Forecasting Boston's Future: A Time Series Exploration

Addressed to:
The Boston Planning and Development Authority (BPDA)

Author:
Aman Thakur

Issues

The Boston Planning and Redevelopment Authority (BPDA) collects and analyses economic data in areas such as employment, housing, travel, and real estate development. This dataset includes data such as the number of domestic and international passengers at Logan Airport, the total number of international flights at the same airport, hotel occupancy rates in Boston, the city's average daily hotel rate, the city's unemployment rate, labour rates, and more.

- Is there a long-term trend in the number of international flights at Logan International Airport?
- What is the annual growth rate of international flights at Logan International Airport?
- Are their seasonal patterns or specific months with consistently higher or lower international flight numbers?
- Which months typically experience the highest and lowest numbers of international flights?
- How does the number of international flights correlate with other economic indicators in the dataset?
- Can I predict the future number of international flights based on historical data?
- Are there periods of consistent increase or decrease in unemployment?
- How do holidays or specific times of the year affect unemployment?

Findings

- The number of international flights at Logan International Airport has been trending upward over the studied period, according to our analysis. Data point to a possible growth trajectory, with a slow increase in the number of international flights.
- One important metric, the annual growth rate, was estimated to be around 4.90% . This percentage represents the average annual growth in international travel, giving our findings a strong statistical basis.
- I consistently see higher international flight activity in July and August during the summer, which suggests a peak travel season. On the other hand, I can observe a decline in international flight numbers in February and November.
- An increase in international travel usually corresponds to a decrease in unemployment. As a result, there appears to be a link between increased international travel and increased job opportunities. Similarly, I have found that an increase in international flights corresponds to an increase in the number of people working in the region. The two appear to be inseparable. These are some pretty compelling trends indicating that the airport's foreign operations may be having a positive impact on the local labour market and workforce participation.
- It seems that I can predict the number of international flights in the future with some degree of accuracy based on historical data analysis. It's important to approach these projections cautiously even though our model's accuracy in forecasting future numbers of international flights based on historical data is encouraging. Flight patterns can be influenced by outside variables, unanticipated events, or market shifts. In order to ensure a comprehensive and flexible approach to our operations, even though the model offers insightful information, it is best to use these predictions as one of many considerations when making decisions.
- The data on unemployment rates show a clear seasonal pattern. In particular, June has a higher unemployment rate, whereas December has a lower rate. Despite the fact that the statistical test indicates that the difference is not statistically significant due to the limited number of data points, the visual interpretation suggests a consistent seasonal trend. While the statistical test may not identify these fluctuations as significant, the observed pattern may have important implications for our understanding of unemployment dynamics throughout the year.

- The holiday seasons, which fall in the same months as winter and summer, are marked by a discernible spike in unemployment. This pattern raises the possibility that seasonal variations and holiday-related periods of the year may be linked to greater unemployment rates. Understanding the seasonal variations in unemployment and the dynamics of the economy at different times of the year can be greatly aided by the knowledge of this data.

Discussion

Autocorrelation(25% in our case) in residuals can indicate that there are underlying patterns in the data that have not been fully captured or that the model may require additional refining. To guarantee the validity and precision of the model's predictions, autocorrelation must be taken into consideration.

Mean Absolute Error was the primary performance evaluation metric (MAE). Predictive accuracy was highest for the AR model, which had the lowest MAE of 147.32. However, the decision-making process extended beyond MAE. Interpretability, complexity, and robustness of the model were considered, recognizing that selecting a model requires considerable consideration.

As compared to both ARIMA and MA models, the AR model performed better in forecasting international flights. This implies that a greater role was played by the autoregressive component in identifying the underlying patterns in the dataset, which takes into account the link between recent and historical observations.

AR models assume that future results will only depend on past values. When working with complicated datasets affected by multiple factors beyond historical trends, care should be used. Furthermore, the necessity of careful data exploration and preprocessing is highlighted by the susceptibility of AR models to noise and outliers.

The ttest_ind (T-test for the means of two independent samples of scores) relies heavily on the magnitude so it is not infallible while dealing with something like the unemployment rate

The ADF test, which revealed non-stationarity in the international flights time series data, was the primary factor in the decision to perform a second differencing even though the KPSS test indicated stationarity following the first one. Because of the ADF test's reputation for being sensitive to trend stationarity, caution was exercised to guarantee a more trustworthy confirmation of stationarity. By addressing any lingering trend components and meeting the objectives of both tests, this second differencing attempted to strengthen the basis for next time series analysis and forecasting.
This method seeks to minimise over-manipulation that could introduce noise while still appropriately altering the data for effective modelling. Because different tests have different sensitivity to stationarity, it makes sure the data is ready for the next steps of the study.

Appendix A: Method

A comma-separated (.csv) file containing a dataset of economic indicators (tracked monthly between January 2013 and December 2019) was downloaded from "Analyse Boston".

A DataFrame df containing information on economic indicators, such as the Year, Month, number of passengers arriving and departing from Logan Airport (logan_intl_flights), and other pertinent metrics. A datetime index is added by manipulating the DataFrame.

The number of lag observations taken into account for modelling trends is indicated by the range of possible AutoRegressive (AR) orders, or p_values_ar..

best_aic_ar is a variable that keeps track of the lowest value of the Akaike Information Criterion (AIC), a metric used to pick models, during the AR order grid search.

best_order_ar is the ideal AR order that impacts the ARIMA model's lagged observation selection, ascertained by minimizing the AIC during grid search.

q_values_ma is a range of possible Moving Average (MA) orders that indicate how many forecast mistakes with lag are taken into account while simulating short-term changes.

best_aic_ma: A variable that keeps track of the lowest AIC value found during the MA order grid search, helping the ARIMA model choose the best lagged forecast errors.

best_order_ma: The ideal MA order found by reducing the AIC during grid search, which affects the ARIMA model's selection of lag-related forecast errors.

The ARIMA model's building blocks are the SARIMAX models for the AR and MA components, respectively, denoted as model_ar and model_ma.

results_ar, results_ma: The information on model parameters and statistical metrics obtained from fitting SARIMAX models to the time series data.

best_model_ar, best_model_ma: SARIMAX models fitted, according to grid search results, with the best AR and MA orders, respectively.

best_results_ar, best_results_ma: The information stored for forecasting and evaluation after fitting the best AR and MA models.

The best AR and MA models are used to anticipate and predict the means of international flights, which are then visualized as best_forecast_ar, best_forecast_mean_ar, best_forecast_ma, and best_forecast_mean_ma.

The ARIMA model's forecasts and projected means on the test data, which integrate the AR and MA components, are called arima_forecast_test and arima_forecast_mean_test.

residuals_ar, residuals_ma: Used in residual analysis, these residuals show the variations between actual and predicted values for the AR and MA models.

lags_ar: The maximum lag taken into account in the residuals of the AR model's Ljung-Box test for autocorrelations.

test_results_ar: The autocorrelations in the residuals of the AR model after the Ljung-Box test.

p_values_ar: P-values that show the significance of autocorrelations at various lags that were taken from

the Ljung-Box test findings.

