**PROJECT: WEBSITE TRAFFIC ANALYSIS**

Project Overview

This project document outlines the goals, objectives, scope, and methodology for a website traffic analysis project. The goal of this project is to gain a deeper understanding of the website's traffic patterns, visitor behavior, and key performance indicators. This information will be used to improve the website's user experience, optimize its marketing campaigns, and increase its conversion rates.

Objectives

The specific objectives of this project are to:

* Identify the website's top traffic sources
* Understand the visitor's journey through the website
* Analyze the website's key performance indicators (KPIs), such as bounce rate, page views per session, and conversion rates
* Identify areas for improvement in the website's user experience and marketing campaigns

Scope

This project will focus on the following aspects of the website's traffic:

* Traffic sources: Where are visitors coming from? Are they coming from search engines, social media, or other websites?
* Visitor behavior: What pages are visitors viewing? How long are they staying on the website? What actions are they taking?
* Key performance indicators (KPIs): What are the website's bounce rate, page views per session, and conversion rates? How do these KPIs compare to industry benchmarks?
* Areas for improvement: What areas of the website's user experience and marketing campaigns can be improved to increase traffic and conversions?

Methodology

The following methodology will be used to collect and analyze the website's traffic data:

1. Data collection: The website's traffic data will be collected using a web analytics tool, such as Google Analytics or Adobe Analytics.
2. Data cleaning and preparation: The traffic data will be cleaned and prepared for analysis. This may involve removing spam and bot traffic, correcting data formatting errors, and aggregating the data into meaningful categories.
3. Data analysis: The traffic data will be analyzed using a variety of statistical and visualization techniques. This analysis will be used to identify trends, patterns, and insights into the website's traffic.
4. Reporting and recommendations: A report will be generated that summarizes the findings of the analysis and provides recommendations for improving the website's user experience and marketing campaigns.

Resources

The following resources will be needed to complete the project:

* Access to the website's web analytics data
* A web analytics tool, such as IBM cognos analytics.

**Conclusion**

* This website traffic analysis project will provide valuable insights into the website's traffic patterns, visitor behavior, and key performance indicators. This information will be used to improve the website's user experience, optimize its marketing campaigns, and increase its conversion rates.

**Problem Definition and Design Thinking**

**Project Definition:**The project involves analyzing website traffic data to gain insights into user behavior, popular pages, and traffic sources. The goal is to help website owners enhance the user experience by understanding how visitors interact with the site. This project encompasses defining the analysis objectives, collecting website traffic data, using IBM Cognos for data visualization, and integrating Python code for advanced analysis.

**Design Thinking:**

1. Analysis Objectives: Define the key insights you want to extract from the website traffic data, such as identifying popular pages, traffic trends, and user engagement metrics.
2. Data Collection: Determine the data sources and methods for collecting website traffic data, including page views, unique visitors, referral sources, and more.
3. Visualization: Plan how to visualize the insights using IBM Cognos to create meaningful dashboards and reports.
4. Python Integration: Consider incorporating machine learning models to predict future traffic trends or user behavior patterns.

**Analysis Objectives:**

* Define specific objectives for your analysis. For example, you could aim to identify the top 10 most visited pages on the website, understand the demographic of the visitors, track how traffic changes over time (daily, weekly, or monthly trends), and determine which referral sources drive the most traffic.
* Ensure that your objectives are aligned with the overall goal of enhancing the user experience.

**Data Collection:**

* Identify the data sources you'll tap into. This might include website analytics tools like Google Analytics, server logs, or custom tracking scripts.
* Define the frequency of data collection. Will it be real-time, daily, weekly, or monthly?
* Consider data storage and retention policies. How long will you keep historical data, and what data privacy and security measures will you implement?

**Visualization:**

* Plan how you'll present the insights to stakeholders. This could involve creating interactive dashboards, reports, or data visualizations.
* Choose appropriate visualization tools. You mentioned using IBM Cognos, which is a good choice for creating meaningful reports and dashboards.
* Consider the audience for your visualizations. Tailor your reports to the needs and preferences of different stakeholders.

**Python Integration:**

* Determine which Python libraries and tools you'll use for data analysis. Libraries like Pandas, Numpy, and Matplotlib or Seaborn for data manipulation and visualization can be useful.
* Decide on the machine learning models you want to implement, if applicable. For predicting future traffic trends or user behavior, time series analysis or machine learning algorithms like regression, clustering, or classification might be relevant.
* Ensure that the integration between IBM Cognos and Python is seamless. You might need to export data from one platform to another or use APIs to connect them.

**NEXT STEPS:**

* Document your entire process, including data sources, data cleaning steps, and analysis methodologies.
* Ensure data quality and perform data cleaning as needed.
* Regularly communicate progress and findings to stakeholders.
* Address ethical and privacy considerations, especially if you are dealing with user data.

There are a number of benefits to using machine learning for website traffic analysis. Some of the key benefits include:

* Accuracy: ML algorithms can be very accurate at predicting website traffic trends and patterns. This can help businesses to make better decisions about their marketing and website optimization strategies.
* Scalability: ML algorithms can be scaled to handle large datasets. This is important for businesses with high-traffic websites.
* Automation: ML algorithms can automate many of the tasks involved in website traffic analysis, such as data collection, cleaning, and analysis. This can free up businesses to focus on other tasks.

Insights: ML algorithms can identify insights and patterns in website traffic data that would be difficult or impossible to find manually. This can help businesses to better understand their visitors and how they are interacting with their website.

**STEPS FOR PROBLEM SOLVING:**

Here is a step-by-step problem-solving guide for website traffic analysis:

1. **Identify the problem**: What specific problem are you trying to solve with website traffic analysis? Are you trying to increase website traffic, improve conversion rates, or identify new traffic sources? Once you know what problem you are trying to solve, you can focus your analysis on the relevant data.
2. **Collect data**: The first step to solving any problem is to collect data. There are a number of different tools and services that you can use to collect website traffic data, such as Google Analytics, Semrush, and Ahrefs.
3. **Clean the data**: Once you have collected your data, you need to clean it to ensure that it is accurate and complete. This may involve removing incomplete or inaccurate data, and correcting any errors.
4. **Analyze the data**: Once you have cleaned your data, you can start to analyze it to identify patterns and trends. You can use a variety of different tools and techniques to analyze your data, such as statistical analysis, data visualization, and machine learning.
5. **Identify the root cause of the problem**: Once you have identified patterns and trends in your data, you need to identify the root cause of the problem. This may involve looking at your website content, design, marketing campaigns, or other factors.
6. **Develop a solution**: Once you have identified the root cause of the problem, you need to develop a solution. This may involve making changes to your website content, design, marketing campaigns, or other factors.
7. **Implement the solution**: Once you have developed a solution, you need to implement it. This may involve making changes to your website, creating new marketing campaigns, or other tasks.
8. **Evaluate the results**: Once you have implemented the solution, you need to evaluate the results to see if it has solved the problem. If not, you may need to make further changes.

**PROBLEM SOLVING VIA DESIGN THINKING:**

Design thinking can be used to solve a wide range of problems, including problems related to website traffic analysis. Here is a step-by-step guide to using design thinking to solve problems with website traffic analysis:

1. **Empathize with your users:** The first step in design thinking is to understand the people you are designing for. In the context of website traffic analysis, this means understanding your website visitors. What are their needs and wants? What are their pain points? You can learn more about your users by conducting user research, such as interviews, surveys, and observation.
2. **Define the problem:** Once you have a good understanding of your users, you can start to define the problem you are trying to solve. Be as specific as possible and focus on the user's needs, rather than your own assumptions. For example, instead of saying "I need to increase website traffic," you could say "I need to help users find the information they need on my website more easily."
3. **Ideate:** Once you have defined the problem, it's time to start generating ideas for solutions. This is where you can get creative and come up with as many ideas as possible, no matter how crazy they may seem. Some examples of ideas for solutions to website traffic analysis problems might include:
   * Improving the website's search function
   * Creating more engaging content
   * Promoting the website on social media
   * Running paid advertising campaigns
4. **Prototype**: Once you have a few ideas for solutions, it's time to start prototyping them. This means creating rough models or sketches of your ideas so that you can test them out and get feedback from users. For example, you could create a prototype of a new search function by creating a simple web page with a search bar and some sample results.

