Times Series Analysis Final Project Report

Consultation based on forecast on Air Passenger Data

Group 15

Ahmed Raza Sharif - 23PGAI0120

Akash Deshwani – 23PGAI0035

Harshada Jadhav - 23PGAI0101

Rohan Mehta – 23PGAI0001

Objective:

To consult for a company based on a forecast of the next time period.

Introduction:

This dataset consists of a time-series data about the number of passengers flown with a flight company over a period of 11 years.

Dataset Link:

https://www.kaggle.com/datasets/ashfakyeafi/air-passenger-data-for-time-series-analysis

Being a time-sensitive and mission-critical business, It is important to schedule flights as per demand.

Why did we choose this dataset?

This is a simple yet relevant dataset with 11 years of passenger data which displays a clear positive increasing trend and visible seasonality. It is real world problem where flight carriers face issues relating to underutilization of flights during off-season and over-booking in peak seasons.

We will try to reduce this issue by effectively forecasting the demand for the next time period and pro-actively schedule flights to maximize utilization and increase revenue.

Methodology:

We have to forecast the passenger demand for the next time period – 12 months in this case. For this, we will apply all the models that we have learnt in class, and select the best model for our forecast.

Python Pre-requisites:

Importing libraries

Importing Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import itertools
import statsmodels.api as sm
import statsmodels.tsa.api as smt
import pmdarima as pm
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_percentage_error, mean_squared_error, mean_absolute_error
from statsmodels.tsa.holtwinters import ExponentialSmoothing
from statsmodels.tsa.holtwinters import SimpleExpSmoothing
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.arima_model import ARIMA
from statsmodels.tsa.statespace.sarimax import SARIMAX
```

Reading csv file

Reading Dataset

```
df = pd.read_csv('AirPassengers - Original.csv')
```

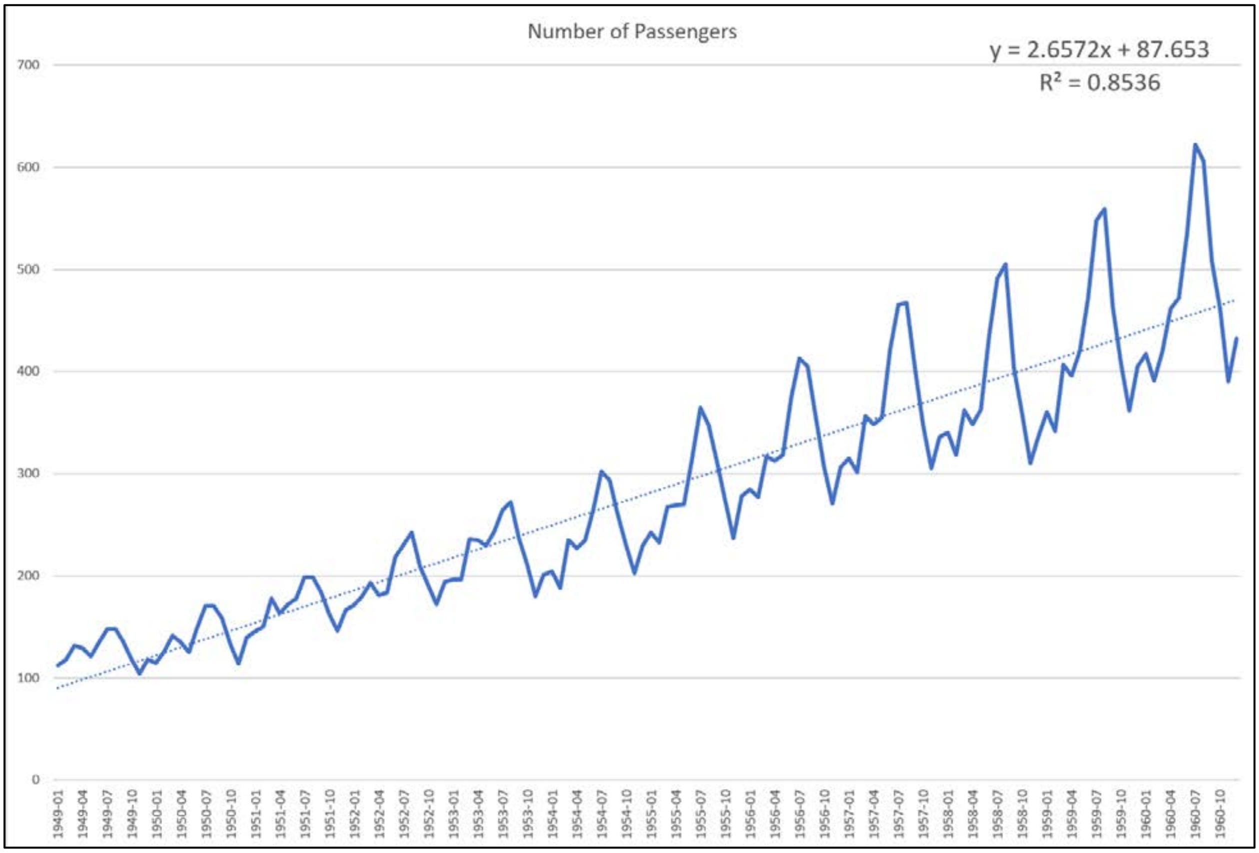
Exploratory Data Analysis:

Dataset

	Month	#Passengers
0	1949-01	112
1	1949-02	118
2	1949-03	132
3	1949-04	129
4	1949-05	121
...	...	...
139	1960-08	606
140	1960-09	508
141	1960-10	461
142	1960-11	390
143	1960-12	432

144 rows × 2 columns

Line plot vs time



Checking for null values

