ISSUES

The data used in this study was collected from the City of Boston's open data hub, specifically from among the 245 available datasets on the Analyze Boston website. The dataset we used is called "Economic Indicators" and contains monthly-tracked information from January 2013 to December 2019 on employment, housing, travel, and real estate development. We used multiple columns to forecast future trends, considering various variable's impacts. We selected data on the average daily rate of hotels in Boston, measured in dollars. With this information, we have developed a plan to answer our questions, as outlined below:

- What is the trend of hotel prices and occupancy in Boston, and what range does it fall under?
- Does the past trend in the hotel industry affect future occupancy and rates?
- What could be the future prices and occupancy rate of hotels in the next six months?
- How accurate are these predictions?
- Is there a possible dependency between hotel average prices and hotel occupancy rates? If yes, how correlated are these two variables?

FINDINGS

- The past trend of the hotel industry does affect the occupancy and rate of hotels. There is a uniform increase in the price and occupancy of the hotels based on yearly data, where the variables increase by a small quantity year after year.
- The value ranges between 157 and 337 USD, which usually increases during the third quarter of the year. The hotels seem to be very busy during the second and third quarters with almost 93% occupancy.
- The predictions for the next six months for hotel occupancies are 0.65, 0.63, 0.68, 0.81, 0.86, 0.88, and 0.91, and hotel prices are 181, 155, 164, 223, 283, 305, and 302.
- The mean absolute error for the prediction of occupancy and prices is 0.03 and 9.07, respectively.
- Yes, a strong positive correlation between hotel occupancy and average daily rates were found with a correlation of 88%.

In Conclusion, hotel occupancy rates affects the hotel price ranges and both the variable seen to be correlated.

DISCUSSION

Time series forecasting gave us a very clear analysis on average hotel rates and occupancy. The data was mostly seasonal, meaning that the prices and rates depend on the season. The hikes differ from season to season, repeating year after year. There could be multiple factors responsible for this. But we focused on future predictions of the data for two variables - hotel occupancy rate and hotel average daily rate.

A 6 month prediction of hotel average prices was done which gives a forecast on the price rates in the Boston city. Likewise, the next year's prediction of hotel occupancy rates was done. There seems to be a good amount of dependency between hotel occupancy rate and prices. The variables are directly dependent in such a way that the hotel prices increase with the increase in occupancy rates. The forecasting models gave us an accurate prediction of the daily hotel price rate. The evaluation methods used clarified how well the model fit the data. The evaluation metrics like mean absolute error, AIC, BIC and Log Likelihood were used in choosing the best hyper parameters for the model.

APPENDIX A: METHOD

a. Data Collection

The Boston Planning and Development Authority has developed a legacy dataset that maintains economic data related to employment, housing, travel, and real-estate development. This data has monthly information about the above mentioned variables from 2013 until 2019. This is used in making future forecasts to help predict price hikes and trends. This trend is then helpful in understanding the increase or decrease in the economy of the city.
The link to the website is given below: https://data.boston.gov/dataset/economic-indicators-legacy-portal

b. Variable Creation

1. The 'Year' variable was extracted from the economic indicators dataset spanning from 2013 to 2019.
2. The 'Month' variable was derived from the economic indicators dataset spanning from January 2013 to December 2019. It captures monthly variations, enabling detailed analysis of economic patterns within this 7-year period.
3. The 'Hotel_occupancy_rate' column represents the percentage of hotel occupancy in Boston. This value ranges between 0 and 1.
4. The 'Hotel_average_daily_rate' column is the average daily Boston hotel room price
5. Date - The date variable was extracted from the year and month columns to be converted into datetime format for the analysis

c. Data Cleaning and Preprocessing

1. **Datetime conversion**: The year and month column were converted into Date column and hence transformed into datetime datatype.
2. **Differencing**: The dataset was converted into stationary dataset by differencing the variables
3. **Null Values**: We have dropped the null values which are created due to transformation.

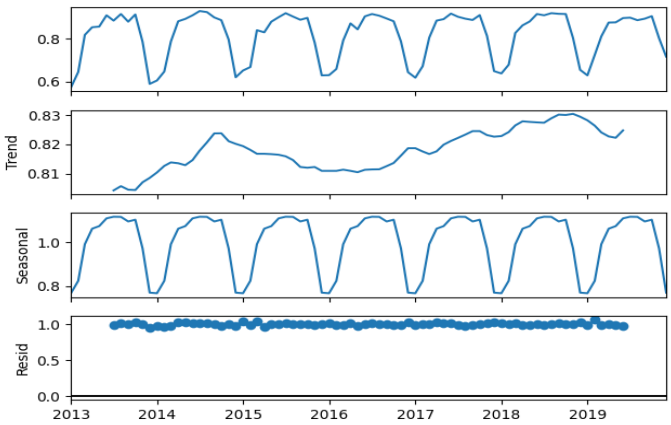
Analytical Methods

1. **Decomposition Analysis**: The decomposition graph visually dissects time series components—trend, seasonal, and residual—facilitating pattern identification, anomaly detection, and model assessment.
2. **The Augmented Dickey-Fuller (ADF) Test**: The ADF test assesses time series stationarity, which is essential for reliable forecasting models. It checks if a dataset exhibits stationarity—the assumption of consistent statistical properties over time such as p-value, critical values etc.
3. **Autocorrelation (ACF) and Partial autocorrelation (PACF) Plot**: The ACF and PACF plot visualizes the autocorrelation and partial autocorrections present within time series data across different time lags. This is used to identify significant values for future lags. These plots are helpful in selecting appropriate models based on their patterns and significant values.
4. **AR Model**: The AR(p) regresses the current value against the p prior observations. This directly incorporates temporal dependency by using lagged points as predictors. Selecting the optimal AR order identifies how long in the past influence persists for modeling serial correlation.
5. **ARIMA Model**: The autoregressive integrated moving average (ARIMA) model is a time series forecasting method. It combines autoregression (AR), differencing (I), and moving averages (MA). ARIMA captures temporal dependencies and trends, making it valuable for predicting future values based on historical patterns in the data.
6. **LLR Test**: The Likelihood Ratio Test (LLR) is employed for model comparison, assessing the goodness of fit between nested models. In time series analysis, it aids in selecting the most fitting model by comparing likelihoods, providing insights into the appropriateness of the chosen representation for the data in question.
7. **Residual test:** There are two types of test qualitative and quantitative analysis. Q-Q plot is plotted to find qualitative analysis which gives the scatter plot of the data on a straight line. A histogram is plotted to check if the data is uniformly distributed which gives the efficiency of the model.

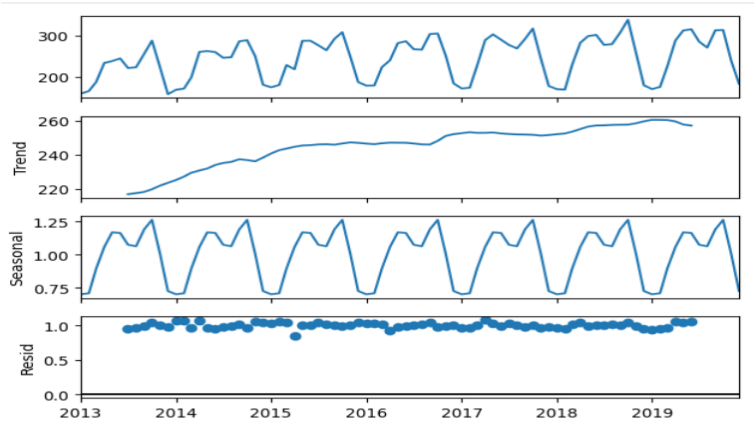
8. **Granger Causalit**: It is used to assess predictive power of one variable over another. The Granger causality test is a statistical hypothesis test for determining whether one time series is useful in forecasting another
9. **Cross Correlation**: It determines the similarity of two sequences of numbers as they move in time relative to each other.
10. **Johansen cointegration test**: A statistical method for analysing cointegration relationships among multiple time series variables.It implies that, while these variables may be non-stationary individually, a combination of these variables exists that is stationary.

APPENDIX B: RESULT

The hotel occupancy rate and hotel average daily rate variables were analyzed to understand the future rates. The models gave us a clear analysis on the trends followed by the variable over a period of 7 years. It was found to be seasonal and increasing in trend from the decomposition plot.



Hotel Occupancy



Hotel Average Daily Rate

ADF test was performed to analyze if the data was stationary using the p-value and critical values. It was found that the hotel daily average rate was stationary since the p-value is 0.0058 which is less than 0.05. However, the hotel occupancy rate was found to be non-stationary with 0.435.

```
Results of Dickey-Fuller Test:
Test Statistic                  -3.597105
p-value                          0.005817
#Lags Used                      11.000000
Number of Observations Used     72.000000
Critical Value (1%)             -3.524624
Critical Value (5%)             -2.902607
Critical Value (10%)            -2.588679
dtype: float64
```

Hotel Average Daily Rate

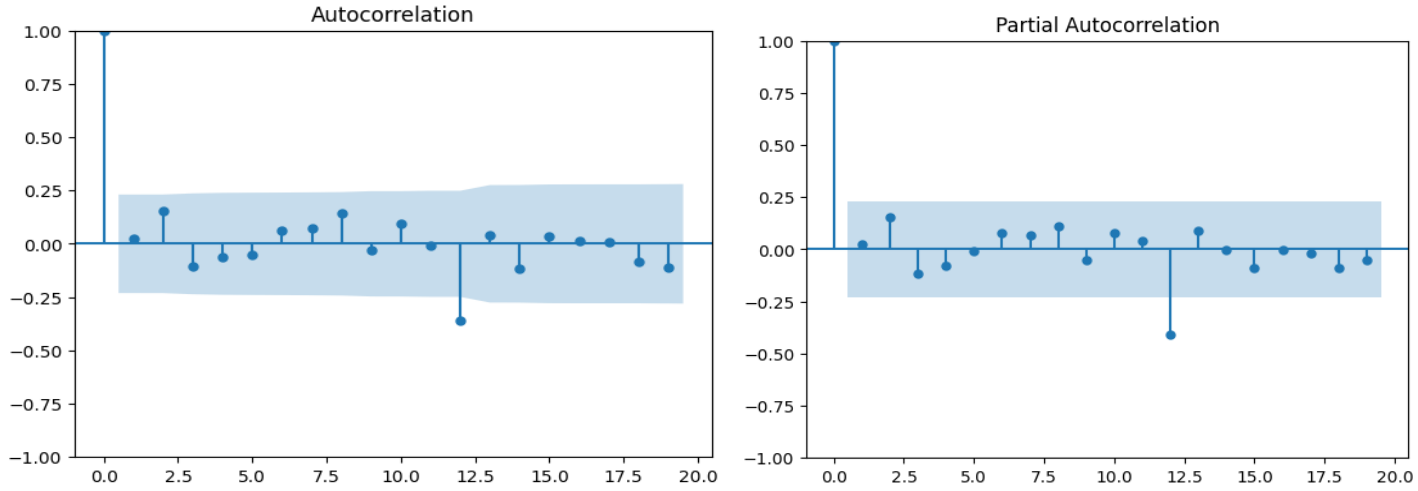
```
Results of Dickey-Fuller Test:
Test Statistic                  -1.690633
p-value                          0.435945
#Lags Used                      11.000000
Number of Observations Used     72.000000
Critical Value (1%)             -3.524624
Critical Value (5%)             -2.902607
Critical Value (10%)            -2.588679
dtype: float64
```