Test: The final step in design thinking is to test your prototypes with users and get their feedback. This will help you to refine your ideas and make sure that you are creating solutions that meet the needs of your users

**Code:**

**import pandas as pd**

**import matplotlib.pyplot as plt**

**# Load the website traffic data from a CSV file**

**website\_traffic\_df = pd.read\_csv('websitetrafficanalysis.csv')**

**# Print the first 5 rows of the DataFrame**

**print(website\_traffic\_df.head())**

**# Print the DataFrame information**

**print(website\_traffic\_df.info())**

**# Create a line graph to visualize the website traffic trend over time**

**plt.plot(website\_traffic\_df['date'], website\_traffic\_df['pageviews'])**

**# Set the title and labels of the graph**

**plt.title('Website Traffic Trend')**

**plt.xlabel('Date')**

**plt.ylabel('Pageviews')**

**# Show the graph**

**Plt.show()**

**OUTPUT:**

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

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

<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 object

5 Unique.Visits 2167 non-null object

6 First.Time.Visits 2167 non-null object

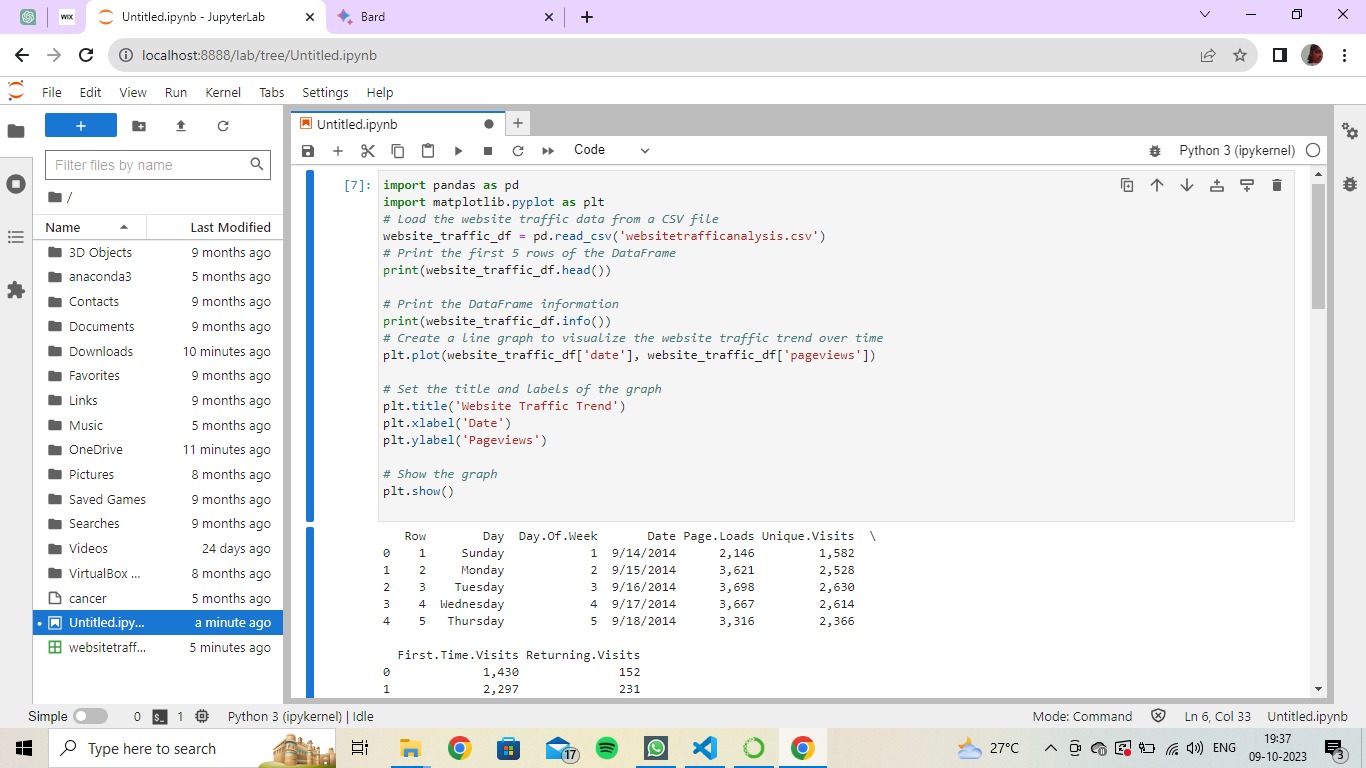
7 Returning.Visits 2167 non-null object

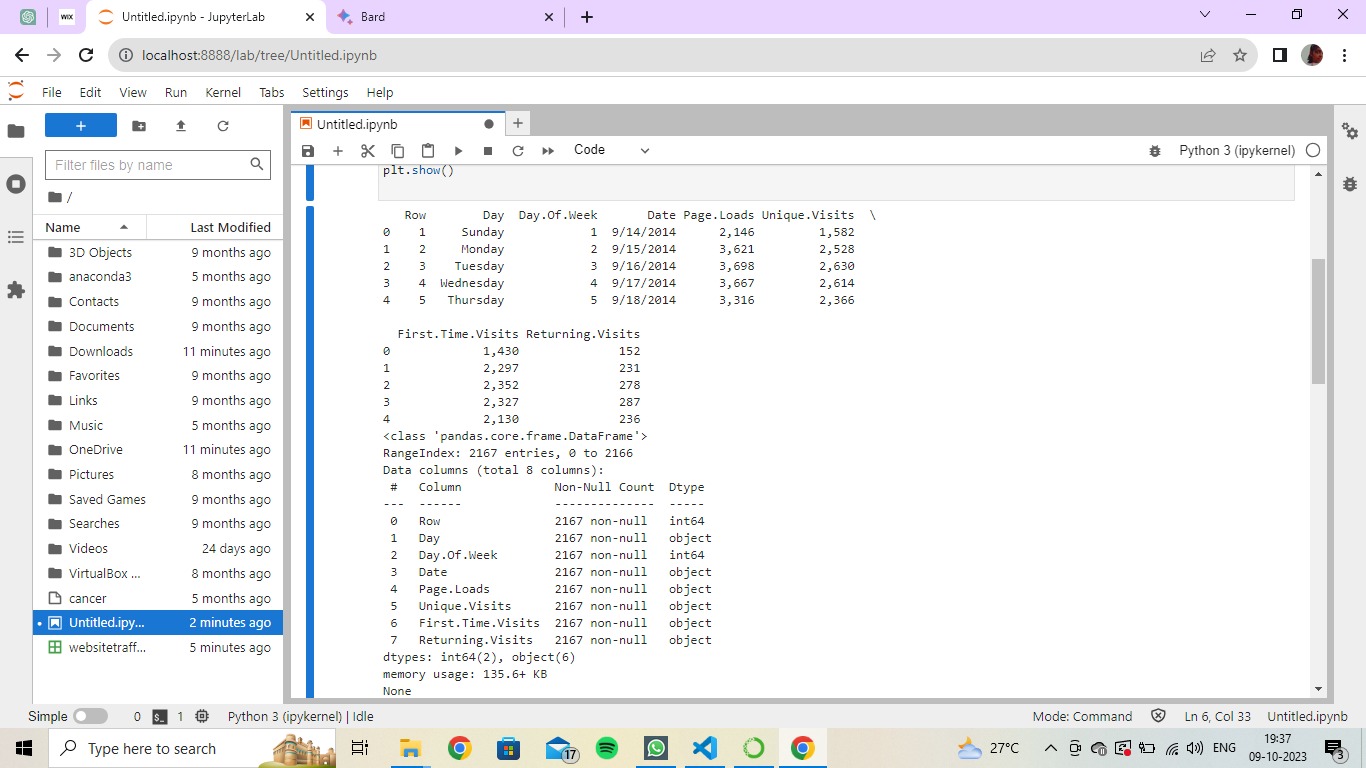
dtypes: int64(2), object(6)

memory usage: 135.6+ KB

None

**OUTPUT IMAGE:**

****

****

**PROBLEM SOLVING USING MACHINE LEARNING**

In website traffic major problem to solve and find the solution using machine learning model:

* One of the major problems with website traffic is that it can be difficult to predict. This can make it difficult for businesses to plan their marketing campaigns and ensure that they have the resources in place to handle increased traffic.
* Another major problem with website traffic is that it can be difficult to understand how users are interacting with a website. This can make it difficult for businesses to improve the user experience and increase conversions.

Machine learning models can be used to solve both of these problems.

**Predicting future traffic trends**

* Machine learning models can be trained on historical traffic data to learn about the factors that influence traffic patterns, such as the time of day, the day of the week, the weather, and special events. Once the model is trained, it can be used to predict future traffic patterns based on these factors.

**Understanding user behavior patterns**

* Machine learning models can also be trained on historical user data to learn about the factors that influence user behavior, such as the user's demographics, interests, and browsing history. Once the model is trained, it can be used to predict future user behavior patterns, such as which pages the user is likely to visit, how long they are likely to stay on each page, and whether they are likely to convert into a customer.

Here is an example of how a machine learning model could be used to solve a major website traffic problem:

**Problem:** A retail website is experiencing a high volume of traffic during peak periods, such as the holiday season. This is causing the website to slow down and crash, resulting in lost sales.

**Solution:** The retail website could use a machine learning model to predict future traffic trends. This information could then be used to scale the website's infrastructure accordingly, ensuring that the website can handle increased traffic without slowing down or crashing.

**ADVANTAGES USING MACHINE LEARNING MODELS:**

Overall, machine learning models can be used to solve a variety of major website traffic problems. By predicting future traffic trends and understanding user behavior patterns, businesses can improve their website performance, optimize their marketing campaigns, and increase conversions.

**MAJOR CAUSE OF WEBSITE TRAFFIC AND SOLVE THE METHODS OF MACHINE LEARING MODELS:**

Website traffic analysis can indeed benefit from machine learning models to solve specific problems.

Here are some common website traffic issues and the machine learning models that can be applied to address them:

1. **Low Traffic:**

* **Machine Learning Model:** Content-Based Recommender System
* **Solution:** Build a recommender system that suggests relevant content to users based on their past behavior and preferences. Algorithms like collaborative filtering or matrix factorization can be used.

1. **High Bounce Rate:**

* **Machine Learning Model:** Clustering Algorithms (e.g., K-Means)
* **Solution:** Cluster users based on their behavior and analyze the behavior of high-bounce-rate clusters separately. Identify common patterns leading to high bounce rates and optimize those pages or content.

1. **Conversion Rate Issues:**

* **Machine Learning Model:** Logistic Regression, Decision Trees
* **Solution:** Use machine learning to analyze user behavior on conversion pages. Predict factors that influence conversions, such as time spent on page, click patterns, or interaction with specific elements, and optimize accordingly.

1. **Mobile Compatibility:**

* **Machine Learning Model:** Image Classification (e.g., Convolutional Neural Networks)
* **Solution:** Train image classification models to automatically detect whether a website element is mobile-friendly. Use these models to flag or fix elements that may cause mobile compatibility issues.

1. **Page Load Speed:**

* **Machine Learning Model**: Regression Models (e.g., Linear Regression)
* **Solution:** Use machine learning to predict page load times based on various factors like page size, number of requests, and server response times. Identify elements that contribute to slow load times and optimize them.

1. **Content Quality:**

* **Machine Learning Model:** Sentiment Analysis, Natural Language Processing (NLP)
* **Solution:** Analyze user-generated content (e.g., comments, reviews) using sentiment analysis to identify areas where content quality can be improved. NLP models can help identify common issues in content.

1. **Technical Errors:**

* **Machine Learning Model:** Anomaly Detection (e.g., Isolation Forest)
* **Solution:** Use anomaly detection to identify unusual patterns in server logs or user interactions that may be indicative of technical errors. Set up alerts for immediate attention.

1. **Security Issues:**

* **Machine Learning Model:** Anomaly Detection, Behavior Analysis
* **Solution:** Train models to detect abnormal user behavior patterns that may indicate security threats, such as SQL injection attempts or unauthorized access.

1. **SEO Problems:**

* **Machine Learning Model:** Regression Models, Text Classification (for content optimization)
* **Solution:** Use regression models to predict SEO rankings based on various factors. Employ NLP and text classification to identify content that needs optimization for specific keywords.

1. **Traffic Sources:**

* **Machine Learning Model**: Clustering, Time Series Analysis
* **Solution**: Cluster user behavior from different traffic sources to understand distinct patterns and optimize your marketing efforts accordingly. Analyze time series data to detect trends and seasonality in traffic sources.

**PREPROCESSING**

**Data Cleaning for Website Traffic Analysis:**

Website traffic data can be complex, containing various issues such as missing values, duplicates, inconsistencies, and outliers. Data cleaning is the initial step in the analysis process, focused on preparing the data for meaningful insights. It involves:

1.**Handling Missing Values**: Identifying and dealing with data points that are incomplete or missing, which can affect the accuracy of analysis.

2.**Removing Duplicates**: Identifying and eliminating duplicate entries in the dataset to avoid skewing results.

3.**Outlier Detection**: Identifying and addressing outliers that can distort patterns and trends in the data.

4**.Data Transformation**: Converting data types, standardizing formats, and making data consistent to ensure accurate analysis.

5.**Data Validation**: Ensuring data quality by validating entries, checking for inaccuracies, and addressing inconsistencies.

The goal of data cleaning is to create a clean, reliable dataset that can serve as a foundation for accurate analysis.

PROGRAM:

import pandas as pd

import matplotlib.pyplot as plt

import plotly.express as px

import plotly.graph\_objects as go

from statsmodels.tsa.seasonal import seasonal\_decompose

from statsmodels.graphics.tsaplots import plot\_pacf

from statsmodels.tsa.arima\_model import ARIMA

import statsmodels.api as sm

#read the csv file

data = pd.read\_csv("daily-website-visitors.csv", \

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

print(data.head())

print(data.info())

#1.handle duplicates

data.dropna(inplace=True)

# 2. Removing Duplicates

data.drop\_duplicates(inplace=True)

#3.conversion of data types

data["Date"] = pd.to\_datetime(data["Date"],

format="%m/%d/%Y")

print(data.info())

#4.checking missing values

print(data.isnull().sum())

#5.Drop rows with missing values

data=data.dropna()

#6.remove the duplicate values

data=data.drop\_duplicates()

**OUTPUT:**

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

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

<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 object

5 Unique.Visits 2167 non-null object

6 First.Time.Visits 2167 non-null object

7 Returning.Visits 2167 non-null object

dtypes: int64(2), object(6)

memory usage: 135.6+ KB

None

<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 datetime64[ns]

4 Page.Loads 2167 non-null object

5 Unique.Visits 2167 non-null object

6 First.Time.Visits 2167 non-null object

7 Returning.Visits 2167 non-null object

dtypes: datetime64[ns](1), int64(2), object(5)

memory usage: 135.6+ KB

None

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

**Data Visualization for Website Traffic Analysis:**

Data visualization is the art of representing data graphically to make it more accessible and understandable. In website traffic analysis, data visualization serves several critical purposes:

**1.Pattern Identification:** Visualizations help identify patterns and trends in website traffic data. Line charts, for instance, can reveal traffic fluctuations over time.

**2.User Behavior Analysis:** Visualizations like heatmaps and click maps provide insights into how users interact with a website, indicating which elements receive the most attention and which paths they follow.

**3.Source and Channel Analysis:** Visualizations can help website owners understand where their traffic comes from, which sources and channels are most effective, and where they should allocate their marketing resources.

**4.Conversion Funnel Analysis:** By creating conversion funnels, website owners can track the user journey, identify drop-off points, and optimize the conversion process.

**5.Performance Monitoring:** Visualizing page load times, error rates, and other performance metrics allows for the identification of issues affecting the user experience.

Data visualization not only helps in understanding the current state of the website but also assists in making data-driven decisions to improve user experience and achieve website objectives.

In conclusion, data cleaning and data visualization are integral to website traffic analysis. They enable website owners and businesses to make sense of complex data, uncover insights, and take informed actions to optimize their websites, enhance user experiences, and achieve their goals. These phases are essential for the success of any data-driven online presence.

