Website Traffic Analysis

Date	**26-09-2023**
Team ID	**1295**
Project Name	**Website Traffic Analysis**

PREPROCESSING THE GIVEN WEBSITE TRAFFIC ANALYSIS DATASET

1. REMOVING THE NULL VALUES

2. OUTLIER DETECTION

3. OBTAINING THE PREPROCESSED DATA

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

```
data =pd.read_csv("/content/daily-website-visitors.csv")
```

```
data
```

```
       Row        Day  Day.Of.Week       Date Page.Loads Unique.Visits  \
0        1     Sunday            1  9/14/2014      2,146         1,582   
1        2     Monday            2  9/15/2014      3,621         2,528   
2        3    Tuesday            3  9/16/2014      3,698         2,630   
3        4  Wednesday            4  9/17/2014      3,667         2,614   
4        5   Thursday            5  9/18/2014      3,316         2,366   
...    ...        ...          ...        ...        ...           ...   
2162  2163   Saturday            7  8/15/2020      2,221         1,696   
2163  2164     Sunday            1  8/16/2020      2,724         2,037   
2164  2165     Monday            2  8/17/2020      3,456         2,638   
2165  2166    Tuesday            3  8/18/2020      3,581         2,683   
2166  2167  Wednesday            4  8/19/2020      2,064         1,564   

     First.Time.Visits Returning.Visits  
0                1,430              152  
1                2,297              231  
2                2,352              278  
3                2,327              287  
4                2,130              236  
```

```
...                ...                ...
2162             1,373                323
2163             1,686                351
2164             2,181                457
2165             2,184                499
2166             1,297                267

[2167 rows x 8 columns]
```

1. REMOVING THE NULL VALUES

```
missing_values = data.isnull().sum()
print(missing_values)
```

```
Row                    0
Day                    0
Day.Of.Week            0
Date                   0
Page.Loads             0
Unique.Visits          0
First.Time.Visits      0
Returning.Visits       0
dtype: int64
```

1.1 VISUALIZING THE NULL VALUED FEATURES TO

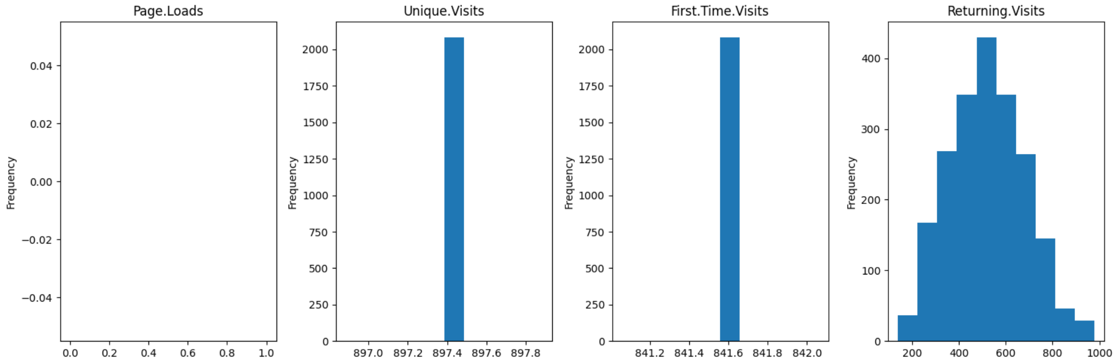
DETERMINE THE METHOD TO FILL THE DATA

SINCE THE DATASET HAS LOW NUMBER OF ROWS REMOVING THEM

WILL RESULT IN LACK OF ACCURACY

```
import matplotlib.pyplot as plt

# Create subplots for each column with missing values
fig, axes = plt.subplots(nrows=1, ncols=4, figsize=(15, 5))

# Plot histograms for page.load, unique.visits, first.time.visits ,
return.visits
data['Page.Loads'].plot(kind='hist', ax=axes[0], title='Page.Loads')
```

```
data['Unique.Visits'].plot(kind='hist', ax=axes[1], title='Unique.Visits')
data['First.Time.Visits'].plot(kind='hist', ax=axes[2],
title='First.Time.Visits')
data['Returning.Visits'].plot(kind='hist', ax=axes[3],
title='Returning.Visits')


plt.tight_layout()
plt.show()
```



