# Problem Statement

Ad Ease is an ads and marketing based company helping businesses elicit maximum clicks @ minimum cost. AdEase is an ad infrastructure to help businesses promote themselves easily, effectively, and economically. The interplay of 3 AI modules - Design, Dispense, and Decipher, come together to make it this an end-to-end 3 step process digital advertising solution for all.

You are working in the Data Science team of Ad ease trying to understand the per page view report for different wikipedia pages for 550 days, and forecasting the number of views so that you can predict and optimize the ad placement for your clients. You are provided with the data of 145k wikipedia pages and daily view count for each of them. Your clients belong to different regions and need data on how their ads will perform on pages in different languages.

#### Importing the libraries

In [ ]:
```
!pip install --upgrade --no-cache-dir gdown
```

```
Looking in indexes: https://pypi.org/simple, (https://pypi.org/simple,) ht
tps://us-python.pkg.dev/colab-wheels/public/simple/ (https://us-python.pk
g.dev/colab-wheels/public/simple/)
Requirement already satisfied: gdown in /usr/local/lib/python3.8/dist-pack
ages (4.6.0)
Requirement already satisfied: six in /usr/local/lib/python3.8/dist-packag
es (from gdown) (1.15.0)
Requirement already satisfied: requests[socks] in /usr/local/lib/python3.
8/dist-packages (from gdown) (2.23.0)
Requirement already satisfied: tqdm in /usr/local/lib/python3.8/dist-packa
ges (from gdown) (4.64.1)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.8/
dist-packages (from gdown) (4.6.3)
Requirement already satisfied: filelock in /usr/local/lib/python3.8/dist-p
ackages (from gdown) (3.8.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python
3.8/dist-packages (from requests[socks]->gdown) (2022.9.24)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in
/usr/local/lib/python3.8/dist-packages (from requests[socks]->gdown) (1.2
4.3)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python
3.8/dist-packages (from requests[socks]->gdown) (3.0.4)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.8/di
st-packages (from requests[socks]->gdown) (2.10)
Requirement already satisfied: PySocks!=1.5.7,>=1.5.6 in /usr/local/lib/py
thon3.8/dist-packages (from requests[socks]->gdown) (1.7.1)
```

In [ ]:
```
import pandas as pd
import numpy as np
import pylab as p
import matplotlib.pyplot as plot
from collections import Counter
import re
import os
import seaborn as sns
```

In [ ]:
```
import warnings
warnings.filterwarnings("ignore")
warnings.simplefilter("ignore")
```

In [ ]:
```
sns.set(rc={'figure.figsize':(11.7,8.27)})
```

In [ ]:
```
import gdown
#url='https://drive.google.com/file/d/1gHYYLqLt6rMyeAyvHf1wvlQ4BLKwjv9W/vie
#url='https://drive.google.com/file/d/1SL_7DoE16m71QpjJXoQUC3cI5aHCIZLv/vie
#url='https://drive.google.com/file/d/11GQSe2Xm4vFD4Xfw3JhOoPLXnBE_LiMe/vie
url='https://drive.google.com/file/d/1CJOMYyg64x3gN52p6OqypN6UUgDnUhkm/view

ider=url.split('/')[-2]
!gdown --id $ider
```

```
/usr/local/lib/python3.8/dist-packages/gdown/cli.py:121: FutureWarning: Op
tion `--id` was deprecated in version 4.3.1 and will be removed in 5.0. Yo
u don't need to pass it anymore to use a file ID.
  warnings.warn(
Downloading...
From: https://drive.google.com/uc?id=1CJOMYyg64x3gN52p6OqypN6UUgDnUhkm (ht
tps://drive.google.com/uc?id=1CJOMYyg64x3gN52p6OqypN6UUgDnUhkm)
To: /content/new_train.csv
100% 425M/425M [00:03<00:00, 125MB/s]
```

In [ ]:
```
train = pd.read_csv('new_train.csv')
```

Reading the dataset and printing head and tail to get basic idea

In [ ]:
```
train.head()
```

Out[12]:

| | Page | 2015-07-01 | 2015-07-02 | 2015-07-03 | 2015-07-04 | 2015-07-05 | 2015-07-06 | 2015-07-07 | 2015-07-0 |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 2NE1_zh.britanica.org_all-access_spider | 18.0 | 11.0 | 5.0 | 13.0 | 14.0 | 9.0 | 9.0 | 22.( |
| **1** | 2PM_zh.britanica.org_all-access_spider | 11.0 | 14.0 | 15.0 | 18.0 | 11.0 | 13.0 | 22.0 | 11.( |
| **2** | 3C_zh.britanica.org_all-access_spider | 1.0 | 0.0 | 1.0 | 1.0 | 0.0 | 4.0 | 0.0 | 3.( |
| **3** | 4minute_zh.britanica.org_all-access_spider | 35.0 | 13.0 | 10.0 | 94.0 | 4.0 | 26.0 | 14.0 | 9.( |
| **4** | 52_Hz_I_Love_You_zh.britanica.org_all-access_s... | NaN | NaN | NaN | NaN | NaN | NaN | NaN | Nal |

5 rows × 551 columns

In [ ]:
```
print(train.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 145063 entries, 0 to 145062
Columns: 551 entries, Page to 2016-12-31
dtypes: float64(550), object(1)
memory usage: 609.8+ MB
None
```

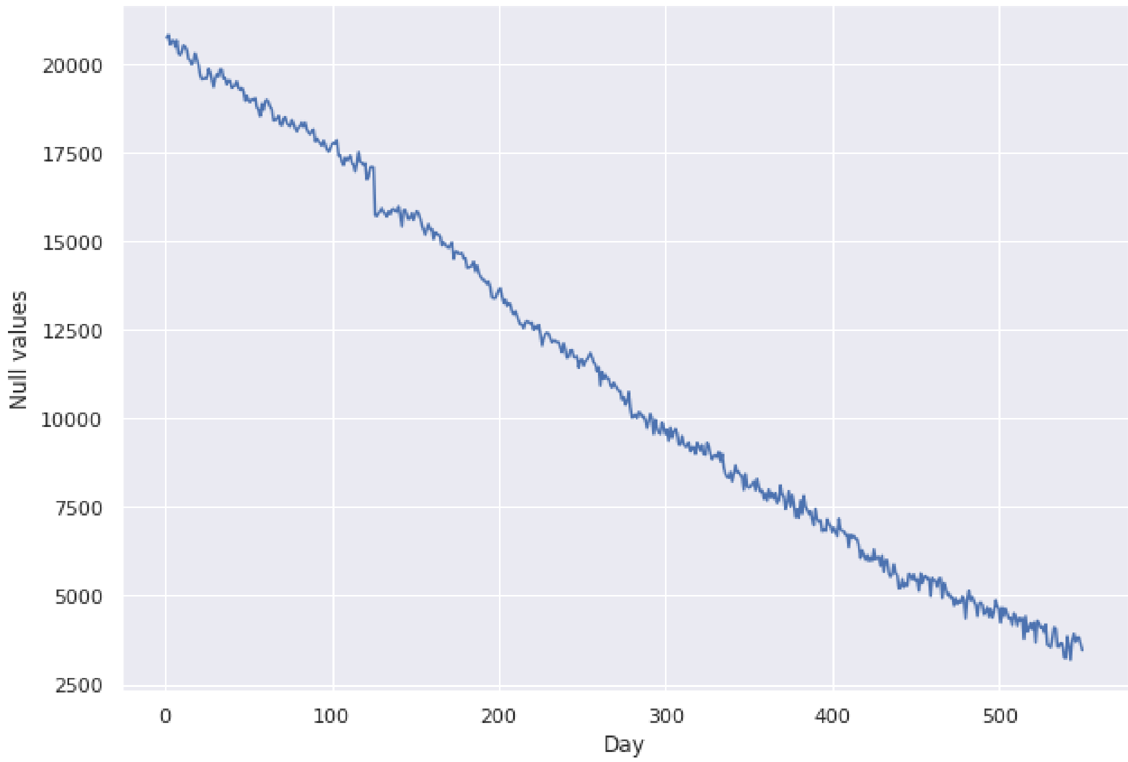
In [ ]:
```
print(train.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 145063 entries, 0 to 145062
Columns: 551 entries, Page to 2016-12-31
dtypes: float64(550), object(1)
memory usage: 609.8+ MB
None
```

We can see that ther are some null values in the data, we will plot them to see how it looks

In [ ]:
```
days = [r for r in range(1, len(train.columns))]
plot.figure(figsize=(10,7))
plot.xlabel('Day')
plot.ylabel('Null values')
plot.plot(days, train.isnull().sum()[1:])
```

Out[15]: [<matplotlib.lines.Line2D at 0x7f9af53dca90>]




We see that the number of nan values decrease with time.

Probable reason: Some website have all nan values in the begining, that can be due to the fact that those were created after that time so there is no traffic reading for that time

In [ ]:
```
print(train.shape)
train=train.dropna(how='all')
#‘all’ : If all values are NA, drop that row or column.
print(train.shape)

train=train.dropna(thresh=300)
print(train.shape)
```

```
(145063, 551)
(145063, 551)
(133617, 551)
```

1. We try droping the rows that have all values as nan, none in our case.
2. We then also drop rows that have nan more than 300 days, because the time series for that would not make much sense
3. We fill all the remaining values with zero assuming there was no traffic on the date that the values are nan for.

