D213 – ADVANCED DATA ANALYTICS

Task 1: Time Series Modeling

WGU - MSDA

Advanced Data Analytics with Churn Dataset

Richard Flores

rflo147@wgu.edu

Student ID: 006771163

Table of Contents

Part I: Research Question

A1. Proposal of Question

Through the use of varied data analysis techniques, we have previously gathered a plethora of insights into customer characteristics and behavior that helps us shape the course of action in the telecommunications company to reduce customer churn and improve annual profits.

The goal of this analysis is to build upon the basic trends and patterns previously discovered and create a more sophisticated statistical model. We will achieve this goal by using a time series model to create, generate forecasts, and report the results of the analysis in a professional manner. In this analysis we will answer the research question:

Will a Time Series analysis of the dataset provide meaningful insights of patterns and trends in the data and allow us to forecast accurate predictions for the future?

A2. Objectives and Goals

For this analysis, the goal will be to use a Time Series to analyze data points in the Churn dataset to discover an internal structure. In the Time series we will generate a thorough accounting of our findings such as seasonal variation or trends. Two objectives are to provide information on stationarity and autocorrelated data which will assist stakeholders and executives with insight into corporate business metrics.

Part II: Method Justification

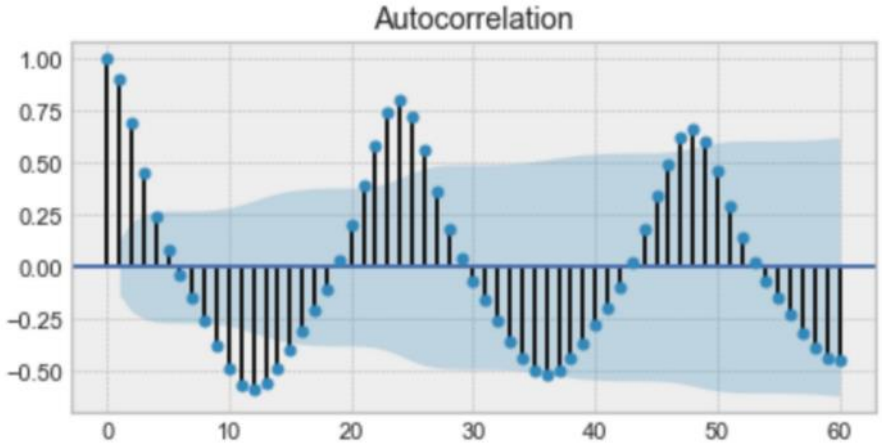
B. Summary of Assumptions

This Time Series analysis is based on the assumptions that the analysis will provide insights into three particular key aspects namely seasonality, stationarity, and autocorrelation.

Seasonality can be explained as observing periodic fluctuations in the data. An example of seasonality would be bandwidth consumption increasing during the evening hours or retail sales increasing after the Thanksgiving holiday. It is important to note that seasonality may be derived through an autocorrelation plot if said plot contains a sinusoidal shape (Peixeiro 2022).

Stationarity exists in a Time Series if the characteristics of the statistical properties remain fixed over time. We can examine stationarity through examination of the variables *constant mean and variance*, relying on covariance being independent of time. The ideal Time Series model is stationary however it is possible to utilize various transformation to change a dataset into a stationary model (Peixeiro 2022).

Autocorrelation is defined by TowardsDataScience as the similarity between observations as a function of the time lag between them (Peixeiro 2022). Autocorrelation is observed by constant and predictable changes in data.



(Figure: Towards Data Science 2022)

From the graph we can observe autocorrelation in the peaks of the data as the highest value occurs every 24th unit in the graph.

Part III: Data Preparation

C1. Line Graph Visualization

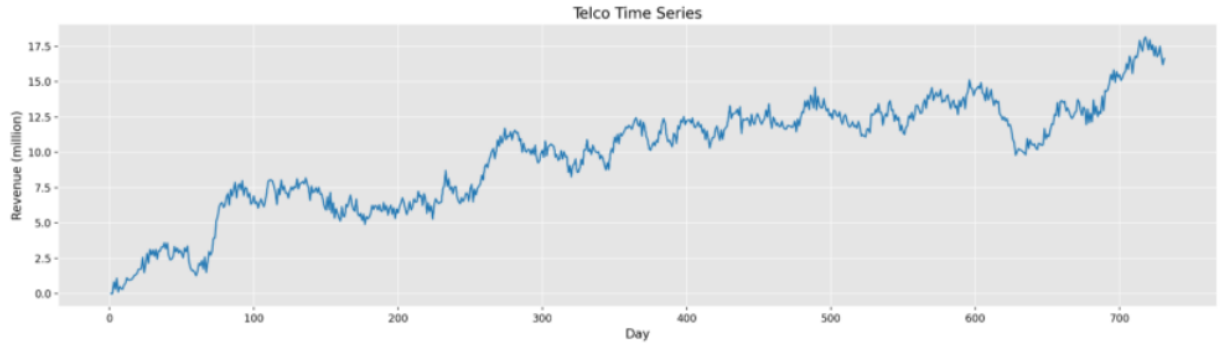
```
# Import Data Science libraries for Data Analysis, Visualization, and Plotting
import numpy as np
import pandas as pd

import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns

# Import adfuller for stationarity test
from statsmodels.tsa.stattools import adfuller

# Import seasonal decompose function from statsmodel
from statsmodels.tsa.seasonal import seasonal_decompose

# Import plot_acf from Statsmodels to calculate Autocorrelation Function
from statsmodels.graphics.tsaplots import plot_acf

# Import auto_arima for future trend analysis
from pmdarima import auto_arima

# Import statsmodels for SARIMAX modelling
import statsmodels.api as sm

# Import sklearn for Root Mean Square calculations
from sklearn.metrics import mean_squared_error
from statsmodels.tools.eval_measures import rmse

# Setup plotting
plt.style.use('ggplot')
COLOR = 'black'
mpl.rcParams['text.color'] = COLOR
mpl.rcParams['axes.labelcolor'] = COLOR
mpl.rcParams['xtick.color'] = COLOR
mpl.rcParams['ytick.color'] = COLOR

# Skip warning messages for clarity
import warnings
warnings.filterwarnings('ignore')
```

```
# Load WGU provided dataset into Pandas dataframe
teleco_df = pd.read_csv('teleco_time_series.csv')
```

```
# Exploration - View header of dataframe
teleco_df.head(10)
```

	Day	Revenue
0	1	0.000000
1	2	0.000793
2	3	0.825542
3	4	0.320332
4	5	1.082554
5	6	0.107654
6	7	0.493901
7	8	0.376698
8	9	0.304075
9	10	0.591748

```python
# Plot a line graph for the time series dataset with labels and complete realization of the data
def plot_df(df, x, y, title='', xlabel='Day', ylabel='Revenue (million)', dpi=200):
    plt.figure(figsize=(20, 5), dpi=dpi)
    plt.plot(x, y, color='tab:blue')
    plt.gca().set(title=title, xlabel=xlabel, ylabel=ylabel)
    plt.show()

plot_df(teleco_df, x=teleco_df['Day'], y=teleco_df['Revenue'],
        title='Telco Time Series')
```



C2. Time Step Formatting

```
# Exploration - Dataframe information
teleco_df.info
```

```
<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]>
```

```
# Exploration - Dataframe variable types
teleco_df.dtypes
```

```
Day          int64
Revenue    float64
dtype: object
```

```
# Data Cleaning - Check for missing values
print(teleco_df.isnull().values.any())
```

```
False
```

```
# Data Cleaning - Check for NAN values
print(teleco_df.isna().values.any())
```

```
False
```

```
# Data Cleaning - Check for duplicate records
print(teleco_df.Day.duplicated().sum())
```

```
0
```

```
# Exploration - Describe dataframe
print(teleco_df.describe())
```

```
              Day     Revenue
count  731.000000  731.000000
mean   366.000000    9.822901
std    211.165812    3.852645
min      1.000000    0.000000
25%    183.500000    6.872836
50%    366.000000   10.785571
75%    548.500000   12.566911
max    731.000000   18.154769
```

The record length described in the info header is 731 rows which corresponds correctly to the 731 records in the dataset. We have verified the dataset is complete with no missing records, NaN values, or duplicate data. Descriptive statistics also show a 731 count for both ‘Day’ and ‘Revenue’ ensuring there are no gaps in the data.

