Part 1

A1.
How can time series modeling be used to identify patterns in daily revenue data from the first two years of operation to predict periods of high customer churn risk in a telecommunications company?

A2.
The primary objectives of the data analysis are to identify revenue trends, analyze seasonal patterns, forecast potential future revenue, detect anomalies in revenue, and correlate these patterns with customer churn. The goal of identifying revenue is to determine the overall direction and magnitude of daily revenue changes over the first two years of operation.

Forecasting future revenue aims to develop predictive models, such as an Autoregressive Integrated Moving Average (ARIMA) model, to project future daily revenue based on historical trends. Detecting anomalies focuses on identifying significant outliers or unusual fluctuations in revenue, which could indicate changes in customer behavior or market conditions.

Finally, correlating revenue patterns with customer churn seeks to establish a connection between observed revenue trends and periods of high customer churn. Although the dataset directly provides revenue data, significant drops or unexpected changes in revenue can indirectly point to times of increased churn. These objectives aim to leverage the dataset effectively, providing the telecommunications company with actionable information to enhance customer retention strategies.

Part 2

B.
The assumptions of a time series model, particularly focusing on stationary and autocorrelated are as follows:

Stationary: Time series models often assume that the data is stationary, meaning the statistical properties such as mean, and variance are constant over time. This is crucial because non-stationary data can lead to misleading results, mistaking trends, or seasonality for noise. Techniques like differencing or log transformation are typically applied to achieve stationary. In the context of ARIMA models (which include ARMA as a special case), stationary also implies that the errors (residuals) are white noise. They are independently and normally distributed with a constant variance.

Autocorrelation: Time series models assume that data points are dependent on past values. This is the foundation for autoregressive (AR) components, where current values are regressed on previous values (e.g., today's value regressed on yesterday's value). For moving average (MA) components, the model incorporates past errors into the prediction. The combination of AR and MA components in ARMA (and ARIMA) models assumes that the current value is influenced by both past values and past errors, making the errors autocorrelated.

White Noise: For accurate model fitting and forecasting, it is essential that the residuals (errors) of the model are white noise. This means they should be uncorrelated, normally distributed, and have a constant variance. Ensuring this helps in validating the model assumptions and improving forecast reliability.

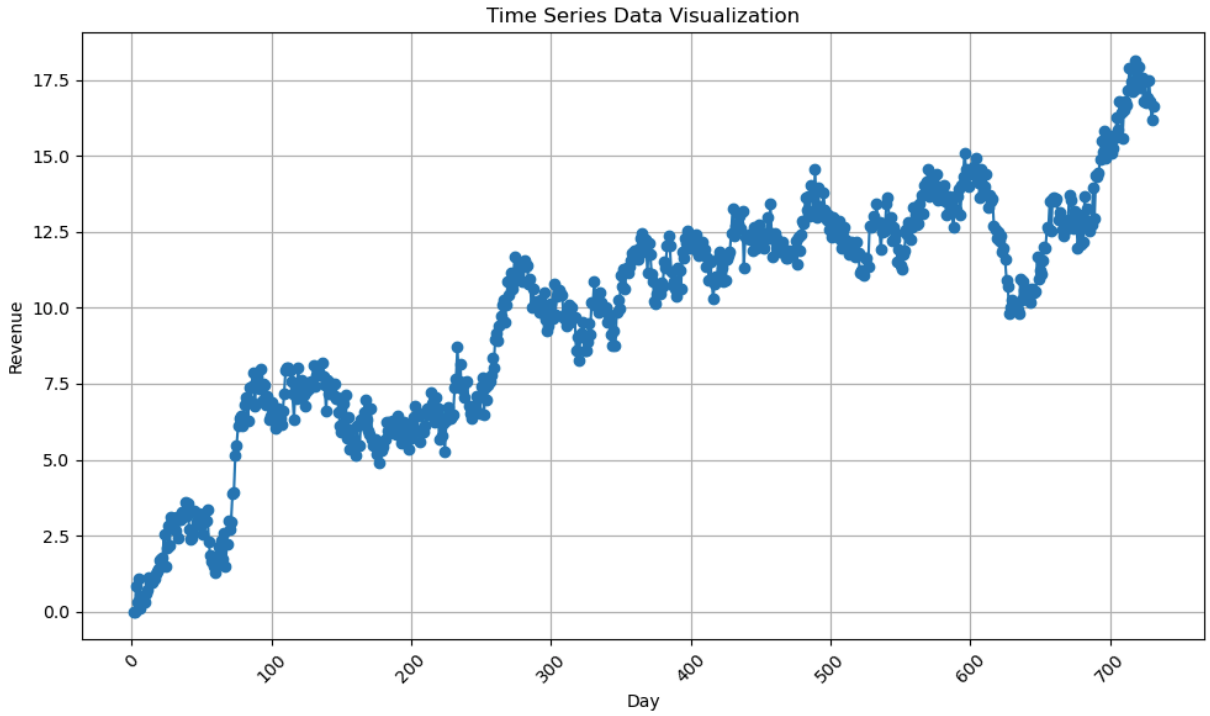
Jasmine Cooper
D213
Task 1: Time Series Modeling
June 15, 2024
Western Governor's University

In text citations:
("First things first", n.d.)
("AR and MA together", n.d.)

Part 3

C1.

Line graph visualizing the realization of the time series:



C2.

