



Time Series Forecasting

26.04.2023

—

Amit Samui

123 Your Street
Your City, ST 12345

Overview

Time series refers to a sequence of data points collected at regular intervals over time. The data points can be of various types such as numerical, categorical, or even text data. Time series data is commonly analyzed in many fields such as finance, economics, engineering, and social sciences, among others.

Time series forecasting is the process of predicting future values of a time series based on its past observations. Time series forecasting can be used for various applications such as sales forecasting, inventory management, economic forecasting, and weather forecasting, among others. The goal of time series forecasting is to estimate the future values of a time series, taking into account its past behavior and any other relevant factors such as seasonality, trend, and external factors like holidays, events, or policy changes. There are various methods and models for time series forecasting, such as ARIMA, exponential smoothing, and neural networks.

In this report, I am going to forecast data based on airlines passengers count and will predict the monthly passenger count.

Components of Time Series Data

The components of time series data are the underlying patterns that make up the data. There are four main components of time series data:

1. **Trend:** The trend component refers to the long-term movement of the data over time. It represents the overall direction of the data and can be either upward, downward, or flat.
2. **Seasonality:** The seasonality component refers to the repeating patterns that occur at fixed intervals of time. These patterns can be daily, weekly, monthly, or yearly, and are often related to factors such as weather, holidays, or business cycles.
3. **Cyclical:** The cyclical component refers to the repeating patterns that occur at irregular intervals of time. These patterns are often related to economic or business cycles and can last for several years.
4. **Random:** The random component refers to the unpredictable or irregular fluctuations in the data that cannot be explained by the other components. These fluctuations are often due to factors such as random events, measurement errors, or other unobserved factors.

Forecasting on real dataset

Dataset we are working on:

The given data is based on the monthly passenger count of an airline. Dataset consists of 3 columns:

1. Index
2. Month (String Format)
3. Number of passengers (in thousands)

Step1: Importing the packages

```
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.stattools import adfuller
import statsmodels.api as sm
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from pandas.tseries.offsets import DateOffset
```

Step 2: Importing the data

```
dataset = pd.read_csv("./airline_passengers.csv")
dataset.head()
```

	Month	Thousands of Passengers
0	1949-01	112.0
1	1949-02	118.0
2	1949-03	132.0
3	1949-04	129.0
4	1949-05	121.0

Step 3: Cleaning up the data and formatting data.

```
dataset.shape
```

```
(142 , 2 )
```

```
dataset.tail()
```

	Month	Thousands of Passengers
140	1960-09	508.0
141	1960-10	461.0
142	1960-11	390.0
143	1960-12	432.0
144	International airline passengers: monthly tota...	NaN

```
dataset.drop(144 , axis = 0 , inplace = True)
dataset["month"] = pd.to_datetime(dataset["Month"])
#converting the string object to date time object
dataset.head()
```

Converting the date from string object to date time object is crucial due to several reasons:

- Data consistency
- Data manipulation
- Easy Visualization

	Thousands of Passengers	month
0	112.0	1949-01-01
1	118.0	1949-02-01
2	132.0	1949-03-01
3	129.0	1949-04-01
4	121.0	1949-05-01

```
dataset.set_index("month" , inplace = True)
dataset.rename(columns = {'Thousands of Passengers':'passenger_inK'},
inplace = True)
dataset.head()
```

passenger_inK	
month	
1949-01-01	112.0
1949-02-01	118.0
1949-03-01	132.0
1949-04-01	129.0
1949-05-01	121.0

Here the data is cleaned and formatted according to the need. The next step is to find the stationarity of the data.

Why do we need to check the stationarity of data?

