Task 3:

Time Series Modeling

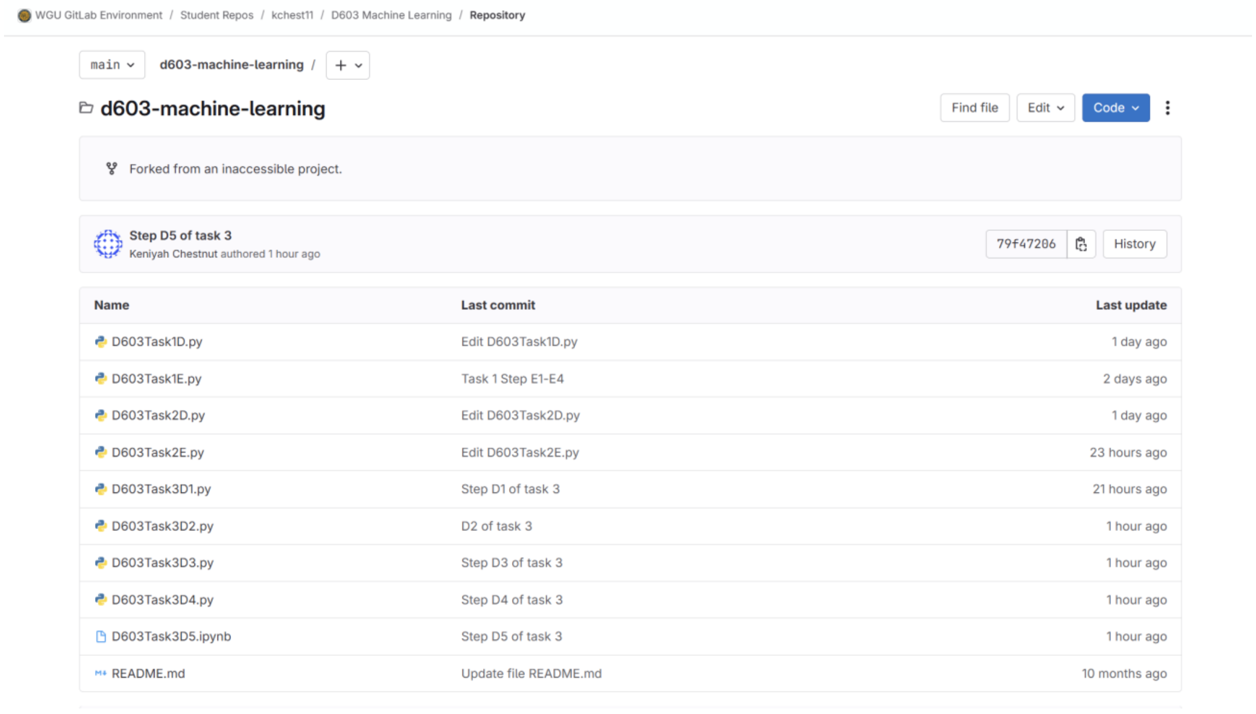
Keniyah Chestnut

Machine Learning — D603

SID: 012601305

A: GITLAB REPOSITORY

I created a subgroup and project in GitLab using the provided link. I cloned the repo to my computer, worked in it locally, and pushed my changes as I completed each part of the task.


WGU GitLab Environment / Student Repos / kchest11 / D603 Machine Learning / Repository

main / d603-machine-learning /

d603-machine-learning

Forked from an inaccessible project.

Step D5 of task 3
Keniyah Chestnut authored 1 hour ago — 79f47206

Name	Last commit	Last update
D603Task1D.py	Edit D603Task1D.py	1 day ago
D603Task1E.py	Task 1 Step E1-E4	2 days ago
D603Task2D.py	Edit D603Task2D.py	1 day ago
D603Task2E.py	Edit D603Task2E.py	23 hours ago
D603Task3D1.py	Step D1 of task 3	21 hours ago
D603Task3D2.py	D2 of task 3	1 hour ago
D603Task3D3.py	Step D3 of task 3	1 hour ago
D603Task3D4.py	Step D4 of task 3	1 hour ago
D603Task3D5.ipynb	Step D5 of task 3	1 hour ago
README.md	Update file README.md	10 months ago

B1: PROPOSAL OF QUESTION

The research question I am addressing is: How does revenue change over time, and can we forecast future revenue trends with a reasonable degree of accuracy? In this time series analysis, I aim to uncover patterns in the data that could inform future hospital revenue expectations and help stakeholders make informed financial decisions.

