Advanced Data Analytics

PA1 – Time Series Modeling

Jason Willis

College of Information Technology,

Western Governors University

Dr. Festus Elleh

Aug 1, 2022

Table of Contents for Each Rubric

Hospital Readmission Problem

For our chain of hospitals to lower readmission concerns, company executives requested a time series data analysis using revenue data. This data set compiled revenue data from the first two years of operation.

A1 – Research Question

Using the previous two years of data, are there any patterns present that can predict the revenue produced by the hospital for the next quarter?

A2 – Objectives and Goals

The goal of our analysis is to leverage time series modeling techniques to forecast the daily revenue of our company's next quarter. Then, to provide our pattern understandings, analysis and findings to company executives so they can understand readmission impacts in current times.

B – Summary of Assumptions

According to Rajbhoj (2019) "[ARIMA] models assume that the series is stationary. This also means that 'stationarity' is a necessary condition to take advantage of these models for any time series." Time series assumes the data is stationary, e.g. normally distributed with constant mean and variance during the same time period. The error term is uncorrelated and randomly distributed, while and the mean and variance are constant over the period of time. There shouldn't be outliers.

In order to meet stationarity requirements, the time series is assumed to experience normal distribution, e.g. it cannot experience seasonality. Additionally, the mean and standard deviation both need to be constant (constant variance), and the autocorrelation is constant. (Reider R., Datacamp, n.d.)

C1 – Line Graph Visualization

Refer to Figure 1 for a line graph of the provided medical data.

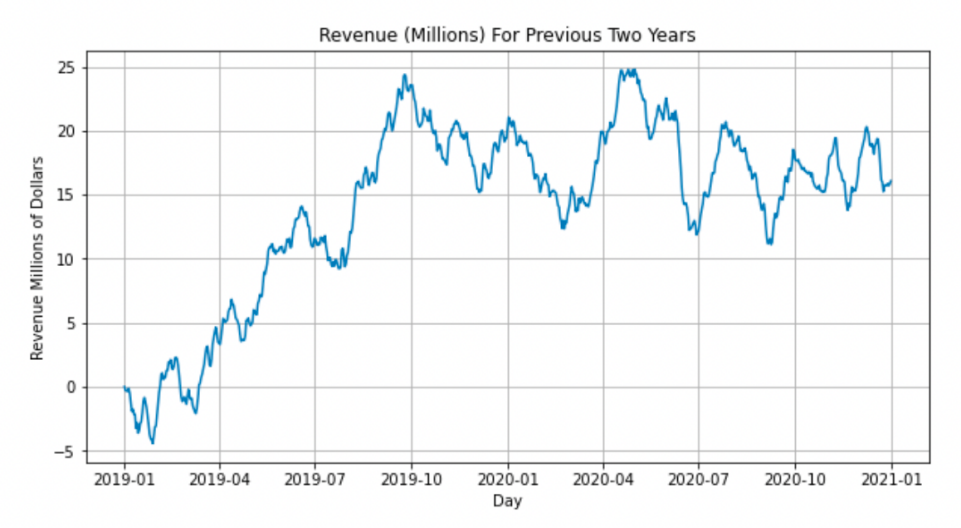

Figure 1 - Linear Graph of Medical Data

C2 – Time Step Formatting

Time step formatting is leveraged to explore for gaps in our data measurements. The time series is 731 days long, with each step of one day. This data set, 731 days long, was formatted to have no gaps or missing data (days).

C3 – Stationarity

The time series is evaluated for stationarity and aligns with the chosen dataset and research question. Please see Figure 2:

```
# Test if data is stationary again

result = adfuller(df_stationary['Revenue'])

print("Test Statistics: ", result[0])
print("p-value: ", result[1])
print("Critical Values: ",result[4])

if result[1] <= 0.05: #Compare result against threshold

    print("Reject null hypothesis, this time series data is stationary.")
else:
    print("Accept null hypothesis, this time series data is non-stationary!")
```

```
Test Statistics:  -17.37477230355706
p-value:  5.1132069788403175e-30
Critical Values:  {'1%': -3.4393520240470554, '5%': -2.8655128165959236, '10%':
-2.5688855736949163}
Reject null hypothesis, this time series data is stationary.
```

Figure 2 - Evaluate Stationarity of Time Series

C4 – Steps to Prepare the Data

In order to prepare the data for analysis, the following steps were performed in accordance with Dr. Elleh's presentation (2022):

- Data was imported and read into the notebook
- Exploratory data analysis was performed to understand the dataset deeper, such as shape, identifying null values (if any), and I chose to convert the day field to a date field with an index.
- Created a line graph to visually view the data

- Tested data to see if it's stationary (p-value is less than the significance level of 0.05). Found the data wasn't stationary, so the data frame was processed using df.diff().dropna() to remove nulls. Then, retested with the augmented Dickey Fuller (ADF) test to verify stationarity.
- Data is split into a training and testing subsets using approximately an 80/20 split.

C5 – Prepared Dataset

The submission includes prepared and cleaned data set: df_cleaned_stationary.csv

D1 – Report Findings and Visualizations

The submission includes annotated findings with visualizations of the data analysis:

• Seasonality – Figure 3 shows cyclical data is present.

• Trends – Figure 4 displays an overall downward direction over time.

• Autocorrelation – Figure 5 displays both autocorrelation function (ACF) and a partial autocorrelation function (PACF). In the visual, we see that the values increasing and decreasing slightly; yet, still stay within the blue shaded band as expected for stationary data.

• Spectral Density – Figure 6 presents a graph displaying autocorrelation time series frequencies.

• Decomposed Timeseries – Figure 7 splits the time series into three components: Trend, Seasonality and Residual.

• Residuals – Figure 8 confirms the lack of trends in the residuals of the decomposed series.

```
# Plot Seasonality

plt.title('Seasonality')
decomp.seasonal.plot();
```

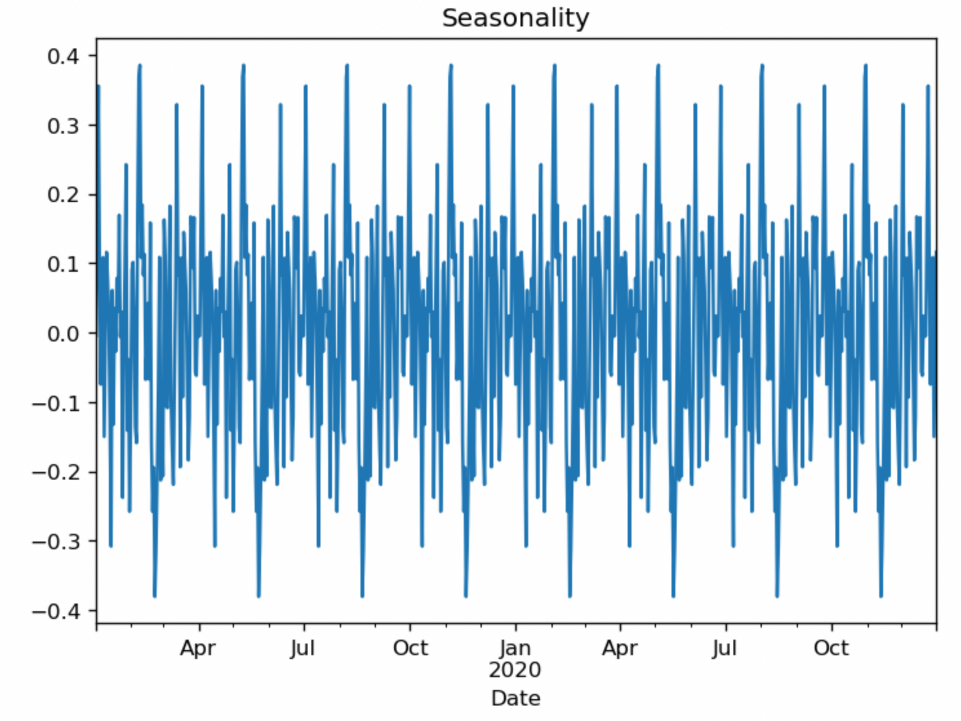

Figure 3 - Looking for Seasonality

```
# View Trend
plt.title('Trend')
decomp.trend.plot();
```

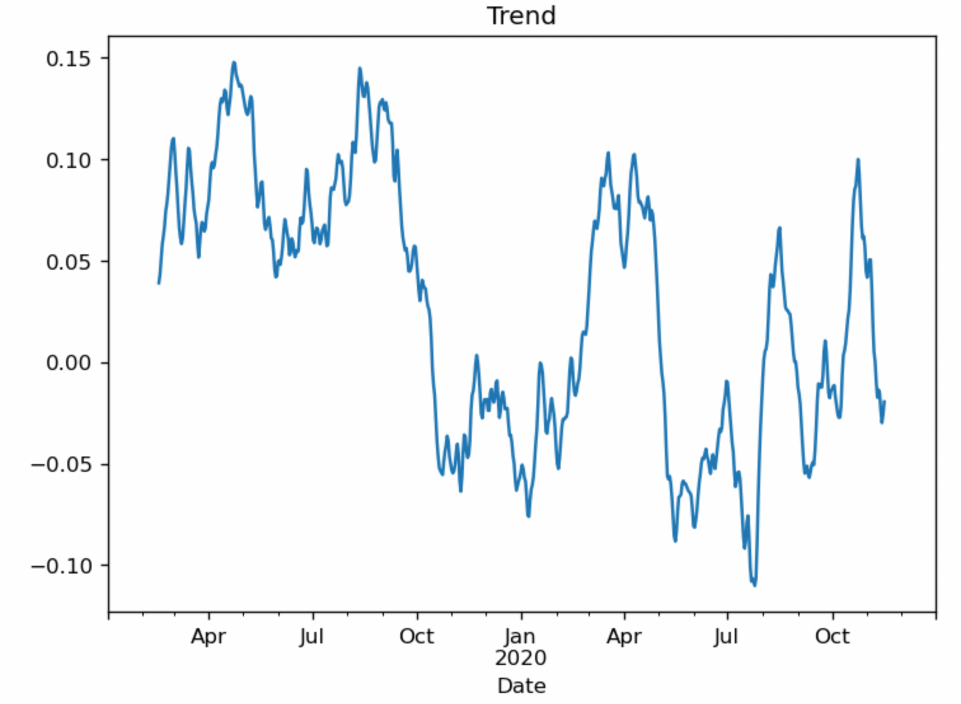

Figure 4 - Look for a Trend

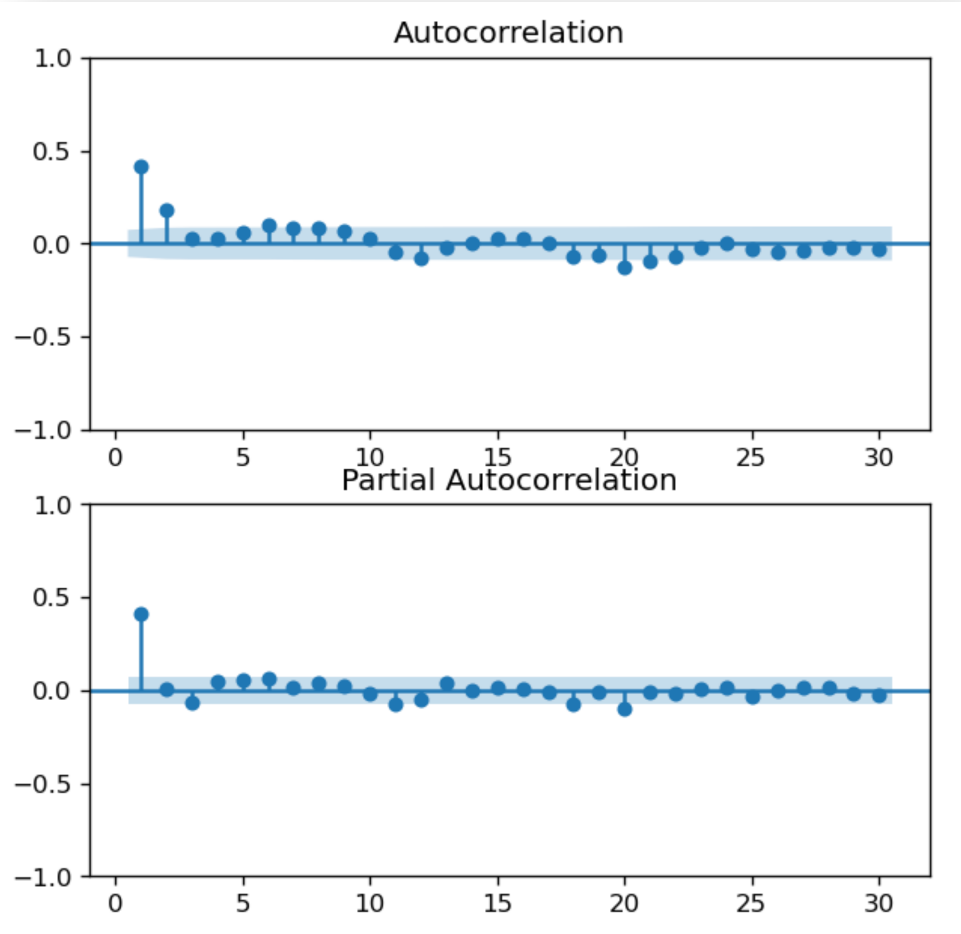

Figure 5 - Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF)

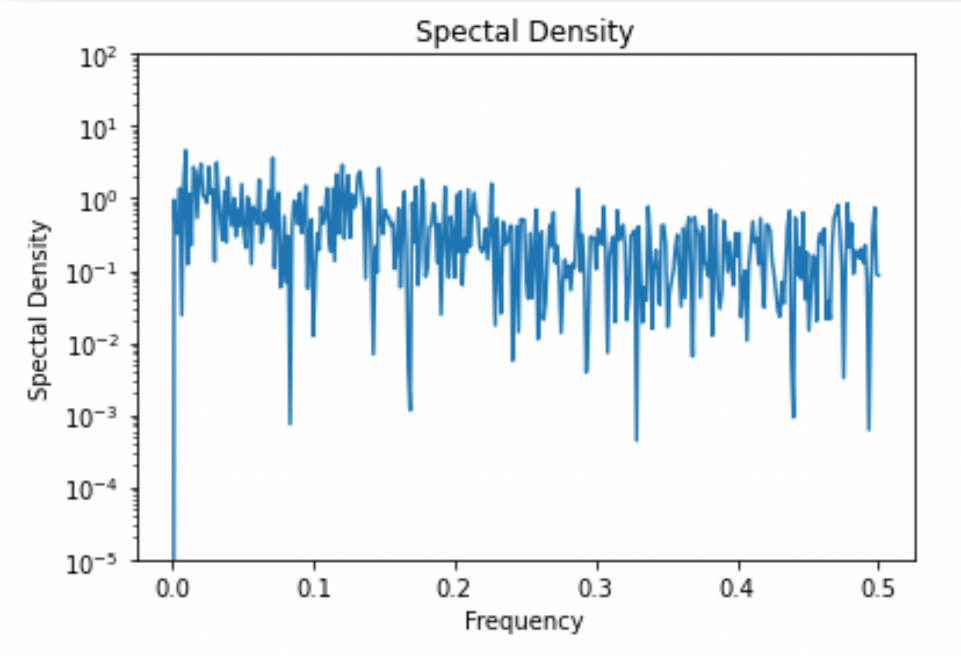

Figure 6 - Spectral Density

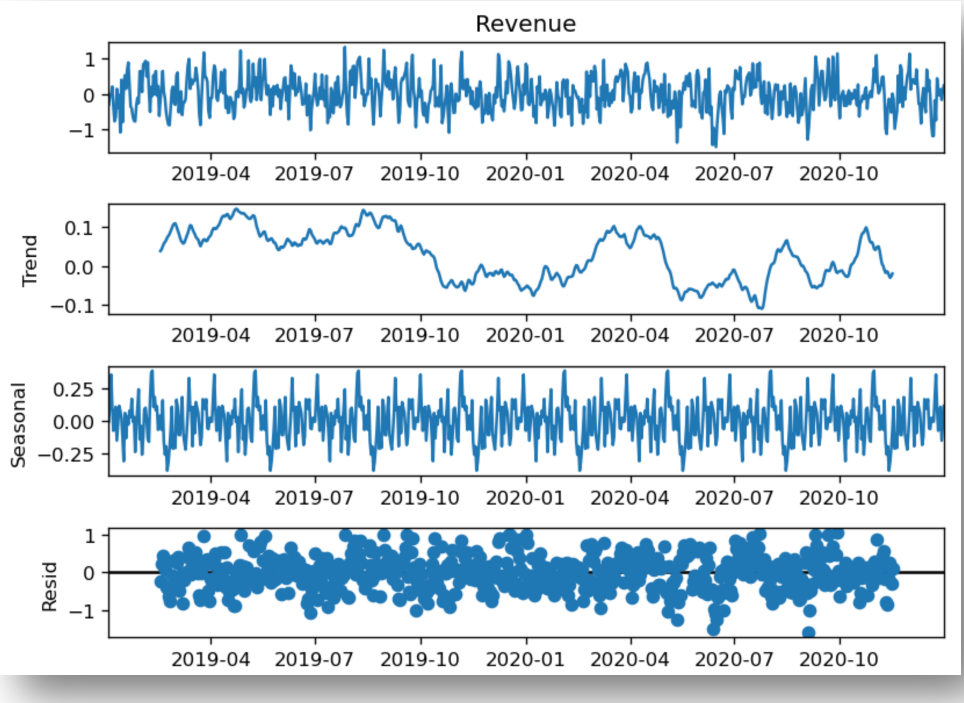

Figure 7 - Decomposing Time Series

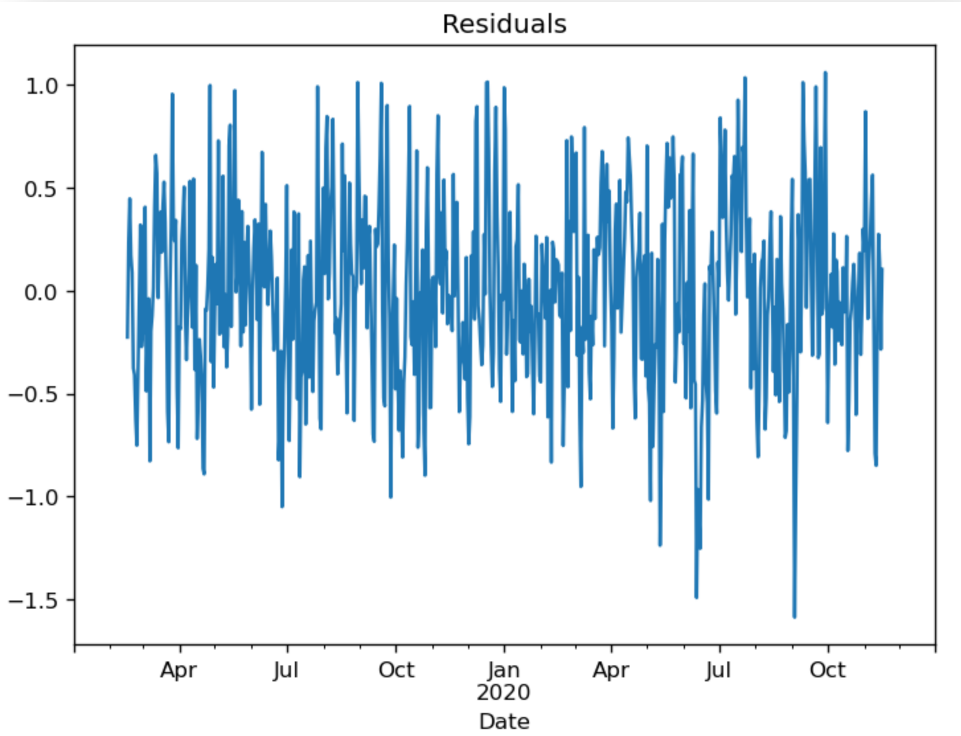

Figure 8 - Residuals of Decomposed Time Series

D2 – ARIMA Model

Ran an autoregressive integrated moving average (ARIMA) to help identify the best model fit. (Figure 9)

```
%time
tqdm.pandas()
model = pm.auto_arima(df_stationary,
                      seasonal=True, m=90,
                      d=1, D=1,
                      start_p=1, start_q=1,
                      max_p=2, maxq=2,
                      maxP=2, maxQ=2,
                      trace=True,
                      error_action='ignore',
                      suppress_warnings=True)
```

```
CPU times: user 2 µs, sys: 0 ns, total: 2 µs
Wall time: 4.05 µs
Performing stepwise search to minimize aic
 ARIMA(1,1,1)(1,1,1)[90]             : AIC=inf, Time=687.89 sec
 ARIMA(0,1,0)(0,1,0)[90]             : AIC=1448.607, Time=7.33 sec
 ARIMA(1,1,0)(1,1,0)[90]             : AIC=inf, Time=56.88 sec
 ARIMA(0,1,1)(0,1,1)[90]             : AIC=inf, Time=349.59 sec
 ARIMA(0,1,0)(1,1,0)[90]             : AIC=inf, Time=33.81 sec
 ARIMA(0,1,0)(0,1,1)[90]             : AIC=inf, Time=159.45 sec
 ARIMA(0,1,0)(1,1,1)[90]             : AIC=inf, Time=297.59 sec
 ARIMA(1,1,0)(0,1,0)[90]             : AIC=1390.122, Time=10.78 sec
 ARIMA(1,1,0)(0,1,1)[90]             : AIC=inf, Time=323.06 sec
 ARIMA(1,1,0)(1,1,1)[90]             : AIC=inf, Time=363.84 sec
 ARIMA(2,1,0)(0,1,0)[90]             : AIC=1366.083, Time=13.73 sec
 ARIMA(2,1,0)(1,1,0)[90]             : AIC=inf, Time=55.67 sec
 ARIMA(2,1,0)(0,1,1)[90]             : AIC=inf, Time=371.44 sec
 ARIMA(2,1,0)(1,1,1)[90]             : AIC=inf, Time=786.04 sec
 ARIMA(2,1,1)(0,1,0)[90]             : AIC=inf, Time=296.54 sec
 ARIMA(1,1,1)(0,1,0)[90]             : AIC=inf, Time=192.69 sec
 ARIMA(2,1,0)(0,1,0)[90] intercept   : AIC=1368.083, Time=33.45 sec

Best model:  ARIMA(2,1,0)(0,1,0)[90]
Total fit time: 4039.779 seconds
```

Figure 9 - Auto ARIMA: Identifying Best Model

D3 – Forecasting Using ARIMA Model

Provided in Figure 10 is an ARIMA derived model forecast.

```
# Plot Data
plt.figure(figsize=(12,4))
#plt.plot(X_test.index, X_test, label='Observed X_test')
plt.plot(np.array(X_test.index), np.array(X_test[['Revenue']]), label='Observed (test set)')

# plot your mean predictions
plt.plot(mean_prediction.index, mean_prediction, color='r', label='Forecast')

# shade upper conf. limit area
#plt.fill_between(lower_limits.index, lower_limits, upper_limits, color='pink')
plt.fill_between(upper_limits.index, upper_limits, lower_limits, color='lightgreen')

## plot mean predictions
#plt.fill_between(mean_prediction.index, mean_prediction, color='brown', label='forecast')

# Annotations: Labels and Legends
plt.title('Forecast Comparing w/Test Data')
plt.xlabel('Date')
plt.ylabel('Revenue ($ Millions)')
plt.legend()
plt.show()
```

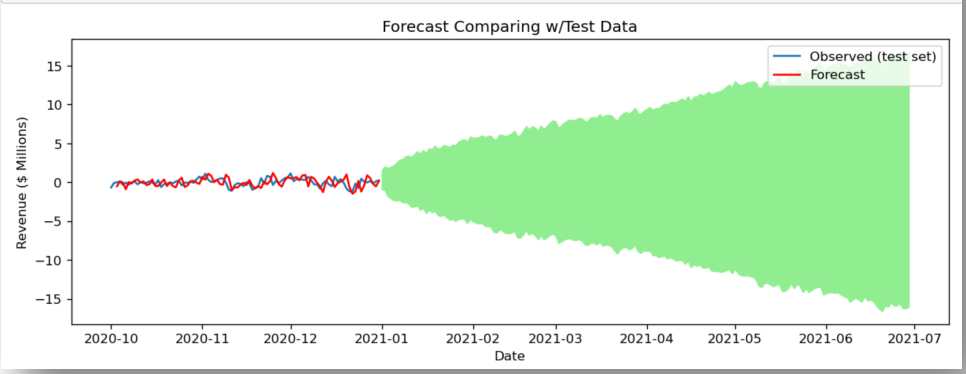

Figure 10 - ARIMA Forecast with Test Data

D4 – Output and Calculations

Provided in Jupyter notebook and pdf are the code for validating stationarity, autocorrelation plots (ACF and PACF), SARIMAX, the model summary, and future predictions.

D5 – Code

Provided in the zipped folder is a Jupyter notebook (D213-AdvancedDataAnalyticsPA1.ipynb) and pdf export (D213-AdvancedDataAnalyticsPA1.pdf) detailing the code used to support the implementation of the time series model.

E1 – Results

ACF and PACF were generated to glean a suitable ARIMA model. Several AR and MA sequence combinations were run to identify the model with the lowest AIC score. Please see Figure 11. Our forecast's prediction interval is 1 day. Our data set contains 2-years of daily revenue; therefore, the ARIMA model identifies the correlations and seasonality in order to predict revenue at a daily interval. Please see Figure 12. Since we are leveraging a 2-year daily revenue data set, our time series data can only forecast up to 1-year of future revenue. Earlier predictions would provide higher accuracy; thus, a date range of 90 days was chosen for our prediction. An error metric helped evaluate of our fitted ARIMA(p, q) model. The MAE results seen in Figure 13 show a low error rate given the number of observations within the data set.

```
# Pick best order by aic

best_aic = np.inf
best_order = None
best_mdl = None
rng = range(3)
for p in rng: # loop over p
    for q in rng: #loop over q
        try: #create and fit ARIMA(p,q) model
            model = SARIMAX(df_stationary, order=(p,1,q), trend='c')
            results = model.fit()
            tmp_aic = results.aic
            print(p, q, results.aic, results.bic)
            if tmp_aic < best_aic: # value swap
                best_aic = tmp_aic
                best_order = (p, q)
                best_mdl = tmp_mdl

            # Print order and results
        except:
            print(p,q, None, None)

print('\nBest AIC: {:6.5f} | order: {}'.format(best_aic, best_order))
```

```
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
    6     29     48      1     0     0   2.696D-03   6.022D-01
  F =  0.60220556374000445

CONVERGENCE: REL_REDUCTION_OF_F_<=_FACTR*EPSMCH
2 2 891.2201230604064 918.7701654524584

Best AIC: 887.66351 | order: (1, 1)
```

