T.Y.B.Sc. (Computer Science)
Semester – VI

WORKBOOK FOR

CS – 368

SECTION II

Practical Course on DATA ANALYTICS

Coordinator and Editor

Dr. Poonam Ponde

Nowrosjee Wadia College, Pune
Member, BOS (Computer Science),
Savitribai Phule Pune University.

Assignments Prepared by

Dr. Poonam Ponde
Nowrosjee Wadia College, Pune

Dr. Harsha Patil
Ashoka Center For Business & Computer Studies, Nashik

Prof. Amit Mogal
MVP Samaj's CMCS College, Nashik

Prof. Kamakshi Goyal
Nowrosjee Wadia College, Pune

CS 368 (SECTION II)
PRACTICAL COURSE ON DATA ANALYTICS

Assignment Completion Sheet

Name of Student: ____

Roll Number: __________ **Division: _____________**

Sr. No.	**Assignment Title**	**Marks obtained**	**Signature of Instructor**
1	Linear and Logistic Regression		
2	Frequent itemset and Association rule mining		
3	Text and Social Media Analytics		

Total Marks :__________/ 15

Converted Marks :_______/ ______

Signature of Incharge

Date:

Internal Examiner **External Examiner**

Assignment 1: Linear and Logistic Regression

No. of slots: 02

Objectives

- Apply appropriate analytic techniques and tools to analyze data, create models, and identify insights that can lead to actionable results.
- Apply modeling and data analysis linear and logistic regression techniques to the solution of real world business problems

Reading

You should read the following topics before starting this exercise

- The modeling process, Engineering features and selecting a model, Training the model, Validating the model, Predicting new observations
- Types of machine learning
- Regression models
- Concept of classification, clustering and reinforcement learning

Ready Reference and Self Activity

Machine Learning -

- Machine Learning is said as a subset of **artificial intelligence** that is mainly concerned with the development of algorithms which allow a computer to learn from the data and past experiences on their own. The term machine learning was first introduced by **Arthur Samuel** in **1959**.
- Definition: Machine learning enables a machine to automatically learn from data, improve performance from experiences, and predict things without being explicitly programmed.
- With the help of sample historical data, which is known as **training data**, machine learning algorithms build a **mathematical model** that helps in making predictions or decisions without being explicitly programmed. Machine learning brings computer science and statistics together for creating predictive models.

Machine learning can be classified into three types:

1. Supervised learning 2. Unsupervised learning 3. Reinforcement learning

1) Supervised Learning - Supervised learning is a type of machine learning method in which we provide sample labeled data to the machine learning system in order to train it, and on that basis, it predicts the output.

2) Unsupervised Learning - Unsupervised learning is a learning method in which a machine learns without any supervision.

3) Reinforcement Learning - Reinforcement learning is a feedback-based learning method, in which a learning agent gets a reward for each right action and gets a penalty for each wrong action. The agent learns automatically with these feedbacks and improves its performance. In reinforcement learning, the agent interacts with the environment and explores it. The goal of an agent is to get the most reward points, and hence, it improves its performance.

