```
# Data Dictionary:

# There are two csv files given
# train_1.csv: In the csv file, each row corresponds to a particular article and each column corresponds to a particular date. The values are

# The page name contains data in this format:
# SPECIFIC NAME _ LANGUAGE.wikipedia.org _ ACCESS TYPE _ ACCESS ORIGIN
# having information about the page name, the main domain, the device type used to access the page, and also the request origin(spider or bro

# Exog_Campaign_eng: This file contains data for the dates which had a campaign or significant event that could affect the views for that day

# There’s 1 for dates with campaigns and 0 for remaining dates. It is to be treated as an exogenous variable for models when training and for


#·Evaluation·Criteria·(100·points)

#·Importing·the·dataset·and·doing·usual·exploratory·analysis·steps·like·checking·the·structure·&·characteristics·of·the·dataset·(10·points)

#·Exploratory·Data·Analysis·(20·points)

#·Separating·the·data

#·Analyzing·and·visualizing·the·data

#·Getting·inferences

#·Checking·stationarity·(20·points)

#·Formatting·the·data·for·the·model

#·Dickey·fuller·test

#·Decomposition

#·Differencing

#·Creating·model·training·and·forecasting·with·ARIMA,·SARIMAX·(20·points)

#·ACF·and·PACF·plot.
```

Saved successfully!

```
#·Forecasting·for·different·languages/regions.

#·Plotting·the·final·results

#·Forecasting·with·(20·points)

#·Facebook·prophet
#·Creating·a·pipeline·for·working·with·multiple·series·(10·points)


!pip install pmdarima
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: pmdarima in /usr/local/lib/python3.9/dist-packages (2.0.3)
Requirement already satisfied: pandas>=0.19 in /usr/local/lib/python3.9/dist-packages (from pmdarima) (1.5.3)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.9/dist-packages (from pmdarima) (1.2.0)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.9/dist-packages (from pmdarima) (1.26.15)
Requirement already satisfied: scikit-learn>=0.22 in /usr/local/lib/python3.9/dist-packages (from pmdarima) (1.2.2)
Requirement already satisfied: setuptools!=50.0.0,>=38.6.0 in /usr/local/lib/python3.9/dist-packages (from pmdarima) (67.6.1)
Requirement already satisfied: numpy>=1.21.2 in /usr/local/lib/python3.9/dist-packages (from pmdarima) (1.22.4)
Requirement already satisfied: Cython!=0.29.18,!=0.29.31,>=0.29 in /usr/local/lib/python3.9/dist-packages (from pmdarima) (0.29.34)
Requirement already satisfied: statsmodels>=0.13.2 in /usr/local/lib/python3.9/dist-packages (from pmdarima) (0.13.5)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.9/dist-packages (from pmdarima) (1.10.1)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.9/dist-packages (from pandas>=0.19->pmdarima) (2022.7.1)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.9/dist-packages (from pandas>=0.19->pmdarima) (2.8.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.9/dist-packages (from scikit-learn>=0.22->pmdarima) (3.1.0
Requirement already satisfied: patsy>=0.5.2 in /usr/local/lib/python3.9/dist-packages (from statsmodels>=0.13.2->pmdarima) (0.5.3)
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.9/dist-packages (from statsmodels>=0.13.2->pmdarima) (23.1)
Requirement already satisfied: six in /usr/local/lib/python3.9/dist-packages (from patsy>=0.5.2->statsmodels>=0.13.2->pmdarima) (1.16.0)
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
import seaborn as sns
import pmdarima as pm
```

Importing the dataset and doing usual exploratory analysis steps like checking the structure & characteristics of the dataset

```
df= pd.read_csv('/content/Exog_Campaign_eng')
df.head()
```

	Exog
0	0
1	0
2	0
3	0
4	0

```
df2 = pd.read_csv('/content/train_1.csv')
```

```
df2.head()
```

	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	...	2016-12-22	2016-12-23	2016-12-24	2016-12-25	201 12-
0	2NE1_zh.wikipedia.org_all-access_spider	18.0	11.0	5.0	13.0	14.0	9.0	9.0	22.0	26.0	...	32.0	63.0	15.0	26.0	14
1	2PM_zh.wikipedia.org_all-access_spider	11.0	14.0	15.0	18.0	11.0	13.0	22.0	11.0	10.0	...	17.0	42.0	28.0	15.0	9
2	3C_zh.wikipedia.org_all-access_spider	1.0	0.0	1.0	1.0	0.0	4.0	0.0	3.0	4.0	...	3.0	1.0	1.0	7.0	4
3	4minute_zh.wikipedia.org_all-access_spider	35.0	13.0	10.0	94.0	4.0	26.0	14.0	9.0	11.0	...	32.0	10.0	26.0	27.0	16
4	...org_all-ess_s...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	48.0	9.0	25.0	13.0	3

Saved successfully!

```
df2.shape
```

```
(145063, 551)
```

```
df2.isna().sum()
```

```
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
```

```
df2.columns
```

```
Index(['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',
       ...
       '2016-12-22', '2016-12-23', '2016-12-24', '2016-12-25', '2016-12-26',
       '2016-12-27', '2016-12-28', '2016-12-29', '2016-12-30', '2016-12-31'],
      dtype='object', length=551)
```

```
page=df2['Page'].str.rsplit("_",n=3,expand=True)
page.columns = ['Name','wiki','acc_type','acc_origin']
```

```
page.head()
```

	Name	wiki	acc_type	acc_origin
0	2NE1	zh.wikipedia.org	all-access	spider
1	2PM	zh.wikipedia.org	all-access	spider
2	3C	zh.wikipedia.org	all-access	spider
3	4minute	zh.wikipedia.org	all-access	spider
4	52_Hz_I_Love_You	zh.wikipedia.org	all-access	spider

```
import re
def find_language(url):
    res = re.search('[a-z][a-z].wikipedia.org',url)
    if res:
        return res[0][0:2]
    return 'na'



x = []
for i in page['wiki']:
  x.append(find_language(i))


page['lang'] = x


page.head()
```

