
WEB TRAFFIC
GO
PLAN
PASSWORD
12345

Website Traffic Forecasting using Python

The dataset I am using for Website Traffic Forecasting is collected from the daily traffic data of **thecleverprogrammer.com**. It contains data about daily traffic data from June 2021 to June 2022. **You can download the dataset from here.** Now let's get started with the task of website traffic forecasting by importing the necessary Python libraries and the **dataset**:

```
import pandas as pd
import matplotlib.pyplot as plt
import plotly.express as px
import plotly.graph_objects as go
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.graphics.tsaplots import plot_pacf
from statsmodels.tsa.arima_model import ARIMA
import statsmodels.api as sm

data = pd.read_csv("Thecleverprogrammer.csv")
print(data.head())
```

```
         Date  Views
0  01/06/2021   7831
1  02/06/2021   7798
2  03/06/2021   7401
3  04/06/2021   7054
4  05/06/2021   7973
```

The dataset contains two columns, date and traffic. Before moving forward, I will convert the Date column into Datetime data type:

```
data["Date"] = pd.to_datetime(data["Date"],
                              format="%d/%m/%Y")
print(data.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 391 entries, 0 to 390
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Date    391 non-null    datetime64[ns]
 1   Views   391 non-null    int64
dtypes: datetime64[ns](1), int64(1)
memory usage: 6.2 KB
None
```

The Date time column was an object initially, so I converted it into a Datetime column. Now let's have a look at the daily traffic of the website:

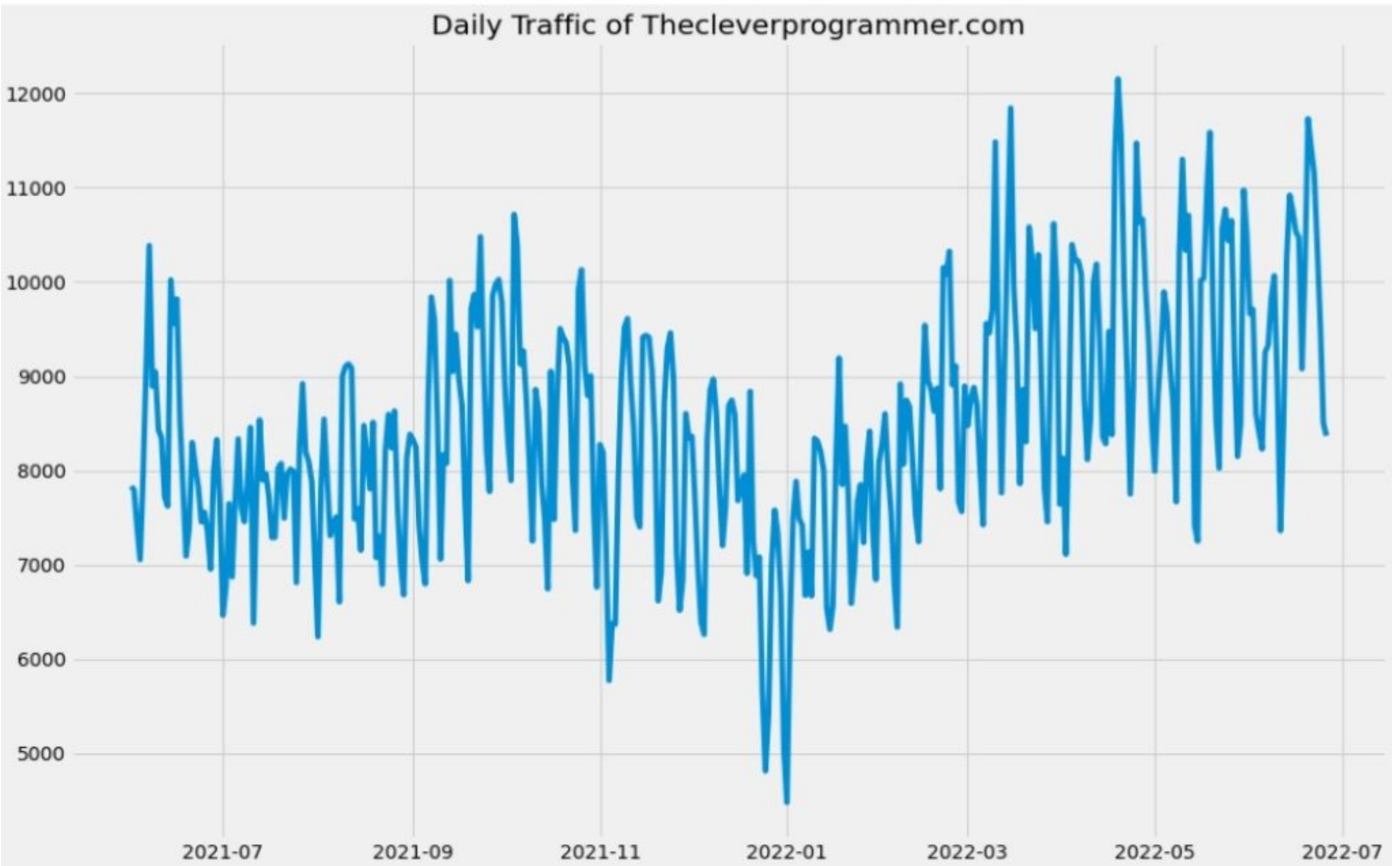
```
plt.style.use('fivethirtyeight')
plt.figure(figsize=(15, 10))
plt.plot(data["Date"], data["Views"])
plt.title("Daily Traffic of Thecleverprogrammer.com")
plt.show()
```

```
print(data.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 391 entries, 0 to 390
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype         
---  ------  --------------  -----         
 0   Date    391 non-null    datetime64[ns]
 1   Views   391 non-null    int64         
dtypes: datetime64[ns](1), int64(1)
memory usage: 6.2 KB
None
```

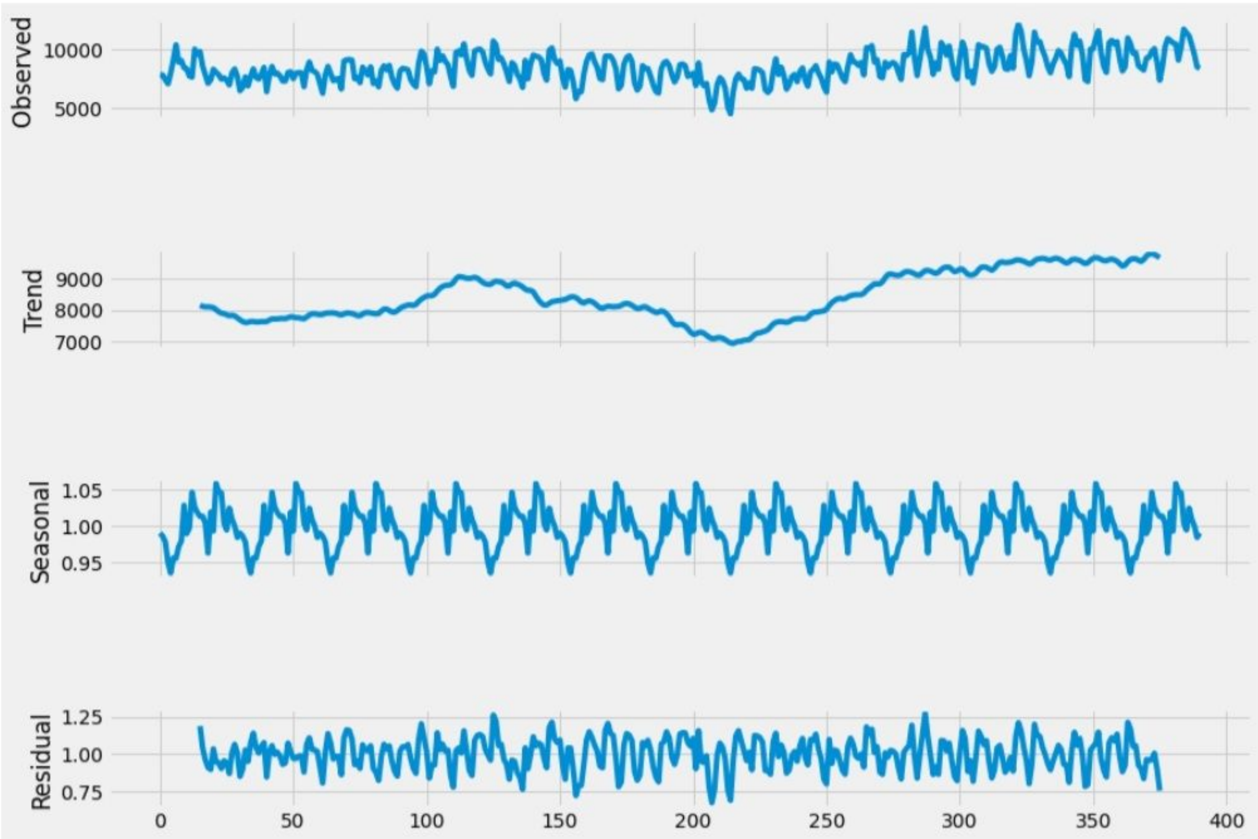
The Date time column was an object initially, so I converted it into a Datetime column. Now let's have a look at the daily traffic of the website:

```
plt.style.use('fivethirtyeight')
plt.figure(figsize=(15, 10))
plt.plot(data["Date"], data["Views"])
plt.title("Daily Traffic of Thecleverprogrammer.com")
plt.show()
```



