BusinessCase

by Lohith Kumar Kasula

Context

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

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

Concepts Applied:

Exploratory data analysis

Time Series forecasting- ARIMA, SARIMAX, and Prophet

In [1]:
```
!pip install colorama
!pip install category_encoders
!pip install -U scikit-learn
!pip install pycountry
```

```
Collecting colorama
  Downloading colorama-0.4.6-py2.py3-none-any.whl.metadata (17 kB)
Downloading colorama-0.4.6-py2.py3-none-any.whl (25 kB)
Installing collected packages: colorama
Successfully installed colorama-0.4.6
Collecting category_encoders
  Downloading category_encoders-2.8.1-py3-none-any.whl.metadata (7.9 kB)
Requirement already satisfied: numpy>=1.14.0 in /usr/local/lib/python3.11/dist-pac
kages (from category_encoders) (2.0.2)
Requirement already satisfied: pandas>=1.0.5 in /usr/local/lib/python3.11/dist-pac
kages (from category_encoders) (2.2.2)
Requirement already satisfied: patsy>=0.5.1 in /usr/local/lib/python3.11/dist-pack
ages (from category_encoders) (1.0.1)
Requirement already satisfied: scikit-learn>=1.6.0 in /usr/local/lib/python3.11/di
st-packages (from category_encoders) (1.6.1)
Requirement already satisfied: scipy>=1.0.0 in /usr/local/lib/python3.11/dist-pack
ages (from category_encoders) (1.14.1)
Requirement already satisfied: statsmodels>=0.9.0 in /usr/local/lib/python3.11/dis
t-packages (from category_encoders) (0.14.4)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.1
1/dist-packages (from pandas>=1.0.5->category_encoders) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-pack
ages (from pandas>=1.0.5->category_encoders) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-pa
ckages (from pandas>=1.0.5->category_encoders) (2025.2)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/dist-pac
kages (from scikit-learn>=1.6.0->category_encoders) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/d
ist-packages (from scikit-learn>=1.6.0->category_encoders) (3.6.0)
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.11/dist-p
ackages (from statsmodels>=0.9.0->category_encoders) (24.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages
(from python-dateutil>=2.8.2->pandas>=1.0.5->category_encoders) (1.17.0)
Downloading category_encoders-2.8.1-py3-none-any.whl (85 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 85.7/85.7 kB 2.2 MB/s eta 0:00:00
Installing collected packages: category_encoders
Successfully installed category_encoders-2.8.1
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.11/dist-pack
ages (1.6.1)
Requirement already satisfied: numpy>=1.19.5 in /usr/local/lib/python3.11/dist-pac
kages (from scikit-learn) (2.0.2)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.11/dist-pack
ages (from scikit-learn) (1.14.1)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/dist-pac
kages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/d
ist-packages (from scikit-learn) (3.6.0)
Collecting pycountry
  Downloading pycountry-24.6.1-py3-none-any.whl.metadata (12 kB)
Downloading pycountry-24.6.1-py3-none-any.whl (6.3 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 6.3/6.3 MB 25.2 MB/s eta 0:00:00
Installing collected packages: pycountry
Successfully installed pycountry-24.6.1
```

In [2]:

```python
import numpy as np
import pandas as pd
import pickle
import re
import statsmodels.api as sm
from scipy.stats import ttest_ind, ttest_rel, chi2, chi2_contingency, chisquare, \
f_oneway, spearmanr, pearsonr, norm, shapiro, kstest
from statsmodels.api import OLS
import matplotlib.pyplot as plt
import seaborn as sns
```

```
from google.colab import drive
from statsmodels.graphics.gofplots import qqplot
from sklearn.preprocessing import MinMaxScaler, StandardScaler
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
from statsmodels.compat import lzip
import statsmodels.stats.api as sms
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.model_selection import train_test_split
from colorama import Fore, Back, Style
from sklearn.impute import SimpleImputer
import category_encoders as ce
from sklearn.preprocessing import OrdinalEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report, roc_curve, roc_auc_score, \
confusion_matrix, ConfusionMatrixDisplay, precision_recall_curve, accuracy_score
import warnings
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.model_selection import KFold, cross_validate
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
from imblearn.over_sampling import SMOTE
from IPython.display import Image
from six import StringIO
from sklearn.tree import export_graphviz
import pydot
import calendar
import datetime as dt
from scipy.stats import uniform
import gc
```

In [3]:
```
warnings.filterwarnings("ignore", category=UserWarning)
drive.mount('/content/drive', force_remount=True)
df = pd.read_csv('/content/drive/MyDrive/Scaler_DSML_Digital_Notes/BusinessCases/11
pd.set_option('display.max_columns', None)
pd.options.display.float_format = '{:,.0f}'.format
# df_copy = df.copy()
df.head(5)
```

```
Mounted at /content/drive
```

Out[3]:

	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
0	2NE1_zh.wikipedia.org_all-access_spider	18	11	5	13	14	9	9	22
1	2PM_zh.wikipedia.org_all-access_spider	11	14	15	18	11	13	22	11
2	3C_zh.wikipedia.org_all-access_spider	1	0	1	1	0	4	0	3
3	4minute_zh.wikipedia.org_all-access_spider	35	13	10	94	4	26	14	9
4	52_Hz_I_Love_You_zh.wikipedia.org_all-access_s...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

EDA

In [ ]:
```
df.shape
```

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

In [ ]:
```
df.info(verbose=True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 145063 entries, 0 to 145062
Data columns (total 551 columns):
 #    Column      Dtype
---   ------      -----
 0    Page        object
 1    2015-07-01  float64
 2    2015-07-02  float64
 3    2015-07-03  float64
 4    2015-07-04  float64
 5    2015-07-05  float64
 6    2015-07-06  float64
 7    2015-07-07  float64
 8    2015-07-08  float64
 9    2015-07-09  float64
 10   2015-07-10  float64
 11   2015-07-11  float64
 12   2015-07-12  float64
 13   2015-07-13  float64
 14   2015-07-14  float64
 15   2015-07-15  float64
 16   2015-07-16  float64
 17   2015-07-17  float64
 18   2015-07-18  float64
 19   2015-07-19  float64
 20   2015-07-20  float64
 21   2015-07-21  float64
 22   2015-07-22  float64
 23   2015-07-23  float64
 24   2015-07-24  float64
 25   2015-07-25  float64
 26   2015-07-26  float64
 27   2015-07-27  float64
 28   2015-07-28  float64
 29   2015-07-29  float64
 30   2015-07-30  float64
 31   2015-07-31  float64
 32   2015-08-01  float64
 33   2015-08-02  float64
 34   2015-08-03  float64
 35   2015-08-04  float64
 36   2015-08-05  float64
 37   2015-08-06  float64
 38   2015-08-07  float64
 39   2015-08-08  float64
 40   2015-08-09  float64
 41   2015-08-10  float64
 42   2015-08-11  float64
 43   2015-08-12  float64
 44   2015-08-13  float64
 45   2015-08-14  float64
 46   2015-08-15  float64
 47   2015-08-16  float64
 48   2015-08-17  float64
 49   2015-08-18  float64
 50   2015-08-19  float64
 51   2015-08-20  float64
 52   2015-08-21  float64
 53   2015-08-22  float64
 54   2015-08-23  float64
 55   2015-08-24  float64
 56   2015-08-25  float64
 57   2015-08-26  float64
 58   2015-08-27  float64
```

```
59   2015-08-28  float64
60   2015-08-29  float64
61   2015-08-30  float64
62   2015-08-31  float64
63   2015-09-01  float64
64   2015-09-02  float64
65   2015-09-03  float64
66   2015-09-04  float64
67   2015-09-05  float64
68   2015-09-06  float64
69   2015-09-07  float64
70   2015-09-08  float64
71   2015-09-09  float64
72   2015-09-10  float64
73   2015-09-11  float64
74   2015-09-12  float64
75   2015-09-13  float64
76   2015-09-14  float64
77   2015-09-15  float64
78   2015-09-16  float64
79   2015-09-17  float64
80   2015-09-18  float64
81   2015-09-19  float64
82   2015-09-20  float64
83   2015-09-21  float64
84   2015-09-22  float64
85   2015-09-23  float64
86   2015-09-24  float64
87   2015-09-25  float64
88   2015-09-26  float64
89   2015-09-27  float64
90   2015-09-28  float64
91   2015-09-29  float64
92   2015-09-30  float64
93   2015-10-01  float64
94   2015-10-02  float64
95   2015-10-03  float64
96   2015-10-04  float64
97   2015-10-05  float64
98   2015-10-06  float64
99   2015-10-07  float64
100  2015-10-08  float64
101  2015-10-09  float64
102  2015-10-10  float64
103  2015-10-11  float64
104  2015-10-12  float64
105  2015-10-13  float64
106  2015-10-14  float64
107  2015-10-15  float64
108  2015-10-16  float64
109  2015-10-17  float64
110  2015-10-18  float64
111  2015-10-19  float64
112  2015-10-20  float64
113  2015-10-21  float64
114  2015-10-22  float64
115  2015-10-23  float64
116  2015-10-24  float64
117  2015-10-25  float64
118  2015-10-26  float64
119  2015-10-27  float64
120  2015-10-28  float64
121  2015-10-29  float64
122  2015-10-30  float64
```

```
123  2015-10-31  float64
124  2015-11-01  float64
125  2015-11-02  float64
126  2015-11-03  float64
127  2015-11-04  float64
128  2015-11-05  float64
129  2015-11-06  float64
130  2015-11-07  float64
131  2015-11-08  float64
132  2015-11-09  float64
133  2015-11-10  float64
134  2015-11-11  float64
135  2015-11-12  float64
136  2015-11-13  float64
137  2015-11-14  float64
138  2015-11-15  float64
139  2015-11-16  float64
140  2015-11-17  float64
141  2015-11-18  float64
142  2015-11-19  float64
143  2015-11-20  float64
144  2015-11-21  float64
145  2015-11-22  float64
146  2015-11-23  float64
147  2015-11-24  float64
148  2015-11-25  float64
149  2015-11-26  float64
150  2015-11-27  float64
151  2015-11-28  float64
152  2015-11-29  float64
153  2015-11-30  float64
154  2015-12-01  float64
155  2015-12-02  float64
156  2015-12-03  float64
157  2015-12-04  float64
158  2015-12-05  float64
159  2015-12-06  float64
160  2015-12-07  float64
161  2015-12-08  float64
162  2015-12-09  float64
163  2015-12-10  float64
164  2015-12-11  float64
165  2015-12-12  float64
166  2015-12-13  float64
167  2015-12-14  float64
168  2015-12-15  float64
169  2015-12-16  float64
170  2015-12-17  float64
171  2015-12-18  float64
172  2015-12-19  float64
173  2015-12-20  float64
174  2015-12-21  float64
175  2015-12-22  float64
176  2015-12-23  float64
177  2015-12-24  float64
178  2015-12-25  float64
179  2015-12-26  float64
180  2015-12-27  float64
181  2015-12-28  float64
182  2015-12-29  float64
183  2015-12-30  float64
184  2015-12-31  float64
185  2016-01-01  float64
186  2016-01-02  float64
```

```
187  2016-01-03  float64
188  2016-01-04  float64
189  2016-01-05  float64
190  2016-01-06  float64
191  2016-01-07  float64
192  2016-01-08  float64
193  2016-01-09  float64
194  2016-01-10  float64
195  2016-01-11  float64
196  2016-01-12  float64
197  2016-01-13  float64
198  2016-01-14  float64
199  2016-01-15  float64
200  2016-01-16  float64
201  2016-01-17  float64
202  2016-01-18  float64
203  2016-01-19  float64
204  2016-01-20  float64
205  2016-01-21  float64
206  2016-01-22  float64
207  2016-01-23  float64
208  2016-01-24  float64
209  2016-01-25  float64
210  2016-01-26  float64
211  2016-01-27  float64
212  2016-01-28  float64
213  2016-01-29  float64
214  2016-01-30  float64
215  2016-01-31  float64
216  2016-02-01  float64
217  2016-02-02  float64
218  2016-02-03  float64
219  2016-02-04  float64
220  2016-02-05  float64
221  2016-02-06  float64
222  2016-02-07  float64
223  2016-02-08  float64
224  2016-02-09  float64
225  2016-02-10  float64
226  2016-02-11  float64
227  2016-02-12  float64
228  2016-02-13  float64
229  2016-02-14  float64
230  2016-02-15  float64
231  2016-02-16  float64
232  2016-02-17  float64
233  2016-02-18  float64
234  2016-02-19  float64
235  2016-02-20  float64
236  2016-02-21  float64
237  2016-02-22  float64
238  2016-02-23  float64
239  2016-02-24  float64
240  2016-02-25  float64
241  2016-02-26  float64
242  2016-02-27  float64
243  2016-02-28  float64
244  2016-02-29  float64
245  2016-03-01  float64
246  2016-03-02  float64
247  2016-03-03  float64
248  2016-03-04  float64
249  2016-03-05  float64
250  2016-03-06  float64
```

```
251  2016-03-07  float64
252  2016-03-08  float64
253  2016-03-09  float64
254  2016-03-10  float64
255  2016-03-11  float64
256  2016-03-12  float64
257  2016-03-13  float64
258  2016-03-14  float64
259  2016-03-15  float64
260  2016-03-16  float64
261  2016-03-17  float64
262  2016-03-18  float64
263  2016-03-19  float64
264  2016-03-20  float64
265  2016-03-21  float64
266  2016-03-22  float64
267  2016-03-23  float64
268  2016-03-24  float64
269  2016-03-25  float64
270  2016-03-26  float64
271  2016-03-27  float64
272  2016-03-28  float64
273  2016-03-29  float64
274  2016-03-30  float64
275  2016-03-31  float64
276  2016-04-01  float64
277  2016-04-02  float64
278  2016-04-03  float64
279  2016-04-04  float64
280  2016-04-05  float64
281  2016-04-06  float64
282  2016-04-07  float64
283  2016-04-08  float64
284  2016-04-09  float64
285  2016-04-10  float64
286  2016-04-11  float64
287  2016-04-12  float64
288  2016-04-13  float64
289  2016-04-14  float64
290  2016-04-15  float64
291  2016-04-16  float64
292  2016-04-17  float64
293  2016-04-18  float64
294  2016-04-19  float64
295  2016-04-20  float64
296  2016-04-21  float64
297  2016-04-22  float64
298  2016-04-23  float64
299  2016-04-24  float64
300  2016-04-25  float64
301  2016-04-26  float64
302  2016-04-27  float64
303  2016-04-28  float64
304  2016-04-29  float64
305  2016-04-30  float64
306  2016-05-01  float64
307  2016-05-02  float64
308  2016-05-03  float64
309  2016-05-04  float64
310  2016-05-05  float64
311  2016-05-06  float64
312  2016-05-07  float64
313  2016-05-08  float64
314  2016-05-09  float64
```

```
315  2016-05-10  float64
316  2016-05-11  float64
317  2016-05-12  float64
318  2016-05-13  float64
319  2016-05-14  float64
320  2016-05-15  float64
321  2016-05-16  float64
322  2016-05-17  float64
323  2016-05-18  float64
324  2016-05-19  float64
325  2016-05-20  float64
326  2016-05-21  float64
327  2016-05-22  float64
328  2016-05-23  float64
329  2016-05-24  float64
330  2016-05-25  float64
331  2016-05-26  float64
332  2016-05-27  float64
333  2016-05-28  float64
334  2016-05-29  float64
335  2016-05-30  float64
336  2016-05-31  float64
337  2016-06-01  float64
338  2016-06-02  float64
339  2016-06-03  float64
340  2016-06-04  float64
341  2016-06-05  float64
342  2016-06-06  float64
343  2016-06-07  float64
344  2016-06-08  float64
345  2016-06-09  float64
346  2016-06-10  float64
347  2016-06-11  float64
348  2016-06-12  float64
349  2016-06-13  float64
350  2016-06-14  float64
351  2016-06-15  float64
352  2016-06-16  float64
353  2016-06-17  float64
354  2016-06-18  float64
355  2016-06-19  float64
356  2016-06-20  float64
357  2016-06-21  float64
358  2016-06-22  float64
359  2016-06-23  float64
360  2016-06-24  float64
361  2016-06-25  float64
362  2016-06-26  float64
363  2016-06-27  float64
364  2016-06-28  float64
365  2016-06-29  float64
366  2016-06-30  float64
367  2016-07-01  float64
368  2016-07-02  float64
369  2016-07-03  float64
370  2016-07-04  float64
371  2016-07-05  float64
372  2016-07-06  float64
373  2016-07-07  float64
374  2016-07-08  float64
375  2016-07-09  float64
376  2016-07-10  float64
377  2016-07-11  float64
378  2016-07-12  float64
```

```
379  2016-07-13  float64
380  2016-07-14  float64
381  2016-07-15  float64
382  2016-07-16  float64
383  2016-07-17  float64
384  2016-07-18  float64
385  2016-07-19  float64
386  2016-07-20  float64
387  2016-07-21  float64
388  2016-07-22  float64
389  2016-07-23  float64
390  2016-07-24  float64
391  2016-07-25  float64
392  2016-07-26  float64
393  2016-07-27  float64
394  2016-07-28  float64
395  2016-07-29  float64
396  2016-07-30  float64
397  2016-07-31  float64
398  2016-08-01  float64
399  2016-08-02  float64
400  2016-08-03  float64
401  2016-08-04  float64
402  2016-08-05  float64
403  2016-08-06  float64
404  2016-08-07  float64
405  2016-08-08  float64
406  2016-08-09  float64
407  2016-08-10  float64
408  2016-08-11  float64
409  2016-08-12  float64
410  2016-08-13  float64
411  2016-08-14  float64
412  2016-08-15  float64
413  2016-08-16  float64
414  2016-08-17  float64
415  2016-08-18  float64
416  2016-08-19  float64
417  2016-08-20  float64
418  2016-08-21  float64
419  2016-08-22  float64
420  2016-08-23  float64
421  2016-08-24  float64
422  2016-08-25  float64
423  2016-08-26  float64
424  2016-08-27  float64
425  2016-08-28  float64
426  2016-08-29  float64
427  2016-08-30  float64
428  2016-08-31  float64
429  2016-09-01  float64
430  2016-09-02  float64
431  2016-09-03  float64
432  2016-09-04  float64
433  2016-09-05  float64
434  2016-09-06  float64
435  2016-09-07  float64
436  2016-09-08  float64
437  2016-09-09  float64
438  2016-09-10  float64
439  2016-09-11  float64
440  2016-09-12  float64
441  2016-09-13  float64
442  2016-09-14  float64
```

```
443  2016-09-15  float64
444  2016-09-16  float64
445  2016-09-17  float64
446  2016-09-18  float64
447  2016-09-19  float64
448  2016-09-20  float64
449  2016-09-21  float64
450  2016-09-22  float64
451  2016-09-23  float64
452  2016-09-24  float64
453  2016-09-25  float64
454  2016-09-26  float64
455  2016-09-27  float64
456  2016-09-28  float64
457  2016-09-29  float64
458  2016-09-30  float64
459  2016-10-01  float64
460  2016-10-02  float64
461  2016-10-03  float64
462  2016-10-04  float64
463  2016-10-05  float64
464  2016-10-06  float64
465  2016-10-07  float64
466  2016-10-08  float64
467  2016-10-09  float64
468  2016-10-10  float64
469  2016-10-11  float64
470  2016-10-12  float64
471  2016-10-13  float64
472  2016-10-14  float64
473  2016-10-15  float64
474  2016-10-16  float64
475  2016-10-17  float64
476  2016-10-18  float64
477  2016-10-19  float64
478  2016-10-20  float64
479  2016-10-21  float64
480  2016-10-22  float64
481  2016-10-23  float64
482  2016-10-24  float64
483  2016-10-25  float64
484  2016-10-26  float64
485  2016-10-27  float64
486  2016-10-28  float64
487  2016-10-29  float64
488  2016-10-30  float64
489  2016-10-31  float64
490  2016-11-01  float64
491  2016-11-02  float64
492  2016-11-03  float64
493  2016-11-04  float64
494  2016-11-05  float64
495  2016-11-06  float64
496  2016-11-07  float64
497  2016-11-08  float64
498  2016-11-09  float64
499  2016-11-10  float64
500  2016-11-11  float64
501  2016-11-12  float64
502  2016-11-13  float64
503  2016-11-14  float64
504  2016-11-15  float64
505  2016-11-16  float64
506  2016-11-17  float64
```

```
 507  2016-11-18  float64
 508  2016-11-19  float64
 509  2016-11-20  float64
 510  2016-11-21  float64
 511  2016-11-22  float64
 512  2016-11-23  float64
 513  2016-11-24  float64
 514  2016-11-25  float64
 515  2016-11-26  float64
 516  2016-11-27  float64
 517  2016-11-28  float64
 518  2016-11-29  float64
 519  2016-11-30  float64
 520  2016-12-01  float64
 521  2016-12-02  float64
 522  2016-12-03  float64
 523  2016-12-04  float64
 524  2016-12-05  float64
 525  2016-12-06  float64
 526  2016-12-07  float64
 527  2016-12-08  float64
 528  2016-12-09  float64
 529  2016-12-10  float64
 530  2016-12-11  float64
 531  2016-12-12  float64
 532  2016-12-13  float64
 533  2016-12-14  float64
 534  2016-12-15  float64
 535  2016-12-16  float64
 536  2016-12-17  float64
 537  2016-12-18  float64
 538  2016-12-19  float64
 539  2016-12-20  float64
 540  2016-12-21  float64
 541  2016-12-22  float64
 542  2016-12-23  float64
 543  2016-12-24  float64
 544  2016-12-25  float64
 545  2016-12-26  float64
 546  2016-12-27  float64
 547  2016-12-28  float64
 548  2016-12-29  float64
 549  2016-12-30  float64
 550  2016-12-31  float64
dtypes: float64(550), object(1)
memory usage: 609.8+ MB
```

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

Out[ ]:

	2015-07-01	2015-07-02	2015-07-03	2015-07-04	2015-07-05	2015-07-06	2015-07-07	2015-07-08
count	124,323	124,247	124,519	124,409	124,404	124,580	124,399	124,769
mean	1,196	1,204	1,134	1,170	1,218	1,290	1,239	1,193
std	72,754	74,215	69,610	72,574	73,796	80,544	75,763	68,200
min	0	0	0	0	0	0	0	0
25%	13	13	12	13	14	11	13	13
50%	109	108	105	105	113	113	115	117
75%	524	519	504	487	540	555	551	554
max	20,381,245	20,752,194	19,573,967	20,439,645	20,772,109	22,544,669	21,210,887	19,107,911

In [ ]:
```
# SPECIFICNAME_LANGUAGE.wikipedia.org_ACCESSTYPE_ACCESSORIGIN
# 2NE1_zh.wikipedia.org_all-access_spider

# pattern = r'(?P<specific_name>\S+)_(?P<language>\S+)\.wikipedia\.org_(?P<access_t
pattern = '(?P<specific_name>\S+)_(?P<language>\S+)\.wikipedia\.org_(?P<access_type
```

In [ ]:
```
import regex
regex = regex.compile(pattern)

def split_page_fields(string):
  match_obj = regex.match(string)
  match_list = []
  if match_obj:
    match_list.append(match_obj.group('specific_name') if match_obj.group('specific
    match_list.append(match_obj.group('language') if match_obj.group('language') el
    match_list.append(match_obj.group('access_type') if match_obj.group('access_typ
    match_list.append(match_obj.group('access_origin') if match_obj.group('access_o
    return match_list
  else:
    return np.full(4,np.nan)
```

In [ ]:
```
split_page_fields('2NE1_zh.wikipedia.org_all-access_spider')
```

Out[ ]:
```
['2NE1', 'zh', 'all-access', 'spider']
```

In [ ]:
```
df[['Specific_Name', 'Language','Access_Type', 'Access_Origin']] = df['Page'].apply
```

In [ ]:
```
df.columns
```

Out[ ]:
```
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',
       ...
       '2016-12-26', '2016-12-27', '2016-12-28', '2016-12-29', '2016-12-30',
       '2016-12-31', 'Specific_Name', 'Language', 'Access_Type',
       'Access_Origin'],
      dtype='object', length=555)
```

In [ ]:
```
df.isna().sum(axis=1)
```

Out[ ]:

	0
0	0
1	0
2	0
3	0
4	291
...	...
145058	544
145059	550
145060	550
145061	550
145062	550

145063 rows × 1 columns

dtype: int64

In [ ]:
```
null_rows = df[df.iloc[:,1:].isna().all(axis=1) == True].index
print('Total Null values', len(null_rows))
df = df.drop(null_rows)
df1 =df.copy()
df1.head(5)
```

```
Total Null values 127
```

Out[ ]:

	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
0	2NE1_zh.wikipedia.org_all-access_spider	18	11	5	13	14	9	9	22
1	2PM_zh.wikipedia.org_all-access_spider	11	14	15	18	11	13	22	11
2	3C_zh.wikipedia.org_all-access_spider	1	0	1	1	0	4	0	3
3	4minute_zh.wikipedia.org_all-access_spider	35	13	10	94	4	26	14	9
4	52_Hz_I_Love_You_zh.wikipedia.org_all-access_s...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

In [ ]:
```
for i in ['Language', 'Access_Type', 'Access_Origin']:
  print(f'{i} - {df[i].nunique()}  {list(df[i].dropna().unique())}')
```

```
Language - 7  ['zh', 'fr', 'en', 'ru', 'de', 'ja', 'es']
Access_Type - 3  ['all-access', 'desktop', 'mobile-web']
Access_Origin - 2  ['spider', 'all-agents']
```

In [ ]:
```
df['Language'].value_counts()
```

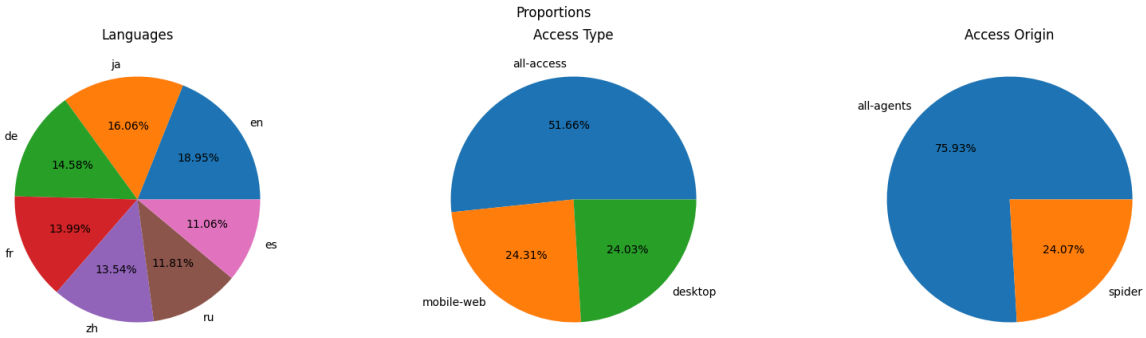
Out[ ]:

Language	count
en	24108
ja	20431
de	18547
fr	17802
zh	17229
ru	15022
es	14069

dtype: int64

In [ ]:
```
import pycountry

def get_language_name(code):
    try:
        return pycountry.languages.get(alpha_2=code).name
    except:
        return code  # return original if not found

df['Language_Full'] = df['Language'].apply(get_language_name)
```

In [ ]:
```
from IPython.display import display, HTML
plt.figure(figsize=(20,5))
plt.suptitle('Proportions')
plt.subplot(1,3,1)
plt.title('Languages')
plt.pie(df['Language'].value_counts(), autopct='%.2f%%', labels=df['Language'].valu
plt.subplot(1,3,2)
plt.title('Access Type')
plt.pie(df['Access_Type'].value_counts(), autopct='%.2f%%', labels=df['Access_Type'
plt.subplot(1,3,3)
plt.title('Access Origin')
plt.pie(df['Access_Origin'].value_counts(), autopct='%.2f%%', labels=df['Access_Ori
plt.show()

html = f"""
<div style="display: flex; gap: 40px; justify-content: space-around;">
    <div>
        <h4>Language</h4>
        {df['Language'].value_counts().to_frame().to_html()}
    </div>
    <div>
        <h4>Access Type</h4>
        {df['Access_Type'].value_counts().to_frame().to_html()}
    </div>
    <div>
        <h4>Access Origin</h4>
        {df['Access_Origin'].value_counts().to_frame().to_html()}
    </div>
</div>
"""

display(HTML(html))
```



Language

Language	count
en	24108
ja	20431
de	18547
fr	17802
zh	17229
ru	15022
es	14069

Access Type

Access_Type	count
all-access	65713
mobile-web	30923
desktop	30572

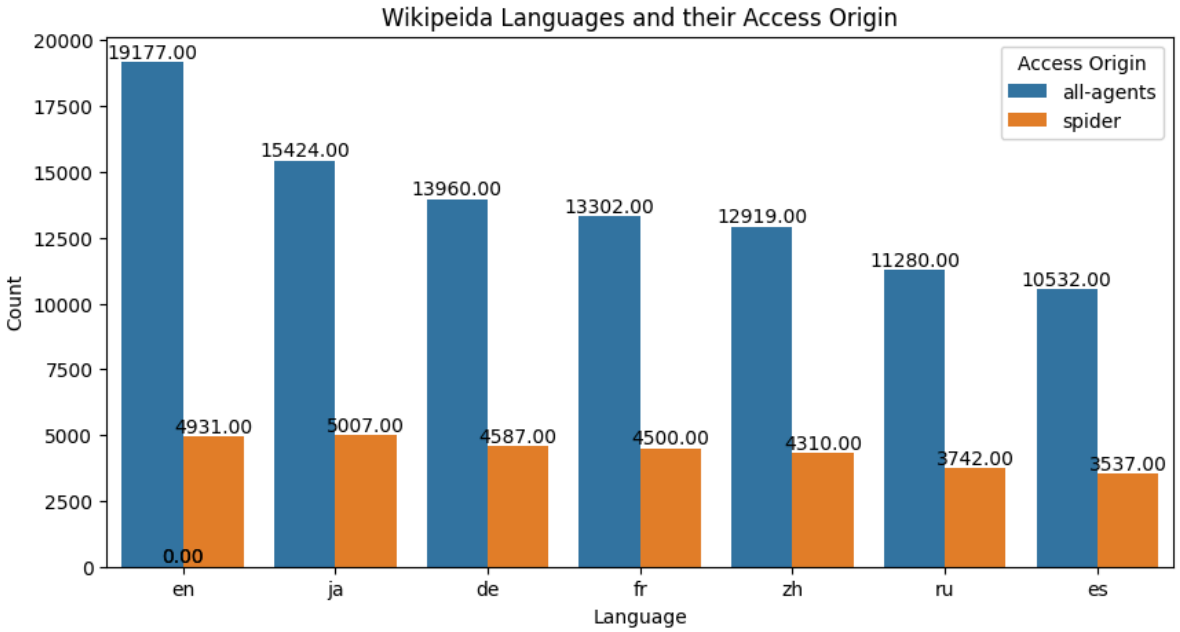
Access Origin

Access_Origin	count
all-agents	96594
spider	30614

In [ ]:
```
def annotate(ax, rotation = False):
   for patch in ax.patches:  # Loop through each bar
        if rotation:  # For horizontal bars
            x = patch.get_width()  # Get the width (value of the bar)
            y = patch.get_y() + patch.get_height() / 2  # Center the annotation ver
            ax.annotate(f"{x:.2f}", (x + 0.5, y), ha='left', va='center')  # Adjust
        else:  # For vertical bars
            x = patch.get_x() + patch.get_width() / 2  # Center the annotation hori
            y = patch.get_height()  # Get the height (value of the bar)
            ax.annotate(f"{y:.2f}", (x, y + 0.5), ha='center', va='bottom')  # Adju
```

In [ ]:
```
lag_acc_orn = df[['Language','Access_Origin']]
lag_acc_orn_count = lag_acc_orn.value_counts().reset_index()
display(lag_acc_orn_count)
plt.figure(figsize=(10, 5))
ax = sns.barplot(x='Language', y='count', hue='Access_Origin', data=lag_acc_orn_cou
plt.title('Wikipeida Languages and their Access Origin')
plt.legend(title='Access Origin')
plt.xlabel('Language')
plt.ylabel('Count')
annotate(ax)
plt.show()

del lag_acc_orn_count, lag_acc_orn
gc.collect()
```

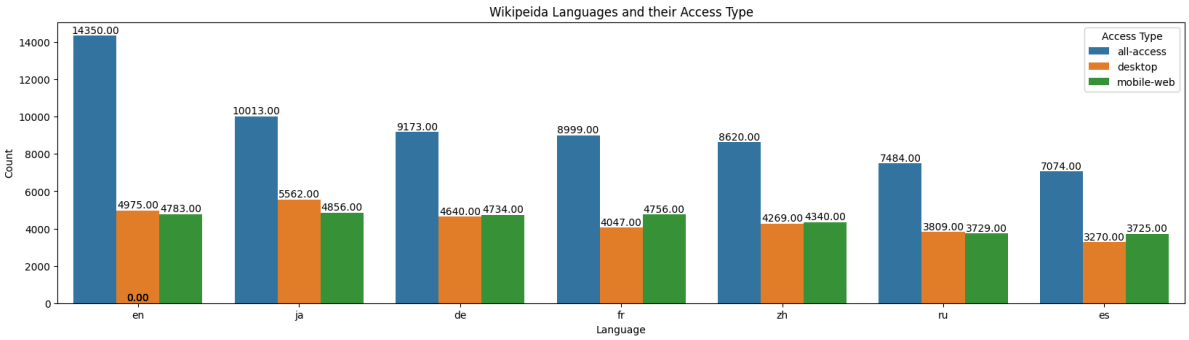
	Language	Access_Origin	count
0	en	all-agents	19177
1	ja	all-agents	15424
2	de	all-agents	13960
3	fr	all-agents	13302
4	zh	all-agents	12919
5	ru	all-agents	11280
6	es	all-agents	10532
7	ja	spider	5007
8	en	spider	4931
9	de	spider	4587
10	fr	spider	4500
11	zh	spider	4310
12	ru	spider	3742
13	es	spider	3537