	Name	wiki	acc_type	acc_origin	lang
0	2NE1	zh.wikipedia.org	all-access	spider	zh
1	2PM	zh.wikipedia.org	all-access	spider	zh
2	3C	zh.wikipedia.org	all-access	spider	zh
3	4minute	zh.wikipedia.org	all-access	spider	zh
		.org	all-access	spider	zh

Saved successfully!

```
page.lang.value_counts()
```

```
en    24108
ja    20431
de    18547
na    17855
fr    17802
zh    17229
ru    15022
es    14069
Name: lang, dtype: int64
```

```
train=pd.melt(df2[list(df2.columns[-5:])+['Page']], id_vars='Page', var_name='date', value_name='Visits')


result = pd.concat([page,df2], axis=1, join='inner')


result.drop('Page',axis=1,inplace=True)


result
```

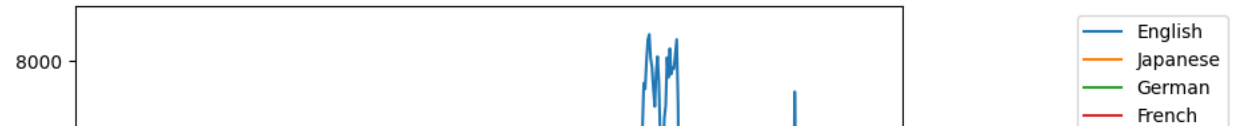
	Name	wiki	acc_type	acc_origin	lang	2015-07-01	2015-07-02	2015-07-03	2015-07-04	2015-07-05	...	20 12
0	2NE1	zh.wikipedia.org	all-access	spider	zh	18.0	11.0	5.0	13.0	14.0	...	3
1	2PM	zh.wikipedia.org	all-access	spider	zh	11.0	14.0	15.0	18.0	11.0	...	1
2	3C	zh.wikipedia.org	all-access	spider	zh	1.0	0.0	1.0	1.0	0.0	...	
3	4minute	zh.wikipedia.org	all-access	spider	zh	35.0	13.0	10.0	94.0	4.0	...	3
4	52_Hz_I_Love_You	zh.wikipedia.org	all-access	spider	zh	NaN	NaN	NaN	NaN	NaN	...	4
...	...	...	...	...	...	...	...	...	...	...	...	
145058	Underworld_(serie_de_películas)	es.wikipedia.org	all-access	spider	es	NaN	NaN	NaN	NaN	NaN	...	N
145059	Resident_Evil:_Capítulo_Final	es.wikipedia.org	all-access	spider	es	NaN	NaN	NaN	NaN	NaN	...	N

Exploratory Data Analysis

```
lang_set = {}
lang_set['en'] = result[result.lang=='en'].iloc[:,0:-1]
lang_set['ja'] = result[result.lang == 'ja'].iloc[:, 0:-1]
lang_set['de'] = result[result.lang == 'de'].iloc[:, 0:-1]
lang_set['fr'] = result[result.lang == 'fr'].iloc[:, 0:-1]
lang_set['ru'] = result[result.lang == 'ru'].iloc[:, 0:-1]
lang_set['es'] = result[result.lang == 'es'].iloc[:, 0:-1]
lang_set['zh'] = result[result.lang == 'zh'].iloc[:, 0:-1]
lang_set['na'] = result[result.lang == 'na'].iloc[:, 0:-1]


total_view = {}
```

Saved successfully!

```
iloc[:, 5:].sum(axis=0) / lang_set[key].shape[0]

days = [r for r in range(total_view['en'].shape[0])]

plt.figure(figsize=(8, 6))
labels={'ja':'Japanese','de':'German','en' : 'English','no_lang':'Media_File','fr':'French','zh':'Chinese',
        'ru':'Russian','es':'Spanish','na':'Null'}

for key in total_view:
    plt.plot(days,total_view[key],label = labels[key])

plt.legend(loc = 'upper left', bbox_to_anchor = (1.2, 1))
plt.show()
```



train

	Page	date	Visits
0	2NE1_zh.wikipedia.org_all-access_spider	2016-12-27	20.0
1	2PM_zh.wikipedia.org_all-access_spider	2016-12-27	30.0
2	3C_zh.wikipedia.org_all-access_spider	2016-12-27	4.0
3	4minute_zh.wikipedia.org_all-access_spider	2016-12-27	11.0
4	52_Hz_I_Love_You_zh.wikipedia.org_all-access_s...	2016-12-27	11.0
...	...	...	...
725310	Underworld_(serie_de_películas)_es.wikipedia.o...	2016-12-31	10.0
725311	Resident_Evil:_Capítulo_Final_es.wikipedia.org...	2016-12-31	NaN
725312	Enamorándome_de_Ramón_es.wikipedia.org_all-acc...	2016-12-31	NaN
725313	Hasta_el_último_hombre_es.wikipedia.org_all-ac...	2016-12-31	NaN
725314	Francisco_el_matemático_(serie_de_televisión_d...	2016-12-31	NaN

725315 rows × 3 columns

Checking stationarity

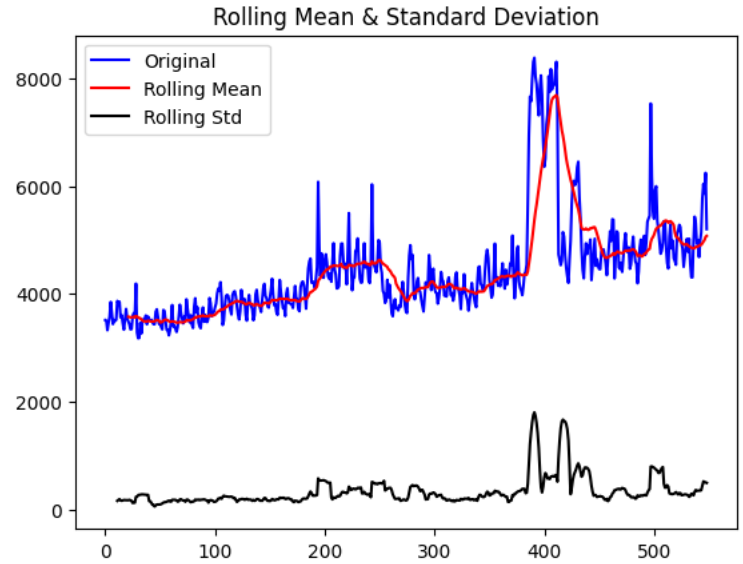
Null hypothesis: TS is non-stationary.