The proportion of significant autocorrelations in the AR model residuals is expressed as percentage_autocorrelations_ar.

mae_arima: Mean Absolute Error (MAE) computed to assess how well the ARIMA model performs using the test set of data.

By using a Moving Average (MA) model, short-term fluctuations were explained and random walk behaviors were better understood. Moreover, underlying patterns were revealed through the disentanglement of trends, seasonality, and residuals made possible by the Seasonal-Trend decomposition using the LOESS (STL) approach.

The relationships between foreign flights and economic indicators were examined through correlation analyses, which provided insight into possible influencing factors.

T-tests for hypothesis testing examined changes in the unemployment rate over time, revealing statistically significant variations. By using ARIMA models for Time Series Forecasting, I was able to provide predictive insights by looking into the future. Forecasting accuracy was measured using evaluation metrics like Mean Squared Error (MSE) and Root Mean Squared Error (RMSE).

Appendix C has relevant Python code.

Appendix B: Results

The number of international flights at Logan International Airport is increasing over time, as evidenced by the positive annual growth rate. The growth rate is approximately 4.90

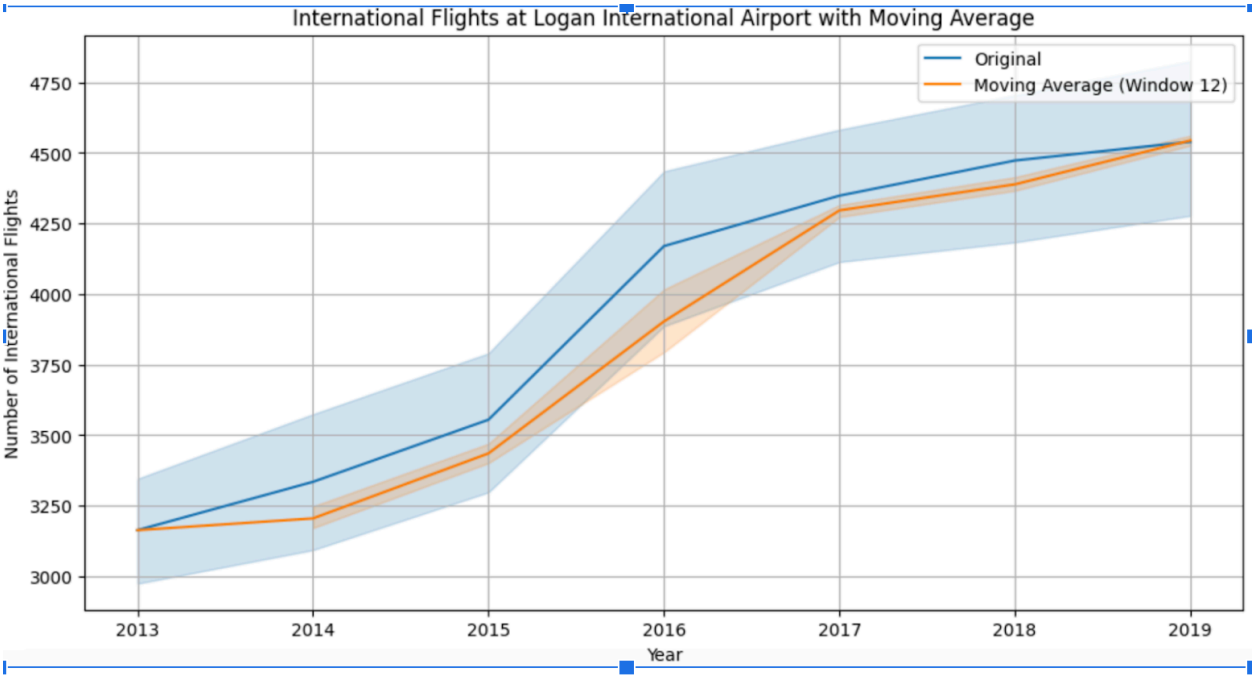

Figure 1: International Flight at Logan international airport with moving average

The Seasonal Breakdown of International Flights at Logan International Airport is shown below.

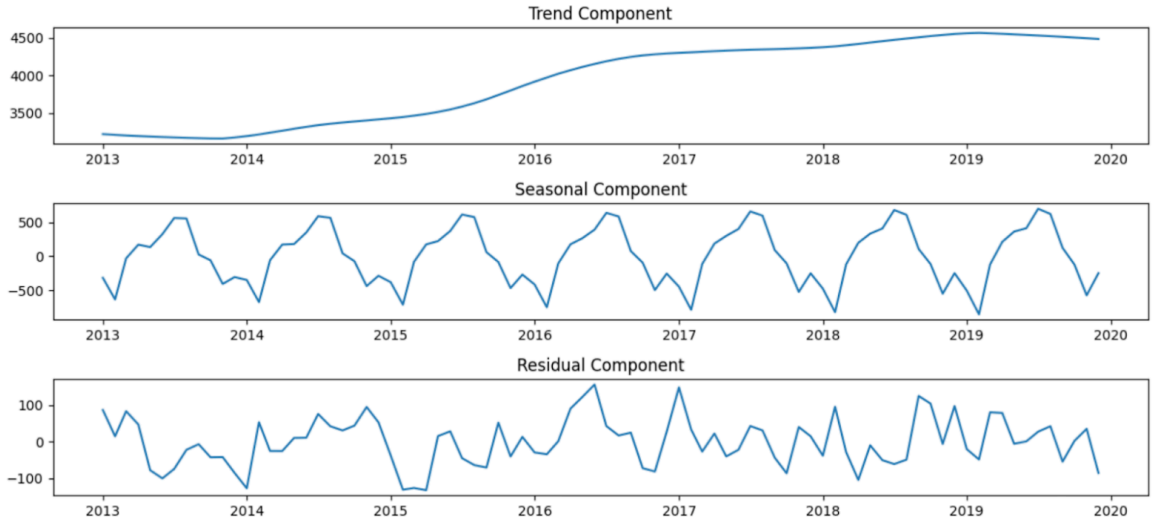

Figure 2: International Flight at Logan international airport with moving average

Patterns were discovered, with July and August consistently having higher international flight numbers, while February and November had lower numbers. I calculated the mean by aggregating seasonal values by month after STL decomposition. The plot of monthly mean values is shown below.

