bap-exp7

April 17, 2025

EXP 7 : Forecasting using ARIMA

Name : Harsh Warghade

UID : 2022700069

Class : CSE-DS

1 Import Necessary Libraries

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

from statsmodels.tsa.stattools import adfuller
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.arima.model import ARIMA
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

import warnings
warnings.filterwarnings("ignore")
```

2 Load the Time Series Dataset

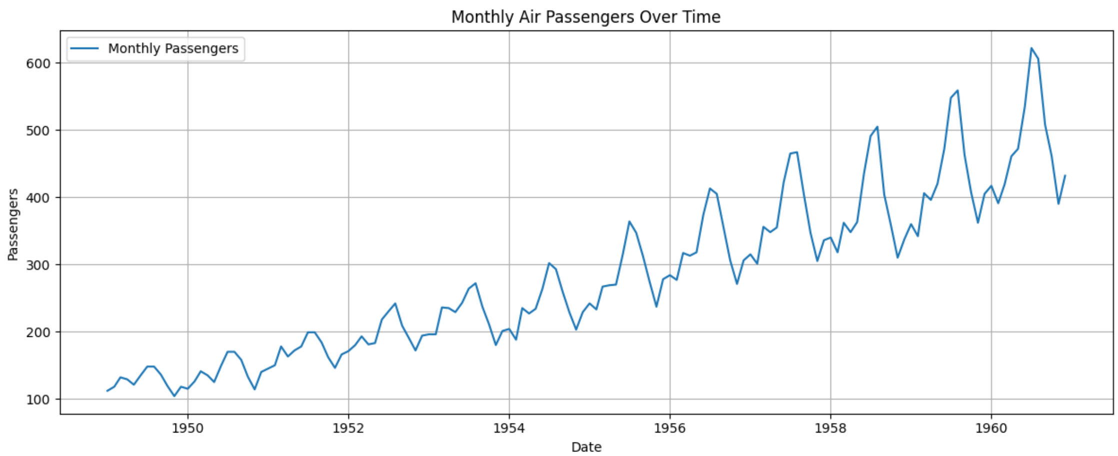
[4]:
```
# Q2: Load the Time Series Dataset
file_path = "/content/AirPassengers.csv"
df = pd.read_csv(file_path, parse_dates=["Month"], index_col="Month")
df.head()
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 144 entries, 1949-01-01 to 1960-12-01
Data columns (total 1 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   #Passengers  144 non-null    int64
```

```
dtypes: int64(1)
memory usage: 2.2 KB
```

3 Visualize the Time Series

[5]:
```
plt.figure(figsize=(12, 5))
plt.plot(df["#Passengers"], label="Monthly Passengers")
plt.title("Monthly Air Passengers Over Time")
plt.xlabel("Date")
plt.ylabel("Passengers")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```



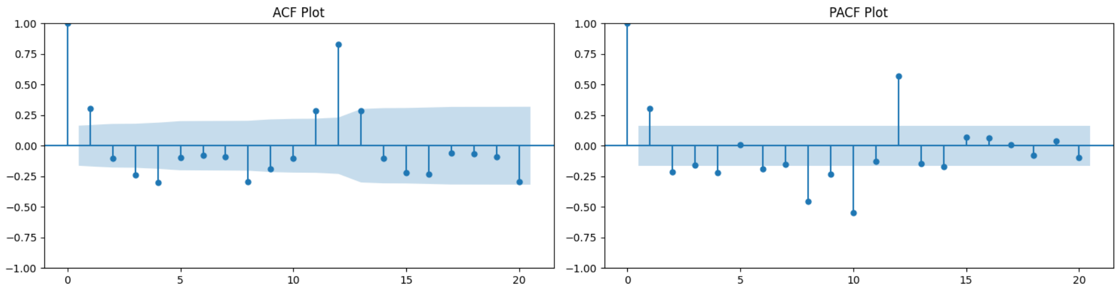
4 Check for Stationarity using ADF Test

[6]:
```
# Perform Augmented Dickey-Fuller test on original data
adf_result = adfuller(df["#Passengers"])
adf_output = {
    "ADF Statistic": adf_result[0],
    "p-value": adf_result[1],
    "Used Lags": adf_result[2],
    "Number of Observations": adf_result[3]
}

# Apply first differencing
df_diff = df["#Passengers"].diff().dropna()

# Re-run ADF on differenced data
```

```
adf_diff_result = adfuller(df_diff)
adf_diff_output = {
    "ADF Statistic": adf_diff_result[0],
    "p-value": adf_diff_result[1],
    "Used Lags": adf_diff_result[2],
    "Number of Observations": adf_diff_result[3]
}
```

5 Determine ARIMA Parameters (p, d, q) using ACF and PACF plots

[7]:
```
fig, axes = plt.subplots(1, 2, figsize=(15, 4))
plot_acf(df_diff, ax=axes[0], lags=20)
axes[0].set_title("ACF Plot")
plot_pacf(df_diff, ax=axes[1], lags=20)
axes[1].set_title("PACF Plot")
plt.tight_layout()
plt.show()
```



6 Train-Test Split (80/20)

[8]:
```
train_size = int(len(df) * 0.8)
train, test = df["#Passengers"][:train_size], df["#Passengers"][train_size:]
```

7 Fit ARIMA Model - based on ACF and PACF we'll choose p=2, d=1, q=2 as a starting point

[9]:
```
model = ARIMA(train, order=(2, 1, 2))
model_fit = model.fit()
```

8 Forecast and Compare

[10]:
```
forecast = model_fit.forecast(steps=len(test))
forecast = pd.Series(forecast, index=test.index)
```

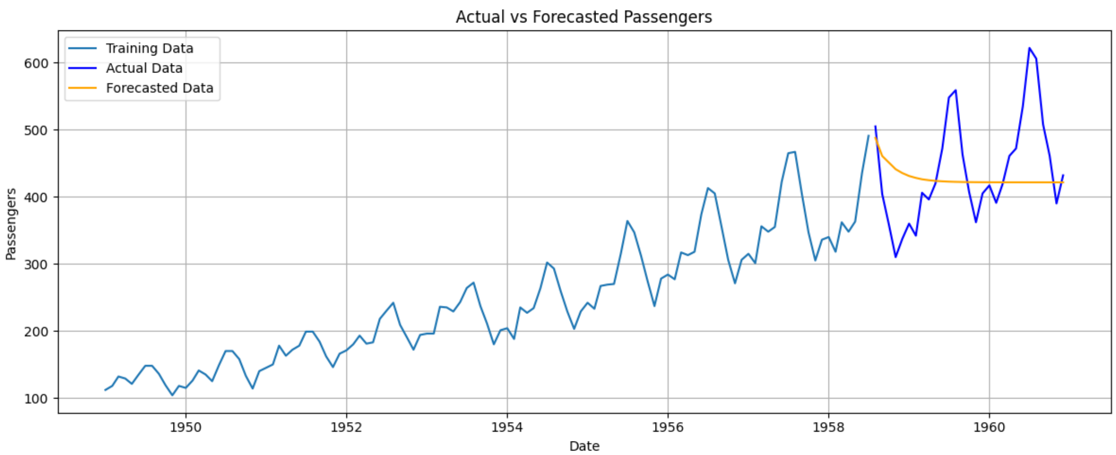
9 Evaluate the Model

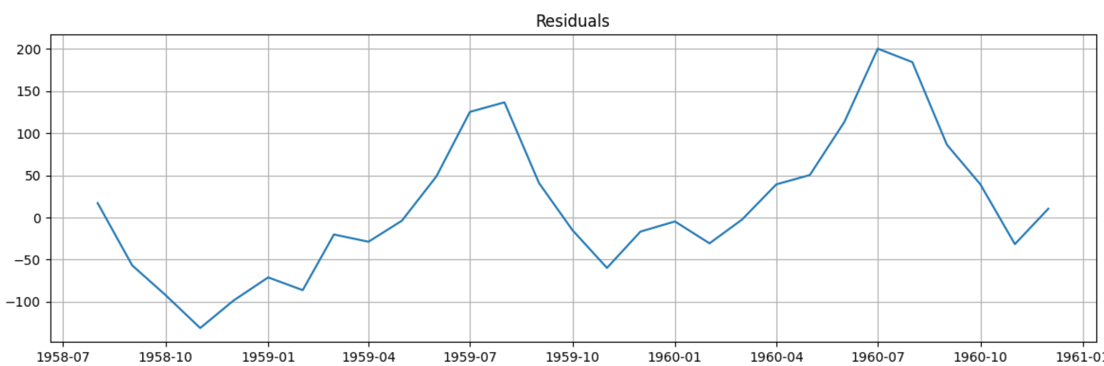
[12]:
```
mae = mean_absolute_error(test, forecast)
rmse = mean_squared_error(test, forecast)
mape = np.mean(np.abs((test - forecast) / test)) * 100
r2 = r2_score(test, forecast)
```

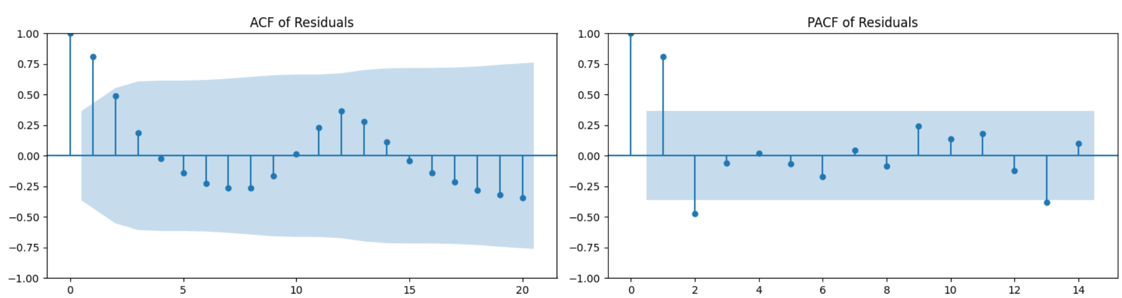
10 Visualize Results

[14]:
```
plt.figure(figsize=(12, 5))
plt.plot(train, label="Training Data")
plt.plot(test, label="Actual Data", color="blue")
plt.plot(forecast, label="Forecasted Data", color="orange")
plt.title("Actual vs Forecasted Passengers")
plt.xlabel("Date")
plt.ylabel("Passengers")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()

# Residual Plot
residuals = test - forecast
plt.figure(figsize=(12, 4))
plt.plot(residuals)
plt.title("Residuals")
plt.grid(True)
plt.tight_layout()
plt.show()

# ACF and PACF of Residuals
fig, axes = plt.subplots(1, 2, figsize=(15, 4))
plot_acf(residuals, ax=axes[0], lags=20)
axes[0].set_title("ACF of Residuals")
plot_pacf(residuals, ax=axes[1], lags=14)
axes[1].set_title("PACF of Residuals")
plt.tight_layout()
plt.show()

# Return evaluation metrics
```

```
{
    "ADF Test (Original)": adf_output,
    "ADF Test (Differenced)": adf_diff_output,
    "Chosen ARIMA Order": (2, 1, 2),
    "MAE": mae,
    "RMSE": rmse,
    "MAPE": mape,
    "R² Score": r2
}
```







```
[14]: {'ADF Test (Original)': {'ADF Statistic': np.float64(0.8153688792060498),
        'p-value': np.float64(0.991880243437641),
        'Used Lags': 13,
        'Number of Observations': 130},
       'ADF Test (Differenced)': {'ADF Statistic': np.float64(-2.8292668241700047),
        'p-value': np.float64(0.05421329028382478),
        'Used Lags': 12,
        'Number of Observations': 130},
       'Chosen ARIMA Order': (2, 1, 2),
       'MAE': 63.545311250127014,
       'RMSE': 6808.3970474928465,
       'MAPE': np.float64(14.216158861347736),
       'R² Score': -0.11530974198470823}
```