The test results comprise of a Test Statistic and some Critical Values for difference confidence levels. If the 'Test Statistic' is less than the Critical Value', we can reject the null hypothesis and say that the series is stationary.

Saved successfully!

```
t adfuller

def test_stationarity(x):


    #Determing rolling statistics
    rolmean = x.rolling(window=22,center=False).mean()

    rolstd = x.rolling(window=12,center=False).std()

    #Plot rolling statistics:
    orig = plt.plot(x.values, color='blue',label='Original')
    mean = plt.plot(rolmean.values, color='red', label='Rolling Mean')
    std = plt.plot(rolstd.values, color='black', label = 'Rolling Std')
    plt.legend(loc='best')
    plt.title('Rolling Mean & Standard Deviation')
    plt.show(block=False)

    #Perform Dickey Fuller test
    result=adfuller(x)
    print('ADF Stastistic: %f'%result[0])
    print('p-value: %f'%result[1])
    pvalue=result[1]
    for key,value in result[4].items():
         if result[0]>value:
            print("The graph is non stationery")
            break
         else:
            print("The graph is stationery")
            break;
    print('Critical values:')
    for key,value in result[4].items():
        print('\t%s: %.3f' % (key, value))
```

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

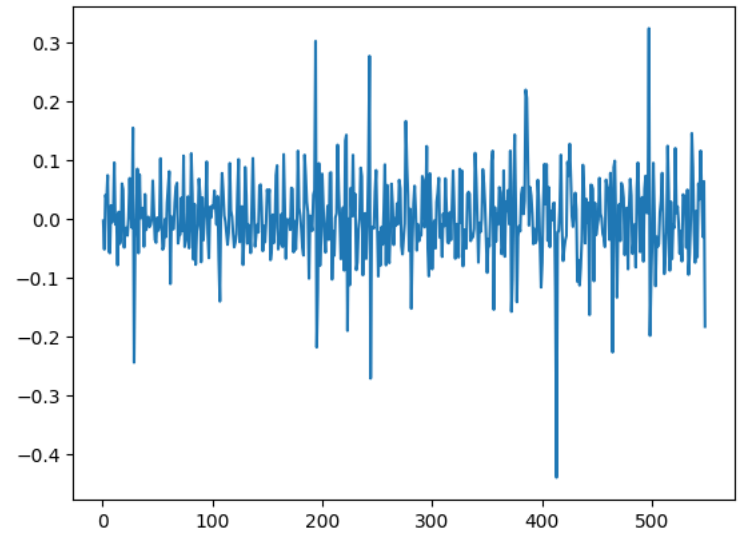



```
ADF Stastistic: -2.245201
p-value: 0.190251
The graph is non stationery
Critical values:
	1%: -3.443
	5%: -2.867
	10%: -2.570
```

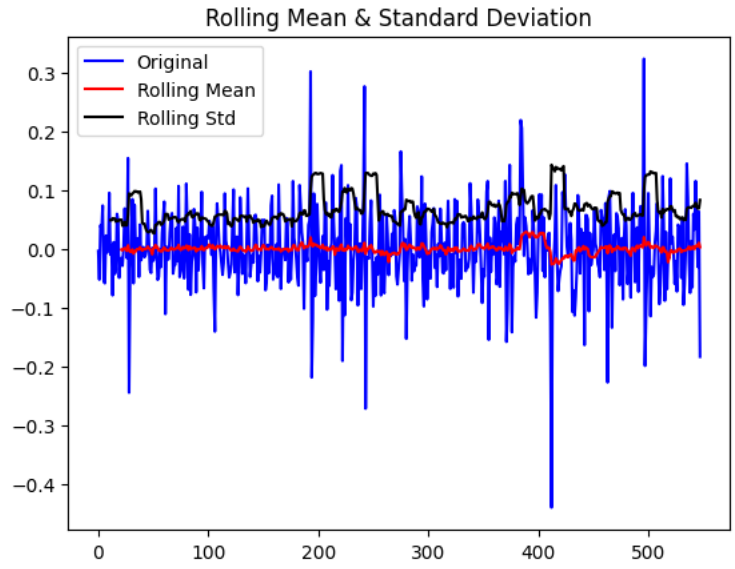
```
ts_log = np.log(total_view['en'])
from statsmodels.tsa.seasonal import seasonal_decompose
decomposition = seasonal_decompose(ts_log.values, model='multiplicative',period = 7)

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

Saved successfully!

```
ts_log_diff = ts_log - ts_log.shift()
plt.plot(ts_log_diff.values)
plt.show()
```




```
ts_log_diff.dropna(inplace=True)
test_stationarity(ts_log_diff)
```



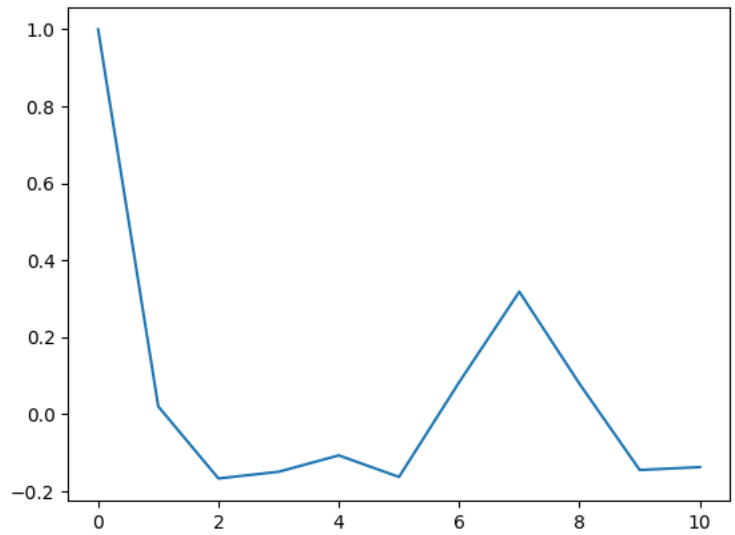

```
ADF Stastistic: -8.130271
p-value: 0.000000
The graph is stationery
```

The ADF statistic is much lesser than critical value at 1%. So there is 99% confidence interval that our graph is now stationery. Now we can apply the ARIMA model

Plotting the ACF and PACF plots

```
from statsmodels.tsa.stattools import acf, pacf

lag_acf = acf(ts_log_diff, nlags=10)
lag_pacf = pacf(ts_log_diff, nlags=10, method='ols')
```

Saved successfully!