Hotel Occupancy

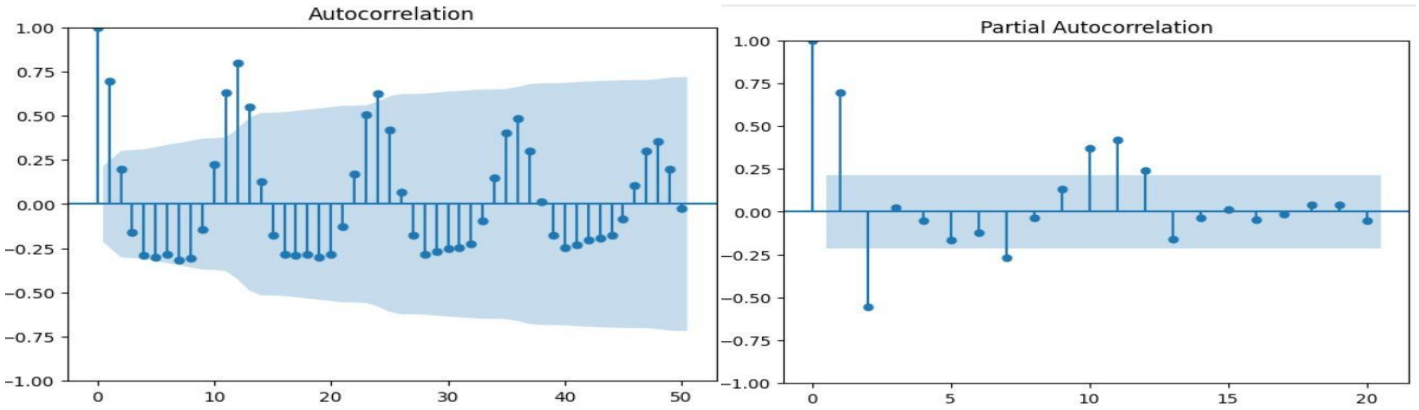
Hence, the hotel occupancy rate variable was transformed using a differencing method in order to make it stationary. After differencing, a p-value of 0.0397 was obtained making the data stationary.

```
ADF Statistic: -2.9509362745172987
p-value: 0.039737366609597 91
Critical Values:
   1%: -3.5443688564814813
   5%: -2.9110731481481484
   10%: -2.5931902777777776
The time series is likely stationary (reject the null hypothesis)
```

The ACF and PACF graphs were plotted which showed the significant values that could be used as lags in our models for best accuracy. From the two graphs plotted for each, it was clear that the significant values of hotel occupancy rates are 1 and 12 and the significant values of hotel daily average rates are 1, 2 10, 11 and 12. Hence, the models were tested on these values as lags.



Hotel Occupancy



Hotel Average Daily Rate

It is clear from the graph that ARIMA model could be used for training the hotel occupancy data and AR for hotel average daily rate since it is forming sinusoidal decreasing waves.