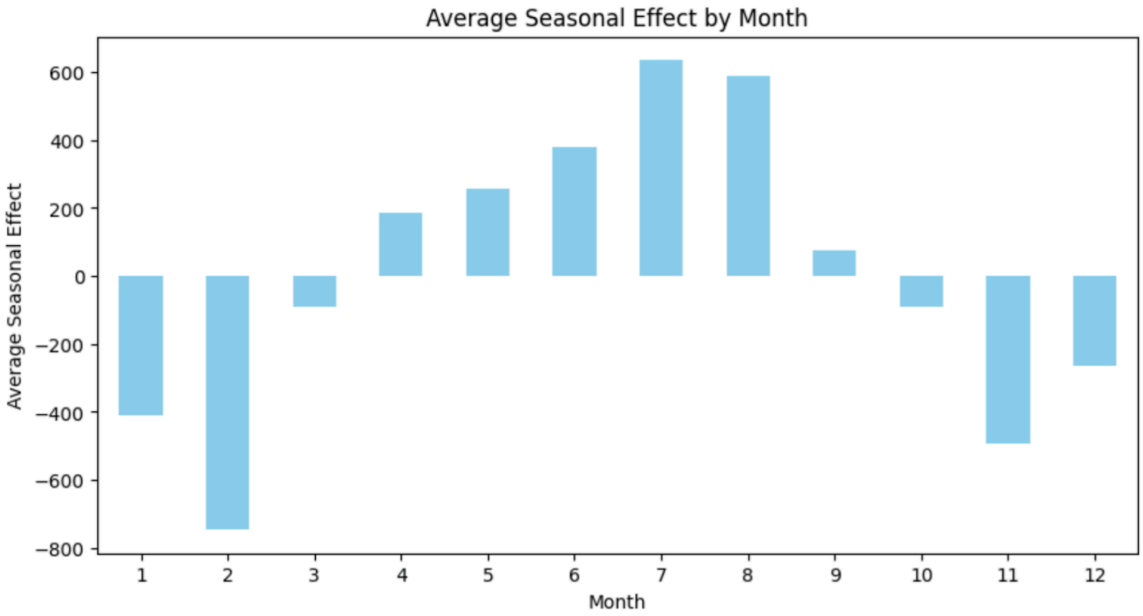

Figure 3: Average Seasonal Effect by Month

As can be seen, the number of international flights is usually highest in July and August and lowest in February and November.

The number of international flights correlates negatively with the unemployment rate and positively with the labor force participation rate with a Pearson correlation coefficient of -0.6239526755612207 and 0.7185056582429975 respectively.

The plot below depicts the time series data of the unemployment rate and the monthly average of unemployment rates after calculating the average unemployment rate for each month.

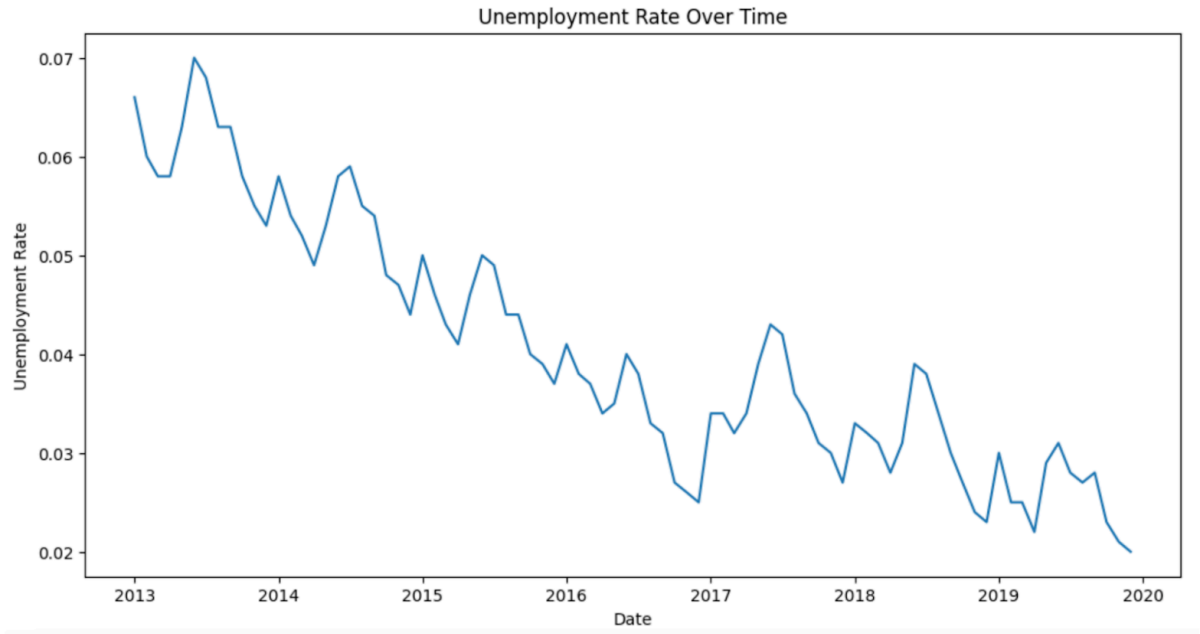

Figure 4: Unemployment rate over time

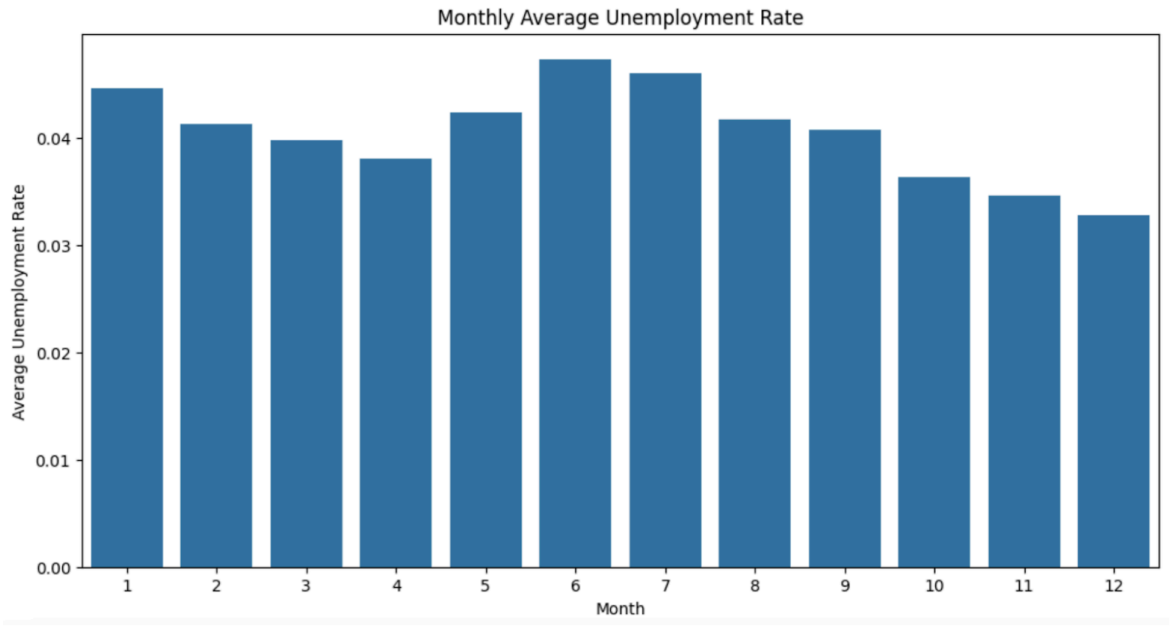

Figure 5: Monthly Average Unemployment Rate

The visualisations of second-order differenced data for Logan International flights and autocorrelation functions were created to aid in the selection of appropriate parameters for an ARIMA model, which is a critical step in time series forecasting.