The analysis was conducted with daily intervals for the time step formatting of the realization. The sequence spans a total of 731 days. Within this period, a total of 731 measurements were expected. Since there is no missing data, all 731 measurements were recorded, indicating that there were no gaps in the data. Upon reviewing the frequency of measurements, it was found that all intervals between consecutive recordings were consistently one day, with 730 occurrences of 1-day intervals. This confirms that that the measurements were taken daily without any missing days.

C3.

The stationary of the time series was assessed using multiple diagnostic tests and visual inspections.

The results of the Augmented Dickey-Fuller (ADF) and Kwiatkowski-Phillips-Schmidt-Shin (KPSS) tests provided valuable insights into the stationary of the time series. The ADF test yielded a p-value of 0.32,

which is greater than the conventional significance level of 0.05. Consequently, I fail to reject the null hypothesis, suggesting non-stationary. Conversely, the KPSS test produced a p-value of 0.01, failing below the significance threshold of 0.05. Therefore, we reject the null hypothesis and conclude that the series is non-stationary based on the KPSS test.

Visual inspections of the Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) plots further confirmed the findings of the diagnostic tests. The ACF plot displayed a slow decay, indicative of non-stationary. However, interpreting the PACF plot proved to be more challenging. While a sharp cutoff observed after lag 3, subsequent lags displayed only marginal significance. This ambiguity makes it difficult to conclusively determine the stationary of the series based solely on PACF. Considering the consistent indications of non-stationary from the ADF, KPSS, and ACF analyses, the PACF's inconclusive nature aligns with overall assessment of non-stationary.

While the PACF provided some insights, its interpretation was inconclusive in isolation, emphasizing the importance of integrating multiple diagnostic techniques in assessing time series stationary. In summary, the combined evidence from the ADF, KPSS, and ACF analyses suggests that the time series exhibits characteristics of non-stationary.

C4.

Steps to prepare data for analysis:

1. Import Necessary Libraries: I began by importing essential libraries for data manipulation (pandas, numpy), visualization (missingno, matplotlib), and time series analysis (statsmodels).
2. Load Data: The time series data was loaded from a CSV file into a pandas DataFrame. I displayed the first 10 rows to verify that the data was loaded correctly and to check for any missing values.
3. Visualize Missing Data: Using the missingno library, I created a bar plot to visualize missing data in the DataFrame, which helps in identifying columns with missing values.
4. Visualize Time Series Data: I plotted the time series data to observe the trend of Revenue over Day, ensuring a clear understanding of the data's behavior.
5. Check Stationary: I performed the ADF and KPSS tests to determine that the time series data is non-stationary. A p-value less than the significance level, 0.05, indicates the series is stationary for the ADF test. Regarding the KPSS test, a p-value greater than the significance level, 0.05, suggests the series is stationary. I also plotted the ACF to understand the autocorrelation in the time series up to 50 lags and plotted the PACF to understand the autocorrelation in the data up to 50 lags.
6. Split data: I split the dataset into training and test sets. The training set contains all data except the last 30 days, which are reserved for testing. This approach ensures that the model is trained on historical data and tested on future data.
7. To achieve stationary, I differenced the training data and plotted the differenced data to visualize the changes. This step ensures that the training data is stationary which is crucial requirements for fitting the ARIMA model.

By following these steps, I prepared the data effectively for time series analysis and forecasting the ARIMA model. This thorough preparation ensures that your model will be trained on a solid foundation of clean and well-understood data.

C5.

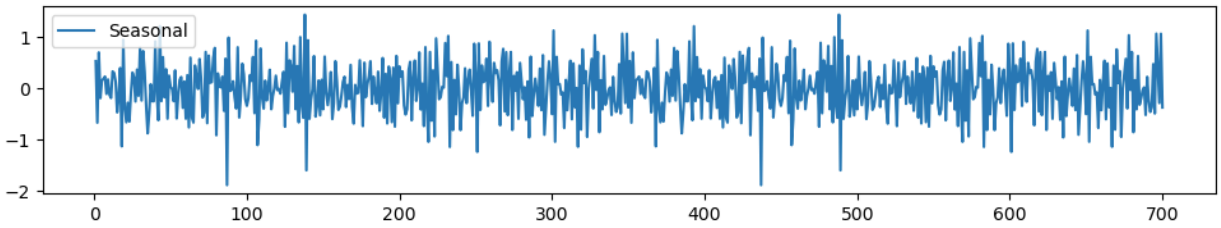
Please see a copy of the cleaned data set attached to the submission titled ‘cleaned_data.csv’.

In text citations:
(“Random Walk”, n.d.)
(“Autocorrelation Function”, n.d.)
(“Stationary”, n.d.)

Part 4

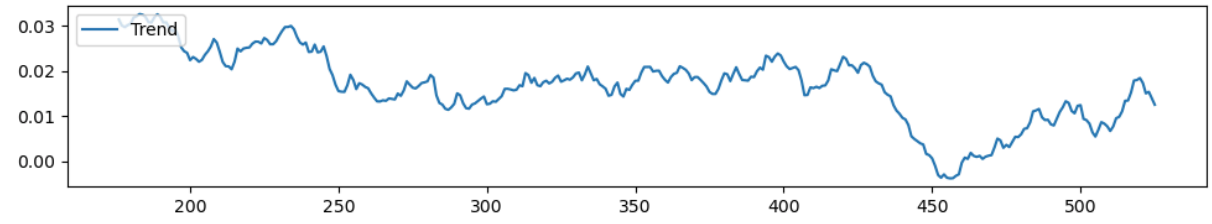
D1.