C3. Stationarity

```
# From statsmodel use adfuller package for stationarity test
adft = adfuller(teleco_df.loc[:, 'Revenue'].values, autolag='AIC')

output_teleco_df = pd.DataFrame({'Values':[adft[0], adft[1], adft[2], adft[3],
                                 adft[4]['1%'], adft[4]['5%'], adft[4]['10%']],
                                'Metric':['adf', 'pvalue', 'usedlag',
                                        'nobs', '1% critical value',
                                         '5% critical value', '10% critical value',]})

# Print Adfuller Stationarity results
print(output_teleco_df)
```

```
       Values              Metric
0   -1.924612                 adf
1    0.320573              pvalue
2    1.000000            usedlag
3  729.000000                nobs
4   -3.439352   1% critical value
5   -2.865513   5% critical value
6   -2.568886  10% critical value
```

The test for stationarity is performed with the Statsmodel module adfuller. The adfuller function reports adf, pvalue, usedlag, nobs (number of observations), and 1%, 5%, and 10% critical values with an overall p-value of 0.32.

C4. Steps to Prepare the Data

- Import the 'telco_time_series.csv' dataset into a Pandas data frame for analysis.
- Describe the various features and data to prepare relevant items.
- Create a view of the summary statistics.
- Review record data to check for anomalies, outliers, missing data and other data that could become obstacles in the analysis.
- Split data to include training and test sets for time series modeling.
- Export manipulated Dataframe to .CSV for analysis.

```
# Use index function to split data and create training set
teleco_df['Day'] = teleco_df.index
teleco_train = teleco_df.iloc[:len(teleco_df) - 365]

teleco_train['teleco_train'] = teleco_train['Revenue']
del teleco_train['Day']
del teleco_train['Revenue']
```

```
# Create test set
teleco_test = teleco_df.iloc[len(teleco_df) - 365:]

# Test Day and Revenue
teleco_test['teleco_test'] = teleco_test['Revenue']
del teleco_test['Day']
del teleco_test['Revenue']
```

```
# Plot training and test sets
plt.plot(teleco_train, color='green')
plt.plot(teleco_test, color='blue')
plt.title('Training and Test Sets')
plt.xlabel('Days')
plt.ylabel('Revenue (Millions)')
sns.set()
plt.show()
```

The data has been successfully split into training and testing sets with a plot for visualization of corresponding green and blue colors.

C5. Prepared Dataset

```
# Extract Prepared Telco Dataset to CSV
teleco_df.to_csv('prepared_time_series.csv')
```

Dataset has been exported for submission and review.

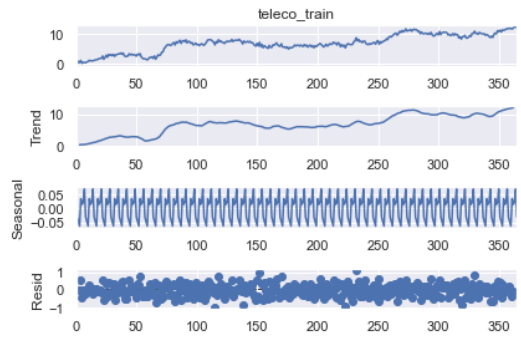
Part IV: Model Identification and Analysis

D1. Report Findings and Visualizations

Visualizations

Code and Output for Statsmodel Decompose analytics on Trend, Seasonality, and Residual Time elements.

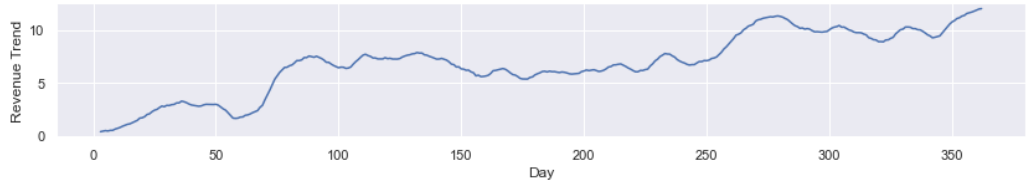
```
# Prepare for seasonality assessment using statsmodel decompose function
decompose = seasonal_decompose(teleco_train['teleco_train'], model='additive', period=7)
decompose.plot()
plt.show()
```



Code and Plot for Decompose Trend.

```
# Decompose and plot Trend
decompose_trend = decompose.trend

ax = decompose_trend.plot(figsize=(14,2))
ax.set_xlabel('Day')
ax.set_ylabel('Revenue Trend')
```

```
Text(0, 0.5, 'Revenue Trend')
```



Code and Plot for Autocorrelation.

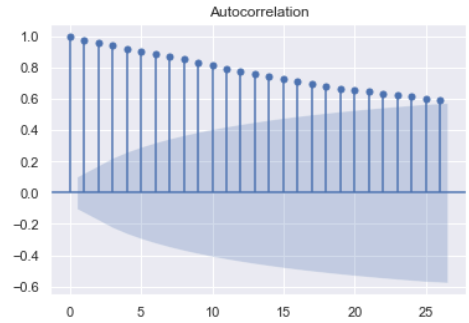
```
# Use autocorr feature to calculate autocorrelation with 31 day lag
autocorrelation_lag1 = teleco_df['Revenue'].autocorr(lag=31)
print("Period Lag: 31 days: ", autocorrelation_lag1)
```

```
Period Lag: 31 days:  0.8690526347331857
```

```
# Recalculate autocorrelation with 62 day and 182 day periods
autocorrelation_lag3 = teleco_df['Revenue'].autocorr(lag=62)
print("Period Lag: 62 days: ", autocorrelation_lag3)

autocorrelation_lag6 = teleco_df['Revenue'].autocorr(lag=182)
print("Period Lag: 182 days: ", autocorrelation_lag6)
```

```
Period Lag: 62 days:  0.7758400879703407
Period Lag: 182 days:  0.8139870392893127
```

```
# Use statsmodel to plot autocorrelation function (ACF)
plot_acf(teleco_train)
plt.show()
```

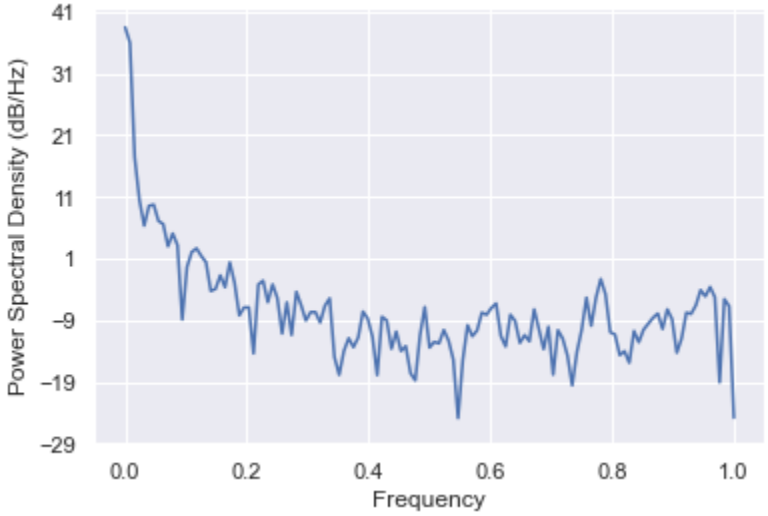


Code and Plot for Spectral Density.

```
# Calculate and plot power spectral density
plt.psd(teleco_df['Revenue'])
```