Figure 11 - Choosing Best AIC Score

```
# Plot Training Data
plt.figure(figsize=(12,4))
plt.plot(np.array(X_train.index), np.array(X_train[['Revenue']]), label='Observed (Training Set)')

# plot your mean predictions
plt.plot(mean_prediction_test.index, mean_prediction_test, color='r', label='Forecast (Testing Set)')

# shade upper conf. limit area
#plt.fill_between(lower_limits.index, lower_limits, upper_limits, color='pink')
plt.fill_between(upper_limits_test.index, upper_limits_test, lower_limits_test, color='lightgreen')

## plot mean predictions
#plt.fill_between(mean_prediction.index, mean_prediction, color='brown', label='forecast')

# Annotations: Labels and Legends
plt.title('Forecast Comparing w/Test Data')
plt.xlabel('Date')
plt.ylabel('Revenue ($ Millions)')
plt.legend()
plt.show()
```

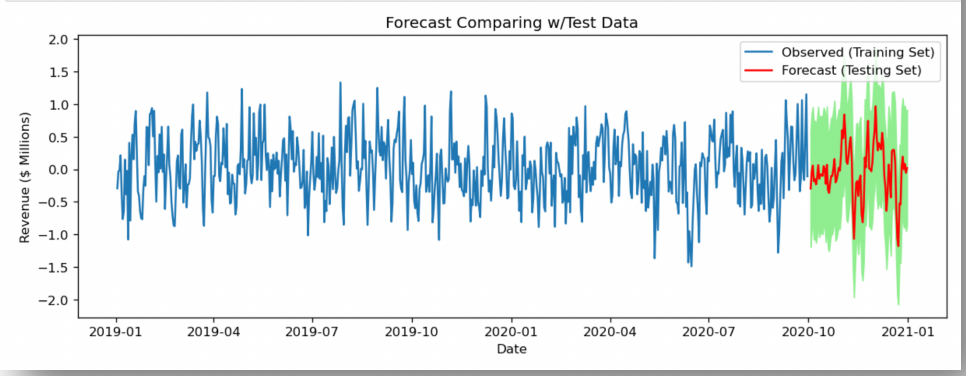

Figure 12 - Forecasted Revenue Projections (2021)

```
# Plot forecast
plt.figure(figsize=(12,4))
#plt.plot(df_stationary.index, df_stationary, label='Observed')
plt.plot(np.array(df_stationary.index), np.array(df_stationary[['Revenue']].reset_index(drop=Tr

# Plot mean predictions

plt.plot(mean_forecast.index, mean_forecast, color='r', label='Forecast')

# shade conf. limit area
#plt.fill_between(lower_limits.index, lower_limits, upper_limits, color='pink')
plt.fill_between(upper_limits.index, upper_limits, lower_limits, color='lightgreen')



# Annotations: Labels and Legends
plt.title('Forecasted 2021 Revenue Projections')
plt.xlabel('Date')
plt.ylabel('Revenue ($ Millions)')
plt.xticks(rotation=45)
plt.legend()
plt.show()
```

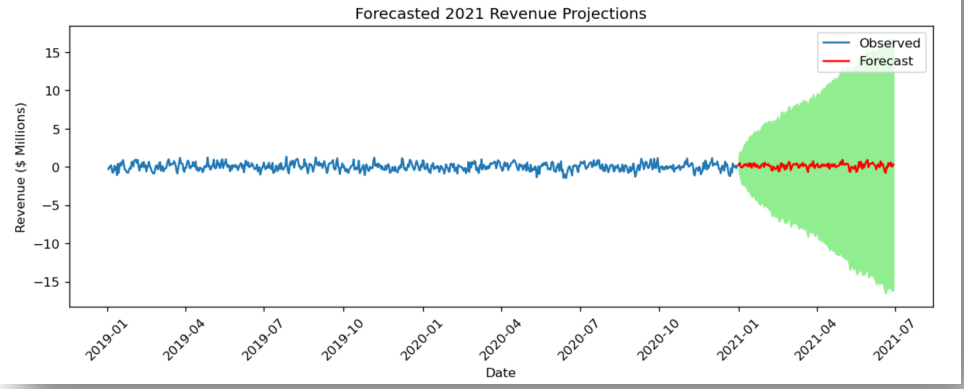

Figure 12 - Forecasted Revenue Projections (2021) Cont…

A model run using the data only from the training set and forecasted out to the test set

```
start=len(X_train) # A model run using the data only from the training set
end=len(X_train)+len(X_test)-1
pred=results.predict(start=start, end=end, typ='levels') # forecasted out to the test set
# Set index for plotting
pred.index=df_stationary.index[start:end+1]
print(pred)
```

```
Date
2020-10-01    0.691880
2020-10-02   -0.355448
2020-10-03   -0.516314
2020-10-04    0.111306
2020-10-05   -0.037037
                ...
2020-12-27    0.883200
2020-12-28    0.541385
2020-12-29   -0.219236
2020-12-30   -0.530855
2020-12-31    0.164098
Name: predicted_mean, Length: 92, dtype: float64
```

```
pred.plot(legend=True)
X_test['Revenue'].plot(figsize=(12,4),legend=True);
```

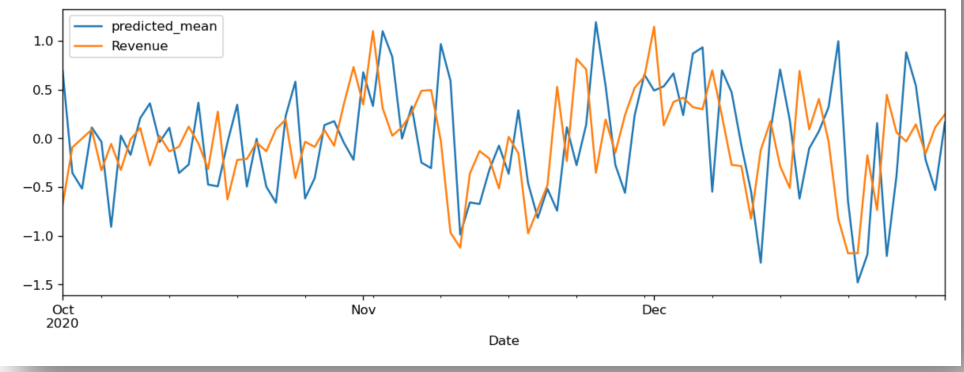

Figure 12 - Forecasted Revenue Projections (2021) Cont…

Standard Error Metric: Train, Test and Actual Data

```
# Print mean absolute error
mae = np.mean(np.abs(a_model.resid))
mae_train = np.mean(np.abs(a_model_train.resid))
mae_test = np.mean(np.abs(a_model_test.resid))

print("Actual - Mean Absolute Error Data: ", mae)
print("Actual - Mean Absolute Error Training Data: ", mae_train)
print("Actual - Mean Absolute Error Test Data: ", mae_test)
```

```
Actual - Mean Absolute Error Data:  0.49873094599791684
Actual - Mean Absolute Error Training Data:  0.49425653996990915
Actual - Mean Absolute Error Test Data:  0.35953135297954647
```

```
from sklearn.metrics import mean_squared_error
from math import sqrt
rmse = sqrt(mean_squared_error(pred,X_test['Revenue']))
print("RMSE of test data: ", rmse)
```

```
RMSE of test data:  0.6713765772122239
```

Figure 13 - Standard Error Metrics

E2 – Annotated Visualization

Provided in Figure 12 is an annotated model of our forecast, with out-of-sample predictions post Dec 31, 2021. When comparing this test data forecast against the final model, a strong correlation is seen.

E3 – Recommendations

The accuracy of our forecast (90 days) provides high confidence levels for predicting our future revenue. While you can utilize this forecast as a predictive guide, continue to reduce readmission rates which will directly lower penalty payments while also improving hospital/patient confidence. Additional forecast length (180 days) could be calculated with our current data set if needed.

F – Reporting

Provided in this document is an accurately created both VS-Code and Jupyter Notebooks. Both a Jupyter notebook (D213-AdvancedDataAnalyticsPA1.ipynb) and pdf export (D213-AdvancedDataAnalyticsPA1.pdf has been provided in the zipped package.

G – Sources for Third-Party Code

- Help using Markdown: https://www.markdownguide.org/basic-syntax/
- MacTeX: https://tug.org/mactex/mactex-download.html
- Matplotlib Help: https://matplotlib.org/2.1.2/api/_as_gen/matplotlib.pyplot.plot.html
- Numpy Help: https://numpy.org/doc/stable/
- Pandas Help: https://pandas.pydata.org/docs/user_guide/index.html#user-guide
- Python Help: https://docs.python.org/3.9/library/index.html
- Scipy.stats Help: https://docs.scipy.org/doc/scipy/reference/tutorial/stats.html

References

Elleh, F. (2022). Lecture: D213 T1 – Welcome to D213 Advanced Data Analytics. Western Governors University. Found Here: https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=efceba6c-e8ef-47a2-b859-aec400fe18e7

Becker, D., Reider, R., Hull, I., Biswanath, H., et. al. (n.d.) Advanced Data Analytics DataCamp. Found Here: https://app.datacamp.com/learn/custom-tracks/custom-advanced-data-analytics

Rajbhoj A. (2019). ARIMA Simplified. *Toward Data Science. Found Here:* https://towardsdatascience.com/arima-simplified-b63315f27cbc