Presence or lack of a seasonal component:



The plot of the seasonal component shows minor fluctuations around the zero line. The lack of peaks and troughs in the plot suggests that seasonality is not present in the data or not highly significant. The data does not exhibit strong periodic patterns that are typically associated with a clear seasonal component.

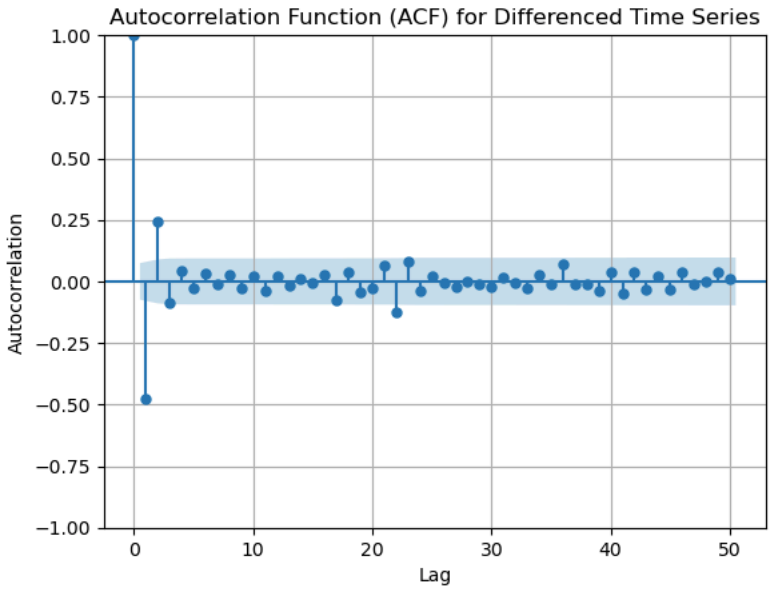
Trends:



This plot shows the trend component of the time series, highlighting the underlying direction in which the data is moving over time. The trend component displays long-term movements without a strong consistent upward or downward trend. This indicates that the data is relatively stable over time with only minor variations. Notably, there is a significant drop in revenue around the 450th day. Since there is no sustained upward or downward movement, it suggests that the time series lacks a strong trend

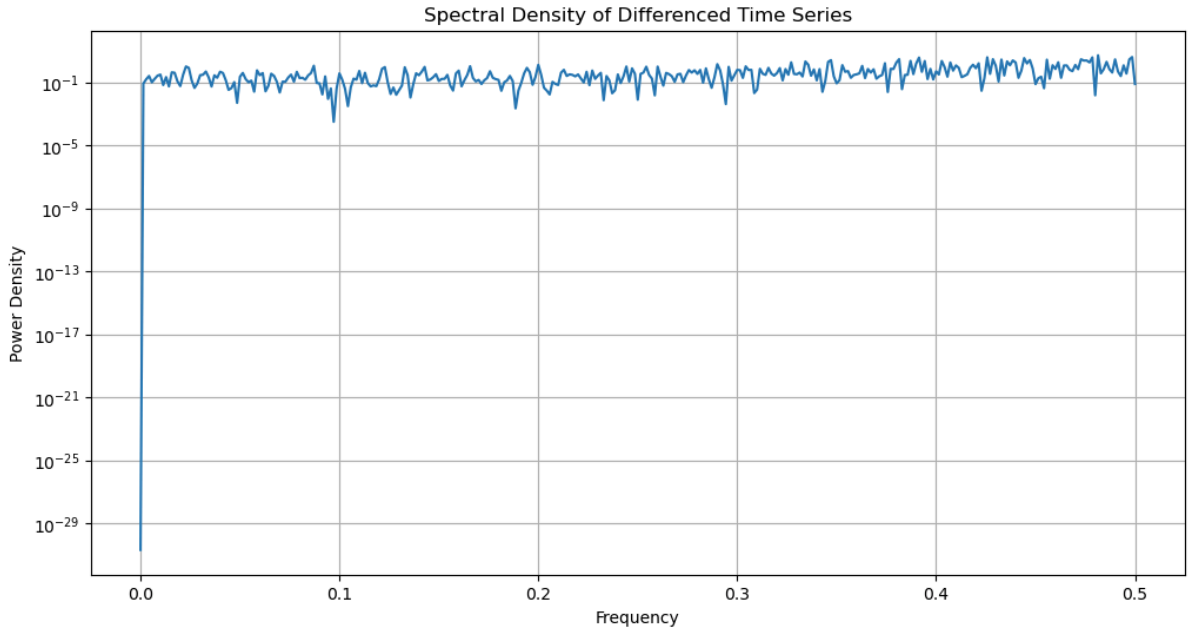
component. This stability in the trend component implies that the time series is stationary concerning its trend.

The autocorrelation function:



The ACF plot for the differenced time series shows significant autocorrelation at lag 1, lag 2, and lag 3 with subsequent lags displaying much lower correlations. The initial spikes at lags 1, 2, and 3 indicate some short-term dependencies in the data. The rapid drop-off in autocorrelation after lag 3 suggests that the differencing has been effective in achieving stationary, as no strong autocorrelation patterns remain beyond these initial lags. This indicates that the data is likely stationary after differencing, with dependencies largely confined to the first few lags.