Our website traffic data is seasonal because the traffic on the website increases during the weekdays and decreases during the weekends. It is valuable to know if the dataset is seasonal or not while working on the problem of Time Series Forecasting. Below is how we can have a look at whether our dataset is stationary or seasonal:

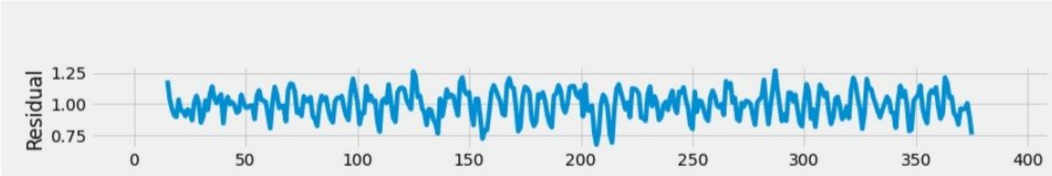
```
result = seasonal_decompose(data["Views"],
                            model='multiplicative',
                            freq = 30)
fig = plt.figure()
fig = result.plot()
fig.set_size_inches(15, 10)
```



I will be using the Seasonal ARIMA (SARIMA) model to forecast traffic on the website. Before using the SARIMA model, it is necessary to find the p, d, and q values. You can learn how to find p, d, and q values from **here**.

As the data is not stationary, the value of d is 1. To find the values of p and q, we can use the autocorrelation and partial autocorrelation plots:

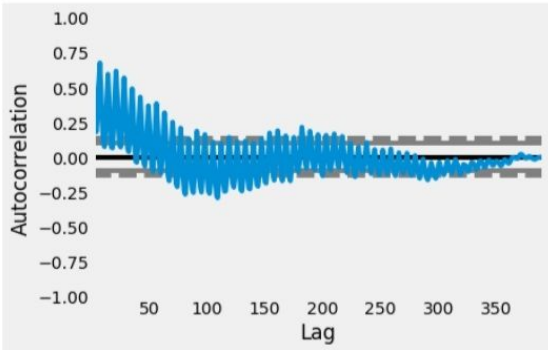
```
pd.plotting.autocorrelation_plot(data["Views"])
```



I will be using the Seasonal ARIMA (SARIMA) model to forecast traffic on the website. Before using the SARIMA model, it is necessary to find the p, d, and q values. You can learn how to find p, d, and q values from **here**.

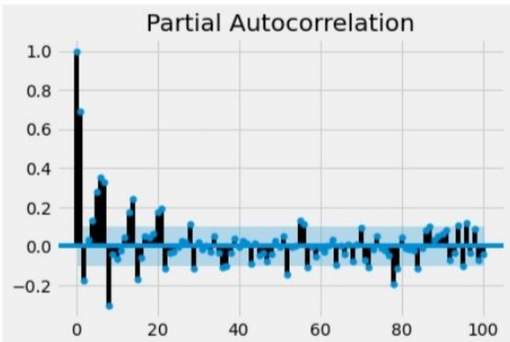
As the data is not stationary, the value of d is 1. To find the values of p and q, we can use the autocorrelation and partial autocorrelation plots:

```
pd.plotting.autocorrelation_plot(data["Views"])
```