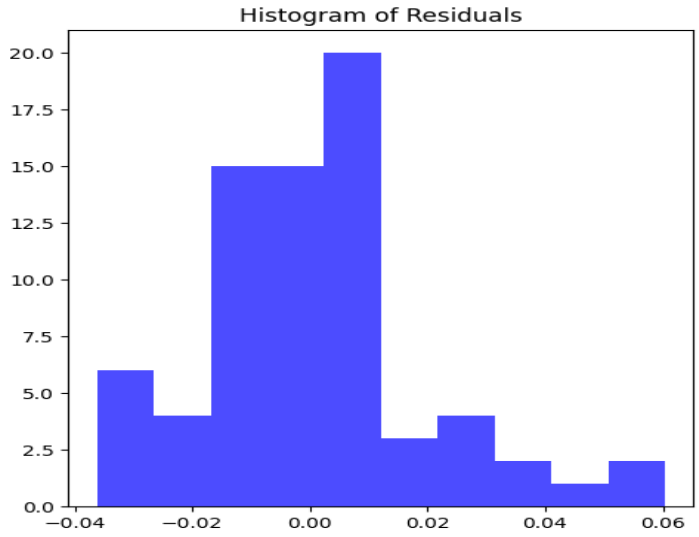
SARIMAX Results

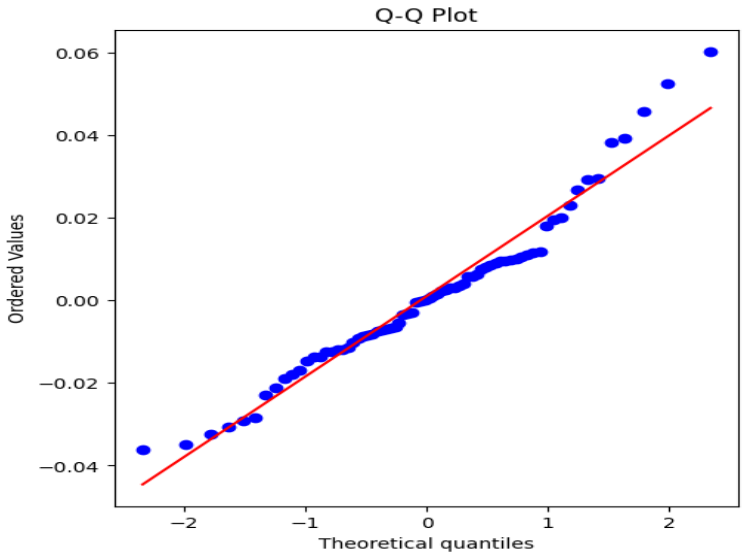
Dep. Variable:	stationary_data	No. Observations:	84
Model:	ARIMA(12, 0, 12)	Log Likelihood	183.133
Date:	Sat, 09 Dec 2023	AIC	-314.266
Time:	21:51:44	BIC	-251.065
Sample:	0	HQIC	-288.860
	- 84		
Covariance Type:	opg		

The model result shown above proves to be the best with an order of 12,0,12 as p value and q value by achieving an AIC of -314.266 and BIC of -261.065. Hence, further analysis and visualizations were done to confirm the model accuracy.

First, residual analysis was done to visualize the QQ plot and understand the correlations. QQ plot showed a scatter plot of the data which is seen to be present almost in a straight line proving that the data fits best for the model.

The histogram of residuals were plotted to check if the data is uniformly distributed.





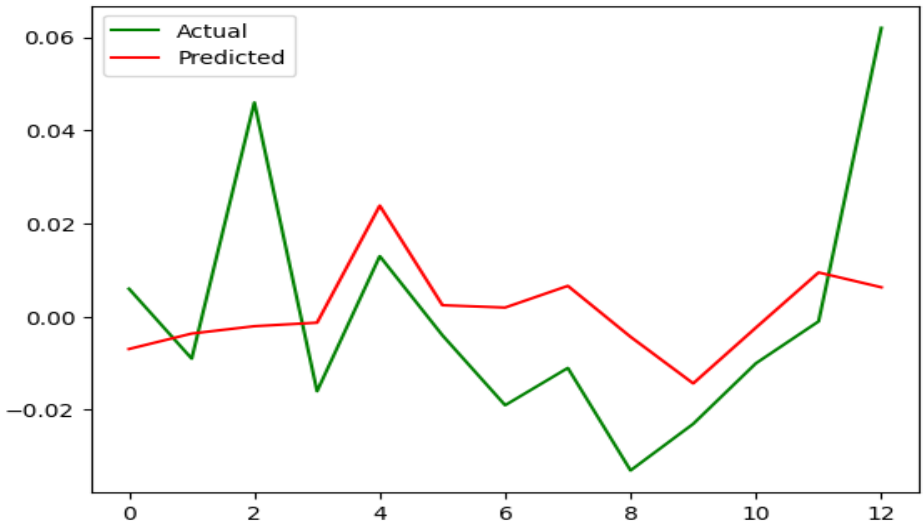
Whereas, the AR model for hotel average daily rate with a lags of 11, 12 were found to be the best fit. The likelihood value was considered and is found to be significantly less with -361.455. Also, the LLR p value is matching with p values of models of lag 11 and 12.