**PROGRAM:**

import pandas as pd

import matplotlib.pyplot as plt

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

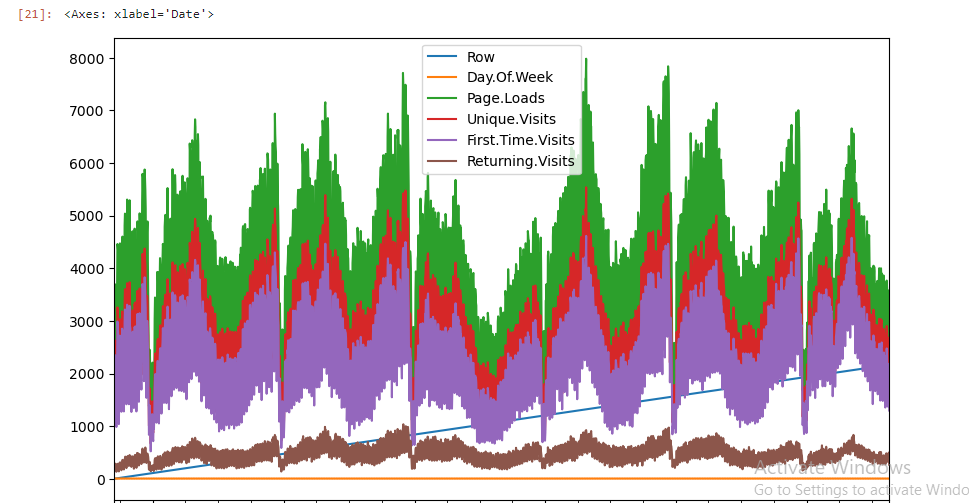
index\_col = 'Date', thousands = ',', parse\_dates=True)

df.head()

#visualize the figure

df.plot(figsize=(10,6))

**OUTPUT:**

****

**CREATING VISULATION USING IBM COGNOS INTEGRATION PYTHON CODE FOR ADVANCED ANALYSIS**

Here is how you can continue building your analysis by creating visualizations using IBM Cognos and integrating Python code for advanced analysis:

Step 1: Create a data module in IBM Cognos

A data module in IBM Cognos is a logical representation of a data source. It provides a way to organize and manage your data in a way that is easy to use for reporting and analysis.

To create a data module, you will need to connect to your data source and define the dimensions and measures that you want to include. Once you have created the data module, you can publish it to IBM Cognos so that it can be used in dashboards and reports.

Step 2: Create a dashboard in IBM Cognos

A dashboard in IBM Cognos is a visual representation of data that provides you with insights into your business. You can create dashboards to track key performance indicators (KPIs), monitor trends, and identify opportunities.

To create a dashboard, you will need to add visualizations to the dashboard canvas. You can add visualizations from the IBM Cognos library or create your own custom visualizations using Python.

Step 3: Integrate Python code into your IBM Cognos dashboard

You can integrate Python code into your IBM Cognos dashboard to create custom visualizations and perform more complex analyses on your data. To do this, you will need to use the IBM Cognos Analytics Python API.

The IBM Cognos Analytics Python API provides a set of Python functions that you can use to interact with IBM Cognos Analytics. You can use the API to read and write data, create and modify visualizations, and schedule reports.

Step 4: Deploy your dashboard to IBM Cognos

Once you have created your dashboard, you can deploy it to IBM Cognos so that other users can view it. To do this, you will need to publish the dashboard to a Cognos content store.

Step 5: Use your dashboard to gain insights

Once your dashboard is deployed, you can use it to gain insights into your data. You can explore the different visualizations and interact with the dashboard to get a deeper understanding of your data.

**Here are some examples of how you can use IBM Cognos and Python to perform advanced analysis on your website traffic data:**

* Time series analysis

You can use Python to perform time series analysis on your website traffic data to identify trends and seasonality. For example, you can use Python to create a line chart that shows the total number of visitors to your website over time. You can also use Python to calculate the average number of visitors to your website per day, week, or month.

* User segmentation

You can use Python to segment your users based on their behavior and demographics. For example, you can segment your users based on the pages they visit, the sources they come from, and the devices they use. You can also segment your users based on their age, gender, location, and interests.

Once you have segmented your users, you can use IBM Cognos to create dashboards and reports that show how each segment is interacting with your website. For example, you can create a dashboard that shows the top pages visited by each segment, the average time spent on each page, and the bounce rate for each page.

* Machine learning-based predictions

You can use machine learning to predict future website traffic trends and user behavior. For example, you can train a machine learning model to predict the number of visitors to your website on a given day based on historical data, such as the day of the week, the time of year, and the weather forecast.

You can also use machine learning to predict which users are most likely to churn. This information can be used to target these users with retention campaigns.

**MACHINE LEARNING MODELS:**

To use machine learning to predict future website traffic trends and user behavior using IBM Cognos and Python, you can follow these steps:

1. Prepare your data

The first step is to prepare your website traffic data for machine learning. This involves cleaning the data, removing any outliers, and encoding categorical variables. You can use Python to do this.

2. Choose a machine learning algorithm

There are many different machine learning algorithms that you can use to predict website traffic. Some popular algorithms include:

* Linear regression
* Logistic regression
* Support vector machines
* Decision trees
* Random forests
* Neural networks

You can choose an algorithm based on the specific problem that you are trying to solve. For example, if you are trying to predict the total number of visitors to your website on a given day, you could use a linear regression algorithm. If you are trying to predict whether a user is likely to churn, you could use a logistic regression algorithm.

3. Train the machine learning model

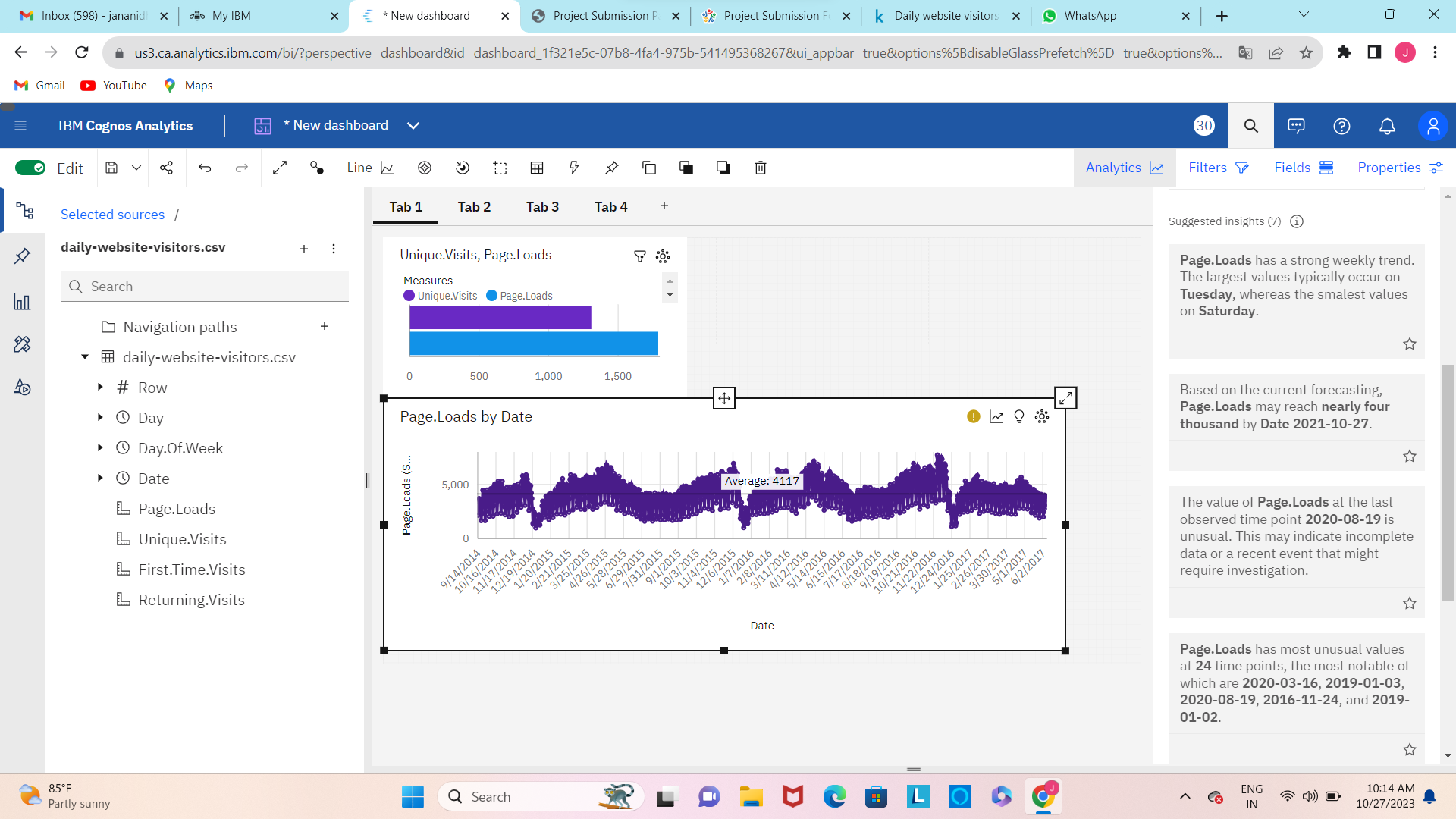
Once you have chosen a machine learning algorithm, you need to train the model on your website traffic data. You can use Python to do this.

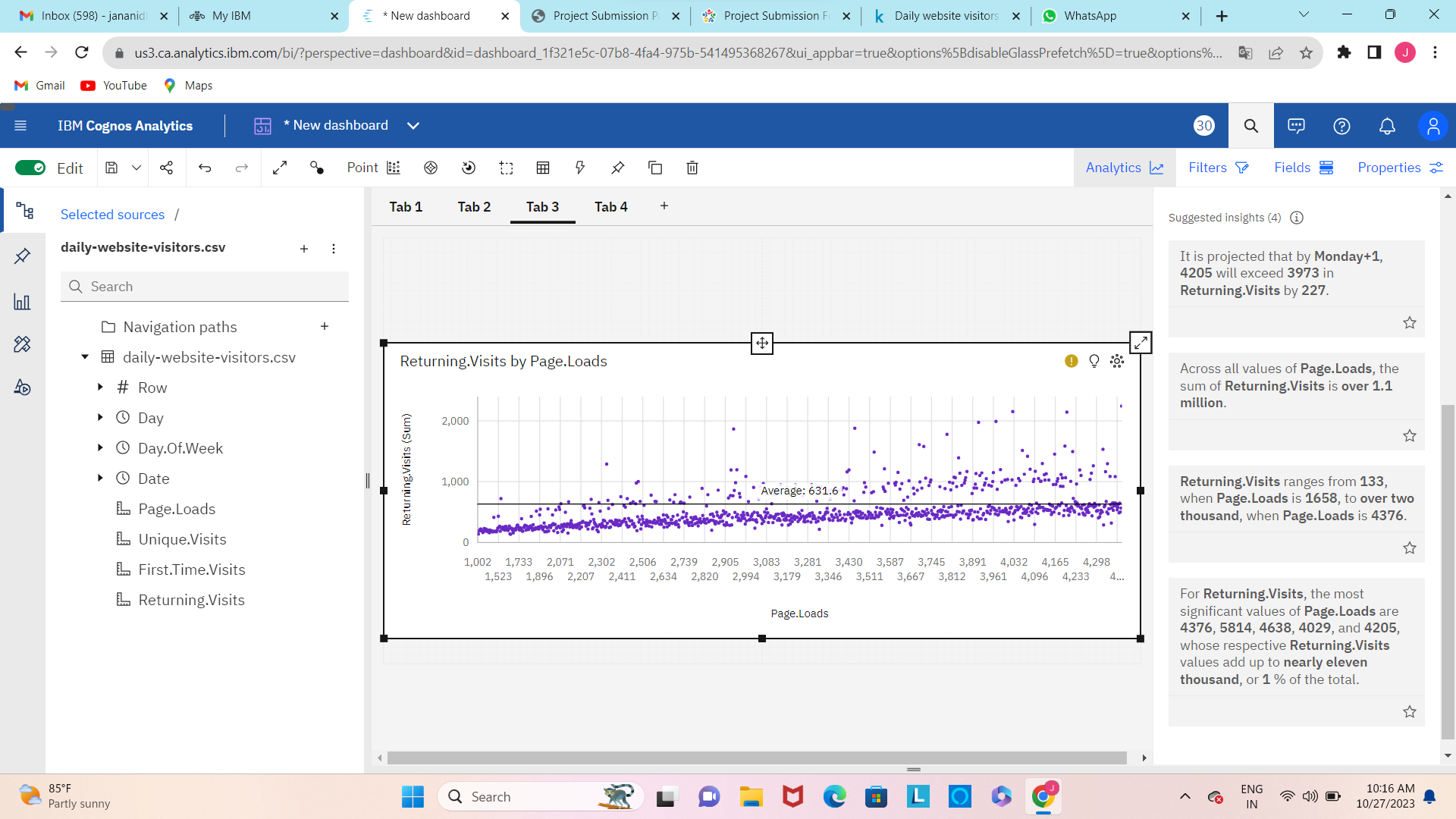
4. Evaluate the machine learning model

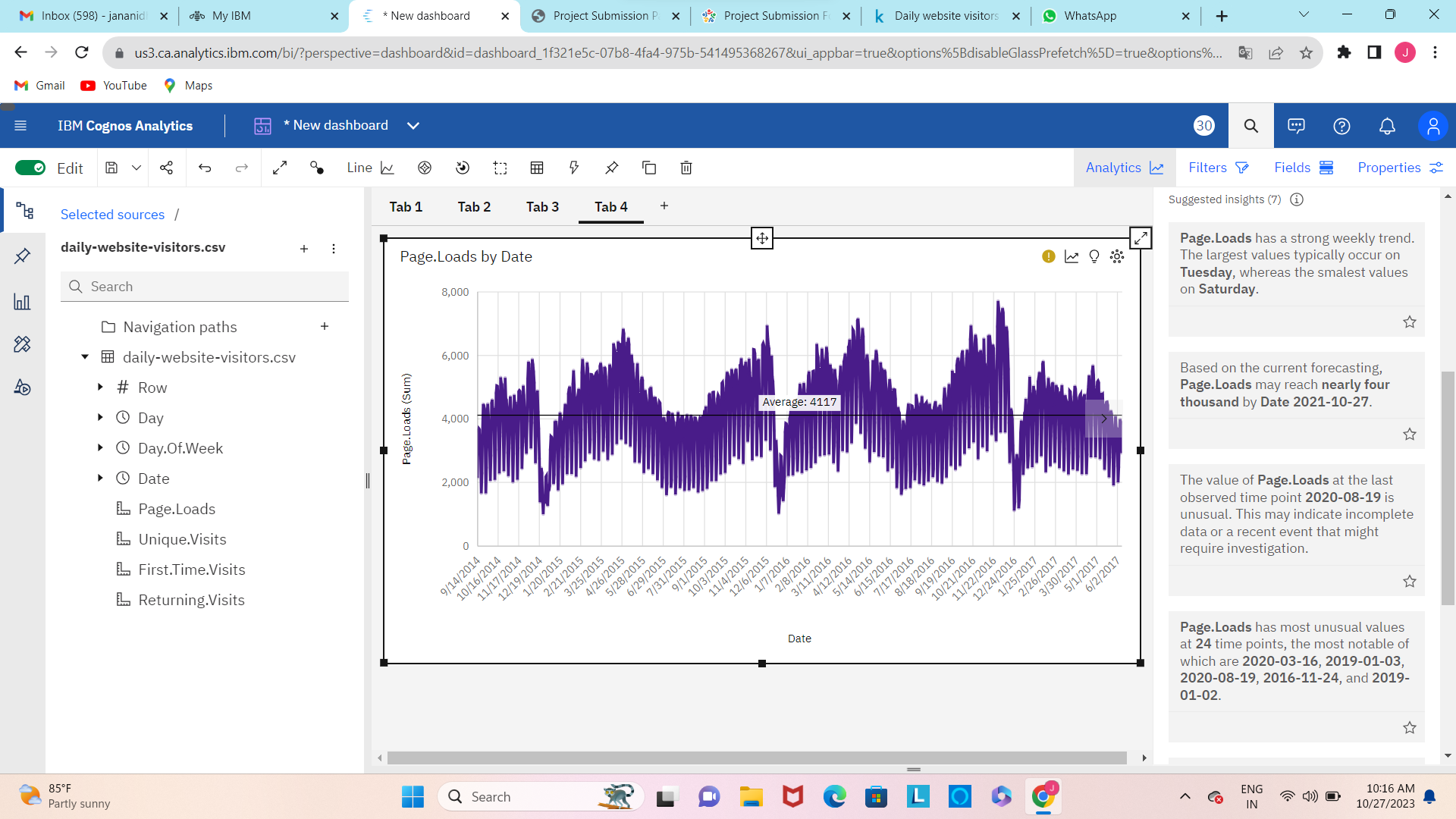
Once the machine learning model is trained, you need to evaluate its performance on a held-out test set. This will give you an idea of how well the model will perform on new data.