Out[ ]: 6589

In [ ]:
```
lag_acc_type = df[['Language','Access_Type']]
lag_acc_type_count = lag_acc_type.value_counts().reset_index()
display(lag_acc_type_count)
plt.figure(figsize=(20, 5))
ax = sns.barplot(x='Language', y='count', hue='Access_Type', data=lag_acc_type_cour
plt.title('Wikipeida Languages and their Access Type')
plt.legend(title='Access Type')
plt.xlabel('Language')
plt.ylabel('Count')
annotate(ax)
plt.show()

del  lag_acc_type, lag_acc_type_count
gc.collect()
```

	Language	Access_Type	count
0	en	all-access	14350
1	ja	all-access	10013
2	de	all-access	9173
3	fr	all-access	8999
4	zh	all-access	8620
5	ru	all-access	7484
6	es	all-access	7074
7	ja	desktop	5562
8	en	desktop	4975
9	ja	mobile-web	4856
10	en	mobile-web	4783
11	fr	mobile-web	4756
12	de	mobile-web	4734
13	de	desktop	4640
14	zh	mobile-web	4340
15	zh	desktop	4269
16	fr	desktop	4047
17	ru	desktop	3809
18	ru	mobile-web	3729
19	es	mobile-web	3725
20	es	desktop	3270



Out[ ]:
```
5305
```

In [ ]:
```
column_names = df.columns[~df.columns.isin(['Page','Specific_Name', 'Language', 'Ac
column_names = column_names.insert(0, 'Language_Full')
column_names
```

Out[ ]:
```
Index(['Language_Full', '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)
```

In [ ]:
```
df['Language_Full'] = df['Language_Full'].fillna('Unknown')
df_zero_nan_fill = df[column_names].fillna(0)
```

In [ ]:
```
df_lang_click = pd.DataFrame({'Language':df_zero_nan_fill.Language_Full.to_list() ,
df_lang_click
```

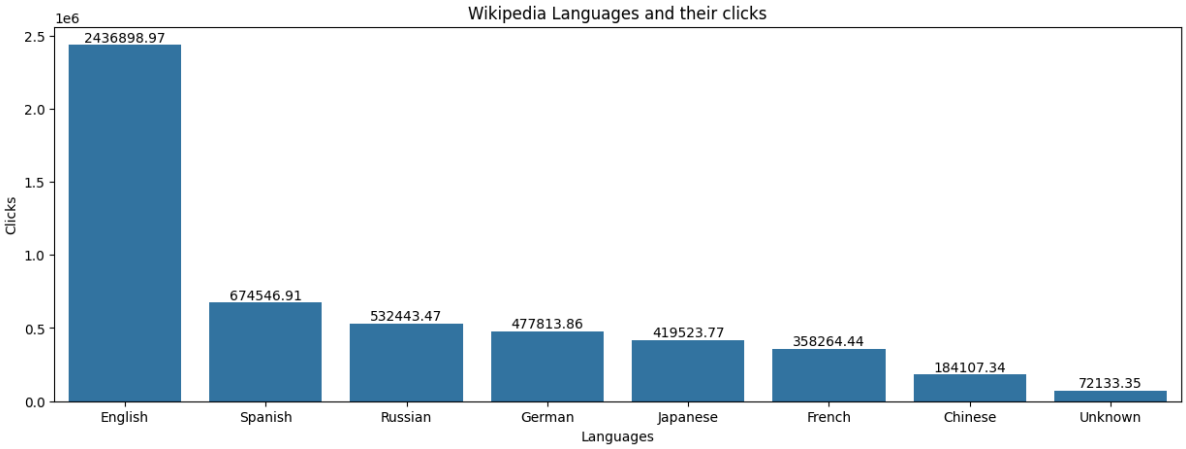
Out[ ]:

	Language	Clicks
0	Chinese	11,966
1	Chinese	13,966
2	Chinese	2,862
3	Chinese	9,419
4	Chinese	2,662
...	...	...
144931	Spanish	56
144932	Spanish	0
144933	Spanish	0
144934	Spanish	0
144935	Spanish	0

144936 rows × 2 columns

In [ ]:
```
agg_df_lang_click = df_lang_click.groupby(by='Language')['Clicks'].agg('sum')
sorted_agg_lang_click_df = agg_df_lang_click.reset_index().sort_values(by='Clicks',
sorted_agg_lang_click_df.reset_index(inplace=True, drop=True)

display(sorted_agg_lang_click_df)
plt.figure(figsize=(15, 5))
ax = sns.barplot(data=sorted_agg_lang_click_df.loc[:, :], x='Language', y='Clicks')
plt.title('Wikipedia Languages and their clicks')
plt.xlabel('Languages')
plt.ylabel('Clicks')
annotate(ax)
plt.show()

del agg_df_lang_click, df_lang_click, sorted_agg_lang_click_df
gc.collect()
```

	Language	Clicks
0	English	2,436,899
1	Spanish	674,547
2	Russian	532,443
3	German	477,814
4	Japanese	419,524
5	French	358,264
6	Chinese	184,107
7	Unknown	72,133



Out[ ]: 6405

In [ ]:
```
#Separating the pages datasets based on the Languages
df.head()
```

Out[ ]:

	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
0	2NE1_zh.wikipedia.org_all-access_spider	18	11	5	13	14	9	9	22
1	2PM_zh.wikipedia.org_all-access_spider	11	14	15	18	11	13	22	11
2	3C_zh.wikipedia.org_all-access_spider	1	0	1	1	0	4	0	3
3	4minute_zh.wikipedia.org_all-access_spider	35	13	10	94	4	26	14	9
4	52_Hz_I_Love_You_zh.wikipedia.org_all-access_s...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

In [ ]:
```
df[df['Language_Full'] == 'Unknown'].loc[:, 'Page'].head(10)
```

Out[ ]:

	Page
13332	Accueil_commons.wikimedia.org_all-access_spider
13333	Atlas_of_Asia_commons.wikimedia.org_all-access...
13334	Atlas_of_Europe_commons.wikimedia.org_all-acce...
13335	Atlas_of_World_War_II_commons.wikimedia.org_al...
13336	Atlas_of_colonialism_commons.wikimedia.org_all...
13337	Atlas_of_the_United_Kingdom_commons.wikimedia....
13338	Atlas_of_the_United_States_commons.wikimedia.o...
13339	Bikini_commons.wikimedia.org_all-access_spider
13340	Campaign:OFBA2016_commons.wikimedia.org_all-ac...
13341	Catalogue_of_Wilhelm_von_Gloeden's_pictures_co...

dtype: object

In [ ]:

```
string_columns = ['Language_Full', 'Specific_Name', 'Access_Type', 'Access_Origin']

date_columns = df.columns[1:-5]

print('String Columns :', string_columns, '\n')
print('Date Columns :', date_columns, '\n')

columns = []

columns.extend(string_columns)
columns.extend(date_columns)
print('Columns :', columns)
```

```
String Columns : ['Language_Full', 'Specific_Name', 'Access_Type', 'Access_Origi
n']

Date Columns : Index(['2015-07-01', '2015-07-02', '2015-07-03', '2015-07-04', '201
5-07-05',
       '2015-07-06', '2015-07-07', '2015-07-08', '2015-07-09', '2015-07-10',
       ...
       '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=550)

Columns : ['Language_Full', 'Specific_Name', 'Access_Type', 'Access_Origin', '2015
-07-01', '2015-07-02', '2015-07-03', '2015-07-04', '2015-07-05', '2015-07-06', '20
15-07-07', '2015-07-08', '2015-07-09', '2015-07-10', '2015-07-11', '2015-07-12',
'2015-07-13', '2015-07-14', '2015-07-15', '2015-07-16', '2015-07-17', '2015-07-1
8', '2015-07-19', '2015-07-20', '2015-07-21', '2015-07-22', '2015-07-23', '2015-07
-24', '2015-07-25', '2015-07-26', '2015-07-27', '2015-07-28', '2015-07-29', '2015-
07-30', '2015-07-31', '2015-08-01', '2015-08-02', '2015-08-03', '2015-08-04', '201
5-08-05', '2015-08-06', '2015-08-07', '2015-08-08', '2015-08-09', '2015-08-10', '2
015-08-11', '2015-08-12', '2015-08-13', '2015-08-14', '2015-08-15', '2015-08-16',
'2015-08-17', '2015-08-18', '2015-08-19', '2015-08-20', '2015-08-21', '2015-08-2
2', '2015-08-23', '2015-08-24', '2015-08-25', '2015-08-26', '2015-08-27', '2015-08
-28', '2015-08-29', '2015-08-30', '2015-08-31', '2015-09-01', '2015-09-02', '2015-
09-03', '2015-09-04', '2015-09-05', '2015-09-06', '2015-09-07', '2015-09-08', '201
5-09-09', '2015-09-10', '2015-09-11', '2015-09-12', '2015-09-13', '2015-09-14', '2
015-09-15', '2015-09-16', '2015-09-17', '2015-09-18', '2015-09-19', '2015-09-20',
'2015-09-21', '2015-09-22', '2015-09-23', '2015-09-24', '2015-09-25', '2015-09-2
6', '2015-09-27', '2015-09-28', '2015-09-29', '2015-09-30', '2015-10-01', '2015-10
-02', '2015-10-03', '2015-10-04', '2015-10-05', '2015-10-06', '2015-10-07', '2015-
10-08', '2015-10-09', '2015-10-10', '2015-10-11', '2015-10-12', '2015-10-13', '201
5-10-14', '2015-10-15', '2015-10-16', '2015-10-17', '2015-10-18', '2015-10-19', '2
015-10-20', '2015-10-21', '2015-10-22', '2015-10-23', '2015-10-24', '2015-10-25',
'2015-10-26', '2015-10-27', '2015-10-28', '2015-10-29', '2015-10-30', '2015-10-3
1', '2015-11-01', '2015-11-02', '2015-11-03', '2015-11-04', '2015-11-05', '2015-11
-06', '2015-11-07', '2015-11-08', '2015-11-09', '2015-11-10', '2015-11-11', '2015-
11-12', '2015-11-13', '2015-11-14', '2015-11-15', '2015-11-16', '2015-11-17', '201
5-11-18', '2015-11-19', '2015-11-20', '2015-11-21', '2015-11-22', '2015-11-23', '2
015-11-24', '2015-11-25', '2015-11-26', '2015-11-27', '2015-11-28', '2015-11-29',
'2015-11-30', '2015-12-01', '2015-12-02', '2015-12-03', '2015-12-04', '2015-12-0
5', '2015-12-06', '2015-12-07', '2015-12-08', '2015-12-09', '2015-12-10', '2015-12
-11', '2015-12-12', '2015-12-13', '2015-12-14', '2015-12-15', '2015-12-16', '2015-
12-17', '2015-12-18', '2015-12-19', '2015-12-20', '2015-12-21', '2015-12-22', '201
5-12-23', '2015-12-24', '2015-12-25', '2015-12-26', '2015-12-27', '2015-12-28', '2
015-12-29', '2015-12-30', '2015-12-31', '2016-01-01', '2016-01-02', '2016-01-03',
'2016-01-04', '2016-01-05', '2016-01-06', '2016-01-07', '2016-01-08', '2016-01-0
9', '2016-01-10', '2016-01-11', '2016-01-12', '2016-01-13', '2016-01-14', '2016-01
-15', '2016-01-16', '2016-01-17', '2016-01-18', '2016-01-19', '2016-01-20', '2016-
01-21', '2016-01-22', '2016-01-23', '2016-01-24', '2016-01-25', '2016-01-26', '201
6-01-27', '2016-01-28', '2016-01-29', '2016-01-30', '2016-01-31', '2016-02-01', '2
016-02-02', '2016-02-03', '2016-02-04', '2016-02-05', '2016-02-06', '2016-02-07',
'2016-02-08', '2016-02-09', '2016-02-10', '2016-02-11', '2016-02-12', '2016-02-1
3', '2016-02-14', '2016-02-15', '2016-02-16', '2016-02-17', '2016-02-18', '2016-02
-19', '2016-02-20', '2016-02-21', '2016-02-22', '2016-02-23', '2016-02-24', '2016-
02-25', '2016-02-26', '2016-02-27', '2016-02-28', '2016-02-29', '2016-03-01', '201
6-03-02', '2016-03-03', '2016-03-04', '2016-03-05', '2016-03-06', '2016-03-07', '2
016-03-08', '2016-03-09', '2016-03-10', '2016-03-11', '2016-03-12', '2016-03-13',
'2016-03-14', '2016-03-15', '2016-03-16', '2016-03-17', '2016-03-18', '2016-03-1
9', '2016-03-20', '2016-03-21', '2016-03-22', '2016-03-23', '2016-03-24', '2016-03
-25', '2016-03-26', '2016-03-27', '2016-03-28', '2016-03-29', '2016-03-30', '2016-
03-31', '2016-04-01', '2016-04-02', '2016-04-03', '2016-04-04', '2016-04-05', '201
6-04-06', '2016-04-07', '2016-04-08', '2016-04-09', '2016-04-10', '2016-04-11', '2
016-04-12', '2016-04-13', '2016-04-14', '2016-04-15', '2016-04-16', '2016-04-17',
'2016-04-18', '2016-04-19', '2016-04-20', '2016-04-21', '2016-04-22', '2016-04-2
3', '2016-04-24', '2016-04-25', '2016-04-26', '2016-04-27', '2016-04-28', '2016-04
```

```
-29', '2016-04-30', '2016-05-01', '2016-05-02', '2016-05-03', '2016-05-04', '2016-
05-05', '2016-05-06', '2016-05-07', '2016-05-08', '2016-05-09', '2016-05-10', '201
6-05-11', '2016-05-12', '2016-05-13', '2016-05-14', '2016-05-15', '2016-05-16', '2
016-05-17', '2016-05-18', '2016-05-19', '2016-05-20', '2016-05-21', '2016-05-22',
'2016-05-23', '2016-05-24', '2016-05-25', '2016-05-26', '2016-05-27', '2016-05-2
8', '2016-05-29', '2016-05-30', '2016-05-31', '2016-06-01', '2016-06-02', '2016-06
-03', '2016-06-04', '2016-06-05', '2016-06-06', '2016-06-07', '2016-06-08', '2016-
06-09', '2016-06-10', '2016-06-11', '2016-06-12', '2016-06-13', '2016-06-14', '201
6-06-15', '2016-06-16', '2016-06-17', '2016-06-18', '2016-06-19', '2016-06-20', '2
016-06-21', '2016-06-22', '2016-06-23', '2016-06-24', '2016-06-25', '2016-06-26',
'2016-06-27', '2016-06-28', '2016-06-29', '2016-06-30', '2016-07-01', '2016-07-0
2', '2016-07-03', '2016-07-04', '2016-07-05', '2016-07-06', '2016-07-07', '2016-07
-08', '2016-07-09', '2016-07-10', '2016-07-11', '2016-07-12', '2016-07-13', '2016-
07-14', '2016-07-15', '2016-07-16', '2016-07-17', '2016-07-18', '2016-07-19', '201
6-07-20', '2016-07-21', '2016-07-22', '2016-07-23', '2016-07-24', '2016-07-25', '2
016-07-26', '2016-07-27', '2016-07-28', '2016-07-29', '2016-07-30', '2016-07-31',
'2016-08-01', '2016-08-02', '2016-08-03', '2016-08-04', '2016-08-05', '2016-08-0
6', '2016-08-07', '2016-08-08', '2016-08-09', '2016-08-10', '2016-08-11', '2016-08
-12', '2016-08-13', '2016-08-14', '2016-08-15', '2016-08-16', '2016-08-17', '2016-
08-18', '2016-08-19', '2016-08-20', '2016-08-21', '2016-08-22', '2016-08-23', '201
6-08-24', '2016-08-25', '2016-08-26', '2016-08-27', '2016-08-28', '2016-08-29', '2
016-08-30', '2016-08-31', '2016-09-01', '2016-09-02', '2016-09-03', '2016-09-04',
'2016-09-05', '2016-09-06', '2016-09-07', '2016-09-08', '2016-09-09', '2016-09-1
0', '2016-09-11', '2016-09-12', '2016-09-13', '2016-09-14', '2016-09-15', '2016-09
-16', '2016-09-17', '2016-09-18', '2016-09-19', '2016-09-20', '2016-09-21', '2016-
09-22', '2016-09-23', '2016-09-24', '2016-09-25', '2016-09-26', '2016-09-27', '201
6-09-28', '2016-09-29', '2016-09-30', '2016-10-01', '2016-10-02', '2016-10-03', '2
016-10-04', '2016-10-05', '2016-10-06', '2016-10-07', '2016-10-08', '2016-10-09',
'2016-10-10', '2016-10-11', '2016-10-12', '2016-10-13', '2016-10-14', '2016-10-1
5', '2016-10-16', '2016-10-17', '2016-10-18', '2016-10-19', '2016-10-20', '2016-10
-21', '2016-10-22', '2016-10-23', '2016-10-24', '2016-10-25', '2016-10-26', '2016-
10-27', '2016-10-28', '2016-10-29', '2016-10-30', '2016-10-31', '2016-11-01', '201
6-11-02', '2016-11-03', '2016-11-04', '2016-11-05', '2016-11-06', '2016-11-07', '2
016-11-08', '2016-11-09', '2016-11-10', '2016-11-11', '2016-11-12', '2016-11-13',
'2016-11-14', '2016-11-15', '2016-11-16', '2016-11-17', '2016-11-18', '2016-11-1
9', '2016-11-20', '2016-11-21', '2016-11-22', '2016-11-23', '2016-11-24', '2016-11
-25', '2016-11-26', '2016-11-27', '2016-11-28', '2016-11-29', '2016-11-30', '2016-
12-01', '2016-12-02', '2016-12-03', '2016-12-04', '2016-12-05', '2016-12-06', '201
6-12-07', '2016-12-08', '2016-12-09', '2016-12-10', '2016-12-11', '2016-12-12', '2
016-12-13', '2016-12-14', '2016-12-15', '2016-12-16', '2016-12-17', '2016-12-18',
'2016-12-19', '2016-12-20', '2016-12-21', '2016-12-22', '2016-12-23', '2016-12-2
4', '2016-12-25', '2016-12-26', '2016-12-27', '2016-12-28', '2016-12-29', '2016-12
-30', '2016-12-31']
```

In [ ]:
```
all_lang_df = df[columns]
all_lang_df.head()
```

Out[ ]:

	Language_Full	Specific_Name	Access_Type	Access_Origin	2015-07-01	2015-07-02	2015-07-03	2015-07-04	2015-07-0
0	Chinese	2NE1	all-access	spider	18	11	5	13	1
1	Chinese	2PM	all-access	spider	11	14	15	18	1
2	Chinese	3C	all-access	spider	1	0	1	1	
3	Chinese	4minute	all-access	spider	35	13	10	94	
4	Chinese	52_Hz_I_Love_You	all-access	spider	NaN	NaN	NaN	NaN	NaI

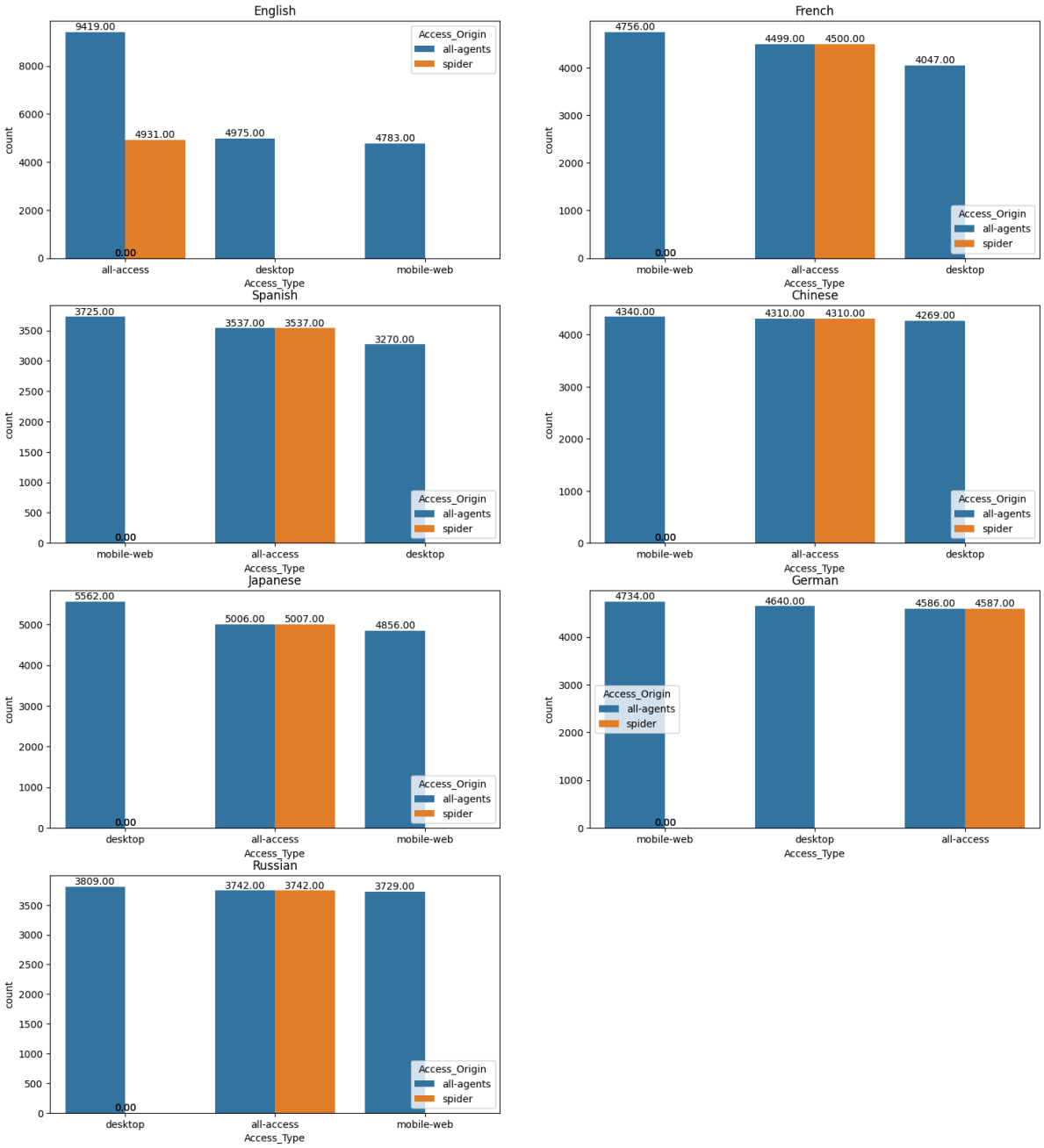
In [ ]:
```
languages_list = ['English', 'French', 'Spanish', 'Chinese', 'Japanese', 'German',

for lang in languages_list:
```

```
  with open(f'/content/drive/MyDrive/Scaler_DSML_Digital_Notes/BusinessCases/11_AdE
    pickle.dump(all_lang_df[all_lang_df['Language_Full'] == lang], fp)
```

In [ ]:
```
def get_access_type_origin_value_cpounts(df_dict):

  plot_dict = {}

  for lag, data in df_dict.items():

    acc_type_ogn_df = data[['Access_Type', 'Access_Origin']].value_counts()
    plot_dict.update({lag:acc_type_ogn_df.reset_index()})

  plt.figure(figsize=(18,20))
  plt.suptitle('Access Type and Access Origin Count of all Langauges')

  for i in range(1,8):
    plt.subplot(4,2,i)
    plt.title(list(plot_dict.keys())[0])
    ax = sns.barplot(data=plot_dict.get(list(plot_dict.keys())[0]), x='Access_Type'
    annotate(ax)
    plot_dict.pop(list(plot_dict.keys())[0])

  plt.show()
```

In [ ]:
```
all_lang_df_objs_dict = {}

for lang in languages_list:
  with open(f'/content/drive/MyDrive/Scaler_DSML_Digital_Notes/BusinessCases/11_AdE
    all_lang_df_objs_dict.update({lang:pickle.load(fp)})
get_access_type_origin_value_cpounts(all_lang_df_objs_dict)
# all_lang_df_objs_dict = {'English':english_df, 'French':french_df, 'Spanish':span
# get_access_type_origin_value_cpounts(all_lang_df_objs_dict)
```

Access Type and Access Origin Count of all Langauges



In [ ]:
```
# en_acc_type_ogn_df = english_df[['Access_Type', 'Access_Origin']].value_counts()
# en_acc_type_ogn_df = en_acc_type_ogn_df.reset_index()
# display(en_acc_type_ogn_df)
# plt.figure(figsize=(10,5))
# ax = sns.barplot(data=en_acc_type_ogn_df, x='Access_Type', y='count', hue='Access
# annotate(ax)
# plt.title('Wikipidea pages of English Language with theit Access Type and Access
# plt.show()
```

In [ ]:
```
for k, v in all_lang_df_objs_dict.items():
  print(f'{k} Langauge data frame has {len(v[v.duplicated()])} duplicate values')
```

```
English Langauge data frame has 0 duplicate values
French Langauge data frame has 0 duplicate values
Spanish Langauge data frame has 0 duplicate values
Chinese Langauge data frame has 0 duplicate values
Japanese Langauge data frame has 0 duplicate values
German Langauge data frame has 0 duplicate values
Russian Langauge data frame has 0 duplicate values
```

In [ ]:
```
english_dates_df = all_lang_df_objs_dict['English'].drop(columns = string_columns)
english_dates_df.head()
```

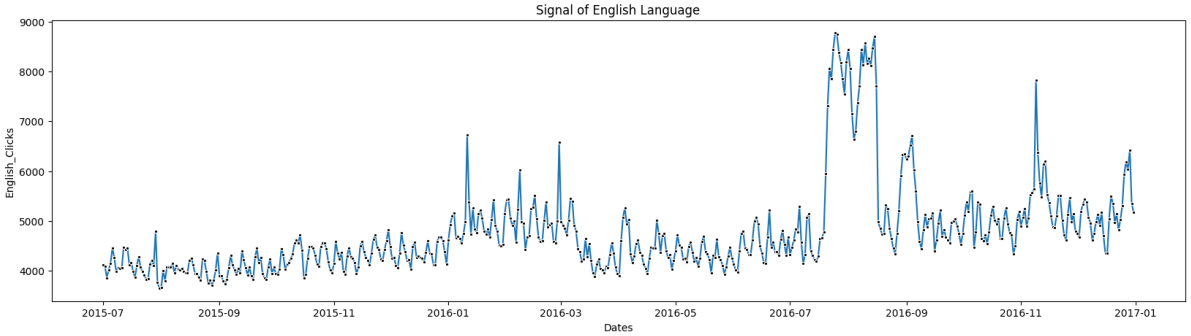
Out[ ]:

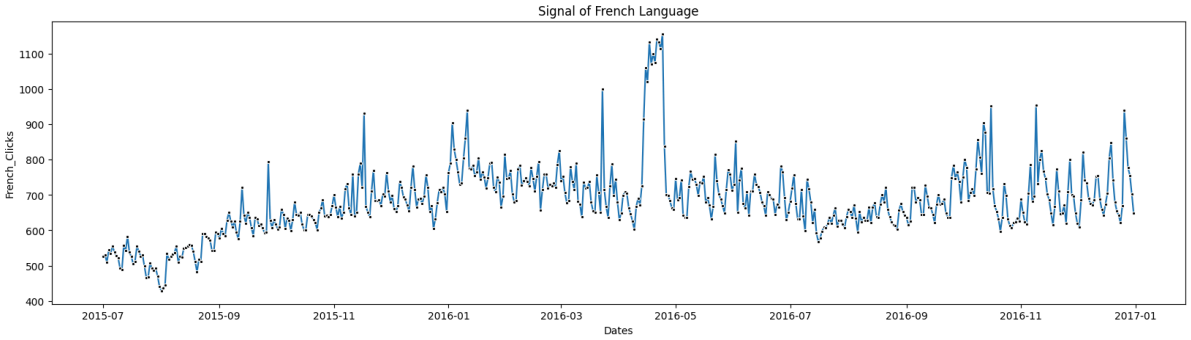
	2015-07-01	2015-07-02	2015-07-03	2015-07-04	2015-07-05	2015-07-06	2015-07-07	2015-07-08	2015-07-09	2015-07-10	2015-07-11	2015-07-12	20 07-
8357	3	4	7	4	4	2	3	7	2	NaN	3	1	
8358	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
8359	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	0	NaN	NaN	N
8360	2,403	20,136	1,850	1,432	1,351	1,792	1,710	2,039	1,957	1,640	1,263	1,373	1,(
8361	982	881	794	979	1,191	1,057	1,184	860	1,259	1,257	745	657	8

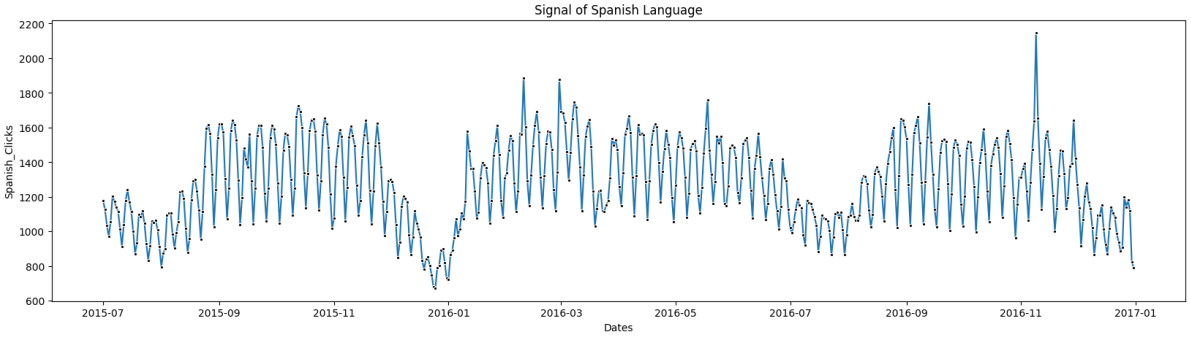
In [ ]:
```
from functools import reduce
def get_all_langs_df_melted(df_dict):
  all_langs_melf_df_dict = {}
  for k, v in df_dict.items():
    dates_df = df_dict[k].drop(columns = string_columns)
    dates_melt_df = dates_df.melt(var_name='Dates', value_name=f'{k}_Clicks')
    dates_melt_df['Dates'] = pd.to_datetime(dates_melt_df['Dates'])
    dates_melt_df.sort_values(by='Dates', ascending=True, inplace=True)
    all_langs_melf_df_dict.update({k:dates_melt_df})

  return  all_langs_melf_df_dict

all_lang_melt_df_dict = get_all_langs_df_melted(all_lang_df_objs_dict)

del all_lang_df_objs_dict
gc.collect()
```

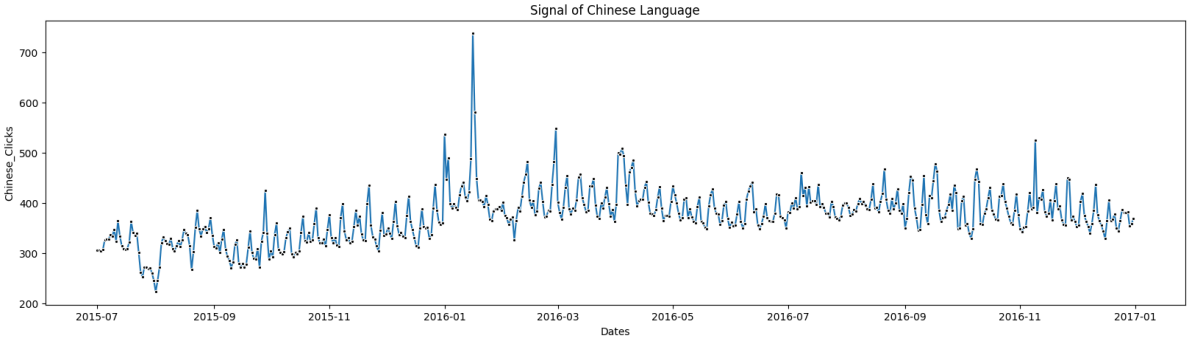
Out[ ]:
```
21916
```

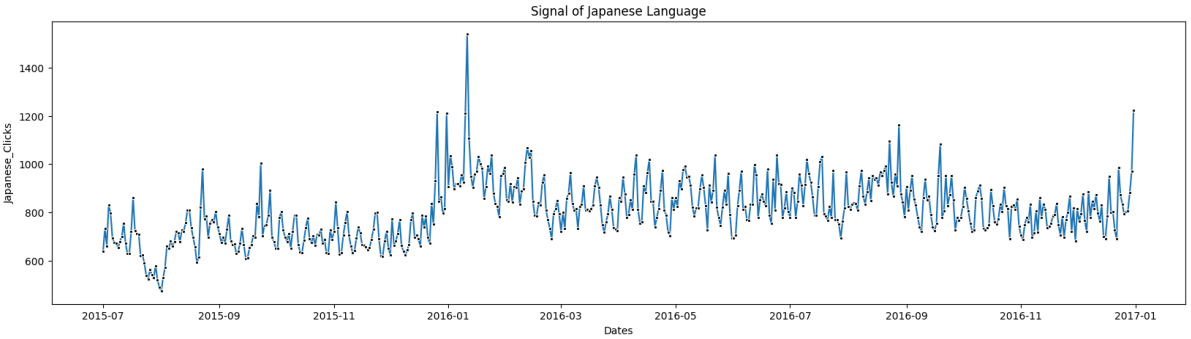
In [ ]:
```
for k, v in all_lang_melt_df_dict.items():
  plt.figure(figsize=(20,5))
  plt.title(f'Signal of {k} Language')
  sns.lineplot(data=v, x='Dates', y=f'{k}_Clicks',errorbar=None, marker='o', marker
  plt.show()
```

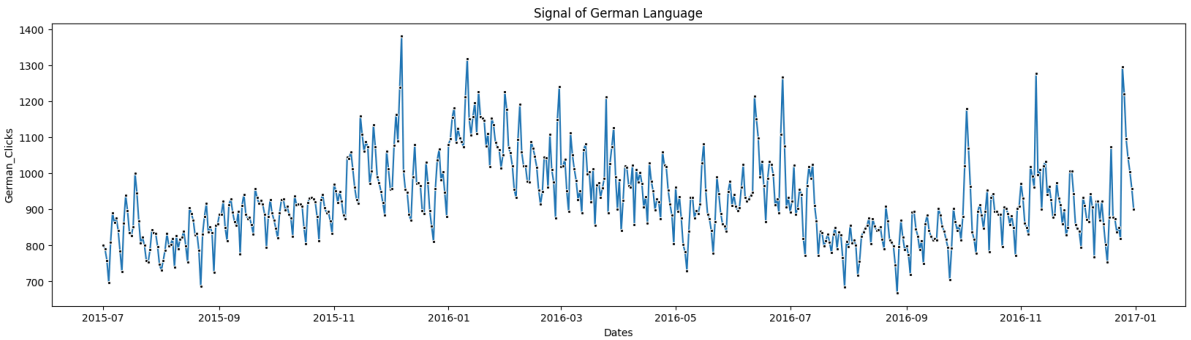


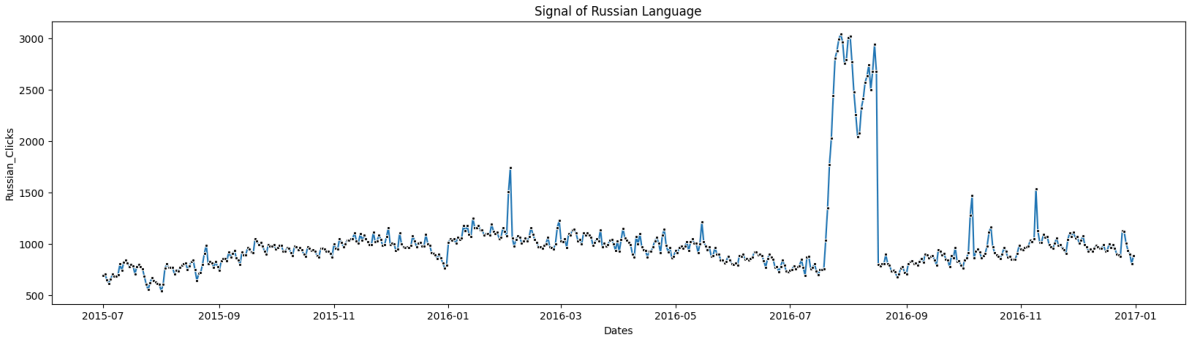

Signal of French Language
French_Clicks
Dates


Signal of Spanish Language
Spanish_Clicks
Dates


Signal of Chinese Language
Chinese_Clicks
Dates


Signal of Japanese Language
Japanese_Clicks
Dates


Signal of German Language
German_Clicks
Dates



In [ ]:
```
for k, v in all_lang_melt_df_dict.items():
  with open(f'/content/drive/MyDrive/Scaler_DSML_Digital_Notes/BusinessCases/11_AdE
    pickle.dump(all_lang_melt_df_dict[k], fp)
```

In [ ]:
```
%whos
```

```
Variable                                Type                            Data/Info
-------------------------------------------------------------------------------
Back                                    AnsiBack                        <colorama.ans
i.AnsiBack object at 0x7bdaf4a01510>
ConfusionMatrixDisplay                  type                            <class 'sklea
rn.metrics._<...>.ConfusionMatrixDisplay'>
DecisionTreeClassifier                  ABCMeta                         <class 'sklea
rn.tree._cla<...>.DecisionTreeClassifier'>
Fore                                    AnsiFore                        <colorama.ans
i.AnsiFore object at 0x7bdaf4a01090>
GradientBoostingClassifier              ABCMeta                         <class 'sklea
rn.ensemble.<...>dientBoostingClassifier'>
GridSearchCV                            ABCMeta                         <class 'sklea
rn.model_sel<...>on._search.GridSearchCV'>
HTML                                    type                            <class 'IPyth
on.core.display.HTML'>
Image                                   type                            <class 'IPyth
on.core.display.Image'>
KFold                                   ABCMeta                         <class 'sklea
rn.model_selection._split.KFold'>
LogisticRegression                      type                            <class 'sklea
rn.linear_mo<...>stic.LogisticRegression'>
MinMaxScaler                            type                            <class 'sklea
rn.preproces<...>sing._data.MinMaxScaler'>
OLS                                     type                            <class 'stats
models.regression.linear_model.OLS'>
OrdinalEncoder                          type                            <class 'sklea
rn.preproces<...>encoders.OrdinalEncoder'>
RandomForestClassifier                  ABCMeta                         <class 'sklea
rn.ensemble.<...>.RandomForestClassifier'>
RandomizedSearchCV                      ABCMeta                         <class 'sklea
rn.model_sel<...>arch.RandomizedSearchCV'>
SARIMAX                                 type                            <class 'stats
models.tsa.s<...>tespace.sarimax.SARIMAX'>
SMOTE                                   ABCMeta                         <class 'imble
arn.over_sam<...>pling._smote.base.SMOTE'>
SimpleImputer                           type                            <class 'sklea
rn.impute._base.SimpleImputer'>
StandardScaler                          type                            <class 'sklea
rn.preproces<...>ng._data.StandardScaler'>
StringIO                                type                            <class '_io.S
tringIO'>
Style                                   AnsiStyle                       <colorama.ans
i.AnsiStyle <...>object at 0x7bdaf4a019d0>
accuracy_score                          function                        <function acc
uracy_score at 0x7bda97d3bb00>
all_lang_df                             DataFrame                              Langua
ge_Full     <...>44936 rows x 554 columns]
all_lang_melt_df_dict                   dict                            n=7
annotate                                function                        <function ann
otate at 0x7bda6a921120>
ax                                      Axes                            Axes(0.125,0.
11;0.775x0.77)
calendar                                module                          <module 'cale
ndar' from '<...>/python3.11/calendar.py'>
ce                                      module                          <module 'cate
gory_encoder<...>ry_encoders/__init__.py'>
chi2                                    chi2_gen                        <scipy.stats.
_continuous_<...>object at 0x7bda9f32abd0>
chi2_contingency                        function                        <function chi
2_contingency at 0x7bda9e818b80>
chisquare                               function                        <function chi
square at 0x7bda9eaffba0>
classification_report                   function                        <function cla
```

```
ssification_<...>report at 0x7bda97d6cfe0>
column_names                          Index                         Index(['Langu
age_Full', '<...>ype='object', length=551)
columns                               list                          n=554
confusion_matrix                      function                      <function con
fusion_matrix at 0x7bda97d3bc40>
cross_validate                        function                      <function cro
ss_validate at 0x7bda97514900>
date_columns                          Index                         Index(['2015-
07-01', '201<...>ype='object', length=550)
df                                    DataFrame
<...>44936 rows x 556 columns]
df1                                   DataFrame
<...>44936 rows x 555 columns]
df_zero_nan_fill                      DataFrame                            Langua
ge_Full  201<...>44936 rows x 551 columns]
display                               function                      <function dis
play at 0x7bdaf54e9e40>
drive                                 module                        <module 'goog
le.colab.dri<...>s/google/colab/drive.py'>
dt                                    module                        <module 'date
time' from '<...>/python3.11/datetime.py'>
english_dates_df                      DataFrame                             2015-
07-01  2015-<...>24108 rows x 550 columns]
export_graphviz                       function                      <function exp
ort_graphviz at 0x7bda967c4900>
f_oneway                              function                      <function f_o
neway at 0x7bda9eafc680>
fp                                    BufferedWriter                <_io.Buffered
Writer name=<...>Analysis/Russian.pickle'>
gc                                    module                        <module 'gc'
(built-in)>
get_access_type_origin_value_cpounts  function                      <function get
_access_type<...>pounts at 0x7bda6a8c3ec0>
get_all_langs_df_melted               function                      <function get
_all_langs_d<...>melted at 0x7bd9e2530180>
get_language_name                     function                      <function get
_language_name at 0x7bda68fbdf80>
html                                  str                           \n<div style
="display: fl<...>le>\n    </div>\n</div>\n
i                                     str                           Access_Origin
k                                     str                           Russian
kstest                                function                      <function kst
est at 0x7bda9eb08ea0>
lang                                  str                           Russian
languages_list                        list                          n=7
lzip                                  function                      <function lzi
p at 0x7bdaa175c720>
mean_absolute_error                   function                      <function mea
n_absolute_error at 0x7bda97d8d940>
mean_squared_error                    function                      <function mea
n_squared_error at 0x7bda97d8dd00>
norm                                  norm_gen                      <scipy.stats.
_continuous_<...>object at 0x7bda9f319f90>
np                                    module                        <module 'nump
y' from '/us<...>kages/numpy/__init__.py'>
null_rows                             Index                         Index([13882,
13946, 1404<...>type='int64', length=127)
pattern                               str                           (?P<specific_
name>\S+)_(?<...>+)_(?P<access_origin>\S+)
pd                                    module                        <module 'pand
as' from '/u<...>ages/pandas/__init__.py'>
pearsonr                              function                      <function pea
rsonr at 0x7bda9eafd620>
```

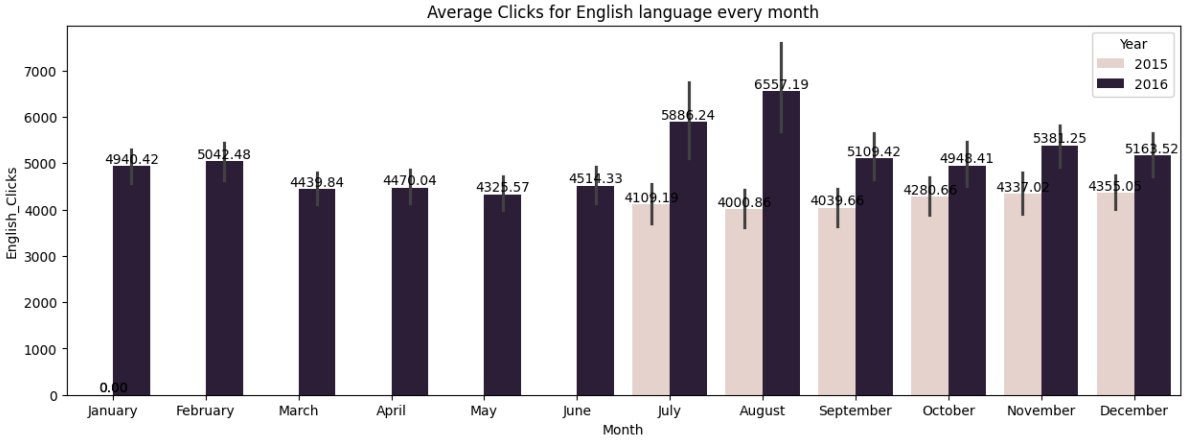
```
pickle                                  module                          <module 'pick
le' from '/u<...>ib/python3.11/pickle.py'>
plt                                     module                          <module 'matp
lotlib.pyplo<...>es/matplotlib/pyplot.py'>
precision_recall_curve                  function                        <function pre
cision_recal<...>_curve at 0x7bda97d6f880>
pycountry                               module                          <module 'pyco
untry' from <...>s/pycountry/__init__.py'>
pydot                                   module                          <module 'pydo
t' from '/us<...>kages/pydot/__init__.py'>
qqplot                                  function                        <function qqp
lot at 0x7bda98a88860>
r2_score                                function                        <function r2_
score at 0x7bda97d8e520>
re                                      module                          <module 're'
from '/usr/l<...>thon3.11/re/__init__.py'>
reduce                                  builtin_function_or_method      <built-in fun
ction reduce>
regex                                   Pattern                         regex.Regex
('(?P<specific<...>n>\\S+)', flags=regex.V0)
roc_auc_score                           function                        <function roc
_auc_score at 0x7bda97d6f600>
roc_curve                               function                        <function roc
_curve at 0x7bda97d6f9c0>
shapiro                                 function                        <function sha
piro at 0x7bda9e81b1a0>
sm                                      module                          <module 'stat
smodels.api'<...>ages/statsmodels/api.py'>
sms                                     module                          <module 'stat
smodels.stat<...>tatsmodels/stats/api.py'>
sns                                     module                          <module 'seab
orn' from '/<...>ges/seaborn/__init__.py'>
spearmanr                               function                        <function spe
armanr at 0x7bda9eafd940>
split_page_fields                       function                        <function spl
it_page_fields at 0x7bda91486b60>
string_columns                          list                            n=4
train_test_split                        function                        <function tra
in_test_split at 0x7bda974f3100>
ttest_ind                               function                        <function tte
st_ind at 0x7bda9eafef20>
ttest_rel                               function                        <function tte
st_rel at 0x7bda9eaff4c0>
uniform                                 uniform_gen                     <scipy.stats.
_continuous_<...>object at 0x7bdaa080f8d0>
v                                       DataFrame
Dates  Russi<...>8262100 rows x 2 columns]
variance_inflation_factor               function                        <function var
iance_inflat<...>factor at 0x7bda974c4f40>
warnings                                module                          <module 'warn
ings' from '<...>/python3.11/warnings.py'>
```

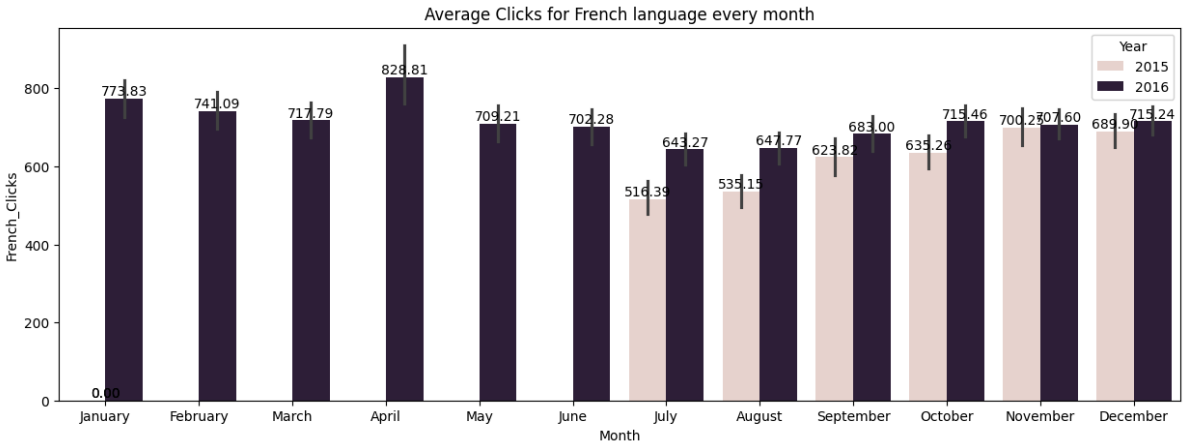
In [ ]:
```
del all_lang_melt_df_dict
gc.collect()
```

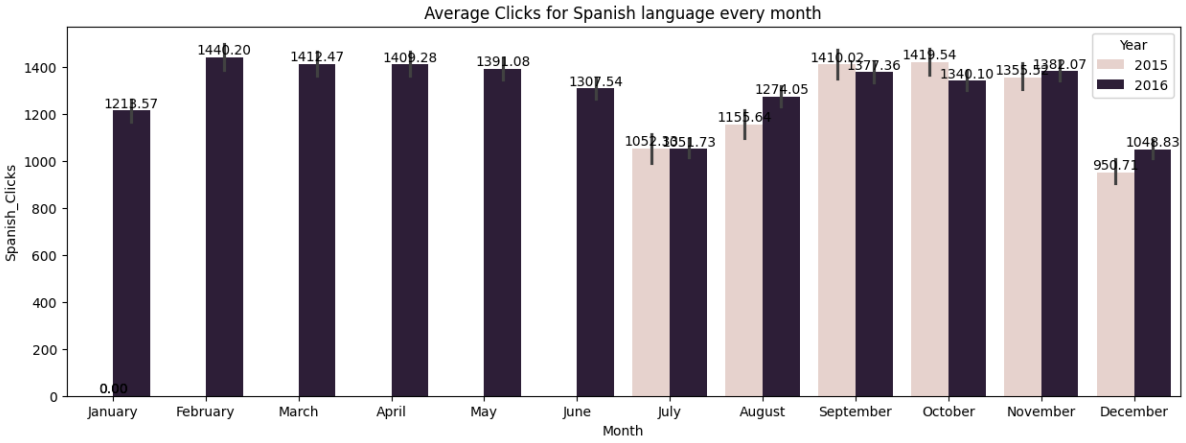
Out[ ]:
```
12987
```

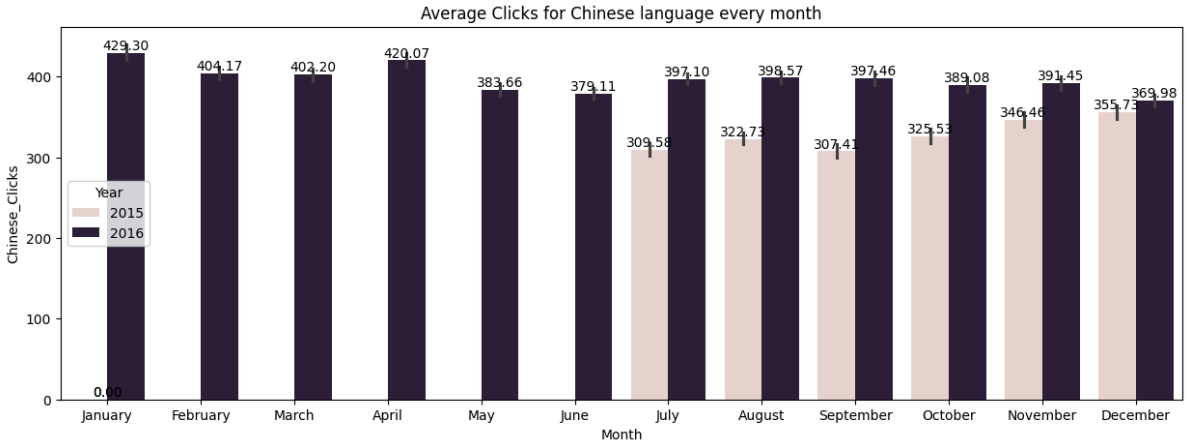
In [ ]:
```
month_order = ['January', 'February', 'March', 'April', 'May', 'June',
               'July', 'August', 'September', 'October', 'November', 'December']

for lang in languages_list:
  with open(f'/content/drive/MyDrive/Scaler_DSML_Digital_Notes/BusinessCases/11_AdE
      dataframe = pickle.load(fp)
      dataframe['Month'] = dataframe['Dates'].dt.month_name()
```

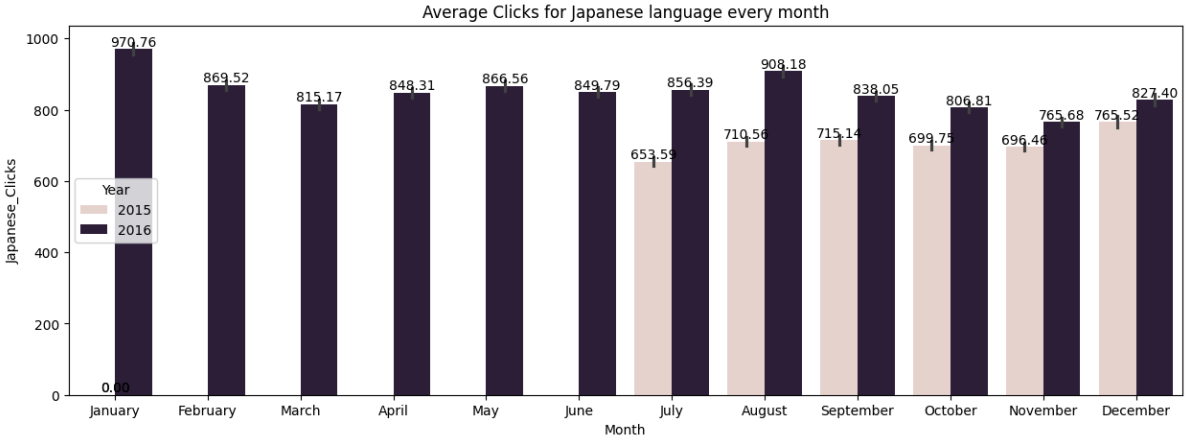
```
    dataframe['Year'] = dataframe['Dates'].dt.year
    plt.figure(figsize=(15,5))
    plt.title(f'Average Clicks for {lang} language every month')
    ax = sns.barplot(data=dataframe, x='Month', y=f'{lang}_Clicks', hue='Year', c
    annotate(ax)
    plt.show()
    del dataframe
    gc.collect()
```

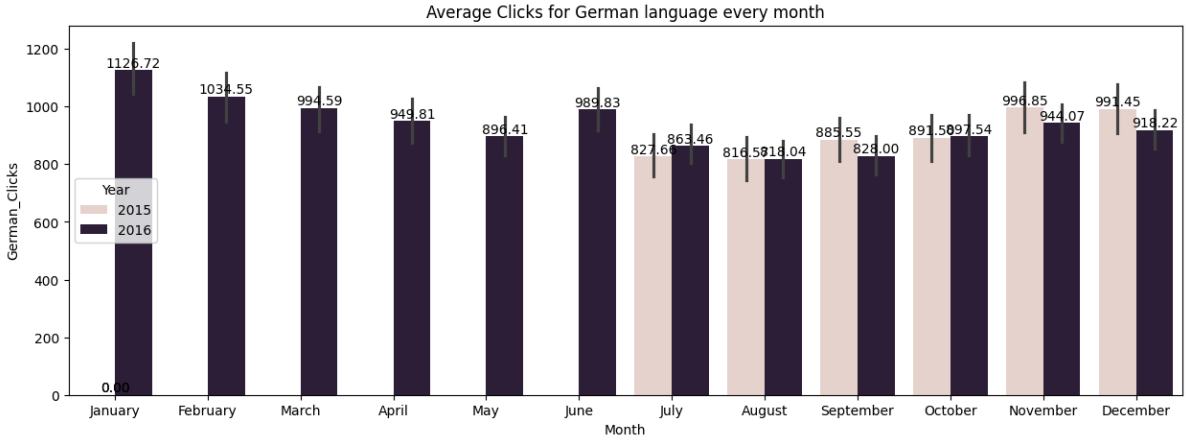


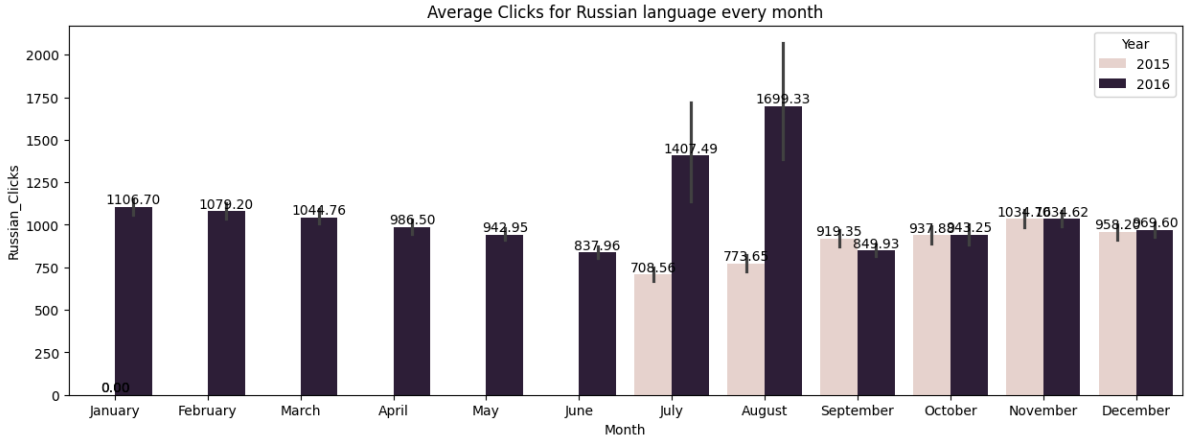






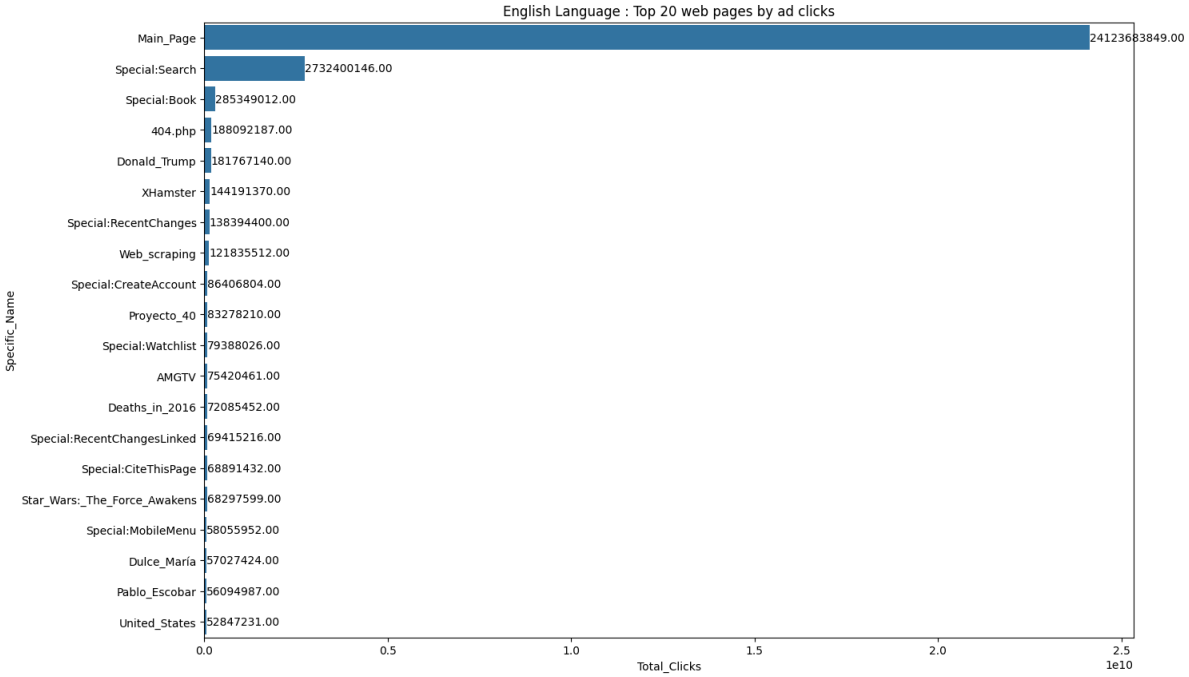


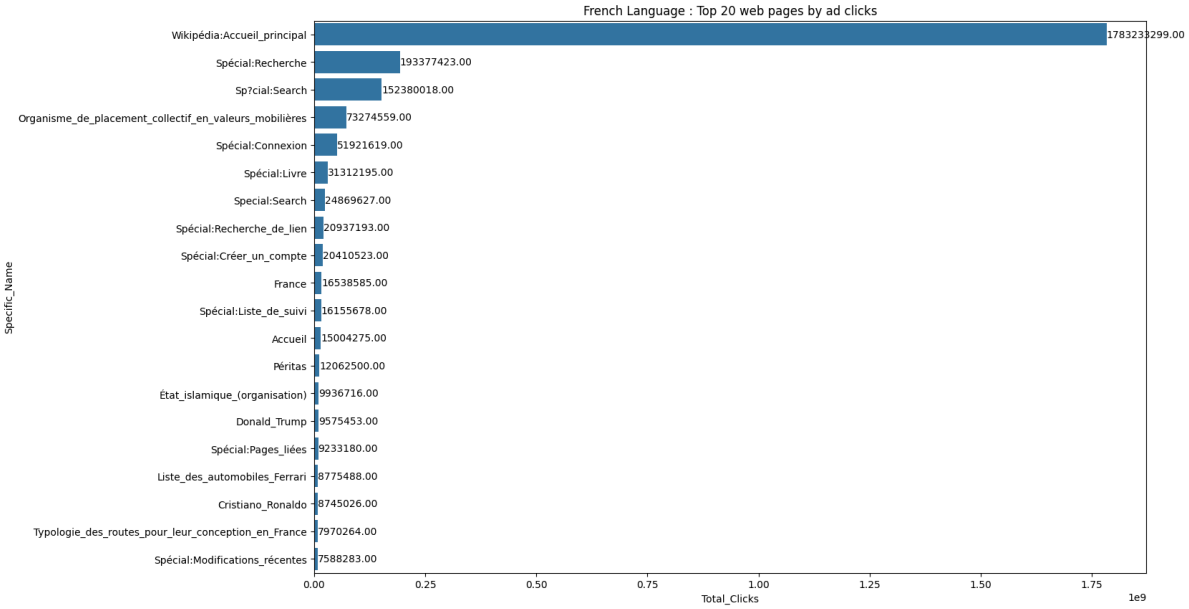


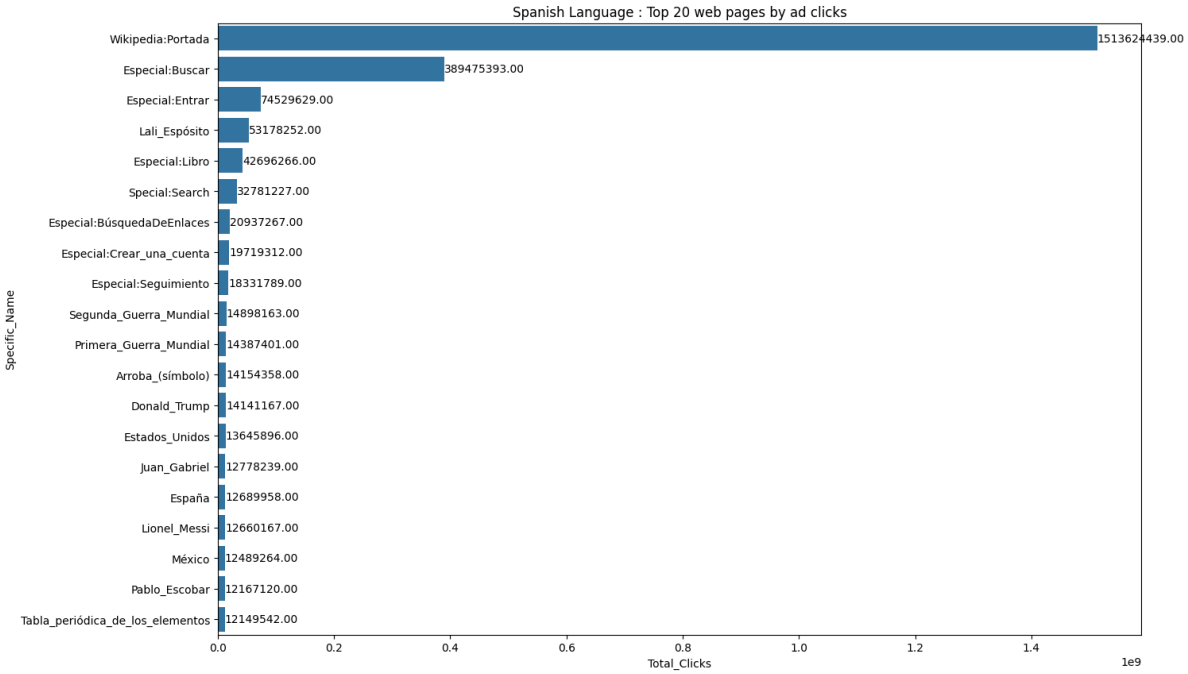


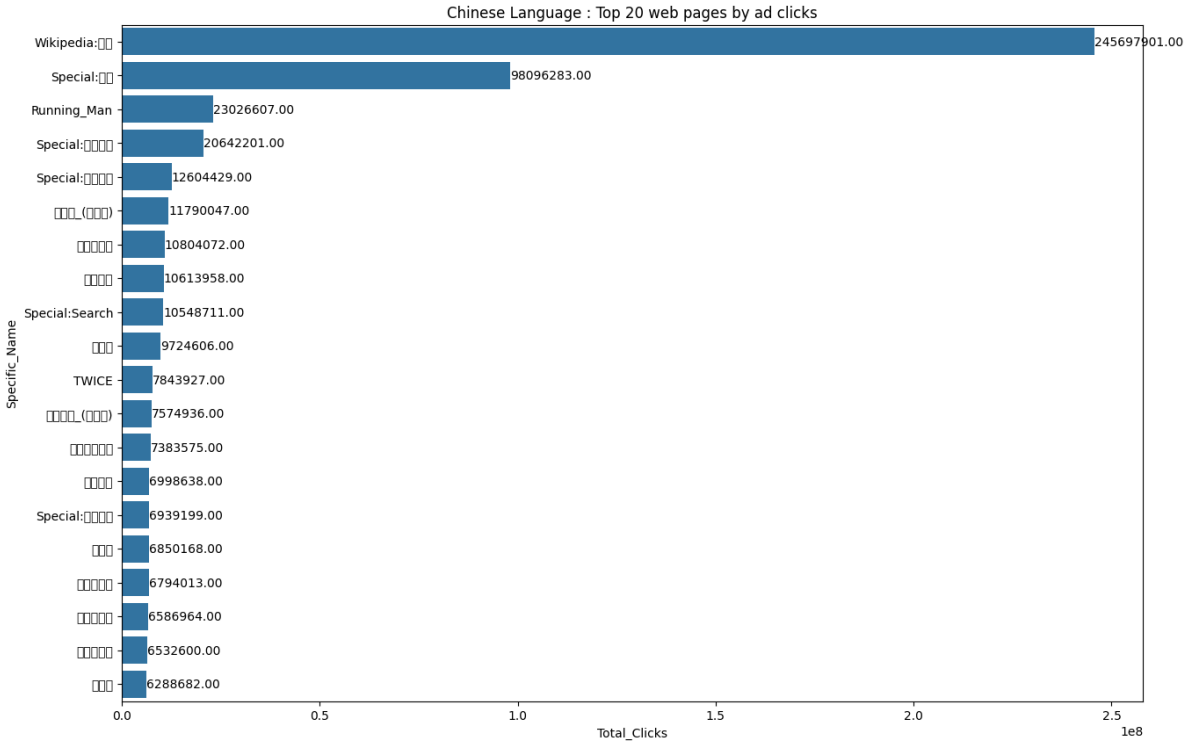
In [ ]:
```
for lang in languages_list:

  with open(f'/content/drive/MyDrive/Scaler_DSML_Digital_Notes/BusinessCases/11_AdE
    dataframe = pickle.load(fp)


    df_columns = dataframe.columns[4:]
```

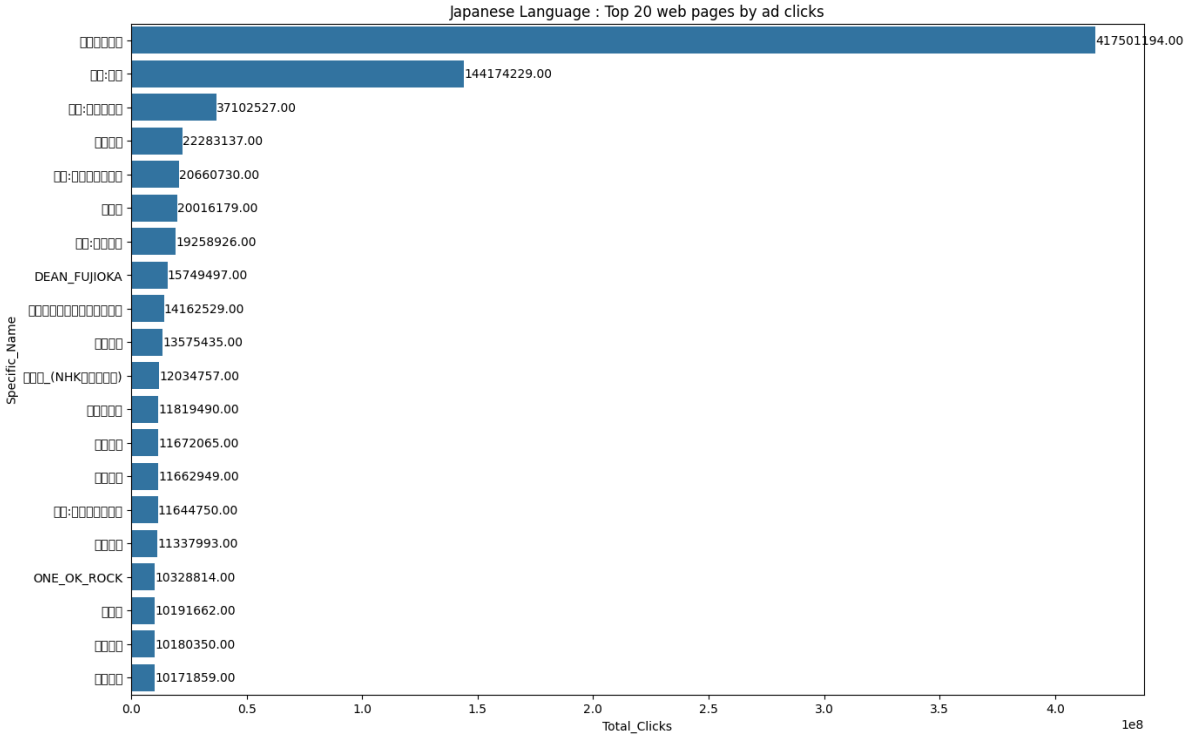
```
    df_columns  = df_columns.insert(0, 'Specific_Name')

    only_dates_df = dataframe[df_columns]

    warnings.filterwarnings("ignore", category=pd.errors.SettingWithCopyWarning)
    only_dates_df.loc[:,'Total_Clicks'] = only_dates_df.iloc[:,1:].sum(axis=1)
    only_dates_df.loc[:,['Specific_Name', 'Total_Clicks']].sort_values(by='Total_Cl
    only_dates_df = only_dates_df.groupby(by='Specific_Name')['Total_Clicks'].agg('
    only_dates_df = only_dates_df.reset_index()
    plt.figure(figsize=(15, 10))
    plt.title(f'{lang} Language : Top 20 web pages by ad clicks')
    ax=sns.barplot(data=only_dates_df.head(20), x='Total_Clicks', y='Specific_Name'
    annotate(ax, rotation=True)
    plt.show()
    print(' ')

    del dataframe, only_dates_df, df_columns
    gc.collect()
```

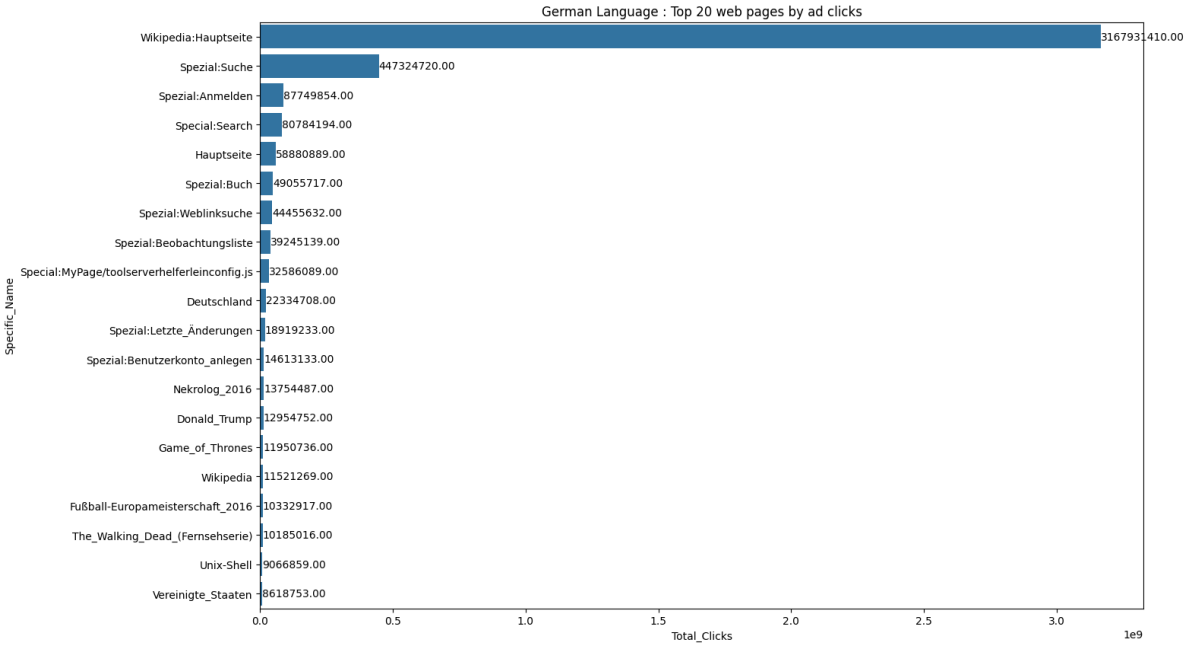


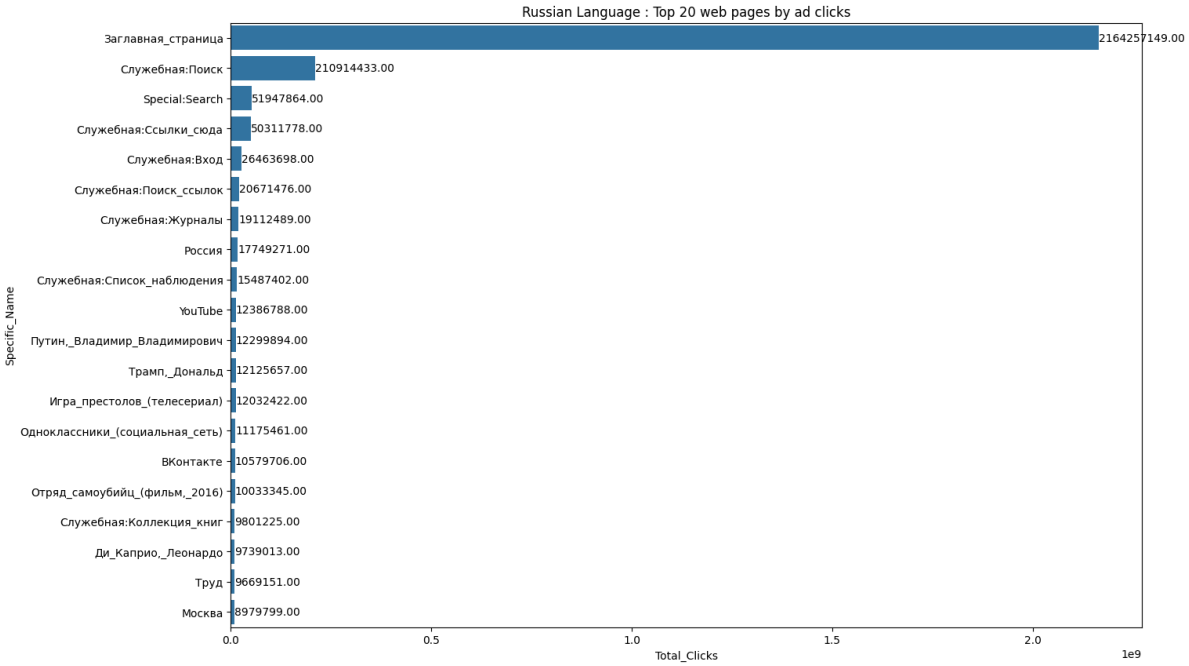



Spanish Language : Top 20 web pages by ad clicks
Specific_Name
Wikipedia:Portada 1513624439.00
Especial:Buscar 389475393.00
Especial:Entrar 74529629.00
Lali_Espósito 53178252.00
Especial:Libro 42696266.00
Special:Search 32781227.00
Especial:BúsquedaDeEnlaces 20937267.00
Especial:Crear_una_cuenta 19719312.00
Especial:Seguimiento 18331789.00
Segunda_Guerra_Mundial 14898163.00
Primera_Guerra_Mundial 14387401.00
Arroba_(símbolo) 14154358.00
Donald_Trump 14141167.00
Estados_Unidos 13645896.00
Juan_Gabriel 12778239.00
España 12689958.00
Lionel_Messi 12660167.00
México 12489264.00
Pablo_Escobar 12167120.00
Tabla_periódica_de_los_elementos 12149542.00
0.0 0.2 0.4 0.6 0.8 1.0 1.2 1.4
Total_Clicks
1e9


Chinese Language : Top 20 web pages by ad clicks
Specific_Name
Wikipedia:□□ 245697901.00
Special:□□ 98096283.00
Running_Man 23026607.00
Special:□□□□ 20642201.00
Special:□□□□ 12604429.00
□□□_(□□□) 11790047.00
□□□□□ 10804072.00
□□□□ 10613958.00
Special:Search 10548711.00
□□□ 9724606.00
TWICE 7843927.00
□□□□_(□□□) 7574936.00
□□□□□□ 7383575.00
□□□□ 6998638.00
Special:□□□□ 6939199.00
□□□ 6850168.00
□□□□□ 6794013.00
□□□□□ 6586964.00
□□□□□ 6532600.00
□□□ 6288682.00
0.0 0.5 1.0 1.5 2.0 2.5
Total_Clicks
1e8

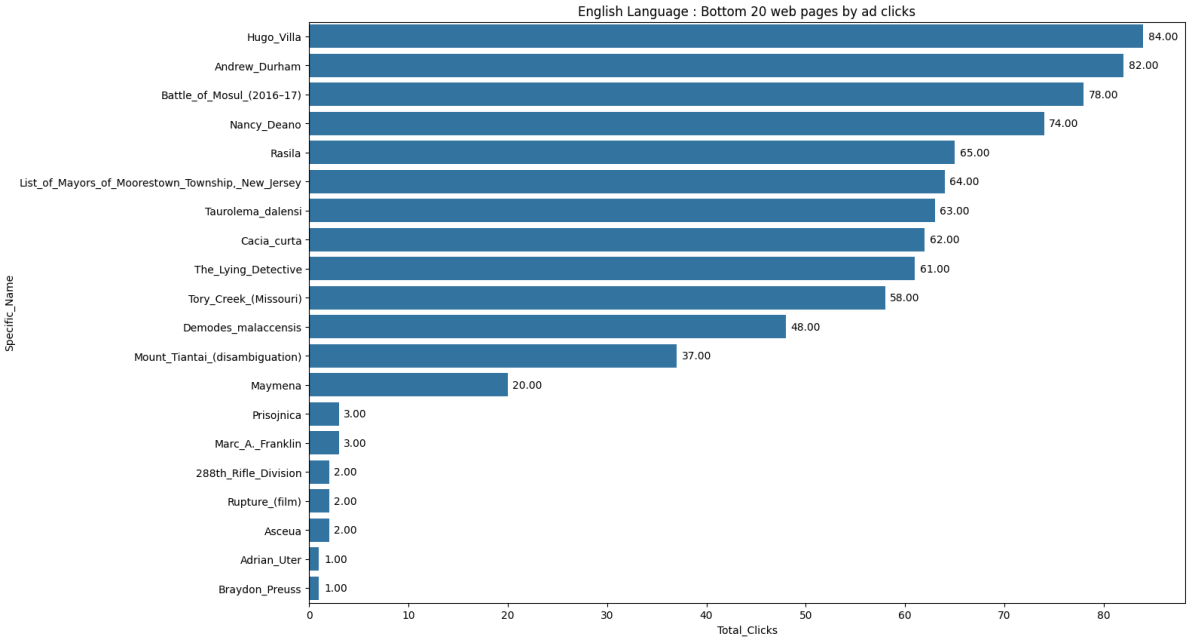

Japanese Language : Top 20 web pages by ad clicks
□□□□□□□
□□:□□
□□:□□□□□
□□□□
□□:□□□□□□□
□□□
□□:□□□□
DEAN_FUJIOKA
□□□□□□□□□□□□□□□
□□□□
□□□_(NHK□□□□)
□□□□□
□□□□
□□□□
□□:□□□□□□□
□□□□
ONE_OK_ROCK
□□□
□□□□
□□□□
Specific_Name
417501194.00
144174229.00
37102527.00
22283137.00
20660730.00
20016179.00
19258926.00
15749497.00
14162529.00
13575435.00
12034757.00
11819490.00
11672065.00
11662949.00
11644750.00
11337993.00
10328814.00
10191662.00
10180350.00
10171859.00
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
1e8
Total_Clicks

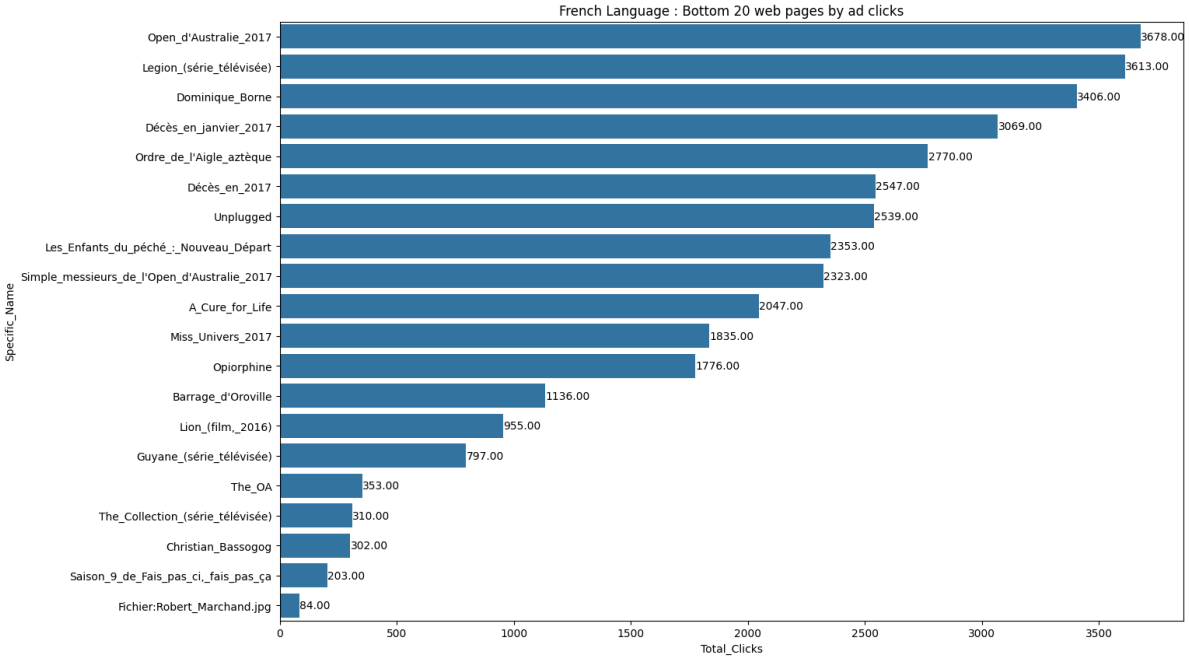

German Language : Top 20 web pages by ad clicks
Wikipedia:Hauptseite
Spezial:Suche
Spezial:Anmelden
Special:Search
Hauptseite
Spezial:Buch
Spezial:Weblinksuche
Spezial:Beobachtungsliste
Special:MyPage/toolserverhelferleinconfig.js
Deutschland
Spezial:Letzte_Änderungen
Spezial:Benutzerkonto_anlegen
Nekrolog_2016
Donald_Trump
Game_of_Thrones
Wikipedia
Fußball-Europameisterschaft_2016
The_Walking_Dead_(Fernsehserie)
Unix-Shell
Vereinigte_Staaten
Specific_Name
3167931410.00
447324720.00
87749854.00
80784194.00
58880889.00
49055717.00
44455632.00
39245139.00
32586089.00
22334708.00
18919233.00
14613133.00
13754487.00
12954752.00
11950736.00
11521269.00
10332917.00
10185016.00
9066859.00
8618753.00
0.0
0.5
1.0
1.5
2.0
2.5
3.0
1e9
Total_Clicks

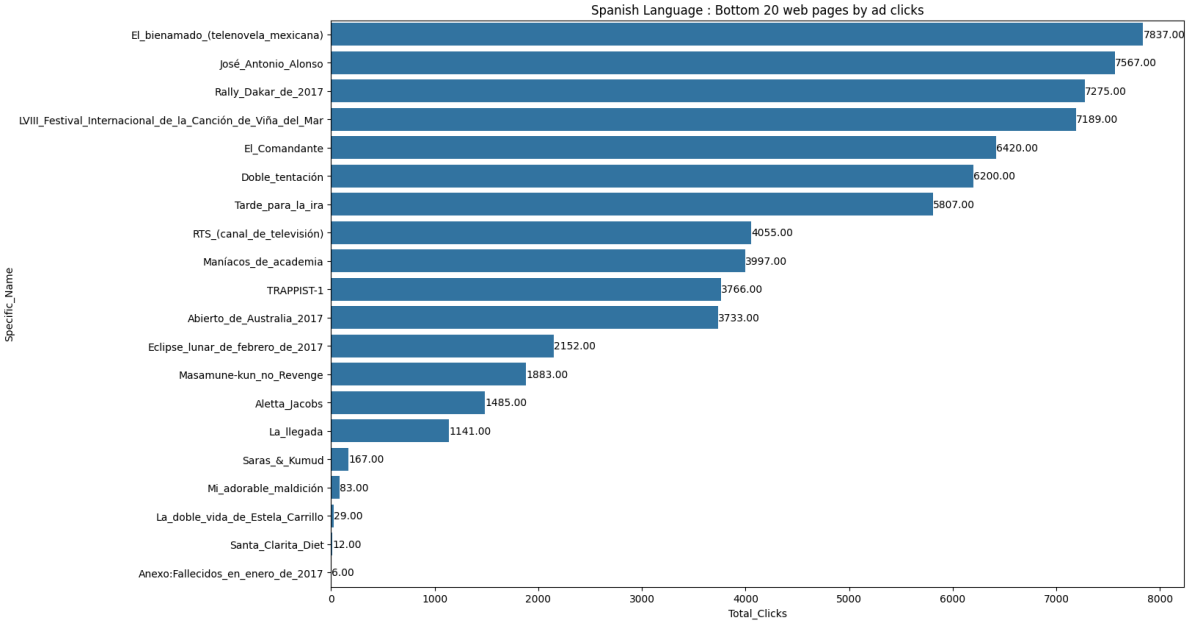


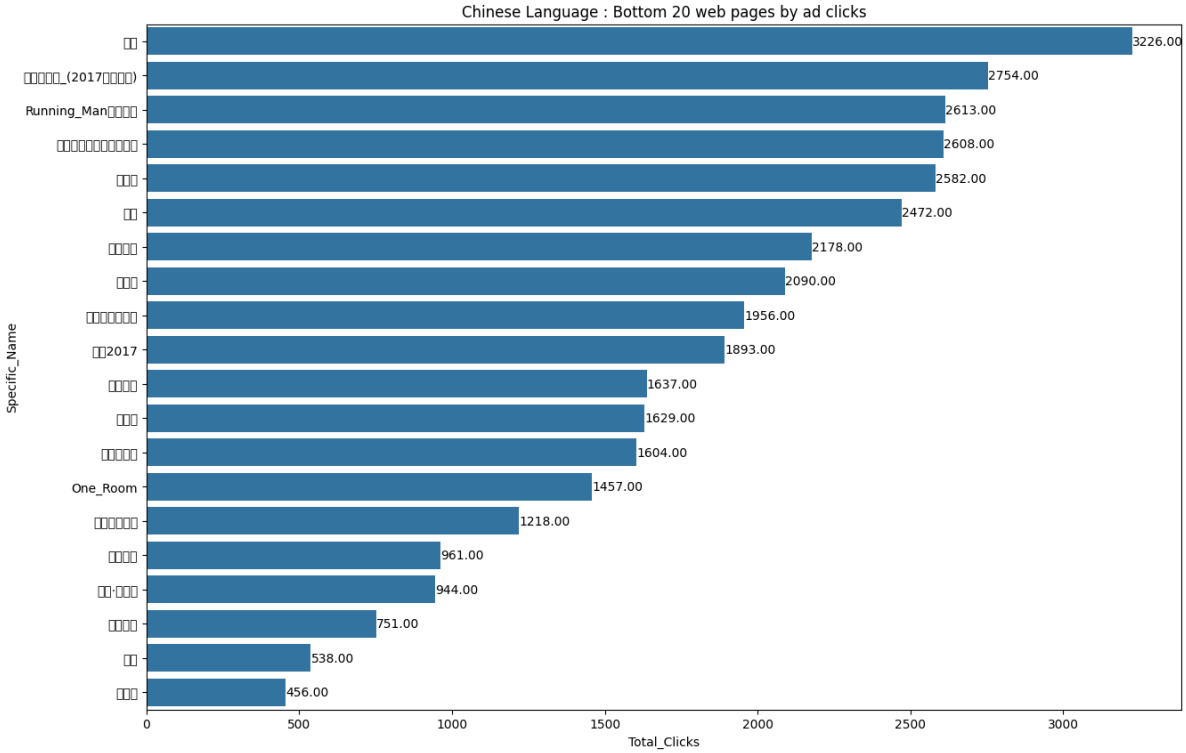
In [ ]:

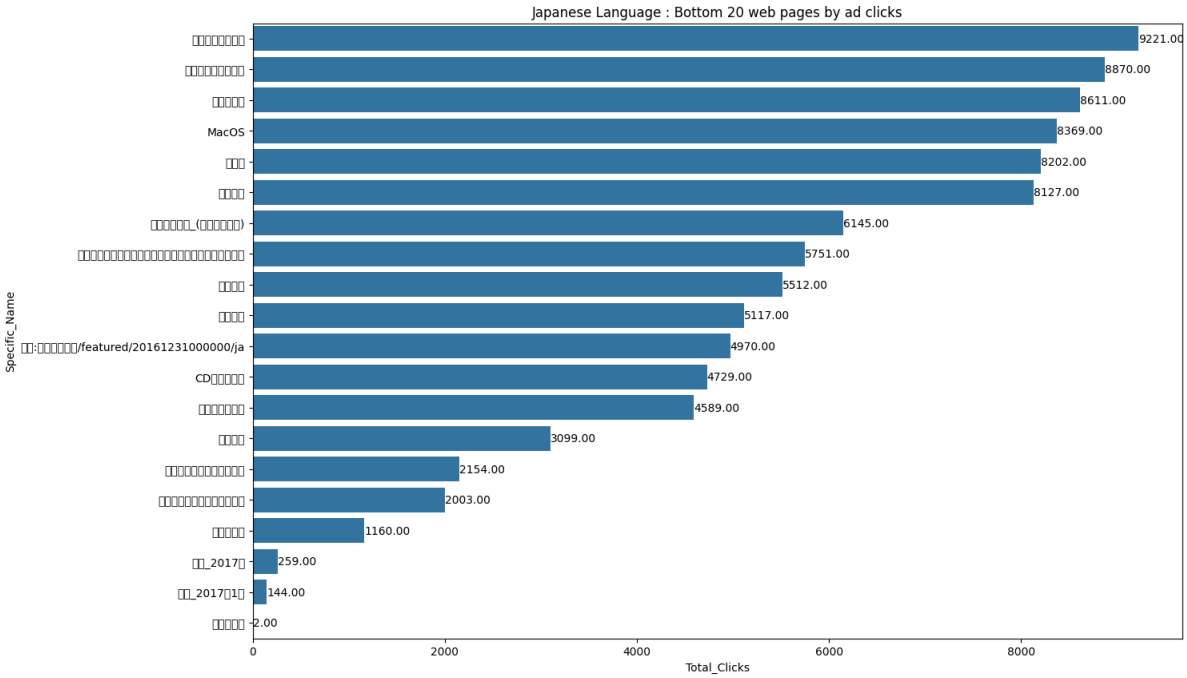
```
for lang in languages_list:

  with open(f'/content/drive/MyDrive/Scaler_DSML_Digital_Notes/BusinessCases/11_AdE
    dataframe = pickle.load(fp)


    df_columns = dataframe.columns[4:]
    df_columns  = df_columns.insert(0, 'Specific_Name')

    only_dates_df = dataframe[df_columns]

    warnings.filterwarnings("ignore", category=pd.errors.SettingWithCopyWarning)
    only_dates_df.loc[:,'Total_Clicks'] = only_dates_df.iloc[:,1:].sum(axis=1)
    only_dates_df.loc[:,['Specific_Name', 'Total_Clicks']].sort_values(by='Total_Cl
    only_dates_df = only_dates_df.groupby(by='Specific_Name')['Total_Clicks'].agg('
    only_dates_df = only_dates_df.reset_index()
    plt.figure(figsize=(15, 10))
    only_dates_df = only_dates_df[only_dates_df['Total_Clicks'] != 0]
    plt.title(f'{lang} Language : Bottom 20 web pages by ad clicks')
    ax=sns.barplot(data=only_dates_df.tail(20), x='Total_Clicks', y='Specific_Name'
    annotate(ax, rotation=True)
    plt.show()
    print(' ')
    del dataframe, only_dates_df, df_columns
    gc.collect()
```

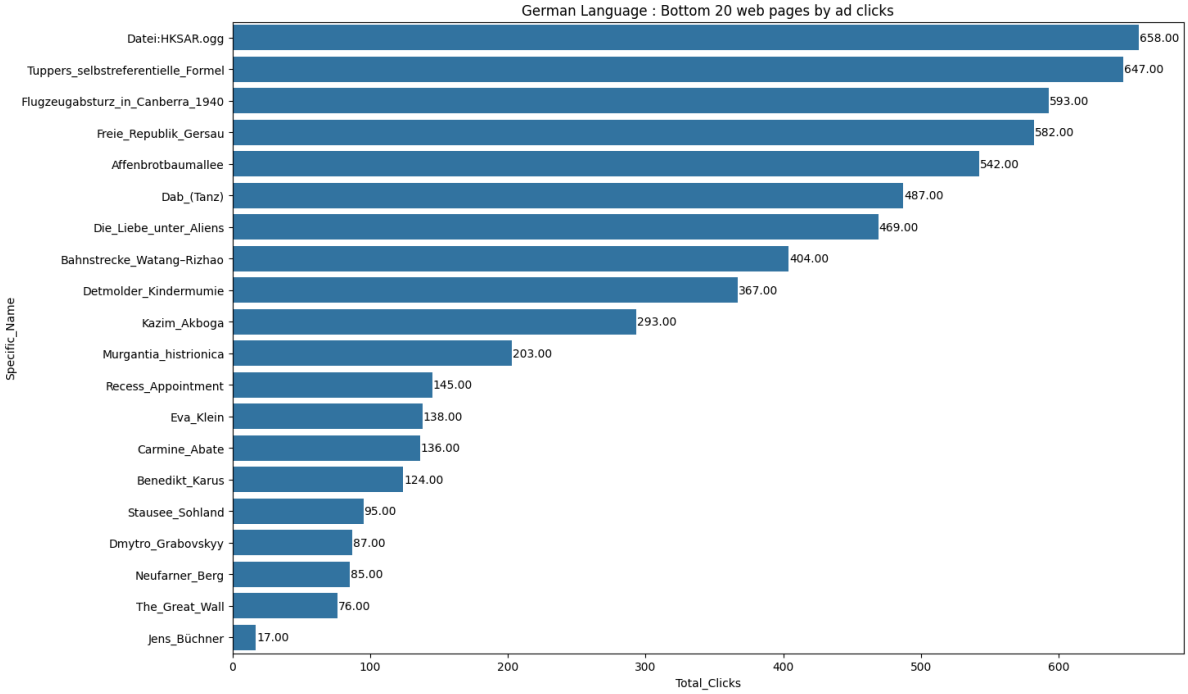

English Language : Bottom 20 web pages by ad clicks
Hugo_Villa
Andrew_Durham
Battle_of_Mosul_(2016–17)
Nancy_Deano
Rasila
List_of_Mayors_of_Moorestown_Township,_New_Jersey
Taurolema_dalensi
Cacia_curta
The_Lying_Detective
Tory_Creek_(Missouri)
Demodes_malaccensis
Mount_Tiantai_(disambiguation)
Maymena
Prisojnica
Marc_A._Franklin
288th_Rifle_Division
Rupture_(film)
Asceua
Adrian_Uter
Braydon_Preuss
84.00
82.00
78.00
74.00
65.00
64.00
63.00
62.00
61.00
58.00
48.00
37.00
20.00
3.00
3.00
2.00
2.00
2.00
1.00
1.00
Specific_Name
0
10
20
30
40
50
60
70
80
Total_Clicks

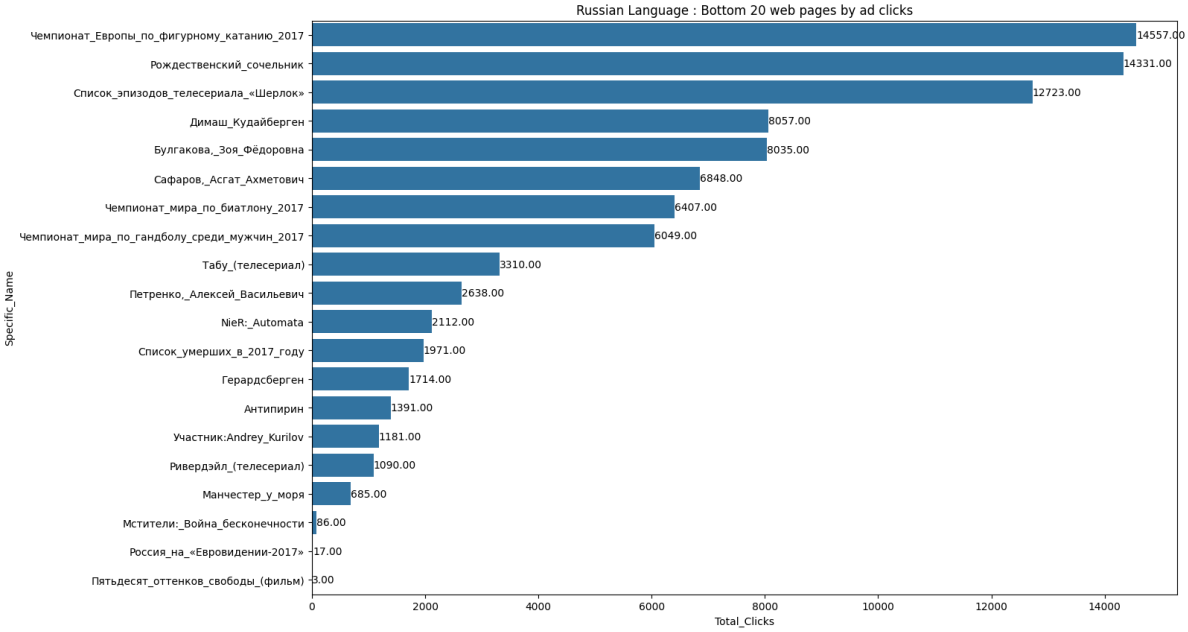

French Language : Bottom 20 web pages by ad clicks
Open_d'Australie_2017
Legion_(série_télévisée)
Dominique_Borne
Décès_en_janvier_2017
Ordre_de_l'Aigle_aztèque
Décès_en_2017
Unplugged
Les_Enfants_du_péché_:_Nouveau_Départ
Simple_messieurs_de_l'Open_d'Australie_2017
A_Cure_for_Life
Miss_Univers_2017
Opiorphine
Barrage_d'Oroville
Lion_(film,_2016)
Guyane_(série_télévisée)
The_OA
The_Collection_(série_télévisée)
Christian_Bassogog
Saison_9_de_Fais_pas_ci,_fais_pas_ça
Fichier:Robert_Marchand.jpg
3678.00
3613.00
3406.00
3069.00
2770.00
2547.00
2539.00
2353.00
2323.00
2047.00
1835.00
1776.00
1136.00
955.00
797.00
353.00
310.00
302.00
203.00
84.00
Specific_Name
0
500
1000
1500
2000
2500
3000
3500
Total_Clicks


Spanish Language : Bottom 20 web pages by ad clicks
El_bienamado_(telenovela_mexicana)
José_Antonio_Alonso
Rally_Dakar_de_2017
LVIII_Festival_Internacional_de_la_Canción_de_Viña_del_Mar
El_Comandante
Doble_tentación
Tarde_para_la_ira
RTS_(canal_de_televisión)
Maníacos_de_academia
TRAPPIST-1
Abierto_de_Australia_2017
Eclipse_lunar_de_febrero_de_2017
Masamune-kun_no_Revenge
Aletta_Jacobs
La_llegada
Saras_&_Kumud
Mi_adorable_maldición
La_doble_vida_de_Estela_Carrillo
Santa_Clarita_Diet
Anexo:Fallecidos_en_enero_de_2017
7837.00
7567.00
7275.00
7189.00
6420.00
6200.00
5807.00
4055.00
3997.00
3766.00
3733.00
2152.00
1883.00
1485.00
1141.00
167.00
83.00
29.00
12.00
6.00
Specific_Name
0
1000
2000
3000
4000
5000
6000
7000
8000
Total_Clicks


Chinese Language : Bottom 20 web pages by ad clicks
Specific_Name
Total_Clicks
3226.00
2754.00
2613.00
2608.00
2582.00
2472.00
2178.00
2090.00
1956.00
1893.00
1637.00
1629.00
1604.00
1457.00
1218.00
961.00
944.00
751.00
538.00
456.00


Japanese Language : Bottom 20 web pages by ad clicks
Specific_Name
Total_Clicks
9221.00
8870.00
8611.00
8369.00
8202.00
8127.00
6145.00
5751.00
5512.00
5117.00
4970.00
4729.00
4589.00
3099.00
2154.00
2003.00
1160.00
259.00
144.00
2.00





In [ ]:
```
with open('/content/drive/MyDrive/Scaler_DSML_Digital_Notes/BusinessCases/11_AdEase
  english_df = pickle.load(fp)
```

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

In [227…
```
english_df[english_df['English_Clicks'] == 0]
```

Out[227]:

	Dates	English_Clicks
16155	2015-07-01	0
16431	2015-07-01	0
16493	2015-07-01	0
16206	2015-07-01	0
16311	2015-07-01	0
...	...	...
13242252	2016-12-31	0
13242251	2016-12-31	0
13242736	2016-12-31	0
13243749	2016-12-31	0
13243748	2016-12-31	0

34950 rows × 2 columns

In [228...

```
english_grpd_dates_df = english_df.groupby(by='Dates')['English_Clicks'].agg('mean'
english_grpd_dates_df.tail()

english_grpd_dates_df.sort_values( ascending=False)
```

Out[228]:

	English_Clicks
Dates	
2016-07-25	8,773
2016-07-26	8,739
2016-08-15	8,708
2016-08-10	8,570
2016-08-14	8,470
...	...
2015-08-26	3,752
2015-09-04	3,743
2015-08-28	3,705
2015-08-01	3,670
2015-07-31	3,650

550 rows × 1 columns

dtype: float64

In [229...

```
english_train_df = english_grpd_dates_df.iloc[:-50]
english_test_df = english_grpd_dates_df.iloc[-50:]
```

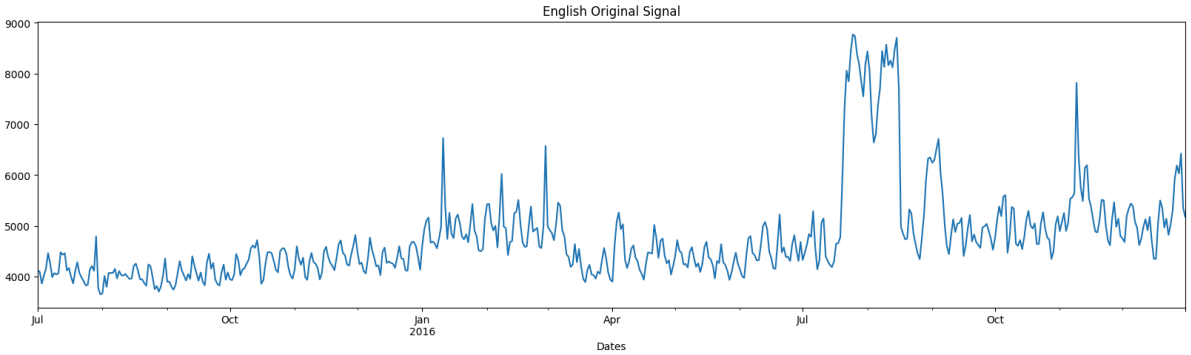
In [230...

```
print('Training_Samples :', len(english_train_df))
print('Testing_Samples :', len(english_test_df))
```

```
Training_Samples : 500
Testing_Samples : 50
```

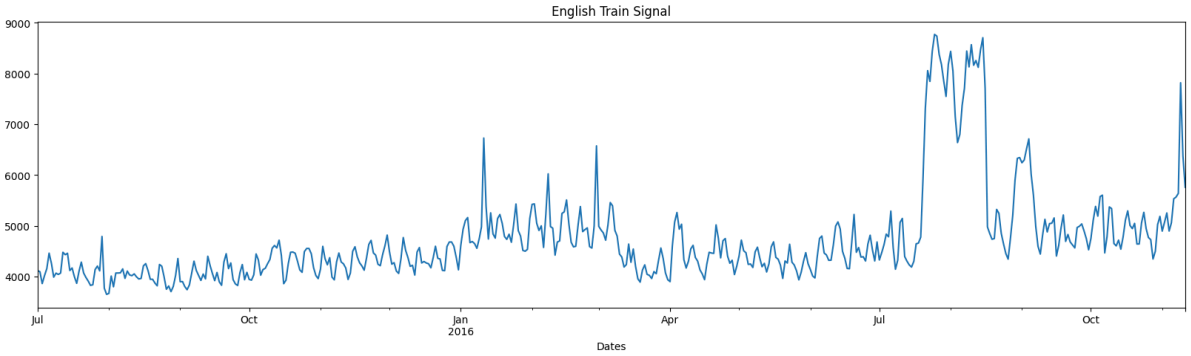
In [231…

```
plt.figure(figsize=(20,5))
plt.title('English Original Signal')
english_grpd_dates_df.plot()
plt.show()
```



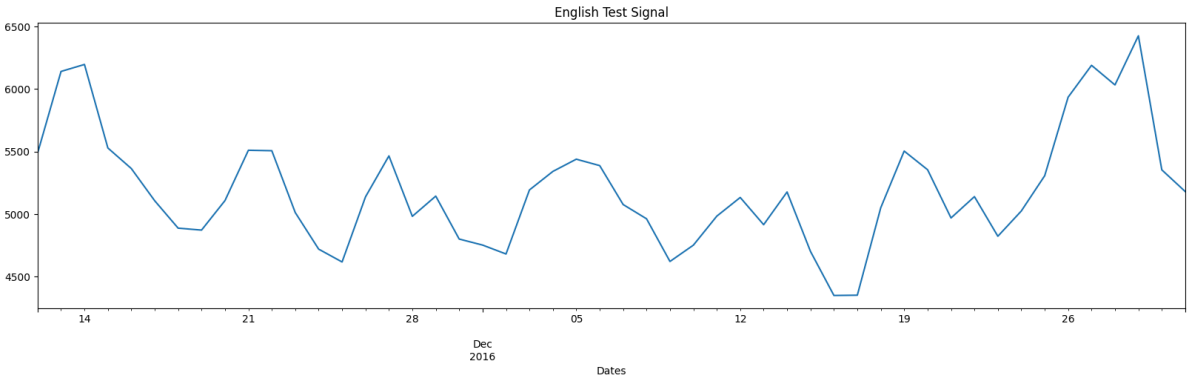
In [232…

```
plt.figure(figsize=(20,5))
plt.title('English Train Signal')
english_train_df.plot()
plt.show()
```



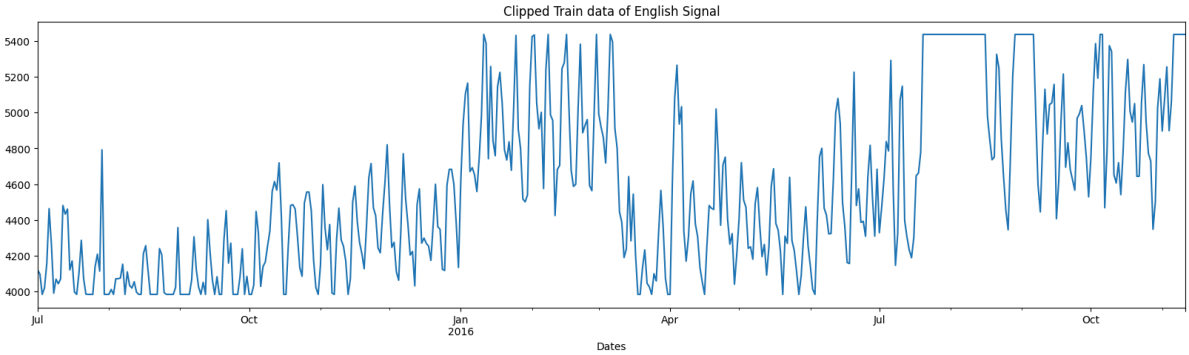
In [233…

```
plt.figure(figsize=(20,5))
plt.title('English Test Signal')
english_test_df.plot()
plt.show()
```

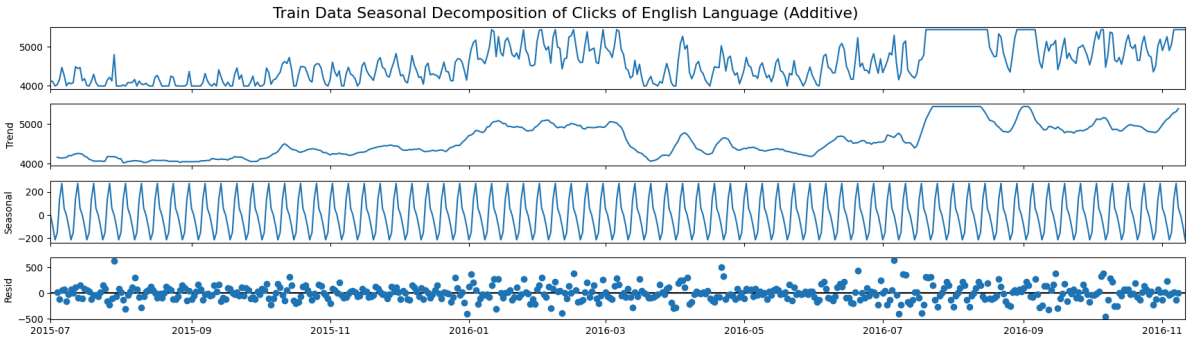


In [234…

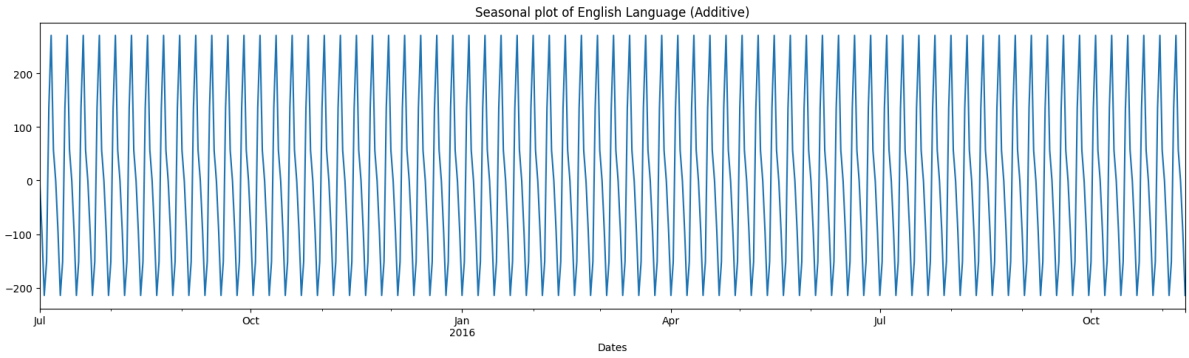
```
cliped_english_df =english_grpd_dates_df.clip(lower = english_grpd_dates_df.quantil
cliped_english_train_df =english_train_df.clip(lower = english_train_df.quantile(0.
plt.figure(figsize=(20,5))
plt.title('Clipped Train data of English Signal')
cliped_english_train_df.plot()
plt.show()
```



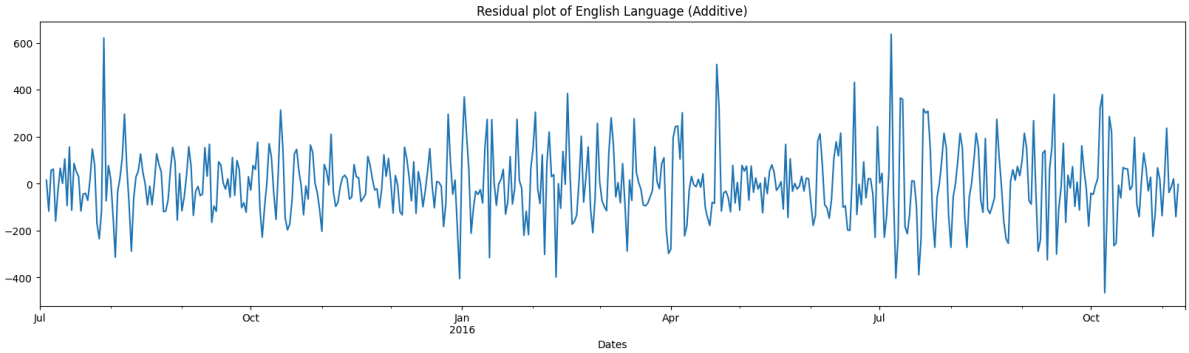
In [235…
```
model_add = sm.tsa.seasonal_decompose(cliped_english_train_df, model='additive')
fig = model_add.plot()
fig.axes[0].set_title("")
fig.suptitle("Train Data Seasonal Decomposition of Clicks of English Language (Addi
fig.set_size_inches(20, 5)
```



In [236…
```
model_add.seasonal.plot(figsize=(20,5))
plt.title('Seasonal plot of English Language (Additive)')
plt.show()
```

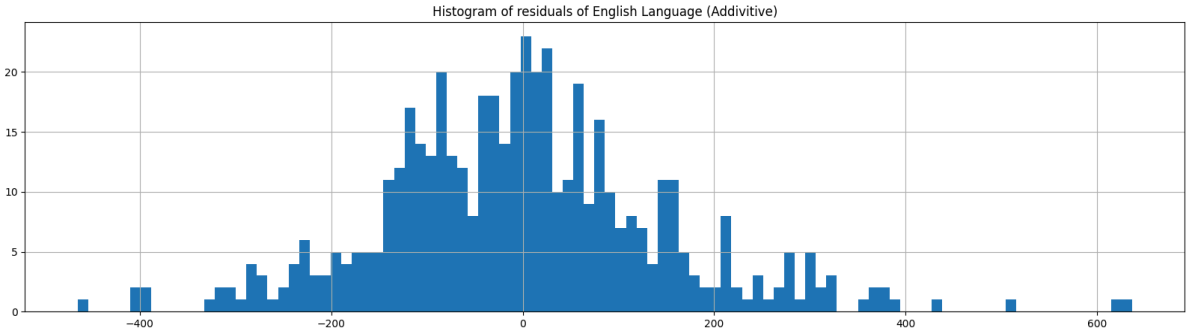


In [237…
```
model_add.resid.plot(figsize=(20,5))
plt.title('Residual plot of English Language (Additive)')
plt.show()
```



In [238…
```
model_add.resid.hist(bins=100,figsize=(20,5))
plt.title('Histogram of residuals of English Language (Addivitive)')
```

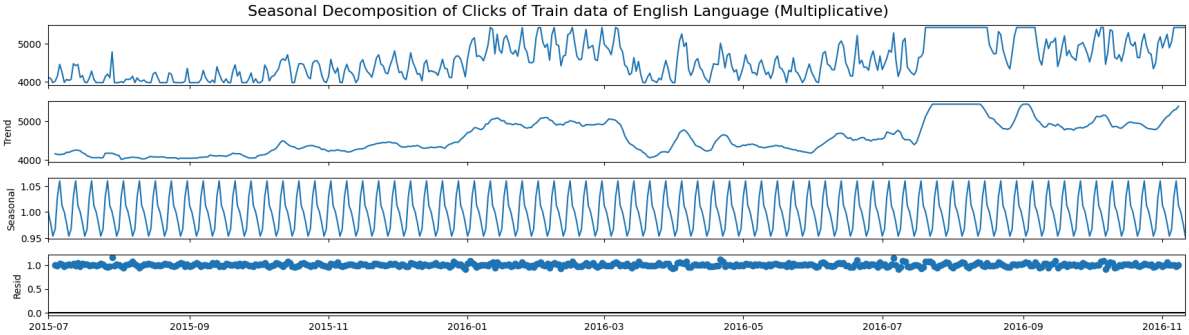
```
plt.show()
```



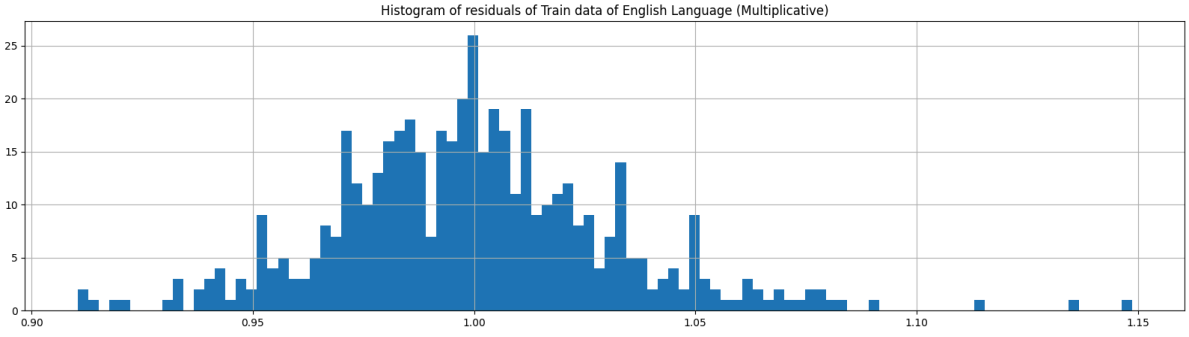
In [239…
```
model_add.resid.mean()
```

Out[239]:
```
np.float64(-0.46609157016702624)
```

In [240…
```
model_multi = sm.tsa.seasonal_decompose(cliped_english_train_df, model='multiplicat
fig = model_multi.plot()
fig.axes[0].set_title("")
fig.suptitle("Seasonal Decomposition of Clicks of Train data of English Language (M
fig.set_size_inches(20, 5)
```



In [241…
```
model_multi.resid.hist(bins=100, figsize=(20,5))
plt.title('Histogram of residuals of Train data of English Language (Multiplicative
plt.show()
```



In [242…
```
model_multi.resid.mean()
```

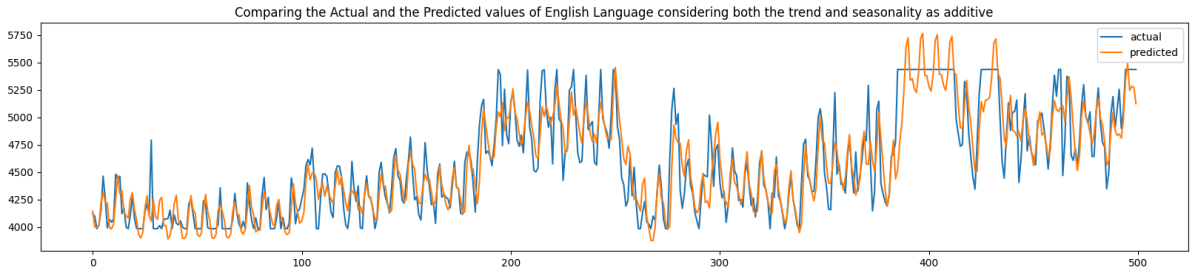
Out[242]:
```
np.float64(0.9997497083183565)
```

In [243…
```
from sklearn.metrics import (mean_absolute_error as mae
                             , mean_squared_error as mse
                             , mean_absolute_percentage_error as mape)

def performance(actual, predicated):
  print(f'MAE : {round(mae(actual, predicated),4)}')
  print(f'MSE : {round(mse(actual, predicated),4)}')
  print(f'MAPE : {round(mape(actual, predicated),4)}')
```

Modelling using the Triple Exponential smoothing (TES)

Modelling using TES (Trend : Additive, Sesasonality : Additive)

In [244...]
```
tes_model_both_additive = sm.tsa.ExponentialSmoothing(cliped_english_train_df.reset
plt.figure(figsize=(20,4))
cliped_english_train_df.reset_index().English_Clicks.plot(label='actual')
tes_model_both_additive.fittedvalues.plot(label='predicted')
plt.title('Comparing the Actual and the Predicted values of English Language consic
plt.legend()
plt.show()
```



In [245...]
```
tes_model_both_additive.fittedvalues.tail(5)
```

Out[245]:

	0
495	5,490
496	5,245
497	5,283
498	5,273
499	5,124

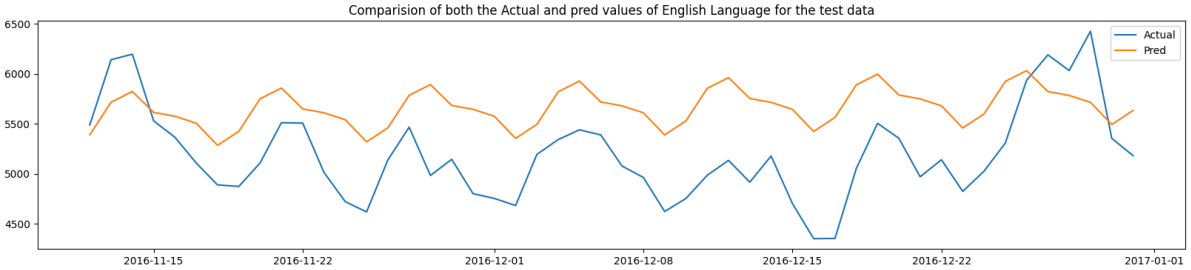
dtype: float64

In [246...]
```
pred = tes_model_both_additive.forecast(steps=50)
english_test_final_df1 = pd.DataFrame({'Actual':english_test_df.to_list(), 'Pred':p
english_test_final_df1.tail(5)
```

Out[246]:

Dates	Actual	Pred
2016-12-27	6,189	5,822
2016-12-28	6,033	5,783
2016-12-29	6,425	5,714
2016-12-30	5,354	5,492
2016-12-31	5,180	5,633

In [247…
```
plt.figure(figsize=(20, 4))  # Create a new figure of desired size
plt.title('Comparision of both the Actual and pred values of English Language for t
plt.plot(english_test_final_df1)  # Plot directly after figure
plt.legend(['Actual', 'Pred'])   # Optional: if it's a DataFrame with columns
plt.show()
```

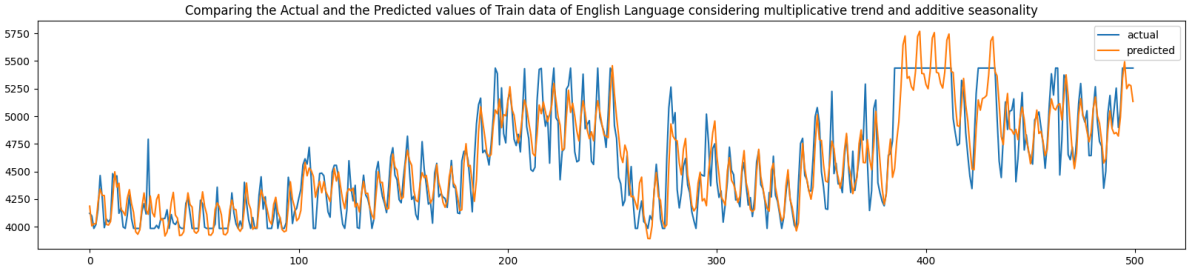


In [248…
```
print('Train Performance Metrics :\n')
performance(cliped_english_train_df.reset_index().English_Clicks,tes_model_both_add
print('\n\n')
print('Test Performance Metrics :\n')
performance(english_test_final_df1.Actual,english_test_final_df1.Pred)
```

```
Train Performance Metrics :

MAE : 149.3091
MSE : 40533.1869
MAPE : 0.0321



Test Performance Metrics :

MAE : 556.6921
MSE : 378119.6854
MAPE : 0.111
```

Modelling using TES (Trend : Multiplicative, Sesasonality : Additive)

In [249…
```
tes_model_t_mul_s_add = sm.tsa.ExponentialSmoothing(cliped_english_train_df.reset_i
plt.figure(figsize=(20,4))
plt.title('Comparing the Actual and the Predicted values of Train data of English L
cliped_english_train_df.reset_index().English_Clicks.plot(label='actual')
tes_model_t_mul_s_add.fittedvalues.plot(label='predicted')
plt.legend()
plt.show()
```



In [250…
```
tes_model_t_mul_s_add.fittedvalues.head()
```

Out[250]:

	0
0	4,185
1	4,007
2	4,024
3	4,028
4	4,222

dtype: float64

In [251…

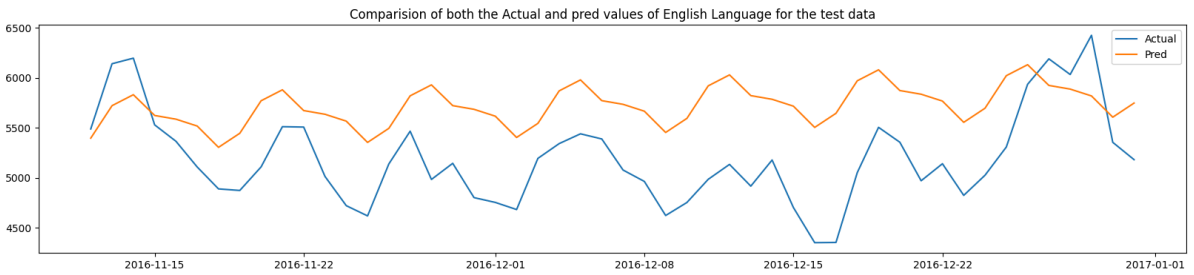
```
pred = tes_model_t_mul_s_add.forecast(steps=50)
english_test_final_df2 = pd.DataFrame({'Actual':english_test_df.to_list(), 'Pred':p
english_test_final_df2.tail(5)
```

Out[251]:

Dates	Actual	Pred
2016-12-27	6,189	5,923
2016-12-28	6,033	5,887
2016-12-29	6,425	5,818
2016-12-30	5,354	5,605
2016-12-31	5,180	5,747

In [252…

```
plt.figure(figsize=(20, 4))  # Create a new figure of desired size
plt.title('Comparision of both the Actual and pred values of English Language for t
plt.plot(english_test_final_df2)  # Plot directly after figure
plt.legend(['Actual', 'Pred'])   # Optional: if it's a DataFrame with columns
plt.show()
```



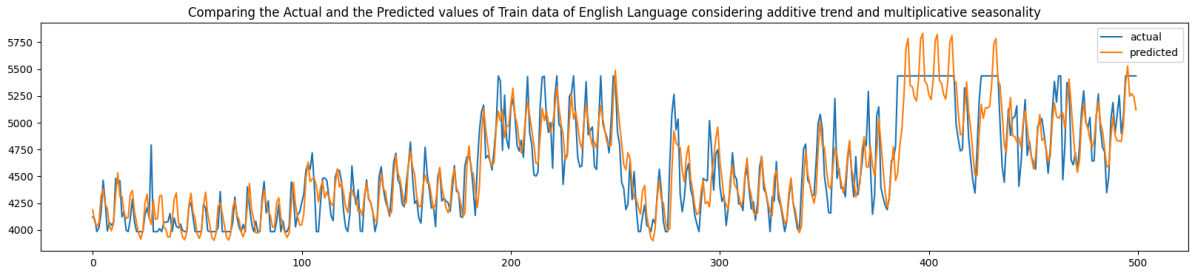
In [253…

```
print('Train Performance Metrics :\n')
performance(cliped_english_train_df.reset_index().English_Clicks,tes_model_t_mul_s_
print('\n\n')
print('Test Performance Metrics :\n')
performance(english_test_final_df2.Actual,english_test_final_df2.Pred)
```

```
Train Performance Metrics :

MAE : 151.0074
MSE : 40938.7038
MAPE : 0.0326



Test Performance Metrics :

MAE : 599.8193
MSE : 437012.4347
MAPE : 0.1198
```

Modelling using TES (Trend : Additive, Sesasonality : Multiplicative)

In [254...
```
tes_model_t_add_s_mul = sm.tsa.ExponentialSmoothing(cliped_english_train_df.reset_i
plt.figure(figsize=(20,4))
plt.title('Comparing the Actual and the Predicted values of Train data of English L
cliped_english_train_df.reset_index().English_Clicks.plot(label='actual')
tes_model_t_add_s_mul.fittedvalues.plot(label='predicted')
plt.legend()
plt.show()
```



In [255...
```
tes_model_t_add_s_mul.fittedvalues.head()
```

Out[255]:

	0
0	4,187
1	4,075
2	4,031
3	4,068
4	4,315

dtype: float64

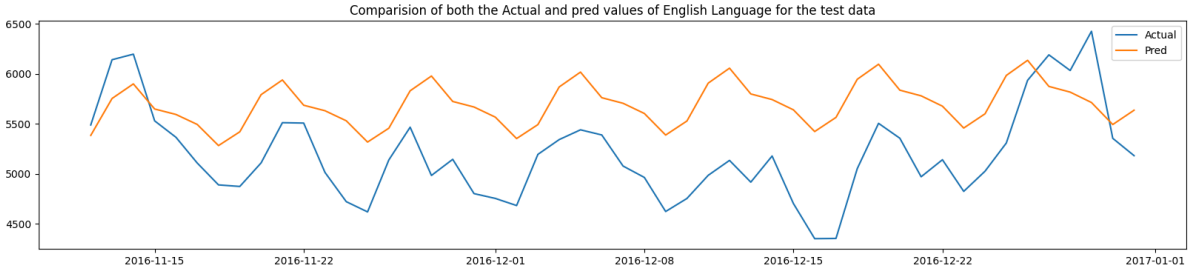
In [256...
```
pred = tes_model_t_add_s_mul.forecast(steps=50)
english_test_final_df3 = pd.DataFrame({'Actual':english_test_df.to_list(), 'Pred':p
english_test_final_df3.tail(5)
```

Out[256]:

Dates	Actual	Pred
2016-12-27	6,189	5,873
2016-12-28	6,033	5,816
2016-12-29	6,425	5,712
2016-12-30	5,354	5,492
2016-12-31	5,180	5,635

In [257…

```
plt.figure(figsize=(20, 4))  # Create a new figure of desired size
plt.title('Comparision of both the Actual and pred values of English Language for t
plt.plot(english_test_final_df3)  # Plot directly after figure
plt.legend(['Actual', 'Pred'])   # Optional: if it's a DataFrame with columns
plt.show()
```



In [258…

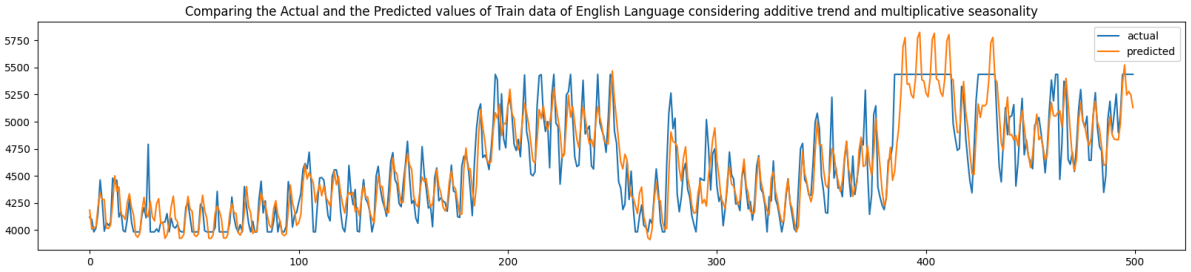
```
print('Train Performance Metrics :\n')
performance(cliped_english_train_df.reset_index().English_Clicks,tes_model_t_add_s_
print('\n\n')
print('Test Performance Metrics :\n')
performance(english_test_final_df3.Actual,english_test_final_df3.Pred)
```

```
Train Performance Metrics :

MAE : 148.396
MSE : 40231.4933
MAPE : 0.0319



Test Performance Metrics :

MAE : 576.4876
MSE : 402304.8536
MAPE : 0.1148
```

Modelling using TES (Trend : Multiplicative, Sesasonality : Multiplicative)

In [259…

```
tes_model_t_mul_s_mul = sm.tsa.ExponentialSmoothing(cliped_english_train_df.reset_i
plt.figure(figsize=(20,4))
plt.title('Comparing the Actual and the Predicted values of Train data of English L
cliped_english_train_df.reset_index().English_Clicks.plot(label='actual')
tes_model_t_mul_s_mul.fittedvalues.plot(label='predicted')
plt.legend()
plt.show()
```



In [260…
```
tes_model_t_mul_s_mul.fittedvalues.head()
```

Out[260]:

	0
0	4,185
1	4,009
2	4,025
3	4,028
4	4,221

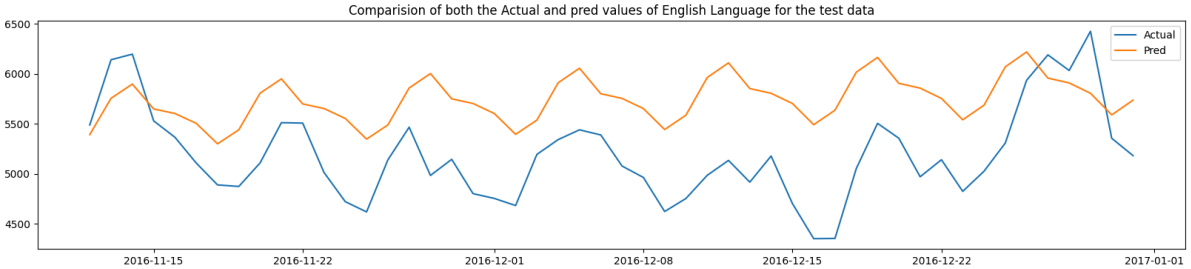
dtype: float64

In [261…
```
pred = tes_model_t_mul_s_mul.forecast(steps=50)
english_test_final_df4 = pd.DataFrame({'Actual':english_test_df.to_list(), 'Pred':p
english_test_final_df4.tail(5)
```

Out[261]:

Dates	Actual	Pred
2016-12-27	6,189	5,956
2016-12-28	6,033	5,908
2016-12-29	6,425	5,804
2016-12-30	5,354	5,588
2016-12-31	5,180	5,736

In [262…
```
plt.figure(figsize=(20, 4))  # Create a new figure of desired size
plt.title('Comparision of both the Actual and pred values of English Language for t
plt.plot(english_test_final_df4)  # Plot directly after figure
plt.legend(['Actual', 'Pred'])   # Optional: if it's a DataFrame with columns
plt.show()
```



In [263…
```
print('Train Performance Metrics :\n')
performance(cliped_english_train_df.reset_index().English_Clicks,tes_model_t_mul_s_
print('\n\n')
```

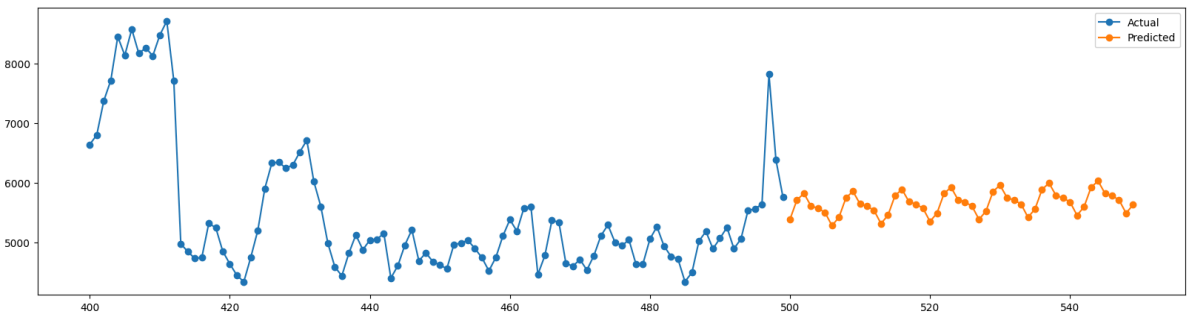
```
print('Test Performance Metrics :\n')
performance(english_test_final_df4.Actual,english_test_final_df4.Pred)
```

```
Train Performance Metrics :

MAE : 151.4279
MSE : 40891.9854
MAPE : 0.0326



Test Performance Metrics :

MAE : 613.0552
MSE : 454758.3768
MAPE : 0.1223
```

Final Model for forcasting the future clicks for the pages of English Language using TES

In [264…

```
pred = tes_model_both_additive.forecast(steps=50)
plt.figure(figsize=(20,5))
plt.plot(english_train_df.reset_index().English_Clicks.tail(100), '-o', label='Actu
plt.plot(pred, '-o', label='Predicted')
plt.legend()
plt.show()
```



Modelling using ARIMA Family of Modelling Techniques

In [415…

```
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.statespace.sarimax import SARIMAX

class Arima_PipeLine:

  def __init__(self, data, forecast_steps=20, exog_data=None):
    self.df = data
    self.exog_data = exog_data
    self.df_grpd = self.preprocess_data()
    self.train = self.df_grpd[:-forecast_steps]
    self.train_pred = None
    print(Fore.GREEN, '*'*50, '\n', f'Train Samples : {len(self.train)}',
          '\n','*'*50, '\n')
    self.test = self.df_grpd[-forecast_steps:]
    print(Fore.GREEN, '*'*50, '\n', f'Test Samples : {len(self.test)}',
          '\n','*'*50)
    self.exog_train = None
    self.exog_test = None
```

```
    self.train_clip = self.train.clip(self.train.quantile(0.10), upper = self.train
    self.forecast_steps = forecast_steps
    self.stationarity = False
    self.forecast_arima = None
    self.forecast_sarima = None
    self.final_order = None
    self.seasonal_order = None
    self.final_order_score = None
    self.Language = None
    self.stationary_data = None
    self.forecast_sarimax = None

  def get_language_info(self):
    pattern = r'(\w+)_Clicks'
    regex = re.compile(pattern)
    search_obj = regex.search(self.df_grpd.name)
    self.Language = search_obj.group(1)
    print(f'\n{Fore.GREEN}', '*'*50, '\n',f'Language : {self.Language}',
          '\n','*'*50)


  def preprocess_data(self):
    self.df.fillna(0, inplace=True)
    grpd_df = self.df.groupby(by='Dates')[self.df.columns[1]].agg('mean')
    grpd_df.sort_index(ascending=True, inplace=True)

    return grpd_df

  def preprocess_exog(self):
    if self.exog_data is not None and not self.exog_data.empty:
      self.exog_train = self.exog_data['Exog'].values[:-50]
      self.exog_test = self.exog_data['Exog'].values[-50:]
      self.exog_train = pd.Series(self.exog_train, index=self.train.index)
      self.exog_test = pd.Series(self.exog_test, index=self.test.index)

  def de_trend_seasonality(self):
    plt.figure(figsize=(20,5))
    self.stationary_data = self.train_clip.diff(1).diff(7)
    plt.title(f'{self.Language} : Stationary time series')
    plt.plot(self.stationary_data)
    plt.show()

  def stationatiy_check(self):
    pvalue = sm.tsa.stattools.adfuller(self.stationary_data.dropna())[1]
    if pvalue <= 0.05:
      self.stationarity = True
      print(f'\n{Fore.GREEN}', '*'*50,'\n',f'Stationarity : {self.stationarity}\n',
    else:
      self.stationarity = False
      print(f'\n{Fore.RED}', '*'*50,'\n', f'Stationarity : {self.stationarity}\n',

  def acf_and_pacf(self):
    fig, ax = plt.subplots(figsize=(20, 5))
    plot_acf(self.stationary_data.dropna(),ax=ax);
    plt.title(f'{self.Language} : ACF Plot - used for finding MA(q)')
    plt.show()

    fig, ax = plt.subplots(figsize=(20, 5))
    plot_pacf(self.stationary_data.dropna(),ax=ax)
    plt.title(f'{self.Language} : PACF Plot - used for finding AR(p)')
    plt.show()

  def grid_search(self , p_values=[0, 1, 2], d_values=[0, 1, 2], q_values=[0, 1, 2]
    self.best_score = float("inf")
```

```
    self.best_order = None

    for p in p_values:
        for d in d_values:
            for q in q_values:
              try:
                if model_name == 'SARIMA':
                  model = SARIMAX(self.train_clip, order=self.final_order, seasonal
                elif model_name == 'SARIMAX':
                  model = SARIMAX(self.train_clip, order=self.final_order, seasonal
                else:
                  model = ARIMA(self.train_clip, order=(p, d, q))
                model_fit = model.fit()
                forecast = model_fit.forecast(steps=self.forecast_steps, exog=self.

                  # MAPE calculation
                mape = np.mean(np.abs((self.test.values - forecast) / self.test.val

                if mape < self.best_score:
                  self.best_score = mape
                  self.best_order = (p, d, q)

              except Exception as e:
                print(f"Skipped order ({p},{d},{q}) due to error: {e}")
                continue

    if (model_name == 'SARIMA') or (model_name == 'SARIMAX'):
      self.seasonal_order = self.best_order
      self.seasonal_order = list(self.seasonal_order)
      self.seasonal_order.append(7)
    else:
      self.final_order = self.best_order

    self.final_order_score = self.best_score
    print(f'{Fore.GREEN}{"*" * 50}')
    print(f'Best order for {model_name} Model is {Fore.BLUE} : {self.final_order} {
    if (model_name == 'SARIMA') or (model_name == 'SARIMAX'):
      print(f'Best Seasonal order for {model_name} Model is {Fore.BLUE} : {tuple(se
    print(f'Best Score for {model_name} Model is {Fore.BLUE} : {round(self.final_or
    print(f'{Fore.GREEN}{"*" * 50}')

  def build_model(self, _exog=False):

    if _exog == True:
      model_dict = {'SARIMAX':[SARIMAX, self.forecast_sarimax]}
    else:
      model_dict = {'ARIMA':[ARIMA, self.forecast_arima],
                    'SARIMA':[SARIMAX, self.forecast_sarima]}

    for _model, m_obj in model_dict.items():

      if _model == 'ARIMA':
        model = ARIMA(self.train_clip, order=self.final_order)
      else :
        if _exog == True:
          self.grid_search(model_name='SARIMAX')
          model = SARIMAX(self.train_clip, order=self.final_order, seasonal_order=s
        else:
          self.grid_search(model_name='SARIMA')
          model = SARIMAX(self.train_clip, order=self.final_order, seasonal_order=s


      fitted = model.fit()
```

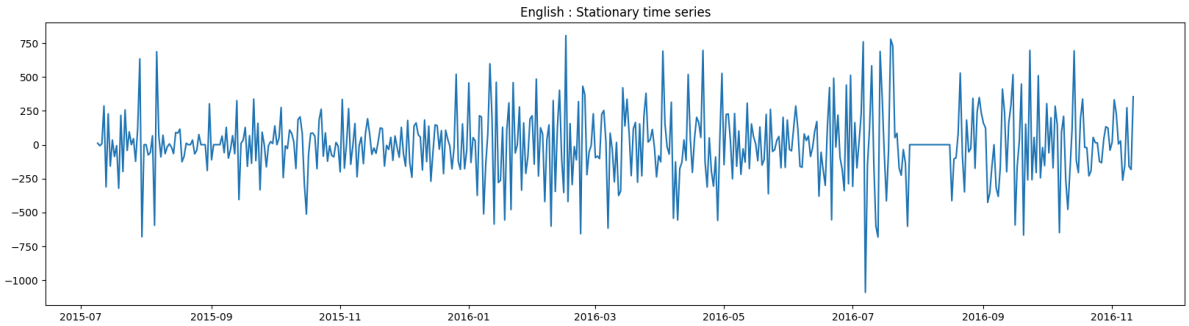
```
      self.train_pred = fitted.predict(start=0, end=len(self.train)-1)

      if fitted:
        print(f'\n{Fore.GREEN}{"*" *50}\n{_model} Model fit Successful\n')

      print(f'Forecasting next {self.forecast_steps} steps\n{"*" *50}')
      if _exog == True:
        forecast_obj = fitted.get_forecast(steps=self.forecast_steps, exog=self.exo
      else:
        forecast_obj = fitted.get_forecast(steps=self.forecast_steps)
      fc = forecast_obj.predicted_mean
      fc_model = pd.Series(fc, index=self.test.index)
      conf = forecast_obj.conf_int(alpha=0.02)

      plt.figure(figsize=(20,5), dpi=100)
      plt.plot(self.train, label='training')
      plt.plot(self.test, label='actual')
      plt.plot(fc_model, label='forecast')

      plt.title(f'{self.Language} Language Forecast vs Actual using {_model}')
      plt.legend(loc='upper left', fontsize=8)
      plt.show()

      m_obj = fc_model

      self.performance(_model, m_obj)

  def calculate_scores(self, data1, data2):
    print(f'MAE : {round(mae(data1, data2),4)}')
    print(f'MSE : {round(mse(data1, data2),4)}')
    print(f'MAPE : {round(mape(data1, data2),4)}')

  def performance(self, model, model_fc):
    print('\n')
    print(f'{Fore.GREEN}{"*" * 50}')
    print(f'Language : {self.Language}\n')
    print(f'Model Type : {model}')
    print(f'Training Performance : {Fore.BLUE}')
    self.calculate_scores(self.train, self.train_pred)
    print(f'{Fore.GREEN}Testing Performance : {Fore.BLUE}')
    self.calculate_scores(self.test, model_fc)
    print(f'{Fore.GREEN}{"*" * 50}')

  def get_forecast(self):
    self.get_language_info()
    self.de_trend_seasonality()
    self.stationatiy_check()
    if self.stationarity == True:
      self.acf_and_pacf()
      self.grid_search()
      self.build_model()
      if self.exog_data is not None and not self.exog_data.empty:
        self.preprocess_exog()
        self.build_model(_exog= True)

    else:
      print(Fore.RED , 'Stopping Modelling...')
      print(Fore.RED , 'Automation failed')
      print(Fore.RED ,  'Try Manually')
```

English Language

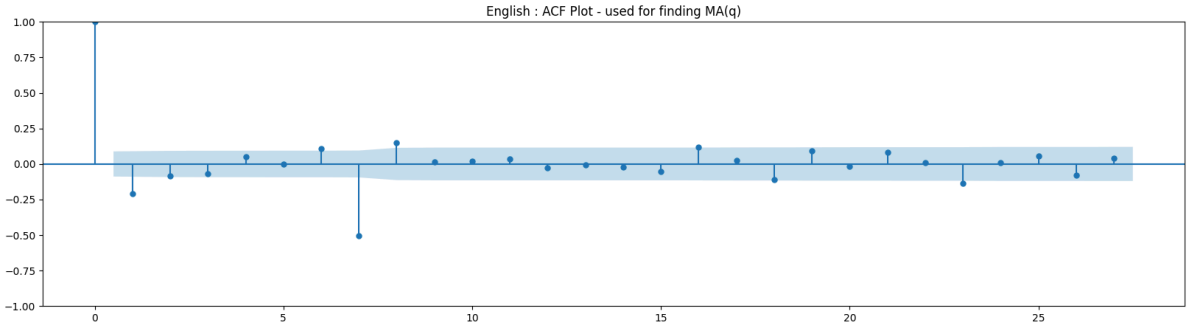
In [416…

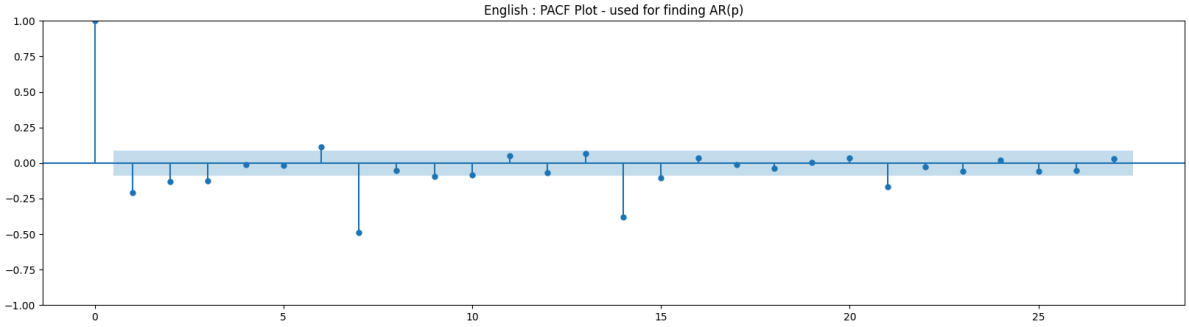
```
with open('/content/drive/MyDrive/Scaler_DSML_Digital_Notes/BusinessCases/11_AdEase
  english_df = pickle.load(fp)

exog_path = '/content/drive/MyDrive/Scaler_DSML_Digital_Notes/BusinessCases/11_AdEa
exog_df = pd.read_csv(exog_path)
arima_english = Arima_PipeLine(english_df, forecast_steps=50, exog_data=exog_df)
arima_english.get_forecast()
```

```
**************************************************
Train Samples : 500
**************************************************

**************************************************
Test Samples : 50
**************************************************

**************************************************
Language : English
**************************************************
```

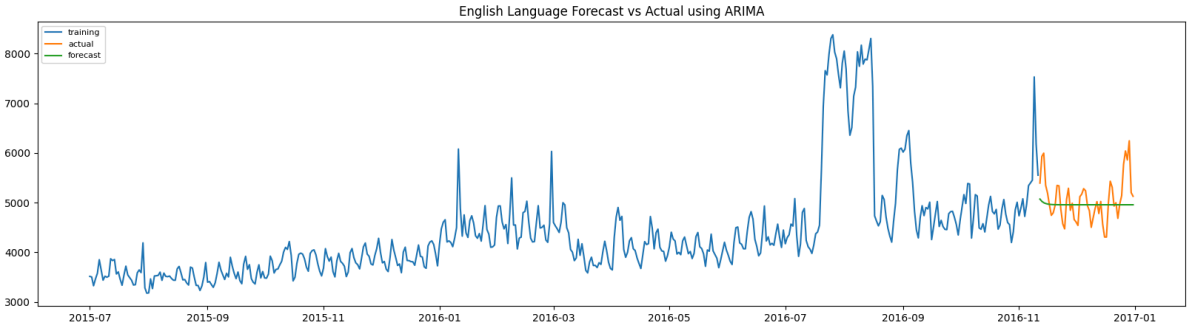



```
**************************************************
Stationarity : True
**************************************************
```

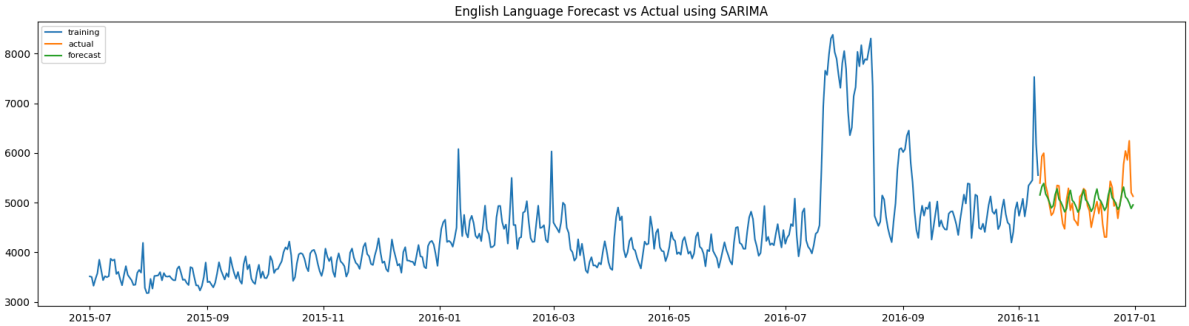




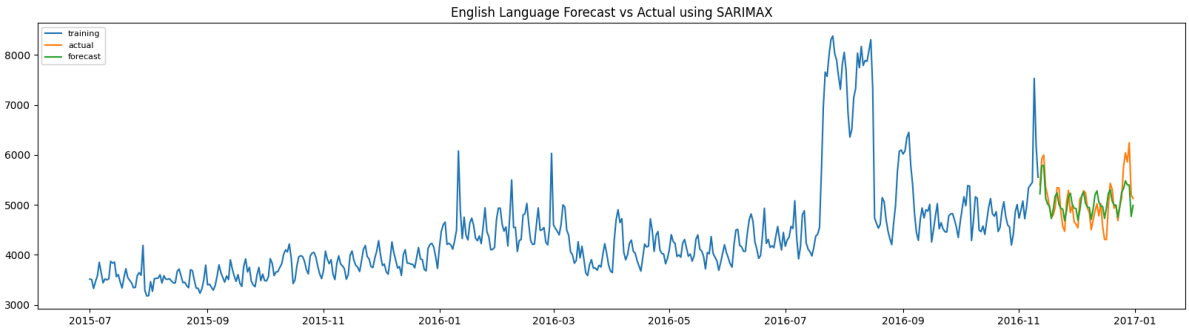
```
************************************************
Best order for ARIMA Model is  : (1, 1, 1)
Best Score for ARIMA Model is  : 6.288
************************************************

************************************************
ARIMA Model fit Successful

Forecasting next 50 steps
************************************************
```




```
************************************************
Language : English

Model Type : ARIMA
Training Performance :
MAE : 350.2584
MSE : 502070.0562
MAPE : 0.066
Testing Performance :
MAE : 327.9523
MSE : 196690.8382
MAPE : 0.0629
************************************************
************************************************
Best order for SARIMA Model is  : (1, 1, 1)
Best Seasonal order for SARIMA Model is  : (2, 1, 2, 7)
Best Score for SARIMA Model is  : 5.26
************************************************

************************************************
SARIMA Model fit Successful

Forecasting next 50 steps
************************************************
```



```
************************************************
Language : English

Model Type : SARIMA
Training Performance :
MAE : 315.9012
MSE : 489478.157
MAPE : 0.0584
Testing Performance :
MAE : 271.2947
MSE : 131824.3361
MAPE : 0.0526
************************************************
************************************************
Best order for SARIMAX Model is  : (1, 1, 1)
Best Seasonal order for SARIMAX Model is  : (0, 2, 2, 7)
Best Score for SARIMAX Model is  : 4.308
************************************************

************************************************
SARIMAX Model fit Successful

Forecasting next 50 steps
************************************************
```




```
************************************************
Language : English

Model Type : SARIMAX
Training Performance :
MAE : 313.1243
MSE : 493088.353
MAPE : 0.0587
Testing Performance :
MAE : 218.729
MSE : 78223.5218
MAPE : 0.0431
************************************************
```

Chinese Language

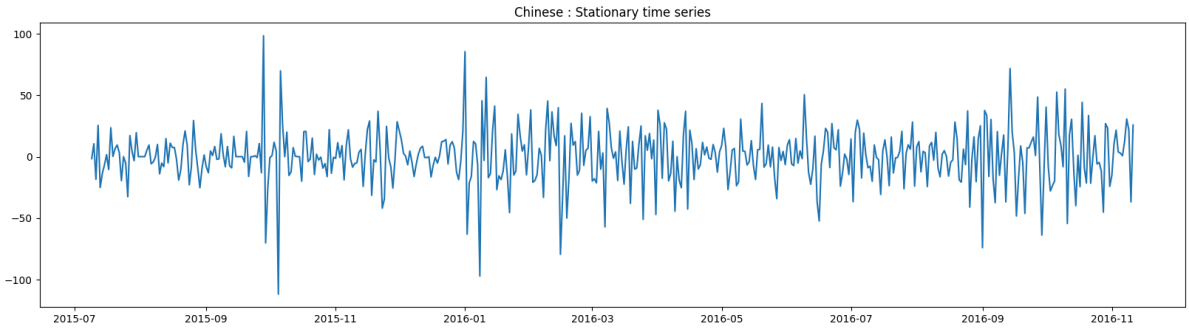
In [417…

```
with open('/content/drive/MyDrive/Scaler_DSML_Digital_Notes/BusinessCases/11_AdEase
  chinese_df = pickle.load(fp)

arima_chinese = Arima_PipeLine(chinese_df, 50)
arima_chinese.get_forecast()
```

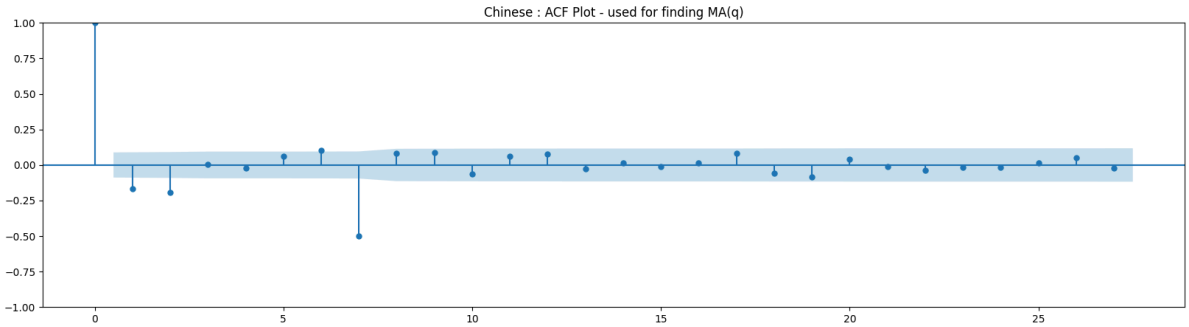
****
Train Samples : 500
****

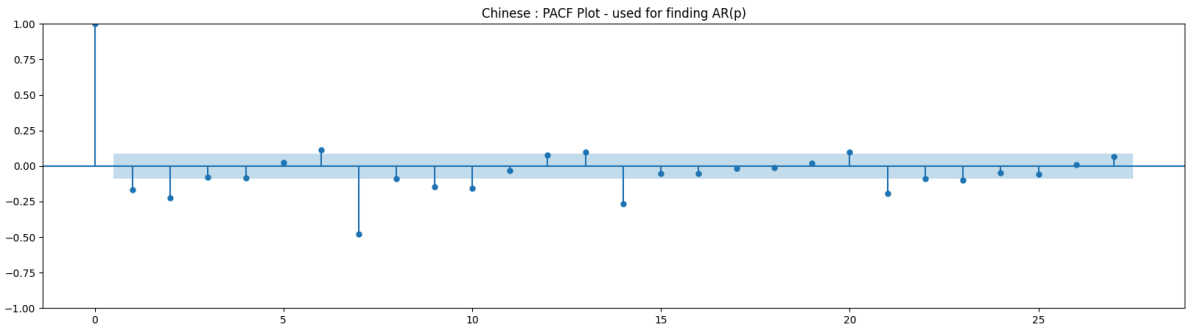
****
Test Samples : 50
****

****
Language : Chinese
****



****
Stationarity : True
****

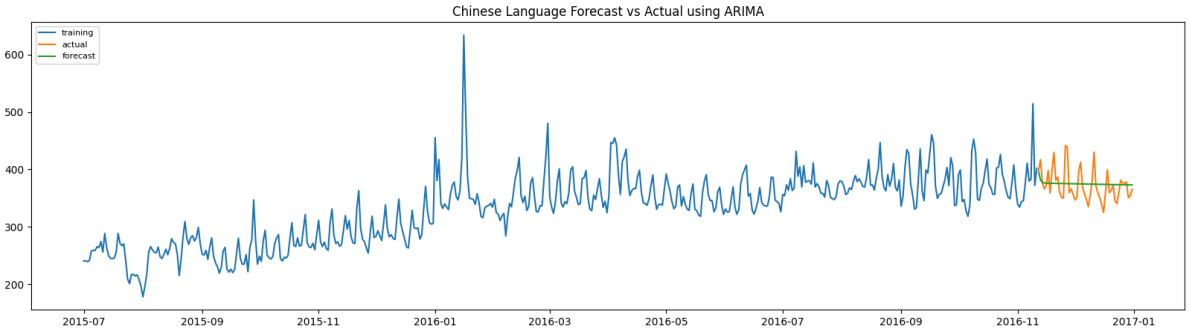




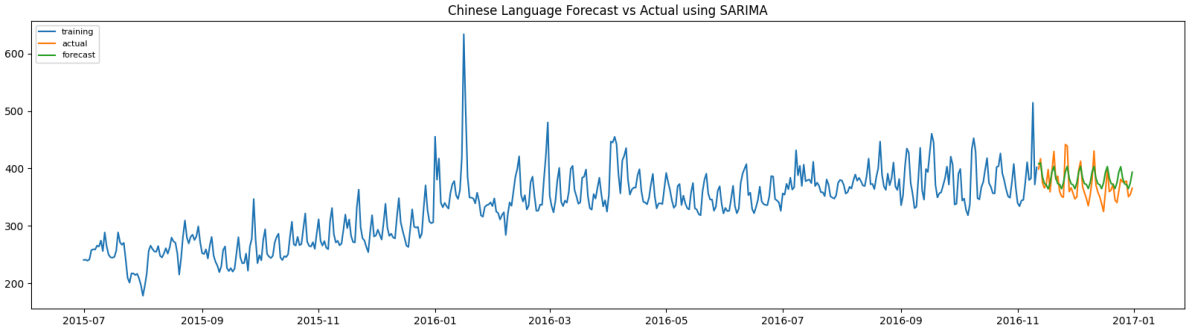
****
Best order for ARIMA Model is : (2, 0, 2)
Best Score for ARIMA Model is : 5.531
****

****
ARIMA Model fit Successful

Forecasting next 50 steps
****




```
**************************************************
Language : Chinese

Model Type : ARIMA
Training Performance :
MAE : 19.7501
MSE : 845.4933
MAPE : 0.0595
Testing Performance :
MAE : 20.7885
MSE : 689.3554
MAPE : 0.0553
**************************************************
**************************************************
Best order for SARIMA Model is  : (2, 0, 2)
Best Seasonal order for SARIMA Model is  : (2, 0, 2, 7)
Best Score for SARIMA Model is  : 4.856
**************************************************

**************************************************
SARIMA Model fit Successful

Forecasting next 50 steps
**************************************************
```



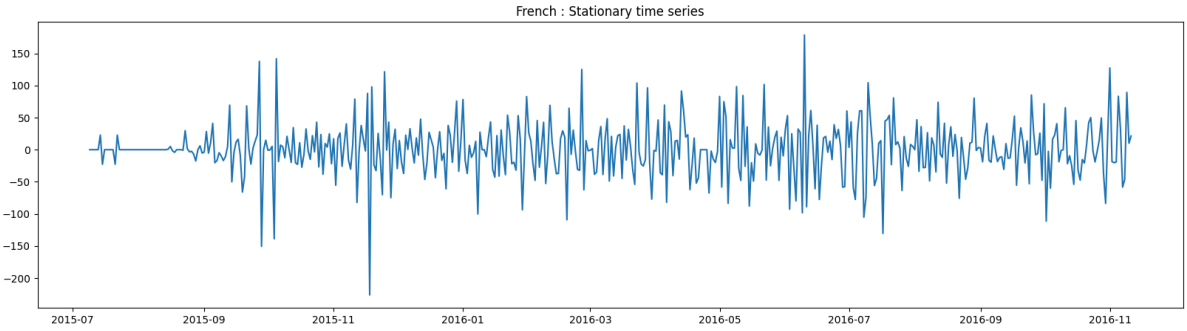

```
**************************************************
Language : Chinese

Model Type : SARIMA
Training Performance :
MAE : 16.3244
MSE : 771.0708
MAPE : 0.0498
Testing Performance :
MAE : 17.8722
MSE : 452.5685
MAPE : 0.0486
**************************************************
```

French Language

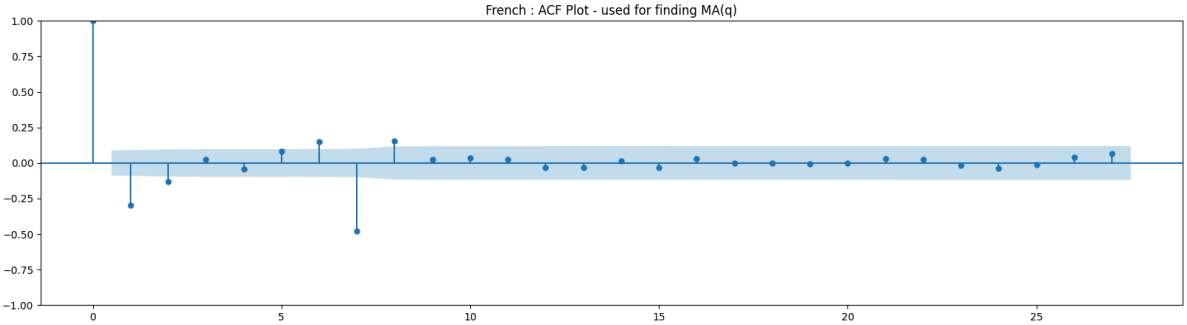
In [418…

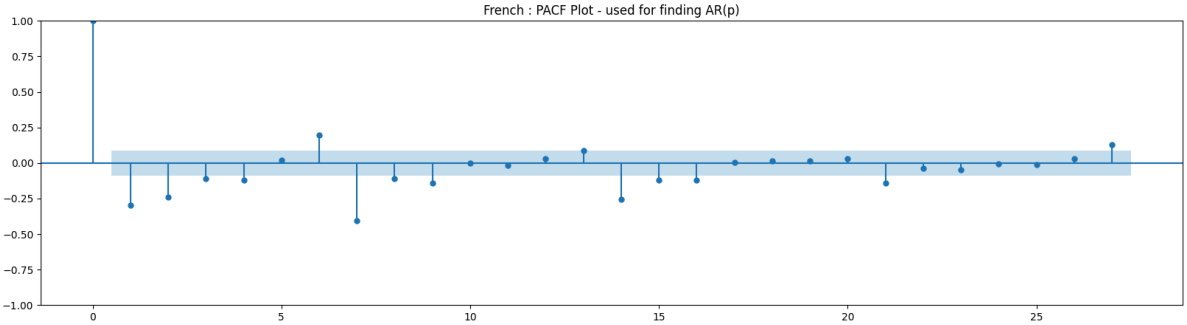
```
with open('/content/drive/MyDrive/Scaler_DSML_Digital_Notes/BusinessCases/11_AdEase
  french_df = pickle.load(fp)

arima_french = Arima_PipeLine(french_df, 50)
arima_french.get_forecast()
```

```
**************************************************
Train Samples : 500
**************************************************

**************************************************
Test Samples : 50
**************************************************

**************************************************
Language : French
**************************************************
```

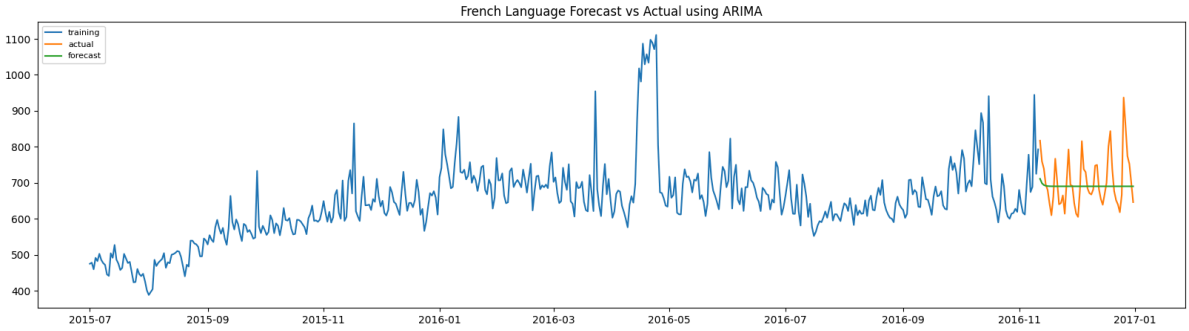



```
**************************************************
Stationarity : True
**************************************************
```






```
**************************************************
Best order for ARIMA Model is  : (1, 1, 1)
Best Score for ARIMA Model is  : 7.228
**************************************************

**************************************************
ARIMA Model fit Successful

Forecasting next 50 steps
**************************************************
```

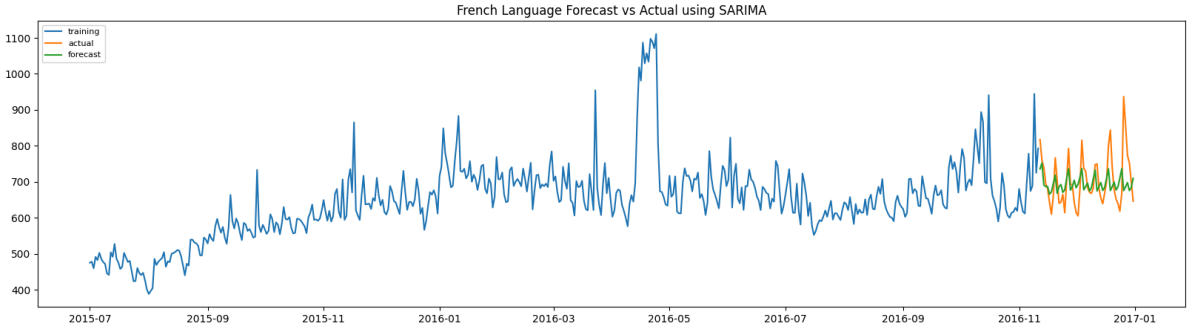


****
Language : French

Model Type : ARIMA
Training Performance :
MAE : 41.7612
MSE : 5239.7775
MAPE : 0.0609
Testing Performance :
MAE : 52.958
MSE : 5034.8212
MAPE : 0.0723
****
****
Best order for SARIMA Model is : (1, 1, 1)
Best Seasonal order for SARIMA Model is : (2, 2, 1, 7)
Best Score for SARIMA Model is : 6.862
****

****
SARIMA Model fit Successful

Forecasting next 50 steps
****



****
Language : French

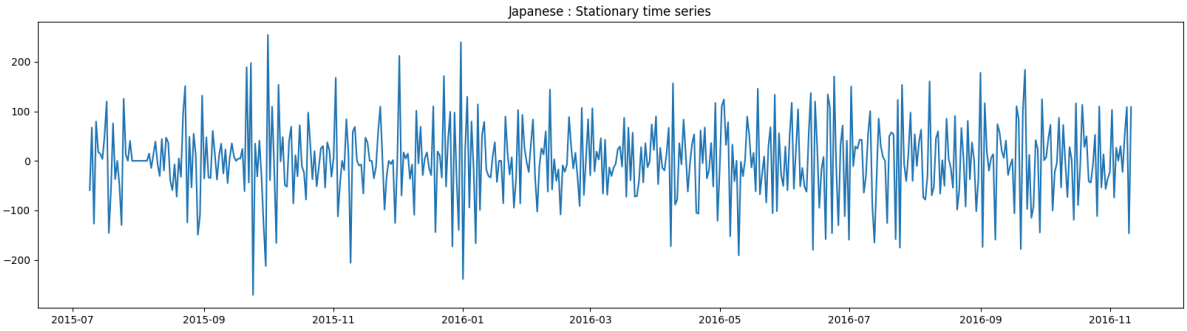
Model Type : SARIMA
Training Performance :
MAE : 39.9052
MSE : 5593.5413
MAPE : 0.059
Testing Performance :
MAE : 50.0245
MSE : 4715.0599
MAPE : 0.0686
****

Japanese Language

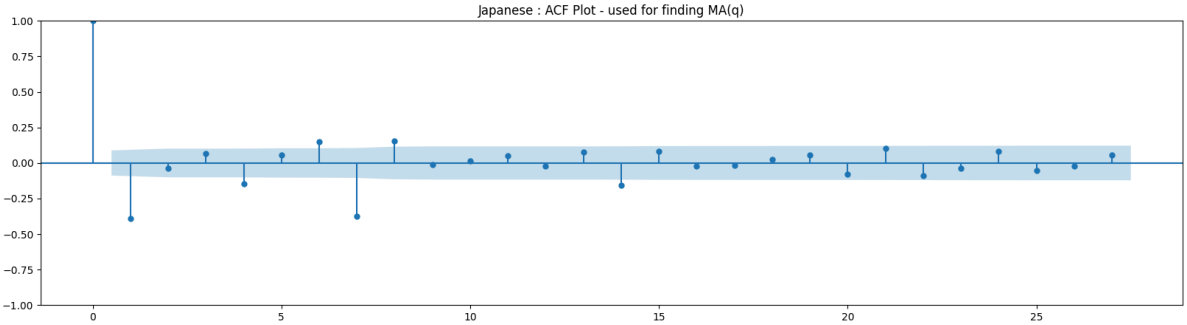
In [419…

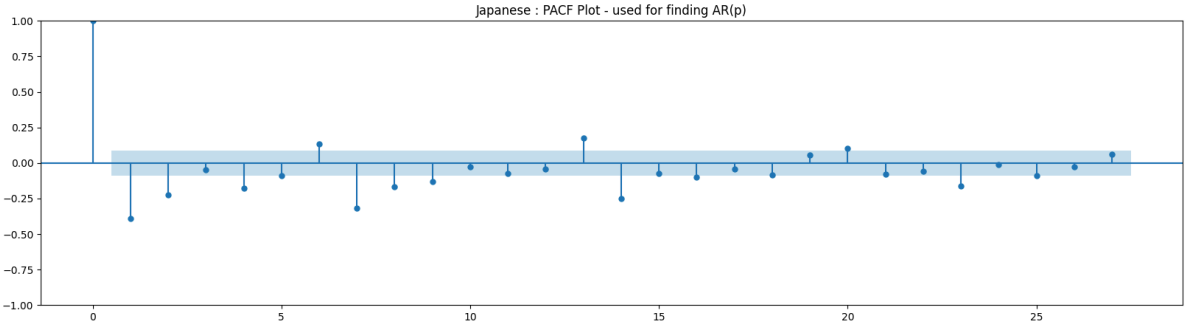
```
with open('/content/drive/MyDrive/Scaler_DSML_Digital_Notes/BusinessCases/11_AdEase
  japanese_df = pickle.load(fp)

arima_japanese = Arima_PipeLine(japanese_df, 50)
arima_japanese.get_forecast()
```

```
**************************************************
Train Samples : 500
**************************************************

**************************************************
Test Samples : 50
**************************************************

**************************************************
Language : Japanese
**************************************************
```

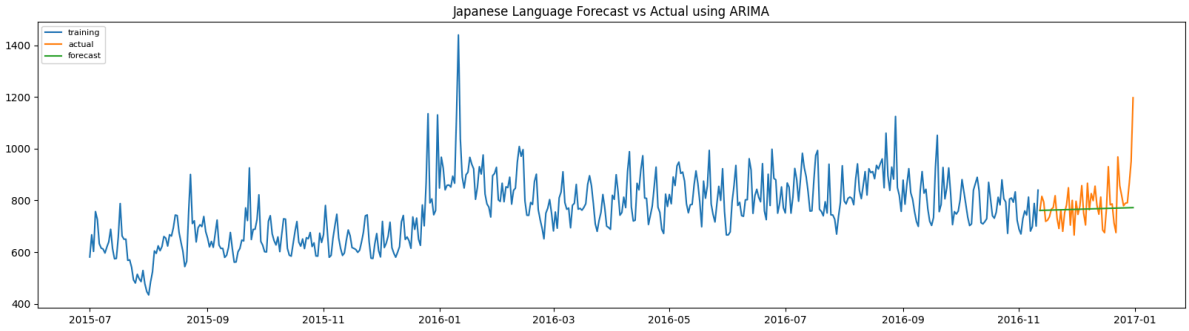



```
**************************************************
Stationarity : True
**************************************************
```

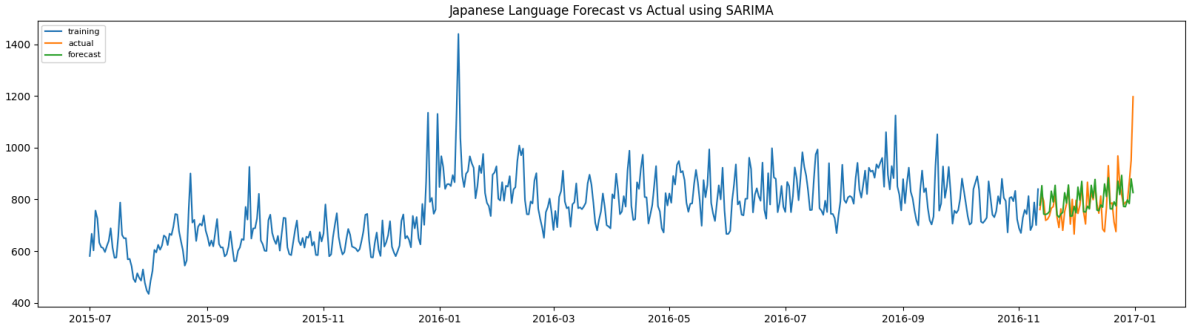





```
**************************************************
Best order for ARIMA Model is  : (0, 2, 2)
Best Score for ARIMA Model is  : 7.057
**************************************************

**************************************************
ARIMA Model fit Successful

Forecasting next 50 steps
**************************************************
```




```
**************************************************
Language : Japanese

Model Type : ARIMA
Training Performance :
MAE : 64.5278
MSE : 7635.3312
MAPE : 0.0854
Testing Performance :
MAE : 59.0251
MSE : 8238.3968
MAPE : 0.0706
**************************************************
**************************************************
Best order for SARIMA Model is  : (0, 2, 2)
Best Seasonal order for SARIMA Model is  : (2, 2, 1, 7)
Best Score for SARIMA Model is  : 6.198
**************************************************

**************************************************
SARIMA Model fit Successful

Forecasting next 50 steps
**************************************************
```



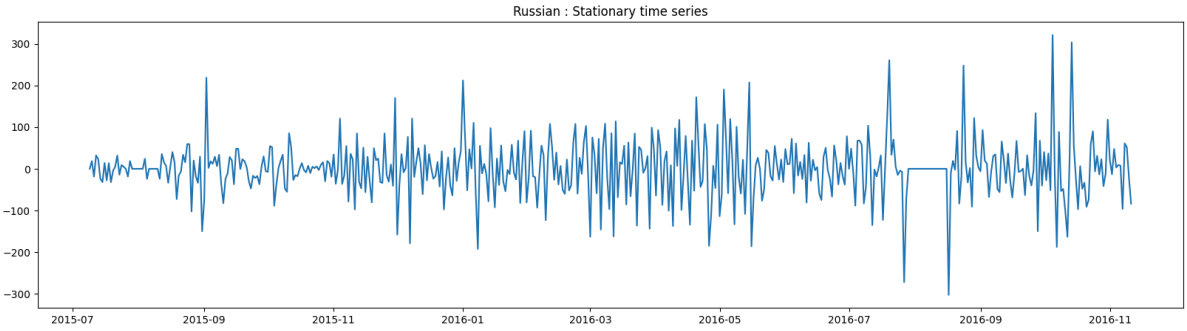

```
**************************************************
Language : Japanese

Model Type : SARIMA
Training Performance :
MAE : 62.4886
MSE : 8174.9262
MAPE : 0.0837
Testing Performance :
MAE : 50.0583
MSE : 5928.38
MAPE : 0.062
**************************************************
```

Russian Language

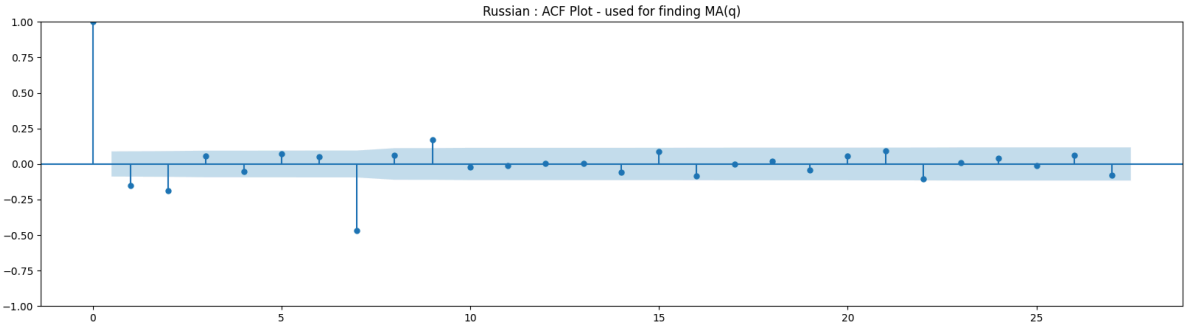
In [420…

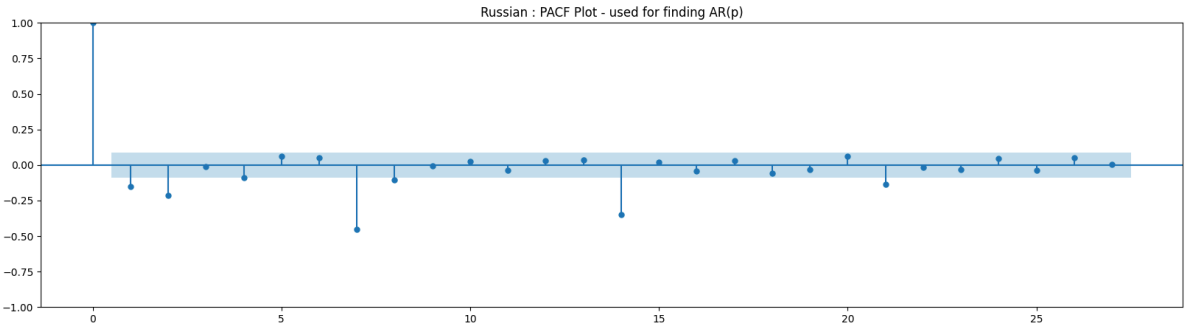
```
with open('/content/drive/MyDrive/Scaler_DSML_Digital_Notes/BusinessCases/11_AdEase
  russian_df = pickle.load(fp)

arima_russian = Arima_PipeLine(russian_df, 50)
arima_russian.get_forecast()
```

```
************************************************
Train Samples : 500
************************************************

************************************************
Test Samples : 50
************************************************

************************************************
Language : Russian
************************************************
```

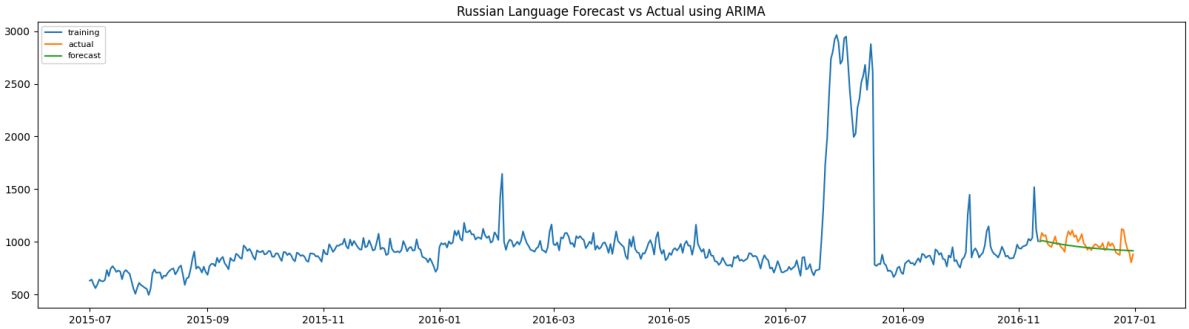



```
************************************************
Stationarity : True
************************************************
```

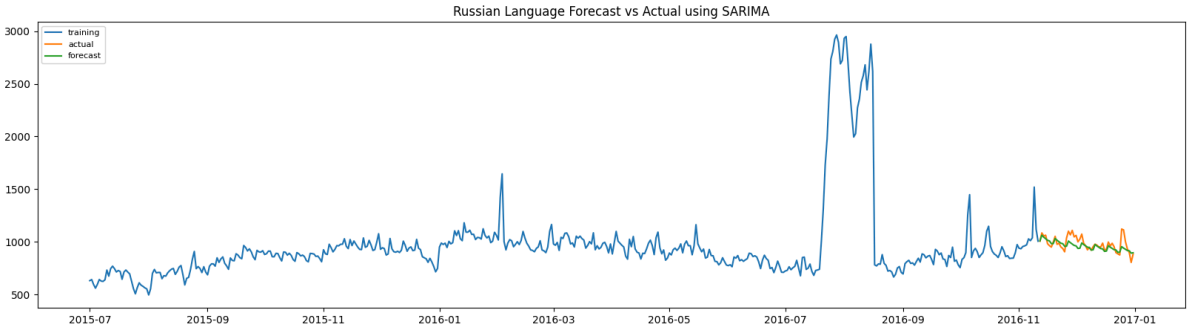





```
************************************************
Best order for ARIMA Model is  : (2, 0, 2)
Best Score for ARIMA Model is  : 5.197
************************************************

************************************************
ARIMA Model fit Successful

Forecasting next 50 steps
************************************************
```




```
**************************************************
Language : Russian

Model Type : ARIMA
Training Performance :
MAE : 125.1416
MSE : 124907.9038
MAPE : 0.0859
Testing Performance :
MAE : 52.5956
MSE : 4899.5789
MAPE : 0.052
**************************************************
Skipped order (2,2,1) due to error: LU decomposition error.
**************************************************
Best order for SARIMA Model is  : (2, 0, 2)
Best Seasonal order for SARIMA Model is  : (0, 1, 1, 7)
Best Score for SARIMA Model is  : 4.194
**************************************************

**************************************************
SARIMA Model fit Successful

Forecasting next 50 steps
**************************************************
```



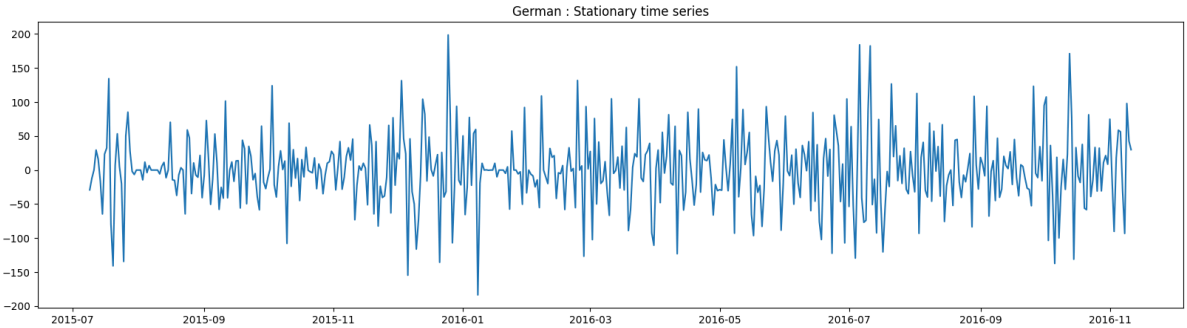

```
**************************************************
Language : Russian

Model Type : SARIMA
Training Performance :
MAE : 127.5656
MSE : 128624.8678
MAPE : 0.0917
Testing Performance :
MAE : 42.4257
MSE : 3405.1473
MAPE : 0.0419
**************************************************
```

German Language

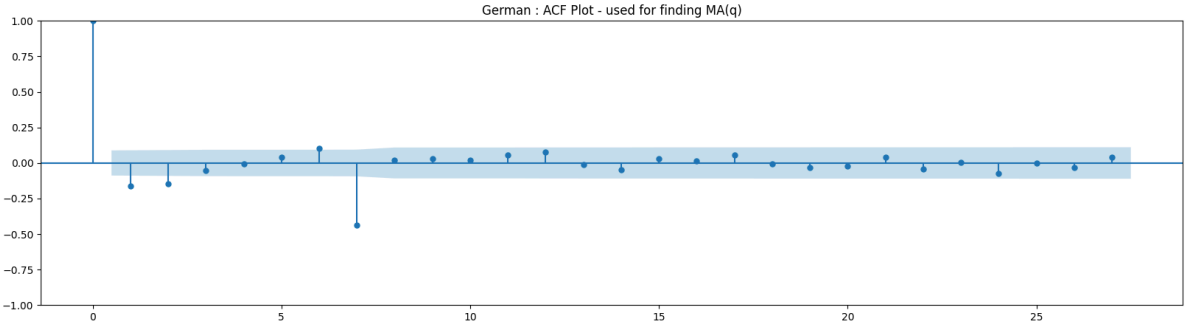
In [421…

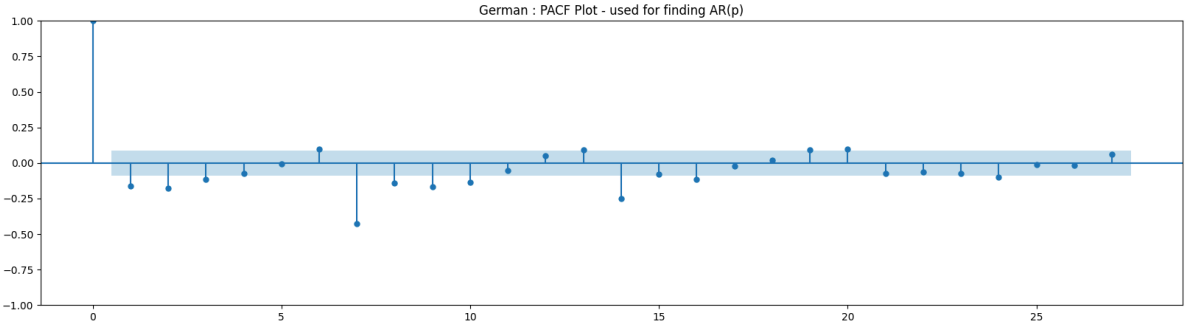
```
with open('/content/drive/MyDrive/Scaler_DSML_Digital_Notes/BusinessCases/11_AdEase
  german_df = pickle.load(fp)

arima_german = Arima_PipeLine(german_df, 50)
arima_german.get_forecast()
```

```
************************************************
Train Samples : 500
************************************************

************************************************
Test Samples : 50
************************************************

************************************************
Language : German
************************************************
```

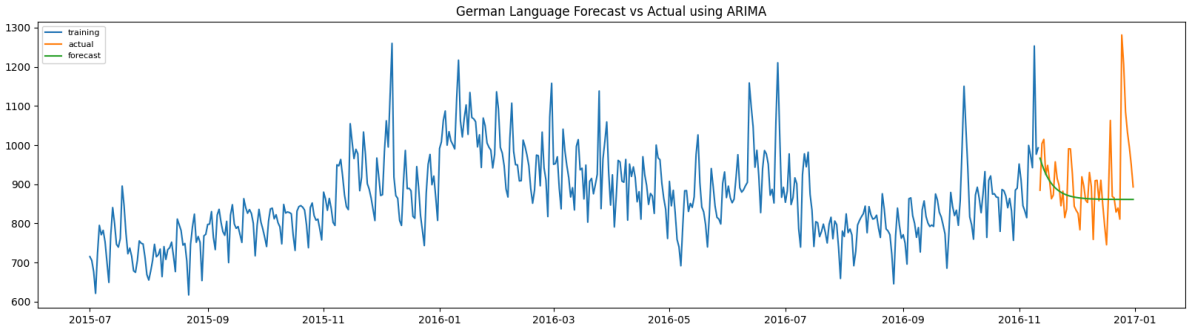



```
************************************************
Stationarity : True
************************************************
```

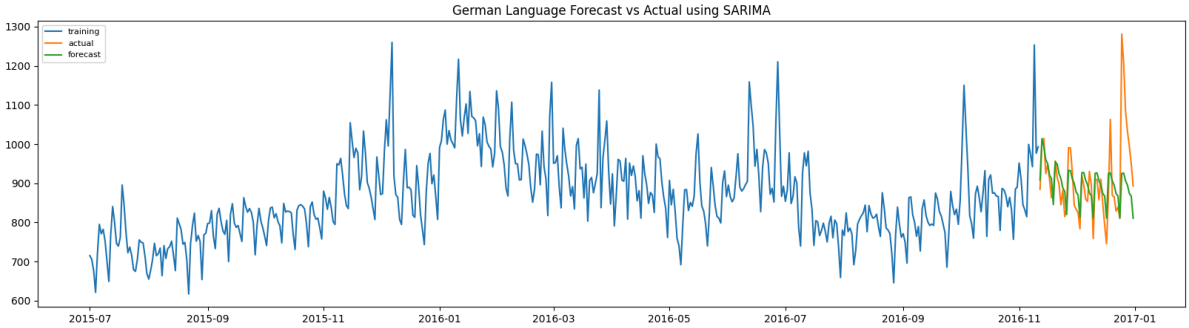





```
************************************************
Best order for ARIMA Model is  : (1, 0, 1)
Best Score for ARIMA Model is  : 6.999
************************************************

************************************************
ARIMA Model fit Successful

Forecasting next 50 steps
************************************************
```




```
************************************************
Language : German

Model Type : ARIMA
Training Performance :
MAE : 53.1575
MSE : 5121.8538
MAPE : 0.0606
Testing Performance :
MAE : 68.3732
MSE : 11289.0248
MAPE : 0.07
************************************************
************************************************
Best order for SARIMA Model is  : (1, 0, 1)
Best Seasonal order for SARIMA Model is  : (1, 1, 1, 7)
Best Score for SARIMA Model is  : 5.781
************************************************

************************************************
SARIMA Model fit Successful

Forecasting next 50 steps
************************************************
```




```
************************************************
Language : German

Model Type : SARIMA
Training Performance :
MAE : 49.9297
MSE : 10396.3025
MAPE : 0.059
Testing Performance :
MAE : 55.9529
MSE : 7357.9905
MAPE : 0.0578
************************************************
```

Modelling using FB Prophet

In [54]:
```
!pip install pystan
!pip install prophet
```

```
Collecting pystan
  Downloading pystan-3.10.0-py3-none-any.whl.metadata (3.7 kB)
Requirement already satisfied: aiohttp<4.0,>=3.6 in /usr/local/lib/python3.11/dist
-packages (from pystan) (3.11.15)
Collecting clikit<0.7,>=0.6 (from pystan)
  Downloading clikit-0.6.2-py2.py3-none-any.whl.metadata (1.6 kB)
Collecting httpstan<4.14,>=4.13 (from pystan)
  Downloading httpstan-4.13.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_
64.whl.metadata (6.2 kB)
Requirement already satisfied: numpy>=1.19 in /usr/local/lib/python3.11/dist-packa
ges (from pystan) (2.0.2)
Collecting pysimdjson<7,>=5.0.2 (from pystan)
  Downloading pysimdjson-6.0.2-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86
_64.whl.metadata (1.9 kB)
Requirement already satisfied: setuptools in /usr/local/lib/python3.11/dist-packag
es (from pystan) (75.2.0)
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in /usr/local/lib/python3.1
1/dist-packages (from aiohttp<4.0,>=3.6->pystan) (2.6.1)
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.11/dist-
packages (from aiohttp<4.0,>=3.6->pystan) (1.3.2)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.11/dist-pac
kages (from aiohttp<4.0,>=3.6->pystan) (25.3.0)
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.11/dist
-packages (from aiohttp<4.0,>=3.6->pystan) (1.5.0)
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.11/di
st-packages (from aiohttp<4.0,>=3.6->pystan) (6.4.3)
Requirement already satisfied: propcache>=0.2.0 in /usr/local/lib/python3.11/dist-
packages (from aiohttp<4.0,>=3.6->pystan) (0.3.1)
Requirement already satisfied: yarl<2.0,>=1.17.0 in /usr/local/lib/python3.11/dist
-packages (from aiohttp<4.0,>=3.6->pystan) (1.19.0)
Collecting crashtest<0.4.0,>=0.3.0 (from clikit<0.7,>=0.6->pystan)
  Downloading crashtest-0.3.1-py3-none-any.whl.metadata (748 bytes)
Collecting pastel<0.3.0,>=0.2.0 (from clikit<0.7,>=0.6->pystan)
  Downloading pastel-0.2.1-py2.py3-none-any.whl.metadata (1.9 kB)
Collecting pylev<2.0,>=1.3 (from clikit<0.7,>=0.6->pystan)
  Downloading pylev-1.4.0-py2.py3-none-any.whl.metadata (2.3 kB)
Collecting appdirs<2.0,>=1.4 (from httpstan<4.14,>=4.13->pystan)
  Downloading appdirs-1.4.4-py2.py3-none-any.whl.metadata (9.0 kB)
Collecting marshmallow<4.0,>=3.10 (from httpstan<4.14,>=4.13->pystan)
  Downloading marshmallow-3.26.1-py3-none-any.whl.metadata (7.3 kB)
Collecting webargs<9.0,>=8.0 (from httpstan<4.14,>=4.13->pystan)
  Downloading webargs-8.7.0-py3-none-any.whl.metadata (6.6 kB)
Requirement already satisfied: packaging>=17.0 in /usr/local/lib/python3.11/dist-p
ackages (from marshmallow<4.0,>=3.10->httpstan<4.14,>=4.13->pystan) (24.2)
Requirement already satisfied: idna>=2.0 in /usr/local/lib/python3.11/dist-package
s (from yarl<2.0,>=1.17.0->aiohttp<4.0,>=3.6->pystan) (3.10)
Downloading pystan-3.10.0-py3-none-any.whl (13 kB)
Downloading clikit-0.6.2-py2.py3-none-any.whl (91 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 91.8/91.8 kB 1.9 MB/s eta 0:00:00
Downloading httpstan-4.13.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_6
4.whl (45.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 45.5/45.5 MB 11.4 MB/s eta 0:00:00
Downloading pysimdjson-6.0.2-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_6
4.whl (1.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.9/1.9 MB 83.6 MB/s eta 0:00:00
Downloading appdirs-1.4.4-py2.py3-none-any.whl (9.6 kB)
Downloading crashtest-0.3.1-py3-none-any.whl (7.0 kB)
Downloading marshmallow-3.26.1-py3-none-any.whl (50 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 50.9/50.9 kB 5.5 MB/s eta 0:00:00
Downloading pastel-0.2.1-py2.py3-none-any.whl (6.0 kB)
Downloading pylev-1.4.0-py2.py3-none-any.whl (6.1 kB)
Downloading webargs-8.7.0-py3-none-any.whl (31 kB)
Installing collected packages: pylev, appdirs, pysimdjson, pastel, marshmallow, cr
ashtest, webargs, clikit, httpstan, pystan
```

```
Successfully installed appdirs-1.4.4 clikit-0.6.2 crashtest-0.3.1 httpstan-4.13.0
marshmallow-3.26.1 pastel-0.2.1 pylev-1.4.0 pysimdjson-6.0.2 pystan-3.10.0 webargs
-8.7.0
Requirement already satisfied: prophet in /usr/local/lib/python3.11/dist-packages
(1.1.6)
Requirement already satisfied: cmdstanpy>=1.0.4 in /usr/local/lib/python3.11/dist-
packages (from prophet) (1.2.5)
Requirement already satisfied: numpy>=1.15.4 in /usr/local/lib/python3.11/dist-pac
kages (from prophet) (2.0.2)
Requirement already satisfied: matplotlib>=2.0.0 in /usr/local/lib/python3.11/dist
-packages (from prophet) (3.10.0)
Requirement already satisfied: pandas>=1.0.4 in /usr/local/lib/python3.11/dist-pac
kages (from prophet) (2.2.2)
Requirement already satisfied: holidays<1,>=0.25 in /usr/local/lib/python3.11/dist
-packages (from prophet) (0.70)
Requirement already satisfied: tqdm>=4.36.1 in /usr/local/lib/python3.11/dist-pack
ages (from prophet) (4.67.1)
Requirement already satisfied: importlib-resources in /usr/local/lib/python3.11/di
st-packages (from prophet) (6.5.2)
Requirement already satisfied: stanio<2.0.0,>=0.4.0 in /usr/local/lib/python3.11/d
ist-packages (from cmdstanpy>=1.0.4->prophet) (0.5.1)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.11/dist-p
ackages (from holidays<1,>=0.25->prophet) (2.8.2)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-
packages (from matplotlib>=2.0.0->prophet) (1.3.2)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-pack
ages (from matplotlib>=2.0.0->prophet) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist
-packages (from matplotlib>=2.0.0->prophet) (4.57.0)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist
-packages (from matplotlib>=2.0.0->prophet) (1.4.8)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-p
ackages (from matplotlib>=2.0.0->prophet) (24.2)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.11/dist-package
s (from matplotlib>=2.0.0->prophet) (11.1.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-
packages (from matplotlib>=2.0.0->prophet) (3.2.3)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-pack
ages (from pandas>=1.0.4->prophet) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-pa
ckages (from pandas>=1.0.4->prophet) (2025.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages
(from python-dateutil->holidays<1,>=0.25->prophet) (1.17.0)
```