```
df.isnull().sum()
```

```
Month          0
#Passengers    0
dtype: int64
```

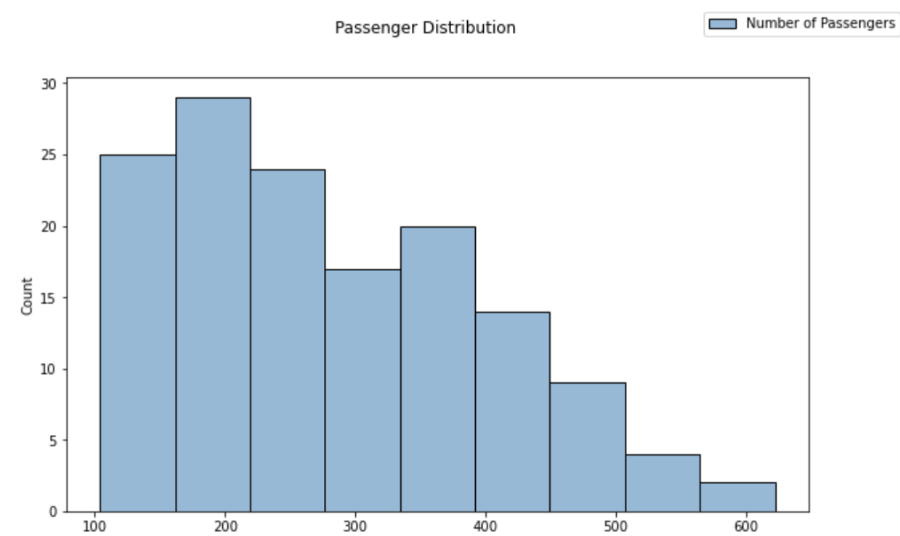
Checking basic statistics

```
df.describe()
```

	Passengers
count	144.000000
mean	280.298611
std	119.966317
min	104.000000
25%	180.000000
50%	265.500000
75%	360.500000
max	622.000000

Checking the distribution of passenger column

```
fig, ax1 = plt.subplots(1, figsize=(10, 6))
sns.histplot(data=df, ax=ax1, legend='')
plt.suptitle('Passenger Distribution')
fig.legend(['Number of Passengers'])
plt.show()
```



1. **Moving Average Model** - In time series analysis, the moving-average model (MA model), also known as moving-average process, is a common approach for modeling univariate time series. The moving-average model specifies that the output variable is cross-correlated with a non-identical to itself random-variable. The moving-average model should not be confused with the moving average, a distinct concept despite some similarities.

We have performed 12 month moving average by excel (attached along with report).

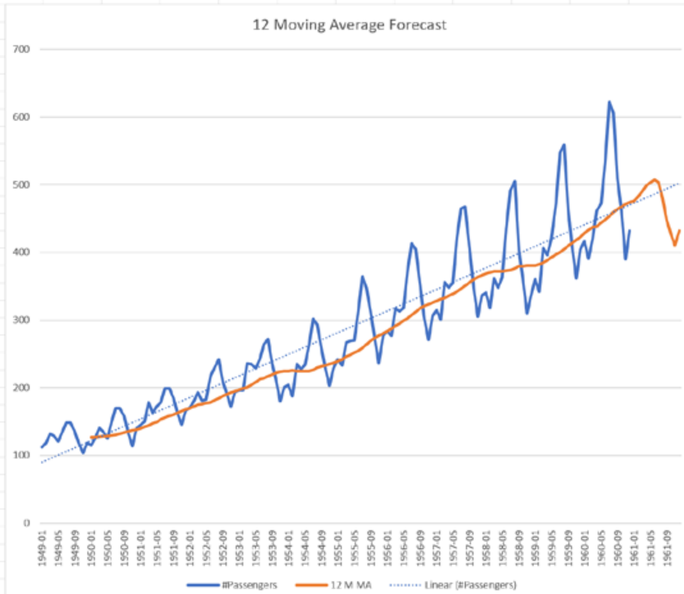
12 month MAD: 35.2481

12 month MSE: 2472.34

12 month RMSE: 49.72

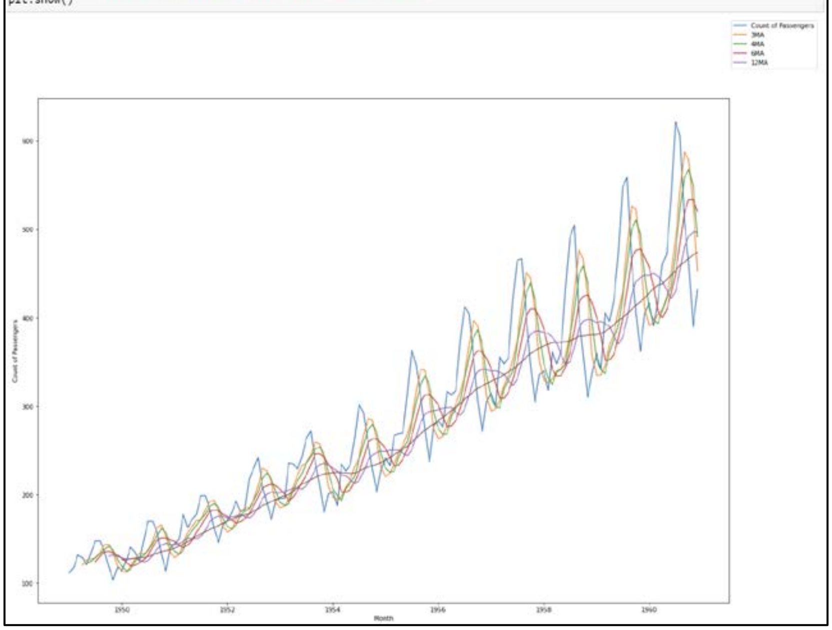
12 month MPE: 6.81%

Month	#Passengers	12 M MA	12 M Dev	Abs Dev	12M SE	MAPE	MPE
1949-01	112						
1949-02	118						
1949-03	132						
1949-04	129						
1949-05	121						
1949-06	135						
1949-07	148						
1949-08	148						
1949-09	136						
1949-10	119						
1949-11	104						
1949-12	118						
1950-01	115	126.667	-11.6667	11.6667	136.111	10.14%	-9.21%
1950-02	126	126.917	-0.91667	0.91667	0.84028	0.73%	-0.72%
1950-03	141	127.583	13.4167	13.4167	180.007	9.52%	10.52%
1950-04	135	128.333	6.66667	6.66667	44.4444	4.94%	5.19%
1950-05	125	128.833	-3.83333	3.83333	14.6944	3.07%	-2.98%
1950-06	149	129.167	19.8333	19.8333	393.361	13.31%	15.35%
1950-07	170	130.333	39.6667	39.6667	1573.44	23.33%	30.43%
1950-08	170	132.167	37.8333	37.8333	1431.36	22.25%	28.63%
1950-09	158	134	24	24	576	15.19%	17.91%
1950-10	133	135.833	-2.83333	2.83333	8.02778	2.13%	-2.09%
1950-11	114	137	-23	23	529	20.18%	-16.79%
1950-12	140	137.833	2.16667	2.16667	4.69444	1.55%	1.57%
1951-01	145	139.667	5.33333	5.33333	28.4444	3.68%	3.82%
1951-02	150	142.167	7.83333	7.83333	61.3611	5.22%	5.51%
1951-03	178	144.167	33.8333	33.8333	1144.69	19.01%	23.47%
1951-04	163	147.25	15.75	15.75	248.063	9.66%	10.70%
1951-05	172	149.583	22.4167	22.4167	502.507	13.03%	14.99%
1951-06	178	153.5	24.5	24.5	600.25	13.76%	15.96%
1951-07	199	155.917	43.0833	43.0833	1856.17	21.65%	27.63%
1951-08	199	158.333	40.6667	40.6667	1653.78	20.44%	25.68%
1951-09	184	160.75	23.25	23.25	540.563	12.64%	14.46%
1951-10	162	162.917	-0.91667	0.91667	0.84028	0.57%	-0.56%



We have also performed 3 month, 4 month, 6 month and 12 month to visualize the differences in python.