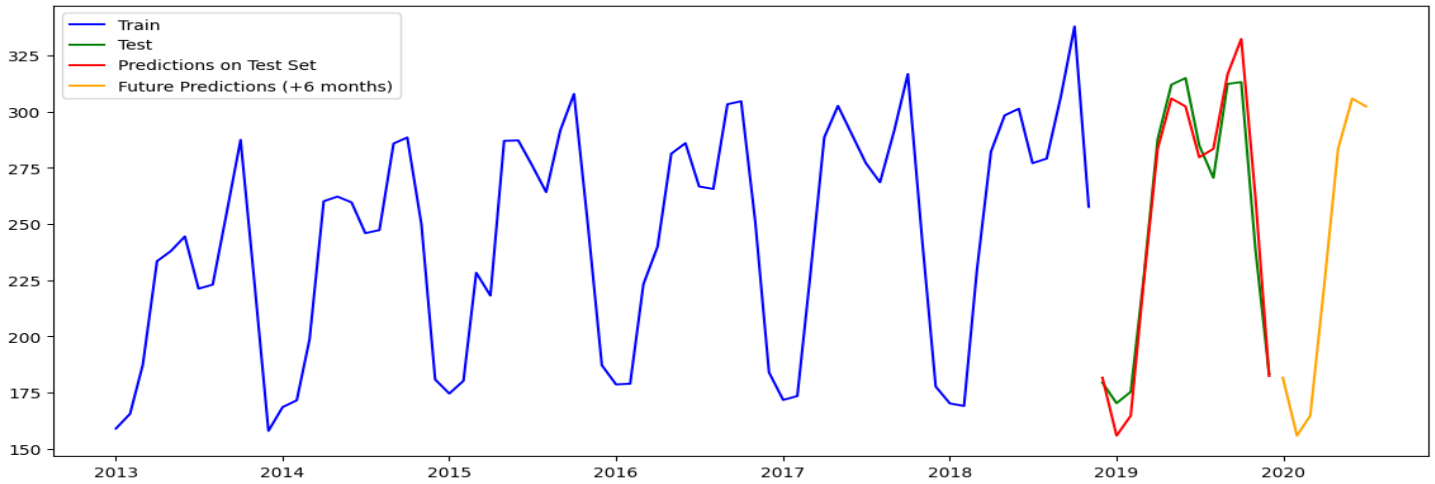
SARIMAX Results

Dep. Variable:	hotel_avg_daily_rate	No. Observations:	84
Model:	ARIMA(11, 0, 0)	Log Likelihood	-361.455
Date:	Mon, 04 Dec 2023	AIC	748.911
Time:	17:32:03	BIC	780.511
Sample:	01-01-2013	HQIC	761.614
	- 12-01-2019		
Covariance Type:	opg		

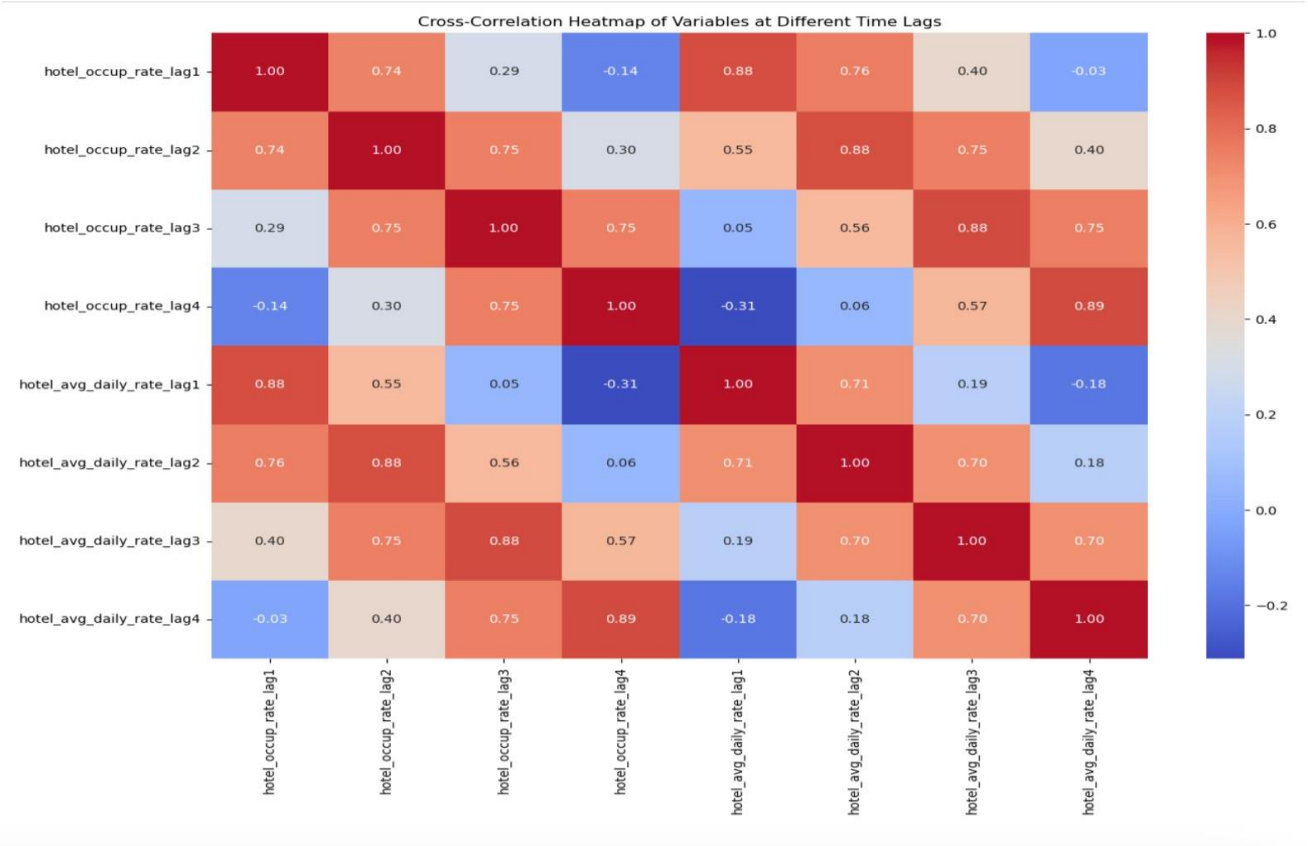
LLR test p-value: 0.0

The future forecasting of the hotel occupancy rates and hotel average daily rates were performed using the models and the predictions are shown in the graph below.





Furthermore, the possibility of dependencies of these two variables were studied giving us the following results. The initial analysis aimed to discern correlations and relationships among Hotel Occupancy Rate, Hotel Average Daily Rate from 2013 to 2019. Results revealed a strong positive correlation between the hotel metrics, and a robust correlation between hotel occupancy rate and average daily rate.
The cross-correlation plot shows strong positive correlations between hotel occupancy and average rate data i.e. is 0.88.



The Granger Causality Test for the columns ‘hotel_avg_daily_rate' and 'hotel_occup_rate'. After performing this test for 4 times we noticed that in the 1st lag the p-value was close to 0. Therefore the null hypothesis was rejected which stated that the past values of one variable do not significantly help in predicting the future values of another.

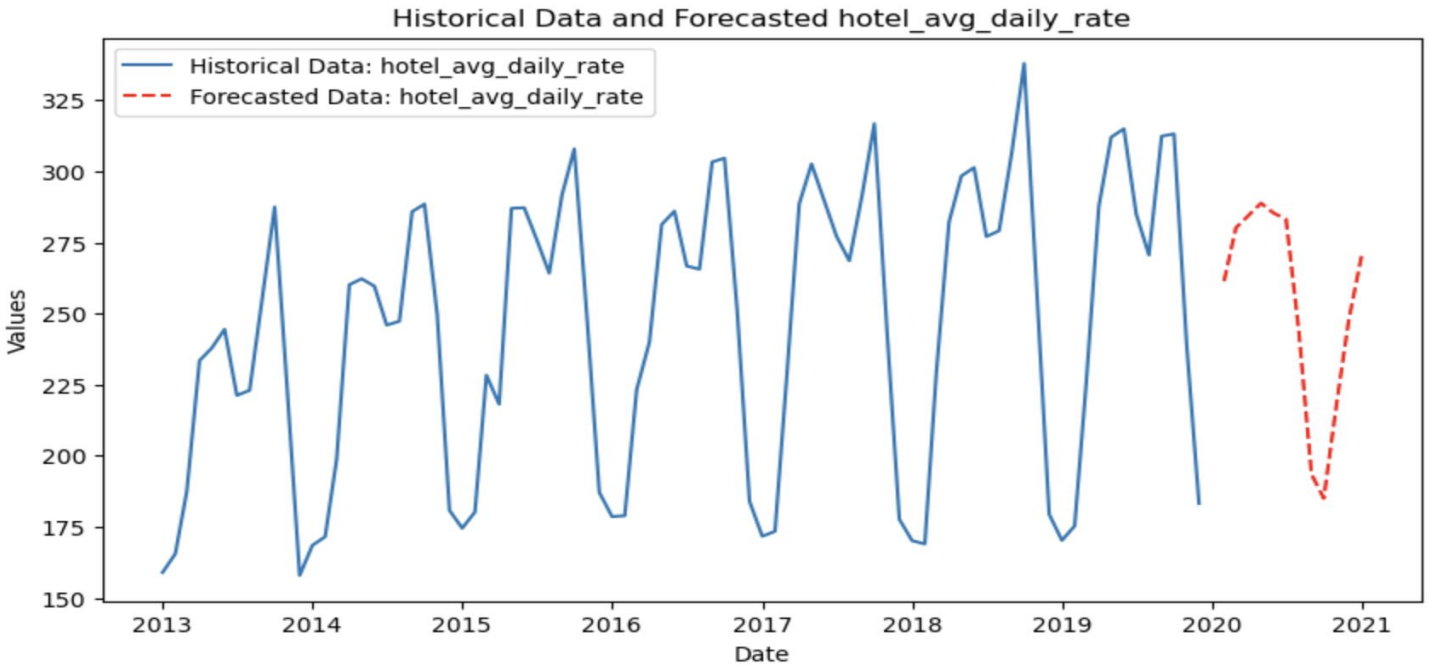
```
Granger Causality
number of lags (no zero) 1
ssr based F test:         F=16.5917 , p=0.0001  , df_denom=80, df_num=1
ssr based chi2 test:   chi2=17.2139 , p=0.0000  , df=1
likelihood ratio test: chi2=15.6427 , p=0.0001  , df=1
parameter F test:         F=16.5917 , p=0.0001  , df_denom=80, df_num=1

Granger Causality
number of lags (no zero) 2
ssr based F test:         F=6.9879  , p=0.0016  , df_denom=77, df_num=2
ssr based chi2 test:   chi2=14.8832 , p=0.0006  , df=2
likelihood ratio test: chi2=13.6765 , p=0.0011  , df=2
parameter F test:         F=6.9879  , p=0.0016  , df_denom=77, df_num=2

Granger Causality
number of lags (no zero) 3
ssr based F test:         F=10.7012 , p=0.0000  , df_denom=74, df_num=3
ssr based chi2 test:   chi2=35.1404 , p=0.0000  , df=3
likelihood ratio test: chi2=29.1884 , p=0.0000  , df=3
parameter F test:         F=10.7012 , p=0.0000  , df_denom=74, df_num=3

Granger Causality
number of lags (no zero) 4
ssr based F test:         F=10.0316 , p=0.0000  , df_denom=71, df_num=4
ssr based chi2 test:   chi2=45.2130 , p=0.0000  , df=4
likelihood ratio test: chi2=35.8392 , p=0.0000  , df=4
parameter F test:         F=10.0316 , p=0.0000  , df_denom=71, df_num=4
```

After that we performed Cointegration test which gave us the result that this was assessed to check whether the 'hotel_occup_rate' and hotel_avg_daily_rate' variables have a stable long-term relationship, which can influence the decision to model them together using an ARIMA framework with one as an exogenous variable for the other. We concluded this as trace_statistic value was greater then the trace_statistic threshold and max_eigenvalue_statistic_h was greater than max_eigenvalue_threshold_h.



