 0.1: 115.84322495835717
Alpha = 0.2, Beta = 0.2: 113.46745650860139
Alpha = 0.2, Beta = 0.3: 114.84615840772273
Alpha = 0.2, Beta = 0.4: 121.27804499079105
Alpha = 0.2, Beta = 0.5: 129.50186093051383
Alpha = 0.3, Beta = 0.1: 119.3315632311921
Alpha = 0.3, Beta = 0.2: 121.63731297607796
Alpha = 0.3, Beta = 0.3: 127.50192013625788
```

```
Alpha = 0.3, Beta = 0.4: 133.77825277814222
Alpha = 0.3, Beta = 0.5: 137.72410636785264
Alpha = 0.4, Beta = 0.1: 123.98107887101547
Alpha = 0.4, Beta = 0.2: 128.37820634313283
Alpha = 0.4, Beta = 0.3: 133.923691349116
Alpha = 0.4, Beta = 0.4: 137.85754263047141
Alpha = 0.4, Beta = 0.5: 139.56557822201108
Alpha = 0.5, Beta = 0.1: 128.67308885721792
Alpha = 0.5, Beta = 0.2: 133.935795811512
Alpha = 0.5, Beta = 0.3: 139.1284374767425
Alpha = 0.5, Beta = 0.4: 142.8260548762863
Alpha = 0.5, Beta = 0.5: 145.57951821057736
```



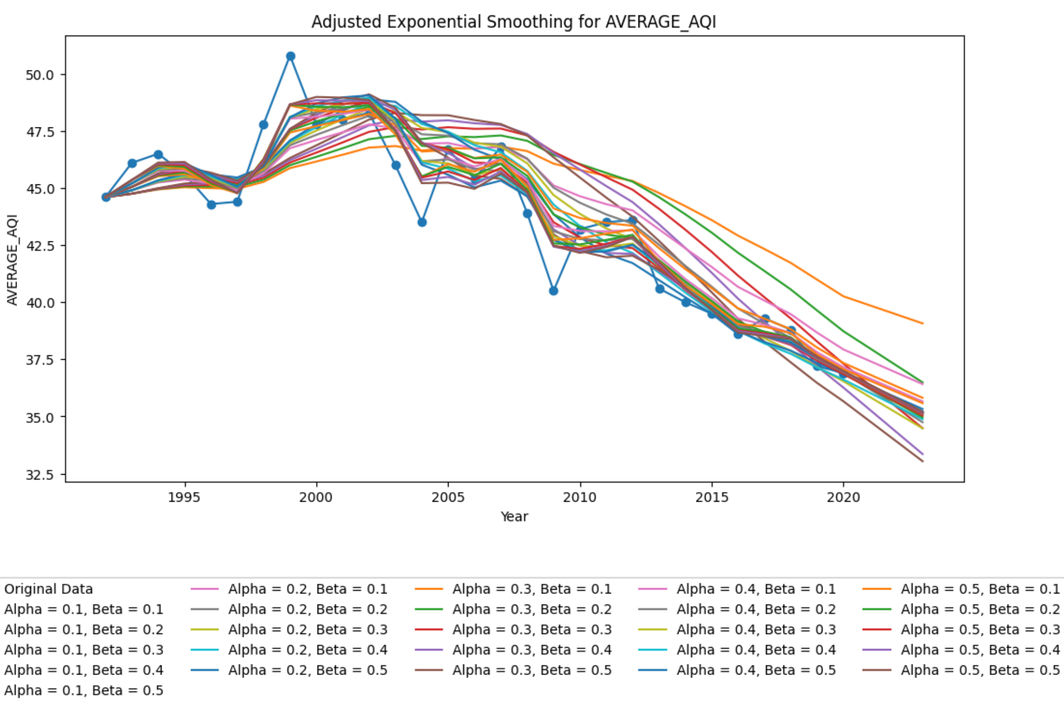

```
Predicted values for AVERAGE_PALMER_INDEX with different alphas and betas for
2023:
Alpha = 0.1, Beta = 0.1: 0.21784629181885665
Alpha = 0.1, Beta = 0.2: 0.49788948584365006
Alpha = 0.1, Beta = 0.3: 1.316622747610437
Alpha = 0.1, Beta = 0.4: 2.3131140785560795
Alpha = 0.1, Beta = 0.5: 3.166461163388019
Alpha = 0.2, Beta = 0.1: 1.378748826665374
Alpha = 0.2, Beta = 0.2: 2.2624539730906537
Alpha = 0.2, Beta = 0.3: 2.8675754227698658
Alpha = 0.2, Beta = 0.4: 2.9321757818067007
Alpha = 0.2, Beta = 0.5: 2.708510923813656
```

```
Alpha = 0.3, Beta = 0.1: 2.0746374356263666
Alpha = 0.3, Beta = 0.2: 2.785243594613153
Alpha = 0.3, Beta = 0.3: 3.1055553676762644
Alpha = 0.3, Beta = 0.4: 3.3295228488445994
Alpha = 0.3, Beta = 0.5: 3.760548896768127
Alpha = 0.4, Beta = 0.1: 2.4893815571553515
Alpha = 0.4, Beta = 0.2: 3.1416223333238236
Alpha = 0.4, Beta = 0.3: 3.574376461667469
Alpha = 0.4, Beta = 0.4: 4.0357404261717775
Alpha = 0.4, Beta = 0.5: 4.528608750070693
Alpha = 0.5, Beta = 0.1: 2.7596472288982796
Alpha = 0.5, Beta = 0.2: 3.40978185252443
Alpha = 0.5, Beta = 0.3: 3.895276790564624
Alpha = 0.5, Beta = 0.4: 4.338651161859319
Alpha = 0.5, Beta = 0.5: 4.698269948223878
```




```
Predicted values for Temp (F) with different alphas and betas for 2023:
Alpha = 0.1, Beta = 0.1: 55.11479702279913
Alpha = 0.1, Beta = 0.2: 55.76805879455259
Alpha = 0.1, Beta = 0.3: 56.359294128842926
Alpha = 0.1, Beta = 0.4: 56.85557086004925
Alpha = 0.1, Beta = 0.5: 57.27129683642844
Alpha = 0.2, Beta = 0.1: 56.06311410668438
Alpha = 0.2, Beta = 0.2: 56.77081021534277
Alpha = 0.2, Beta = 0.3: 57.24698013705663
```

```
Alpha = 0.2, Beta = 0.4: 57.511692937002785
Alpha = 0.2, Beta = 0.5: 57.56847759221083
Alpha = 0.3, Beta = 0.1: 56.374010297169015
Alpha = 0.3, Beta = 0.2: 56.88216663024885
Alpha = 0.3, Beta = 0.3: 57.03692691554167
Alpha = 0.3, Beta = 0.4: 56.90126591446707
Alpha = 0.3, Beta = 0.5: 56.56206186294368
Alpha = 0.4, Beta = 0.1: 56.41805314130589
Alpha = 0.4, Beta = 0.2: 56.71477173780192
Alpha = 0.4, Beta = 0.3: 56.64284915267621
Alpha = 0.4, Beta = 0.4: 56.35560085648354
Alpha = 0.4, Beta = 0.5: 56.003792941929376
Alpha = 0.5, Beta = 0.1: 56.384490382276624
Alpha = 0.5, Beta = 0.2: 56.54889683695227
Alpha = 0.5, Beta = 0.3: 56.40630417736612
Alpha = 0.5, Beta = 0.4: 56.16116569246934
Alpha = 0.5, Beta = 0.5: 55.96324955080124
```




```
Predicted values for AVERAGE_AQI with different alphas and betas for 2023:
Alpha = 0.1, Beta = 0.1: 39.07456736365714
Alpha = 0.1, Beta = 0.2: 36.49048086776565
Alpha = 0.1, Beta = 0.3: 34.485782214827104
Alpha = 0.1, Beta = 0.4: 33.35467087295799
Alpha = 0.1, Beta = 0.5: 33.04121260023639
Alpha = 0.2, Beta = 0.1: 36.41682679335022
```