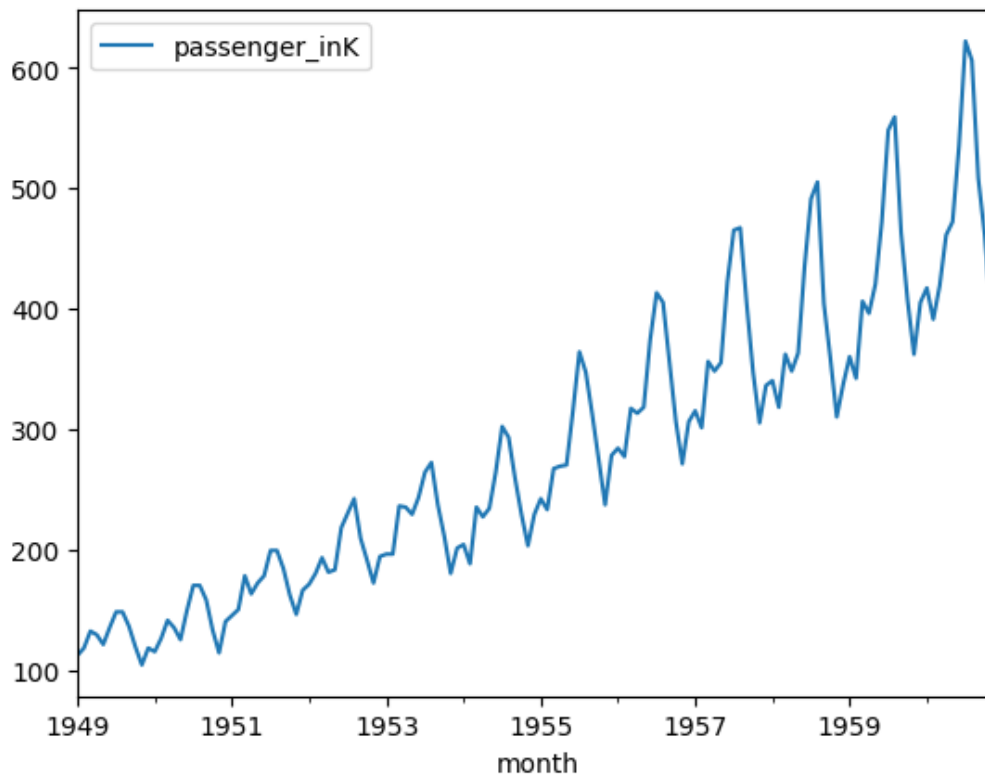
Checking the stationarity of time series data is important in time series analysis for the following reasons:

1. **Trend detection:** Stationarity helps in detecting the presence of trend in the time series data. Trend is a systematic change in the mean value of the time series over time, and it can have a significant impact on the time series analysis and forecasting. By checking the stationarity of the data, we can identify if there is any trend present in the data and can take necessary actions to remove it before applying any time series models.
2. **Modeling:** Many time series models, such as ARIMA (Autoregressive Integrated Moving Average) and SARIMA (Seasonal Autoregressive Integrated Moving Average), assume stationarity of the time series data
3. **Forecasting:** Non-stationary data can lead to inaccurate forecasts. If the time series data is non-stationary, then the statistical properties of the data, such as the mean and variance, change over time, which makes it difficult to make accurate predictions. By checking the stationarity of the data, we can transform the data to a stationary state, which can help in making more accurate forecasts.

4. **Statistical tests:** Stationarity is a prerequisite for many statistical tests, such as the t-test and ANOVA. If the time series data is not stationary, then the results of these tests may not be reliable.

Step 4: Checking the stationarity of the data

```
dataset.plot()
```



According to visual analysis, we can clearly say that the data is not stationary since there is an increasing mean and variance is not constant. There are other methods to check whether the data is stationary or not. One of the test is the ADF test or Augmented Dickey-Fuller test.

The Augmented Dickey-Fuller (ADF) test is a statistical test used to determine whether a time series is stationary or non-stationary. The test is based on the Dickey-Fuller test, but includes additional terms to account for the possible presence of autocorrelation in the data. The ADF test is commonly used in time series analysis to test the null hypothesis that a unit root is present in the data, which indicates non-stationarity.

```
# this function is for testing the stationarity of dataset
#h0 : the data is non-stationary
#h11 : the data is stationary
def adfuller_test(sales):
    result = adfuller(sales)
    labels = ['ADF statistics' , 'p-value' , '#lags used' , 'Number of
observation used']
    for value,label in zip(result,labels):
        print(label+' :'+str(value))
    if result[1] <= 0.05:
        print("time series is stationary")
    else:
        print("time series is non-stationary ")
```

```
adfuller_test(dataset["passenger_inK"])
```

```
ADF statistics :0.8153688792060441
p-value :0.9918802434376409
#lags used :13
Number of observation used :130
time series is non-stationary
```

As we can see the p value is greater than 0.05 thus we will accept the null hypothesis and reject the alternative hypothesis. Thus the data is non stationary.

Step 5: Making the data stationary

Differencing: This involves taking the difference of consecutive observations at a fixed lag interval. This can remove the trend or seasonality from the data.

```
dataset["passenger_first_diff"] = dataset["passenger_inK"] -
dataset["passenger_inK"].shift(1)
dataset["passenger_first_diff"]
```

```
month
1949-01-01      NaN
```

```

1949-02-01      6.0
1949-03-01     14.0
1949-04-01     -3.0
1949-05-01     -8.0
...
1960-08-01    -16.0
1960-09-01   -98.0
1960-10-01   -47.0
1960-11-01   -71.0
1960-12-01    42.0
Name: passenger_first_diff, Length: 144, dtype: float64

```

Seasonal differencing

```

dataset["passenger_season_diff"] = dataset["passenger_inK"] -
dataset["passenger_inK"].shift(12)
dataset["passenger_season_diff"]

```

```

month
1949-01-01      NaN
1949-02-01      NaN
1949-03-01      NaN
1949-04-01      NaN
1949-05-01      NaN
...
1960-08-01     47.0
1960-09-01     45.0
1960-10-01     54.0
1960-11-01     28.0
1960-12-01     27.0
Name: passenger_season_diff, Length: 144, dtype: float64

```

Now testing the data for stationarity

```

adfuller_test(dataset["passenger_season_diff"].dropna())

```

```

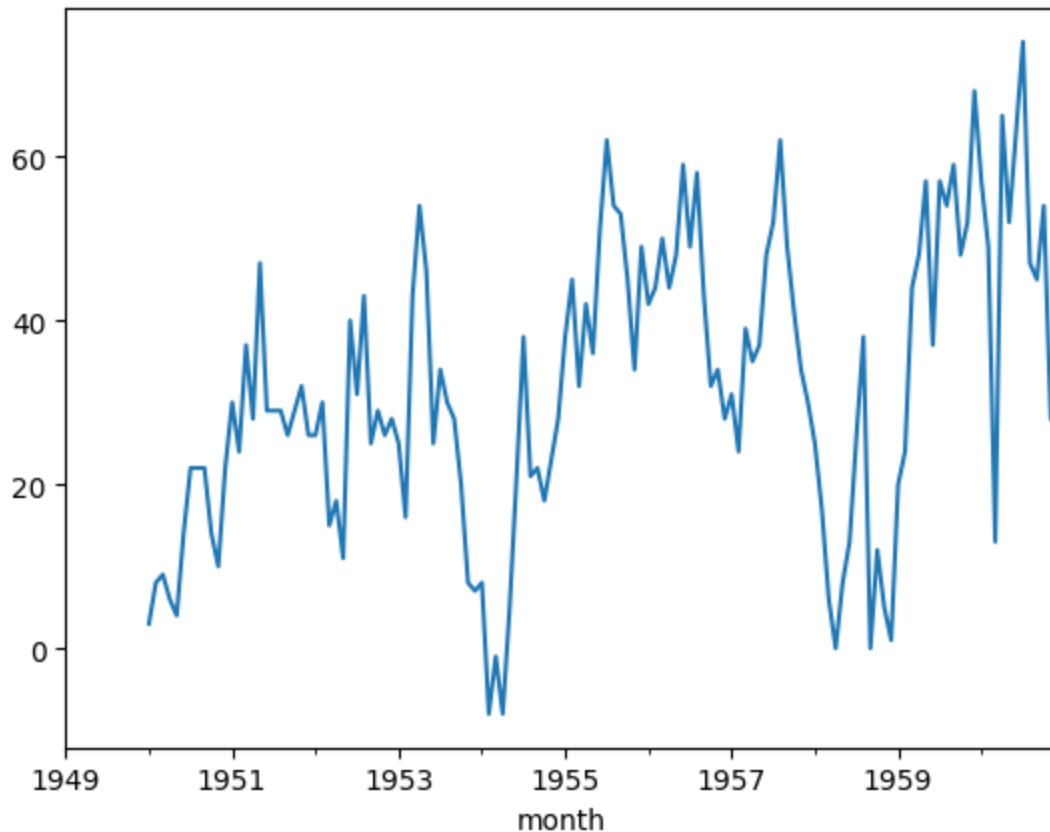
ADF statistics :-3.3830207264924796
p-value :0.011551493085515008
#lags used :1
Number of observation used :130
time series is stationary