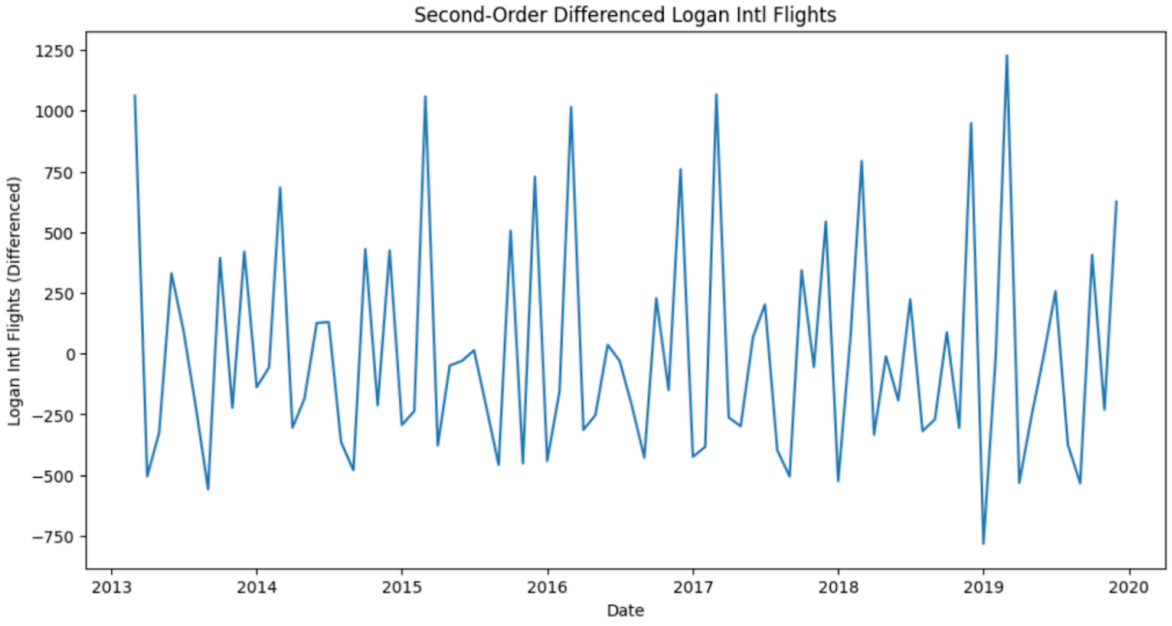

Figure 6: Second Order Differenced Logan International Flight

A grid search is used to find the best parameters for an ARIMA model's AutoRegressive (AR) and Moving Average (MA) components. The script iterates through all possible AR and MA orders, fitting SARIMAX models and calculating AIC scores. The best AR and MA component orders are determined by the lowest

AIC values. After fitting the chosen AR and MA models to the differenced data, their forecasts are plotted alongside the original differenced time series for visual comparison.

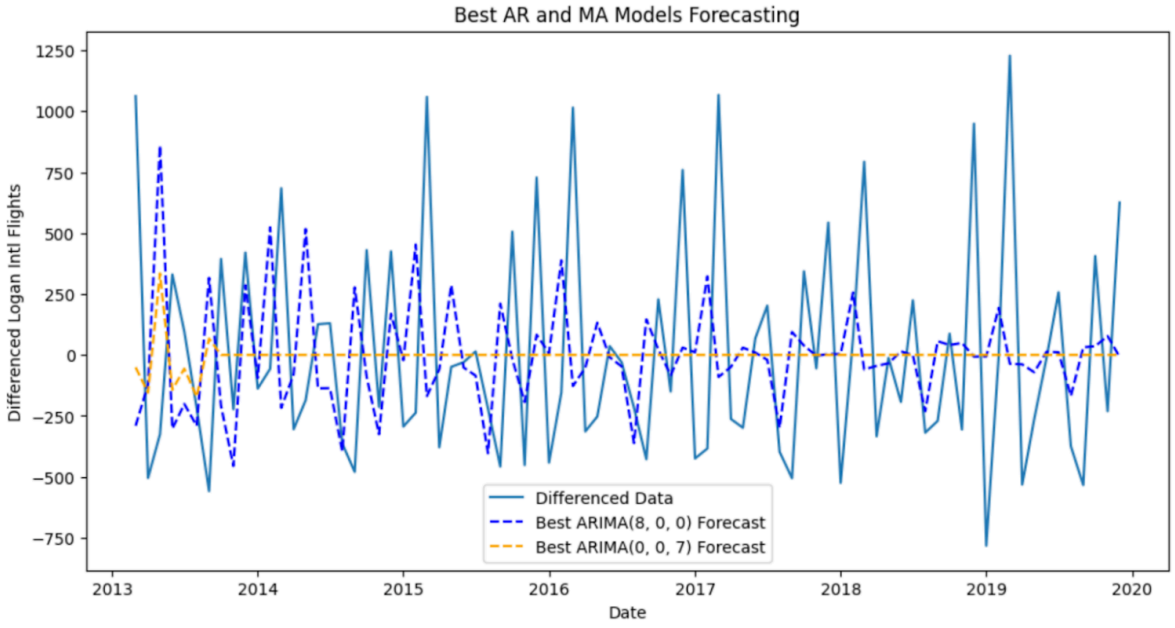

Figure 7: Best AR and MA models Forecasting

The code then implements a train-test split on time series data, using 90% of the data for training and 10% for testing. The training data is then fitted with the best-fitting AR and MA models from the previous grid search. Forecasts are generated for the test set, and the Mean Absolute Error (MAE) is calculated to assess prediction accuracy. The code visualises the models' performance by plotting the original training and testing data as well as the forecasted values. Both the AR and MA models have their own visualisations.

