Healthcare Project - Time Series Analysis for Forecasting Revenue

We use time series analysis to forecast revenue for the next 12 months.

```
# Modules
import pandas as pd
import numpy as np
import statsmodels.api as sm
from statsmodels.tsa.holtwinters import ExponentialSmoothing
import matplotlib.pyplot as plt
```

```
# Load data
path = "C:/Users/rvrei/Documents/Healthcare_df.csv"
healthcare_df = pd.read_csv(path)
healthcare_df.head()
```

	claim_id	patient_id	procedure_id	claim_date	claim_amount	claim_status	insurance_provider	procedure_type	pr
0	CLM0001	PAT0001	41	2024-03-29	1997.79	approved	Blue Shield	CT Scan	
1	CLM0001	PAT0001	41	2024-03-29	1997.79	approved	Blue Shield	CT Scan	
2	CLM0016	PAT0016	41	2023-09-16	1080.34	approved	Aetna	MRI	
3	CLM0016	PAT0016	41	2023-09-16	1080.34	approved	Aetna	MRI	
4	CLM0004	PAT0004	14	2023-02-03	3073.56	approved	Blue Shield	Lab Test	

5 rows × 21 columns

```
# Aggregate revenue by month
healthcare_df['admission_date'] = pd.to_datetime(healthcare_df['admission_date']) # Convert to a datetime type
time_serie_revenue = healthcare_df.resample('M', on='admission_date').revenue.sum()
```

```
time_serie_revenue.head()
```

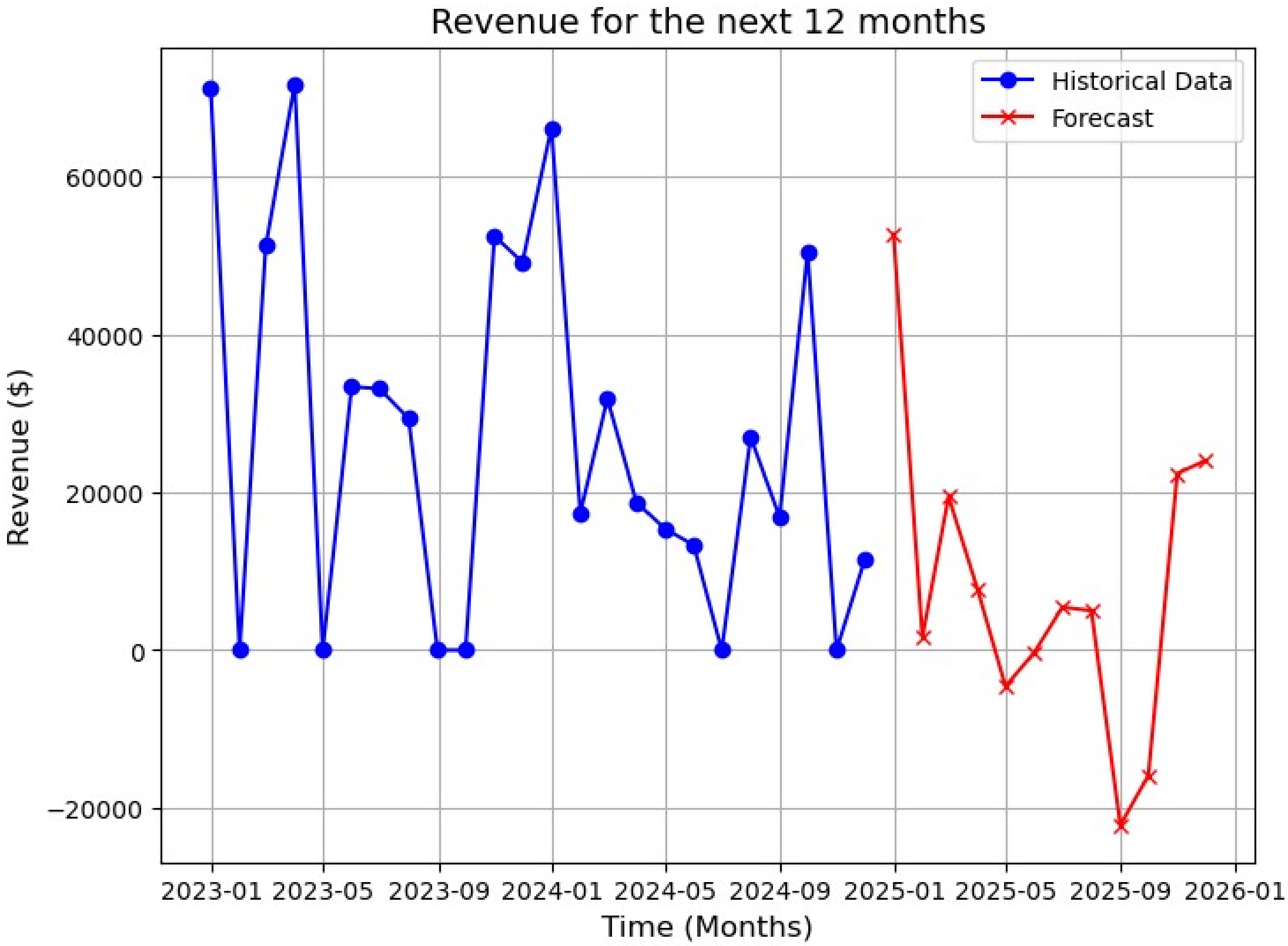
```
admission_date
2022-12-31    71211.95
2023-01-31        0.00
2023-02-28    51256.94
2023-03-31    71780.35
2023-04-30        0.00
Freq: M, Name: revenue, dtype: float64
```

```
# Time Series Forecast for the next 12 months
model = ExponentialSmoothing(time_serie_revenue, trend='add', seasonal='add', seasonal_periods=12)
model_fit = model.fit()
forecast = model_fit.forecast(steps=12)
```

```
C:\Users\rvrei\anaconda3\lib\site-packages\statsmodels\tsa\holtwinters\model.py:917: ConvergenceWarning: Optimizatio
  ConvergenceWarning,
```

```
# Plotting the results: Visualize both the historical data and the forecasted values
plt.figure(figsize=(8, 6))

# Plot the historical data in blue with markers
plt.plot(time_serie_revenue.index, time_serie_revenue, label='Historical Data', color='blue', marker='o')

# Plot the forecasted data for the next 12 months in red with different markers.
forecast_index = pd.date_range(time_serie_revenue.index[-1] + pd.Timedelta(days=1), periods=12, freq='M')
plt.plot(forecast_index, forecast, label='Forecast', color='red', marker='x')

# Add labels to the axes
plt.xlabel('Time (Months)', fontsize=12)  # Label for x-axis (Time in months)
plt.ylabel('Revenue ($)', fontsize=12)    # Label for y-axis (Revenue in dollars)
plt.title('Revenue for the next 12 months', fontsize=14)

# Show the plot
plt.legend()
plt.grid(True)
plt.show()
```

```
# Print forecasted values
print("Forecasted Values:")
print(forecast)
```



```
Forecasted Values:
2024-12-31    52782.087326
2025-01-31     1646.459218
2025-02-28    19476.800462
2025-03-31     7736.344047
```

```
2025-04-30    -4604.638748
2025-05-31     -267.708485
2025-06-30     5463.936737
2025-07-31     4999.198037
2025-08-31   -22223.181891
2025-09-30   -15923.654791
2025-10-31    22366.307003
2025-11-30    23991.604622
Freq: M, dtype: float64
```

```
# Information about the optimization process used to fit the model, and the outcome of the Maximum
# Likelihood Estimation (MLE) process to estimate the parameters of the Exponential Smoothing model.
print(model_fit.mle_retvals)
```

```
     fun: 12055696603.551268
     jac: array([ 1.7438336e+07, -1.2491520e+06,  9.7268480e+07, -1.9289600e+05,
       -2.5601280e+06,  2.4320000e+03,  4.3904000e+04, -2.6496000e+04,
       -9.8944000e+04,  1.9072000e+04, -3.3536000e+04,  2.6240000e+04,
       -3.1872000e+04, -6.7968000e+04, -1.1865600e+05,  6.1312000e+04,
        3.3152000e+04])
 message: 'Inequality constraints incompatible'
    nfev: 18
     nit: 1
    njev: 1
  status: 4
 success: False
       x: array([ 5.00000000e-03,  5.00000000e-03,  1.42142857e-01,  3.38180238e+04,
       -9.19692596e+02,  4.13564536e+04, -5.98333601e+03,  7.86155524e+03,
       -8.03614767e+03, -1.13064748e+04, -9.62759184e+03,  1.70573941e+03,
       -2.60388851e+03, -3.18559393e+04, -2.88382460e+04,  2.52338277e+04,
        2.20940482e+04])
```

```
# The results: a detailed summary of the model's performance, statistical significance, and fit quality.
print(model_fit.summary())
```

```
                       ExponentialSmoothing Model Results
================================================================================
Dep. Variable:                  revenue   No. Observations:                   24
```

```
Model:             ExponentialSmoothing   SSE                    12055696603.551
Optimized:                         True   AIC                            512.834
Trend:                         Additive   BIC                            531.683
Seasonal:                      Additive   AICC                           649.634
Seasonal Periods:                    12   Date:                 Tue, 03 Dec 2024
Box-Cox:                          False   Time:                         09:40:11
Box-Cox Coeff.:                    None
=================================================================================
                          coeff                 code              optimized
---------------------------------------------------------------------------------
smoothing_level                0.0050000                alpha                 True
smoothing_trend                0.0050000                 beta                 True
smoothing_seasonal             0.1421429                gamma                 True
initial_level                  33818.024                  l.0                 True
initial_trend                 -919.69260                  b.0                 True
initial_seasons.0              41356.454                  s.0                 True
initial_seasons.1             -5983.3360                  s.1                 True
initial_seasons.2              7861.5552                  s.2                 True
initial_seasons.3             -8036.1477                  s.3                 True
initial_seasons.4             -11306.475                  s.4                 True
initial_seasons.5             -9627.5918                  s.5                 True
initial_seasons.6              1705.7394                  s.6                 True
initial_seasons.7             -2603.8885                  s.7                 True
initial_seasons.8             -31855.939                  s.8                 True
initial_seasons.9             -28838.246                  s.9                 True
initial_seasons.10             25233.828                 s.10                 True
initial_seasons.11             22094.048                 s.11                 True
---------------------------------------------------------------------------------
```

When attempting to forecast with the model, we encountered the following issues:

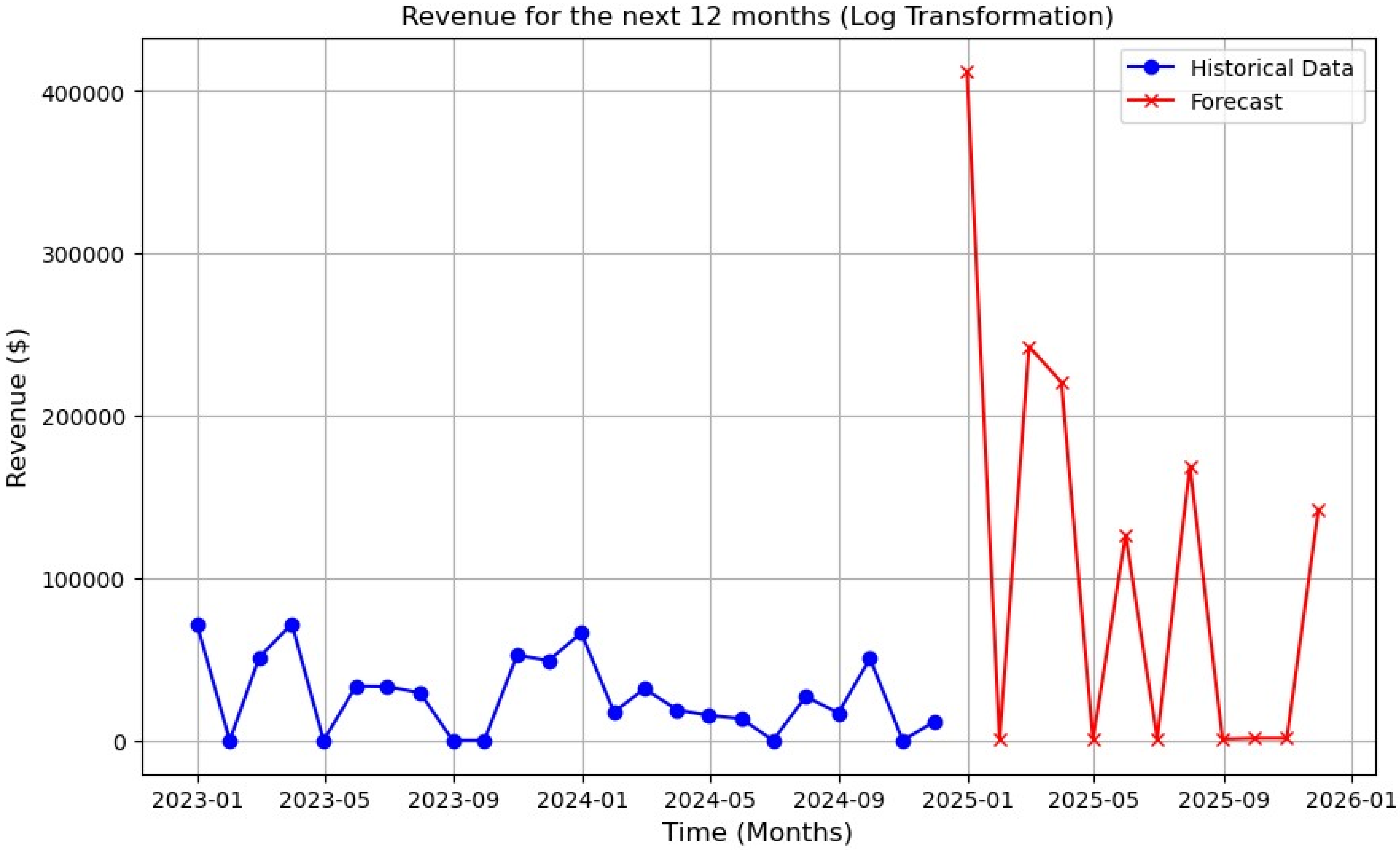
> - Optimization failed to converge: The presence of a large Jacobian array with high values and a status of 4 indicates that the model had difficulty converging. This could be due to the model's high sensitivity to initial parameters or poorly conditioned data, such as the presence of zeros or extreme values.
> - Incompatible Inequality Constraints: This suggests that the model's parameters couldn't be adjusted to fit the data while adhering to the imposed constraints.

Time Series Forecasting: Log Transformation

We apply a log transformation to stabilize the variance and make any growth patterns in the data easier to model and forecast accurately. Additionally, this helps address some convergence issues and optimization problems related to data conditioning.

```
# Apply Log Transformation to the data (with small constant to avoid log(0))
data_log = np.log(time_serie_revenue + 1)

# Fit the Exponential Smoothing model on the log-transformed data trend and seasonality set to
# "additive" (trend='add', seasonal='add'). The seasonality period is set to 12 months
model_log = ExponentialSmoothing(data_log, trend='add', seasonal='add', seasonal_periods=12)
model_fit_log = model_log.fit()

# Forecasting for the next 12 months
forecast_log = model_fit_log.forecast(steps=12)

# Inverse the Log Transformation to return the forecast back to the original scale
forecast1 = np.exp(forecast_log) - 1


# Plotting the results: Visualize both the historical data and the forecasted values
plt.figure(figsize=(10, 6))

# Plot the historical data in blue with markers
plt.plot(time_serie_revenue.index, time_serie_revenue, label='Historical Data', color='blue', marker='o')

# Plot the forecasted data for the next 12 months in red with different markers.
forecast1_index = pd.date_range(time_serie_revenue.index[-1] + pd.Timedelta(days=1), periods=12, freq='M')
plt.plot(forecast1_index, forecast1, label='Forecast', color='red', marker='x')

# Add labels and title
plt.title('Revenue for the next 12 months (Log Transformation)')
plt.xlabel('Time (Months)', fontsize=12)  # Label for x-axis (Time in months)
plt.ylabel('Revenue ($)', fontsize=12)    # Label for y-axis (Revenue in dollars)
```

```
# Show the plot
plt.grid(True)
plt.legend()
plt.show()

# Print forecasted values
print("Forecasted Values (in original scale):")
print(forecast1)
```



```
Forecasted Values (in original scale):
```

```
Forecasted values (in original scale):
2024-12-31    412526.496556
2025-01-31       791.435779
2025-02-28    242728.196180
2025-03-31    220460.709676
2025-04-30       744.722673
2025-05-31    126680.851798
2025-06-30      1092.390955
2025-07-31    168953.216164
2025-08-31       779.840774
2025-09-30      1349.346649
2025-10-31      1375.497076
2025-11-30    142396.582263
Freq: M, dtype: float64
```

```
print(model_fit_log.summary())
```

```
                       ExponentialSmoothing Model Results
================================================================================
Dep. Variable:                  revenue   No. Observations:                   24
Model:             ExponentialSmoothing   SSE                            307.355
Optimized:                         True   AIC                             93.199
Trend:                         Additive   BIC                            112.048
Seasonal:                      Additive   AICC                           229.999
Seasonal Periods:                    12   Date:                 Tue, 03 Dec 2024
Box-Cox:                          False   Time:                         09:40:18
Box-Cox Coeff.:                    None
=================================================================================
                          coeff                 code              optimized
---------------------------------------------------------------------------------
smoothing_level              1.4901e-08                alpha                 True
smoothing_trend              1.4847e-08                 beta                 True
smoothing_seasonal             0.000000                gamma                 True
initial_level                 6.5836278                  l.0                 True
initial_trend                 0.0996090                  b.0                 True
initial_seasons.0             3.8562043                  s.0                 True
initial_seasons.1            -2.4983514                  s.1                 True
initial_seasons.2             3.1266298                  s.2                 True
initial_seasons.3             2.9307984                  s.3                 True
initial_seasons.4            -2.8579362                  s.4                 True
```

```
          initial_seasons.4          -2.8579362                  s.4               True
          initial_seasons.5           2.1775351                  s.5               True
          initial_seasons.6          -2.6744690                  s.6               True
          initial_seasons.7           2.2662659                  s.7               True
          initial_seasons.8          -3.2103549                  s.8               True
          initial_seasons.9          -2.7622186                  s.9               True
          initial_seasons.10         -2.8426471                 s.10               True
          initial_seasons.11          1.6968250                 s.11               True
          --------------------------------------------------------------------------------
```

```
print(model_fit_log.mle_retvals)
```

```
          fun: 307.35475694313
          jac: array([ 3.07354771e+02,  0.00000000e+00,  3.07354752e+02,  2.25067139e-04,
             3.11279297e-03,  2.28881836e-05, -4.95910645e-05,  1.52587891e-05,
             1.90734863e-05, -1.14440918e-05,  4.19616699e-05, -3.81469727e-05,
             3.43322754e-05,  8.77380371e-05,  1.02996826e-04, -2.67028809e-05,
             3.43322754e-05])
      message: 'Optimization terminated successfully'
         nfev: 207
          nit: 11
         njev: 11
       status: 0
      success: True
            x: array([ 1.49011612e-08,  1.48468089e-08,  0.00000000e+00,  6.58362777e+00,
             9.96090424e-02,  3.85620431e+00, -2.49835140e+00,  3.12662977e+00,
             2.93079835e+00, -2.85793622e+00,  2.17753508e+00, -2.67446897e+00,
             2.26626592e+00, -3.21035491e+00, -2.76221859e+00, -2.84264705e+00,
             1.69682500e+00])
```