In [ ]:

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.

Data Dictionary:

There are two csv files given

train_1.csv: In the csv file, each row corresponds to a particular article and each column corresponds to a particular date. The values are the number of visits on that date.

The page name contains data in this format:

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

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

Exog_Campaign_eng: This file contains data for the dates which had a campaign or significant event that could affect the views for that day. The data is just for pages in English.

There's 1 for dates with campaigns and 0 for remaining dates. It is to be treated as an exogenous variable for models when training and forecasting data for pages in English

---

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

In [1]:

```
import pandas as pd
import numpy as np
import pylab as p
import matplotlib.pyplot as plot
from collections import Counter
import re
import os
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
warnings.simplefilter("ignore")
```

In [2]:

```
!gdown 1RPOLd_74kMrpkTg35X843hOrZIsSJSMF
```

```
Downloading...
From (original): https://drive.google.com/uc?id=1RPOLd_74kMrpkTg35X843hOrZIsSJSMF
From (redirected): https://drive.google.com/uc?id=1RPOLd_74kMrpkTg35X843hOrZIsSJSMF&confirm=t&uuid=a5693cd9-998f-4943-b928-b9be08e8e0ae
To: /content/train_1.csv
100% 278M/278M [00:01<00:00, 143MB/s]
```

In [3]:

```
train = pd.read_csv('train_1.csv')
```

In [4]:

```
train.shape
```

Out[4]:

```
(145063, 551)
```

In [5]:

```
train.head(5)
```

Out[5]:

	Page	2015-07-01	2015-07-02	2015-07-03	2015-07-04	2015-07-05	2015-07-06	2015-07-07	2015-07-08	2015-07-09	...	2016-12-22	2016-12-23	2016-12-24	2016-12-25	2016-12-26	2016-12-27	2016-12-28
0	2NE1_zh.wikipedia.org_all-access_spider	18.0	11.0	5.0	13.0	14.0	9.0	9.0	22.0	26.0	...	32.0	63.0	15.0	26.0	14.0	20.0	22.0
1	2PM_zh.wikipedia.org_all-access_spider	11.0	14.0	15.0	18.0	11.0	13.0	22.0	11.0	10.0	...	17.0	42.0	28.0	15.0	9.0	30.0	52.0

	Page	2015-07-01	2015-07-02	2015-07-03	2015-07-04	2015-07-05	2015-07-06	2015-07-07	2015-07-08	2015-07-09	...	2016-12-22	2016-12-23	2016-12-24	2016-12-25	2016-12-26	2016-12-27	2(1
2	3C_zh.wikipedia.org_all-access_spider	1.0	0.0	1.0	1.0	0.0	4.0	0.0	3.0	4.0	...	3.0	1.0	1.0	7.0	4.0	4.0	
3	4minute_zh.wikipedia.org_all-access_spider	35.0	13.0	10.0	94.0	4.0	26.0	14.0	9.0	11.0	...	32.0	10.0	26.0	27.0	16.0	11.0	
4	52_Hz_I_Love_You_zh.wikipedia.org_all-access_s...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	48.0	9.0	25.0	13.0	3.0	11.0	

5 rows × 551 columns

In [6]:
```
from datetime import datetime
array_of_date=train.columns[1:]
date_objects = [datetime.strptime(date_str, '%Y-%m-%d') for date_str in array_of_date]
```

In [7]:
```
len(date_objects)
```

Out[7]:
```
550
```

In [8]:
```
date_objects[0]
```

Out[8]:
```
datetime.datetime(2015, 7, 1, 0, 0)
```

In [9]:
```
date_objects[-1]
```

Out[9]:
```
datetime.datetime(2016, 12, 31, 0, 0)
```

In [10]:
```
min(date_objects)
```

Out[10]:
```
datetime.datetime(2015, 7, 1, 0, 0)
```

In [11]:
```
max(date_objects)
```

Out[11]:
```
datetime.datetime(2016, 12, 31, 0, 0)
```

In [12]:
```
difference = max(date_objects)-min(date_objects)
```

In [13]:
```
difference.days
```

Out[13]:
```
549
```

we have data from 01-07-2015 to 31-12-2016

- Diff between start and end date is of 549 days
- so we have data of 550 days which is matched with the length of the date array

In [14]:
```
print(train.info())
```

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

In [15]:
```
train.isnull()
```

Out[15]:

	Page	2015-07-01	2015-07-02	2015-07-03	2015-07-04	2015-07-05	2015-07-06	2015-07-07	2015-07-08	2015-07-09	...	2016-12-22	2016-12-23	2016-12-24	2016-12-25	2016-12-26	2016-12-27	2016-12-28	2016-12-29	2016-12-30	201 12-3
0	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	Fal
1	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	Fal
2	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	Fal
3	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	Fal
4	False	True	True	True	True	True	True	True	True	True	...	False	False	False	False	False	False	False	False	False	Fal
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	

	Page	2015-07-01	2015-07-02	2015-07-03	2015-07-04	2015-07-05	2015-07-06	2015-07-07	2015-07-08	2015-07-09	...	2016-12-22	2016-12-23	2016-12-24	2016-12-25	2016-12-26	2016-12-27	2016-12-28	2016-12-29	2016-12-30	201 12-3
145058	False	True	True	True	True	True	True	True	True	True	...	True	True	True	True	False	False	False	False	False	Fal
145059	False	True	True	True	True	True	True	True	True	True	...	True	True	True	True	True	True	True	True	True	Tr
145060	False	True	True	True	True	True	True	True	True	True	...	True	True	True	True	True	True	True	True	True	Tr
145061	False	True	True	True	True	True	True	True	True	True	...	True	True	True	True	True	True	True	True	True	Tr
145062	False	True	True	True	True	True	True	True	True	True	...	True	True	True	True	True	True	True	True	True	Tr

145063 rows × 551 columns

In [16]:
```
na_count=train.isnull().sum()[1:].values
```

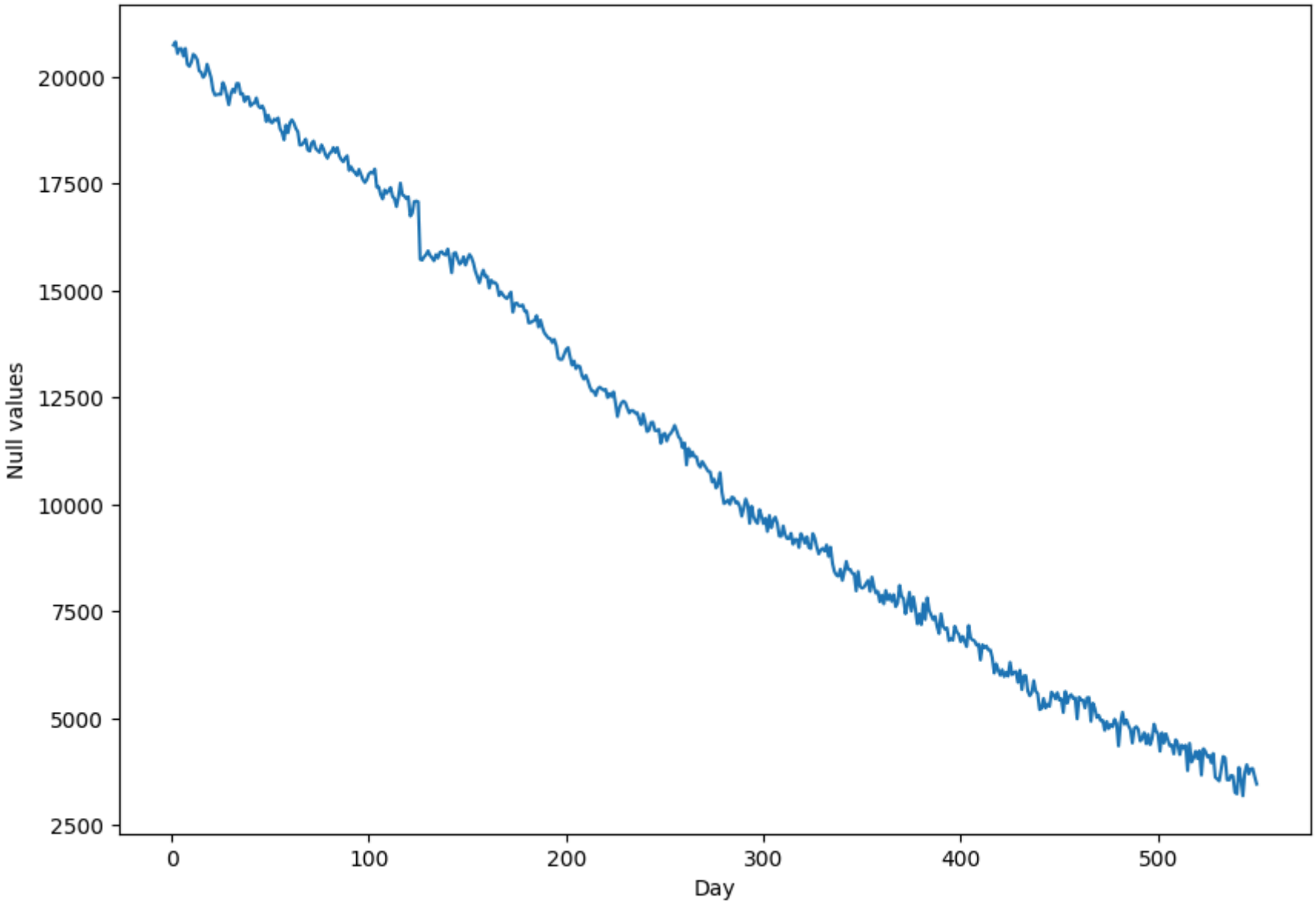
In [17]:
```
len(na_count)
```

Out[17]: 550

In [18]:
```
map_days_to_number = [r for r in range(1, len(train.columns))]
```

In [19]:
```
plot.figure(figsize=(10,7))
plot.xlabel('Day')
plot.ylabel('Null values')
plot.plot(map_days_to_number,na_count)
```

Out[19]: [<matplotlib.lines.Line2D at 0x7f8a2ac646a0>]



In [19]:

We see that na values decreased with time may be some website was not been created during inital time ,or not been recorded

In [20]:
```
df1=train.isnull().sum(axis=1).to_frame()
df2=train['Page'].to_frame()
merged_df = pd.concat([df1, df2], axis=1)
merged_df.rename(columns={0:"count_of_na"},inplace=True)
merged_df.columns
```

Out[20]:
```
Index(['count_of_na', 'Page'], dtype='object')
```

In [21]:
```
sorted_df = merged_df.sort_values(by='count_of_na', ascending=False)
```

In [22]:
```
sorted_df.iloc[0:10]['Page']
```

Out[22]:
```
145062    Francisco_el_matemático_(serie_de_televisión_d...
90284     特別:フィード項目/featured/20170213000000/ja_ja.wikipe...
90291     特別:フィード項目/featured/20170220000000/ja_ja.wikipe...
90290     特別:フィード項目/featured/20170219000000/ja_ja.wikipe...
90289     特別:フィード項目/featured/20170218000000/ja_ja.wikipe...
90288     特別:フィード項目/featured/20170217000000/ja_ja.wikipe...
90287     特別:フィード項目/featured/20170216000000/ja_ja.wikipe...
90286     特別:フィード項目/featured/20170215000000/ja_ja.wikipe...
90285     特別:フィード項目/featured/20170214000000/ja_ja.wikipe...
90283     特別:フィード項目/featured/20170211000000/ja_ja.wikipe...
Name: Page, dtype: object
```

Top ten pages which has high nan values

In [22]:
```

```

Handling NA values

In [23]:
```
print(train.shape)
```

```
(145063, 551)
```

In [24]:
```
train1=train.dropna(thresh=300)
```

In [25]:
```
print(train1.shape)
```

```
(133617, 551)
```

We have removed the pages (rows) which has nan counts greater than 300

We fill all the remaining values with zero assuming there was no traffic on the date that the values are nan for

In [26]:
```
train1=train1.fillna(0)
train1.head()
```

Out[26]:

	Page	2015-07-01	2015-07-02	2015-07-03	2015-07-04	2015-07-05	2015-07-06	2015-07-07	2015-07-08	2015-07-09	...	2016-12-22	2016-12-23	2016-12-24	2016-12-25	2016-12-26	2016-12-27	2016-12-28	20 12-
0	2NE1_zh.wikipedia.org_all-access_spider	18.0	11.0	5.0	13.0	14.0	9.0	9.0	22.0	26.0	...	32.0	63.0	15.0	26.0	14.0	20.0	22.0	1
1	2PM_zh.wikipedia.org_all-access_spider	11.0	14.0	15.0	18.0	11.0	13.0	22.0	11.0	10.0	...	17.0	42.0	28.0	15.0	9.0	30.0	52.0	4
2	3C_zh.wikipedia.org_all-access_spider	1.0	0.0	1.0	1.0	0.0	4.0	0.0	3.0	4.0	...	3.0	1.0	1.0	7.0	4.0	4.0	6.0	
3	4minute_zh.wikipedia.org_all-access_spider	35.0	13.0	10.0	94.0	4.0	26.0	14.0	9.0	11.0	...	32.0	10.0	26.0	27.0	16.0	11.0	17.0	1
5	5566_zh.wikipedia.org_all-access_spider	12.0	7.0	4.0	5.0	20.0	8.0	5.0	17.0	24.0	...	16.0	27.0	8.0	17.0	32.0	19.0	23.0	1

5 rows × 551 columns

In [27]:
```
train1.isnull().sum().max()
```

Out[27]:
```
0
```

In [28]:
```
train1.isnull().sum(axis=1).max()
```

Out[28]: 0

Splitting the page columns based on SPECIFIC NAME *LANGUAGE.wikipedia.org* ACCESS TYPE _ ACCESS ORIGIN

In [29]:
```
def split_page(page):
  w = re.split('_|\.', page)

  return ' '.join(w[:-5]), w[-5], w[-2], w[-1]
```

In [30]:
```
li = list(train1.Page.apply(split_page))
print(len(li))
df = pd.DataFrame(li)
df.columns = ['Title', 'Language', 'Access_type','Access_origin']
```

```
133617
```

In [31]:
```
df.reset_index(inplace=True)
```

In [32]:
```
train1.reset_index(inplace=True)
```

In [33]:
```
df_2 = pd.concat([train1, df], axis = 1)
```

In [34]:
```
df_2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 133617 entries, 0 to 133616
Columns: 557 entries, index to Access_origin
dtypes: float64(550), int64(2), object(5)
memory usage: 567.8+ MB
```

In [34]:
```
```

In [35]:
```
df_2
```

Out[35]:

	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	...	2016-12-27	2016-12-28	2016-12-29	2016-12-30
0	0	2NE1_zh.wikipedia.org_all-access_spider	18.0	11.0	5.0	13.0	14.0	9.0	9.0	22.0	...	20.0	22.0	19.0	18.0
1	1	2PM_zh.wikipedia.org_all-access_spider	11.0	14.0	15.0	18.0	11.0	13.0	22.0	11.0	...	30.0	52.0	45.0	26.0
2	2	3C_zh.wikipedia.org_all-access_spider	1.0	0.0	1.0	1.0	0.0	4.0	0.0	3.0	...	4.0	6.0	3.0	4.0
3	3	4minute_zh.wikipedia.org_all-access_spider	35.0	13.0	10.0	94.0	4.0	26.0	14.0	9.0	...	11.0	17.0	19.0	10.0
4	5	5566_zh.wikipedia.org_all-access_spider	12.0	7.0	4.0	5.0	20.0	8.0	5.0	17.0	...	19.0	23.0	17.0	17.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
133612	145012	Legión_(Marvel_Comics)_es.wikipedia.org_all-ac...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	4.0	4.0	1.0	2.0
133613	145013	Referéndum_sobre_la_permanencia_del_Reino_Unid...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	3.0	3.0	10.0	11.0
133614	145014	Salida_del_Reino_Unido_de_la_Unión_Europea_es....	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	8.0	22.0	13.0	18.0
133615	145015	Amar,_después_de_amar_es.wikipedia.org_all-acc...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	7.0	5.0	43.0	12.0
133616	145016	Anexo:89.º_Premios_Óscar_es.wikipedia.org_all-...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	1.0	1.0

133617 rows × 557 columns

In [36]:

```
df_2.shape
```

Out[36]: (133617, 557)

In [37]:
```
df_2.groupby('Access_type').count()
```

Out[37]:

Access_type	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	...	2016-12-26	2016-12-27	2016-12-28	2016-12-29	2016-12-30	2016-12-31	index	Title	Langu
all-access	68718	68718	68718	68718	68718	68718	68718	68718	68718	68718	...	68718	68718	68718	68718	68718	68718	68718	68718	68
desktop	31600	31600	31600	31600	31600	31600	31600	31600	31600	31600	...	31600	31600	31600	31600	31600	31600	31600	31600	31
mobile-web	33299	33299	33299	33299	33299	33299	33299	33299	33299	33299	...	33299	33299	33299	33299	33299	33299	33299	33299	33

3 rows × 556 columns

In [38]:
```
df_2.groupby('Access_origin').count()
```

Out[38]:

Access_origin	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	...	2016-12-26	2016-12-27	2016-12-28	2016-12-29	2016-12-30	2016-12-31	i
all-agents	101303	101303	101303	101303	101303	101303	101303	101303	101303	101303	...	101303	101303	101303	101303	101303	101303	10
spider	32314	32314	32314	32314	32314	32314	32314	32314	32314	32314	...	32314	32314	32314	32314	32314	32314	3

2 rows × 556 columns

In [39]:
```
df_2['Access_origin'].value_counts()
```

Out[39]:
```
Access_origin
all-agents    101303
spider         32314
Name: count, dtype: int64
```

In [40]:
```
df_2['Access_type'].value_counts()
```

Out[40]:
```
Access_type
all-access    68718
mobile-web    33299
desktop       31600
Name: count, dtype: int64
```

In [41]:
```
df_2['Language'].value_counts()
```

Out[41]:
```
Language
en          22486
ja          19295
de          17362
fr          16948
zh          15211
ru          14270
es          13551
commons      8266
www          6228
Name: count, dtype: int64
```

In [42]:
```
import re

def language_pre(Page):
    val = re.search('[[a-z][a-z]\.([a-z]+)\.org',Page)
    if val:
        return val[0][0:2]
    else:
      return "undefined_lag"
```

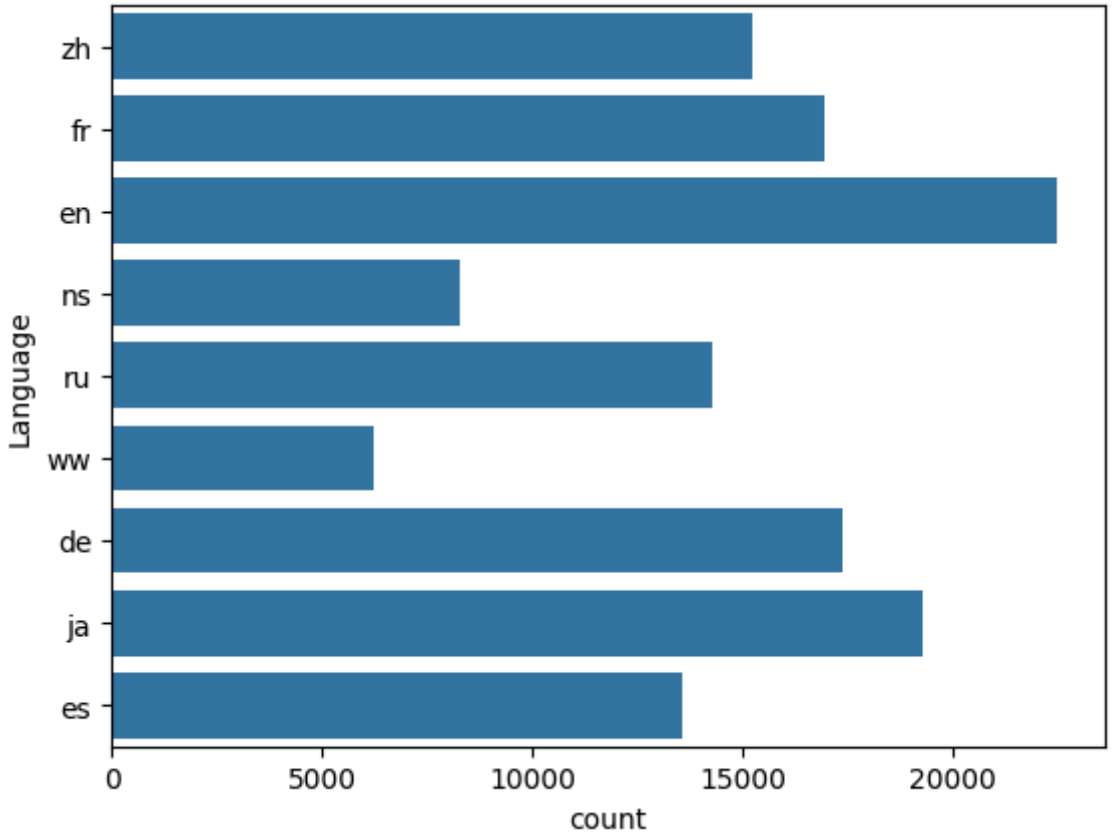
In [43]:
```
df_2['Language']=df_2['Page'].apply(lambda x: language_pre(str(x)))
```

In [44]:
```
df_2['Language'].value_counts()
```

Out[44]:
```
Language
en    22486
ja    19295
de    17362
fr    16948
zh    15211
ru    14270
es    13551
ns     8266
ww     6228
Name: count, dtype: int64
```

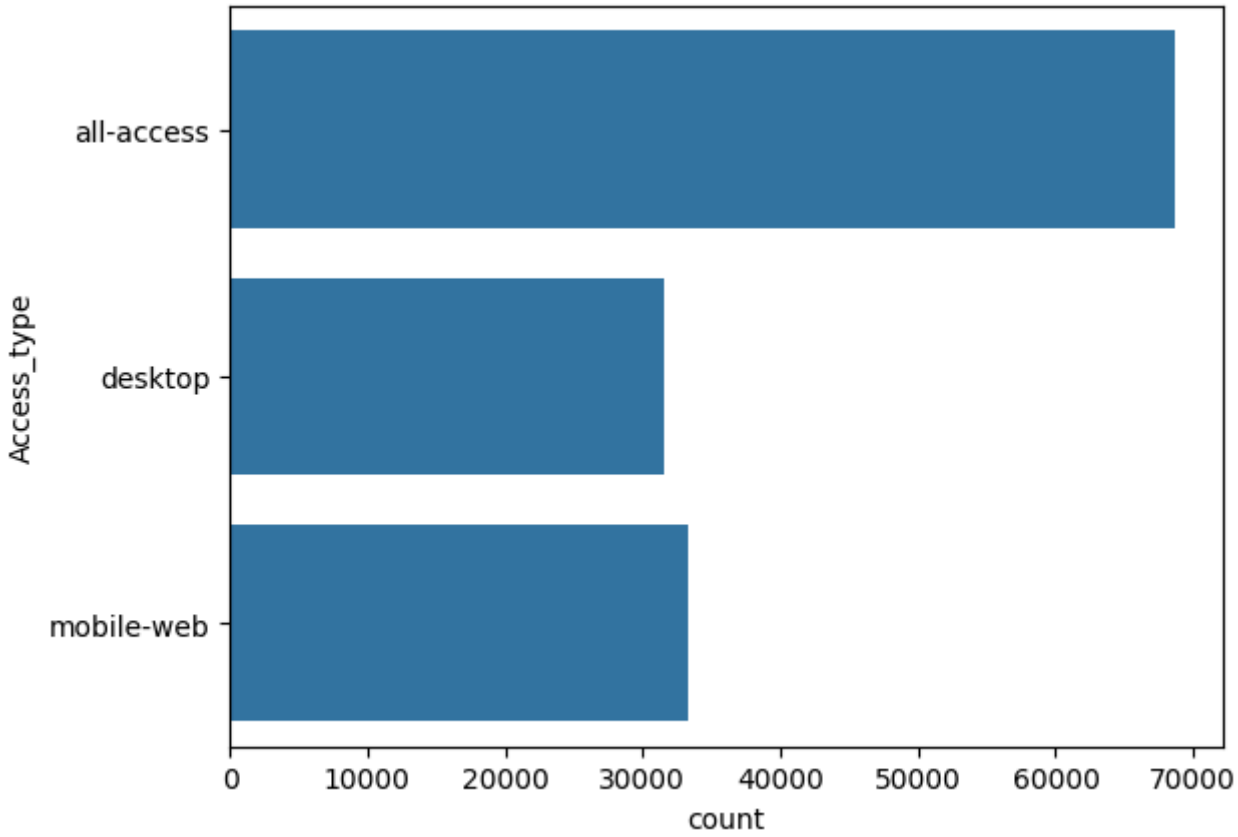
In [45]:
```
sns.countplot(df_2['Language'])
```

Out[45]: <Axes: xlabel='count', ylabel='Language'>



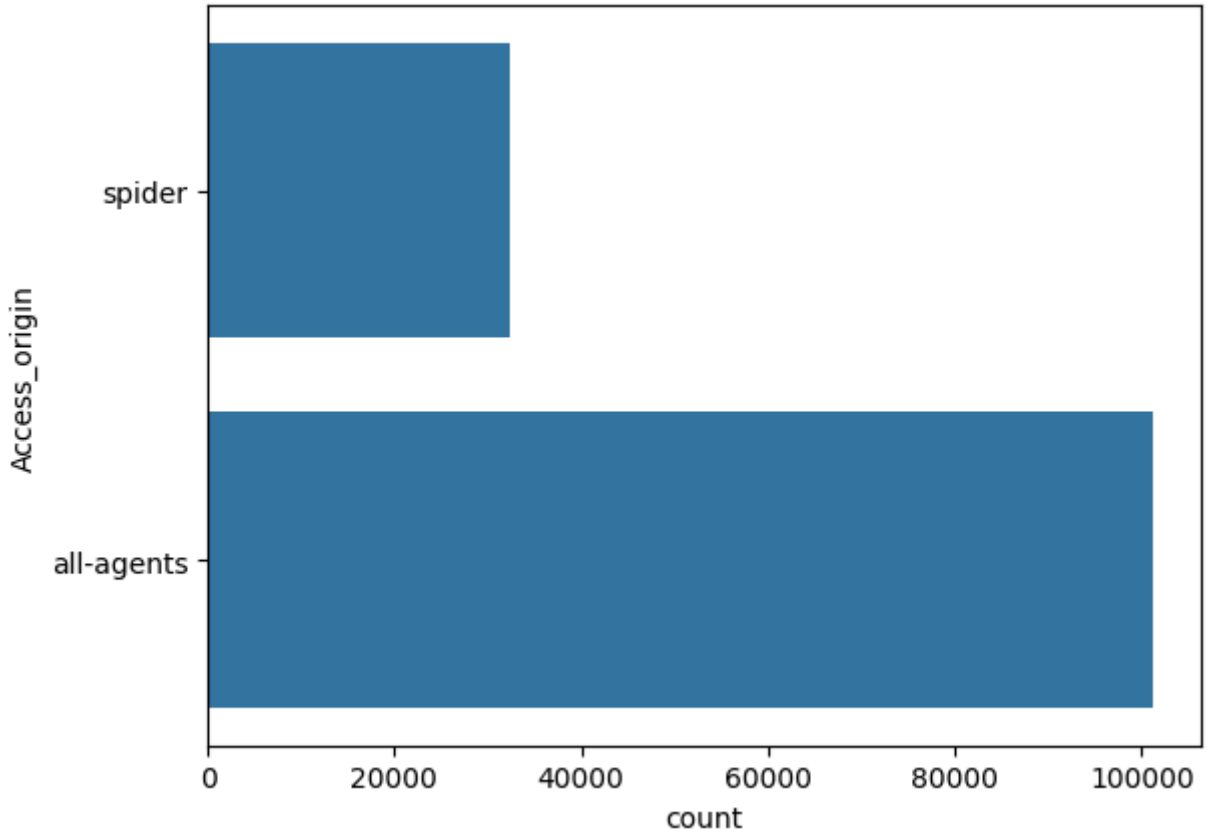
In [46]:
```
sns.countplot(df_2['Access_type'])
```

Out[46]: <Axes: xlabel='count', ylabel='Access_type'>



In [47]:
```
sns.countplot(df_2['Access_origin'])
```

Out[47]: <Axes: xlabel='count', ylabel='Access_origin'>



usage from desktop and mobile is almost the same

Access is orgainic and less from spider

More english content

In [48]:
```
df_2.shape
```

Out[48]:
```
(133617, 557)
```

In [49]:
```
df_2.groupby('Language').count()
```

Out[49]:

Language	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	...	2016-12-26	2016-12-27	2016-12-28	2016-12-29	2016-12-30	2016-12-31	index	Title	Access_t
de	17362	17362	17362	17362	17362	17362	17362	17362	17362	17362	...	17362	17362	17362	17362	17362	17362	17362	17362	17
en	22486	22486	22486	22486	22486	22486	22486	22486	22486	22486	...	22486	22486	22486	22486	22486	22486	22486	22486	22
es	13551	13551	13551	13551	13551	13551	13551	13551	13551	13551	...	13551	13551	13551	13551	13551	13551	13551	13551	13
fr	16948	16948	16948	16948	16948	16948	16948	16948	16948	16948	...	16948	16948	16948	16948	16948	16948	16948	16948	16
ja	19295	19295	19295	19295	19295	19295	19295	19295	19295	19295	...	19295	19295	19295	19295	19295	19295	19295	19295	19
ns	8266	8266	8266	8266	8266	8266	8266	8266	8266	8266	...	8266	8266	8266	8266	8266	8266	8266	8266	8
ru	14270	14270	14270	14270	14270	14270	14270	14270	14270	14270	...	14270	14270	14270	14270	14270	14270	14270	14270	14
ww	6228	6228	6228	6228	6228	6228	6228	6228	6228	6228	...	6228	6228	6228	6228	6228	6228	6228	6228	6
zh	15211	15211	15211	15211	15211	15211	15211	15211	15211	15211	...	15211	15211	15211	15211	15211	15211	15211	15211	15

9 rows × 556 columns

In [50]:
```
df_2.drop(columns=["Page","index"],inplace=True)
```

In [51]:
```
df_2
```

Out[51]:

	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	...	2016-12-26	2016-12-27	2016-12-28	2016-12-29	2016-12-30	2016-12-31	Title	Language	A
0	18.0	11.0	5.0	13.0	14.0	9.0	9.0	22.0	26.0	24.0	...	14.0	20.0	22.0	19.0	18.0	20.0	2NE1	zh	
1	11.0	14.0	15.0	18.0	11.0	13.0	22.0	11.0	10.0	4.0	...	9.0	30.0	52.0	45.0	26.0	20.0	2PM	zh	
2	1.0	0.0	1.0	1.0	0.0	4.0	0.0	3.0	4.0	4.0	...	4.0	4.0	6.0	3.0	4.0	17.0	3C	zh	
3	35.0	13.0	10.0	94.0	4.0	26.0	14.0	9.0	11.0	16.0	...	16.0	11.0	17.0	19.0	10.0	11.0	4minute	zh	
4	12.0	7.0	4.0	5.0	20.0	8.0	5.0	17.0	24.0	7.0	...	32.0	19.0	23.0	17.0	17.0	50.0	5566	zh	

	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	...	2016-12-26	2016-12-27	2016-12-28	2016-12-29	2016-12-30	2016-12-31	Title	Language
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
133612	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	2.0	4.0	4.0	1.0	2.0	2.0	Legión (Marvel Comics)	es
133613	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	6.0	3.0	3.0	10.0	11.0	3.0	Referéndum sobre la permanencia del Reino Unid...	es
133614	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	43.0	8.0	22.0	13.0	18.0	14.0	Salida del Reino Unido de la Unión Europea	es
133615	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	8.0	7.0	5.0	43.0	12.0	25.0	Amar, después de amar	es
133616	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	1.0	0.0	0.0	1.0	1.0	0.0	Anexo:89 º Premios Óscar	es

133617 rows × 554 columns

In [52]:
```
df_v1=df_2.drop(columns=["Access_type","Access_origin","Title"])
```

In [53]:
```
df_v1
```

Out[53]:

	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	...	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	Lan
0	18.0	11.0	5.0	13.0	14.0	9.0	9.0	22.0	26.0	24.0	...	63.0	15.0	26.0	14.0	20.0	22.0	19.0	18.0	20.0	
1	11.0	14.0	15.0	18.0	11.0	13.0	22.0	11.0	10.0	4.0	...	42.0	28.0	15.0	9.0	30.0	52.0	45.0	26.0	20.0	

	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	...	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	Lar
2	1.0	0.0	1.0	1.0	0.0	4.0	0.0	3.0	4.0	4.0	...	1.0	1.0	7.0	4.0	4.0	6.0	3.0	4.0	17.0	
3	35.0	13.0	10.0	94.0	4.0	26.0	14.0	9.0	11.0	16.0	...	10.0	26.0	27.0	16.0	11.0	17.0	19.0	10.0	11.0	
4	12.0	7.0	4.0	5.0	20.0	8.0	5.0	17.0	24.0	7.0	...	27.0	8.0	17.0	32.0	19.0	23.0	17.0	17.0	50.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
133612	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	3.0	2.0	4.0	2.0	4.0	4.0	1.0	2.0	2.0	
133613	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	16.0	8.0	3.0	6.0	3.0	3.0	10.0	11.0	3.0	
133614	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	36.0	23.0	182.0	43.0	8.0	22.0	13.0	18.0	14.0	
133615	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	30.0	27.0	14.0	8.0	7.0	5.0	43.0	12.0	25.0	
133616	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	1.0	0.0	1.0	0.0	0.0	1.0	1.0	0.0	

133617 rows × 551 columns

In [54]:

```
df_language=df_v1.groupby('Language').mean().transpose()
df_language
```

Out[54]:

Language	de	en	es	fr	ja	ns	ru	ww	zh
2015-07-01	763.765926	3767.328604	1127.485204	499.092872	614.637160	137.917614	663.199229	56.036127	272.498521
2015-07-02	753.362861	3755.158765	1077.485425	502.297852	705.813216	142.449189	674.677015	61.494701	272.906778
2015-07-03	723.074415	3565.225696	990.895949	483.007553	637.451671	139.099685	625.329783	52.223988	271.097167
2015-07-04	663.537323	3711.782932	930.303151	516.275785	800.897435	115.011493	588.171829	49.482177	273.712379
2015-07-05	771.358657	3833.433025	1011.759575	506.871666	768.352319	127.912896	626.385354	54.235549	291.977713
...	...	...	...	...	...	...	...	...	...
2016-12-27	1119.596936	6314.335275	1070.923400	840.590217	808.541436	237.444108	998.374071	46.206326	363.066991
2016-12-28	1062.284069	6108.874144	1108.996753	783.585379	807.430163	228.058190	945.054730	112.867373	369.049701

Language	de	en	es	fr	ja	ns	ru	ww	zh
2016-12-29	1033.939062	6518.058525	1058.660320	763.209169	883.752786	227.904065	909.352207	48.636159	340.526330
2016-12-30	981.786430	5401.792360	807.551177	710.502773	979.278777	227.333777	815.475123	61.438022	342.745316
2016-12-31	937.842875	5280.643467	776.934322	654.060656	1228.720808	195.896564	902.600210	56.019428	352.184275

550 rows × 9 columns

In [55]:
```
df_v1.groupby('Language').mean()
```

Out[55]:

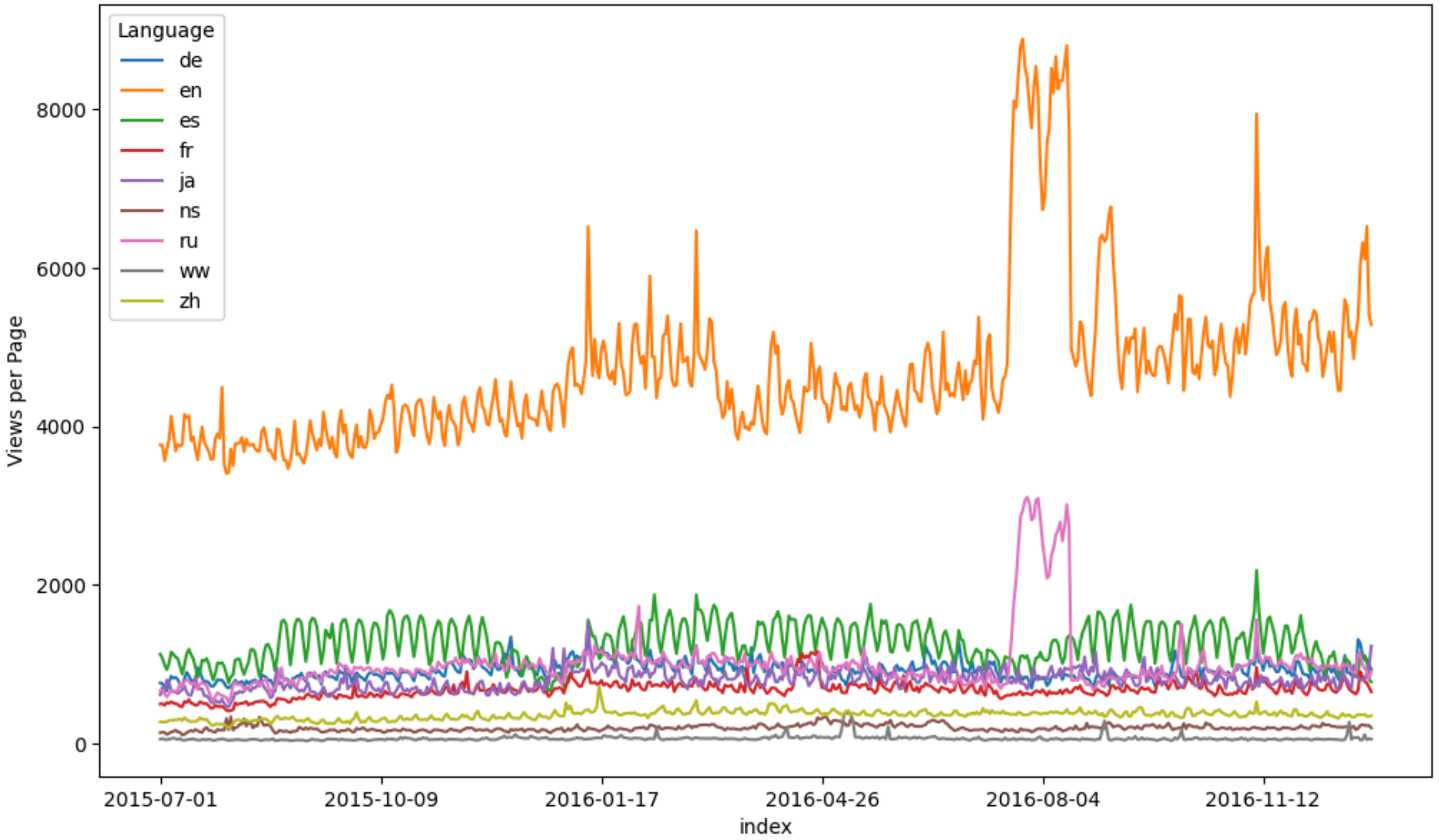
Language	2015-07-01	2015-07-02	2015-07-03	2015-07-04	2015-07-05	2015-07-06	2015-07-07	2015-07-08	2015-07-09	2015-07-10	...	2016-12-
de	763.765926	753.362861	723.074415	663.537323	771.358657	849.080636	823.422532	835.455305	804.329513	749.593653	...	862.5667
en	3767.328604	3755.158765	3565.225696	3711.782932	3833.433025	4127.429067	3906.341724	3685.854621	3771.183714	3749.860313	...	5191.4534
es	1127.485204	1077.485425	990.895949	930.303151	1011.759575	1153.091801	1123.942440	1090.832190	1070.245812	972.952328	...	931.3328
fr	499.092872	502.297852	483.007553	516.275785	506.871666	528.072752	510.426481	501.033632	495.848124	467.943179	...	658.2314
ja	614.637160	705.813216	637.451671	800.897435	768.352319	669.544493	651.298938	647.372428	631.090645	655.678362	...	687.5593
ns	137.917614	142.449189	139.099685	115.011493	127.912896	145.855311	152.569320	168.919913	146.516453	140.302565	...	199.3631
ru	663.199229	674.677015	625.329783	588.171829	626.385354	674.764681	659.297617	656.209530	672.200981	769.787596	...	908.7786
ww	56.036127	61.494701	52.223988	49.482177	54.235549	62.849550	69.478484	59.679351	53.176943	63.997913	...	51.9629
zh	272.498521	272.906778	271.097167	273.712379	291.977713	293.490632	293.170337	300.824206	298.983433	310.819736	...	335.9481

9 rows × 550 columns

In [56]:
```
df_language.reset_index(inplace=True)
df_language.set_index('index', inplace=True)
```

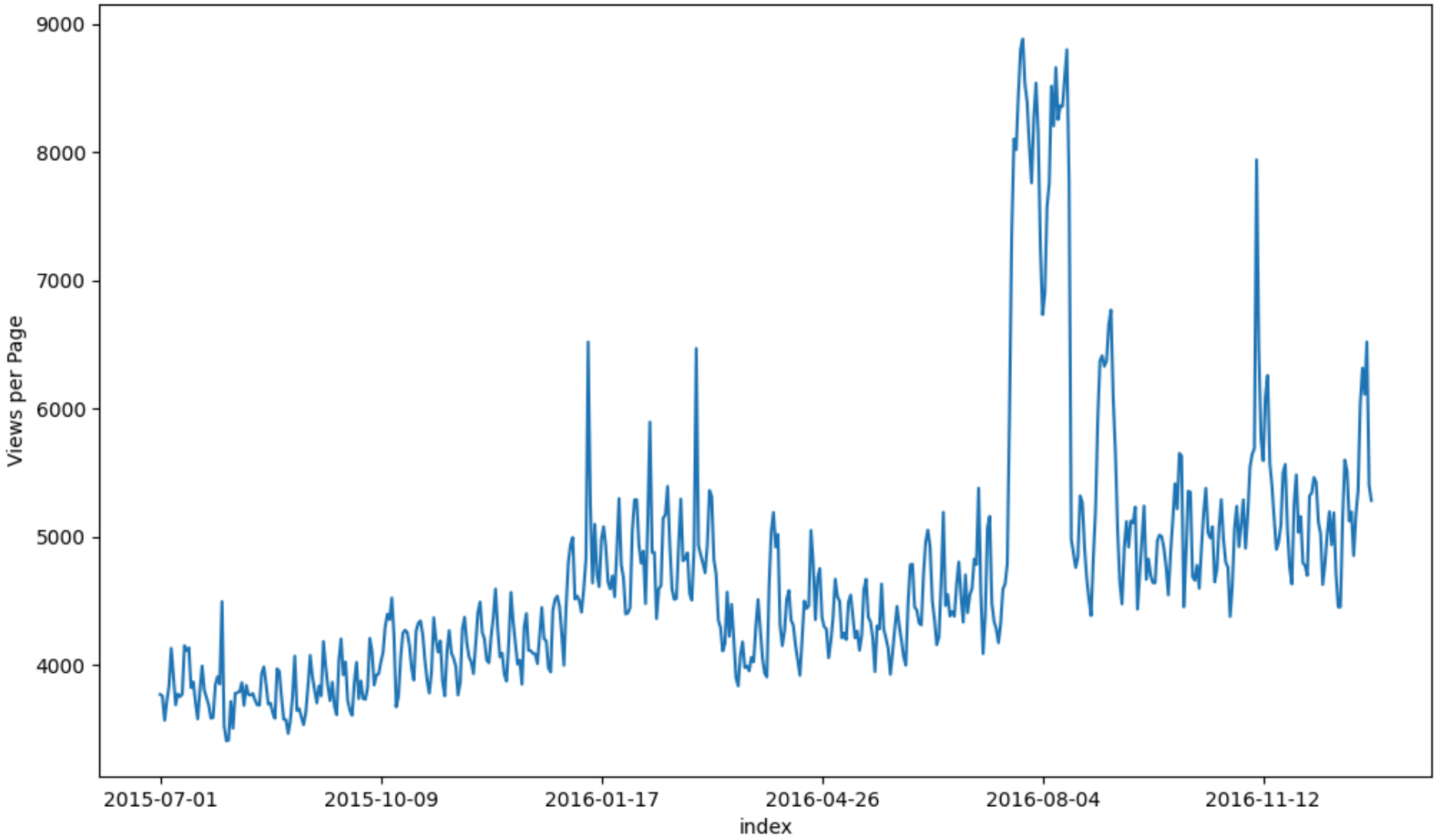
In [57]:
```
df_language.plot(figsize=(12,7))
plot.ylabel('Views per Page')
```

Out[57]:
```
Text(0, 0.5, 'Views per Page')
```



In [58]:
```
df_language['en'].plot(figsize=(12,7))
plot.ylabel('Views per Page')
```

Out[58]: Text(0, 0.5, 'Views per Page')



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

In [58]:

In [59]:
```
df_final=df_language.copy()
```

In [60]:
```
df_final.shape
```

Out[60]:
```
(550, 9)
```

In [60]:

Checking stationarity

Null Hypothesis (H0): The time series has a unit root; it is non-stationary.

Alternative Hypothesis (H1): The time series is stationary (with or without a trend).

if the p-value is less than 0.05, we reject the null hypothesis else we fail to reject null hypothesis

In [61]:
```
from statsmodels.tsa.stattools import adfuller
def aduf_test(x):
    result=adfuller(x)
    if(result[1]<0.05):
      print("stationary")
    else:
      print("not stationary")
    print('ADF Stastistic: %f'%result[0])
    print('p-value: %f'%result[1])
```

In [62]:
```
aduf_test(df_final['en'])
```

```
not stationary
ADF Stastistic: -2.373563
p-value: 0.149337
```

In [62]:

p value is greater than 0.05 so we fail to recject H0

this data is non-stationary

In [63]:
```
ts=df_final[['en']]
```

In [63]:
```

```

In [64]:
```
ts = ts.reset_index()
```

In [65]:
```
ts.index = pd.to_datetime(ts.index)
```

In [66]:
```
ts.set_index('index', inplace=True)
```

In [67]:
```
ts
```

Out[67]:

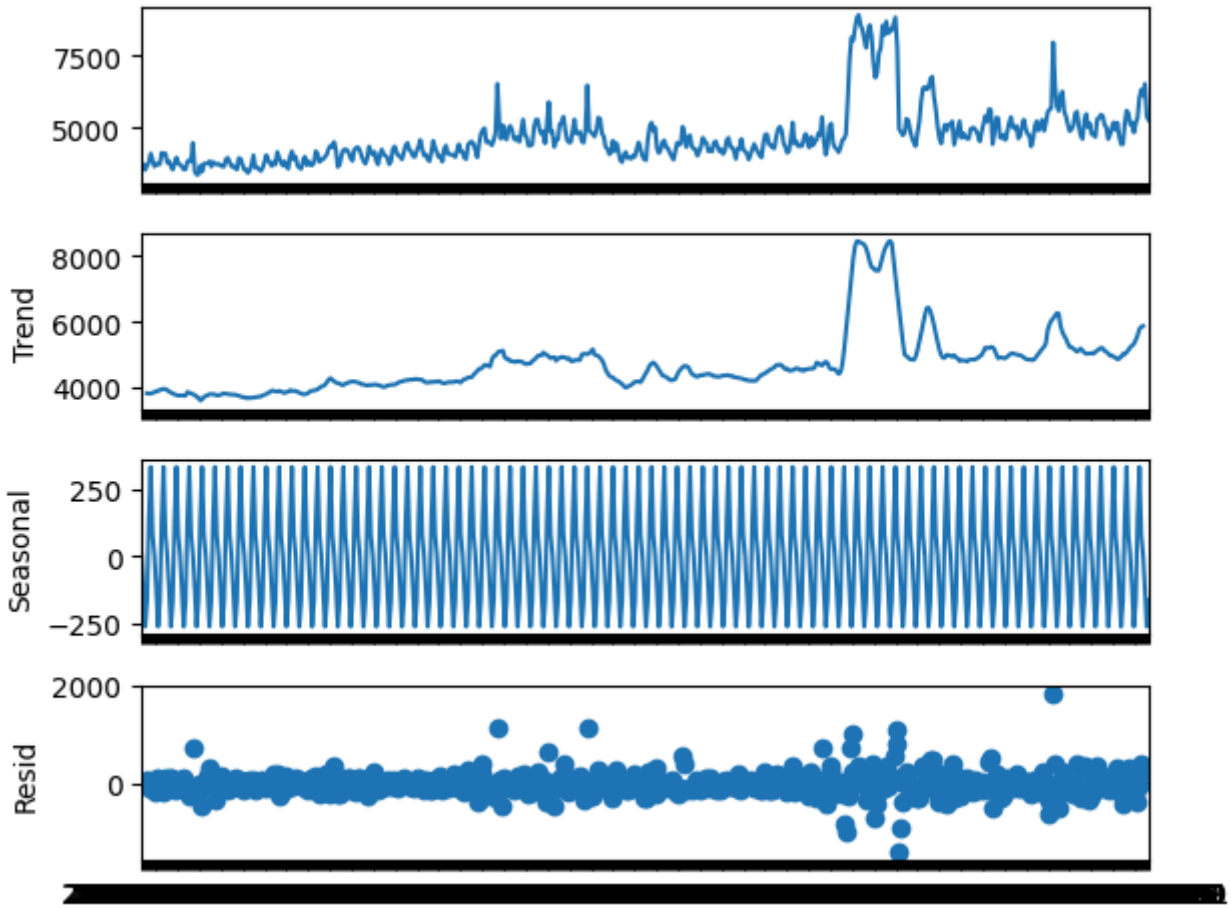
Language	en
index	
2015-07-01	3767.328604
2015-07-02	3755.158765
2015-07-03	3565.225696
2015-07-04	3711.782932
2015-07-05	3833.433025
...	...
2016-12-27	6314.335275
2016-12-28	6108.874144
2016-12-29	6518.058525

Language	en
index	
2016-12-30	5401.792360
2016-12-31	5280.643467

550 rows × 1 columns

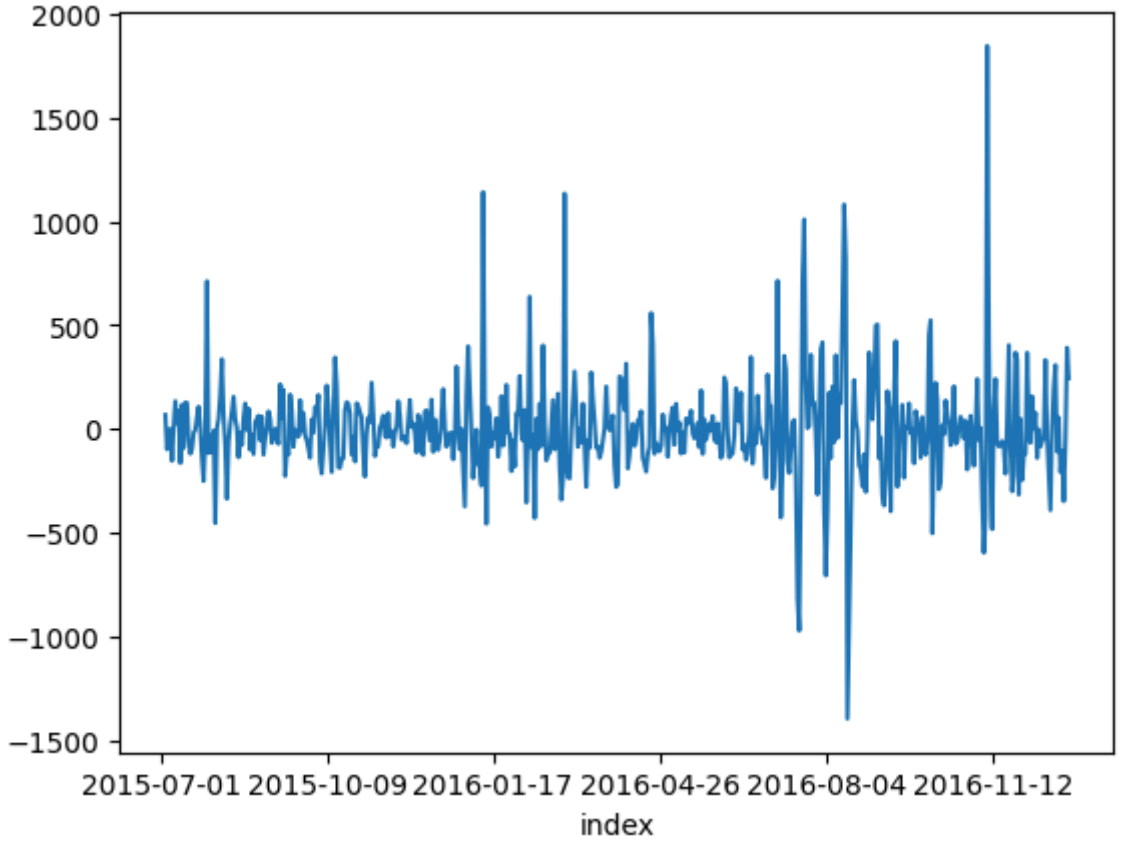
In [68]:

```
import statsmodels.api as sm
model = sm.tsa.seasonal_decompose(ts, period=7)
model.plot();
```



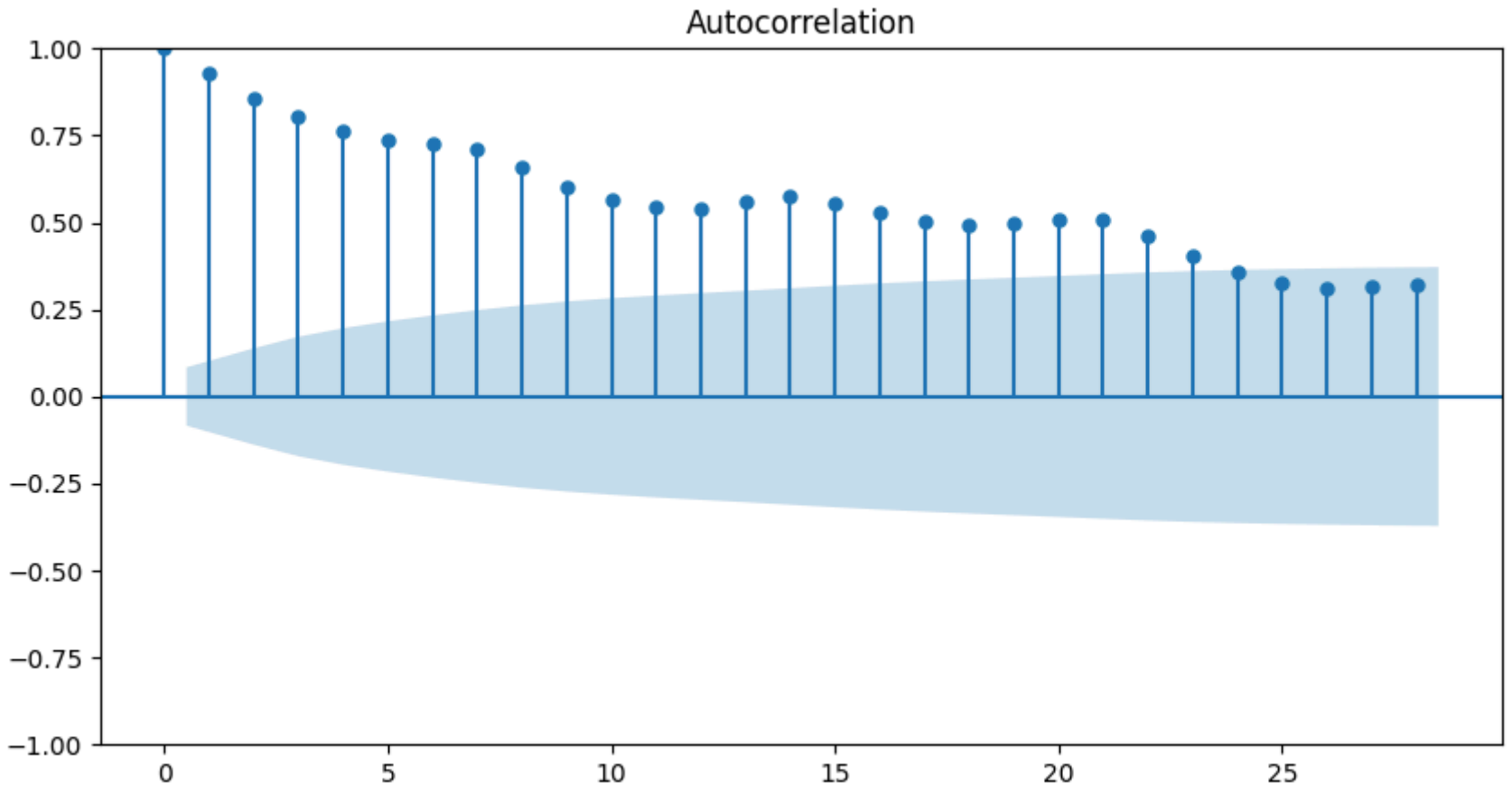
In [69]:
```
model.resid.plot()
```

Out[69]:
```
<Axes: xlabel='index'>
```



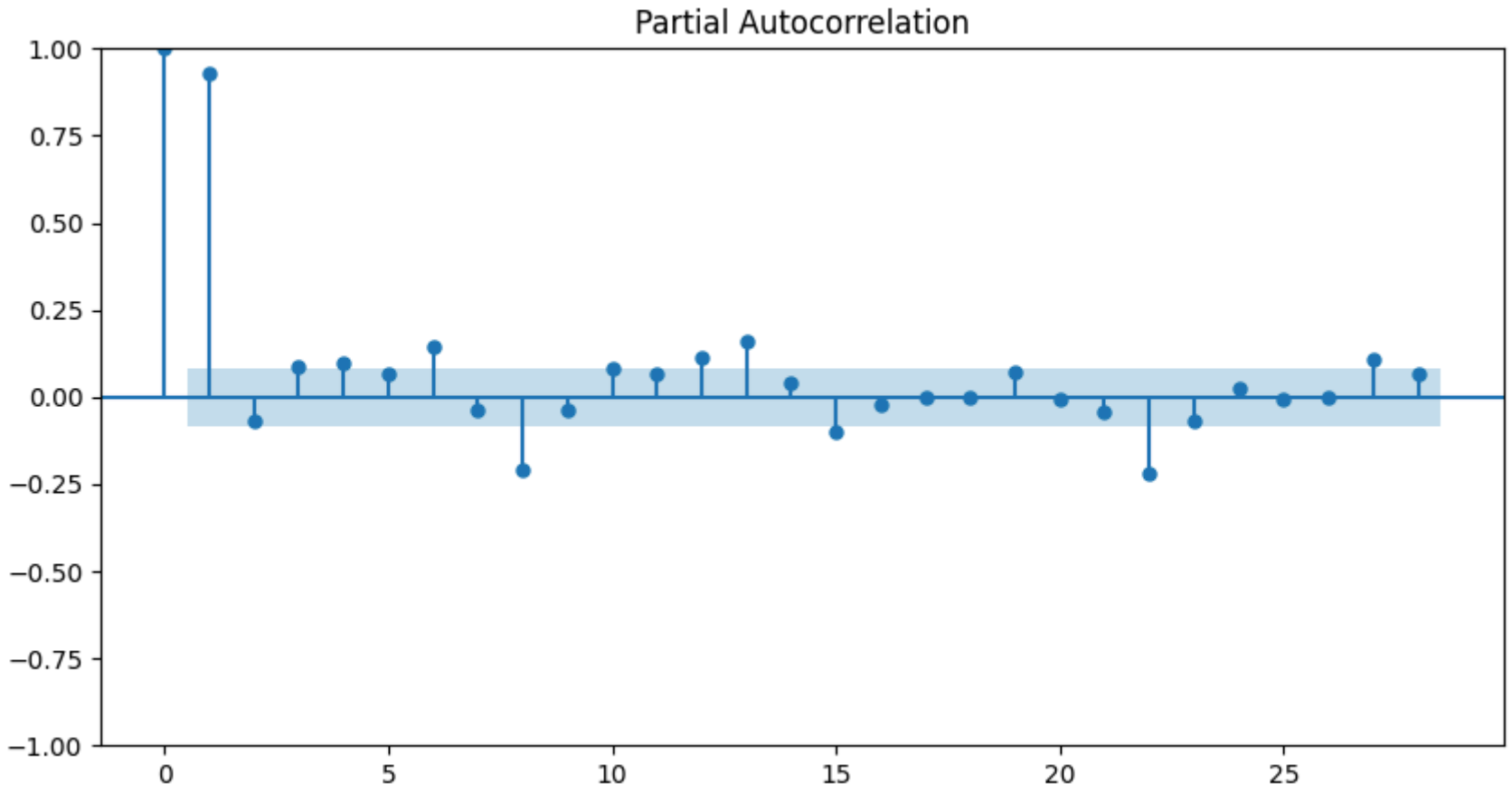
In [70]:

```
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
import matplotlib.pyplot as plt
fig, ax = plt.subplots(figsize=(10, 5))
acf=plot_acf(ts,ax=ax)
```



In [71]:

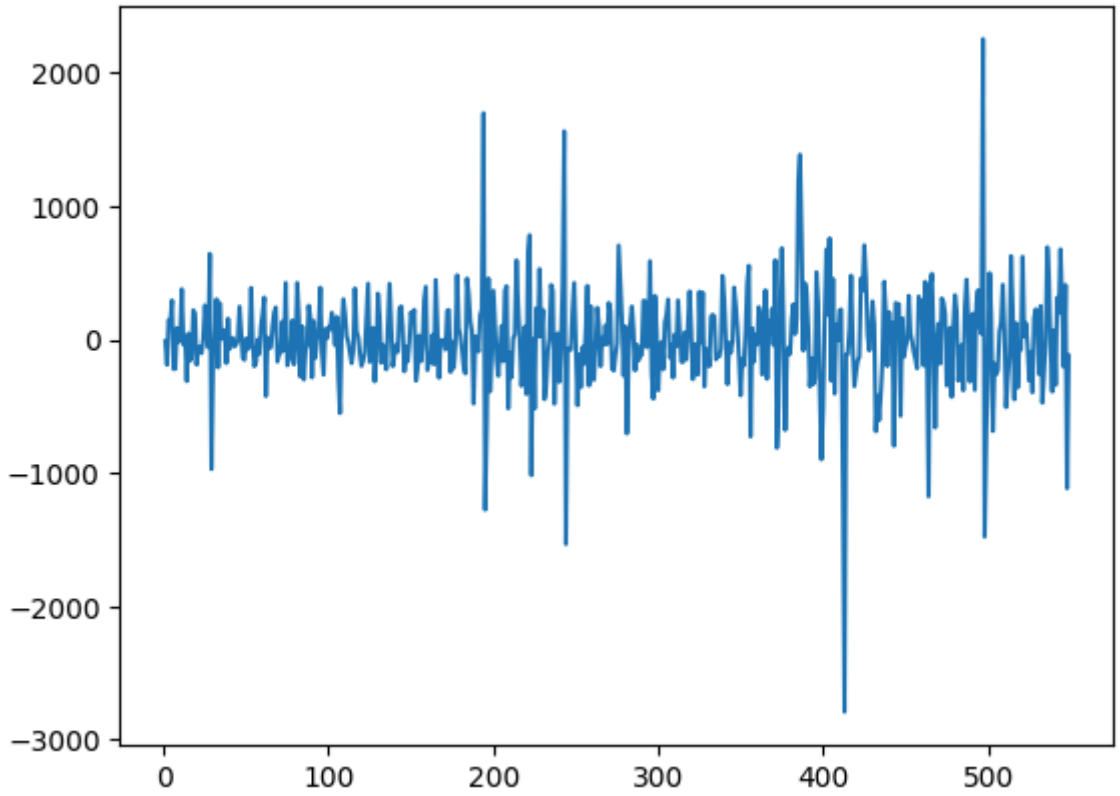
```
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
import matplotlib.pyplot as plt
fig, ax = plt.subplots(figsize=(10, 5))
pacf=plot_pacf(ts,ax=ax)
```



In [72]:
```
aduf_test(model.resid.dropna())
```

```
stationary
ADF Stastistic: -11.437316
p-value: 0.000000
```

In [73]:
```
ts_diff = ts - ts.shift(1)
plot.plot(ts_diff.values)
plot.show()
```



In [74]:
```
ts_diff.dropna(inplace=True)
aduf_test(ts_diff)
```

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

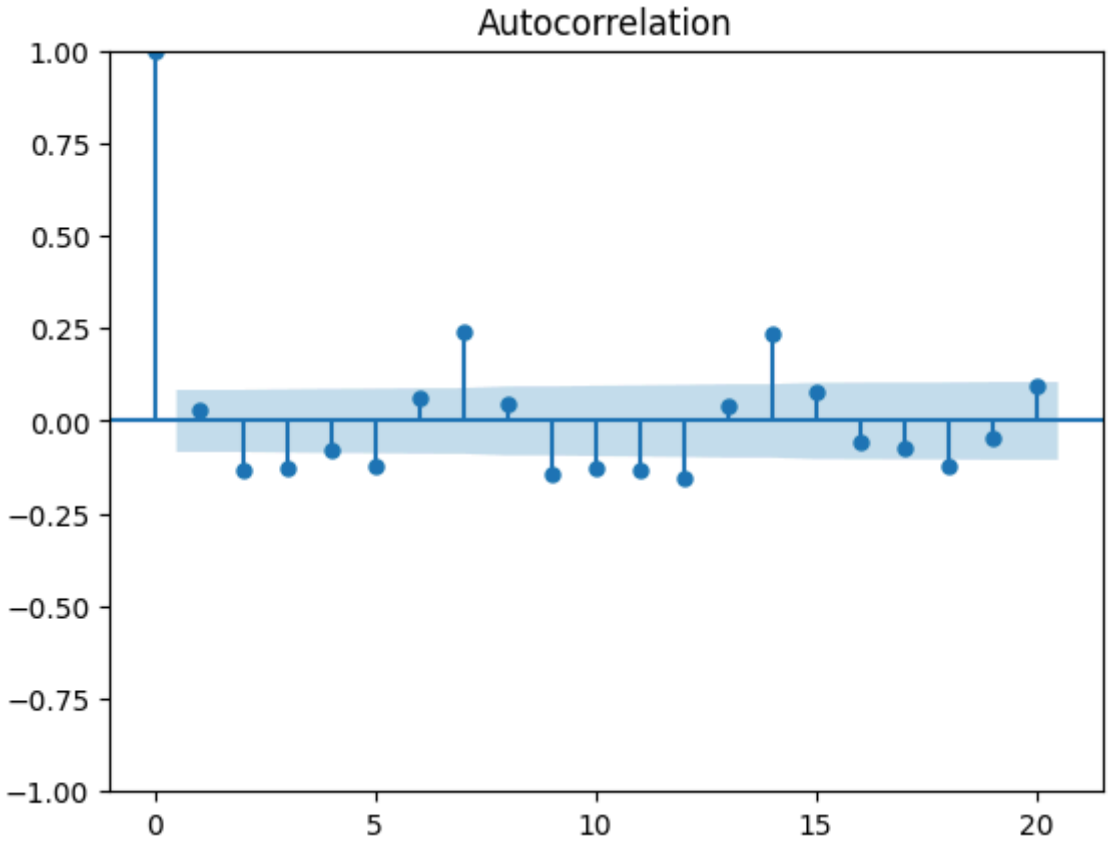
In [74]:

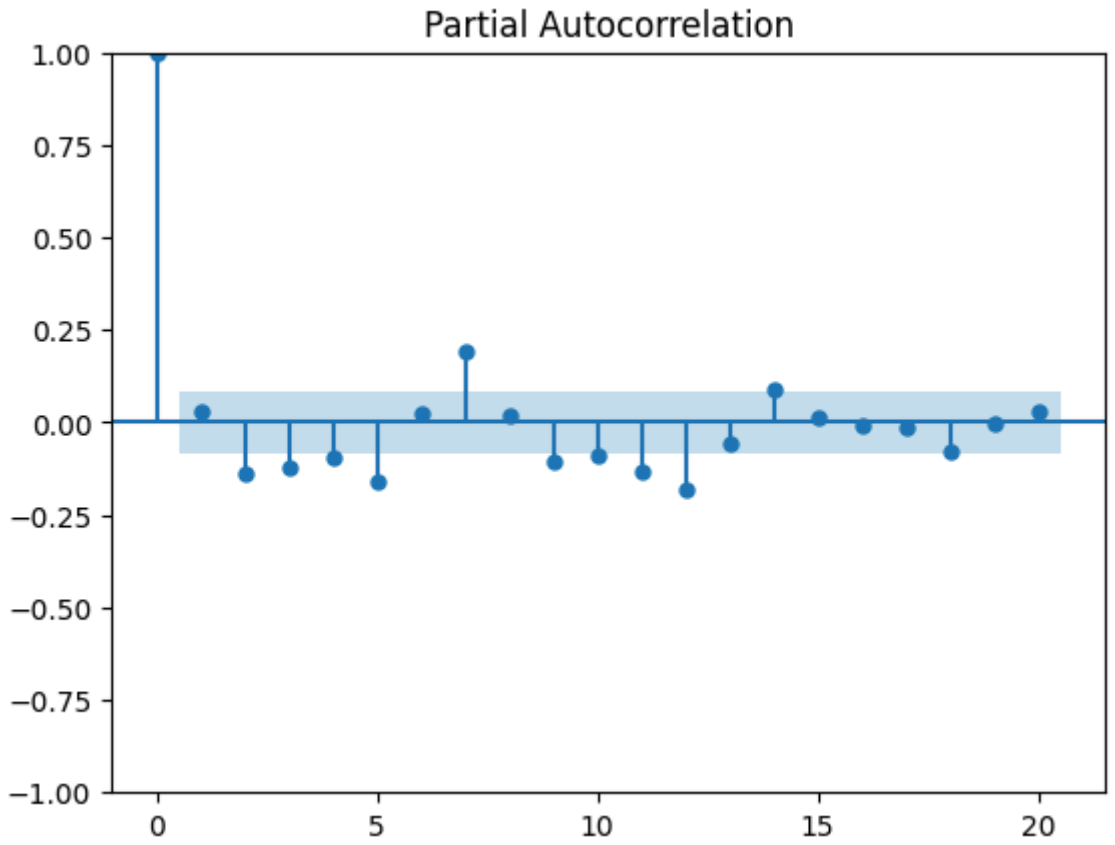
In [74]:

In [74]:

In [74]:

In [75]:
```
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
acf=plot_acf(ts_diff,lags=20)
pacf=plot_pacf(ts_diff,lags=20)
```





In [76]:
```
ts.diff(7) #
```

Out[76]:

Language	en
index	
2015-07-01	NaN
2015-07-02	NaN
2015-07-03	NaN
2015-07-04	NaN
2015-07-05	NaN

Language	en
index	
...	...
2016-12-27	803.065463
2016-12-28	988.209730
2016-12-29	1326.605088
2016-12-30	551.432358
2016-12-31	118.205016

550 rows × 1 columns

In [76]:

ARIMA Model (ARIMA(p,d,q)): When to use: Consider when the time series data exhibits trends or seasonality that require differencing to achieve stationarity. If the ACF and PACF plots show significant spikes, suggesting the need for both autoregressive (AR) and moving average (MA) terms.

AR Model (AR(p)): When to use: Use when the time series data exhibits significant autocorrelation at multiple lags, as indicated by significant spikes in the PACF plot. If the ACF plot decays slowly and the PACF plot has a sharp cutoff after p lags, suggesting the need for autoregressive terms only.

MA Model (MA(q)): When to use: Use when the time series data exhibits significant autocorrelation at multiple lags, as indicated by significant spikes in the ACF plot. If the PACF plot decays slowly and the ACF plot has a sharp cutoff after q lags, suggesting the need for moving average terms only.

ARMA Model (ARMA(p,q)): When to use: Use when the time series data is stationary but exhibits autocorrelation that can be captured by a combination of AR and MA terms. If either the ACF or PACF plot shows significant spikes, suggesting the need for either AR or MA terms, but not both.

In [76]:

In [76]:

In [77]:
```
ts
```

Out[77]:

Language	en
index	
2015-07-01	3767.328604
2015-07-02	3755.158765
2015-07-03	3565.225696
2015-07-04	3711.782932
2015-07-05	3833.433025
...	...
2016-12-27	6314.335275
2016-12-28	6108.874144
2016-12-29	6518.058525
2016-12-30	5401.792360
2016-12-31	5280.643467

550 rows × 1 columns

In [77]:

ARIMA MODEL

In [78]:

```
from statsmodels.tsa.statespace.sarimax import SARIMAX
from pandas import DataFrame
from statsmodels.tsa.arima.model import ARIMA
```

Multistep forecasting

In [79]:

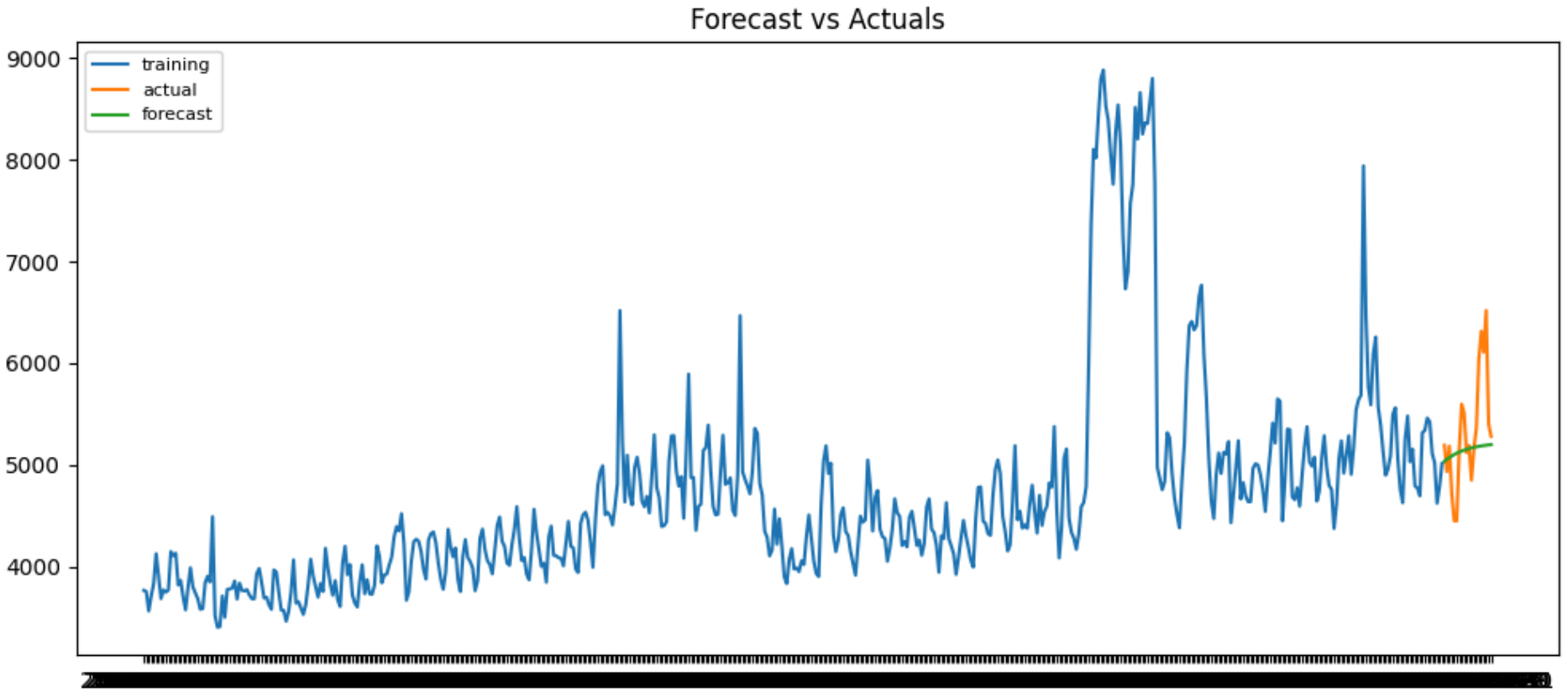
```
train = ts[:-20]
test = ts[-20:]
```

In [97]:
```
train.shape
```

Out[97]:
```
(530, 1)
```

In [80]:
```
model = ARIMA(train, order=(1, 1, 1))
fitted = model.fit()

# Forecast
fc = fitted.forecast(20, alpha=0.02)
fc_series = pd.Series(fc, index=test.index)
plot.figure(figsize=(12,5), dpi=100)
plot.plot(train, label='training')
plot.plot(test, label='actual')
plot.plot(fc_series, label='forecast')

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

def mean_absolute_percentage_error(y_true, y_pred):
    return np.mean(np.abs((y_true - y_pred) / y_true)) * 100

mape = mean_absolute_percentage_error(test.values, fc_series.values.reshape((-1,1)))
print("Mean Absolute Percentage Error (MAPE): {:.2f}%".format(mape))
```

```
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: No frequency information was provide
so inferred frequency D will be used.
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: No frequency information was provided,
so inferred frequency D will be used.
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: No frequency information was provided,
so inferred frequency D will be used.
  self._init_dates(dates, freq)
Mean Absolute Percentage Error (MAPE): 7.22%
```



In [81]:
```
test.values.shape
```

Out[81]:
```
(20, 1)
```

In [82]:
```
fc_series.values.reshape((-1,1)).shape
```

Out[82]:
```
(20, 1)
```

In [83]:
```
import warnings
warnings.filterwarnings("ignore")
p_values = range(5)  # Example: Test p values from 0 to 2
d_values = range(2)  # Example: Test d values from 0 to 2
q_values = range(5)  # Example: Test q values from 0 to 2
```

```python
# Iterate over different combinations of p, d, and q
for p in p_values:
    for d in d_values:
        for q in q_values:
            # Fit the ARIMA model
            model = ARIMA(train, order=(p, d, q))
            fitted = model.fit()

            # Forecast
            fc= fitted.forecast(20, alpha=0.05)  # alpha is the significance level (default is 0.05)
            fc_series = pd.Series(fc, index=test.index)

            # Calculate MAPE
            def mean_absolute_percentage_error(y_true, y_pred):
                return np.mean(np.abs((y_true - y_pred) / y_true)) * 100

            mape = mean_absolute_percentage_error(test.values, fc_series.values.reshape((-1,1)))
            print("MAPE for (p={}, d={}, q={}): {:.2f}%".format(p, d, q, mape))
```

```
MAPE for (p=0, d=0, q=0): 12.36%
MAPE for (p=0, d=0, q=1): 12.23%
MAPE for (p=0, d=0, q=2): 11.83%
MAPE for (p=0, d=0, q=3): 11.74%
MAPE for (p=0, d=0, q=4): 11.49%
MAPE for (p=0, d=1, q=0): 8.56%
MAPE for (p=0, d=1, q=1): 8.48%
MAPE for (p=0, d=1, q=2): 8.96%
MAPE for (p=0, d=1, q=3): 8.66%
MAPE for (p=0, d=1, q=4): 8.64%
MAPE for (p=1, d=0, q=0): 10.48%
MAPE for (p=1, d=0, q=1): 10.62%
MAPE for (p=1, d=0, q=2): 10.40%
MAPE for (p=1, d=0, q=3): 9.97%
MAPE for (p=1, d=0, q=4): 9.97%
MAPE for (p=1, d=1, q=0): 8.50%
MAPE for (p=1, d=1, q=1): 7.22%
MAPE for (p=1, d=1, q=2): 8.22%
MAPE for (p=1, d=1, q=3): 7.25%
MAPE for (p=1, d=1, q=4): 7.58%
MAPE for (p=2, d=0, q=0): 10.60%
MAPE for (p=2, d=0, q=1): 10.61%
MAPE for (p=2, d=0, q=2): 9.77%
MAPE for (p=2, d=0, q=3): 9.95%
MAPE for (p=2, d=0, q=4): 9.61%
MAPE for (p=2, d=1, q=0): 8.89%
```

```
MAPE for (p=2, d=1, q=1): 8.34%
MAPE for (p=2, d=1, q=2): 8.20%
MAPE for (p=2, d=1, q=3): 8.99%
MAPE for (p=2, d=1, q=4): 8.96%
MAPE for (p=3, d=0, q=0): 10.46%
MAPE for (p=3, d=0, q=1): 9.91%
MAPE for (p=3, d=0, q=2): 10.67%
MAPE for (p=3, d=0, q=3): 9.34%
MAPE for (p=3, d=0, q=4): 8.97%
MAPE for (p=3, d=1, q=0): 8.85%
MAPE for (p=3, d=1, q=1): 7.24%
MAPE for (p=3, d=1, q=2): 8.36%
MAPE for (p=3, d=1, q=3): 8.87%
MAPE for (p=3, d=1, q=4): 9.06%
MAPE for (p=4, d=0, q=0): 10.24%
MAPE for (p=4, d=0, q=1): 9.92%
MAPE for (p=4, d=0, q=2): 10.76%
MAPE for (p=4, d=0, q=3): 11.03%
MAPE for (p=4, d=0, q=4): 8.73%
MAPE for (p=4, d=1, q=0): 8.80%
MAPE for (p=4, d=1, q=1): 8.62%
MAPE for (p=4, d=1, q=2): 7.55%
MAPE for (p=4, d=1, q=3): 8.97%
MAPE for (p=4, d=1, q=4): 9.96%
```

In [83]:

best model ARIMA (p=1, d=1, q=1): 7.22%

In [83]:

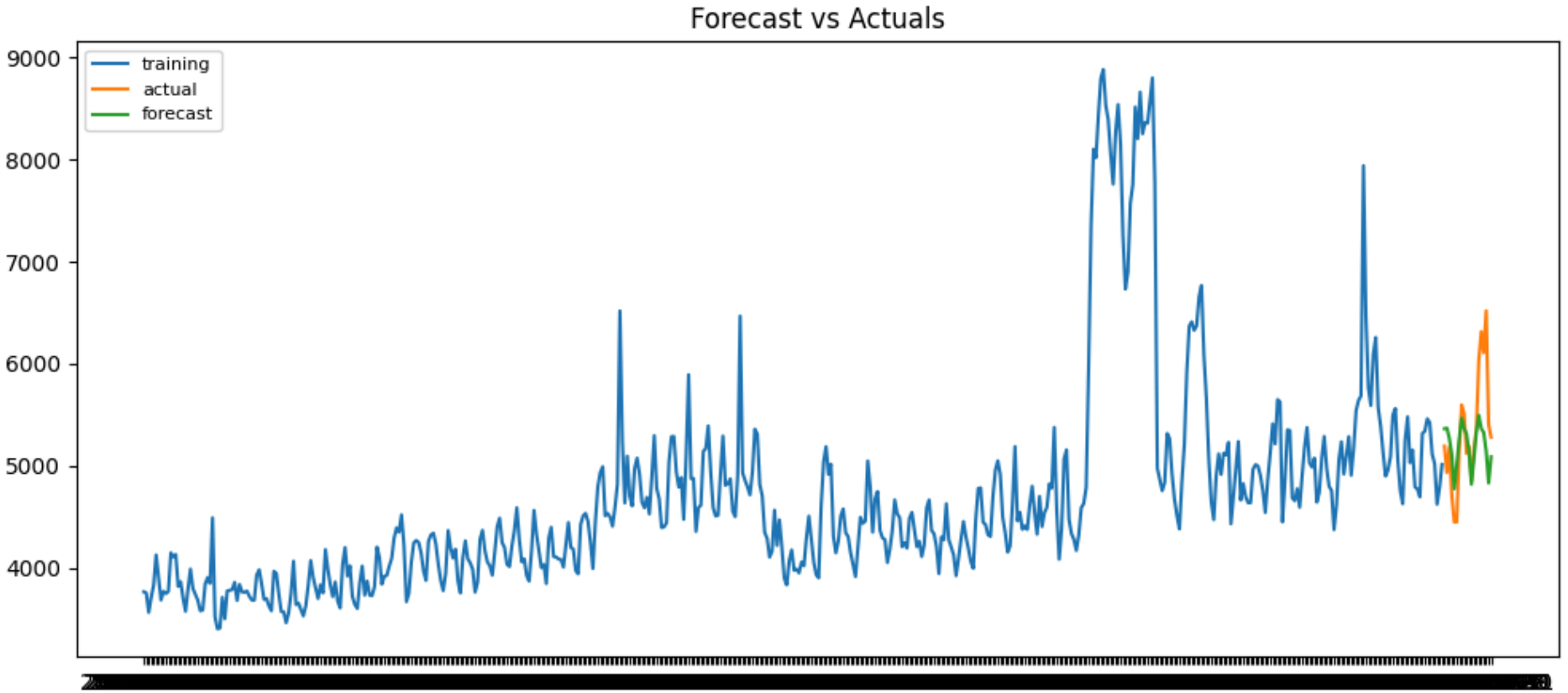
SARIMA

In [84]:

```python
order = (0, 0, 2)  # Order of non-seasonal ARIMA components (p, d, q)
seasonal_order = (1, 1, 1, 7)  # Order of seasonal ARIMA components (P, D, Q, S)

model = SARIMAX(train, order=order, seasonal_order=seasonal_order)
fitted = model.fit()
# Forecast
fc = fitted.forecast(20, alpha=0.02)
fc_series = pd.Series(fc, index=test.index)
```

```
plot.figure(figsize=(12,5), dpi=100)
plot.plot(train, label='training')
plot.plot(test, label='actual')
plot.plot(fc_series, label='forecast')

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

def mean_absolute_percentage_error(y_true, y_pred):
    return np.mean(np.abs((y_true - y_pred) / y_true)) * 100

mape = mean_absolute_percentage_error(test.values, fc_series.values.reshape((-1,1)))
print("Mean Absolute Percentage Error (MAPE): {:.2f}%".format(mape))
```

```
Mean Absolute Percentage Error (MAPE): 6.46%
```



In [86]:

In [ ]:

```
```

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

In [91]:

```
!ls
```

```
Exog_Campaign_eng  sample_data  train_1.csv
```

In [92]:

```
! touch Exog_Campaign_eng
```

In [94]:

```
ex_df = pd.read_csv('Exog_Campaign_eng.csv')
ex_df.head()
```

Out[94]:

	Exog
0	0
1	0
2	0
3	0
4	0

In [95]:

```
ex_df.shape
```

Out[95]:

```
(550, 1)
```

In [96]:

```
ts.shape
```

Out[96]:

```
(550, 1)
```

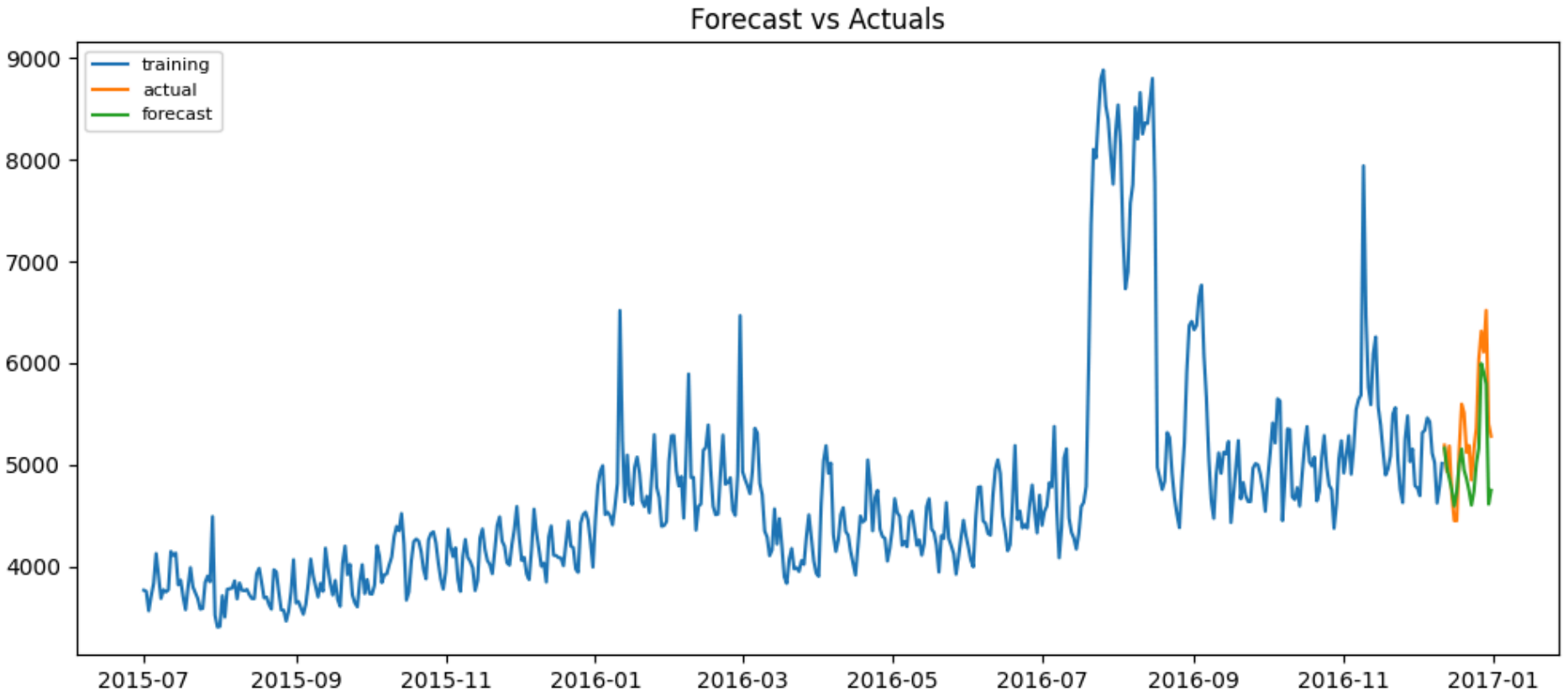
In [99]:

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

In [100…

```
import statsmodels.api as sm
train=ts[:530]
test=ts[530:]
model=sm.tsa.statespace.SARIMAX(train,order=(1, 1, 1),seasonal_order=(1,1,1,7),exog=exog[:530])
results=model.fit()

fc=results.forecast(20,dynamic=True,exog=pd.DataFrame(exog[530:]))
fc_series = pd.Series(fc)
train.index=train.index.astype('datetime64[ns]')
test.index=test.index.astype('datetime64[ns]')
plot.figure(figsize=(12,5), dpi=100)
plot.plot(train, label='training')
plot.plot(test, label='actual')
plot.plot(fc_series, label='forecast')
plot.title('Forecast vs Actuals')
plot.legend(loc='upper left', fontsize=8)
def mean_absolute_percentage_error(y_true, y_pred):
    return np.mean(np.abs((y_true - y_pred) / y_true)) * 100

mape = mean_absolute_percentage_error(test.values, fc_series.values.reshape((-1,1)))
print("Mean Absolute Percentage Error (MAPE): {:.2f}%".format(mape))
```

```
Mean Absolute Percentage Error (MAPE): 6.53%
```

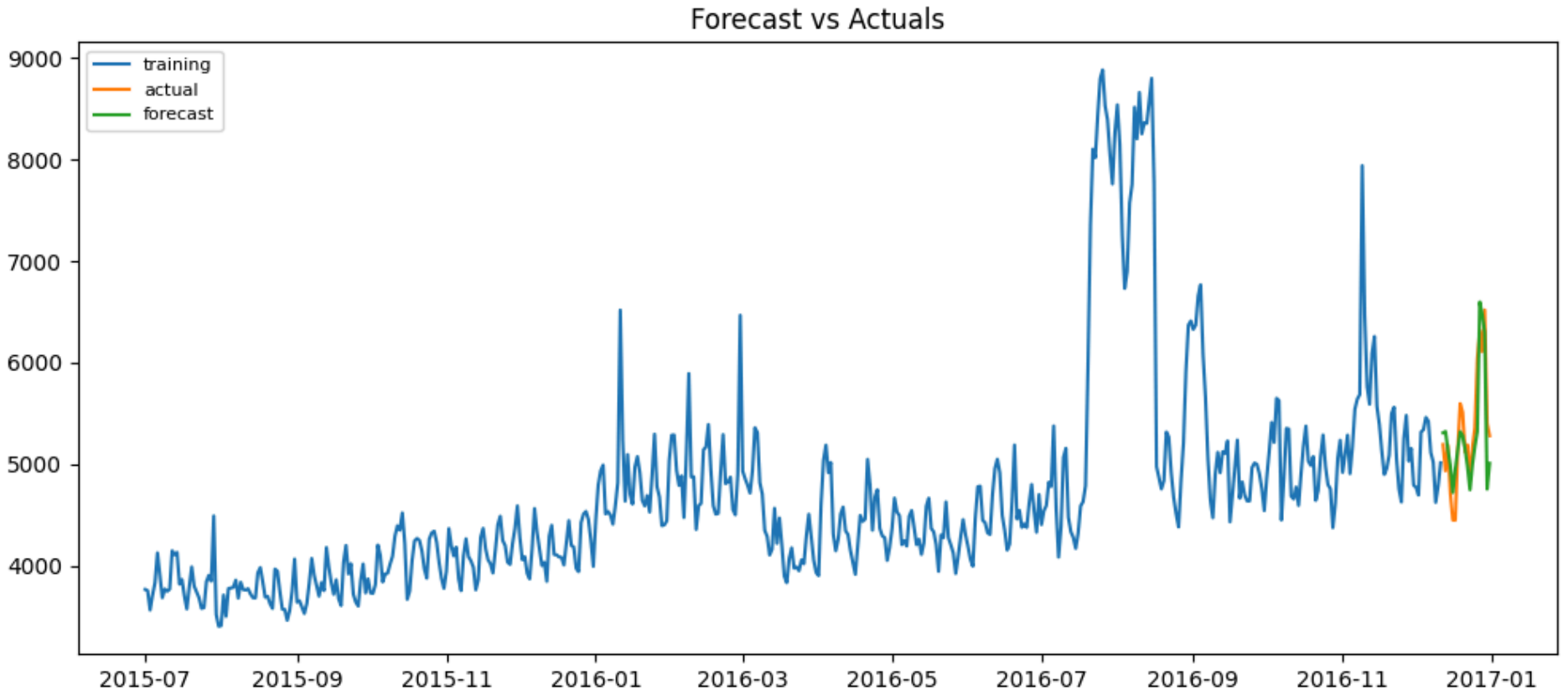


In [101…

```
order = (0, 0, 2)  # Order of non-seasonal ARIMA components (p, d, q)
seasonal_order = (1, 1, 1, 7)  # Order of seasonal ARIMA components (P, D, Q, S)


import statsmodels.api as sm
train=ts[:530]
test=ts[530:]
model=sm.tsa.statespace.SARIMAX(train,order=order,seasonal_order=seasonal_order,exog=exog[:530])
results=model.fit()

fc=results.forecast(20,dynamic=True,exog=pd.DataFrame(exog[530:]))
fc_series = pd.Series(fc)
train.index=train.index.astype('datetime64[ns]')
test.index=test.index.astype('datetime64[ns]')
plot.figure(figsize=(12,5), dpi=100)
plot.plot(train, label='training')
```

```
plot.plot(test, label='actual')
plot.plot(fc_series, label='forecast')
plot.title('Forecast vs Actuals')
plot.legend(loc='upper left', fontsize=8)
def mean_absolute_percentage_error(y_true, y_pred):
    return np.mean(np.abs((y_true - y_pred) / y_true)) * 100

mape = mean_absolute_percentage_error(test.values, fc_series.values.reshape((-1,1)))
print("Mean Absolute Percentage Error (MAPE): {:.2f}%".format(mape))
```

```
Mean Absolute Percentage Error (MAPE): 4.98%
```



In [ ]:

we have build our final model SARIMAX with order = (0, 0, 2) # Order of non-seasonal ARIMA components (p, d, q) seasonal_order = (1, 1, 1, 7)

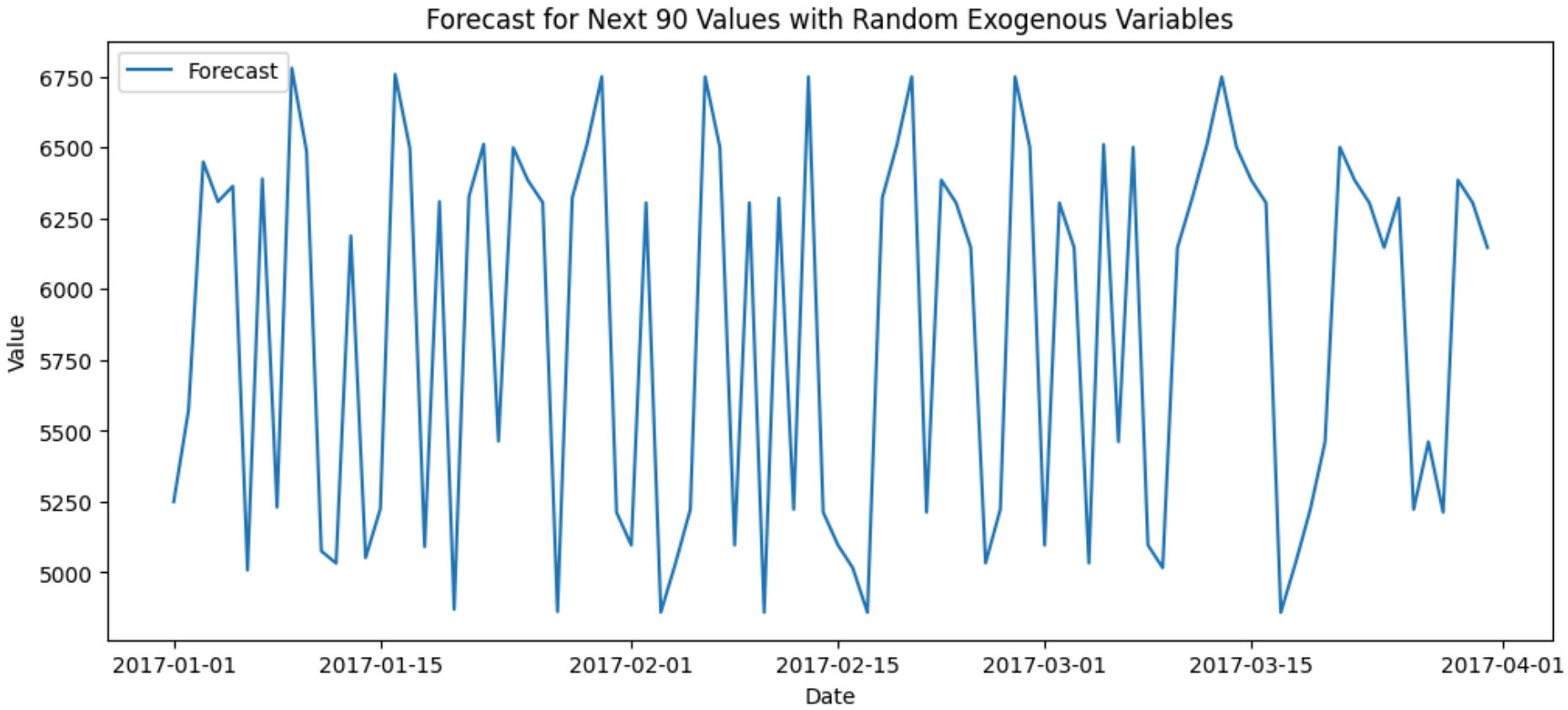
Mean Absolute Percentage Error (MAPE): 4.98%

In [ ]:

FINAL MODEL WITH ALL TRAINING DATA AND PREDECTING FOR NEXT 3 MONTHS

In [110…

```
import statsmodels.api as sm

train = ts  # Using all data for training
exog_train = exog  # Assuming exogenous variables are available for all training data

order = (0, 0, 2)  # Order of non-seasonal ARIMA components (p, d, q)
seasonal_order = (1, 1, 1, 7)  # Order of seasonal ARIMA components (P, D, Q, S)

model = sm.tsa.statespace.SARIMAX(train, order=order, seasonal_order=seasonal_order, exog=exog_train)
results = model.fit()

# Forecast for next 90 values
random_exog = np.random.randint(2, size=90)

# Forecast for next 90 values with random exogenous variables
forecast_values = results.forecast(steps=90, dynamic=True, exog=pd.DataFrame(random_exog))

# Plotting forecast
forecast_index = pd.date_range(start=train.index[-1], periods=91, freq='D')[1:]  # Generating dates for forecast
forecast_series = pd.Series(forecast_values, index=forecast_index)

plt.figure(figsize=(12, 5), dpi=100)
plt.plot(forecast_series, label='Forecast')
plt.title('Forecast for Next 90 Values with Random Exogenous Variables')
plt.xlabel('Date')
plt.ylabel('Value')
plt.legend(loc='upper left')
plt.show()
```



In [110…

In [110…

In [ ]:

Facebook Prophet

In [111…

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

```
Collecting pystan~=2.14
  Downloading pystan-2.19.1.1.tar.gz (16.2 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 16.2/16.2 MB 38.8 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: Cython!=0.25.1,>=0.22 in /usr/local/lib/python3.10/dist-packages (from pystan~=2.14) (3.0.10)
Requirement already satisfied: numpy>=1.7 in /usr/local/lib/python3.10/dist-packages (from pystan~=2.14) (1.25.2)
Building wheels for collected packages: pystan
  Building wheel for pystan (setup.py) ... done
  Created wheel for pystan: filename=pystan-2.19.1.1-cp310-cp310-linux_x86_64.whl size=61975259 sha256=d2818a92fb8090bd39686e42199a
4d5bbd99ce9baebb2231cde3e65f3cfd3a19
  Stored in directory: /root/.cache/pip/wheels/3d/1c/94/4516243362eedbedad15ac4389691ee3bf2d45bec2639c9d8b
Successfully built pystan
Installing collected packages: pystan
Successfully installed pystan-2.19.1.1
Collecting fbprophet
  Downloading fbprophet-0.7.1.tar.gz (64 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 64.0/64.0 kB 2.1 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: Cython>=0.22 in /usr/local/lib/python3.10/dist-packages (from fbprophet) (3.0.10)
Collecting cmdstanpy==0.9.5 (from fbprophet)
  Downloading cmdstanpy-0.9.5-py3-none-any.whl (37 kB)
Requirement already satisfied: pystan>=2.14 in /usr/local/lib/python3.10/dist-packages (from fbprophet) (2.19.1.1)
Requirement already satisfied: numpy>=1.15.4 in /usr/local/lib/python3.10/dist-packages (from fbprophet) (1.25.2)
Requirement already satisfied: pandas>=1.0.4 in /usr/local/lib/python3.10/dist-packages (from fbprophet) (2.0.3)
Requirement already satisfied: matplotlib>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from fbprophet) (3.7.1)
Collecting LunarCalendar>=0.0.9 (from fbprophet)
  Downloading LunarCalendar-0.0.9-py2.py3-none-any.whl (18 kB)
Collecting convertdate>=2.1.2 (from fbprophet)
  Downloading convertdate-2.4.0-py3-none-any.whl (47 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 47.9/47.9 kB 6.0 MB/s eta 0:00:00
Requirement already satisfied: holidays>=0.10.2 in /usr/local/lib/python3.10/dist-packages (from fbprophet) (0.47)
Collecting setuptools-git>=1.2 (from fbprophet)
  Downloading setuptools_git-1.2-py2.py3-none-any.whl (10 kB)
Requirement already satisfied: python-dateutil>=2.8.0 in /usr/local/lib/python3.10/dist-packages (from fbprophet) (2.8.2)
Requirement already satisfied: tqdm>=4.36.1 in /usr/local/lib/python3.10/dist-packages (from fbprophet) (4.66.2)
Collecting pymeeus<=1,>=0.3.13 (from convertdate>=2.1.2->fbprophet)
  Downloading PyMeeus-0.5.12.tar.gz (5.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 5.8/5.8 MB 74.3 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Collecting ephem>=3.7.5.3 (from LunarCalendar>=0.0.9->fbprophet)
  Downloading ephem-4.1.5-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.8/1.8 MB 60.0 MB/s eta 0:00:00
Requirement already satisfied: pytz in /usr/local/lib/python3.10/dist-packages (from LunarCalendar>=0.0.9->fbprophet) (2023.4)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=2.0.0->fbprophet) (1.
2.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=2.0.0->fbprophet) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=2.0.0->fbprophet) (4.
```

```
51.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=2.0.0->fbprophet) (1.
4.5)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=2.0.0->fbprophet) (24.
0)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=2.0.0->fbprophet) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=2.0.0->fbprophet) (3.
1.2)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.0.4->fbprophet) (2024.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.0->fbprophet) (1.16.
0)
Building wheels for collected packages: fbprophet, pymeeus
  error: subprocess-exited-with-error

  × python setup.py bdist_wheel did not run successfully.
  │ exit code: 1
  ╰─> See above for output.

  note: This error originates from a subprocess, and is likely not a problem with pip.
  Building wheel for fbprophet (setup.py) ... error
  ERROR: Failed building wheel for fbprophet
  Running setup.py clean for fbprophet
  Building wheel for pymeeus (setup.py) ... done
  Created wheel for pymeeus: filename=PyMeeus-0.5.12-py3-none-any.whl size=732001 sha256=3f52376f51eef3e4273bad1a98682acdd08421b2e3
2af7e9b2a4c1607ea5040b
  Stored in directory: /root/.cache/pip/wheels/d6/67/78/aa2e8d108639dd23a5e9e72a4fc88bb44f5541894382712f48
Successfully built pymeeus
Failed to build fbprophet
ERROR: Could not build wheels for fbprophet, which is required to install pyproject.toml-based projects
```

In [112…

```
ts_df = ts.reset_index().copy()
ts_df.columns = [['ds', 'y']]
```

In [113…

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

Out[113…

	ds	y	exog
0	2015-07-01	3767.328604	0
1	2015-07-02	3755.158765	0

	ds	y	exog
2	2015-07-03	3565.225696	0
3	2015-07-04	3711.782932	0
4	2015-07-05	3833.433025	0

In [114…

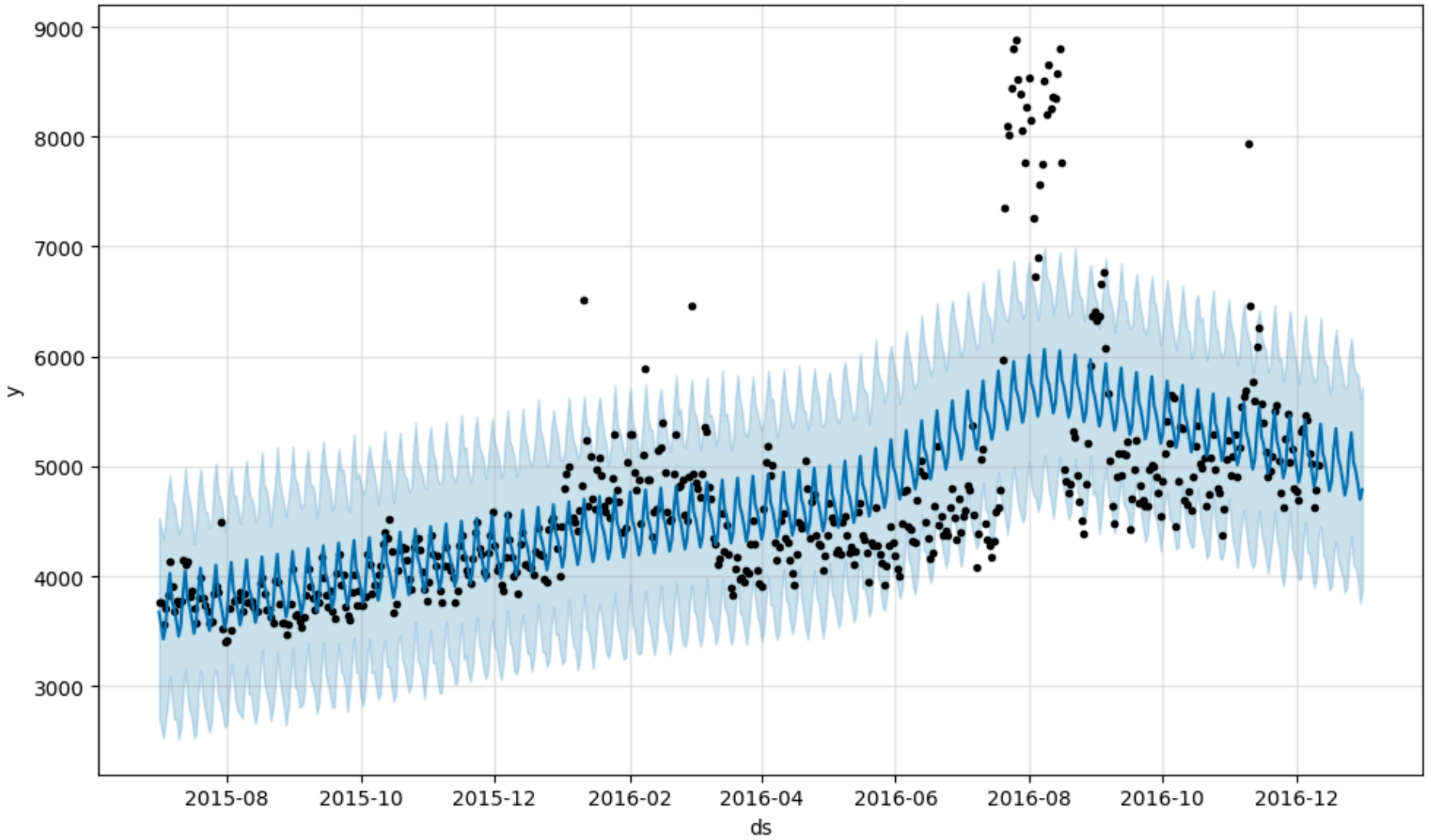
```
df2[:-20].info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 530 entries, 0 to 529
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   ds      530 non-null    object 
 1   y       530 non-null    float64
 2   exog    530 non-null    int64  
dtypes: float64(1), int64(1), object(1)
memory usage: 12.5+ KB
```

In [115…

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

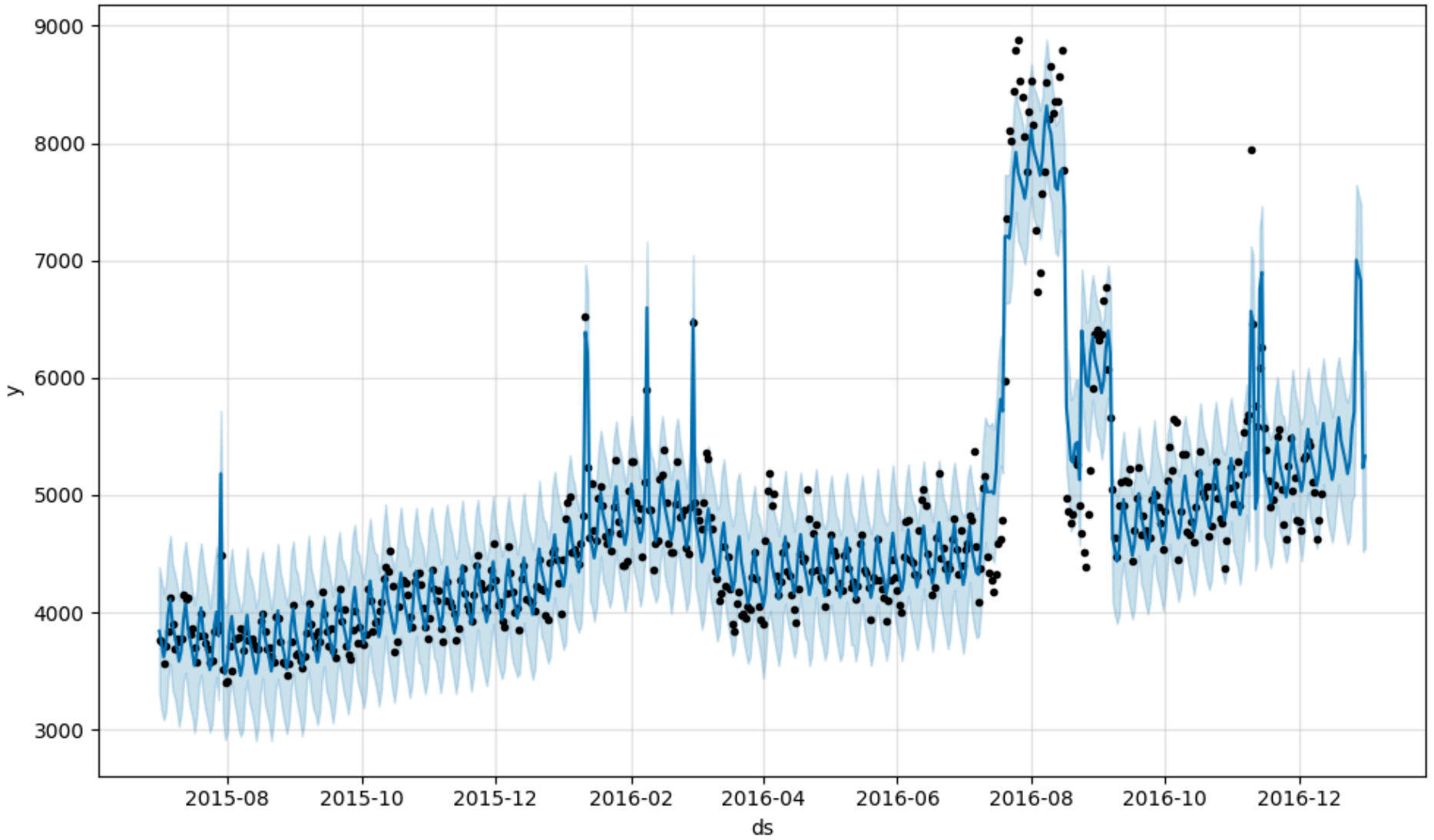
```
INFO:prophet:Disabling yearly seasonality. Run prophet with yearly_seasonality=True to override this.
INFO:prophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to override this.
DEBUG:cmdstanpy:input tempfile: /tmp/tmp6hy4gx9n/7xhv4_lq.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmp6hy4gx9n/ozk72bl8.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_model/prophet_model.bin', 'random', 'seed=9610
3', 'data', 'file=/tmp/tmp6hy4gx9n/7xhv4_lq.json', 'init=/tmp/tmp6hy4gx9n/ozk72bl8.json', 'output', 'file=/tmp/tmp6hy4gx9n/prophet_
model9eqccc2a/prophet_model-20240422203315.csv', 'method=optimize', 'algorithm=lbfgs', 'iter=10000']
20:33:15 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
20:33:15 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
```



In [116…

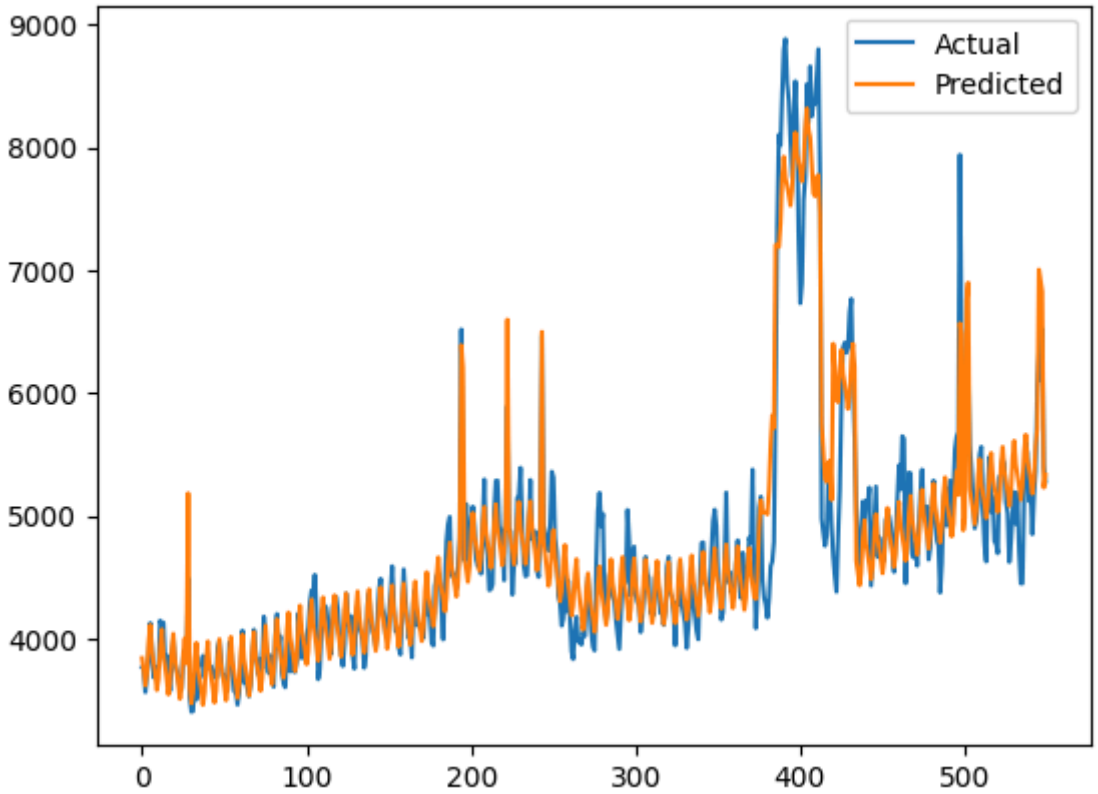
```
model2=Prophet(interval_width=0.9, weekly_seasonality=True, changepoint_prior_scale=1)
model2.add_regressor('exog')
model2.fit(df2[:-20])
forecast2 = model2.predict(df2)
fig = model2.plot(forecast2)
```

```
INFO:prophet:Disabling yearly seasonality. Run prophet with yearly_seasonality=True to override this.
INFO:prophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to override this.
DEBUG:cmdstanpy:input tempfile: /tmp/tmp6hy4gx9n/5w39ifox.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmp6hy4gx9n/jzqm9a7c.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_model/prophet_model.bin', 'random', 'seed=7091
5', 'data', 'file=/tmp/tmp6hy4gx9n/5w39ifox.json', 'init=/tmp/tmp6hy4gx9n/jzqm9a7c.json', 'output', 'file=/tmp/tmp6hy4gx9n/prophet_
modeltuw_qexy/prophet_model-20240422203412.csv', 'method=optimize', 'algorithm=lbfgs', 'iter=10000']
20:34:12 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
20:34:12 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
```



In [117…

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

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



In [119…

```
mape = np.mean(np.abs(forecast2['yhat'][-20:] - df2['y'][-20:].values)/np.abs(df2['y'][-20:].values))
print("mape:",mape*100)
```

```
mape: 6.622248919924485
```

In [ ]:

In [ ]:

For All languages

In [125…

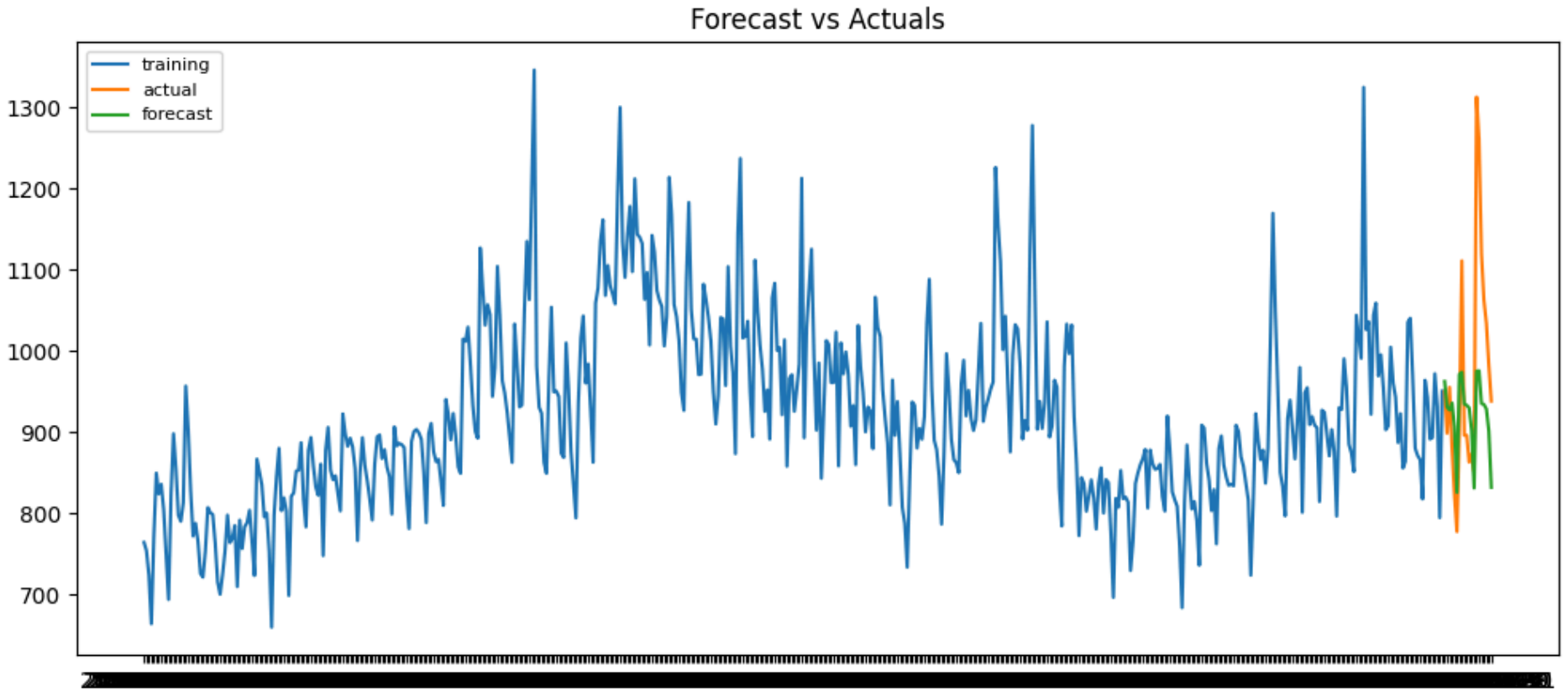
```
def all_arima(train,test):
    order = (0, 0, 2)
    seasonal_order = (1, 1, 1, 7)
```

```
    model=sm.tsa.statespace.SARIMAX(train,order=order,seasonal_order=seasonal_order)
    fitted = model.fit(disp=-1)

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



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

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

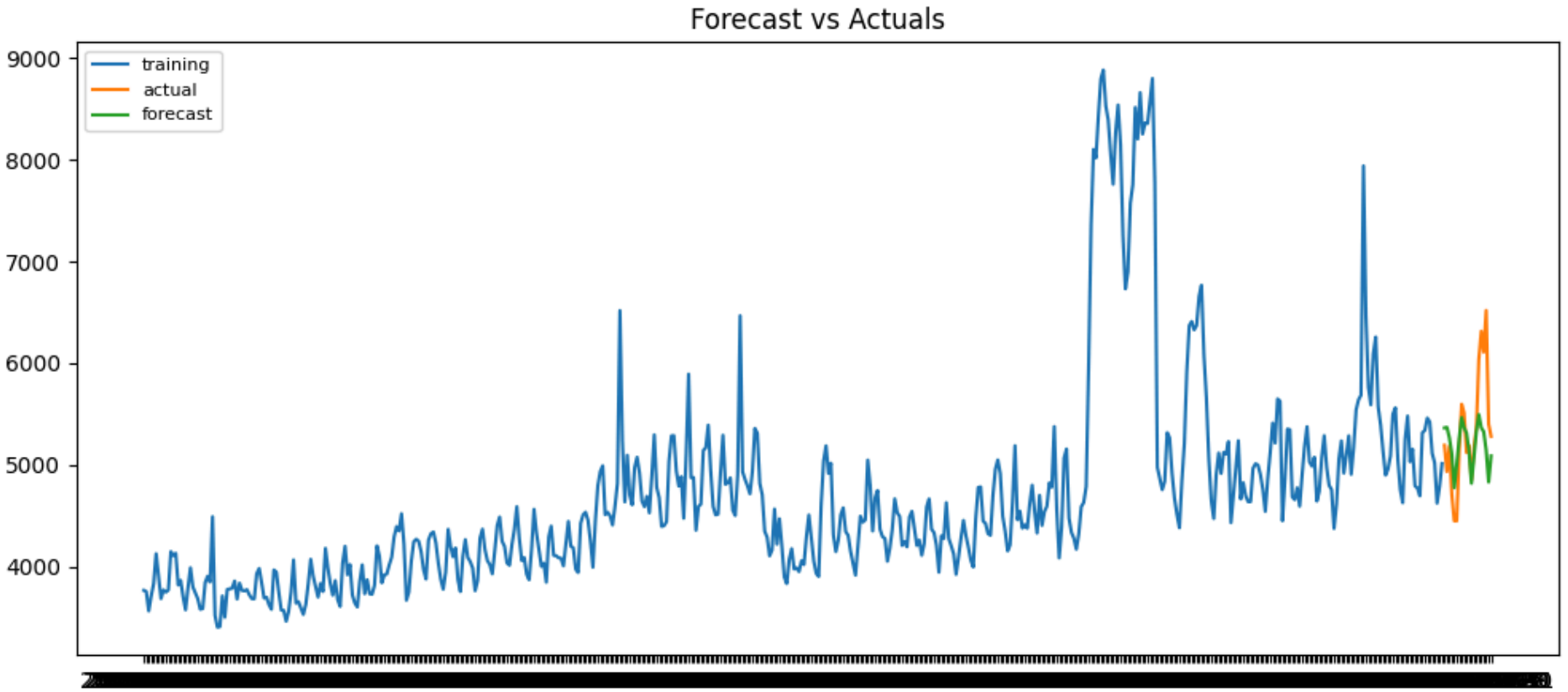
In [126…

```
import warnings
warnings.filterwarnings("ignore")
views_prediction={}
for c in df_final:
    print("language: ",c)
    ts=(df_final[c])
    train = ts[:530]
    test = ts[530:]
    fc=all_arima(train,test)
    views_prediction[c]=fc
```

```
language:  de
```

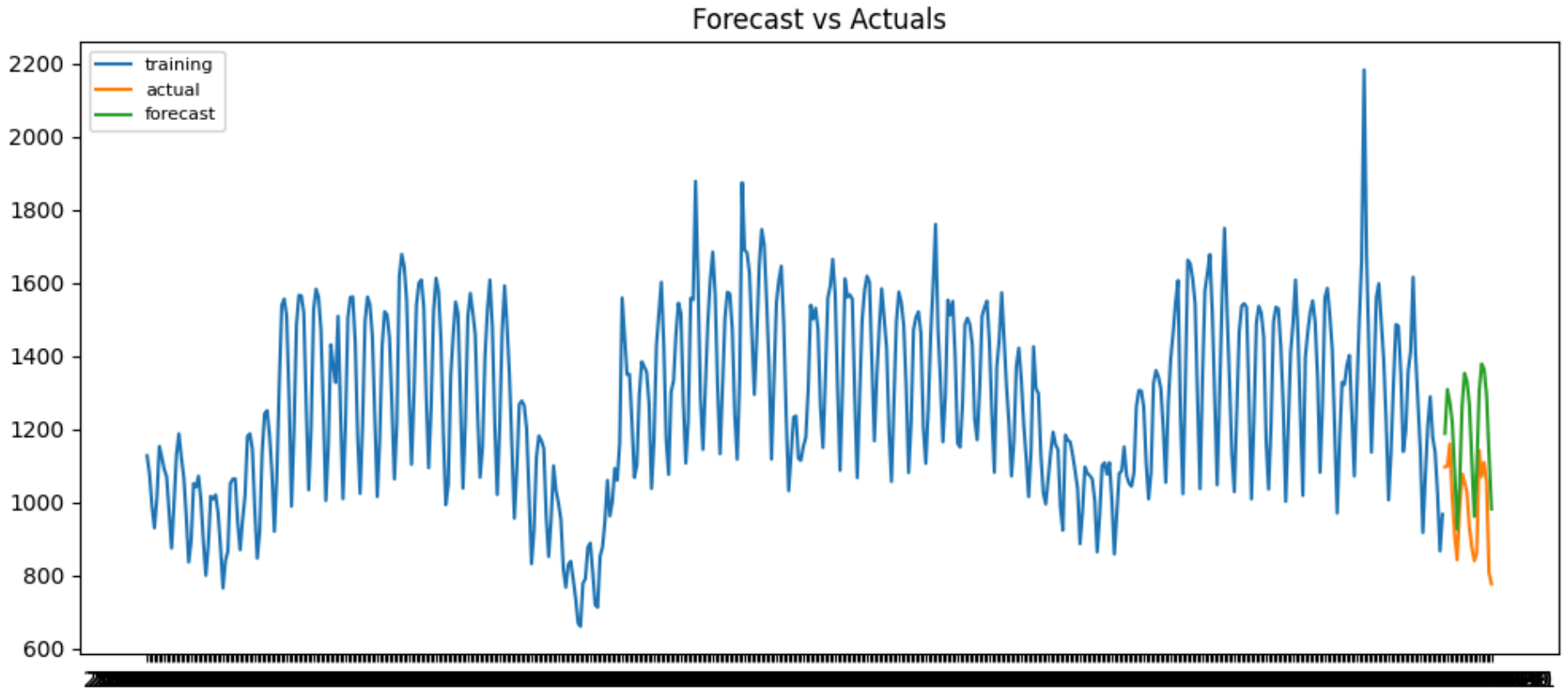



```
mape: 0.08938502866589969
rsme: 126.79126628113846
--------------------------------------------------------------------------------------------------------------
language:  en
```



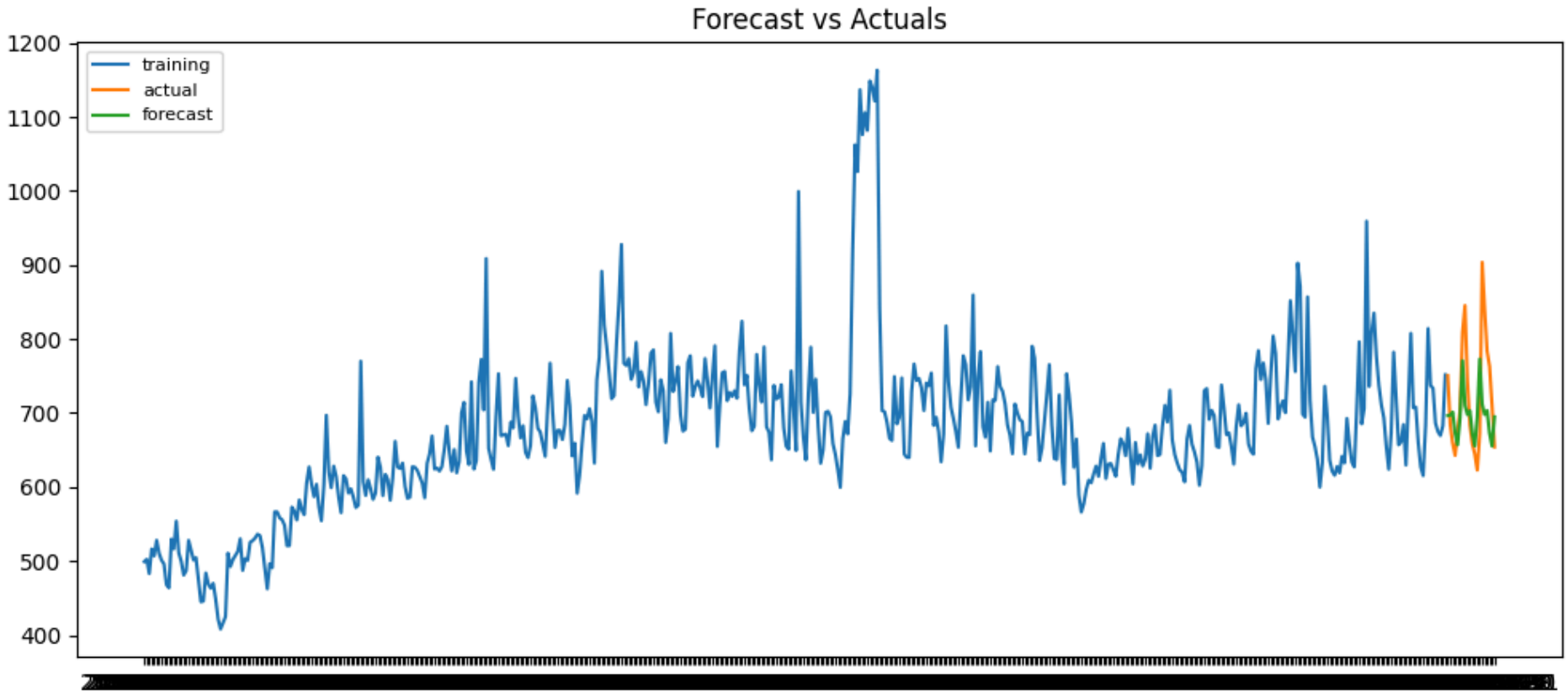

```
mape: 0.06456299170798344
rsme: 497.72723511225075
```

```
language:  es
```




```
mape: 0.2136990062970406
rsme: 222.61710179754917
```

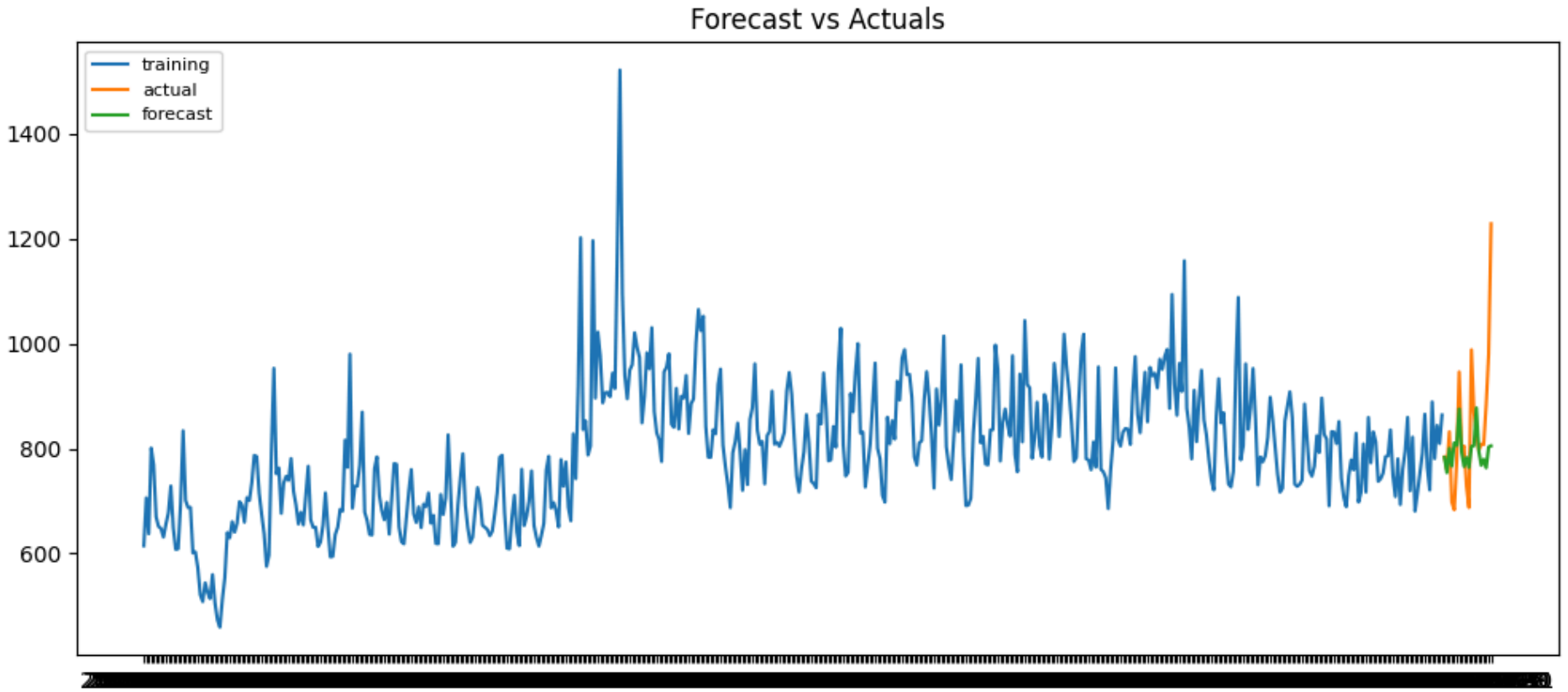
```
language:  fr
```



mape: 0.07937508591347406
rsme: 77.65958501218367

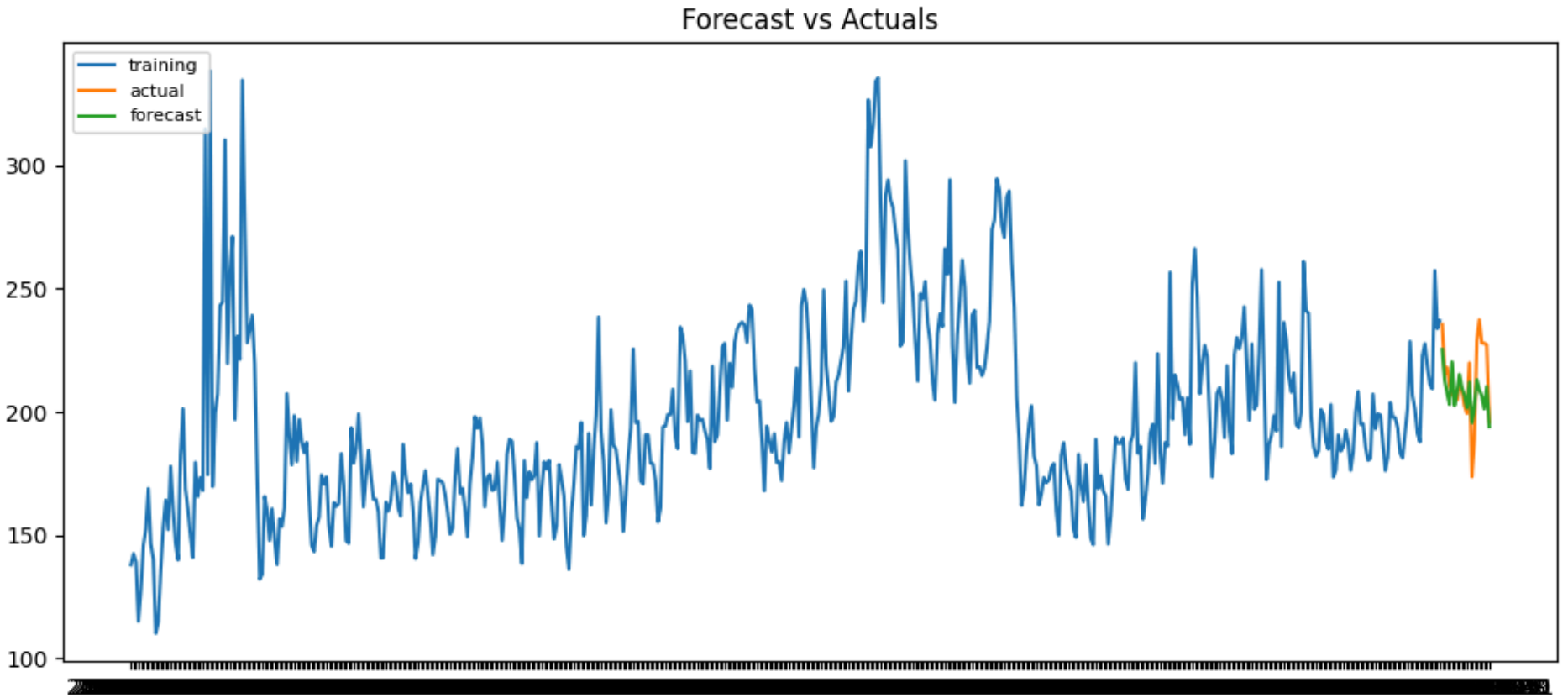
---

language: ja




```
mape: 0.08757977132740317
rsme: 123.72735166919114
```

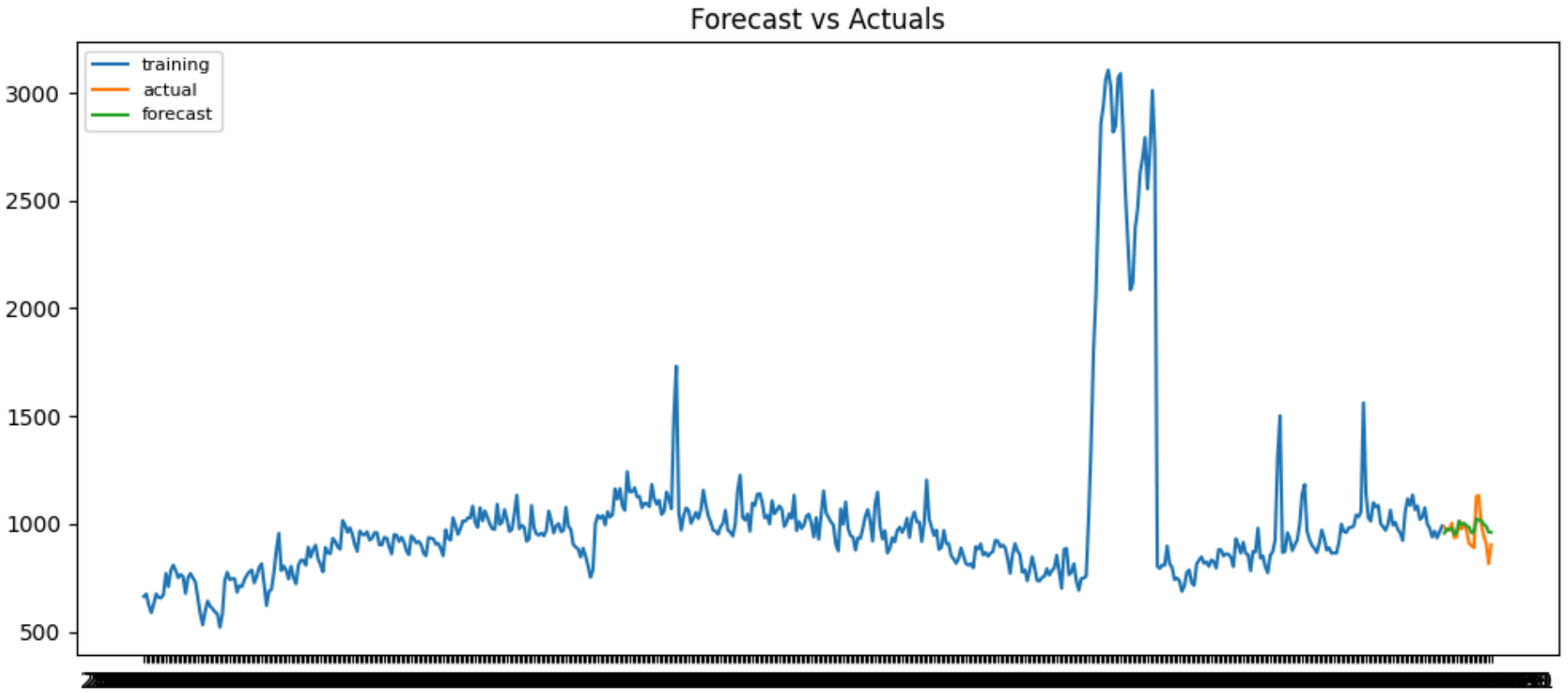
```
language:  ns
```



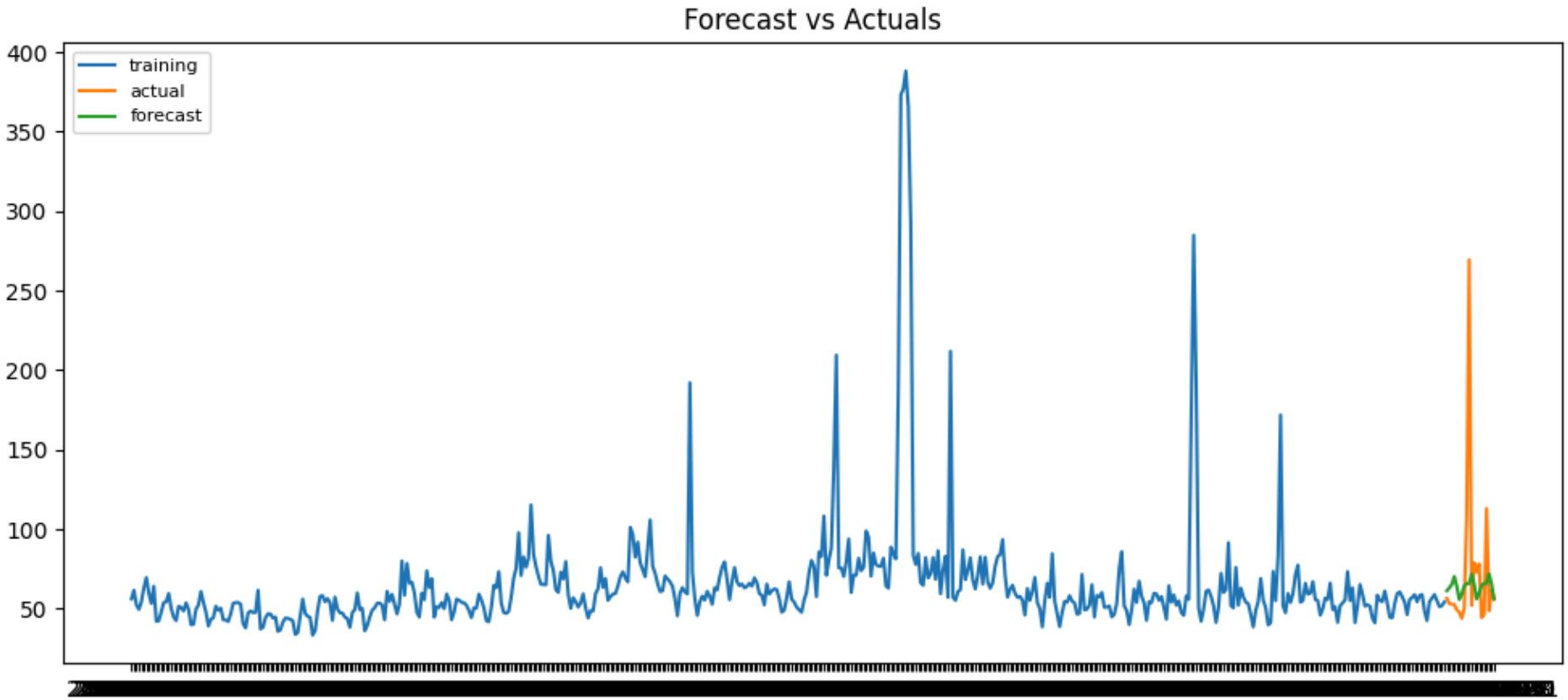
mape: 0.05069596536819856
rsme: 13.713035186540496

---

language: ru



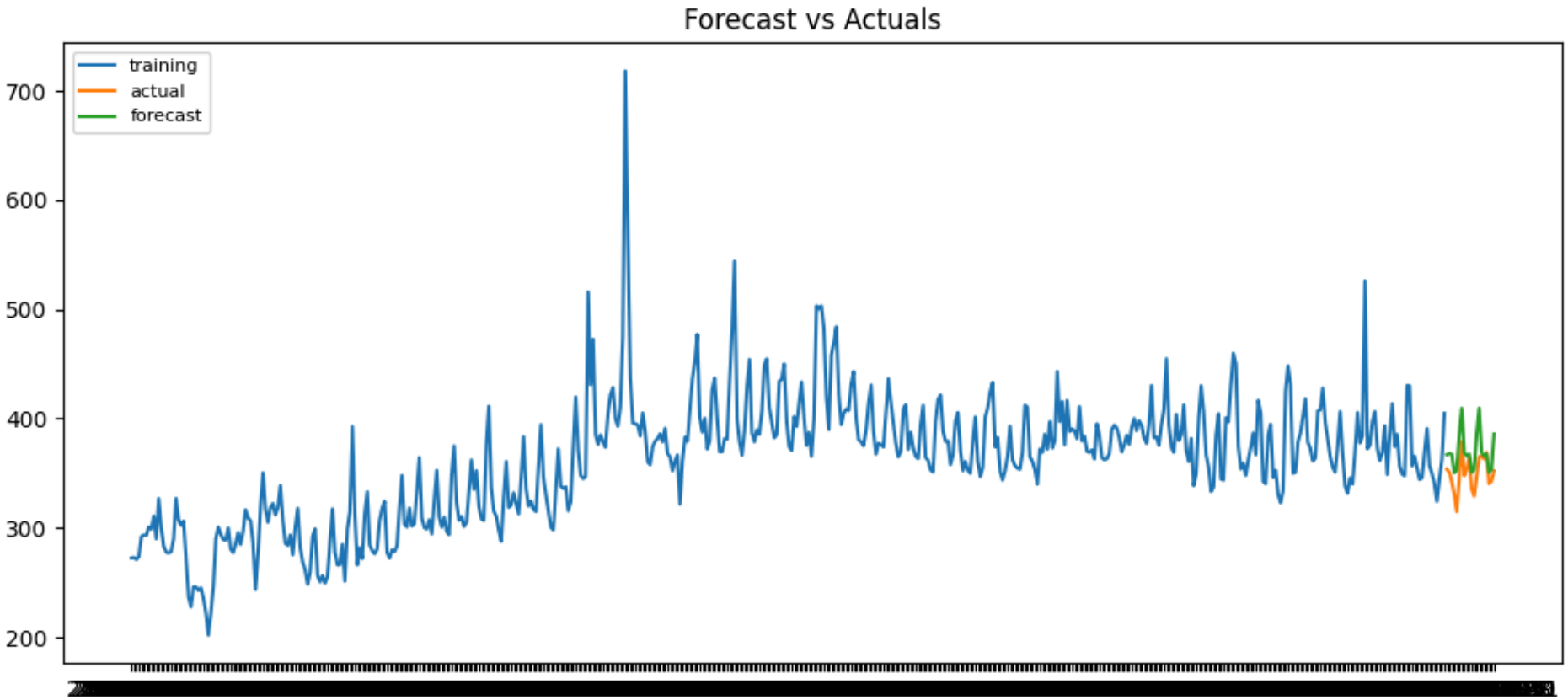

```
mape: 0.048854293625396186
rsme: 61.56141000125506
--------------------------------------------------------------------------------------------------------------
language:  ww
```



mape: 0.2974271887706468
rsme: 49.71392652011668

---

language: zh




```
mape: 0.058636242351694026
rsme: 24.312481622969326
```

In [120…

```
de
en
es
fr
ja
ns
ru
ww
zh
```

In [ ]: