Healthcare Project - Time Series Analysis for Forecasting Claims Volume

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

```
# Modules
import pandas as pd
import numpy as np
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 claims amount by month
healthcare_df['claim_date'] = pd.to_datetime(healthcare_df['claim_date']) # Convert into datetime type
monthly_claims = healthcare_df.resample('M', on='claim_date').claim_amount.sum()
monthly_claims.head()
```

```
claim_date
2022-11-30    8004.99
2022-12-31     298.81
2023-01-31    3598.75
2023-02-28    6147.12
2023-03-31       0.00
Freq: M, Name: claim_amount, dtype: float64
```

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

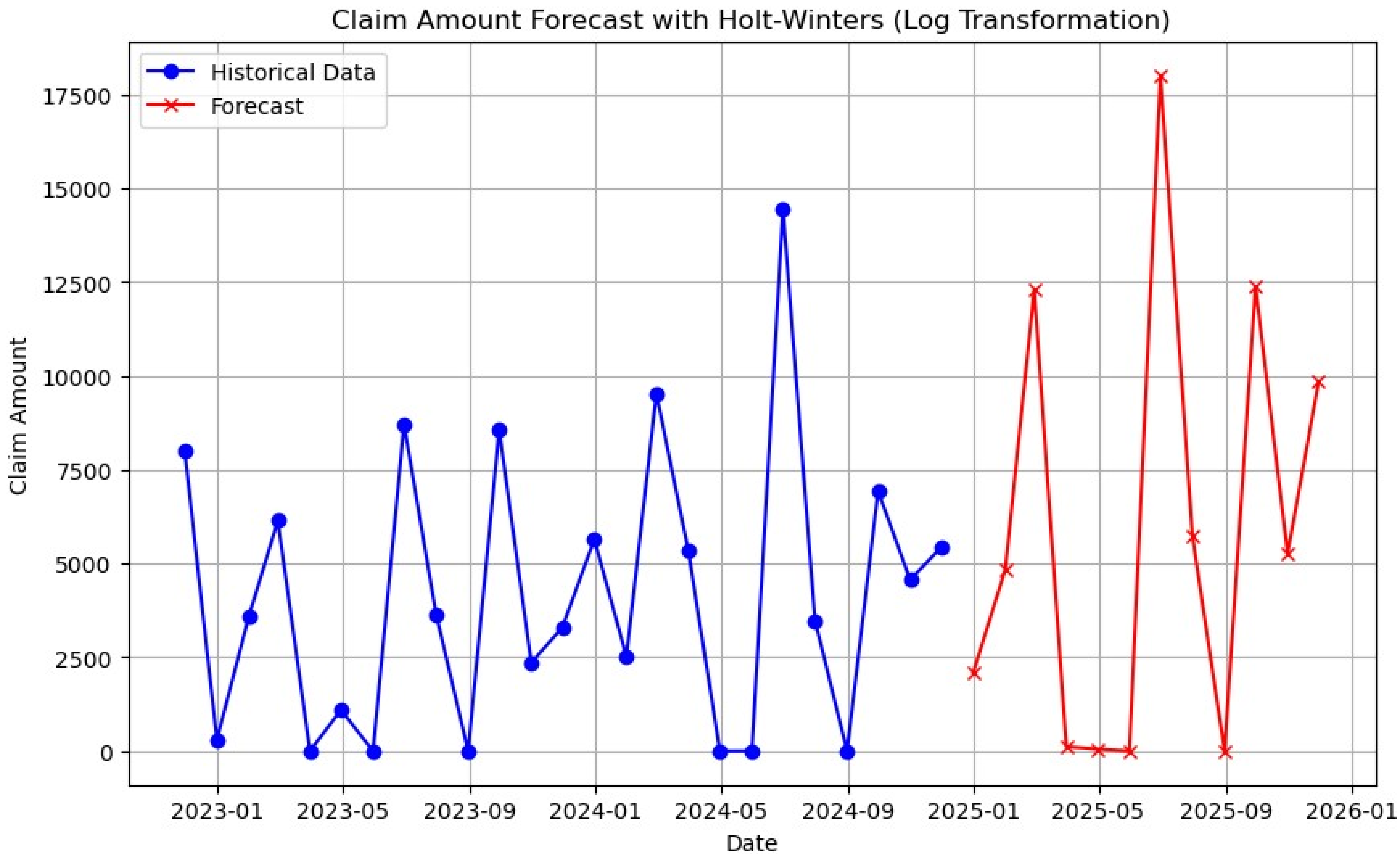
> - Failure to Converge: The optimizer was unable to find a solution that met the required conditions for the Holt-Winters model.
> - Incompatible Inequality Constraints: This suggests that the model's parameters couldn't be adjusted to fit the data while adhering to the imposed constraints.
> - Optimization Struggles: 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.

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(monthly_claims + 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 = ExponentialSmoothing(data_log, trend='add', seasonal='add', seasonal_periods=12)
model_fit = model.fit()

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

```
# Inverse the Log Transformation to return the forecast back to the original scale
forecast = 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(monthly_claims.index, monthly_claims, 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(monthly_claims.index[-1] + pd.Timedelta(days=1), periods=12, freq='M')
plt.plot(forecast_index, forecast, label='Forecast', color='red', marker='x')

# Add labels and title
plt.title('Claim Amount Forecast with Holt-Winters (Log Transformation)')
plt.xlabel('Date')
plt.ylabel('Claim Amount')
plt.legend()

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

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




```
Forecasted Values (in original scale):
2024-12-31     2089.089011
2025-01-31     4840.605791
2025-02-28    12290.723438
2025-03-31      116.288199
```

```
2025-04-30       52.174624
2025-05-31        0.605953
2025-06-30    18021.066779
2025-07-31     5727.064374
2025-08-31        0.605945
2025-09-30    12385.526839
2025-10-31     5268.914685
2025-11-30     9847.415567
Freq: M, dtype: float64
```

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

```
                       ExponentialSmoothing Model Results
================================================================================
Dep. Variable:                claim_amount   No. Observations:                   25
Model:                ExponentialSmoothing   SSE                             65.809
Optimized:                            True   AIC                             56.197
Trend:                            Additive   BIC                             75.699
Seasonal:                         Additive   AICC                           170.197
Seasonal Periods:                       12   Date:                 Sun, 01 Dec 2024
Box-Cox:                             False   Time:                         20:25:36
Box-Cox Coeff.:                       None
=================================================================================
                          coeff                 code              optimized
---------------------------------------------------------------------------------
smoothing_level          1.4901e-08                alpha                True
smoothing_trend          1.4877e-08                 beta                True
smoothing_seasonal       4.5826e-11                gamma                True
initial_level             5.9255482                  l.0                True
initial_trend             0.0263181                  b.0                True
initial_seasons.0         2.2957463                  s.0                True
initial_seasons.1         1.0351420                  s.1                True
initial_seasons.2         1.8488636                  s.2                True
initial_seasons.3         2.7542252                  s.3                True
initial_seasons.4        -1.9241403                  s.4                True
initial_seasons.5        -2.7415113                  s.5                True
initial_seasons.6        -6.2676935                  s.6                True
initial_seasons.7         3.0316234                  s.7                True
```

```
initial_seasons.8            1.8590859                  s.8                True
initial_seasons.9           -6.3466529                  s.9                True
initial_seasons.10           2.5776813                 s.10                True
initial_seasons.11           1.6967680                 s.11                True
-----------------------------------------------------------------------------------
```

```
# 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: 65.80864175803032
     jac: array([ 6.58086424e+01,  0.00000000e+00,  6.58086405e+01,  1.91402435e-03,
        2.95305252e-02,  1.72615051e-03,  2.67028809e-05,  1.05857849e-04,
        6.48498535e-05, -1.73568726e-04,  2.44140625e-04, -3.81469727e-05,
        2.95639038e-05, -4.67300415e-05, -5.72204590e-05, -4.19616699e-05,
        7.24792480e-05])
 message: 'Optimization terminated successfully'
    nfev: 226
     nit: 12
    njev: 12
  status: 0
 success: True
       x: array([ 1.49011612e-08,  1.48767579e-08,  4.58261873e-11,  5.92554816e+00,
        2.63181467e-02,  2.29574628e+00,  1.03514196e+00,  1.84886360e+00,
        2.75422515e+00, -1.92414027e+00, -2.74151126e+00, -6.26769348e+00,
        3.03162336e+00,  1.85908594e+00, -6.34665289e+00,  2.57768132e+00,
        1.69676801e+00])
```