** NETFLIX SUBSCRIPTIONS FORECASTING USING PYTHON**

```
# Importing Necessary Python libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import plotly.graph_objs as go
import plotly.express as px
import plotly.io as pio
pio.templates.default = "plotly_white"
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
```

```
df = pd.read_csv('/content/Netflix-Subscriptions.csv')
print(df)
```

```
print(df.head())
```

```
  Time Period  Subscribers
0  01/04/2013     34240000
1  01/07/2013     35640000
2  01/10/2013     38010000
3  01/01/2014     41430000
4  01/04/2014     46130000
```

```
#The dataset contains subscription counts of Netflix at the start of each quarter from 2013 to 2023. Before moving forward, let’s convert the
```

```
df['Time Period'] = pd.to_datetime(df['Time Period'],
                                     format='%d/%m/%Y')
print(df.head())
```

```
  Time Period  Subscribers
0  2013-04-01     34240000
1  2013-07-01     35640000
2  2013-10-01     38010000
3  2014-01-01     41430000
4  2014-04-01     46130000
```

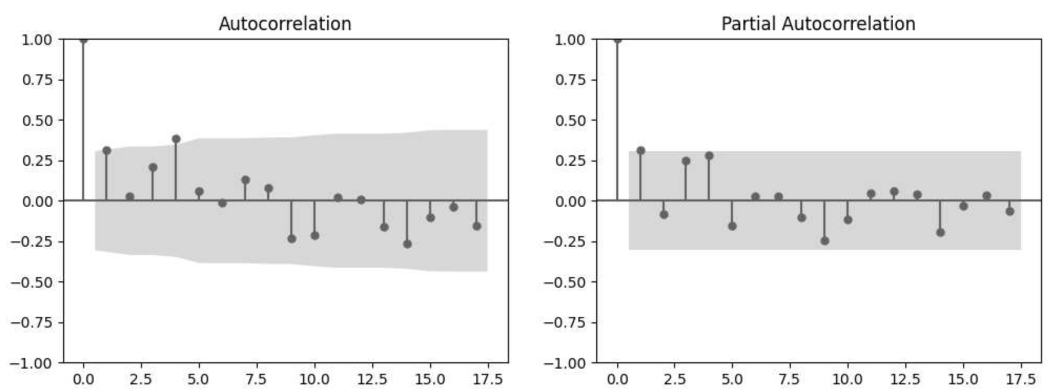
```
#Now let’s have a look at the quarterly subscription growth of Netflix:
fig = go.Figure()
fig.add_trace(go.Scatter(x=df['Time Period'], y=df['Subscribers'],mode='lines', name='Subscribers'))
fig.update_layout(title='Netflix Quarterly Subscriptions Growth', xaxis_title='Date',yaxis_title='Netflix Subscriptions in millions')
fig.show()
```

```
#In the above graph, we can see that the growth of Netflix subscribers is not seasonal. So we can use a forecasting technique like ARIMA in t

#Now let’s have a look at the quarterly growth rate of subscribers at Netflix:
```

```
# Calculate the quarterly growth rate
df['Quarterly Growth Rate'] = df['Subscribers'].pct_change() * 100

# Create a new column for bar color (green for positive growth, red for negative growth)
df['Bar Color'] = df['Quarterly Growth Rate'].apply(lambda x: 'green' if x > 0 else 'red')

# Plot the quarterly growth rate using bar graphs
fig = go.Figure()
fig.add_trace(go.Bar(
    x=df['Time Period'],
    y=df['Quarterly Growth Rate'],
    marker_color=df['Bar Color'],
    name='Quarterly Growth Rate'
))
fig.update_layout(title='Netflix Quarterly Subscriptions Growth Rate',
                  xaxis_title='Time Period',
                  yaxis_title='Quarterly Growth Rate (%)')
fig.show()
```

```
#Now let’s have a look at the yearly growth rate:
```

```
# Calculate the yearly growth rate
df['Year'] = df['Time Period'].dt.year
yearly_growth = df.groupby('Year')['Subscribers'].pct_change().fillna(0) * 100

# Create a new column for bar color (green for positive growth, red for negative growth)
df['Bar Color'] = yearly_growth.apply(lambda x: 'green' if x > 0 else 'red')

# Plot the yearly subscriber growth rate using bar graphs
fig = go.Figure()
fig.add_trace(go.Bar(
    x=df['Year'],
    y=yearly_growth,
    marker_color=df['Bar Color'],
    name='Yearly Growth Rate'
))
fig.update_layout(title='Netflix Yearly Subscriber Growth Rate',
                  xaxis_title='Year',
                  yaxis_title='Yearly Growth Rate (%)')
fig.show()


#Using ARIMA for Forecasting Netflix Quarterly Subscriptions
#Now let’s get started with Time Series Forecasting using ARIMA to forecast the number of subscriptions of Netflix using Python


time_series = df.set_index('Time Period')['Subscribers']


differenced_series = time_series.diff().dropna()

# Plot ACF and PACF of differenced time series
fig, axes = plt.subplots(1, 2, figsize=(12, 4))
plot_acf(differenced_series, ax=axes[0])
plot_pacf(differenced_series, ax=axes[1])
plt.show()
```




```
#Now here’s how to use the ARIMA model on our data


p, d, q = 1, 1, 1
model = ARIMA(time_series, order=(p, d, q))
results = model.fit()
print(results.summary())
```