```
plt.plot(lag_pacf)
plt.show()
```

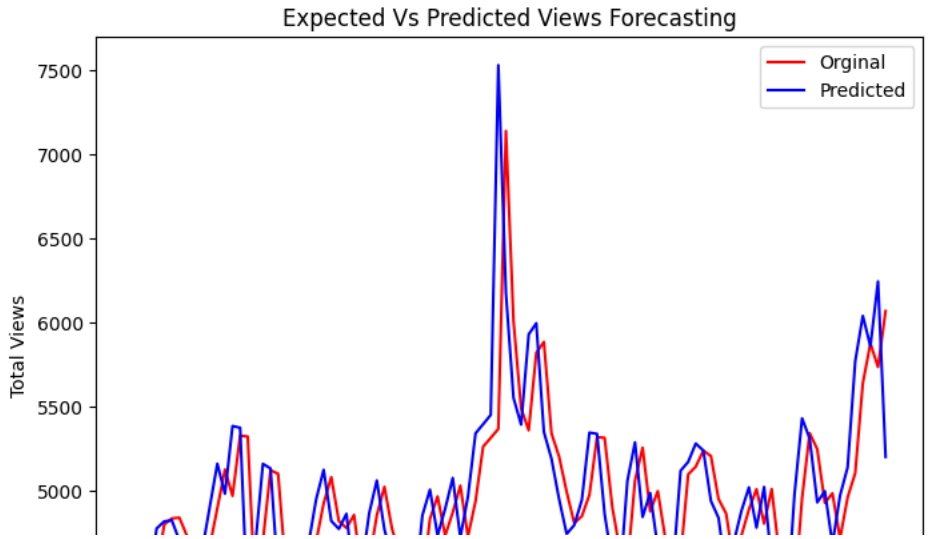
Creating model training and forecasting with ARIMA, SARIMAX

```
−0.2
```

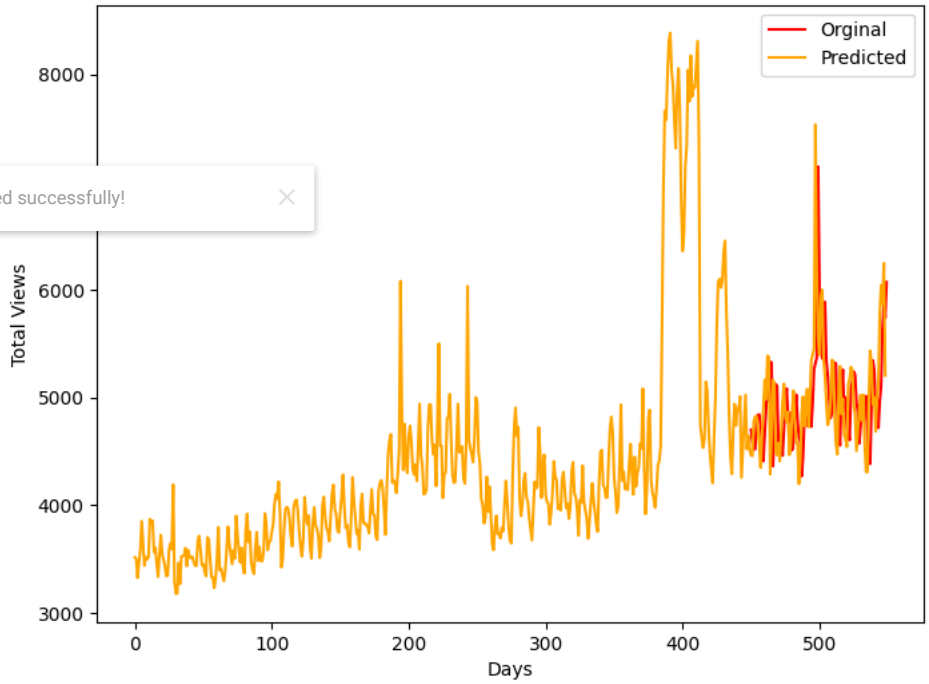
```
from statsmodels.tsa.arima.model import ARIMA

model = ARIMA(ts_log.values, order=(0,1,1))
results_ARIMA = model.fit()


size = int(len(ts_log)-100)
train_arima, test_arima = ts_log[0:size], ts_log[size:len(ts_log)]
history = [x for x in train_arima]
predictions = list()
originals = list()
error_list = list()

print('\n')
for t in range(len(test_arima)):
    model = ARIMA(history, order=(1, 1, 1))
    model_fit = model.fit()
    output = model_fit.forecast()
    pred_value = output[0]
    original_value = test_arima[t]
    history.append(original_value)
    history = history[1:]
    error = ((abs(pred_value - original_value)) / original_value) * 100
    error_list.append(error)
    predictions.append(float(pred_value))
    originals.append(float(original_value))

print('\n Means Error in Predicting Test Case Articles : %f ' % (sum(error_list)/float(len(error_list))), '%')
```

```
 Printing Predicted vs Expected Values...


/usr/local/lib/python3.9/dist-packages/statsmodels/base/model.py:604: ConvergenceWarning: Maximum Likelihood optimization failed to conv
  warnings.warn("Maximum Likelihood optimization failed to "
/usr/local/lib/python3.9/dist-packages/statsmodels/base/model.py:604: ConvergenceWarning: Maximum Likelihood optimization failed to conv
  warnings.warn("Maximum Likelihood optimization failed to "

 Means Error in Predicting Test Case Articles : 5.383650  %
```

```
plt.figure(figsize=(8, 6))
test_day = [t+450 for t in range(len(test_arima))]
labels={'Orginal','Predicted'}
plt.plot(test_day, predictions, color= 'red')
plt.plot(test_day, originals, color = 'blue')
plt.title('Expected Vs Predicted Views Forecasting')
plt.ylabel('Total Views')
plt.legend(labels)
plt.show()
```




```
plt.figure(figsize=(8, 6))
test_day = [t+450
            for t in range(len(test_arima))]
labels={'Orginal','Predicted'}
plt.plot(test_day, predictions, color= 'red')
plt.plot(days, total_view['en'], color = 'orange')
plt.xlabel('Days')
plt.ylabel('Total Views')
plt.legend(labels)
plt.show()
```



Saved successfully!

```
print(results_ARIMA.summary())
```

```
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                  549
Model:                 ARIMA(0, 1, 1)   Log Likelihood                 682.259
Date:                Fri, 21 Apr 2023   AIC                          -1360.518
Time:                        12:28:04   BIC                          -1351.905
Sample:                             0   HQIC                         -1357.151
                                  - 549
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ma.L1          0.0293      0.028      1.052      0.293      -0.025       0.084
sigma2         0.0049      0.000     30.542      0.000       0.005       0.005
```

```
================================================================================
Ljung-Box (L1) (Q):                   0.01   Jarque-Bera (JB):               538.66
Prob(Q):                              0.91   Prob(JB):                         0.00
Heteroskedasticity (H):               2.39   Skew:                            -0.30
Prob(H) (two-sided):                  0.00   Kurtosis:                         7.82
================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```

```
sxmodel = pm.auto_arima(ts_log.values, exogenous=df['Exog'],start_p=1, start_q=1,test='adf',max_p=3, max_q=3, m=12,start_P=0, seasonal=True,d
```

```
sxmodel.summary()
```

```
Performing stepwise search to minimize aic
 ARIMA(1,0,1)(0,1,1)[12] intercept   : AIC=inf, Time=4.48 sec
 ARIMA(0,0,0)(0,1,0)[12] intercept   : AIC=-419.959, Time=0.13 sec
 ARIMA(1,0,0)(1,1,0)[12] intercept   : AIC=-1078.979, Time=1.31 sec
 ARIMA(0,0,1)(0,1,1)[12] intercept   : AIC=inf, Time=2.65 sec
 ARIMA(0,0,0)(0,1,0)[12]             : AIC=-420.260, Time=0.07 sec
 ARIMA(1,0,0)(0,1,0)[12] intercept   : AIC=-930.994, Time=0.08 sec
 ARIMA(1,0,0)(2,1,0)[12] intercept   : AIC=-1206.715, Time=4.50 sec
 ARIMA(1,0,0)(2,1,1)[12] intercept   : AIC=inf, Time=7.31 sec
 ARIMA(1,0,0)(1,1,1)[12] intercept   : AIC=inf, Time=2.62 sec
 ARIMA(0,0,0)(2,1,0)[12] intercept   : AIC=-526.410, Time=1.54 sec
 ARIMA(2,0,0)(2,1,0)[12] intercept   : AIC=-1205.558, Time=5.83 sec
 ARIMA(1,0,1)(2,1,0)[12] intercept   : AIC=-1205.781, Time=5.65 sec
 ARIMA(0,0,1)(2,1,0)[12] intercept   : AIC=-910.511, Time=3.49 sec
 ARIMA(2,0,1)(2,1,0)[12] intercept   : AIC=-1204.601, Time=7.48 sec
 ARIMA(1,0,0)(2,1,0)[12]             : AIC=-1208.159, Time=1.36 sec
 ARIMA(1,0,0)(1,1,0)[12]             : AIC=-1080.652, Time=0.55 sec
 ARIMA(1,0,0)(2,1,1)[12]             : AIC=inf, Time=4.61 sec
 ARIMA(1,0,0)(1,1,1)[12]             : AIC=inf, Time=1.25 sec
 ARIMA(0,0,0)(2,1,0)[12]             : AIC=-523.261, Time=0.39 sec
 ARIMA(2,0,0)(2,1,0)[12]             : AIC=-1206.970, Time=1.53 sec
 ARIMA(1,0,1)(2,1,0)[12]             : AIC=-1207.188, Time=1.61 sec
 ARIMA(0,0,1)(2,1,0)[12]             : AIC=-908.660, Time=1.32 sec
 ARIMA(2,0,1)(2,1,0)[12]             : AIC=-1206.014, Time=2.42 sec

Best model:  ARIMA(1,0,0)(2,1,0)[12]
Total fit time: 62.218 seconds
```

SARIMAX Results

Dep. Variable:	y	**No. Observations:**	549
	×(2, 1, 0, 12)	**Log Likelihood**	608.080
		AIC	-1208.159
Time:	12:29:06	**BIC**	-1191.015
Sample:	0	**HQIC**	-1201.452
	- 549		
Covariance Type:	opg		

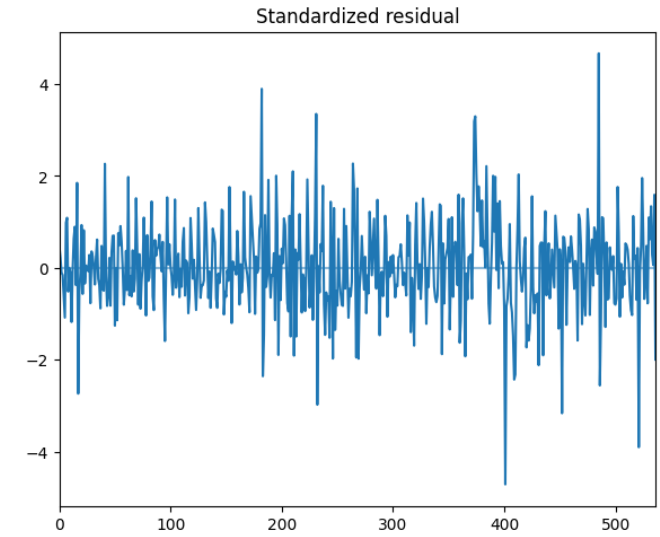
	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.8687	0.017	51.313	0.000	0.835	0.902
ar.S.L12	-0.7507	0.034	-21.938	0.000	-0.818	-0.684
ar.S.L24	-0.4757	0.030	-15.790	0.000	-0.535	-0.417
sigma2	0.0060	0.000	24.298	0.000	0.005	0.006

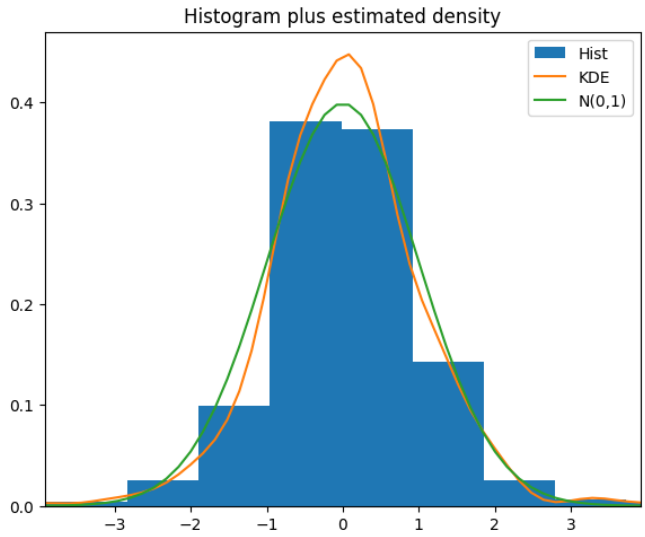
Ljung-Box (L1) (Q):	0.56	**Jarque-Bera (JB):**	138.58
Prob(Q):	0.46	**Prob(JB):**	0.00
Heteroskedasticity (H):	2.54	**Skew:**	0.03
Prob(H) (two-sided):	0.00	**Kurtosis:**	5.49

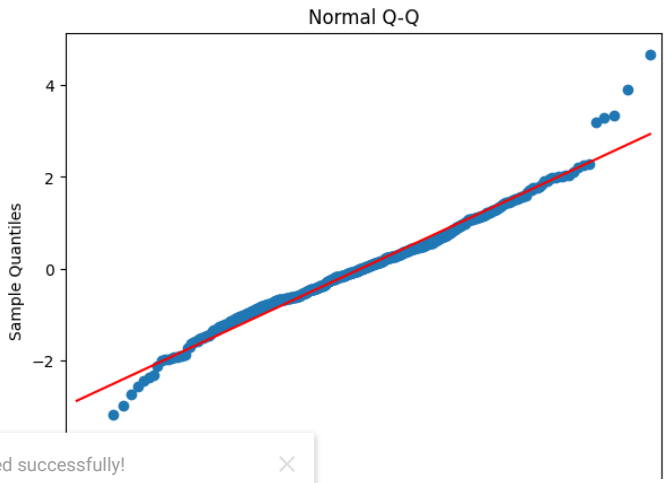
Saved successfully!

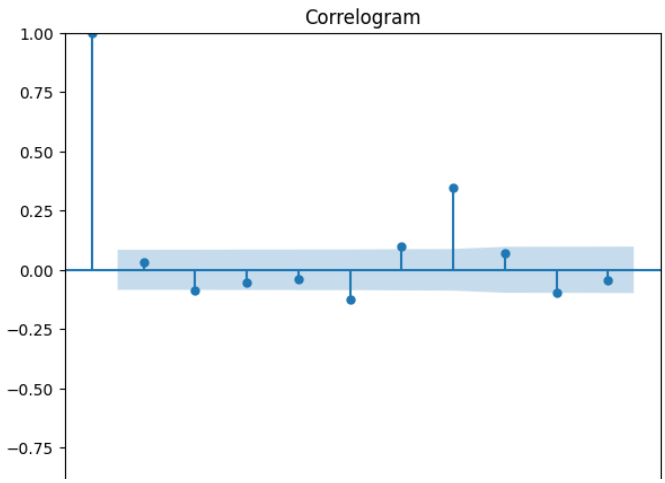
Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

```
sxmodel.plot_diagnostics(figsize=(15,12))
plt.show()
```









Saved successfully!

