PERFORMANCE ASSESSMENT

Task 1 | Time Series Analysis

KAILI HAMILTON

Masters of Science in Data Analytics, Western Governors University

Course: D213

Instructor: Dr. Kesselly Kamara

Program Mentor: Krissy Bryant

March, 2024

TABLE OF CONTENTS

A1

One research question that I will answer using time series analysis is, “What is the forecast of the revenue for the telecommunications company for the next three months (90 days)?”

A2

The objectives and goals of this time series analysis is to make a ninety-day forecast for the company’s revenue by building a predictive model that analyzes trends and seasonality of the revenue for the past two years.

B

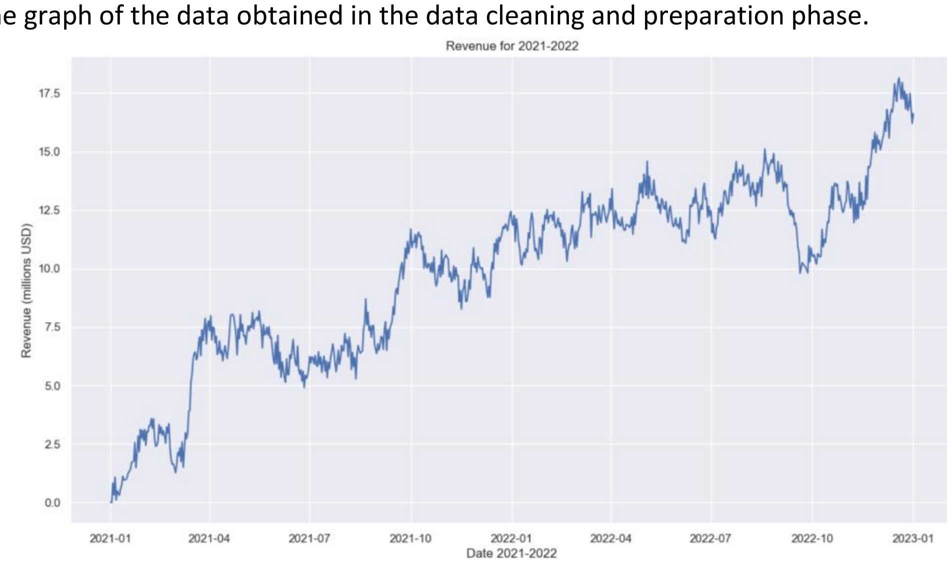
The assumptions of a time series model are that “the time series data should be stationary. This means that it is normally distributed and the mean and variance are constant over that period of time. The series has no trends and it is not growing nor shrinking. The variance is constant and the autocorrelation is constant. [Furthermore], the error in time series analysis should be uncorrelated, there should be no outliers in the series, and the residuals are not autocorrelated” (Karama, n.d.).

“Autocorrelation measures a set of current values against a set of past values to see if they correlate. This also means that future values can be predicted as a linear function of past values” (Kamara, n.d.).

The residuals of a time series should not be autocorrelated (Kamara, n.d.).

C1

Here is a line graph of the data obtained in the data cleaning and preparation phase.



C2

The time step formatting of this realization is daily. Revenue is recorded each day and there are no gaps in the measurement – there is a value for each day for the 731 days of the data set and there are no null values. The length of the sequence is 731 days.

C3

The stationarity of the time series was evaluated using the Augmented Dickey-Fuller Test (ADFuller). The time series was found to be non-stationary, and thus required differencing. The ADFuller Test has a null hypothesis that the time series is non-stationary and the alternative hypothesis is that the data is stationary. The p-value calculated from this test exceeded our critical value of 0.05 and thus we fail to reject the null hypothesis. This means that the data is non-stationary.

```
from statsmodels.tsa.stattools import adfuller
```

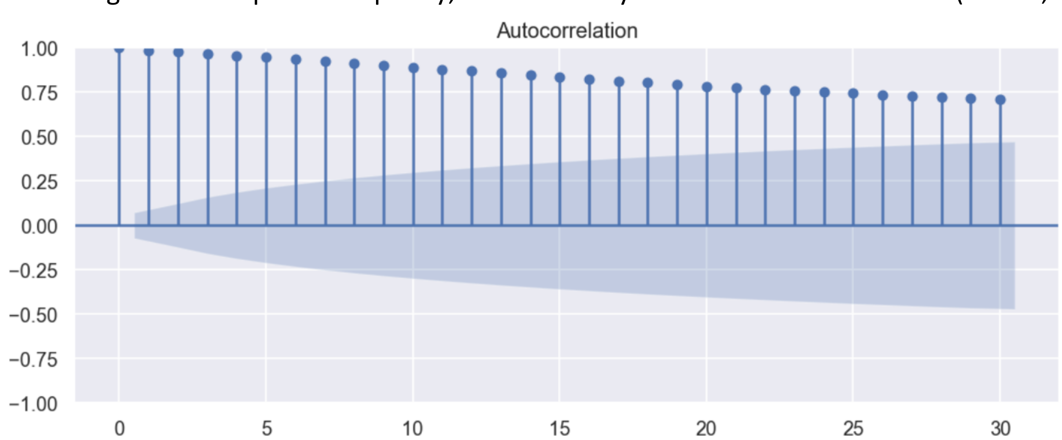
```
result = adfuller(df['Revenue'])
print("Test Statistics: ", result[0])
print("p-value: ", result[1])
print("Critical Values: ", result[4])
```

```
Test Statistics:  -1.9246121573101809
p-value:  0.32057281507939783
Critical Values:  {'1%': -3.4393520240470554, '5%': -2.8655128165959236, '10%': -2.5688855736949163}
```

```
#null hypothesis: time series is non-stationary
#alternative hypothesis: time series is stationary

if result[1] <= 0.05:
    print("Reject the null hypothesis, the time series is stationary.")
else:
    print("Fail to reject the null hypothesis, the time series is non-stationary.")
```

```
Fail to reject the null hypothesis, the time series is non-stationary.
```

Also, the Autocorrelation Function (ACF) clearly indicates that the time series is non-stationary because values at each lag do not drop to zero quickly, like stationary time series should behave (Reider, n.d.).



Visually, the results of the ADFuller test makes sense because we can see the time series has a trend, upward over time (see the line graph in part C1).

Because the time series is non-stationary it needs to be differenced. It only takes one order of differencing for the null hypothesis of the ADFuller Test to be rejected.

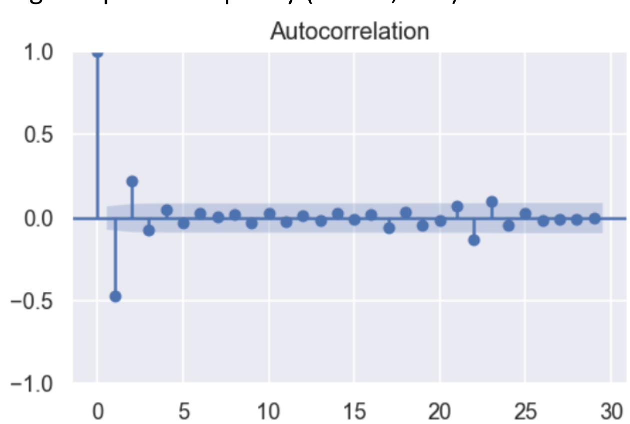
```
df_stationary = df.diff().dropna() #differenced the data
df_stationary.head()
```

	Revenue
Date	
2021-01-02	0.000793
2021-01-03	0.824749
2021-01-04	-0.505210
2021-01-05	0.762222
2021-01-06	-0.974900

```
result = adfuller(df_stationary['Revenue'])
print("Test Statistics: ", result[0])
print("p-value: ", result[1])
print("Critical Values: ", result[4])

#null hypothesis: time series is non-stationary
#alternative hypothesis: time series is stationary

if result[1] <= 0.05:
    print("Reject the null hypothesis, the time series is stationary.")
else:
    print("Fail to reject the null hypothesis, the time series is non-stationary.")
```

```
Test Statistics:  -44.874527193875984
p-value:  0.0
Critical Values:  {'1%': -3.4393520240470554, '5%': -2.8655128165959236, '10%': -2.5688855736949163}
Reject the null hypothesis, the time series is stationary.
```

So, now that the data are stationary with one order of differencing we can see this in the ACF plot because the values at the lags drop to zero quickly (Reider, n.d.).



C4

The steps used to prepare the data for analysis are explained below.

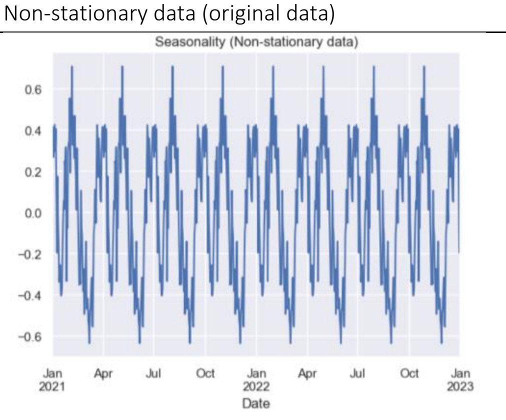
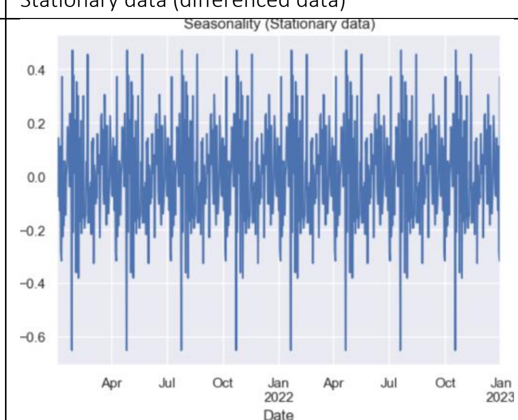
1. The data was imported into a pandas dataframe.
2. Exploratory data analysis was performed to determine the shape of the data (731 rows), if there were any null values (there were none), and visualizing the time series (line graph).
3. The time series was converted from day number to a date using date_time(). The starting date is set at 2021-01-01 and the ending date is 2023-01-01.
4. Next, the stationarity of the time series was determined. The ADFuller test and the ACF indicates that the time series is non-stationary.
5. The data were differenced with just one order of differencing to become stationary. ADFuller test and ACF were ran and plotted again to indicates that the time series is stationary with one order of differencing.
6. Lastly, the time series was split into a train/test split and the resulting dataframes were exported.

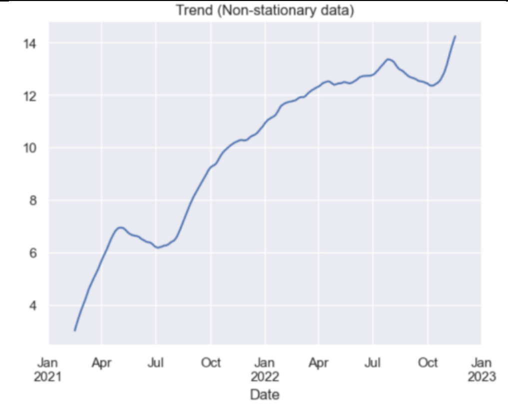
C5

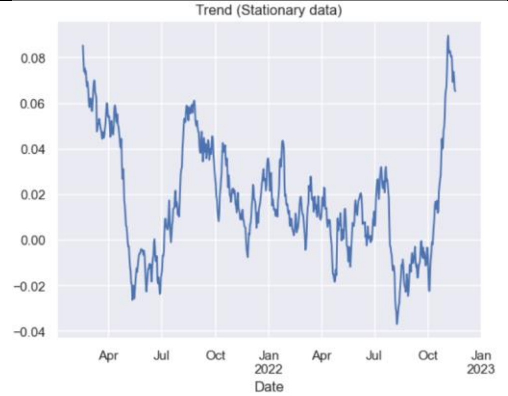
A copy of the cleaned data set, including the train/test split sets, has been provided with the submission of this performance assessment.

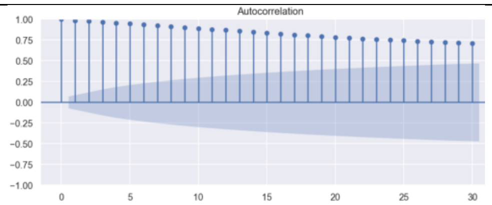
D1

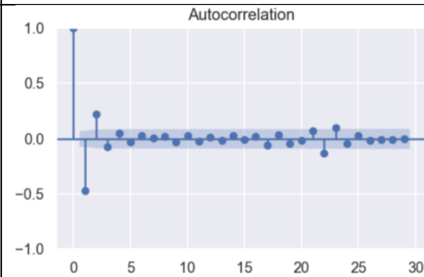
The time series has been analyzed. Below are annotated visualizations for the data analysis (Kamara, n.d.).

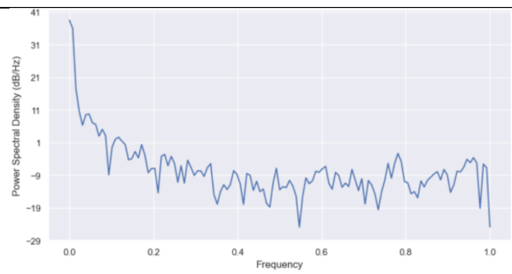
	Non-stationary data (original data)	Stationary data (differenced data)
Presence or lack of a seasonal component	 There is the presence of a seasonal component to the time series. The periodicity is approximately 90 days (3 months).	 Once the data have been differenced, the seasonality resembles white noise and the seasonal component has been removed.

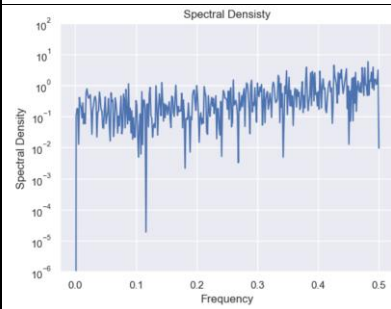
<table>
<tr>
<td>Trends</td>
<td>

There is an upward trend in the time series, as time increases, revenue increases.</td>
<td>

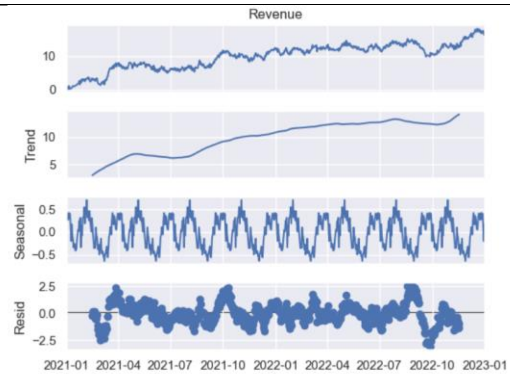
The trend has been removed in the differenced data.</td>
</tr>
<tr>
<td>Autocorrelation function (ACF)</td>
<td>

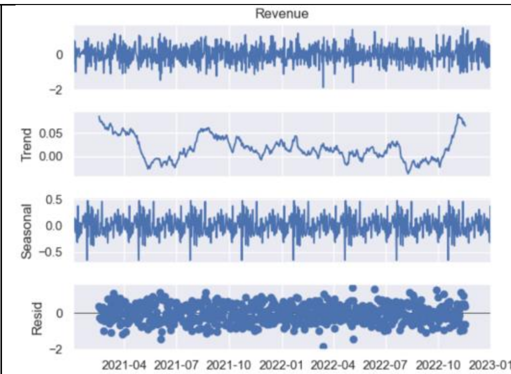
The ACF decreases slowly for increasing number of lags, which indicates non-stationarity.</td>
<td>

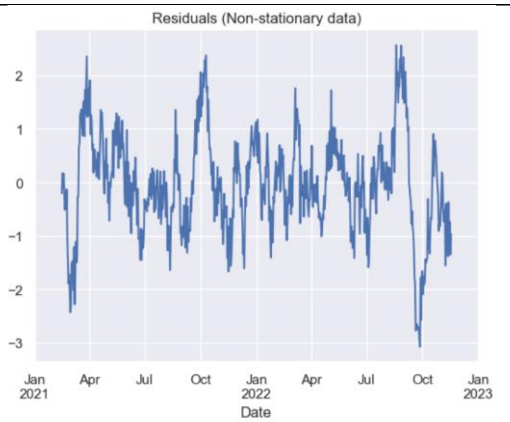
The ACF decreases to zero rapidly, an indication that the data is now stationary.</td>
</tr>
<tr>
<td>Spectral density</td>
<td>

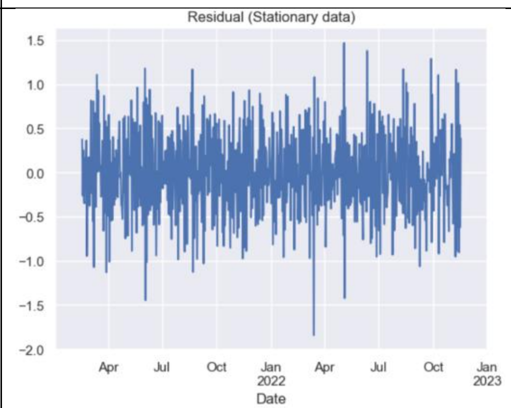
The peaks indicate frequencies at which the time series has significant variance.</td>
<td>

The differenced spectral density resembles white noise with low variance indicating there isn’t a specific periodic pattern.</td>
</tr>
</table>

<table>
<tr>
<td>Decomposed time series</td>
<td>

The decomposed time series shows the trend, seasonality, and residuals. These components are displayed and explained in their corresponding parts of this table.</td>
<td>

The decomposed time series shows the trend, seasonality, and residuals. These components are displayed and explained in their corresponding parts of this table.</td>
</tr>
<tr>
<td>Confirmation of the lack of trends in the residuals of the decomposed series</td>
<td>

The residual plot shows the time series after the trend and seasonality have been removed. The residuals here show a pattern of alternating positive and negative values.</td>
<td>

The residuals here have a mean of zero and a constant variance and are not autocorrelated, confirming the lack of trends in the residuals.</td>
</tr>
</table>

D2

An ARIMA (autoregressive integrated moving average) model that accounts for the observed trend and seasonality is identified and justified here.

To find the parameters to use in the ARIMA model we need those that produce the lowest AIC score. These parameters are (p, q, d). “P is the order of the autoregressive part, specifically the lag of differencing series. D is the degree of differencing. Q is the order of the moving average part, specifically the lag of errors. Auto-Arima was performed to find the parameters for the best model fit.
A variety of parameters were used to fit the model (see the code file for all trials), but the best performing model, the one with the lowest AIC score, was obtained from the parameters produced by

auto_arima(). These parameters are (1, 1, 0). This is true for the whole data set, the training set, and the testing set, displayed below.

Original data:

```
Best model:  ARIMA(1,1,0)(0,0,0)[0] intercept
Total fit time: 1.235 seconds
```

[49]:

SARIMAX Results

Dep. Variable:	y	**No. Observations:**	731
Model:	SARIMAX(1, 1, 0)	**Log Likelihood**	-488.561
Date:	Tue, 05 Mar 2024	**AIC**	983.122
Time:	18:06:02	**BIC**	996.901
Sample:	01-01-2021	**HQIC**	988.438
	- 01-01-2023		
Covariance Type:	opg		

Training data:

```
Best model:  ARIMA(1,1,0)(0,0,0)[0]
Total fit time: 1.335 seconds
```

SARIMAX Results

Dep. Variable:	y	**No. Observations:**	638
Model:	SARIMAX(1, 1, 0)	**Log Likelihood**	-420.775
Date:	Tue, 05 Mar 2024	**AIC**	845.551
Time:	18:06:04	**BIC**	854.464
Sample:	01-01-2021	**HQIC**	849.011
	- 09-30-2022		
Covariance Type:	opg		

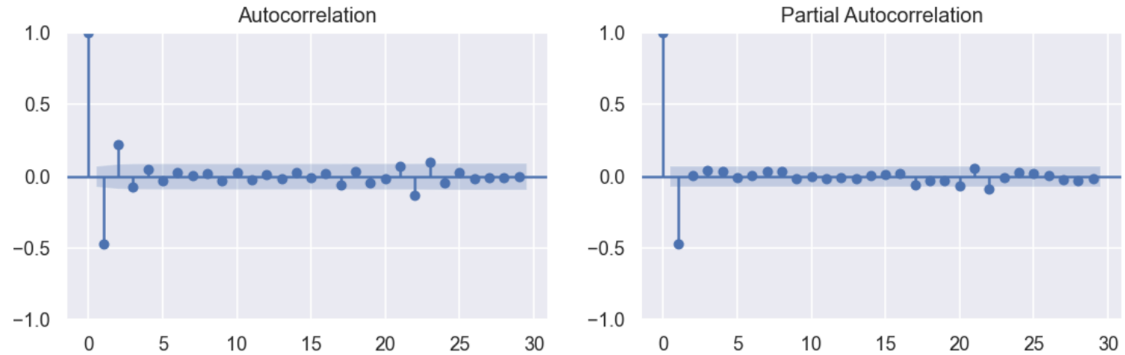
Testing data:

```
Best model:  ARIMA(1,1,0)(0,0,0)[0] intercept
Total fit time: 0.366 seconds
```

SARIMAX Results

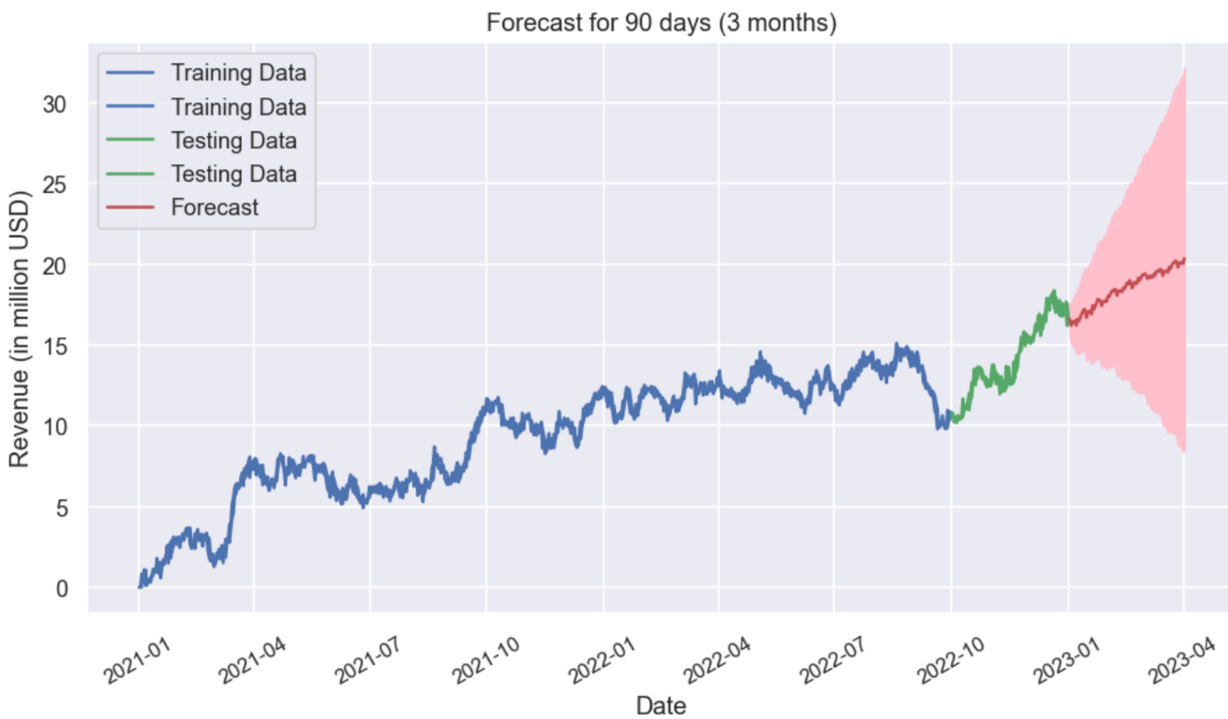
Dep. Variable:	y	**No. Observations:**	93
Model:	SARIMAX(1, 1, 0)	**Log Likelihood**	-66.623
Date:	Tue, 05 Mar 2024	**AIC**	139.246
Time:	18:06:05	**BIC**	146.811
Sample:	10-01-2022	**HQIC**	142.299
	- 01-01-2023		
Covariance Type:	opg		

The ACF and PACF functions help determine the p and q parameters for the ARIMA model.



D3

A forecast for 90 days was performed using the ARIMA model identified in part D2. A visualization of the training set, testing, set, and forecast is displayed here.



D4

The output and calculations of the analysis are as follows:

Stationarity:

```
from statsmodels.tsa.stattools import adfuller
```

```
result = adfuller(df['Revenue'])
print("Test Statistics: ", result[0])
print("p-value: ", result[1])
print("Critical Values: ", result[4])
```

```
Test Statistics:  -1.9246121573101809
p-value:  0.32057281507939783
Critical Values:  {'1%': -3.4393520240470554, '5%': -2.8655128165959236, '10%': -2.5688855736949163}
```

```
#null hypothesis: time series is non-stationary
#alternative hypothesis: time series is stationary

if result[1] <= 0.05:
    print("Reject the null hypothesis, the time series is stationary.")
else:
    print("Fail to reject the null hypothesis, the time series is non-stationary.")
```

```
Fail to reject the null hypothesis, the time series is non-stationary.
```

```
df_stationary = df.diff().dropna() #differenced the data
df_stationary.head()
```

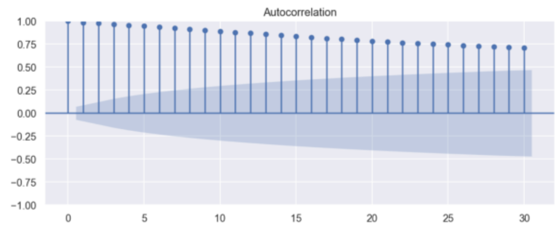
	Revenue
Date	
2021-01-02	0.000793
2021-01-03	0.824749
2021-01-04	-0.505210
2021-01-05	0.762222
2021-01-06	-0.974900

```
result = adfuller(df_stationary['Revenue'])
print("Test Statistics: ", result[0])
print("p-value: ", result[1])
print("Critical Values: ", result[4])

#null hypothesis: time series is non-stationary
#alternative hypothesis: time series is stationary

if result[1] <= 0.05:
    print("Reject the null hypothesis, the time series is stationary.")
else:
    print("Fail to reject the null hypothesis, the time series is non-stationary.")
```

```
Test Statistics:  -44.874527193875984
p-value:  0.0
Critical Values:  {'1%': -3.4393520240470554, '5%': -2.8655128165959236, '10%': -2.5688855736949163}
Reject the null hypothesis, the time series is stationary.
```

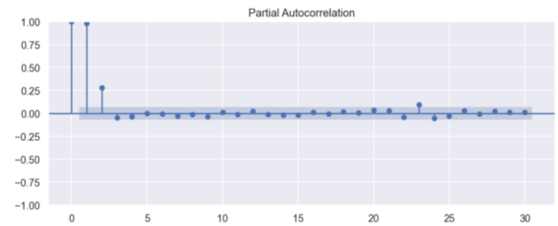
ACF & PACF:

```
#grpahing the ACF plot

fig = plt.figure(figsize=(10,8))
ax1 = fig.add_subplot(211)
fig = sm.graphics.tsa.plot_acf(df.Revenue, lags=30, ax=ax1) # @ 30 lags
```

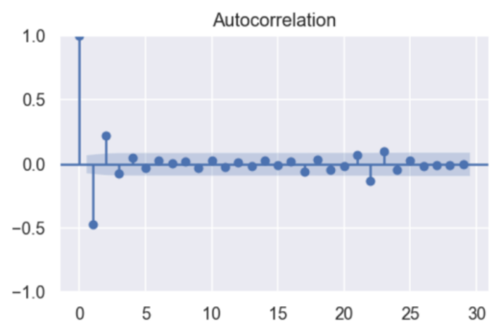



```
#graphing the PACF plot

fig = plt.figure(figsize=(10,8))
ax2 = fig.add_subplot(212)
fig=sm.graphics.tsa.plot_pacf(df.Revenue, lags=30, ax=ax2) # @ 30 lags
plt.show
```

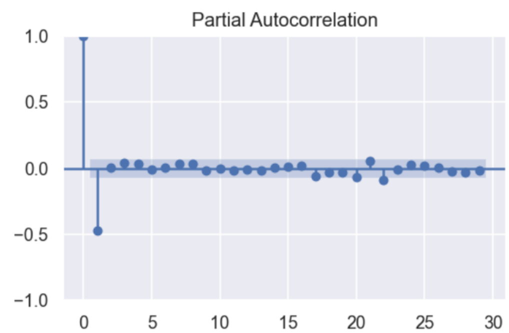
```
<function matplotlib.pyplot.show(close=None, block=None)>
```




```
plot_acf(df_stationary)
#plot_pacf(df_stationary)
plt.show()
```




```
plot_pacf(df_stationary)
plt.show()
```



Auto Arima | data set:

auto_arima

```
from pmdarima import auto_arima
import warnings
warnings.filterwarnings('ignore')
```

```
pip install pmdarima
```

```
model_auto = auto_arima(df['Revenue'], trace=True, suppress_warnings=True)
model_auto.summary()
```

```
Performing stepwise search to minimize aic
 ARIMA(2,1,2)(0,0,0)[0] intercept   : AIC=987.305, Time=0.45 sec
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=1162.819, Time=0.05 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=983.122, Time=0.07 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=1019.369, Time=0.09 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=1162.139, Time=0.05 sec
 ARIMA(2,1,0)(0,0,0)[0] intercept   : AIC=985.104, Time=0.09 sec
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=985.106, Time=0.05 sec
 ARIMA(2,1,1)(0,0,0)[0] intercept   : AIC=986.045, Time=0.35 sec
 ARIMA(1,1,0)(0,0,0)[0]             : AIC=984.710, Time=0.03 sec

Best model:  ARIMA(1,1,0)(0,0,0)[0] intercept
Total fit time: 1.235 seconds
```

SARIMAX Results

Dep. Variable:	y	**No. Observations:**	731
Model:	SARIMAX(1, 1, 0)	**Log Likelihood**	-488.561
Date:	Tue, 05 Mar 2024	**AIC**	983.122
Time:	18:06:02	**BIC**	996.901
Sample:	01-01-2021	**HQIC**	988.438
	- 01-01-2023		
Covariance Type:	opg		

Manual Arima | data set:

```
model_manual = ARIMA(df['Revenue'], order=(1,1,0))
model_manual = model_manual.fit()
model_manual.summary()
```

SARIMAX Results

Dep. Variable:	Revenue	**No. Observations:**	731
Model:	ARIMA(1, 1, 0)	**Log Likelihood**	-490.355
Date:	Tue, 05 Mar 2024	**AIC**	984.710
Time:	18:06:02	**BIC**	993.896
Sample:	01-01-2021	**HQIC**	988.254
	- 01-01-2023		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-0.4667	0.033	-14.213	0.000	-0.531	-0.402
sigma2	0.2243	0.013	17.782	0.000	0.200	0.249

Ljung-Box (L1) (Q):	0.00	**Jarque-Bera (JB):**	2.07
Prob(Q):	0.98	**Prob(JB):**	0.36
Heteroskedasticity (H):	1.02	**Skew:**	-0.02
Prob(H) (two-sided):	0.89	**Kurtosis:**	2.74

```
model_manual_2 = ARIMA(df['Revenue'], order=(2,1,0))
model_manual_2 = model_manual_2.fit()
model_manual_2.summary()
```

SARIMAX Results

Dep. Variable:	Revenue	**No. Observations:**	731
Model:	ARIMA(2, 1, 0)	**Log Likelihood**	-490.320
Date:	Tue, 05 Mar 2024	**AIC**	986.641
Time:	18:06:03	**BIC**	1000.420
Sample:	01-01-2021	**HQIC**	991.957
	- 01-01-2023		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-0.4621	0.037	-12.472	0.000	-0.535	-0.389
ar.L2	0.0098	0.036	0.270	0.787	-0.061	0.081
sigma2	0.2243	0.013	17.760	0.000	0.200	0.249

Ljung-Box (L1) (Q):	0.03	**Jarque-Bera (JB):**	2.10
Prob(Q):	0.87	**Prob(JB):**	0.35
Heteroskedasticity (H):	1.02	**Skew:**	-0.02
Prob(H) (two-sided):	0.88	**Kurtosis:**	2.74

```
model_manual_3 = ARIMA(df['Revenue'], order=(2,1,1))
model_manual_3 = model_manual_3.fit()
model_manual_3.summary()
```

SARIMAX Results

Dep. Variable:	Revenue	**No. Observations:**	731
Model:	ARIMA(2, 1, 1)	**Log Likelihood**	-489.428
Date:	Tue, 05 Mar 2024	**AIC**	986.856
Time:	18:06:03	**BIC**	1005.228
Sample:	01-01-2021	**HQIC**	993.944
	- 01-01-2023		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.3234	0.287	1.126	0.260	-0.239	0.886
ar.L2	0.3936	0.127	3.096	0.002	0.144	0.643
ma.L1	-0.7772	0.297	-2.617	0.009	-1.359	-0.195
sigma2	0.2237	0.013	17.613	0.000	0.199	0.249

Ljung-Box (L1) (Q):	0.18	**Jarque-Bera (JB):**	2.13
Prob(Q):	0.67	**Prob(JB):**	0.34
Heteroskedasticity (H):	1.02	**Skew:**	-0.03
Prob(H) (two-sided):	0.88	**Kurtosis:**	2.74

Auto Arima | training set:

```
model_train_1 = auto_arima(X_train['Revenue'], trace=True, suppress_warnings=True)
model_train_1.summary()
```

```
Performing stepwise search to minimize aic
 ARIMA(2,1,2)(0,0,0)[0] intercept   : AIC=849.940, Time=0.26 sec
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=990.877, Time=0.06 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=845.820, Time=0.06 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=874.381, Time=0.08 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=989.550, Time=0.03 sec
 ARIMA(2,1,0)(0,0,0)[0] intercept   : AIC=847.765, Time=0.07 sec
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=847.775, Time=0.09 sec
 ARIMA(2,1,1)(0,0,0)[0] intercept   : AIC=849.047, Time=0.36 sec
 ARIMA(1,1,0)(0,0,0)[0]             : AIC=845.551, Time=0.03 sec
 ARIMA(2,1,0)(0,0,0)[0]             : AIC=847.461, Time=0.03 sec
 ARIMA(1,1,1)(0,0,0)[0]             : AIC=847.478, Time=0.05 sec
 ARIMA(0,1,1)(0,0,0)[0]             : AIC=874.373, Time=0.06 sec
 ARIMA(2,1,1)(0,0,0)[0]             : AIC=848.504, Time=0.15 sec

Best model:  ARIMA(1,1,0)(0,0,0)[0]
Total fit time: 1.335 seconds
```

SARIMAX Results

Dep. Variable:	y	**No. Observations:**	638
Model:	SARIMAX(1, 1, 0)	**Log Likelihood**	-420.775
Date:	Tue, 05 Mar 2024	**AIC**	845.551
Time:	18:06:04	**BIC**	854.464
Sample:	01-01-2021	**HQIC**	849.011
	- 09-30-2022		
Covariance Type:	opg		

Manual Arima | training set:

```
model_train_2 = ARIMA(X_train['Revenue'], order=(1, 1, 0))
model_train_2 = model_train_2.fit()
model_train_2.summary()
```

SARIMAX Results

Dep. Variable:	Revenue	**No. Observations:**	638
Model:	ARIMA(1, 1, 0)	**Log Likelihood**	-420.775
Date:	Tue, 05 Mar 2024	**AIC**	845.551
Time:	18:06:05	**BIC**	854.464
Sample:	01-01-2021	**HQIC**	849.011
	- 09-30-2022		
Covariance Type:	opg		

```
model_train_3 = ARIMA(X_train['Revenue'], order=(2, 1, 1)
model_train_3 = model_train_3.fit()
model_train_3.summary()
```

SARIMAX Results

Dep. Variable:	Revenue	**No. Observations:**	638
Model:	ARIMA(2, 1, 1)	**Log Likelihood**	-420.252
Date:	Tue, 05 Mar 2024	**AIC**	848.504
Time:	18:06:05	**BIC**	866.331
Sample:	01-01-2021	**HQIC**	855.425
	- 09-30-2022		
Covariance Type:	opg		

```
model_train_4 = ARIMA(X_train['Revenue'], order=(1, 1, 2))
model_train_4 = model_train_4.fit()
model_train_4.summary()
```

SARIMAX Results

Dep. Variable:	Revenue	**No. Observations:**	638
Model:	ARIMA(1, 1, 2)	**Log Likelihood**	-419.874
Date:	Tue, 05 Mar 2024	**AIC**	847.748
Time:	18:06:05	**BIC**	865.575
Sample:	01-01-2021	**HQIC**	854.669
	- 09-30-2022		
Covariance Type:	opg		

Auto Arima | test set:

```
model_test_1 = auto_arima(X_test['Revenue'], trace=True, suppress_warnings=True)
model_test_1.summary()
```

```
Performing stepwise search to minimize aic
 ARIMA(2,1,2)(0,0,0)[0] intercept   : AIC=143.398, Time=0.10 sec
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=171.250, Time=0.03 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=139.246, Time=0.02 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=146.022, Time=0.04 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=170.343, Time=0.01 sec
 ARIMA(2,1,0)(0,0,0)[0] intercept   : AIC=141.030, Time=0.03 sec
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=140.997, Time=0.04 sec
 ARIMA(2,1,1)(0,0,0)[0] intercept   : AIC=142.995, Time=0.07 sec
 ARIMA(1,1,0)(0,0,0)[0]             : AIC=140.859, Time=0.02 sec

Best model:  ARIMA(1,1,0)(0,0,0)[0] intercept
Total fit time: 0.366 seconds
```

SARIMAX Results

Dep. Variable:	y	**No. Observations:**	93
Model:	SARIMAX(1, 1, 0)	**Log Likelihood**	-66.623
Date:	Tue, 05 Mar 2024	**AIC**	139.246
Time:	18:06:05	**BIC**	146.811
Sample:	10-01-2022	**HQIC**	142.299
	- 01-01-2023		
Covariance Type:	opg		

Manual Arima | test set:

```
model_test_2 = ARIMA(X_test['Revenue'], order=(1, 1, 2))
model_test_2 = model_test_2.fit()
model_test_2.summary()
```

SARIMAX Results

Dep. Variable:	Revenue	**No. Observations:**	93
Model:	ARIMA(1, 1, 2)	**Log Likelihood**	-66.997
Date:	Tue, 05 Mar 2024	**AIC**	141.994
Time:	18:06:05	**BIC**	152.081
Sample:	10-01-2022	**HQIC**	146.065
	- 01-01-2023		
Covariance Type:	opg		

```
model_test_3 = ARIMA(X_test['Revenue'], order=(2, 1, 1))
model_test_3 = model_test_3.fit()
model_test_3.summary()
```

SARIMAX Results

Dep. Variable:	Revenue	**No. Observations:**	93
Model:	ARIMA(2, 1, 1)	**Log Likelihood**	-67.749
Date:	Tue, 05 Mar 2024	**AIC**	143.498
Time:	18:06:05	**BIC**	153.586
Sample:	10-01-2022	**HQIC**	147.570
	- 01-01-2023		
Covariance Type:	opg		

```
model_test_4 = ARIMA(X_test['Revenue'], order=(1, 1, 1))
model_test_4 = model_test_4.fit()
model_test_4.summary()
```

SARIMAX Results

Dep. Variable:	Revenue	**No. Observations:**	93
Model:	ARIMA(1, 1, 1)	**Log Likelihood**	-68.427
Date:	Tue, 05 Mar 2024	**AIC**	142.855
Time:	18:06:05	**BIC**	150.420
Sample:	10-01-2022	**HQIC**	145.908
	- 01-01-2023		
Covariance Type:	opg		

Arima Model:

ARIMA Result for Selected Order

```
model = ARIMA(df['Revenue'], order=(1,1,0), seasonal_order=(5, 1, 0, 12))
results = model.fit()
results.summary()
```

SARIMAX Results

Dep. Variable:	Revenue	**No. Observations:**	731
Model:	ARIMA(1, 1, 0)x(5, 1, 0, 12)	**Log Likelihood**	-536.240
Date:	Tue, 05 Mar 2024	**AIC**	1086.481
Time:	21:57:37	**BIC**	1118.516
Sample:	01-01-2021	**HQIC**	1098.850
	- 01-01-2023		
Covariance Type:	opg		

Forecasting:

```
# index future dates

index_future_dates = pd.date_range(start='2023-01-02', end='2023-03-31'
print(index_future_dates)
```

```
DatetimeIndex(['2023-01-02', '2023-01-03', '2023-01-04', '2023-01-05',
               '2023-01-06', '2023-01-07', '2023-01-08', '2023-01-09',
               '2023-01-10', '2023-01-11', '2023-01-12', '2023-01-13',
               '2023-01-14', '2023-01-15', '2023-01-16', '2023-01-17',
               '2023-01-18', '2023-01-19', '2023-01-20', '2023-01-21',
               '2023-01-22', '2023-01-23', '2023-01-24', '2023-01-25',
               '2023-01-26', '2023-01-27', '2023-01-28', '2023-01-29',
               '2023-01-30', '2023-01-31', '2023-02-01', '2023-02-02',
               '2023-02-03', '2023-02-04', '2023-02-05', '2023-02-06',
               '2023-02-07', '2023-02-08', '2023-02-09', '2023-02-10',
               '2023-02-11', '2023-02-12', '2023-02-13', '2023-02-14',
               '2023-02-15', '2023-02-16', '2023-02-17', '2023-02-18',
               '2023-02-19', '2023-02-20', '2023-02-21', '2023-02-22',
               '2023-02-23', '2023-02-24', '2023-02-25', '2023-02-26',
               '2023-02-27', '2023-02-28', '2023-03-01', '2023-03-02',
               '2023-03-03', '2023-03-04', '2023-03-05', '2023-03-06',
               '2023-03-07', '2023-03-08', '2023-03-09', '2023-03-10',
               '2023-03-11', '2023-03-12', '2023-03-13', '2023-03-14',
               '2023-03-15', '2023-03-16', '2023-03-17', '2023-03-18',
               '2023-03-19', '2023-03-20', '2023-03-21', '2023-03-22',
               '2023-03-23', '2023-03-24', '2023-03-25', '2023-03-26',
               '2023-03-27', '2023-03-28', '2023-03-29', '2023-03-30',
               '2023-03-31'],
              dtype='datetime64[ns]', freq='D')
```

```
results.forecast(90)
```

```
2023-01-02    16.594642
2023-01-03    16.201729
2023-01-04    16.417148
2023-01-05    16.375852
2023-01-06    16.497566
                ...
2023-03-28    19.810484
2023-03-29    20.022815
2023-03-30    20.119597
2023-03-31    20.079046
2023-04-01    20.038105
Freq: D, Name: predicted_mean, Length: 90, dtype: float64
```

```
results.summary()
```

SARIMAX Results

Dep. Variable:	Revenue	**No. Observations:**	731
Model:	ARIMA(1, 1, 0)x(5, 1, 0, 12)	**Log Likelihood**	-536.240
Date:	Tue, 05 Mar 2024	**AIC**	1086.481
Time:	18:06:13	**BIC**	1118.516
Sample:	01-01-2021	**HQIC**	1098.850
	- 01-01-2023		
Covariance Type:	opg		

```
prediction = pd.DataFrame(results.predict(n_periods=12), index=df.index)
prediction.columns = ['Revenue']
prediction
```

	Revenue
Date	
2021-01-01	0.000000
2021-01-02	0.000000
2021-01-03	0.000793
2021-01-04	0.825542
2021-01-05	0.320333

```
from statsmodels.tsa.arima.model import ARIMA
```

```
diff_forecast = results.get_forecast(steps=30)
mean_forecast = diff_forecast.predicted_mean
confidence_intervals = diff_forecast.conf_int()

lower_limits = confidence_intervals.loc[:, 'lower Revenue']
upper_limits = confidence_intervals.loc[:, 'upper Revenue']

prediction = results.get_prediction(start=len(df), end=len(df)+90)
mean_prediction = prediction.predicted_mean

confidence_intervals = prediction.conf_int()
lower_limits = confidence_intervals.loc[:, 'lower Revenue']
upper_limits = confidence_intervals.loc[:, 'upper Revenue']

#display tial of mean predition
mean_prediction.tail(30)
```

```
#display training, testing, and forecast

plt.figure(figsize=(10,5))
plt.plot(X_train, label='Training Data', color='b')
plt.plot(X_test, label='Testing Data', color='g')
plt.plot(mean_prediction, label='Forecast', color='r')

plt.fill_between(lower_limits.index, lower_limits, upper_limits, color='pink')
plt.title('Forecast for 90 days (3 months)')
plt.xlabel('Date')
plt.ylabel('Revenue (in million USD)')
plt.xticks(rotation=30, fontsize=10)

plt.legend(loc='upper left')
plt.show()
```

Evaluation Metrics:

```
from sklearn.metrics import r2_score
```

```
#generate and display r2_score function results

df['predicted_revenue'] = prediction
print('R^2 for original data: ', r2_score(df['Revenue'], df['predicted_revenue']))

X_train['predicted_revenue'] = prediction
print('R^2 for training data: ', r2_score(X_train['Revenue'], X_train['predicted_revenue']))

X_test['predicted_revenue'] = prediction
print('R^2 for testing data: ', r2_score(X_test['Revenue'], X_test['predicted_revenue']))
```

```
R^2 for original data:  0.9825082975864294
R^2 for training data:  0.9807146728568572
R^2 for testing data:  0.9414089387470044
```

```
from sklearn.metrics import mean_squared_error
```

```
mse = mean_squared_error(df['Revenue'], prediction[:len(df)])
mse_train = mean_squared_error(X_train['Revenue'], prediction[:len(X_train)])
mse_test = mean_squared_error(X_test['Revenue'], prediction[len(X_train):])
print("Mean Squared Error (MSE) on data: ", mse)
print("Mean Squared Error (MSE) on training set:", mse_train)
print("Mean Squared Error (MSE) on testing set:", mse_test)
```

```
Mean Squared Error (MSE) on data:  0.259271963534594
Mean Squared Error (MSE) on training set: 0.2527672605208619
Mean Squared Error (MSE) on testing set: 0.3038956250696593
```

D5

The code used to support the implementation of the time series model is included with the submission of this performance assessment.

E1

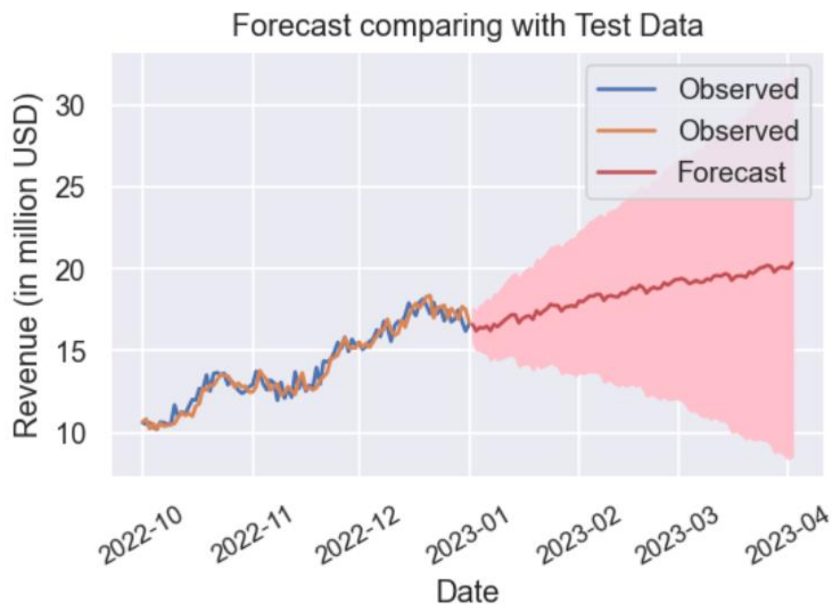
The results of the data analysis are discussed here.

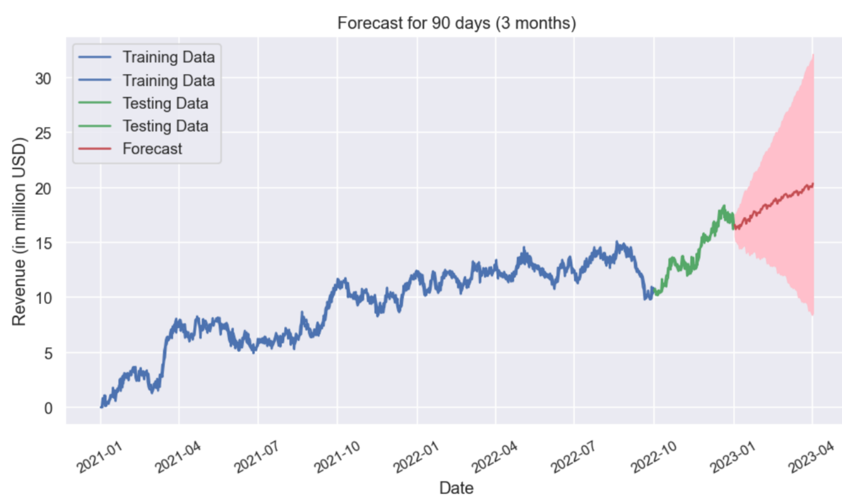
- The selection of the ARIMA model was determined by generating the ACF and PACF. Then various order combinations for AR and MA were ran to find the lowest AIC score. The parameters from Auto Arima (1, 1, 0) generated the lowest AIC for the data, training set, and test sets. The data required one order of differencing to make the data stationary.
- The prediction interval of the forecast is 1 day. The time series data is 731 one days (2 years) of daily revenue. Therefore, ARIMA model identifies correlations and seasonality to predict revenue at a day interval (Kamara, n.d.).
- A justification for the forecast length of 90 days is that the seasonality indicated that there is a periodicity to the data every 90 days, or every three months, or every quarter. So, the forecast predicts the revenue for the next quarter for the telecommunications company. Because the time series contains 2-year daily revenue, we can forecast at most to 1 year. Predictions earlier than one year will be more accurate (Kamara, n.d.).
- The model evaluation procedure is fitting the parameters on ARIMA that generate the lowest AIC score. Auto-arima was used to find the suitable seasonal order (Kamara, n.d.). The error metrics used in this analysis are R^2 and MSE. The model can explain 98% of the variance in the data set and training set and 94% of the variance of the test set. The MSE is low for the data, training, and testing sets 0.25-0.30. These metrics indicate that the model is a good fit.

E2

An annotated visualization of the forecast of the final model compared to the test set is displayed here.

```python
#display training, testing, forecast, and confidence intervals

plt.plot(X_test.index, X_test, label='Observed')
plt.plot(mean_prediction.index, mean_prediction, color='r', label='Forecast')
plt.fill_between(lower_limits.index, lower_limits, upper_limits, color='pink')
plt.title("Forecast comparing with Test Data")
plt.xlabel('Date')
plt.ylabel('Revenue (in million USD)')
plt.xticks(rotation=30, fontsize=10)
plt.legend()
plt.show()
```






```
from statsmodels.tsa.arima.model import ARIMA
```

```
diff_forecast = results.get_forecast(steps=30)
mean_forecast = diff_forecast.predicted_mean
confidence_intervals = diff_forecast.conf_int()

lower_limits = confidence_intervals.loc[:, 'lower Revenue']
upper_limits = confidence_intervals.loc[:, 'upper Revenue']

prediction = results.get_prediction(start=len(df), end=len(df)+90)
mean_prediction = prediction.predicted_mean

confidence_intervals = prediction.conf_int()
lower_limits = confidence_intervals.loc[:, 'lower Revenue']
upper_limits = confidence_intervals.loc[:, 'upper Revenue']

#display tial of mean predition
mean_prediction.tail(30)
```

```
#display training, testing, and forecast

plt.figure(figsize=(10,5))
plt.plot(X_train, label='Training Data', color='b')
plt.plot(X_test, label='Testing Data', color='g')
plt.plot(mean_prediction, label='Forecast', color='r')

plt.fill_between(lower_limits.index, lower_limits, upper_limits, color='pink')
plt.title('Forecast for 90 days (3 months)')
plt.xlabel('Date')
plt.ylabel('Revenue (in million USD)')
plt.xticks(rotation=30, fontsize=10)

plt.legend(loc='upper left')
plt.show()
```

E3

A recommended course of action is described here. The time series has a good performance accuracy to forecast the next 90 days of future revenue. The company can use this forecast to plan for network capacity and expansion for its customers (Kamara, n.d.). The revenue is projected to trend upward. The company needs to decide what to do with this additional revenue. It could be directed to company investments or to their employees or invest in a more quality product.

F

JupyterLab is the interactive development environment used to run this analysis. An HTML document of the executed notebook presentation is included with the submission of this performance assessment.

G

Kamara, Kesselly. "D213 Task 1 Cohort Webinar PPT". D213 Western Governors University, n.d., wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=8d32afc8-d66f-42d5-9117-b100016cf9ac.

Reider, Rob. "Time Series Analysis in Python". Datacamp, n.d., app.datacamp.com/learn/courses/time-series-analysis-in-python.

H

Kamara, Kesselly. "D213 Task 1 Cohort Webinar PPT". D213 Western Governors University, n.d., wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=8d32afc8-d66f-42d5-9117-b100016cf9ac

Reider, Rob. "Time Series Analysis in Python". Datacamp, n.d., app.datacamp.com/learn/courses/time-series-analysis-in-python.

I

Professional communication is demonstrated throughout the content and presentation of this PA.