```
figure, ax1 = plt.subplots(1, figsize=(20, 15))
ax1.plot(df_1['Passengers'])
ax1.plot(df_1['3MA'])
ax1.plot(df_1['4MA'])
ax1.plot(df_1['6MA'])
ax1.plot(df_1['9MA'])
ax1.plot(df_1['12MA'])
ax1.set_xlabel('Month')
ax1.set_ylabel('Count of Passengers')
figure.legend(['Count of Passengers','3MA', '4MA', '6MA', '12MA'])
plt.show()
```

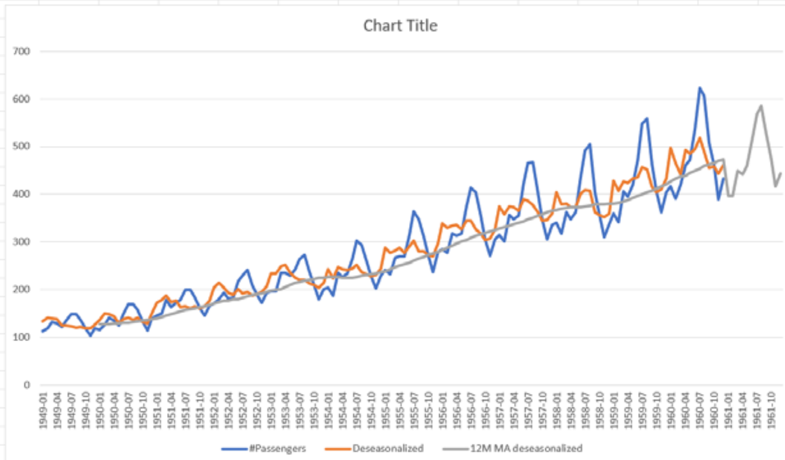


2. **Seasonal Index –** In this method, we seasonalize the data, perform moving averages and then reapply seasonality to the forecast.

We have performed the forecast with excel.

Month	#Passengers	Deseasonalized	12M MA deseasonalized
1949-01	112	133.3456483	
1949-02	118	140.3894214	
1949-03	132	138.9440955	
1949-04	129	137.9966486	
1949-05	121	124.4725292	
1949-06	135	124.5396367	
1949-07	148	123.174075	
1949-08	148	119.5233735	
1949-09	136	121.9053223	
1949-10	119	118.3116286	
1949-11	104	118.0345411	
1949-12	118	126.028362	
1950-01	115	136.9174067	127.222
1950-02	126	149.9073482	127.52
1950-03	141	148.4175565	128.313
1950-04	135	144.4150974	129.102
1950-05	125	128.5873235	129.637
1950-06	149	137.4548582	129.98
1950-07	170	141.4837348	131.056
1950-08	170	137.2903614	132.582
1950-09	158	141.6253009	134.063
1950-10	133	132.2306437	135.706
1950-11	114	129.3840163	136.866
1950-12	140	149.5251753	137.812
1951-01	145	172.6349911	139.77
1951-02	150	178.4611289	142.746
1951-03	178	187.3640075	145.126
1951-04	163	174.3678583	148.371
1951-05	172	176.9361572	150.867
1951-06	178	164.2078172	154.897
1951-07	199	165.6191954	157.126
1951-08	199	160.7104819	159.137
1951-09	184	164.9307301	161.089
1951-10	162	161.0628894	163.031
1951-11	146	165.7023366	165.434
1951-12	166	177.2941364	168.46
1952-01	171	203.5902309	170.774
1952-02	180	214.1533546	173.354
1952-03	193	203.1531093	176.328
1952-04	181	193.6232046	177.644

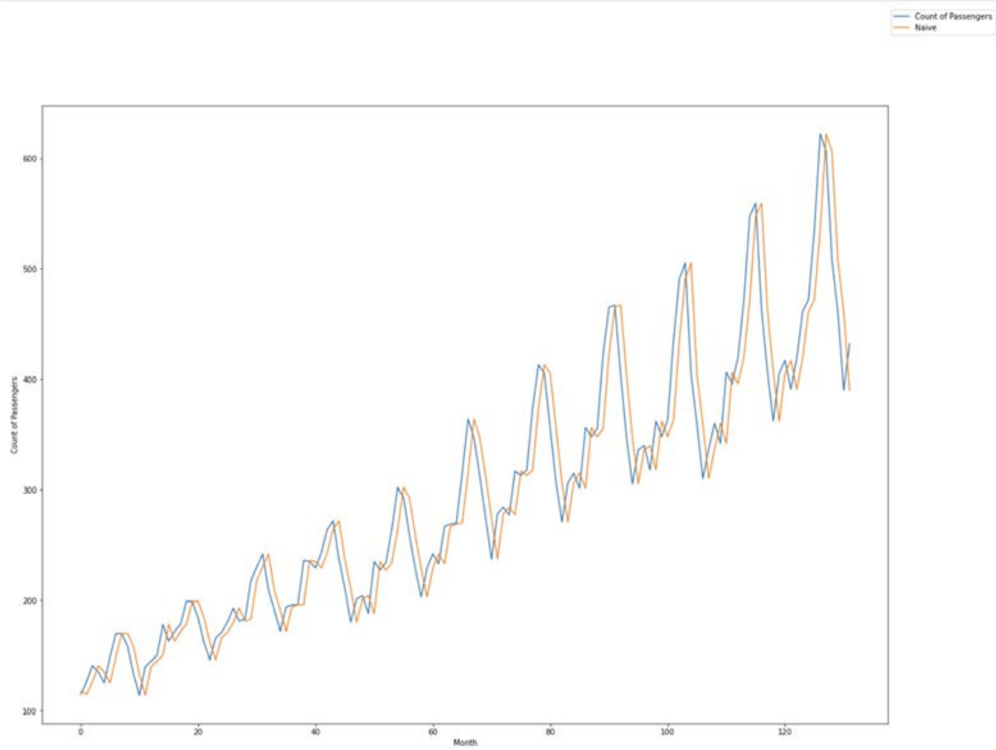
Month	1949	1950	1951	1952	1953	1954	1955	1956	1957	1958	1959	1960	Monthly Average	Seasonal Index
Jan	112	115	145	171	196	204	203	237	315	340	360	417	234.583	83.99%
Feb	118	126	150	180	196	188	229	278	301	318	342	391	234.75	84.05%
Mar	132	141	178	193	236	235	242	284	356	362	406	419	265.333	95.00%
Apr	129	135	163	181	235	227	233	277	348	348	396	461	261.083	93.48%
May	121	125	172	183	229	234	267	317	355	363	420	472	271.5	97.21%
Jun	135	149	178	218	243	264	269	313	422	435	472	535	302.75	108.40%
Jul	148	170	199	230	264	302	270	318	465	491	548	622	335.583	120.16%
Aug	148	170	199	242	272	293	315	374	467	505	559	606	345.833	123.83%
Sep	136	158	184	209	237	259	364	413	404	404	463	508	311.583	111.56%
Oct	119	133	162	191	211	229	347	405	347	359	407	461	280.917	100.58%
Nov	104	114	146	172	180	203	312	355	305	310	362	390	246.083	88.11%
Dec	118	140	166	194	201	229	274	306	336	337	405	432	261.5	93.63%
	126.667	139.667	170.167	197	225	238.917	277.083	323.083	368.417	381	428.333	476.167	279.292	100.00%



3. **Naïve forecast model –** This model is a naïve model, wherein the next forecast is assumed to be the previous value.

Method	RMSE	MAPE
Naive method	34.92	9.07

