# D213 Advanced Data Analytics - ARIMA – Performance Assessment

Steven Schindler

Advanced Data Analytics – D213

Sewell, Williams; PhD

College of I.T., Western Governors University

June 21, 2023

**Part I:  Research Question**

## A1.

Using the telecommunications data file, we can use forecasting techniques to ask: What is the revenue forecast for the upcoming quarter?

## A2.

The goal of this analysis is to use time series to forecast revenue for the upcoming first quarter(2022) for the company by analyzing the company’s revenue over a two-year period(2020-2021).

# **Part II:  Method Justification**

## B.

An assumption of the time series analysis is that the data should be stationary. This means that the mean and variance are constant over a period of time and the series has zero trend as it is not growing or shrinking. Stationarity also means that the autocorrelation is constant; how each value in the time series is related to its neighbor stays the same.( Hyndman, R.J., & Athanasopoulos, G. (2018). ) Autocorrelation is a mathematical representation between a given time series and a lagged version of itself.(InfluxData. (n.d.).)

# **Part III:  Data Preparation**

## C1.

Below is a line graph representing the revenue in millions from 2020 to 2022 two years later.

## 

## C2.

*dafr = pd.read\_csv('teleco\_time\_series.csv')*

*dafr.dropna()*

*dafr = dafr[dafr['Revenue']> 0]*

*dafr['Date'] = pd.date\_range(start = datetime(2020,1,1),periods = dafr.shape[0],freq='24H')*

*dafr = dafr.set\_index(['Date'])*

*dafr = dafr.drop(['Day'],axis = 1)*

*dafr.shape*

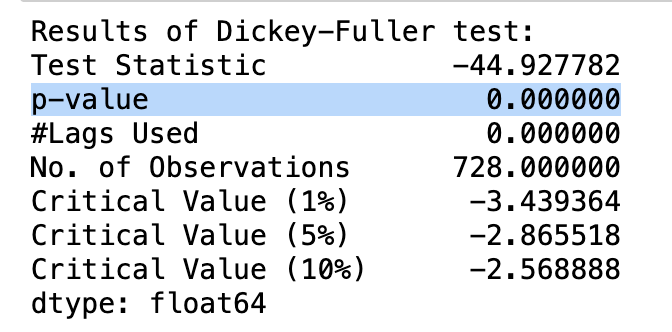
The above code checks for gaps in the time series by removing null values after the series is read into a data frame using the dropna() function. It also removes anywhere the revenue is 0 or less than 0. Then the data ‘Day’ is converted to a datetime format and then set as the index. After the removal of zero revenue the length of the sequence is 730 days.

## C3.

Running the Dickey-Fuller test on the original data set we see that the p-value is greater than .05 at 0.39. This says that the original data is not stationary.

A screenshot of a computer

Description automatically generated with medium confidence

Using differrencing on the data set and then running the ADfuller test again we now have a p-value of 0.000 this being less than .05 the data is now stationary.

## C4.

The first step was to read the data into a pandas data frame called df. Then the Day column was converted into a Date column. Null values were than dropped and stationarity was tested using the Adfuller test. Stationarity was coerced using differencing and finally the Data was split into training and test sets. The training set uses 80% and the testing set uses 20%. The code to split the sets is listed below:

*Xtrain = df\_stationarity.iloc[:-145]*

*Xtest = df\_stationarity.iloc[-145:]*

*print("X\_train shape: ",Xtrain.shape)*

*print("X\_test shape: ",Xtest.shape)*

## C5.

The cleaned dataset is called cleaned\_D213.csv and the training and testing sets are called X\_train.csv and X\_test.csv respectively.

# **Part IV:  Model Identification and Analysis**

## D1.

A picture containing text, diagram, plot, line

Description automatically generatedA picture containing text, plot, screenshot, line

Description automatically generatedBelow are the visuals for seasonality, trends, acf and pacf.

## A picture containing text, screenshot, line, diagram Description automatically generatedA picture containing text, screenshot, line, plot Description automatically generated

A picture containing text, screenshot, plot, diagram

Description automatically generatedA picture containing text, screenshot, plot, line

Description automatically generatedNext are the visualizations for spectral density, the residuals and the decomposed time series.

## A picture containing text, screenshot, font, line Description automatically generated

## D2.

A screenshot of a computer

Description automatically generated with low confidence Using auto\_arima we get the best model with the lowest aic is (5,1,0)(0,0,0)[0].

## D3.

A picture containing plot, text, line, diagram

Description automatically generated Below is a graph of the revenue with the forecasted model using the derived ARIMA model:

Next is a picture of just the forecasted first quarter for 2022:

## A picture containing text, screenshot, line, diagram Description automatically generated

## D4.

The following is my code to make and validate stationarity, autocorrelation plots and the sarimax model:

**Stationarity:**

*from statsmodels.tsa.stattools import adfuller*

*print ('Dickey-Fuller test: ')*

*dfte = adfuller(dafr['Revenue'], autolag='AIC')*

*dfout = pd.Series(dfte[0:4], index=['Test Statistic','pvalue','#Lags','No.Observations'])*

*for key,value in dfte[4].items():*

*dfout['Critical Value (%s) '%key] = value # Critical Values should always be more than the test statistic*

*print(dfout)*

*df\_stationarity = dafr.diff().dropna()*

*from statsmodels.tsa.stattools import adfuller*

*print ('Dickey-Fuller test: ')*

*dfte = adfuller(dafr['Revenue'], autolag='AIC')*

*dfout = pd.Series(dfte[0:4], index=['Test Statistic','pvalue','#Lags','No.Observations'])*

*for key,value in dfte[4].items():*

*dfout['Critical Value (%s) '%key] = value # Critical Values should always be more than the test statistic*

*print(dfout)*

**Autocorrelation plots:**

*from statsmodels.tsa.stattools import acf, pacf*

*from statsmodels.graphics.tsaplots import plot\_acf,plot\_pacf*

*plot\_acf(df\_stationarity)*

*plot\_pacf(df\_stationarity)*

**Sarimax:**

*from statsmodels.tsa.statespace.sarimax import SARIMAX*

*warnings.filterwarnings("ignore")*

*models = SARIMAX(df\_stationarity,order = (5,1,0),seasonal\_order = (0,0,0,0),error\_action="ignore",supress\_warnings = 'true')*

*results = models.fit()*

*print(results.summary())*

The model summary for the SARIMAX:

## A screenshot of a computer Description automatically generated with low confidence

The mean forecast predictions for the first quarter of 2022:

## A picture containing text, font, screenshot, white Description automatically generated

## D5.

Code is listed below and in the provided Juypter notebook file.

***import numpy as np***

***import pandas as pd***

***import matplotlib.pylab as plt***

***import warnings***

***warnings.filterwarnings("ignore")***

***from datetime import datetime***

***from statsmodels.tsa.seasonal import seasonal\_decompose***

***from statsmodels.tsa.statespace.sarimax import SARIMAX***

***dafr = pd.read\_csv('teleco\_time\_series.csv')***

***dafr.dropna()***

***dafr = dafr[dafr['Revenue']> 0]***

***dafr['Date'] = pd.date\_range(start = datetime(2020,1,1),periods = dafr.shape[0],freq='24H')***

***dafr = dafr.set\_index(['Date'])***

***dafr = dafr.drop(['Day'],axis = 1)***

***dafr.shape***

***plt.xlabel('Date')***

***plt.ylabel('Revenue ''(in millions)''')***

***plt.plot(dafr)***

***from statsmodels.tsa.stattools import adfuller***

***print ('Dickey-Fuller test: ')***

***dfte = adfuller(dafr['Revenue'], autolag='AIC')***

***dfout = pd.Series(dfte[0:4], index=['Test Statistic','pvalue','#Lags','No.Observations'])***

***for key,value in dfte[4].items():***

***dfout['Critical Value (%s) '%key] = value # Critical Values should always be more than the test statistic***

***print(dfout)***

***df\_stationarity = dafr.diff().dropna()***

***from statsmodels.tsa.stattools import adfuller***

***print ('Dickey-Fuller test: ')***

***dfte = adfuller(dafr['Revenue'], autolag='AIC')***

***dfout = pd.Series(dfte[0:4], index=['Test Statistic','pvalue','#Lags','No.Observations'])***

***for key,value in dfte[4].items():***

***dfout['Critical Value (%s) '%key] = value # Critical Values should always be more than the test statistic***

***print(dfout)***

***Xtrain = df\_stationarity.iloc[:-145]***

***Xtest = df\_stationarity.iloc[-145:]***

***print("X\_train shape: ",Xtrain.shape)***

***print("X\_test shape: ",Xtest.shape)***

***df\_stationarity.to\_csv("cleaned\_D213.csv")***

***Xtrain.to\_csv('X\_train.csv')***

***Xtest.to\_csv('X\_test.csv')***

***decom = seasonal\_decompose(dafr['Revenue'], period=90)***

***plt.title('Seasonality')***

***decom.seasonal.plot()***

***plt.title('Trend')***

***decom.trend.plot()***

***from statsmodels.tsa.stattools import acf, pacf***

***from statsmodels.graphics.tsaplots import plot\_acf,plot\_pacf***

***plot\_acf(df\_stationarity)***

***plot\_pacf(df\_stationarity)***

***from scipy import signal***

***f, Pxx = signal.periodogram(df\_stationarity['Revenue'])***

***plt.semilogy(f, Pxx)***

***plt.ylim([1e-6, 1e2])***

***plt.title('Spectral Density')***

***plt.xlabel('Frequency')***

***plt.ylabel('Spectral Density')***

***plt.show()***

***decom.plot() # Plot decomposition***

***plt.show() # Check for seasonality in the data***

***plt.title('Residuals')***

***decom.resid.plot()***

***from pmdarima.arima import auto\_arima***

***stepwise\_fit = auto\_arima(df\_stationarity,start\_p=0,start\_q=0,d=1,seasonal=True,trace=True,suppress\_warnings=True)***

***stepwise\_fit.summary()***

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

***warnings.filterwarnings("ignore")***

***model = ARIMA(df\_stationarity['Revenue'], order=(5,1,0))***

***results\_ARIMA = model.fit()***

***model\_fit = model.fit()***

***print(model\_fit.summary())***

***from statsmodels.tsa.statespace.sarimax import SARIMAX***

***warnings.filterwarnings("ignore")***

***models = SARIMAX(df\_stationarity,order = (5,1,0),seasonal\_order = (0,0,0,0),error\_action="ignore",supress\_warnings = 'true')***

***results = models.fit()***

***print(results.summary())***

***pred = results.get\_prediction(start = -90)***

***mean\_pred = pred.predicted\_mean***

***confidence = pred.conf\_int()***

***lower = confidence.loc[:,'lower Revenue']***

***upper = confidence.loc[:,'upper Revenue']***

***print(mean\_pred)***

***plt.figure(figsize=(12,4))***

***plt.plot(Xtest.index, Xtest,label='test set')***

***plt.plot(mean\_pred.index, mean\_pred,color='r',label='forecast')***

***plt.fill\_between(lower.index,lower,upper,color='pink')***

***plt.title('Forcast compared with test data')***

***plt.xlabel('Date')***

***plt.ylabel('Revenue in millions')***

***plt.legend()***

***plt.show()***

***forecast = results.get\_forecast(steps=145)***

***meanforecast = forecast.predicted\_mean***

***confidence = forecast.conf\_int()***

***lower = confidence.loc[:,'lower Revenue']***

***upper = confidence.loc[:,'upper Revenue']***

***print(meanforecast)***

***plt.figure(figsize=(12,4))***

***plt.plot(df\_stationarity.index, df\_stationarity,label='observed')***

***plt.plot(meanforecast.index, meanforecast,color='r',label='forecast')***

***plt.fill\_between(lower.index,lower,upper,color='pink')***

***plt.title('Revenue with forcasted projections for first quarter in 2022')***

***plt.xlabel('Date')***

***plt.ylabel('Revenue in millions')***

***plt.xticks(rotation=45)***

***plt.legend()***

***plt.show()***

***plt.title('Revenue with forcasted projections in first quarter 2022')***

***plt.xlabel('Date')***

***plt.ylabel('Revenue in millions')***

***meanforecast.plot()***

***from sklearn.metrics import mean\_absolute\_error***

***mean\_absolute\_error(Xtest.values, meanforecast.values)***

# **Part V:  Data Summary and Implications**

## E1.

The selection of the Arima model was chosen using auto Arima to find the lowest AIC. This outputted a best Arima model of ARIMA(5,1,0)(0,0,0)[0]. The code to get this best model is shown below:

*from pmdarima.arima import auto\_arima*

*stepwise\_fit = auto\_arima(df\_stationarity,start\_p=0,start\_q=0,d=1,seasonal=True,trace=True,suppress\_warnings=True)*

*stepwise\_fit.summary()*

The prediction interval of the forecast is 1 as our data is two years at a daily interval. Because of this the ARIMA model identifies seasonality and correlations of daily revenue.

The model can only predict up to one year as there is only two years of data however, since we are only predicting for a quarter the forecast length is 3 months.

Auto ARIMA was used as the model evaluation procedure to find the lowest AIC. The mean square error and mean absolute error were both calculated at .36 and .48 respectively.

## E2.

Below is an annotated visualization of the final model compared to the test set:

## A picture containing text, plot, font, line Description automatically generated

## E3.

A picture containing text, screenshot, line, diagram

Description automatically generated

Based on the above visualizations it looks like the revenue is going to be steadily negative around Jan 17th therefore, I recommend for the company to investigate ways of reducing cost and increase efforts for retention.

## References:

InfluxData. (n.d.). Autocorrelation in Time Series Data. Retrieved from <https://www.influxdata.com/blog/autocorrelation-in-time-series-data/>

Hyndman, R.J., & Athanasopoulos, G. (2018). Stationarity. In Forecasting: Principles and Practice (2nd Edition). OTexts. Retrieved from <https://otexts.com/fpp2/stationarity.html>