1.2 REPLACING THE NULL VALUES WITH THEIR

RESPECTIVE MEAN VALUES

```
# Fill missing values with mean
data['Page.Loads'].fillna(data['Page.Loads'].mean(), inplace=True)
data['Unique.Visits'].fillna(data['Unique.Visits'].mean(), inplace=True)
data['First.Time.Visits'].fillna(data['First.Time.Visits'].mean(),
inplace=True)
data['Returning.Visits'].fillna(data['Returning.Visits'].mean(),
inplace=True)

# Verify that there are no more missing values
missing_values_after_filling = data.isnull().sum()
print(missing_values_after_filling)
```

```
Row                     0
Day                     0
Day.Of.Week             0
Date                    0
Page.Loads           2167
Unique.Visits           0
First.Time.Visits       0
Returning.Visits        0
dtype: int64
```

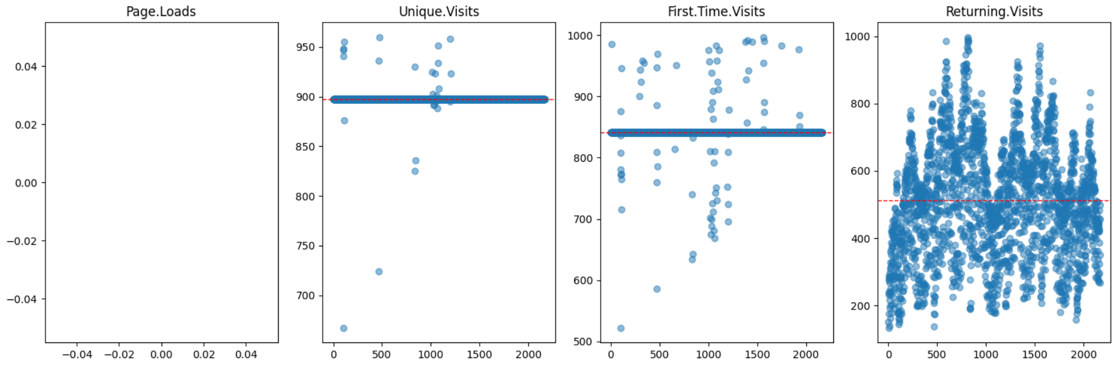
2. OUTLIER DETECTION

```
import numpy as np
import pandas as pd

# Define a function to detect outliers using IQR
def detect_outliers(column):
    Q1 = np.percentile(column, 25)
    Q3 = np.percentile(column, 75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return (column < lower_bound) | (column > upper_bound)

# Specify the numeric columns to detect outliers
numeric_columns = ['Page.Loads', 'Unique.Visits', 'First.Time.Visits',
'Returning.Visits']

# Convert columns to numeric data type (if needed)
data[numeric_columns] = data[numeric_columns].apply(pd.to_numeric,
errors='coerce')

# Apply outlier detection to each numerical column separately
outliers = data[numeric_columns].apply(detect_outliers)

# Print the number of outliers for each column
print(outliers.sum())
```

```
Page.Loads            0
Unique.Visits        26
First.Time.Visits    77
Returning.Visits      5
dtype: int64
```

2.1 USING SCATTER PLOT TO VISUALIZE THE OUTLIERS

```
# Calculate mean values
mean_Page_Loads = data['Page.Loads'].mean()
mean_Unique_Visits = data['Unique.Visits'].mean()
mean_First_Time_Visits = data['First.Time.Visits'].mean()
mean_Returing_Visits = data['Returning.Visits'].mean()

# Create subplots for each column
fig, axes = plt.subplots(nrows=1, ncols=4, figsize=(15, 5))

# Plot scatter plots for page.load, unique.visits, first.time.visits ,
return.visits  against index
axes[0].scatter(data.index, data['Page.Loads'], alpha=0.5)
axes[0].axhline(mean_Page_Loads, color='red', linestyle='dashed',
linewidth=1)
axes[0].set_title('Page.Loads')

axes[1].scatter(data.index, data['Unique.Visits'], alpha=0.5)
axes[1].axhline(mean_Unique_Visits, color='red', linestyle='dashed',
linewidth=1)
axes[1].set_title('Unique.Visits')

axes[2].scatter(data.index, data['First.Time.Visits'], alpha=0.5)
axes[2].axhline(mean_First_Time_Visits, color='red', linestyle='dashed',
linewidth=1)
axes[2].set_title('First.Time.Visits')

axes[3].scatter(data.index, data['Returning.Visits'], alpha=0.5)
axes[3].axhline(mean_Returing_Visits, color='red', linestyle='dashed',
linewidth=1)
axes[3].set_title('Returning.Visits')

plt.tight_layout()
plt.show()
```



2.2 REMOVING THE OUTLIER

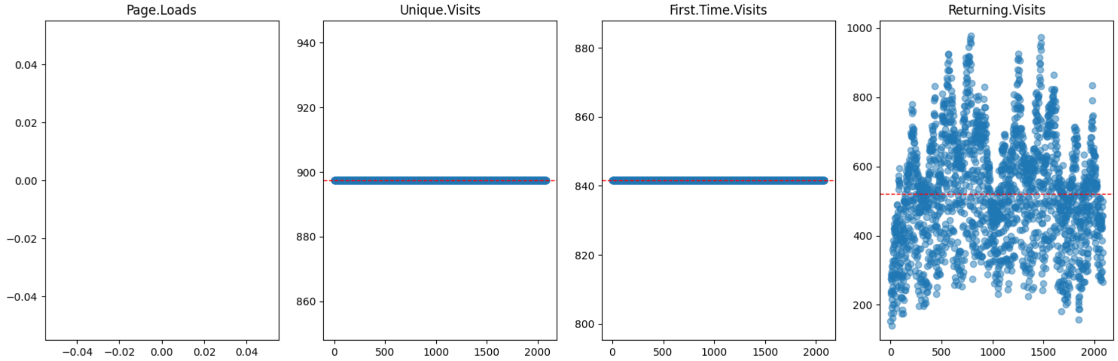
```
import numpy as np

# Define a function to detect outliers using IQR
def detect_outliers(column):
    Q1 = np.percentile(column, 25)
    Q3 = np.percentile(column, 75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return (column < lower_bound) | (column > upper_bound)
```

```
# Apply outlier detection to numerical columns (page.load, unique.visits,
first.time.visits , return.visits)
outliers_page_loads = detect_outliers(data['Page.Loads'])
outliers_uniqus_visits = detect_outliers(data['Unique.Visits'])
outliers_first_time_visits = detect_outliers(data['First.Time.Visits'])
outliers_returning_visits= detect_outliers(data['Returning.Visits'])

# Remove outliers
data = data[~(outliers_page_loads | outliers_uniqus_visits  |
outliers_first_time_visits | outliers_returning_visits )]

# Reset index after removing rows
data.reset_index(drop=True, inplace=True)

# Verify that outliers are removed

print(f'Number of rows after removing outliers: {data.shape[0]}')
```

```
Number of rows after removing outliers: 2085
```

```
# Calculate mean values
mean_page_loads = data['Page.Loads'].mean()
mean_uniqus_visits = data['Unique.Visits'].mean()
mean_first_time_visits = data['First.Time.Visits'].mean()
mean_returning_visits = data['Returning.Visits'].mean()

# Create subplots for each column
fig, axes = plt.subplots(nrows=1, ncols=4, figsize=(15, 5))

# Plot scatter plots for page.load, unique.visits, first.time.visits ,
return.visits against index
axes[0].scatter(data.index, data['Page.Loads'], alpha=0.5)
axes[0].axhline(mean_page_loads, color='red', linestyle='dashed',
linewidth=1)
axes[0].set_title('Page.Loads')

axes[1].scatter(data.index, data['Unique.Visits'], alpha=0.5)
axes[1].axhline(mean_uniqus_visits, color='red', linestyle='dashed',
linewidth=1)
axes[1].set_title('Unique.Visits')

axes[2].scatter(data.index, data['First.Time.Visits'], alpha=0.5)
axes[2].axhline(mean_first_time_visits, color='red', linestyle='dashed',
linewidth=1)
axes[2].set_title('First.Time.Visits')

axes[3].scatter(data.index, data['Returning.Visits'], alpha=0.5)
axes[3].axhline(mean_returning_visits, color='red', linestyle='dashed',
linewidth=1)
```

```
axes[3].set_title('Returning.Visits')

plt.tight_layout()
plt.show()
```



3.OBTAINING THE PREPROCESSED DATA

```
# Assuming 'data' is your cleaned DataFrame

data.to_csv('/content/daily-website-visitors.csv', index=False)
```

WORKING WITH IBM COGNOS

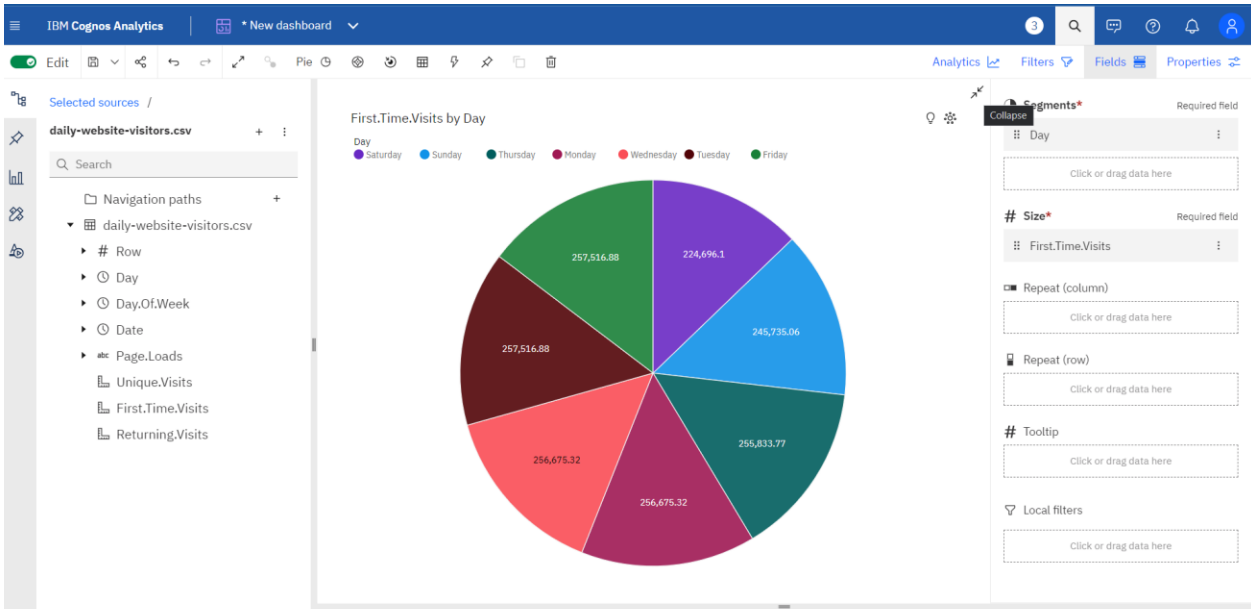
INSIGHTS GATHERED FROM IBM COGNOS

First.Time.Visits has a moderate upward trend.

