BOSTON'S ECONOMIC DIARY

Tracing the Contours of Growth and Challenge: A Detailed Account from 2013-2019

Addressed To:
Boston Planning and Development Authority (BPDA)
BPDAmarketing@boston.gov
Boston, MA

Citywide Analytics Team
City of Boston
analyticsteam@boston.gov
Phone: 617-635-4783
Boston, MA

Co-Authored By:
Shubham Singh (02131135)
Sai Gopal Jarabana (02073973)
Piyush Arvind Kolte (02085765)

ISSUES

This is the report to analyze the **Economy Indicators** dataset from year 2013 to 2019 as per the **Analyze Boston** data hub which they have posted on their website on April 5, 2022. The dataset contains 19 variables and that too in different sectors of economic growth of the Boston city. Some of the topics are Boston Logan International Flights and Passengers, Hotel Occupancy and its Average Daily Rate, Pipeline Total Development Cost as per the square fit, Medical Housing Price and Housing Sales Volume, etc. These sectors are evidently dedicated to the economic challenges of the city.

After hovering over the variables and the data points given in their particular fields, we came to the decision that we worked on the topics Logan International Flights and Passengers and Hotel Average Daily Rate which leads to the correlation in between the variables and the outcomes related to the forecasting in between them.

According to The Analyze Boston website, the goal of this project is to examine the challenges faced by Logan Airport and hotel occupancy in Boston city:

- Was the data given as per the passengers who were departed and arrived at the Boston Logan International Airport were correlated with the Logan International Flights?
- Analyzing the facts about the International Flights from the Boston Airport in between 7 years as per the time series, so what was the frequency and the behavior of the flights as per the passengers travelled in the given tenure and how it affects to the economic sector?
- Was there any predictive forecast correlation in between the Logan Passengers and Logan International flights in the next few months and how it going to affect the economic areas of the city?
- The primary issue we're addressing through these analyses is understanding the complex interplay between various sectors of Boston's economy. Specifically, we're looking at how factors like airport traffic (Logan Passengers and International Flights) and the hotel industry (Average Daily Rates) correlate with and potentially influence the job market and broader economic trends in the city.

FINDINGS

- According to the economic data of Analyze Boston, the data points which were given as per the Logan International flights were very uneven, scattered, and non-stationary. Our analysis is defined on the basis of stationary data which we made later while doing the analysis and got the context that the data is already separated which gives the information about the correlation in between the logan passengers travelled to and from Logan International Flights and on visualizing them on the graph it shows that behavior of both the variables are commonly distributed which leads to the economic growth in the Boston city and at the airport and that too affects positively in the employment.

- Considering the data for the prediction of the logan passengers and logan international flights, we train and test the models and got the outcome that passengers travelling from and to Boston International Airport are going to increase which is relatively leads to the international flights also. The predicted data we gathered from the time series analysis is for the next 3 months. We can analyze the forecasting as per our analysis required, but the botom line is that it is just the prediction and the accuracy should match to the result, that’s why we have forecasted for the next 3 months only.

- Our analysis uncovered a significant correlation between airport traffic (including both passengers and international flights) and key economic indicators such as job availability. This relationship suggests that increased airport activities are likely to have a positive impact on Boston's job market. Additionally, the hotel industry's performance, reflected in rising rates and occupancy, appears to be interconnected with broader economic health, potentially influencing employment opportunities.

- Utilizing a neural network model to predict Boston's total job numbers based on various economic indicators yielded accurate results. The model effectively captured and explained a significant portion of the fluctuations in the job market. This accuracy indicates that the variables incorporated into the model, like airport traffic and hotel industry metrics, are reliable predictors of employment trends in the city.

DISCUSSION

- In the course of our analysis into Logan International Flights and Logan Passengers, a significant pattern in the data became apparent. This growth may have wider ramifications for linked sectors in addition to reflecting the rise in travel demand. The increased level of economic activity at the airport may have an impact on things like hotel average daily rates (ADR) and job prospects both at the airport and in the greater Boston area. Greater employment possibilities and economic prosperity are frequently correlated with an increasing airport ecosystem. The increasing trend in employment generation highlights the diverse effects of Logan International Airport on the regional economy, establishing it as a crucial element of Boston's economic environment.

- At first, the data points we analyzed from the Boston Airport's international flights were irregular and non-stationary. But we were able to attain stationarity by carefully preprocessing and differencing. The correlation research then revealed a clear connection between the travel habits of Logan passengers and Logan International Flights, indicating a reciprocal effect. The behavior of both variables showed a common distribution when this association was shown on a graph. The mutually beneficial link between flight frequencies and passengers is a sign of a healthy economy. This beneficial impact also extends to jobs, underscoring the airport's function as a spur for local economic growth.

- Predictive models indicate a positive association between Logan Passengers and Logan International flights in our investigation. According to the estimate, there will likely be more passengers traveling to and from Boston foreign Airport, which will coincide with an increase in foreign flights. It's important to remember that predictions are inherently unpredictable and that their accuracy might change. Our prediction is only valid for the upcoming three months out of caution. Although this observation suggests possible economic effects on the city, the precision of these projections depends on dynamic variables and outside influences. The data show the forecasting for the upcoming year and with the help of that we can assume the future prediction of the airport that the volume of flights will running in future.

- Our study reveals a direct link between the number of people flying into Boston and job availability. Simply put, more flights coming into the city tend to boost the local economy, leading to more job openings. Using advanced modeling techniques akin to a smart assistant, we predicted future job trends in Boston. The accuracy of these predictions confirms that factors like airport traffic and hotel industry rates are effective in forecasting job availability in the city.

Appendix A: Method

Data Collection:
Our analysis leverages a meticulously curated dataset of economic indicators reflecting the vibrancy and challenges of Boston's economy from 2013 to 2019. This dataset, drawn from reliable and authoritative sources, encompasses the following sectors:
Tourism: Tracked by 'logan_passengers' and 'logan_intl_flights', which count the passengers and international flights at Boston Logan International Airport, respectively.
Hotel Market: Represented by 'hotel_occup_rate' and 'hotel_avg_daily_rate', measuring Boston's hotel occupancy and average daily rates.
Labour Market: Evaluated through 'total_jobs', 'unemp_rate', and 'labor_force_part_rate', indicating the number of jobs, the unemployment rate, and the labor force participation rate in Boston.
Real Estate Market: Assessed using 'med_housing_price' and 'housing_sales_vol', indicating the median housing sales price and the volume of housing sales.

Variable Creation:
The analysis hinged on meticulously selected key variables to capture the diverse economic landscape of Boston City:
Temporal Data: Year and month were critical for the time series analysis, allowing for the tracking of economic trends over time.
Tourism Metrics: 'logan_passengers' and 'logan_intl_flights' recorded the flow of domestic and international travellers through Logan Airport, essential for gauging tourism activity.
Hotel Market Indicators: 'hotel_occup_rate' and 'hotel_avg_daily_rate' provided insights into the health of Boston's hotel sector, indicating occupancy rates and average revenue per available room.
Labour Market Dynamics: Variables like 'total_jobs', 'unemp_rate', and 'labor_force_part_rate' reflected the employment situation in the city, from job availability to workforce engagement.
Housing Market Figures: 'med_housing_price' and 'housing_sales_vol' were pivotal in assessing the real estate market, tracking median sales prices and the number of transactions.
These variables were thoughtfully chosen to ensure a holistic view of the city's economic status, mirroring the demographic, geographic, and situational factors typically involved in a comprehensive socio-economic study. Each variable was measured in units appropriate to its nature—people for passengers, flights for air traffic, percentage for rates, and dollars for financial figures—ensuring precision and clarity in the analysis.

Analytic Methods:
To dissect and interpret these variables, we employed a comprehensive suite of analytic methods:

- **Summary Statistics:** To establish a baseline understanding of each variable's distribution, central tendency, and variability.
- **Correlation Analysis:** To examine the relationships between economic sectors, determining how variables such as tourism and labour statistics correlate with housing market dynamics.
- **Time Series Decomposition:** To separate trend, seasonal, and irregular components within our temporal data, providing clarity on underlying patterns.
- **Stationarity Tests:** Including both Augmented Dickey-Fuller and KPSS tests to confirm the time series data's suitability for modelling.
- **Differencing:** Applied to transform non-stationary time series data into a stationary series, a prerequisite for many forecasting models.
- **ACF and PACF Analysis**: To identify the appropriate autoregressive and moving average components for ARIMA modelling.

- **ARIMA Modelling:** Utilized for its robustness in forecasting economic indicators, informed by our earlier ACF and PACF analysis.

These analyses were carried out using Python and its rich ecosystem of libraries. We utilized pandas for data manipulation and handling, matplotlib and seaborn for visualization to elucidate trends and relationships, and statsmodels for time series analysis. This methodological approach, leveraging the robust capabilities of Python's scientific stack, was meticulously designed to afford a nuanced, multi-dimensional understanding of Boston's economic indicators.

Appendix B: Result

We initiated our data analysis with the computation of summary statistics for key economic indicators. This initial step was instrumental in understanding the data's distribution and setting the stage for deeper analysis. Summary statistics provided us with measures of central tendency, variability, and distribution shape for variables such as 'total_jobs', 'unemp_rate', 'logan_passengers', 'logan_intl_flights', and 'med_housing_price'.

```
Date Range: 2013-2019 with monthly data.
Logan Passengers: Average ~3,015,647, ranging from ~1,878,731 to ~4,120,937.
International Flights: Average ~3,941, ranging from 2,587 to 5,260.
Hotel Occupancy Rate: Average 81.77%, min 57.2%, max 93.1%.
Hotel Average Daily Rate: Average $244.42, ranging from $157.89 to $337.92.
Total Jobs: Average ~356K, ranging from ~323K to ~393K.
Unemployment Rate: Average 4.04%, ranging from 2% to 7%.
Median Housing Price: High variability with a standard deviation of ~$221,099.
```

Fig: Summary Statistics of Key Economic Indicators

Through these metrics, we identified the average levels of employment, tourism activity, and housing market health, as well as their range and fluctuations over the studied period. The precise figures obtained will be presented in a tabular form for clear and concise reference.
To decipher the relationships between the labor market, tourism, and housing sector, we constructed a correlation matrix. This matrix was a key tool in identifying and quantifying the strength and direction of associations among the variables.

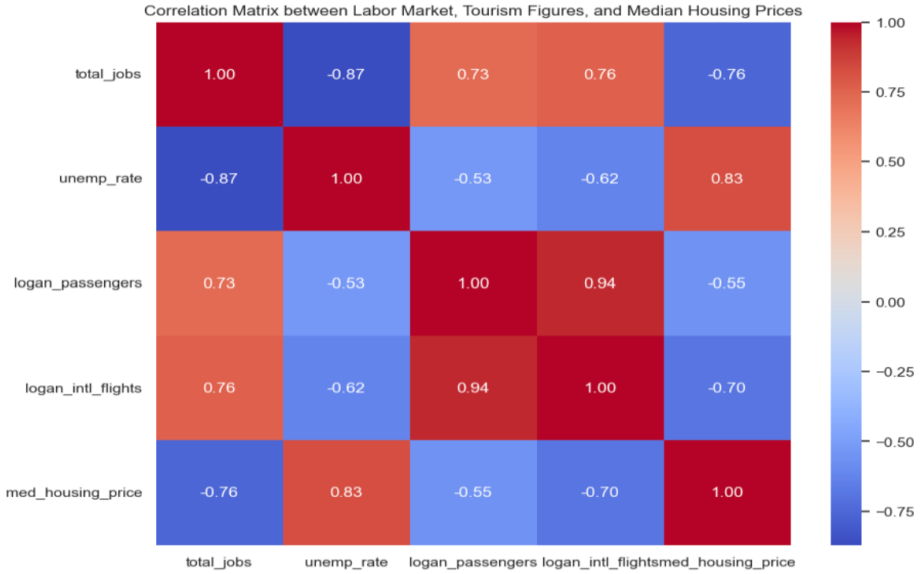


Fig: Correlation Matrix of Boston Economic Indicators

A strong negative correlation between 'total_jobs' and 'unemp_rate' highlighted the inverse relationship typically observed between employment and unemployment levels.
Positive correlations of 'logan_passengers' and 'logan_intl_flights' with 'total_jobs' suggested a link between tourism activity and job availability.
An intriguing negative correlation was observed between 'total_jobs' and 'med_housing_price', prompting further investigation into the underlying causes.
The housing market's median price showed a positive correlation with the unemployment rate, a counterintuitive finding that warrants a deeper dive into the economic factors at play.

```
Neural Network MSE: 0.27982413926835253, R2: 0.7123094084489661
```

The neural network's adeptness in modeling these complex relationships was evident in its performance metrics – achieving an MSE of about 0.2798 and an R2 score of 0.7123, effectively explaining over 70% of the variance in 'total_jobs'. This not only highlights the model's accuracy but also its proficiency in deciphering the intricate web of factors that influence employment trends in an urban environment.

The decomposition of the components (trends, seasonal, and residual) is done as per the logan passengers and international flights. The graph for both the variables showing that the passengers are consistently raising as the time goes away and this affects to the international flights which is also uniformly moving from 2013 to 2019. The curves for both the graphs are looking consistently same by which we can assume that their correlation in between these variables are strongly connected which can also affect the economic sector in other fields in the city.
The decomposition components as trends, seasonal, and residual captured the variations that repeat over a fixed periods, such as yearly cycle. By analyzing of the each components independently, it states for the further analysis and the ready for the understandings of the time series behaviour.

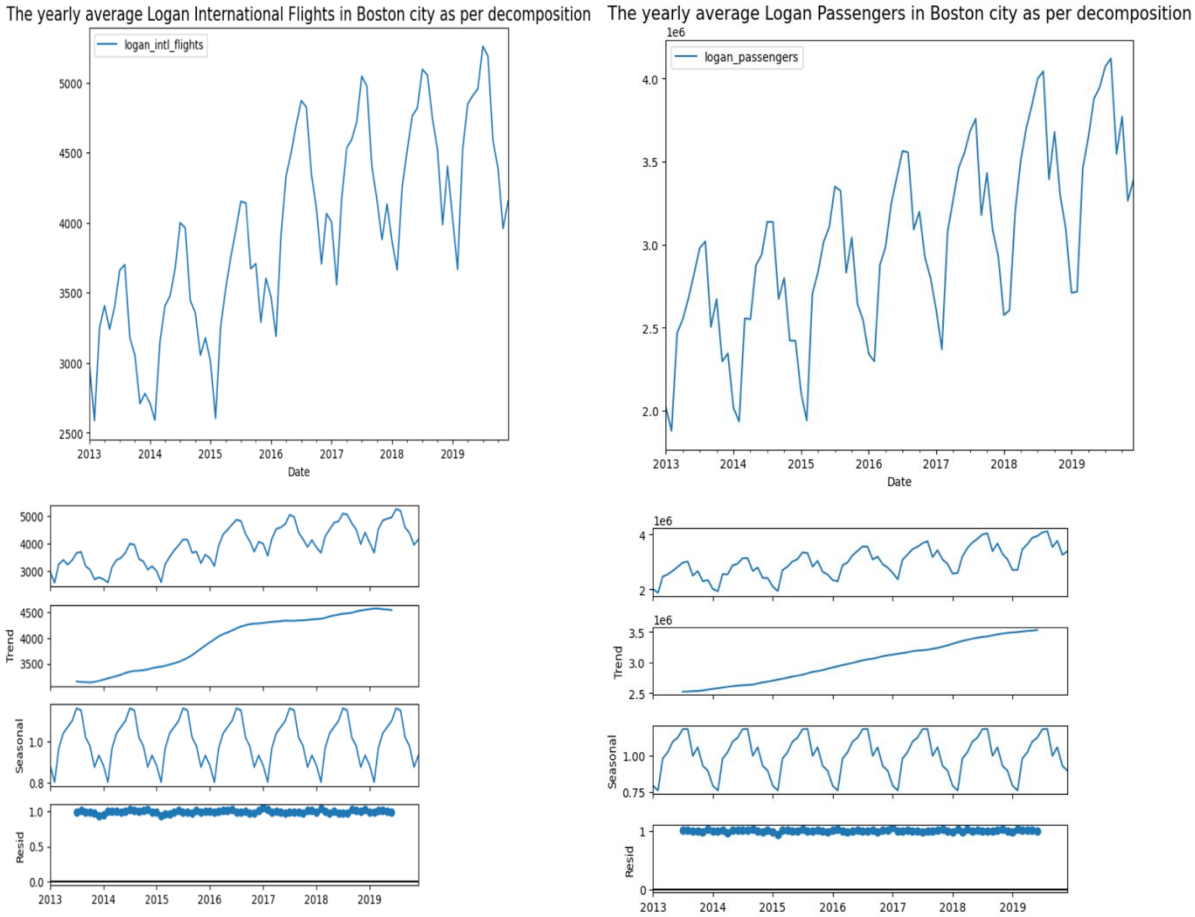


Fig: Decomposition components behaviour for logan flights and logan passengers

Later, in the analysis we have checked our data with respect to the logan international flights whether it is stationary or not by using the two prominent tests Augmented-Dickey-Fuller (ADF) and Kwiatkowski-Phillips-Schmidt-Shin (KPSS). After performing both the tests, we came to the result that our time series is non-stationary and it fails to reject the null hypothesis in ADF test and rejects the null hypothesis in KPSS test. In summary, for the ADF test, rejecting the null hypothesis means the time series is stationary and For the KPSS test, failing to reject the null hypothesis means the time series is stationary.
These two tests for the stationarity are having their factors in vice-versa condition, but while performing both, we got the result that our time series is not stationary as far now.

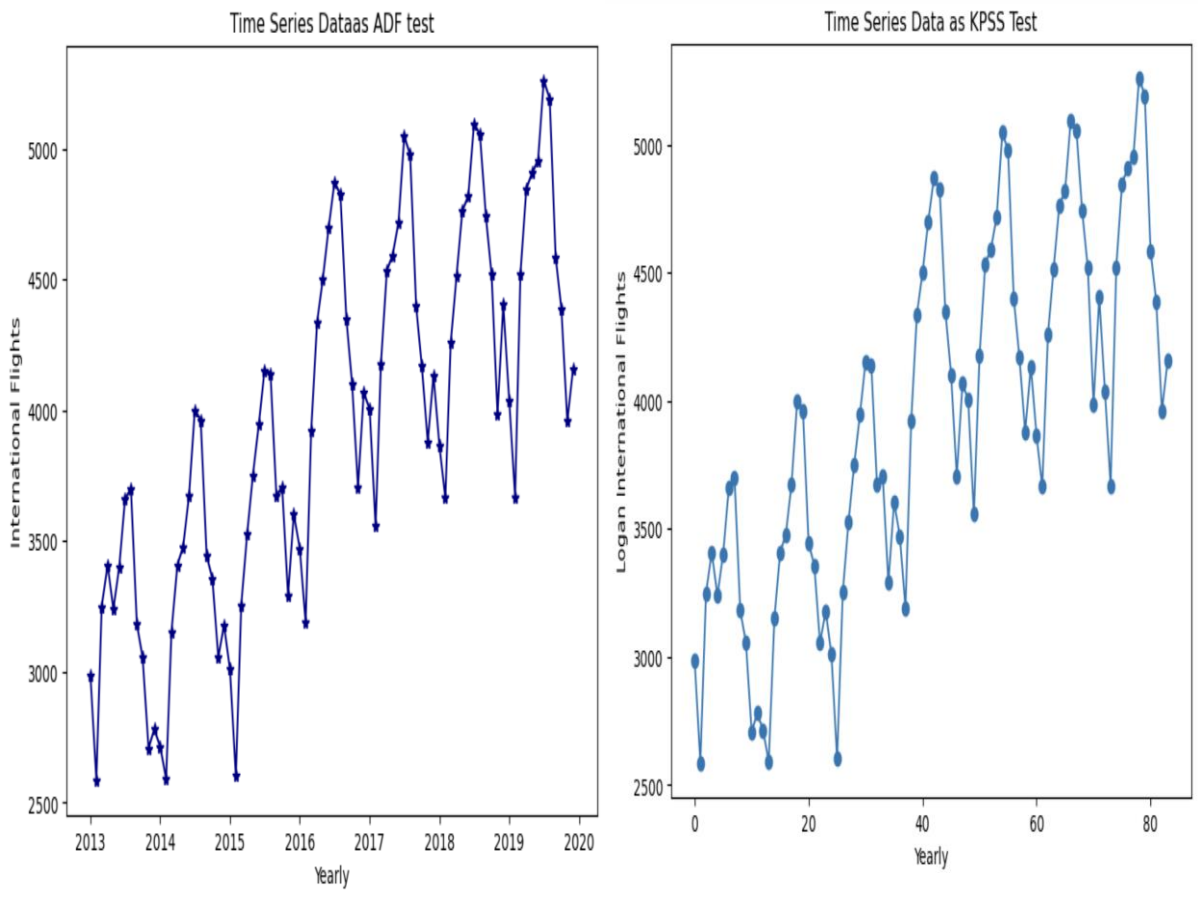


Fig: Stationarity test by using ADF and KPSS model

Moving forward in analysis, to make the time series stationary we perform the transformation method and in that we choose the differencing technique to differencing the dataset by using the lags. Below the first plot shows the original time series data to get the plot like that firstly we difference the data with a lag of 12 but we didn't achieve the result. After that we performed the task again and got the result at the lag of 24 where one can see the below graph which shows the lag 1 with the final result and the outcome is time series become stationary as per the ADF test. The

statistics also defined the stationarity of the time series model and the p-value which we got is lower than the optimal value which is 0.05.