p = 5

```
plot_pacf(data["Views"], lags = 100)
```



q = 2

Now here's how we can train a SARIMA model for the task of website traffic forecasting:

```
p, d, q = 5, 1, 2
model=sm.tsa.statespace.SARIMAX(data['Views'],
                                order=(p, d, q),
                                seasonal_order=(p, d, q, 12))
model=model.fit()
print(model.summary())
```

```
                                     Statespace Model Results
==========================================================================================
Dep. Variable:                                Views   No. Observations:                391
Model:             SARIMAX(5, 1, 2)x(5, 1, 2, 12)   Log Likelihood             -3099.402
Date:                              Tue, 28 Jun 2022   AIC                         6228.803
Time:                                      07:01:10   BIC                         6287.827
Sample:                                           0   HQIC                        6252.229
                                              - 391
Covariance Type:                                opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.7808      0.134      5.836      0.000       0.519       1.043
```

```
model=model.fit()
print(model.summary())
```

```
                                 Statespace Model Results
==========================================================================================
Dep. Variable:                                 Views   No. Observations:              391
Model:             SARIMAX(5, 1, 2)x(5, 1, 2, 12)   Log Likelihood              -3099.402
Date:                           Tue, 28 Jun 2022   AIC                          6228.803
Time:                                   07:01:10   BIC                          6287.827
Sample:                                        0   HQIC                         6252.229
                                           - 391
Covariance Type:                             opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.7808      0.134      5.836      0.000       0.519       1.043
ar.L2         -0.7973      0.135     -5.920      0.000      -1.061      -0.533
ar.L3         -0.1442      0.170     -0.850      0.395      -0.477       0.188
ar.L4         -0.1833      0.151     -1.210      0.226      -0.480       0.114
ar.L5         -0.1548      0.139     -1.117      0.264      -0.426       0.117
ma.L1         -1.1826      0.094    -12.515      0.000      -1.368      -0.997
ma.L2          0.8856      0.078     11.304      0.000       0.732       1.039
ar.S.L12      -0.2606      4.608     -0.057      0.955      -9.293       8.772
ar.S.L24       0.0428      0.781      0.055      0.956      -1.488       1.573
ar.S.L36      -0.1880      0.246     -0.764      0.445      -0.670       0.294
ar.S.L48      -0.2151      0.959     -0.224      0.823      -2.095       1.664
ar.S.L60       0.0127      0.986      0.013      0.990      -1.920       1.946
ma.S.L12      -0.6902      4.611     -0.150      0.881      -9.728       8.348
ma.S.L24      -0.0994      3.637     -0.027      0.978      -7.228       7.029
sigma2      1.257e+06   1.59e+05      7.914      0.000    9.46e+05    1.57e+06
===================================================================================
Ljung-Box (Q):                      102.98   Jarque-Bera (JB):                 1.32
Prob(Q):                              0.00   Prob(JB):                         0.52
Heteroskedasticity (H):               1.03   Skew:                             0.14
Prob(H) (two-sided):                  0.85   Kurtosis:                         3.01
===================================================================================
```

Now let's forecast traffic on the website for the next 50 days:


```
predictions = model.predict(len(data), len(data)+50)
print(predictions)
```

```
391     9874.390136
392    10786.957398
393    10757.445305
394     9863.890552
395     8765.031698
396     8212.310651
397     8929.181869
398     9685.809771
399    10270.622236
400    10625.904093
401     9854.870630
402     9362.193417
403     9040.021193
404     9081.558484
405    10538.993124
406    11003.816870
407    10897.859601
408    10083.291284
409     9445.806523
410     8629.901288
411     9184.420361
412    10392.770399
413    10593.941868
414    10788.128238
415    10263.101427
```

```
predictions = model.predict(len(data), len(data)+50)
print(predictions)
```

```
391     9874.390136
392    10786.957398
393    10757.445305
394     9863.890552
395     8765.031698
396     8212.310651
397     8929.181869
398     9685.809771
399    10270.622236
400    10625.904093
401     9854.870630
402     9362.193417
403     9040.021193
404     9081.558484
405    10538.993124
406    11003.816870
407    10897.859601
408    10083.291284
409     9445.806523
410     8629.901288
411     9184.420361
412    10392.770399
413    10593.941868
414    10788.128238
415    10263.101427
416     9449.467789
417     9040.226113
418     9168.972091
419     9887.094079
420    10218.658067
421    10715.657122
422     9899.224399
423     9541.622897
424     9065.810941
425     8825.335634
426    10137.936392
427    10839.866240
428    10905.862922
429    10411.640309
430     9451.211368
431     8698.339931
432     8725.534103
433    10060.678587
434    10506.263524
435    10842.515622
436    10485.387495
437     9335.244813
438     9175.122336
439     9357.034382
440    10295.910655
441    11162.934817
dtype: float64
```

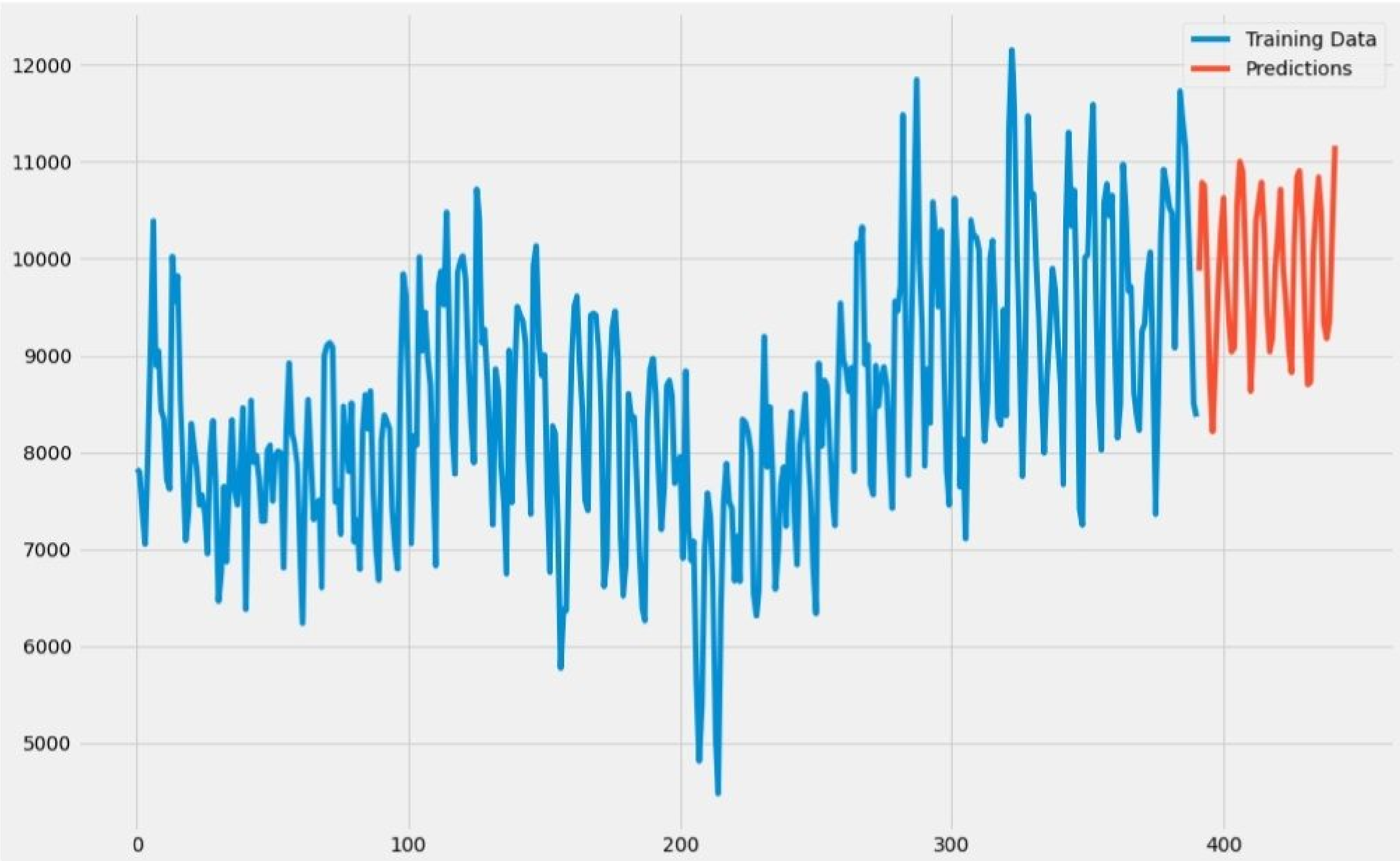
Here's how we can plot the predictions:

```
data["Views"].plot(legend=True, label="Training Data",
                   figsize=(15, 10))
predictions.plot(legend=True, label="Predictions")
```

```
435     10842.515622
436     10485.387495
437      9335.244813
438      9175.122336
439      9357.034382
440     10295.910655
441     11162.934817
dtype: float64
```

Here's how we can plot the predictions:

```
data["Views"].plot(legend=True, label="Training Data",
                               figsize=(15, 10))
predictions.plot(legend=True, label="Predictions")
```



Summary

So this is how you can forecast website traffic for a particular period. Website traffic prediction is one of the best data science project ideas you can mention on your resume. I hope this article has been helpful for you to learn website traffic prediction using the Python programming language. Feel free to ask valuable questions in the comments section below.


Thank you