In [ ]:
```
train=train.fillna(0)
train.tail()
```

Out[17]:

| | Page | 2015-07-01 | 2015-07-02 | 2015-07-03 | 2015-07-04 | 2015-07-05 |
|---|---|---|---|---|---|---|
| **145012** | Legión_(Marvel_Comics)_es.britanica.org_all-ac... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **145013** | Referéndum_sobre_la_permanencia_del_Reino_Unid... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **145014** | Salida_del_Reino_Unido_de_la_Unión_Europea_es.... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **145015** | Amar,_después_de_amar_es.britanica.org_all-acc... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **145016** | Anexo:89.º_Premios_Óscar_es.britanica.org_all-... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |

5 rows × 551 columns

# *EDA*

The page values are in this format

**SPECIFIC NAME _ LANGUAGE.britanica.org _ ACCESS TYPE _ ACCESS ORIGIN**

having information about page name, the main domain, device type used to access the page, and also the request origin(spider or browser agent)

In [ ]:
```
#Usage of Regex
def split_page(page):
    w = re.split('_|\.', page)
    print(w)
    return ' '.join(w[:-5]), w[-5], w[-2], w[-1]

split_page('2NE1_zh.britanica.org_all-access_spider')
```

```
['2NE1', 'zh', 'britanica', 'org', 'all-access', 'spider']
```

Out[18]: ('2NE1', 'zh', 'all-access', 'spider')

In [ ]:
```
def split_page(page):
  w = re.split('_|\.', page)
  return ' '.join(w[:-5]), w[-5], w[-2], w[-1]

li = list(train.Page.apply(lambda x: split_page(str(x))))
df = pd.DataFrame(li)
df.columns = ['Title', 'Language', 'Access_type','Access_origin']
df = pd.concat([train, df], axis = 1)
```

We split the page name and get that information joining it with a temporary database. below we get some rows to see the structure of the data

In [ ]:
```
df.head()
```

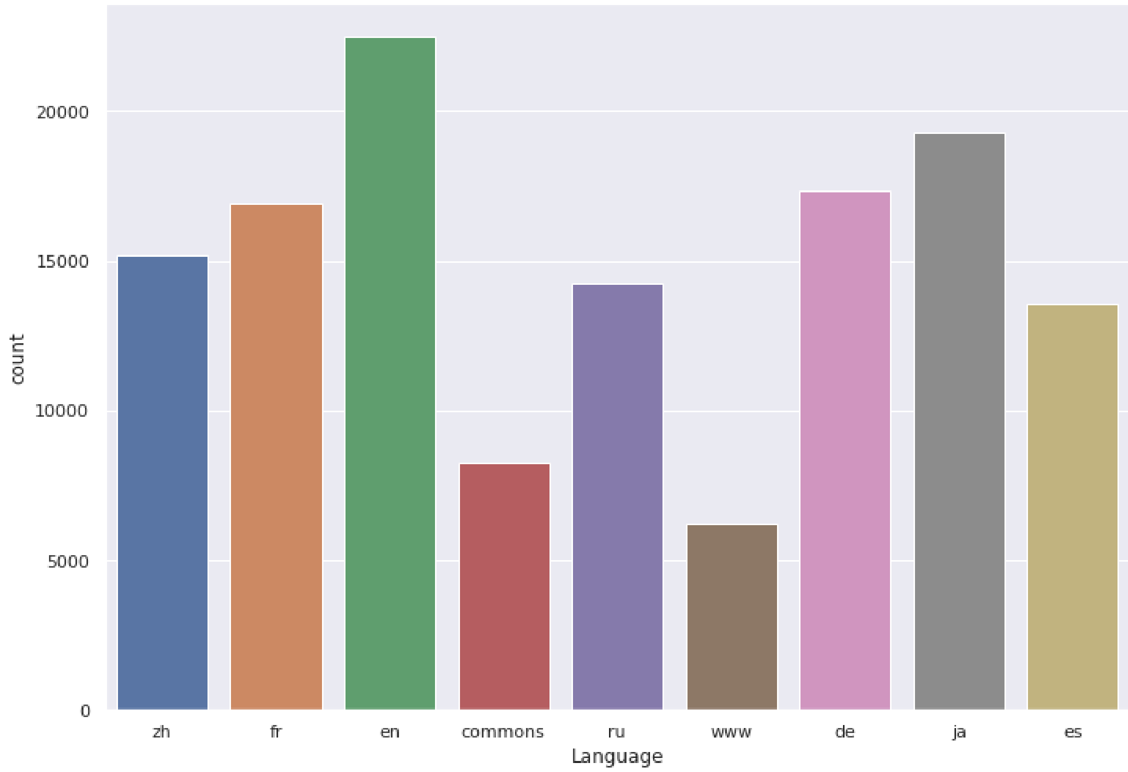
Out[20]:

| | Page | 2015-07-01 | 2015-07-02 | 2015-07-03 | 2015-07-04 | 2015-07-05 | 2015-07-06 | 2015-07-07 | 2015-07-08 | 2015-07-09 | . |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 2NE1_zh.britanica.org_all-access_spider | 18.0 | 11.0 | 5.0 | 13.0 | 14.0 | 9.0 | 9.0 | 22.0 | 26.0 | . |
| **1** | 2PM_zh.britanica.org_all-access_spider | 11.0 | 14.0 | 15.0 | 18.0 | 11.0 | 13.0 | 22.0 | 11.0 | 10.0 | . |
| **2** | 3C_zh.britanica.org_all-access_spider | 1.0 | 0.0 | 1.0 | 1.0 | 0.0 | 4.0 | 0.0 | 3.0 | 4.0 | . |
| **3** | 4minute_zh.britanica.org_all-access_spider | 35.0 | 13.0 | 10.0 | 94.0 | 4.0 | 26.0 | 14.0 | 9.0 | 11.0 | . |
| **4** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | . |

5 rows × 555 columns

In [ ]:
```
sns.countplot(df['Language'])
```

Out[21]: <matplotlib.axes._subplots.AxesSubplot at 0x7f9af603af10>

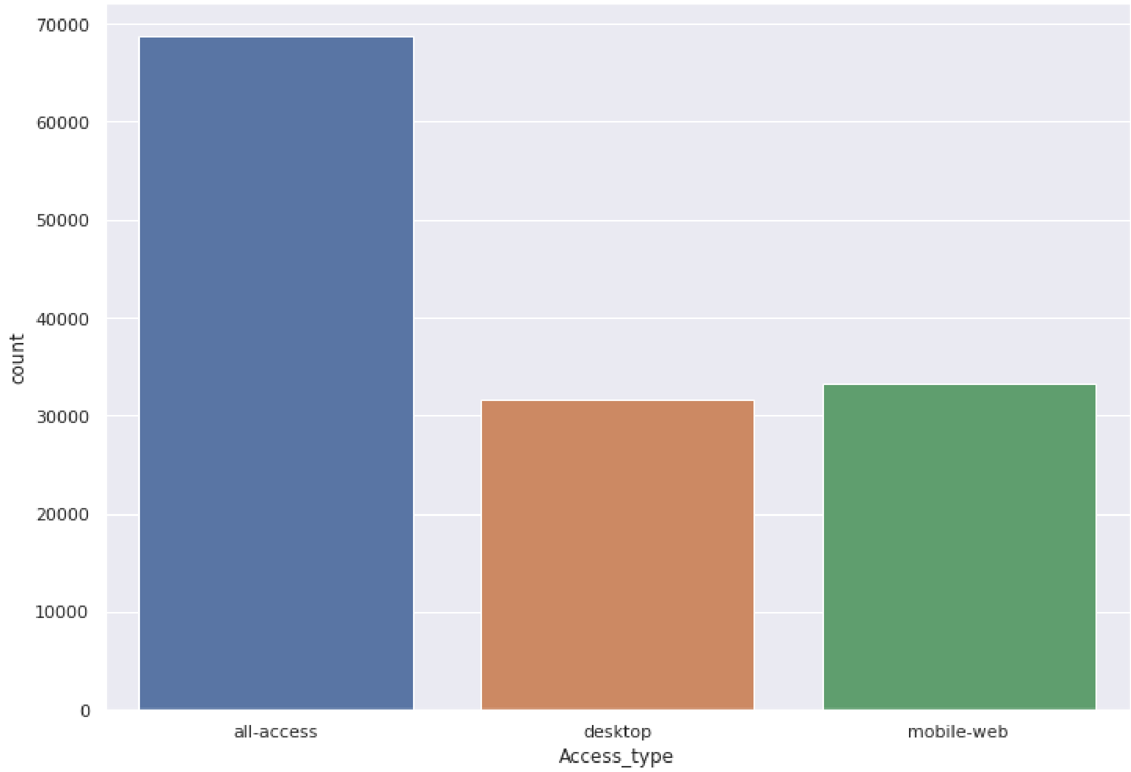



This above is the comparision number of articles in each language

{'ja':'Japanese', 'de':'German', 'en' : 'English', 'no_lang':'Media_File', 'fr':'French', 'zh':'Chinese', 'ru':'Russian', 'es':'Spanish'}

In [ ]:
```
sns.countplot(df['Access_type'])
```

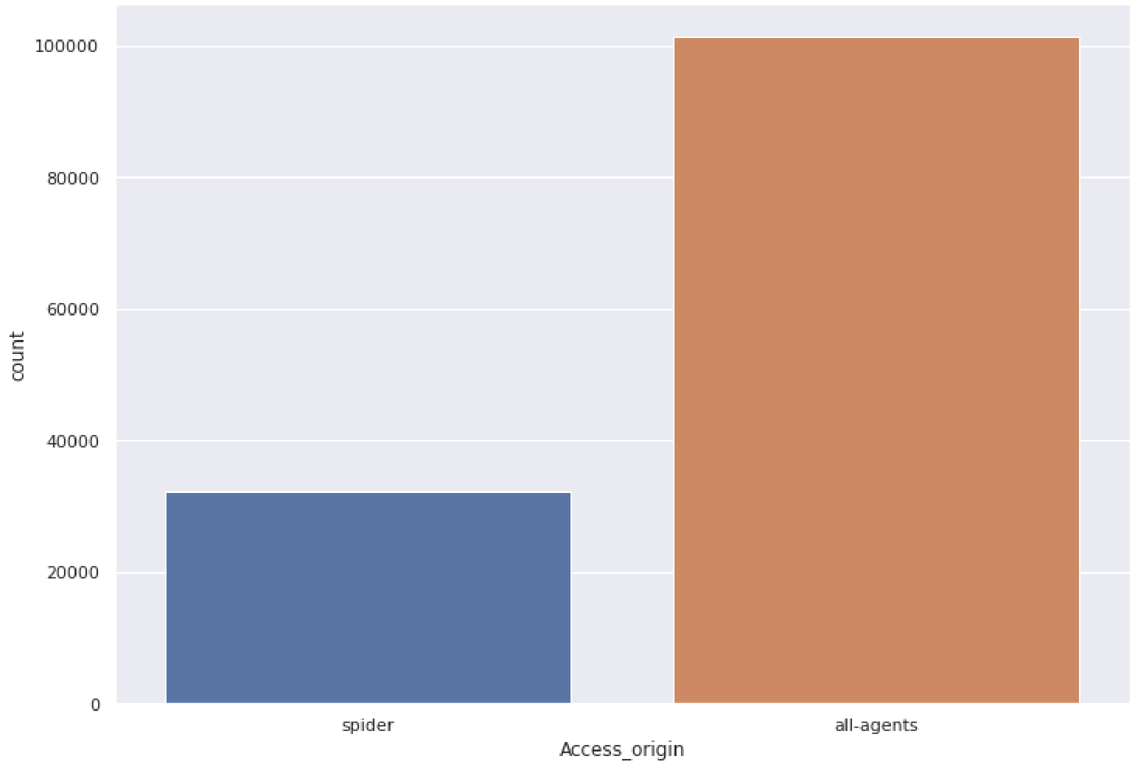
Out[22]: <matplotlib.axes._subplots.AxesSubplot at 0x7f9af249c220>




This comparision shows that usage from desktop and mobile is almost the same

In [ ]:
```
sns.countplot(df['Access_origin'])
```

Out[23]: <matplotlib.axes._subplots.AxesSubplot at 0x7f9af23f3880>



This shows that organic view is far more than that of spiders or bots

**Now we want to compare the views for different languages **

In [ ]:

```
#here we see that the languages are not treated properly as there are commo
df.groupby('Language').count()
```

Out[24]:

| | Page | 2015-07-01 | 2015-07-02 | 2015-07-03 | 2015-07-04 | 2015-07-05 | 2015-07-06 | 2015-07-07 | 2015-07-08 | 2015-07-09 | ... | 2016 12-2 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Language** | | | | | | | | | | | | |
| **commons** | 7672 | 7672 | 7672 | 7672 | 7672 | 7672 | 7672 | 7672 | 7672 | 7672 | ... | 767 |
| **de** | 15946 | 15946 | 15946 | 15946 | 15946 | 15946 | 15946 | 15946 | 15946 | 15946 | ... | 1594 |
| **en** | 20758 | 20758 | 20758 | 20758 | 20758 | 20758 | 20758 | 20758 | 20758 | 20758 | ... | 2075 |
| **es** | 12268 | 12268 | 12268 | 12268 | 12268 | 12268 | 12268 | 12268 | 12268 | 12268 | ... | 1226 |
| **fr** | 15418 | 15418 | 15418 | 15418 | 15418 | 15418 | 15418 | 15418 | 15418 | 15418 | ... | 1541 |
| **ja** | 17132 | 17132 | 17132 | 17132 | 17132 | 17132 | 17132 | 17132 | 17132 | 17132 | ... | 1713 |
| **ru** | 12955 | 12955 | 12955 | 12955 | 12955 | 12955 | 12955 | 12955 | 12955 | 12955 | ... | 1295 |
| **www** | 5743 | 5743 | 5743 | 5743 | 5743 | 5743 | 5743 | 5743 | 5743 | 5743 | ... | 574 |
| **zh** | 14845 | 14845 | 14845 | 14845 | 14845 | 14845 | 14845 | 14845 | 14845 | 14845 | ... | 1484 |

9 rows × 554 columns

In [ ]:
```
df[df['Language']=='commons']
```

Out[25]:

| | Page | 2015-07-01 | 2015-07-02 | 2015-07-03 | 2015-07-04 | 2015-07-05 |
|---|---|---|---|---|---|---|
| **12271** | Burning_Man_en.britanica.org_desktop_all-agents | 1693.0 | 1490.0 | 1186.0 | 1099.0 | 1051.0 |
| **12272** | Cali_Cartel_en.britanica.org_desktop_all-agents | 348.0 | 363.0 | 214.0 | 252.0 | 257.0 |
| **12273** | Call_of_Duty:_Modern_Warfare_2_en.britanica.or... | 806.0 | 768.0 | 700.0 | 725.0 | 723.0 |
| **12274** | Calvin_Harris_en.britanica.org_desktop_all-agents | 7114.0 | 5599.0 | 7685.0 | 15844.0 | 9390.0 |
| **12275** | Carl_Sagan_en.britanica.org_desktop_all-agents | 1808.0 | 1759.0 | 1838.0 | 1631.0 | 1701.0 |
| **...** | ... | ... | ... | ... | ... | ... |
| **75274** | Ash_Wednesday_en.britanica.org_mobile-web_all-... | 170.0 | 169.0 | 165.0 | 166.0 | 186.0 |
| **75275** | Ashley_Williams_(footballer)_en.britanica.org_... | 112.0 | 102.0 | 135.0 | 147.0 | 120.0 |
| **75276** | Assassin's_Creed_(film)_en.britanica.org_mobil... | 28.0 | 15.0 | 24.0 | 24.0 | 27.0 |
| **75277** | Aubrey_Plaza_en.britanica.org_mobile-web_all-a... | 3067.0 | 2952.0 | 3459.0 | 3310.0 | 3294.0 |
| **75278** | Australia_Plus_en.britanica.org_mobile-web_all... | 17.0 | 11.0 | 14.0 | 6.0 | 10.0 |

8266 rows × 555 columns

In [ ]:
```
# Checking another way of fetching the language out of the string
def lang(Page):
    val = re.search('[a-z][a-z].britanica.org',Page)
    if val:
        #print(val)
        #print(val[0][0:2] )

        return val[0][0:2]

    return 'no_lang'

df['Language']=df['Page'].apply(lambda x: lang(str(x)))
```

In [ ]:
```
df.groupby('Language').count() #now the count has increased. You can go bac
```

Out[27]:

| | Page | 2015-07-01 | 2015-07-02 | 2015-07-03 | 2015-07-04 | 2015-07-05 | 2015-07-06 | 2015-07-07 | 2015-07-08 | 2015-07-09 | ... | 2016-12-2 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Language** | | | | | | | | | | | | |
| **de** | 17362 | 17362 | 17362 | 17362 | 17362 | 17362 | 17362 | 17362 | 17362 | 17362 | ... | 1736 |
| **en** | 22486 | 22486 | 22486 | 22486 | 22486 | 22486 | 22486 | 22486 | 22486 | 22486 | ... | 2248 |
| **es** | 13551 | 13551 | 13551 | 13551 | 13551 | 13551 | 13551 | 13551 | 13551 | 13551 | ... | 1355 |
| **fr** | 16948 | 16948 | 16948 | 16948 | 16948 | 16948 | 16948 | 16948 | 16948 | 16948 | ... | 1694 |
| **ja** | 19295 | 19295 | 19295 | 19295 | 19295 | 19295 | 19295 | 19295 | 19295 | 19295 | ... | 1929 |
| **no_lang** | 14494 | 14494 | 14494 | 14494 | 14494 | 14494 | 14494 | 14494 | 14494 | 14494 | ... | 1449 |
| **ru** | 14270 | 14270 | 14270 | 14270 | 14270 | 14270 | 14270 | 14270 | 14270 | 14270 | ... | 1427 |
| **zh** | 15211 | 15211 | 15211 | 15211 | 15211 | 15211 | 15211 | 15211 | 15211 | 15211 | ... | 1521 |

8 rows × 554 columns

In [ ]:
```
df_language=df.groupby('Language').mean().transpose()
df_language
```

Out[28]:

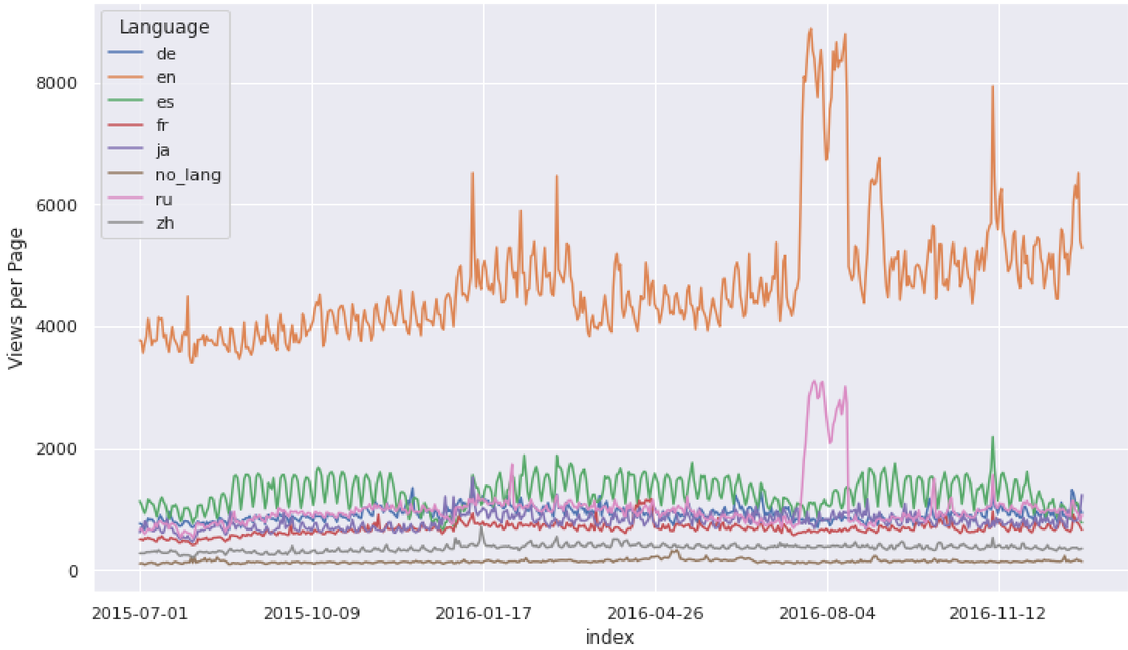
| Language | de | en | es | fr | ja | no_lang | |
|---|---|---|---|---|---|---|---|
| **2015-07-01** | 763.765926 | 3767.328604 | 1127.485204 | 499.092872 | 614.637160 | 102.733545 | 663. |
| **2015-07-02** | 753.362861 | 3755.158765 | 1077.485425 | 502.297852 | 705.813216 | 107.663447 | 674. |
| **2015-07-03** | 723.074415 | 3565.225696 | 990.895949 | 483.007553 | 637.451671 | 101.769629 | 625. |
| **2015-07-04** | 663.537323 | 3711.782932 | 930.303151 | 516.275785 | 800.897435 | 86.853871 | 588. |
| **2015-07-05** | 771.358657 | 3833.433025 | 1011.759575 | 506.871666 | 768.352319 | 96.254105 | 626. |
| **...** | ... | ... | ... | ... | ... | ... | |
| **2016-12-27** | 1119.596936 | 6314.335275 | 1070.923400 | 840.590217 | 808.541436 | 155.270181 | 998. |
| **2016-12-28** | 1062.284069 | 6108.874144 | 1108.996753 | 783.585379 | 807.430163 | 178.561267 | 945. |
| **2016-12-29** | 1033.939062 | 6518.058525 | 1058.660320 | 763.209169 | 883.752786 | 150.873534 | 909. |
| **2016-12-30** | 981.786430 | 5401.792360 | 807.551177 | 710.502773 | 979.278777 | 156.049193 | 815. |
| **2016-12-31** | 937.842875 | 5280.643467 | 776.934322 | 654.060656 | 1228.720808 | 135.792052 | 902. |

550 rows × 8 columns

In [ ]:
```
df_language.reset_index(inplace=True)
df_language.set_index('index', inplace=True)
```

In [ ]:
```
df_language.plot(figsize=(12,7))
plot.ylabel('Views per Page')
```

Out[30]: Text(0, 0.5, 'Views per Page')

