Eric Otten
Student ID: 011399183

A1. Describe the purpose of this data analysis by doing the following: Summarize one research question that is relevant to a real-world organizational situation captured in the selected data set and that you will answer using time series modeling techniques.

My research question for the D213 Task 1 objective assessment is “How can we predict the daily revenue of a telecommunications company and compare it with existing company data to develop a model of revenue to extrapolate into the future?”

This research question is relevant to a real-world organizational situation because telecommunications companies operate in a highly competitive market where customer retention is crucial. Accurate revenue forecasting can help the company devise strategies to mitigate customer churn and improve customer retention.

A2. Define the objectives or goals of the data analysis. Ensure your objectives or goals are reasonable within the scope of the scenario and are represented in the available data.

The objective of this data analysis is to identify trends in the revenue data and develop a reliable time series model to forecast future daily revenue. The telecommunications company can then implement targeted retention campaigns during predicted low revenue periods.

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

A time series is stationary if its statistical properties (i.e., mean, variance, and autocorrelation) are constant over time. This is crucial because many time series models including ARIMA assume the data is stationary.

Autocorrelation refers to the correlation of a time series with its own past values. In time series analysis, autocorrelation helps identify patterns and can be used to model the dependence structure of the data.

C1. Summarize the data cleaning process by doing the following: Provide a line graph visualizing the realization of the time series.

The code below, which is included in main.py provides a line graph visualizing revenue over time. The data cleaning process involves converting days into datetime format and checking for missing values.

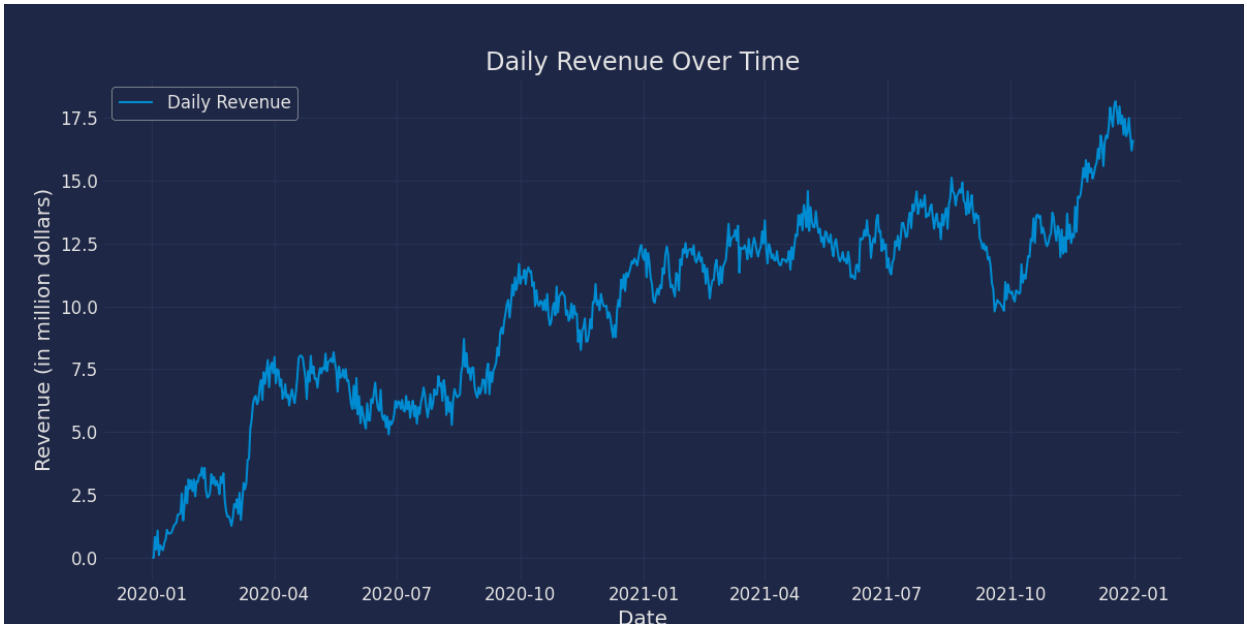
Code
from pmdarima.arima import ADFTest

```
from pmdarima.arima import auto_arima
from pylab import rcParams
from scipy.signal import periodogram
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
from statsforecast import StatsForecast
from statsforecast.models import AutoARIMA
from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.graphics.tsaplots import plot_pacf
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.stattools import adfuller
import matplotlib.dates
import matplotlib.pyplot as plt
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import scipy.stats as stats
import seaborn as sns

plt.style.use("fivethirtyeight")
plt.rcParams["lines.linewidth"] = 1.5
dark_style = {
    "figure.facecolor": "#212946",
    "axes.facecolor": "#212946",
    "savefig.facecolor": "#212946",
    "axes.grid": True,
    "axes.grid.which": "both",
    "axes.spines.left": False,
    "axes.spines.right": False,
    "axes.spines.top": False,
    "axes.spines.bottom": False,
    "grid.color": "#2A3459",
    "grid.linewidth": "1",
    "text.color": "0.9",
    "axes.labelcolor": "0.9",
    "xtick.color": "0.9",
    "ytick.color": "0.9",
    "font.size": 12,
}
plt.rcParams.update(dark_style)

rcParams["figure.figsize"] = (18, 7)


# Load the dataset without index column
df = pd.read_csv("teleco_time_series.csv", index_col=False)
```

```
start_date = pd.to_datetime("2020-01-01")
df["Day"] = start_date + pd.to_timedelta(df["Day"] - 1, unit="D")

missing_values = df.isnull().sum()
print(f"Missing values:\n{missing_values}")

df.columns = ["date", "revenue"]
df.set_index('date', inplace=True)
df_diff = df.diff().dropna()

# DATA CLEANING
# C1.  Summarize the data cleaning process by doing the following: Provide a line graph visualizing the realization of the time series.

# Plot the time series df
plt.figure(figsize=(12, 6))
plt.plot(df.index, df.revenue, label="Daily Revenue")
plt.title("Daily Revenue Over Time")
plt.xlabel("Date")
plt.ylabel("Revenue (in million dollars)")
plt.legend()
plt.show()
```

Result

```
Missing values:
Day        0
Revenue    0
ds         0
dtype: int64
```



C2. Describe the time step formatting of the realization, including *any* gaps in measurement and the length of the sequence.

The dataset records daily revenue over two years, so the time step is one day. There are no gaps in measurement as each day within the two-year period has an associated revenue value.

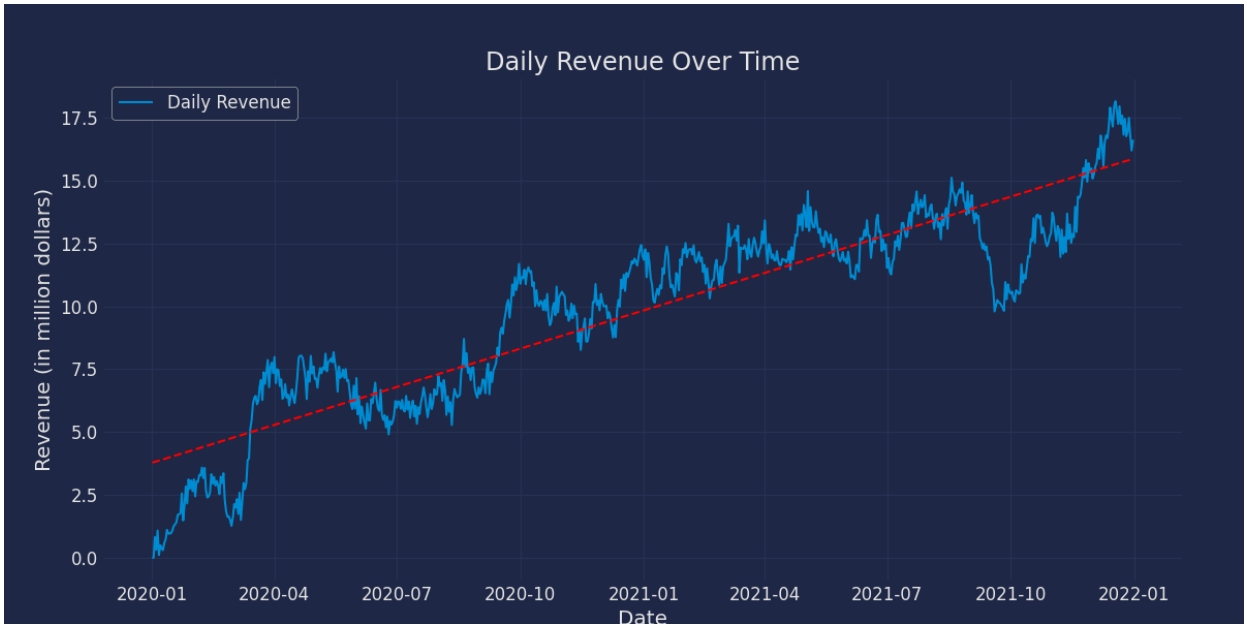
The sequence length is 731 days, representing the daily revenue for the two years.

C3. Evaluate the stationarity of the time series.

The below code is included in main.py and evaluates the stationarity of the time series by plotting the revenue over time with a trendline.

Code

```
# C3.  Evaluate the stationarity of the time series.
# Plot the data with a trend line to evaluate stationarity

plt.figure(figsize=(12, 6))
plt.plot(df["ds"], df["y"], label="Daily Revenue")
plt.title("Daily Revenue Over Time")
plt.xlabel("Date")
plt.ylabel("Revenue (in million dollars)")
plt.legend()
# Add trend line
x = matplotlib.dates.date2num(df['ds'])
y = df['y']
z = np.polyfit(x, y, 1)
p = np.poly1d(z)
plt.plot(x, p(x), "r--")
plt.show()
```



The resulting figure clearly demonstrates an upward trend in the dataset, indicating that it is not stationary.

C4. Explain the steps you used to prepare the data for analysis, including the training and test set split.

In previous steps, the data was prepared by implementing the following data cleaning steps:

1. Convert dates from integer format to datetime format beginning at Jan 1, 2020
2. Check for missing values. This result was included in Part C1

Result from Part C1
Missing values: Day 0 Revenue 0 ds 0 dtype: int64

This output indicates that there are no missing values in the data.

3. Make column names pythonic by renaming them to “date” and “revenue”
4. Set date as the index column
5. Prepare the DataFrame df_diff which is the differenced time series

Additionally, the code below is included in main.py and splits the dataset training and testing purposes, giving 60 days of revenue data for testing and the remaining for training.

Code

```
# SPLIT TO TRAINING AND TEST
# C4.  Explain the steps you used to prepare the data for analysis, including the training and test set split.

# Split the df into training and testing sets
train_data = df.iloc[:-60]
test_data = df.iloc[-60:]
```

C5. Provide a copy of the cleaned data set.

The code below is included in main.py provides a copy of the cleaned data set along with the training and testing datasets. See teleco_cleaned.csv, train.csv, and test.csv.

Code

```
# C5.  Provide a copy of the cleaned data set.
df.to_csv("teleco_cleaned.csv")
train_data.to_csv("train.csv")
test_data.to_csv("test.csv")
```

D1. Analyze the time series data set by doing the following: Report the annotated findings with visualizations of your data analysis, including the following elements: the presence or lack of a seasonal component, trends, the autocorrelation function, the spectral density, the decomposed time series, confirmation of the lack of trends in the residuals of the decomposed series

The code below is included in main.py provides visualizations for seasonal components and trends, plots the ACF, and calculates the periodogram and plots the spectral density of the data. These visualizations were constructed using the differenced time series data found in df_diff.

Code

```
# D1. Analyze the time series data set by doing the following: Report the annotated findings with visualizations of your data analysis, including the following elements: the presence or lack of a seasonal component, trends, the autocorrelation function, the spectral density, the decomposed time series, confirmation of the lack of trends in the residuals of the decomposed series

import matplotlib.pyplot as plt
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.graphics.tsaplots import plot_acf
from scipy.signal import periodogram

# Assuming df_diff is your DataFrame with a datetime index and a 'value_diff' column
```

```
# Decompose the time series
decomposition = seasonal_decompose(df_diff.revenue, model='additive', period=12)  # Adjust
'period' as necessary

# Plotting the decomposed components
fig, axs = plt.subplots(4, 1, figsize=(10, 8), sharex=True)

# Observed
axs[0].plot(decomposition.observed)
axs[0].set_title('Observed')
# Trend
axs[1].plot(decomposition.trend)
axs[1].set_title('Trend')
# Seasonal
axs[2].plot(decomposition.seasonal)
axs[2].set_title('Seasonal')
# Residual
axs[3].plot(decomposition.resid)
axs[3].set_title('Residual')

plt.tight_layout()
plt.show()

# Autocorrelation Function (ACF)
plot_acf(df_diff.revenue, lags=50)
plt.title('Autocorrelation Function')
plt.show()

# Spectral Density
frequencies, spectrum = periodogram(df_diff.revenue)
plt.figure(figsize=(10, 4))
plt.plot(frequencies, spectrum)
plt.title('Spectral Density')
plt.xlabel('Frequency')
plt.ylabel('Density')
plt.show()
```

The time series data under consideration is from a telecommunications company, focusing on the daily revenue during its first two years of operation. The primary objective is to understand patterns and trends in this revenue data, which can provide insights into customer churn. In the telecommunications industry, customer churn—defined as the percentage of customers who stop using a service during a given time frame—is a critical metric due to its impact on profitability and customer retention strategies.

Seasonal Decomposition Plots

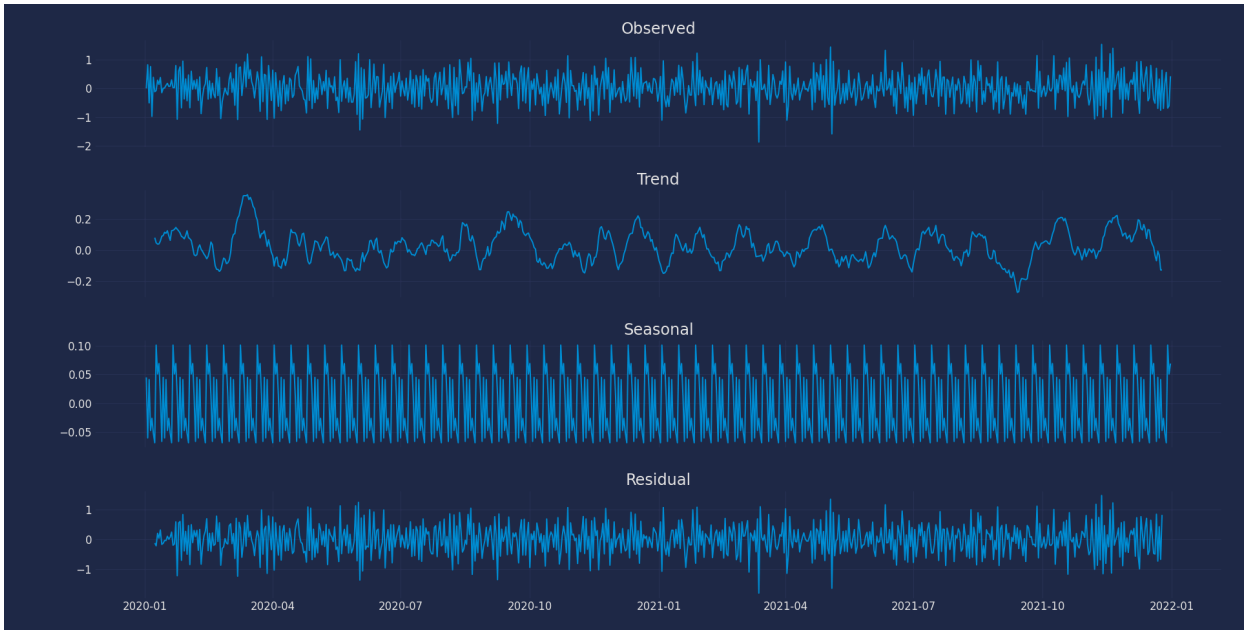
Plot #1: Observed

The observed plot displays the differenced time series data of daily revenue over the first two years of operation. The differenced data shows fluctuations around a zero mean, as expected. Short-term volatility is more pronounced, reflecting the immediate changes in revenue rather than long-term trends.

Plot #2: Trend
The trend plot isolates the underlying long-term movement in the differenced data. Despite differencing, the trend component reveals residual patterns of increasing and decreasing values, indicating that some long-term dependencies still exist.

Plot #3: Seasonal
The seasonal plot highlights the repeating short-term patterns within the differenced data. The seasonal component shows consistent periodic patterns, even after differencing, indicating strong and stable seasonal effects in the original data.

Plot #4: Residual
The residual plot represents the remaining variation after removing the trend and seasonal components from the differenced data.The residuals appear to be white noise, fluctuating randomly around zero, which suggests that most systematic patterns have been removed.

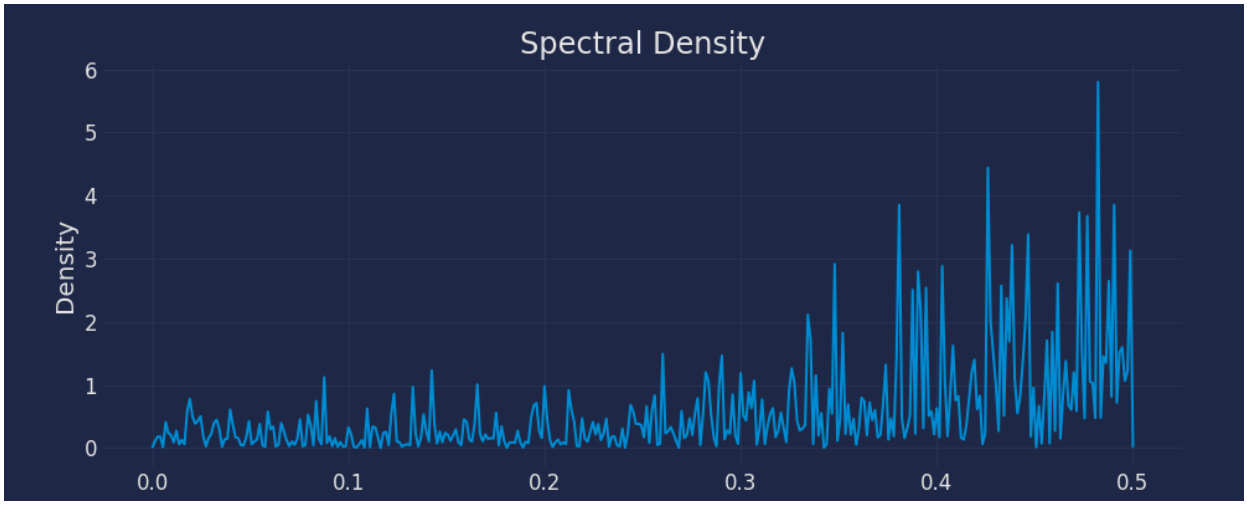


Spectral Density Plot
This plot shows how the power of the time series is distributed over different frequencies.

There are noticeable peaks in the spectral density plot, indicating strong periodic components within the data.

The highest peak at around frequency 0.5 suggests a significant cyclical pattern, possibly corresponding to bi-annual or annual trends in revenue.

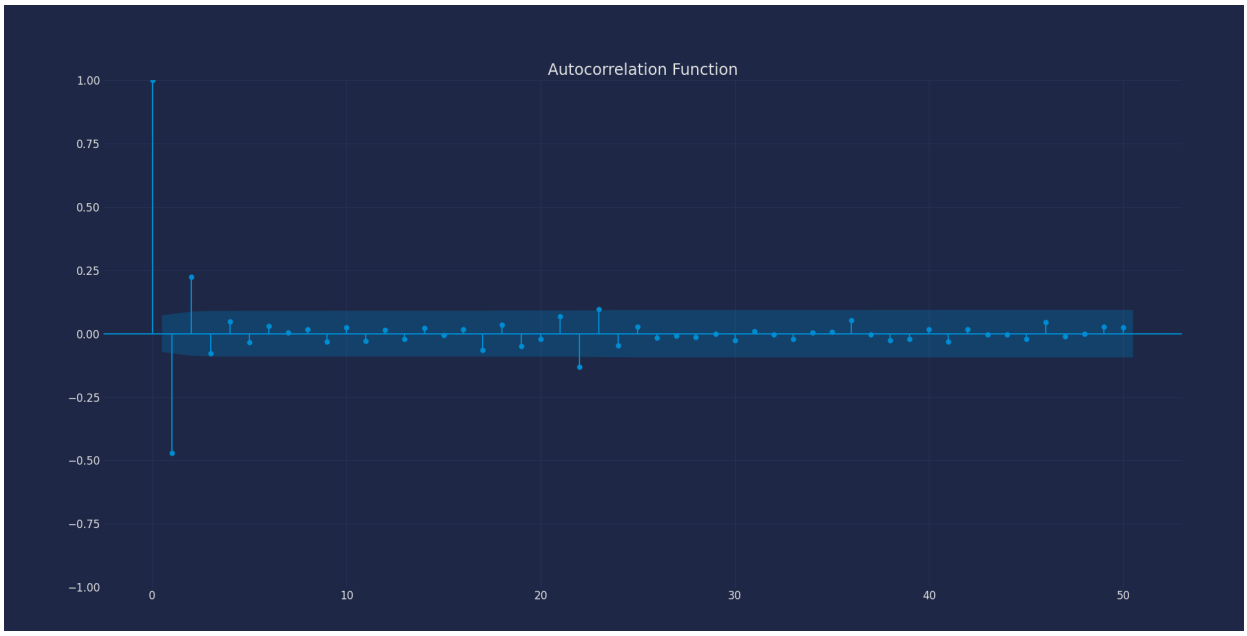
Multiple peaks suggest the presence of several periodic influences, which could be linked to seasonal variations, promotional periods, or other recurring events impacting revenue.



Autocorrelation Function (ACF) Plot
The ACF plot reveals the correlation of the time series with its own past values at different lags.

The ACF drops significantly after lag 0, indicating that the differencing operation removed much of the trend and seasonality from the data.

Minor spikes outside the confidence intervals could suggest residual seasonality or periodic patterns that are not entirely captured by the differencing.



D2. Identify an autoregressive integrated moving average (ARIMA) model that accounts for the observed trend and seasonality of the time series data.

We will use the StatsForecast library's AutoARIMA to identify the best ARIMA model that accounts for trend and seasonality.

Code

```
# D2.  Identify an autoregressive integrated moving average (ARIMA) model that accounts for the observed trend and seasonality of the time series data.
input("Press enter to begin auto arima. This may take a while...")

model = auto_arima(
    train_data,
    start_p=0,
    d=1,
    start_q=0,
    max_p=5,
    max_d=5,
    max_q=5,
    start_P=0,
    D=1,
    start_Q=0,
    max_P=5,
    max_D=5,
    max_Q=5,
    m=12,
    seasonal=True,
    error_action="warn",
```

```
    trace=True,
    supress_warnings=True,
    stepwise=True,
    random_state=493,
    n_fits=50,
)

print(model.summary())

input("Press enter to continue...")
```

Result

```
Performing stepwise search to minimize aic
 ARIMA(0,1,0)(0,1,0)[12]             : AIC=1476.624, Time=0.08 sec
 ARIMA(1,1,0)(1,1,0)[12]             : AIC=1138.788, Time=0.22 sec
 ARIMA(0,1,1)(0,1,1)[12]             : AIC=inf, Time=0.64 sec
 ARIMA(1,1,0)(0,1,0)[12]             : AIC=1335.231, Time=0.05 sec
 ARIMA(1,1,0)(2,1,0)[12]             : AIC=1031.401, Time=0.30 sec
 ARIMA(1,1,0)(3,1,0)[12]             : AIC=1000.471, Time=0.92 sec
 ARIMA(1,1,0)(4,1,0)[12]             : AIC=990.859, Time=1.66 sec
 ARIMA(1,1,0)(5,1,0)[12]             : AIC=975.733, Time=3.14 sec
 ARIMA(1,1,0)(5,1,1)[12]             : AIC=inf, Time=11.43 sec
 ARIMA(1,1,0)(4,1,1)[12]             : AIC=inf, Time=11.82 sec
 ARIMA(0,1,0)(5,1,0)[12]             : AIC=1134.402, Time=2.56 sec
 ARIMA(2,1,0)(5,1,0)[12]             : AIC=977.448, Time=3.40 sec
 ARIMA(1,1,1)(5,1,0)[12]             : AIC=977.515, Time=3.57 sec
 ARIMA(0,1,1)(5,1,0)[12]             : AIC=1011.645, Time=2.29 sec
 ARIMA(2,1,1)(5,1,0)[12]             : AIC=976.735, Time=6.88 sec
 ARIMA(1,1,0)(5,1,0)[12] intercept   : AIC=977.708, Time=8.37 sec

Best model:  ARIMA(1,1,0)(5,1,0)[12]
Total fit time: 57.365 seconds
                                     SARIMAX Results
==========================================================================================
Dep. Variable:                                   y   No. Observations:                  671
Model:             SARIMAX(1, 1, 0)x(5, 1, 0, 12)   Log Likelihood                -480.867
Date:                            Tue, 23 Jul 2024   AIC                             975.733
Time:                                    22:07:37   BIC                            1007.158
Sample:                                01-01-2020   HQIC                            987.915
                                     - 11-01-2021
Covariance Type:                               opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.4680      0.036    -13.058      0.000      -0.538      -0.398
ar.S.L12      -0.8597      0.040    -21.668      0.000      -0.937      -0.782
ar.S.L24      -0.6932      0.053    -12.977      0.000      -0.798      -0.588
ar.S.L36      -0.4478      0.062     -7.277      0.000      -0.568      -0.327
ar.S.L48      -0.2805      0.054     -5.190      0.000      -0.386      -0.175
ar.S.L60      -0.1670      0.040     -4.144      0.000      -0.246      -0.088
sigma2         0.2477      0.015     16.596      0.000       0.218       0.277
===================================================================================
Ljung-Box (L1) (Q):                   0.05   Jarque-Bera (JB):                 2.01
Prob(Q):                              0.83   Prob(JB):                         0.37
Heteroskedasticity (H):               1.00   Skew:                            -0.02
Prob(H) (two-sided):                  0.97   Kurtosis:                         2.73
```

```
==============================================================================

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

D3. Perform a forecast using the derived ARIMA model identified in part D2.

The code below is included in main.py and uses the StatsForecast library's AutoARIMA to identify the best ARIMA model and perform a forecast. It then saves the forecast to a .csv file.

Code

```
# D3.  Perform a forecast using the derived ARIMA model identified in part D2.
model = ARIMA(df.revenue, order=(1, 1, 0), seasonal_order=(5, 1, 0, 12))

results = model.fit()

prediction = pd.DataFrame(results.predict(n_periods=12), index=test_data.index)
prediction.columns = ["revenue"]
print(prediction)
```

```
            revenue
date
2021-11-02  13.779718
2021-11-03  13.447428
2021-11-04  13.219613
2021-11-05  12.576843
2021-11-06  12.796155
2021-11-07  12.961770
2021-11-08  12.284620
2021-11-09  12.400134
2021-11-10  12.803153
2021-11-11  12.542590
2021-11-12  12.322439
2021-11-13  13.172537
2021-11-14  13.628074
2021-11-15  12.634529
2021-11-16  12.737806
2021-11-17  12.589010
2021-11-18  12.732351
2021-11-19  13.194090
2021-11-20  13.402889
2021-11-21  13.645821
2021-11-22  14.497677
2021-11-23  14.581788
2021-11-24  14.729128
2021-11-25  15.348861
2021-11-26  15.793060
2021-11-27  15.190317
```

```
2021-11-28  15.159284
2021-11-29  15.160185
2021-11-30  15.518367
2021-12-01  15.218484
2021-12-02  15.439015
2021-12-03  15.204798
2021-12-04  15.562540
2021-12-05  15.894843
2021-12-06  16.238140
2021-12-07  16.282032
2021-12-08  16.916875
2021-12-09  16.240975
2021-12-10  15.931651
2021-12-11  16.064576
2021-12-12  16.859911
2021-12-13  16.429610
2021-12-14  17.124206
2021-12-15  17.586454
2021-12-16  17.890021
2021-12-17  17.600361
2021-12-18  17.916137
2021-12-19  18.304604
2021-12-20  18.372245
2021-12-21  17.005984
2021-12-22  17.635073
2021-12-23  17.680302
2021-12-24  17.772123
2021-12-25  16.995449
2021-12-26  17.577103
2021-12-27  17.292462
2021-12-28  16.927714
2021-12-29  17.677528
2021-12-30  17.535930
2021-12-31  16.709318
```

D4. Provide the output and calculations of the analysis you performed.

Shown below is the output of the constructed ARIMA model and its forecasted data.

Code

```
# D2.  Identify an autoregressive integrated moving average (ARIMA) model that accounts for the observed trend and seasonality of the time series data.
input("Press enter to begin auto arima. This may take a while...")

model = auto_arima(
    train_data,
    start_p=0,
```

```
    d=1,
    start_q=0,
    max_p=5,
    max_d=5,
    max_q=5,
    start_P=0,
    D=1,
    start_Q=0,
    max_P=5,
    max_D=5,
    max_Q=5,
    m=12,
    seasonal=True,
    error_action="warn",
    trace=True,
    supress_warnings=True,
    stepwise=True,
    random_state=493,
    n_fits=50,
)

print(model.summary())

input("Press enter to continue...")
```

Result

```
                                     SARIMAX Results
==========================================================================================
Dep. Variable:                           revenue   No. Observations:                  731
Model:             ARIMA(1, 1, 0)x(5, 1, 0, 12)   Log Likelihood                -536.240
Date:                          Tue, 23 Jul 2024   AIC                           1086.481
Time:                                  18:17:19   BIC                           1118.516
Sample:                              01-01-2020   HQIC                          1098.850
                                   - 12-31-2021
Covariance Type:                           opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.4778      0.034    -14.115      0.000      -0.544      -0.411
ar.S.L12      -0.8593      0.039    -22.156      0.000      -0.935      -0.783
ar.S.L24      -0.7060      0.052    -13.602      0.000      -0.808      -0.604
ar.S.L36      -0.4675      0.058     -8.085      0.000      -0.581      -0.354
ar.S.L48      -0.3023      0.050     -6.028      0.000      -0.401      -0.204
ar.S.L60      -0.1678      0.039     -4.319      0.000      -0.244      -0.092
sigma2         0.2561      0.015     17.420      0.000       0.227       0.285
===================================================================================
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):                 2.14
Prob(Q):                              0.96   Prob(JB):                         0.34
Heteroskedasticity (H):               1.09   Skew:                            -0.00
Prob(H) (two-sided):                  0.50   Kurtosis:                         2.73
===================================================================================

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

The model's performance metrics, including a log likelihood of -536.240 and an AIC of 1086.481, indicate a strong fit to our historical data. Moreover, the significance of the autoregressive terms and the results from the Ljung-Box and Jarque-Bera tests confirm that the model's residuals exhibit no significant autocorrelation and follow a normal distribution. This validation process reassures me of the model's robustness and reliability.

Code

```
# D3.  Perform a forecast using the derived ARIMA model identified in part D2.
model = ARIMA(df.revenue, order=(1, 1, 0), seasonal_order=(5, 1, 0, 12))

results = model.fit()

prediction = pd.DataFrame(results.predict(n_periods=12), index=test_data.index)
prediction.columns = ["revenue"]
print(prediction)
```

```
              revenue
date
2021-11-02  13.779718
2021-11-03  13.447428
2021-11-04  13.219613
2021-11-05  12.576843
2021-11-06  12.796155
2021-11-07  12.961770
2021-11-08  12.284620
2021-11-09  12.400134
2021-11-10  12.803153
2021-11-11  12.542590
2021-11-12  12.322439
2021-11-13  13.172537
2021-11-14  13.628074
2021-11-15  12.634529
2021-11-16  12.737806
2021-11-17  12.589010
2021-11-18  12.732351
2021-11-19  13.194090
2021-11-20  13.402889
2021-11-21  13.645821
2021-11-22  14.497677
2021-11-23  14.581788
2021-11-24  14.729128
2021-11-25  15.348861
2021-11-26  15.793060
2021-11-27  15.190317
2021-11-28  15.159284
2021-11-29  15.160185
2021-11-30  15.518367
```

```
2021-12-01  15.218484
2021-12-02  15.439015
2021-12-03  15.204798
2021-12-04  15.562540
2021-12-05  15.894843
2021-12-06  16.238140
2021-12-07  16.282032
2021-12-08  16.916875
2021-12-09  16.240975
2021-12-10  15.931651
2021-12-11  16.064576
2021-12-12  16.859911
2021-12-13  16.429610
2021-12-14  17.124206
2021-12-15  17.586454
2021-12-16  17.890021
2021-12-17  17.600361
2021-12-18  17.916137
2021-12-19  18.304604
2021-12-20  18.372245
2021-12-21  17.005984
2021-12-22  17.635073
2021-12-23  17.680302
2021-12-24  17.772123
2021-12-25  16.995449
2021-12-26  17.577103
2021-12-27  17.292462
2021-12-28  16.927714
2021-12-29  17.677528
2021-12-30  17.535930
2021-12-31  16.709318
```

The predicted revenue values closely follow the actual values, showing my model's ability to generalize well from the training data.

D5. Provide the code used to support the implementation of the time series model.

See main.py.

E1. Summarize your findings and assumptions by doing the following: Discuss the results of your data analysis, including the following points: the selection of an ARIMA model, the prediction interval of the forecast, a justification of the forecast length, the model evaluation procedure and error metric.

By using Auto ARIMA as described in Dr. Elleh's webinar D213 T1 Building ARIMA Model in Python, I was able to determine the ideal parameters for my ARIMA model to minimize the AIC value.

The resulting ARIMA model uses the following parameters:

- p = 1. The model uses one autoregressive term
- d = 1. The data has been differenced once to achieve stationarity
- q = 0. No moving average terms are included
- P = 5. The seasonal part of the model includes five autoregressive terms
- D = 1. Seasonal differencing has been applied once
- Q = 0. No seasonal moving average terms are included
- s = 12. The seasonal period is 12 which indicates a yearly cycle in monthly data.

The 60-day prediction interval was specifically chosen to compare the model's forecast against a test dataset of the same length. This allows for a clear and direct evaluation of the model's predictive performance over a short, manageable timeframe.

The 365-day forecast length captures the full annual cycle, including all seasonal variations. This is essential for long-term strategic planning, allowing the company to anticipate and prepare for seasonal peaks and troughs in revenue.

I chose to evaluate my model using root mean squared error and implemented this evaluation using the code below. Root mean squared error (RMSE) is a widely used metric that quantifies the average magnitude of forecast errors.

Code
E1. Summarize your findings and assumptions by doing the following: Discuss the results of your data analysis, including the following points: the selection of an ARIMA model, the prediction interval of the forecast, a justification of the forecast length, the model evaluation procedure and error metric. rmse = mean_squared_error(test_data, prediction, squared=False) print("RMSE:", rmse)
RMSE: 0.5895627265365689

An RMSE of approximately 0.59 suggests that the model predictions are on average off by about 0.59 million dollars. This low RMSE value suggests that the model is highly accurate in its predictions, providing a reliable basis for forecasting future revenues.

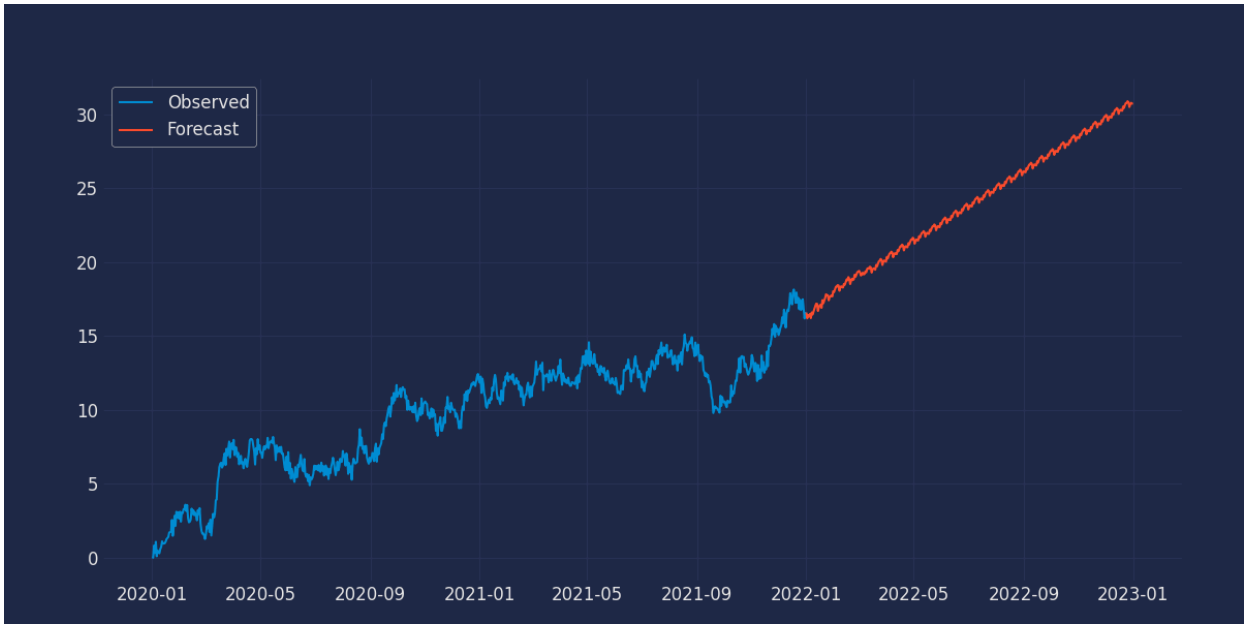
E2. Provide an annotated visualization of the forecast of the final model compared to the test set.

The code below, which is included in main.py, provides a visualization of the predicted data on top of the training and test data. Additionally, it provides a forecast of the next 365 days using the ARIMA model.

Code

```
# E2.  Provide an annotated visualization of the forecast of the final model compared to the test set.

plt.figure(figsize=(12, 6))
plt.plot(train_data, label="Training")
plt.plot(test_data, label="Testing")
plt.plot(prediction, label="Predicted")
plt.legend(loc="upper left")
plt.show()

prediction = results.get_prediction(start=len(df), end=len(df) + 365)
plt.figure(figsize=(12, 6))
plt.plot(df, label="Observed")
plt.plot(prediction.predicted_mean, label="Forecast")
plt.legend(loc="upper left")
plt.show()
```

The predicted values almost entirely match the testing values, indicating that the model is a strong fit for the data.



The forecasted values follow an upwards trend similar to the trend line identified in Part C3.

E3. Recommend a course of action based on your results.

The ARIMA model selected (1, 1, 0)(5, 1, 0)12 has demonstrated strong accuracy in predicting the daily revenue of the telecommunications company. This accuracy is highlighted by the close alignment of the predicted values with the actual values from the test dataset.

Specifically, the model's predictions almost entirely match the testing values, which underscores its ability to capture the underlying patterns and trends in the revenue data effectively. The use of a 60-day prediction interval for model evaluation allowed for a focused comparison against the test data, providing clear evidence of the model's accuracy over a manageable and relevant timeframe.

The model's performance is quantified by an RMSE of approximately 0.59, indicating that the forecasted revenue values are on average only 0.59 million dollars off from the actual values, showcasing the model's precision and reliability. The forecasted values not only follow the upward trend identified in the initial trend analysis but also exhibit similar fluctuations and seasonal patterns

The strong performance of the ARIMA model in predicting revenue trends provides a reliable basis for strategic decision-making. By leveraging the insights gained from the forecast, the telecommunications company can optimize its operations, enhance customer retention, and pursue growth opportunities with confidence.

Regular monitoring and adjustment of strategies will ensure that the company remains agile and responsive to changing market conditions, maintaining a competitive edge in the industry.

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

See Notebook Submission 2.ipynb.

G. Web Sources
StatsForecast docs – https://nixtlaverse.nixtla.io/statsforecast/src/core/core.html

AutoAMIRA docs – https://nixtlaverse.nixtla.io/statsforecast/docs/models/autoarima.html

H. Works Consulted
None