```python
figure, ax1 = plt.subplots(1, figsize=(20, 15))
ax1.plot(df_2['Passengers'])
ax1.plot(df_2['Naive'])
ax1.set_xlabel('Month')
ax1.set_ylabel('Count of Passengers')
figure.legend(['Count of Passengers','Naive', 'Naive_seasonal'])
plt.show()
```



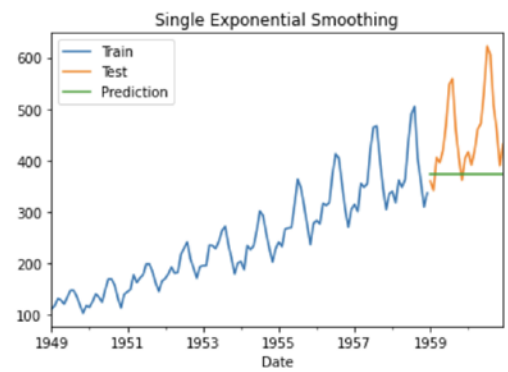
4. **Simple exponential smoothing –** In this model, we use exponential smoothing assuming no trend not seasonality

This does not provide a good forecast

MAE: 82.53

```python
ses_model = SimpleExpSmoothing(df_train).fit(smoothing_level=best_alpha)
y_pred = ses_model.forecast(24)

def plot_prediction(y_pred, label):
    df_train["Passengers"].plot(legend=True, label="Train")
    df_test["Passengers"].plot(legend=True, label="Test")
    y_pred.plot(legend=True, label="Prediction")
    plt.title(label)
    plt.show()
plot_prediction(y_pred, "Single Exponential Smoothing")
```



5. **Holt's model** - In this model, we use exponential smoothing assuming trend but not seasonality

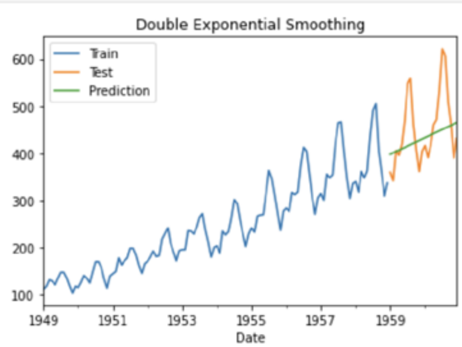
This also does not provide a good forecast

MAE: 54.1

Plotting predicted forecast with Double Exponential Smoothing.

It also does not give a good forecast.