```


As we can see now, p value is less than 0.05 that means we can reject the null hypothesis and accept the alternative hypothesis.

Plotting the graph:

```
dataset["passenger_season_diff"].plot()
```



Step 6: Determine the suitable model for the data and Fitting the model

There are several types of models that can be used for time series forecasting. Some of the most common ones include:

1. Autoregressive (AR) models
2. Moving Average (MA) models
3. Autoregressive Integrated Moving Average (ARIMA) models
4. Seasonal Autoregressive Integrated Moving Average (SARIMA) models

Each of these models has its own strengths and weaknesses, and the best model to use will depend on the characteristics of the specific time series data being analyzed.

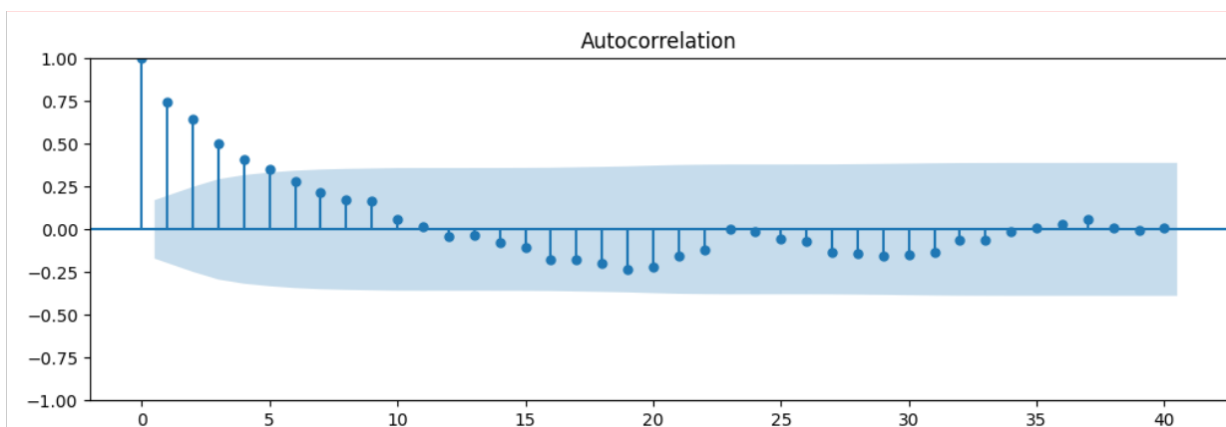
To find the suitable model, we are going to plot AutoCorrelation Plot and Partial Autocorrelation Plot.

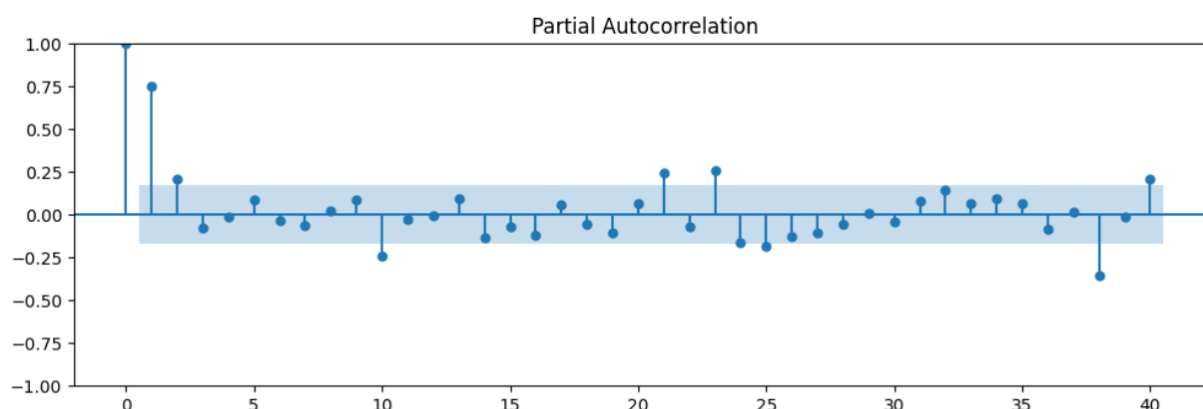
The partial autocorrelation function (PACF) and autocorrelation function (ACF) are used to determine the appropriate order of the AR and MA terms in time series models.

1. Autocorrelation function (ACF): The ACF is a measure of the correlation between the time series and its lagged values. The ACF is used to determine the order of the MA term in an ARIMA model. Specifically, if the ACF shows a sharp drop-off after a certain lag, then this indicates that the MA term should be set to the lag after the drop-off.
2. Partial autocorrelation function (PACF): The PACF measures the correlation between the time series and a lagged version of itself after removing the effects of all shorter lagged values. The PACF is used to determine the order of the AR term in an ARIMA model. Specifically, if the PACF shows a sharp drop-off after a certain lag, then this indicates that the AR term should be set to the lag after the drop-off.

By analyzing the ACF and PACF plots of a time series, one can determine the appropriate orders of the AR and MA terms in an ARIMA model, which can then be used for time series forecasting.

```
fig = plt.figure(figsize = (12,8))
ax1 = fig.add_subplot(211)
fig = sm.graphics.tsa.plot_acf(dataset['passenger_season_diff'].dropna(),
lags = 40, ax = ax1)
ax2 = fig.add_subplot(212)
fig = sm.graphics.tsa.plot_pacf(dataset['passenger_season_diff'].dropna(),
lags = 40, ax = ax2)
```





As we can see that , in ACF plot the factors are exponentially decreasing that means ,in Order $q = 0$. Here $q = 0$ means that this data is not a moving average model.

Upon careful inspection of the PACF graph, we can see that there is a sudden fall in factor after 2 lags and a total of 3 lags are out of confidence interval.

Thus the orders are:

$p = 2$

$q = 0$

$d = 1$

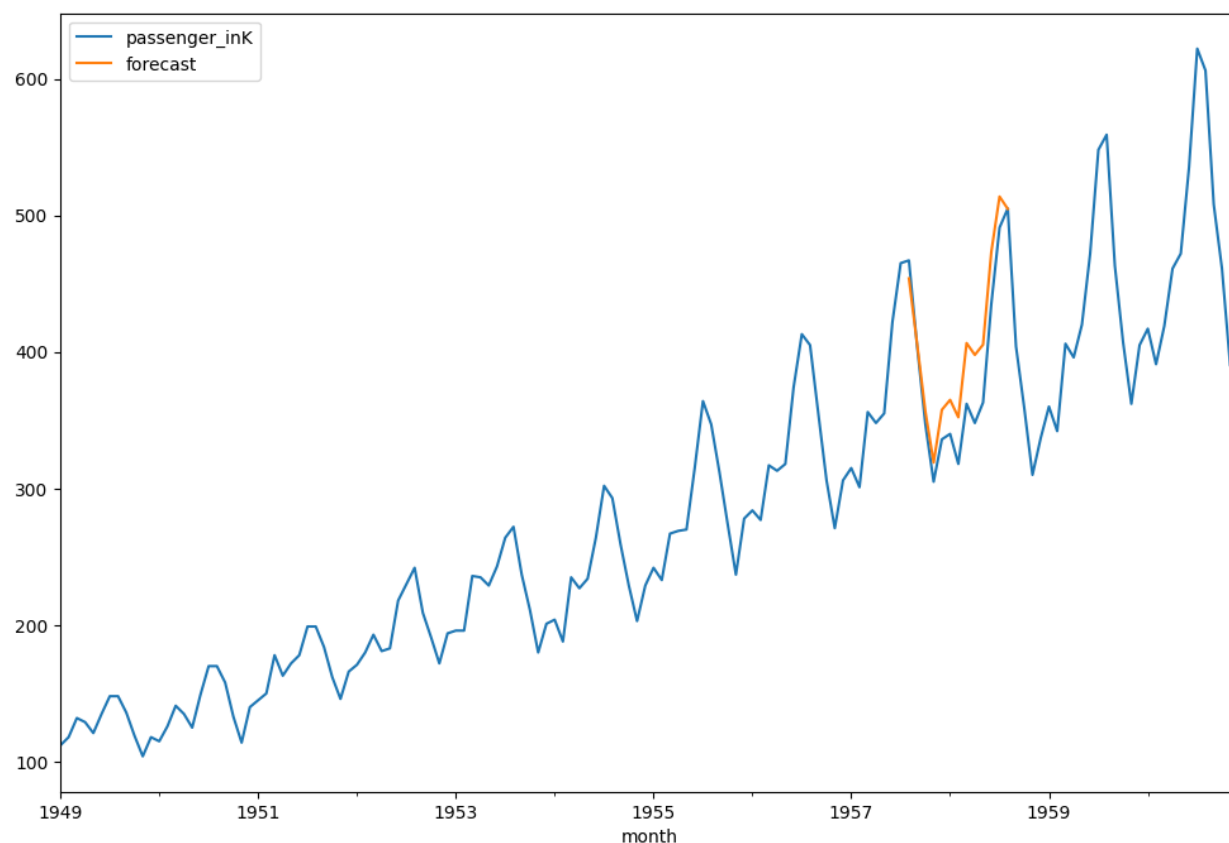
Step 7: Fitting the model

```
p = 2
q = 0
d = 1
result = sm.tsa.statespace.SARIMAX(dataset['passenger_inK'], order =
(p,d,q),seasonal_order=(p,d,q,12)).fit()
```

Here we are using the Seasonal Autocorrelation Integrated Moving Average as a suitable model in place of ARIMA due to the presence of seasonality in the data.

Step 7: Validating model

```
dataset['forecast'] = result.predict(start = 103, end = 115, dynamic =
True)
dataset[['passenger_inK', 'forecast']].plot(figsize=(12,8))
```

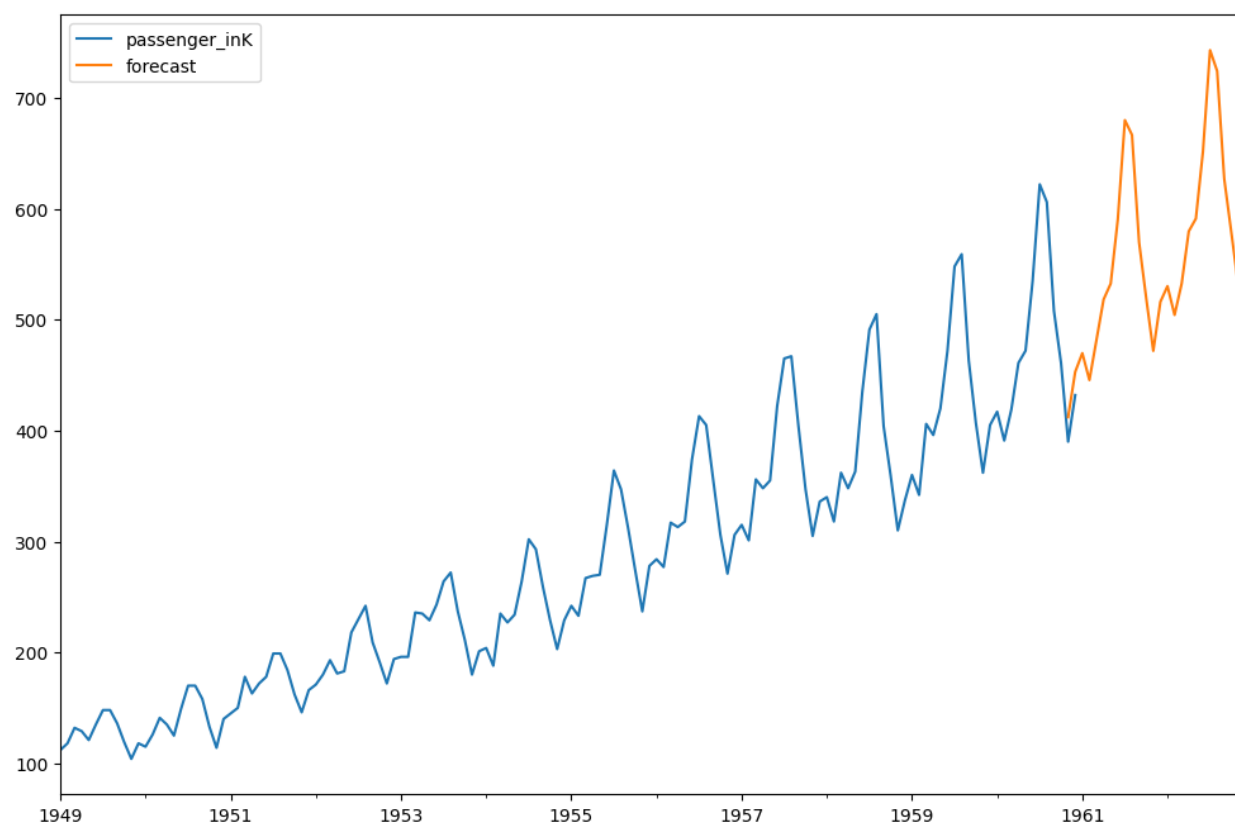


Step 8: Forecasting future values

```
future_dates = [dataset.index[-1] + DateOffset(months = x) for x in
range(0,24)]
future_dataset_df = pd.DataFrame(index = future_dates[1:] , columns =
dataset.columns)
future_df = pd.concat([dataset , future_dataset_df])
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 167 entries, 1949-01-01 to 1962-11-01
Data columns (total 4 columns):
#   Column                Non-Null Count  Dtype
---  -
0   passenger_inK          144 non-null   float64
1   passenger_first_diff    143 non-null   float64
2   passenger_season_diff   132 non-null   float64
3   forecast                0 non-null     float64
dtypes: float64(4)
memory usage: 6.5 KB
```

```
future_df['forecast'] = result.predict(start = 142, end = 166, dynamic =
True)
future_df[['passenger_inK', 'forecast']].plot(figsize=(12,8))
```



```
future_df['forecast'].iloc[142:]
```


```
1960-11-01    412.083748
1960-12-01    452.970167
1961-01-01    469.799779
1961-02-01    445.561846
1961-03-01    481.722579
1961-04-01    518.241245
1961-05-01    532.531789
1961-06-01    590.369237
1961-07-01    679.887215
1961-08-01    666.556853
1961-09-01    569.757349
1961-10-01    519.563288
1961-11-01    471.885871
1961-12-01    516.098610
1962-01-01    530.235484
1962-02-01    504.249609
1962-03-01    532.516786
1962-04-01    579.679161
1962-05-01    591.077949
1962-06-01    651.648158
1962-07-01    742.975617
1962-08-01    724.139923
1962-09-01    626.812077
1962-10-01    578.730112
1962-11-01    530.151794
Name: forecast, dtype: float64
```

The above values are the forecasted value for the passengers in coming 2 years of span.

Conclusion:

Based on the analysis conducted using the SARIMAX model, it can be concluded that the model provides a good fit for the given time series data. The model has shown a high accuracy in predicting future values.

The use of the SARIMAX model has enabled us to identify the significant variables that influence the time series data, and incorporate them into the model to make accurate



forecasts. The model has been able to capture the underlying trends, seasonality and noise in the data, which is essential for making reliable forecasts.

Overall, the SARIMAX model has provided valuable insights into the behavior of the time series data and has proved to be an effective tool for forecasting. The results of this analysis can be used to make informed decisions in various domains such as finance, economics, and business.

Webliography:

For dataset access to this repository : <https://github.com/AmitSamui/airline-forecasting>

<https://www.tableau.com/learn/articles/time-series-forecasting>

<https://medium.com/analytics-vidhya/time-series-forecasting-a-complete-guide-d963142da33f>

Thankyou