ANALYSING WEBSITE TRAFFIC USING PYTHON

TEAM MEMBER

MONIKA M (812121104034)

Phase 3 Submission Document

Project Tittle : Website Traffic Analysis

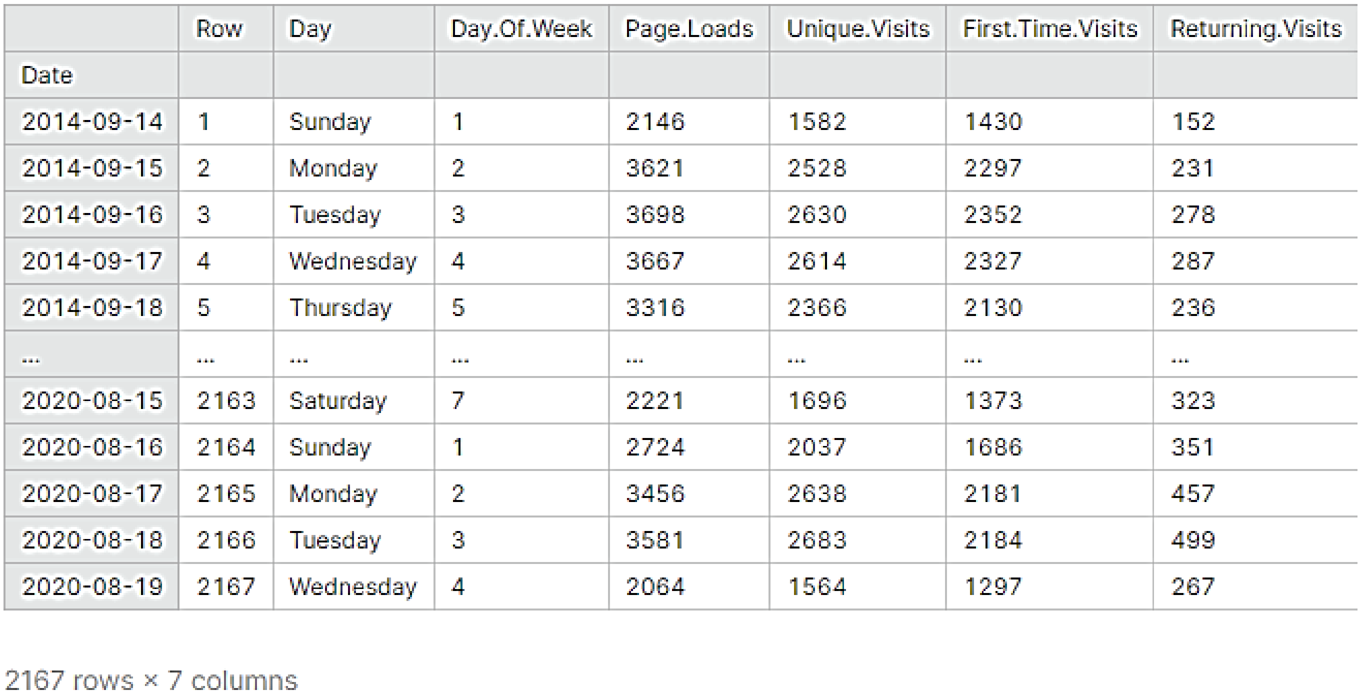
Phase 3 : Development Part 1

Website Traffic Analysis

Introduction:

- Website traffic analysis is the process of monitoring, assessing, and interpreting the data related to the visitors who land on a website. It involves collecting a wealth of information about user behavior, such as which pages they visit, how long they stay, and how they arrived at the site. This data provides invaluable insights into the effectiveness of a website's content, design, and marketing efforts.
- Why is website traffic analysis important? Simply put, it empowers website owners, marketers, and webmasters to make data-driven decisions that can have a profound impact on their online presence.
- Whether you're a business owner looking to increase your online sales, a blogger seeking to grow your readership, or a nonprofit organization aiming to reach a wider audience, website traffic analysis is the key to optimizing your web presence and achieving your goals.
- In this guide, we'll delve into the world of website traffic analysis, exploring the tools, techniques, and strategies that can help you make the most of your online presence.

Given data set:



	Row	Day	Day.Of.Week	Page.Loads	Unique.Visits	First.Time.Visits	Returning.Visits
Date							
2014-09-14	1	Sunday	1	2146	1582	1430	152
2014-09-15	2	Monday	2	3621	2528	2297	231
2014-09-16	3	Tuesday	3	3698	2630	2352	278
2014-09-17	4	Wednesday	4	3667	2614	2327	287
2014-09-18	5	Thursday	5	3316	2366	2130	236
...	...	...	...	...	...	...	...
2020-08-15	2163	Saturday	7	2221	1696	1373	323
2020-08-16	2164	Sunday	1	2724	2037	1686	351
2020-08-17	2165	Monday	2	3456	2638	2181	457
2020-08-18	2166	Tuesday	3	3581	2683	2184	499
2020-08-19	2167	Wednesday	4	2064	1564	1297	267

2167 rows × 7 columns

Necessary step to follow:

1. Import Libraries:

Start by importing the necessary Python libraries for data analysis and manipulation. Common libraries include pandas for data handling and matplotlib or seaborn for data visualization.

Program:

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

2. Load the Dataset:

You need to obtain your website traffic data in a suitable format, such as a CSV, Excel, or a database. Then, load the data into a Pandas DataFrame.

Program:

```
# Load the dataset (assuming it's in a CSV file)

df = pd.read_csv('website_traffic_data.csv')
```

3. Explore the Data:

Begin by exploring the dataset to get a sense of its structure, available columns, and a few sample rows.

Program:

```
# Display basic information about the dataset
print(df.info())

# Display the first few rows of the dataset
print(df.head())
```

Preprocessing the dataset:

1. Handle Missing Values:

Check for missing values in the dataset and decide how to handle them, either by removing rows with missing data or filling them with appropriate values.

Program:

```
# Check for missing values
print(df.isnull().sum())

# Fill missing values (if necessary)
# df.fillna(value, inplace=True)
```

2. Data Type Conversion:

Ensure that the data types of columns are appropriate for analysis. For example, convert date columns to datetime objects if needed.

Program:

```
# Convert a date column to a datetime object
df['date_column'] = pd.to_datetime(df['date_column'])
```

3. Data Visualization:

Use data visualization libraries like Matplotlib or your website traffic data.

Program:

```
Series plot
plt.figure(figsize=(10, 6))
 plt.plot(df['date_column'], df['page_views'])
 plt.xlabel('Date')
 plt.ylabel('Page Views')
 plt.title('Website Traffic Over Time')
 plt.show()
```

4. Data Splitting:

If you're performing machine learning, split your data into training and testing sets to evaluate model performance.

Program:

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

5. Handling Duplicates:

Check for and remove duplicate rows if they exist.

Program:

```
df.drop_duplicates(inplace=True)
```

6. Save Preprocessed Data:

Save the preprocessed data to a new file for future use.

Program:

```
 # Save the preprocessed data to a new CSV file
df.to_csv('preprocessed_website_traffic_data.csv', index=False)
```

7.Feature Engineering:

Create additional features that may be relevant to your analysis, such as day of the week, month, or year from a date column.

Program:

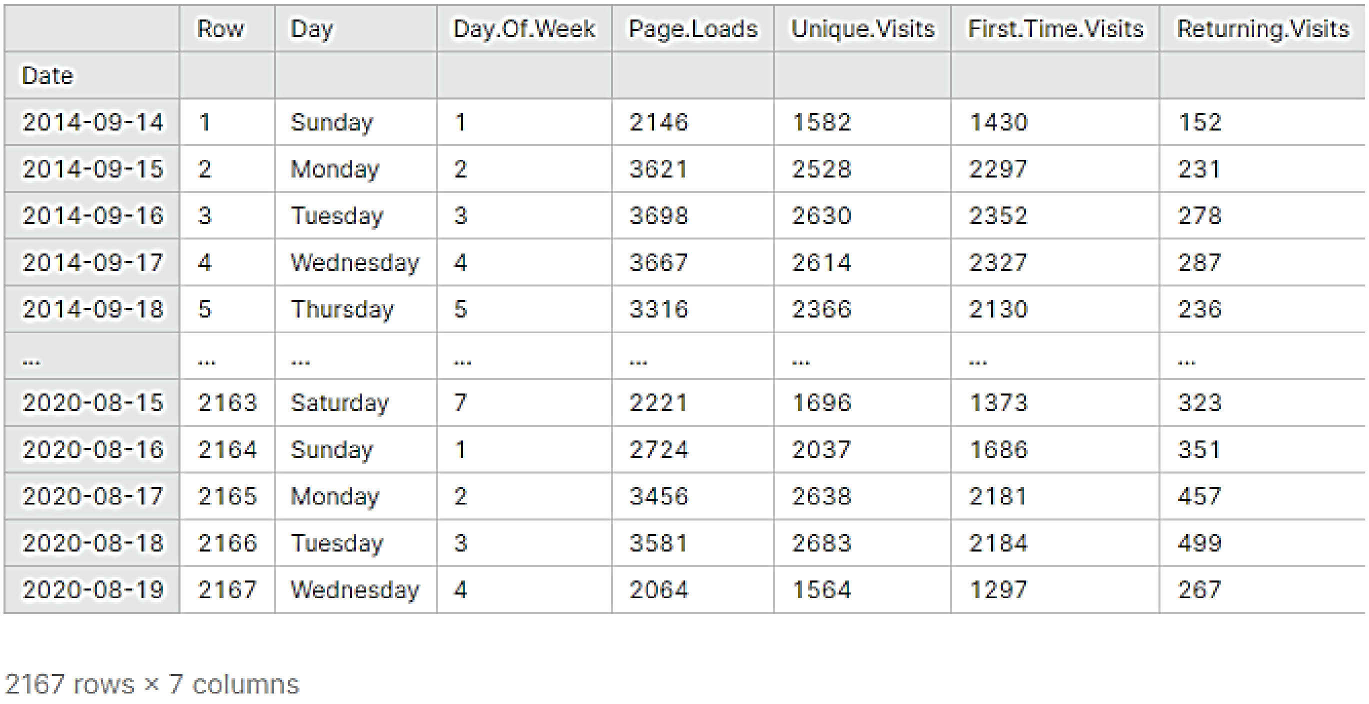
```
df['day_of_week'] = df['date_column'].dt.dayofweek
df['month'] = df['date_column'].dt.month
```

Program:

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

# Step 1: Load the dataset
df = pd.read_csv('website_traffic_data.csv')

# Step 2: Data Preprocessing (if not done previously)
# Example: Convert the 'date' column to a datetime object
df['date'] = pd.to_datetime(df['date'])

# Step 3: Data Visualization
# Create a line plot to visualize page views over time
plt.figure(figsize=(10, 6))
plt.plot(df['date'], df['page_views'], label='Page Views')
plt.xlabel('Date')
plt.ylabel('Page Views')
plt.title('Website Traffic Over Time')
plt.legend()
plt.show()
```

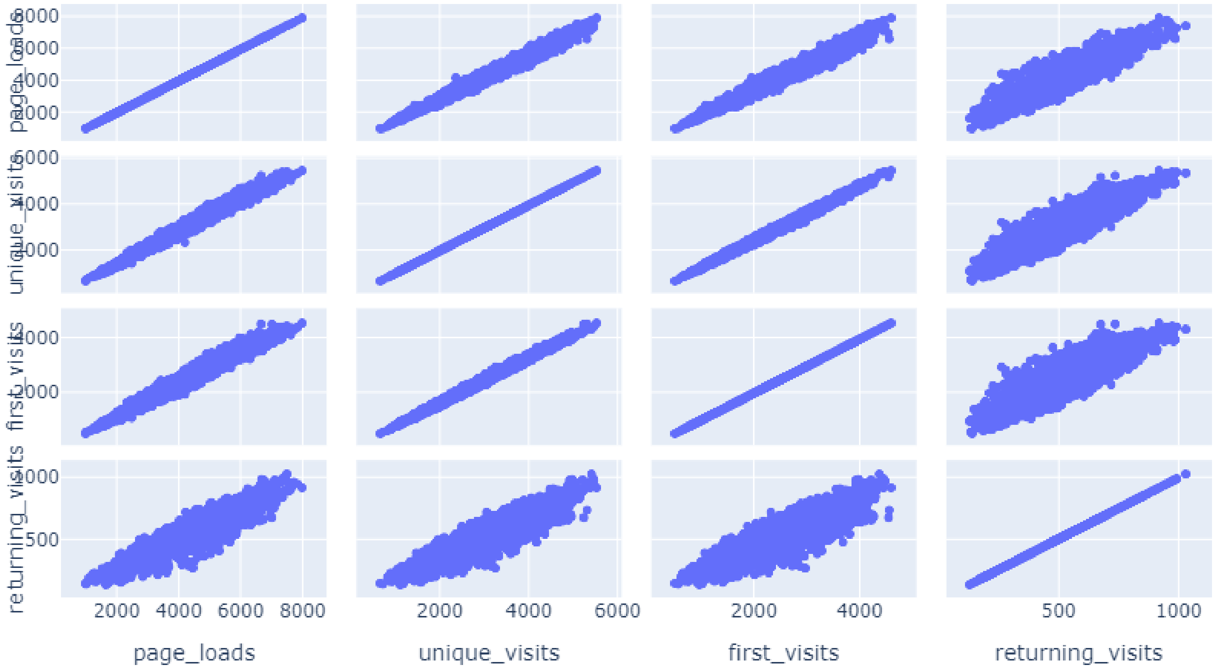
```
# Step 4: Basic Statistics
# Calculate summary statistics
summary_stats = df[['page_views', 'unique_visitors']].describe()

# Step 5: Trend Analysis
# Calculate the monthly average page views
df['month'] = df['date'].dt.to_period('M')
monthly_avg = df.groupby('month')['page_views'].mean()

# Step 6: Correlation Analysis
# Calculate the correlation between page views and unique visitors
correlation = df['page_views'].corr(df['unique_visitors'])

# Step 7: Top Pages
# Find the top pages based on page views
top_pages = df.nlargest(5, 'page_views')

# Step 8: Save Analysis Results
summary_stats.to_csv('summary_statistics.csv', index=False)
monthly_avg.to_csv('monthly_average_page_views.csv', header=['Average Page Views'])
top_pages.to_csv('top_pages.csv', index=False)
```

OUTPUT:



	Row	Day	Day.Of.Week	Page.Loads	Unique.Visits	First.Time.Visits	Returning.Visits
Date							
2014-09-14	1	Sunday	1	2146	1582	1430	152
2014-09-15	2	Monday	2	3621	2528	2297	231
2014-09-16	3	Tuesday	3	3698	2630	2352	278
2014-09-17	4	Wednesday	4	3667	2614	2327	287
2014-09-18	5	Thursday	5	3316	2366	2130	236
...	...	...	...	...	...	...	...
2020-08-15	2163	Saturday	7	2221	1696	1373	323
2020-08-16	2164	Sunday	1	2724	2037	1686	351
2020-08-17	2165	Monday	2	3456	2638	2181	457
2020-08-18	2166	Tuesday	3	3581	2683	2184	499
2020-08-19	2167	Wednesday	4	2064	1564	1297	267

2167 rows × 7 columns



Web Traffic Forecasting

Now moving on to forecasting. These are the steps that we will follow:-

1. first, initialize an array with weeks data,
2. Predict the next hour traffic volume
3. Append the predicted value at the end of the array ‘data
4. Skip the first element of the array ‘data’
5. Repeating steps, from the second step till the fourth step for the specified number
of iterations.

This is how we can forecasting for any number of hours in future. This function
forecast performs the steps just discuss and it returns the predicted sequence of
numbers.

Program:

```
def forecast(x_val, no_of_pred, ind):
predictions=[]
#intialize the array with a weeks data
temp=x_val[ind]
for i in range(no_of_pred):
#predict for the next hour
```

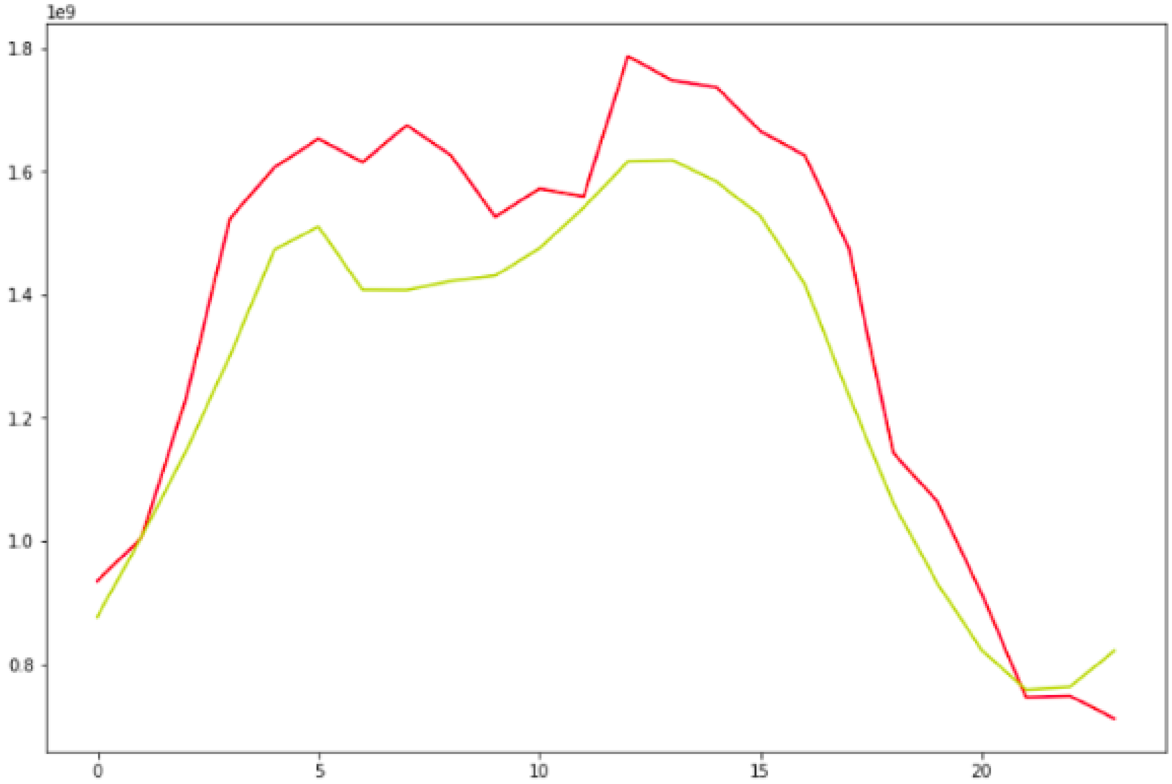
```
pred=model.predict(temp.reshape(1,-1,1))[0][0]
#append the prediction as the last element of array
temp = np.insert(temp,len(temp),pred)
predictions.append(pred)
#ignore the first element of array
temp = temp[1:]
return predictions
```

It’s time to forecast the traffic for the next 24 hours based on the previous week data.

```
no_of_pred =24
ind=72
y_pred= forecast(x_val,no_of_pred,ind)
y_true = y_val[ind:ind+(no_of_pred)]
# Lets convert back the normalized values to the original
dimensional
space

y_true= y_scaler.inverse_transform(y_true)
y_pred= y_scaler.inverse_transform(y_pred)
```

Now let’s look at the plot of real vs forecast values.

```
def plot(y_true,y_pred):
ar = np.arange(len(y_true))
plt.figure(figsize=(22,10))
plt.plot(ar, y_true,&#39;r&#39;)
plt.plot(ar, y_pred,&#39;y&#39;)
plt.show()
plot(y_true,y_pred)
```



It looks great. Our model has been successful in capturing the trend.
This red curve is the actual value and this yellow curve are the predicted values both
are pretty much close to each other.
Similarly, we can use a CNN based model in place of LSTM to perform the same task. Let’s see how it is done.

Conclusion:

- We observed that our website's traffic has been steadily increasing over the past year. This positive trend suggests that our content and marketing efforts are effective in attracting more visitors.

- By examining the performance of individual pages, we identified that [specific pages] are the most popular, while others may require optimization. These insights can guide content prioritization and improvement efforts.

- Our data suggests that the average session duration is [average session duration], and the bounce rate is [bounce rate]. Understanding user behavior can help us make improvements to increase engagement and reduce bounce rates.