# **DATA ANALYSIS based Website Traffic Analysis**

**PHASE** **3 : DEVELOPMENT PART -1**

**(Loading And Preprocessing** **The Dataset)**

AIM:

Loading and preprocessing a dataset for website traffic analysis is a crucial step in any web analytics project. It involves gathering data from various sources, cleaning and transforming it for analysis.

Here's a step-by-step guide on how to approach this task:

CREATE PYTHON FILE:

A python files contain various libraries and features for machine learning and data analysis. I have created a python file for this project.

IMPORTANT LIBRARIES:

To perform machine learning and website traffic analysis in Python, you can use a combination of various libraries. Here are some commonly used libraries for each of these tasks:

Numpy:

The most common way to include Numpy in your code is by writing “import numpy as np” at the beginning of each file in which you intend to use it. This will allow you to call functions from the library without needing to write out the entire name “numpy” each time.

Sklearn:

neighbors provides functionality for unsupervised and supervised neighbors-based learning methods. Unsupervised nearest neighbors is the foundation of many other learning methods, notably manifold learning and spectral clustering.

Matplotlib.pyplot:

That specific import line merely imports the module "matplotlib. pyplot" and binds that to the name "plt".

Pandas:

Importing pandas as pd: an essential Python library for data scientists. Once you import it, you can take your data analysis to a whole new level. As a general purpose programming language, Python has all the features necessary to analyze and gain insights from data

CODES:

Just some imports for usecase:

import math, time, datetime

import random as rd

Data Manipulation

import numpy as np

import pandas as pd

Visualization

import matplotlib.pyplot as plt

import seaborn as sns

import geopandas as gpd

plt.style.use('seaborn-whitegrid')

Preprocessing

from sklearn.preprocessing import OneHotEncoder, LabelEncoder, label\_binarize

Machine learning:

from sklearn.model\_selection import train\_test\_split

from sklearn.svm import LinearSVC

IMPORTING THE MODEL:

Some of the important model for taking your data.

*# importing models*  
*import* pandas as pd  
import numpy as np  
  
import matplotlib.pyplot as plt  
import geopandas as gpd  
import seaborn as sns  
  
from datetime import timedelta  
from pathlib import Path  
from warnings import simplefilter  
simplefilter('ignore')

MATPLOTLIB DEFAULTS:

*# matplotlib defaults*  
*plt*.style.use('seaborn-whitegrid')  
plt.rc('figure', autolayout = True, figsize = (15,4), titlesize = 10, titleweight ='bold')  
plt.rc('axes', labelweight = 'bold', labelsize = 10, titlesize = 10, titleweight = 'bold', titlepad = 10)

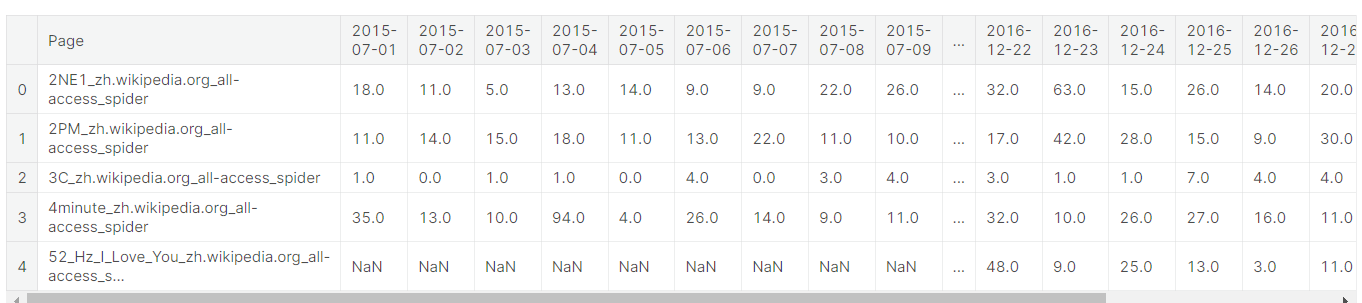
MATPLOTLIB CONFIGURATION FRO HIGER IMAGES**:**

*# matplotlib configuration fro higer images*  
*%*matplotlib inline  
%config InlineBackend.figure\_formats = 'retina'

LOAD THE DATASET:

The head() method returns a specified number of rows, string from the top. The head() method returns the first 5 rows if a number is not specified.

*# load the dataset*  
*data\_dir* = Path(".../input/web-traffic-time-series-forecasting/")  
df = pd.read\_csv(data\_dir / "train\_1.csv.zip")  
df.head()





CHECKING FOR MISSING VALUES:  
 returns the number of missing values in the dataset.

*# checking for missing values*  
*missing\_values* = df.isnull().sum()  
missing\_values

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

missing\_values[missing\_values > 0]

2015-07-01 20740  
2015-07-02 20816  
2015-07-03 20544  
2015-07-04 20654  
2015-07-05 20659  
 ...   
2016-12-27 3701  
2016-12-28 3822  
2016-12-29 3826  
2016-12-30 3635  
2016-12-31 3465  
Length: 550, dtype: int64

total\_missing\_values = missing\_values.sum()  
total\_missing\_values

SHAPE OF THE DATAFRAME:  
*# shape of the dataframe*  
*df*.shape

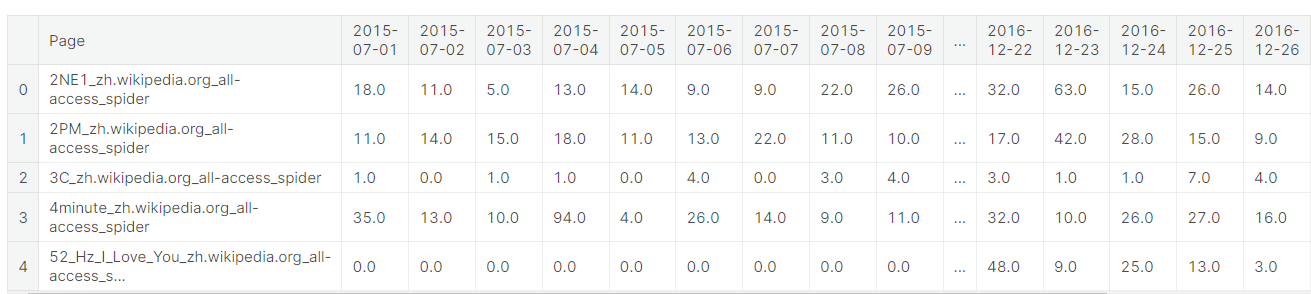
total\_data = np.product(df.shape)  
total\_data

MISSING PERCENT:

*# calculating the total percent of missing values*  
*missing\_percent* = (total\_missing\_values / total\_data) \* 100   
missing\_percent

FILLING NULL COULUMNS AND ROWS:

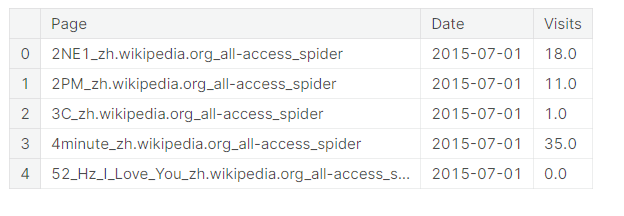
*df\_fill* = df.copy()  
df\_fill = df\_fill.fillna(0)  
df\_fill.head()





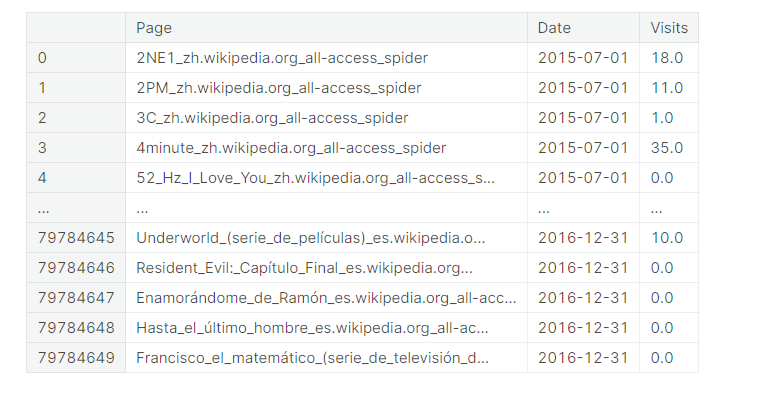
SEPT:

*train\_data* = pd.melt(df\_fill,id\_vars = ['Page'], value\_name = 'Visits', var\_name = 'Date')  
train\_data.head()



train\_data.shape

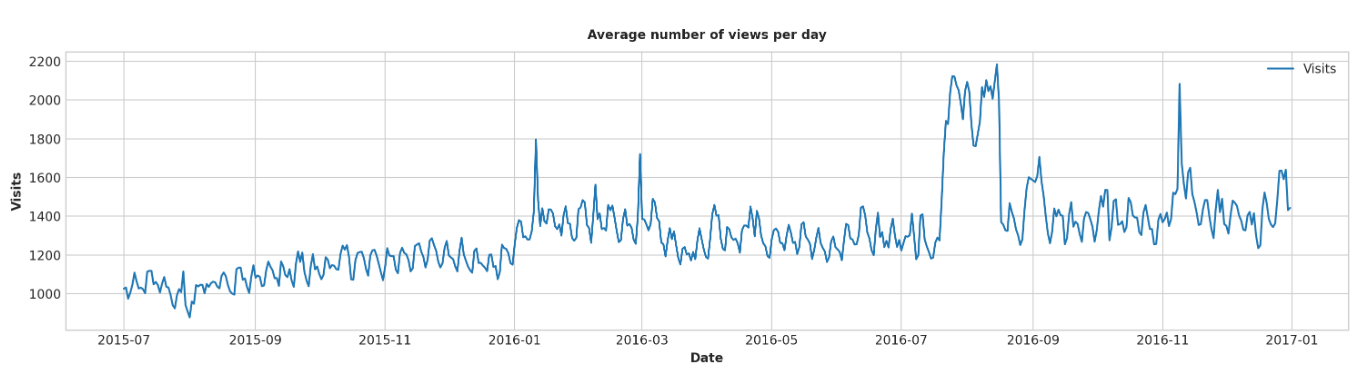
train\_data['Date'] = pd.DatetimeIndex(train\_data['Date'])  
train\_data



VISUALISING :

Visual representation of the given website traffic dataset which represents the Average number of views per day.

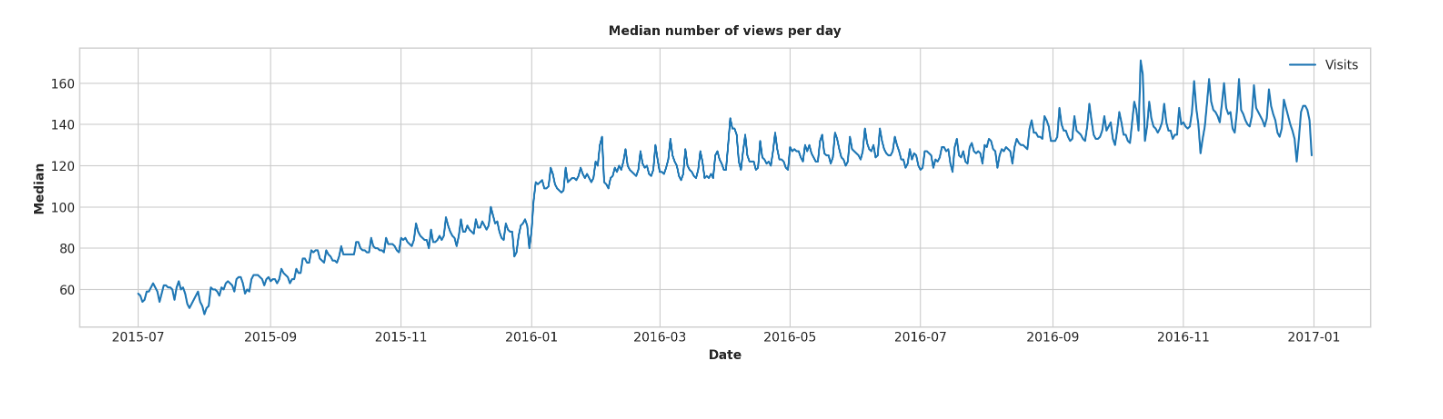
*# Visualising for Mean*  
*temp* = train\_data.groupby('Date')['Visits'].mean()  
plt.xlabel('Date')  
plt.ylabel('Visits')  
plt.title('Average number of views per day')  
plt.plot(temp, label='Visits')  
plt.legend()  
plt.show()



Visual representation of the given website traffic dataset which represents the Median number of views per day.

*# Visualising for Median Number of Data*  
*temp* = train\_data.groupby('Date')['Visits'].median()  
plt.xlabel('Date')  
plt.ylabel('Median')  
plt.title('Median number of views per day')  
plt.plot(temp, label='Visits')  
plt.legend()

plt.show()



CONCLUSION:

Thus, this general process should provide a solid framework for loading and preprocessing your website traffic analysis dataset. It involves visual representation which can be more effective understanding the analysis of website traffics. Help to improve business and in many fields of virtual life.