# D213 Advanced Data Analytics Performance Task 1

Sean Simmons

WDU Data Analytics

MSDA D213

March 2023

## Part I: Research Question

### A. Describe the purpose of this data analysis by doing the following:

#### 1. Summarize one research question that is relevant to a real-world organizational situation captured in the selected data set and that you will answer using time series modeling techniques.

#### 2. Define the objectives or goals of the data analysis. Ensure that your objectives or goals are reasonable within the scope of the scenario and are represented in the available data.

1. One research question relevant to the telecommunications company is, “Can you use the daily revenue data from the past 2 years to predict revenue 250 days into the future? “ I will use the telecommunications company's time series dataset and time series modeling techniques to answer this question.

2. The goal of this analysis is to build a predictive model that will forecast future revenue by day using the daily revenue history from the past 2 years. This model will analyze the revenue by day to identify trends, seasonality, and a variety of factors related to time. This goal is reasonable within the scope of the scenario and represented in the data because we are given revenue data by day, called a time series.

## Part II: Method Justification

### B. Summarize the assumptions of a time series model including stationarity and autocorrelated data.

The assumptions of a time series model are related to the data set itself. Firstly, it is assumed the data is stationary. For stationarity, it is assumed the series is normally distributed and the mean and variance are constant over the given period of time; aka it is not changing, growing, shrinking, and the variance and autocorrelation are constant. In summary the distribution does not change. As stated, for autocorrelation measures the degree of similarity and is assumed to be constant. Also, residuals are assumed to NOT be autocorrelated and there are no outliers in the series. Finally, it is assumed the error term is randomly distributed (Prabhakaran 2019).

## Part III: Data Preparation

## C. Summarize the data cleaning process by doing the following:

### 1. Provide a line graph visualizing the realization of the time series.

### 2. Describe the time step formatting of the realization, including any gaps in measurement and the length of the sequence.

### 3. Evaluate the stationarity of the time series.

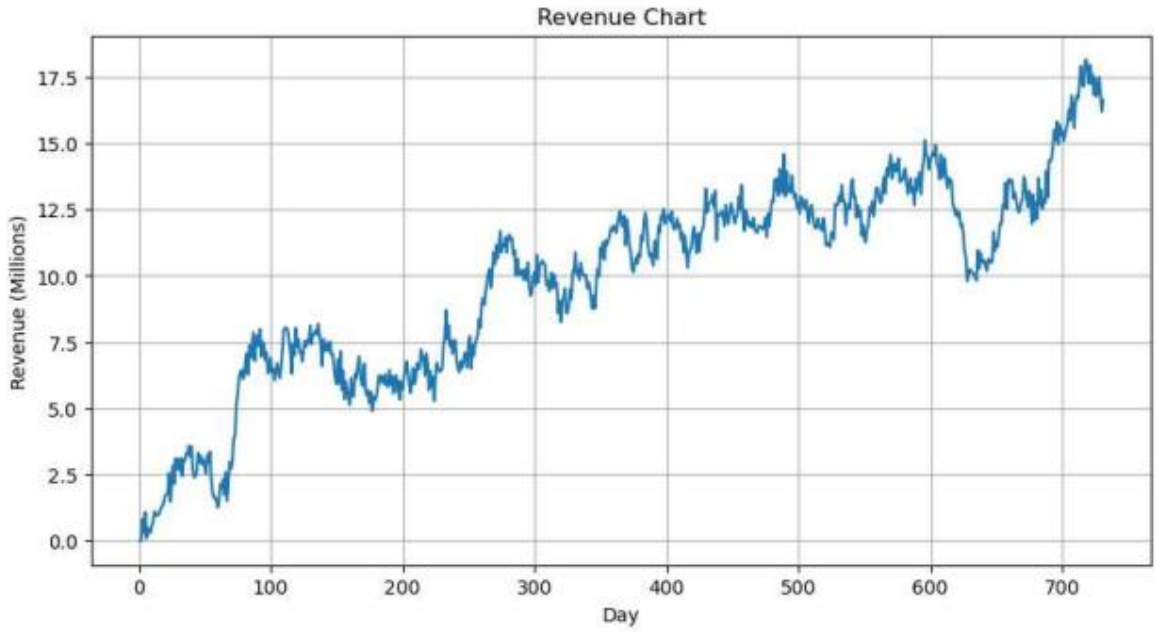
### 4. Explain the steps used to prepare the data for analysis, including the training and test set split.

### 5. Provide a copy of the cleaned dataset.

1. The line graph visualizing the realization of the time series is found below:

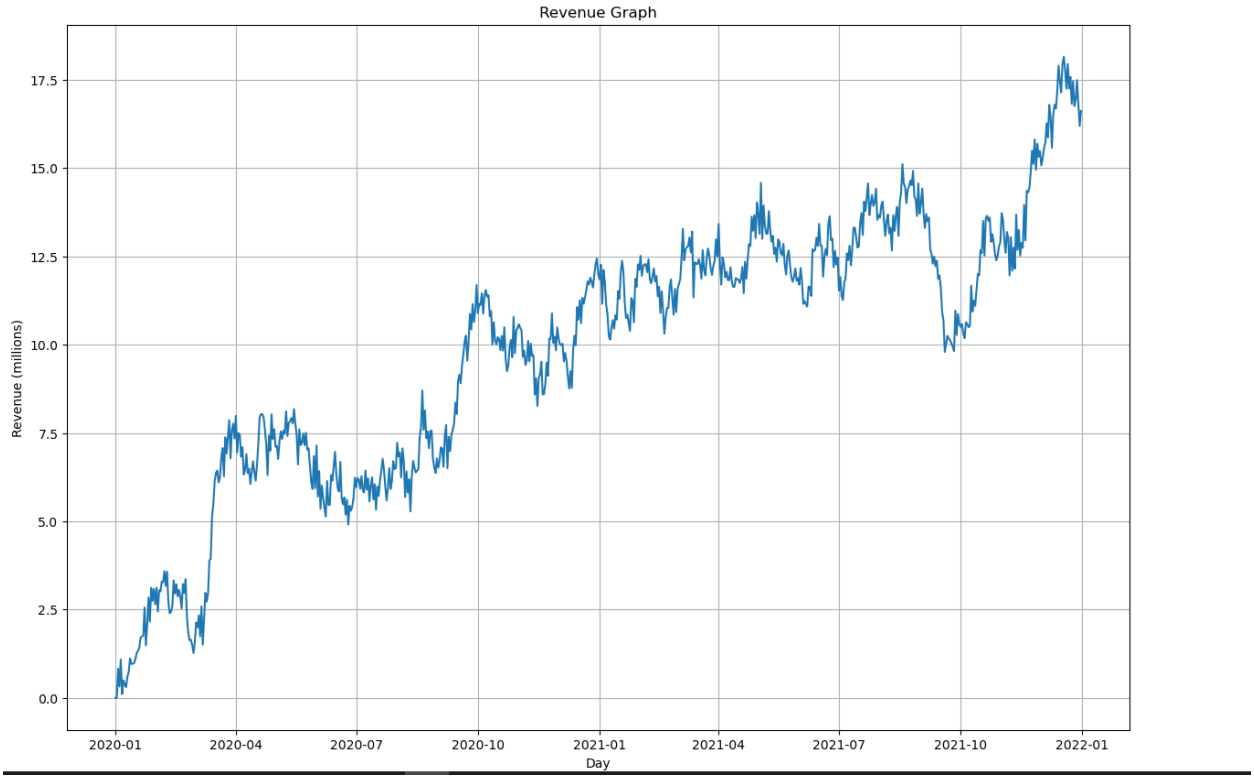
Realization before indexing

```
#Create a line graph to visualize the realization of the time series.
mpl.figure(figsize=(10,5))
mpl.plot(df.Revenue)
mpl.title('Revenue Chart')
mpl.xlabel('Day')
mpl.ylabel('Revenue (Millions)')
mpl.grid(True)
mpl.show()
```



Realization after indexing to date format

```
mpl.figure(figsize=(15,10))
mpl.plot(df.Revenue)
mpl.title('Revenue Graph')
mpl.xlabel('Day')
mpl.ylabel('Revenue (millions)')
mpl.grid(True)
mpl.show()
```




2. The time step formatting of the realization, including the gaps in measurement and length, is as follows: There are no gaps present and we have 731 days. We converted it to a time series after checking the data for missing values and NA values.

```
Floats
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 731 entries, 0 to 730
Data columns (total 1 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   Revenue  731 non-null    float64
dtypes: float64(1)
memory usage: 5.8 KB
None
Integers
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 731 entries, 0 to 730
Data columns (total 1 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Day     731 non-null    int64
dtypes: int64(1)
memory usage: 5.8 KB
None
Objects
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 731 entries, 0 to 730
Empty DataFrameNone
Dataset Information
<bound method DataFrame.info of      Day    Revenue
0      1   0.000000
1      2   0.000793
2      3   0.825542
3      4   0.320332
4      5   1.082554
..   ...        ...
726  727  16.931559
727  728  17.490666
728  729  16.803638
729  730  16.194813
730  731  16.620798

[731 rows x 2 columns]>
```

[122…]
```
#Check for missing values
df.isna().sum()
```

t[122]:
```
Day        0
Revenue    0
dtype: int64
```

```
#View summary statistics
print("Means")
print(df.mean())
print("Medians")
print(df.median())
```

e/Documents/WGU/D213/D213_Task1_Code.ipynb

D213_Task1_Code

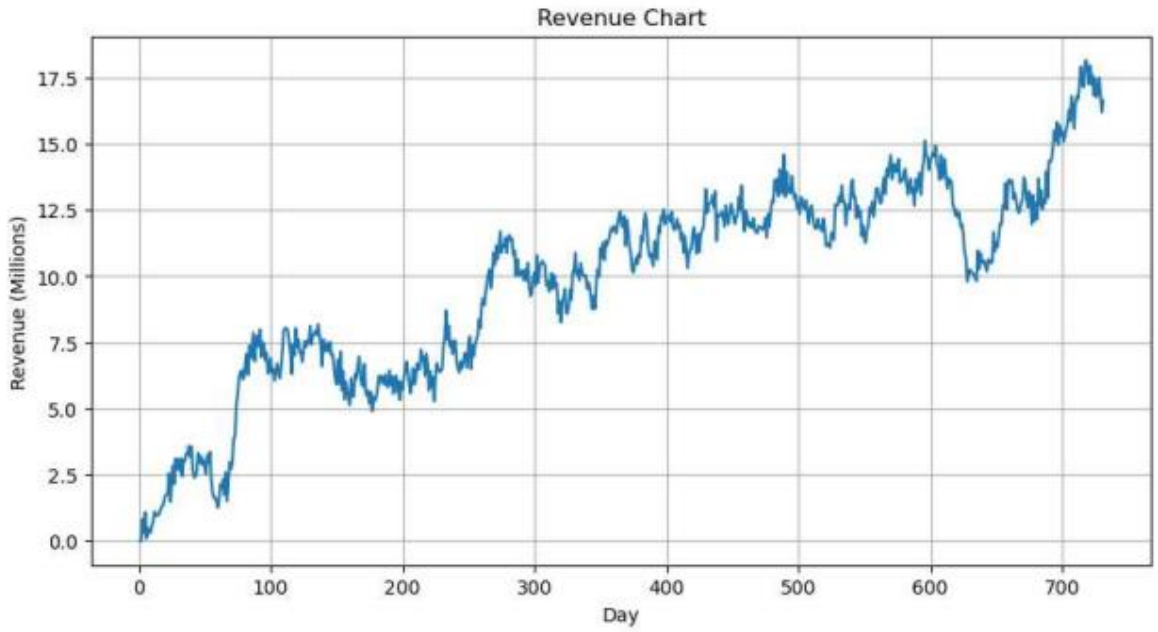
```
Means
Day        366.000000
Revenue      9.822901
dtype: float64
Medians
Day        366.000000
Revenue     10.785571
dtype: float64
```

```
#Index our Day and Revenue
X=df.Day
Y=df.Revenue

df.set_index('Day', inplace=True)
df.index
```

```
Int64Index([  1,   2,   3,   4,   5,   6,   7,   8,   9,  10,
            ...
            722, 723, 724, 725, 726, 727, 728, 729, 730, 731],
           dtype='int64', name='Day', length=731)
```

```
#Create a line graph to visualize the realization of the time series.
mpl.figure(figsize=(10,5))
mpl.plot(df.Revenue)
mpl.title('Revenue Chart')
mpl.xlabel('Day')
mpl.ylabel('Revenue (Millions)')
mpl.grid(True)
mpl.show()
```



```
#C2- Time step formatting
from datetime import datetime
df['Date'] = (pd.date_range(start=datetime(2020, 1, 1),
    periods=df.shape[0], freq='24H'))
df.set_index('Date',inplace=True)
df
```

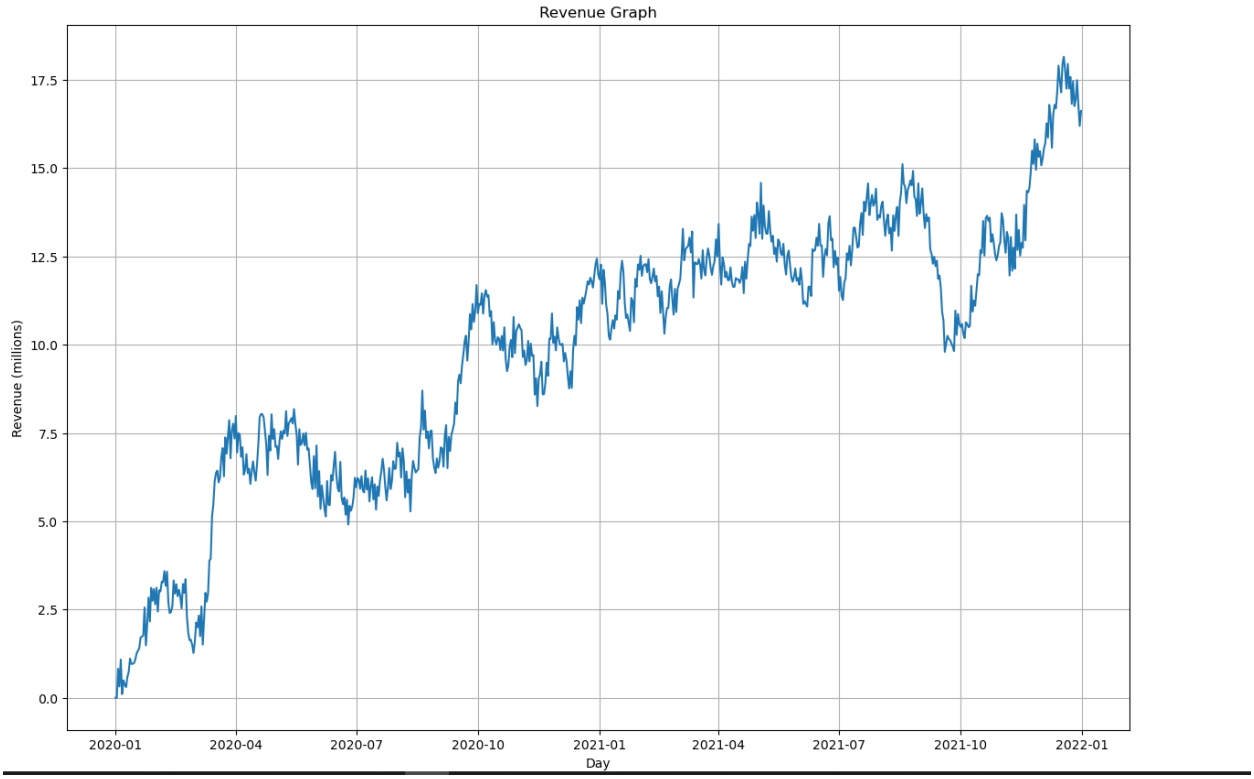
ee/Documents/WGU/D213/D213_Task1_Code.ipynb

D213_Task1_Code

| Date | Revenue |
|---|---|
| **2020-01-01** | 0.000000 |
| **2020-01-02** | 0.000793 |
| **2020-01-03** | 0.825542 |
| **2020-01-04** | 0.320332 |
| **2020-01-05** | 1.082554 |
| **...** | ... |
| **2021-12-27** | 16.931559 |
| **2021-12-28** | 17.490666 |
| **2021-12-29** | 16.803638 |
| **2021-12-30** | 16.194813 |
| **2021-12-31** | 16.620798 |

731 rows × 1 columns

```
mpl.figure(figsize=(15,10))
mpl.plot(df.Revenue)
mpl.title('Revenue Graph')
mpl.xlabel('Day')
mpl.ylabel('Revenue (millions)')
mpl.grid(True)
mpl.show()
```




3. The stationarity of the time series was tested with the ADFuller test. As seen below, our test statistics value is -1.92, which is not below the critical values and our p value is larger than .05, so this time series is not stationary. To make it stationary, I used differencing to remove the linear trend found in graphing. Only one order of differencing was required to create stationarity and it is used to help stabilize the mean. The differencing method here was able to eliminate the linear trend. The stationarity was tested with an ACF plot and if it is stationary, the plot should drop to 0 very quickly. I included before and after plots so we could verify our methods worked and it indeed did work (Brownlee 2017).

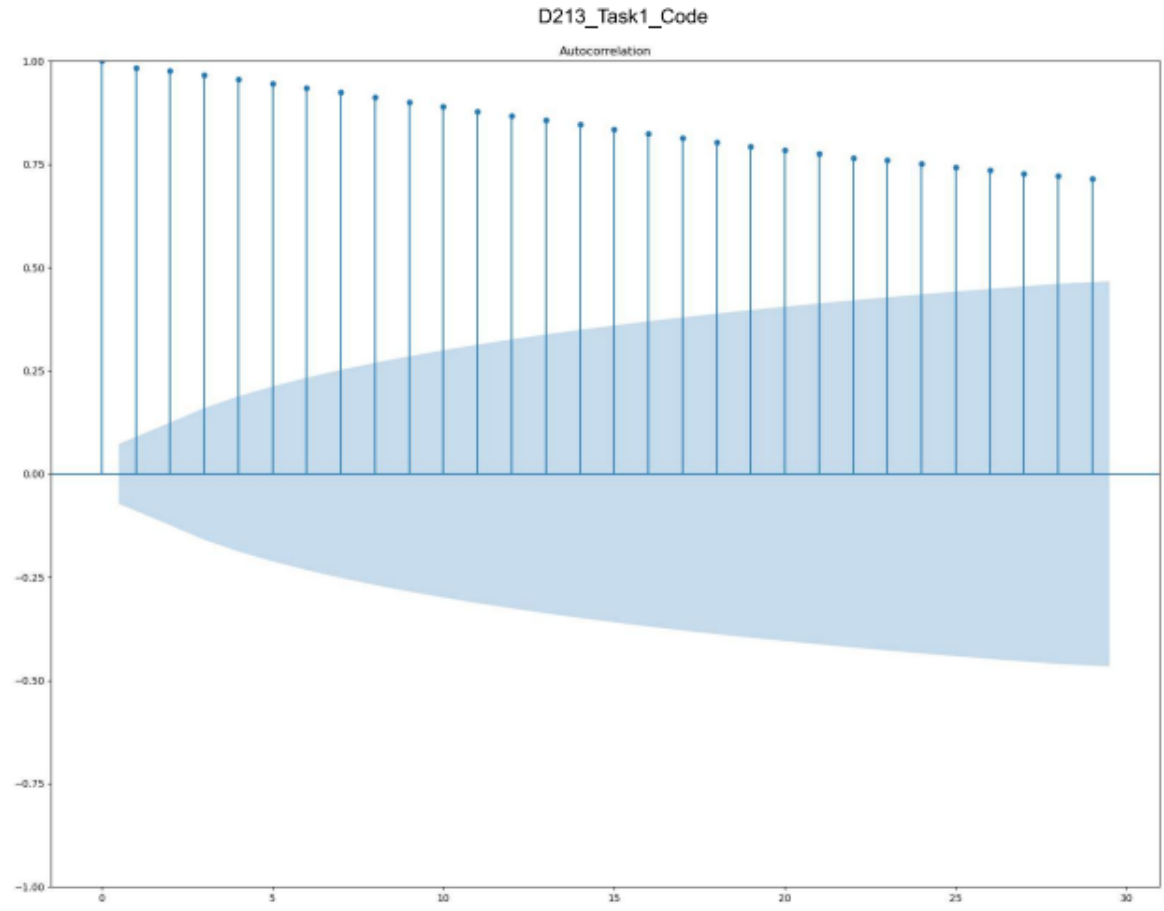
```
#C3 Let's see the stationarity of the time series using the ADFuller Test
result = adfuller(df['Revenue'])
print("Test statistics: ", result[0])
print("p-value: ", result[1])
print("Critical values: ", result[4])

#if the p-value is above .5 we fail to reject the null hypothesis, the time series is
#If the p-value is below .5, we reject the null hypothesis, the time series is statior
#Here, the p-value is below .5, but the test statistic is not below the critical value
```

```
Test statistics:  -1.924612157310183
p-value:  0.3205728150793967
Critical values:  {'1%': -3.4393520240470554, '5%': -2.8655128165959236, '10%': -2.56
88855736949163}
```

```
#Initial ACF
af = plot_acf(df)
af.set_size_inches(20, 15)
mpl.show(af)
```





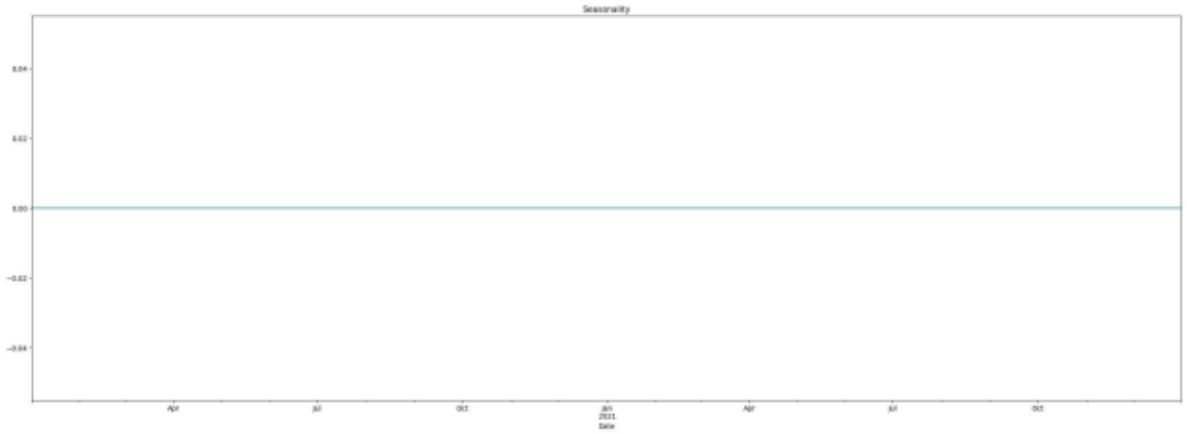
```
#Use the built in differencing function
df = df.diff().dropna()
df
```

| Date | Revenue |
|---|---|
| **2020-01-02** | 0.000793 |
| **2020-01-03** | 0.824749 |
| **2020-01-04** | -0.505210 |
| **2020-01-05** | 0.762222 |
| **2020-01-06** | -0.974900 |
| **...** | ... |
| **2021-12-27** | 0.170280 |
| **2021-12-28** | 0.559108 |
| **2021-12-29** | -0.687028 |
| **2021-12-30** | -0.608824 |
| **2021-12-31** | 0.425985 |

730 rows × 1 columns

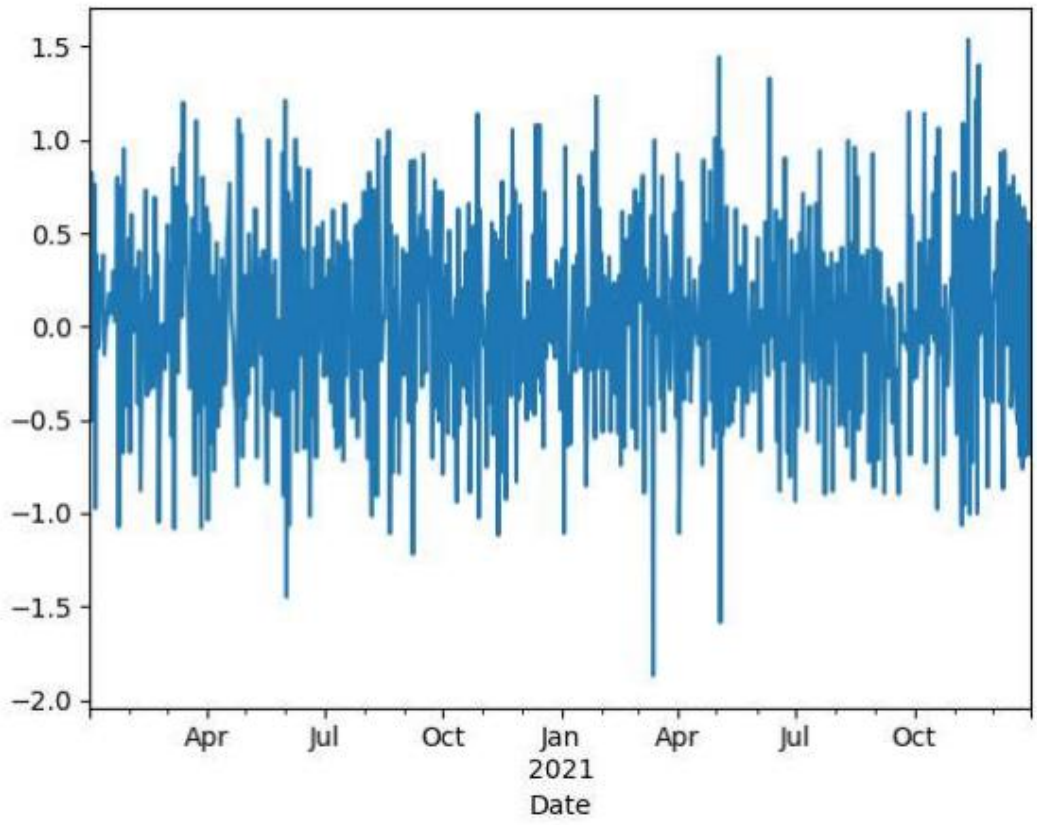
```
#D1a: Lets evaluate our model
from statsmodels.tsa.seasonal import seasonal_decompose
result = seasonal_decompose(df['Revenue'], model='additive', period=1)
mpl.title('Seasonality')
result.seasonal.plot(figsize=(30, 10))
#The straight line here shows us there is no seasonal trend to our data
```

```
<AxesSubplot:title={'center':'Seasonality'}, xlabel='Date'>
```



```
#D1b: Let's look at trends
result.trend.plot()
```

```
<AxesSubplot:xlabel='Date'>
```

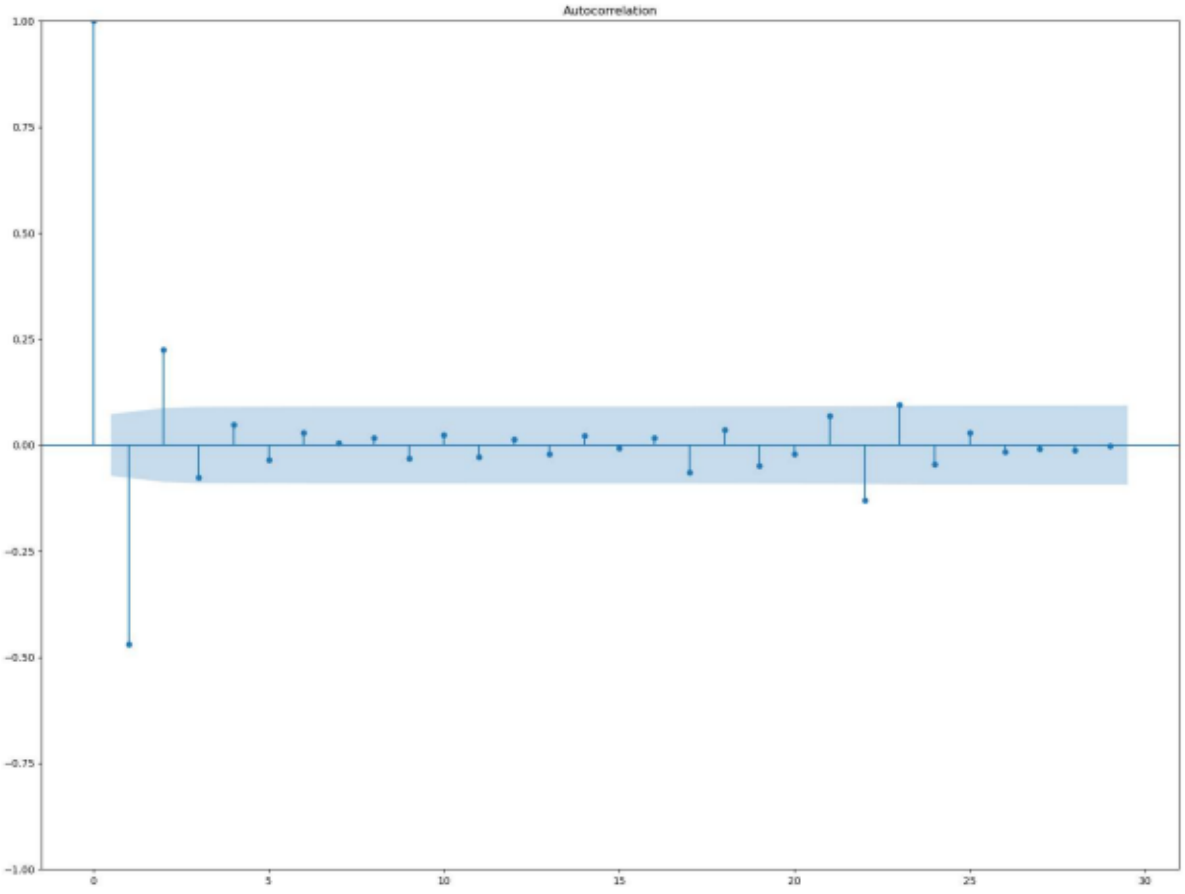


```
#D1c: Autocorrelation function
af = plot_acf(df)
```

e/Documents/WGU/D213/D213_Task1_Code.ipynb 7/21

D213_Task1_Code

```
af.set_size_inches(20, 15)
mpl.show(af)
#If we are stationary there should be a quick drop to 0
```



4. The steps used to prepare the data for analysis, including the training and test set, is as follows:
    a. Import the data into my coding environment using pandas (in python)
    b. View the data type and summary information to prepare for the modeling
    c. Perform EDA with the above step and check summary statistics, such as mean/median.
    d. Check for missing and null values
    e. Convert Day to Date
    f. Check for stationarity testing time series Adfuller test (verify with ACF)
    g. Perform differencing on the data to make it stationary
    h. Perform ACF plotting and trend analysis to verify stationarity.
    i. Split the data into training (75%) and test (25%) sets.
    j. Extract the prepared data set to csv file
5. The cleaned dataset has been provided as "D213_Task1_clean.csv" The training and test sets have been provided as well and appropriately named "Task1_Train.csv" and "Task1_test.csv."

## Part IV: Model Identification and Analysis

## D. Analyze the time series dataset by doing the following:

### 1. Report the annotated findings with visualizations of your data analysis, including the following elements:

- **the presence or lack of a seasonal component**
- **trends**
- **auto correlation function**
- **spectral density**
- **the decomposed time series**
- **confirmation of the lack of trends in the residuals of the decomposed series**

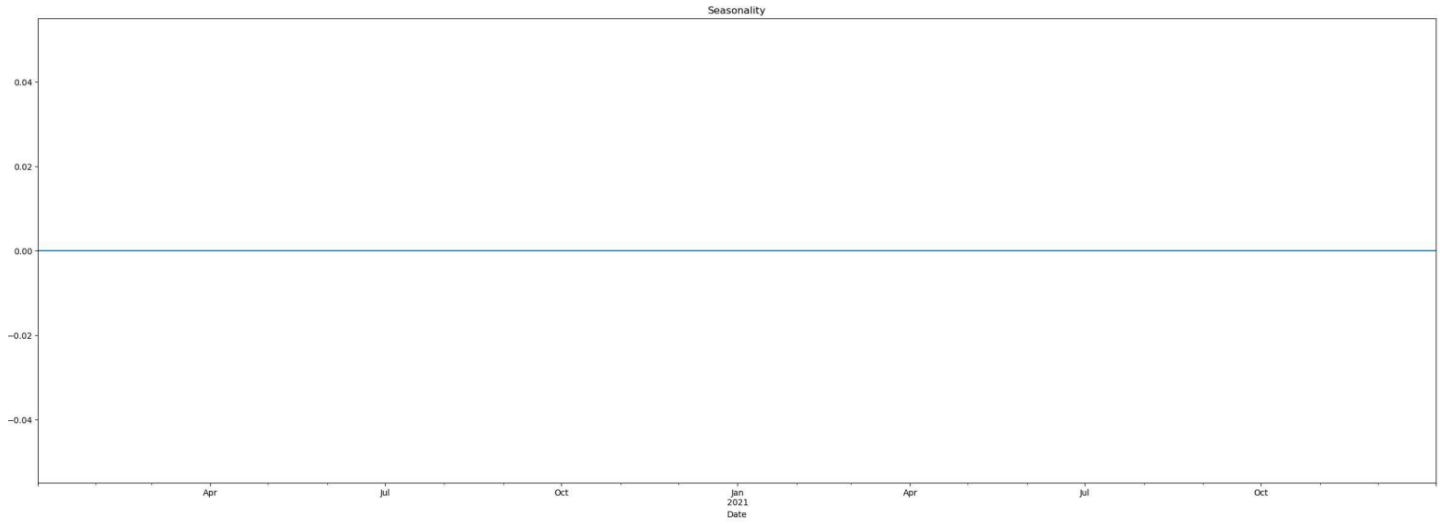
**2. Identify an autoregressive integrated moving average (ARIMA) model that takes into account the observed trend and seasonality of the time series data.**

**3. Perform a forecast using the derived ARIMA model.**

**4. Provide the output and calculations of the analysis you performed.**

**5. Provide the code used to support the implementation of the time series model.**

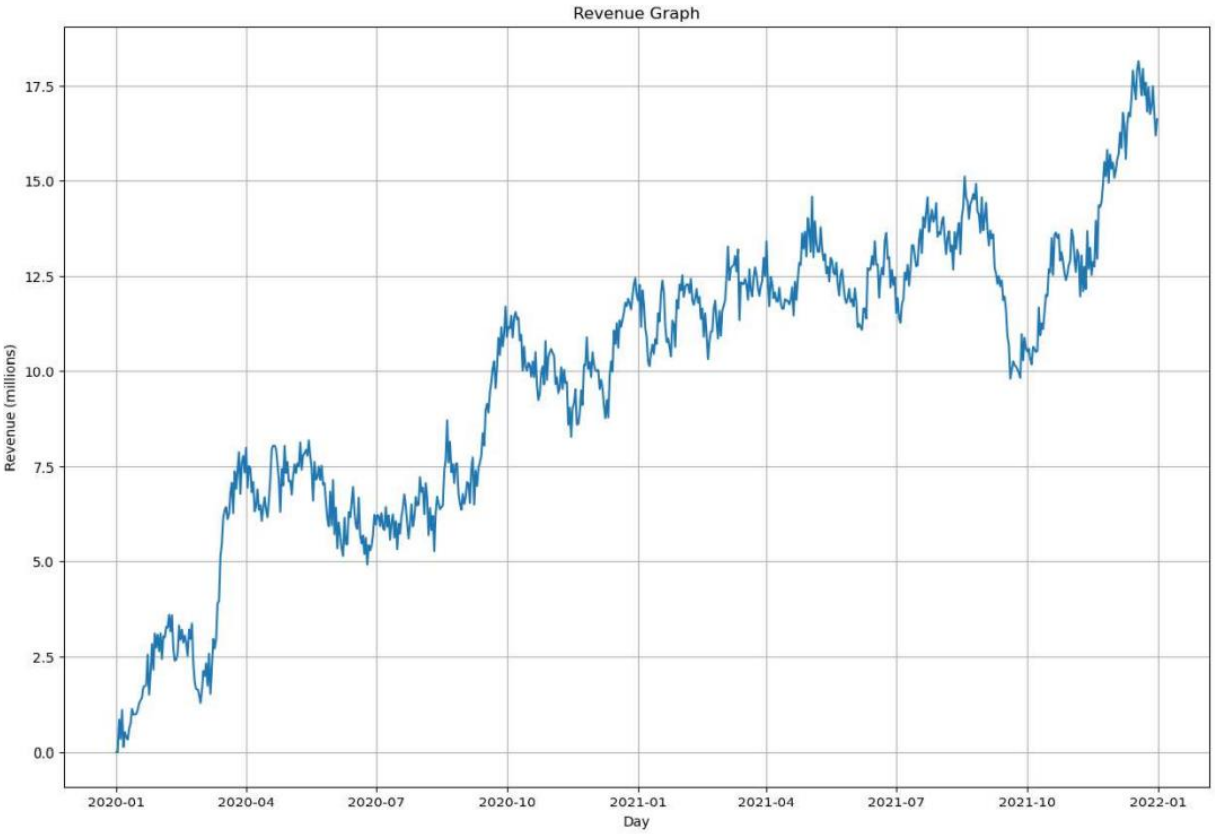
1. The annotated findings with visualizations are below:
    a. The presence or lack of a seasonal component- There is no seasonality in this dataset and it was tested using the seasonality function:




Seasonality. The blue line represents a seasonal trend. If it remains at zero, there is no seasonal trend and if it jumps away from zero it indicates a seasonal trend present at the jump.

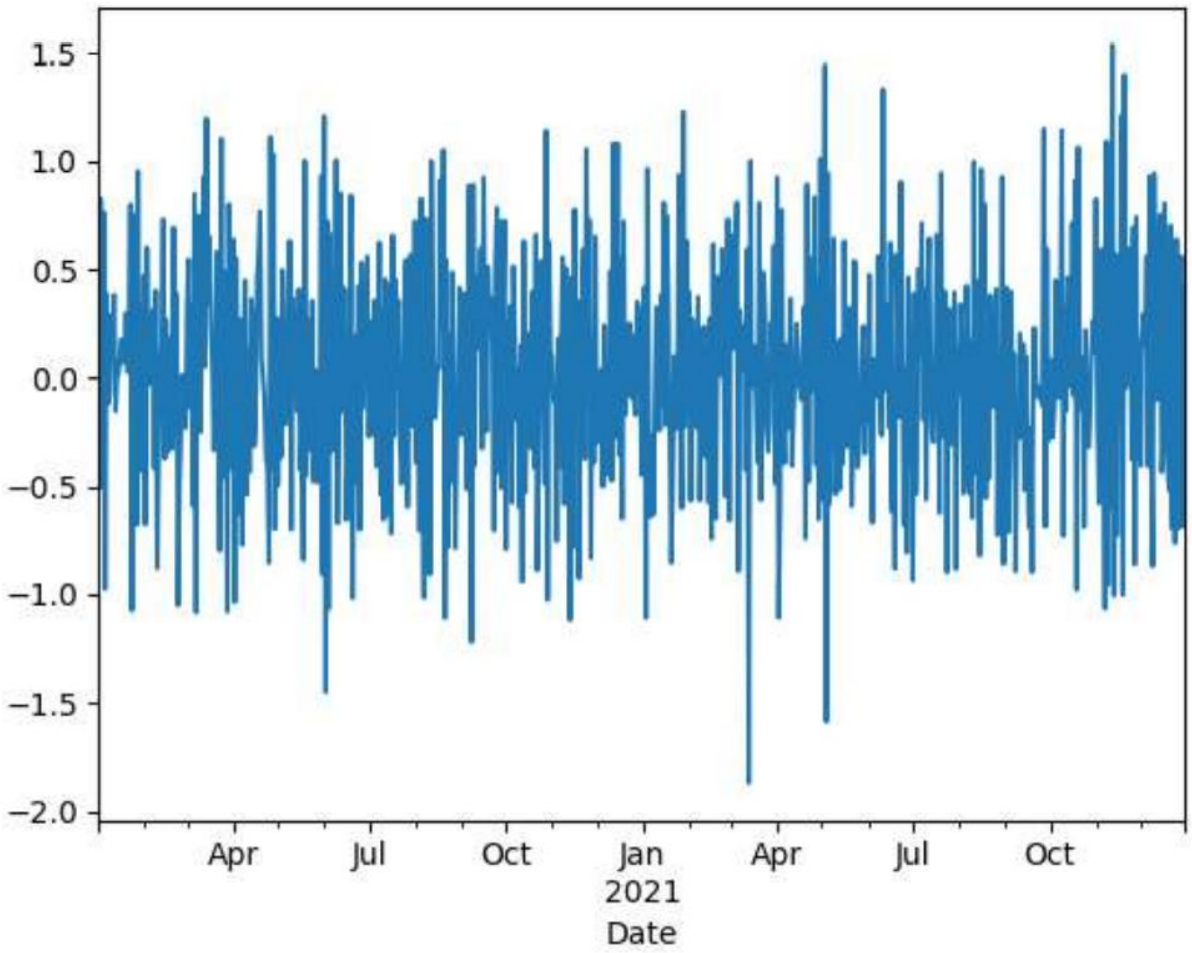
b. Trends- There was a linear trend before we differenced the time series, now there is no linear trend. Please see the before and after image of this trend.

Before Differencing




Revenue Graph. The blue line represents the revenue in millions of dollars and it is increasing in a linear fashion from 0 to 17.5.

Revenue (Millions) After differencing

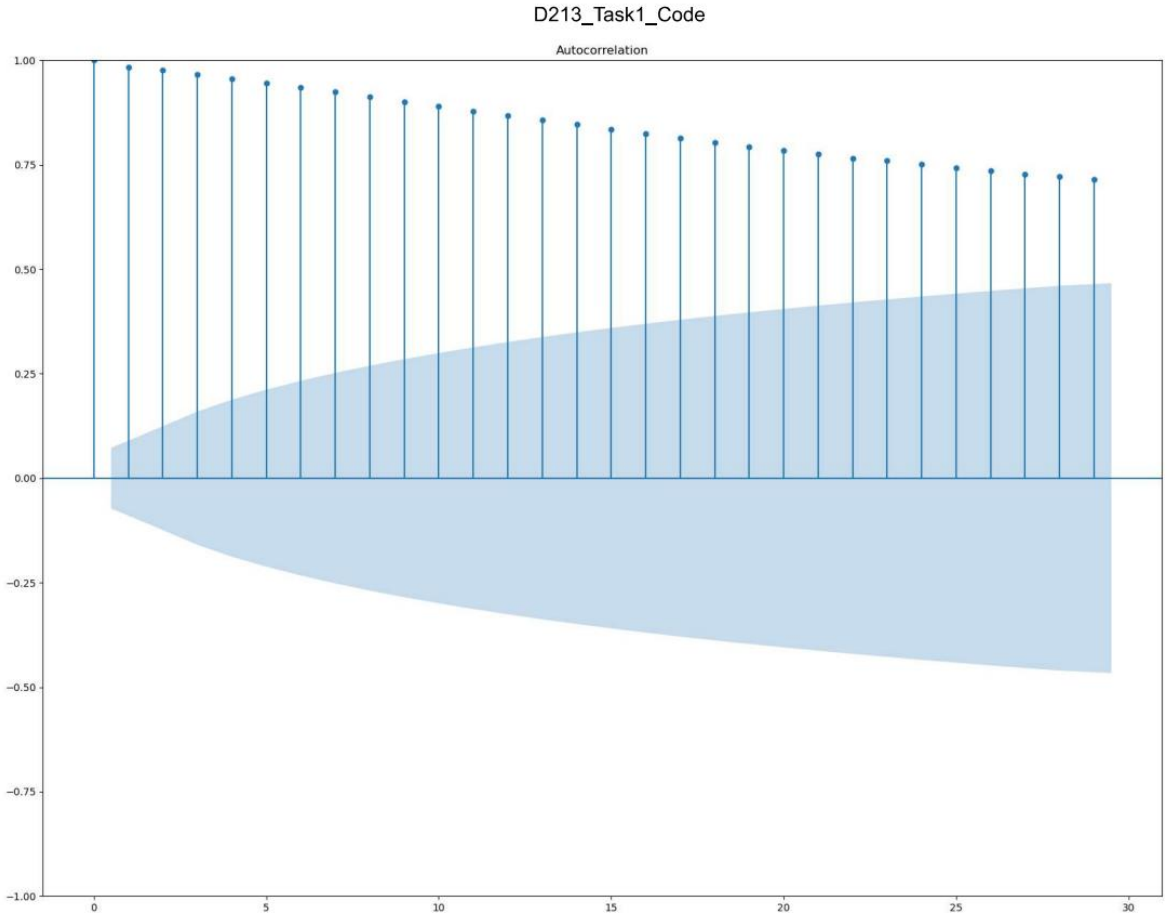



Revenue (Millions). This is a trend analysis plot with revenue in blue and the time series across the x axis. If there is a linear trend we expect to see large and consistent changes across the y axis that remain. Instead, we still have a steady line at 0 with minor peaks and valleys that remain surrounding 0 indicating we do not have a growing linear trend.

c. Autocorrelation Function (ACF)- A plot of the autocorrelation of a time series. The autocorrelation shows how the data is correlating across periods. A value between 0 and 1, represents a positive correlation, and negative values would

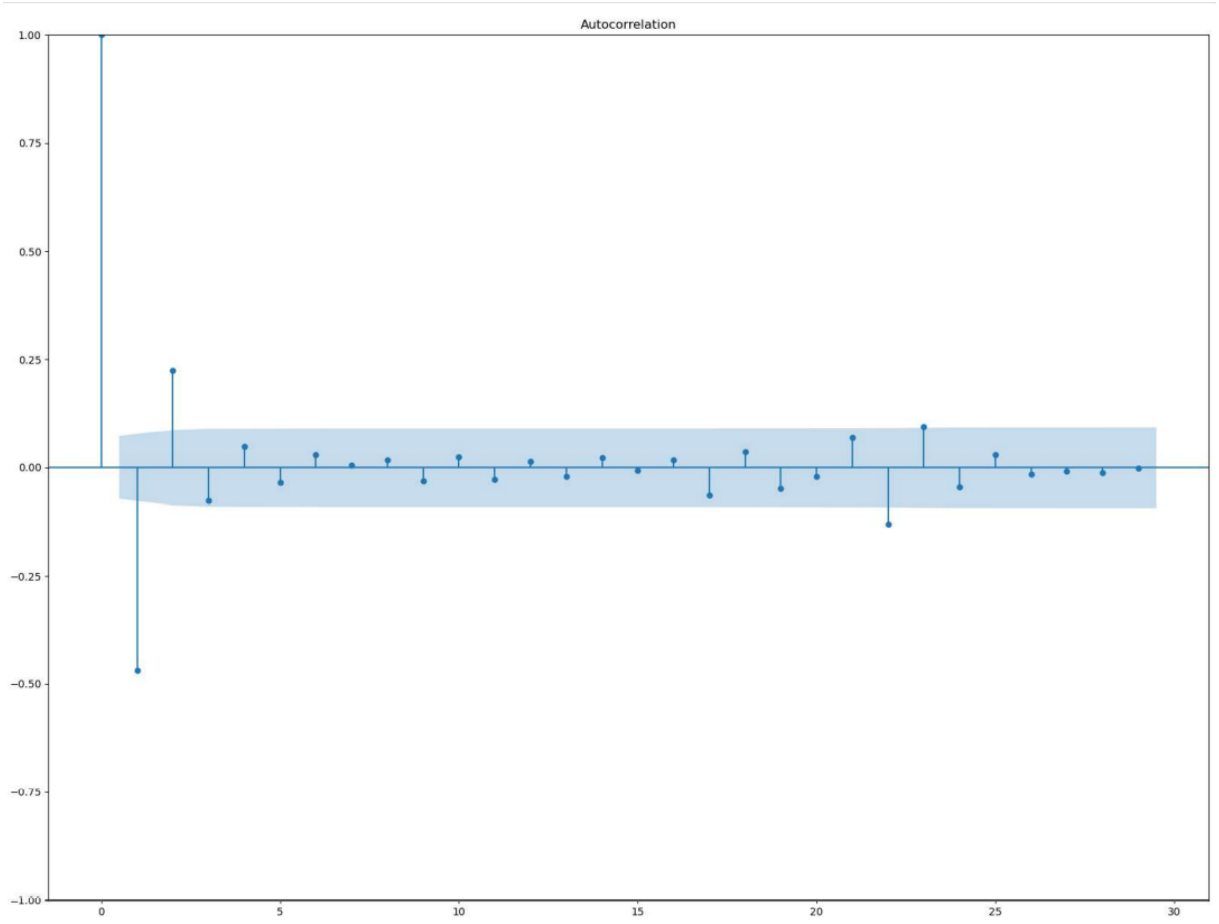
show our data is not correlated. We expect to see a high peak that very quickly goes to 0. Please see the before and after ACF plot:

ACF plot of the initial time series before differencing



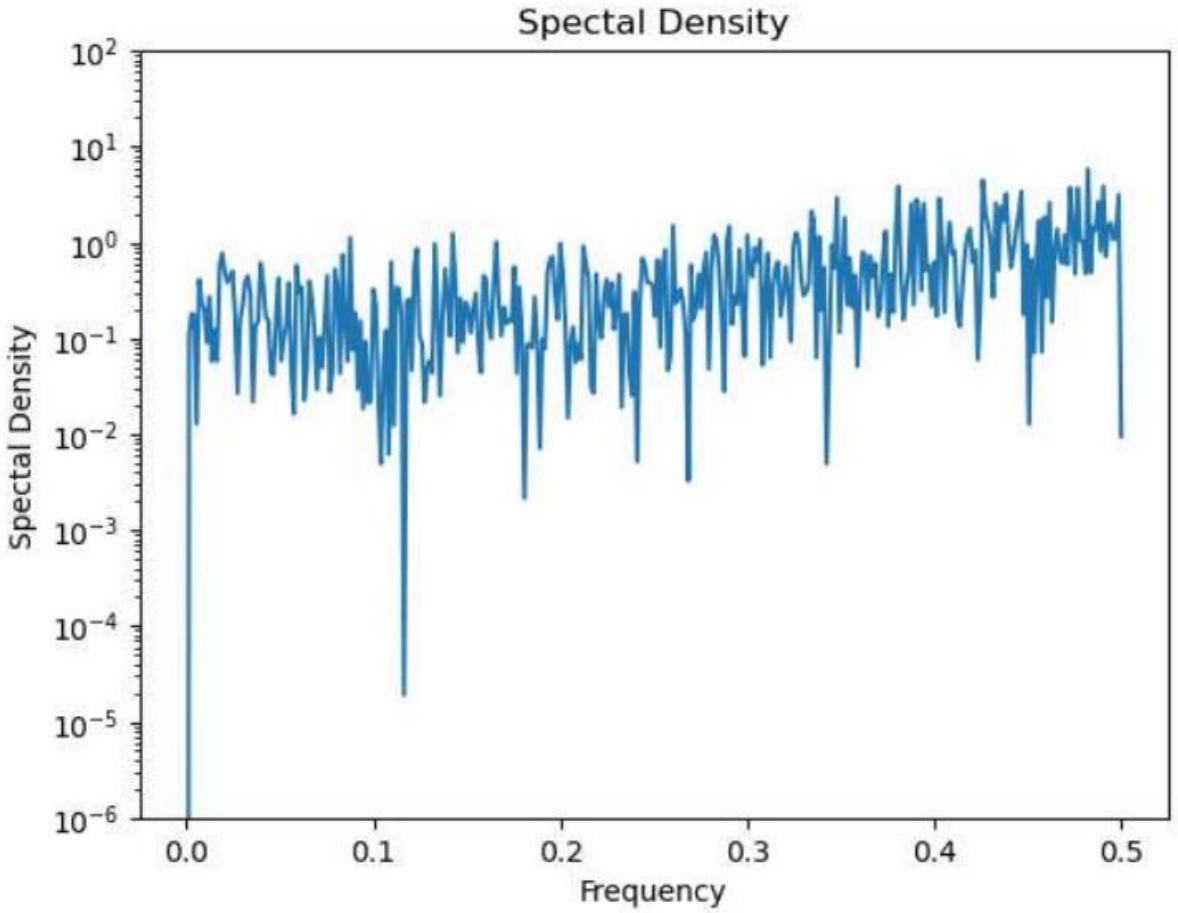

ACF plot of the initial time series. Here we can see the peak does not quickly go to zero or within the blue range (statistically insignificant) and instead continues at a significant level. We know from this we need to make our data stationary.

Time series plot after differencing




Time series plot after differencing. Here we can see the initial peak quickly falls to 0 and in the blue range, which is the statistically insignificant marker. Our data now has stationarity.

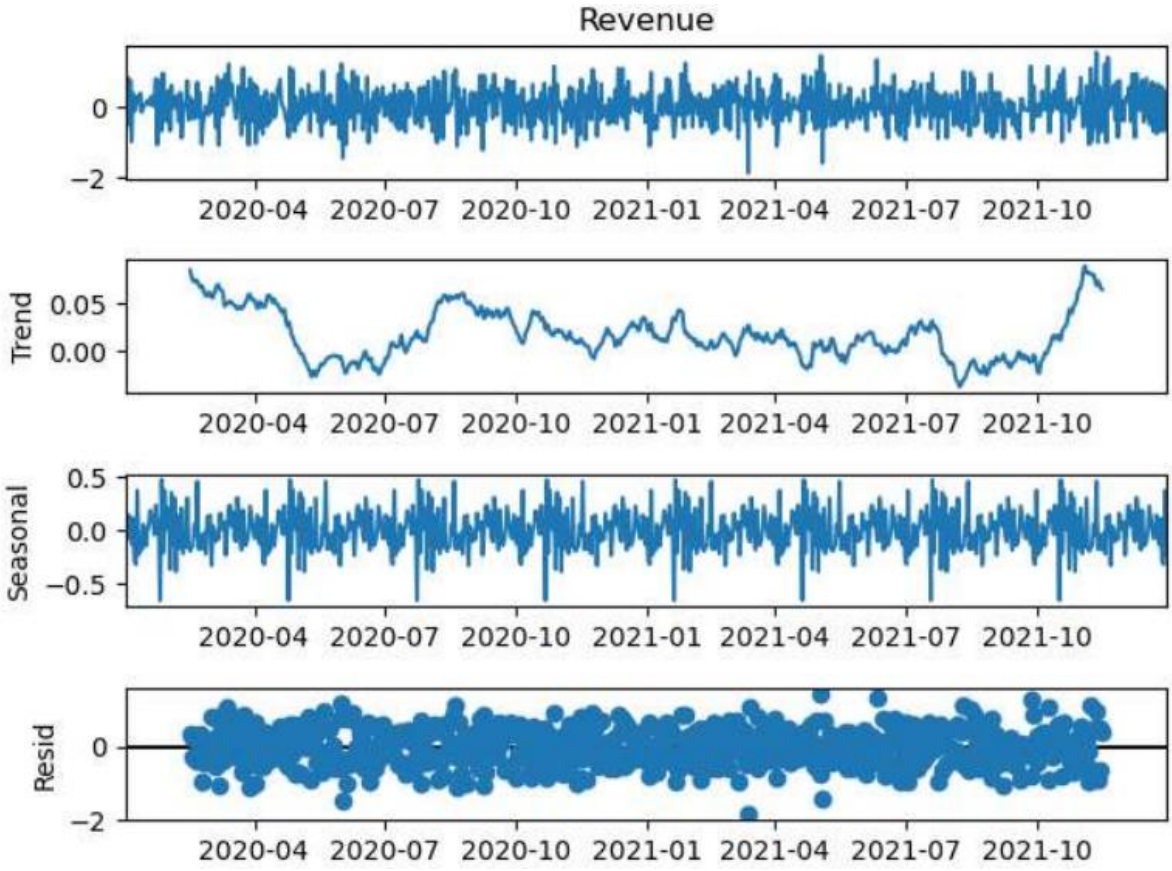
d. Spectral Density- Shows frequencies (number of observations before the seasonal pattern repeats) related to the autocovariance time domain. Autocovariance is the covariance between two elements of the time series. Spectral Density is directly related to the autocovariance time-domain representation and represents the frequency domain of a time series. We are looking for a trend if there is a pattern in regularly repeated time intervals.




Spectral Density. This plot shows the number of observations (frequency) and the autocovariance (spectral density). The blue line shows us the spectral density. If the line is not relatively constant, aka there are not large alterations to the density, then it is cyclical. Ours is relatively constant and we can see our time series is not cyclical.

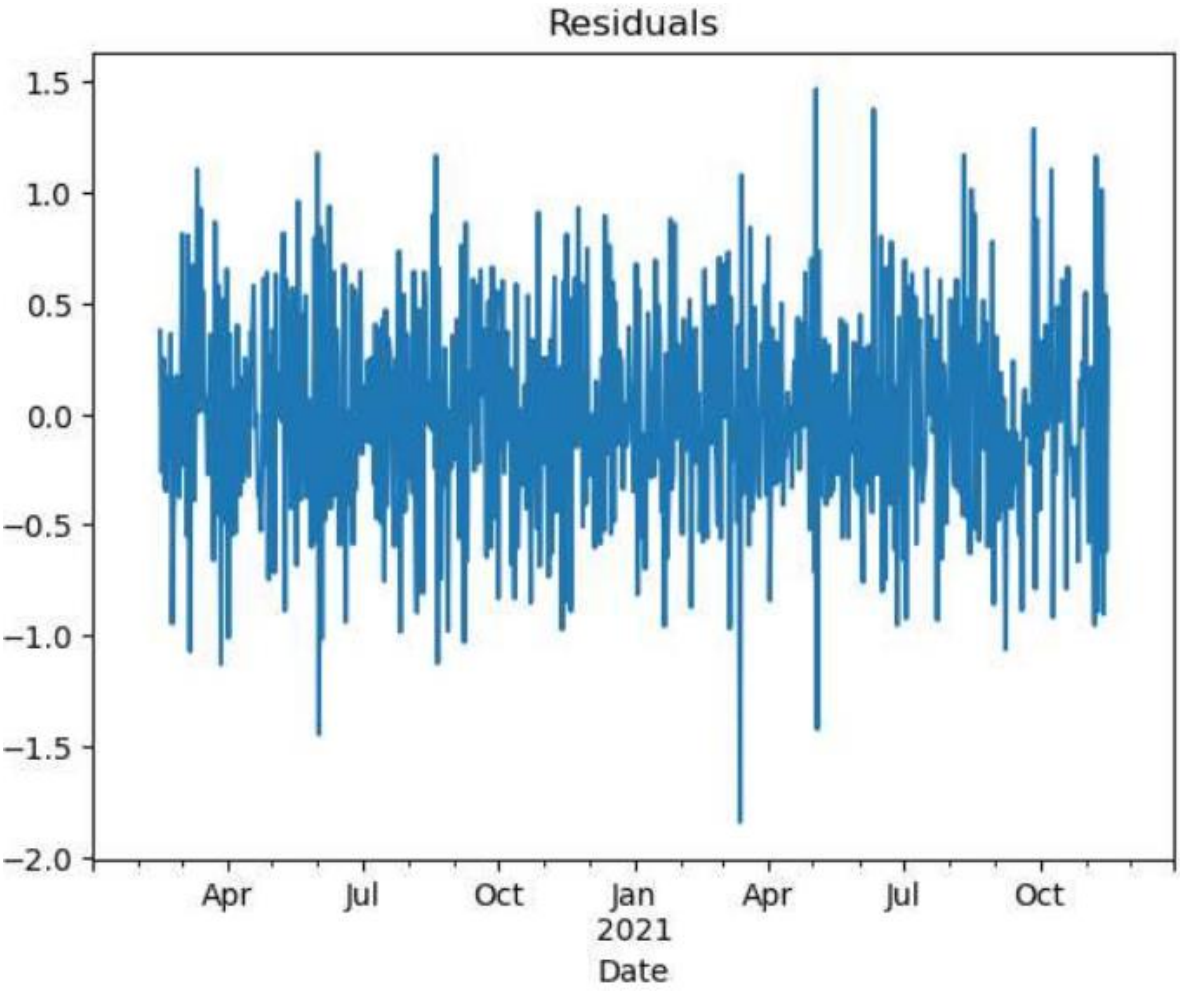
e. Decomposed Time series- When we decompose a time series, we split it into trend, seasonality, and residual. This helps provide a summary of the components of the time series as a visual representation. We have already assessed the trend and seasonality, but this is another way to check for their presence.

Decomposed time series




Decomposed time series. This is a collection of plots that break down our telecommunications time series into various components. We have the top representation which shows the revenue over time graph and below, a representation of the trend. From both of these we can see there is no identifiable trend that has a large impact on our dataset. Next is the seasonal trend showing if there is a cyclical nature to the data. The lack of variance shows there is not a seasonal aspect to it. Finally, we have the residual plot. Residuals confirm the lack of trends in our data.

f. Confirmation of the lack of trends in the residuals of the decomposed series. The residuals is the timer series after removing trend and seasonality. Here we can see the series does not have any other trends or identifiable variations across the date range.




Residuals. This plot shows residuals over time by removing the trend and seasonality components. We are looking for residuals that do not have a large variance outside of -2 and 2, which we have. This shows the difference between observations and fitted values are low.

2. An autoregressive integrated moving average (ARIMA)- To choose the ARIMA model because I checked the ACF and spectral density and took into account the lack of seasonality/trend of the differenced data. We have a financial time series and the best model to use ended up being the ARIMA model from statsmodels.tsa.arima.mode. The p values and standard deviation are within acceptable ranges.

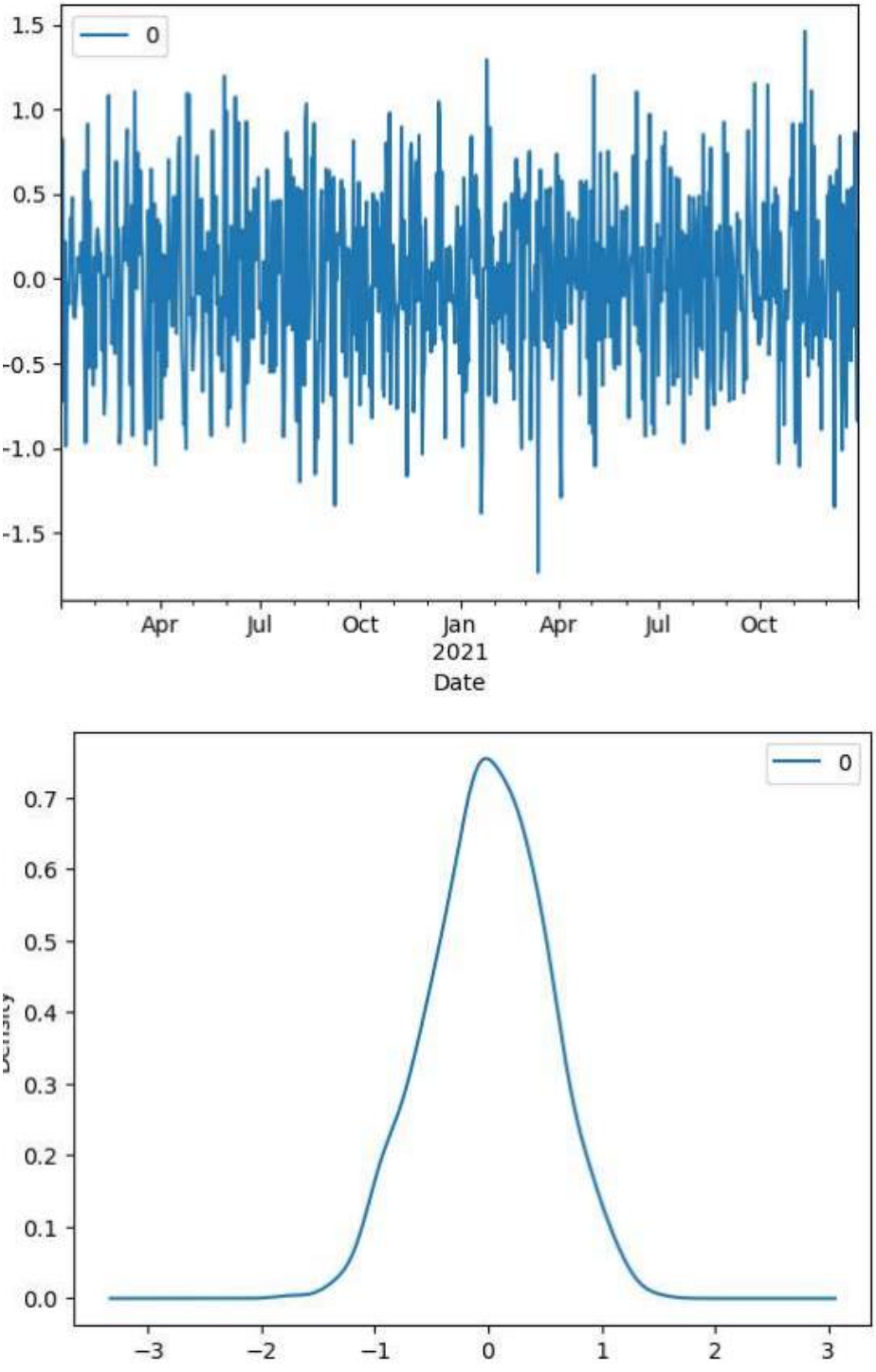
```
#D2: Autoregressive integrated moving average (ARIMA).
# fit model
df.index = df.index.to_period('D')
model = ARIMA(df, order=(5,1,0), freq = "D")
model_fit = model.fit()
# summary of fit model
print(model_fit.summary())
# line plot of residuals
residuals = DataFrame(model_fit.resid)
residuals.plot()
mpl.show()
# density plot of residuals
residuals.plot(kind='kde')
mpl.show()
# summary stats of residuals
print(residuals.describe())
```

```
residuals = DataFrame(model_fit.resid)
residuals.plot()
mpl.show()
# density plot of residuals
residuals.plot(kind='kde')
mpl.show()
# summary stats of residuals
print(residuals.describe())
```

```
                               SARIMAX Results
==============================================================================
Dep. Variable:                Revenue   No. Observations:                  730
Model:                 ARIMA(5, 1, 0)   Log Likelihood                -545.622
Date:                Wed, 01 Mar 2023   AIC                           1103.243
Time:                        17:10:00   BIC                           1130.793
Sample:                    01-02-2020   HQIC                          1113.873
                         - 12-31-2021
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -1.2947      0.036    -35.703      0.000      -1.366      -1.224
ar.L2         -1.0175      0.057    -17.802      0.000      -1.130      -0.906
ar.L3         -0.7079      0.065    -10.935      0.000      -0.835      -0.581
ar.L4         -0.4271      0.058     -7.311      0.000      -0.542      -0.313
ar.L5         -0.1813      0.037     -4.882      0.000      -0.254      -0.109
sigma2         0.2609      0.015     17.960      0.000       0.232       0.289
===================================================================================
Ljung-Box (L1) (Q):                   0.74   Jarque-Bera (JB):                 3.05
Prob(Q):                              0.39   Prob(JB):                         0.22
Heteroskedasticity (H):               0.99   Skew:                            -0.12
Prob(H) (two-sided):                  0.92   Kurtosis:                         2.79
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```

Residuals (top) and density of residuals (bottom)




Residuals (top) and density of residuals (bottom).

| | 0 |
|---|---|
| count | 730.000000 |
| mean | -0.002052 |
| std | 0.511771 |
| min | -1.734059 |
| 25% | -0.354040 |
| 50% | 0.007620 |
| 75% | 0.347918 |
| max | 1.460947 |

3. A forecast using the derived ARIMA model is shown below. We are looking at revenue data for a predicted 250 days. To validate the data, we tested our predictions and looked for the root-mean square which shows the difference between the estimated and actual values of the model. The RMSE is 0.517. We want the RMSE to be as close to 0 as possible, so .517 is rather disappointing but does not deem this model as useless.

```
#Predict
from sklearn.metrics import mean_squared_error
from math import sqrt
X = df.values
size = int(len(X) * 0.66)
train, test = X[0:size], X[size:len(X)]
history = [x for x in train]
predictions = list()
# walk-forward validation
for t in range(len(test)):
    model = ARIMA(history, order=(5,1,0))
    model_fit = model.fit()
    output = model_fit.forecast()
    yhat = output[0]
    predictions.append(yhat)
    obs = test[t]
    history.append(obs)
    print('predicted=%f, expected=%f' % (yhat, obs))
#Evaluate our forecast
rmse = sqrt(mean_squared_error(test, predictions))
print('Test RMSE: %.3f' % rmse)
```

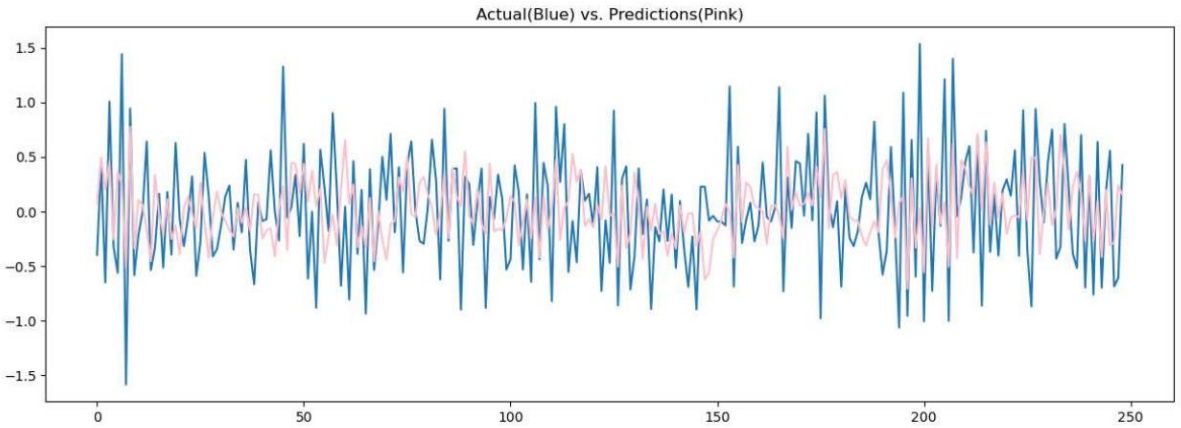
```
predicted=0.258186, expected=0.459377
predicted=-0.124528, expected=0.750848
predicted=0.023465, expected=-0.432122
predicted=0.705158, expected=-0.325217
predicted=0.346282, expected=0.802984
predicted=-0.165041, expected=0.205935
predicted=0.225539, expected=-0.385952
predicted=0.362889, expected=-0.517971
predicted=0.162307, expected=0.701432
predicted=-0.269716, expected=-0.696344
```

ee/Documents/WGU/D213/D213_Task1_Code.ipynb

D213_Task1_Coc

```
predicted=0.331193, expected=0.327692
predicted=-0.275620, expected=-0.761868
predicted=0.097753, expected=0.637298
predicted=-0.415888, expected=-0.697778
predicted=0.198563, expected=0.170280
predicted=-0.309482, expected=0.559108
predicted=-0.283029, expected=-0.687028
predicted=0.240091, expected=-0.608824
predicted=0.151032, expected=0.425985
Test RMSE: 0.517
```

```
#D3: Forecast using the derived ARIMA model against outcome
mpl.figure(figsize=(15,5))
mpl.title('Actual(Blue) vs. Predictions(Pink)')
mpl.plot(test)
mpl.plot(predictions, color='pink')
mpl.show()
```




Actual(Blue) vs. Predictions(Pink) is a 250 day forecast using the ARIMA model.

4. The output and calculations of the analysis are provided in the python code file, pdf of the notebook, and html of the notebook. I have also included the graphs at each step of this analysis for 1-3. I did not copy and paste into this specific section to avoid redundancy and excessive length, however, all of the components are provided in the visualizations above and in the two files provided.

5. The code used to support the implementation of the time series model has been provided as “D213_Task1_code” and can be seen in the pdf and html file.

## Part V: Data Summary and Implications

### E. Summarize your findings and assumptions, including the following points:

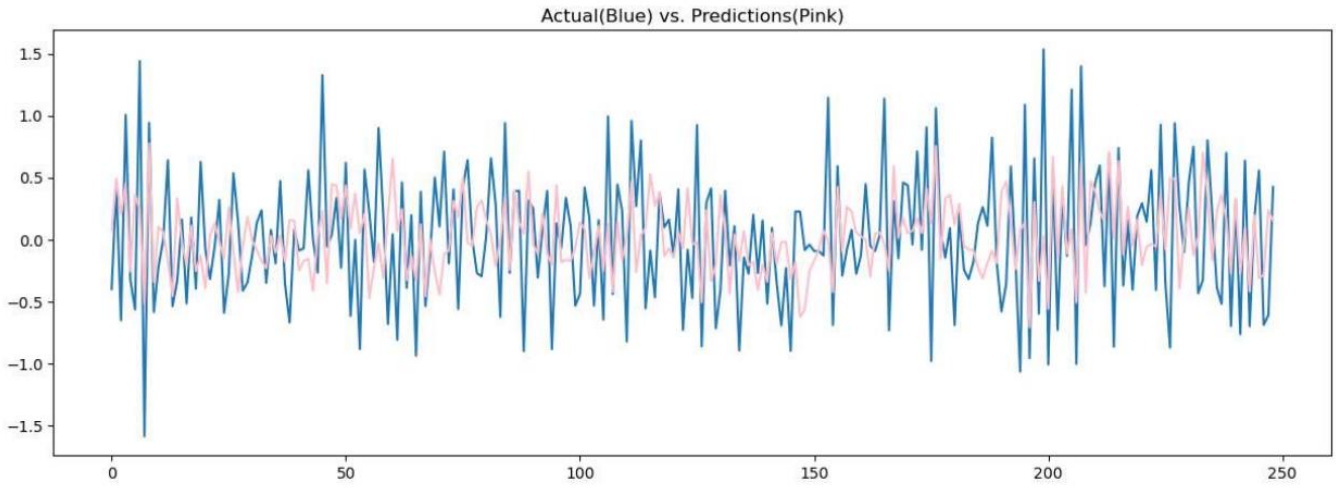
1. **Discuss the results of your data analysis, including the following:**
   - **the selection of an ARIMA model**
   - **the prediction interval of the forecast**
   - **a justification of the forecast length**
   - **the model evaluation procedure and error metric**
2. **Provide an annotated visualization of the forecast of the final model compared to the test set.**
3. **Recommend a course of action based on your results.**

1. The results of the analysis are as follows: In summary, we were able to create a model that can be used to forecast the revenue per day over the next 250 days.
   a. Selection of an ARIMA model- For this analysis we used an ARIMA model due to the identity of the time series and the low AIC score it gave us: 1103. We also needed to avoid one that uses seasonality and trends since we already found it does not exist in our dataset after differencing.
   b. Prediction interval of the forecast- The prediction interval of our forecast is one day. The time series model consists of data from 2 years and our ARIMA model identifies seasonality and correlations to predict revenue at an interval of one day.
   c. Justification of the forecast length- For a two year time series and making a forecast from it, if you go into long term prediction beyond one year, you reduce

the accuracy available. One year would be a viable prediction and going even less than one year would give us greater accuracy. As a result, we have gone for a little less than one year, about 250 days for the forecast. Our prediction model matches up to our test data.

d. Model evaluation procedure and error metric: We evaluated this model in several ways. Firstly, we used a model with an acceptable and relative low AIC score from what I was expecting. Secondly, we used root mean square error (RMSE) to evaluate our forecast. We had .517 RMSE, which is far from 0 and whose acceptability would need to be determined by the organization moving forward. I find it acceptable for this model specific model and it shows we need more analysis.

2. The annotated visualizations of the forecast final model compared to the test set:




Actual(Blue) vs. Predictions(Pink). This is a plot comparing the forecast of our final model with the test set. We can see the predictions follow the test data very well and have less variations. The forecast is very similar to the test data and follows the same microtrends with spikes and valleys.

3. A course of action I recommend based on these results is to use this forecast model to get an idea of what the revenue will look like for the next 250 days. This can inform business decisions and if no changes are made to practice, give a metric that is expected. However, I suggest the organization put more resources into trying to raise revenue beyond what this forecast shows by supporting departments as deemed necessary. The company can also use this forecast to plan for expansions. With this being said, the RMSE of .517 is large and I would recommend the organization to run more tests to try and create a model more accurate than this forecast.

## Part VI: Reporting

### F. Create your report from part E using an industry-relevant interactive development environment (e.g., a Jupyter Notebook). Include a PDF or HTML document of your executed notebook presentation.

The html has been included as “D213_Task1_Code.html”

The pdf has been included as “D213_Task1_Code_backup.pdf”

### G. List the web sources used to acquire data or segments of third-party code to support the application.

References

Brownlee, J. (2017, February 21). *How to Difference a Time Series Dataset with Python*. MachineLearningMastery.com. https://machinelearningmastery.com/difference-time-series-dataset-python/

Prabhakaran, S. (2019, February 13). *Time Series Analysis in Python - A Comprehensive Guide with Examples – ML+*. Machine Learning Plus. https://www.machinelearningplus.com/time-series/time-series-analysis-python

## H. Acknowledge sources, using in-text citations and references, for content that is quoted, paraphrased, or summarized.

References

Brownlee, J. (2017, February 21). *How to Difference a Time Series Dataset with Python*. MachineLearningMastery.com. https://machinelearningmastery.com/difference-time-series-dataset-python/

Prabhakaran, S. (2019, February 13). *Time Series Analysis in Python - A Comprehensive Guide with Examples – ML+*. Machine Learning Plus. https://www.machinelearningplus.com/time-series/time-series-analysis-python

## I. Demonstrate professional communication in the content and presentation of your submission.

This aspect of the rubric is evaluated through the entirety of this report and I hope professionalism has shown continuously.