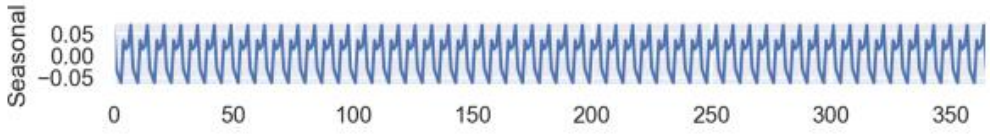
```
(array([6.97387711e+03, 3.91439441e+03, 5.47611144e+01, 1.10791953e+01,
        4.25439684e+00, 8.90767103e+00, 9.32436541e+00, 5.07699490e+00,
        4.50820186e+00, 1.97142105e+00, 3.17930644e+00, 2.05465010e+00,
        1.27856111e-01, 9.12357670e-01, 1.58928494e+00, 1.82600308e+00,
        1.37345831e+00, 1.08602805e+00, 3.71664573e-01, 4.01447712e-01,
        6.67019362e-01, 4.28134847e-01, 1.08303874e+00, 5.05461139e-01,
        1.49612021e-01, 2.01156666e-01, 2.01929017e-01, 3.62482513e-02,
        4.74386760e-01, 5.46619399e-01, 2.44460915e-01, 4.75225583e-01,
        2.85007556e-01, 7.59631735e-02, 2.43183009e-01, 7.19503505e-02,
        3.60475608e-01, 2.18971480e-01, 1.23421972e-01, 1.69234109e-01,
        1.69371921e-01, 1.14748035e-01, 2.14144408e-01, 2.82736486e-01,
        3.17059554e-02, 1.62458482e-02, 3.98324693e-02, 6.38895068e-02,
        4.57061837e-02, 6.54718394e-02, 1.72598760e-01, 1.32802101e-01,
        6.74716970e-02, 1.61089989e-02, 1.42266347e-01, 1.23640967e-01,
        4.30737122e-02, 8.17619769e-02, 3.95102804e-02, 4.74833201e-02,
        1.73085344e-02, 1.32485092e-02, 7.14595244e-02, 2.04348554e-01,
        4.51051907e-02, 5.55609932e-02, 5.26341931e-02, 8.68327071e-02,
        5.86340275e-02, 2.88124318e-02, 3.23398046e-03, 2.85277556e-02,
        1.03859619e-01, 6.85761289e-02, 8.53467489e-02, 1.65815053e-01,
        1.53444767e-01, 1.97776090e-01, 2.33413022e-01, 6.95161339e-02,
        4.68937467e-02, 1.53353843e-01, 1.21391009e-01, 5.34892233e-02,
        7.17250734e-02, 5.68272472e-02, 1.87708648e-01, 8.92439810e-02,
        4.25453689e-02, 9.79721726e-02, 1.63975102e-02, 8.61687997e-02,
        6.31239429e-02, 3.34886298e-02, 1.09875888e-02, 3.83717041e-02,
        8.81504336e-02, 2.89832055e-01, 1.01621880e-01, 2.88982189e-01,
        5.81784635e-01, 3.29483923e-01, 8.00152896e-02, 7.31104074e-02,
        3.39216170e-02, 3.93543037e-02, 2.55189466e-02, 8.36160222e-02,
        5.48857747e-02, 8.78315482e-02, 1.10183077e-01, 1.37188728e-01,
        1.60562556e-01, 9.01524835e-02, 1.87554502e-01, 1.29971998e-01,
        3.71787543e-02, 6.31650566e-02, 1.65356465e-01, 1.58164919e-01,
        2.17681939e-01, 3.85248684e-01, 3.04221135e-01, 4.32125608e-01,
        2.94750935e-01, 1.22618052e-02, 2.74845469e-01, 2.10555482e-01,
        3.28604031e-03]),
```

```
array([0.       , 0.0078125, 0.015625 , 0.0234375, 0.03125  , 0.0390625,
       0.046875 , 0.0546875, 0.0625   , 0.0703125, 0.078125 , 0.0859375,
       0.09375  , 0.1015625, 0.109375 , 0.1171875, 0.125    , 0.1328125,
       0.140625 , 0.1484375, 0.15625  , 0.1640625, 0.171875 , 0.1796875,
       0.1875   , 0.1953125, 0.203125 , 0.2109375, 0.21875  , 0.2265625,
       0.234375 , 0.2421875, 0.25     , 0.2578125, 0.265625 , 0.2734375,
       0.28125  , 0.2890625, 0.296875 , 0.3046875, 0.3125   , 0.3203125,
       0.328125 , 0.3359375, 0.34375  , 0.3515625, 0.359375 , 0.3671875,
       0.375    , 0.3828125, 0.390625 , 0.3984375, 0.40625  , 0.4140625,
       0.421875 , 0.4296875, 0.4375   , 0.4453125, 0.453125 , 0.4609375,
       0.46875  , 0.4765625, 0.484375 , 0.4921875, 0.5      , 0.5078125,
       0.515625 , 0.5234375, 0.53125  , 0.5390625, 0.546875 , 0.5546875,
       0.5625   , 0.5703125, 0.578125 , 0.5859375, 0.59375  , 0.6015625,
       0.609375 , 0.6171875, 0.625    , 0.6328125, 0.640625 , 0.6484375,
       0.65625  , 0.6640625, 0.671875 , 0.6796875, 0.6875   , 0.6953125,
       0.703125 , 0.7109375, 0.71875  , 0.7265625, 0.734375 , 0.7421875,
       0.75     , 0.7578125, 0.765625 , 0.7734375, 0.78125  , 0.7890625,
       0.796875 , 0.8046875, 0.8125   , 0.8203125, 0.828125 , 0.8359375,
       0.84375  , 0.8515625, 0.859375 , 0.8671875, 0.875    , 0.8828125,
       0.890625 , 0.8984375, 0.90625  , 0.9140625, 0.921875 , 0.9296875,
       0.9375   , 0.9453125, 0.953125 , 0.9609375, 0.96875  , 0.9765625,
       0.984375 , 0.9921875, 1.       ]))
```



Report of Findings

1. **Presence or lack of a seasonal component.**

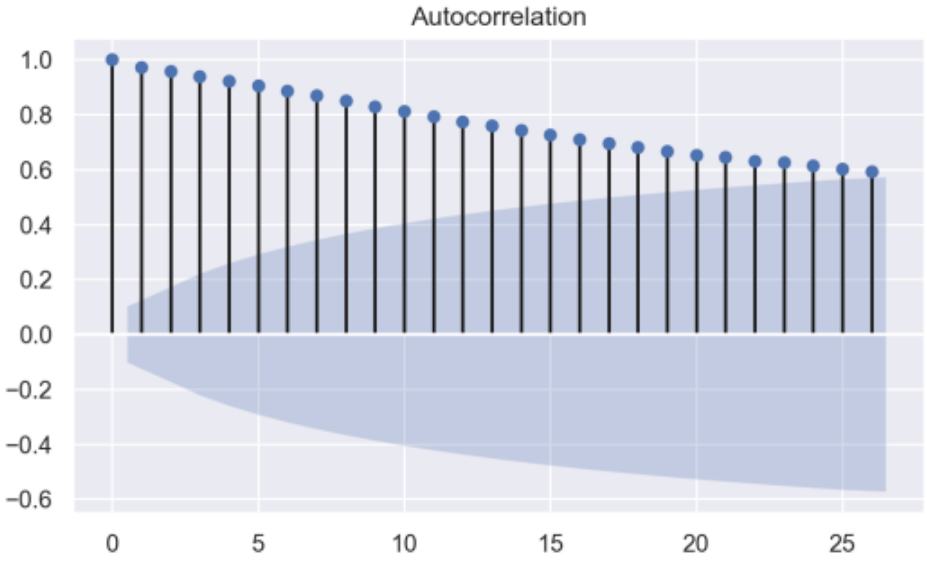


From the decompose analysis we definitely see a recurring period in the data suggesting a certain repeating period. Therefor we can conclude there is a definite observation of seasonality in the dataset.

2. **Trends**.

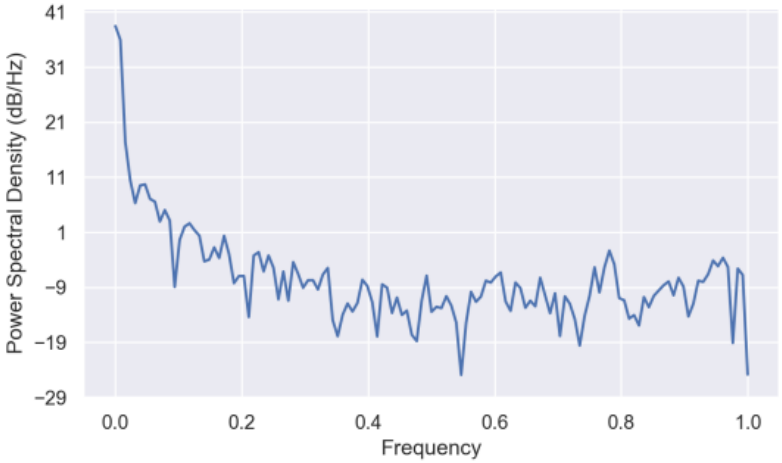
From the decompose analysis we observe a gradual trend upwards over time over the dataset period of two years.

3. **Auto Correlation Function.**



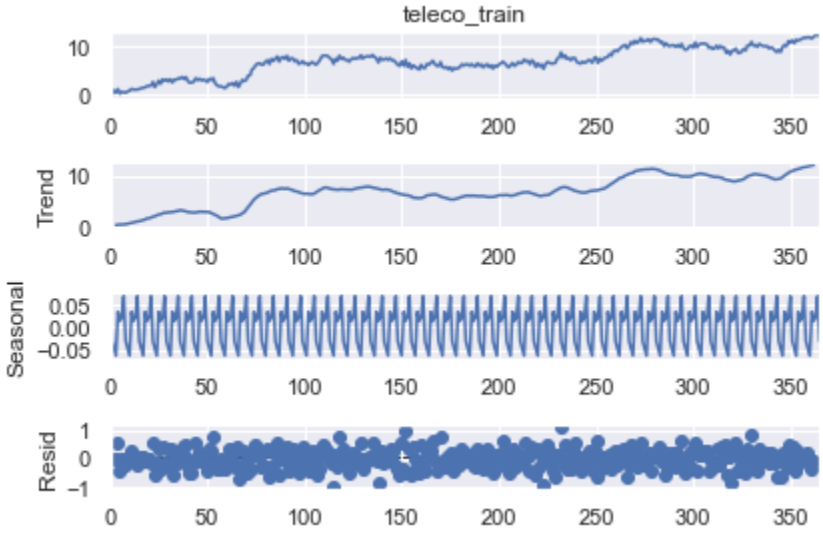
The autocorrelation function displays a high degree of correlation at lag measurements of 31, 62, and 182 days with corresponding values of 0.86, 0.77, and 0.81.

4. **Spectral Density.**



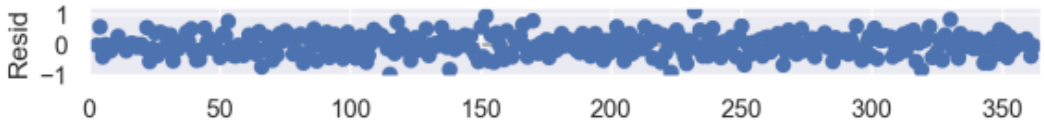
The spectral density plot function is derived from Welch's average periodogram algorithm. The plot represents a series of sine waves derived from the calculated array of which the values will be useful in our later analysis.

5. **Decomposed Time Series.**



The Decomposed Time series is composed of Trend, Seasonality, and Residual Time elements. These elements are broken down into the corresponding categories in our Report of Findings for further analysis.

6. **Confirmation of the lack of trends in the residuals of the decomposed series.**



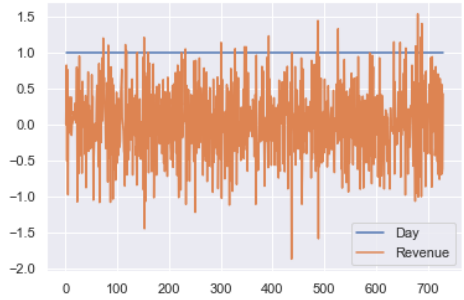
From the decompose analysis we can observe the residuals in the series. We do not observe a trend in the residuals data.

D2. Arima Model

```
# Set first difference for ARIMA Model
first_diff = teleco_df.diff().dropna()
```

```
# Use previously invoked adfuller module from statsmodel on difference
result = adfuller(first_diff['Revenue'])
```

```
# Create visualization of difference
fig, ax = plt.subplots()
first_diff.plot(ax=ax)
plt.show()
```




```
# Print calculated ADF result and pvalue
print('ADF Result:', result[0])
print('pvalue:', result[1])
```

```
ADF Result: -44.874527193875984
pvalue: 0.0
```

```
# Install pmdarima
!pip install pmdarima
```

```
Requirement already satisfied: pmdarima in c:\users\richard\anaconda3\lib\site-packages (1.8.5)
Requirement already satisfied: scipy>=1.3.2 in c:\users\richard\anaconda3\lib\site-packages (from pmdarima) (1.6.2)
Requirement already satisfied: joblib>=0.11 in c:\users\richard\anaconda3\lib\site-packages (from pmdarima) (1.0.1)
Requirement already satisfied: urllib3 in c:\users\richard\anaconda3\lib\site-packages (from pmdarima) (1.26.4)
Requirement already satisfied: setuptools!=50.0.0,>=38.6.0 in c:\users\richard\anaconda3\lib\site-packages (from pmdarima) (52.
0.0.post20210125)
Requirement already satisfied: scikit-learn>=0.22 in c:\users\richard\anaconda3\lib\site-packages (from pmdarima) (0.24.1)
Requirement already satisfied: pandas>=0.19 in c:\users\richard\anaconda3\lib\site-packages (from pmdarima) (1.2.4)
Requirement already satisfied: numpy>=1.19.3 in c:\users\richard\anaconda3\lib\site-packages (from pmdarima) (1.22.2)
Requirement already satisfied: statsmodels!=0.12.0,>=0.11 in c:\users\richard\anaconda3\lib\site-packages (from pmdarima) (0.1
2.2)
Requirement already satisfied: Cython!=0.29.18,>=0.29 in c:\users\richard\anaconda3\lib\site-packages (from pmdarima) (0.29.23)
Requirement already satisfied: python-dateutil>=2.7.3 in c:\users\richard\anaconda3\lib\site-packages (from pandas>=0.19->pmdar
ima) (2.8.1)
Requirement already satisfied: pytz>=2017.3 in c:\users\richard\anaconda3\lib\site-packages (from pandas>=0.19->pmdarima) (202
1.1)
Requirement already satisfied: six>=1.5 in c:\users\richard\anaconda3\lib\site-packages (from python-dateutil>=2.7.3->pandas>=
0.19->pmdarima) (1.15.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\richard\anaconda3\lib\site-packages (from scikit-learn>=0.22->p
mdarima) (2.1.0)
Requirement already satisfied: patsy>=0.5 in c:\users\richard\anaconda3\lib\site-packages (from statsmodels!=0.12.0,>=0.11->pmd
arima) (0.5.1)
```

```
# Perform ARIMA fit on dataset
auto_arima_fit = auto_arima(teleco_df['Revenue'], start_P=1,
                            start_q=1,
                            max_p=3,
                            max_q=3,
                            m=12,
                            seasonal=True,
                            d=None,
                            D=1,
                            trace=True,
                            error_action='ignore',
                            suppress_warnings=True,
                            stepwise=True)

auto_arima_fit.summary()
```

```
Performing stepwise search to minimize aic
 ARIMA(2,0,1)(1,1,1)[12] intercept   : AIC=inf, Time=5.78 sec
 ARIMA(0,0,0)(0,1,0)[12] intercept   : AIC=2367.159, Time=0.04 sec
 ARIMA(1,0,0)(1,1,0)[12] intercept   : AIC=1419.537, Time=1.20 sec
 ARIMA(0,0,1)(0,1,1)[12] intercept   : AIC=1969.738, Time=0.83 sec
 ARIMA(0,0,0)(0,1,0)[12]             : AIC=2399.547, Time=0.04 sec
 ARIMA(1,0,0)(0,1,0)[12] intercept   : AIC=1568.311, Time=0.21 sec
 ARIMA(1,0,0)(2,1,0)[12] intercept   : AIC=1320.755, Time=2.40 sec
 ARIMA(1,0,0)(2,1,1)[12] intercept   : AIC=inf, Time=7.90 sec
 ARIMA(1,0,0)(1,1,1)[12] intercept   : AIC=inf, Time=5.16 sec
 ARIMA(0,0,0)(2,1,0)[12] intercept   : AIC=2339.965, Time=0.96 sec
 ARIMA(2,0,0)(2,1,0)[12] intercept   : AIC=1147.041, Time=4.26 sec
 ARIMA(2,0,0)(1,1,0)[12] intercept   : AIC=1256.245, Time=2.08 sec
 ARIMA(2,0,0)(2,1,1)[12] intercept   : AIC=inf, Time=9.04 sec
 ARIMA(2,0,0)(1,1,1)[12] intercept   : AIC=inf, Time=5.49 sec
 ARIMA(3,0,0)(2,1,0)[12] intercept   : AIC=1148.348, Time=5.71 sec
 ARIMA(2,0,1)(2,1,0)[12] intercept   : AIC=1148.544, Time=5.07 sec
 ARIMA(1,0,1)(2,1,0)[12] intercept   : AIC=1195.703, Time=4.33 sec
 ARIMA(3,0,1)(2,1,0)[12] intercept   : AIC=1132.898, Time=14.84 sec
 ARIMA(3,0,1)(1,1,0)[12] intercept   : AIC=1235.930, Time=7.56 sec
 ARIMA(3,0,1)(2,1,1)[12] intercept   : AIC=inf, Time=15.24 sec
 ARIMA(3,0,1)(1,1,1)[12] intercept   : AIC=inf, Time=8.65 sec
 ARIMA(3,0,2)(2,1,0)[12] intercept   : AIC=1132.912, Time=14.56 sec
 ARIMA(2,0,2)(2,1,0)[12] intercept   : AIC=1142.858, Time=6.22 sec
 ARIMA(3,0,1)(2,1,0)[12]             : AIC=1135.916, Time=3.30 sec

Best model:  ARIMA(3,0,1)(2,1,0)[12] intercept
Total fit time: 130.913 seconds
```

SARIMAX Results

Dep. Variable:	y	**No. Observations:**	731
Model:	SARIMAX(3, 0, 1)x(2, 1, [], 12)	**Log Likelihood**	-558.449
Date:	Thu, 21 Apr 2022	**AIC**	1132.898
Time:	20:39:22	**BIC**	1169.520
Sample:	0	**HQIC**	1147.037
	- 731		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
intercept	0.0052	0.003	1.779	0.075	-0.001	0.011
ar.L1	1.3864	0.055	25.180	0.000	1.278	1.494
ar.L2	0.0927	0.070	1.325	0.185	-0.044	0.230
ar.L3	-0.4888	0.037	-13.172	0.000	-0.561	-0.416
ma.L1	-0.8941	0.054	-16.578	0.000	-1.000	-0.788
ar.S.L12	-0.7110	0.038	-18.840	0.000	-0.785	-0.637
ar.S.L24	-0.3823	0.039	-9.810	0.000	-0.459	-0.306
sigma2	0.2733	0.016	17.572	0.000	0.243	0.304

Ljung-Box (L1) (Q):	0.44	**Jarque-Bera (JB):**	2.19
Prob(Q):	0.50	**Prob(JB):**	0.33
Heteroskedasticity (H):	1.07	**Skew:**	0.01
Prob(H) (two-sided):	0.59	**Kurtosis:**	2.73

```
# Use index to create a dataframe split and remove old entries
teleco_df['Day'] = teleco_df.index
teleco_train = teleco_df.iloc[:len(teleco_df) - 365]

teleco_train['teleco_train'] = teleco_train['Revenue']
del teleco_train['Day']
del teleco_train['Revenue']
```

```
# Test telco dataframe set
teleco_test = teleco_df.iloc[len(teleco_df) - 365:]

teleco_test['teleco_test'] = teleco_test['Revenue']
del teleco_test['Day']
del teleco_test['Revenue']
```

```
# Plot training and test sets
plt.plot(teleco_train, color='green')
plt.plot(teleco_test, color='blue')
plt.title('Training and Test split')
plt.xlabel('Days')
plt.ylabel('Revenue (Millions)')
sns.set()
plt.show()
```


```
# Create SARIMAX model
model = sm.tsa.SARIMAX(teleco_train,
                       order=(3, 0, 2),
                       seasonal_order=(2, 1, 0, 12),
                       enforce_stationarity=False,
                       enforce_invertibility=False)

SARIMAX_results = model.fit()

# Print SARIMAX Table
print(SARIMAX_results.summary())
```

```
                                SARIMAX Results
==========================================================================================
Dep. Variable:                      teleco_train   No. Observations:                  366
Model:             SARIMAX(3, 0, 2)x(2, 1, [], 12)   Log Likelihood                -247.972
Date:                             Thu, 21 Apr 2022   AIC                            511.945
Time:                                     20:39:24   BIC                            542.264
Sample:                                          0   HQIC                           524.043
                                               - 366
Covariance Type:                               opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          1.3587      0.117     11.635      0.000       1.130       1.588
ar.L2          0.0684      0.181      0.378      0.705      -0.286       0.423
ar.L3         -0.4383      0.109     -4.014      0.000      -0.652      -0.224
ma.L1         -0.8505      0.127     -6.711      0.000      -1.099      -0.602
ma.L2          0.0593      0.119      0.499      0.618      -0.173       0.292
ar.S.L12      -0.8350      0.063    -13.343      0.000      -0.958      -0.712
ar.S.L24      -0.4178      0.060     -6.929      0.000      -0.536      -0.300
sigma2         0.2660      0.023     11.636      0.000       0.221       0.311
===================================================================================
Ljung-Box (L1) (Q):                   0.02   Jarque-Bera (JB):                 1.22
Prob(Q):                              0.89   Prob(JB):                         0.54
Heteroskedasticity (H):               1.03   Skew:                             0.03
Prob(H) (two-sided):                  0.88   Kurtosis:                         2.71
===================================================================================
```

Previously during the adfuller analysis we observed our data was not stationary as required for the Arima analysis. The data has been manipulated to convert the dataset into a stationary frame. Furthermore, the data has been fitted into a required model to allow analysis in the ARIMA model in the subsequent section.

D3. Forecasting using Arima Model

We will be using the ARIMA model to observe forecasting. To accomplish this, we will create an initial forecast, summarize the confidence intervals, create a prediction on our testing set, and then retest on the full set based on the prediction's outcome.

```
# Create SARIMAX forecast
result = SARIMAX_results.get_forecast()

# Print forecast analysis
test_1 = teleco_test['teleco_test'].values.astype('float32')
forecast = result.predicted_mean
print('Expected Result: %.2f' % forecast)
print('Forecast Result: %.2f' % test_1[0])
print('Standard Error: %.2f' % result.se_mean)
```

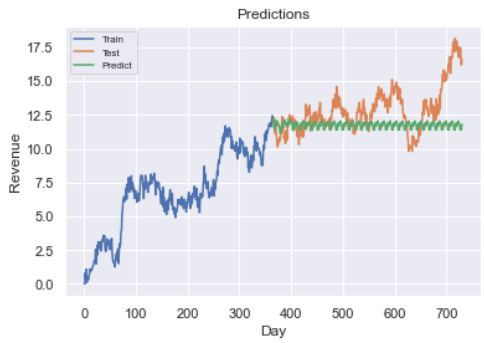
```
Expected Result: 12.24
Forecast Result: 11.85
Standard Error: 0.52
```

```
# Create four confidence intervals
intervals = [0.2, 0.1, 0.05, 0.01]
for a in intervals:
 ci = result.conf_int(alpha=a)
 print('%.1f%% Confidence Interval: %.2f between %.2f and %.2f' % ((1 - a) * 100, forecast, ci['lower teleco_train'], ci['upper t

ci
```

```
80.0% Confidence Interval: 12.24 between 11.58 and 12.90
90.0% Confidence Interval: 12.24 between 11.39 and 13.09
95.0% Confidence Interval: 12.24 between 11.23 and 13.25
99.0% Confidence Interval: 12.24 between 10.91 and 13.57
```

	lower teleco_train	upper teleco_train
366	10.910625	13.567535

```
start = len(teleco_train)
end = len(teleco_train) + len(teleco_test) - 1

# SARIMAX Prediction Plot
predictions = SARIMAX_results.predict(start, end, typ = 'levels').rename('Predictions')
plt.plot(teleco_train, label = 'Train')
plt.plot(teleco_test, label = 'Test')
plt.plot(predictions, label = 'Predict')
plt.title('Predictions')
plt.xlabel('Day')
plt.ylabel('Revenue')
plt.legend(loc='upper left', fontsize = 8)
plt.show()
```



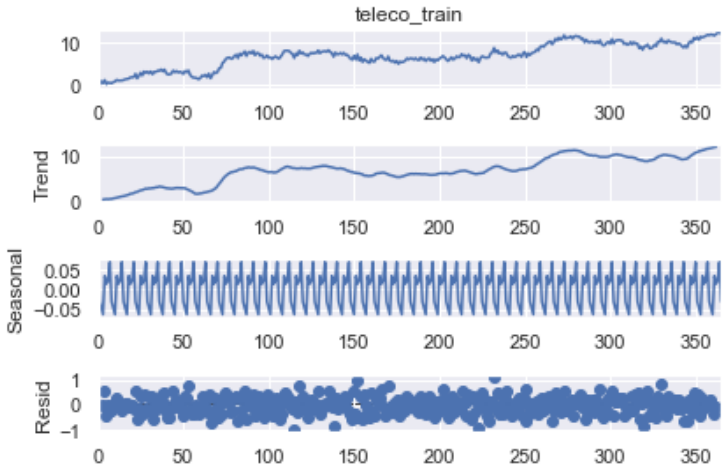
```
# Use full model to train dataset
model = sm.tsa.statespace.SARIMAX(teleco_df['Revenue'],
                                  order=(3, 0, 2),
                                  seasonal_order=(2, 1, 0, 12),
                                  enforce_stationarity=False,
                                  enforce_invertibility=False)

results = model.fit()
```

```
# Create forecast from results
forecast = results.predict(start = len(teleco_df['Revenue']),
                           end = (len(teleco_df['Revenue']) - 1) + 365,
                           typ = 'level').rename('Teleco Forecast')

# Plot forecast
teleco_df['Revenue'].plot(figsize = (12, 5), legend = True)
forecast.plot(legend = True)
```