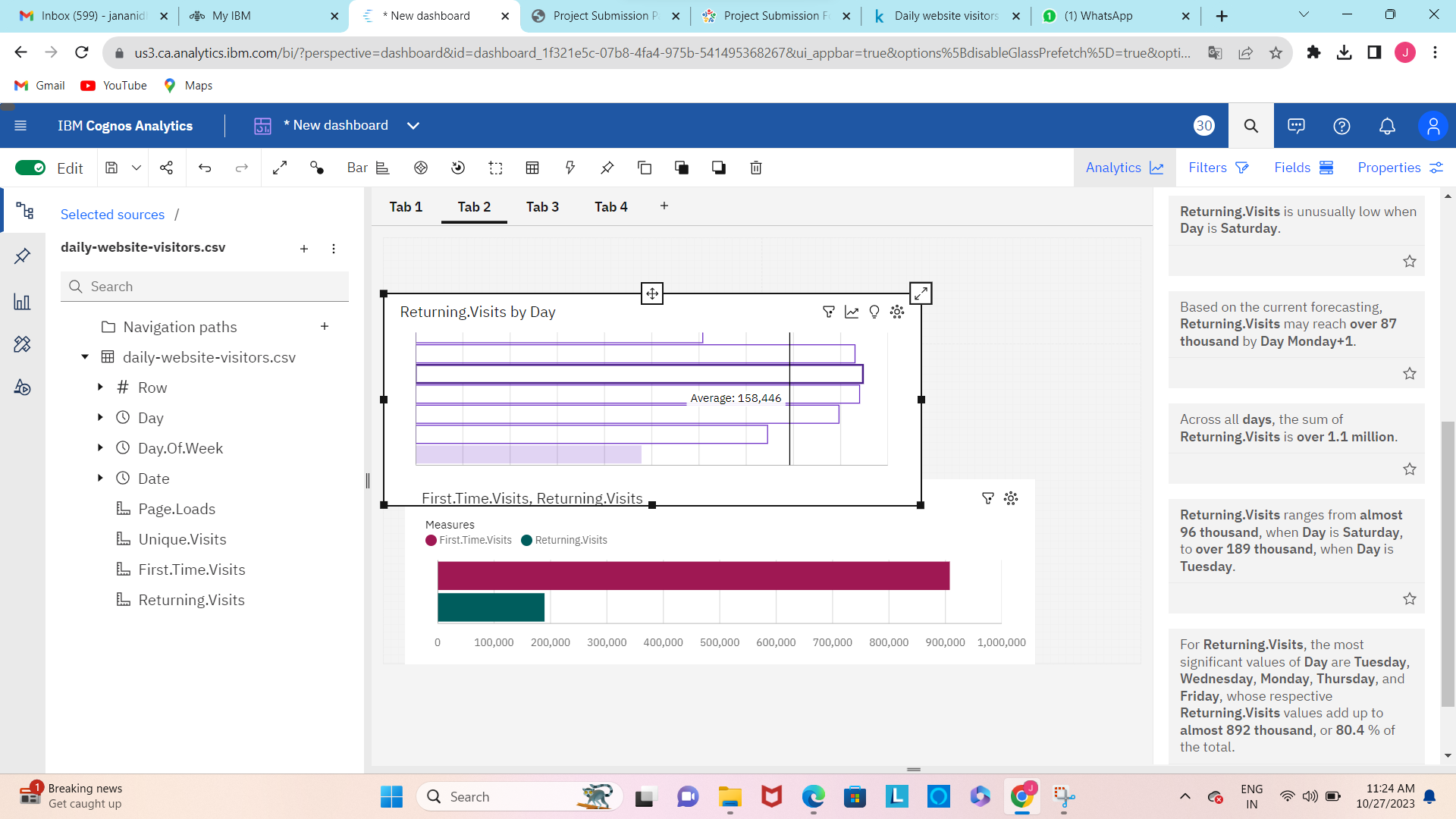
5. Deploy the machine learning model

Once you are satisfied with the performance of the machine learning model, you can deploy it to production. You can use IBM Cognos to do this**.**

**VISUALIZATION USING IBM COGNOS**

****

****



**MACHINE LEARNING PREDICTION USING LINEAR REGRESSION**

**from sklearn import neighbors**

**import matplotlib.pyplot as plt**

**import pandas as pd**

**df = pd.read\_csv("daily-website-visitors.csv", thousands=',')**

**df = df.drop(['Row'], axis=1)**

**df.head()**

**OUTPUT:**

**Day Day.Of.Week Date Page.Loads Unique.Visits First.Time.Visits Returning.Visits**

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

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

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

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

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

**Linear / Non-Linear Relationship**

**pd.plotting.scatter\_matrix(df, figsize=(20,10))**

**OUTPUT:**

array([[<AxesSubplot:xlabel='Day.Of.Week', ylabel='Day.Of.Week'>,

<AxesSubplot:xlabel='Page.Loads', ylabel='Day.Of.Week'>,

<AxesSubplot:xlabel='Unique.Visits', ylabel='Day.Of.Week'>,

<AxesSubplot:xlabel='First.Time.Visits', ylabel='Day.Of.Week'>,

<AxesSubplot:xlabel='Returning.Visits', ylabel='Day.Of.Week'>],

[<AxesSubplot:xlabel='Day.Of.Week', ylabel='Page.Loads'>,

<AxesSubplot:xlabel='Page.Loads', ylabel='Page.Loads'>,

<AxesSubplot:xlabel='Unique.Visits', ylabel='Page.Loads'>,

<AxesSubplot:xlabel='First.Time.Visits', ylabel='Page.Loads'>,

<AxesSubplot:xlabel='Returning.Visits', ylabel='Page.Loads'>],

[<AxesSubplot:xlabel='Day.Of.Week', ylabel='Unique.Visits'>,

<AxesSubplot:xlabel='Page.Loads', ylabel='Unique.Visits'>,

<AxesSubplot:xlabel='Unique.Visits', ylabel='Unique.Visits'>,

<AxesSubplot:xlabel='First.Time.Visits', ylabel='Unique.Visits'>,

<AxesSubplot:xlabel='Returning.Visits', ylabel='Unique.Visits'>],

[<AxesSubplot:xlabel='Day.Of.Week', ylabel='First.Time.Visits'>,

<AxesSubplot:xlabel='Page.Loads', ylabel='First.Time.Visits'>,

<AxesSubplot:xlabel='Unique.Visits', ylabel='First.Time.Visits'>,

<AxesSubplot:xlabel='First.Time.Visits', ylabel='First.Time.Visits'>,

<AxesSubplot:xlabel='Returning.Visits', ylabel='First.Time.Visits'>],

[<AxesSubplot:xlabel='Day.Of.Week', ylabel='Returning.Visits'>,

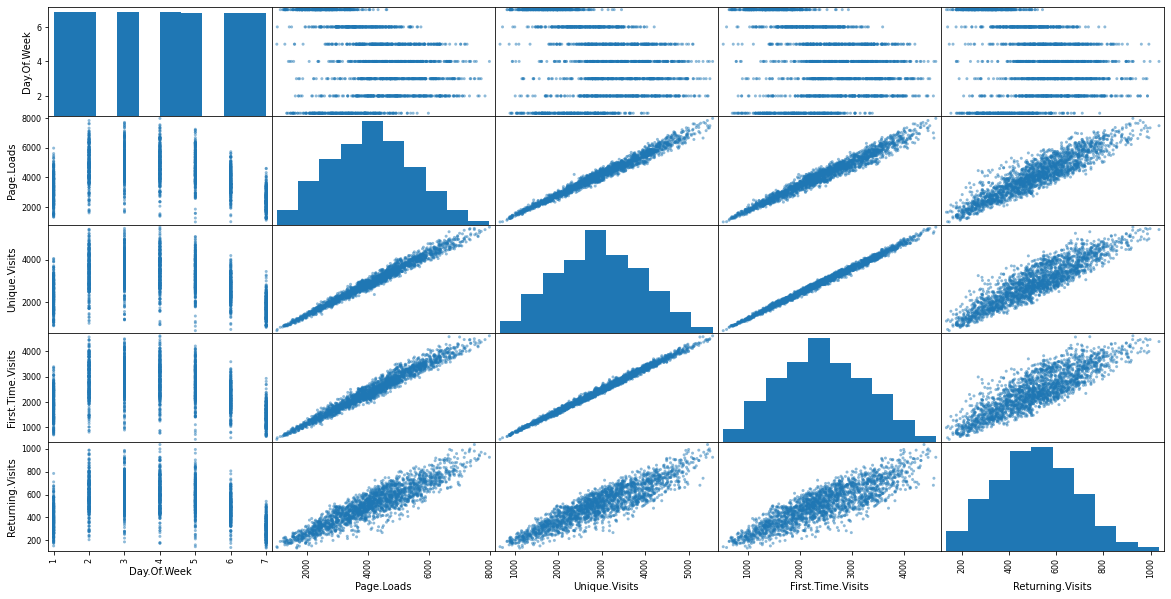
<AxesSubplot:xlabel='Page.Loads', ylabel='Returning.Visits'>,

