**WEBSITE TRAFFIC ANALYSIS**

****

**Phase 5 submission document**

Project title: website traffic analysis

Phase 5:

Project documentation &submission

Topic:

In this session we will document the complete project and prepare it for submission.

**INTRODUCTION:-**

* Web traffic analytics refers to collecting data about who comes to your website and what they do when they get there. That data is crucial to building effective sales and marketing strategies. While most people assume more traffic is always better, that's not always true.
* The most popular (and free) tool for website analysis is Google Analytics 4. It's perfect for beginners and has an advanced paid version — Google Analytics 360 — for websites with a lot of traffic. Google Analytics 4 shows data about website traffic and data on user behavior in the form of events and sessions.

**OBJECTIVE:-**

* The goal of Cognos Analytics is to provide qualitative insights that help you to understand your data and its relationships, and to do so automatically for a wide variety of types of data. Cognos Analytics aims at providing results similar to a professional statistician without getting in the way of the business user.

**DATA.COLLECTION:-**

* Data is collected from the Rational solution for Collaborative Lifecycle Management (CLM) applications periodically by data collection jobs.
* collection jobs are commonly known as *extract*, *transform*, and *load* or *ETL* jobs. Data Collection Component uses the ETL process to extract data from various products, for example CLM products, transform it and ultimately store the data into the data warehouse for reporting purposes. When you author reports and you want the CLM applicable metrics, Data Collection Component fills in all the metrics tables from the data warehouse.
* Data modules are containers that describe data and rules for combining and shaping data to prepare it for analysis and visualization in IBM Cognos Analytics with Watson.

**DATA.VISUALIZATION:-**

IBM Cognos Analytics provides a number of recommended visualizations based on the data that you are working with. Use color in your table or crosstab visualizations to see the distribution of your data and highlight exceptional data points.

**DATA.EXPLORATION**:-

Explore is a flexible workspace where you can discover and analyze data. You can also explore an existing visualization from a dashboard or story. Uncover hidden relationships and identify patterns that turn your data into insights.

**GIVEN.DATA.LINK:-**

[**https://www.kaggle.com/datasets/bobnau/daily-website-visitors**](https://www.kaggle.com/datasets/bobnau/daily-website-visitors)

**DATASET:-**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Row** | **Day** | **Day.Of.Week** | **Date** | **Page.Loads** | **Unique.Visits** | **First.Time.Visits** | **Returning.Visits** |
| **1** | **Sunday** | **1** | **9/14/2014** | **2,146** | **1,582** | **1,430** | **152** |
| **2** | **Monday** | **2** | **9/15/2014** | **3,621** | **2,528** | **2,297** | **231** |
| **3** | **Tuesday** | **3** | **9/16/2014** | **3,698** | **2,630** | **2,352** | **278** |
| **4** | **Wednesday** | **4** | **9/17/2014** | **3,667** | **2,614** | **2,327** | **287** |
| **5** | **Thursday** | **5** | **9/18/2014** | **3,316** | **2,366** | **2,130** | **236** |
| **6** | **Friday** | **6** | **9/19/2014** | **2,815** | **1,863** | **1,622** | **241** |
| **7** | **Saturday** | **7** | **9/20/2014** | **1,658** | **1,118** | **985** | **133** |
| **8** | **Sunday** | **1** | **9/21/2014** | **2,288** | **1,656** | **1,481** | **175** |
| **9** | **Monday** | **2** | **9/22/2014** | **3,638** | **2,586** | **2,312** | **274** |
| **10** | **Tuesday** | **3** | **9/23/2014** | **4,462** | **3,257** | **2,989** | **268** |
| **11** | **Wednesday** | **4** | **9/24/2014** | **4,414** | **3,175** | **2,891** | **284** |
| **12** | **Thursday** | **5** | **9/25/2014** | **4,315** | **3,029** | **2,743** | **286** |
| **13** | **Friday** | **6** | **9/26/2014** | **3,323** | **2,249** | **2,033** | **216** |
| **14** | **Saturday** | **7** | **9/27/2014** | **1,656** | **1,180** | **1,040** | **140** |
| **15** | **Sunday** | **1** | **9/28/2014** | **2,465** | **1,806** | **1,613** | **193** |

**MANDOTARY.STEPS:-**

Import the libraries:

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

**PYTHON.PROGRAMS:-**

import re

log\_file\_path = "access.log"

def parse\_log(log\_file):

with open(log\_file, "r") as file:

log\_data = file.readlines()

log\_entries = []

for line in log\_data:

match = re.match(r'(\S+) (\S+) (\S+) \[([\w:/]+\s[+\-]\d{4})\] "(\S+) (\S+) (\S+)" (\d{3}) (\d+)', line)

if match:

entry = {

"ip": match.group(1),

"user": match.group(2),

"timestamp": match.group(4),

"method": match.group(5),

"url": match.group(6),

"status\_code": match.group(8),

"bytes\_sent": match.group(9)

}

log\_entries.append(entry)

return log\_entries

log\_entries = parse\_log(log\_file\_path)

def analyze\_traffic(log\_entries):

page\_views = len(log\_entries)

unique\_ips = len(set(entry["ip"] for entry in log\_entries))

return page\_views, unique\_ips

page\_views, unique\_ips = analyze\_traffic(log\_entries)

print(f"Page Views: {page\_views}")

print(f"Unique Visitors: {unique\_ips}")

IN:

import math

from scipy.stats import norm

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from IPython.core.display import HTML

import os

for dirname, \_, filenames in os.walk('/kaggle/input'):

for filename in filenames:

print(os.path.join(dirname, filename))

df = pd.read\_csv("/kaggle/input/daily-website-visitors/daily-website-visitors.csv", \

index\_col = 'Date', thousands = ',', parse\_dates=True)

df.head()

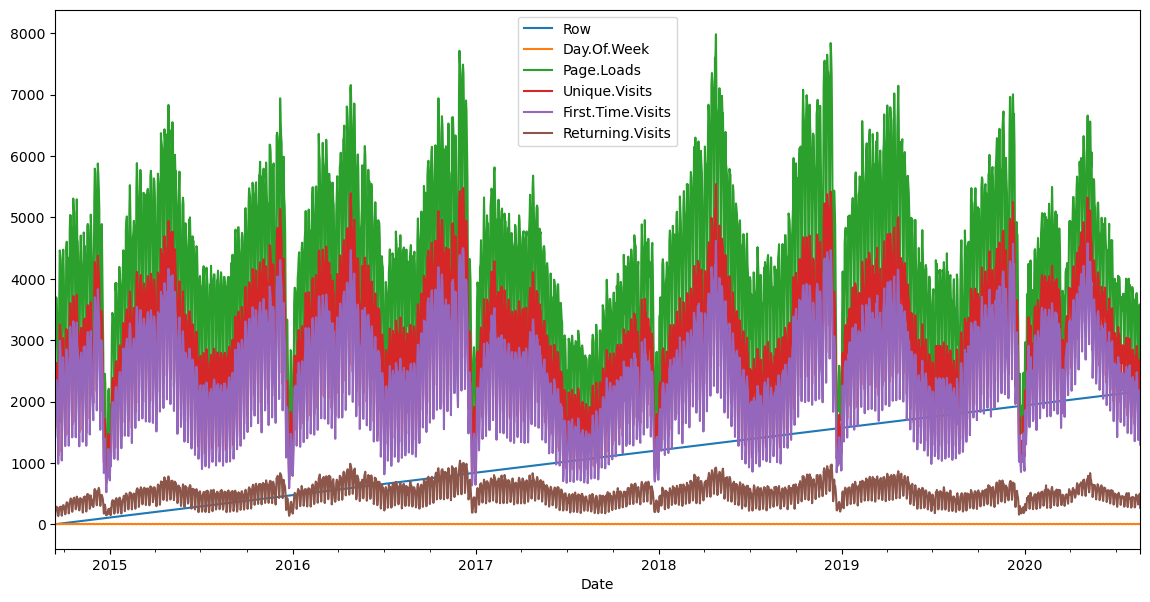
OUT :

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Row | Day | Day.Of.Week | Date | Page.Loads | Unique.Visits | First.Time.Visits |
| 1 | Sunday | 1 | 9/14/2014 | 2,146 | 1,582 | 1,430 |
| 2 | Monday | 2 | 9/15/2014 | 3,621 | 2,528 | 2,297 |
| 3 | Tuesday | 3 | 9/16/2014 | 3,698 | 2,630 | 2,352 |
| 4 | Wednesday | 4 | 9/17/2014 | 3,667 | 2,614 | 2,327 |

IN:

df.plot(figsize=(14,7))

OUT:



IN:

def prob(t, n, lmbda):

return math.pow(lmbda \* t, n)/math.factorial(n)\*math.exp(-lmbda\*t)

mean = df['Page.Loads'].mean()

print( "mean loads per day:", mean)

std = df['Page.Loads'].std()

print( "std deviation of loads per day:", std)

n = 1

px = np.linspace(1, 8000, 50)

py = np.zeros(50)

for i in range(0, 50):

x = (px[i]-mean)/std

p = norm.pdf(x)

py[i] = 1000\*p

OUT:

mean loads per day: 4116.9893862482695

std deviation of loads per day: 1350.9778426999621

IN:

fig, ax1 = plt.subplots()

df['Page.Loads'].plot.hist(ax = ax1, label='Page.Loads')

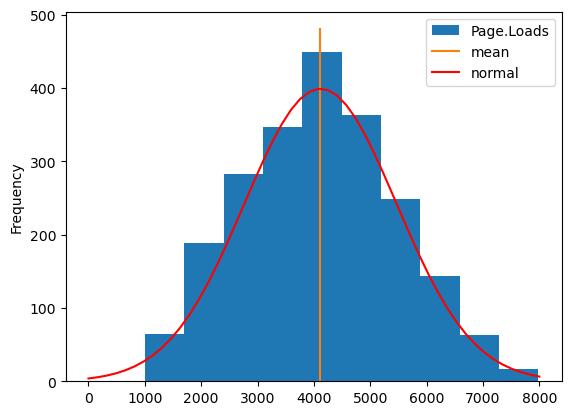
plt.plot([mean, mean], [0, 480], label='mean')

plt.plot(px, py, label='normal', color='red')

plt.legend()

plt.show()

OUT:



IN:

fig, ax1 = plt.subplots()

df['Page.Loads'].plot(ax = ax1, label='Page.Loads')

plt.plot([df.index[0], df.index[-1]], [mean, mean], color='red')

upper = mean + 1.96\*std

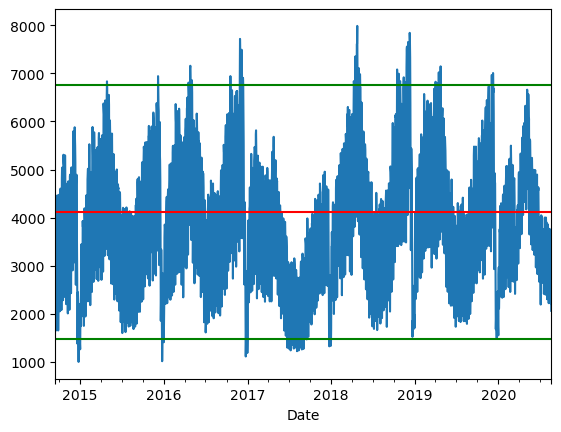
lower = mean - 1.96\*std

plt.plot([df.index[0], df.index[-1]], [upper, upper], color='green')

plt.plot([df.index[0], df.index[-1]], [lower, lower], color='green')

plt.show()

OUT:



IN:

import pandas as pd

FILE\_LOCATION = ('/kaggle/input/daily-website-visitors/daily-website-visitors.csv')

whole\_dataset=pd.read\_csv(FILE\_LOCATION,index\_col='Date',thousands=',')

whole\_dataset.index = pd.to\_datetime(whole\_dataset.index)

whole\_dataset

OUT:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Row | Day | Day.Of.week | Date | Page.Loads | Unique.Visits | First.Time.Visits | Returning.Visits |
| 1 | Sunday | 1 | 9/14/2014 | 2,146 | 1,582 | 1,430 | 152 |
| 2 | Monday | 2 | 9/15/2014 | 3,621 | 2,528 | 2,297 | 231 |
| 3 | Tuesday | 3 | 9/16/2014 | 3,698 | 2,630 | 2,352 | 278 |
| 4 | Wednesday | 4 | 9/17/2014 | 3,667 | 2,614 | 2,327 | 287 |
| 5 | Thursday | 5 | 9/18/2014 | 3,316 | 2,366 | 2,130 | 236 |
| 6 | Friday | 6 | 9/19/2014 | 2,815 | 1,863 | 1,622 | 241 |
| 7 | Saturday | 7 | 9/20/2014 | 1,658 | 1,118 | 985 | 133 |
| 8 | Sunday | 1 | 9/21/2014 | 2,288 | 1,656 | 1,481 | 175 |
| 9 | Monday | 2 | 9/22/2014 | 3,638 | 2,586 | 2,312 | 274 |

IN:

import matplotlib.pyplot as plt

fig, axs = plt.subplots(3, figsize=(12, 5))

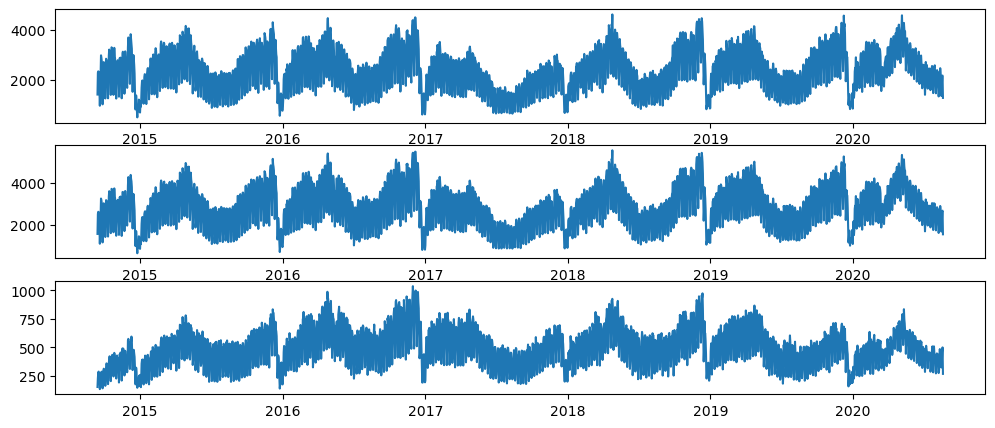
axs[0].plot(whole\_dataset['First.Time.Visits'])

axs[1].plot(whole\_dataset['Unique.Visits'])

axs[2].plot(whole\_dataset['Returning.Visits'])

plt.show()

**OUT:**

****  
IN:

target\_column = whole\_dataset['Returning.Visits']

target\_column

**OUT:**

Date

2014-09-14 152

2014-09-15 231

2014-09-16 278

2014-09-17 287

2014-09-18 236

...

2020-08-15 323

2020-08-16 351

2020-08-17 457

2020-08-18 499

2020-08-19 267

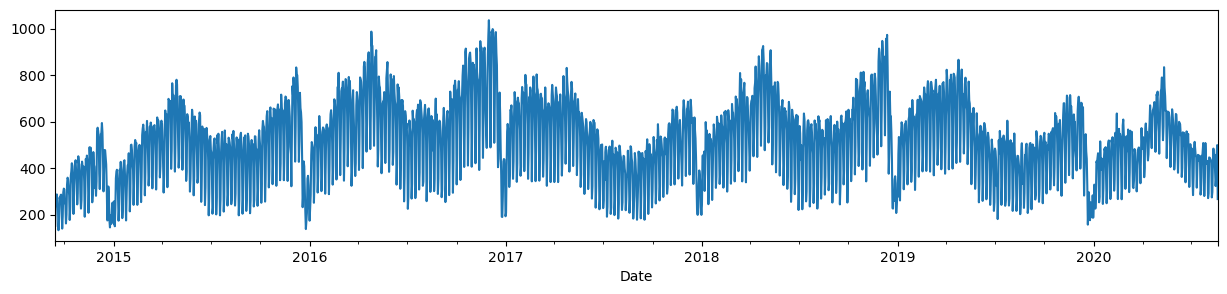
Name: Returning.Visits, Length: 2167, dtype: int64

In:

target\_column.plot(figsize=(15, 3))

plt.show()

out:



In:

TEST\_DATA\_PERCENTAGE = 0.1

TEST\_DATA\_BOUNDARY\_INDEX = int((1 - TEST\_DATA\_PERCENTAGE) \* len(target\_column))

print(f"Train data:\tReturning Visits [:{TEST\_DATA\_BOUNDARY\_INDEX}] ({TEST\_DATA\_BOUNDARY\_INDEX + 1})")

print(f"Test data:\tReturning Visits [{TEST\_DATA\_BOUNDARY\_INDEX}:] ({len(target\_column) - TEST\_DATA\_BOUNDARY\_INDEX})")

print(f"\nLast target on train data: {target\_column[TEST\_DATA\_BOUNDARY\_INDEX]}")

out:

Train data: Returning Visits [:1950] (1951)

Test data: Returning Visits [1950:] (217)

In:

target\_column[TEST\_DATA\_BOUNDARY\_INDEX10:TEST\_DATA\_BOUNDARY\_INDEX+10].

values, (list(train\_dataset)[-1][0][-1].numpy(), list(train\_dataset)[-1][1][-1].numpy())

out:

(array([429, 423, 442, 464, 372, 253, 277, 515, 434, 394, 441, 413, 246, 314, 443, 484, 473, 490, 353, 249]),

(array([277, 515, 434]), 394))

Plot the train and test datasets

**In:**

import numpy as np

import matplotlib.dates as mdates

def plot\_time\_series(predictions = None, start\_index=1500):

timesteps = pd.to\_datetime(target\_column.index)

fig,ax = plt.subplots(1,figsize=(15,5)) ax.xaxis.set\_major\_locator(mdates.MonthLocator(bymonth=(1, 7)))

ax.xaxis.set\_minor\_locator(mdates.MonthLocator()) ax.xaxis.set\_major\_formatter(mdates.DateFormatter('%Y-%b'))

*# Plot train dataset* plt.plot(timesteps[start\_index:TEST\_DATA\_BOUNDARY\_INDEX], target\_column[start\_index:TEST\_DATA\_BOUNDARY\_INDEX],

color='blue')

*# Plot test dataset*

plt.plot(timesteps[TEST\_DATA\_BOUNDARY\_INDEX:], target\_column[TEST\_DATA\_BOUNDARY\_INDEX:],

color='green', linewidth=0.4)

if predictions is not None:

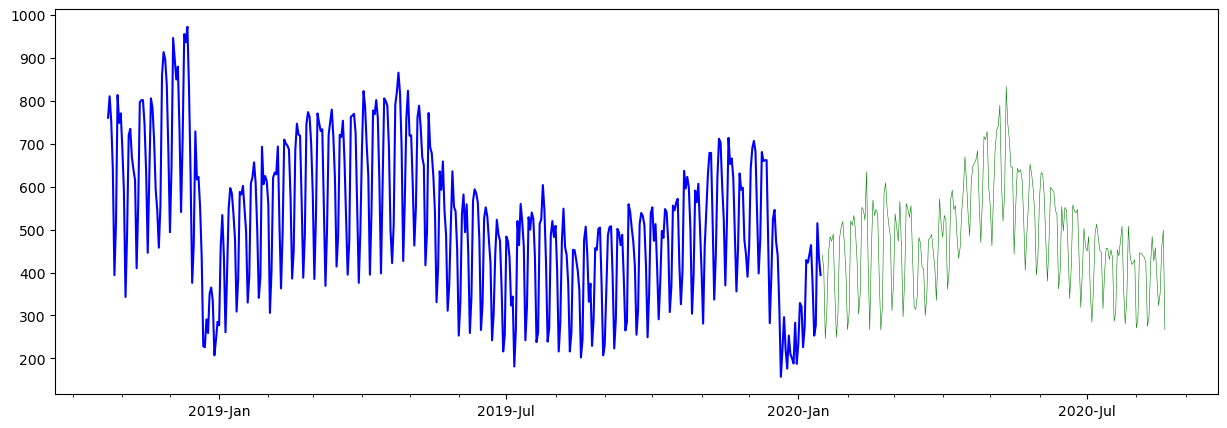
pred\_timesteps = timesteps[TEST\_DATA\_BOUNDARY\_INDEX:]

plt.plot(pred\_timesteps, predictions, linewidth=0.4, color='red')

plt.scatter(pred\_timesteps, predictions, s=0.4, color='red')

plot\_time\_series()

out:



**In:**

import tensorflow as tf

from tensorflow.keras.layers import Layer

from tensorflow.keras import Model

class **NaiveForecastLayer**(Model):

def \_\_init\_\_(self):

super().\_\_init\_\_()

def call(self, inputs):

result = inputs[:, -1]

return result[:, tf.newaxis]

baseline\_model = NaiveForecastLayer()

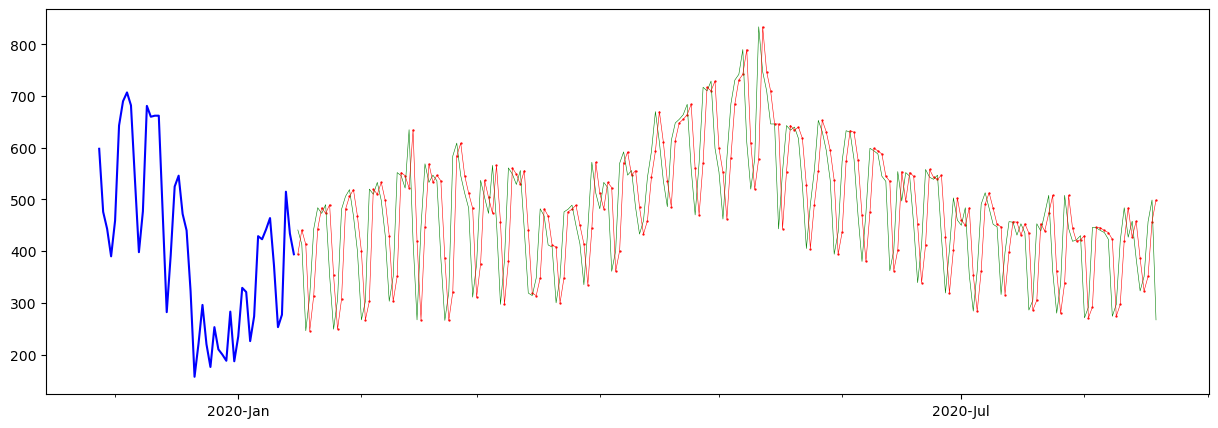
baseline\_model.\_name = 'model\_0'

baseline\_model.compile(metrics=[tf.keras.metrics.MeanAbsoluteError()])

baseline\_predictions = baseline\_model.predict(test\_dataset)

plot\_time\_series(baseline\_predictions.ravel(), start\_index=1900)

out:



In:

y\_true = target\_column[TEST\_DATA\_BOUNDARY\_INDEX : ]

len(y\_true), y\_true

out:

(217,

Date

2020-01-16 441

2020-01-17 413

2020-01-18 246

2020-01-19 314

2020-01-20 443

...

2020-08-15 323

2020-08-16 351

2020-08-17 457

2020-08-18 499

2020-08-19 267

Name: Returning.Visits, Length: 217, dtype: int64)

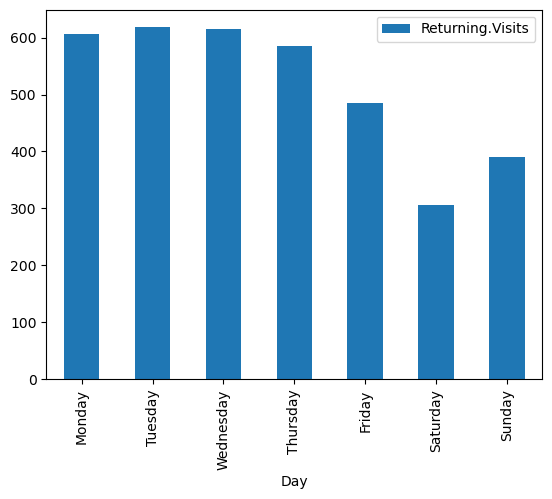
**In:**

DAYS\_OF\_WEEK = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']

pd.DataFrame(dataset\_by\_day['Returning.Visits'].mean()).loc[DAYS\_OF\_WEEK].plot(kind='bar')

**out:**

<Axes: xlabel='Day'>

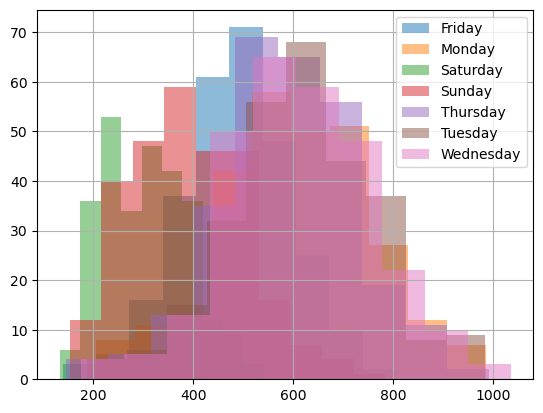


**In:**

dataset\_by\_day['Returning.Visits'].hist(legend=True, alpha=0.5)

plt.show()

**out:**

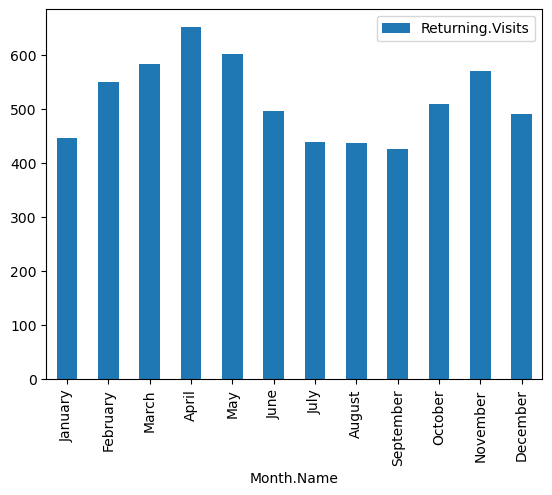


**In:**

pd.DataFrame(dataset\_group\_by\_month['Returning.Visits'].mean()).loc[MONTH\_NAMES].plot(kind='bar')

plt.show()

**out:**



**In:**

from tensorflow.data import Dataset

model3\_history = model\_3.fit(x=[dataset2\_rv\_history\_features, X\_cat\_encoded], y=train\_dataset2, epochs=5)

pd.DataFrame(model3\_history.history).plot()

out:

Epoch 1/5

61/61 [==============================] - 3s 7ms/step - loss: 232.3113

Epoch 2/5

61/61 [==============================] - 0s 7ms/step - loss: 105.8665

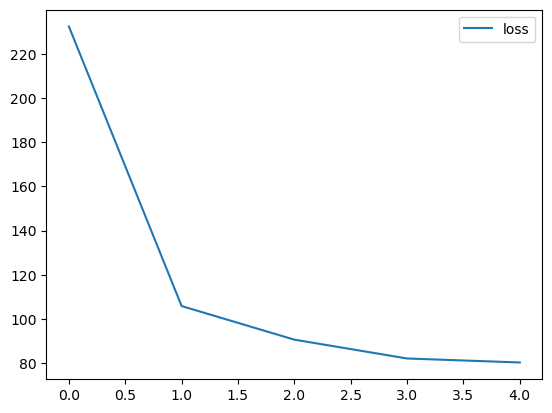
Epoch 3/5

61/61 [==============================] - 0s 7ms/step - loss: 90.6746

Epoch 4/5

61/61 [==============================] - 0s 7ms/step - loss: 82.1568

Epoch 5/561/61 [==============================]- 0s 7ms/step - loss: 80.3541



<Axes: >

In:

y\_dataset = test\_dataset2['Returning.Visits']

y\_dataset

out:

Date

2020-01-16 441

2020-01-17 413

2020-01-18 246

2020-01-19 314

2020-01-20 443

...

2020-08-15 323

2020-08-16 351

2020-08-17 457

2020-08-18 499

2020-08-19 267

Name: Returning.Visits, Length: 217, dtype: int64

In:

def evaluate\_model\_predictions(y\_true, predictions, model\_name):

metrics = evaluate\_predictions(y\_true, predictions)

MODEL\_METRICS.loc[model\_name] = metrics

plot\_time\_series(predictions.ravel(), start\_index=1900)

return metrics

evaluate\_model\_predictions(y\_dataset, model\_3\_preds, 'model\_3 (multi-input)')

out:

{'mae': 72.46600053497174,

'mse': 8797.218902262757,

'rmse': 93.79349072437147,

'mape': 0.15300125677432075}

In:

import numpy as np

import pandas as pd

import pandas\_profiling

import warnings

warnings.filterwarnings('ignore')

import datetime

from datetime import date

import seaborn as sns

import matplotlib.pyplot as plt

%matplotlib inline

sns.set\_style("whitegrid")

# import chart\_studio.plotly as py

import cufflinks as cf

import plotly.express as px

from plotly.offline import download\_plotlyjs, init\_notebook\_mode, plot, iplot

init\_notebook\_mode(connected=True)

cf.go\_offline()

import pandas\_profiling

import plotly.graph\_objects as go

from sklearn.model\_selection import train\_test\_split, cross\_val\_score, GridSearchCV

from sklearn.metrics import accuracy\_score

from sklearn.svm import SVR

from sklearn.linear\_model import LinearRegression

from sklearn.tree import DecisionTreeRegressor

import xgboost as xg

df=pd.read\_csv('../input/daily-website-visitors/daily-website-visitors.csv')

df.rename(columns = {'Day.Of.Week':'day\_of\_week'

,'Page.Loads':'page\_loads'

,'Unique.Visits':'unique\_visits'

,'First.Time.Visits':'first\_visits'

,'Returning.Visits':'returning\_visits'}, inplace = True)

df=df.replace(',','',regex=True)

df['page\_loads']=df['page\_loads'].astype(int)

df['unique\_visits']=df['unique\_visits'].astype(int)

df['first\_visits']=df['first\_visits'].astype(int)

df['returning\_visits']=df['returning\_visits'].astype(int)

df

out:

Row Day day\_of\_week Date page\_loads unique\_visit first\_visits

0 1 Sunday 1 9/14/2014 2146 1582 1430 152

1 2 Monday 2 9/15/2014 3621 2528 2297 231

2 3 Tuesday 3 9/16/2014 3698 2630 2352 278

3 4 Wednesday 4 9/17/2014 3667 2614 2327 287

4 5 Thursday 5 9/18/2014 3316 2366 2130 236

... ... ... ... ... ... ... ... ...

2162 2163 Saturday 7 8/15/2020 2221 1696 1373 323

2163 2164 Sunday 1 8/16/2020 2724 2037 1686 351

In:

df.isna().sum()

out:

Row 0

Day 0

day\_of\_week 0

Date 0

page\_loads 0

unique\_visits 0

first\_visits 0

returning\_visits 0

dtype: int64

In:

df.duplicated().sum()

out:

0

In:

df.info()

out:

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 2167 entries, 0 to 2166

Data columns (total 8 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Row 2167 non-null int64

1 Day 2167 non-null object

2 day\_of\_week 2167 non-null int64

3 Date 2167 non-null object

4 page\_loads 2167 non-null int64

5 unique\_visits 2167 non-null int64

6 first\_visits 2167 non-null int64

7 returning\_visits 2167 non-null int64

dtypes: int64(6), object(2)

memory usage: 135.6+ KB

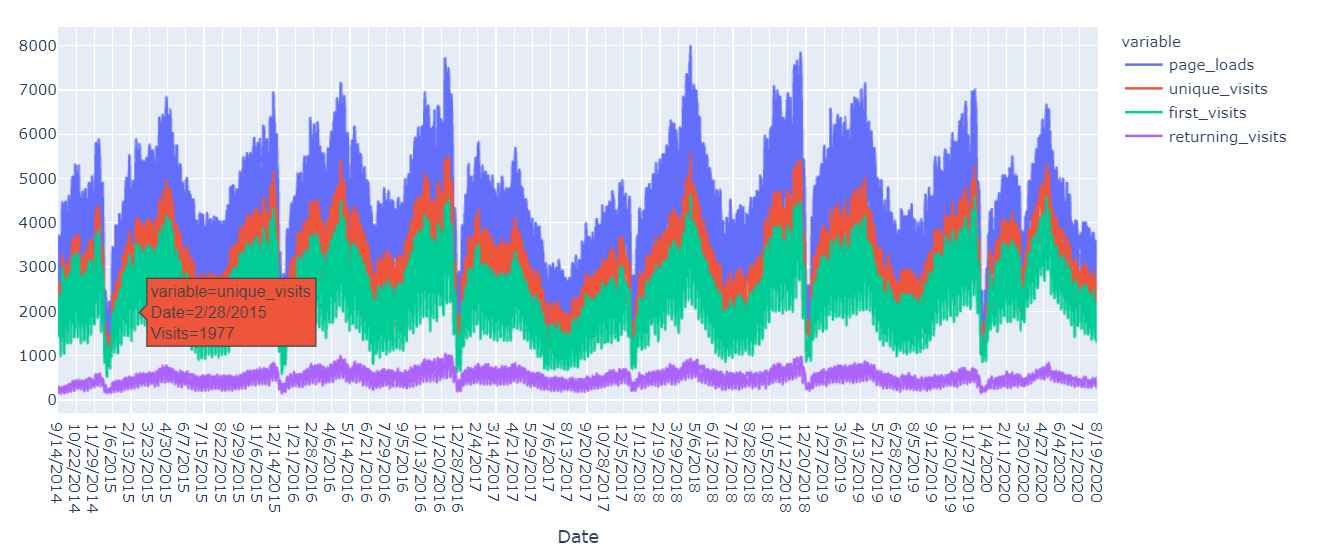
In:

px.line(df,x='Date',y=['page\_loads' ,'unique\_visits' ,'first\_visits' ,'returning\_visits'],

labels={'value':'Visits'}

,title='Page Loads & visitors over Time')

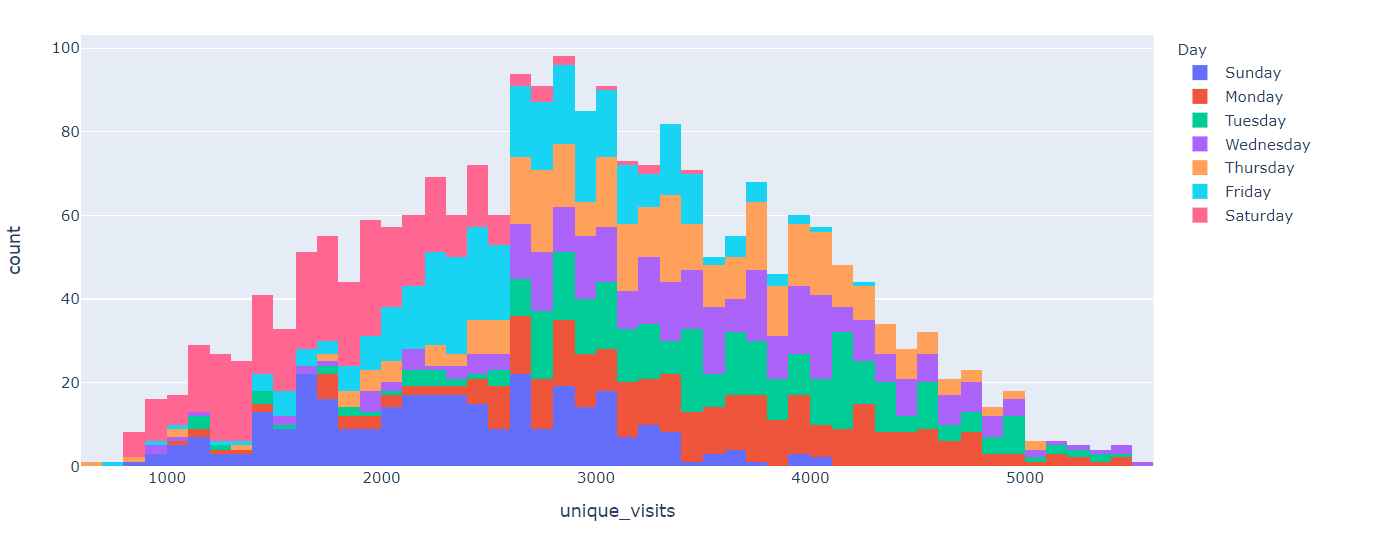
Out:



In:

px.histogram(df,x='unique\_visits',color='Day',title='unique visits for each day')

out:

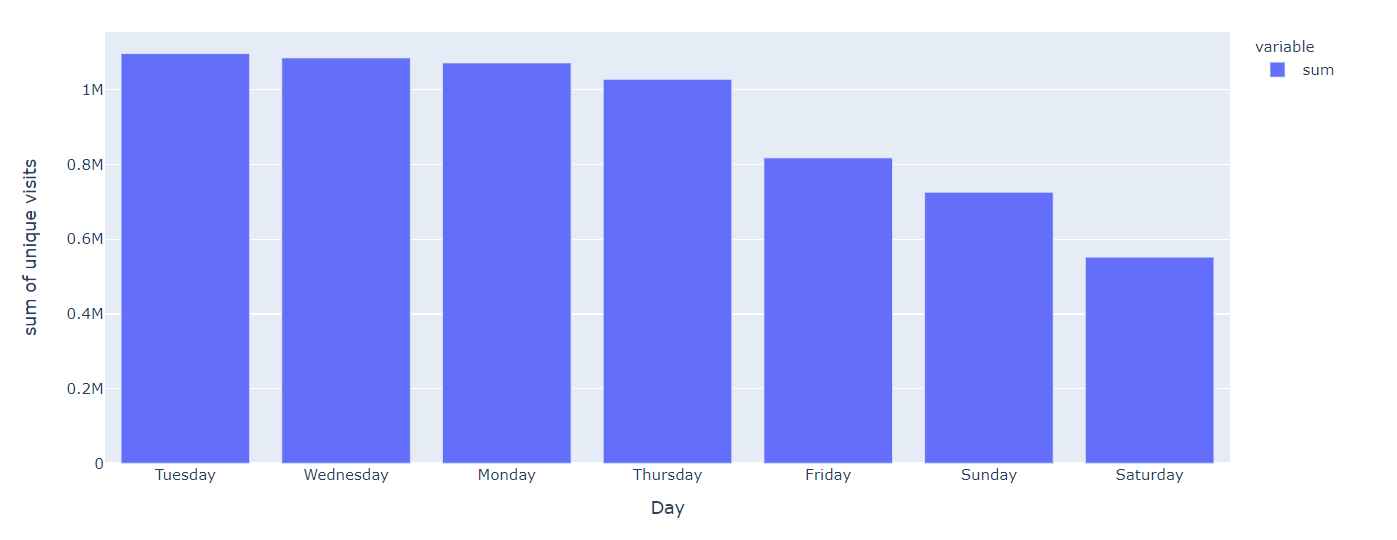


In:

day\_imp=df.groupby(['Day'])['unique\_visits'].agg(['sum']).sort\_values(by='sum',ascending=False)

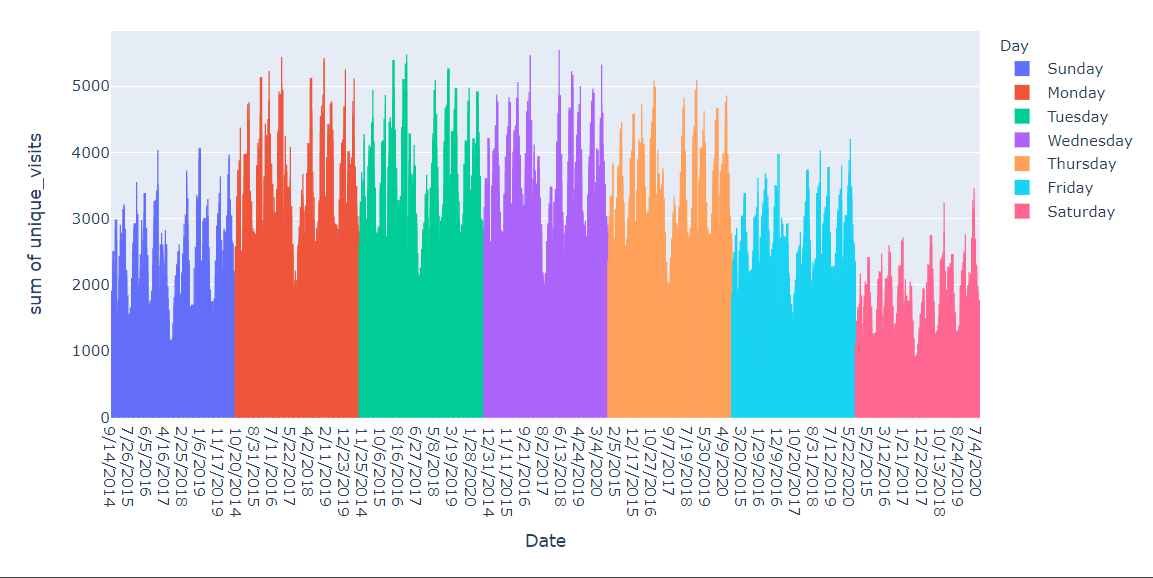
px.bar(day\_imp,labels={'value':'sum of unique visits'},title='Sum of Unique visits for each day')

out:



In:

px.histogram(df,x='Date',y='unique\_visits',color='Day',title='Sum of unique visits for each day over Time')

out:

In:

sums=df.groupby(['Day'])[['page\_loads' ,'unique\_visits' ,'first\_visits' ,'returning\_visits']].sum().sort\_values(

by='unique\_visits',ascending=False)

sums

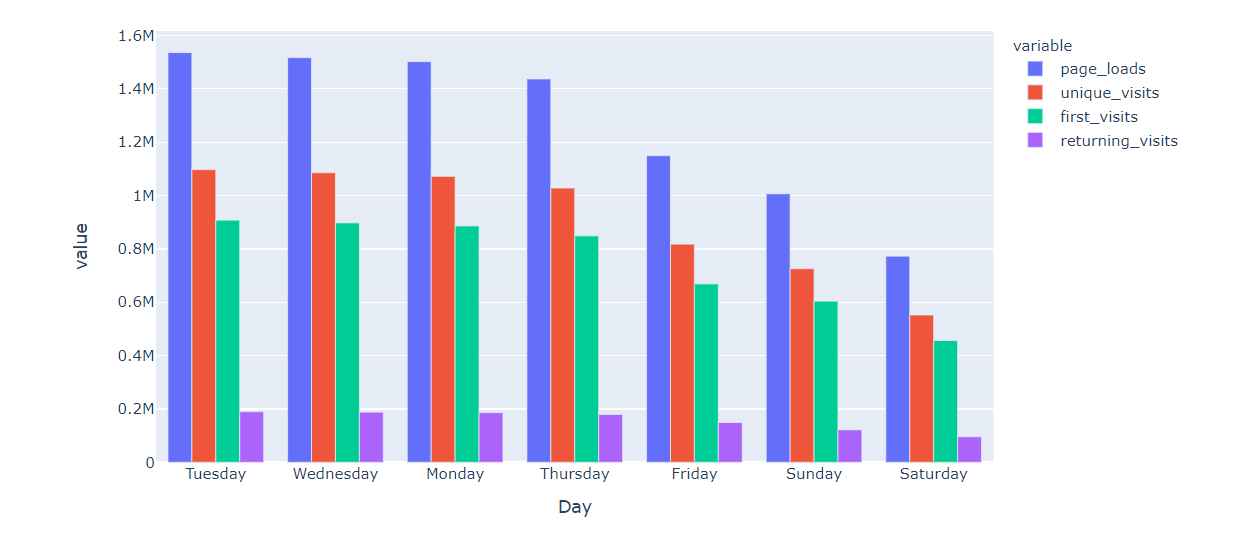
out:

| page\_loads | unique\_visits | first\_visits | returning\_visits |
| --- | --- | --- | --- |
| Day |  |  |  |
| Tuesday | 1536154 | 1097181 | 907752 |
| Wednesday | 1517114 | 1085624 | 897602 |

In:

px.bar(sums,barmode='group',title='Sum of page loads and visits for each of their days')

out:

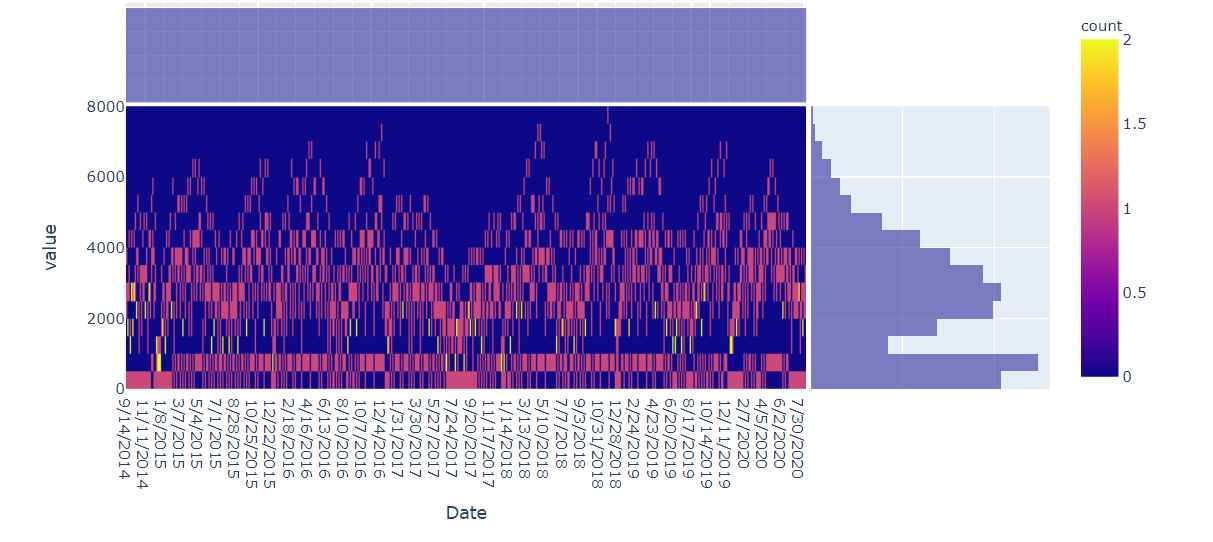


In:

px.density\_heatmap(df, x='Date',y=['page\_loads' ,'unique\_visits' ,'first\_visits' ,'returning\_visits']

*color\_continuous\_scale="Viridis"*

,marginal\_x="histogram", marginal\_y="histogram",title='Correlation for each data point')

Out: 

In:

fig, ax = plt.subplots()

fig.set\_size\_inches(8, 6)

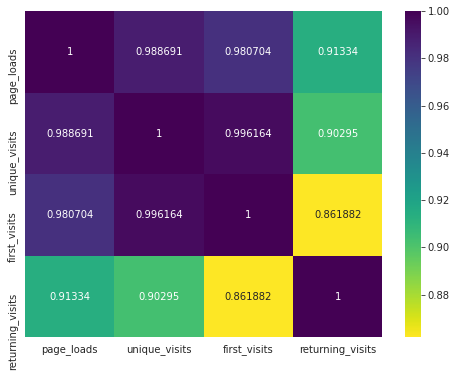
sns.heatmap(df[['page\_loads' ,'unique\_visits' ,'first\_visits' ,'returning\_visits']].corr(),

annot=True,

cmap='viridis\_r',

fmt='g')

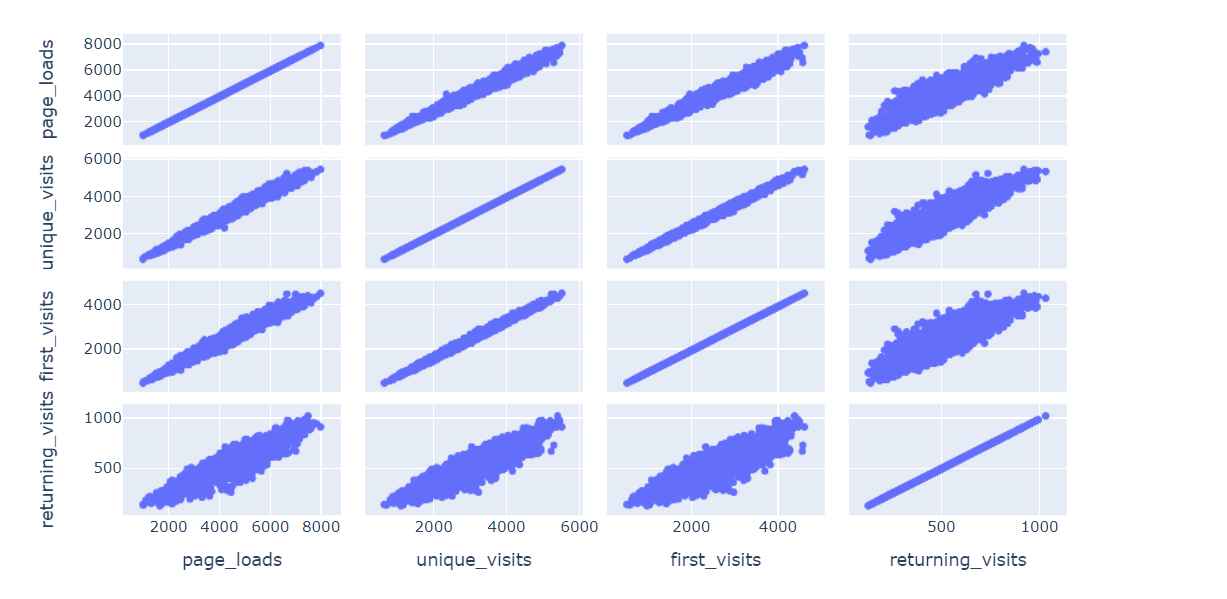
out:



In:

px.scatter\_matrix(df[['page\_loads' ,'unique\_visits' ,'first\_visits' ,'returning\_visits']])

out:



In:

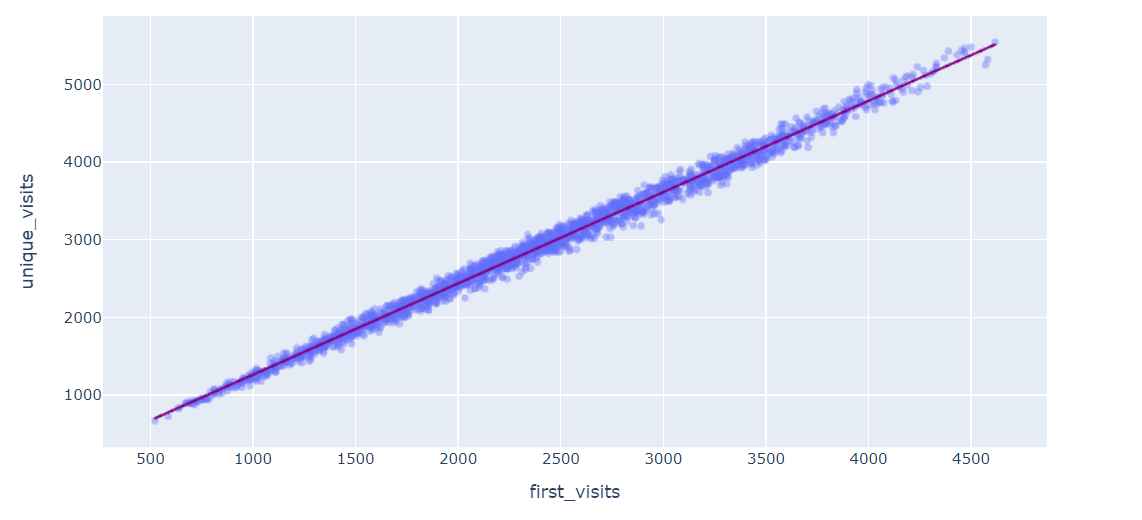
px.scatter(

df, x='first\_visits', y='unique\_visits',opacity=0.4,

trendline='ols', trendline\_color\_override='purple',title="Regression line for unique visits and first visits"

)

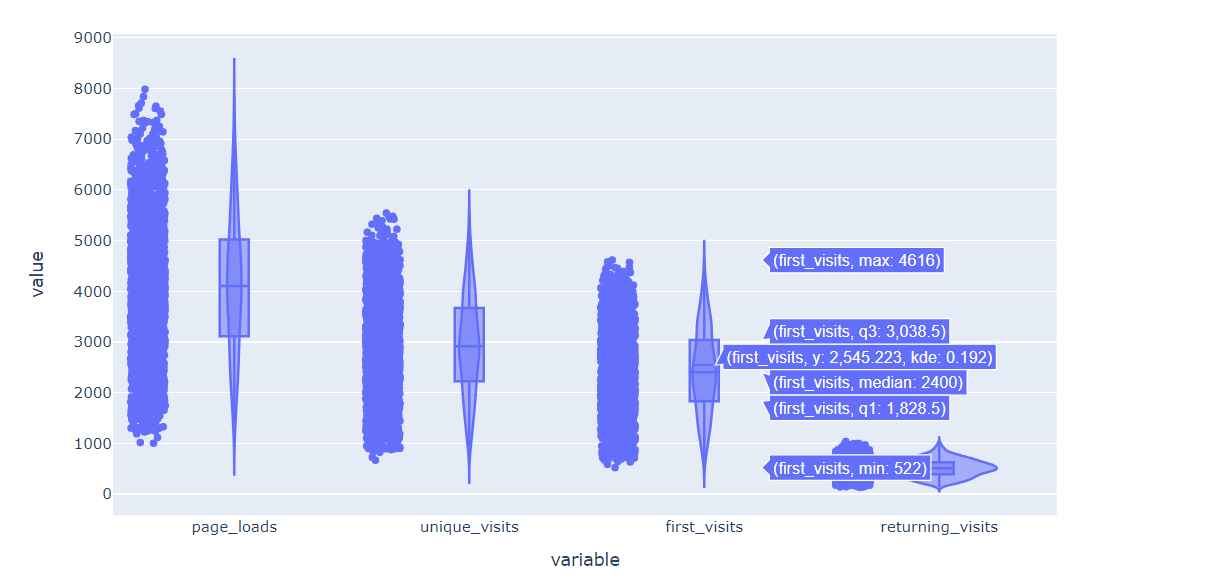
Out:



In:

px.violin(df,y=['page\_loads' ,'unique\_visits' ,'first\_visits' ,'returning\_visits'],box=True,points='all')

out:



In:

regressor2=LinearRegression(fit\_intercept=False,normalize=True)

regressor2.fit(X\_train, y\_train)

out:

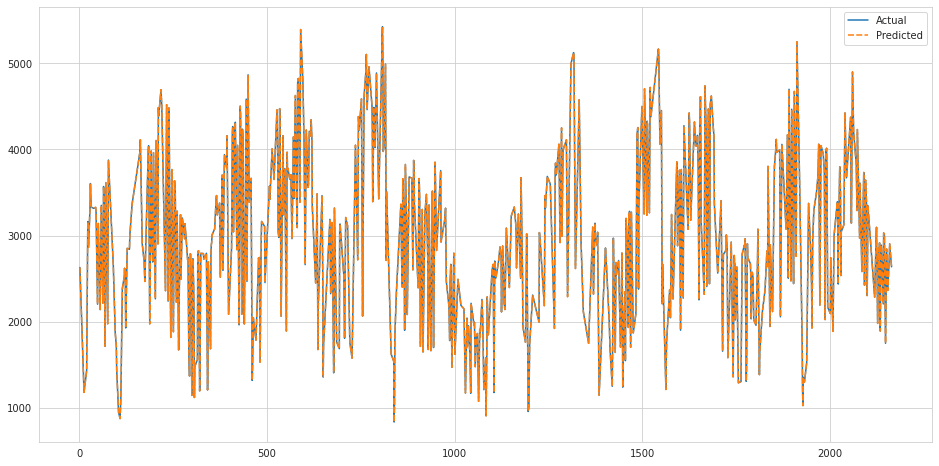
LinearRegression(fit\_intercept=False, normalize=True)

In:

plt.figure(figsize=(16,8))

sns.lineplot(data=lr2)

out:



In:

regressor2.score(X\_test,y\_test)\*100

out:

100.0

In:

svr\_rbf = SVR(kernel='rbf', C=1e3, gamma=0.00001)

svr\_rbf.fit(X\_train, y\_train)

out 19:

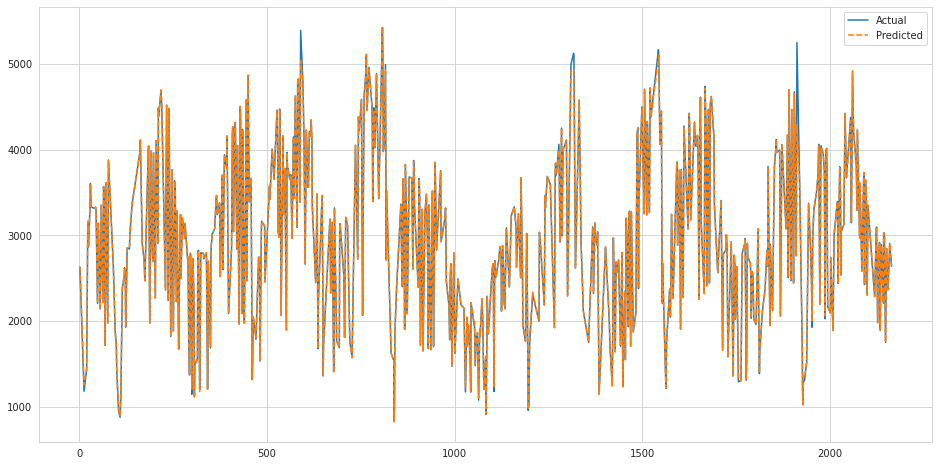
SVR(C=1000.0, gamma=1e-05)

In:

plt.figure(figsize=(16,8))

sns.lineplot(data=svr)

out:



In:

svr\_rbf.score(X\_test,y\_test)\*100

out:

99.80054455767926

In:

xgb\_r = xg.XGBRegressor(objective ='reg:squarederror',n\_estimators = 10, seed = 123)

xgb\_r.fit(X\_train, y\_train)

out:

XGBRegressor(base\_score=0.5, booster='gbtree', colsample\_bylevel=1,

colsample\_bynode=1, colsample\_bytree=1, gamma=0, gpu\_id=-1,

importance\_type='gain', interaction\_constraints='',

learning\_rate=0.300000012, max\_delta\_step=0, max\_depth=6,

min\_child\_weight=1, missing=nan, monotone\_constraints='()',

n\_estimators=10, n\_jobs=4, num\_parallel\_tree=1, random\_state=123,

reg\_alpha=0, reg\_lambda=1, scale\_pos\_weight=1, seed=123,

subsample=1, tree\_method='exact', validate\_parameters=1,

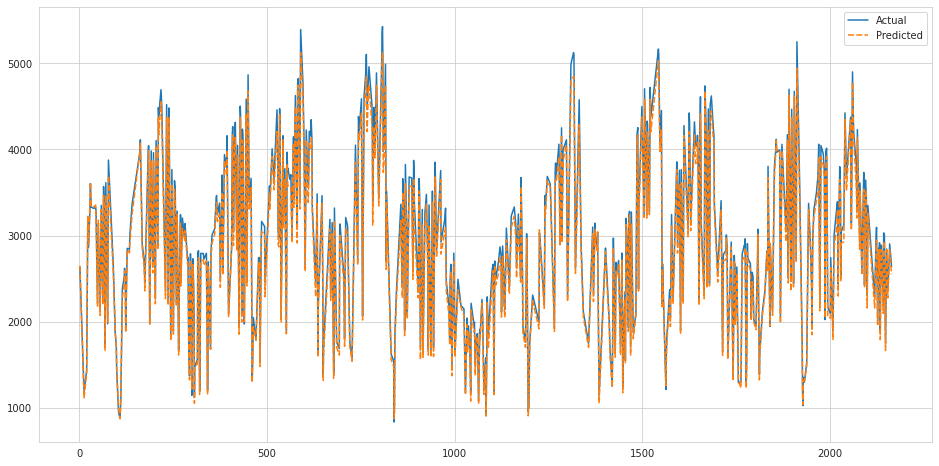
verbosity=None)

In:

plt.figure(figsize=(16,8))

sns.lineplot(data=xgb\_df)

out:



In:

xgb\_r.score(X\_test,y\_test)\*100

out:

98.7655882096893

**BENEFITS:-**

* Brings together your website data.
* Guides you to your target audience.
* Allows your business to track customer behavior.
* Provides actionable insights for improving customer experience.
* Helps build effective, customer-driven marketing campaigns.
* One of the main benefits of web analytics is that it can help businesses identify how they can make their website more effective. With all of the data that's collected (page views, conversion rates, bounce rates, etc.), businesses can start to see which pages are working well and which pages are not.
* This tells you how your web visitors are interacting with your information. Learning what programs and devices they use to find your content can help you better optimize your website to give them the best user experience when they visit.

**CONCLUSION:-**

Website traffic is important for many reasons. The more people see your site, the more potential customers you will have. The number of visitors to your website becomes the number of opportunities your business has at giving an impression, generating qualified leads, sharing your brand and building relationships