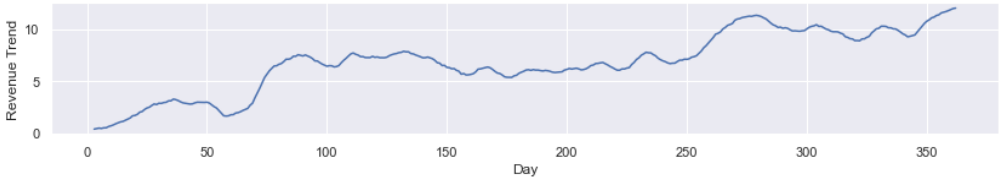
```
<AxesSubplot:>
```

D4. Output and Calculations

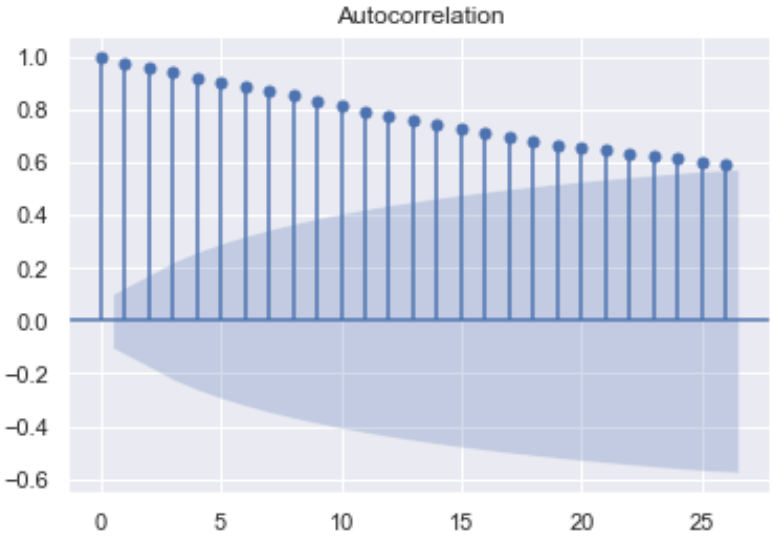
Decompose Elements



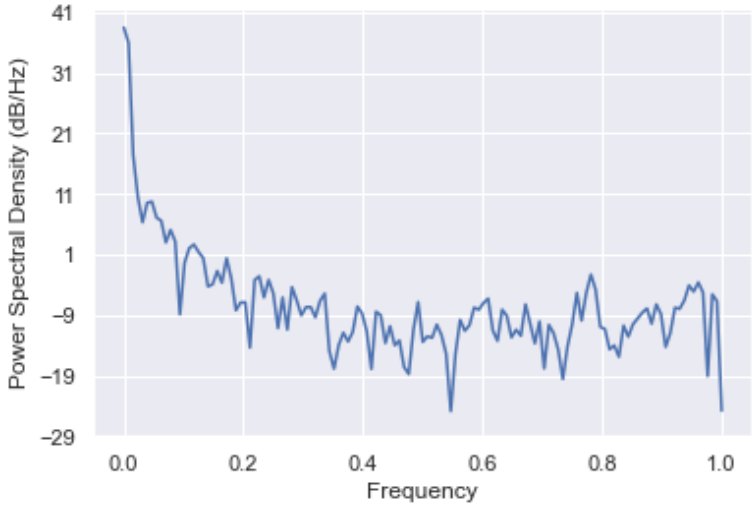
Decompose Trend



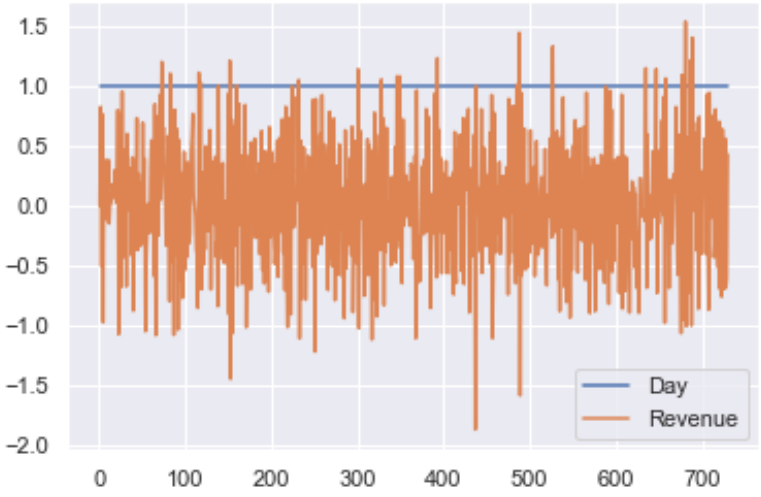
Autocorrelation



Spectral Density



ARIMA Difference



Sarimax Results

SARIMAX Results

Dep. Variable:	y	**No. Observations:**	731
Model:	SARIMAX(3, 0, 1)x(2, 1, [], 12)	**Log Likelihood**	-558.449
Date:	Fri, 22 Apr 2022	**AIC**	1132.898
Time:	19:37:28	**BIC**	1169.520
Sample:	0	**HQIC**	1147.037
	- 731		
Covariance Type:	opg		

Covariance Type:						
	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
intercept	0.0052	0.003	1.779	0.075	-0.001	0.011
ar.L1	1.3864	0.055	25.180	0.000	1.278	1.494
ar.L2	0.0927	0.070	1.325	0.185	-0.044	0.230
ar.L3	-0.4888	0.037	-13.172	0.000	-0.561	-0.416
ma.L1	-0.8941	0.054	-16.578	0.000	-1.000	-0.788
ar.S.L12	-0.7110	0.038	-18.840	0.000	-0.785	-0.637
ar.S.L24	-0.3823	0.039	-9.810	0.000	-0.459	-0.306
sigma2	0.2733	0.016	17.572	0.000	0.243	0.304

Ljung-Box (L1) (Q):	0.44	**Jarque-Bera (JB):**	2.19
Prob(Q):	0.50	**Prob(JB):**	0.33
Heteroskedasticity (H):	1.07	**Skew:**	0.01
Prob(H) (two-sided):	0.59	**Kurtosis:**	2.73

SARIMAX Results

```
==============================================================================
============
Dep. Variable:                          teleco_train   No. Observations:
366
Model:             SARIMAX(3, 0, 2)x(2, 1, [], 12)   Log Likelihood
-247.972
Date:                             Fri, 22 Apr 2022   AIC
511.945
Time:                                     19:32:59   BIC
542.264
Sample:                                          0   HQIC
524.043
                                               - 366
Covariance Type:                               opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          1.3587      0.117     11.635      0.000       1.130       1.588
ar.L2          0.0684      0.181      0.378      0.705      -0.286       0.423
ar.L3         -0.4383      0.109     -4.014      0.000      -0.652      -0.224
ma.L1         -0.8505      0.127     -6.711      0.000      -1.099      -0.602
ma.L2          0.0593      0.119      0.499      0.618      -0.173       0.292
ar.S.L12      -0.8350      0.063    -13.343      0.000      -0.958      -0.712
ar.S.L24      -0.4178      0.060     -6.929      0.000      -0.536      -0.300
sigma2         0.2660      0.023     11.636      0.000       0.221       0.311
==============================================================================
====
Ljung-Box (L1) (Q):                   0.02   Jarque-Bera (JB):
1.22
Prob(Q):                              0.89   Prob(JB):
0.54
Heteroskedasticity (H):               1.03   Skew:
0.03
Prob(H) (two-sided):                  0.88   Kurtosis:
2.71
==============================================================================
```

Sarimax Forecast

```
Expected Result: 12.24
Forecast Result: 11.85
Standard Error: 0.52
```

Confidence Intervals

```
80.0% Confidence Interval: 12.24 between 11.58 and 12.90
90.0% Confidence Interval: 12.24 between 11.39 and 13.09
95.0% Confidence Interval: 12.24 between 11.23 and 13.25
99.0% Confidence Interval: 12.24 between 10.91 and 13.57
```

	lower teleco_train	upper teleco_train
366	10.910625	13.567535

MSE & RMSE Calculations

```
# Calculate MSE
MSE = mean_squared_error(teleco_test['teleco_test'], predictions)
print('MSE: ', round(MSE, 4))
```

```
MSE:  4.0294
```

```
# Calculate RMSE
RMSE = rmse(teleco_test['teleco_test'], predictions)
print('RMSE: ', round(RMSE, 4))
```

```
RMSE:  2.0073
```

D5. Code

Code provided in full in calculations above.

Part V: Data Summary and Implications

E1. Results

1. **Selection of an ARIMA model**
   For this Time Series analysis, the optimal model selected from our ARIMA analysis is the Sarimax model. From analyzing the generated Arima fitted data, the Sarimax model possess reliable d, p, and q values as seen in our calculations.

SARIMAX Results

Dep. Variable:	y	**No. Observations:**	731
Model:	SARIMAX(3, 0, 1)x(2, 1, [], 12)	**Log Likelihood**	-558.449
Date:	Fri, 22 Apr 2022	**AIC**	1132.898
Time:	19:42:44	**BIC**	1169.520
Sample:	0	**HQIC**	1147.037
	- 731		
Covariance Type:	opg		

   In addition, the Sarimax results contains an optimal AIC metric as seen below.