In [55]:

```python
from prophet import Prophet
import pandas as pd
from colorama import Fore
import logging
import matplotlib.pyplot as plt
from sklearn.metrics import mean_absolute_error as mae, mean_squared_error as mse
import numpy as np
import re

class Prophet_Pipeline:
    def __init__(self, signal_data, forecast_steps=20, exog_data=None):
        self.signal_data = signal_data
        self.exog_data = exog_data
        self.forecast_steps = forecast_steps
        self.train = None
        self.test = None
        self.df_grpd_series = None
        self.df_prophet = None
        self.forecast = None
        self.y_true = None
```

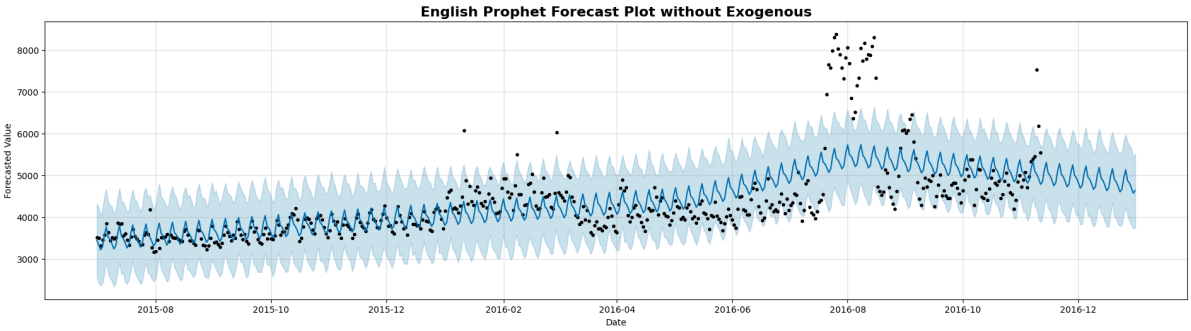
```
        self.y_pred = None
        self.Language = None
        self.model = None
        self.forecast_with_exog = None

    def get_language_info(self):
        pattern = r'(\w+)_Clicks'
        regex = re.compile(pattern)
        search_obj = regex.search(self.df_grpd_series.name)
        self.Language = search_obj.group(1)
        print(f'\n{Fore.GREEN}{"*"*50}\nLanguage : {self.Language}\n{"*"*50}')

    def preprocess_data(self):
        self.signal_data.fillna(0, inplace=True)

        self.df_grpd_series = self.signal_data.groupby(by='Dates')[self.signal_data
        self.get_language_info()
        self.df_grpd = self.df_grpd_series.sort_index(ascending=True).reset_index()
        self.df_grpd.columns = ['ds', 'y']

        self.df_prophet = self.df_grpd
        self.train = self.df_prophet[:-self.forecast_steps]
        self.test = self.df_prophet[-self.forecast_steps:]

        print(Fore.GREEN + '*'*50)
        print(f'Train Samples : {len(self.train)}')
        print('*'*50)
        print(f'Test Samples : {len(self.test)}')
        print('*'*50)

        if self.exog_data is not None:
            self.exog_data.index.name = 'Dates'
            self.exog_data = self.exog_data.sort_index()
            self.exog_data = self.exog_data.reset_index()
            self.exog_train = self.exog_data[:-self.forecast_steps]
            self.exog_test = self.exog_data[-self.forecast_steps:]
        else:
            self.exog_train = self.exog_test = None

    def build_model(self, use_exog=False):
        self.model = Prophet(weekly_seasonality=True)

        if use_exog and self.exog_data is not None:
            for col in self.exog_data.columns[1:]:
                self.model.add_regressor(col)
            self.model.fit(pd.concat([self.train, self.exog_train.iloc[:, 1:]], axi
            future = self.model.make_future_dataframe(periods=self.forecast_steps)
            future = pd.concat([future, self.exog_data.iloc[:, 1:]], axis=1)
            self.forecast_with_exog = self.model.predict(future)
        else:
            self.model.fit(self.train)
            future = self.model.make_future_dataframe(periods=self.forecast_steps)
            self.forecast = self.model.predict(future)

        logging.getLogger('prophet').setLevel(logging.WARNING)
        logging.getLogger('cmdstanpy').setLevel(logging.WARNING)

    def plot_forecast(self, use_exog=False):
        forecast = self.forecast_with_exog if use_exog else self.forecast
        fig = self.model.plot(forecast)
        fig.set_size_inches(20, 5)
        title = f"{self.Language} Prophet Forecast Plot {'with Exogenous' if use_ex
        plt.title(title, fontsize=16, fontweight='bold')
        plt.xlabel("Date")
```

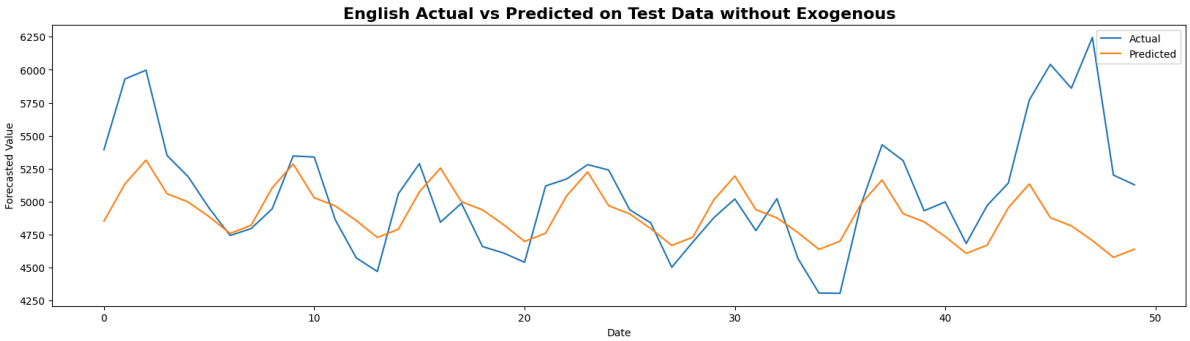
```
        plt.ylabel("Forecasted Value")
        plt.grid(True)
        plt.show()

    def plot_test_comparison(self, use_exog=False):
        forecast = self.forecast_with_exog if use_exog else self.forecast
        y_true = self.test['y'].values
        y_pred = forecast['yhat'][-self.forecast_steps:].values

        plt.figure(figsize=(20, 5))
        title = f"{self.Language} Actual vs Predicted on Test Data {'with Exogenous
        plt.title(title, fontsize=16, fontweight='bold')
        plt.plot(y_true, label='Actual')
        plt.plot(y_pred, label='Predicted')
        plt.xlabel("Date")
        plt.ylabel("Forecasted Value")
        plt.legend()
        plt.show()

    def calculate_scores(self, actual, predicted):
        print(f'MAE : {round(mae(actual, predicted), 4)}')
        print(f'MSE : {round(mse(actual, predicted), 4)}')
        print(f'MAPE : {round(np.mean(np.abs((actual - predicted) / actual)) * 100,

    def performance(self, use_exog=False):
        forecast = self.forecast_with_exog if use_exog else self.forecast
        model_type = "Prophet with Exogenous" if use_exog else "Prophet"

        train_pred = forecast[['ds', 'yhat']].set_index('ds').loc[self.train['ds']]
        test_pred = forecast[['ds', 'yhat']].set_index('ds').loc[self.test['ds']]

        print('\n')
        print(f'{Fore.GREEN}{"*" * 50}')
        print(f'Language : {self.Language}\n')
        print(f'Model Type : {model_type}')

        print(f'Training Performance : {Fore.BLUE}')
        self.calculate_scores(self.train['y'].values, train_pred['yhat'].values)

        print(f'{Fore.GREEN}Testing Performance : {Fore.BLUE}')
        self.calculate_scores(self.test['y'].values, test_pred['yhat'].values)

        print(f'{Fore.GREEN}{"*" * 50}')

    def forecast_pipeline(self):
        self.preprocess_data()

        print(f'{Fore.CYAN}Running model WITHOUT exogenous variables...\n')
        self.build_model(use_exog=False)
        self.plot_forecast(use_exog=False)
        self.plot_test_comparison(use_exog=False)
        self.performance(use_exog=False)

        if self.exog_data is not None:
            print(f'\n{Fore.CYAN}Running model WITH exogenous variables...\n')
            self.build_model(use_exog=True)
            self.plot_forecast(use_exog=True)
            self.plot_test_comparison(use_exog=True)
            self.performance(use_exog=True)
```

English Language

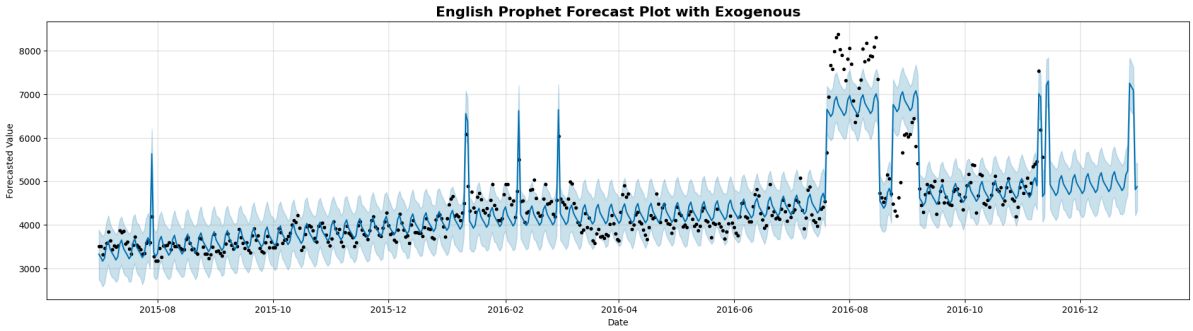
In [56]:
```
with open('/content/drive/MyDrive/Scaler_DSML_Digital_Notes/BusinessCases/11_AdEase
  english_df = pickle.load(fp)


exog_path = '/content/drive/MyDrive/Scaler_DSML_Digital_Notes/BusinessCases/11_AdEa
exog_df = pd.read_csv(exog_path)

prophet_english = Prophet_Pipeline(english_df, forecast_steps=50, exog_data=exog_df
prophet_english.forecast_pipeline()
```

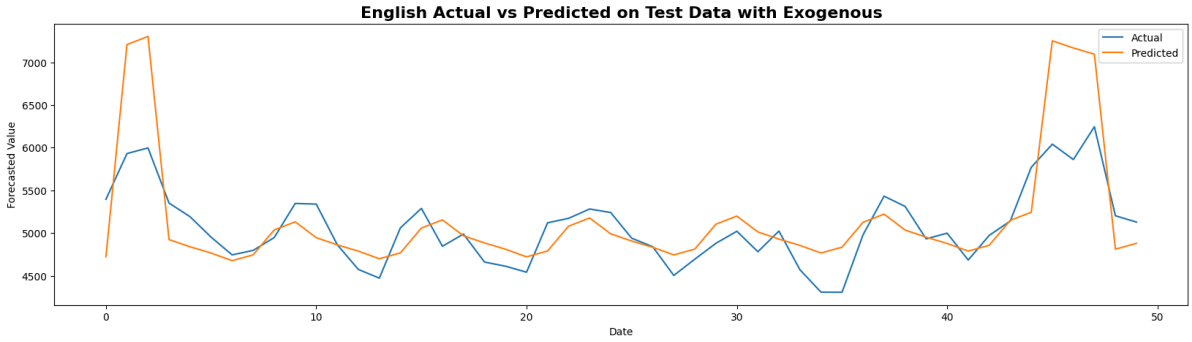
```
**************************************************
Language : English
**************************************************
**************************************************
Train Samples : 500
**************************************************
Test Samples : 50
**************************************************
Running model WITHOUT exogenous variables...
```






```
**************************************************
Language : English

Model Type : Prophet
Training Performance :
MAE : 444.011
MSE : 474047.7791
MAPE : 8.9841
Testing Performance :
MAE : 300.8145
MSE : 183333.8337
MAPE : 5.6586
**************************************************

Running model WITH exogenous variables...
```



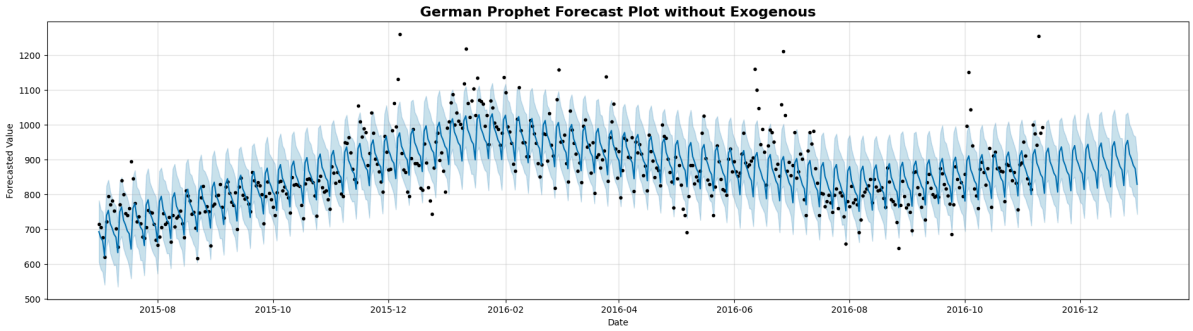


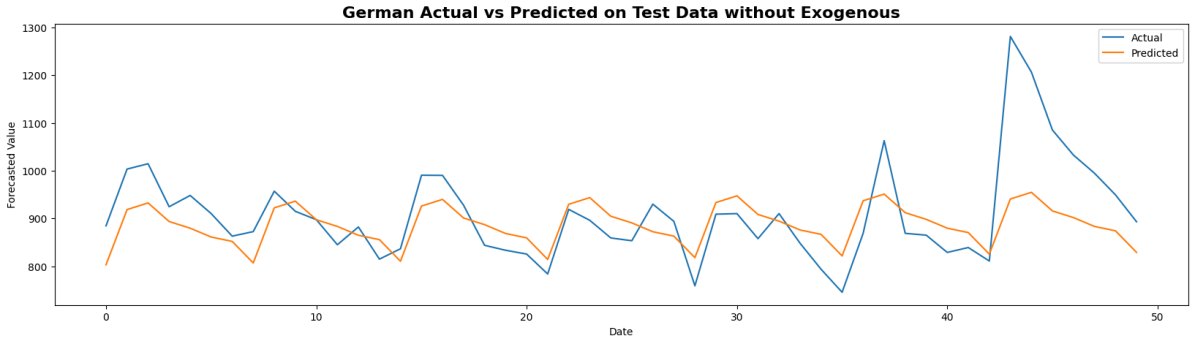

```
**************************************************
Language : English

Model Type : Prophet with Exogenous
Training Performance :
MAE : 280.4493
MSE : 194768.6384
MAPE : 5.9323
Testing Performance :
MAE : 312.9439
MSE : 206932.013
MAPE : 5.8895
**************************************************
```

German Language

In [57]:
```
with open('/content/drive/MyDrive/Scaler_DSML_Digital_Notes/BusinessCases/11_AdEase
  german_df = pickle.load(fp)

prophet_german = Prophet_Pipeline(german_df, forecast_steps=50)
prophet_german.forecast_pipeline()
```

```
**************************************************
Language : German
**************************************************
**************************************************
Train Samples : 500
**************************************************
Test Samples : 50
**************************************************
Running model WITHOUT exogenous variables...
```



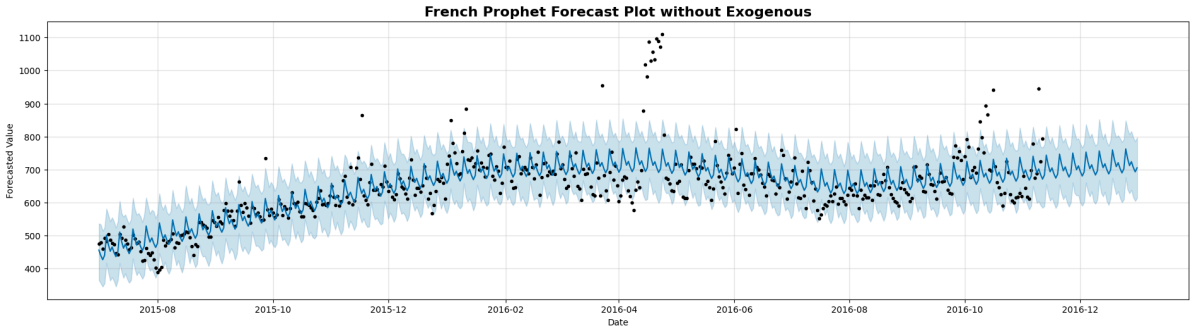


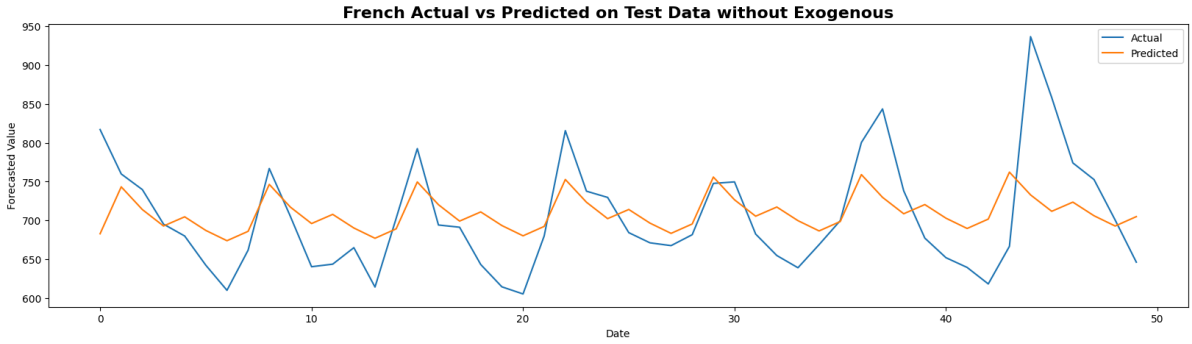

```
**************************************************
Language : German

Model Type : Prophet
Training Performance :
MAE : 48.9072
MSE : 4579.1191
MAPE : 5.5308
Testing Performance :
MAE : 60.5408
MSE : 7060.5736
MAPE : 6.2908
**************************************************
```

French Language

In [58]:
```
with open('/content/drive/MyDrive/Scaler_DSML_Digital_Notes/BusinessCases/11_AdEase
  french_df = pickle.load(fp)

prophet_french = Prophet_Pipeline(french_df, forecast_steps=50)
prophet_french.forecast_pipeline()
```

```
**************************************************
Language : French
**************************************************
**************************************************
Train Samples : 500
**************************************************
Test Samples : 50
**************************************************
Running model WITHOUT exogenous variables...
```



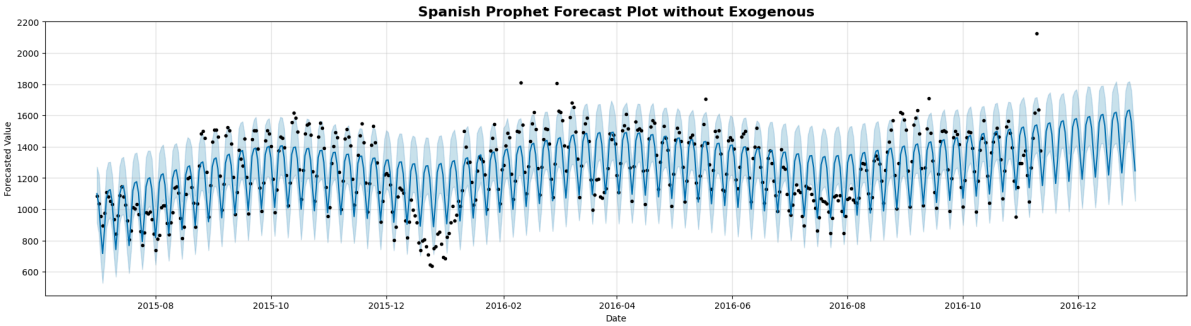


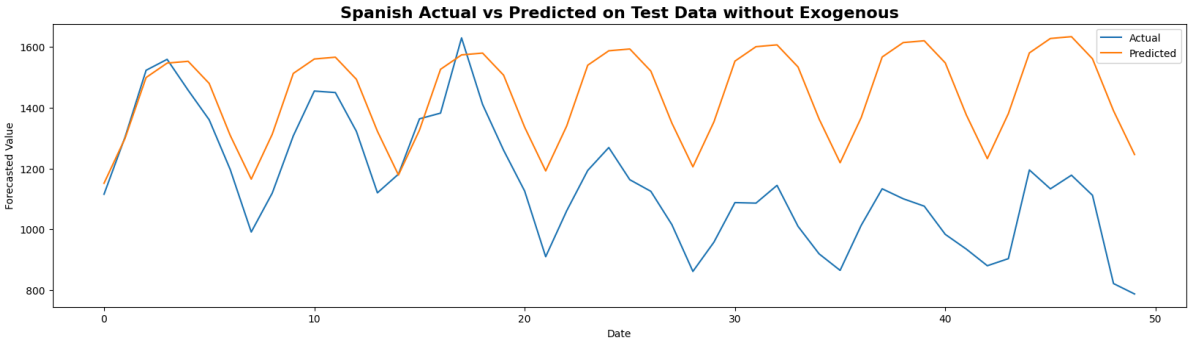

```
**************************************************
Language : French

Model Type : Prophet
Training Performance :
MAE : 41.6471
MSE : 4786.4578
MAPE : 6.0613
Testing Performance :
MAE : 46.2071
MSE : 3679.0141
MAPE : 6.4765
**************************************************
```

Spanish Language

In [59]:
```
with open('/content/drive/MyDrive/Scaler_DSML_Digital_Notes/BusinessCases/11_AdEase
  spanish_df = pickle.load(fp)

prophet_spanish= Prophet_Pipeline(spanish_df, forecast_steps=50)
prophet_spanish.forecast_pipeline()
```

```
**************************************************
Language : Spanish
**************************************************
**************************************************
Train Samples : 500
**************************************************
Test Samples : 50
**************************************************
Running model WITHOUT exogenous variables...
```



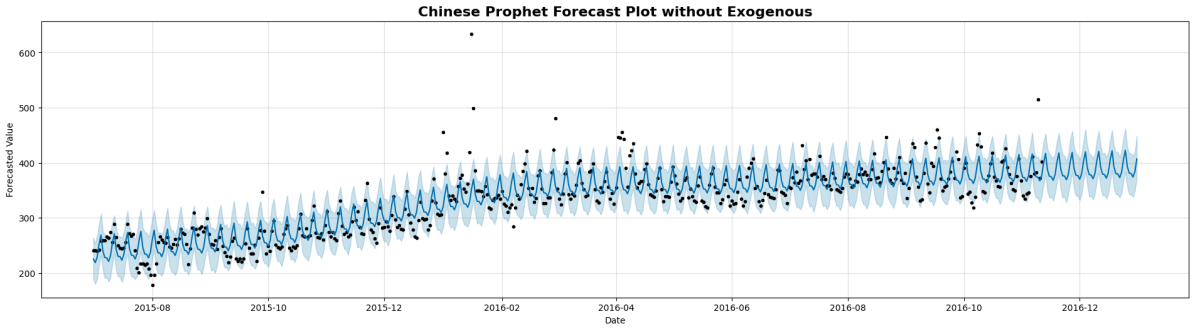


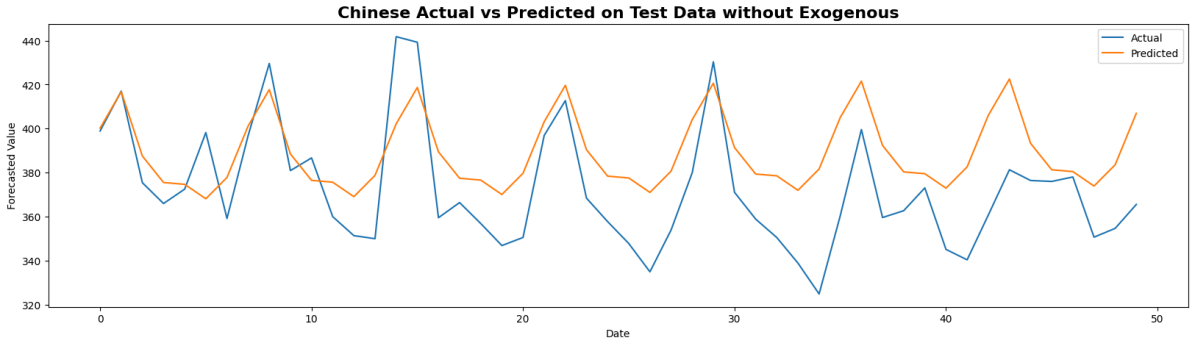

```
**************************************************
Language : Spanish

Model Type : Prophet
Training Performance :
MAE : 106.258
MSE : 20399.0849
MAPE : 9.3343
Testing Performance :
MAE : 296.5329
MSE : 117611.8329
MAPE : 28.2914
**************************************************
```

Chinese Language

In [60]:
```
with open('/content/drive/MyDrive/Scaler_DSML_Digital_Notes/BusinessCases/11_AdEase
  chinese_df = pickle.load(fp)

prophet_chinese= Prophet_Pipeline(chinese_df, forecast_steps=50)
prophet_chinese.forecast_pipeline()
```

```
**************************************************
Language : Chinese
**************************************************
**************************************************
Train Samples : 500
**************************************************
Test Samples : 50
**************************************************
Running model WITHOUT exogenous variables...
```



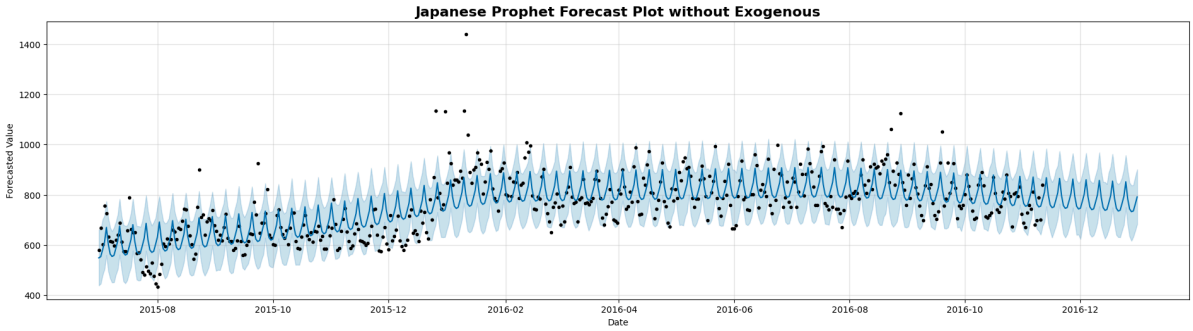


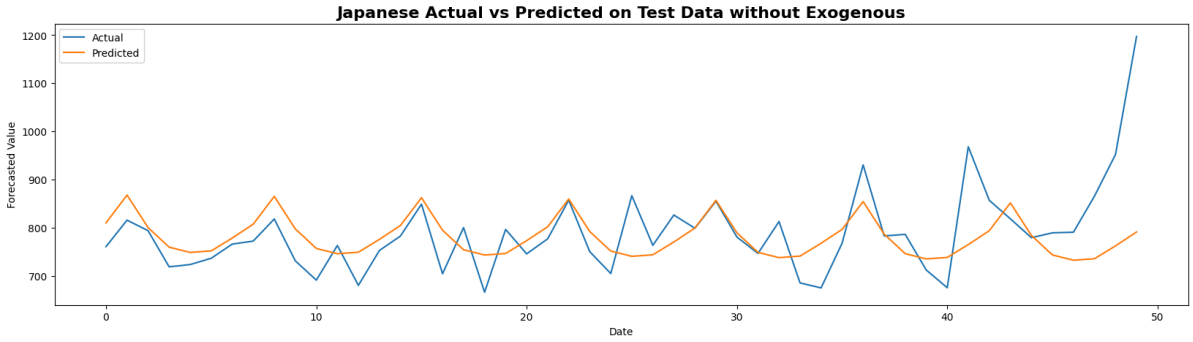

```
**************************************************
Language : Chinese

Model Type : Prophet
Training Performance :
MAE : 19.7365
MSE : 860.6872
MAPE : 6.0054
Testing Performance :
MAE : 21.6437
MSE : 641.9401
MAPE : 5.9599
**************************************************
```

Japanese Language

In [61]:
```
with open('/content/drive/MyDrive/Scaler_DSML_Digital_Notes/BusinessCases/11_AdEase
  japanese_df = pickle.load(fp)

prophet_japanese= Prophet_Pipeline(japanese_df, forecast_steps=50)
prophet_japanese.forecast_pipeline()
```

```
**************************************************
Language : Japanese
**************************************************
**************************************************
Train Samples : 500
**************************************************
Test Samples : 50
**************************************************
Running model WITHOUT exogenous variables...
```



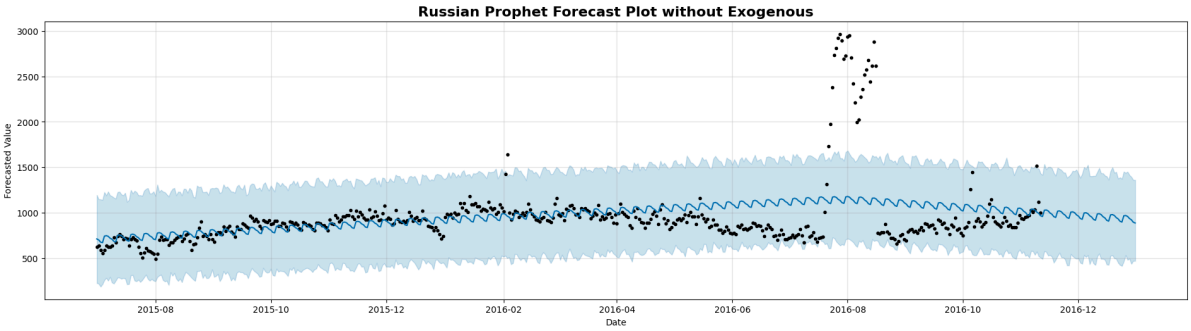


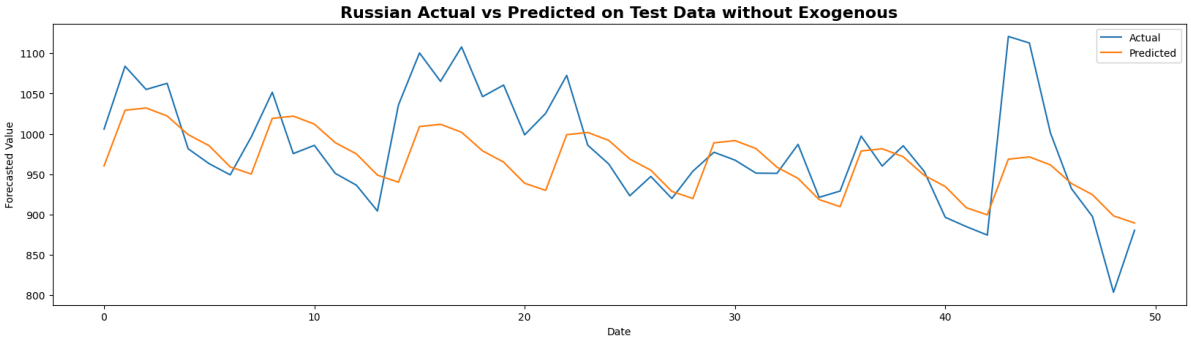

```
**************************************************
Language : Japanese

Model Type : Prophet
Training Performance :
MAE : 61.1714
MSE : 7010.5267
MAPE : 8.1098
Testing Performance :
MAE : 55.5112
MSE : 7398.5343
MAPE : 6.6616
**************************************************
```

Russian Language

In [62]:
```
with open('/content/drive/MyDrive/Scaler_DSML_Digital_Notes/BusinessCases/11_AdEase
  russian_df = pickle.load(fp)

prophet_russian= Prophet_Pipeline(russian_df, forecast_steps=50)
prophet_russian.forecast_pipeline()
```

```
**************************************************
Language : Russian
**************************************************
**************************************************
Train Samples : 500
**************************************************
Test Samples : 50
**************************************************
Running model WITHOUT exogenous variables...
```






```
**************************************************
Language : Russian

Model Type : Prophet
Training Performance :
MAE : 197.9298
MSE : 136731.1737
MAPE : 17.9234
Testing Performance :
MAE : 42.4347
MSE : 2987.1412
MAPE : 4.2137
**************************************************
```

In [62]: