

# Adease\_Time\_Series

July 31, 2024

```
[5]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from datetime import datetime
import seaborn as sns
import seaborn.objects as so
import re
from itertools import product

from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.statespace.sarimax import SARIMAX
from prophet import Prophet

sns.set(style = 'darkgrid')
pd.set_option('display.max_columns', None)
pd.options.display.max_colwidth = 100
plt.rcParams["figure.figsize"] = (15,7)
import warnings # supress warnings
warnings.filterwarnings('ignore')
```

```
[1]: !gdown 10rQ7yyQzKdpF1E5nLnJvIgOIjP8JOQxP
```

Downloading...

From (original):

<https://drive.google.com/uc?id=10rQ7yyQzKdpF1E5nLnJvIgOIjP8JOQxP>

From (redirected): <https://drive.google.com/uc?id=10rQ7yyQzKdpF1E5nLnJvIgOIjP8JOQxP&confirm=t&uuiid=93b6461b-afb8-49d1-ac14-cbdc1383fd4e>

To: /content/train\_1.csv

100% 278M/278M [00:05<00:00, 53.0MB/s]

```
[2]: !gdown 179_whZdRlKUQsdfh5wX05GamAqo0lqvn
```

Downloading...

From: [https://drive.google.com/uc?id=179\\_whZdRlKUQsdfh5wX05GamAqo0lqvn](https://drive.google.com/uc?id=179_whZdRlKUQsdfh5wX05GamAqo0lqvn)  
 To: /content/Exog\_Campaign\_eng  
 100% 1.10k/1.10k [00:00<00:00, 3.70MB/s]

```
[6]: data = pd.read_csv('/content/train_1.csv')
     exog = pd.read_csv('/content/Exog_Campaign_eng')
```

```
[7]: raw_data = data.copy(deep=True)
```

```
[8]: data.head()
```

```
[8]:
```

		Page	2015-07-01	\
0	2NE1_zh.wikipedia.org_all-access_spider		18.0	
1	2PM_zh.wikipedia.org_all-access_spider		11.0	
2	3C_zh.wikipedia.org_all-access_spider		1.0	
3	4minute_zh.wikipedia.org_all-access_spider		35.0	
4	52_Hz_I_Love_You_zh.wikipedia.org_all-access_spider		NaN	

	2015-07-02	2015-07-03	2015-07-04	2015-07-05	2015-07-06	2015-07-07	\
0	11.0	5.0	13.0	14.0	9.0	9.0	
1	14.0	15.0	18.0	11.0	13.0	22.0	
2	0.0	1.0	1.0	0.0	4.0	0.0	
3	13.0	10.0	94.0	4.0	26.0	14.0	
4	NaN	NaN	NaN	NaN	NaN	NaN	

	2015-07-08	2015-07-09	2015-07-10	2015-07-11	2015-07-12	2015-07-13	\
0	22.0	26.0	24.0	19.0	10.0	14.0	
1	11.0	10.0	4.0	41.0	65.0	57.0	
2	3.0	4.0	4.0	1.0	1.0	1.0	
3	9.0	11.0	16.0	16.0	11.0	23.0	
4	NaN	NaN	NaN	NaN	NaN	NaN	

	2015-07-14	2015-07-15	2015-07-16	2015-07-17	2015-07-18	2015-07-19	\
0	15.0	8.0	16.0	8.0	8.0	16.0	
1	38.0	20.0	62.0	44.0	15.0	10.0	
2	6.0	8.0	6.0	4.0	5.0	1.0	
3	145.0	14.0	17.0	85.0	4.0	30.0	
4	NaN	NaN	NaN	NaN	NaN	NaN	

	2015-07-20	2015-07-21	2015-07-22	2015-07-23	2015-07-24	2015-07-25	\
0	7.0	11.0	10.0	20.0	18.0	15.0	
1	47.0	24.0	17.0	22.0	9.0	39.0	
2	2.0	3.0	8.0	8.0	6.0	6.0	
3	22.0	9.0	10.0	11.0	7.0	7.0	
4	NaN	NaN	NaN	NaN	NaN	NaN	

	2015-07-26	2015-07-27	2015-07-28	2015-07-29	2015-07-30	2015-07-31	\
--	------------	------------	------------	------------	------------	------------	---

0	14.0	49.0	10.0	16.0	18.0	8.0	
1	13.0	11.0	12.0	21.0	19.0	9.0	
2	2.0	2.0	3.0	2.0	4.0	3.0	
3	11.0	9.0	11.0	44.0	8.0	14.0	
4	NaN	NaN	NaN	NaN	NaN	NaN	
	2015-08-01	2015-08-02	2015-08-03	2015-08-04	2015-08-05	2015-08-06	\
0	5.0	9.0	7.0	13.0	9.0	7.0	
1	15.0	33.0	8.0	8.0	7.0	13.0	
2	3.0	5.0	3.0	5.0	4.0	2.0	
3	19.0	10.0	17.0	17.0	10.0	7.0	
4	NaN	NaN	NaN	NaN	NaN	NaN	
	2015-08-07	2015-08-08	2015-08-09	2015-08-10	2015-08-11	2015-08-12	\
0	4.0	11.0	10.0	5.0	9.0	9.0	
1	2.0	23.0	12.0	27.0	27.0	36.0	
2	5.0	1.0	4.0	5.0	0.0	0.0	
3	10.0	1.0	8.0	27.0	19.0	16.0	
4	NaN	NaN	NaN	NaN	NaN	NaN	
	2015-08-13	2015-08-14	2015-08-15	2015-08-16	2015-08-17	2015-08-18	\
0	9.0	9.0	13.0	4.0	15.0	25.0	
1	23.0	58.0	80.0	60.0	69.0	42.0	
2	7.0	3.0	5.0	1.0	6.0	2.0	
3	2.0	84.0	22.0	14.0	47.0	25.0	
4	NaN	NaN	NaN	NaN	NaN	NaN	
	2015-08-19	2015-08-20	2015-08-21	2015-08-22	2015-08-23	2015-08-24	\
0	9.0	5.0	6.0	20.0	3.0	14.0	
1	161.0	94.0	77.0	78.0	20.0	24.0	
2	5.0	0.0	3.0	1.0	0.0	1.0	
3	14.0	11.0	12.0	27.0	8.0	17.0	
4	NaN	NaN	NaN	NaN	NaN	NaN	
	2015-08-25	2015-08-26	2015-08-27	2015-08-28	2015-08-29	2015-08-30	\
0	46.0	5.0	5.0	13.0	4.0	9.0	
1	13.0	14.0	26.0	8.0	82.0	22.0	
2	1.0	2.0	4.0	2.0	1.0	1.0	
3	43.0	3.0	19.0	14.0	20.0	43.0	
4	NaN	NaN	NaN	NaN	NaN	NaN	
	2015-08-31	2015-09-01	2015-09-02	2015-09-03	2015-09-04	2015-09-05	\
0	10.0	9.0	11.0	11.0	11.0	9.0	
1	11.0	81.0	37.0	9.0	40.0	47.0	
2	3.0	4.0	3.0	6.0	6.0	4.0	
3	4.0	5.0	37.0	23.0	14.0	12.0	
4	NaN	NaN	NaN	NaN	NaN	NaN	

	2015-09-06	2015-09-07	2015-09-08	2015-09-09	2015-09-10	2015-09-11	\
0	15.0	5.0	10.0	7.0	4.0	8.0	
1	18.0	23.0	6.0	2.0	7.0	16.0	
2	3.0	3.0	2.0	9.0	7.0	2.0	
3	13.0	22.0	12.0	12.0	6.0	27.0	
4	NaN	NaN	NaN	NaN	NaN	NaN	

	2015-09-12	2015-09-13	2015-09-14	2015-09-15	2015-09-16	2015-09-17	\
0	9.0	10.0	6.0	13.0	16.0	6.0	
1	10.0	34.0	14.0	31.0	20.0	23.0	
2	3.0	1.0	3.0	1.0	6.0	7.0	
3	5.0	7.0	24.0	8.0	9.0	10.0	
4	NaN	NaN	NaN	NaN	NaN	NaN	

	2015-09-18	2015-09-19	2015-09-20	2015-09-21	2015-09-22	2015-09-23	\
0	24.0	9.0	11.0	12.0	8.0	14.0	
1	14.0	16.0	34.0	15.0	30.0	13.0	
2	1.0	2.0	5.0	2.0	3.0	8.0	
3	12.0	19.0	7.0	7.0	18.0	15.0	
4	NaN	NaN	NaN	NaN	NaN	NaN	

	2015-09-24	2015-09-25	2015-09-26	2015-09-27	2015-09-28	2015-09-29	\
0	6.0	6.0	11.0	14.0	6.0	10.0	
1	30.0	15.0	25.0	17.0	8.0	12.0	
2	5.0	0.0	4.0	1.0	5.0	3.0	
3	7.0	9.0	10.0	9.0	14.0	8.0	
4	NaN	NaN	NaN	NaN	NaN	NaN	

	2015-09-30	2015-10-01	2015-10-02	2015-10-03	2015-10-04	2015-10-05	\
0	20.0	7.0	15.0	8.0	15.0	5.0	
1	17.0	10.0	21.0	18.0	30.0	13.0	
2	0.0	1.0	8.0	2.0	1.0	3.0	
3	17.0	6.0	8.0	7.0	5.0	3.0	
4	NaN	NaN	NaN	NaN	NaN	NaN	

	2015-10-06	2015-10-07	2015-10-08	2015-10-09	2015-10-10	2015-10-11	\
0	8.0	8.0	5.0	11.0	165.0	34.0	
1	7.0	15.0	23.0	20.0	15.0	9.0	
2	0.0	0.0	5.0	3.0	3.0	0.0	
3	9.0	5.0	6.0	8.0	8.0	11.0	
4	NaN	NaN	NaN	NaN	NaN	NaN	

	2015-10-12	2015-10-13	2015-10-14	2015-10-15	2015-10-16	2015-10-17	\
0	6.0	13.0	8.0	9.0	11.0	26.0	
1	47.0	14.0	11.0	16.0	12.0	7.0	
2	2.0	5.0	2.0	5.0	10.0	5.0	

3	6.0	7.0	28.0	15.0	8.0	7.0	
4	NaN	NaN	NaN	NaN	NaN	NaN	
	2015-10-18	2015-10-19	2015-10-20	2015-10-21	2015-10-22	2015-10-23	\
0	18.0	3.0	5.0	12.0	6.0	16.0	
1	15.0	14.0	12.0	18.0	29.0	39.0	
2	6.0	1.0	4.0	4.0	1.0	3.0	
3	7.0	12.0	5.0	11.0	3.0	7.0	
4	NaN	NaN	NaN	NaN	NaN	NaN	
	2015-10-24	2015-10-25	2015-10-26	2015-10-27	2015-10-28	2015-10-29	\
0	19.0	9.0	10.0	11.0	11.0	7.0	
1	11.0	14.0	28.0	17.0	20.0	17.0	
2	13.0	2.0	1.0	3.0	2.0	1.0	
3	23.0	6.0	3.0	8.0	8.0	39.0	
4	NaN	NaN	NaN	NaN	NaN	NaN	
	2015-10-30	2015-10-31	2015-11-01	2015-11-02	2015-11-03	2015-11-04	\
0	9.0	10.0	24.0	6.0	6.0	8.0	
1	36.0	13.0	11.0	14.0	14.0	14.0	
2	10.0	5.0	6.0	2.0	5.0	2.0	
3	4.0	10.0	6.0	8.0	9.0	16.0	
4	NaN	NaN	NaN	NaN	NaN	NaN	
	2015-11-05	2015-11-06	2015-11-07	2015-11-08	2015-11-09	2015-11-10	\
0	16.0	13.0	10.0	10.0	6.0	5.0	
1	33.0	14.0	13.0	18.0	13.0	11.0	
2	2.0	3.0	2.0	6.0	3.0	2.0	
3	9.0	8.0	8.0	7.0	5.0	5.0	
4	NaN	NaN	NaN	NaN	NaN	NaN	
	2015-11-11	2015-11-12	2015-11-13	2015-11-14	2015-11-15	2015-11-16	\
0	20.0	6.0	47.0	9.0	9.0	12.0	
1	8.0	10.0	11.0	81.0	14.0	20.0	
2	1.0	2.0	3.0	1.0	1.0	2.0	
3	12.0	8.0	15.0	9.0	12.0	5.0	
4	NaN	NaN	NaN	NaN	NaN	NaN	
	2015-11-17	2015-11-18	2015-11-19	2015-11-20	2015-11-21	2015-11-22	\
0	11.0	17.0	15.0	14.0	11.0	97.0	
1	6.0	16.0	18.0	9.0	12.0	10.0	
2	2.0	3.0	2.0	2.0	5.0	7.0	
3	7.0	6.0	12.0	7.0	6.0	33.0	
4	NaN	NaN	NaN	NaN	NaN	NaN	
	2015-11-23	2015-11-24	2015-11-25	2015-11-26	2015-11-27	2015-11-28	\
0	11.0	12.0	11.0	14.0	15.0	12.0	

1	8.0	11.0	14.0	47.0	13.0	13.0	
2	2.0	3.0	4.0	6.0	1.0	3.0	
3	5.0	11.0	6.0	4.0	32.0	9.0	
4	NaN	NaN	NaN	NaN	NaN	NaN	
	2015-11-29	2015-11-30	2015-12-01	2015-12-02	2015-12-03	2015-12-04	\
0	104.0	5.0	22.0	45.0	75.0	29.0	
1	6.0	10.0	8.0	8.0	8.0	18.0	
2	6.0	3.0	3.0	4.0	2.0	2.0	
3	17.0	2.0	10.0	10.0	5.0	7.0	
4	NaN	NaN	NaN	NaN	NaN	NaN	
	2015-12-05	2015-12-06	2015-12-07	2015-12-08	2015-12-09	2015-12-10	\
0	34.0	20.0	12.0	25.0	9.0	62.0	
1	31.0	16.0	15.0	10.0	13.0	9.0	
2	4.0	3.0	1.0	5.0	5.0	4.0	
3	11.0	8.0	10.0	6.0	17.0	11.0	
4	NaN	NaN	NaN	NaN	NaN	NaN	
	2015-12-11	2015-12-12	2015-12-13	2015-12-14	2015-12-15	2015-12-16	\
0	20.0	19.0	8.0	23.0	13.0	16.0	
1	32.0	161.0	6.0	20.0	8.0	11.0	
2	2.0	4.0	5.0	4.0	2.0	1.0	
3	20.0	11.0	15.0	18.0	10.0	15.0	
4	NaN	NaN	NaN	NaN	NaN	NaN	
	2015-12-17	2015-12-18	2015-12-19	2015-12-20	2015-12-21	2015-12-22	\
0	34.0	36.0	11.0	18.0	12.0	24.0	
1	13.0	8.0	19.0	7.0	9.0	16.0	
2	6.0	1.0	1.0	3.0	1.0	3.0	
3	12.0	12.0	12.0	8.0	13.0	9.0	
4	NaN	NaN	NaN	NaN	NaN	NaN	
	2015-12-23	2015-12-24	2015-12-25	2015-12-26	2015-12-27	2015-12-28	\
0	30.0	27.0	44.0	35.0	53.0	11.0	
1	11.0	6.0	38.0	11.0	17.0	13.0	
2	5.0	3.0	3.0	0.0	5.0	3.0	
3	11.0	4.0	12.0	9.0	6.0	12.0	
4	NaN	NaN	NaN	NaN	NaN	NaN	
	2015-12-29	2015-12-30	2015-12-31	2016-01-01	2016-01-02	2016-01-03	\
0	26.0	13.0	18.0	9.0	16.0	6.0	
1	12.0	12.0	9.0	7.0	15.0	14.0	
2	2.0	2.0	2.0	2.0	0.0	3.0	
3	9.0	9.0	6.0	7.0	7.0	11.0	
4	NaN	NaN	NaN	NaN	NaN	NaN	

	2016-01-04	2016-01-05	2016-01-06	2016-01-07	2016-01-08	2016-01-09	\
0	19.0	20.0	19.0	22.0	30.0	14.0	
1	14.0	11.0	13.0	12.0	12.0	24.0	
2	3.0	3.0	4.0	4.0	8.0	3.0	
3	7.0	14.0	9.0	21.0	9.0	10.0	
4	NaN	NaN	NaN	NaN	NaN	NaN	
	2016-01-10	2016-01-11	2016-01-12	2016-01-13	2016-01-14	2016-01-15	\
0	16.0	22.0	15.0	15.0	26.0	16.0	
1	15.0	38.0	18.0	26.0	15.0	12.0	
2	5.0	8.0	1.0	4.0	0.0	3.0	
3	13.0	10.0	13.0	16.0	8.0	10.0	
4	NaN	NaN	NaN	NaN	NaN	NaN	
	2016-01-16	2016-01-17	2016-01-18	2016-01-19	2016-01-20	2016-01-21	\
0	13.0	27.0	18.0	13.0	32.0	31.0	
1	14.0	40.0	19.0	13.0	39.0	19.0	
2	6.0	3.0	1.0	3.0	3.0	3.0	
3	7.0	13.0	18.0	8.0	50.0	8.0	
4	NaN	NaN	NaN	NaN	NaN	NaN	
	2016-01-22	2016-01-23	2016-01-24	2016-01-25	2016-01-26	2016-01-27	\
0	16.0	38.0	18.0	9.0	14.0	10.0	
1	16.0	19.0	11.0	76.0	14.0	19.0	
2	1.0	3.0	8.0	4.0	3.0	2.0	
3	33.0	6.0	22.0	9.0	84.0	28.0	
4	NaN	NaN	NaN	NaN	NaN	NaN	
	2016-01-28	2016-01-29	2016-01-30	2016-01-31	2016-02-01	2016-02-02	\
0	24.0	8.0	15.0	18.0	10.0	23.0	
1	26.0	19.0	17.0	30.0	17.0	17.0	
2	5.0	6.0	3.0	6.0	5.0	6.0	
3	11.0	7.0	14.0	16.0	49.0	71.0	
4	NaN	NaN	NaN	NaN	NaN	NaN	
	2016-02-03	2016-02-04	2016-02-05	2016-02-06	2016-02-07	2016-02-08	\
0	17.0	11.0	26.0	14.0	8.0	12.0	
1	17.0	19.0	11.0	175.0	10.0	5.0	
2	7.0	3.0	1.0	5.0	1.0	2.0	
3	29.0	22.0	6.0	34.0	16.0	14.0	
4	NaN	NaN	NaN	NaN	NaN	NaN	
	2016-02-09	2016-02-10	2016-02-11	2016-02-12	2016-02-13	2016-02-14	\
0	9.0	11.0	34.0	17.0	29.0	11.0	
1	12.0	7.0	12.0	14.0	19.0	11.0	
2	0.0	1.0	4.0	3.0	3.0	9.0	
3	9.0	12.0	24.0	18.0	8.0	26.0	

4	NaN	NaN	NaN	NaN	NaN	NaN	
	2016-02-15	2016-02-16	2016-02-17	2016-02-18	2016-02-19	2016-02-20	\
0	9.0	14.0	21.0	12.0	11.0	13.0	
1	19.0	17.0	15.0	19.0	15.0	9.0	
2	4.0	7.0	5.0	10.0	2.0	3.0	
3	8.0	8.0	13.0	21.0	9.0	10.0	
4	NaN	NaN	NaN	NaN	NaN	NaN	
	2016-02-21	2016-02-22	2016-02-23	2016-02-24	2016-02-25	2016-02-26	\
0	11.0	13.0	16.0	13.0	19.0	21.0	
1	20.0	6.0	11.0	6.0	15.0	20.0	
2	3.0	4.0	2.0	3.0	5.0	3.0	
3	14.0	12.0	9.0	10.0	20.0	15.0	
4	NaN	NaN	NaN	NaN	NaN	NaN	
	2016-02-27	2016-02-28	2016-02-29	2016-03-01	2016-03-02	2016-03-03	\
0	14.0	11.0	35.0	18.0	42.0	15.0	
1	35.0	34.0	21.0	17.0	22.0	26.0	
2	6.0	4.0	5.0	5.0	2.0	1.0	
3	26.0	24.0	19.0	10.0	12.0	8.0	
4	NaN	NaN	NaN	NaN	NaN	NaN	
	2016-03-04	2016-03-05	2016-03-06	2016-03-07	2016-03-08	2016-03-09	\
0	5.0	21.0	56.0	9.0	20.0	17.0	
1	16.0	16.0	28.0	19.0	17.0	15.0	
2	4.0	7.0	2.0	2.0	5.0	1.0	
3	16.0	13.0	8.0	17.0	12.0	34.0	
4	NaN	NaN	NaN	NaN	NaN	NaN	
	2016-03-10	2016-03-11	2016-03-12	2016-03-13	2016-03-14	2016-03-15	\
0	18.0	8.0	9.0	17.0	9.0	10.0	
1	11.0	7.0	15.0	11.0	36.0	16.0	
2	0.0	3.0	3.0	1.0	2.0	4.0	
3	10.0	9.0	9.0	15.0	10.0	12.0	
4	NaN	NaN	NaN	NaN	NaN	NaN	
	2016-03-16	2016-03-17	2016-03-18	2016-03-19	2016-03-20	2016-03-21	\
0	14.0	17.0	6.0	18.0	13.0	11.0	
1	22.0	18.0	46.0	17.0	15.0	17.0	
2	2.0	2.0	3.0	4.0	7.0	1.0	
3	8.0	11.0	9.0	28.0	17.0	11.0	
4	NaN	NaN	NaN	NaN	NaN	NaN	
	2016-03-22	2016-03-23	2016-03-24	2016-03-25	2016-03-26	2016-03-27	\
0	12.0	11.0	8.0	15.0	11.0	20.0	
1	12.0	17.0	14.0	15.0	14.0	15.0	



2	1.0	10.0	9.0	5.0	1.0	6.0	
3	13.0	10.0	10.0	10.0	16.0	12.0	
4	NaN	NaN	NaN	NaN	NaN	NaN	
	2016-03-28	2016-03-29	2016-03-30	2016-03-31	2016-04-01	2016-04-02	\
0	59.0	11.0	18.0	17.0	12.0	14.0	
1	28.0	36.0	23.0	12.0	25.0	18.0	
2	7.0	4.0	6.0	2.0	4.0	155.0	
3	12.0	13.0	25.0	25.0	18.0	18.0	
4	NaN	NaN	NaN	NaN	NaN	NaN	
	2016-04-03	2016-04-04	2016-04-05	2016-04-06	2016-04-07	2016-04-08	\
0	13.0	9.0	490.0	189.0	102.0	38.0	
1	18.0	16.0	20.0	17.0	16.0	13.0	
2	155.0	83.0	48.0	31.0	16.0	6.0	
3	23.0	27.0	39.0	11.0	16.0	9.0	
4	NaN	NaN	NaN	NaN	NaN	NaN	
	2016-04-09	2016-04-10	2016-04-11	2016-04-12	2016-04-13	2016-04-14	\
0	126.0	71.0	21.0	57.0	79.0	17.0	
1	15.0	19.0	14.0	20.0	37.0	16.0	
2	13.0	8.0	8.0	5.0	7.0	3.0	
3	26.0	14.0	15.0	10.0	23.0	17.0	
4	NaN	NaN	NaN	NaN	NaN	NaN	
	2016-04-15	2016-04-16	2016-04-17	2016-04-18	2016-04-19	2016-04-20	\
0	17.0	23.0	16.0	23.0	18.0	22.0	
1	15.0	11.0	42.0	10.0	14.0	61.0	
2	4.0	6.0	7.0	10.0	9.0	7.0	
3	74.0	114.0	8.0	15.0	15.0	15.0	
4	NaN	NaN	38.0	159.0	9.0	4.0	
	2016-04-21	2016-04-22	2016-04-23	2016-04-24	2016-04-25	2016-04-26	\
0	44.0	6.0	31.0	17.0	25.0	40.0	
1	39.0	17.0	17.0	41.0	35.0	16.0	
2	8.0	4.0	6.0	5.0	2.0	7.0	
3	12.0	14.0	14.0	23.0	21.0	11.0	
4	1.0	10.0	9.0	2.0	0.0	5.0	
	2016-04-27	2016-04-28	2016-04-29	2016-04-30	2016-05-01	2016-05-02	\
0	19.0	15.0	15.0	29.0	18.0	16.0	
1	9.0	64.0	22.0	22.0	66.0	33.0	
2	3.0	7.0	6.0	3.0	1.0	6.0	
3	19.0	9.0	10.0	11.0	14.0	9.0	
4	0.0	3.0	55.0	234.0	57.0	5.0	
	2016-05-03	2016-05-04	2016-05-05	2016-05-06	2016-05-07	2016-05-08	\

0	13.0	20.0	22.0	19.0	11.0	50.0	
1	30.0	16.0	18.0	45.0	17.0	88.0	
2	2.0	1.0	3.0	8.0	3.0	5.0	
3	5.0	10.0	20.0	22.0	16.0	9.0	
4	4.0	4.0	0.0	9.0	9.0	6.0	
	2016-05-09	2016-05-10	2016-05-11	2016-05-12	2016-05-13	2016-05-14	\
0	22.0	39.0	23.0	21.0	23.0	22.0	
1	23.0	18.0	12.0	12.0	13.0	13.0	
2	4.0	7.0	5.0	2.0	5.0	0.0	
3	10.0	42.0	22.0	7.0	7.0	54.0	
4	6.0	6.0	10.0	7.0	5.0	4.0	
	2016-05-15	2016-05-16	2016-05-17	2016-05-18	2016-05-19	2016-05-20	\
0	16.0	19.0	35.0	16.0	12.0	15.0	
1	5.0	11.0	13.0	11.0	22.0	10.0	
2	3.0	12.0	4.0	2.0	4.0	6.0	
3	7.0	9.0	13.0	5.0	10.0	12.0	
4	6.0	4.0	2.0	6.0	5.0	3.0	
	2016-05-21	2016-05-22	2016-05-23	2016-05-24	2016-05-25	2016-05-26	\
0	13.0	14.0	10.0	21.0	20.0	19.0	
1	13.0	17.0	10.0	14.0	18.0	9.0	
2	4.0	5.0	9.0	4.0	5.0	7.0	
3	18.0	23.0	23.0	17.0	6.0	14.0	
4	3.0	2.0	5.0	5.0	8.0	8.0	
	2016-05-27	2016-05-28	2016-05-29	2016-05-30	2016-05-31	2016-06-01	\
0	14.0	12.0	15.0	17.0	16.0	21.0	
1	16.0	17.0	6.0	15.0	18.0	10.0	
2	1.0	5.0	1.0	5.0	4.0	5.0	
3	13.0	13.0	9.0	11.0	35.0	8.0	
4	6.0	3.0	7.0	7.0	6.0	6.0	
	2016-06-02	2016-06-03	2016-06-04	2016-06-05	2016-06-06	2016-06-07	\
0	27.0	13.0	11.0	15.0	14.0	18.0	
1	11.0	16.0	10.0	12.0	12.0	13.0	
2	7.0	7.0	5.0	3.0	4.0	1.0	
3	12.0	15.0	10.0	25.0	9.0	8.0	
4	2.0	8.0	3.0	7.0	8.0	3.0	
	2016-06-08	2016-06-09	2016-06-10	2016-06-11	2016-06-12	2016-06-13	\
0	18.0	10.0	11.0	14.0	18.0	14.0	
1	9.0	16.0	19.0	19.0	11.0	15.0	
2	9.0	3.0	4.0	6.0	2.0	2.0	
3	8.0	10.0	14.0	9.0	11.0	303.0	
4	4.0	5.0	2.0	1.0	1.0	1.0	

	2016-06-14	2016-06-15	2016-06-16	2016-06-17	2016-06-18	2016-06-19	\
0	13.0	17.0	15.0	14.0	234.0	8.0	
1	10.0	20.0	25.0	9.0	14.0	10.0	
2	1.0	16.0	6.0	3.0	3.0	6.0	
3	29.0	121.0	69.0	39.0	25.0	27.0	
4	2.0	8.0	6.0	1.0	0.0	4.0	

	2016-06-20	2016-06-21	2016-06-22	2016-06-23	2016-06-24	2016-06-25	\
0	62.0	26.0	22.0	8.0	22.0	15.0	
1	14.0	18.0	25.0	13.0	24.0	14.0	
2	1.0	6.0	1.0	4.0	3.0	5.0	
3	54.0	39.0	24.0	22.0	20.0	14.0	
4	2.0	6.0	2.0	2.0	2.0	1.0	

	2016-06-26	2016-06-27	2016-06-28	2016-06-29	2016-06-30	2016-07-01	\
0	69.0	11.0	18.0	23.0	12.0	20.0	
1	13.0	14.0	24.0	16.0	15.0	13.0	
2	1.0	6.0	5.0	1.0	4.0	5.0	
3	12.0	8.0	17.0	11.0	15.0	19.0	
4	5.0	2.0	2.0	2.0	3.0	10.0	

	2016-07-02	2016-07-03	2016-07-04	2016-07-05	2016-07-06	2016-07-07	\
0	17.0	15.0	16.0	18.0	21.0	15.0	
1	11.0	12.0	28.0	28.0	17.0	27.0	
2	4.0	2.0	4.0	3.0	4.0	2.0	
3	20.0	11.0	36.0	19.0	35.0	22.0	
4	1.0	3.0	4.0	2.0	3.0	4.0	

	2016-07-08	2016-07-09	2016-07-10	2016-07-11	2016-07-12	2016-07-13	\
0	30.0	115.0	56.0	45.0	17.0	18.0	
1	48.0	184.0	64.0	24.0	92.0	31.0	
2	0.0	1.0	3.0	12.0	4.0	7.0	
3	14.0	17.0	15.0	12.0	34.0	20.0	
4	1.0	1.0	9.0	0.0	1.0	6.0	

	2016-07-14	2016-07-15	2016-07-16	2016-07-17	2016-07-18	2016-07-19	\
0	15.0	18.0	14.0	15.0	15.0	24.0	
1	34.0	49.0	21.0	36.0	32.0	16.0	
2	5.0	6.0	6.0	6.0	3.0	3.0	
3	25.0	15.0	18.0	19.0	13.0	17.0	
4	2.0	5.0	2.0	2.0	3.0	2.0	

	2016-07-20	2016-07-21	2016-07-22	2016-07-23	2016-07-24	2016-07-25	\
0	22.0	18.0	30.0	12.0	13.0	18.0	
1	16.0	19.0	22.0	22.0	19.0	18.0	
2	3.0	5.0	5.0	2.0	11.0	6.0	

3	16.0	11.0	22.0	43.0	8.0	13.0
4	11.0	1.0	4.0	4.0	2.0	10.0

	2016-07-26	2016-07-27	2016-07-28	2016-07-29	2016-07-30	2016-07-31	\
0	17.0	31.0	26.0	29.0	12.0	19.0	
1	18.0	17.0	35.0	49.0	19.0	25.0	
2	2.0	2.0	3.0	7.0	5.0	4.0	
3	16.0	8.0	19.0	14.0	9.0	13.0	
4	5.0	3.0	10.0	2.0	5.0	7.0	

	2016-08-01	2016-08-02	2016-08-03	2016-08-04	2016-08-05	2016-08-06	\
0	19.0	57.0	17.0	20.0	49.0	10.0	
1	24.0	39.0	19.0	29.0	30.0	16.0	
2	5.0	3.0	3.0	9.0	7.0	2.0	
3	13.0	16.0	10.0	10.0	11.0	17.0	
4	2.0	5.0	8.0	2.0	5.0	1.0	

	2016-08-07	2016-08-08	2016-08-09	2016-08-10	2016-08-11	2016-08-12	\
0	19.0	26.0	41.0	23.0	30.0	55.0	
1	54.0	15.0	39.0	19.0	17.0	60.0	
2	1.0	5.0	6.0	7.0	13.0	3.0	
3	32.0	21.0	16.0	23.0	15.0	55.0	
4	1.0	2.0	6.0	6.0	2.0	1.0	

	2016-08-13	2016-08-14	2016-08-15	2016-08-16	2016-08-17	2016-08-18	\
0	17.0	24.0	14.0	12.0	49.0	42.0	
1	12.0	77.0	63.0	12.0	9.0	34.0	
2	5.0	6.0	2.0	4.0	1.0	2.0	
3	17.0	17.0	15.0	7.0	13.0	11.0	
4	3.0	2.0	3.0	4.0	3.0	2.0	

	2016-08-19	2016-08-20	2016-08-21	2016-08-22	2016-08-23	2016-08-24	\
0	37.0	13.0	30.0	20.0	33.0	20.0	
1	30.0	13.0	20.0	29.0	10.0	14.0	
2	7.0	2.0	2.0	4.0	4.0	2.0	
3	11.0	8.0	22.0	5.0	7.0	18.0	
4	0.0	13.0	4.0	2.0	4.0	3.0	

	2016-08-25	2016-08-26	2016-08-27	2016-08-28	2016-08-29	2016-08-30	\
0	14.0	40.0	15.0	18.0	26.0	8.0	
1	23.0	15.0	12.0	25.0	22.0	144.0	
2	5.0	3.0	2.0	3.0	5.0	4.0	
3	9.0	13.0	27.0	15.0	19.0	7.0	
4	3.0	1.0	3.0	5.0	2.0	3.0	

	2016-08-31	2016-09-01	2016-09-02	2016-09-03	2016-09-04	2016-09-05	\
0	25.0	21.0	20.0	25.0	19.0	23.0	

1	31.0	31.0	17.0	66.0	78.0	19.0	
2	2.0	5.0	7.0	5.0	2.0	7.0	
3	9.0	14.0	14.0	9.0	16.0	11.0	
4	2.0	4.0	3.0	39.0	4.0	3.0	
	2016-09-06	2016-09-07	2016-09-08	2016-09-09	2016-09-10	2016-09-11	\
0	18.0	19.0	18.0	55.0	16.0	65.0	
1	44.0	43.0	35.0	13.0	13.0	25.0	
2	6.0	11.0	10.0	5.0	19.0	7.0	
3	7.0	14.0	13.0	11.0	9.0	9.0	
4	1.0	5.0	5.0	5.0	5.0	8.0	
	2016-09-12	2016-09-13	2016-09-14	2016-09-15	2016-09-16	2016-09-17	\
0	11.0	11.0	13.0	20.0	21.0	13.0	
1	15.0	37.0	38.0	22.0	28.0	19.0	
2	11.0	4.0	10.0	3.0	4.0	6.0	
3	9.0	11.0	15.0	28.0	10.0	24.0	
4	15.0	13.0	63.0	2.0	2.0	3.0	
	2016-09-18	2016-09-19	2016-09-20	2016-09-21	2016-09-22	2016-09-23	\
0	24.0	20.0	13.0	32.0	16.0	10.0	
1	46.0	24.0	22.0	43.0	58.0	26.0	
2	3.0	4.0	8.0	10.0	3.0	3.0	
3	8.0	20.0	19.0	12.0	31.0	14.0	
4	6.0	10.0	2.0	8.0	4.0	3.0	
	2016-09-24	2016-09-25	2016-09-26	2016-09-27	2016-09-28	2016-09-29	\
0	13.0	44.0	17.0	13.0	72.0	40.0	
1	20.0	27.0	35.0	20.0	31.0	24.0	
2	1.0	10.0	5.0	4.0	4.0	3.0	
3	9.0	40.0	15.0	83.0	60.0	19.0	
4	3.0	6.0	4.0	1.0	5.0	9.0	
	2016-09-30	2016-10-01	2016-10-02	2016-10-03	2016-10-04	2016-10-05	\
0	19.0	14.0	13.0	12.0	14.0	10.0	
1	24.0	94.0	18.0	20.0	18.0	16.0	
2	4.0	1.0	3.0	6.0	6.0	6.0	
3	15.0	15.0	12.0	23.0	17.0	20.0	
4	1.0	6.0	4.0	0.0	4.0	9.0	
	2016-10-06	2016-10-07	2016-10-08	2016-10-09	2016-10-10	2016-10-11	\
0	26.0	13.0	22.0	14.0	23.0	12.0	
1	38.0	54.0	29.0	49.0	25.0	72.0	
2	3.0	5.0	11.0	6.0	3.0	7.0	
3	26.0	11.0	13.0	9.0	44.0	7.0	
4	6.0	8.0	13.0	4.0	7.0	6.0	