We concluded that hotel average daily rate can be forecasted using hotel occupancy rate.

After training the ARIMA model and assigning the parameters p, d, q with 12, 0, 12 on variables hotel average rates and hotel occupancy, forecast the average daily rates of the hotel by using hotel occupancy variable as exogenous variable. We can also see that the trend is following the past pattern.

APPENDIX C: Data and Code

In this appendix anyone can replicate our analysis with the help of python code. Use the git hub repository.
https://github.com/Milkyy-way/Project-3-MTH-522.git

1) Importing the Libraries

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

2) Creating new data column

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

3) Decomposition of variable with time

```
df.set_index('Date', inplace=True)
analysis = df[['hotel_occup_rate']].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();
```

4) ADF testing

```
def adf_test(timeseries):
    print ('Results of Dickey-Fuller Test:')
    dftest = adfuller(timeseries, autolag='AIC')
    dfoutput = pd.Series(dftest[0:4], index=['Test Statistic','p-value','#Lags Used','Number of Observations Used'])
    for key,value in dftest[4].items():
        dfoutput['Critical Value (%s)'%key] = value
    print (dfoutput)
adf_test(df['hotel_occup_rate'])
```

5) Transformation of data

```
df['stationary_data_occup'] = df['hotel_occup_rate'] - df['hotel_occup_rate'].shift(12)
```

6) ACF and PACF plots

```
fig = plot_acf(df['stationary_data'].dropna())
plt.show()
```

```
pacf_plot=plot_pacf(df.stationary_data.dropna())
```

7) Fitting of ARIMA model for best lag

```
arma_11 = sm.tsa.ARIMA(df['stationary_data'], order=(1, 0, 1))
arma_results = arma_11.fit()
print(arma_results.summary())
```

8) Residual model

```
model = sm.tsa.SARIMAX(df['stationary_data'].dropna(), order=(12, 0, 12), trend='c')
model_fit = model.fit(disp=True)
residuals = model_fit.resid
residuals_mean = np.mean(residuals)
residuals_std = np.std(residuals)
residuals_min = np.min(residuals)
residuals_25th_percentile = np.percentile(residuals, 25)
residuals_median = np.median(residuals)
residuals_75th_percentile = np.percentile(residuals, 75)
residuals_max = np.max(residuals)
print("Residual Summary Statistics:")
print(f"Mean: {residuals_mean}")
print(f"Standard Deviation: {residuals_std}")
print(f"Minimum: {residuals_min}")
print(f"25th Percentile: {residuals_25th_percentile}")
print(f"Median: {residuals_median}")
print(f"75th Percentile: {residuals_75th_percentile}")
print(f"Maximum: {residuals_max}")
plt.figure(figsize=(12, 6))
plt.plot(residuals)
stat, p = acorr_ljungbox(residuals, lags=10)
# Print the Ljung-Box Test Results
print("\nLjung-Box Test Results:")
print(f"Ljung-Box Test Statistic: {stat}")
print(f"P-value: {p}")
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.hist(residuals, bins='auto', color='blue', alpha=0.7)
plt.title('Histogram of Residuals')
plt.subplot(1, 2, 2)
stats.probplot(residuals, dist="norm", plot=plt)
plt.title('Q-Q Plot')
plt.show()
sm.graphics.tsa.plot_acf(residuals, lags=20)
plt.title('Autocorrelation Function (ACF) of Residuals')
plt.show()
```

9) LLR Test

```
def LLR_test(mod_1, mod_2, DF=1):
    L1=mod_1.fit().llf
    L2=mod_2.fit().llf
    LR=(2*(L2-L1))
    p=chi2.sf(LR, DF).round(3)
    return p
```

10) AR model with future prediction

```
from pandas import date_range
forecast_steps = 6
X = df['hotel_avg_daily_rate'].values
train, test = X[:len(X)-13], X[len(X)-13:]
model = AutoReg(train, lags=12)
model_fit = model.fit()
predictions = model_fit.predict(start=len(train), end=len(train)+len(test)-1, dynamic=False)
future_dates = date_range(start=df.index[-1], periods=forecast_steps + 1, freq='M')
future_predictions = model_fit.predict(start=len(train), end=len(train) + forecast_steps, dynamic=False)
pyplot.figure(figsize=(14, 6))
pyplot.plot(df.index[:-13], train, color='blue', label='Train')
pyplot.plot(df.index[-13:], test, color='green', label='Test')
pyplot.plot(df.index[-13:], predictions, color='red', label='Predictions on Test Set')
pyplot.plot(future_dates, future_predictions, color='orange', label=f'Future Predictions (+{forecast_steps} months)')
pyplot.legend()
pyplot.show()
```

11) ARIMA model

```
X = df['stationary_data'].values
train, test = X[1:len(X)-13], X[len(X)-13:]
order = (12, 0, 12)
model = ARIMA(train, order=order)
model_fit = model.fit()
predictions = model_fit.predict(start=len(train), end=len(train)+len(test)-1, typ='levels')
mae = mean_absolute_error(test, predictions)
print("Mean Absolute Error:", mae)
for i in range(len(predictions)):
    print('Predicted=%f, Expected=%f' % (predictions[i], test[i]))
rmse = sqrt(mean_squared_error(test, predictions))
print('Test RMSE: %.3f' % rmse)
pyplot.plot(test, color='green', label='Actual')
pyplot.plot(predictions, color='red', label='Predicted')
pyplot.legend()
pyplot.show()
```

12) Cross correlation heatmap

```
columns_of_interest = ['hotel_occup_rate', 'hotel_avg_daily_rate' ]
cross_corr = pd.DataFrame()
for col in columns_of_interest:
    for lag in range(1, lags + 1):
        cross_corr[f'{col}_lag{lag}'] = df[col].shift(-lag)
corr_matrix = cross_corr.corr()
plt.figure(figsize=(15, 10))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Cross-Correlation Heatmap of Variables at Different Time Lags')
plt.show()
```

13) Impulse response function using VAR model

```
columns_of_interest = ['hotel_occup_rate', 'hotel_avg_daily_rate']
def normalize(df):
    return (df - df.min()) / (df.max() - df.min())
df_normalized = normalize(df[columns_of_interest])
data_for_var = df_normalized.to_numpy()
var_model = VAR(data_for_var)
results = var_model.fit(maxlags=4, ic='aic')
num_periods = 10
plt.figure(figsize=(10, 6))
irf = results.irf(num_periods)
for i, column in enumerate(columns_of_interest):
    impulse_responses = irf.orth_irfs[:, i, i]
    plt.plot(impulse_responses, label=column)
plt.title('Impulse Response Functions')
plt.xlabel('Time')
plt.ylabel('Response')
plt.legend()
plt.show()
```

14) Granger causality test

```
columns_of_interest = ['hotel_avg_daily_rate', 'hotel_occup_rate']
data_for_test = df[columns_of_interest]
max_lag = 4
granger_test_result = grangercausalitytests(data_for_test, max_lag, verbose=True)
```

15) Johansen cointegration test

```
model1 = coint_johansen(df[['hotel_occup_rate', 'hotel_avg_daily_rate']], det_order=0, k_ar_diff=2)
eigenvalues_h = model1.eig
eigenvectors_h = model1.evec
trace_statistic_h = model1.lr1
critical_values_trace_h = model1.cvt
max_eigenvalue_statistic_h = model1.lr2
critical_values_max_eigenvalue_h = model1.cvm
print('for hotel occupancy rate and hotel avg daily rate')
print("Eigenvalues:")
print(eigenvalues_h)
print("\nEigenvectors:")
print(eigenvectors_h)
print("\nTrace Statistic and Critical Values:")
print("Trace Statistic:", trace_statistic_h)
print("Critical Values for Trace Statistic:")
print(critical_values_trace_h)
print("\nMaximum Eigenvalue Statistic and Critical Values:")
print("Maximum Eigenvalue Statistic:", max_eigenvalue_statistic_h)
print("Critical Values for Maximum Eigenvalue Statistic:")
print(critical_values_max_eigenvalue_h)
```

16) ARIMA model on hotel_occup_rate and hotel_avg_daily_rate

```
df['Date'] = pd.to_datetime(df[['Year', 'Month']].assign(day=1))
df.set_index('Date', inplace=True)
model = coint_johansen(df[['stationary_data_occup', 'hotel_avg_daily_rate']], det_order=0, k_ar_diff=2)
coint_result = model.lr1
columns_of_interest = ['stationary_data_occup', 'hotel_avg_daily_rate']
train_size = int(len(df) * 0.8)
train_data = df[columns_of_interest].iloc[:train_size]
test_data = df[columns_of_interest].iloc[train_size:]
p = 12  # AR order
d = 0  # Differencing order
q = 12  # MA order
arima_model = ARIMA(train_data['hotel_avg_daily_rate'], exog=train_data['stationary_data_occup'], order=(p, d, q))
arima_result = arima_model.fit()
test_predictions = arima_result.forecast(steps=len(test_data), exog=test_data['stationary_data_occup'])
plt.figure(figsize=(10, 5))
plt.plot(test_data.index, test_predictions, label='Predicted Test Data', linestyle='--')
plt.plot(test_data.index, test_data['hotel_avg_daily_rate'], label='Actual Test Data: hotel_avg_daily_rate')
plt.xlabel('Date')
plt.ylabel('Values')
plt.legend()
plt.title('Testing Data vs Predicted Testing Data (ARIMA Forecast for hotel_avg_daily_rate)')
plt.show()
```

17) Forecasting on hotel_occup_rate and hotel_avg_daily_rate

```
forecast_steps = 12
last_20_hotel_occup = df['stationary_data_occup'].iloc[-20:]  # Retrieve the last 20 values
future_exog = last_20_hotel_occup.tolist() * (forecast_steps // 20)  # Repeat the last 20 values based on forecast steps

remaining_steps = forecast_steps % 20
future_exog += last_20_hotel_occup.tolist()[:remaining_steps]
future_forecast = arima_result.forecast(steps=forecast_steps, exog=future_exog)
last_date = df.index[-1]
future_dates = pd.date_range(start=last_date + pd.DateOffset(months=1), periods=forecast_steps, freq='M')
plt.figure(figsize=(10, 5))
plt.plot(df.index, df['hotel_avg_daily_rate'], label='Historical Data: hotel_avg_daily_rate')
plt.xlabel('Date')
plt.ylabel('Values')
plt.title('Historical Data and Forecasted hotel_avg_daily_rate')
plt.plot(future_dates, future_forecast, label='Forecasted Data: hotel_avg_daily_rate', linestyle='--', color='red')
plt.legend()
plt.show()
```