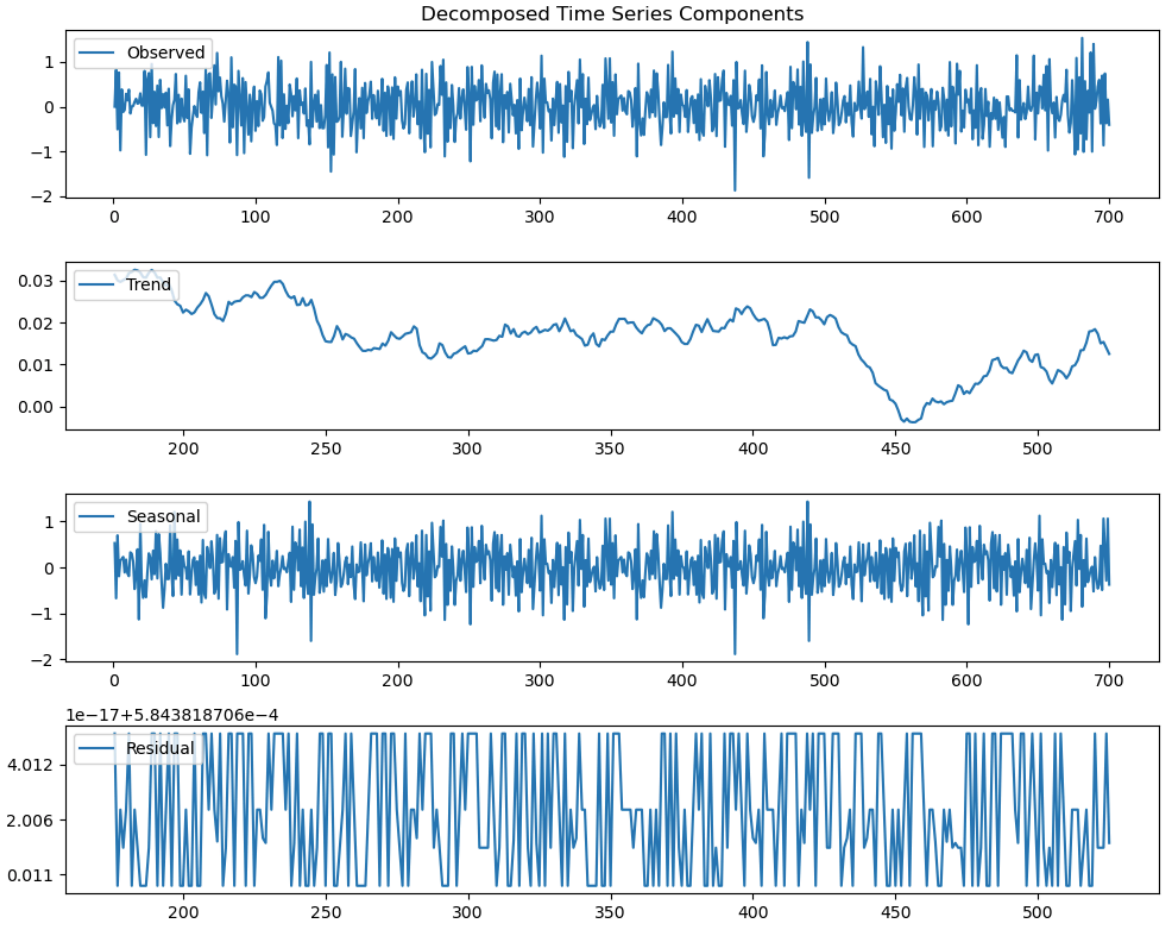
The Spectral density:



The spectral density plot of the differenced time series shows a relatively flat distribution of power frequencies, with no significant peaks. The flat spectral density indicates that the differenced time series does not contain strong periodic components of regular cycles. This supports the conclusion that the data lacks significant seasonal or cyclic behavior, reinforcing the earlier findings from the seasonal component analysis.

Jasmine Cooper
D213
Task 1: Time Series Modeling
June 15, 2024
Western Governor's University

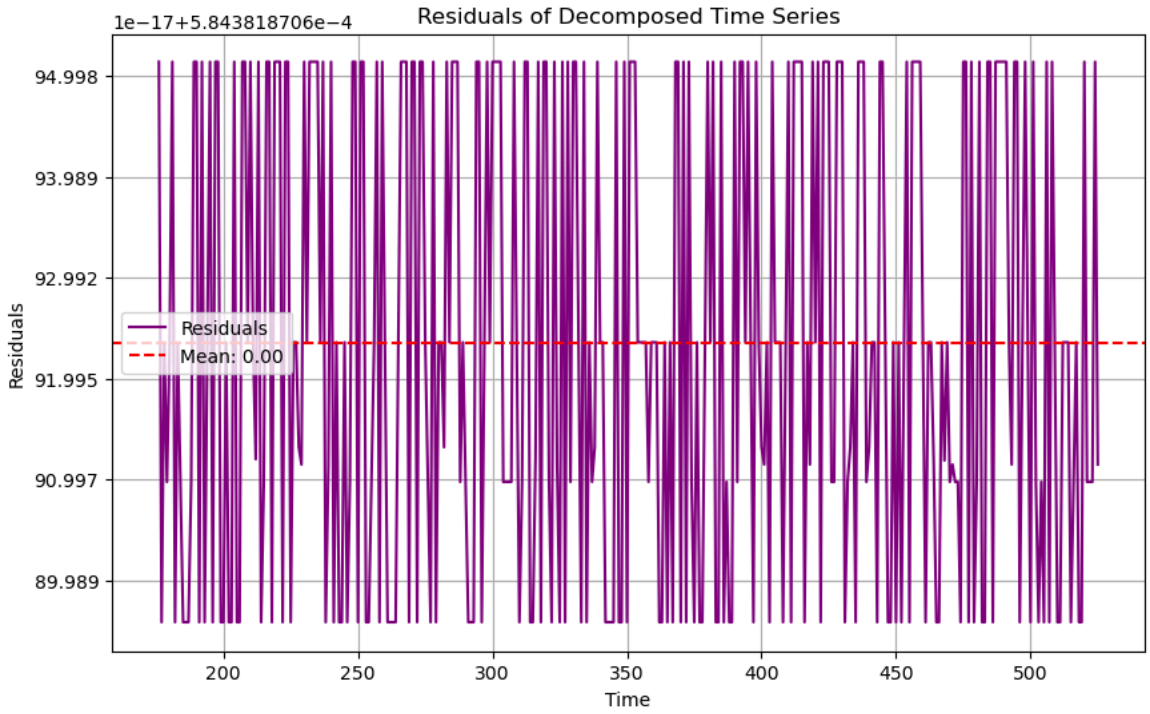
The decomposed time series:

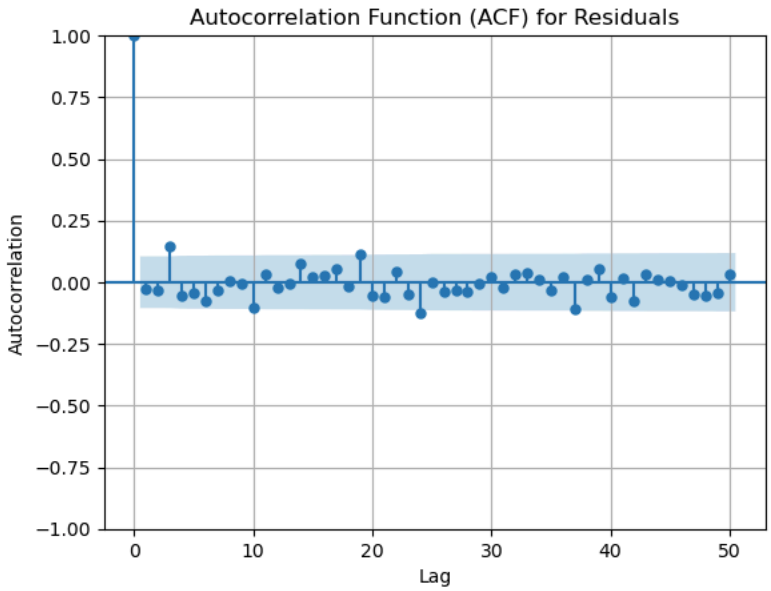


The decomposition of the time series data helps in understanding the individual contributions of trend and seasonal components. The decomposed time series includes the original time series data, trend, and seasonal components, as well as the residuals. The original time series data displays the combined effects of trend, seasonality, and random noise. By decomposing the series, we can analyze each component separately. The trend component shows slight fluctuations, indicating minor long-term movements in the data. This indicates that while there are small variations, the overall trend is relatively stable, suggesting no major long-term direction in the data. The seasonal component, as previously discussed, reveals repetitive patterns at regular intervals. In this case, the seasonal variations are relatively weak, with minor fluctuations around the zero line. The weak seasonality suggests that the data does not have a strong periodic pattern, which aligns with the findings from the spectral density analysis. The residuals, which should ideally be random noise, confirm that the decomposition has effectively separated the systematic components (trend and seasonality) from the random noise.

Jasmine Cooper
D213
Task 1: Time Series Modeling
June 15, 2024
Western Governor's University

Confirmation of the lack of trends in the residuals of the decomposed series:





The residuals of the decomposed series have a mean around zero, suggesting that systematic components (such as trend and seasonality) have been effectively removed during the decomposition

Jasmine Cooper
D213
Task 1: Time Series Modeling
June 15, 2024
Western Governor's University

process, leaving behind fluctuations that do not exhibit a consistent bias over time. The absence of any discernable pattern or trend in the residuals further supports the effectiveness of the decomposition. With no clear structure left in the residuals, they resemble white noise, which is a key characteristic of stationary time series data. The ACF plot for the residuals of the decomposed time series shows the autocorrelation of the residuals at different lags. There is a significant peak at lag 1, but subsequent lags fall within the confidence bounds. The significant peak at lag 1 indicates some short-term dependencies in the residuals. The fact that autocorrelation values for most lags fall within the confidence bounds indicates that the residuals do not exhibit significant autocorrelation beyond the first lag. This lack of discernable pattern suggests that the residuals are indeed randomly distributed, resembling white noise.

This analysis confirms the remaining residuals are purely random noise, validating the effectiveness of the decomposition. Furthermore, that the residuals are appropriately modeled as white noise, indicating successful decomposition of the time series.

D2.

Based on the analysis of the time series data, an autoregressive integrated moving average (ARIMA) model was identified that effectively captures the observed trend and seasonality. The model includes a first-order differencing term (d=1) to account for trend, indicating that the data required one level of differencing to achieve stationary. Additionally, an autoregressive term of order 1 (p=1) was incorporated to capture the dependence between consecutive observations.

The absence of significant seasonal patterns in the data, as confirmed by the seasonal component analysis, led to the exclusion of seasonal differencing or seasonal autoregressive terms in the ARIMA model terms (D=0, P=0). The auto_arima function, therefore, selected the most appropriate non-seasonal ARIMA model to capture the underlying patterns in the dataset.

The selected ARIMA model, ARIMA (1,1,0)(0,0,0)[7], achieved the lowest Akaike Information Criterion (AIC) value of 935.52, among the tested models, indicating a good balance between the model fit and complexity. The corresponding Bayesian Information Criterion (BIC) was 949.17.

The performance of the ARIMA model was evaluated using the Root Mean Squared Error (RMSE), which yielded a value of 1.49. This represents the average magnitude of forecasting errors.

Overall, the selected ARIMA model demonstrates its capability to effectively capture the observed trend and seasonality in the time series data.

D3.

The ARIMA model was derived and used to forecast revenue for the next 30 days beyond the training period. Additionally, predictions were made based on the test data, and both forecasts and predictions were visualized with confidence intervals. Model performance metrics (RMSE, AIC, BIC) were also calculated to assess accuracy.

D4.

The outputs (charts) of the analysis performed are above in each respective section. The calculations (code) of the analysis performed:

```
#the presence or lack of a seasonal component
#trends

from statsmodels.tsa.seasonal import seasonal_decompose

# Decompose the differenced time series data
# Adjust seasonal_period based on the data frequency and observed seasonality
seasonal_period = 350

decomposition = seasonal_decompose(training_data['Revenue_diff'], model='additive',
period=seasonal_period)

# Plot decomposed components
plt.figure(figsize=(10, 8))

plt.subplot(411)
plt.plot(decomposition.observed, label='Observed')
plt.legend(loc='upper left')

plt.subplot(412)
plt.plot(decomposition.trend.dropna(), label='Trend')
plt.legend(loc='upper left')

plt.subplot(413)
plt.plot(decomposition.seasonal.dropna(), label='Seasonal')
plt.legend(loc='upper left')

plt.subplot(414)
plt.plot(decomposition.resid.dropna(), label='Residual')
plt.legend(loc='upper left')

plt.tight_layout()
plt.show()

#the acf

from statsmodels.graphics.tsaplots import plot_acf

# Plot ACF for the differenced data
plt.figure(figsize=(12, 6))
plot_acf(training_data['Revenue_diff'], lags=50)
plt.title('Autocorrelation Function (ACF) for Differenced Time Series')
plt.xlabel('Lag')
plt.ylabel('Autocorrelation')
```

```
plt.grid(True)
plt.show()

#spectral density

from scipy.signal import periodogram

# Calculate and plot the spectral density
frequencies, power_density = periodogram(training_data['Revenue_diff'])

plt.figure(figsize=(12, 6))
plt.semilogy(frequencies, power_density)
plt.title('Spectral Density of Differenced Time Series')
plt.xlabel('Frequency')
plt.ylabel('Power Density')
plt.grid(True)
plt.show()


seasonal_period = 350

# Decompose the differenced time series data
decomposition = seasonal_decompose(training_data['Revenue_diff'], model='additive',
period=seasonal_period)

# Plot decomposed components
plt.figure(figsize=(10, 8))

plt.subplot(411)
plt.plot(decomposition.observed, label='Observed')
plt.legend(loc='upper left')
plt.title('Decomposed Time Series Components')

plt.subplot(412)
plt.plot(decomposition.trend.dropna(), label='Trend')
plt.legend(loc='upper left')

plt.subplot(413)
plt.plot(decomposition.seasonal.dropna(), label='Seasonal')
plt.legend(loc='upper left')

plt.subplot(414)
plt.plot(decomposition.resid.dropna(), label='Residual')
plt.legend(loc='upper left')

plt.tight_layout()
plt.show()
```

```
#confirmation of the lack of trends in the residuals of the decomposed series
# Plot residuals
plt.figure(figsize=(10, 6))
plt.plot(decomposition.resid.dropna(), label='Residuals', color='purple')
plt.title('Residuals of Decomposed Time Series')
plt.xlabel('Time')
plt.ylabel('Residuals')
plt.legend()
plt.grid(True)

# Print the center of the residuals
residual_center = decomposition.resid.mean()
plt.axhline(y=residual_center, color='r', linestyle='--', label=f'Mean: {residual_center:.2f}')
plt.legend()

plt.show()

# Plot ACF for the residuals
plt.figure(figsize=(12, 6))
plot_acf(decomposition.resid.dropna(), lags=50)
plt.title('Autocorrelation Function (ACF) for Residuals')
plt.xlabel('Lag')
plt.ylabel('Autocorrelation')
plt.grid(True)
plt.show()
```

D5.

The code used to support the implementation of the time series model:

```
pip install pmdarima
from pmdarima import auto_arima
import matplotlib.pyplot as plt
import numpy as np

# Fit auto ARIMA model
model = auto_arima(training_data['Revenue'], trace=True, stepwise=True)

# Fit the best model to the entire training data
fit_model = model.fit(training_data['Revenue'])

# Forecast the next 30 days
forecast, conf_int = fit_model.predict(n_periods=30, return_conf_int=True)

# Predict against test data
predictions, conf_int_pred = fit_model.predict(n_periods=len(testing_data), return_conf_int=True)
```

```
# Plot the combined graph with confidence intervals
plt.figure(figsize=(10, 6))

# Plot training data
plt.plot(training_data['Day'], training_data['Revenue'], label='Training Data')

# Plot testing data
plt.plot(testing_data['Day'], testing_data['Revenue'], label='Actual Revenue', color = 'lightblue')

# Plot forecast
plt.plot(testing_data['Day'], forecast, color='green', label='Forecast', linestyle='-')
plt.fill_between(testing_data['Day'], conf_int[:, 0], conf_int[:, 1], color='lightgreen', alpha=0.2)

# Plot predictions against testing data
plt.plot(testing_data['Day'], predictions, color='orange', label='Predictions', linestyle='--')
plt.fill_between(testing_data['Day'], conf_int_pred[:, 0], conf_int_pred[:, 1], color='darkorange',
alpha=0.2)

plt.title('Revenue Forecast with Confidence Intervals')
plt.xlabel('Day')
plt.ylabel('Revenue')
plt.legend()
plt.show()

# Evaluate the model performance
from sklearn.metrics import mean_squared_error

# Calculate RMSE
mse = mean_squared_error(testing_data['Revenue'], predictions)
rmse = np.sqrt(mse)

# Output RMSE, AIC, BIC
print(f'Root Mean Squared Error (RMSE): {rmse}')
print(f'AIC: {fit_model.aic()}')
print(f'BIC: {fit_model.bic()}')
```

In text citations:
(“Model choice and residual analysis”, n.d.)
(“ARIMA models”, n.d.)
(“Seasonal time series”, n.d.)

Part 5

E1.

Selection of an ARIMA model:

The ARIMA model selected is ARIMA (1,1,0), which achieved the lowest Akaike Information Criterion (AIC) value of 935.52 among the tested models, indicating an optimal balance between model fit and complexity. The corresponding Bayesian Information Criterion (BIC) was 949.17. The selection process was automated using auto_arima. This function simplifies model selection by analyzing the data and determining the best-fit parameters (p,d,q) based on statistical criteria. It performs tests, such as ADF, to determine if differencing is required to achieve stationary, thus identifying the appropriate value for the differencing parameter (d).

Although the process is automated, auto_arima, implicitly considers the ACF and PACF plots. These plots help guide the selection of the autoregressive (p) and moving average parameters. For instance, a slow decay in the ACF plot suggests non-stationary, indicating the need for differencing. Sharp cutoffs in the ACF and PACF plots provide insights about potential values for p and q. The function evaluates multiple ARIMA configurates and selects the model with the lowest AIC and BIC values, balancing model fit and complexity and helping to avoid overfitting.

Prediction Interval of the Forecast:

The prediction interval provides a range within which future values are expected to lie with a certain probability, considering the uncertainty in the model's predictions. This is crucial for understanding the potential variability in the forecasted values and aids in risk management and decision-making process.

Justification of the Forecast Length:

The chosen forecast length of 30 days is suitable for the given dataset, which spans over two years. This length allows for short-term predictions that are practical and actionable for business planning and decision-making. Given that the data is daily, a 30-day forecast provides a one-month outlook, which is practical for planning and operations. Additionally, ARIMA models tend to perform well with short to medium-term forecasts, and a 30-day forecast is within this range. The substantial data history ensures that the model has enough information to accurately capture underlying trends and patterns. Short-term forecast typically offers higher accuracy and reliability, making them ideal for immediate strategic use.

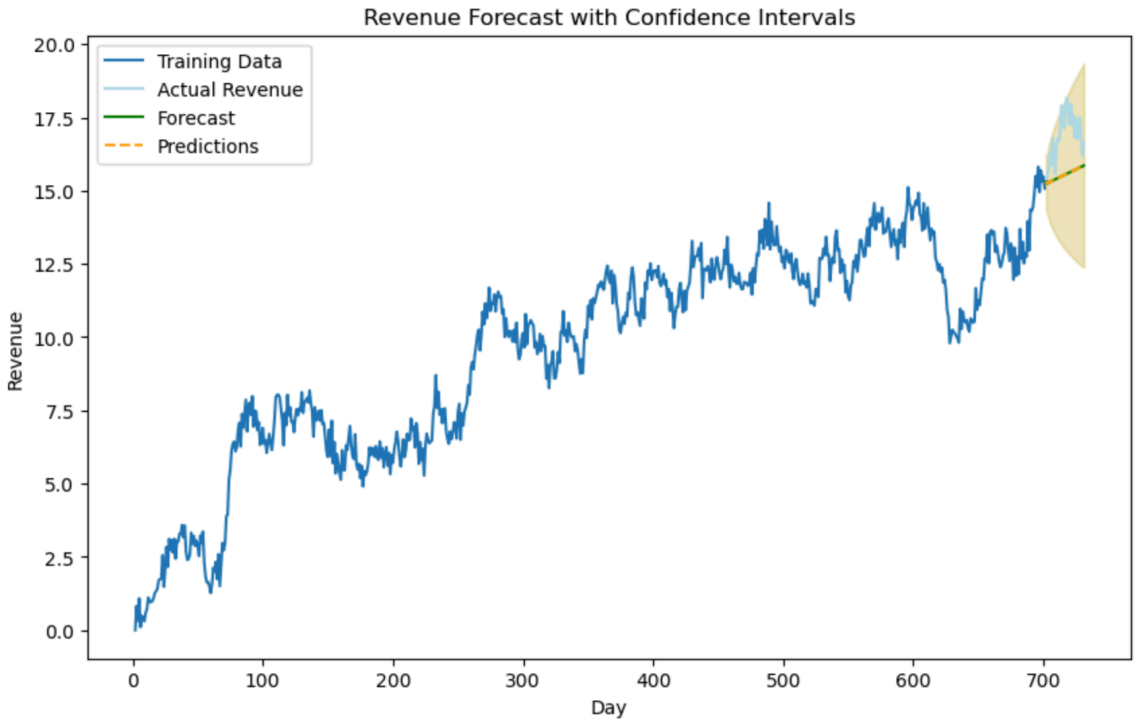
Model Evaluation Procedure and Error Metric:

The model's performance was evaluated using various error metrics to measure how well the model's predictions match the actual data. Metrics such as AIC, BIC, and RMSE were utilized. Lower values for AIC and BIC indicate a better fit. The RMSE, which measures the average magnitude of the prediction errors was calculated to be 1.49, indicating the average between the model's predictions and the actual values. This low RMSE value reflects the model's accuracy in forecasting the revenue. By testing various combinations of (p,d,q) and comparing their AIC and BIC values, auto_arima identifies the most appropriate ARIMA model for the given time series, ensuring a reliable and robust forecast.

Jasmine Cooper
D213
Task 1: Time Series Modeling
June 15, 2024
Western Governor's University

E2.

Please see an annotated visualization of the forecast of the final model compared to the test set.



E3.

Leverage the 30-day revenue forecast to refine your short-term business strategies. Use the forecast to adjust resource allocation, manage inventory levels, and plan targeted marketing campaigns. This approach will help maintain profitability, avoid overstocking, and boost sales during anticipated low-revenue periods. Regularly update the ARIMA model with new data to ensure accurate and actionable forecasts.

In text citations:
(“Forecasting ARIMA”, n.d.)

Jasmine Cooper
D213
Task 1: Time Series Modeling
June 15, 2024
Western Governor’s University

("ARIMA models", n.d.)
("ARIMA – integrated ARIMA", n.d.)

Part 6

Citations for code:

DataCamp. (n.d.). Autocorrelation Function [Video file]. Retrieved from https://campus.datacamp.com/courses/time-series-analysis-in-python/some-simple-time-series?ex=1

DataCamp. (n.d.). White Noise [Video file]. Retrieved from https://campus.datacamp.com/courses/time-series-analysis-in-python/some-simple-time-series?ex=4

DataCamp. (n.d.). Random Walk [Video file]. Retrieved from https://campus.datacamp.com/courses/time-series-analysis-in-python/some-simple-time-series?ex=6

DataCamp. (n.d.). Stationary [Video file]. Retrieved from https://campus.datacamp.com/courses/time-series-analysis-in-python/some-simple-time-series?ex=11

DataCamp. (n.d.). Model choice and residual analysis [Video file]. Retrieved from https://campus.datacamp.com/courses/arima-models-in-r/fitting-arma-models?ex=8

DataCamp. (n.d.). ARIMA – integrated ARIMA [Video file]. Retrieved from https://campus.datacamp.com/courses/arima-models-in-r/arima-models?ex=1

DataCamp. (n.d.). Forecasting ARIMA [Video file]. Retrieved from https://campus.datacamp.com/courses/arima-models-in-r/arima-models?ex=8

DataCamp. (n.d.). ARIMA models [Video file]. Retrieved from https://campus.datacamp.com/courses/forecasting-in-r/forecasting-with-arima-models?ex=5

DataCamp. (n.d.). Seasonal time series [Video file]. Retrieved from https://campus.datacamp.com/courses/arima-models-in-python/seasonal-arima-models?ex=1

Citations for content:

DataCamp. (n.d.). Autocorrelation Function [Video file]. Retrieved from https://campus.datacamp.com/courses/time-series-analysis-in-python/some-simple-time-series?ex=1

DataCamp. (n.d.). White Noise [Video file]. Retrieved from https://campus.datacamp.com/courses/time-series-analysis-in-python/some-simple-time-series?ex=4

DataCamp. (n.d.). Random Walk [Video file]. Retrieved from https://campus.datacamp.com/courses/time-series-analysis-in-python/some-simple-time-series?ex=6

DataCamp. (n.d.). Stationary [Video file]. Retrieved from https://campus.datacamp.com/courses/time-series-analysis-in-python/some-simple-time-series?ex=11

DataCamp. (n.d.). First things first [Video file]. Retrieved from https://campus.datacamp.com/courses/arima-models-in-r/time-series-data-and-models?ex=1

DataCamp. (n.d.). AR and MA together [Video file]. Retrieved from https://campus.datacamp.com/courses/arima-models-in-r/fitting-arma-models?ex=5

DataCamp. (n.d.). Model choice and residual analysis [Video file]. Retrieved from https://campus.datacamp.com/courses/arima-models-in-r/fitting-arma-models?ex=8

DataCamp. (n.d.). ARIMA – integrated ARIMA [Video file]. Retrieved from https://campus.datacamp.com/courses/arima-models-in-r/arima-models?ex=1

DataCamp. (n.d.). Forecasting ARIMA [Video file]. Retrieved from https://campus.datacamp.com/courses/arima-models-in-r/arima-models?ex=8

DataCamp. (n.d.). ARIMA models [Video file]. Retrieved from https://campus.datacamp.com/courses/forecasting-in-r/forecasting-with-arima-models?ex=5

DataCamp. (n.d.). Seasonal time series [Video file]. Retrieved from https://campus.datacamp.com/courses/arima-models-in-python/seasonal-arima-models?ex=1