Forecasting with Facebook Prophet

```
!pip install pystan~=2.14
!pip install fbprophet
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting pystan~=2.14
  Downloading pystan-2.19.1.1.tar.gz (16.2 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 16.2/16.2 MB 70.5 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: Cython!=0.25.1,>=0.22 in /usr/local/lib/python3.9/dist-packages (from pystan~=2.14) (0.29.34)
Requirement already satisfied: numpy>=1.7 in /usr/local/lib/python3.9/dist-packages (from pystan~=2.14) (1.22.4)
Building wheels for collected packages: pystan
  Building wheel for pystan (setup.py) ... done
  Created wheel for pystan: filename=pystan-2.19.1.1-cp39-cp39-linux_x86_64.whl size=61827094 sha256=03e83b4751e83c3559696d4fcb094fb9
  Stored in directory: /root/.cache/pip/wheels/b8/36/bf/7ec7e363f796373cea3eb9ea94e83f5bbbb586d2edbf7e3417
Successfully built pystan
Installing collected packages: pystan
  Attempting uninstall: pystan
    Found existing installation: pystan 3.6.0
    Uninstalling pystan-3.6.0:
      Successfully uninstalled pystan-3.6.0
Successfully installed pystan-2.19.1.1
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting fbprophet
  Using cached fbprophet-0.7.1.tar.gz (64 kB)
  Preparing metadata (setup.py) ... done
Requirement already satisfied: Cython>=0.22 in /usr/local/lib/python3.9/dist-packages (from fbprophet) (0.29.34)
Requirement already satisfied: cmdstanpy==0.9.5 in /usr/local/lib/python3.9/dist-packages (from fbprophet) (0.9.5)
Requirement already satisfied: pystan>=2.14 in /usr/local/lib/python3.9/dist-packages (from fbprophet) (2.19.1.1)
Requirement already satisfied: numpy>=1.15.4 in /usr/local/lib/python3.9/dist-packages (from fbprophet) (1.22.4)
Requirement already satisfied: pandas>=1.0.4 in /usr/local/lib/python3.9/dist-packages (from fbprophet) (1.5.3)
Requirement already satisfied: matplotlib>=2.0.0 in /usr/local/lib/python3.9/dist-packages (from fbprophet) (3.7.1)
Requirement already satisfied: LunarCalendar>=0.0.9 in /usr/local/lib/python3.9/dist-packages (from fbprophet) (0.0.9)
```

```
Requirement already satisfied: convertdate>=2.1.2 in /usr/local/lib/python3.9/dist-packages (from fbprophet) (2.4.0)
Requirement already satisfied: holidays>=0.10.2 in /usr/local/lib/python3.9/dist-packages (from fbprophet) (0.22)
Requirement already satisfied: setuptools-git>=1.2 in /usr/local/lib/python3.9/dist-packages (from fbprophet) (1.2)
Requirement already satisfied: python-dateutil>=2.8.0 in /usr/local/lib/python3.9/dist-packages (from fbprophet) (2.8.2)
Requirement already satisfied: tqdm>=4.36.1 in /usr/local/lib/python3.9/dist-packages (from fbprophet) (4.65.0)
Requirement already satisfied: pymeeus<=1,>=0.3.13 in /usr/local/lib/python3.9/dist-packages (from convertdate>=2.1.2->fbprophet) (0.5
Requirement already satisfied: hijri-converter in /usr/local/lib/python3.9/dist-packages (from holidays>=0.10.2->fbprophet) (2.2.4)
Requirement already satisfied: korean-lunar-calendar in /usr/local/lib/python3.9/dist-packages (from holidays>=0.10.2->fbprophet) (0.3
Requirement already satisfied: ephem>=3.7.5.3 in /usr/local/lib/python3.9/dist-packages (from LunarCalendar>=0.0.9->fbprophet) (4.1.4
Requirement already satisfied: pytz in /usr/local/lib/python3.9/dist-packages (from LunarCalendar>=0.0.9->fbprophet) (2022.7.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.9/dist-packages (from matplotlib>=2.0.0->fbprophet) (4.39.3
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.9/dist-packages (from matplotlib>=2.0.0->fbprophet) (23.1)
Requirement already satisfied: importlib-resources>=3.2.0 in /usr/local/lib/python3.9/dist-packages (from matplotlib>=2.0.0->fbprophet
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.9/dist-packages (from matplotlib>=2.0.0->fbprophet) (3.0.9)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.9/dist-packages (from matplotlib>=2.0.0->fbprophet) (1.0.7)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.9/dist-packages (from matplotlib>=2.0.0->fbprophet) (1.4.4
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.9/dist-packages (from matplotlib>=2.0.0->fbprophet) (0.11.0)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.9/dist-packages (from matplotlib>=2.0.0->fbprophet) (8.4.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.9/dist-packages (from python-dateutil>=2.8.0->fbprophet) (1.16.0)
Requirement already satisfied: zipp>=3.1.0 in /usr/local/lib/python3.9/dist-packages (from importlib-resources>=3.2.0->matplotlib>=2.0
Building wheels for collected packages: fbprophet
  Building wheel for fbprophet (setup.py) ... done
  Created wheel for fbprophet: filename=fbprophet-0.7.1-py3-none-any.whl size=9436653 sha256=1007a9258d5c95cc6afca65a5c59ee0d83bec543
  Stored in directory: /root/.cache/pip/wheels/da/a4/bb/dbed5db92b2183a753dd96cc8a56706a61484ff3959988bdaa
Successfully built fbprophet
Installing collected packages: fbprophet
Successfully installed fbprophet-0.7.1
```

```
from fbprophet import Prophet


y = ts_log.values


ds= result.columns[5:-1]



data = {'ds':ds,'y':y}


df = pd.DataFrame(data)
```

Saved successfully!

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 549 entries, 0 to 548
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   ds      549 non-null    object
 1   y       549 non-null    float64
dtypes: float64(1), object(1)
memory usage: 8.7+ KB
```

```
df['ds'] = df['ds'].astype('datetime64[ns]')


df.info()
```

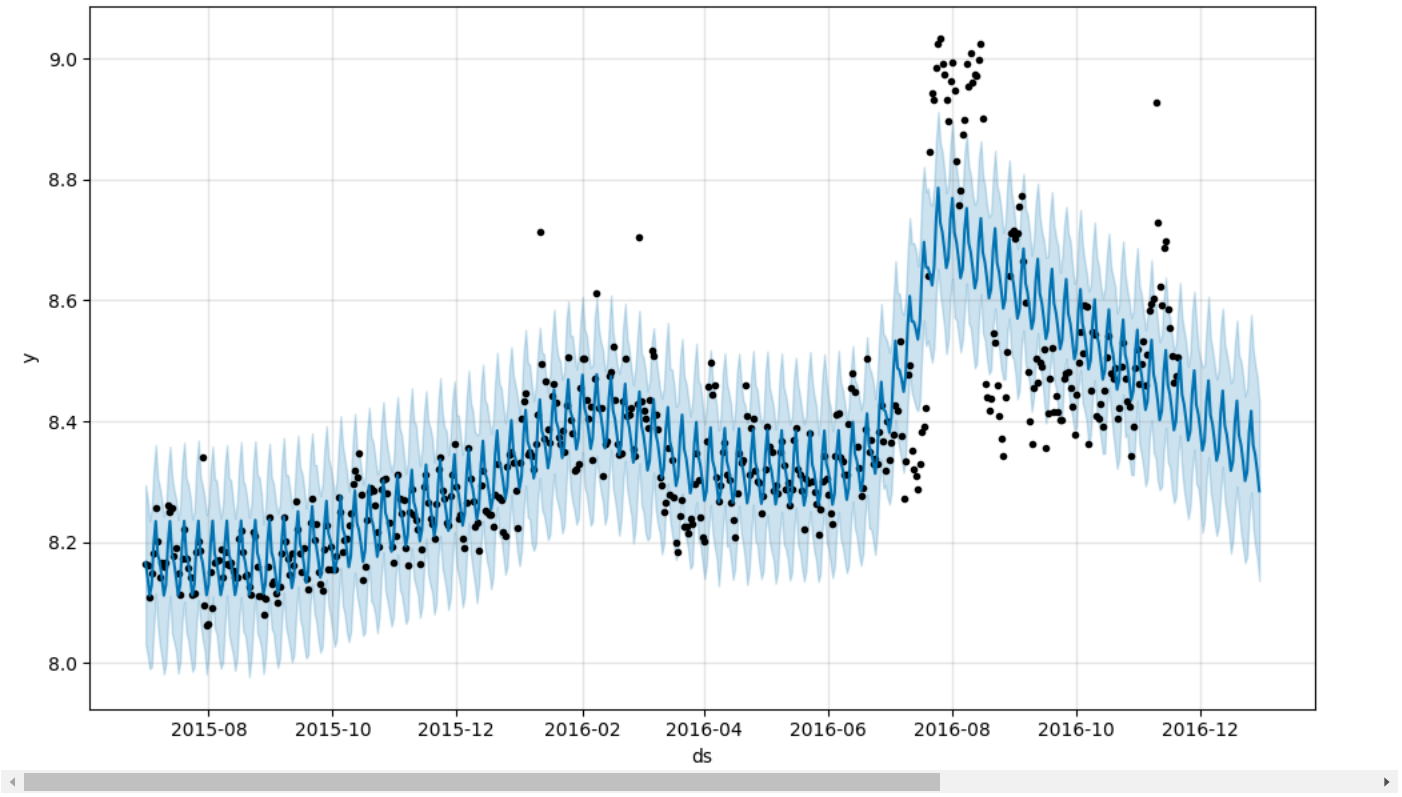
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 549 entries, 0 to 548
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   ds      549 non-null    datetime64[ns]
 1   y       549 non-null    float64
dtypes: datetime64[ns](1), float64(1)
memory usage: 8.7 KB
```

```
model = Prophet()
model.fit(df[['ds','y']][:-40])
```

```
INFO:fbprophet:Disabling yearly seasonality. Run prophet with yearly_seasonality=True to override this.
INFO:fbprophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to override this.
/usr/local/lib/python3.9/dist-packages/fbprophet/forecaster.py:891: FutureWarning: The frame.append method is deprecated and will be rem
  components = components.append(new_comp)
<fbprophet.forecaster.Prophet at 0x7f56f778f9a0>
```

```
future = model.make_future_dataframe(periods = 40,freq = 'D')
forecast =model.predict(future)
fg = model.plot(forecast)
```

```
/usr/local/lib/python3.9/dist-packages/fbprophet/forecaster.py:891: FutureWarning: The frame.append method is deprecated and will be rem
  components = components.append(new_comp)
/usr/local/lib/python3.9/dist-packages/fbprophet/forecaster.py:891: FutureWarning: The frame.append method is deprecated and will be rem
  components = components.append(new_comp)
```



Saved successfully!

from the data visualizations

1. There are 7 languages plus the media pages. The languages used here are: English, Japanese, German, French, Chinese, Russian, and Spanish. Article count as per Language : ({'en': 24108, 'ja': 20431, 'de': 18547, 'no_lang': 17855, 'fr': 17802, 'zh': 17229, 'ru': 15022, 'es': 14069})
2. English shows a much higher number of views per page, as might be expected since Wikipedia is a US-based site.

What does the decomposition of series do?

Decomposition is task in which the Time Series data is decomposed into several component or extracting seasonality, trend from a series data. These components are defined as follows:

Level: The average value in the series. Trend: The increasing or decreasing value in the series. Seasonality: The repeating short-term cycle in the series. Noise: The random variation in the series.

What level of differencing gave you a stationary series?

Log-transform stationarity is been applied as After Dickley Fuller test, seems that the data is not stationary and after log tranformation the data is turned out to stationary.

Difference between arima, sarima & sarimax.

$$Y_t = \beta_2 + \omega_1 \varepsilon_{t-1} + \omega_2 \varepsilon_{t-2} + \ldots + \omega_q \varepsilon_{t-q} + \varepsilon_t$$

Arima: The ARIMA model is an ARMA model yet with a preprocessing step included in the model that we represent I(d) is the difference order, which is the number of transformations needed to make the data stationary. So, an ARIMA model is simply an ARMA model on the differenced time series.

SARIMA:

$$y_t = c + \sum_{n=1}^{p} \alpha_n y_{t-n} + \sum_{n=1}^{q} \theta_n \epsilon_{t-n} + \sum_{n=1}^{P} \phi_n y_{t-sn} + \sum_{n=1}^{Q} \eta_n \epsilon_{t-sn} + \epsilon_t$$

Searson ARIMA also known as SARIMA is a very similar to the ARIMA model, except that there is an additional set of autoregressive and moving average components.The additional lags are offset by the frequency of seasonality (ex. 12 — monthly, 24 — hourly).

SARIMAX:

$$d_t = c + \sum_{n=1}^{p} \alpha_n d_{t-n} + \sum_{n=1}^{q} \theta_n \epsilon_{t-n} + \sum_{n=1}^{r} \beta_n x_{n_t} + \sum_{n=1}^{P} \phi_n d_{t-sn} + \sum_{n=1}^{Q} \eta_n \epsilon_{t-sn} + \epsilon_t$$

This model takes into account exogenous variables, or in other words, use external data in our forecast. Some real-world examples of exogenous variables include gold price, oil price, outdoor temperature, exchange rate.

Saved successfully!