B2: OBJECTIVES AND GOALS

The primary objective of this analysis is to understand how revenue evolves across time and whether a statistical model can accurately project future revenue. By evaluating past revenue patterns, I will develop a predictive model that can support resource planning and reduce uncertainty in operational forecasting.

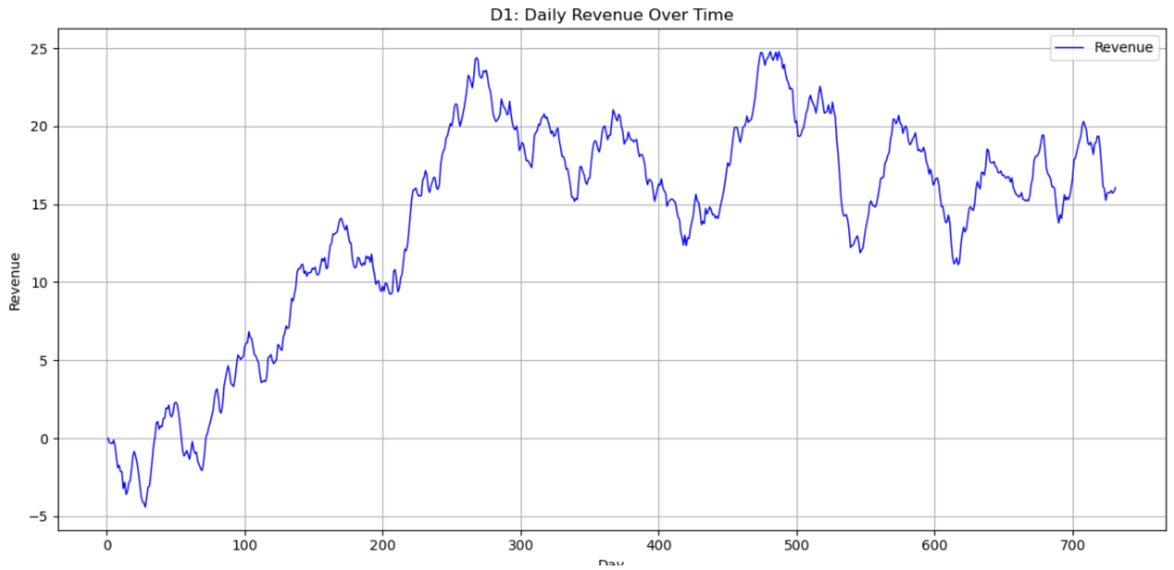
C: SUMMARY OF ASSUMPTIONS

This model is based on two key assumptions. First, we assume the data exhibits autocorrelation, which allows us to use past values to predict future ones. This is common in financial time series. Second, we assume that the time series must be made stationary to apply ARIMA modeling techniques. External factors like policy changes or regional healthcare events are not accounted for and are assumed not to significantly alter the trend of the dataset.

C2: SUMMARY OF TECHNIQUE ASSUMPTION

This model is based on two key assumptions. First, we assume the data exhibits autocorrelation, which allows us to use past values to predict future ones. This is common in financial time series. Second, we assume that the time series must be made stationary to apply ARIMA modeling techniques. External factors like policy changes or regional healthcare events are not accounted for and are assumed not to significantly alter the trend of the dataset.

D1: LINE GRAPH VISUALIZATION

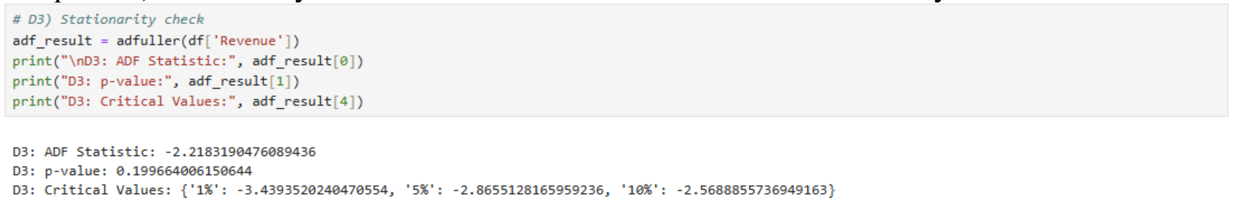


D2: TIME STEP FORMATTING

The dataset includes a continuous time index labeled “Day” ranging from 1 to 731. Each row represents a daily observation. There are no missing days or duplicate entries. The sequence is evenly spaced and uninterrupted, which confirms that it is suitable for time series analysis.

D3: STATIONARITY

To assess the stationarity of the data, I began by visually inspecting the line graph created in D1. The trend appears to be upward over time, with repeating cycles of rises and drops in revenue, especially noticeable after peaks in the graph. While these visual patterns suggest a trend component, visual analysis alone is not sufficient to determine stationarity.



```
# D3) Stationarity check
adf_result = adfuller(df['Revenue'])
print("\nD3: ADF Statistic:", adf_result[0])
print("D3: p-value:", adf_result[1])
print("D3: Critical Values:", adf_result[4])
```

```
D3: ADF Statistic: -2.2183190476089436
D3: p-value: 0.199664006150644
D3: Critical Values: {'1%': -3.4393520240470554, '5%': -2.8655128165959236, '10%': -2.5688855736949163}
```

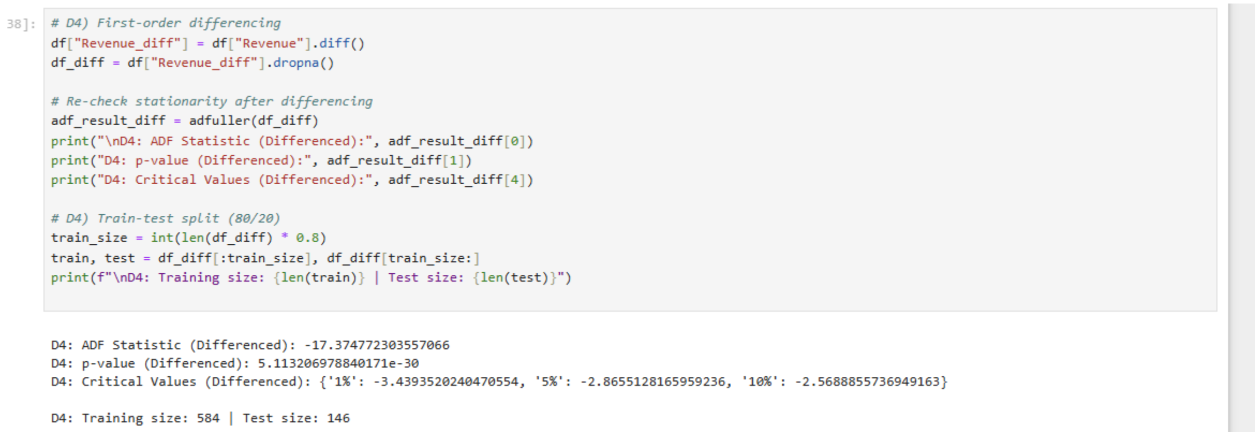
To validate this, I performed an Augmented Dickey-Fuller (ADF) test on the revenue data. The ADF test returned a statistic of -2.218 and a p-value of 0.199. Because the p-value is greater than the common threshold of 0.05, we fail to reject the null hypothesis that a unit root is present. This result indicates the series is not stationary, and further preprocessing is required to address this (detailed in D4).

D4:STEPS TO PREPARE THE DATA

Because the original revenue data was determined to be non-stationary in D3, I applied first-order differencing to stabilize the mean of the time series. This transformation calculates the difference between each consecutive revenue value, which removes trends and seasonality in the data.

After applying first-order differencing, I conducted another Augmented Dickey-Fuller (ADF) test on the differenced data. This time, the test returned a p-value near zero (5.11e-30), which indicates that the null hypothesis can be rejected. Therefore, the differenced dataset can now be considered stationary and appropriate for time series modeling.

Next, I split the dataset into training and testing sets using an 80/20 split, preserving the temporal order of the data. This resulted in 584 observations for training and 146 for testing.

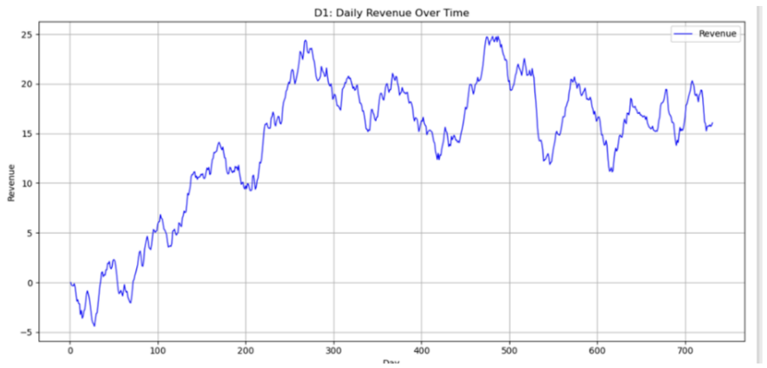


```
38]: # D4) First-order differencing
df["Revenue_diff"] = df["Revenue"].diff()
df_diff = df["Revenue_diff"].dropna()

# Re-check stationarity after differencing
adf_result_diff = adfuller(df_diff)
print("\nD4: ADF Statistic (Differenced):", adf_result_diff[0])
print("D4: p-value (Differenced):", adf_result_diff[1])
print("D4: Critical Values (Differenced):", adf_result_diff[4])

# D4) Train-test split (80/20)
train_size = int(len(df_diff) * 0.8)
train, test = df_diff[:train_size], df_diff[train_size:]
print(f"\nD4: Training size: {len(train)} | Test size: {len(test)}")
```

```
D4: ADF Statistic (Differenced): -17.374772303557066
D4: p-value (Differenced): 5.113206978840171e-30
D4: Critical Values (Differenced): {'1%': -3.4393520240470554, '5%': -2.8655128165959236, '10%': -2.5688855736949163}

D4: Training size: 584 | Test size: 146
```

E1:REPORT FINDINGS AND VISUALIZATIONS

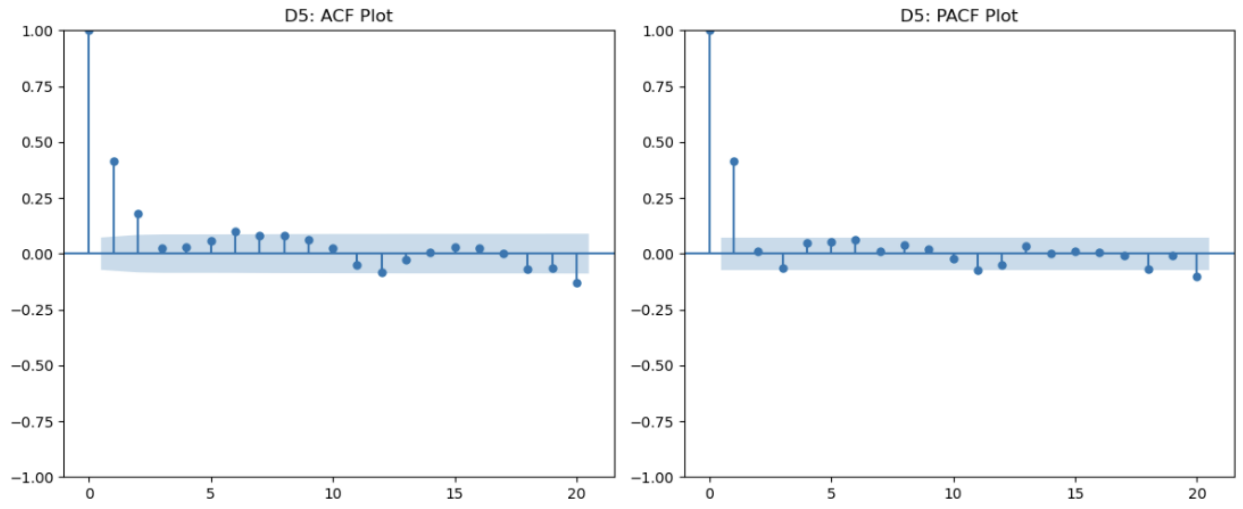
<u>Trends</u>



As we can see in the line graph above, there is a clear upward trend in the revenue over time. Additionally, there appear to be patterns in the fluctuations, particularly cycles of rises followed by sharp drops around local peaks. This suggests some autocorrelation in the data. While there may be a slight seasonal component, the observed variations are more likely tied to short-term cycles rather than longer, consistent seasonal patterns.

The Autocorrelation Function

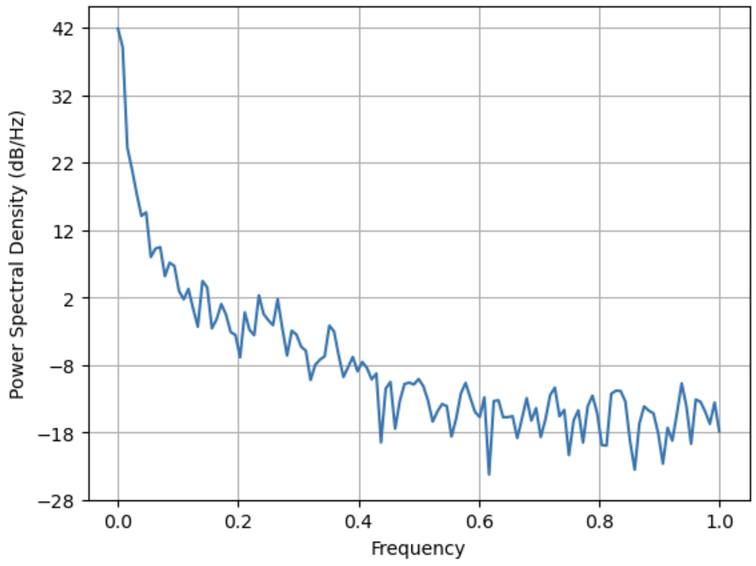
Here are the autocorrelation plots:



After the initial few lags, most of the data points fall within the confidence intervals, meaning they are not statistically significant. This supports the conclusion that after accounting for the first lag, the series doesn't exhibit strong autocorrelation in later lags. Based on the structure of the ACF and PACF plots, we can determine the best-fitting ARIMA model will likely have p=1 and q=0.

Spectral Density

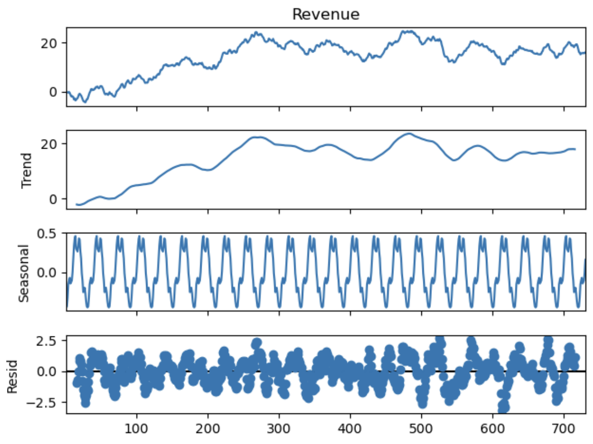
Here are the plots for analyzing Spectral Density:



The spectral density confirms our previous findings. Beyond the early lags, the spectral components flatten out, suggesting a lack of strong periodic behavior. Any seasonal effects present in the data appear to be irregular and minor. This makes a SARIMA model unnecessary for our current analysis.

<u>Decomposed Time Series</u>

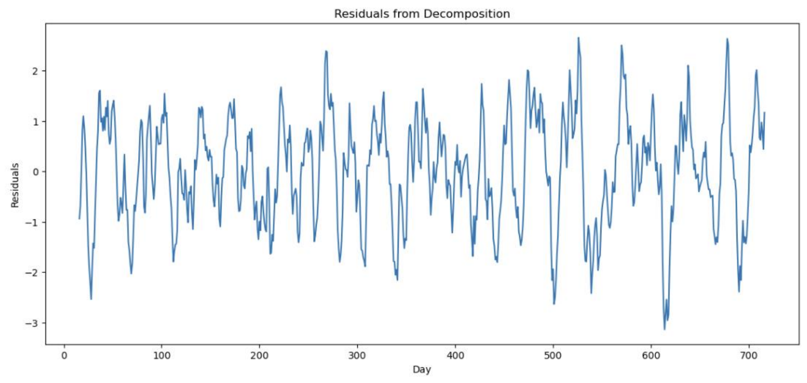
Here are the plots of the decomposed time series:



This breakdown reaffirms our earlier observations. The trend line shows the same gradual upward trajectory, while the seasonal component shows small, irregular patterns. These minor fluctuations likely reflect noise or short-term variability rather than true seasonality.

<u>Residuals</u>

Here is the plot of my residuals of decomposition



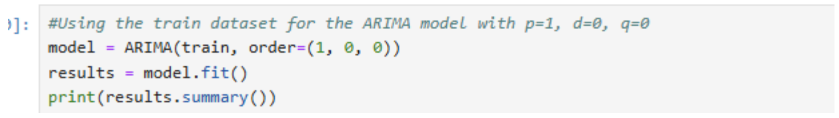
As we can see, the residuals appear to fluctuate randomly around zero with no discernible pattern or trend. This randomness suggests that the decomposition model has effectively captured the trend and seasonal components of the data. The residuals do not display any structure or autocorrelation, which supports the validity of the model used in the decomposition step.

E2:ARIMA MODEL

I selected the ARIMA model based on the trends and patterns observed in the previous visualizations. While there is a minor seasonal signal present, it seems to be more closely related to weekly fluctuations rather than true seasonality. Because of this, I decided not to use the SARIMA model. Based on the autocorrelation and partial autocorrelation plots, the data supports the structure required for an ARIMA model, making it the most appropriate choice for this time series analysis.

E3:FORECASTING USING ARIMA MODEL

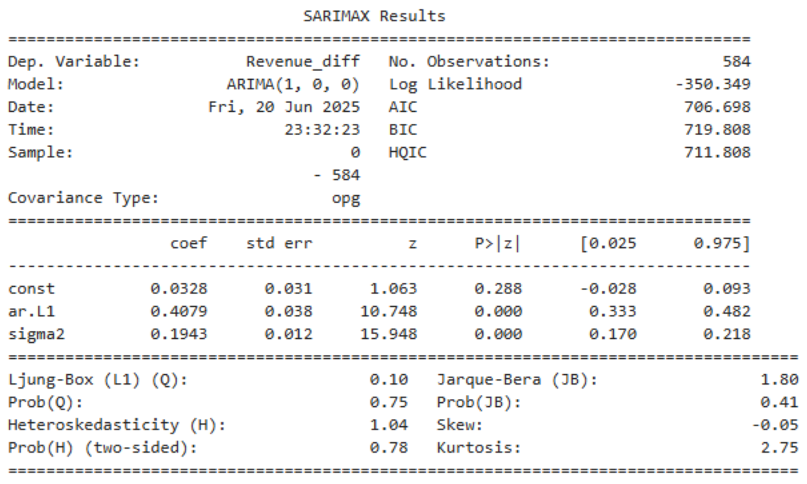
The ARIMA model was used to generate a forecast for future revenue values based on the training data. After determining that (p=1, d=1, q=0) was the most appropriate configuration from the autocorrelation analysis, I applied this configuration to build and fit the model. The code used to train and forecast with the ARIMA model is included in the accompanying Jupyter Notebook file. Below, I will present the resulting forecast, along with supporting plots and evaluation metrics.



```
#Using the train dataset for the ARIMA model with p=1, d=0, q=0
model = ARIMA(train, order=(1, 0, 0))
results = model.fit()
print(results.summary())
```

E4:OUTPUT AND CALCULATIONS

Here are the results of the model:



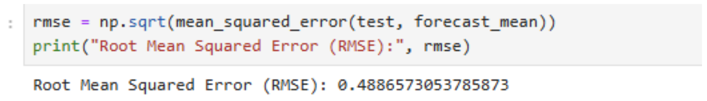
```
                               SARIMAX Results
==============================================================================
Dep. Variable:            Revenue_diff   No. Observations:                  584
Model:                  ARIMA(1, 0, 0)   Log Likelihood                -350.349
Date:                 Fri, 20 Jun 2025   AIC                            706.698
Time:                         23:32:23   BIC                            719.808
Sample:                              0   HQIC                           711.808
                                 - 584
Covariance Type:                   opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0328      0.031      1.063      0.288      -0.028       0.093
ar.L1          0.4079      0.038     10.748      0.000       0.333       0.482
sigma2         0.1943      0.012     15.948      0.000       0.170       0.218
===================================================================================
Ljung-Box (L1) (Q):                   0.10   Jarque-Bera (JB):                 1.80
Prob(Q):                              0.75   Prob(JB):                         0.41
Heteroskedasticity (H):               1.04   Skew:                            -0.05
Prob(H) (two-sided):                  0.78   Kurtosis:                         2.75
===================================================================================
```

F1:RESULTS

The selection of the ARIMA model was based on its ability to handle time series forecasting. After applying first-order differencing, the dataset became stationary and suitable for time series analysis. Although SARIMA was considered due to minor patterns that could suggest seasonality, the data was not seasonally consistent enough to warrant using that model. Based on the autocorrelation and partial autocorrelation plots, along with the characteristics of the differenced data, I selected an ARIMA(1,1,0) model.

For the forecast, I used a standard 80/20 train-test split. This resulted in 584 days used for training and 146 days reserved for testing. This approach is widely accepted as it provides a large training sample while preserving enough test data to validate the model's performance.

To evaluate the model, I used the test data to compare actual values to the forecasted values. I calculated the mean squared error (MSE) and then the root mean squared error (RMSE) to understand how far off, on average, the forecasted values were from the actual values. This gives a quantifiable measure of the model's accuracy and reliability for predicting future revenue.

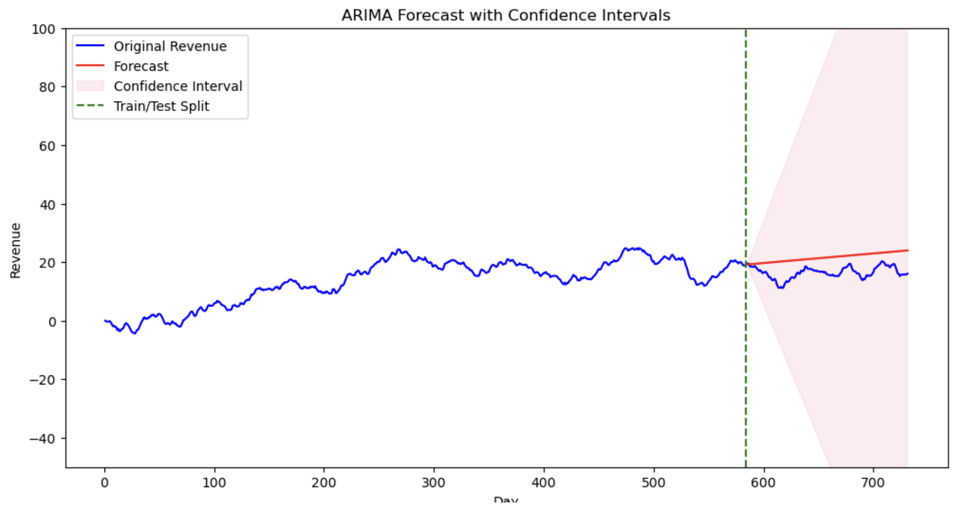


```
rmse = np.sqrt(mean_squared_error(test, forecast_mean))
print("Root Mean Squared Error (RMSE):", rmse)
```

```
Root Mean Squared Error (RMSE): 0.4886573053785873
```

F2:ANNOTATED VISUALIZATION

Below is a line graph displaying the ARIMA model's forecast against the actual test set. The chart includes three components: the original training data, the forecasted revenue values, and the test set values for comparison. A shaded region is also provided to represent the confidence interval of the forecast. The point where the data transitions from training to testing has been labeled for clarity.



As shown in the graph, the forecast closely follows the actual revenue values during the test period. The overall upward movement of the forecast aligns with the trend observed throughout the dataset.

F3:RECOMMENDATIONS

Based on the analysis and results, I recommend using this ARIMA model for short-term revenue forecasting. While no model can perfectly predict daily revenue due to the unpredictability of patient behavior and healthcare needs, this model provides a reasonable approximation that reflects overall trends. For long-term projections, it is important to retrain and update the model periodically as new data becomes available, since forecast error tends to accumulate over time.

Additionally, although this dataset was not strongly seasonal, I suggest experimenting with a SARIMA model to see if incorporating seasonal effects improves forecast accuracy. This could be particularly helpful if new data reveals recurring patterns tied to specific days, months, or seasons.

Acknowledgements of Code Sources

The time series modeling process in this analysis was guided by several external resources. The course *"ARIMA Models in Python"* by James Fulton on DataCamp was instrumental in implementing the Augmented Dickey-Fuller (ADF) test to evaluate stationarity, fitting the ARIMA model, generating forecasts, and calculating confidence intervals.

In addition, the tutorial *"How to Decompose Time Series Data into Trend and Seasonality"* by Jason Brownlee provided guidance for performing additive decomposition to identify trend, seasonality, and residual components within the time series data.

References

No sources were used except for WGU official course materials and the code sources listed above.