<AxesSubplot:xlabel='Unique.Visits', ylabel='Returning.Visits'>,

<AxesSubplot:xlabel='First.Time.Visits', ylabel='Returning.Visits'>,

<AxesSubplot:xlabel='Returning.Visits', ylabel='Returning.Visits'>]],

dtype=object)



**import seaborn as sns**

**corr = df.corr()**

**sns.heatmap(corr,**

**xticklabels=corr.columns,**

**yticklabels=corr.columns, cmap="Blues", annot=True)**

**OUTPUT:**

<AxesSubplot:>



**Data Splits**

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

**Y = df['Unique.Visits']**

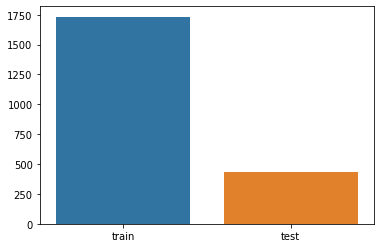
**X = df[['Page.Loads']]**

**x\_train, x\_test,y\_train,y\_test = train\_test\_split(X,Y,test\_size =0.2)**

**sns.barplot(x=['train','test'], y=[y\_train.count(), y\_test.count()])**

**OUTPUT:**

**<AxesSubplot:>**

****

# Linear Regression

**from sklearn.linear\_model import LinearRegression**

**clf = LinearRegression()**

**clf.fit(x\_train,y\_train)**

**OUTPUT:**

**LinearRegression()**

**preds = clf.predict(x\_test)**

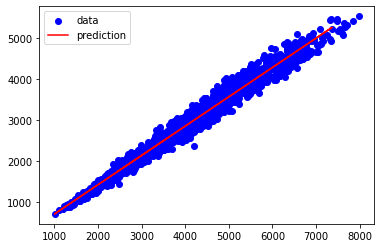
**plt.scatter(x\_train, y\_train, color="blue", label="data")**

**plt.plot(x\_test, preds, color="red", label="prediction")**

**plt.legend()**

**OUTPUT:**

<matplotlib.legend.Legend at 0x7f76c2698410>



**import sklearn.metrics as metrics**

**import numpy as np**

**mae = metrics.mean\_absolute\_error(y\_test, preds)**

**mse = metrics.mean\_squared\_error(y\_test, preds)**

**rmse = np.sqrt(mse) # or mse\*\*(0.5)**

**r2 = metrics.r2\_score(y\_test,preds)**

**print("Results of Linear Regression:")**

**print("MAE:",mae)**

**print("MSE:", mse)**

**print("RMSE:", rmse)**

**print("R-Squared:", r2)**

**OUTPUT:**

Results of Linear Regression:

MAE: 110.70796662588398

MSE: 20212.04529554275

RMSE: 142.169072922147

R-Squared: 0.978179

**CODE:**

**import pandas as pd**

**import matplotlib.pyplot as plt**

**import plotly.express as px**

**import plotly.graph\_objects as go**

**from statsmodels.tsa.seasonal import seasonal\_decompose**

**from statsmodels.graphics.tsaplots import plot\_pacf**

**from statsmodels.tsa.arima\_model import ARIMA**

**import statsmodels.api as sm**

**data = pd.read\_csv("Website.csv")**

**print(data.head())**

**Date Views**

**0 01/06/2021 7831**

**1 02/06/2021 7798**

**2 03/06/2021 7401**

**3 04/06/2021 7054**

**4 05/06/2021 7973**

**data["Date"] = pd.to\_datetime(data["Date"],**

**format="%d/%m/%Y")**

**print(data.info())**

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

**RangeIndex: 391 entries, 0 to 390**

**Data columns (total 2 columns):**

**# Column Non-Null Count Dtype**

**--- ------ -------------- -----**

**0 Date 391 non-null datetime64[ns]**

**1 Views 391 non-null int64**

**dtypes: datetime64[ns](1), int64(1)**

**memory usage: 6.2 KB**

**None**

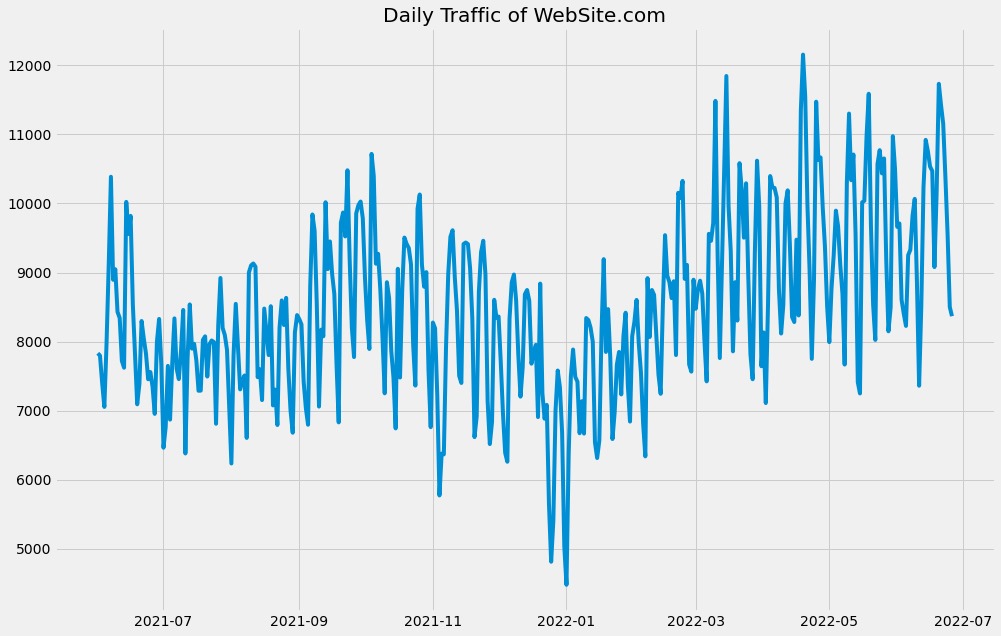
**plt.style.use('fivethirtyeight')**

**plt.figure(figsize=(15, 10))**

**plt.plot(data["Date"], data["Views"])**

**plt.title("Daily Traffic of WebSite.com")**

**plt.show()**

**OUTPUT:**

**from statsmodels.tsa.seasonal import seasonal\_decompose**

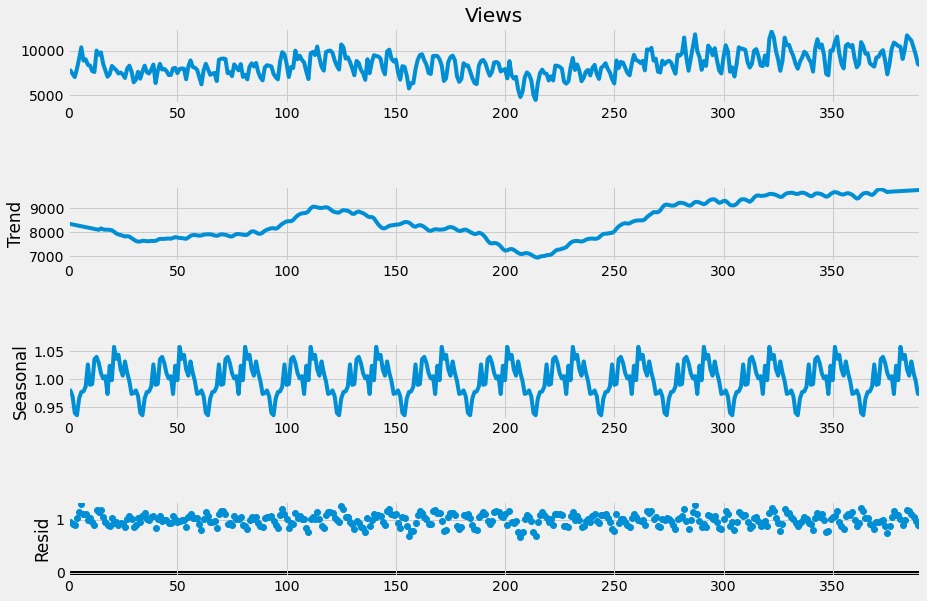
**result = seasonal\_decompose(data["Views"], model='multiplicative',extrapolate\_trend='freq', period=30)**

