NLM3 Task 1: Time Series Modeling

Steven Oldenburg

Western Governors University

D213: Advanced Data Analytics

04/1/2024

NLM3 Task 1: Time Series Modeling

A1: Research Question

Can we use time series modeling techniques to identify seasonal trends in our revenue data?

A2: Defined Goal

The goal of this is to identify seasonal trends for forecasting purposes. Our company is looking back and reflecting on their first two years of operations to use this data to help predict churn rates in future analysis. The company plans to do this by examining revenue drops which most likely translate into customer churn. If we can identify any seasonal trends this will shed light on additional insights on how to decipher the executive's goal of analyzing revenue and customer churn rates. For instance, if revenue increases during Black Friday, this is a trend we can account for and expect in future predictions, ignoring this can lead to poor understanding of yearly business metrics.

B1: Summary of Assumptions

Stationary Assumptions:

Stationary data assumptions are that the data has a constant mean and doesn't exhibit trends or seasonal fluctuations. It also assumes that there is a constant rate of variance around the mean. Finally, that correlation of its data points and its lagged values remains the same.

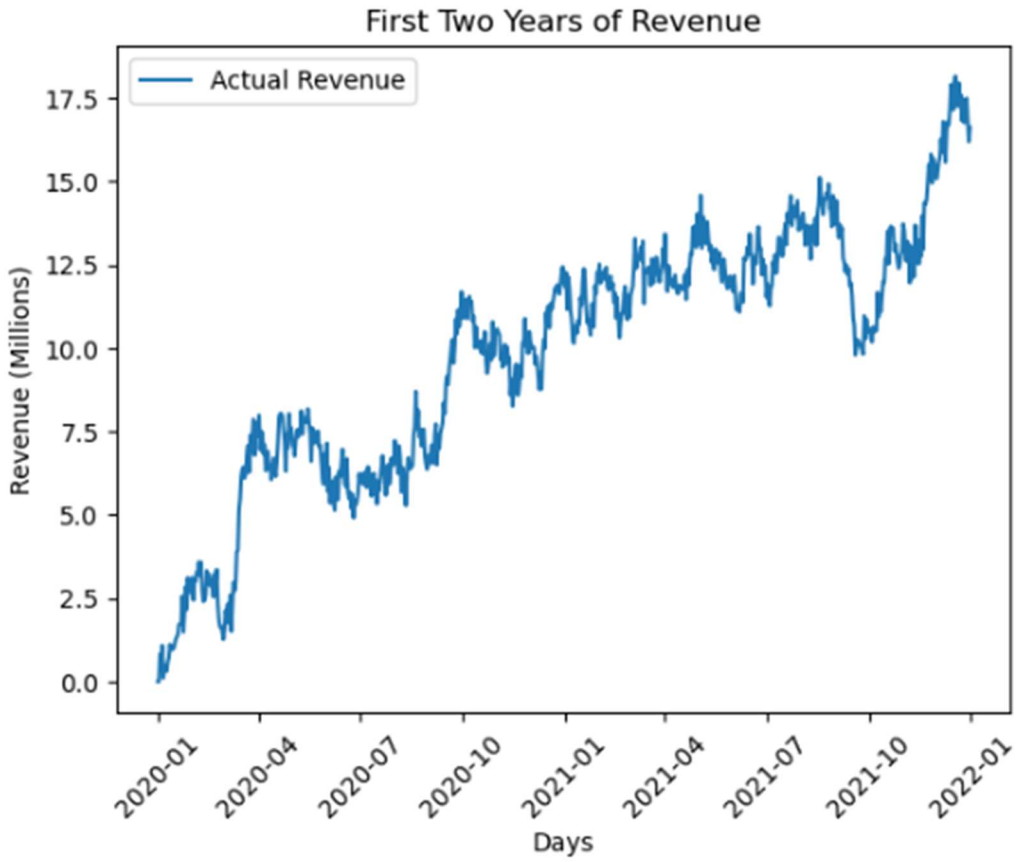
Autocorrelation:

Autocorrelation means predicting the future based on the past. If there is no autocorrelation, it means we cannot learn the relationship between past and future values. If there is a positive correlation, high revenue will be followed by more high revenue. If there is a negative correlation high values will be followed by opposite values. So basically when we have data that

is autocorrelated its current data point is based on its lagged data point depending on if the relationship is positive or negative.

These two concepts are important because without it we would not be able to make predictions accurately or reliably in the time series.

C1: Line Graph Visualization



C2: Time Step Formatting

The time series consists of 2 years of data where daily revenue is recorded, the two years consists of 731 datapoints. Most likely there is an extra day for a leap year, the data dictionary did not specify a year, so I used 01-01-2020 as a place value. The date time is at a frequency of one day per datapoint. Formatting the date as a datetime object helps to create different types of possible

aggregations. There are no gaps in the measurement. The data was downloaded from WGU as part of the assignment. While the bulk of sequencing will done with date and time, occasionally exceptions will be made when needed.

C3: Stationarity

The data is not stationary. It has an obvious upward trend. The first few data points start at very low revenue. The first 5 days of revenue range from 0 to 1 million dollars, the last 5 days of revenue range from 16 to 17.5 million dollars per day. Also looking at the line graph submitted under C1 in this paper you can see an upward trend.

```
   Day   Revenue
0    1  0.000000
1    2  0.000793
2    3  0.825542
3    4  0.320332
4    5  1.082554
     Day     Revenue
726  727  16.931559
727  728  17.490666
728  729  16.803638
729  730  16.194813
730  731  16.620798
```

C4: Steps to Prepare the Data

To prepare the data for data analysis I needed to:

1. Check for gaps and null values.
2. Set the Day column to an actual date time object.
3. Set that column to the index.
4. The revenue data needs to be stationary, removing trends.
5. Separating data into both training and test sets.

Checking for Null/Gaps

I check the beginning, end, null values. Then I subtracted the index from the day numerical value leaving every data point as one when stored in missing, the sum added to 731 which means every data point is equal to 1 and there are no missing values.

```
#Subtract index from days each value should be 1.
missing = df['Day'] - df.index
print("731 means no gaps:  ", missing.sum())
print(df.isnull().sum())
print("Duplicates:  ",df.duplicated().sum())
print(df.head())
print(df.tail())
df.columns = ['day', 'revenue']
```

```
731 means no gaps:   731
Day        0
Revenue    0
dtype: int64
Duplicates:   0
   Day   Revenue
0    1  0.000000
1    2  0.000793
2    3  0.825542
3    4  0.320332
4    5  1.082554
     Day    Revenue
726  727  16.931559
727  728  17.490666
728  729  16.803638
729  730  16.194813
730  731  16.620798
```

I set the date to 01-01-2020 because I was not provided with the date in the data dictionary. I then set the date to the index.

```
df['day'] = pd.to_datetime(df.index, unit='D', origin='2020-01-01')
df.set_index('day', inplace=True)
stationary = df
df
```

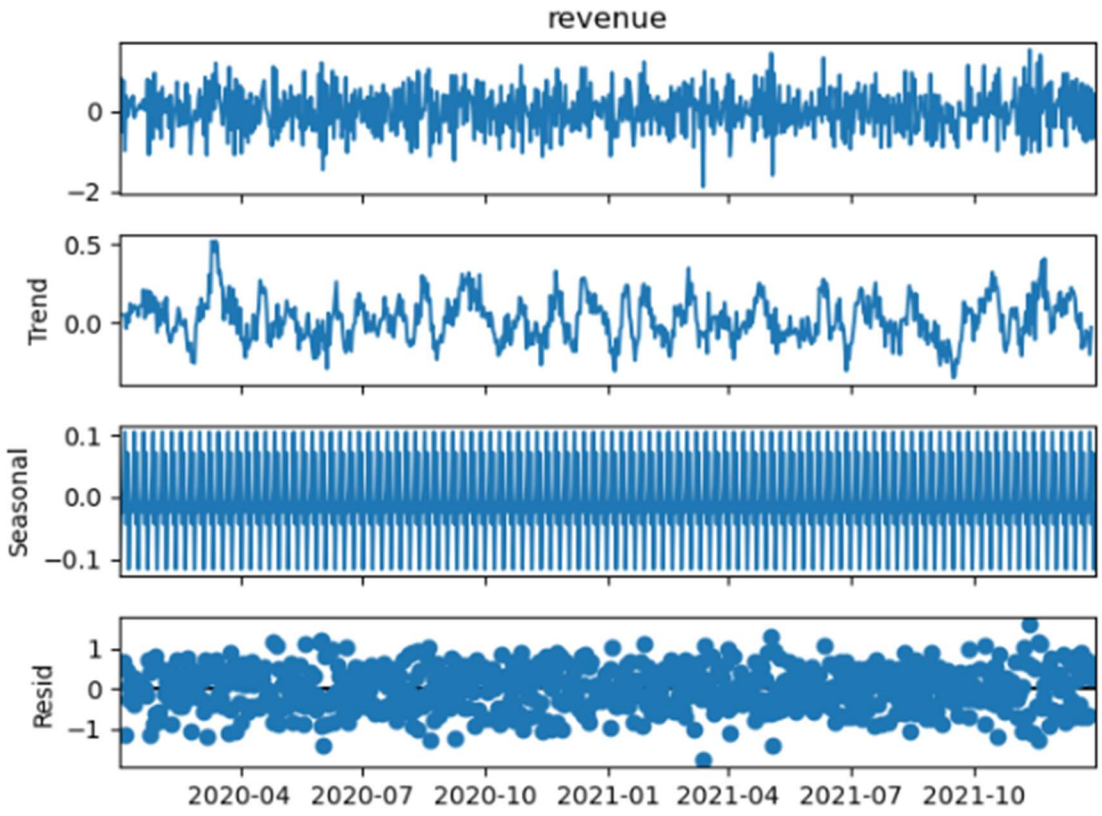
I create another data frame called stationary which becomes stationary because I take the difference from each successive data point. This creates one null value since it has no difference to subtract. This is why the. dropna() was included, to drop this empty value.

```
stationary = df.diff().dropna()
```

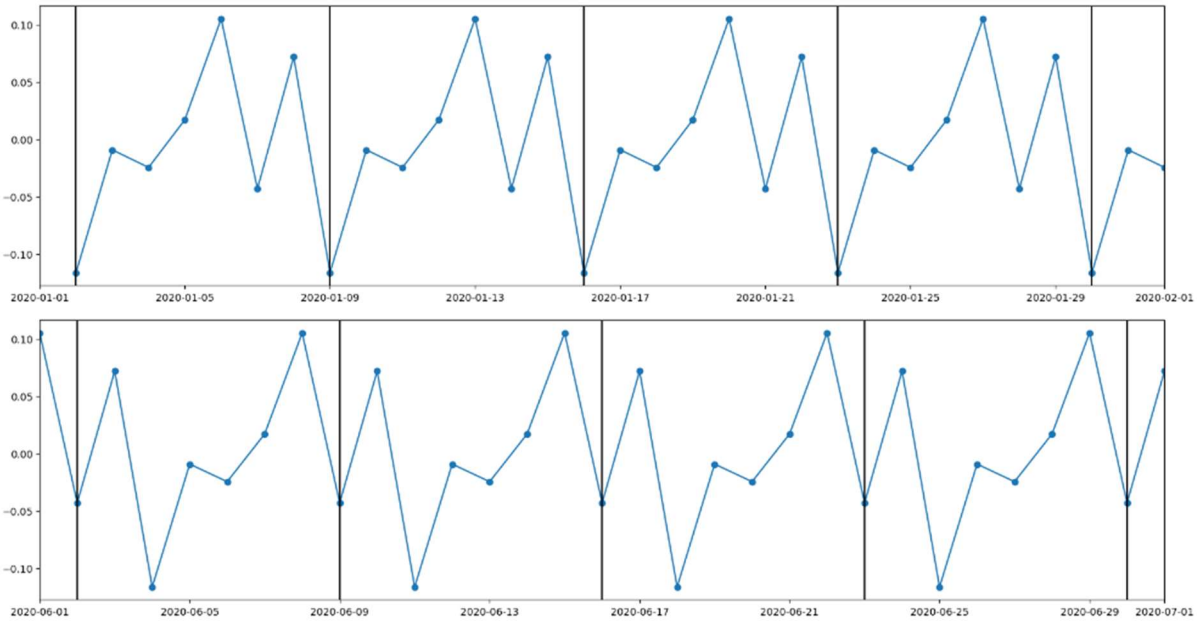
C5: Cleaned Data

I included the cleaned data attached as “task1_train.csv” and “task1_test.csv”.

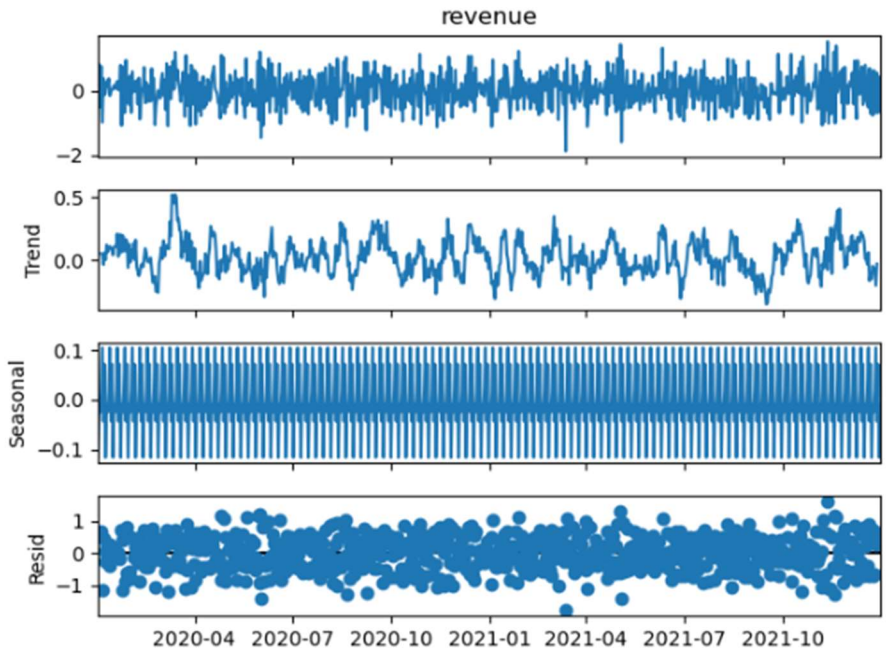
D1: Report Findings and Visualizations



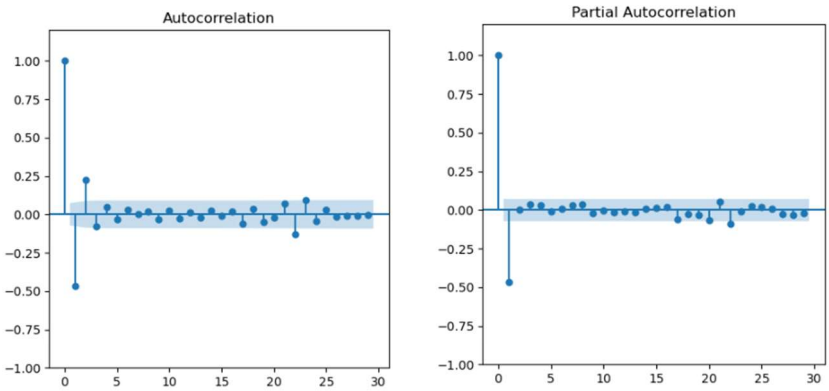
1. Seasonal Component: Above we can see a seasonal component in the stationary data. It has a repeating frequency or pattern of 7 days. We can see this in the two graphs below that show the pattern extends to the end of the period. While its possible to be a trend in our customer base, it is most likely how we measure our data. For instance, if recorded revenue has a cutoff time during the weekend and will spill over into Monday.



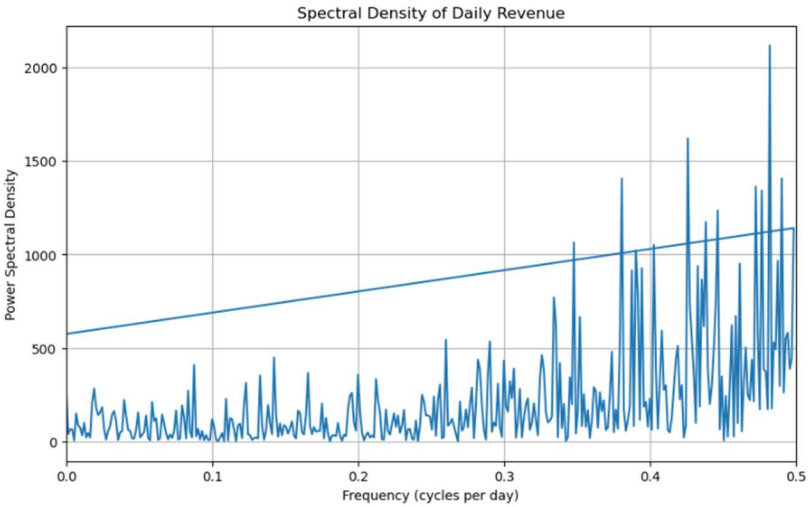
2. Trends: As you can see in the first figure above the overall trend was eliminated when the data was made stationary. The first figure provided before the data was made stationary showed an upwards trend of the data before it was cleaned.
3. Decomposed time series: The decomposed time series shows the data being stationary, a distinct seasonal trend and that some of the residuals fluctuate quite a bit.



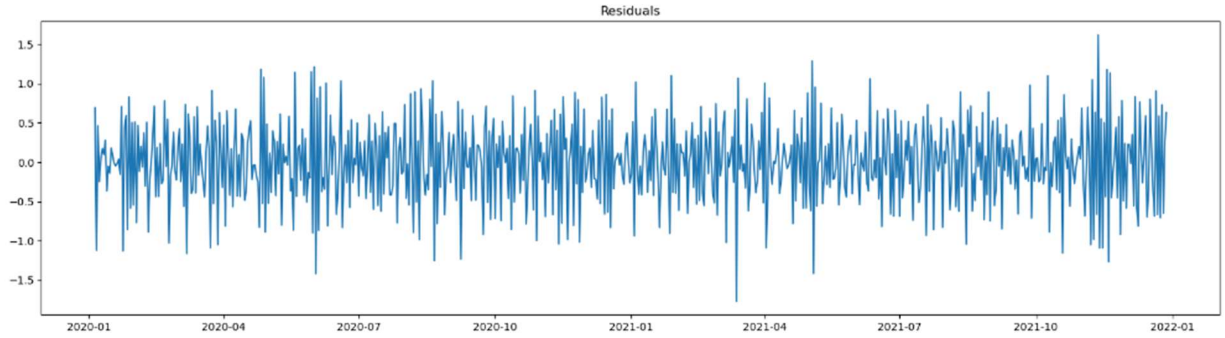
4. Autocorrelation Function: The ACF plot shows that the second value depends on the first and the third value depends on the second value showing that a one-day lag affects the current value. The PACF plot demonstrates this as well since it starts at one, drops to -.5 and becomes insignificant for all other lags beyond that point. This suggests an AR model is the best starting point.



5. Spectral Density: The plot shows from left to right the power or impact of a specific frequency, values closer to zero represent shorter periods while values closer to .5 represent longer periods such as quarterly or yearly patterns. This graph did not have any values above .5 and power leakage is manifested as a diagonal line across the graph.



6. Confirmation of Lack of Trends in Residuals: There are no specific residual trends in the data. There is an increased density of residuals towards the beginning and end but these to do continue to manifest and trail off.



D2: Arima Model

As suggested by the autocorrelation plots, I used a AR(1) model. I used this because the data seems to be mostly influenced by the previous days data.

```
train = train.asfreq('D')
test = test.asfreq('D')
model = ARIMA(train, order=(1, 0, 0), seasonal_order=(0, 1, 1, 7))

# Fit the model
model_fit = model.fit()

print(model_fit.summary())
```

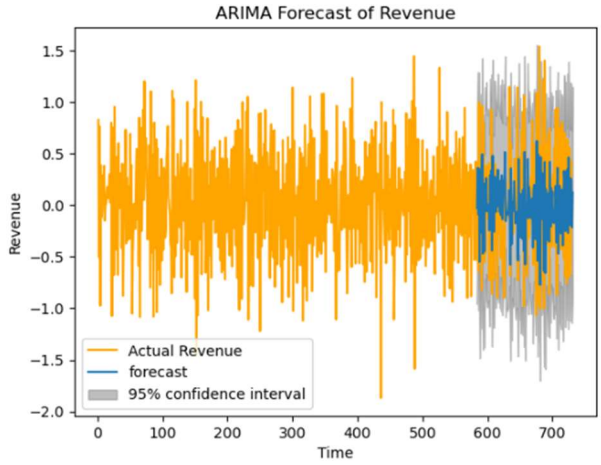
```
                                SARIMAX Results
==========================================================================================
Dep. Variable:                          revenue   No. Observations:                  584
Model:             ARIMA(1, 0, 0)x(0, 1, [1], 7)   Log Likelihood                -392.125
Date:                           Sun, 24 Mar 2024   AIC                            790.249
Time:                                   13:03:12   BIC                            803.323
Sample:                                 01-02-2020   HQIC                           795.348
                                      - 08-07-2021
Covariance Type:                            opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.4612      0.037    -12.307      0.000      -0.535      -0.388
ma.S.L7       -0.9894      0.034    -28.917      0.000      -1.057      -0.922
sigma2         0.2179      0.015     14.730      0.000       0.189       0.247
===================================================================================
Ljung-Box (L1) (Q):                   0.02   Jarque-Bera (JB):                 0.55
Prob(Q):                              0.88   Prob(JB):                         0.76
Heteroskedasticity (H):               0.93   Skew:                            -0.04
Prob(H) (two-sided):                  0.64   Kurtosis:                         2.88
===================================================================================
```

The coefficient formula can be simplified as this:

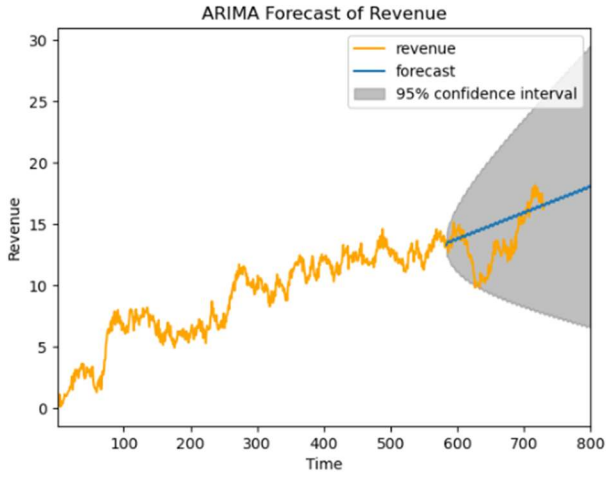
Current Days Revenue = -0.4612 * Previous Day Revenue - 0.9894 * Previous Weekly Error + Current Days Error

D3: Forecasting Using Arima Model

Testing the 146 rows of data against actual values with the stationary data didn't provide a good insight into the forecasting data.



To get a better understanding of the forecast I created another training set from the original data and plotted it.



As you can see the forecast is like a trend because it's a modified version of the mean of data and the confidence interval. The confidence interval is that there is a 95% chance the value will be located inside the shaded area, it works like a range of possibilities and as the days increase so does the interval since each day builds on another.

D4: Output and Calculations

Dickey Fuller test showing data is not stationary before I made adjustments.

```
ADF Statistic: -1.924612157310183
p-value: 0.3205728150793967
Critical Values:
        1%: -3.4393520240470554
        5%: -2.8655128165959236
        10%: -2.5688855736949163
```

Dickey Fuller test showing data is stationary after I made adjustments.

```
ADF Statistic: -44.87452719387599
p-value: 0.0
Critical Values:
        1%: -3.4393520240470554
        5%: -2.8655128165959236
        10%: -2.5688855736949163
```

Summary Results of Arima Model:

```
                                SARIMAX Results
==========================================================================================
Dep. Variable:                            revenue   No. Observations:                  584
Model:             ARIMA(1, 0, 0)x(0, 1, [1], 7)   Log Likelihood                -392.125
Date:                           Sun, 24 Mar 2024   AIC                            790.249
Time:                                   13:48:29   BIC                            803.323
Sample:                                 01-02-2020 HQIC                           795.348
                                      - 08-07-2021
Covariance Type:                              opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.4612      0.037    -12.307      0.000      -0.535      -0.388
ma.S.L7       -0.9894      0.034    -28.917      0.000      -1.057      -0.922
sigma2         0.2179      0.015     14.730      0.000       0.189       0.247
===================================================================================
Ljung-Box (L1) (Q):                   0.02   Jarque-Bera (JB):                 0.55
Prob(Q):                              0.88   Prob(JB):                         0.76
Heteroskedasticity (H):               0.93   Skew:                            -0.04
Prob(H) (two-sided):                  0.64   Kurtosis:                         2.88
===================================================================================
```

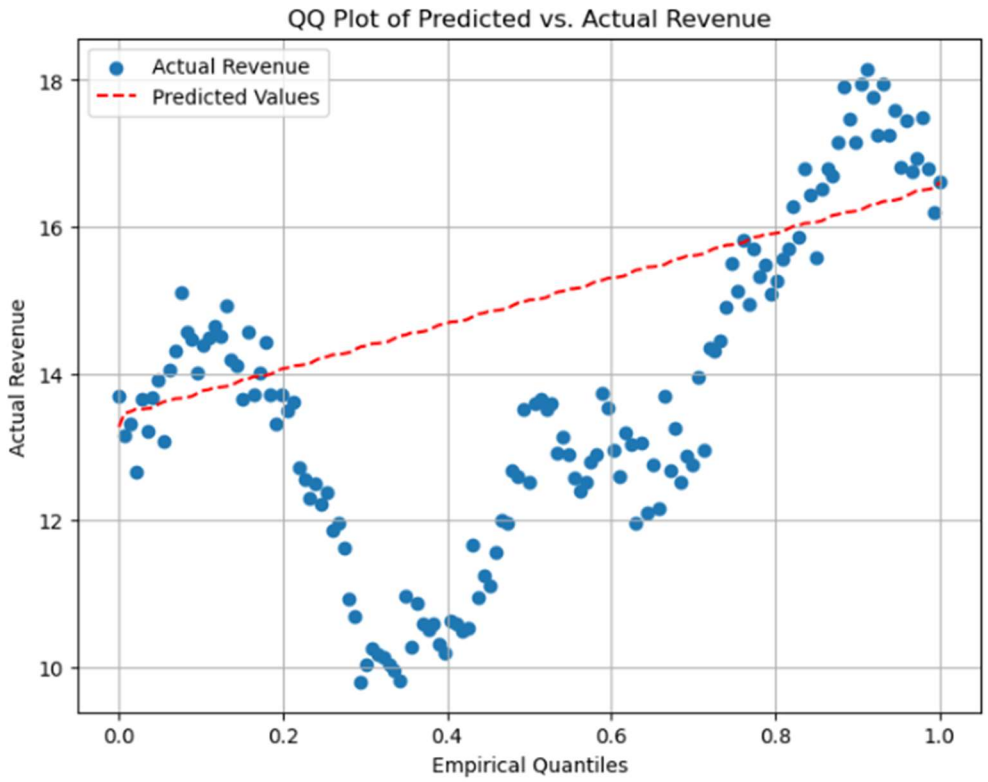
RMSE of Forecast again testing data:

```
#Calculate RMSE
predicted_revenue_testing = model_fit.forecast(steps=147)
rmse_testing = np.sqrt(mean_squared_error(test1, predicted_revenue_testing))

print("RMSE (Testing):", rmse_testing)
average_revenue = test1.revenue.mean()
percentage_error = (rmse_testing / average_revenue) *100
print(f"Percentage Error: {percentage_error:.2f}%")
```

```
RMSE (Testing): 2.2498366127301854
Percentage Error: 16.49%
```

QQ plot



D5: Provide the Code

Attached is the Python code “Task1a.ipynb”

E1: Results

Selection of Arima Model:

I chose a Arima(1, 0, 0) with a seasonal order of 7 days because of the seasonal pattern I found. I choose this based on the autocorrelation information that was found where current values are dependent on past values using a 1-day lag. The equation can be summarized as this: Current Days Revenue = -0.4612 * Previous Day Revenue - 0.9894 * Previous Weekly Error + Current Days Error

Prediction Interval of the Forecast:

The prediction interval of the forecast is 216 steps in total. This was from the 80% training data and 20% test data.

Justification of Forecast Length:

Because of the long amount of time this covered which was 2 years, while only have 731 data points I chose to use an 80/20 split because I wanted to try and use the maximum amount of training data while still making a decent overall prediction time. 216 days is about 60% of a full year, so it's a significant amount of time to be forecasted. The split of the data as 80% training and 20% testing is on the maximum size of training while still following a common practice from the literature I've read or code I have observed being somewhere between 20%-30% testing data.

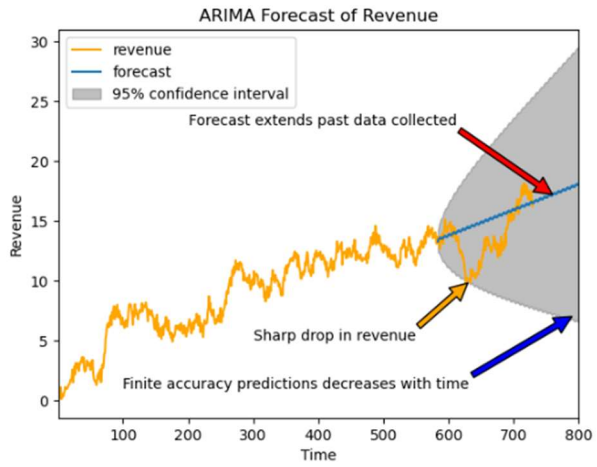
Model Evaluation Procedure/Metric:

To evaluate the model, I first used visualization to gain a better understanding of the data. Plotting the stationary test data did not reveal much useful information so I turned to plotting the original data which was not stationary. This revealed the forecast and the confidence interval or the range of revenue each step would have. I followed this up by calculating the RMSE which

was 2.25. When decoding what this means in raw numbers of the data it means the predicted forecast is off by 16.49%. This was found by dividing 2.25 by the average revenue which was 13.65 and multiplying that by 100.

The goal of this research was to identify seasonal trends which may help in our companies' efforts to retain customers. The only seasonal trend which was identified is a weekly trend which may have more to do with how our company collects and reports its revenue data. For example, a drop in revenue on Sunday is followed a large increase in revenue on Monday which may mean that revenue data recorded afterhours on Sunday may be reported on Monday. No yearly, quarterly or monthly seasonal trends were identified. This may be due to the small amount of data to work with. At around 09/08/2021 in the data or around 620 days into the data a steep decline in revenue which continues to drop outside of the normal daily lag correlation. This warrants further investigation and reasons for this drop should be identified in future research as this specific drop is exactly what the company is looking for to identify trends of churn, even if I can't directly identify it as a seasonal trend. One thing to note is that this drop is within the 95% confidence interval, show it is inside our forecast limits even if extremely unlikely before a correction occurs.

E2: Annotated Visualizations



E3: Course of Action

My recommendation is to first and foremost continue to collect more data to build a better prediction model going forward. Next, we should address the seasonal and weekly trend issue. To address this issue, my recommendation is to make sure our reporting team or automated reporting is consistent. Going forward we could also try to use other modeling techniques that may increase the forecasting accuracy to better detect the seasonality or weekly trends. Finally, and most importantly, we should investigate the sharp revenue drop to see if that can be explained by market forces or our own actions. If it is market forces, we can investigate how we can possibly shield our company from this in the future. If it was due to our own actions or something we can change in our business model, investigating this will provide us with further actions we can take to prevent churn rates using our revenue data.

Third-Party Code

No third-party code was used.

References

Geeks for Geeks (n.d.). Matplotlib.pyplot.annotate() in Python. Retrieved from https://www.geeksforgeeks.org/matplotlib-pyplot-annotate-in-python/