Add insight to favorites

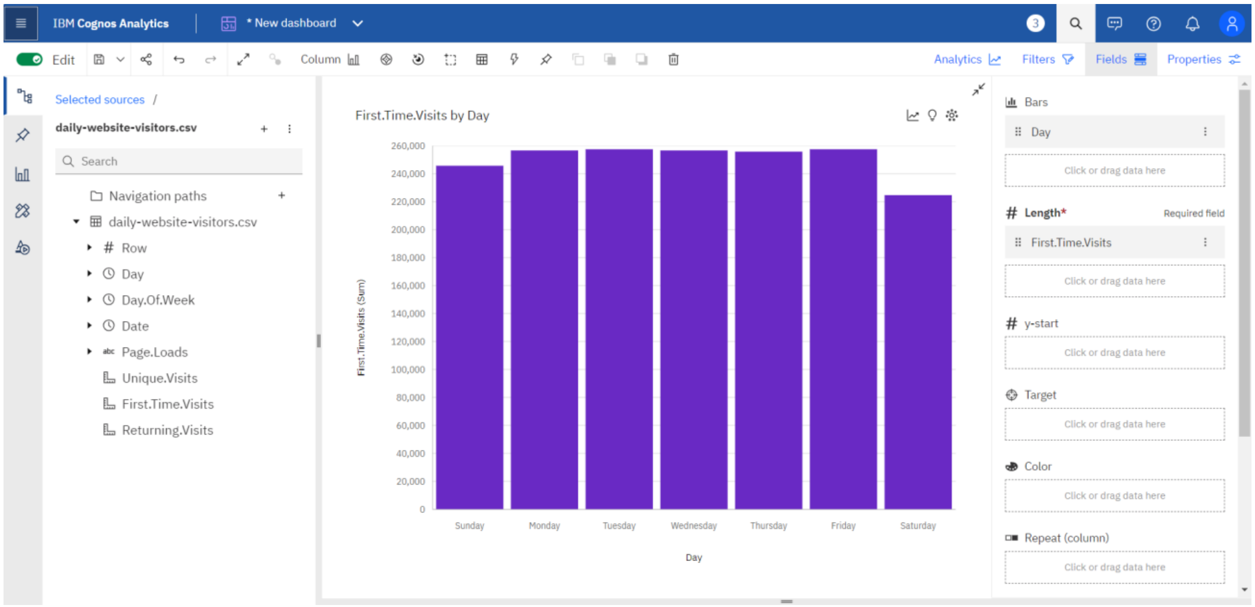
- Based on the current forecasting, First.Time.Visits may reach over 239 thousand by Day Monday+1.
- Friday (14.7 %), Tuesday (14.7 %), Monday (14.6 %), Wednesday (14.6 %), and Thursday (14.6 %) are the most frequently occurring categories of Day with a combined count of 1526 items with First.Time.Visits values (73.2 % of the total) .
- Over all days, the average of First.Time.Visits is 841.6.
- The total number of results for First.Time.Visits, across all days, is over two thousand.

```
First.Time.Visits ranges from nearly 225 thousand, when Day is Saturday, to
nearly 258 thousand, when Day is Tuesday.
```



First.Time.Visits has a moderate upward trend.

- Based on the current forecasting, First.Time.Visits may reach over 239 thousand by Day Monday+1.
- Over all days, the sum of First.Time.Visits is almost 1.8 million.
- First.Time.Visits ranges from nearly 225 thousand, when Day is Saturday, to nearly 258 thousand, when Day is Tuesday.
- For First.Time.Visits, the most significant values of Day are Friday, Tuesday, Monday, Wednesday, and Thursday, whose respective First.Time.Visits values add up to almost 1.3 million, or 73.2 % of the total.

- 

Returning.Visits by Day

Based on the current forecasting, Returning.Visits may reach 183.4 by Day Monday+1.

Add insight to favorites

The total number of results for Day, across all Returning.Visits, is over two thousand.

Based on the current forecasting, Unique.Visits may reach 1 by Day.Of.Week 9.

- 3 (14.7 %), 6 (14.7 %), 2 (14.6 %), 4 (14.6 %), and 5 (14.6 %) are the most frequently occurring categories of Day.Of.Week with a combined count of 1526 items with First.Time.Visits values (73.2 % of the total) .

- 3 (14.7 %), 6 (14.7 %), 2 (14.6 %), 4 (14.6 %), and 5 (14.6 %) are the most frequently occurring categories of Day.Of.Week with a combined count of 1526 items with Returning.Visits values (73.2 % of the total) .

- Over all values of Day.Of.Week, the average of First.Time.Visits is 841.6.

- Over all values of Day.Of.Week, the average of Returning.Visits is 521.4.

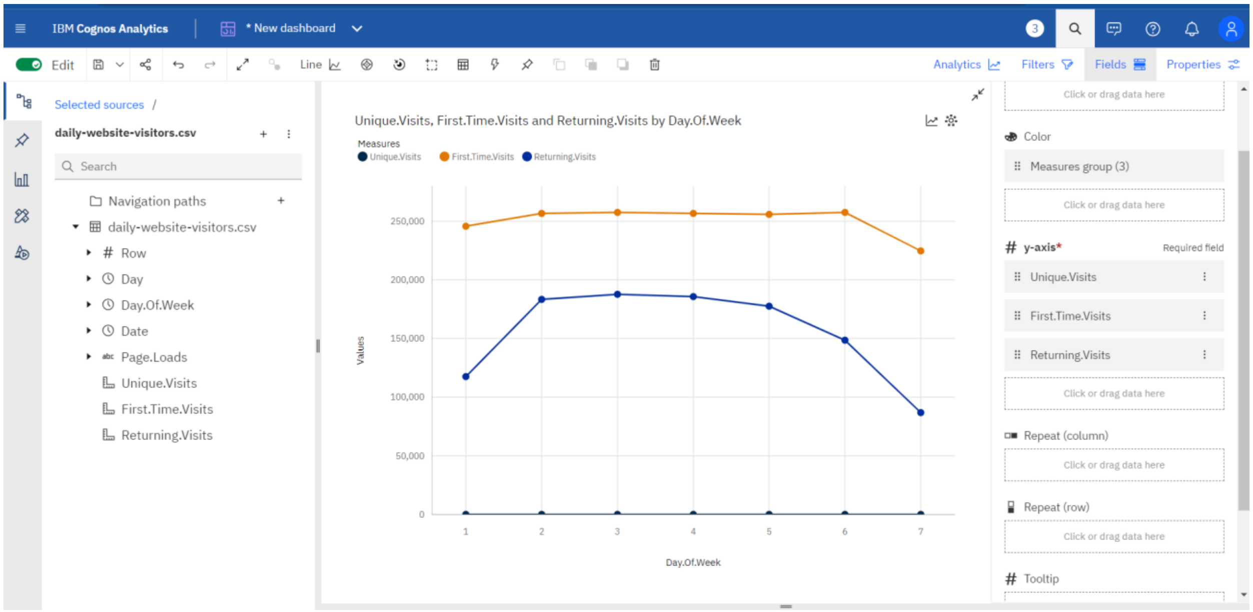
- The total number of results for First.Time.Visits, across all Day.Of.Week, is over two thousand.

- The total number of results for Returning.Visits, across all Day.Of.Week, is over two thousand.

- The total number of results for Unique.Visits, across all Day.Of.Week, is over two thousand.

- First.Time.Visits ranges from nearly 225 thousand, when Day.Of.Week is 7, to nearly 258 thousand, when Day.Of.Week is 3.

- Returning.Visits ranges from nearly 87 thousand, when Day.Of.Week is 7, to almost 188 thousand, when Day.Of.Week is 3.



Color
Measures group (3)

y-axis* Required field
Unique.Visits
First.Time.Visits
Returning.Visits

CONCLUSION

In this phase the given website traffic analysis dataset in preprocessed through such activities like removing he null values and outliers. Then the dataset in loaded into the IBM COGNOS to perform various visualizations and insights collected from them about the website traffic.