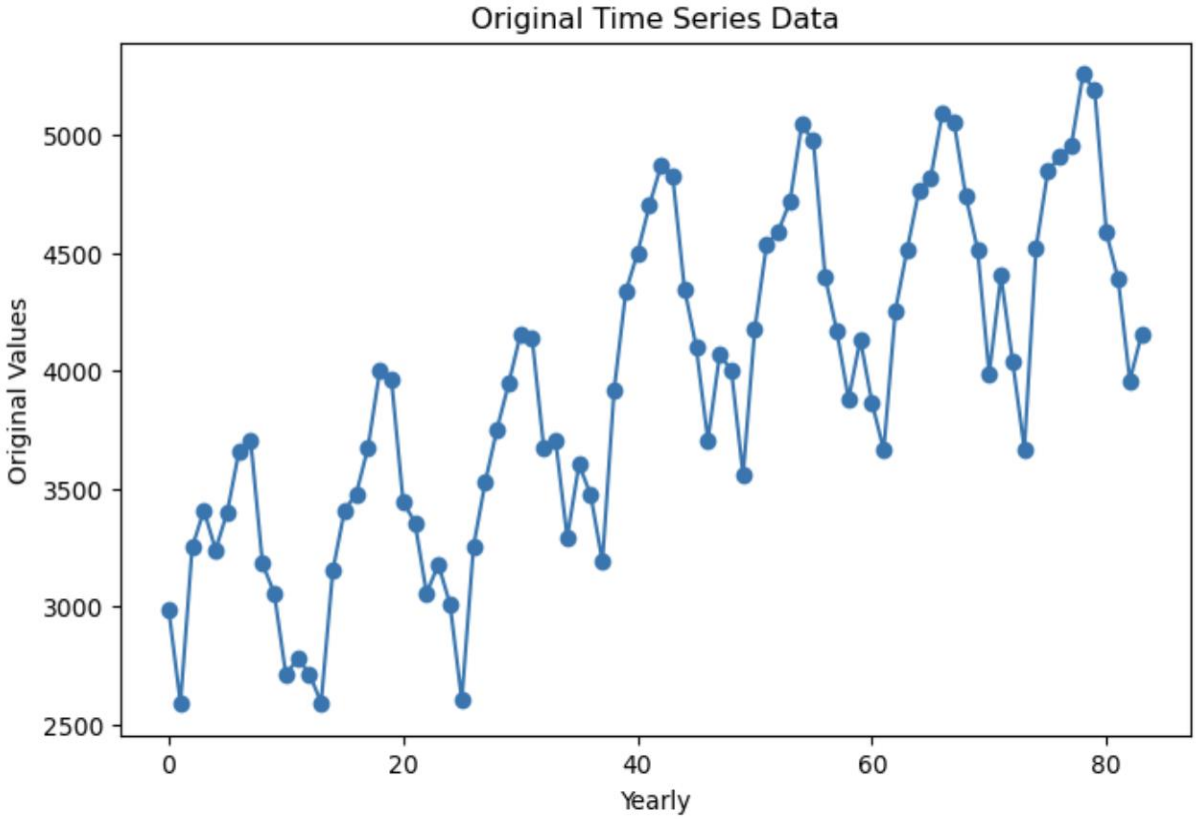
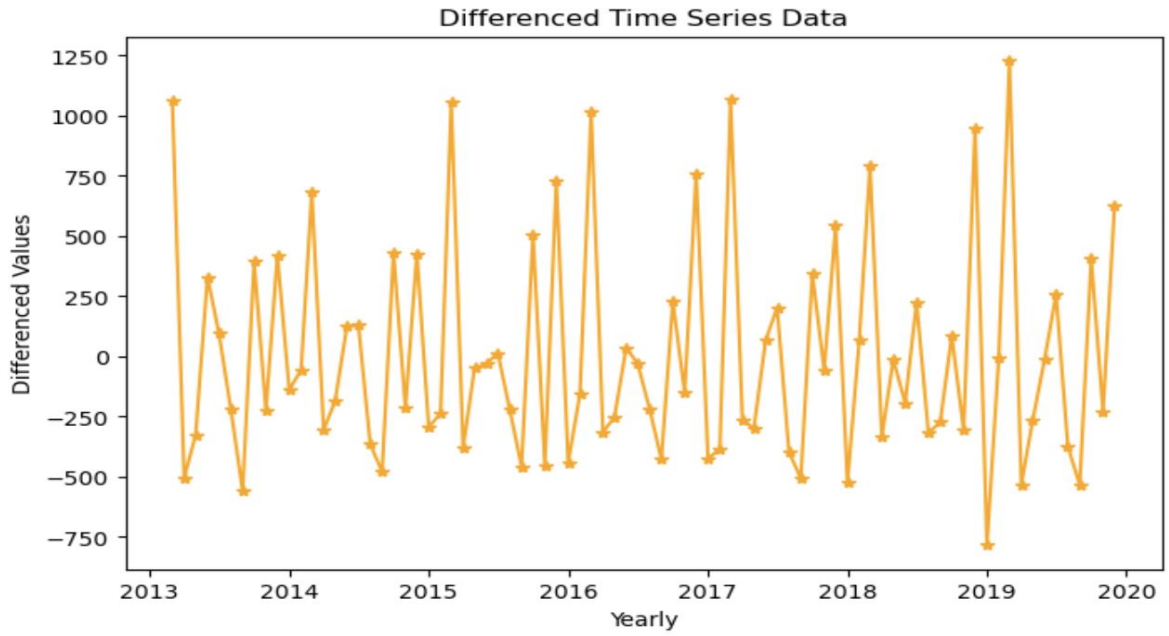


Fig: Plot for original time data




```
ADF Statistic: -13.171270506497267
p-value: 1.249002760595815e-24
Critical Values:
   1%: -3.526004646825607
   5%: -2.9032002348069774
   10%: -2.5889948363419957
The time series is likely stationary (reject the null hypothesis)
```

Fig: Plot for Differenced Time Series Data with a shift of 2

Further in the analysis, we implemented the Autocorrelation Function (ACF) and Partial-Autocorrelation Function (PACF) to pre-build the model the for Autoregressive (AR) Model and Moving Average (MA) Model. While working on ACF for building a pre-model, I got the plot for Logan Passengers and Logan International Flights. The number of lags to include in the plot is determined by the 'lag' parameter, which you can change to suit your needs. The width of the confidence interval surrounding the autocorrelation values is determined by the 'alpha' parameter. Library imported "from statsmodels.graphics.tsaplots import plot_acf".
In contrast, the PACF eliminates the impact of the intermediate lags from the measurement of the correlation between a time series and its lagged values. By removing the influence of the intermediate lags, it aids in determining the direct relationship between observations at various lags. Same as ACF, just need to update the library and I performed the Partial Autocorrelation Function (PACF) for the both the variables. Library imported "from statsmodels.graphics.tsaplots import plot_pacf".

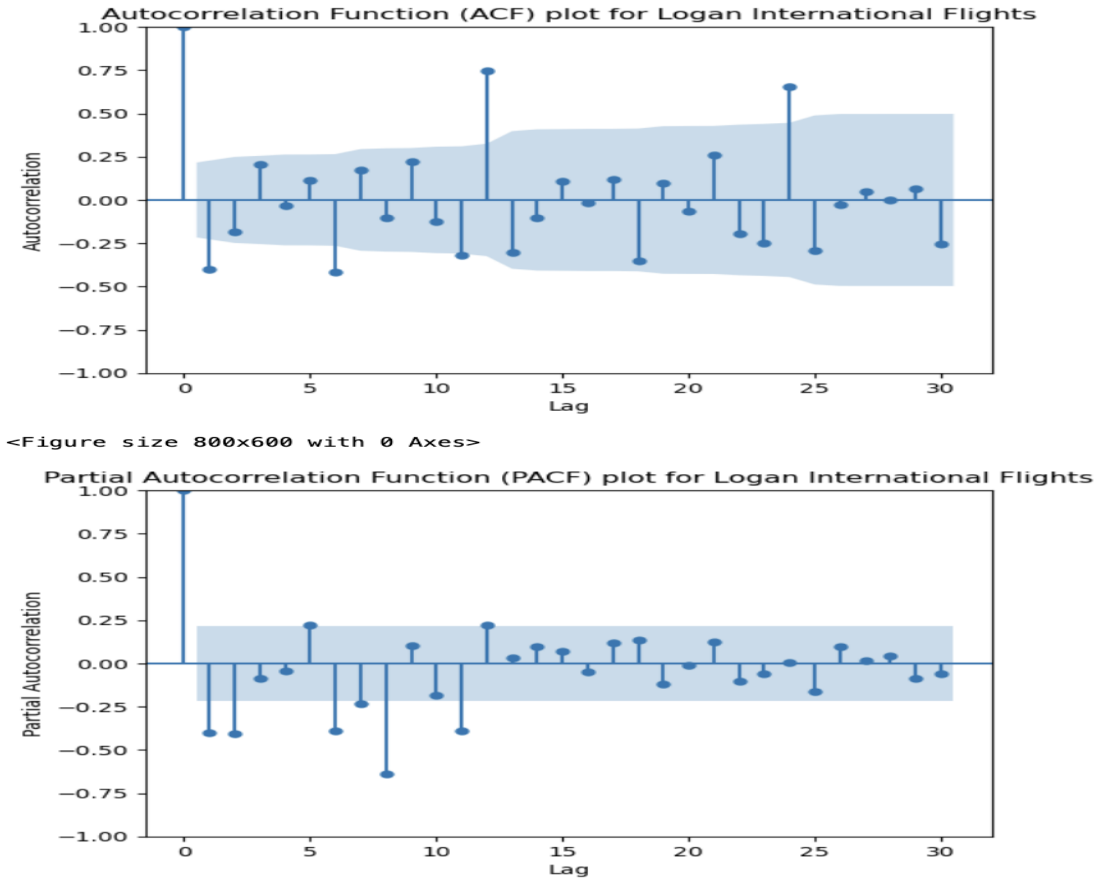


Fig: Plot of ACF and PACF for Logan International Flights

To implement the Autoregressive (AR) Model and Moving Average (MA) Model in the time series is a crucial part of the analysis. Time series analysis requires the use of autoregressive (AR) and moving average (MA) models because they offer a framework for comprehending and predicting temporal trends in data. The main aspect to use AR model is for the modelling memory technique i.e., they take into account observations from both the recent and older periods. How far back in time these dependencies stretch is indicated by the AR model's order. By mitigating transient fluctuations, MA models facilitate the identification of long-term patterns within the data and gives the stats for the logan international flights which shows a very close curves towards the original time series. This shows that our analysis is working in an excellent way.

To build the Autoregressive Integrated Moving Average (ARIMA) model we have to extend ARMA model by incorporating differencing to address non-stationarity which we had already performed in our analysis.

```
Mean Absolute Error 212.43580264345053
predicted=764.006729, expected=949.000000
predicted=-508.317921, expected=-783.000000
predicted=96.415909, expected=-7.000000
predicted=724.290375, expected=1228.000000
predicted=-282.238634, expected=-532.000000
predicted=26.424697, expected=-264.000000
predicted=-245.701229, expected=-12.000000
predicted=255.515707, expected=257.000000
predicted=-359.539927, expected=-375.000000
predicted=-203.276198, expected=-534.000000
predicted=17.028579, expected=406.000000
predicted=-288.671206, expected=-231.000000
predicted=751.666465, expected=625.000000
```

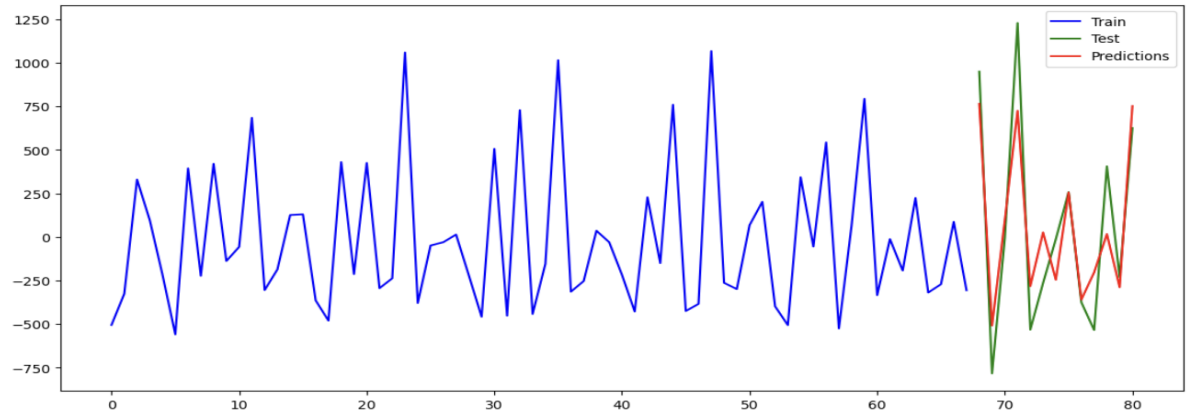
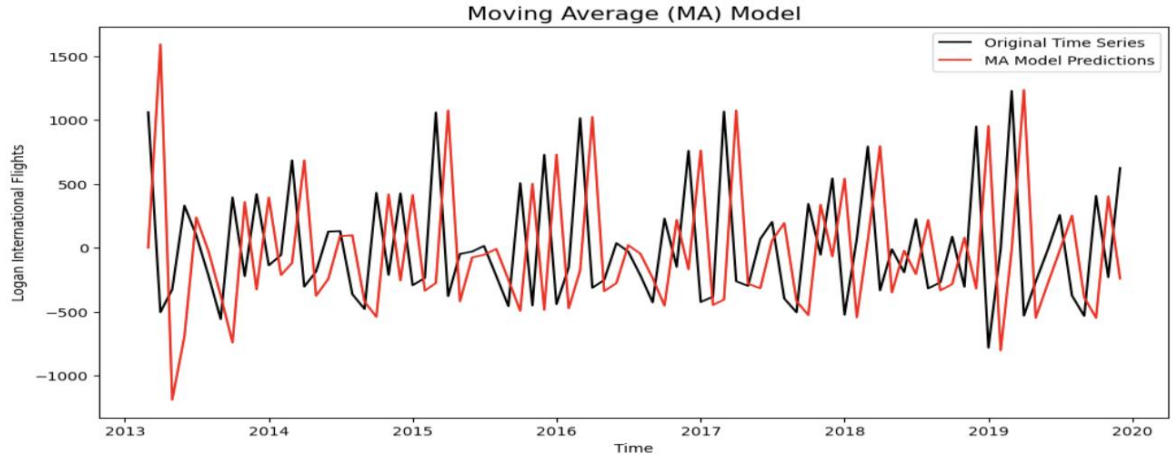


Fig: AR model of Logan Flights as per the Time and predicted and expected value

At last, we performed the last step of the time series in which we have build the ARIMA model and forecasted the data for the logan international flights and that too with logan passengers by using the train and test model for the next 3 months.



SARIMAX Results

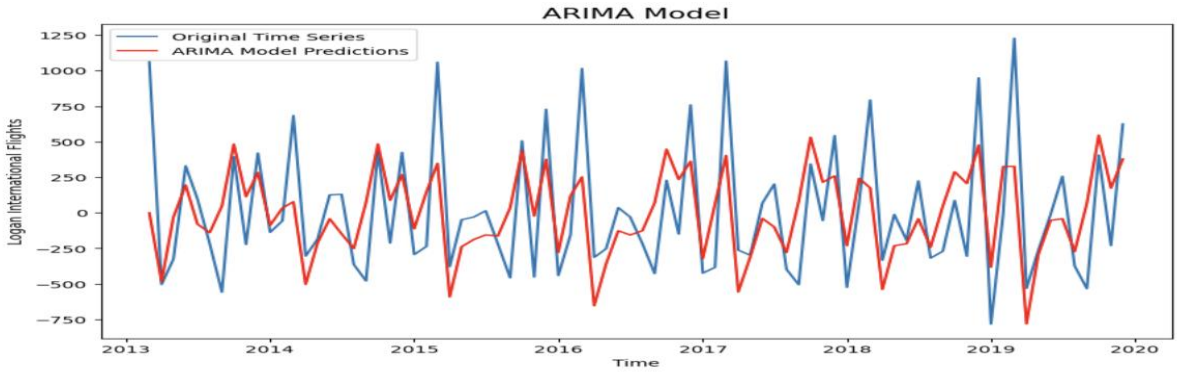
Dep. Variable:	logan_stationary	No. Observations:	82
Model:	ARIMA(0, 2, 1)	Log Likelihood	−644.757
Date:	Sun, 10 Dec 2023	AIC	1293.514
Time:	21:56:29	BIC	1298.278
Sample:	03-01-2013	HQIC	1295.424
	- 12-01-2019		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ma.L1	−0.9993	5.632	−0.177	0.859	−12.037	10.038
sigma2	5.681e+05	3.22e+06	0.176	0.860	−5.75e+06	6.88e+06

Ljung-Box (L1) (Q):	25.48	Jarque-Bera (JB):	2.12
Prob(Q):	0.00	Prob(JB):	0.35
Heteroskedasticity (H):	1.56	Skew:	−0.34
Prob(H) (two-sided):	0.25	Kurtosis:	2.60

Fig: MA Model of Logan International Flights and the SARIMAX Result

It is very obvious to forecast any data and make a future prediction, but everything is rely on the data what we have got from the analysis. The train and test model we build for the logan international flights shows that the flights will still run in the same pattern as it was running before and this leads to the increase in number of the flights as it significantly increasing. If we properly hover over the curves it shows that the highest number of flights running in the starting and ending of the year. The prediction curve is above the test model and later for the upcoming months it shows that the average flights would be running in the next year by the forecasting technique.



SARIMAX Results

Dep. Variable:	logan_stationary	No. Observations:	82
Model:	ARIMA(1, 0, 1)	Log Likelihood	−595.668
Date:	Sun, 10 Dec 2023	AIC	1199.336
Time:	21:49:21	BIC	1208.963
Sample:	03−01−2013	HQIC	1203.201
	− 12−01−2019		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
const	−0.5975	1.838	−0.325	0.745	−4.201	3.006
ar.L1	0.0880	0.158	0.558	0.577	−0.221	0.397
ma.L1	−1.0000	0.192	−5.196	0.000	−1.377	−0.623
sigma2	1.134e+05	1.7e−06	6.68e+10	0.000	1.13e+05	1.13e+05

Ljung−Box (L1) (Q):	0.04	Jarque−Bera (JB):	1.99
Prob(Q):	0.85	Prob(JB):	0.37
Heteroskedasticity (H):	1.17	Skew:	0.37
Prob(H) (two−sided):	0.69	Kurtosis:	2.83

Fig: ARIMA Model Prediction

The below picture shows the ARIMA forecasting model for Logan International Flights and by visualizing that we can conclude that the yellow line curve shows the future prediction of the international flights at Boston airport. The curve shows the future time prediction forecasting for the year 2020.

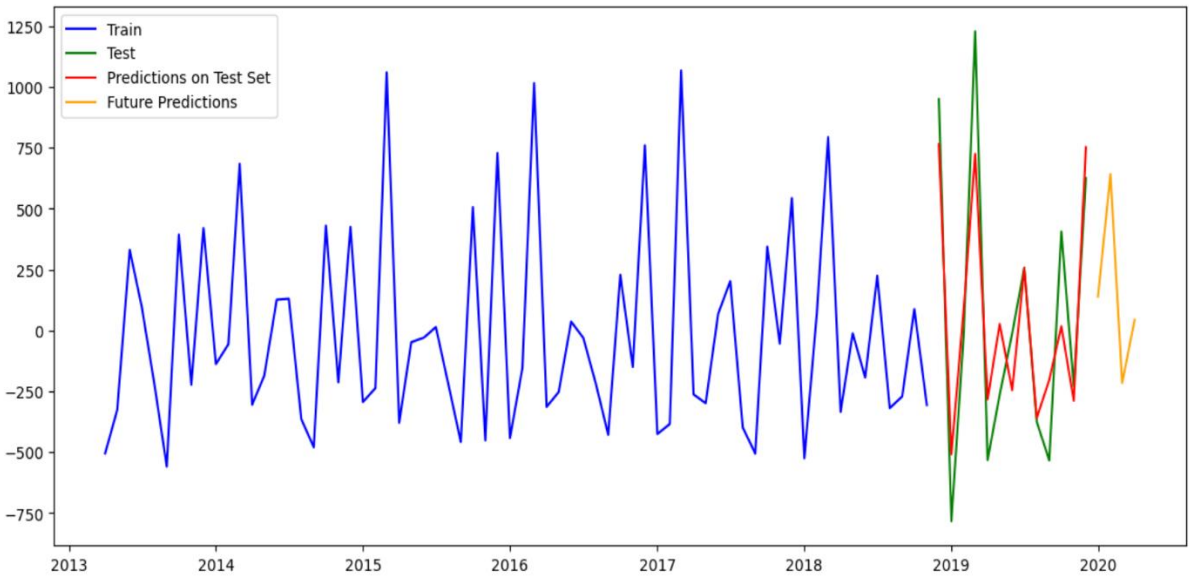


Fig: ARIMA model and forecasting of Logan International Flights for future

The above snippets shows the train and test model for the logan flights at the Boston Airport. The prediction shows the curve that the number of flights is getting increased as per the test model and slightly decreasing as compare to the train model. Before this, we have seen that the prediction curve is also getting increased for the international flights and that it correlates the variables as both are performing uniformly and this can be the major-affect in other sectors of the city like the rate of employment at the airport or in the city and other fields such as medical housing, hotel average daily rates and hotel occupancy in the city.

Appendix C: Code

pandas (pd) A powerful data manipulation and analysis library, providing data structures like DataFrames for efficient data handling.numpy (np) A fundamental package for scientific computing with support for large, multi-dimensional arrays and matrices. matplotlib.pyplot (plt) A data visualization library for creating static, animated, and interactive plots. statsmodels (sm) A library for estimating and testing statistical models, including time series analysis and econometrics. scipy.stats.distributions.chi2 Part of SciPy, it provides statistical functions, including the chi-squared distribution. %matplotlib inline A magic command in Jupyter notebooks that allows inline rendering of matplotlib plots. statsmodels.tsa.stattools Time series analysis tools, including the Augmented Dickey-Fuller test for unit root testing. statsmodels.tsa.ar_model.AutoReg AutoRegressive model implementation for time series forecasting. sklearn.metrics.mean_absolute_error A metric from scikit-learn for evaluating the mean absolute error between predicted and true values. The square root function from the Python standard library. statsmodels.graphics.tsaplots.plot_acf/pacf Function for plotting autocorrelation function (ACF) plots. Function for plotting partial autocorrelation function (PACF) plots. statsmodels.tsa.arima.model.ARIMA AutoRegressive Integrated Moving Average (ARIMA) model for time series forecasting.

- Necessary Imports.

```
import pandas as pd
import numpy as np
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import pyplot
import statsmodels.api as sm
from scipy.stats.distributions import chi2
%matplotlib inline
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.ar_model import AutoReg
from sklearn.metrics import mean_absolute_error
from math import sqrt
from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.graphics.tsaplots import plot_pacf
from statsmodels.tsa.arima.model import ARIMA
```

Fig: Important imports of python library

- Importing data and Summary Statistics.

```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

# Loading the dataset
file_path = 'economic-indicators.csv'
data = pd.read_csv(file_path)

# Construct a concise summary statistics report that can be displayed on a single screen.
summary_report = f"""
Date Range: {int(data['Year'].min())}-{int(data['Year'].max())} with monthly data.
Logan Passengers: Average ~{data['logan_passengers'].mean():,.0f}, ranging from ~{data['logan_passengers'].min():,.0
International Flights: Average ~{data['logan_intl_flights'].mean():,.0f}, ranging from {data['logan_intl_flights'].m
Hotel Occupancy Rate: Average {data['hotel_occup_rate'].mean()*100:.2f}%, min {data['hotel_occup_rate'].min()*100:.1
Hotel Average Daily Rate: Average ${data['hotel_avg_daily_rate'].mean():,.2f}, ranging from ${data['hotel_avg_daily_
Total Jobs: Average ~{data['total_jobs'].mean()/1000:.0f}K, ranging from ~{data['total_jobs'].min()/1000:.0f}K to ~{
Unemployment Rate: Average {data['unemp_rate'].mean()*100:.2f}%, ranging from {data['unemp_rate'].min()*100:.0f}% to
Median Housing Price: High variability with a standard deviation of ~${data['med_housing_price'].std():,.0f}.
"""

print(summary_report)
```

Fig: Code for summary statistics

- Heatmap of Correlation Matrix for Economic Indicators.

```
# Selecting relevant columns for the correlation analysis
# Focus on labor market data (total_jobs, unemp_rate), tourism figures (logan_passengers, logan_intl_flights),
# and median housing prices
columns_of_interest = ['total_jobs', 'unemp_rate', 'logan_passengers', 'logan_intl_flights', 'med_housing_price']
selected_data = data[columns_of_interest]

# Calculating the correlation matrix
correlation_matrix = selected_data.corr()

# Plotting the correlation matrix using a heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Correlation Matrix between Labor Market, Tourism Figures, and Median Housing Prices")
plt.show()
```

Fig: Code for Correlation Matrix

- Neural Network Analysis of Economic Indicators and Labor Market Dynamics in Boston.

```
from sklearn.preprocessing import StandardScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
import pandas as pd


# Preparing data for Neural Network Model
features = ['logan_passengers', 'logan_intl_flights', 'hotel_occup_rate', 'hotel_avg_daily_rate', 'med_housing_price
X = data[features]
y = data['total_jobs']

# Standardizing the data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
y_scaled = scaler.fit_transform(y.values.reshape(-1, 1))

# Splitting the scaled data
X_train_scaled, X_test_scaled, y_train_scaled, y_test_scaled = train_test_split(X_scaled, y_scaled, test_size=0.3, r

# Neural Network model
nn_model = Sequential()
nn_model.add(Dense(64, activation='relu', input_shape=(X_train_scaled.shape[1],)))
nn_model.add(Dropout(0.2))
nn_model.add(Dense(32, activation='relu'))
nn_model.add(Dense(1))

# Compile the model
nn_model.compile(optimizer='adam', loss='mean_squared_error')

# Train the model
nn_model.fit(X_train_scaled, y_train_scaled, epochs=100, batch_size=32, verbose=1)

# Evaluating the Neural Network model
nn_predictions = nn_model.predict(X_test_scaled)
nn_mse = mean_squared_error(y_test_scaled, nn_predictions)
nn_r2 = r2_score(y_test_scaled, nn_predictions)

print(f"Neural Network MSE: {nn_mse}, R2: {nn_r2}")
```

Fig: Code for Neural Network Analysis

- Decomposition of components (trends, seasonal, and residual) along decompose library.

```
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.stattools import adfuller

df['Date']=pd.to_datetime(df[['Year', 'Month']].assign(DAY=1))
df.plot(x='Date', y='logan_intl_flights', figsize=(8,6))
plt.title('The yearly average Logan International Flights in Boston city as per decomposition', fontsize=15)
df.set_index('Date', inplace=True)
analysis = df[['logan_intl_flights']].copy()
decompose_result_mult = seasonal_decompose(analysis, model="multiplicative")
trend = decompose_result_mult.trend
seasonal = decompose_result_mult.seasonal
residual = decompose_result_mult.resid
decompose_result_mult.plot();
```

Fig: Code for Decomposition of components

- Augmented Dickey-Fuller test.

```
df['Year'] = pd.to_datetime(df['Year'])

# Plot the original time series data
plt.figure(figsize=(8, 6))
plt.plot(df['logan_intl_flights'], marker='*', color='navy', linestyle='-')
plt.title('Time Series Dataas ADF test')
plt.xlabel('Yearly')
plt.ylabel('International Flights')
plt.show()

# Take the first difference
df['stationary_data'] = df['logan_intl_flights'] - df['logan_intl_flights'].shift(1)

# Perform Augmented Dickey-Fuller test
result = adfuller(df['logan_intl_flights'].dropna(), autolag='AIC')

# Extract and print the test results
adf_statistic = result[0]
p_value = result[1]
critical_values = result[4]

print(f'ADF Statistic: {adf_statistic}')
print(f'p-value: {p_value}')
print('Critical Values:')
for key, value in critical_values.items():
    print(f'   {key}: {value}')

# Interpret the results
if p_value <= 0.05:
    print('The time series is likely stationary (reject the null hypothesis)')
else:
    print('The time series is likely non-stationary (fail to reject the null hypothesis)')
```

Fig: Code for ADF test to check the stationarity of time series

- Differencing to transform the data and make it stationary.

```
# Plot the original time series data
plt.figure(figsize=(8, 5))
plt.plot(df['logan_intl_flights'], marker='o', linestyle='-')
plt.title('Original Time Series Data')
plt.xlabel('Yearly')
plt.ylabel('Original Values')
plt.show()

# Applying differencing technique to make the data stationary
df['logan_stationary']=df['logan_intl_flights'].diff().diff()

# Plot the differenced time series data
plt.figure(figsize=(8, 5))
plt.plot(df['logan_stationary'], color='orange', marker='*', linestyle='-')
plt.title('Differenced Time Series Data')
plt.xlabel('Yearly')
plt.ylabel('Differenced Values')
plt.show()

# Perform Augmented Dickey-Fuller test
result = adfuller(df['logan_stationary'].dropna(), autolag='AIC')

# Extract and print the test results
adf_statistic = result[0]
p_value = result[1]
critical_values = result[4]

print(f'ADF Statistic: {adf_statistic}')
print(f'p-value: {p_value}')
print('Critical Values:')
for key, value in critical_values.items():
    print(f'   {key}: {value}')

# Interpret the results
if p_value <= 0.05:
    print('The time series is likely stationary (reject the null hypothesis)')
else:
    print('The time series is likely non-stationary (fail to reject the null hypothesis)')
```

Fig: Code for differencing to make the data stationary

- Autocorrelation Function and Partial-Autocorrelation Function.

```
data_flight = df['logan_stationary']

# Plot the Autocorrelation Function (ACF)
plt.figure(figsize=(8, 6))
plot_acf(data_flight, lags=30, alpha=0.05)
plt.title('Autocorrelation Function (ACF) plot for Logan International Flights')
plt.xlabel('Lag')
plt.ylabel('Autocorrelation')
plt.show()

# Plot the Partial Autocorrelation Function (PACF)
plt.figure(figsize=(8, 6))
plot_pacf(data_flight, lags=30)
plt.title('Partial Autocorrelation Function (PACF) plot for Logan International Flights')
plt.xlabel('Lag')
plt.ylabel('Partial Autocorrelation')
plt.show()
```

Fig: Code to implement ACF and PACF

- Building Autoregressive (AR) Model to make the predictions.

```
# Applying AR model in the time series analysis

df.set_index('Date', inplace=True)
X = df['logan_stationary'].values
train, test = X[1:len(X)-13], X[len(X)-13:]
model = AutoReg(train, lags=11)
model_fit = model.fit()
predictions = model_fit.predict(start=len(train), end=len(train)+len(test)-1, dynamic=False)
mse = mean_absolute_error(test, predictions)
print("Mean Absolute Error", mse)
for i in range(len(predictions)):
    print('predicted=%f, expected=%f' % (predictions[i], test[i]))
rmse = sqrt(mean_absolute_error(test, predictions))
pyplot.figure(figsize=(14, 6))
pyplot.plot(range(len(train)), train, color='blue', label='Train')
pyplot.plot(range(len(train), len(train) + len(test)), test, color='green', label='Test')
pyplot.plot(range(len(train), len(train) + len(test)), predictions, color='red', label='Predictions')
pyplot.legend()
pyplot.show()
```

Fig: Code for AR model

- Building Moving Average (MA) model to make the predictions.

```
# Applying The MA Model

time_series = df['logan_stationary']

# Define the order of the MA model
order = (0, 2, 1)  # (p-autoregressive, d-differencing, q-moving average)

# Fit the MA model
ma_model = ARIMA(time_series, order=order)
ma_results = ma_model.fit()

# Generate predictions
predictions = ma_results.predict()

# Plot the original time series and the predicted values
plt.figure(figsize=(12, 6))
plt.plot(time_series, label='Original Time Series', color='black')
plt.plot(predictions, label='MA Model Predictions', color='red')
plt.legend()
plt.title('Moving Average (MA) Model', fontsize=15)
plt.xlabel('Time')
plt.ylabel('Logan International Flights')
plt.show()
print(ma_results.summary())
```

Fig: Code for MA model

- Building the ARIMA model to make the predictions for the forecasting.

```
# Define the order of the ARIMA model
order = (1, 0, 1)  # (p-autoregressive, d-differencing, q-moving average)

# Fit the ARIMA model
arima_model = ARIMA(time_series, order=order)
arima_results = arima_model.fit()

# Generate predictions
predictions = arima_results.predict()

# Plot the original time series and the predicted values
plt.figure(figsize=(10, 6))
plt.plot(time_series, label='Original Time Series')
plt.plot(predictions, label='ARIMA Model Predictions', color='red')
plt.legend()
plt.title('ARIMA Model', fontsize=15)
plt.xlabel('Time')
plt.ylabel('Logan International Flights')
plt.show()
print(arima_results.summary())
```

Fig: Code for ARIMA model

- The forecasting for the future prediction.

```
from pandas import date_range
from sklearn.metrics import mean_absolute_error
from math import sqrt
import matplotlib.pyplot as plt
from statsmodels.tsa.api import AutoReg

# Assuming df is your DataFrame and 'Date' is the name of the datetime column
#df.set_index('Date', inplace=True)

# Extract the time series data
X = df['logan_stationary'].values

# Split the data into train and test sets
train, test = X[1:len(X)-13], X[len(X)-13:]

# Fit the AutoReg model
model = AutoReg(train, lags=11)
model_fit = model.fit()

# Make predictions on the test set
predictions = model_fit.predict(start=len(train), end=len(train)+len(test)-1, dynamic=False)

# Calculate Mean Absolute Error
mse = mean_absolute_error(test, predictions)
print("Mean Absolute Error:", mse)

# Print predicted and expected values
for i in range(len(predictions)):
    print('predicted=%f, expected=%f' % (predictions[i], test[i]))

# Forecast future values
forecast_steps = 3  # Change this to the number of steps you want to forecast into the future
future_dates = date_range(start=df.index[-1], periods=forecast_steps + 1, freq='M')
future_predictions = model_fit.predict(start=len(X), end=len(X) + forecast_steps, dynamic=False)

# Plotting
plt.figure(figsize=(14, 6))
plt.plot(df.index[1:len(X)-13], train, color='blue', label='Train')
plt.plot(df.index[len(X)-13:], test, color='green', label='Test')
plt.plot(df.index[len(X)-13:], predictions, color='red', label='Predictions on Test Set')
plt.plot(future_dates, future_predictions, color='orange', label=f'Future Predictions')
plt.legend()
plt.show()
```

Fig: Code for the future prediction

References

https://data.boston.gov/dataset/economic-indicators-legacy-portal

Contributions

All the co-author have performed equally in the project.