**fig = plt.figure()**

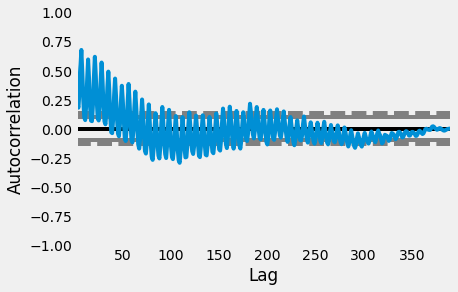
**fig = result.plot()**

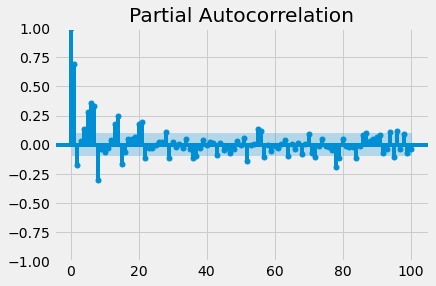
**fig.set\_size\_inches(15, 10)**

**OUTPUT:**

****

**plot\_pacf(data["Views"], lags = 100**

****

****

**p, d, q = 5, 1, 2**

**model=sm.tsa.statespace.SARIMAX(data['Views'],**

**order=(p, d, q),**

**seasonal\_order=(p, d, q, 12))**

**model=model.fit()**

**print(model.summary())**

**OUTPUT:**

**SARIMAX Results**

**==========================================================================================**

**Dep. Variable: Views No. Observations: 391**

**Model: SARIMAX(5, 1, 2)x(5, 1, 2, 12) Log Likelihood -3099.467**

**Date: Fri, 31 Mar 2023 AIC 6228.933**

**Time: 23:53:54 BIC 6287.957**

**Sample: 0 HQIC 6252.359**

**- 391**

**Covariance Type: opg**

**==============================================================================**

**coef std err z P>|z| [0.025 0.975]**

**------------------------------------------------------------------------------**

**ar.L1 0.7785 0.134 5.790 0.000 0.515 1.042**

**ar.L2 -0.8003 0.134 -5.951 0.000 -1.064 -0.537**

**ar.L3 -0.1439 0.169 -0.850 0.395 -0.475 0.188**

**ar.L4 -0.1830 0.151 -1.212 0.226 -0.479 0.113**

**ar.L5 -0.1593 0.139 -1.147 0.251 -0.432 0.113**

**ma.L1 -1.1799 0.096 -12.302 0.000 -1.368 -0.992**

**ma.L2 0.8830 0.079 11.164 0.000 0.728 1.038**

**ar.S.L12 -0.2655 4.706 -0.056 0.955 -9.489 8.958**

**ar.S.L24 0.0402 0.798 0.050 0.960 -1.524 1.605**

**ar.S.L36 -0.1866 0.244 -0.764 0.445 -0.665 0.292**

**ar.S.L48 -0.2174 0.971 -0.224 0.823 -2.121 1.686**

**ar.S.L60 0.0108 1.013 0.011 0.991 -1.974 1.996**

**ma.S.L12 -0.6849 4.709 -0.145 0.884 -9.914 8.544**

**ma.S.L24 -0.1025 3.710 -0.028 0.978 -7.374 7.169**

**sigma2 1.257e+06 1.59e+05 7.914 0.000 9.46e+05 1.57e+06**

**===================================================================================**

**Ljung-Box (L1) (Q): 0.00 Jarque-Bera (JB): 1.31**

**Prob(Q): 1.00 Prob(JB): 0.52**

**Heteroskedasticity (H): 1.03 Skew: 0.14**

**Prob(H) (two-sided): 0.86 Kurtosis: 3.01**

**predictions=model.predict(len(data),len(data)+50)**

**print(predictions)**

**OUTPUT:**

**391 9877.390741**

**392 10795.539039**

**393 10760.027140**

**394 9859.469163**

**395 8768.340243**

**396 8220.386498**

**397 8934.781852**

**398 9691.221147**

**399 10276.023043**

**400 10623.217947**

**401 9846.080910**

**402 9358.619001**

**403 9044.210850**

**404 9088.171287**

**405 10544.303967**

**406 11004.539945**

**407 10889.086614**

**408 10073.663141**

**409 9443.450824**

**410 8634.197490**

**411 9188.860108**

**412 10400.395571**

**413 10595.944638**

**414 10784.074228**

**415 10258.259946**

**416 9450.516229**

**417 9052.232545**

**418 9181.449413**

**419 9900.238884**

**420 10226.056903**

**421 10711.820877**

**422 9893.199616**

**423 9542.918529**

**424 9075.749724**

**425 8834.478515**

**426 10144.241610**

**427 10840.511772**

**428 10898.827148**

**429 10402.301196**

**430 9448.255097**

**431 8701.545390**

**432 8732.720077**

**433 10068.060242**

**434 10505.394196**

**435 10833.152585**

**436 10476.852952**

**437 9332.115091**

**438 9183.498204**

**439 9370.223284**

**440 10305.766357**

**441 11165.818361**

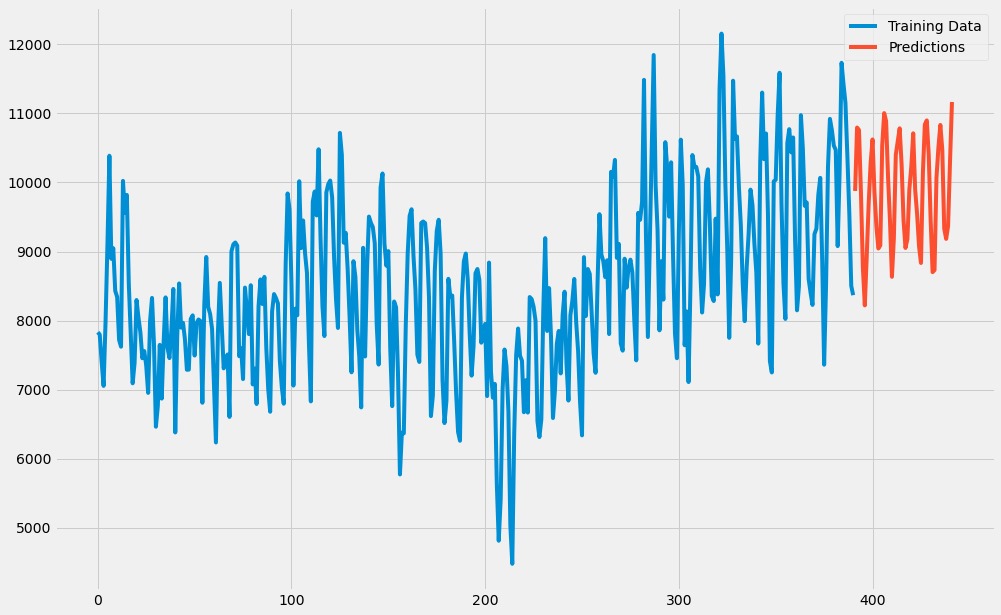
**Name: predicted\_mean, dtype: float64**

**data["Views"].plot(legend=True, label="Training Data",**

**figsize=(15, 10))**

**predictions.plot(legend=True, label="Predictions")1**

**FINAL OUTPUT:**

****