```
Alpha = 0.2, Beta = 0.2: 34.74825433475162
Alpha = 0.2, Beta = 0.3: 34.48027334382126
Alpha = 0.2, Beta = 0.4: 34.88078794503642
Alpha = 0.2, Beta = 0.5: 35.3267652931249
Alpha = 0.3, Beta = 0.1: 35.81627115287876
Alpha = 0.3, Beta = 0.2: 34.9639863491969
Alpha = 0.3, Beta = 0.3: 35.03724374934936
Alpha = 0.3, Beta = 0.4: 35.19962837658302
Alpha = 0.3, Beta = 0.5: 35.208506603142965
Alpha = 0.4, Beta = 0.1: 35.6560805612026
Alpha = 0.4, Beta = 0.2: 35.097888840552855
Alpha = 0.4, Beta = 0.3: 35.126084804993084
Alpha = 0.4, Beta = 0.4: 35.17577353524874
Alpha = 0.4, Beta = 0.5: 35.2080142795299
Alpha = 0.5, Beta = 0.1: 35.579808626563846
Alpha = 0.5, Beta = 0.2: 35.13256288305871
Alpha = 0.5, Beta = 0.3: 35.12857697179536
Alpha = 0.5, Beta = 0.4: 35.163233497993666
Alpha = 0.5, Beta = 0.5: 35.200781066708956
Accuracy metrics for num_fires for 2023:
Alpha = 0.1, Beta = 0.1:
MAPE: 16.33%
MSE: 171913183.40
RMSE: 13111.57
----------------------------------------
Alpha = 0.1, Beta = 0.2:
MAPE: 12.40%
MSE: 113490620.31
RMSE: 10653.20
----------------------------------------
Alpha = 0.1, Beta = 0.3:
MAPE: 7.76%
MSE: 47553575.03
RMSE: 6895.91
----------------------------------------
Alpha = 0.1, Beta = 0.4:
MAPE: 9.93%
MSE: 55932067.94
RMSE: 7478.77
----------------------------------------
Alpha = 0.1, Beta = 0.5:
MAPE: 11.68%
MSE: 120684842.71
RMSE: 10985.67
----------------------------------------
Alpha = 0.2, Beta = 0.1:
MAPE: 9.01%
MSE: 56309103.45
```

```
RMSE: 7503.94
----------------------------------------
Alpha = 0.2, Beta = 0.2:
MAPE: 9.49%
MSE: 51028486.81
RMSE: 7143.42
----------------------------------------
Alpha = 0.2, Beta = 0.3:
MAPE: 10.83%
MSE: 86591596.22
RMSE: 9305.46
----------------------------------------
Alpha = 0.2, Beta = 0.4:
MAPE: 11.14%
MSE: 100674454.90
RMSE: 10033.67
----------------------------------------
Alpha = 0.2, Beta = 0.5:
MAPE: 10.69%
MSE: 86126040.73
RMSE: 9280.41
----------------------------------------
Alpha = 0.3, Beta = 0.1:
MAPE: 8.58%
MSE: 46815178.45
RMSE: 6842.16
----------------------------------------
Alpha = 0.3, Beta = 0.2:
MAPE: 9.86%
MSE: 59094344.34
RMSE: 7687.28
----------------------------------------
Alpha = 0.3, Beta = 0.3:
MAPE: 10.05%
MSE: 64330784.15
RMSE: 8020.65
----------------------------------------
Alpha = 0.3, Beta = 0.4:
MAPE: 9.53%
MSE: 55349802.87
RMSE: 7439.74
----------------------------------------
Alpha = 0.3, Beta = 0.5:
MAPE: 8.78%
MSE: 49307539.13
RMSE: 7021.93
----------------------------------------
Alpha = 0.4, Beta = 0.1:
```

```
MAPE: 8.96%
MSE: 48525802.48
RMSE: 6966.05
----------------------------------------
Alpha = 0.4, Beta = 0.2:
MAPE: 9.71%
MSE: 56629980.25
RMSE: 7525.29
----------------------------------------
Alpha = 0.4, Beta = 0.3:
MAPE: 9.60%
MSE: 54983006.15
RMSE: 7415.05
----------------------------------------
Alpha = 0.4, Beta = 0.4:
MAPE: 9.21%
MSE: 49967193.99
RMSE: 7068.75
----------------------------------------
Alpha = 0.4, Beta = 0.5:
MAPE: 8.97%
MSE: 47282898.12
RMSE: 6876.26
----------------------------------------
Alpha = 0.5, Beta = 0.1:
MAPE: 9.18%
MSE: 50537074.08
RMSE: 7108.94
----------------------------------------
Alpha = 0.5, Beta = 0.2:
MAPE: 9.73%
MSE: 57103521.02
RMSE: 7556.69
----------------------------------------
Alpha = 0.5, Beta = 0.3:
MAPE: 9.72%
MSE: 56752087.66
RMSE: 7533.40
----------------------------------------
Alpha = 0.5, Beta = 0.4:
MAPE: 9.73%
MSE: 56208072.96
RMSE: 7497.20
----------------------------------------
Alpha = 0.5, Beta = 0.5:
MAPE: 10.03%
MSE: 61203994.86
RMSE: 7823.30
```

```
----------------------------------------
Accuracy metrics for avg_fire_size for 2023:
Alpha = 0.1, Beta = 0.1:
MAPE: 24.73%
MSE: 706.03
RMSE: 26.57
----------------------------------------
Alpha = 0.1, Beta = 0.2:
MAPE: 25.76%
MSE: 736.48
RMSE: 27.14
----------------------------------------
Alpha = 0.1, Beta = 0.3:
MAPE: 22.83%
MSE: 707.49
RMSE: 26.60
----------------------------------------
Alpha = 0.1, Beta = 0.4:
MAPE: 20.38%
MSE: 750.92
RMSE: 27.40
----------------------------------------
Alpha = 0.1, Beta = 0.5:
MAPE: 20.95%
MSE: 796.55
RMSE: 28.22
----------------------------------------
Alpha = 0.2, Beta = 0.1:
MAPE: 24.89%
MSE: 711.86
RMSE: 26.68
----------------------------------------
Alpha = 0.2, Beta = 0.2:
MAPE: 23.51%
MSE: 685.49
RMSE: 26.18
----------------------------------------
Alpha = 0.2, Beta = 0.3:
MAPE: 23.01%
MSE: 660.90
RMSE: 25.71
----------------------------------------
Alpha = 0.2, Beta = 0.4:
MAPE: 24.35%
MSE: 674.71
RMSE: 25.98
----------------------------------------
Alpha = 0.2, Beta = 0.5:
```

```
MAPE: 26.68%
MSE: 785.94
RMSE: 28.03
----------------------------------------
Alpha = 0.3, Beta = 0.1:
MAPE: 26.39%
MSE: 752.15
RMSE: 27.43
----------------------------------------
Alpha = 0.3, Beta = 0.2:
MAPE: 26.48%
MSE: 752.75
RMSE: 27.44
----------------------------------------
Alpha = 0.3, Beta = 0.3:
MAPE: 27.75%
MSE: 821.52
RMSE: 28.66
----------------------------------------
Alpha = 0.3, Beta = 0.4:
MAPE: 29.46%
MSE: 956.56
RMSE: 30.93
----------------------------------------
Alpha = 0.3, Beta = 0.5:
MAPE: 30.71%
MSE: 1077.13
RMSE: 32.82
----------------------------------------
Alpha = 0.4, Beta = 0.1:
MAPE: 28.43%
MSE: 839.32
RMSE: 28.97
----------------------------------------
Alpha = 0.4, Beta = 0.2:
MAPE: 29.14%
MSE: 893.11
RMSE: 29.88
----------------------------------------
Alpha = 0.4, Beta = 0.3:
MAPE: 30.31%
MSE: 1004.36
RMSE: 31.69
----------------------------------------
Alpha = 0.4, Beta = 0.4:
MAPE: 31.18%
MSE: 1106.21
RMSE: 33.26
```

```
----------------------------------------
Alpha = 0.4, Beta = 0.5:
MAPE: 31.43%
MSE: 1148.68
RMSE: 33.89
----------------------------------------
Alpha = 0.5, Beta = 0.1:
MAPE: 30.49%
MSE: 965.19
RMSE: 31.07
----------------------------------------
Alpha = 0.5, Beta = 0.2:
MAPE: 31.34%
MSE: 1062.61
RMSE: 32.60
----------------------------------------
Alpha = 0.5, Beta = 0.3:
MAPE: 32.24%
MSE: 1190.51
RMSE: 34.50
----------------------------------------
Alpha = 0.5, Beta = 0.4:
MAPE: 32.77%
MSE: 1291.93
RMSE: 35.94
----------------------------------------
Alpha = 0.5, Beta = 0.5:
MAPE: 34.07%
MSE: 1365.13
RMSE: 36.95
----------------------------------------
Accuracy metrics for AVERAGE_PALMER_INDEX for 2023:
Alpha = 0.1, Beta = 0.1:
MAPE: 119.90%
MSE: 7.72
RMSE: 2.78
----------------------------------------
Alpha = 0.1, Beta = 0.2:
MAPE: 118.45%
MSE: 6.99
RMSE: 2.64
----------------------------------------
Alpha = 0.1, Beta = 0.3:
MAPE: 163.15%
MSE: 4.96
RMSE: 2.23
----------------------------------------
Alpha = 0.1, Beta = 0.4:
```

```
MAPE: 249.60%
MSE: 3.70
RMSE: 1.92
----------------------------------------
Alpha = 0.1, Beta = 0.5:
MAPE: 351.82%
MSE: 3.87
RMSE: 1.97
----------------------------------------
Alpha = 0.2, Beta = 0.1:
MAPE: 221.97%
MSE: 4.69
RMSE: 2.16
----------------------------------------
Alpha = 0.2, Beta = 0.2:
MAPE: 287.06%
MSE: 3.82
RMSE: 1.95
----------------------------------------
Alpha = 0.2, Beta = 0.3:
MAPE: 351.40%
MSE: 3.87
RMSE: 1.97
----------------------------------------
Alpha = 0.2, Beta = 0.4:
MAPE: 362.34%
MSE: 3.94
RMSE: 1.98
----------------------------------------
Alpha = 0.2, Beta = 0.5:
MAPE: 335.31%
MSE: 3.81
RMSE: 1.95
----------------------------------------
Alpha = 0.3, Beta = 0.1:
MAPE: 294.48%
MSE: 3.98
RMSE: 1.99
----------------------------------------
Alpha = 0.3, Beta = 0.2:
MAPE: 355.45%
MSE: 3.91
RMSE: 1.98
----------------------------------------
Alpha = 0.3, Beta = 0.3:
MAPE: 380.59%
MSE: 4.08
RMSE: 2.02
```

```
----------------------------------------
Alpha = 0.3, Beta = 0.4:
MAPE: 391.36%
MSE: 4.20
RMSE: 2.05
----------------------------------------
Alpha = 0.3, Beta = 0.5:
MAPE: 418.13%
MSE: 4.64
RMSE: 2.15
----------------------------------------
Alpha = 0.4, Beta = 0.1:
MAPE: 347.95%
MSE: 3.97
RMSE: 1.99
----------------------------------------
Alpha = 0.4, Beta = 0.2:
MAPE: 398.96%
MSE: 4.24
RMSE: 2.06
----------------------------------------
Alpha = 0.4, Beta = 0.3:
MAPE: 430.74%
MSE: 4.67
RMSE: 2.16
----------------------------------------
Alpha = 0.4, Beta = 0.4:
MAPE: 464.79%
MSE: 5.38
RMSE: 2.32
----------------------------------------
Alpha = 0.4, Beta = 0.5:
MAPE: 504.47%
MSE: 6.46
RMSE: 2.54
----------------------------------------
Alpha = 0.5, Beta = 0.1:
MAPE: 382.93%
MSE: 4.12
RMSE: 2.03
----------------------------------------
Alpha = 0.5, Beta = 0.2:
MAPE: 431.75%
MSE: 4.62
RMSE: 2.15
----------------------------------------
Alpha = 0.5, Beta = 0.3:
MAPE: 468.25%
```

```
MSE: 5.31
RMSE: 2.31
----------------------------------------
Alpha = 0.5, Beta = 0.4:
MAPE: 502.86%
MSE: 6.21
RMSE: 2.49
----------------------------------------
Alpha = 0.5, Beta = 0.5:
MAPE: 532.77%
MSE: 7.14
RMSE: 2.67
----------------------------------------
Accuracy metrics for Temp (F) for 2023:
Alpha = 0.1, Beta = 0.1:
MAPE: 1.78%
MSE: 1.09
RMSE: 1.04
----------------------------------------
Alpha = 0.1, Beta = 0.2:
MAPE: 0.89%
MSE: 0.36
RMSE: 0.60
----------------------------------------
Alpha = 0.1, Beta = 0.3:
MAPE: 0.70%
MSE: 0.22
RMSE: 0.47
----------------------------------------
Alpha = 0.1, Beta = 0.4:
MAPE: 0.98%
MSE: 0.50
RMSE: 0.70
----------------------------------------
Alpha = 0.1, Beta = 0.5:
MAPE: 1.51%
MSE: 1.02
RMSE: 1.01
----------------------------------------
Alpha = 0.2, Beta = 0.1:
MAPE: 0.61%
MSE: 0.13
RMSE: 0.36
----------------------------------------
Alpha = 0.2, Beta = 0.2:
MAPE: 0.88%
MSE: 0.43
RMSE: 0.66
```

```
----------------------------------------
Alpha = 0.2, Beta = 0.3:
MAPE: 1.63%
MSE: 1.07
RMSE: 1.03
----------------------------------------
Alpha = 0.2, Beta = 0.4:
MAPE: 2.10%
MSE: 1.61
RMSE: 1.27
----------------------------------------
Alpha = 0.2, Beta = 0.5:
MAPE: 2.28%
MSE: 1.82
RMSE: 1.35
----------------------------------------
Alpha = 0.3, Beta = 0.1:
MAPE: 0.56%
MSE: 0.16
RMSE: 0.41
----------------------------------------
Alpha = 0.3, Beta = 0.2:
MAPE: 1.13%
MSE: 0.57
RMSE: 0.75
----------------------------------------
Alpha = 0.3, Beta = 0.3:
MAPE: 1.41%
MSE: 0.79
RMSE: 0.89
----------------------------------------
Alpha = 0.3, Beta = 0.4:
MAPE: 1.29%
MSE: 0.65
RMSE: 0.81
----------------------------------------
Alpha = 0.3, Beta = 0.5:
MAPE: 0.87%
MSE: 0.31
RMSE: 0.56
----------------------------------------
Alpha = 0.4, Beta = 0.1:
MAPE: 0.59%
MSE: 0.18
RMSE: 0.43
----------------------------------------
Alpha = 0.4, Beta = 0.2:
MAPE: 0.90%
```

```
MSE: 0.40
RMSE: 0.63
----------------------------------------
Alpha = 0.4, Beta = 0.3:
MAPE: 0.84%
MSE: 0.34
RMSE: 0.59
----------------------------------------
Alpha = 0.4, Beta = 0.4:
MAPE: 0.51%
MSE: 0.15
RMSE: 0.39
----------------------------------------
Alpha = 0.4, Beta = 0.5:
MAPE: 0.36%
MSE: 0.05
RMSE: 0.22
----------------------------------------
Alpha = 0.5, Beta = 0.1:
MAPE: 0.56%
MSE: 0.17
RMSE: 0.41
----------------------------------------
Alpha = 0.5, Beta = 0.2:
MAPE: 0.68%
MSE: 0.26
RMSE: 0.51
----------------------------------------
Alpha = 0.5, Beta = 0.3:
MAPE: 0.57%
MSE: 0.18
RMSE: 0.42
----------------------------------------
Alpha = 0.5, Beta = 0.4:
MAPE: 0.40%
MSE: 0.08
RMSE: 0.29
----------------------------------------
Alpha = 0.5, Beta = 0.5:
MAPE: 0.44%
MSE: 0.06
RMSE: 0.25
----------------------------------------
Accuracy metrics for AVERAGE_AQI for 2023:
Alpha = 0.1, Beta = 0.1:
MAPE: 4.91%
MSE: 3.67
RMSE: 1.92
```

```
----------------------------------------
Alpha = 0.1, Beta = 0.2:
MAPE: 1.10%
MSE: 0.28
RMSE: 0.53
----------------------------------------
Alpha = 0.1, Beta = 0.3:
MAPE: 5.86%
MSE: 4.97
RMSE: 2.23
----------------------------------------
Alpha = 0.1, Beta = 0.4:
MAPE: 8.79%
MSE: 11.05
RMSE: 3.32
----------------------------------------
Alpha = 0.1, Beta = 0.5:
MAPE: 9.88%
MSE: 13.93
RMSE: 3.73
----------------------------------------
Alpha = 0.2, Beta = 0.1:
MAPE: 1.87%
MSE: 0.73
RMSE: 0.86
----------------------------------------
Alpha = 0.2, Beta = 0.2:
MAPE: 5.74%
MSE: 4.82
RMSE: 2.19
----------------------------------------
Alpha = 0.2, Beta = 0.3:
MAPE: 6.54%
MSE: 6.23
RMSE: 2.50
----------------------------------------
Alpha = 0.2, Beta = 0.4:
MAPE: 5.76%
MSE: 4.92
RMSE: 2.22
----------------------------------------
Alpha = 0.2, Beta = 0.5:
MAPE: 4.76%
MSE: 3.47
RMSE: 1.86
----------------------------------------
Alpha = 0.3, Beta = 0.1:
MAPE: 3.46%
```

```
MSE: 1.94
RMSE: 1.39
----------------------------------------
Alpha = 0.3, Beta = 0.2:
MAPE: 5.37%
MSE: 4.28
RMSE: 2.07
----------------------------------------
Alpha = 0.3, Beta = 0.3:
MAPE: 5.27%
MSE: 4.14
RMSE: 2.03
----------------------------------------
Alpha = 0.3, Beta = 0.4:
MAPE: 4.91%
MSE: 3.63
RMSE: 1.91
----------------------------------------
Alpha = 0.3, Beta = 0.5:
MAPE: 4.87%
MSE: 3.57
RMSE: 1.89
----------------------------------------
Alpha = 0.4, Beta = 0.1:
MAPE: 3.90%
MSE: 2.41
RMSE: 1.55
----------------------------------------
Alpha = 0.4, Beta = 0.2:
MAPE: 5.09%
MSE: 3.87
RMSE: 1.97
----------------------------------------
Alpha = 0.4, Beta = 0.3:
MAPE: 5.05%
MSE: 3.81
RMSE: 1.95
----------------------------------------
Alpha = 0.4, Beta = 0.4:
MAPE: 4.94%
MSE: 3.66
RMSE: 1.91
----------------------------------------
Alpha = 0.4, Beta = 0.5:
MAPE: 4.87%
MSE: 3.58
RMSE: 1.89
----------------------------------------
```

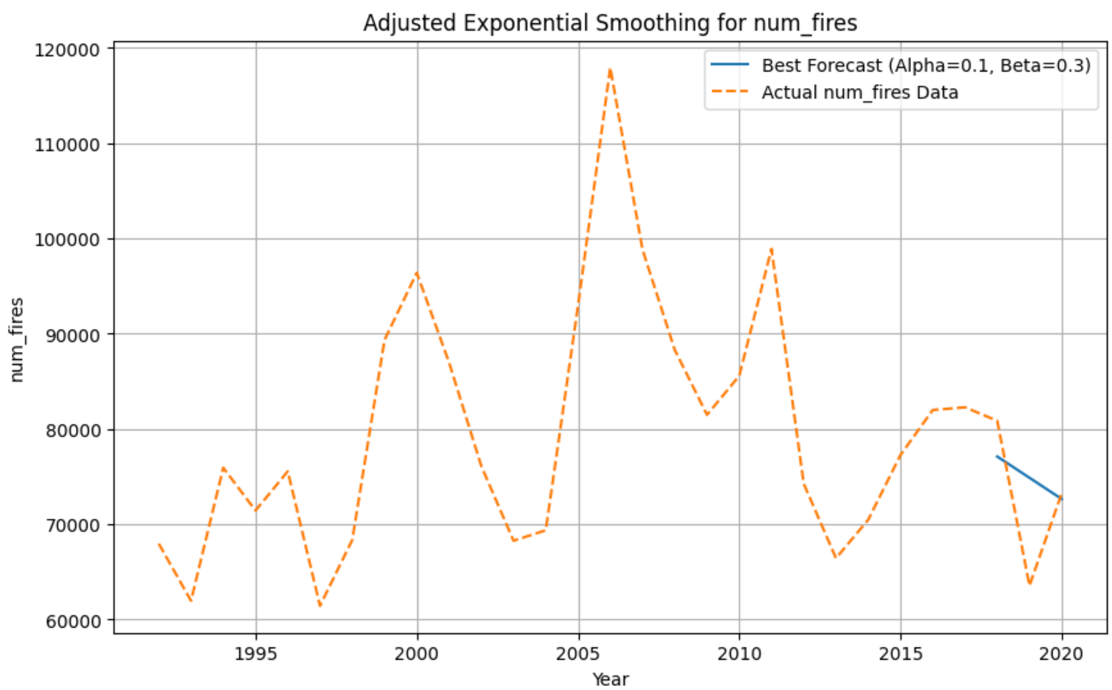
```
Alpha = 0.5, Beta = 0.1:
MAPE: 4.12%
MSE: 2.66
RMSE: 1.63
----------------------------------------
Alpha = 0.5, Beta = 0.2:
MAPE: 5.03%
MSE: 3.78
RMSE: 1.94
----------------------------------------
Alpha = 0.5, Beta = 0.3:
MAPE: 5.03%
MSE: 3.79
RMSE: 1.95
----------------------------------------
Alpha = 0.5, Beta = 0.4:
MAPE: 4.95%
MSE: 3.68
RMSE: 1.92
----------------------------------------
Alpha = 0.5, Beta = 0.5:
MAPE: 4.86%
MSE: 3.56
RMSE: 1.89
----------------------------------------
```

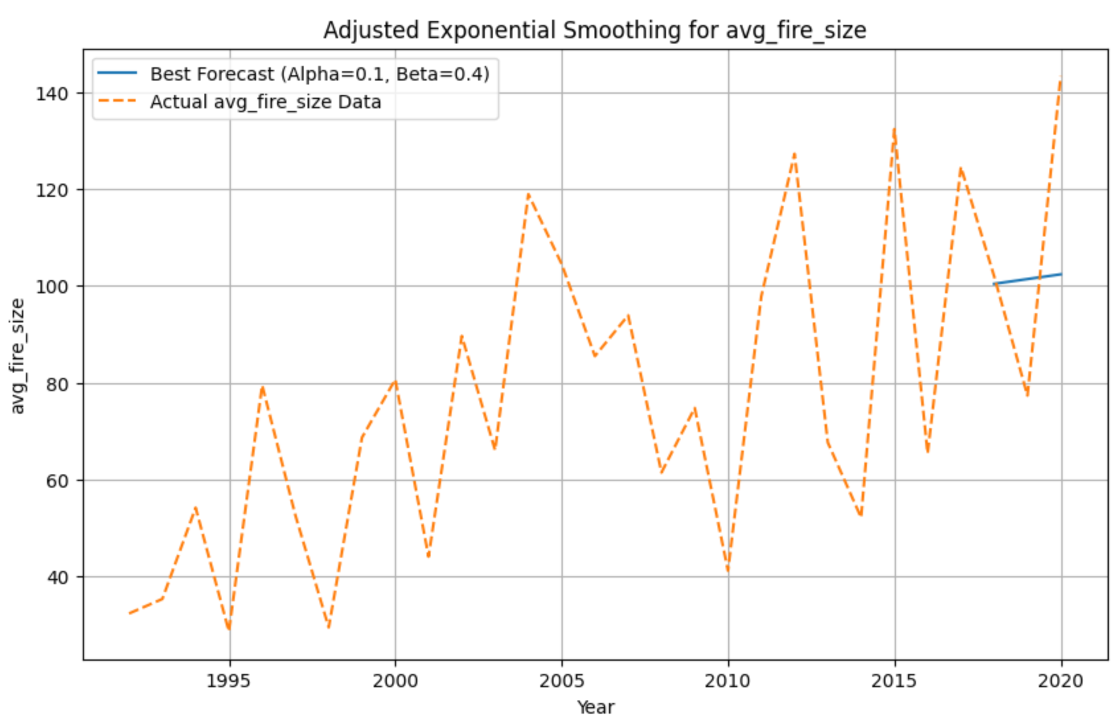
[47]:

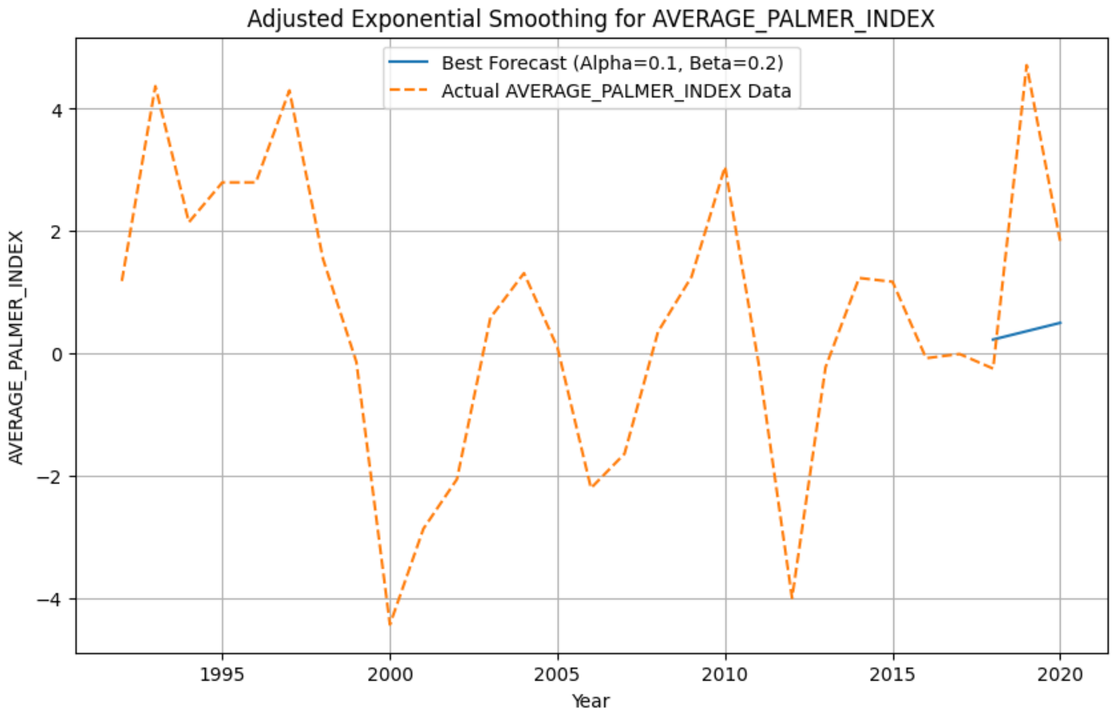
```
import sqlite3
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Define the Adjusted Exponential Smoothing function
def adjusted_exponential_smoothing(data, alpha, beta, forecast_years):
    forecast = [data.iloc[0]]  # Initial forecast is the first data point
    trend = [0]  # Initial trend is 0
    for i in range(1, len(data) + forecast_years):
        if i < len(data):
            forecast.append(alpha * data.iloc[i] + (1 - alpha) * (forecast[i -
 ↪1] + trend[i - 1]))
            trend.append(beta * (forecast[i] - forecast[i - 1]) + (1 - beta) *
 ↪trend[i - 1])
        else:
            forecast.append(forecast[-1] + trend[-1])
    return forecast

# Connect to the SQLite database
```

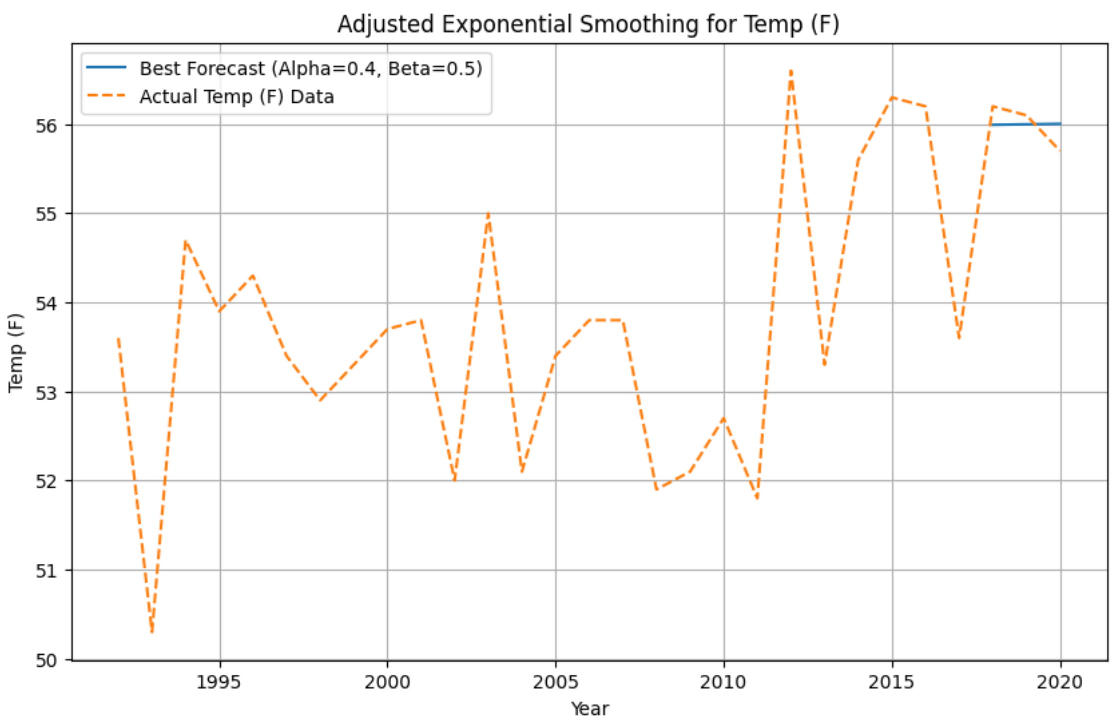
```
conn = sqlite3.connect('.database')  # Replace 'your_database.db' with the␣
↪actual database filename

# Define a list of alpha and beta values to try
alpha_values = [0.1, 0.2, 0.3, 0.4, 0.5]
beta_values = [0.1, 0.2, 0.3, 0.4, 0.5]

# Create a dictionary to store the forecasts for each column, alpha, and beta␣
↪value
forecasts_aes = {column: {alpha: {} for alpha in alpha_values} for column in␣
↪data.columns[1:]}

# Create dictionaries to store MAPE and MAD values for each combination of␣
↪alpha and beta
mape_values = {column: {alpha: {} for alpha in alpha_values} for column in data.
↪columns[1:]}
mad_values = {column: {alpha: {} for alpha in alpha_values} for column in data.
↪columns[1:]}

# Load data from the SQLite database
data = pd.read_sql_query("SELECT * FROM Data", conn)

# Forecasting parameters
forecast_years = 2023 - data['year'].max()  # Forecast to the year 2023

# Iterate through each column, alpha, and beta, and calculate forecasts
for column in data.columns[1:]:
    for alpha in alpha_values:
        for beta in beta_values:
            forecasts_aes[column][alpha][beta] =␣
↪adjusted_exponential_smoothing(data[column], alpha, beta, forecast_years)
            forecasted_values =␣
↪forecasts_aes[column][alpha][beta][-forecast_years:]
            actual_values = data[column].tail(forecast_years).values
            mape = np.mean(np.abs((actual_values - forecasted_values) /␣
↪actual_values)) * 100
            mad = np.mean(np.abs(actual_values - forecasted_values))
            mape_values[column][alpha][beta] = mape
            mad_values[column][alpha][beta] = mad

# Find the alpha and beta values that give the most accurate forecast based on␣
↪MAPE
best_alpha_beta = {}
for column in data.columns[1:]:
    min_mape = float('inf')
    for alpha, beta_dict in mape_values[column].items():
```

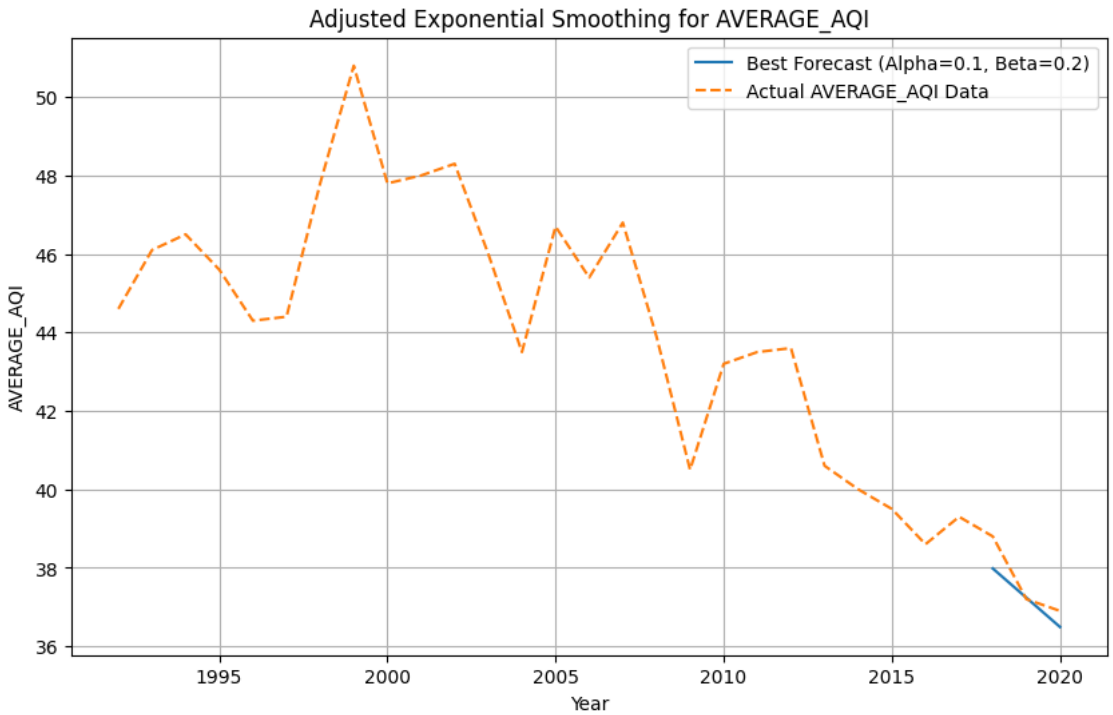
```python
        for beta, mape in beta_dict.items():
            if mape < min_mape:
                min_mape = mape
                best_alpha_beta[column] = (alpha, beta)

# Plot the best forecasts for each column based on the best alpha and beta␣
 ↪values
for column, (best_alpha, best_beta) in best_alpha_beta.items():
    best_forecast = forecasts_aes[column][best_alpha][best_beta]
    years = data['year'].tail(forecast_years).values
    plt.figure(figsize=(10, 6))
    plt.plot(years, best_forecast[-forecast_years:], label=f'Best Forecast␣
 ↪(Alpha={best_alpha}, Beta={best_beta})')
    plt.plot(data['year'], data[column], label=f'Actual {column} Data',␣
 ↪linestyle='--')
    plt.title(f'Adjusted Exponential Smoothing for {column}')
    plt.xlabel('Year')
    plt.ylabel(column)
    plt.legend()
    plt.grid(True)
    plt.show()

# Print the best alpha and beta values for each column based on MAPE
for column, (best_alpha, best_beta) in best_alpha_beta.items():
    print(f'Best Alpha and Beta for {column} based on MAPE:')
    print(f'Alpha: {best_alpha}, Beta: {best_beta}')
    print(f'MAPE: {mape_values[column][best_alpha][best_beta]:.2f}%')
    print(f'MAD: {mad_values[column][best_alpha][best_beta]:.2f}')
    print('-' * 40)

# Close the database connection
conn.close()
```


Adjusted Exponential Smoothing for num_fires
Best Forecast (Alpha=0.1, Beta=0.3)
Actual num_fires Data
num_fires
Year
120000
110000
100000
90000
80000
70000
60000
1995
2000
2005
2010
2015
2020


Adjusted Exponential Smoothing for avg_fire_size
Best Forecast (Alpha=0.1, Beta=0.4)
Actual avg_fire_size Data
avg_fire_size
Year
140
120
100
80
60
40
1995
2000
2005
2010
2015
2020


Adjusted Exponential Smoothing for AVERAGE_PALMER_INDEX
Best Forecast (Alpha=0.1, Beta=0.2)
Actual AVERAGE_PALMER_INDEX Data
AVERAGE_PALMER_INDEX
4
2
0
−2
−4
1995
2000
2005
2010
2015
2020
Year


Adjusted Exponential Smoothing for Temp (F)
Best Forecast (Alpha=0.4, Beta=0.5)
Actual Temp (F) Data
Temp (F)
56
55
54
53
52
51
50
1995
2000
2005
2010
2015
2020
Year



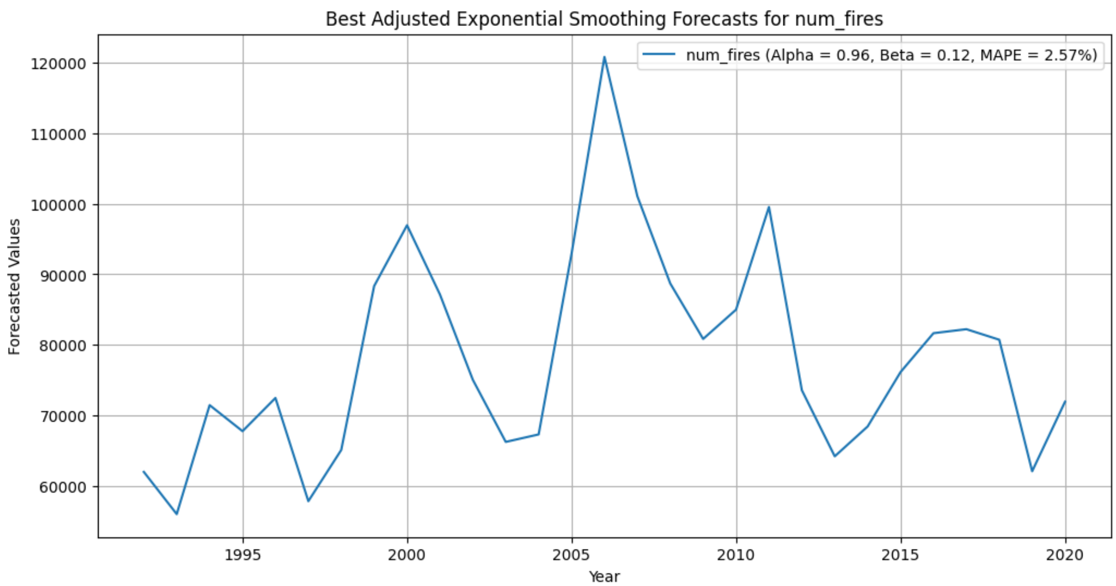

```
Best Alpha and Beta for num_fires based on MAPE:
Alpha: 0.1, Beta: 0.3
MAPE: 7.76%
MAD: 5228.22
----------------------------------------
Best Alpha and Beta for avg_fire_size based on MAPE:
Alpha: 0.1, Beta: 0.4
MAPE: 20.38%
MAD: 22.16
----------------------------------------
Best Alpha and Beta for AVERAGE_PALMER_INDEX based on MAPE:
Alpha: 0.1, Beta: 0.2
MAPE: 118.45%
MAD: 2.06
----------------------------------------
Best Alpha and Beta for Temp (F) based on MAPE:
Alpha: 0.4, Beta: 0.5
MAPE: 0.36%
MAD: 0.20
----------------------------------------
Best Alpha and Beta for AVERAGE_AQI based on MAPE:
Alpha: 0.1, Beta: 0.2
MAPE: 1.10%
MAD: 0.42
```

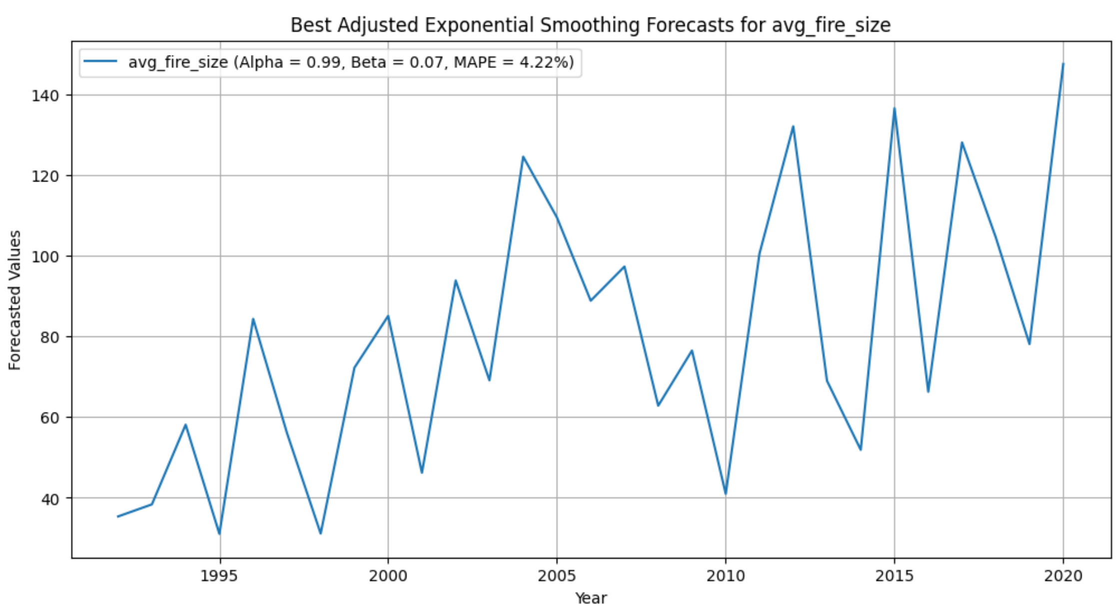
----

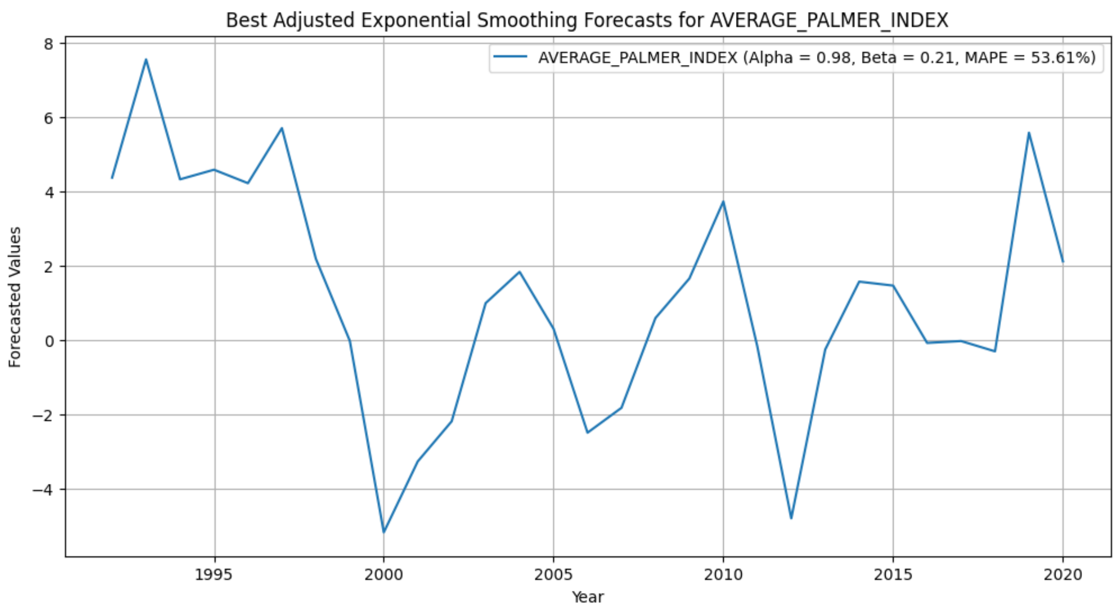
[48]:
```
import sqlite3
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

# Connect to the SQLite database
conn = sqlite3.connect('.database')  # Replace '.database' with your database␣
↪file path

# Query data from the database
query = "SELECT * FROM Data"  # Replace 'Data' with your table name
data = pd.read_sql(query, conn)

# Close the database connection
conn.close()

# Define the range of alpha and beta values you want to test
alpha_values = np.arange(0.01, 1.0, 0.01)
beta_values = np.arange(0.01, 1.0, 0.01)

# Dictionary to store forecasts for different combinations of alpha and beta
forecasts_aes = {}

# Iterate over columns (variables) in your data
for column in data.columns[1:]:
    forecasts_aes[column] = {}
    for alpha in alpha_values:
        forecasts_aes[column][alpha] = {}
        for beta in beta_values:
            # Calculate forecasts using adjusted exponential smoothing for the␣
↪current alpha and beta
            def adjusted_exponential_smoothing(series, alpha, beta):
                forecasts = []
                level, trend = series[0], series[1] - series[0]
                for i in range(len(series)):
                    if i == 0:
                        forecast = level + trend
                    else:
                        last_level, level = level, alpha * series[i] + (1 -␣
↪alpha) * (level + trend)
                        trend = beta * (level - last_level) + (1 - beta) * trend
                        forecast = level + trend
                    forecasts.append(forecast)
                return forecasts
```

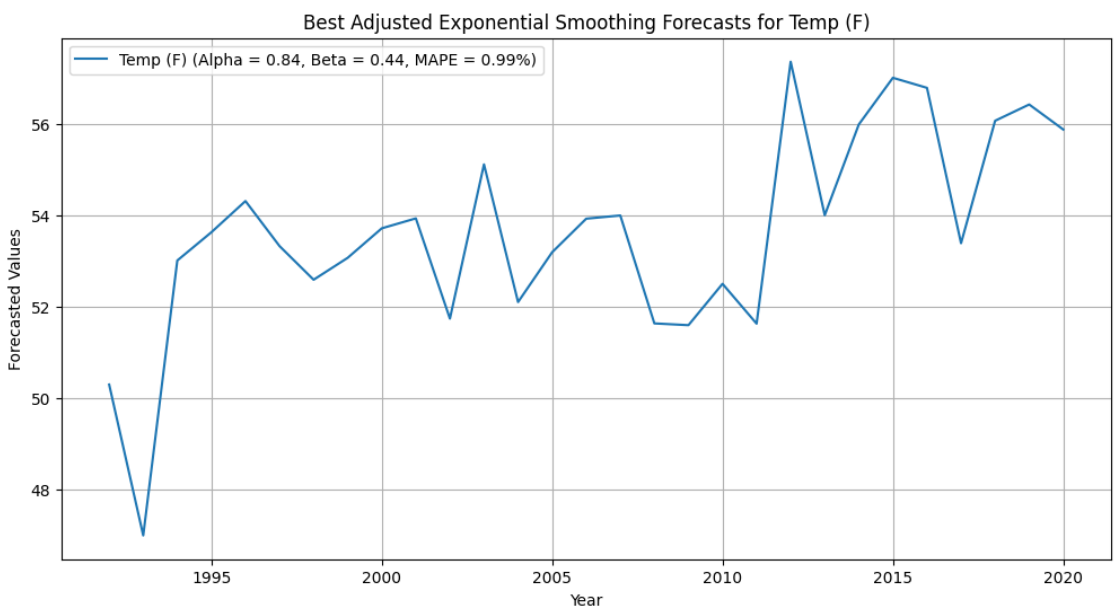
```python
            forecast_years = len(data)
            forecasts = adjusted_exponential_smoothing(data[column], alpha, beta)

            # Calculate MAPE for the current forecasts
            actual = data[column].values
            mape = np.mean(np.abs((actual - forecasts) / actual)) * 100

            # Store the MAPE and forecasts for the current alpha and beta
            forecasts_aes[column][alpha][beta] = {'MAPE': mape, 'Forecasts': forecasts}

# Find the combination of alpha and beta with the lowest MAPE for each column
best_alpha_beta = {}
for column in forecasts_aes:
    best_mape = float('inf')
    for alpha, beta_dict in forecasts_aes[column].items():
        for beta, result in beta_dict.items():
            mape = result['MAPE']
            if mape < best_mape:
                best_mape = mape
                best_alpha_beta[column] = {'Alpha': alpha, 'Beta': beta, 'MAPE': mape}

# Plot the best forecasts for each column on separate graphs
for column, params in best_alpha_beta.items():
    best_alpha = params['Alpha']
    best_beta = params['Beta']
    best_forecasts = forecasts_aes[column][best_alpha][best_beta]['Forecasts']

    plt.figure(figsize=(12, 6))
    years = range(data['year'].min(), data['year'].min() + len(best_forecasts)) # Adjusted range of years
    plt.plot(years, best_forecasts, label=f'{column} (Alpha = {best_alpha:.2f}, Beta = {best_beta:.2f}, MAPE = {params["MAPE"]:.2f}%)')

    # Print the forecast for 2023
    forecast_2023 = best_forecasts[-1]  # Get the last forecasted value
    print(f'Forecast for {column} in 2023: {forecast_2023:.2f}')

    plt.title(f'Best Adjusted Exponential Smoothing Forecasts for {column}')
    plt.xlabel('Year')
    plt.ylabel('Forecasted Values')
    plt.legend()
    plt.grid(True)

plt.show()
```

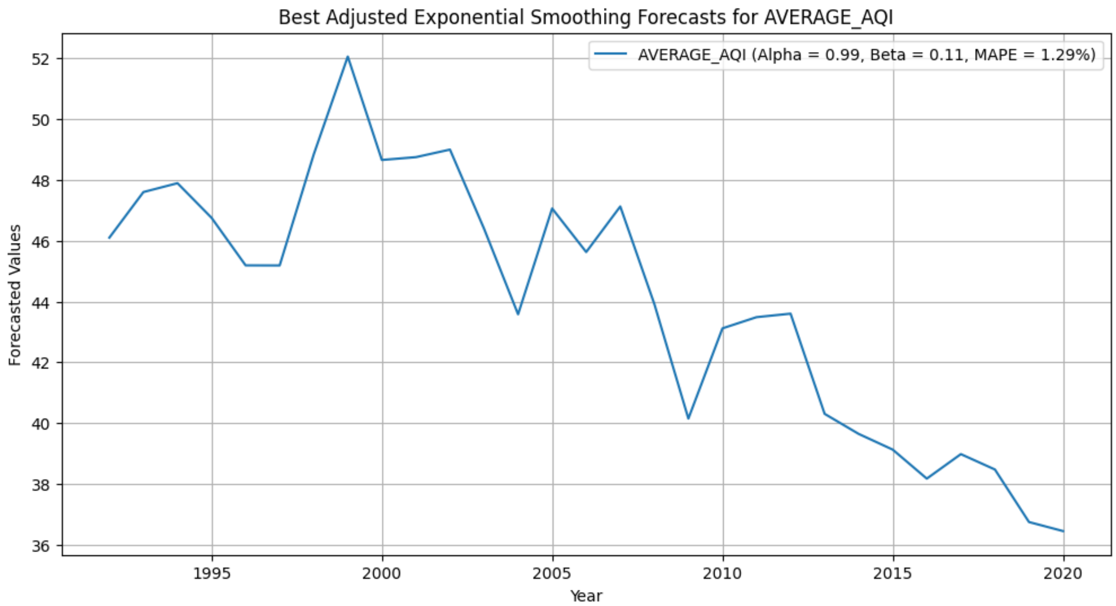
```
Forecast for num_fires in 2023: 71931.08
Forecast for avg_fire_size in 2023: 147.64
Forecast for AVERAGE_PALMER_INDEX in 2023: 2.12
Forecast for Temp (F) in 2023: 55.88
Forecast for AVERAGE_AQI in 2023: 36.45
```






Best Adjusted Exponential Smoothing Forecasts for AVERAGE_PALMER_INDEX
AVERAGE_PALMER_INDEX (Alpha = 0.98, Beta = 0.21, MAPE = 53.61%)
Forecasted Values
Year
8
6
4
2
0
−2
−4
1995
2000
2005
2010
2015
2020


Best Adjusted Exponential Smoothing Forecasts for Temp (F)
Temp (F) (Alpha = 0.84, Beta = 0.44, MAPE = 0.99%)
Forecasted Values
Year
56
54
52
50
48
1995
2000
2005
2010
2015
2020



1 Time Series Forecasting with Adjustable Exponential Smoothing (AES)

The code I've developed is a Python script designed for time series forecasting using Adjustable Exponential Smoothing (AES) on data fetched from an SQLite database. The primary goal of this code is to facilitate accurate predictions for various variables over time. Here's a detailed breakdown of what my code accomplishes:

1. **Database Connection**: The script establishes a connection with an SQLite database named `.database`. It accesses a specific table called `Data` to retrieve the necessary time series data. This data typically includes multiple columns representing different variables of interest, with a crucial 'year' column indicating the corresponding years.

2. **Adjustable Exponential Smoothing**: For time series forecasting, I've implemented an Adjustable Exponential Smoothing function called `adjusted_exponential_smoothing`. This function takes three crucial inputs: the time series data, an 'alpha' value (which controls the smoothing), and a 'beta' value (responsible for managing the trend component). The function uses these inputs to generate forecasts.

3. **Hyperparameter Tuning**: One standout feature of my code is its ability to perform hyperparameter tuning. It systematically explores various combinations of 'alpha' and 'beta' values, applying the AES method each time to calculate forecasts. To assess forecast accuracy, I use the Mean Absolute Percentage Error (MAPE), a popular metric.

4. **Model Selection**: After generating forecasts for all combinations of 'alpha' and 'beta,' I select the best model for each variable. The best model is determined by the combination of 'alpha' and 'beta' that yields the lowest MAPE. This ensures the use of the most accurate model for predicting future values of each variable.

5. **Visualization**: To visualize these forecasts effectively, I employ the `matplotlib` library. I create individual line plots for each variable, showcasing the best forecasts alongside their corresponding 'alpha,' 'beta,' and MAPE values. This visual representation makes it easy to interpret and compare forecasted trends over time.
6. **Forecast Output**: As a finishing touch, I've included the option to print out the forecasted values for the year 2023. This allows users to quickly access predictions for that specific year, aiding in planning and decision-making.

In summary, my code combines data retrieval, adjustable exponential smoothing, hyperparameter tuning, model selection, and visualization to provide a comprehensive solution for accurate time series forecasting from an SQLite database. It's a versatile tool that can assist in making informed predictions for various variables.