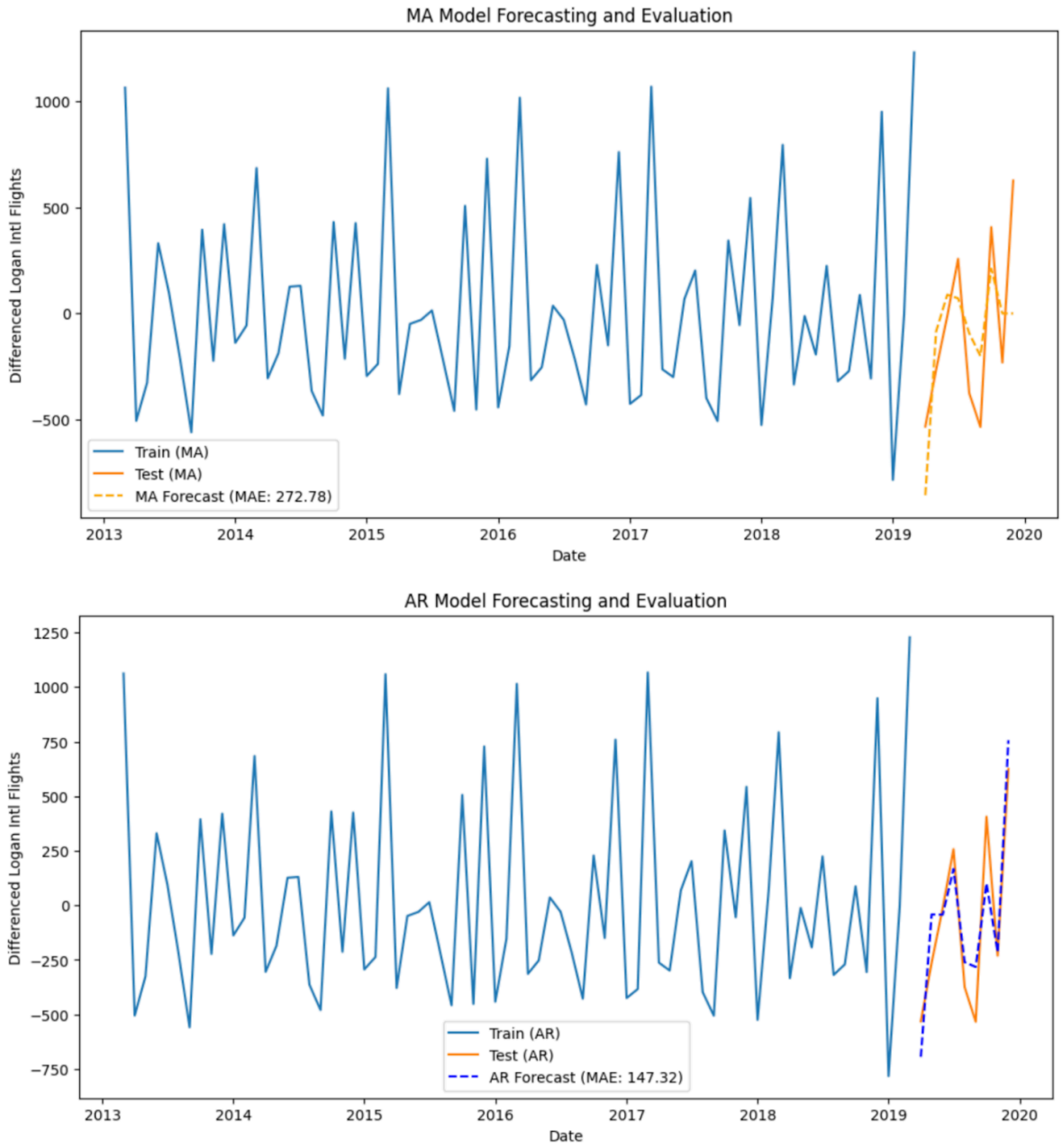

Figure 8: MA and AR Model Forecasting and Evaluation

The parameters for an ARIMA model are then formed by combining the best orders for the AutoRegressive (AR) and Moving Average (MA) components. The ARIMA model is then fitted to the training data, and forecasts for the test set are generated. The ARIMA model's predictions on the test data are evaluated using the Mean Absolute Error (MAE). The obtained MAE for the ARIMA Model is 175.8160293501777.

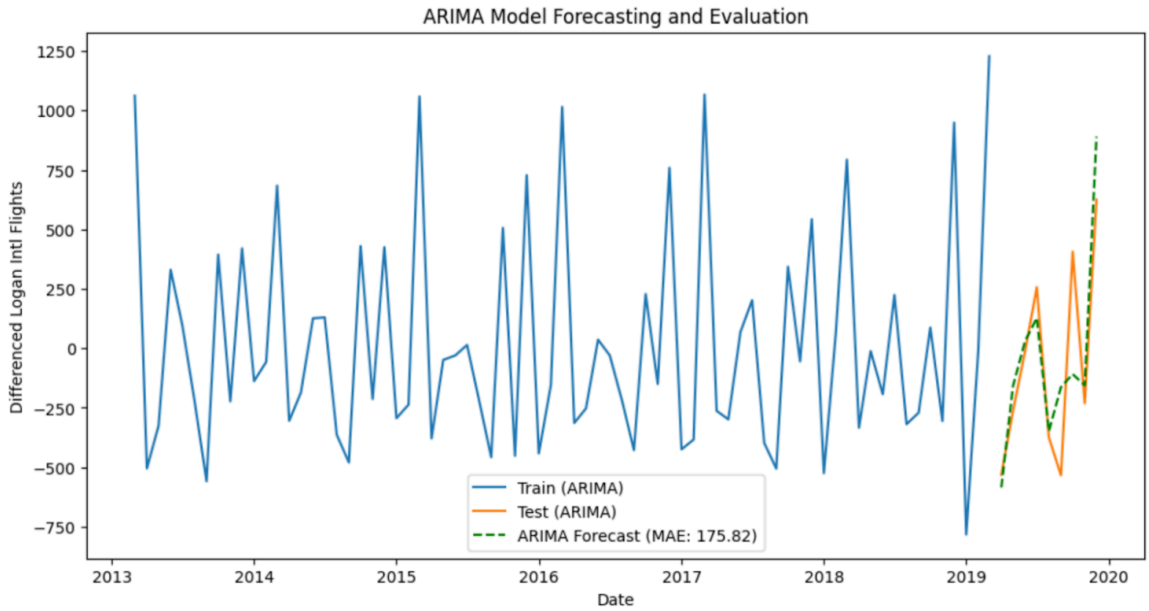

Figure 9: Arima Model Forecasting and Evaluation

The AutoRegressive (AR) and Moving Average (MA) models were then subjected to residual analysis. On the test data, residuals, or the differences between actual and forecasted values, are computed for each model. The residuals are then plotted over time, allowing for visual inspection of any patterns or trends. For unbiased residuals, the horizontal dashed line at zero serves as a reference. This analysis provides insights into the AR and MA models' performance and potential areas for improvement.