```python
des_model = ExponentialSmoothing(df_train, trend="add").fit(smoothing_level=best_alpha, smoothing_trend=best_beta)
y_pred = des_model.forecast(24)
plot_prediction(y_pred, "Double Exponential Smoothing")
```

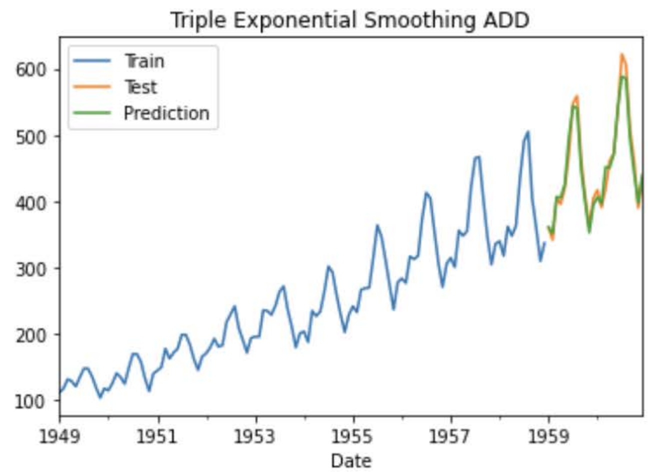


6. **Winter's model** - In this model, we use exponential smoothing assuming trend and seasonality

Winter's model is able to provide a pretty good forecast.

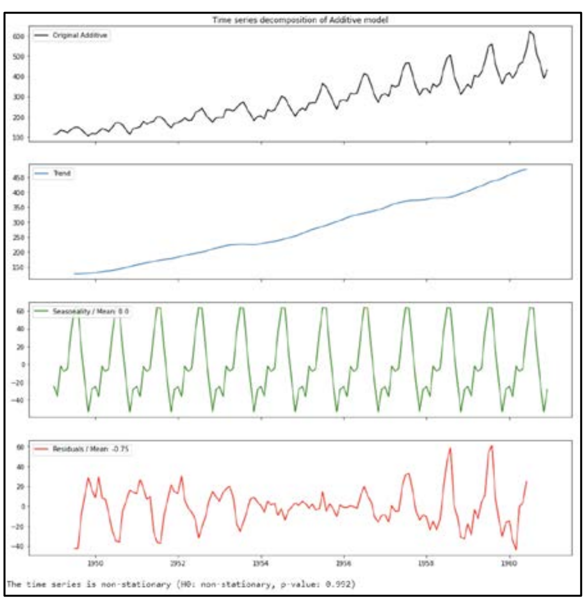
MAE: 11.99

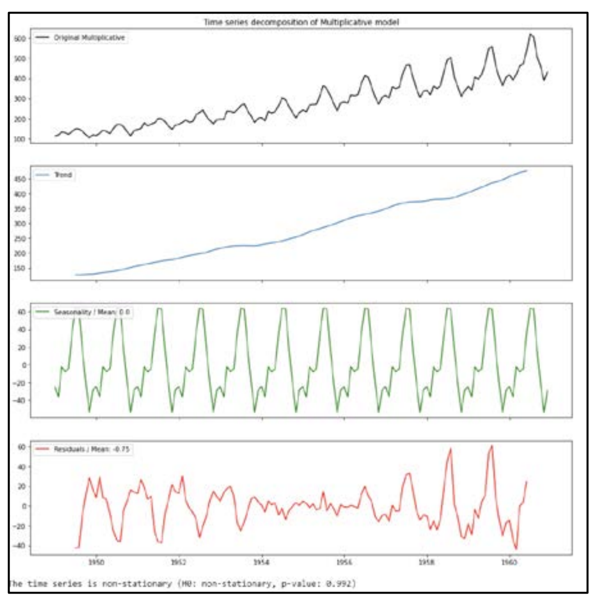
```
tes_model = ExponentialSmoothing(df_train, trend="add", seasonal="add", seasonal_periods=12)
y_pred = tes_model.forecast(24)
plot_prediction(y_pred, "Triple Exponential Smoothing ADD")
```



7. **A.R.I.M.A** - The ARIMA model is specified by three parameters: (p,d,q), where p is the number of autoregressive terms, d is the number of times the data have been differenced to make it stationary, and q is the number of moving average terms. The process of differencing involves subtracting the previous observation from the current one to remove trend or seasonality in the data.

Performed Dickey-Fuller test to check for stationarity.

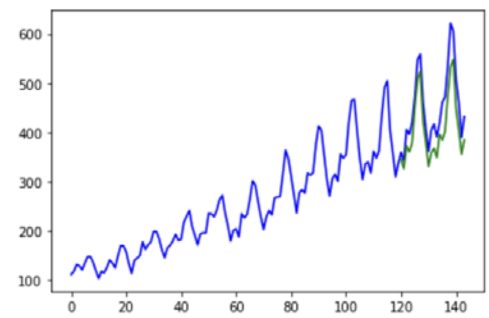




Both additive models and multiplicative models are **non-stationary** with p-value = 0.992

ARIMA Forecast

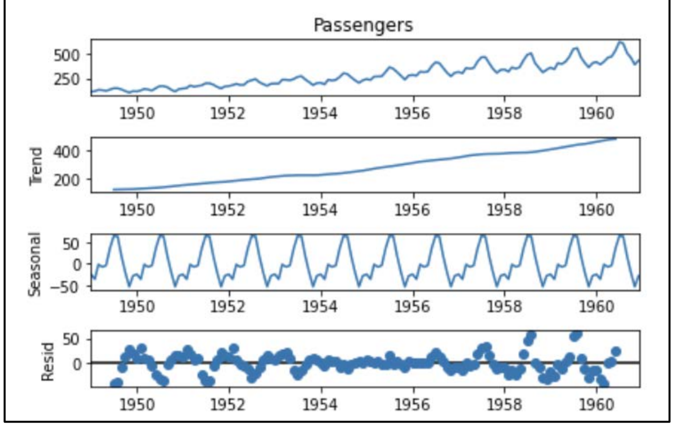
```python
model = pm.auto_arima(df_train, seasonal=True, m=12)
forecasts = model.predict(df_test.shape[0])
x = np.arange(y.shape[0])
plt.plot(x, y, c='blue')
plt.plot(x[df_train.shape[0]:], forecasts, c='green')
plt.show()
```

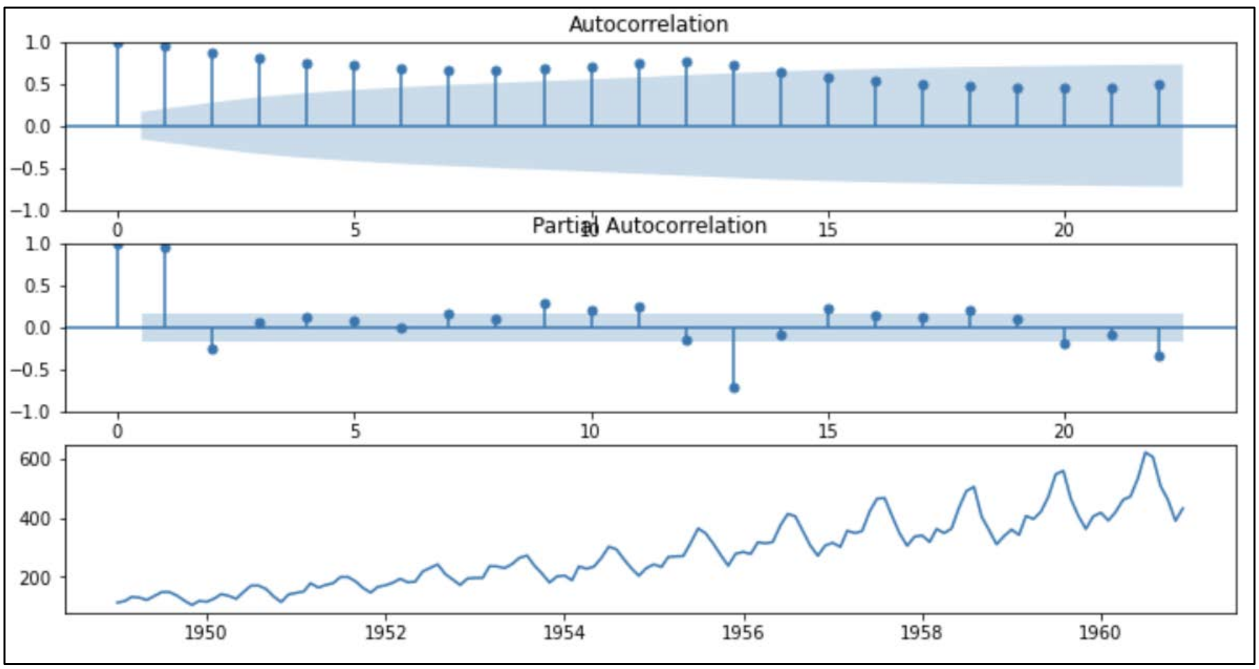


```python
model.get_params()
```

```
{'maxiter': 50,
 'method': 'lbfgs',
 'order': (2, 0, 0),
 'out_of_sample_size': 0,
 'scoring': 'mse',
 'scoring_args': {},
 'seasonal_order': (0, 1, 0, 12),
 'start_params': None,
 'suppress_warnings': True,
 'trend': None,
 'with_intercept': True}
```

```python
seasonal_decompose(x=df["Passengers"], period=12).plot()
plt.show()
```





8. S.A.R.I.M.A.X - A SARIMA model is a combination of two models: an autoregressive (AR) model and a moving average (MA) model. The seasonal component of the model is denoted by the "S" in the acronym. The "I" stands for "integrated," which refers to the use of differencing to make the time series stationary.

Checked parameters at all combinations of p, d, q.

SARIMA (Seasonal AutoRegressive Integrated Moving Average)

```
p = range(0, 3)
d = range(0, 3)
q = range(0, 3)
i = list(itertools.product(p, d, q))
seasonal_i = [(x[0], x[1], x[2], 12) for x in list(itertools.product(p, d, q))]
print('Examples of parameter combinations for Seasonal ARIMA...')
for j in i:
    for k in seasonal_i:
        print('SARIMAX: {} x {}'.format(j, k))
temp = []
for j in i:
    for k in seasonal_i:
        try:
            mod = SARIMAX(df_train, order=j, seasonal_order=k, enforce_stationarity=False, enforce_
            results = mod.fit()
            temp.append([j, k, results.aic, results.bic])
            print('SARIMA{}x{}12 - AIC:{} BIC:{}'.format(j, k, results.aic, results.bic))
        except:
            continue
```

```
SARIMA(2, 2, 2)x(2, 1, 0, 12)12 - AIC:624.119097784893 BIC:640.7932842276102

C:\Users\coola\anaconda3\lib\site-packages\statsmodels\base\model.py:604: ConvergenceWarning: Maxim
tion failed to converge. Check mle_retvals
  warnings.warn("Maximum Likelihood optimization failed to "

SARIMA(2, 2, 2)x(2, 1, 1, 12)12 - AIC:620.9174378310843 BIC:639.9736509084753
SARIMA(2, 2, 2)x(2, 1, 2, 12)12 - AIC:614.8450943025557 BIC:636.1701249747589
SARIMA(2, 2, 2)x(2, 2, 0, 12)12 - AIC:533.5202727379693 BIC:549.0568266742021

C:\Users\coola\anaconda3\lib\site-packages\statsmodels\base\model.py:604: ConvergenceWarning: Maxim
tion failed to converge. Check mle_retvals
  warnings.warn("Maximum Likelihood optimization failed to "

SARIMA(2, 2, 2)x(2, 2, 1, 12)12 - AIC:522.6098657180412 BIC:540.3659273594501
SARIMA(2, 2, 2)x(2, 2, 2, 12)12 - AIC:3129.429867260472 BIC:3149.2721008349904
```

Received best results (Least AIC, BIC) at (2,1,1), (0,2,1,12)

SARIMAX Summary Table

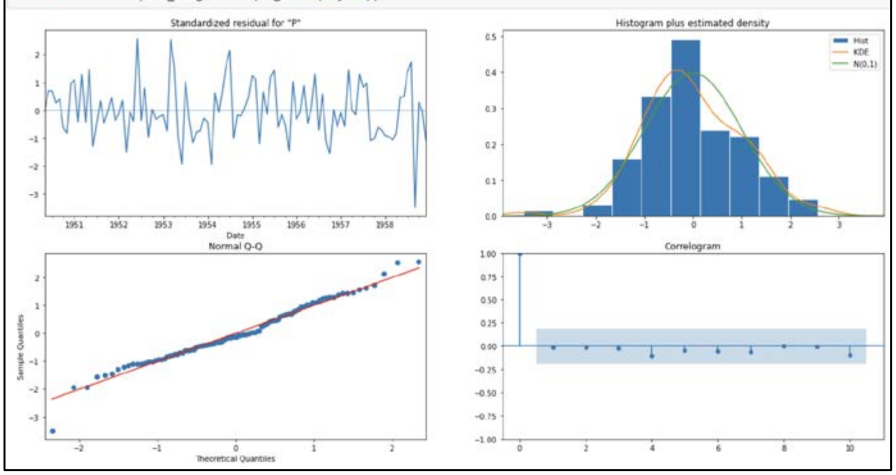
```python
# Best Value, Least AIC BIC at (2,1,1),(0,2,1,12)
sarima_df = df_train
sarimod = SARIMAX(sarima_df, order = (2, 1, 1), seasonal_order=(0, 2, 1, 12),enforce_stati
modelfit = sarimod.fit()
print(modelfit.summary())
```