SARIMAX Results

Dep. Variable:	y	**No. Observations:**	731
Model:	SARIMAX(3, 0, 1)x(2, 1, [], 12)	**Log Likelihood**	-558.449
Date:	Fri, 22 Apr 2022	**AIC**	1132.898
Time:	19:42:44	**BIC**	1169.520
Sample:	0	**HQIC**	1147.037
	- 731		
Covariance Type:	opg		

2. **Prediction interval of the forecast**
   The Prediction interval was tested at ranges of 0.2, 0.1, 0.05, and 0.01 resulting in the following confidence intervals:

```
80.0% Confidence Interval: 12.24 between 11.58 and 12.90
90.0% Confidence Interval: 12.24 between 11.39 and 13.09
95.0% Confidence Interval: 12.24 between 11.23 and 13.25
99.0% Confidence Interval: 12.24 between 10.91 and 13.57
```

   The intervals show at 95.0% confidence the observations will fall inside the range between 11.23 & 13.25.

3. **Justification of the forecast length**

The dataset provided contains a period of two years in length and our forecast was selected for a period of one year. A one-year period provides sufficient information for stakeholders and executives to make informed decisions for continued operations. The previous two years show an overall upward trend in revenue; however, the forecast provides an unreliable flat and sinusoidal shape that does not provide meaningful insight.

4. **Model evaluation procedure and error metric**

We have calculated Mean Square Error and Root Mean Square Error as follows:

```
# Calculate MSE
MSE = mean_squared_error(teleco_test['teleco_test'], predictions)
print('MSE: ', round(MSE, 4))
```

```
MSE:  4.0294
```

```
# Calculate RMSE
RMSE = rmse(teleco_test['teleco_test'], predictions)
print('RMSE: ', round(RMSE, 4))
```

```
RMSE:  2.0073
```

The lower the MSE and RMSE values obtained, the more accurate and reliable our analysis becomes.

E2. Annotated Visualization

```
# Create predictions for test set
start = len(teleco_train)
end = len(teleco_train) + len(teleco_test) - 1

# Plot predictions
predictions = SARIMAX_results.predict(start, end, typ = 'levels').rename('Predictions')
plt.plot(teleco_train, label = 'Train')
plt.plot(teleco_test, label = 'Test')
plt.plot(predictions, label = 'Prediction')
plt.title('Revenue Predictions')
plt.xlabel('Day')
plt.ylabel('Revenue')
plt.legend(loc='upper left', fontsize = 8)
plt.show()
```


```
# Create forecast from prediction
forecast = results.predict(start = len(teleco_df['Revenue']),
                           end = (len(teleco_df['Revenue']) - 1) + 365,
                           typ = 'level').rename('Teleco Forecast')

# Plot forecast
teleco_df['Revenue'].plot(figsize = (12, 5), legend = True)
forecast.plot(legend = True)
```

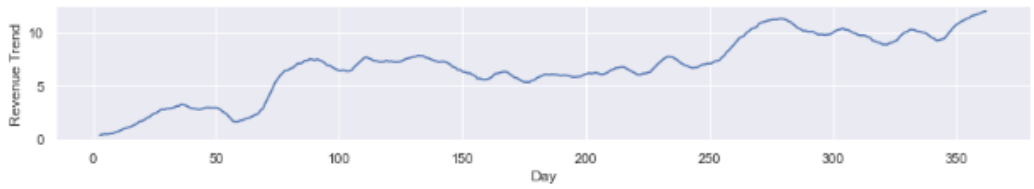
```
<AxesSubplot:>
```

E3. Recommendations

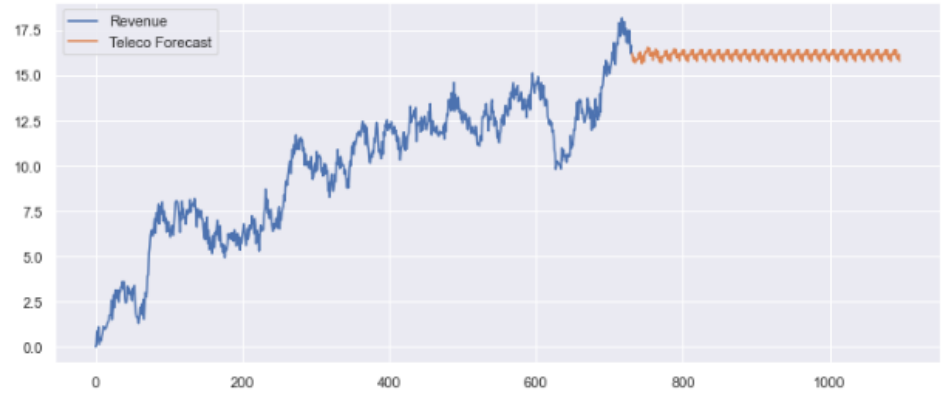
The initial research question proposed is:

Will a Time Series analysis of the dataset provide meaningful insights of patterns and trends in the data and allow us to forecast accurate predictions for the future?

In this Time Series analysis our greatest insight is the observation that over the two-year period in the provided dataset there is an upward trend in revenue as seen in the following graph.



The Time Series analysis shows the following prediction of revenue for the next year based on analysis of the previous two years:



The Time Series forecast begins between the 600 and 800 day period on the x-axis and the results are unremarkable. After observing a noticeable upward trend in revenue for the first 600 days, the time series forecast shows a horizontal trend neither indicating a rise or loss in profits. It is possible that the algorithm predicts the horizontal line to be the apex of possible revenue for the company signifying significant changes are necessary to Telcom operations or sales in order to break the revenue barrier. However, the more likely scenario is that the time series analysis is not the proper technique for casting a revenue forecast or the dataset needs to be refined.

We can say with certainty that there appears an obvious upward trend in revenue representing a positive effect of the Telcom companies' operations and with previous recommendations in prior analysis that profits are likely to continue and furthermore increase. But in order to have a meaningful Time Series Forecast we will need to further examine and develop the dataset by increasing the number of years observed and producing more reliable data that can be analyzed.

Part VI: Reporting

F. Reporting

The required Jupyter Notebook file has been exported and attached to the submission under "Task 1 – Time Series Jupyter Notebook.pdf".

G. Sources for Third-Party Code

Time Series Modeling, pt. 3: Seasonal-trend decomposition. | notebook.community. (n.d.). Retrieved April 22, 2022, from https://notebook.community/jrmontag/Data-Science-45min-Intros/time-series/03%20-%20Seasonal-Trend%20Decomposition

DataCamp. (n.d.). *Time series with Python*. DataCamp. Retrieved April 23, 2022, from https://www.datacamp.com/tracks/time-series-with-python

DataCamp. (n.d.). *Time Series Analysis in python course*. DataCamp. Retrieved April 23, 2022, from https://www.datacamp.com/courses/introduction-to-time-series-analysis-in-python

DataCamp. (n.d.). *Manipulating time series data in python course*. DataCamp. Retrieved April 23, 2022, from https://www.datacamp.com/courses/manipulating-time-series-data-in-python

DataCamp. (n.d.). *Visualizing Time Series data in python course*. DataCamp. Retrieved April 23, 2022, from https://www.datacamp.com/courses/visualizing-time-series-data-in-python

H. Sources

Forecasting: Principles and practice (2nd ed). Otexts. Chapter 6 Time series decomposition. (n.d.). Retrieved April 22, 2022, from https://otexts.com/fpp2/decomposition.html

Peixeiro, M. (2022, April 3). *The Complete Guide to Time Series Analysis and forecasting*. Medium. Retrieved April 20, 2022, from https://towardsdatascience.com/the-complete-guide-to-time-series-analysis-and-forecasting-70d476bfe775

Statsmodels.tsa.stattools.adfuller¶. statsmodels. (n.d.). Retrieved April 23, 2022, from https://www.statsmodels.org/stable/generated/statsmodels.tsa.stattools.adfuller.html