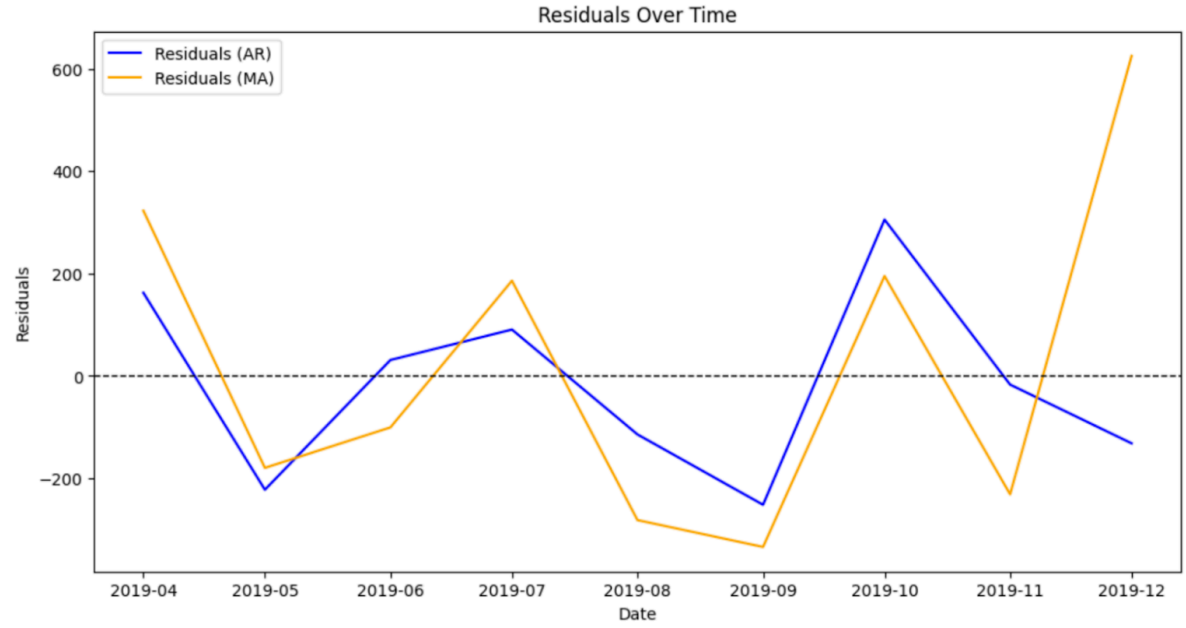

Figure 10: Residuals Over Time

The AutoRegressive (AR) model's residuals are then subjected to the Ljung-Box test. The test determines if the residuals at different lags show statistically significant autocorrelations. The test's p-values,

which show the likelihood of finding the specified residuals assuming no autocorrelation, are reported. Next, using a predetermined significance threshold, the algorithm determines the proportion of autocorrelations that are statistically significant. Lastly, a thorough examination of the autocorrelations in the residuals of the AR model is provided by printing each lag's individual p-values together with their significance level.

With p-values of 0.0353 and 0.0400, respectively, the Ljung-Box test results for the residuals of the AutoRegressive (AR) model show that the autocorrelations at lags 5 and 6 are statistically significant. This implies that the residuals at these delays exhibit a significant pattern or association. The autocorrelations at the remaining lags—1, 2, 3, 4, 7, and 8 show that they are not statistically significant, though, since the p-values for these lags are higher than the significance level of 0.05. It is determined that 25% of the residual autocorrelations are significant overall. According to the interpretation, autocorrelation is present in the residuals of the AR model, however it is most noticeable at lags 5 and 6.

```
P-values for Ljung-Box Test (AR): 1    0.171994
2    0.116802
3    0.105995
4    0.135048
5    0.035253
6    0.039975
7    0.059187
8    0.077761
Name: lb_pvalue, dtype: float64
Percentage of Autocorrelations in Residuals (AR): 25.00%
Lag 1: p-value = 0.1720, Not Significant
Lag 2: p-value = 0.1168, Not Significant
Lag 3: p-value = 0.1060, Not Significant
Lag 4: p-value = 0.1350, Not Significant
Lag 5: p-value = 0.0353, Significant
Lag 6: p-value = 0.0400, Significant
Lag 7: p-value = 0.0592, Not Significant
Lag 8: p-value = 0.0778, Not Significant
Total significant autocorrelations: 2
```

Appendix C: Code

[1]:
```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df = pd.read_csv('economic-indicators.csv')

df['Date'] = pd.to_datetime(df[['Year', 'Month']].assign(DAY=1))
df.set_index('Date', inplace=True)

plt.figure(figsize=(12, 6))
sns.lineplot(x='Year', y='logan_intl_flights', data=df)
plt.title('International Flights at Logan International Airport Over Time')
plt.xlabel('Year')
plt.ylabel('Number of International Flights')
plt.grid(True)
plt.show()
```

[3]:
```python
#Moving Average (Window 12)
window_size = 12
df['MA'] = df['logan_intl_flights'].rolling(window=window_size).mean()

plt.figure(figsize=(12, 6))
sns.lineplot(x='Year', y='logan_intl_flights', data=df, label='Original')
sns.lineplot(x='Year', y='MA', data=df, label=f'Moving Average (Window {window_size})')
plt.title('International Flights at Logan International Airport with Moving Average')
plt.xlabel('Year')
plt.ylabel('Number of International Flights')
plt.legend()
plt.grid(True)
plt.show()
```

[5]:
```python
#STL Decomposition
from statsmodels.tsa.seasonal import STL

# Perform STL decomposition
seasonal_period = 13  # Assuming seasonality of 12 months
stl = STL(df['logan_intl_flights'], seasonal=seasonal_period)
result = stl.fit()
# Plotting the decomposition components
plt.figure(figsize=(12, 8))

plt.subplot(4, 1, 1)
plt.plot(df.index, result.trend)
plt.title('Trend Component')

plt.subplot(4, 1, 2)
plt.plot(df.index, result.seasonal)
plt.title('Seasonal Component')

plt.subplot(4, 1, 3)
plt.plot(df.index, result.resid)
plt.title('Residual Component')

plt.subplot(4, 1, 4)
plt.plot(df.index, df['logan_intl_flights'], label='Original')
plt.title('Original Time Series')

plt.suptitle('Seasonal Decomposition of International Flights at Logan International Airport')
plt.tight_layout(rect=[0, 0, 1, 0.96])
plt.show()
```

Seasonal Decomposition of International Flights at Logan International Airport

[6]:
```
# Extracting seasonal component from the STL result
seasonal_component = result.seasonal

# Create a differen dataframe with the seasonal component and the dates
seasonal_df = pd.DataFrame({'Date': result.trend.index, 'Seasonal_Component': seasonal_component.values})
seasonal_df.set_index('Date', inplace=True)

seasonal_df['Month'] = seasonal_df.index.month

# Aggregate seasonal values by month and calculate the mean
monthly_seasonal_mean = seasonal_df.groupby('Month')['Seasonal_Component'].mean()

plt.figure(figsize=(12, 6))
plt.plot(seasonal_df.index, seasonal_df['Seasonal_Component'], label='Seasonal Component')
plt.title('Seasonal Patterns by Month')
plt.xlabel('Date')
plt.ylabel('Seasonal Component')
plt.legend()
plt.show()

# Monthly mean values plot
plt.figure(figsize=(10, 5))
monthly_seasonal_mean.plot(kind='bar', color='skyblue')
plt.title('Average Seasonal Effect by Month')
plt.xlabel('Month')
plt.ylabel('Average Seasonal Effect')
plt.xticks(rotation=0)
plt.show()
```

[7]:
```
#Annual growth rate

initial_value = df.loc[df.index.min(), 'logan_intl_flights']
final_value = df.loc[df.index.max(), 'logan_intl_flights']

# number of years
num_years = (df.index.max() - df.index.min()).days / 365.25

annual_growth_rate = (final_value / initial_value) ** (1 / num_years) - 1

print(f"Annual Growth Rate: {annual_growth_rate * 100:.2f}%")
```

```
Annual Growth Rate: 4.90%
```

[8]:
```
# Correlation(Pearson) between Number of Flights and other relevant economic indicators

correlation = df['logan_intl_flights'].corr(df['labor_force_part_rate'])
correlation1 = df['logan_intl_flights'].corr(df['unemp_rate'])
print(f"Correlation between International Flights and Labor Force Participation Rate: {correlation}")
print(f"Correlation between International Flights and Unemployment Rate: {correlation1}")
```

```
Correlation between International Flights and Labor Force Participation Rate: 0.7185056582429975
Correlation between International Flights and Unemployment Rate: -0.6239526755612207
```

[9]:
```
# Unemployment Rate time series
plt.figure(figsize=(12, 6))
plt.plot(df['unemp_rate'])
plt.title('Unemployment Rate Over Time')
plt.xlabel('Date')
plt.ylabel('Unemployment Rate')
plt.show()

# STL decomposition
stl = STL(df['unemp_rate'], seasonal=13)  # Assuming a seasonal period of 12 months
result = stl.fit()

# Plot the decomposition components
fig = result.plot()
plt.show()
```

[ ]:
```
# average unemployment rate for each month
monthly_avg_unemp = df.groupby('Month')['unemp_rate'].mean().sort_values()

# Visualize the monthly average unemployment rates
plt.figure(figsize=(12, 6))
sns.barplot(x=monthly_avg_unemp.index, y=monthly_avg_unemp.values)
plt.title('Monthly Average Unemployment Rate')
plt.xlabel('Month')
plt.ylabel('Average Unemployment Rate')
plt.show()
```

[12]:
```
# Test for significance. Note that ttest_ind relies heavily on the magnitude so it is not infallible while dealing with something like unemplo
from scipy.stats import ttest_ind

# Extract data for June and December
unemployment_june = df[df.index.month == 6]['unemp_rate']
unemployment_december = df[df.index.month == 12]['unemp_rate']

# t-test
t_stat, p_value = ttest_ind(unemployment_june, unemployment_december)
print('t-test value', t_stat)
print('p value', p_value)

if p_value < 0.05:
    print('The difference in unemployment rates between June and December is statistically significant.')
else:
    print('There is no significant difference in unemployment rates between June and December.')
```

```
t-test value 2.1391280094846437
p value 0.053674458485613565
There is no significant difference in unemployment rates between June and December.
```

[22]:
```
#Model Creation

#AR model
df = pd.read_csv('economic-indicators.csv')
df['Date'] = pd.to_datetime(df[['Year', 'Month']].assign(DAY=1))
df.set_index('Date', inplace=True)
df.index.freq = 'MS'
# Apply second-order differencing
df['logan_intl_flights_diff2'] = df['logan_intl_flights'].diff().diff()
print(df)
# Drop NaN values generated by differencing
df = df.dropna()

# Perform Augmented Dickey-Fuller test
result_adfuller = adfuller(df['logan_intl_flights_diff2'])

# Print the results
print("ADF Statistic:", result_adfuller[0])
print("p-value:", result_adfuller[1])
print("Critical Values:", result_adfuller[4])

# Interpret the results
if result_adfuller[1] <= 0.05:
    print("Reject the null hypothesis. The series is likely stationary.")
else:
    print("Fail to reject the null hypothesis. The series may still be non-stationary.")

# Visualize the differenced time series
plt.figure(figsize=(12, 6))
plt.plot(df['logan_intl_flights_diff2'])
plt.title('Second-Order Differenced Logan Intl Flights')
plt.xlabel('Date')
plt.ylabel('Logan Intl Flights (Differenced)')
plt.show()
```

```
# Visualize the differenced time series
plt.figure(figsize=(12, 6))
plt.plot(df['logan_intl_flights_diff2'])
plt.title('Second-Order Differenced Logan Intl Flights')
plt.xlabel('Date')
plt.ylabel('Logan Intl Flights (Differenced)')
plt.show()

# ACF and PACF plots to determine model orders
lag_acf = acf(df['logan_intl_flights_diff2'], nlags=20)
lag_pacf = pacf(df['logan_intl_flights_diff2'], nlags=20, method='ols')

# Plot ACF
plt.figure(figsize=(4, 4))
plot_acf(df['logan_intl_flights_diff2'], lags=20)
plt.title('Autocorrelation Function (ACF) - Differenced')
plt.show()

# Plot PACF
plt.figure(figsize=(4, 4))
plot_pacf(df['logan_intl_flights_diff2'], lags=20)
plt.title('Partial Autocorrelation Function (PACF) - Differenced')
plt.show()
```

[23]:
```
ar_order = 1
ar_model = SARIMAX(df['logan_intl_flights_diff2'], order=(ar_order, 0, 0))
ar_results = ar_model.fit(disp=False)

# Forecasting on the original data
ar_forecast = ar_results.get_forecast(steps=len(df))
ar_forecast_mean = ar_forecast.predicted_mean

# Plotting AR Model
plt.figure(figsize=(12, 6))
plt.plot(df.index, df['logan_intl_flights_diff2'], label='Differenced Data')
plt.plot(df.index, ar_forecast_mean, label='AR Forecast', linestyle='dashed', color='red')
plt.title(f'AR({ar_order}) Model Forecasting')
plt.xlabel('Date')
plt.ylabel('Differenced Logan Intl Flights')
plt.legend()
plt.show()

# MA Model
ma_order = 1
ma_model = SARIMAX(df['logan_intl_flights_diff2'], order=(0, 0, ma_order))
ma_results = ma_model.fit(disp=False)

# Forecasting on the original data
ma_forecast = ma_results.get_forecast(steps=len(df))
ma_forecast_mean = ma_forecast.predicted_mean

# Plotting MA Model
plt.figure(figsize=(12, 6))
plt.plot(df.index, df['logan_intl_flights_diff2'], label='Differenced Data')
plt.plot(df.index, ma_forecast_mean, label='MA Forecast', linestyle='dashed', color='green')
plt.title(f'MA({ma_order}) Model Forecasting')
plt.xlabel('Date')
plt.ylabel('Differenced Logan Intl Flights')
plt.legend()
plt.show()
```