	2016-10-12	2016-10-13	2016-10-14	2016-10-15	2016-10-16	2016-10-17	\
0	8.0	50.0	13.0	10.0	16.0	14.0	
1	144.0	36.0	97.0	179.0	29.0	12.0	
2	6.0	0.0	2.0	4.0	4.0	3.0	
3	18.0	4.0	36.0	34.0	10.0	8.0	
4	9.0	3.0	21.0	6.0	13.0	10.0	

	2016-10-18	2016-10-19	2016-10-20	2016-10-21	2016-10-22	2016-10-23	\
0	10.0	24.0	10.0	20.0	10.0	26.0	
1	21.0	42.0	53.0	41.0	19.0	25.0	
2	6.0	4.0	3.0	4.0	1.0	6.0	
3	21.0	7.0	6.0	12.0	15.0	9.0	
4	2.0	3.0	6.0	7.0	10.0	6.0	

	2016-10-24	2016-10-25	2016-10-26	2016-10-27	2016-10-28	2016-10-29	\
0	25.0	16.0	19.0	20.0	12.0	19.0	
1	19.0	15.0	21.0	21.0	27.0	33.0	
2	5.0	5.0	2.0	3.0	3.0	2.0	
3	13.0	21.0	13.0	10.0	21.0	15.0	
4	6.0	4.0	173.0	5.0	10.0	10.0	

	2016-10-30	2016-10-31	2016-11-01	2016-11-02	2016-11-03	2016-11-04	\
0	50.0	16.0	30.0	18.0	25.0	14.0	
1	15.0	24.0	13.0	11.0	14.0	26.0	
2	2.0	6.0	1.0	3.0	3.0	3.0	
3	103.0	22.0	15.0	12.0	11.0	15.0	
4	18.0	20.0	11.0	5.0	6.0	33.0	

	2016-11-05	2016-11-06	2016-11-07	2016-11-08	2016-11-09	2016-11-10	\
0	20.0	8.0	67.0	13.0	41.0	10.0	
1	11.0	21.0	14.0	14.0	54.0	5.0	
2	2.0	10.0	2.0	2.0	2.0	7.0	
3	7.0	12.0	13.0	9.0	8.0	21.0	
4	13.0	10.0	22.0	11.0	8.0	4.0	

	2016-11-11	2016-11-12	2016-11-13	2016-11-14	2016-11-15	2016-11-16	\
0	21.0	13.0	8.0	15.0	14.0	12.0	
1	10.0	12.0	11.0	14.0	28.0	23.0	
2	3.0	6.0	4.0	2.0	4.0	6.0	
3	16.0	38.0	13.0	14.0	17.0	26.0	
4	10.0	13.0	11.0	8.0	6.0	10.0	

	2016-11-17	2016-11-18	2016-11-19	2016-11-20	2016-11-21	2016-11-22	\
0	6.0	11.0	10.0	42.0	21.0	24.0	
1	20.0	9.0	12.0	11.0	14.0	14.0	
2	5.0	4.0	4.0	3.0	3.0	9.0	
3	14.0	10.0	9.0	23.0	15.0	7.0	

4	14.0	6.0	9.0	6.0	16.0	14.0	
	2016-11-23	2016-11-24	2016-11-25	2016-11-26	2016-11-27	2016-11-28	\
0	14.0	11.0	204.0	14.0	45.0	33.0	
1	15.0	15.0	11.0	20.0	13.0	19.0	
2	3.0	5.0	4.0	0.0	1.0	4.0	
3	10.0	7.0	10.0	14.0	17.0	11.0	
4	13.0	15.0	14.0	16.0	9.0	178.0	
	2016-11-29	2016-11-30	2016-12-01	2016-12-02	2016-12-03	2016-12-04	\
0	28.0	18.0	14.0	47.0	15.0	14.0	
1	621.0	57.0	17.0	23.0	19.0	21.0	
2	5.0	8.0	8.0	1.0	1.0	2.0	
3	9.0	11.0	5.0	10.0	8.0	17.0	
4	64.0	12.0	10.0	11.0	6.0	8.0	
	2016-12-05	2016-12-06	2016-12-07	2016-12-08	2016-12-09	2016-12-10	\
0	18.0	20.0	14.0	16.0	14.0	20.0	
1	47.0	28.0	22.0	22.0	65.0	27.0	
2	5.0	3.0	3.0	3.0	7.0	3.0	
3	13.0	23.0	40.0	16.0	17.0	41.0	
4	7.0	9.0	8.0	5.0	11.0	8.0	
	2016-12-11	2016-12-12	2016-12-13	2016-12-14	2016-12-15	2016-12-16	\
0	60.0	22.0	15.0	17.0	19.0	18.0	
1	17.0	17.0	13.0	9.0	18.0	22.0	
2	9.0	8.0	3.0	210.0	5.0	4.0	
3	17.0	8.0	9.0	18.0	12.0	12.0	
4	4.0	15.0	5.0	8.0	8.0	6.0	
	2016-12-17	2016-12-18	2016-12-19	2016-12-20	2016-12-21	2016-12-22	\
0	21.0	21.0	47.0	65.0	17.0	32.0	
1	17.0	15.0	22.0	23.0	19.0	17.0	
2	6.0	2.0	2.0	4.0	3.0	3.0	
3	18.0	13.0	18.0	23.0	10.0	32.0	
4	7.0	15.0	4.0	11.0	7.0	48.0	
	2016-12-23	2016-12-24	2016-12-25	2016-12-26	2016-12-27	2016-12-28	\
0	63.0	15.0	26.0	14.0	20.0	22.0	
1	42.0	28.0	15.0	9.0	30.0	52.0	
2	1.0	1.0	7.0	4.0	4.0	6.0	
3	10.0	26.0	27.0	16.0	11.0	17.0	
4	9.0	25.0	13.0	3.0	11.0	27.0	
	2016-12-29	2016-12-30	2016-12-31				
0	19.0	18.0	20.0				
1	45.0	26.0	20.0				

2	3.0	4.0	17.0
3	19.0	10.0	11.0
4	13.0	36.0	10.0

```
[9]: data.shape
```

```
[9]: (145063, 551)
```

We have 145063 different pages and visits for 550 days

### 0.0.1 Missing values check

```
[10]: #Checking the count of nulls in each column.
data.isnull().sum()
```

```
[10]: Page          0
2015-07-01    20740
2015-07-02    20816
2015-07-03    20544
2015-07-04    20654
...
2016-12-27     3701
2016-12-28     3822
2016-12-29     3826
2016-12-30     3635
2016-12-31     3465
Length: 551, dtype: int64
```

Clearly we have lots of nulls in each column

```
[11]: data.loc[data['Page']=='52_Hz_I_Love_You_zh.wikipedia.org_all-access_spider']
d1 = datetime.strptime('2015-07-01', "%Y-%m-%d")
print('Start date:', d1)

d2 = datetime.strptime('2016-04-16', "%Y-%m-%d")
print('End time:', d2)

# get difference
delta = d2 - d1

# time difference in seconds
print(f"Days difference is {delta} seconds")
```

Start date: 2015-07-01 00:00:00

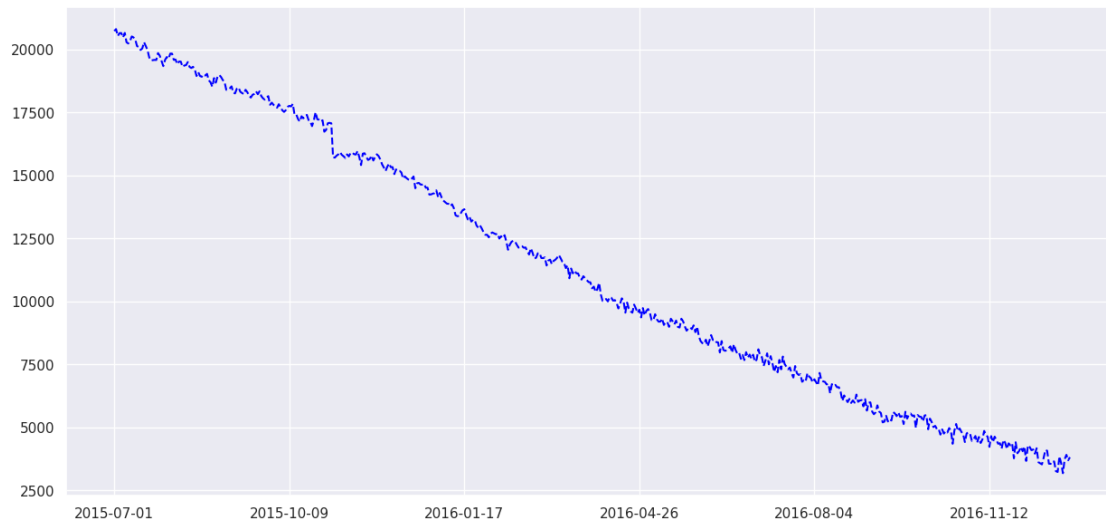
End time: 2016-04-16 00:00:00

Days difference is 290 days, 0:00:00 seconds



- We also have pages where the data hasn't started before a certain date. So We have to remove those records once we have reshaped the data.

```
[12]: data.iloc[:, 1:-3].isnull().sum().plot(color='blue', linestyle='dashed')
plt.show()
```



- The chart above illustrates a decreasing trend in NaN/Null values over time. Recent dates exhibit fewer Null Values compared to earlier dates.
- This phenomenon is plausible because pages created or hosted at later dates naturally lack data for previous dates (dates preceding their creation/hosting).
- To address this, we plan to eliminate rows containing more than 300 Null Values and substitute the remaining Null Values with 0.

```
[13]: data.dropna(thresh = 300, inplace = True)
print(f'Shape of Data : {data.shape}')
```

Shape of Data : (133617, 551)

```
[14]: data.fillna(0, inplace = True)
```

## 0.0.2 Feature Extraction

```
[15]: #Function to Extract Language from Page using Regex
def get_language(name):
    if len(re.findall(r'_{2}).wikipedia.org_', name)) == 1 :
        return re.findall(r'_{2}).wikipedia.org_', name)[0]
    else: return 'Unknown_language'

data['language'] = data['Page'].apply(get_language)
```

```
language_dict ={'de': 'German',
                'en': 'English',
                'es': 'Spanish',
                'fr': 'French',
                'ja': 'Japenese' ,
                'ru': 'Russian',
                'zh': 'Chinese',
                'Unknown_language': 'Unknown_language'}
```

```
data['language'] = data['language'].map(language_dict)
```

```
[16]: def get_access_type(name):
        if len(re.findall(r'all-access|mobile-web|desktop', name)) == 1 :
            return re.findall(r'all-access|mobile-web|desktop', name)[0]
        else: return 'No Access_type'
```

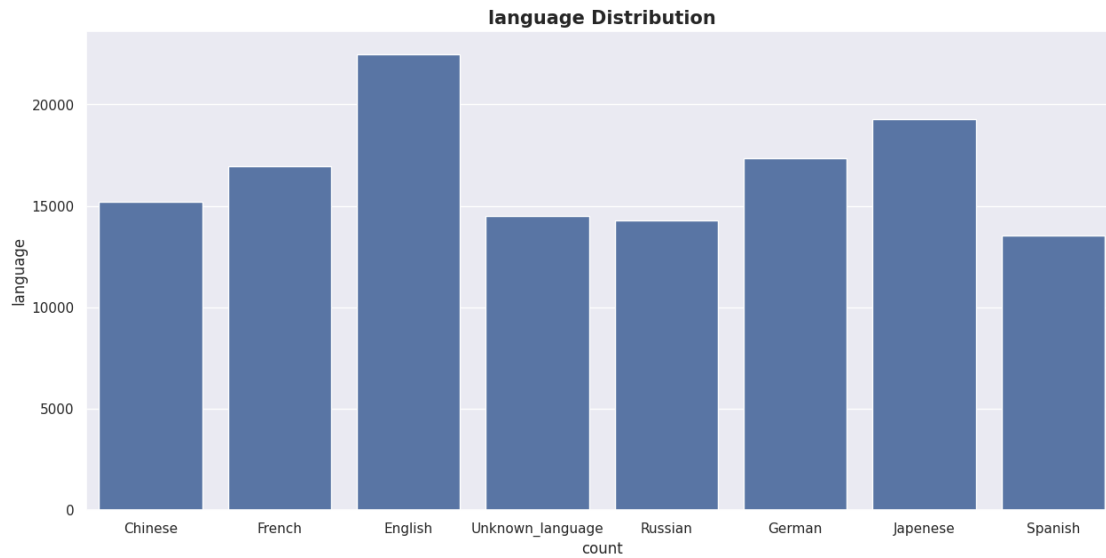
```
data['access_type'] = data['Page'].apply(get_access_type)
```

```
[17]: def get_access_origin(name):
        if len(re.findall(r'[ai].org_(.*)_(.*)$', name)) == 1 :
            return re.findall(r'[ai].org_(.*)_(.*)$', name)[0][1]
        else: return 'No Access_origin'
```

```
data['access_origin'] = data['Page'].apply(get_access_origin)
```

### Plotting count of langauges

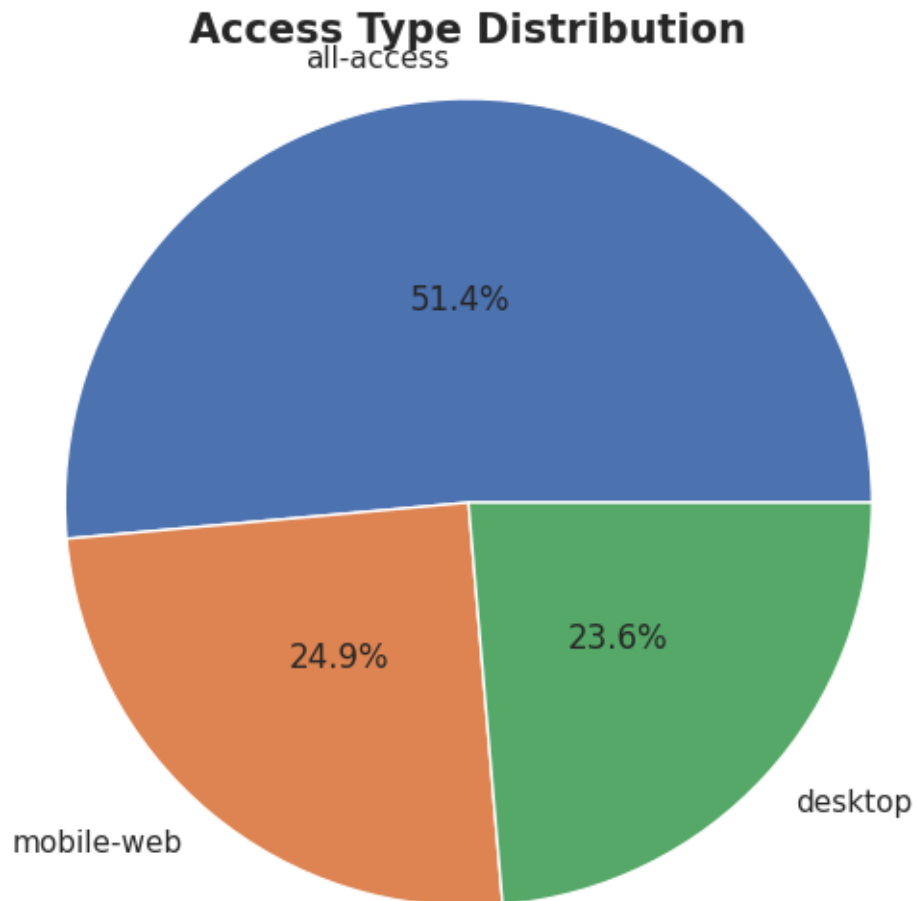
```
[18]: # plt.figure(figsize=(15, 7))
sns.countplot(x='language' , data=data)
plt.title('language Distribution')
plt.xlabel('count')
plt.ylabel('language')
plt.title('language Distribution', fontsize = 15, fontweight = 'bold')
plt.show()
```



### Plotting access type

```
[19]: x = data['access_type'].value_counts().values
      y = data['access_type'].value_counts().index

      plt.figure(figsize=(7, 6))
      plt.pie(x, labels = y, center=(0, 0), radius=1.5, autopct='%1.1f%%',
              pctdistance=0.5)
      plt.title('Access Type Distribution', fontsize = 15, fontweight = 'bold')
      plt.axis('equal')
      plt.show()
```

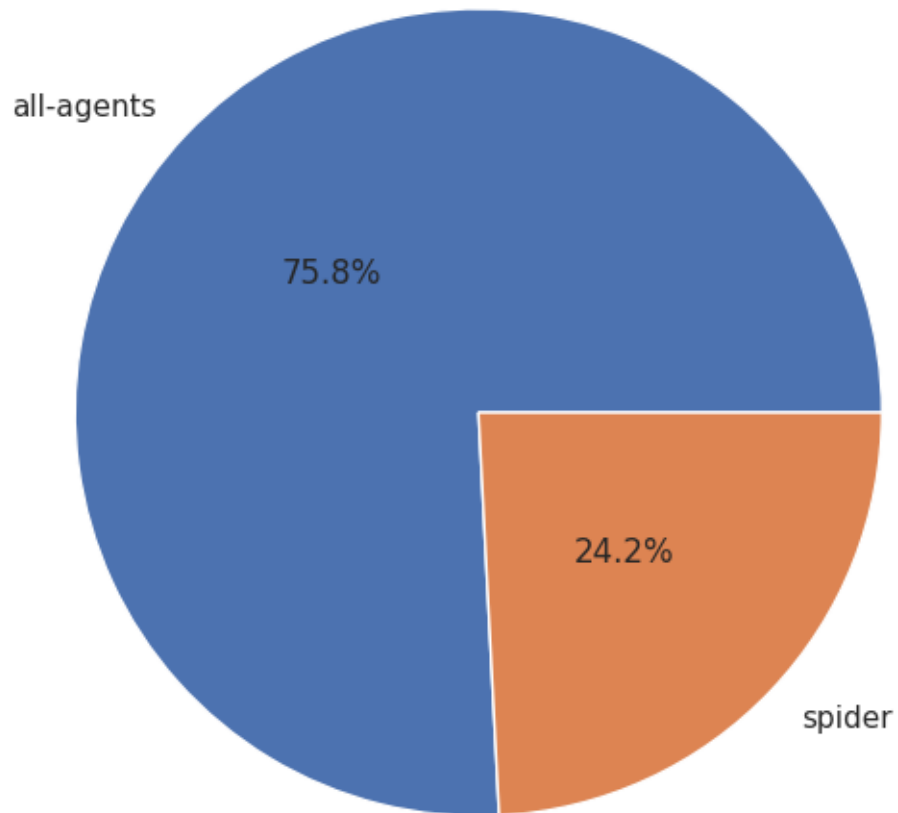


#### Plotting access origin spread

```
[20]: var = 'access_origin'
x = data[var].value_counts().values
y = data[var].value_counts().index

plt.figure(figsize=(7, 6))
plt.pie(x, labels = y, center=(0, 0), radius=1.5, autopct='%1.1f%%',
        pctdistance=0.5)
plt.title(f'{var} Distribution', fontsize = 15, fontweight = 'bold')
plt.axis('equal')
plt.show()
```

## access\_origin Distribution



### 0.0.3 Reshaping data

```
[21]: reshaped = data.melt(id_vars = 
    ↪ ['Page', 'language', 'access_type', 'access_origin'])
```

```
reshaped.sort_values(['Page', 'variable'], inplace=True)
```

```
[22]: reshaped.head()
```

```
[22]:
```

	Page	language	access_type	\
0	2NE1_zh.wikipedia.org_all-access_spider	Chinese	all-access	
1	2PM_zh.wikipedia.org_all-access_spider	Chinese	all-access	
2	3C_zh.wikipedia.org_all-access_spider	Chinese	all-access	
3	4minute_zh.wikipedia.org_all-access_spider	Chinese	all-access	
4	5566_zh.wikipedia.org_all-access_spider	Chinese	all-access	

access_origin	variable	value
---------------	----------	-------

```

0      spider  2015-07-01  18.0
1      spider  2015-07-01  11.0
2      spider  2015-07-01   1.0
3      spider  2015-07-01  35.0
4      spider  2015-07-01  12.0

```

```
[23]: reshaped.columns = ['Page', 'language', 'access_type', 'access_origin', 'Date', 'Visits']
```

```
[24]: reshaped.Date = pd.to_datetime(reshaped.Date, format='%Y-%m-%d')
```

```
[25]: lang_data = reshaped.groupby(['language', 'Date'], as_index=False)['Visits'].sum()
```

```
[26]: lang_data.shape
```

```
[26]: (4400, 3)
```

```
[27]: lang_data.head()
```

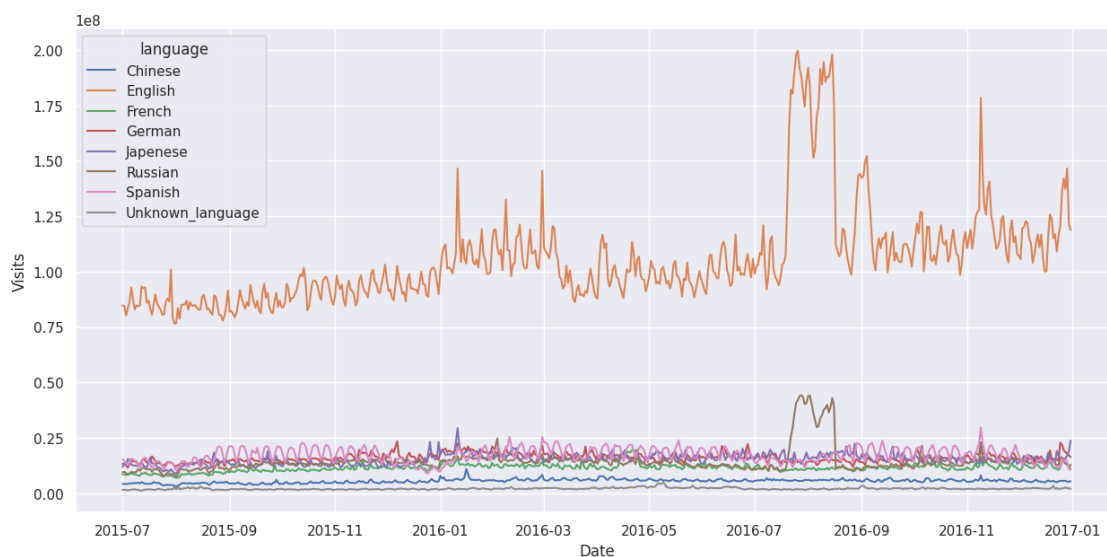
```

[27]:  language      Date      Visits
0  Chinese  2015-07-01  4144975.0
1  Chinese  2015-07-02  4151185.0
2  Chinese  2015-07-03  4123659.0
3  Chinese  2015-07-04  4163439.0
4  Chinese  2015-07-05  4441273.0

```

```
[28]: sns.lineplot(data=lang_data, y='Visits', x='Date', hue='language')
```

```
[28]: <Axes: xlabel='Date', ylabel='Visits'>
```



```
[29]: lang_data.head()
```

```
[29]:   language      Date      Visits
0  Chinese 2015-07-01  4144975.0
1  Chinese 2015-07-02  4151185.0
2  Chinese 2015-07-03  4123659.0
3  Chinese 2015-07-04  4163439.0
4  Chinese 2015-07-05  4441273.0
```

#### 0.0.4 Checking Stationarity using ADF (Augmented Dickey Fuller) Test

```
[30]: #define function for ADF test
def adf_test(timeseries):
    print ('Results of Dickey-Fuller Test:')
    dfctest = adfuller(timeseries, autolag='AIC')
    df_output = pd.Series(dfctest[0:4], index=['Test Statistic','p-value','#Lags_
Used','Number of Observations Used'])
    for key, value in dfctest[4].items():
        df_output['Critical Value (%)' %key] = round(value,2)
    print (df_output)
```

```
[31]: adf_test(lang_data[lang_data['language'] == 'English']['Visits'])
```

```
Results of Dickey-Fuller Test:
Test Statistic          -2.373563
p-value                  0.149337
#Lags Used              14.000000
Number of Observations Used  535.000000
Critical Value (1%)      -3.440000
Critical Value (5%)      -2.870000
Critical Value (10%)     -2.570000
dtype: float64
```

The test statistic > critical value / p\_value > 5%. This implies that the series is not stationary.

#### 0.0.5 Decomposing Time Series

##### 0.1 Time Series Decomposition

Time series decomposition is a statistical technique used to break down a time series into its constituent components in order to understand its underlying structure, trends, seasonality, and irregular fluctuations. The decomposition typically involves separating the time series data into three main components:

1. **Trend ((T<sub>t</sub>)):** The long-term movement or pattern in the data, representing the overall direction in which the time series is moving.
2. **Seasonality ((S<sub>t</sub>)):** The repeating patterns or fluctuations that occur at regular intervals within the time series data.
3. **Residuals ((R<sub>t</sub>)):** The remaining variation in the data after removing the trend and seasonality components.

The time series ( $y_t$ ) can be decomposed into its components as follows:

- **Additive Decomposition:**  $[y_t = T_t + S_t + R_t]$
- **Multiplicative Decomposition:**  $[y_t = T_t \times S_t \times R_t]$

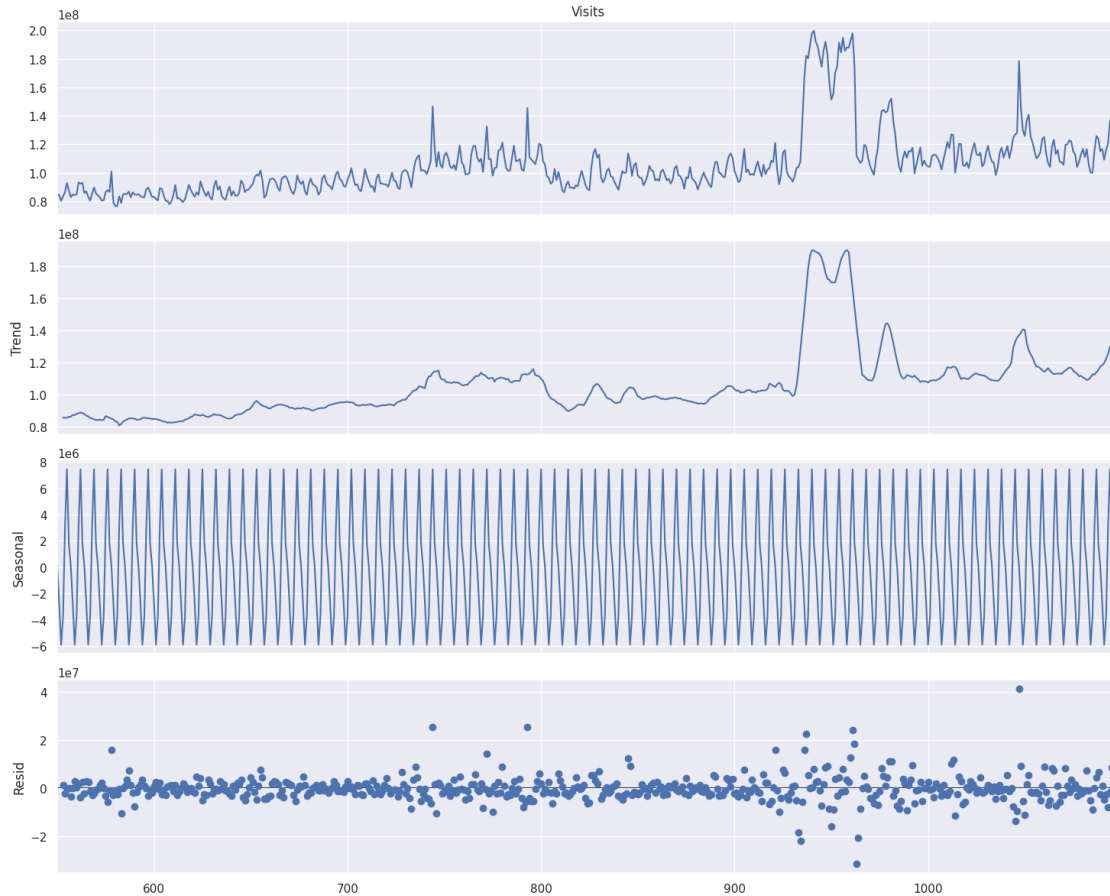
Various techniques such as moving averages, exponential smoothing, or mathematical models can be used to estimate the trend and seasonal components, leaving the residual component as the leftover variation in the data.

```
[32]: ts_english = lang_data[lang_data['language'] == 'English']['Visits']
```

```
[33]: decomposition = seasonal_decompose(ts_english, model='additive', period=7)

fig = decomposition.plot()
fig.set_size_inches((15, 12))
fig.tight_layout()
plt.show()
```





```
[34]: residual = pd.DataFrame(decomposition.resid.fillna(0).values)
      adf_test(residual)
```

Results of Dickey-Fuller Test:

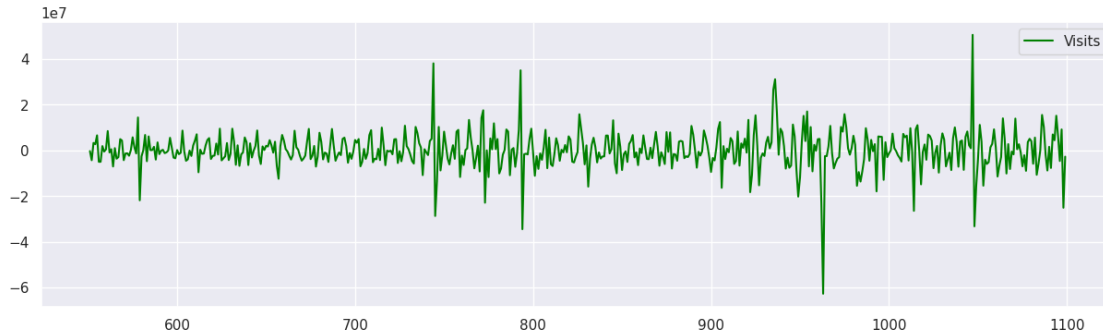
Test Statistic	-1.152195e+01
p-value	4.020092e-21
#Lags Used	1.700000e+01
Number of Observations Used	5.320000e+02
Critical Value (1%)	-3.440000e+00
Critical Value (5%)	-2.870000e+00
Critical Value (10%)	-2.570000e+00
dtype:	float64

Residuals from time-series decomposition are now Stationary

### 0.1.1 Estimating (p,q,d) & Interpreting ACF and PACF plots

```
[35]: ts_diff = pd.DataFrame(ts_english).diff(1)
      ts_diff.dropna(inplace = True)
```

```
[36]: ts_diff.plot(color = 'green', figsize=(15, 4))
      plt.show()
```



```
[37]: adf_test(ts_diff)
```

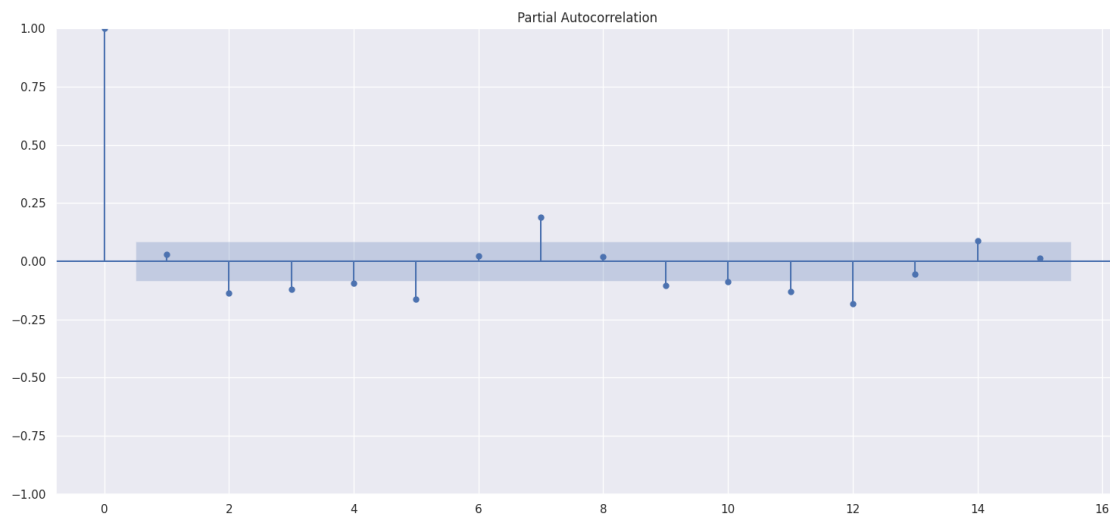
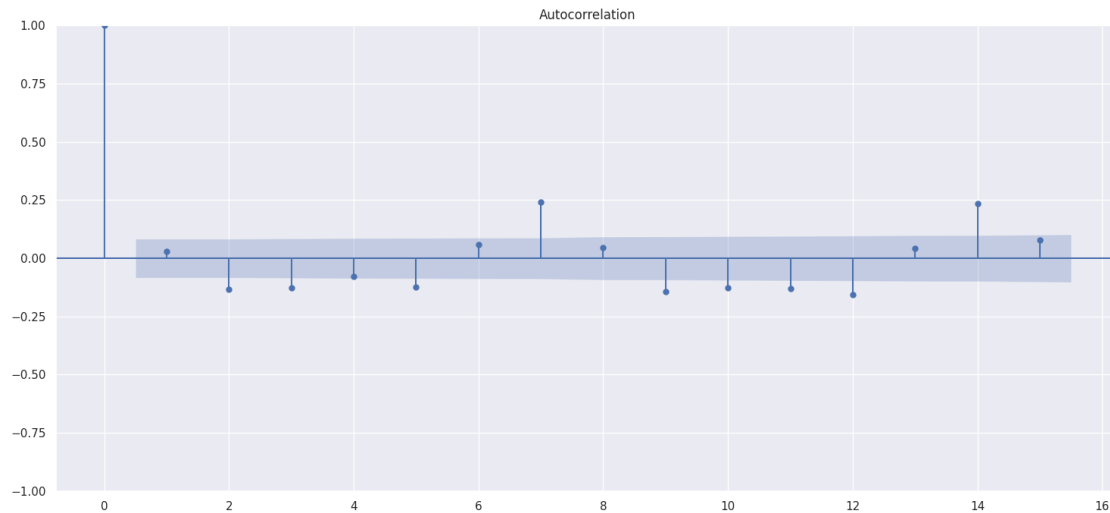
Results of Dickey-Fuller Test:

Test Statistic	-8.273590e+00
p-value	4.721272e-13
#Lags Used	1.300000e+01
Number of Observations Used	5.350000e+02
Critical Value (1%)	-3.440000e+00
Critical Value (5%)	-2.870000e+00
Critical Value (10%)	-2.570000e+00

dtype: float64

We are getting a stationary time series after a differentiation of 1. d can therefore be 1.

```
[38]: acf = plot_acf(ts_diff, lags= 15)
      acf.tight_layout()
      pacf = plot_pacf(ts_diff, lags= 15)
      pacf.tight_layout()
```



## ACF

- If the ACF shows a sharp cutoff after lag 'k', it suggests that an AR(k) model may be appropriate.
- If the ACF decreases gradually, it suggests a non-stationary series, and differencing (d) may be needed.
- If the ACF has a sinusoidal pattern or fluctuates around zero, it suggests a seasonal component.

The ACF shows a sharp cutoff after lag 0, it suggests that an AR(0) model may be appropriate.

## PACF

- If the PACF has a sharp cutoff after lag 'k', it suggests an MA(k) model may be appropriate.
- If the PACF gradually decreases, it suggests an AR component.
- If there are significant spikes at seasonal lags, it suggests a seasonal AR or MA component.

The PACF has a sharp cutoff after lag 0, it suggests an MA(0) model may be appropriate.

```
[39]: ts_english = lang_data[lang_data.language == 'English'][['Date', 'Visits']]
      ts_english.set_index('Date', drop=True, inplace=True)
```

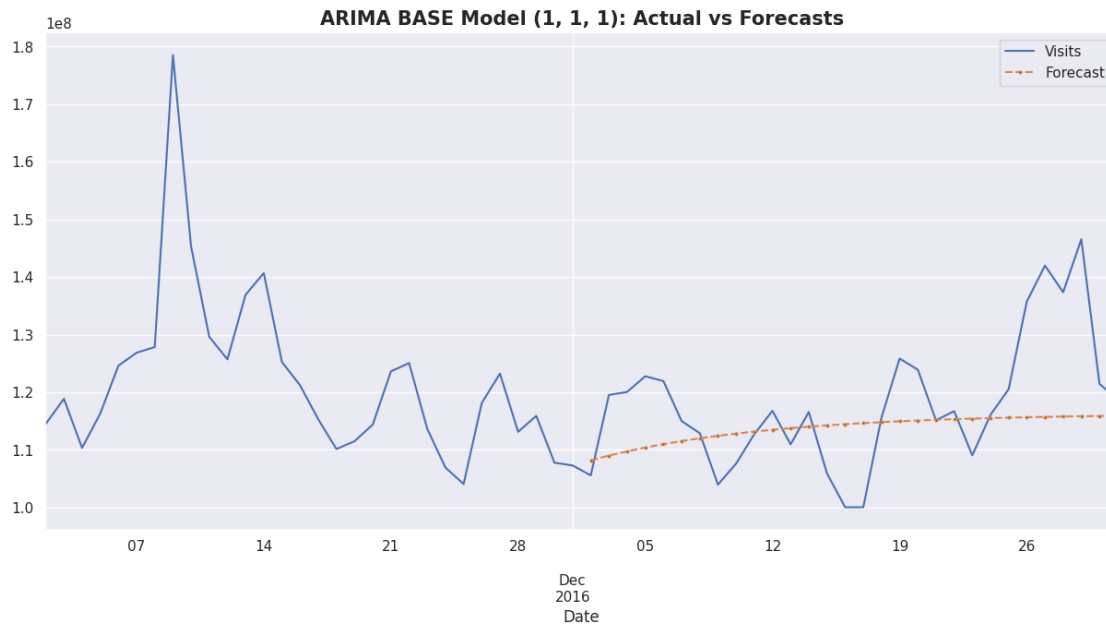
```
[40]: def arima_model(n, order, time_series):
      model = ARIMA(time_series[:-n], order=order)
      model_fit = model.fit()
      forecast = model_fit.forecast(steps=n, alpha=0.05)
      time_series.index = pd.to_datetime(time_series.index)
      forecast.index = pd.to_datetime(forecast.index)
      time_series[-60:].plot(label='Actual')
      forecast.plot(label='Forecast', linestyle='dashed', marker='o',
      ↪markerfacecolor='green', markersize=2)
      plt.legend(loc="upper right")
      plt.title(f'ARIMA BASE Model {order}: Actual vs Forecasts', fontsize=15,
      ↪fontweight='bold')
      plt.show()

      actuals = time_series.values[-n:]
      errors = time_series.values[-n:] - forecast.values

      mape = np.mean(np.abs(errors) / np.abs(actuals))
      rmse = np.sqrt(np.mean(errors**2))

      # Print MAPE & RMSE
      print('-' * 80)
      print(f'MAPE of Model: {np.round(mape, 5)}')
      print('-' * 80)
      print(f'RMSE of Model: {np.round(rmse, 3)}')
      print('-' * 80)
```

```
[41]: arima_model(30, (1,1,1), ts_english)
```



-----  
MAPE of Model: 0.07229  
-----

RMSE of Model: 12071774.914  
-----

```
[42]: def sarimax_model(time_series, n, p=0, d=0, q=0, P=0, D=0, Q=0, s=0, exog = []):

    #Creating SARIMAX Model with order(p,d,q) & seasonal_order=(P, D, Q, s)
    model = SARIMAX(time_series[:-n],
                    order=(p, d, q),
                    seasonal_order=(P, D, Q, s),
                    exog=exog[:-n],
                    initialization='approximate_diffuse')
    model_fit = model.fit()

    # Forecasting last n-values
    model_forecast = model_fit.forecast(n, dynamic=True, exog=pd.
    ↪DataFrame(exog[-n:]))

    # Plotting Actual & Forecasted values
    plt.figure(figsize=(20, 8))
    time_series[-60:].plot(label='Actual')
    model_forecast[-60:].plot(label='Forecast', color='red',
                               linestyle='dashed', marker='o',
    ↪markerfacecolor='green', markersize=5)
```

```

plt.legend(loc="upper right")
plt.title(f'SARIMAX Model ({p},{d},{q}) ({P},{D},{Q},{s}) : Actual vs_
Forecasts', fontsize=15, fontweight='bold')
plt.show()

# Calculating MAPE & RMSE
actuals = time_series.values[-n:]
errors = time_series.values[-n:] - model_forecast.values

mape = np.mean(np.abs(errors) / np.abs(actuals))
rmse = np.sqrt(np.mean(errors ** 2))

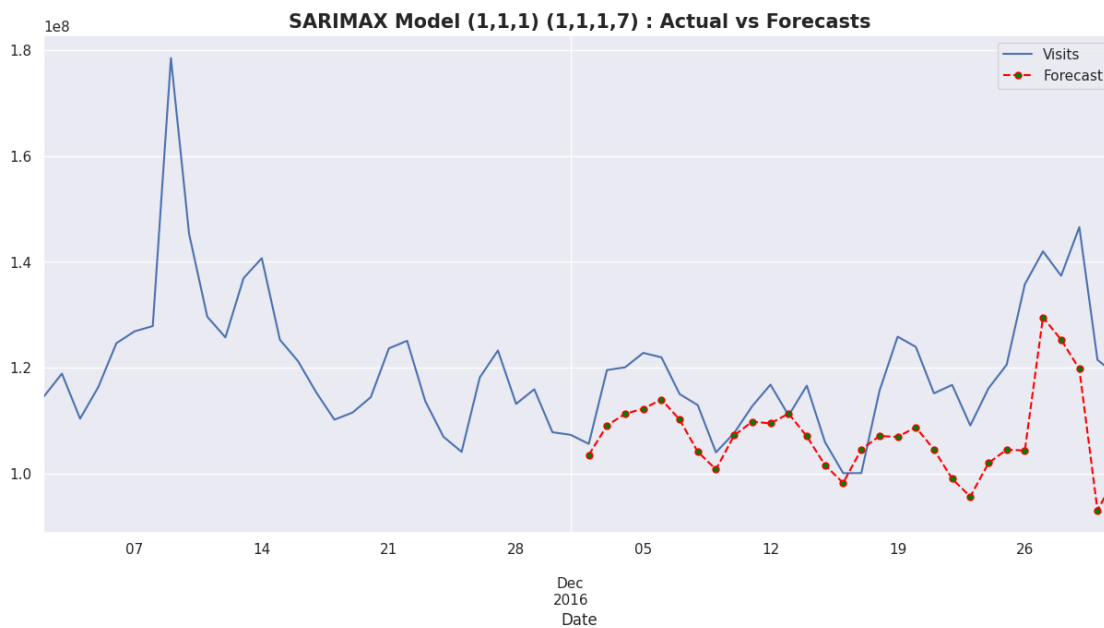
# Printing metrics
print('-' * 80)
print(f'MAPE of Model : {np.round(mape, 5)}')
print('-' * 80)
print(f'RMSE of Model : {np.round(rmse, 3)}')
print('-' * 80)

```

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

```
[44]: time_series = ts_english
test_size= 0.1
p,d,q, P,D,Q,s = 1,1,1,1,1,1,7
n = 30
sarimax_model(time_series, n, p = p, d=d,q=q, P=P, D=D, Q=Q, s=s, exog = exog)
```

<Figure size 2000x800 with 0 Axes>



---

MAPE of Model : 0.11208

---

RMSE of Model : 17326667.279

---

Sarimax algorithm is giving us less than 12 % MAPE.

### 0.1.2 Grid Search

```
[45]: def sarimax_grid_search(time_series, n, param, d_param, s_param, exog=[]):  
    # Creating df for storing results summary  
    param_df = pd.DataFrame(columns=['serial', 'pdq', 'PDQs', 'mape', 'rmse'])  
  
    # Generate all parameter combinations  
    param_combinations = product(param, d_param, param, param, d_param, param,   
    ↪ s_param)  
  
    # Counter for keeping track of iterations  
    counter = 0  
  
    for p, d, q, P, D, Q, s in param_combinations:  
        model = SARIMAX(time_series[:-n],  
                        order=(p, d, q),  
                        seasonal_order=(P, D, Q, s),  
                        exog=exog[:-n],  
                        initialization='approximate_diffuse')  
        model_fit = model.fit()  
  
        model_forecast = model_fit.forecast(n, dynamic=True, exog=pd.  
    ↪ DataFrame(exog[-n:]))  
  
        actuals = time_series.values[-n:]  
        errors = time_series.values[-n:] - model_forecast.values  
  
        mape = np.mean(np.abs(errors) / np.abs(actuals))  
        rmse = np.sqrt(np.mean(errors**2))  
        mape = np.round(mape, 5)  
        rmse = np.round(rmse, 3)  
  
        counter += 1  
        list_row = [counter, (p, d, q), (P, D, Q, s), mape, rmse]  
        param_df.loc[len(param_df)] = list_row  
  
    # Print statement to check progress of Loop
```

```

        print(f'Possible Combination: {counter} out of {len(param)**4 *
↳len(s_param) * len(d_param)**2} calculated')

    return param_df

```

```

[46]: time_series = ts_english
      n = 30
      param = [0,1,2]
      d_param = [0,1]
      s_param = [7]

      english_params = sarimax_grid_search(time_series, n, param,
↳d_param,s_param,exog)

```

```

Possible Combination: 1 out of 324 calculated
Possible Combination: 2 out of 324 calculated
Possible Combination: 3 out of 324 calculated
Possible Combination: 4 out of 324 calculated
Possible Combination: 5 out of 324 calculated
Possible Combination: 6 out of 324 calculated
Possible Combination: 7 out of 324 calculated
Possible Combination: 8 out of 324 calculated
Possible Combination: 9 out of 324 calculated
Possible Combination: 10 out of 324 calculated
Possible Combination: 11 out of 324 calculated
Possible Combination: 12 out of 324 calculated
Possible Combination: 13 out of 324 calculated
Possible Combination: 14 out of 324 calculated
Possible Combination: 15 out of 324 calculated
Possible Combination: 16 out of 324 calculated
Possible Combination: 17 out of 324 calculated
Possible Combination: 18 out of 324 calculated
Possible Combination: 19 out of 324 calculated
Possible Combination: 20 out of 324 calculated
Possible Combination: 21 out of 324 calculated
Possible Combination: 22 out of 324 calculated
Possible Combination: 23 out of 324 calculated
Possible Combination: 24 out of 324 calculated
Possible Combination: 25 out of 324 calculated
Possible Combination: 26 out of 324 calculated
Possible Combination: 27 out of 324 calculated
Possible Combination: 28 out of 324 calculated
Possible Combination: 29 out of 324 calculated
Possible Combination: 30 out of 324 calculated
Possible Combination: 31 out of 324 calculated
Possible Combination: 32 out of 324 calculated
Possible Combination: 33 out of 324 calculated

```



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

Possible Combination: 322 out of 324 calculated  
 Possible Combination: 323 out of 324 calculated  
 Possible Combination: 324 out of 324 calculated

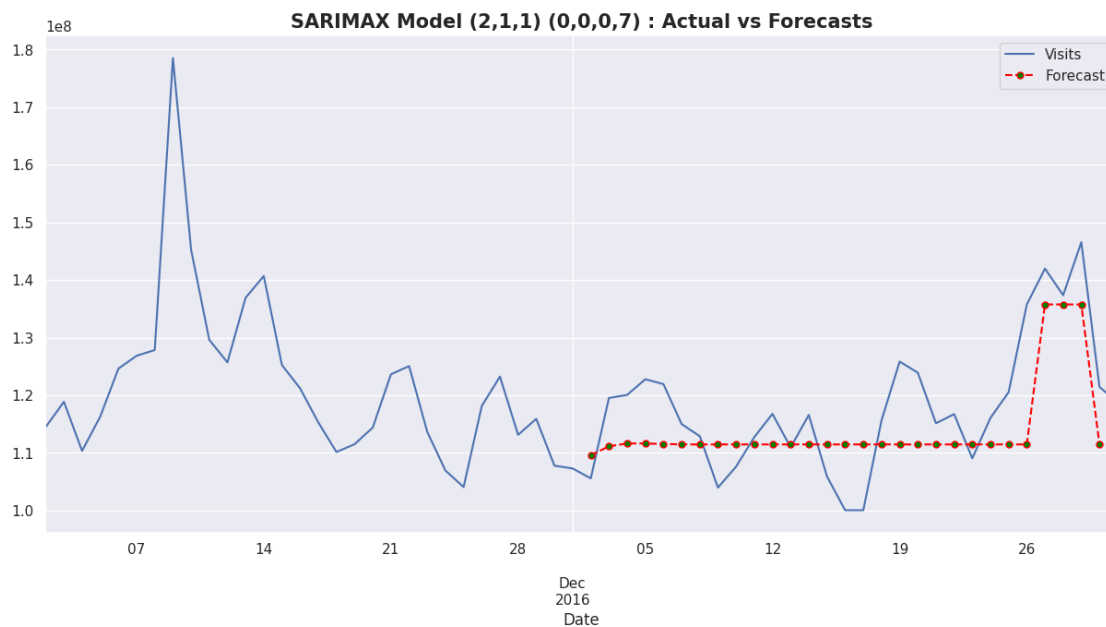
```
[47]: english_params.sort_values(['mape', 'rmse']).head()
```

```
[47]:
```

	serial	pdq	PDQs	mape	rmse
288	289	(2, 1, 1)	(0, 0, 0, 7)	0.08737	1.390974e+07
289	290	(2, 1, 1)	(0, 0, 1, 7)	0.08903	1.411103e+07
290	291	(2, 1, 1)	(0, 0, 2, 7)	0.08988	1.421080e+07
294	295	(2, 1, 1)	(1, 0, 0, 7)	0.09013	1.423713e+07
300	301	(2, 1, 1)	(2, 0, 0, 7)	0.09273	1.456823e+07

```
[48]: time_series = ts_english
p,d,q, P,D,Q,s = 2,1,1, 0,0,0,7
n = 30
sarimax_model(time_series, n, p=p, d=d, q=q, P=P, D=D, Q=Q, s=s, exog = exog)
```

<Figure size 2000x800 with 0 Axes>

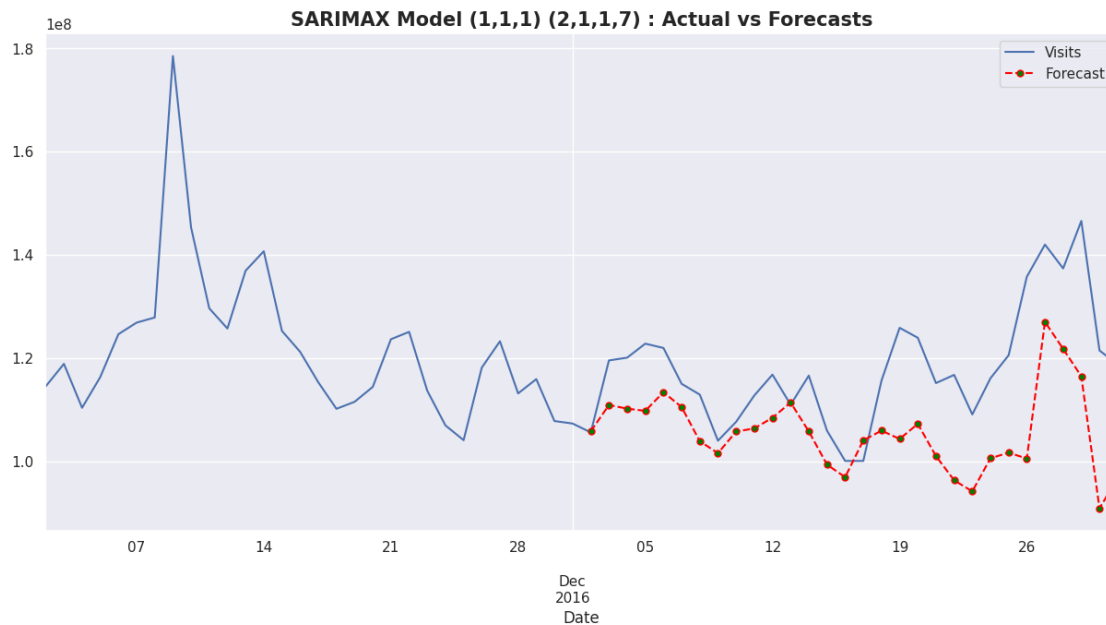


-----  
 MAPE of Model : 0.08737  
 -----

RMSE of Model : 13909735.545  
 -----

```
[49]: time_series = ts_english
p,d,q, P,D,Q,s = 1,1,1, 2,1,1,7
n = 30
sarimax_model(time_series, n, p=p, d=d, q=q, P=P, D=D, Q=Q, s=s, exog = exog)
```

<Figure size 2000x800 with 0 Axes>



MAPE of Model : 0.11871

RMSE of Model : 18268474.479

```
[50]: def pipeline_sarimax_grid_search_without_exog(languages, data_language, n,
param, d_param, s_param):

    best_param_df = pd.DataFrame(columns=['language', 'p', 'd', 'q', 'P', 'D',
'Q', 's', 'mape'])

    for lang in languages:
        print(f'-----')
        print(f'                Finding best parameters for {lang}                ')
        print(f'-----')

        time_series = data_language[data_language['language'] == lang][['Date',
'Visits']]
        time_series.set_index('Date', drop=True, inplace=True)
```



```

best_mape = 100

counter = 0
param_combinations = product(param, d_param, param, param, d_param,
↪param, s_param)

for p, d, q, P, D, Q, s in param_combinations:
    model = SARIMAX(time_series[:-n],
                    order=(p, d, q),
                    seasonal_order=(P, D, Q, s),
                    initialization='approximate_diffuse')
    model_fit = model.fit()
    model_forecast = model_fit.forecast(n, dynamic=True)

    actuals = time_series.values[-n:]
    errors = time_series.values[-n:] - model_forecast.values
    mape = np.mean(np.abs(errors) / np.abs(actuals))

    counter += 1
    if mape < best_mape:
        best_mape = mape
        best_p, best_d, best_q = p, d, q
        best_P, best_D, best_Q = P, D, Q
        best_s = s

    print(f'Possible Combination: {counter} out of
↪{(len(param)**4)*len(s_param)*(len(d_param)**2)} calculated')

    best_mape = np.round(best_mape, 5)
    print(f'-----')
    print(f'Minimum MAPE for {lang} = {best_mape}')
    print(f'Corresponding Best Parameters are {best_p, best_d, best_q,
↪best_P, best_D, best_Q, best_s}')
    print(f'-----')

    best_param_row = [lang, best_p, best_d, best_q, best_P, best_D, best_Q,
↪best_s, best_mape]
    best_param_df.loc[len(best_param_df)] = best_param_row

return best_param_df

```

```

[51]: languages = ['Chinese', 'French', 'German', 'Japenese', 'Russian', 'Spanish']
n = 30
param = [0,1,2]
d_param = [0,1]
s_param = [7]

```

```
best_param_df = pipeline_sarimax_grid_search_without_exog(languages, lang_data,
↳n, param, d_param, s_param)
```

---

#### Finding best parameters for Chinese

---

Possible Combination: 1 out of 324 calculated  
Possible Combination: 2 out of 324 calculated  
Possible Combination: 3 out of 324 calculated  
Possible Combination: 4 out of 324 calculated  
Possible Combination: 5 out of 324 calculated  
Possible Combination: 6 out of 324 calculated  
Possible Combination: 7 out of 324 calculated  
Possible Combination: 8 out of 324 calculated  
Possible Combination: 9 out of 324 calculated  
Possible Combination: 10 out of 324 calculated  
Possible Combination: 11 out of 324 calculated  
Possible Combination: 12 out of 324 calculated  
Possible Combination: 13 out of 324 calculated  
Possible Combination: 14 out of 324 calculated  
Possible Combination: 15 out of 324 calculated  
Possible Combination: 16 out of 324 calculated  
Possible Combination: 17 out of 324 calculated  
Possible Combination: 18 out of 324 calculated  
Possible Combination: 19 out of 324 calculated  
Possible Combination: 20 out of 324 calculated  
Possible Combination: 21 out of 324 calculated  
Possible Combination: 22 out of 324 calculated  
Possible Combination: 23 out of 324 calculated  
Possible Combination: 24 out of 324 calculated  
Possible Combination: 25 out of 324 calculated  
Possible Combination: 26 out of 324 calculated  
Possible Combination: 27 out of 324 calculated  
Possible Combination: 28 out of 324 calculated  
Possible Combination: 29 out of 324 calculated  
Possible Combination: 30 out of 324 calculated  
Possible Combination: 31 out of 324 calculated  
Possible Combination: 32 out of 324 calculated  
Possible Combination: 33 out of 324 calculated  
Possible Combination: 34 out of 324 calculated  
Possible Combination: 35 out of 324 calculated  
Possible Combination: 36 out of 324 calculated  
Possible Combination: 37 out of 324 calculated  
Possible Combination: 38 out of 324 calculated  
Possible Combination: 39 out of 324 calculated  
Possible Combination: 40 out of 324 calculated

[illegible]

[illegible]



[illegible]

[illegible]

Possible Combination: 281 out of 324 calculated  
Possible Combination: 282 out of 324 calculated  
Possible Combination: 283 out of 324 calculated  
Possible Combination: 284 out of 324 calculated  
Possible Combination: 285 out of 324 calculated  
Possible Combination: 286 out of 324 calculated  
Possible Combination: 287 out of 324 calculated  
Possible Combination: 288 out of 324 calculated  
Possible Combination: 289 out of 324 calculated  
Possible Combination: 290 out of 324 calculated  
Possible Combination: 291 out of 324 calculated  
Possible Combination: 292 out of 324 calculated  
Possible Combination: 293 out of 324 calculated  
Possible Combination: 294 out of 324 calculated  
Possible Combination: 295 out of 324 calculated  
Possible Combination: 296 out of 324 calculated  
Possible Combination: 297 out of 324 calculated  
Possible Combination: 298 out of 324 calculated  
Possible Combination: 299 out of 324 calculated  
Possible Combination: 300 out of 324 calculated  
Possible Combination: 301 out of 324 calculated  
Possible Combination: 302 out of 324 calculated  
Possible Combination: 303 out of 324 calculated  
Possible Combination: 304 out of 324 calculated  
Possible Combination: 305 out of 324 calculated  
Possible Combination: 306 out of 324 calculated  
Possible Combination: 307 out of 324 calculated  
Possible Combination: 308 out of 324 calculated  
Possible Combination: 309 out of 324 calculated  
Possible Combination: 310 out of 324 calculated  
Possible Combination: 311 out of 324 calculated  
Possible Combination: 312 out of 324 calculated  
Possible Combination: 313 out of 324 calculated  
Possible Combination: 314 out of 324 calculated  
Possible Combination: 315 out of 324 calculated  
Possible Combination: 316 out of 324 calculated  
Possible Combination: 317 out of 324 calculated  
Possible Combination: 318 out of 324 calculated  
Possible Combination: 319 out of 324 calculated  
Possible Combination: 320 out of 324 calculated  
Possible Combination: 321 out of 324 calculated  
Possible Combination: 322 out of 324 calculated  
Possible Combination: 323 out of 324 calculated  
Possible Combination: 324 out of 324 calculated

-----  
Minimum MAPE for Chinese = 0.03932

Corresponding Best Parameters are (2, 1, 0, 0, 0, 0, 7)  
-----



---

### Finding best parameters for French

---

Possible Combination: 1 out of 324 calculated  
Possible Combination: 2 out of 324 calculated  
Possible Combination: 3 out of 324 calculated  
Possible Combination: 4 out of 324 calculated  
Possible Combination: 5 out of 324 calculated  
Possible Combination: 6 out of 324 calculated  
Possible Combination: 7 out of 324 calculated  
Possible Combination: 8 out of 324 calculated  
Possible Combination: 9 out of 324 calculated  
Possible Combination: 10 out of 324 calculated  
Possible Combination: 11 out of 324 calculated  
Possible Combination: 12 out of 324 calculated  
Possible Combination: 13 out of 324 calculated  
Possible Combination: 14 out of 324 calculated  
Possible Combination: 15 out of 324 calculated  
Possible Combination: 16 out of 324 calculated  
Possible Combination: 17 out of 324 calculated  
Possible Combination: 18 out of 324 calculated  
Possible Combination: 19 out of 324 calculated  
Possible Combination: 20 out of 324 calculated  
Possible Combination: 21 out of 324 calculated  
Possible Combination: 22 out of 324 calculated  
Possible Combination: 23 out of 324 calculated  
Possible Combination: 24 out of 324 calculated  
Possible Combination: 25 out of 324 calculated  
Possible Combination: 26 out of 324 calculated  
Possible Combination: 27 out of 324 calculated  
Possible Combination: 28 out of 324 calculated  
Possible Combination: 29 out of 324 calculated  
Possible Combination: 30 out of 324 calculated  
Possible Combination: 31 out of 324 calculated  
Possible Combination: 32 out of 324 calculated  
Possible Combination: 33 out of 324 calculated  
Possible Combination: 34 out of 324 calculated  
Possible Combination: 35 out of 324 calculated  
Possible Combination: 36 out of 324 calculated  
Possible Combination: 37 out of 324 calculated  
Possible Combination: 38 out of 324 calculated  
Possible Combination: 39 out of 324 calculated  
Possible Combination: 40 out of 324 calculated  
Possible Combination: 41 out of 324 calculated  
Possible Combination: 42 out of 324 calculated  
Possible Combination: 43 out of 324 calculated  
Possible Combination: 44 out of 324 calculated  
Possible Combination: 45 out of 324 calculated

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

Possible Combination: 286 out of 324 calculated  
Possible Combination: 287 out of 324 calculated  
Possible Combination: 288 out of 324 calculated  
Possible Combination: 289 out of 324 calculated  
Possible Combination: 290 out of 324 calculated  
Possible Combination: 291 out of 324 calculated  
Possible Combination: 292 out of 324 calculated  
Possible Combination: 293 out of 324 calculated  
Possible Combination: 294 out of 324 calculated  
Possible Combination: 295 out of 324 calculated  
Possible Combination: 296 out of 324 calculated  
Possible Combination: 297 out of 324 calculated  
Possible Combination: 298 out of 324 calculated  
Possible Combination: 299 out of 324 calculated  
Possible Combination: 300 out of 324 calculated  
Possible Combination: 301 out of 324 calculated  
Possible Combination: 302 out of 324 calculated  
Possible Combination: 303 out of 324 calculated  
Possible Combination: 304 out of 324 calculated  
Possible Combination: 305 out of 324 calculated  
Possible Combination: 306 out of 324 calculated  
Possible Combination: 307 out of 324 calculated  
Possible Combination: 308 out of 324 calculated  
Possible Combination: 309 out of 324 calculated  
Possible Combination: 310 out of 324 calculated  
Possible Combination: 311 out of 324 calculated  
Possible Combination: 312 out of 324 calculated  
Possible Combination: 313 out of 324 calculated  
Possible Combination: 314 out of 324 calculated  
Possible Combination: 315 out of 324 calculated  
Possible Combination: 316 out of 324 calculated  
Possible Combination: 317 out of 324 calculated  
Possible Combination: 318 out of 324 calculated  
Possible Combination: 319 out of 324 calculated  
Possible Combination: 320 out of 324 calculated  
Possible Combination: 321 out of 324 calculated  
Possible Combination: 322 out of 324 calculated  
Possible Combination: 323 out of 324 calculated  
Possible Combination: 324 out of 324 calculated

-----  
Minimum MAPE for French = 0.08091

Corresponding Best Parameters are (0, 1, 2, 0, 0, 0, 7)  
-----

-----  
Finding best parameters for German  
-----

Possible Combination: 1 out of 324 calculated  
Possible Combination: 2 out of 324 calculated

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

Possible Combination: 291 out of 324 calculated  
Possible Combination: 292 out of 324 calculated  
Possible Combination: 293 out of 324 calculated  
Possible Combination: 294 out of 324 calculated  
Possible Combination: 295 out of 324 calculated  
Possible Combination: 296 out of 324 calculated  
Possible Combination: 297 out of 324 calculated  
Possible Combination: 298 out of 324 calculated  
Possible Combination: 299 out of 324 calculated  
Possible Combination: 300 out of 324 calculated  
Possible Combination: 301 out of 324 calculated  
Possible Combination: 302 out of 324 calculated  
Possible Combination: 303 out of 324 calculated  
Possible Combination: 304 out of 324 calculated  
Possible Combination: 305 out of 324 calculated  
Possible Combination: 306 out of 324 calculated  
Possible Combination: 307 out of 324 calculated  
Possible Combination: 308 out of 324 calculated  
Possible Combination: 309 out of 324 calculated  
Possible Combination: 310 out of 324 calculated  
Possible Combination: 311 out of 324 calculated  
Possible Combination: 312 out of 324 calculated  
Possible Combination: 313 out of 324 calculated  
Possible Combination: 314 out of 324 calculated  
Possible Combination: 315 out of 324 calculated  
Possible Combination: 316 out of 324 calculated  
Possible Combination: 317 out of 324 calculated  
Possible Combination: 318 out of 324 calculated  
Possible Combination: 319 out of 324 calculated  
Possible Combination: 320 out of 324 calculated  
Possible Combination: 321 out of 324 calculated  
Possible Combination: 322 out of 324 calculated  
Possible Combination: 323 out of 324 calculated  
Possible Combination: 324 out of 324 calculated

-----  
Minimum MAPE for German = 0.08384

Corresponding Best Parameters are (2, 0, 1, 0, 0, 0, 7)  
-----  
-----

Finding best parameters for Japanese  
-----

Possible Combination: 1 out of 324 calculated  
Possible Combination: 2 out of 324 calculated  
Possible Combination: 3 out of 324 calculated  
Possible Combination: 4 out of 324 calculated  
Possible Combination: 5 out of 324 calculated  
Possible Combination: 6 out of 324 calculated  
Possible Combination: 7 out of 324 calculated

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

Possible Combination: 296 out of 324 calculated  
Possible Combination: 297 out of 324 calculated  
Possible Combination: 298 out of 324 calculated  
Possible Combination: 299 out of 324 calculated  
Possible Combination: 300 out of 324 calculated  
Possible Combination: 301 out of 324 calculated  
Possible Combination: 302 out of 324 calculated  
Possible Combination: 303 out of 324 calculated  
Possible Combination: 304 out of 324 calculated  
Possible Combination: 305 out of 324 calculated  
Possible Combination: 306 out of 324 calculated  
Possible Combination: 307 out of 324 calculated  
Possible Combination: 308 out of 324 calculated  
Possible Combination: 309 out of 324 calculated  
Possible Combination: 310 out of 324 calculated  
Possible Combination: 311 out of 324 calculated  
Possible Combination: 312 out of 324 calculated  
Possible Combination: 313 out of 324 calculated  
Possible Combination: 314 out of 324 calculated  
Possible Combination: 315 out of 324 calculated  
Possible Combination: 316 out of 324 calculated  
Possible Combination: 317 out of 324 calculated  
Possible Combination: 318 out of 324 calculated  
Possible Combination: 319 out of 324 calculated  
Possible Combination: 320 out of 324 calculated  
Possible Combination: 321 out of 324 calculated  
Possible Combination: 322 out of 324 calculated  
Possible Combination: 323 out of 324 calculated  
Possible Combination: 324 out of 324 calculated

-----  
Minimum MAPE for Japanese = 0.0934

Corresponding Best Parameters are (1, 1, 2, 0, 0, 1, 7)  
-----  
-----

Finding best parameters for Russian  
-----

Possible Combination: 1 out of 324 calculated  
Possible Combination: 2 out of 324 calculated  
Possible Combination: 3 out of 324 calculated  
Possible Combination: 4 out of 324 calculated  
Possible Combination: 5 out of 324 calculated  
Possible Combination: 6 out of 324 calculated  
Possible Combination: 7 out of 324 calculated  
Possible Combination: 8 out of 324 calculated  
Possible Combination: 9 out of 324 calculated  
Possible Combination: 10 out of 324 calculated  
Possible Combination: 11 out of 324 calculated  
Possible Combination: 12 out of 324 calculated

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

Possible Combination: 301 out of 324 calculated  
Possible Combination: 302 out of 324 calculated  
Possible Combination: 303 out of 324 calculated  
Possible Combination: 304 out of 324 calculated  
Possible Combination: 305 out of 324 calculated  
Possible Combination: 306 out of 324 calculated  
Possible Combination: 307 out of 324 calculated  
Possible Combination: 308 out of 324 calculated  
Possible Combination: 309 out of 324 calculated  
Possible Combination: 310 out of 324 calculated  
Possible Combination: 311 out of 324 calculated  
Possible Combination: 312 out of 324 calculated  
Possible Combination: 313 out of 324 calculated  
Possible Combination: 314 out of 324 calculated  
Possible Combination: 315 out of 324 calculated  
Possible Combination: 316 out of 324 calculated  
Possible Combination: 317 out of 324 calculated  
Possible Combination: 318 out of 324 calculated  
Possible Combination: 319 out of 324 calculated  
Possible Combination: 320 out of 324 calculated  
Possible Combination: 321 out of 324 calculated  
Possible Combination: 322 out of 324 calculated  
Possible Combination: 323 out of 324 calculated  
Possible Combination: 324 out of 324 calculated

-----  
Minimum MAPE for Russian = 0.06795

Corresponding Best Parameters are (0, 0, 1, 2, 0, 0, 7)  
-----

-----  
Finding best parameters for Spanish  
-----

Possible Combination: 1 out of 324 calculated  
Possible Combination: 2 out of 324 calculated  
Possible Combination: 3 out of 324 calculated  
Possible Combination: 4 out of 324 calculated  
Possible Combination: 5 out of 324 calculated  
Possible Combination: 6 out of 324 calculated  
Possible Combination: 7 out of 324 calculated  
Possible Combination: 8 out of 324 calculated  
Possible Combination: 9 out of 324 calculated  
Possible Combination: 10 out of 324 calculated  
Possible Combination: 11 out of 324 calculated  
Possible Combination: 12 out of 324 calculated  
Possible Combination: 13 out of 324 calculated  
Possible Combination: 14 out of 324 calculated  
Possible Combination: 15 out of 324 calculated  
Possible Combination: 16 out of 324 calculated  
Possible Combination: 17 out of 324 calculated

[illegible]

[illegible]

[illegible]





[illegible]

[illegible]

Possible Combination: 306 out of 324 calculated  
 Possible Combination: 307 out of 324 calculated  
 Possible Combination: 308 out of 324 calculated  
 Possible Combination: 309 out of 324 calculated  
 Possible Combination: 310 out of 324 calculated  
 Possible Combination: 311 out of 324 calculated  
 Possible Combination: 312 out of 324 calculated  
 Possible Combination: 313 out of 324 calculated  
 Possible Combination: 314 out of 324 calculated  
 Possible Combination: 315 out of 324 calculated  
 Possible Combination: 316 out of 324 calculated  
 Possible Combination: 317 out of 324 calculated  
 Possible Combination: 318 out of 324 calculated  
 Possible Combination: 319 out of 324 calculated  
 Possible Combination: 320 out of 324 calculated  
 Possible Combination: 321 out of 324 calculated  
 Possible Combination: 322 out of 324 calculated  
 Possible Combination: 323 out of 324 calculated  
 Possible Combination: 324 out of 324 calculated

-----  
 Minimum MAPE for Spanish = 0.14351  
 Corresponding Best Parameters are (2, 0, 0, 0, 0, 1, 7)  
 -----

```
[52]: best_param_df.sort_values(['mape'], inplace = True)
      best_param_df
```

```
[52]:
```

	language	p	d	q	P	D	Q	s	mape
0	Chinese	2	1	0	0	0	0	7	0.03932
4	Russian	0	0	1	2	0	0	7	0.06795
1	French	0	1	2	0	0	0	7	0.08091
2	German	2	0	1	0	0	0	7	0.08384
3	Japenese	1	1	2	0	0	1	7	0.09340
5	Spanish	2	0	0	0	0	1	7	0.14351

```
[53]: def plot_best_SARIMAX_model(languages, data_language, n, best_param_df):
      for lang in languages:
          # Fetching respective best parameters for that language
          params_lang = best_param_df[best_param_df['language'] == lang].iloc[0]
          p, d, q, P, D, Q, s = params_lang[['p', 'd', 'q', 'P', 'D', 'Q', 's']]

          # Creating language time-series
          time_series = data_language[data_language['language'] == lang][['Date',
          ↪ 'Visits']]
          time_series.set_index('Date', drop=True, inplace=True)

          # Creating SARIMAX Model
```

```

    model = SARIMAX(time_series[:-n], order=(p, d, q),
                    seasonal_order=(P, D, Q, s),
↳ initialization='approximate_diffuse')
    model_fit = model.fit()

    # Creating forecast for last n-values
    model_forecast = model_fit.forecast(n, dynamic=True)

    # Calculating MAPE & RMSE
    actuals = time_series.values[-n:]
    errors = time_series.values[-n:] - model_forecast.values
    mape = np.mean(np.abs(errors) / np.abs(actuals))
    rmse = np.sqrt(np.mean(errors**2))

    # Printing model statistics
    print(f'\n{"-" * 90}')
    print(f'SARIMAX model for {lang} Time Series')
    print(f'Parameters of Model: ({p}, {d}, {q}) ({P}, {D}, {Q}, {s})')
    print(f'MAPE of Model: {np.round(mape, 5)}')
    print(f'RMSE of Model: {np.round(rmse, 3)}')
    print(f'{"-" * 90}')

    # Plotting Actual & Forecasted values
    time_series.index = time_series.index.astype('datetime64[ns]')
    model_forecast.index = model_forecast.index.astype('datetime64[ns]')
    plt.figure(figsize=(20, 8))
    time_series[-60:].plot(label='Actual')
    model_forecast[-60:].plot(label='Forecast', color='red',
                               linestyle='dashed', marker='o',
↳ markerfacecolor='green', markersize=5)
    plt.legend(loc="upper right")
    plt.title(f'SARIMAX Model ({p}, {d}, {q}) ({P}, {D}, {Q}, {s}): Actual
↳ vs Forecasts',
              fontsize=15, fontweight='bold')
    plt.show()

    return 0

```

```

[54]: languages = ['Chinese', 'French', 'German', 'Japanese', 'Russian', 'Spanish']
      n = 30
      plot_best_SARIMAX_model(languages, lang_data, n, best_param_df)

```

---



---

```

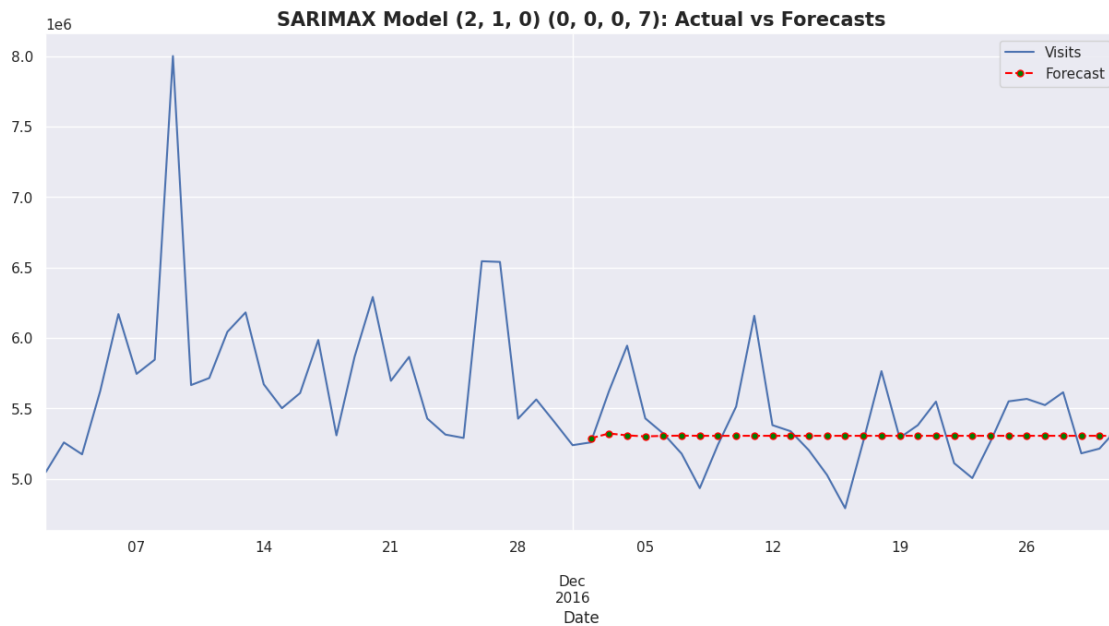
SARIMAX model for Chinese Time Series
Parameters of Model: (2, 1, 0) (0, 0, 0, 7)

```

MAPE of Model: 0.03932  
RMSE of Model: 289943.436

---

<Figure size 2000x800 with 0 Axes>

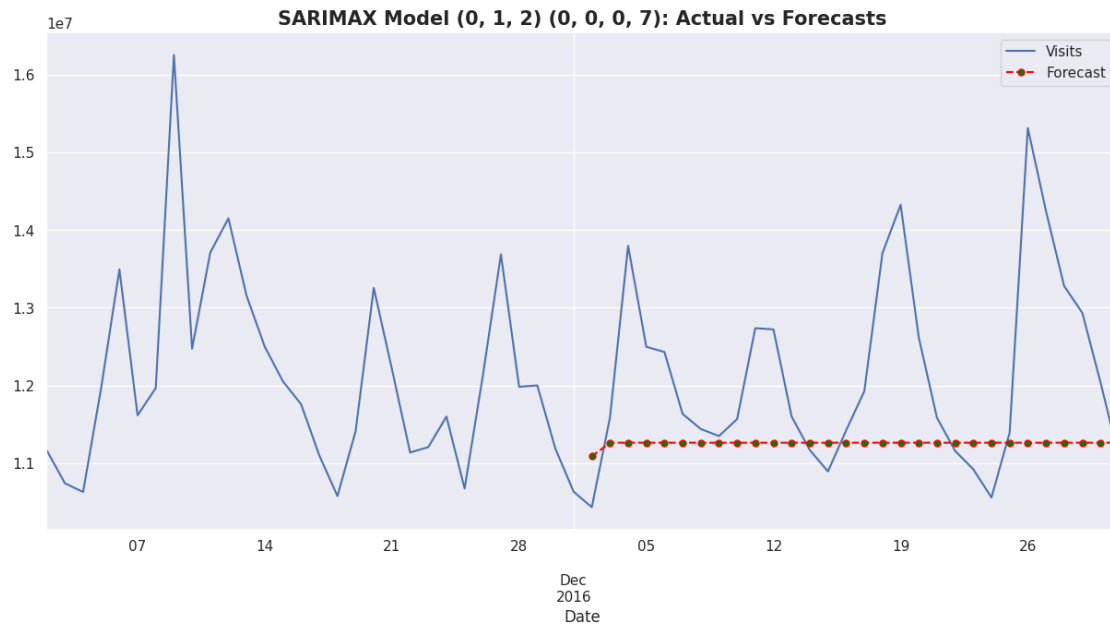


---

SARIMAX model for French Time Series  
Parameters of Model: (0, 1, 2) (0, 0, 0, 7)  
MAPE of Model: 0.08091  
RMSE of Model: 1489350.009

---

<Figure size 2000x800 with 0 Axes>

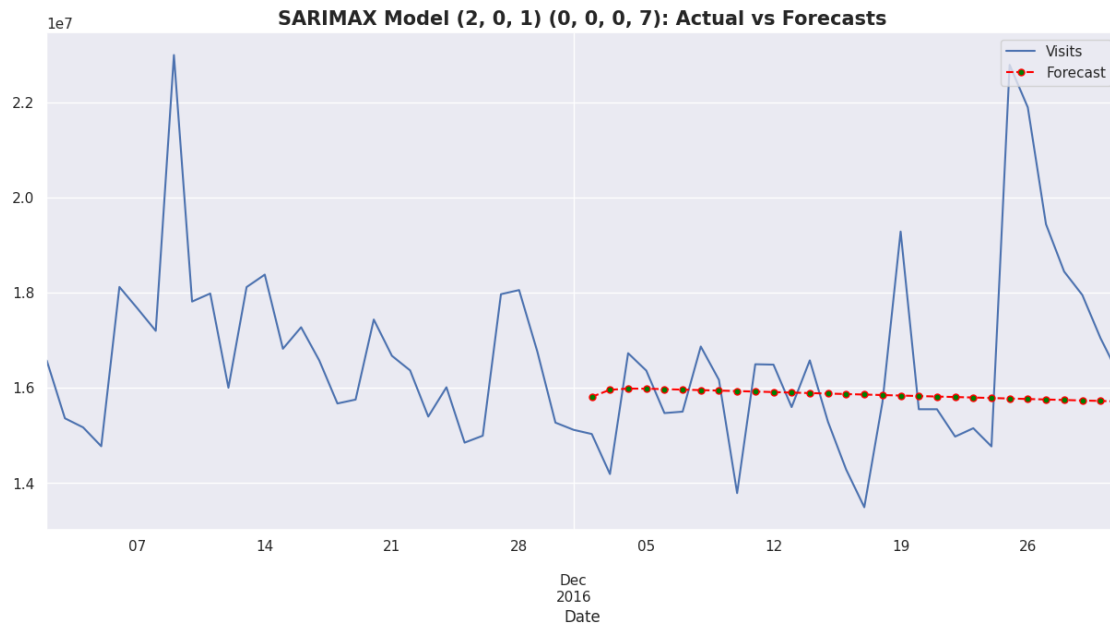


-----

SARIMAX model for German Time Series  
Parameters of Model: (2, 0, 1) (0, 0, 0, 7)  
MAPE of Model: 0.08384  
RMSE of Model: 2195114.679

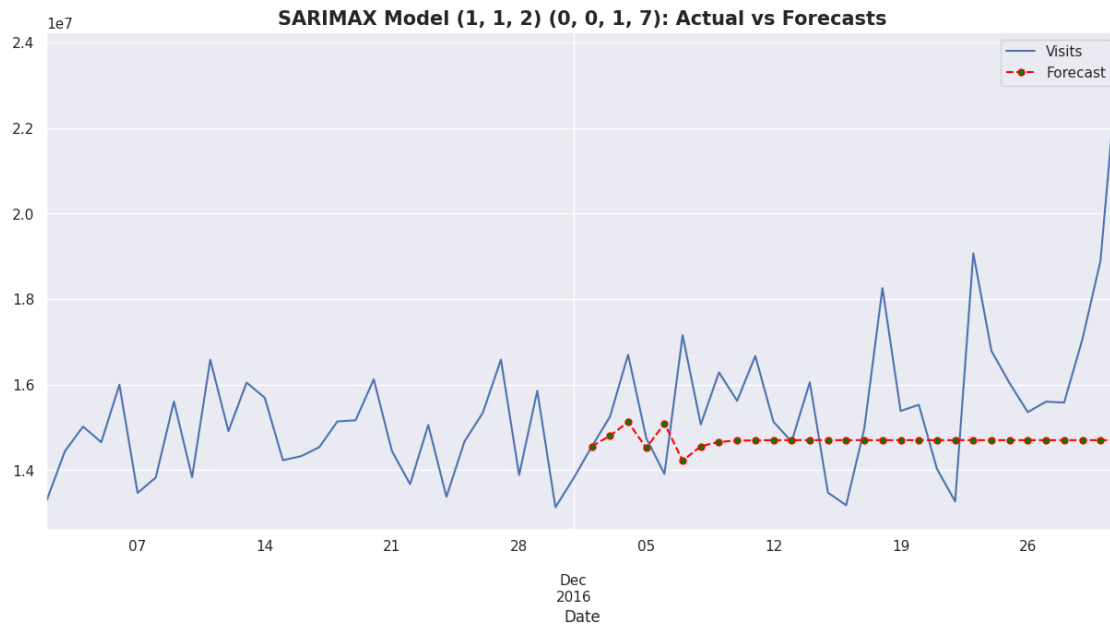
-----

<Figure size 2000x800 with 0 Axes>



-----  
 -----  
 SARIMAX model for Japanese Time Series  
 Parameters of Model: (1, 1, 2) (0, 0, 1, 7)  
 MAPE of Model: 0.0934  
 RMSE of Model: 2400870.834  
 -----  
 -----

<Figure size 2000x800 with 0 Axes>



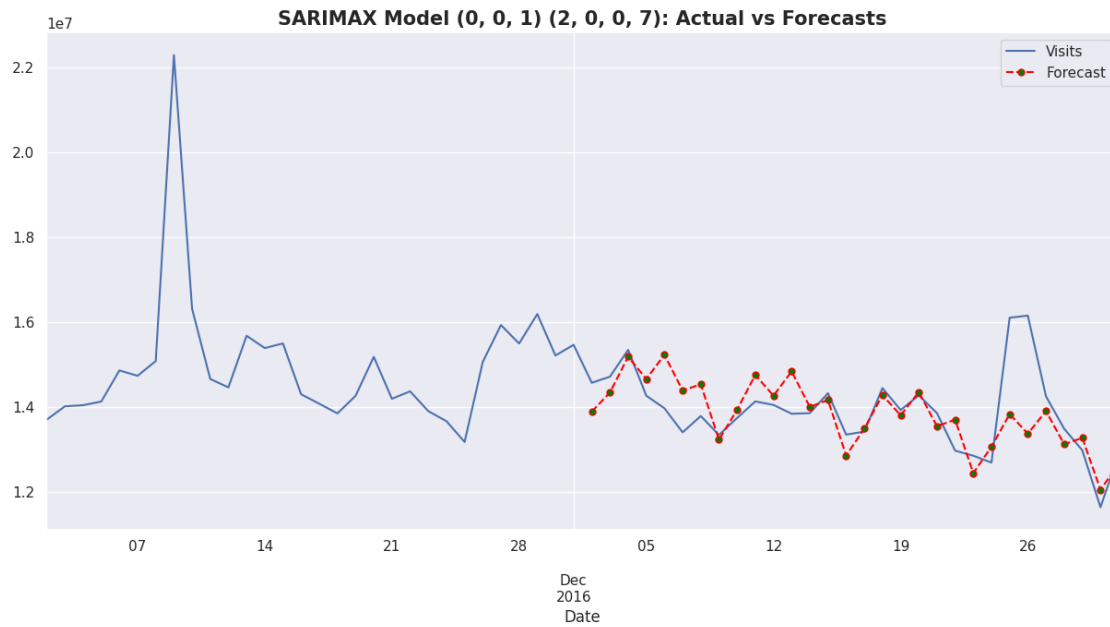
-----

SARIMAX model for Russian Time Series  
Parameters of Model: (0, 0, 1) (2, 0, 0, 7)  
MAPE of Model: 0.06795  
RMSE of Model: 1206324.353

-----

<Figure size 2000x800 with 0 Axes>



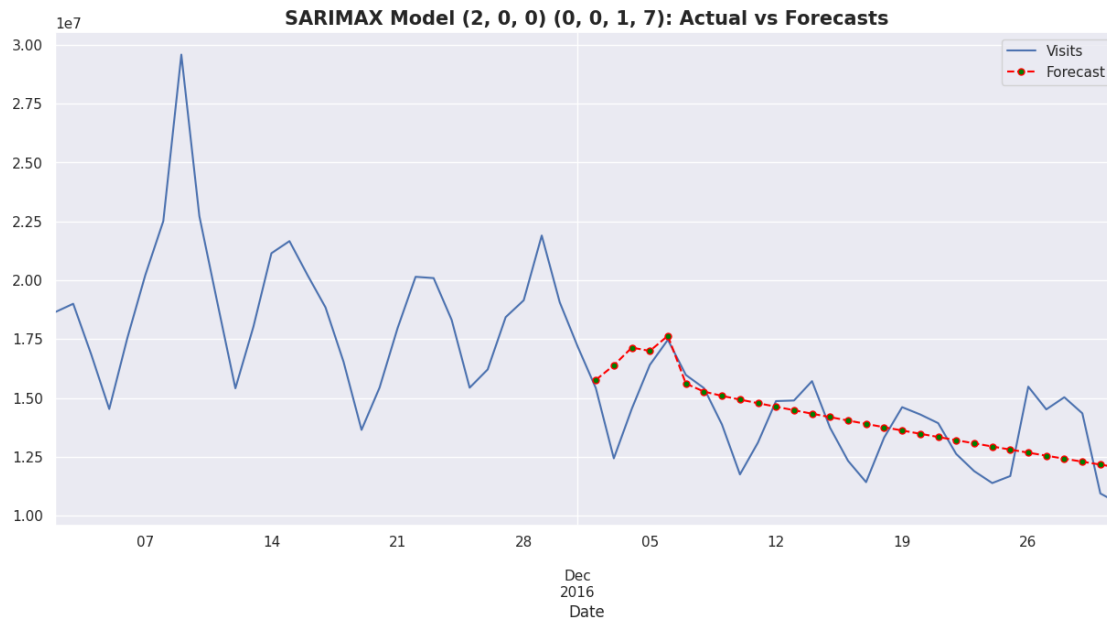


-----

SARIMAX model for Spanish Time Series  
Parameters of Model: (2, 0, 0) (0, 0, 1, 7)  
MAPE of Model: 0.14351  
RMSE of Model: 2344695.867

-----

<Figure size 2000x800 with 0 Axes>



[54]: 0

```
[55]: time_series = lang_data[lang_data['language'] == 'English'][['Date', 'Visits']]
# time_series.set_index('Date', drop=True, inplace=True)
time_series.columns = ['ds', 'y']
time_series['exog'] = exog
```

```
[56]: prophet1 = Prophet(weekly_seasonality=True)
prophet1.fit(time_series[['ds', 'y']][:30])
future = prophet1.make_future_dataframe(periods=30, freq='D')
forecast = prophet1.predict(future)
fig1 = prophet1.plot(forecast)
```

INFO:prophet:Disabling yearly seasonality. Run prophet with yearly\_seasonality=True to override this.

INFO:prophet:Disabling daily seasonality. Run prophet with daily\_seasonality=True to override this.

DEBUG:cmdstanpy:input tempfile: /tmp/tmpiejhvt6k/\_87itkxn.json

DEBUG:cmdstanpy:input tempfile: /tmp/tmpiejhvt6k/v4hc0\_fn.json

DEBUG:cmdstanpy:idx 0

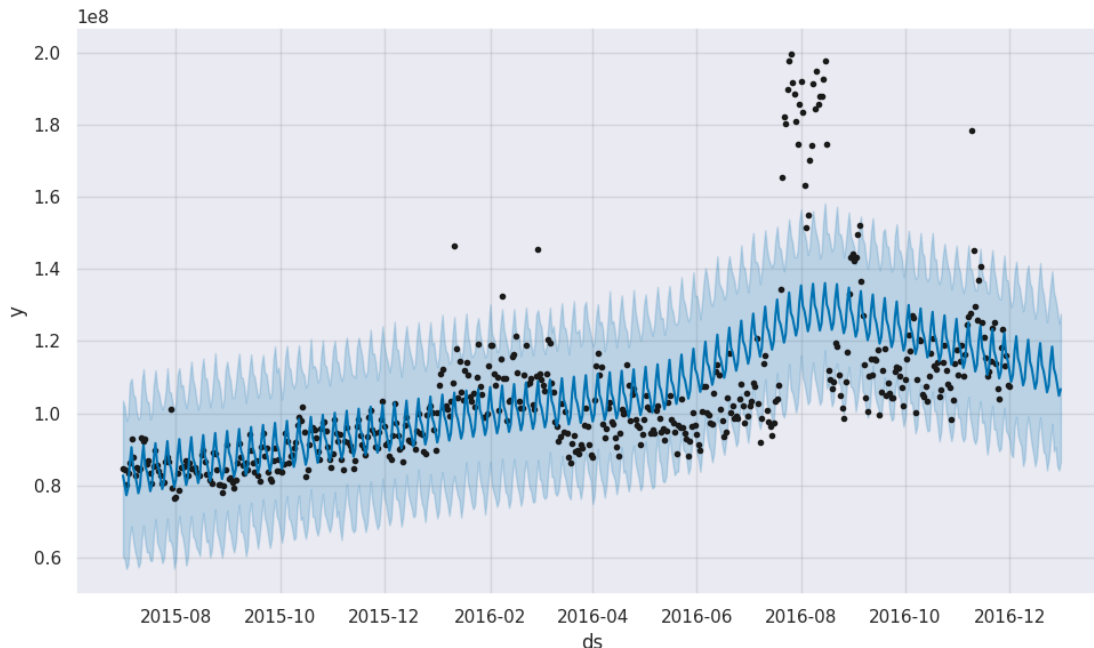
DEBUG:cmdstanpy:running CmdStan, num\_threads: None

DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan\_model/prophet\_model.bin', 'random', 'seed=82994', 'data', 'file=/tmp/tmpiejhvt6k/\_87itkxn.json', 'init=/tmp/tmpiejhvt6k/v4hc0\_fn.json', 'output', 'file=/tmp/tmpiejhvt6k/prophet\_modely3dgkts/prophet\_model-20240731183407.csv', 'method=optimize', 'algorithm=lbgfs', 'iter=10000']

```

18:34:07 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
18:34:07 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing

```



```

[57]: prophet2 = Prophet(weekly_seasonality=True)
prophet2.add_regressor('exog')
prophet2.fit(time_series[:-30])
#future2 = prophet2.make_future_dataframe(periods=30, freq= 'D')
forecast2 = prophet2.predict(time_series)
fig2 = prophet2.plot(forecast2)

```

```

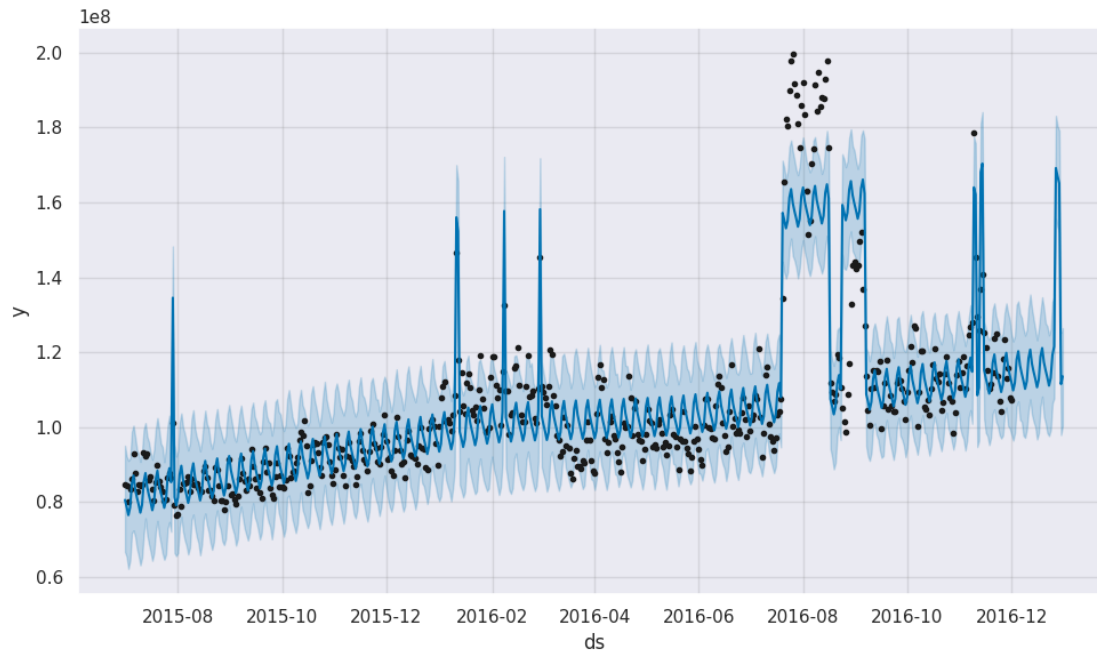
INFO:prophet:Disabling yearly seasonality. Run prophet with
yearly_seasonality=True to override this.
INFO:prophet:Disabling daily seasonality. Run prophet with
daily_seasonality=True to override this.
DEBUG:cmdstanpy:input tempfile: /tmp/tmpiejhvt6k/qojn0guf.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmpiejhvt6k/63plgtae.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-
packages/prophet/stan_model/prophet_model.bin', 'random', 'seed=63648', 'data',
'file=/tmp/tmpiejhvt6k/qojn0guf.json', 'init=/tmp/tmpiejhvt6k/63plgtae.json',
'output',
'file=/tmp/tmpiejhvt6k/prophet_model1mc2ddjk/prophet_model-20240731183408.csv',
'method=optimize', 'algorithm=lbgfs', 'iter=10000']

```

```

18:34:08 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
18:34:09 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing

```

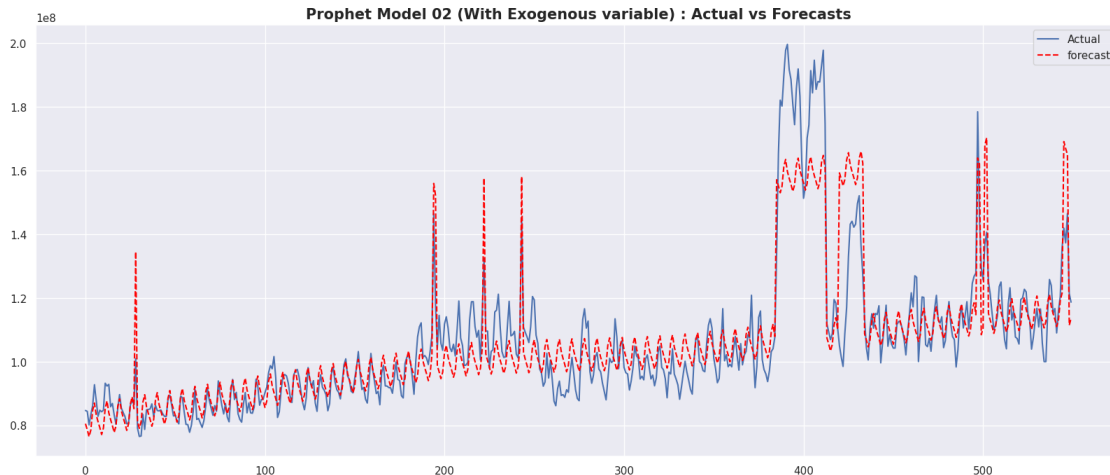


```

[58]: actual = time_series['y'].values
forecast = forecast2['yhat'].values

plt.figure(figsize = (20,8))
plt.plot(actual, label = 'Actual')
plt.plot(forecast, label = 'forecast', color = 'red', linestyle='dashed')
plt.legend(loc="upper right")
plt.title(f'Prophet Model 02 (With Exogenous variable) : Actual vs Forecasts',
        ↪fontsize = 15, fontweight = 'bold')
plt.show()

```



```
[59]: errors = abs(actual - forecast)
      mape = np.mean(errors/abs(actual))
      mape
```

[59]: 0.05983786254333203

FB Prophet Model is able to capture peaks because of exogenous variable and is giving a MAPE of 6%

### 0.1.3 Recommendations

1. Prioritize English language pages due to their low MAPE and high mean visits, making them optimal for advertising efforts to maximize reach and effectiveness.
2. Avoid advertising on Chinese language pages unless there's a specific marketing strategy tailored for Chinese populations, as they have the lowest number of visits.
3. Russian language pages present a promising opportunity for high conversion rates with their decent number of visits and low MAPE if utilized effectively.
4. Despite having the second-highest number of visits, Spanish language pages exhibit the highest MAPE, suggesting that advertisements on these pages may not effectively reach the intended audience.
5. French, German, and Japanese language pages show moderate levels of visits and MAPE. Depending on the target customers, consider advertising campaigns on these pages to capitalize on their potential reach and conversion rates.

### 0.1.4 Questionnaire

1. Defining the problem statements and where can this and modifications of this be used? > Identification of the problem and its applications:
  - The Data Science team at Ad ease aims to analyze per page view reports for various Wikipedia pages spanning 550 days.

- The objective includes forecasting page views to enhance ad placement optimization for clients.
- Dataset encompasses 145k Wikipedia pages with daily view counts.
- Client base extends across diverse regions, necessitating insights into ad performance across different languages.

Importance of forecasting model: - Implementing a robust forecasting model is pivotal in predicting fluctuations in page visits. - This model aids the business team in optimizing marketing expenditure. - Precise prediction of high-traffic days enables strategic ad placement, maximizing audience reach while optimizing spending.

2. Write 3 inferences you made from the data visualizations. > Inferences from Data Visualizations:

- Linguistic Diversity: The data reveals the presence of 7 languages, with English dominating, followed by Japanese, German, and French.
- Access Type Distribution: Three access types are identified—All-access, mobile-web, and desktop—comprising 51.4%, 24.9%, and 23.6% respectively.
- Access-Origin Insights: The dataset illustrates two access origins—‘all-agents’ and ‘spider’—with ‘all-agents’ constituting 75.8% and ‘spider’ 24.2% of the data.

Advertising Strategies: - English Language Dominance: English emerges as the most prominent language, suggesting prioritized advertisement placement due to its low Mean Absolute Percentage Error (MAPE) and high mean visit count. - Chinese Language Considerations: Pages in Chinese exhibit the lowest visit counts, signaling caution in advertisement allocation unless specifically targeting Chinese demographics. - Russian Language Potential: Russian language pages demonstrate a favorable balance between visit count and MAPE, indicating potential for maximum conversion if utilized effectively. - Spanish Language Challenges: Despite being the second-highest in visit count, Spanish pages exhibit the highest MAPE, suggesting potential challenges in advertisement efficacy. - Moderate Performers: French, German, and Japanese languages present medium-level visit counts and MAPE levels, prompting tailored advertisement strategies based on target customer demographics.

## 0.2 Time Series Decomposition

3. What does the decomposition of series do? Time series decomposition is a statistical technique used to break down a time series into its constituent components in order to understand its underlying structure, trends, seasonality, and irregular fluctuations. The decomposition typically involves separating the time series data into three main components:
4. **Trend ((T<sub>t</sub>))**: The long-term movement or pattern in the data, representing the overall direction in which the time series is moving.
5. **Seasonality ((S<sub>t</sub>))**: The repeating patterns or fluctuations that occur at regular intervals within the time series data.
6. **Residuals ((R<sub>t</sub>))**: The remaining variation in the data after removing the trend and seasonality components.

The time series ( $y_t$ ) can be decomposed into its components as follows:

- **Additive Decomposition:**  $[y_t = T_t + S_t + R_t]$

- **Multiplicative Decomposition:**  $[y_t = T_t \times S_t \times R_t]$

Various techniques such as moving averages, exponential smoothing, or mathematical models can be used to estimate the trend and seasonal components, leaving the residual component as the leftover variation in the data.

4. What level of differencing gave you a stationary series?

First order differencing

5. Difference between arima, sarima & sarimax.

**ARIMA (Autoregressive Integrated Moving Average):** - ARIMA is a time series forecasting model that combines autoregression (AR), differencing (I), and moving average (MA) components. - It's suitable for univariate time series data without exogenous variables. - ARIMA(p,d,q) where p represents the autoregressive order, d represents the differencing order, and q represents the moving average order.

**SARIMA (Seasonal Autoregressive Integrated Moving Average):** - SARIMA is an extension of ARIMA that incorporates seasonal components in addition to the non-seasonal ones. - It's suitable for time series data with seasonal patterns. - SARIMA(p,d,q)(P,D,Q)m where P, D, and Q represent the seasonal autoregressive, differencing, and moving average orders respectively, and 'm' represents the seasonal period.

**SARIMAX (Seasonal Autoregressive Integrated Moving Average with Exogenous Variables):** - SARIMAX extends SARIMA by allowing the inclusion of exogenous variables, which are external factors that can influence the time series. - It's suitable for time series data with both seasonal patterns and external variables. - SARIMAX(p,d,q)(P,D,Q)m with exogenous variables.

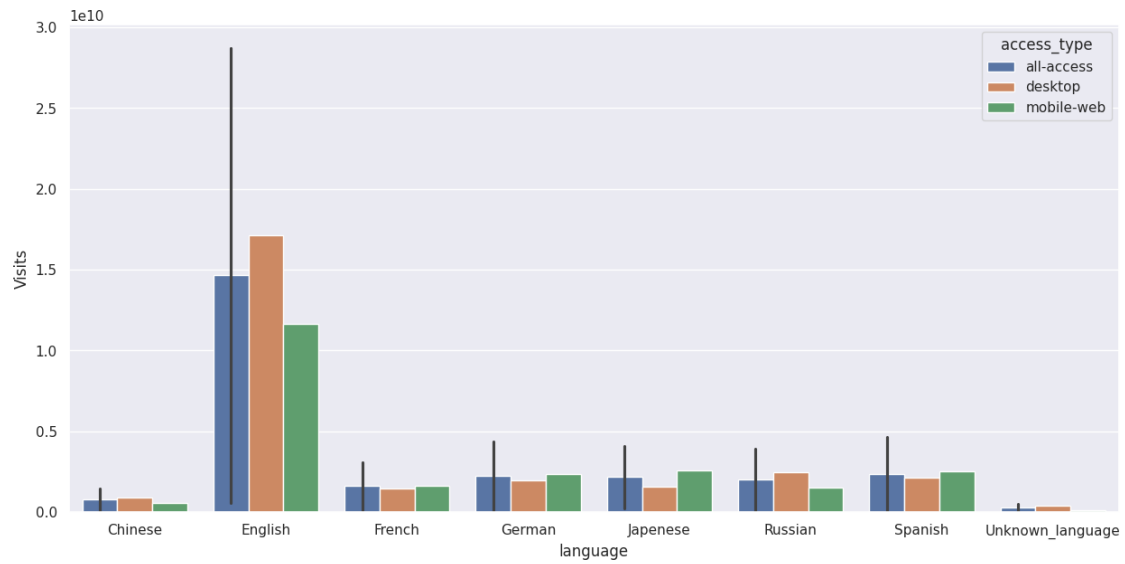
These models are commonly used in time series analysis and forecasting tasks, each offering different capabilities to handle various types of data and patterns.

6. Compare the number of views in different languages

```
[60]: grouped = reshaped.groupby(['language', 'access_type', 'access_origin'],
    ↪as_index=False)['Visits'].sum()
```

```
[61]: sns.barplot(grouped, x="language", y="Visits", hue="access_type")
```

```
[61]: <Axes: xlabel='language', ylabel='Visits'>
```



7. What other methods other than grid search would be suitable to get the model for all languages?
- We can use packages like hyperopt, optuna and sci-kit-optimize
  - We can try and use different models like tsmixer and deep learning models