D213 Task 1 Time Series Modeling

Western Governs University

Table of Contents

PART I: RESEARCH QUESTION **3**

A1. Research Question 3
A2. Objectives and Goals 3

PART II: METHOD JUSTIFICATION **3**

B. Summary of Assumptions 3

PART III: DATA PREPARATION **3**

C1. Line Graph Visualization 3
C2. Time Step Formatting 4
C3. Stationarity 4
C4. Steps to prepare the Data 5
C5. Prepared Dataset 6

PART IV: MODEL IDENTIFICATION AND ANALYSIS **6**

D1. Report findings and visualizations 6
D2. Arima Model 10
D3. Forecasting using Arima Model 12
D4. Output and Calculations 13
D5. Code 16

PART V: DATA SUMMARY AND IMPLICATIONS **16**

E1. Results 16
E2. Annotated Visualization 17
E3. Recommendations 17

PART VI: REPORTING **18**

F. Reporting 18
G. Sources for Third Party Code 18
H. Sources 18

Part I: Research Question

A1. Research Question

Can we accurately forecast the daily revenue of the company for the next quarter by considering increased network capacity expansions for customers?

A2. Objectives and Goals

The objective of this analysis is to build a predictive model that will forecast future daily revenue of the telecommunication company. This analysis will provide insights and identify trends, seasonality, and other factors to make informed decisions regarding marketing and business planning.

Part II: Method Justification

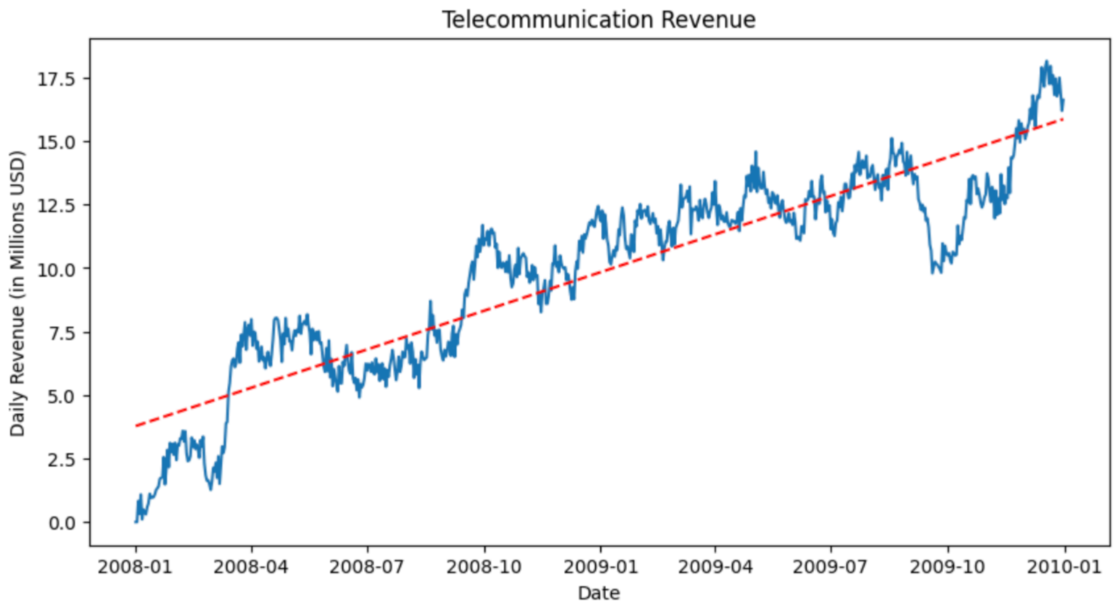
B. Summary of Assumptions

There are assumptions that can be made when completing a time series model. One assumption of a time series modeling is that the time series data should exhibit stationarity. Stationarity implies that the statistical properties of the series do not change over time. Stationarity time series is one with constant mean, constant variance, and consistent autocorrelation structure. Time series modeling also assumes that errors or residual in time series analysis are uncorrelated. This assumes that the model captures all relevant information and any remaining variation due to randomness. There also should be no outliers in the series.

Part III: Data Preparation

C1. Line Graph Visualization

```python
# Set the figure size
plt.figure(figsize=[10, 5])

# Prettify the graph
plt.title("Telecommunication Revenue")
plt.xlabel("Date")
plt.ylabel("Daily Revenue (in Millions USD)")

# Plot time series data
plt.plot(dataset.index, dataset['Revenue'])

# Generate trend line
x = np.arange(len(dataset.index))
y = dataset['Revenue'].values
z = np.polyfit(x, y, 1)
p = np.poly1d(z)

# Generate x-axis dates
x_dates = dataset.index.date

# Plot trend line
plt.plot(x_dates, p(x), "r--")

plt.show()
```



C2. Time Step Formatting

The time series included two data columns 'Day' and 'Revenue' and had 731 rows. The 'Day' values were unique and ranged from 1 to 731. There were no gaps in the measurements. There were no missing values in the 'Revenue' data. The 'Day' column was formatted into DateTime object and labeled 'Date' to allow for the facilitation of potential aggregations or other manipulation of the data.

```
start_date = pd.to_datetime('2008-01-01')
dataset['Date'] = start_date + pd.to_timedelta(dataset['Day'] - 1, unit='D')

# Drop the 'Day' column
dataset.drop('Day', axis=1, inplace=True)

# With datetime column established, set this as index
dataset.set_index('Date', inplace=True)

# Visually inspect final dataframe to verify appearance as expected
print(dataset.head())
print(dataset.tail())
```

C3. Stationarity

The given data set was non stationary. This was evident in the positively trending data graphed on the above graph. The red dotted line is a trend line shows an upward trend. Thus, 'revenue' was increasing over the given period of time. A Dicky Fuller test was used to test frame to return a test static of -1.924612 and a p-value of 0.320573. A p-value of greater than 0.05 confirms that data is non-stationary.

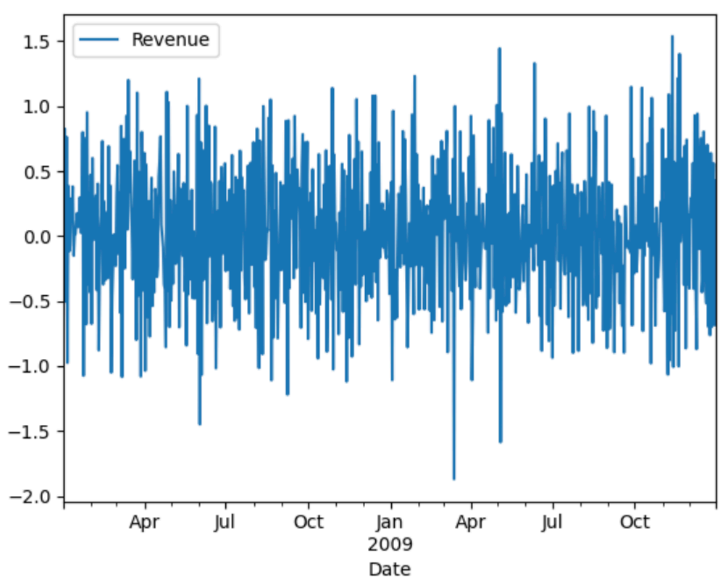
```
print ('Results of Dickey-Fuller test: ')

dftest = adfuller(dataset['Revenue'], autolag='AIC')
dfoutput = pd.Series(dftest[0:4], index=['Test Statistic','p-value','#Lags Used','No. of Observations'])

for key,value in dftest[4].items():
    dfoutput['Critical Value (%s) '%key] = value # Critical Values should always be more than the test statistic

print(dfoutput)
```

```
Results of Dickey-Fuller test:
Test Statistic                -1.924612
p-value                        0.320573
#Lags Used                     1.000000
No. of Observations          729.000000
Critical Value (1%)           -3.439352
Critical Value (5%)           -2.865513
Critical Value (10%)          -2.568886
dtype: float64
```

C4. Steps to prepare the Data

There were several steps preformed to prepare the data for time series analysis. The data must be stationary to complete a ARMIA analysis. As discussed, above the 'Day' column was converted to DatetTime object type. The fixed 'Day' ('Date') column was then turned into the index of the time series, without trend or seasonality.

After the Dickey-Fuller test the different of the data set was taken and all null values were dropped. The Augmented Dickey-Fuller was then performed resulted in a test statistic of -44.8745 and a p-value of 0.0. This made the data set stationary as evident in the graph below.



The data was then split into training and testing data sets. The data is split into two sets to confirm predictions made in the ARMIA model. The training data set makes up to first 80% of the cleaned data and the testing data set makes up the remaining 20% of the cleaned data.

```
from sklearn.model_selection import train_test_split

train, test = train_test_split(dataset_trans, test_size=0.2, shuffle=False, random_state=369)
```

```
train
```

Date	Revenue
2008-01-02	0.000793
2008-01-03	0.824749
2008-01-04	-0.505210
2008-01-05	0.762222
2008-01-06	-0.974900
...	...
2009-08-03	0.113264
2009-08-04	-0.531705
2009-08-05	-0.437835
2009-08-06	0.422243
2009-08-07	0.179940

584 rows × 1 columns

```
test
```

Date	Revenue
2009-08-08	-0.531923
2009-08-09	0.157387
2009-08-10	-0.644689
2009-08-11	0.995057
2009-08-12	-0.438775
...	...
2009-12-27	0.170280
2009-12-28	0.559108
2009-12-29	-0.687028
2009-12-30	-0.608824
2009-12-31	0.425985

146 rows × 1 columns

C5. Prepared Dataset

The cleaned data sets were split into a training and testing data set and are attached.

Save to new files

```
train.to_csv('D213_task1_train_clean.csv')
test.to_csv('D213_task1_test_clean.csv')
```

Part IV: Model Identification and Analysis

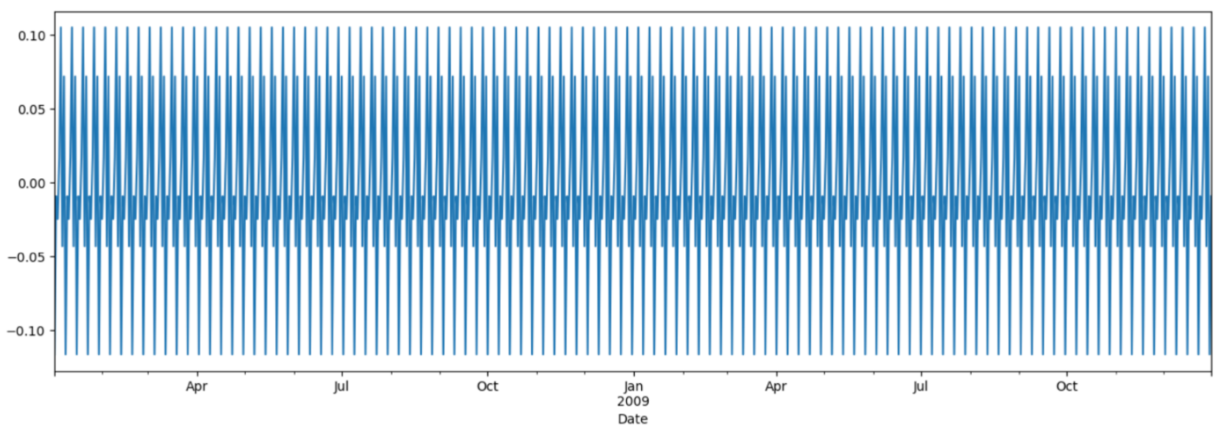
D1. Report findings and visualizations

Annotated findings with visualizations of the data analysis:

1. *Presence or lack of Seasonal Component:*

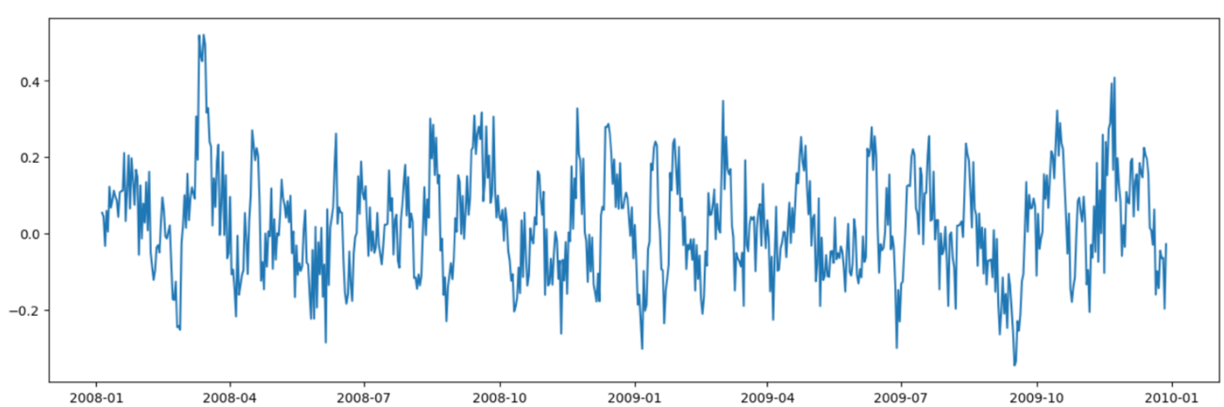
Seasonal decomposition with additive model was then performed on the data. As shown in the graph below the seasonal component has a constant magnitude so seasonal decomposition with an additive model was used to help identify and separate the seasonal, trend, and residual components of a time series (Franco, 2022).

```
# Drop any rows with null values
dataset.dropna(inplace=True)

plt.figure(figsize=[16, 5])

# Perform seasonal decomposition with additive model
decomp = seasonal_decompose(dataset_trans)

# Plot the seasonal component
decomp.seasonal.plot()
```



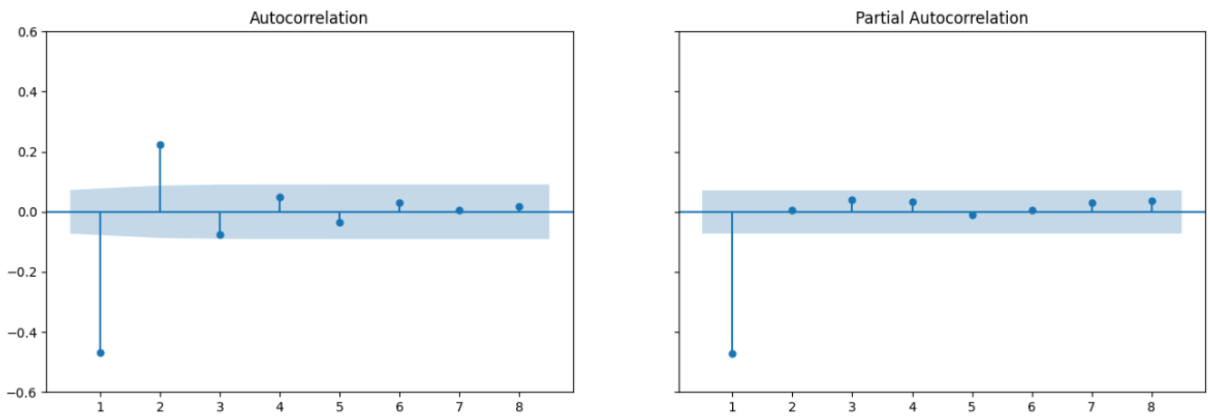
2. *Trends:*

The data was then plotted to verify seasonality and to see if tends in the data remained. As shown in the graph below there are no apparent trends.



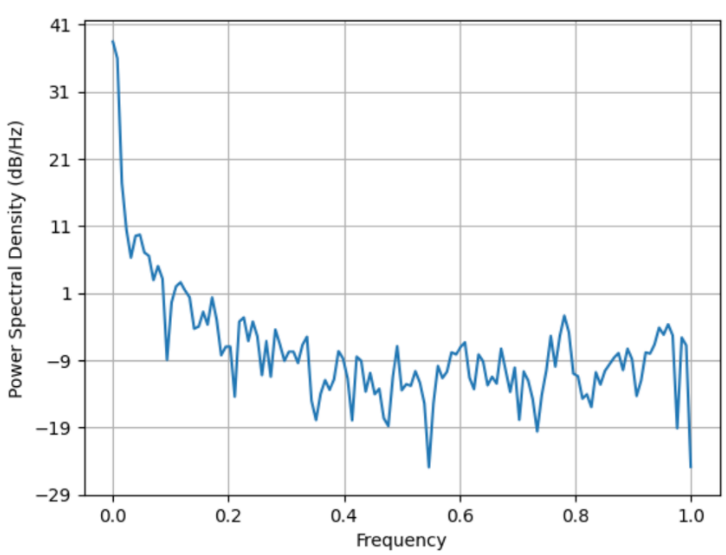
3. Auto Correlation Function:

The autocorrelation function (ACF) and partial autocorrelation function (PACF) are both statistical tools used in time series analysis to identify the underlying pattern and relationships in the data. These can be used to determine an autoregression or moving average model. The ACF and PACF were graphed as shown below. Autoregression (AR) model was chosen based on visualizations of the graphs. In the ACF plot the autocorrelation values decline gradually. In the PACF plot abruptly dropped off indicating a need for AR model.



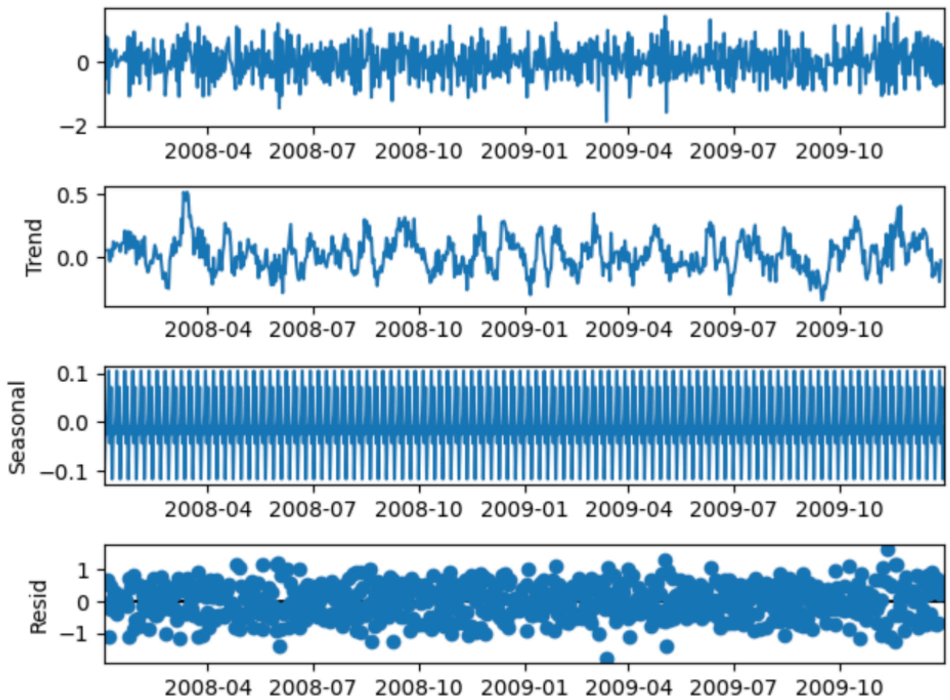
4. Spectral Density

Power spectral density is the power distribution of a signal across different frequencies. The power spectral density provides information about the strength of various frequency components present in the signal (*Siemens DISW*, n.d.).



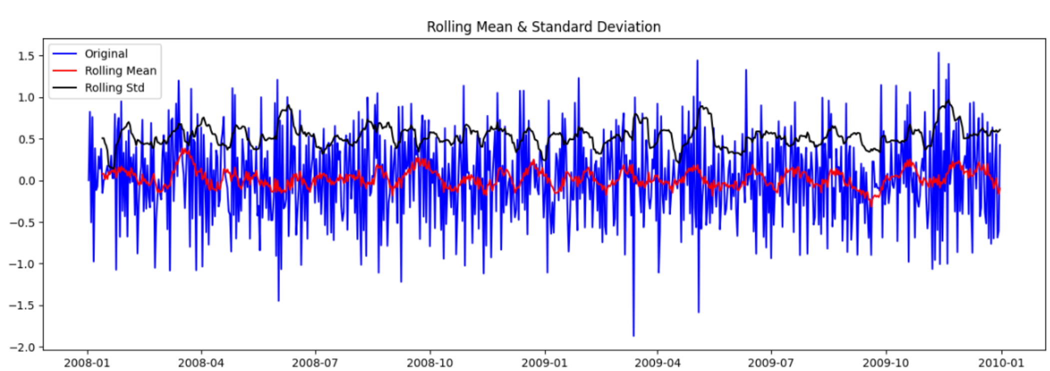
5. Decomposed Time Series

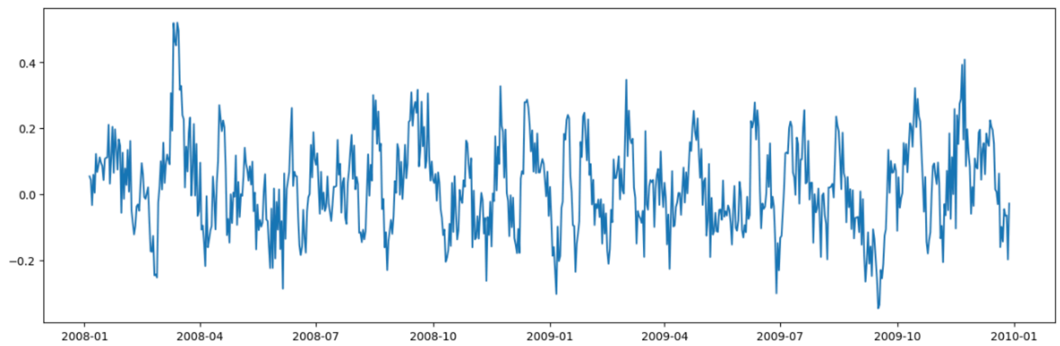
A decomposing time series includes the separation of the time series into different components which are trend, seasonality, and residual components. The additive model was used and is assumed to be the sum of its individual components (*Forecasting: Principles and Practice (2nd Ed)*, n.d.).



6. Confirmation of lack of trends in the residuals of the decompose series:

The rolling mean and standard deviation served as a visual confirmation of the lack of trends in the residuals of decomposition. The residuals of the decomposition were also plotted and show no apparent trend.





D2. Arima Model

The ACF and PACF plots could be used to indicate the best suit for the AR model. A manual approach to fitting the ARIMA model to the data was taken. The model provides the ARMIA model specification of (1,0,0)(0,0,0)[0]. This first set of parentheses (1,0,0) indicate the p, d, q which are the non-seasonal component of the model. The second set indicates the seasonal component. The third set indicate there is no seasonal period which confirms there is no seasonality in the data. The ARMIA model was then fitted to the data using the fixed order (1,0,0).

```
Performing stepwise search to minimize aic
 ARIMA(2,0,2)(0,0,0)[0] intercept   : AIC=987.305, Time=0.36 sec
 ARIMA(0,0,0)(0,0,0)[0] intercept   : AIC=1162.819, Time=0.04 sec
 ARIMA(1,0,0)(0,0,0)[0] intercept   : AIC=983.122, Time=0.05 sec
 ARIMA(0,0,1)(0,0,0)[0] intercept   : AIC=1019.369, Time=0.06 sec
 ARIMA(0,0,0)(0,0,0)[0]             : AIC=1162.139, Time=0.03 sec
 ARIMA(2,0,0)(0,0,0)[0] intercept   : AIC=985.104, Time=0.06 sec
 ARIMA(1,0,1)(0,0,0)[0] intercept   : AIC=985.106, Time=0.04 sec
 ARIMA(2,0,1)(0,0,0)[0] intercept   : AIC=986.045, Time=0.26 sec
 ARIMA(1,0,0)(0,0,0)[0]             : AIC=984.710, Time=0.02 sec

Best model:  ARIMA(1,0,0)(0,0,0)[0] intercept
Total fit time: 0.928 seconds
```

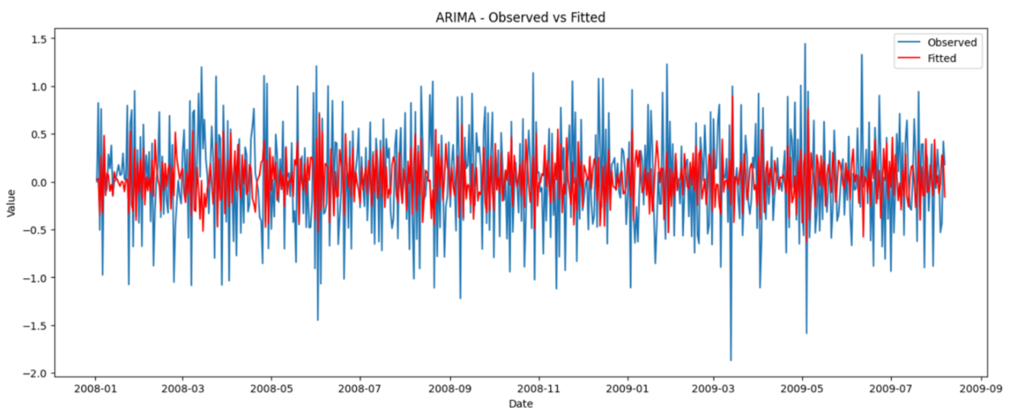
SARIMAX Results

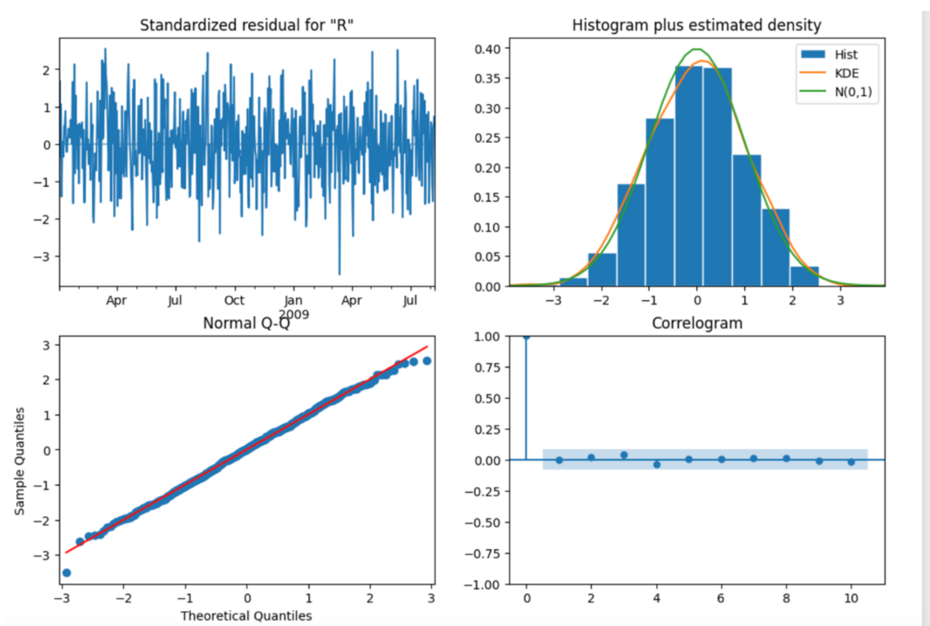
Dep. Variable:	y	**No. Observations:**	730
Model:	SARIMAX(1, 0, 0)	**Log Likelihood**	-488.561
Date:	Tue, 30 May 2023	**AIC**	983.122
Time:	17:46:41	**BIC**	996.901
Sample:	01-02-2008	**HQIC**	988.438
	- 12-31-2009		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
intercept	0.0332	0.018	1.895	0.058	-0.001	0.068
ar.L1	-0.4692	0.033	-14.296	0.000	-0.534	-0.405
sigma2	0.2232	0.013	17.801	0.000	0.199	0.248

Ljung-Box (L1) (Q):	0.00	**Jarque-Bera (JB):**	2.05
Prob(Q):	0.96	**Prob(JB):**	0.36
Heteroskedasticity (H):	1.02	**Skew:**	-0.02
Prob(H) (two-sided):	0.85	**Kurtosis:**	2.74

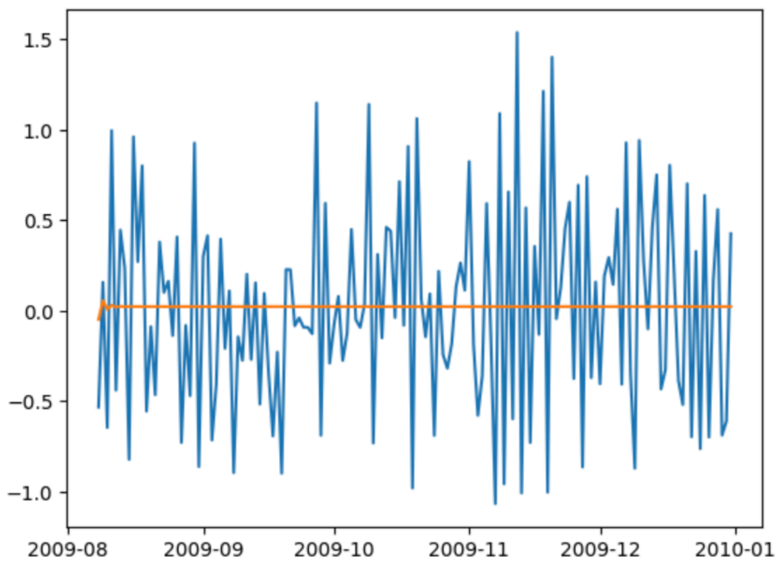
```
                               SARIMAX Results
==============================================================================
Dep. Variable:                Revenue   No. Observations:                  584
Model:                 ARIMA(1, 0, 0)   Log Likelihood                -383.946
Date:                Tue, 30 May 2023   AIC                            773.893
Time:                        17:46:44   BIC                            787.002
Sample:                    01-02-2008   HQIC                           779.002
                         - 08-07-2009
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0234      0.013      1.758      0.079      -0.003       0.049
ar.L1         -0.4597      0.036    -12.654      0.000      -0.531      -0.388
sigma2         0.2180      0.014     16.034      0.000       0.191       0.245
===================================================================================
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):                 1.84
Prob(Q):                              0.96   Prob(JB):                         0.40
Heteroskedasticity (H):               0.97   Skew:                            -0.08
Prob(H) (two-sided):                  0.83   Kurtosis:                         2.77
===================================================================================

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

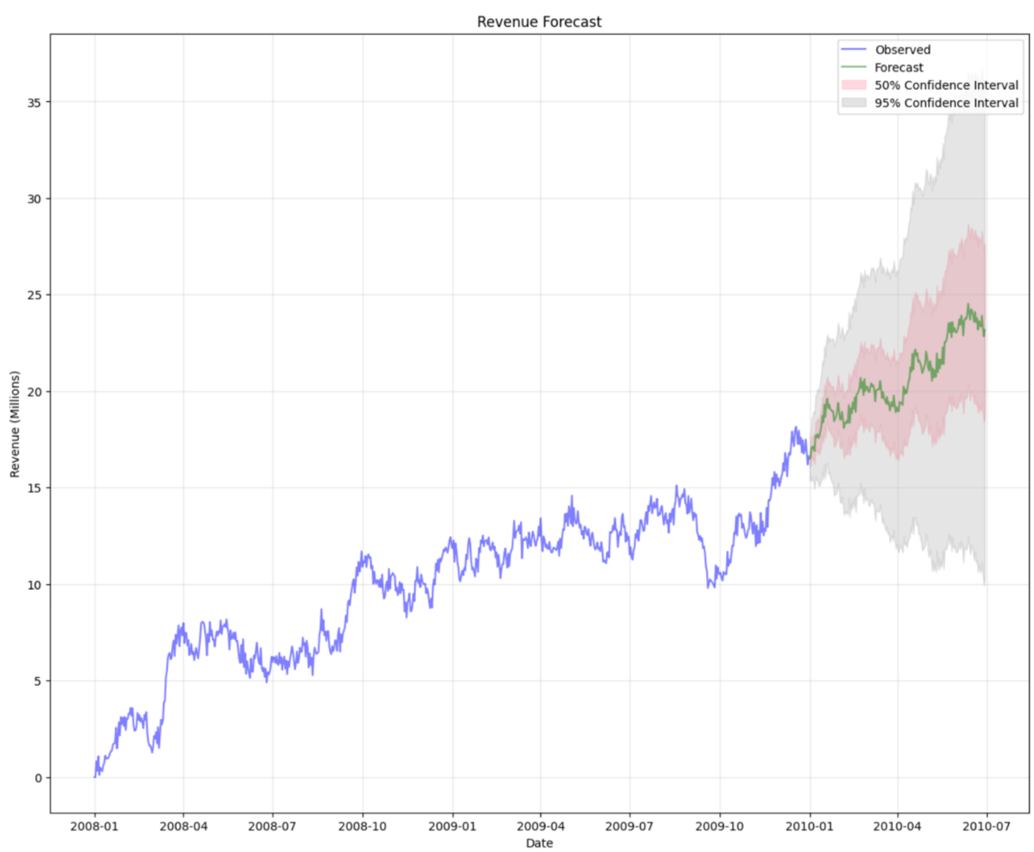




D3. Forecasting using Arima Model

The ARMIA model can be used to predict future values in the time series model. Below is the graph of the forecasted values.





D4. Output and Calculations

1. Code to make/validate stationarity:

Decomposing Data:

```
# Drop any rows with null values
dataset.dropna(inplace=True)

plt.figure(figsize=[16, 5])

# Perform seasonal decomposition with additive model
decomp = seasonal_decompose(dataset_trans)

# Plot the seasonal component
decomp.seasonal.plot()
```

Rolling Mean and Standard Deviations:

```
rolmean = dataset_trans.rolling(window=12).mean() # Monthly level
rolstd = dataset_trans.rolling(window=12).std()
print (rolmean, rolstd)


plt.figure(figsize=[16, 5])
orig = plt.plot(dataset_trans, color = 'blue', label='Original')
mean = plt.plot(rolmean, color ='red', label='Rolling Mean')
std = plt.plot(rolstd, color='black',label='Rolling Std')
plt.legend(loc='best')
plt.title('Rolling Mean & Standard Deviation')
plt.show(block=False)
```

2. Code for autocorrelation plots (ACF & PACF):

ACF & PACF plots:

```
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

# Plot Autocorrelation and Partial Autocorrelation in one figure, sharing a y axis
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=[16,5], sharey=True)

# Plot ACF to 8 lags (only 7 days in a week), ignore zero (zero always = 1)
plot_acf(dataset_trans, lags=8, zero=False, ax=ax1)

# Plot PACF to 8 lags (only 7 days in a week), ignore zero (zero always = 1)
plot_pacf(dataset_trans, lags=8, zero=False, ax=ax2)
```

```
# Zoom in on y axis to see points more clearly
plt.ylim(-0.6, 0.6)

# Display the plot
plt.show()
```

3. Code for using SARIMAX:

Fitting model:

```
stepwise_fit=auto_arima(dataset_trans['Revenue'], trace=True,
suppress_warnings=True)
stepwise_fit.summary()
```

Arima code:

```
from statsmodels.tsa.arima.model import ARIMA
model = ARIMA(train, order=(1, 0, 0), freq='D')
results = model.fit()
print(results.summary())
```

Plot of Observed vs Fitted ARIMA:

```
# Plot the observed values and fitted values
plt.figure(figsize=(16, 6))
plt.plot(train, label='Observed')
plt.plot(results.fittedvalues, color='red', label='Fitted')
plt.title('ARIMA - Observed vs Fitted')
plt.xlabel('Date')
plt.ylabel('Value')
plt.legend()
plt.show()
```

Diagnostic Plot for ARIMA:

```
results.plot_diagnostics(figsize=(12, 8))
```

4. Model Summary:

```
                               SARIMAX Results
==============================================================================
Dep. Variable:                Revenue   No. Observations:                  584
Model:                 ARIMA(1, 0, 0)   Log Likelihood                -383.946
Date:                Tue, 30 May 2023   AIC                            773.893
Time:                        17:46:44   BIC                            787.002
Sample:                    01-02-2008   HQIC                           779.002
                         - 08-07-2009
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0234      0.013      1.758      0.079      -0.003       0.049
ar.L1         -0.4597      0.036    -12.654      0.000      -0.531      -0.388
sigma2         0.2180      0.014     16.034      0.000       0.191       0.245
===================================================================================
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):                 1.84
Prob(Q):                              0.96   Prob(JB):                         0.40
Heteroskedasticity (H):               0.97   Skew:                            -0.08
Prob(H) (two-sided):                  0.83   Kurtosis:                         2.77
===================================================================================

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

5. Predictions on out of model data (future dates)

Using the SARIMAX to make predictions:

```
model = SARIMAX(dataset, order=(1,1,0),seasonal_order=(1,1,0,90), trend='c',
        error_action='ignore', suppress_warnings=True)
results = model.fit()
```

Generate forecasts using ARIMA model:

```
#The prediction of this model, set to 180 days
forecast = results.get_forecast(steps=180)

#The predicted mean
mean_forecast = forecast.predicted_mean

#Establishing the lower and upper confidence limts for a %50 confidence interval
confidence_intervals_50 = forecast.conf_int(alpha=.5)
lower_50 = confidence_intervals_50.iloc[:,0]
upper_50 = confidence_intervals_50.iloc[:,1]

#Establishing the lower and upper confidence limts for a %95confidence interval
confidence_intervals_95 = forecast.conf_int(alpha=.05)
lower_95 = confidence_intervals_95.iloc[:,0]
upper_95 = confidence_intervals_95.iloc[:,1]
```

Generate forecast for future revenue values based on ARIMA model:

```
# Place the forecasted differences into a temporary dataframe
forecast_temp = pd.DataFrame(forecasted.predicted_mean)
# Make consistent names for dataframe for concatenation
forecast_temp.rename(columns={'predicted_mean' : 'revenue'}, inplace=True)
# Concat a copy of Train (thru Aug 07 2009) and a copy of forecasted values (forward from Aug 08 2009)
df_w_forecast = pd.concat([train.copy(), forecast_temp.copy()])
# We've generated one DF with the differences in daily revenue for the entire 2-year period, invert the differences using cumsum
df_w_forecast = df_w_forecast.cumsum()
# Check output to verify expected values
df_w_forecast
```

D5. Code

Code used to implement the model has been provided.

Part V: Data Summary and Implications

E1. Results

1. Select of an ARIMA model:

To identify a suitable ARIMA model the ACF and PACF were first generated. The ACF and PACF plots could be used to indicate the best suit for the AR model. A manual approach to fitting the ARIMA model to the data was taken. The model provides the ARMIA model specification of (1,0,0)(0,0,0)[0]. This first set of parentheses (1,0,0) indicate the p, d, q which are the non-seasonal component of the model. The second set indicates the seasonal component. The third set indicate there is no seasonal period which confirms there is no seasonality in the data. The ARMIA model was then fitted to the data using the fixed order (1,0,0).

2. Prediction interval of the forecast:

The training data for the ARIMA model was built with daily revenue values. Prediction intervals for future values are in one day intervals. The data is a 2-year daily revenue. Therefor the ARIMA model identifies the correlations and seasonality to predict revenue at a one day interval.

3. Justification of the forecast length:

The model was suitable for predicting up to one year of values. As the future values moved beyond the date of known data the confidence interval of the ARIMA model's predictions became wider. Thus, there was larger uncertainty as dates move further in the future. So, 180 days into the future was a sufficient forecast length. Long term predictions will require more historical data.

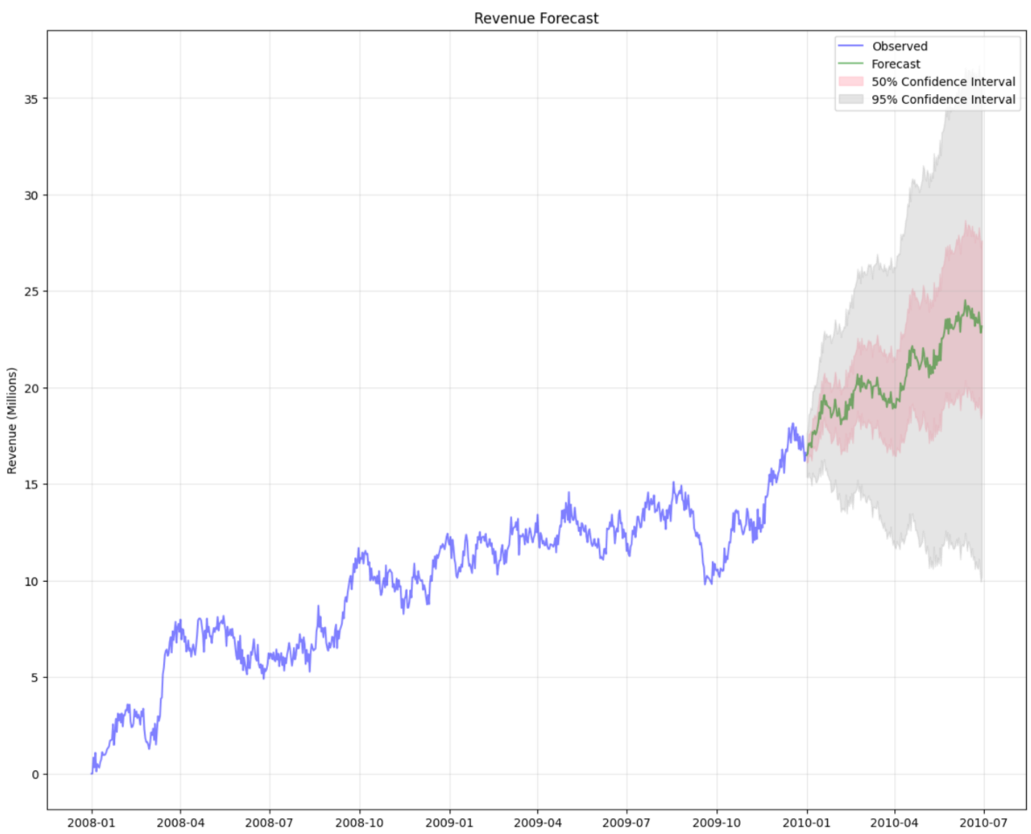
4. Model Evaluation Procedure and Error Metric:

The model was fitted with a p,d,q of 1,0,0. Manual ARIMA was used to find suitable seasonal order. The final model provided an AIC of 771. The model demonstrates statistical significance based on the coefficient 'ar.L1' and sigma had p-value of less than 0.05.

Error Metrics:

The mean absolute error (MAE) of this prediction model is: 15.19275885014206
The mean square error (MSE) of this prediction model is: 232.14181047352
The root mean square error (RMSE) of this prediction model is: 15.23620065743163

E2. Annotated Visualization



E3. Recommendations

The time series has a good performance accuracy to forecast 90 days future revenue. Shorter time forecasts are more accurate to predict terms longer than 2-year data due to decrease in accuracy with longer predictions. Using the model, the company can forecast revenue projections for the next 2 quatres (180 days). The telecommunications company can be used to forecast a plan for network capacity expansion for its customers.

Part VI: Reporting

F. Reporting

Jupyter notebook was used for developing the time series model. Attached is the HTML of the python code for the time series.

G. Sources for Third Party Code

Converting day count to date time. (n.d.). Stack Overflow. https://stackoverflow.com/questions/61389654/converting-day-count-to-date-time

H. Sources

6.1 Time series components | Forecasting: Principles and Practice (2nd ed). (n.d.). https://otexts.com/fpp2/components.html

"ARIMA Models in Python" Datacamp. https://plus.google.com/u/0/+Datacamp/. (n.d.). *Sign in*. DataCamp. https://app.datacamp.com/learn/courses/arima-models-in-python

Franco, D. (2022, October 3). A Visual Guide to Time Series Decomposition Analysis. *Encora*. https://www.encora.com/insights/a-visual-guide-to-time-series-decomposition-analysis

Siemens DISW. (n.d.). https://community.sw.siemens.com/s/article/what-is-a-power-spectral-density-psd