[26]:
```
import itertools

# Grid search for AR orders
p_values_ar = range(0, 10)  # AR orders
best_aic_ar = float('inf')
best_order_ar = None

for p in p_values_ar:
    model_ar = SARIMAX(df['logan_intl_flights_diff2'], order=(p, 0, 0))
    results_ar = model_ar.fit(disp=False)
    aic_ar = results_ar.aic
    if aic_ar < best_aic_ar:
        best_aic_ar = aic_ar
        best_order_ar = (p, 0, 0)

# Grid search for MA orders
q_values_ma = range(0, 10)  # MA orders
best_aic_ma = float('inf')
best_order_ma = None

for q in q_values_ma:
    model_ma = SARIMAX(df['logan_intl_flights_diff2'], order=(0, 0, q))
    results_ma = model_ma.fit(disp=False)
    aic_ma = results_ma.aic
    if aic_ma < best_aic_ma:
        best_aic_ma = aic_ma
        best_order_ma = (0, 0, q)

print("best order MA", best_order_ma)
print("best order AR", best_order_ar)
# Fit the best AR and MA models
best_model_ar = SARIMAX(df['logan_intl_flights_diff2'], order=best_order_ar)
best_results_ar = best_model_ar.fit(disp=False)

best_model_ma = SARIMAX(df['logan_intl_flights_diff2'], order=best_order_ma)
best_results_ma = best_model_ma.fit(disp=False)

# Forecasting on the original data
best_forecast_ar = best_results_ar.get_forecast(steps=len(df))
best_forecast_mean_ar = best_forecast_ar.predicted_mean

best_forecast_ma = best_results_ma.get_forecast(steps=len(df))
```

[27]:
```
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error

# Train-test split (90-10%)
train_size = int(len(df) * 0.9)
train_ar, test_ar = df.iloc[:train_size], df.iloc[train_size:]
train_ma, test_ma = df.iloc[:train_size], df.iloc[train_size:]
end_index_forecast_ar = test_ar.index[-1]
# Fit the best AR and MA models on the training data
best_model_ar = SARIMAX(train_ar['logan_intl_flights_diff2'], order=best_order_ar)
best_results_ar = best_model_ar.fit(disp=False)

best_model_ma = SARIMAX(train_ma['logan_intl_flights_diff2'], order=best_order_ma)
best_results_ma = best_model_ma.fit(disp=False)

# Forecasting on the test set
forecast_ar = best_results_ar.get_forecast(steps=len(test_ar))
forecast_mean_ar = forecast_ar.predicted_mean
print(train_ma['logan_intl_flights_diff2'])
forecast_ma = best_results_ma.get_forecast(steps=len(test_ma))
forecast_mean_ma = forecast_ma.predicted_mean

# Calculate MAE for evaluation
mae_ar = mean_absolute_error(test_ar['logan_intl_flights_diff2'], forecast_mean_ar)
mae_ma = mean_absolute_error(test_ma['logan_intl_flights_diff2'], forecast_mean_ma)

# Plotting the predictions
plt.figure(figsize=(12, 6))
plt.plot(train_ar.index, train_ar['logan_intl_flights_diff2'], label='Train (AR)')
plt.plot(test_ar.index, test_ar['logan_intl_flights_diff2'], label='Test (AR)')
plt.plot(test_ar.index, forecast_mean_ar, label=f'AR Forecast (MAE: {mae_ar:.2f})', linestyle='dashed', color='blue')

plt.title('AR Model Forecasting and Evaluation')
plt.xlabel('Date')
plt.ylabel('Differenced Logan Intl Flights')
plt.legend()
plt.show()

# Plotting the predictions for MA model
plt.figure(figsize=(12, 6))
plt.plot(train_ma.index, train_ma['logan_intl_flights_diff2'], label='Train (MA)')
plt.plot(test_ma.index, test_ma['logan_intl_flights_diff2'], label='Test (MA)')
plt.plot(test_ma.index, forecast_mean_ma, label=f'MA Forecast (MAE: {mae_ma:.2f})', linestyle='dashed', color='orange')

plt.title('MA Model Forecasting and Evaluation')
plt.xlabel('Date')
plt.ylabel('Differenced Logan Intl Flights')
plt.legend()
plt.show()
```

[28]:
```
# Combine AR and MA orders for ARIMA model
best_order_arima = (best_order_ar[0], 0, best_order_ma[2])

# Fit the best ARIMA model on the training data
best_model_arima_train = SARIMAX(train_ar['logan_intl_flights_diff2'], order=best_order_arima)
best_results_arima_train = best_model_arima_train.fit(disp=False)

# Forecast on the test data
arima_forecast_test = best_results_arima_train.get_forecast(steps=len(test_ar))
arima_forecast_mean_test = arima_forecast_test.predicted_mean

# Evaluate ARIMA model
mae_arima = mean_absolute_error(test_ar['logan_intl_flights_diff2'], arima_forecast_mean_test)
print(f'MAE for ARIMA Model: {mae_arima}')

# Plotting the predictions for ARIMA model
plt.figure(figsize=(12, 6))
plt.plot(train_ar.index, train_ar['logan_intl_flights_diff2'], label='Train (ARIMA)')
plt.plot(test_ar.index, test_ar['logan_intl_flights_diff2'], label='Test (ARIMA)')
plt.plot(test_ar.index, arima_forecast_mean_test, label=f'ARIMA Forecast (MAE: {mae_arima:.2f})', linestyle='dashed', color='green')

plt.title('ARIMA Model Forecasting and Evaluation')
plt.xlabel('Date')
plt.ylabel('Differenced Logan Intl Flights')
plt.legend()
plt.show()
```

[29]:
```
#Residual Analysis

# Residuals for AR model
residuals_ar = test_ar['logan_intl_flights_diff2'] - forecast_mean_ar

# Residuals for MA model
residuals_ma = test_ma['logan_intl_flights_diff2'] - forecast_mean_ma

# Plot residuals over time
plt.figure(figsize=(12, 6))
plt.plot(test_ar.index, residuals_ar, label='Residuals (AR)', color='blue')
plt.plot(test_ma.index, residuals_ma, label='Residuals (MA)', color='orange')
plt.axhline(0, color='black', linestyle='--', linewidth=1)
plt.title('Residuals Over Time')
plt.xlabel('Date')
plt.ylabel('Residuals')
plt.legend()
plt.show()
```

[30]:
```
# Plot ACF of residuals with confidence intervals for AR model
plt.figure(figsize=(12, 6))
plot_acf(residuals_ar, lags=3, title='ACF of Residuals (AR)', alpha=0.05)
plt.xlabel('Lags')
plt.ylabel('Autocorrelation')
plt.show()

# Plot ACF of residuals with confidence intervals for MA model
plt.figure(figsize=(12, 6))
plot_acf(residuals_ma, lags=3, title='ACF of Residuals (MA)', alpha=0.05)
plt.xlabel('Lags')
plt.ylabel('Autocorrelation')
plt.show()
```

[32]:
```
from scipy import stats
# Histogram and Q-Q Plot of residuals for AR model
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.hist(residuals_ar, bins=20, color='blue', alpha=0.7, edgecolor='black')
plt.title('Histogram of Residuals (AR)')
plt.xlabel('Residuals')
plt.ylabel('Frequency')

plt.subplot(1, 2, 2)
stats.probplot(residuals_ar, dist='norm', plot=plt)
plt.title('Q-Q Plot of Residuals (AR)')

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