```
                                     SARIMAX Results
==========================================================================================
Dep. Variable:                        Passengers   No. Observations:                  120
Model:             SARIMAX(2, 1, 1)x(0, 2, 1, 12)   Log Likelihood                -305.621
Date:                           Mon, 09 Jan 2023   AIC                            621.243
Time:                                   01:56:38   BIC                            633.215
Sample:                               01-01-1949   HQIC                           626.046
                                    - 12-01-1958
Covariance Type:                             opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.8921      0.836     -1.067      0.286      -2.531       0.747
ar.L2         -0.1616      0.350     -0.462      0.644      -0.847       0.524
ma.L1          0.5617      0.833      0.674      0.500      -1.071       2.194
ma.S.L12      -1.0000   1298.353     -0.001      0.999   -2545.725    2543.725
sigma2        89.6919   1.16e+05      0.001      0.999   -2.28e+05    2.28e+05
===================================================================================
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):                 1.15
Prob(Q):                              0.97   Prob(JB):                         0.56
Heteroskedasticity (H):               0.65   Skew:                             0.04
Prob(H) (two-sided):                  0.26   Kurtosis:                         3.58
===================================================================================
```

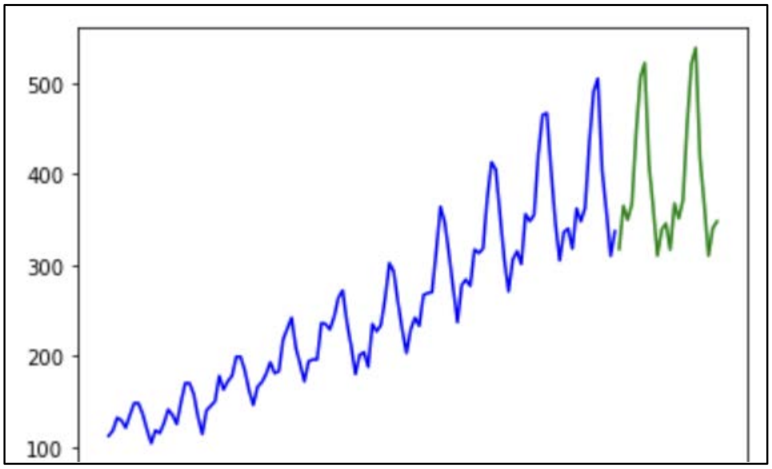
```python
mape = mean_absolute_percentage_error(y[df_train.shape[0]:], forecasts)
mse = mean_squared_error(y[df_train.shape[0]:], forecasts)
mae = mean_absolute_error(y[df_train.shape[0]:], forecasts)
print("MAPE: ", mape)
print("MSE: ", mse)
print("MAE: ", mae)
```

```
MAPE:  0.1497064200742201
MSE:  7177.538832808758
MAE:  69.62434007595839
```

```python
result = modelfit.plot_diagnostics(figsize=(20, 10))
```



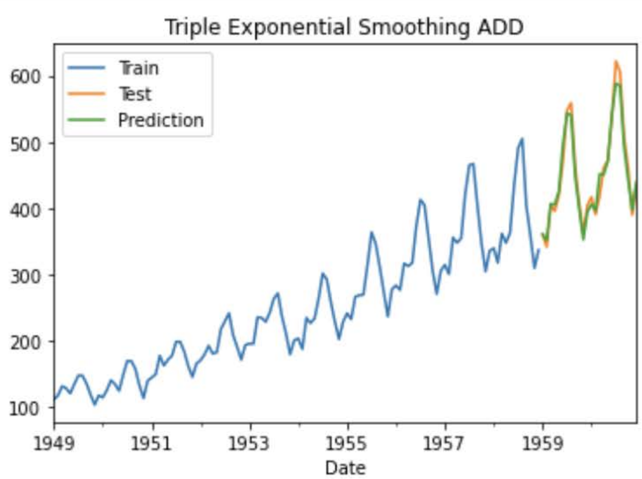
SARIMAX Forecast: It is seen that the features of the time series (blue) have been captured really well in the forecast (green)



Which is the best model and why?

Winter’s Exponential Forecast has provided the best forecast as it is optimized for series having level, trend and seasonality. Also, it has given the least error (MAE = 11.99), hence we will proceed with it.

```
tes_model = ExponentialSmoothing(df_train, trend="add", seasonal="add", seasonal_periods=12)
y_pred = tes_model.forecast(24)
plot_prediction(y_pred, "Triple Exponential Smoothing ADD")
```



Consultation

From the forecast, it is visible that there is a trend in the months of July and December. It is pretty evident as this is the holiday season.

1. For the months of July and December where peak season of demand is seen, It is recommended to rent more aircrafts so as to meet the demand and not invest in fixed assets.
2. For the month of August to November where there is a steep decline in demand, we recommend to offer discounts and incentives for customers to travel. Also, reduce the frequency of flights to improve resource utilization.