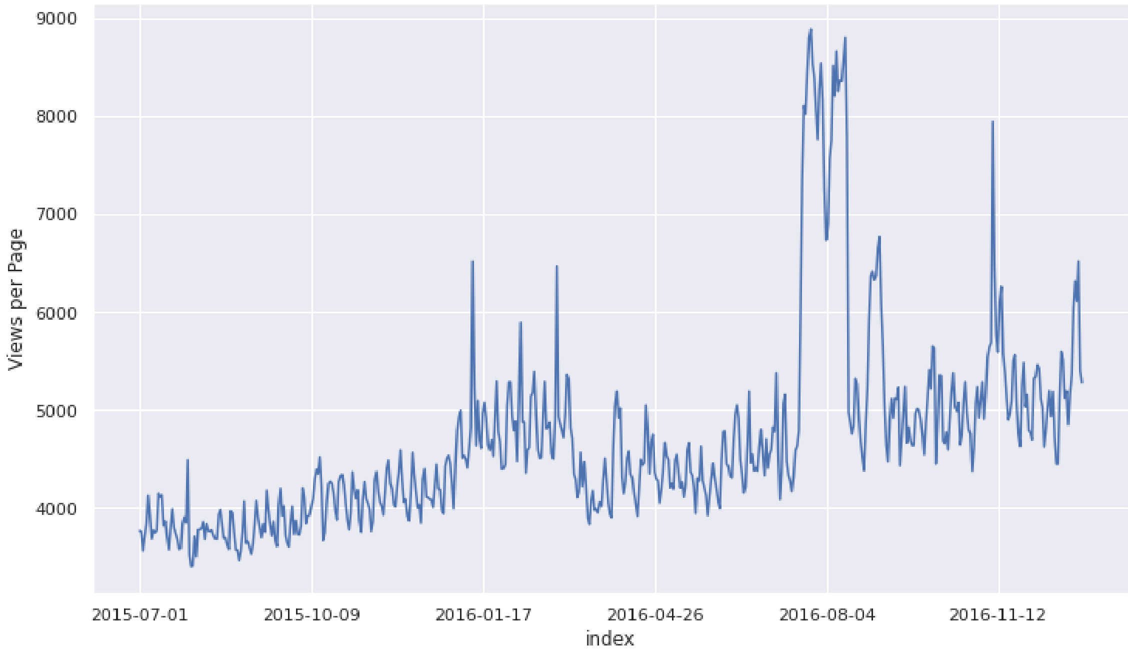



Ploting the data shows that articles in english get the most number of views as compared to different languages, there are some spikes at different times in different laguages

Ploting just for english because we are going to use this for our furthur investigation and predictions

In [ ]:
```
df_language['en'].plot(figsize=(12,7))
plot.ylabel('Views per Page')
```

Out[31]:
```
Text(0, 0.5, 'Views per Page')
```




In [ ]:
```
total_view=df_language.copy()
```

In [ ]:
```
##########################################################################
```

# Checking the stationarity

Dickey-Fuller test

**Here the null hypothesis is that the TS is non-stationary**: The test results comprise of a Test Statistic and some Critical Values for difference confidence levels.

In [ ]:
```
from statsmodels.tsa.stattools import adfuller
def df_test(x):
    result=adfuller(x)
    print('ADF Stastistic: %f'%result[0])
    print('p-value: %f'%result[1])

df_test(total_view['en'])
```

```
ADF Stastistic: -2.373563
p-value: 0.149337
```

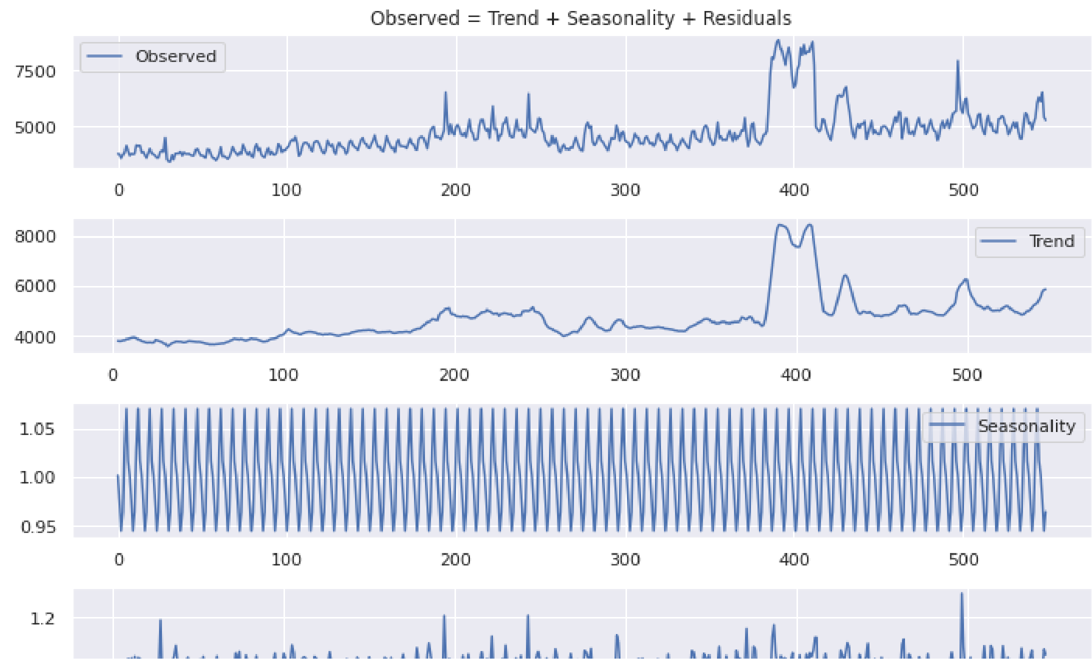
We see that the p value is not low enough(<0.05). Therefore, we can say our series in not stationary as we fail to reject the null hypothesis

# Making the time series stationary

In [ ]:
```
ts=total_view['en']
```

# 1. Remove trend and seasonality with decomposition

In [ ]:
```
# Naive decomposition of our Time Series as explained above
from statsmodels.tsa.seasonal import seasonal_decompose
decomposition = seasonal_decompose(ts.values, model='multiplicative',freq =

""" Additive or multiplicative?
  It’s important to understand what the difference between a multiplicative

  There are three components to a time series:
  - trend how things are overall changing
  - seasonality how things change within a given period e.g. a year, month,
  - error/residual/irregular activity not explained by the trend or the sea

  How these three components interact determines the difference between a m

  In a multiplicative time series, the components multiply together to make

  In an additive time series, the components add together to make the time


"""

trend = decomposition.trend
seasonal = decomposition.seasonal
residual = decomposition.resid

plot.figure(figsize=(10,7))
plot.subplot(411)
plot.title('Observed = Trend + Seasonality + Residuals')
plot.plot(ts.values,label='Observed')
plot.legend(loc='best')
plot.subplot(412)
plot.plot(trend, label='Trend')
plot.legend(loc='best')
plot.subplot(413)
plot.plot(seasonal,label='Seasonality')
plot.legend(loc='best')
plot.subplot(414)
plot.plot(residual, label='Residuals')
plot.legend(loc='best')
plot.tight_layout()
plot.show()
```



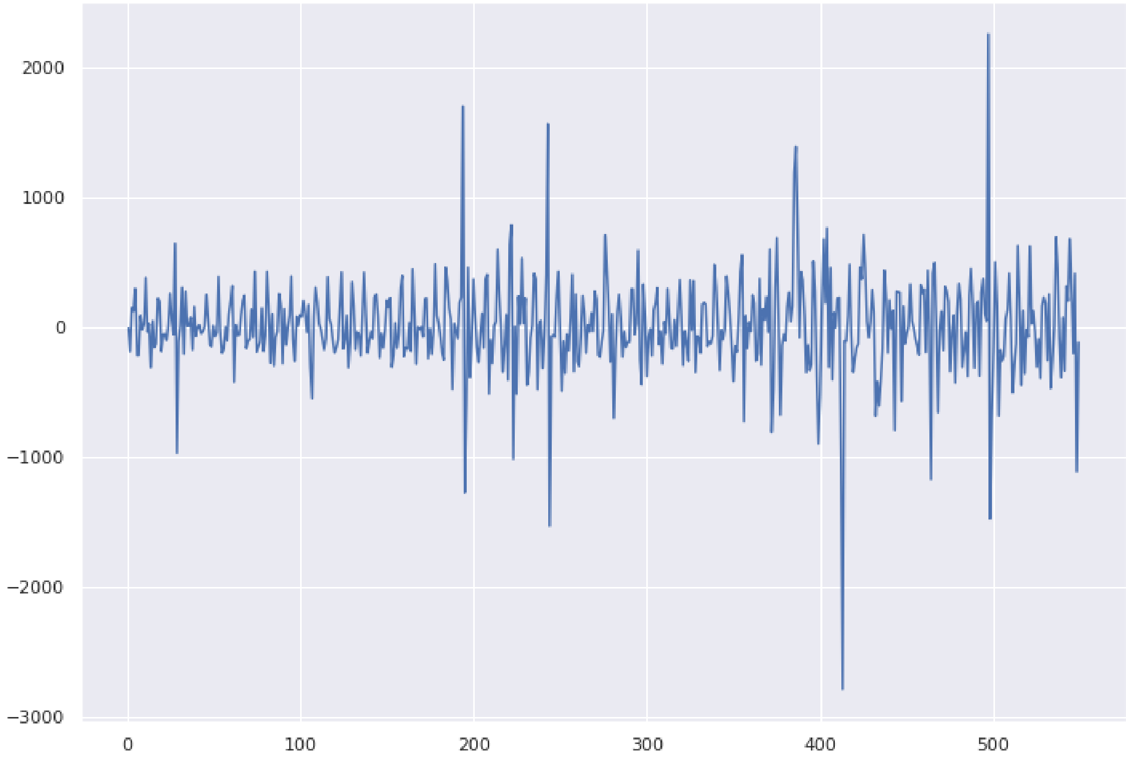

In [ ]:
```
ts_decompose=pd.DataFrame(residual).fillna(0)[0].values
df_test(ts_decompose)
```

```
ADF Stastistic: -3.796320
p-value: 0.002945
```

We can see that aur series is now stationary, we can also try diffrencing to see what results we can get.

# 2. Remove trend and seasonality with differencing

In [ ]:
```
ts_diff = ts - ts.shift(1)
plot.plot(ts_diff.values)
plot.show()
```




In [ ]:
```
ts_diff.dropna(inplace=True)
df_test(ts_diff)
```

```
ADF Stastistic: -8.273590
p-value: 0.000000
```

Also the p value is 0. So we can say that our graph is now stationery. Now we can apply the ARIMA model

**How do we choose p,d,q**

a thumb rule that for choosing the p,q values are when the lag goes below the significant level

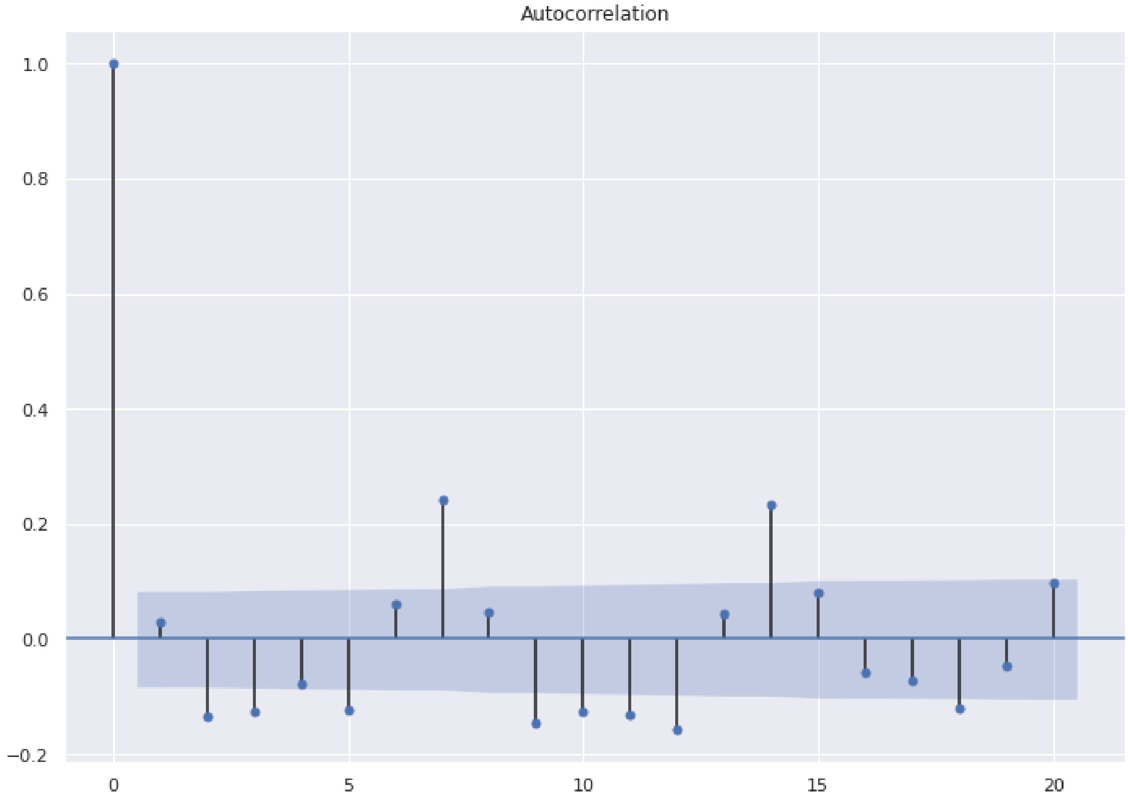
- we use PACF for p, here we see that till lag 5 there are significat lines, if we want our model to be simpler we can start with a smaller number like 3/4
- we use ACF for q. here we can see that lag 4 is below significant level so we will use till lag 3

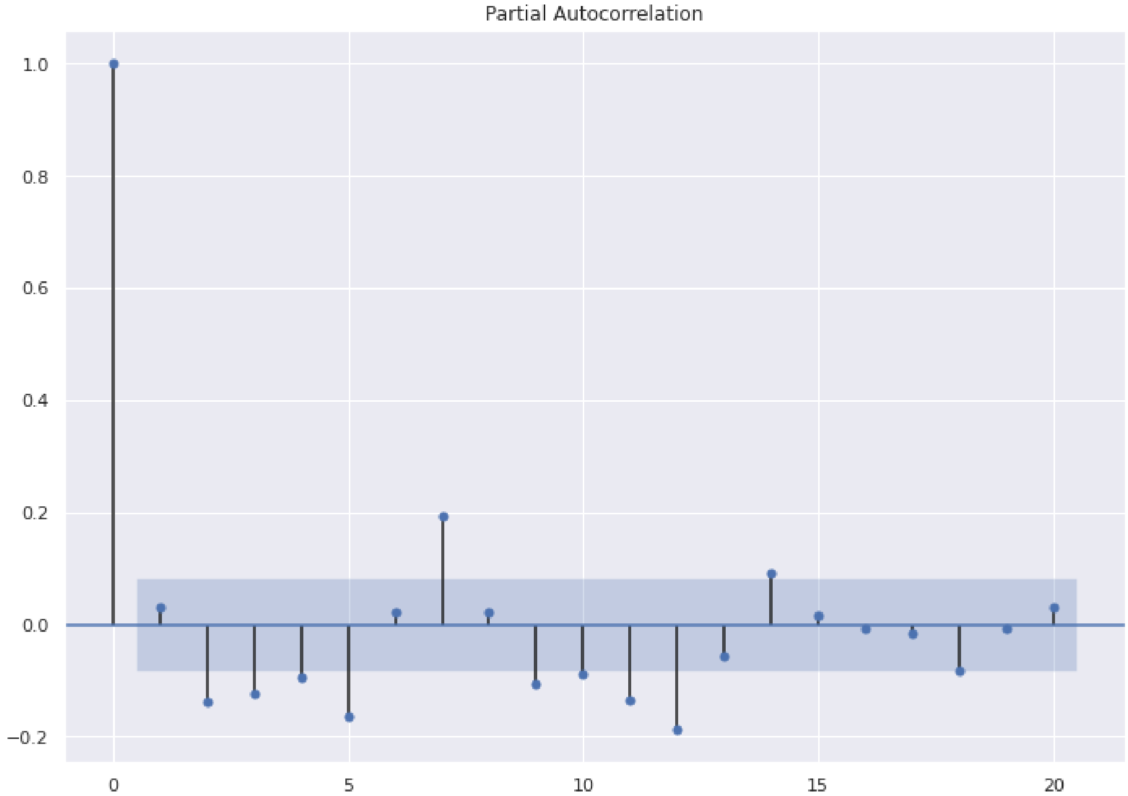
as for d we can see that at 1 diffencing the series becomes stationary so we choose d as 1

# Plot the autocorreltaion and partial auto correlation functions

Plotting the graphs and getting the p,q,d values for arima

In [ ]:
```
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
acf=plot_acf(ts_diff,lags=20)
pacf=plot_pacf(ts_diff,lags=20)
```




[https://people.duke.edu/~rnau/411arim3.htm (https://people.duke.edu/~rnau/411arim3.htm)](https://people.duke.edu/~rnau/411arim3.htm)

In [ ]:

```

```

# ARIMA MODEL

In [ ]:

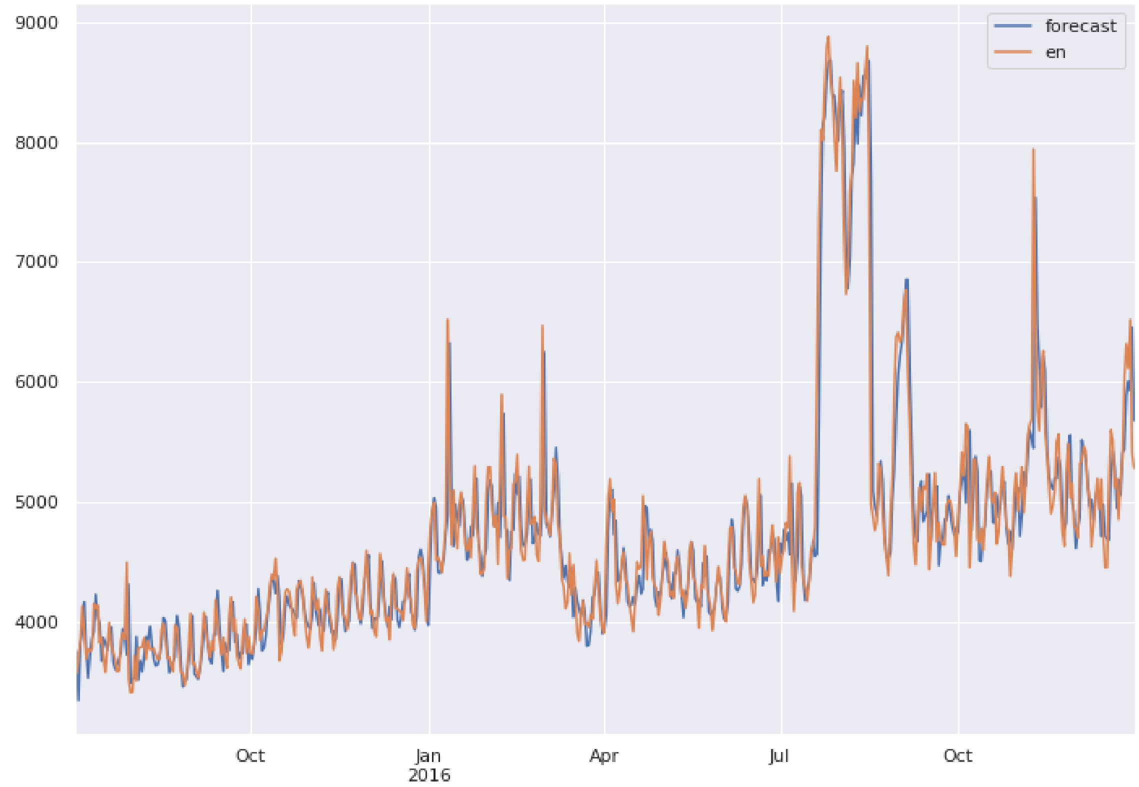
```
from statsmodels.tsa.arima_model import ARIMA
from pandas import DataFrame
```

In [ ]:

```
model = ARIMA(ts, order=(4,1,3))
model_fit = model.fit(disp=0)
```

```
/usr/local/lib/python3.8/dist-packages/statsmodels/tsa/base/tsa_model.py:5
24: ValueWarning: No frequency information was provided, so inferred frequ
ency D will be used.
  warnings.warn('No frequency information was'
/usr/local/lib/python3.8/dist-packages/statsmodels/tsa/base/tsa_model.py:5
24: ValueWarning: No frequency information was provided, so inferred frequ
ency D will be used.
  warnings.warn('No frequency information was'
/usr/local/lib/python3.8/dist-packages/statsmodels/base/model.py:547: Hess
ianInversionWarning: Inverting hessian failed, no bse or cov_params availa
ble
  warnings.warn('Inverting hessian failed, no bse or cov_params '
/usr/local/lib/python3.8/dist-packages/statsmodels/base/model.py:566: Conv
ergenceWarning: Maximum Likelihood optimization failed to converge. Check
mle_retvals
  warnings.warn("Maximum Likelihood optimization failed to "
```

In [ ]:
```
model_fit.plot_predict(dynamic=False)
"""When you set dynamic=True, the model continuously predicts one-step ahea

When you set dynamic=False, the model sequentially predicts one-step-ahead

On your first comparison of plots as you predict from 509 to 533, the reaso

Since out-of-sample approach uses the last predicted value from the previou
"""
plot.show()
```




In [ ]:

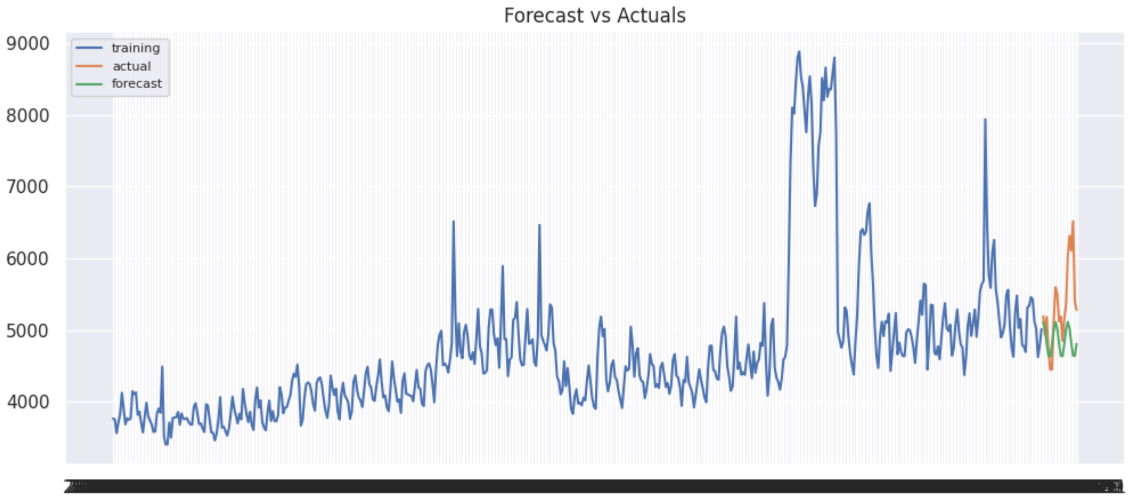
### Multistep forecasting

In [ ]:
```
train = ts[:-20]
test = ts[-20:]
```

In [ ]:
```
model = ARIMA(train, order=(4, 1, 3))
fitted = model.fit(disp=-1)

# Forecast
fc, se, conf = fitted.forecast(20, alpha=0.02)

# Make as pandas series
fc_series = pd.Series(fc, index=test.index)
# Plot
plot.figure(figsize=(12,5), dpi=100)
plot.plot(train, label='training')
plot.plot(test, label='actual')
plot.plot(fc_series, label='forecast')

plot.title('Forecast vs Actuals')
plot.legend(loc='upper left', fontsize=8)
```

```
/usr/local/lib/python3.8/dist-packages/statsmodels/tsa/base/tsa_model.py:5
24: ValueWarning: No frequency information was provided, so inferred frequ
ency D will be used.
  warnings.warn('No frequency information was'
/usr/local/lib/python3.8/dist-packages/statsmodels/tsa/base/tsa_model.py:5
24: ValueWarning: No frequency information was provided, so inferred frequ
ency D will be used.
  warnings.warn('No frequency information was'
/usr/local/lib/python3.8/dist-packages/statsmodels/base/model.py:547: Hess
ianInversionWarning: Inverting hessian failed, no bse or cov_params availa
ble
  warnings.warn('Inverting hessian failed, no bse or cov_params '
/usr/local/lib/python3.8/dist-packages/statsmodels/base/model.py:566: Conv
ergenceWarning: Maximum Likelihood optimization failed to converge. Check
mle_retvals
  warnings.warn("Maximum Likelihood optimization failed to "
```

Out[45]:
```
<matplotlib.legend.Legend at 0x7f9aed53c340>
```




In [ ]:
```
mape = np.mean(np.abs(fc - test.values)/np.abs(test.values))
rmse = np.mean((fc - test.values)**2)**.5
print("mape:",mape)
print("rsme:",rmse)
```

```
mape: 0.0935492560101219
rsme: 706.602453194038
```

we can see that the model does not perform very well for multistep out sample data

from the decomposition we can see that there is a weekly seasonality and still some spikes in the residual, that may be because of some external factors, which we can take into account by using them as our exogenous variable

In [ ]:
```
!gdown 1H9054-eVP9IdANPOblXwX7Nd2r_Sjf1u
```

```
Downloading...
From: https://drive.google.com/uc?id=1H9054-eVP9IdANPOblXwX7Nd2r_Sjf1u (ht
tps://drive.google.com/uc?id=1H9054-eVP9IdANPOblXwX7Nd2r_Sjf1u)
To: /content/Exog_Campaign_eng
100% 1.10k/1.10k [00:00<00:00, 1.79MB/s]
```

In [ ]:
```
ex_df = pd.read_csv('Exog_Campaign_eng')
ex_df.head()
```

Out[48]:

| | Exog |
|---|---|
| **0** | 0 |
| **1** | 0 |
| **2** | 0 |
| **3** | 0 |
| **4** | 0 |

We get the exogenous data from this csv file for english pages

In [ ]:
```
exog=ex_df['Exog'].to_numpy()
```

we will train a sarimax model for that and see if we get anyimprovements from using the two information.

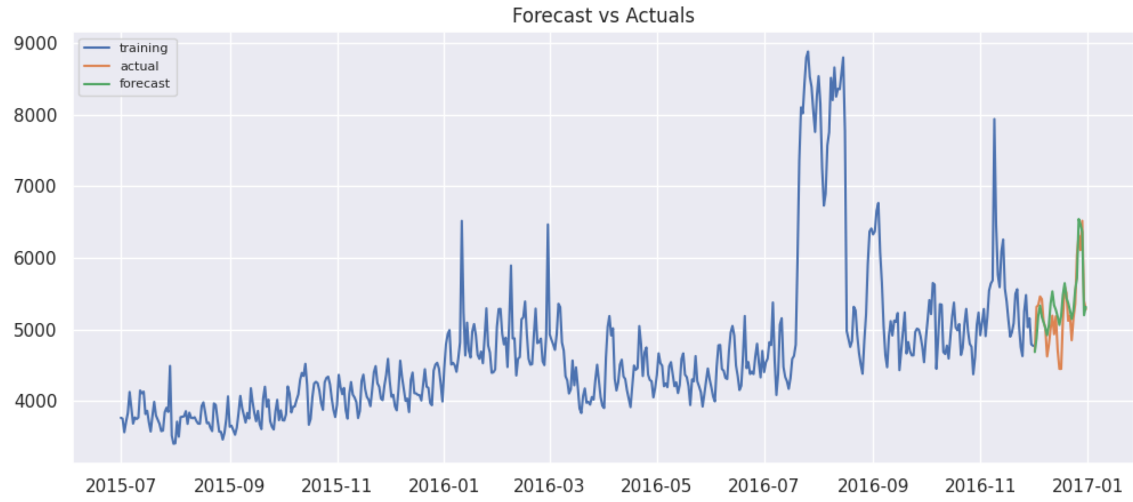
the seasonal order and the values of PDQ are based upon various trials and comparision of the models

- we see a seasonality of 7 from the plots ie: weekly seasonality ( from the plots we can see that afte some insignificant plots we have some significant values repeating at intervals of 7 ie: 7,14 ... )
- the non seasonal order we can keep the same

In [ ]:
```
import statsmodels.api as sm
train=ts[:520]
test=ts[520:]
model=sm.tsa.statespace.SARIMAX(train,order=(4, 1, 3),seasonal_order=(1,1,1
results=model.fit()

fc=results.forecast(30,dynamic=True,exog=pd.DataFrame(exog[520:]))

# Make as pandas series
fc_series = pd.Series(fc)
# Plot
train.index=train.index.astype('datetime64[ns]')
test.index=test.index.astype('datetime64[ns]')
plot.figure(figsize=(12,5), dpi=100)
plot.plot(train, label='training')
plot.plot(test, label='actual')
plot.plot(fc_series, label='forecast')

plot.title('Forecast vs Actuals')
plot.legend(loc='upper left', fontsize=8)
```

```
/usr/local/lib/python3.8/dist-packages/statsmodels/tsa/base/tsa_model.py:5
24: ValueWarning: No frequency information was provided, so inferred frequ
ency D will be used.
  warnings.warn('No frequency information was'
/usr/local/lib/python3.8/dist-packages/statsmodels/tsa/base/tsa_model.py:5
24: ValueWarning: No frequency information was provided, so inferred frequ
ency D will be used.
  warnings.warn('No frequency information was'
/usr/local/lib/python3.8/dist-packages/statsmodels/base/model.py:566: Conv
ergenceWarning: Maximum Likelihood optimization failed to converge. Check
mle_retvals
  warnings.warn("Maximum Likelihood optimization failed to "
```

Out[50]: `<matplotlib.legend.Legend at 0x7f9ae2bd5dc0>`




In [ ]:

In [ ]:
```
mape = np.mean(np.abs(fc - test.values)/np.abs(test.values))
rmse = np.mean((fc - test.values)**2)**.5
print("mape:",mape)
print("rsme:",rmse)
```

```
mape: 0.0476009066291969
rsme: 299.17343793278815
```

The mean absolute percentage error and the root mean squared error is low

# regression for a time series

In [ ]:
```
ts_df=ts.to_frame()
ts_df.head()
```

Out[52]:

| index | en |
|---|---|
| **2015-07-01** | 3767.328604 |
| **2015-07-02** | 3755.158765 |
| **2015-07-03** | 3565.225696 |
| **2015-07-04** | 3711.782932 |
| **2015-07-05** | 3833.433025 |

In [ ]:
```
ts_df.reset_index(level=0, inplace=True)
ts_df['date']=pd.to_datetime(ts_df['index'])
ts_df.drop(['index'],axis=1,inplace=True)
ts_df.head()
```

Out[53]:

| | en | date |
|---|---|---|
| **0** | 3767.328604 | 2015-07-01 |
| **1** | 3755.158765 | 2015-07-02 |
| **2** | 3565.225696 | 2015-07-03 |
| **3** | 3711.782932 | 2015-07-04 |
| **4** | 3833.433025 | 2015-07-05 |

In [ ]:
```
ts_df['day_of_week']=ts_df['date'].dt.day_name()
ts_df.head()
```

Out[54]:

| | en | date | day_of_week |
|---|---|---|---|
| **0** | 3767.328604 | 2015-07-01 | Wednesday |
| **1** | 3755.158765 | 2015-07-02 | Thursday |
| **2** | 3565.225696 | 2015-07-03 | Friday |
| **3** | 3711.782932 | 2015-07-04 | Saturday |
| **4** | 3833.433025 | 2015-07-05 | Sunday |

In [ ]:
```
ts_df=pd.get_dummies(ts_df, columns = ['day_of_week'])
```

In [ ]:
```
ts_df.head()
```

Out[56]:

| | en | date | day_of_week_Friday | day_of_week_Monday | day_of_week_Saturday | da |
|---|---|---|---|---|---|---|
| 0 | 3767.328604 | 2015-07-01 | 0 | 0 | 0 | |
| 1 | 3755.158765 | 2015-07-02 | 0 | 0 | 0 | |
| 2 | 3565.225696 | 2015-07-03 | 1 | 0 | 0 | |
| 3 | 3711.782932 | 2015-07-04 | 0 | 0 | 1 | |
| 4 | 3833.433025 | 2015-07-05 | 0 | 0 | 0 | |

In [ ]:
```
ts_df['exog']=ex_df['Exog']
ts_df['rolling_mean']=ts_df['en'].rolling(7).mean()
```

In [ ]:

In [ ]:
```
ts_df=ts_df.dropna()
ts_df.head()
```

Out[58]:

| | en | date | day_of_week_Friday | day_of_week_Monday | day_of_week_Saturday | d |
|---|---|---|---|---|---|---|
| 6 | 3906.341724 | 2015-07-07 | 0 | 0 | 0 | |
| 7 | 3685.854621 | 2015-07-08 | 0 | 0 | 0 | |
| 8 | 3771.183714 | 2015-07-09 | 0 | 0 | 0 | |
| 9 | 3749.860313 | 2015-07-10 | 1 | 0 | 0 | |
| 10 | 3770.749355 | 2015-07-11 | 0 | 0 | 1 | |

In [ ]:
```
X=ts_df[['day_of_week_Friday',  'day_of_week_Monday',   'day_of_week_Saturd
y=ts_df[['en']]

train_x = X[:-20]
test_x = X[-20:]

train_y = y[:-20]
test_y = y[-20:]
```

In [ ]:
```
from sklearn.linear_model import LinearRegression

# Train and pred
model = LinearRegression()
model.fit(train_x, train_y)
y_pred = (model.predict(test_x))


mape = np.mean(np.abs(y_pred - test_y.values)/np.abs(test_y.values))
print("mape:",mape)
```

```
mape: 0.04523968736329716
```

We can see here that aur mape is better than our arima model but worse than our sarimax model

- Linear Regression Is Limited to Linear Relationships and in our case there is not a lot of linear relationship.
- it would have been better to use a regression based model for forecasting if we can build some better features.
- we have our series data and the exogenous variables, we add the day of week feature, other than that there are not a lot of features that we can build

# using Facebook Prophet

In [ ]:
```
!pip install pystan~=2.14
!pip install fbprophet
```

```
Looking in indexes: https://pypi.org/simple, (https://pypi.org/simple,) ht
tps://us-python.pkg.dev/colab-wheels/public/simple/ (https://us-python.pk
g.dev/colab-wheels/public/simple/)
Requirement already satisfied: pystan~=2.14 in /usr/local/lib/python3.7/di
st-packages (2.19.1.1)
Requirement already satisfied: Cython!=0.25.1,>=0.22 in /usr/local/lib/pyt
hon3.7/dist-packages (from pystan~=2.14) (0.29.32)
Requirement already satisfied: numpy>=1.7 in /usr/local/lib/python3.7/dist
-packages (from pystan~=2.14) (1.21.6)
Looking in indexes: https://pypi.org/simple, (https://pypi.org/simple,) ht
tps://us-python.pkg.dev/colab-wheels/public/simple/ (https://us-python.pk
g.dev/colab-wheels/public/simple/)
Requirement already satisfied: fbprophet in /usr/local/lib/python3.7/dist-
packages (0.7.1)
Requirement already satisfied: holidays>=0.10.2 in /usr/local/lib/python3.
7/dist-packages (from fbprophet) (0.14.2)
Requirement already satisfied: cmdstanpy==0.9.5 in /usr/local/lib/python3.
7/dist-packages (from fbprophet) (0.9.5)
Requirement already satisfied: convertdate>=2.1.2 in /usr/local/lib/python
3.7/dist-packages (from fbprophet) (2.4.0)
Requirement already satisfied: matplotlib>=2.0.0 in /usr/local/lib/python
3.7/dist-packages (from fbprophet) (3.2.2)
Requirement already satisfied: setuptools-git>=1.2 in /usr/local/lib/pytho
n3.7/dist-packages (from fbprophet) (1.2)
Requirement already satisfied: numpy>=1.15.4 in /usr/local/lib/python3.7/d
ist-packages (from fbprophet) (1.21.6)
Requirement already satisfied: tqdm>=4.36.1 in /usr/local/lib/python3.7/di
st-packages (from fbprophet) (4.64.0)
Requirement already satisfied: pystan>=2.14 in /usr/local/lib/python3.7/di
st-packages (from fbprophet) (2.19.1.1)
Requirement already satisfied: python-dateutil>=2.8.0 in /usr/local/lib/py
thon3.7/dist-packages (from fbprophet) (2.8.2)
Requirement already satisfied: Cython>=0.22 in /usr/local/lib/python3.7/di
st-packages (from fbprophet) (0.29.32)
Requirement already satisfied: LunarCalendar>=0.0.9 in /usr/local/lib/pyth
on3.7/dist-packages (from fbprophet) (0.0.9)
Requirement already satisfied: pandas>=1.0.4 in /usr/local/lib/python3.7/d
ist-packages (from fbprophet) (1.3.5)
Requirement already satisfied: pymeeus<=1,>=0.3.13 in /usr/local/lib/pytho
n3.7/dist-packages (from convertdate>=2.1.2->fbprophet) (0.5.11)
Requirement already satisfied: hijri-converter in /usr/local/lib/python3.
7/dist-packages (from holidays>=0.10.2->fbprophet) (2.2.4)
Requirement already satisfied: korean-lunar-calendar in /usr/local/lib/pyt
hon3.7/dist-packages (from holidays>=0.10.2->fbprophet) (0.2.1)
Requirement already satisfied: pytz in /usr/local/lib/python3.7/dist-packa
ges (from LunarCalendar>=0.0.9->fbprophet) (2022.2.1)
Requirement already satisfied: ephem>=3.7.5.3 in /usr/local/lib/python3.7/
dist-packages (from LunarCalendar>=0.0.9->fbprophet) (4.1.3)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/di
st-packages (from matplotlib>=2.0.0->fbprophet) (0.11.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python
3.7/dist-packages (from matplotlib>=2.0.0->fbprophet) (1.4.4)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in
/usr/local/lib/python3.7/dist-packages (from matplotlib>=2.0.0->fbprophet)
(3.0.9)
Requirement already satisfied: typing-extensions in /usr/local/lib/python
3.7/dist-packages (from kiwisolver>=1.0.1->matplotlib>=2.0.0->fbprophet)
(4.1.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-p
ackages (from python-dateutil>=2.8.0->fbprophet) (1.15.0)
```

In [ ]:
```
ts_df['ds']=ts_df['date']
ts_df['y']=ts_df['en']
```

In [ ]:
```
df2=ts_df[['date','en','exog']].copy()
df2.columns = ['ds', 'y', 'exog']
df2.head()
```

Out[174]:

| | ds | y | exog |
|---|---|---|---|
| **6** | 2015-07-07 | 3906.341724 | 0 |
| **7** | 2015-07-08 | 3685.854621 | 0 |
| **8** | 2015-07-09 | 3771.183714 | 0 |
| **9** | 2015-07-10 | 3749.860313 | 0 |
| **10** | 2015-07-11 | 3770.749355 | 0 |

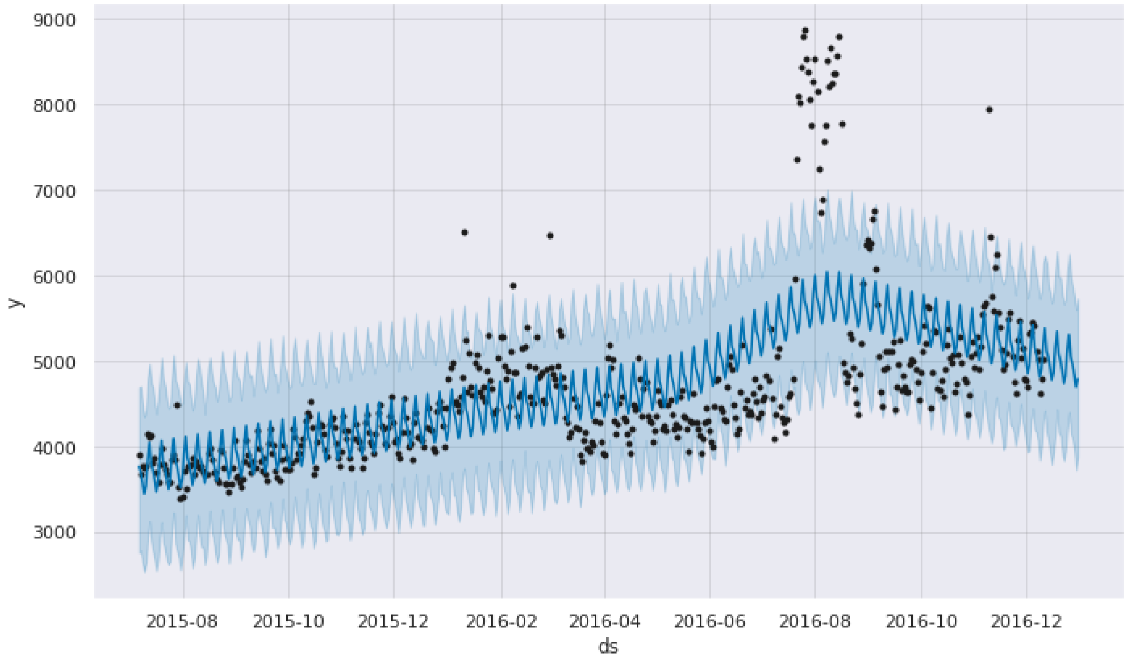
In [ ]:
```
df2[:-20].info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 524 entries, 6 to 529
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   ds      524 non-null    datetime64[ns]
 1   y       524 non-null    float64
 2   exog    524 non-null    int64
dtypes: datetime64[ns](1), float64(1), int64(1)
memory usage: 16.4 KB
```

prophet without exogenous

In [ ]:
```
from fbprophet import Prophet
m = Prophet(weekly_seasonality=True)
m.fit(df2[['ds', 'y']][:-20])
future = m.make_future_dataframe(periods=20,freq="D")
forecast = m.predict(future)
fig = m.plot(forecast)
```

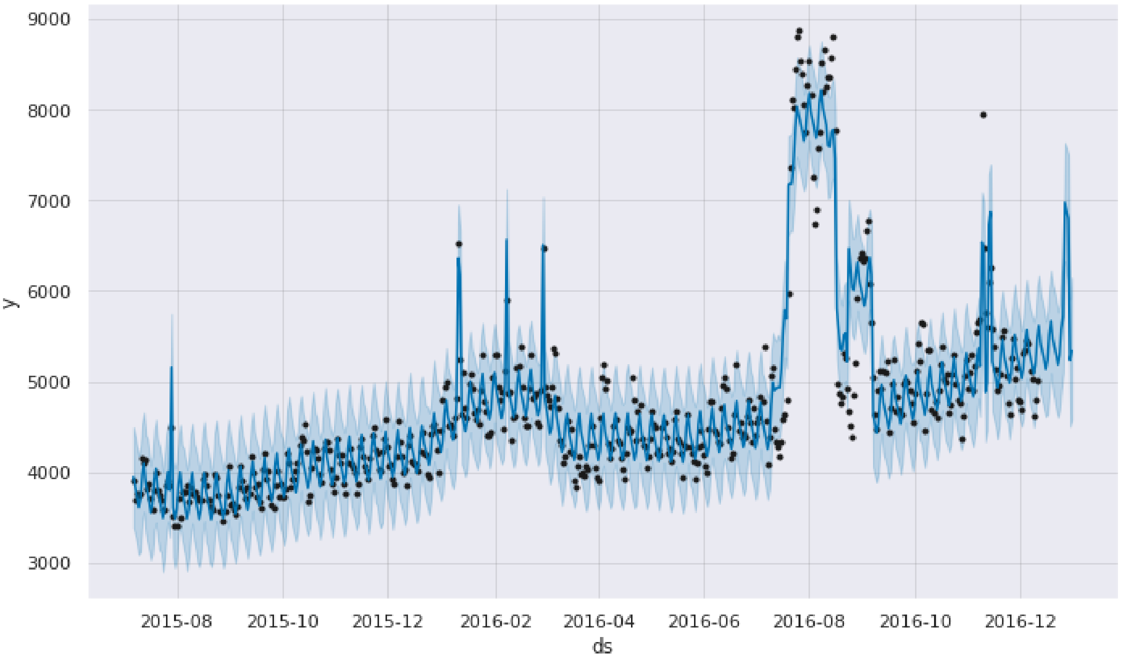
```
INFO:fbprophet:Disabling yearly seasonality. Run prophet with yearly_seaso
nality=True to override this.
INFO:fbprophet:Disabling daily seasonality. Run prophet with daily_seasona
lity=True to override this.
```



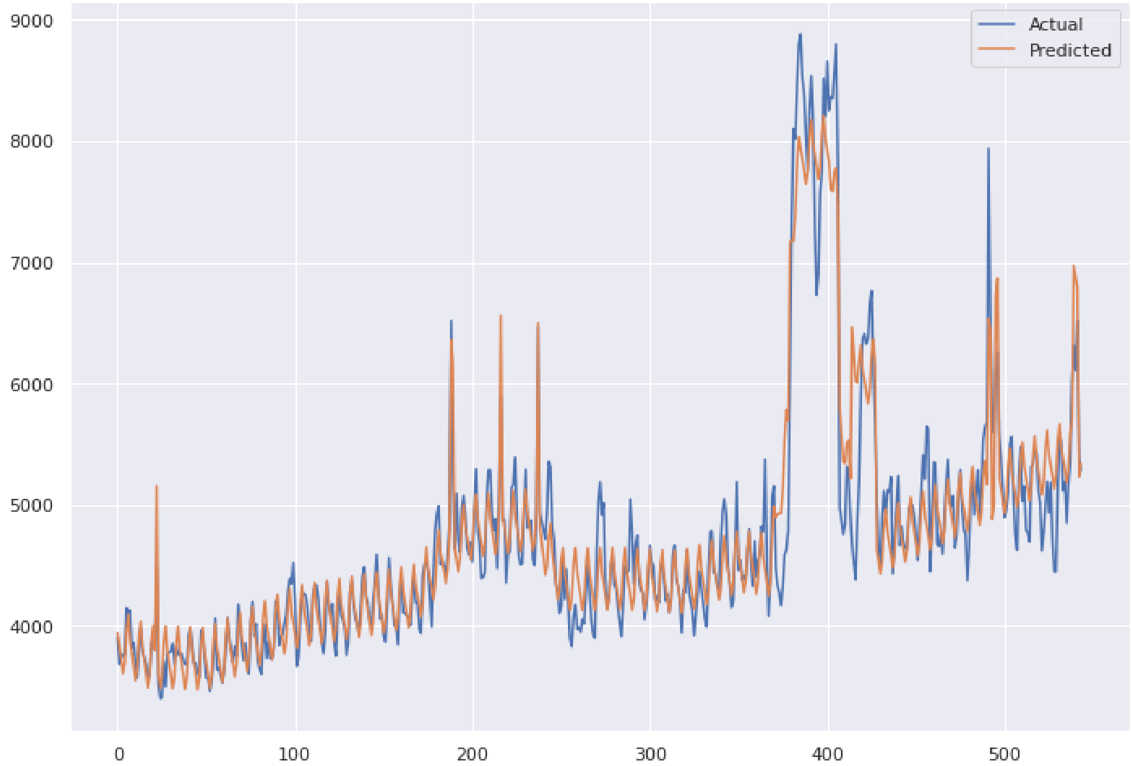

prophet with exogenous

In [ ]:
```
model2=Prophet(interval_width=0.9, weekly_seasonality=True, changepoint_pri
model2.add_regressor('exog')
model2.fit(df2[:-20])
forecast2 = model2.predict(df2)
fig = model2.plot(forecast2)
```

```
INFO:fbprophet:Disabling yearly seasonality. Run prophet with yearly_seaso
nality=True to override this.
INFO:fbprophet:Disabling daily seasonality. Run prophet with daily_seasona
lity=True to override this.
```



In [ ]:
```
y_true = df2['y'].values
y_pred = forecast2['yhat'].values

plot.plot(y_true, label='Actual')
plot.plot(y_pred, label='Predicted')
plot.legend()
plot.show()
```




In [ ]:
```
mape = np.mean(np.abs(forecast2['yhat'][-20:] - df2['y'][-20:].values)/np.a
print("mape:",mape)
```

```
mape: 0.06592815614410931
```

- Prophet does not perform well on non-stationary data because it is difficult to find the actual seasonality and trend of the data if the patterns are inconsistent.

**--------------------------------------------------------------------------------------------------------------------------------------------**

# Comparing the predicted views for different languages

**For doing this we are going to automate the procedure from loading the separate data for each langauge to doing out of sample forecasting for the next month, and then comparing the results.**

In [ ]:
```
def grid_search(ts):
    v=[0,1,2,3]
    mape=100
    val=[0,0,0]
    for p in v:
        for d in v:
            for q in v:
                try:
                    model = ARIMA(ts[:-20], order=(p,d,q))
                    model_fit = model.fit(disp=-1)
                    fc, se, conf = model_fit.forecast(20, alpha=0.02)
                    x = np.mean(np.abs(fc - ts[-20:].values)/np.abs(ts[-20:
                    if(x<mape):
                        mape=x
                        val=[p,d,q]

                except:
                    pass

    return(mape, val)
```

This functions works like a grid search for getting the best value of p,d,q by comparing the mape of all models that we create.

the values of p,d,q that give the least mape score are saved and returned

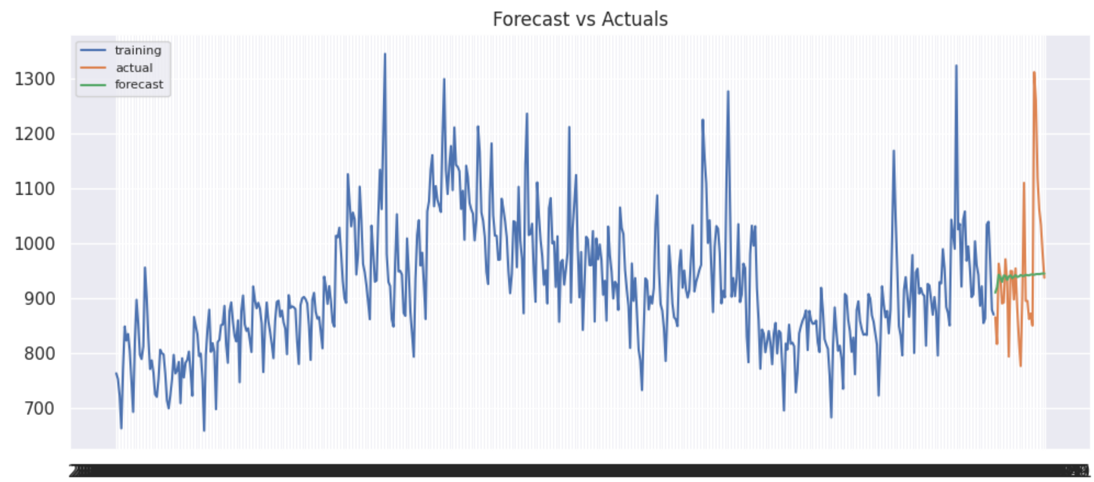
In [ ]:
```
def all_arima(train,test,val):
    model = ARIMA(train, order=(val[0], val[1], val[2]))
    fitted = model.fit(disp=-1)

  # Forecast
    fc, se, conf = fitted.forecast(30, alpha=0.02)



    fc_series = pd.Series(fc, index=test.index)

  # Plot
    plot.figure(figsize=(12,5), dpi=100)
    plot.plot(train, label='training')
    plot.plot(test, label='actual')
    plot.plot(fc_series, label='forecast')
    plot.title('Forecast vs Actuals')
    plot.legend(loc='upper left', fontsize=8)
    plot.show()
    mape = np.mean(np.abs(fc - test.values)/np.abs(test.values))
    rmse = np.mean((fc - test.values)**2)**.5
    print("mape:",mape)
    print("rsme:",rmse)
    return (fc)
```

This function takes the p,d,q values that we calculated earlier and then trains a model on it, does forecast and plots them for visualization.

it also calculates the sum of forecased views for the next 30 days and returns it back

In [ ]:
```
import warnings
warnings.filterwarnings("ignore")
views_prediction={}
for c in total_view:
    print("language: ",c)
    ts=(total_view[c])
    mape,val=grid_search(ts)
    print(mape,val)
    train = ts[:520]
    test = ts[520:]
    fc=all_arima(train,test,val)
    views_prediction[c]=fc
```

```
language:  de
0.09397758421308047 [3, 1, 3]
```




```
mape: 0.08451930259659801
```

- This function is what calls and drives all the other functions.
- It first gets the data for a particular language.
- checks stationarity.
- Gets the optimal p,d,q values from grid search
- uses that value to train the model, forecast and plot it