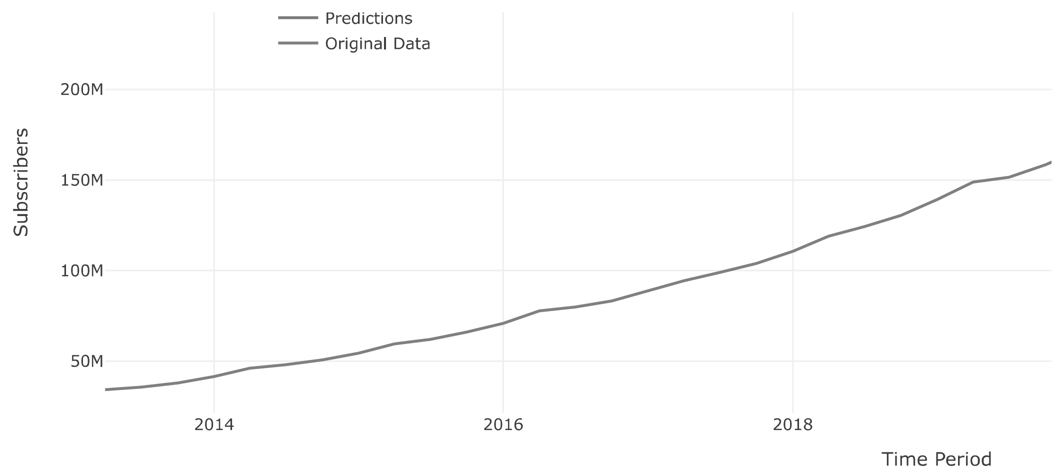
```
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning:

No frequency information was provided, so inferred frequency QS-OCT will be used.

/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning:

No frequency information was provided, so inferred frequency QS-OCT will be used.

/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning:

No frequency information was provided, so inferred frequency QS-OCT will be used.
```

```
                               SARIMAX Results
==============================================================================
Dep. Variable:            Subscribers   No. Observations:                   42
Model:                 ARIMA(1, 1, 1)   Log Likelihood                -672.993
Date:                Sat, 16 Sep 2023   AIC                           1351.986
Time:                        15:46:35   BIC                           1357.127
Sample:                    04-01-2013   HQIC                          1353.858
                         - 07-01-2023
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.9997      0.012     80.765      0.000       0.975       1.024
ma.L1         -0.9908      0.221     -4.476      0.000      -1.425      -0.557
sigma2      1.187e+13   1.57e-14   7.57e+26      0.000    1.19e+13    1.19e+13
===================================================================================
Ljung-Box (L1) (Q):                   3.96   Jarque-Bera (JB):                 4.62
Prob(Q):                              0.05   Prob(JB):                         0.10
Heteroskedasticity (H):               7.27   Skew:                             0.54
Prob(H) (two-sided):                  0.00   Kurtosis:                         4.23
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
[2] Covariance matrix is singular or near-singular, with condition number 7.88e+42. Standard errors may be unstable.
```

```
#Now here’s how to make predictions using the trained model to forecast the number of subscribers for the next five quarters:


future_steps = 5
predictions = results.predict(len(time_series), len(time_series) + future_steps - 1)
predictions = predictions.astype(int)


#Now let’s visualize the results of Netflix Subscriptions Forecasting for the next five quarters:


# Create a DataFrame with the original data and predictions
forecast = pd.DataFrame({'Original': time_series, 'Predictions': predictions})

# Plot the original data and predictions
fig = go.Figure()

fig.add_trace(go.Scatter(x=forecast.index, y=forecast['Predictions'],
                         mode='lines', name='Predictions'))

fig.add_trace(go.Scatter(x=forecast.index, y=forecast['Original'],
                         mode='lines', name='Original Data'))

fig.update_layout(title='Netflix Quarterly Subscription Predictions',
                  xaxis_title='Time Period',
                  yaxis_title='Subscribers',
                  legend=dict(x=0.1, y=0.9),
                  showlegend=True)

fig.show()
```

Netflix Quarterly Subscription Predictions

```
#So this is how you can forecast subscription counts for a given time period using Time Series Forecasting and Python.
```



✓ 0s completed at 9:19 PM