Regression Analysis-

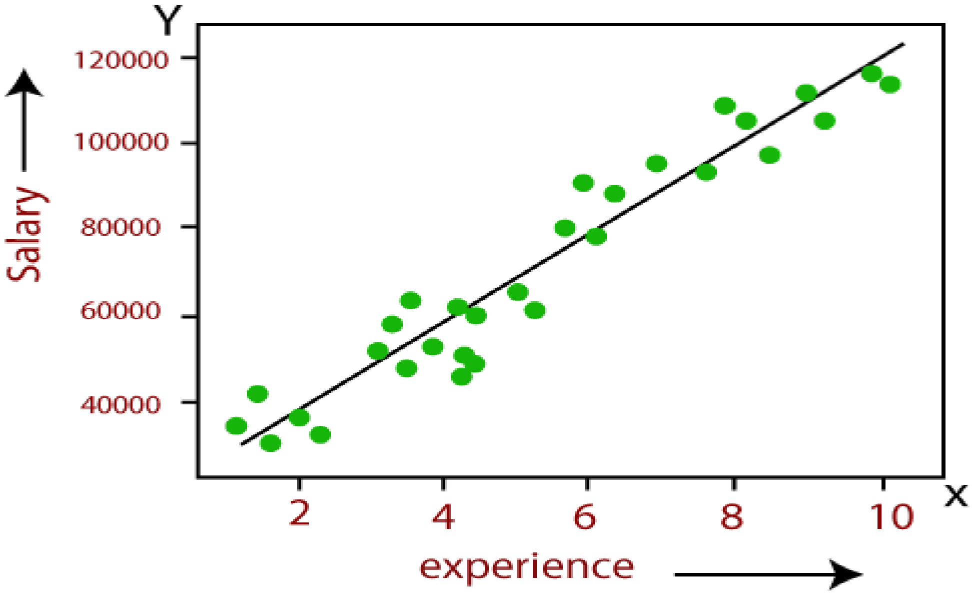
- Regression analysis is a statistical method to model the relationship between a dependent (target) and independent (predictor) variables with one or more independent variables.
- Regression analysis helps us to understand how the value of the dependent variable is changing corresponding to an independent variable when other independent variables are held fixed.
- Regression is a supervised learning technique which helps in finding the correlation between variables and enables us to predict the continuous output variable based on the one or more predictor variables.
- It is mainly used for **prediction, forecasting, time series modeling, and determining the causal-effect relationship between variables**.
- In Regression, we plot a graph between the variables which best fits the given datapoints, using this plot, the machine learning model can make predictions about the data.
- ***"Regression shows a line or curve that passes through all the datapoints on target-predictor graph in such a way that the vertical distance between the datapoints and the regression line is minimum."***
- The distance between datapoints and line tells whether a model has captured a strong relationship or not.

Types of Regression

There are various types of regressions which are used in data science and machine learning. Each type has its own importance on different scenarios, but at the core, all the regression methods analyze the effect of the independent variable on dependent variables. Here in this assignment we will learn Linear Regression and Logistic Regression in detail.

Linear Regression:

- Linear regression is a statistical regression method which is used for predictive analysis.
- It is one of the very simple and easy algorithms which works on regression and shows the relationship between the continuous variables.
- It is used for solving the regression problem in machine learning.
- Linear regression shows the linear relationship between the independent variable (X-axis) and the dependent variable (Y-axis), hence called linear regression.
- If there is only one input variable (x), then such linear regression is called **simple linear regression**. And if there is more than one input variable, then such linear regression is called **multiple linear regression**.
- The relationship between variables in the linear regression model can be explained using the below image. Here we are predicting the salary of an employee on the basis of **the year of experience**.



- Below is the mathematical equation for Linear regression:
    - Y= aX+b
        - **Here,**
            - **Y = dependent variables (target variables),**
            - **X= Independent variables (predictor variables),**
            - **a and b are the linear coefficients**

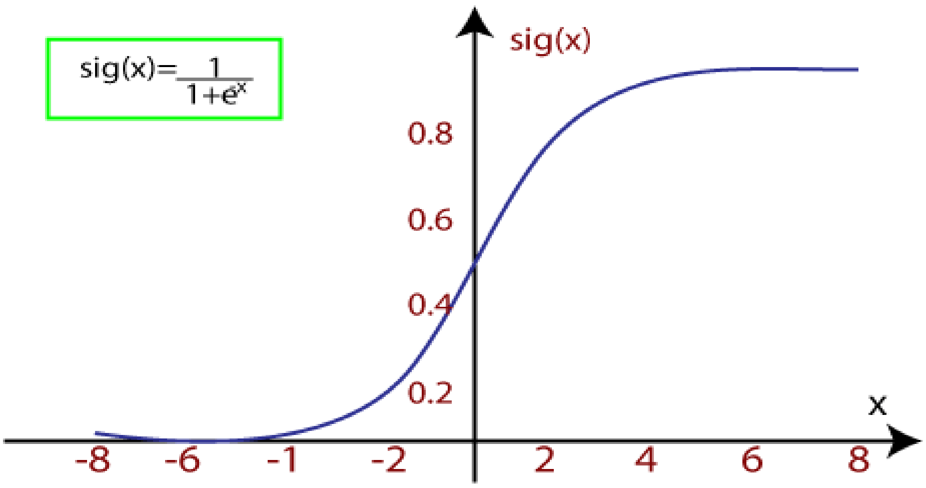
Logistic Regression:

- Logistic regression is another supervised learning algorithm which is used to solve the classification problems. In **classification problems**, we have dependent variables in a binary or discrete format such as 0 or 1.
- Logistic regression algorithm works with the categorical variable such as 0 or 1, Yes or No, True or False, Spam or not spam, etc.
- It is a predictive analysis algorithm which works on the concept of probability.
- Logistic regression is a type of regression, but it is different from the linear regression algorithm in the term how they are used.
- Logistic regression uses **sigmoid function** or logistic function which is a complex cost function. This sigmoid function is used to model the data in logistic regression. The function can be represented as:

$$f(x)= \frac{1}{1+e^{-x}}$$

- f(x)= Output between the 0 and 1 value.
- x= input to the function
- e= base of natural logarithm.

When we provide the input values (data) to the function, it gives the S-curve as follows:



- It uses the concept of threshold levels, values above the threshold level are rounded up to 1, and values below the threshold level are rounded up to 0.

There are three types of logistic regression:

- **Binary**- In this type, the dependent/target variable has two distinct values, either 0 or 1, malignant or benign, passed or failed, admitted or not admitted.
- **Multinominal** - Multinomial Logistic Regression deals with cases when the target or independent variable has three or more possible values. (cats, dogs, lions)
- **Ordinal** – It is used in cases when the target variable is of ordinal nature. In this type, the categories are ordered in a meaningful manner and each category has quantitative significance. (low, medium, high)

Self-Activity

Building a linear regression model in Python

1. Import libraries /packages
2. Reading and understanding the data(eventually do appropriate transformations)
3. Visualizing the data
4. Splitting our Data set in Dependent and Independent variables.
5. Performing simple linear regression (create linear regression model)
6. Residual analysis(Check the results of model fitting to know whether the model is satisfactory)
7. Predictions on the test set (apply the model)

Sample Example -

Goal is to build a linear regression model in Python

For this example we are using car data. Following is the link to download the rquired dataset

https://www.kaggle.com/CooperUnion/cardataset

1. Import libraries /packages

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score,mean_squared_error
%matplotlib inline
```

2. Reading and understanding the data(eventually do appropriate transformations)

```
df = pd.read_csv('C:/TYBSC/car_data.csv') # Importing the data set
df.sample(5) #previewing dataset randomly
```

Then we import the car dataset. And print 5 sample dataset values. At first, we imported our necessary libraries.

```
print(df.shape) # view the dataset shape
print(df['Make'].value_counts()) # viewing Car companies with their cars number
```

Here we print the shape of the dataset and print the different car companies with their total cars.

```
new_df = df[df['Make']=='Volkswagen'] # in this new data set we only take 'Volkswagen' Cars
print(new_df.shape) # Viewing the new dataset shape
print(new_df.isnull().sum()) # Is there any Null or Empty cell presents
new_df = new_df.dropna() # Deleting the rows which have Empty cells
new_df.shape # After deletion Vewing the shape
new_df.isnull().sum() #Is there any Null or Empty cell presents
new_df.sample(2) # Checking the random dataset sample
```

Here we select only ‘Volkswagen’ cars from the large dataset. Because different types of cars have different brand value and higher or lower price. So we take only one car company for better prediction.

Then we view the shape and check if any null cell present or not. We found there are many null cells present. We delete those rows which have null cells. It is very important when you make a dataset for fitting any data model. Then we cross check if any null cells present or not. No null cell found then we print 5 sample dataset values.

```
new_df = new_df[['Engine HP','MSRP']] # We only take the 'Engine HP' and 'MSRP' columns
new_df.sample(5) # Checking the random dataset sample
```

Here we select only 2 specific (‘Engine HP’ and ‘MSRP’) columns from all columns. It is very important to select only those columns which could be helpful for prediction. It depends on your common sense to select those columns. Please select those columns that wouldn’t spoil your prediction. After select only 2 columns, we view our new dataset.

```
X = np.array(new_df[['Engine HP']]) # Storing into X the 'Engine HP' as np.array
y = np.array(new_df[['MSRP']]) # Storing into y the 'MSRP' as np.array
```

```
print(X.shape) # Vewing the shape of X
print(y.shape) # Vewing the shape of y
```

Here we put the **‘Engine HP’** column as a numpy array into **‘X’** variable. And **‘MSRP’** column as a numpy array into **‘y’** variable. Then check the shape of the array.

3. Visualizing the data

```
plt.scatter(X,y,color="red") # Plot a graph X vs y
plt.title('HP vs MSRP')
plt.xlabel('HP')
plt.ylabel('MSRP')
plt.show()
```

   Here we plot a scatter plot graph between **‘MSRP’** and **‘HP’.** After viewing this graph we ensured that we can perform a linear regression for prediction.

4. Splitting our Data set in Dependent and Independent variables.

```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size = 0.25,random_state=15)
# Spliting into train & test dataset
```

   Here we split our ‘X’ and ‘y’ dataset into ‘X_train’, ‘X_test’ and ‘y_train’, ‘y_test’. Here we take 25% data as test dataset and remaining as train dataset. We take the random_state value as 15 for our better prediction.

5. Performing simple linear regression (create linear regression model)

```
regressor = LinearRegression() # Creating a regressior
regressor.fit(X_train,y_train) # Fiting the dataset into the model
```

   We create regressor. And we fit the X_train and y_train into the regressor model.

6. Residual analysis(Check the results of model fitting to know whether the model is satisfactory)

```
plt.scatter(X_test,y_test,color="green") # Plot a graph with X_test vs y_test
plt.plot(X_train,regressor.predict(X_train),color="red",linewidth=3) # Regressior line showing
plt.title('Regression(Test Set)')
plt.xlabel('HP')
plt.ylabel('MSRP')
plt.show()
```

   Here we plot a scatter plot graph between X_test and y_test datasets and we draw a regression line.

```
plt.scatter(X_train,y_train,color="blue")  # Plot a graph with X_train vs y_train
plt.plot(X_train,regressor.predict(X_train),color="red",linewidth=3) # Regressior line showing
plt.title('Regression(training Set)')
plt.xlabel('HP')
plt.ylabel('MSRP')
plt.show()
```

Here we plot the final **X_train vs y_train** scatterplot graph with a **best-fit regression line**. Here we can clearly understand the regression line.

7. Predictions on the test set (apply the model)

Here we print R^2, Mean Error, write function predict the price of car.

```
y_pred = regressor.predict(X_test)
print('R2 score: %.2f' % r2_score(y_test,y_pred)) # Priniting R$^2$ Score

print('Mean Error :',mean_squared_error(y_test,y_pred)) # Priniting the mean error

def car_price(hp): # A function to predict the price according to Horse power
   result = regressor.predict(np.array(hp).reshape(1, -1))
   return(result[0,0])
car_hp = int(input('Enter Volkswagen cars Horse Power : '))
print('This Volkswagen Price will be : ',int(car_price(car_hp))*69,'₹')
```

Linear regression example with Python code and scikit-learn

1. First, let's import linear_model from scikit-learn library:

```
from sklearn import linear_model
```

2. Now take features and labels set to train our program:

```
features = [[2],[1],[5],[10]]
labels = [27, 11, 75, 155]
```

3. After that create our model and fit the label and features to our model:

```
clf = linear_model.LinearRegression()
clf=clf.fit(features,labels)
```

4. In the end, pass data to the model and print the predicted result:

```
predicted = clf.predict([[8]])
print(predicted)
```

Building a logistic regression model

Steps to build Logistic Regression model in Python:

1. Import libraries /packages
2. Reading and understanding the data(do appropriate transformations- cleaning, filling nulls, duplicates, etc…)
3. Splitting our Data set in Dependent and Independent variables.

4. Performing simple linear regression (create logistic regression model)
5. Print the Accuracy and plot the Confusion Matrix
6. Print test data and predicted data Predictions on the test set

Sample Example -

Goal is to build a logistic regression model in Python in order to determine whether candidates would get admitted to a prestigious university.

Here, there are two possible outcomes: **Admitted** (represented by the value of '1') vs. **Rejected** (represented by the value of '0').

You can then build a logistic regression in Python, where:

- The dependent variable represents whether a person gets admitted; and
- The 3 independent variables are the GMAT score, GPA and Years of work experience

1. Import libraries /packages

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
import seaborn as sn
import matplotlib.pyplot as plt
```

2. Reading and understanding the data(eventually do appropriate transformations- cleaning, filling nulls, duplicates, etc…)

```
data = pd.read_csv("C:\TYBSC\Student_Score.csv")     # dataset
```

3. Splitting our Data set in Dependent and Independent variables.

   In our Data set we'll consider **gmat** score, **gpa** and Years of **work_experience** as Independent variable and **admitted** as Dependent Variable.

```
x = dataset.iloc[:, [2,3]].values
y = dataset.iloc[:, 4].values
```

Then, apply train_test_split. For example, you can set the test size to 0.25, and therefore the model testing will be based on 25% of the dataset, while the model training will be based on 75% of the dataset:

```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.25,random_state=0)
```

4. Performing simple logistic regression (create regression model)

```
logistic_regression= LogisticRegression()
```

```
logistic_regression.fit(x_train,y_train)
y_pred=logistic_regression.predict(x_test)
```

5. Print the Accuracy and plot the Confusion Matrix

Then, use the code below to get the Confusion Matrix:

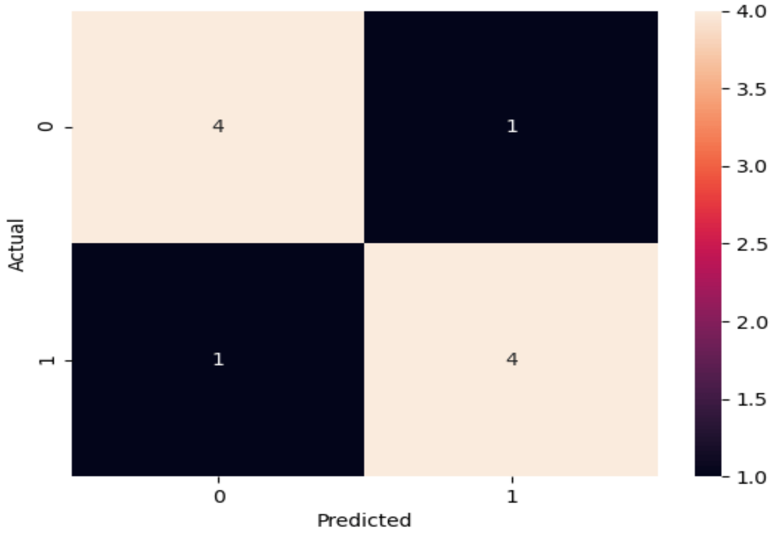
```
confusion_matrix = pd.crosstab(y_test, y_pred, rownames=['Actual'], colnames=['Predicted'])
sn.heatmap(confusion_matrix, annot=True)
```

For the final part, print the Accuracy and plot the Confusion Matrix:

```
print('Accuracy: ',metrics.accuracy_score(y_test, y_pred))
plt.show()
```

When we put all the code components together and Run the code in Python, will get the following Confusion Matrix with an Accuracy of 0.8

(note that depending on sklearn version, may get a different accuracy results. In above code case, the sklearn version is 0.22.2):



As can be observed from the matrix:

TP = True Positives = 4

TN = True Negatives = 4

FP = False Positives = 1

FN = False Negatives = 1

You can then also get the Accuracy using:

Accuracy = (TP+TN) / Total = (4+4) / 10 = 0.8

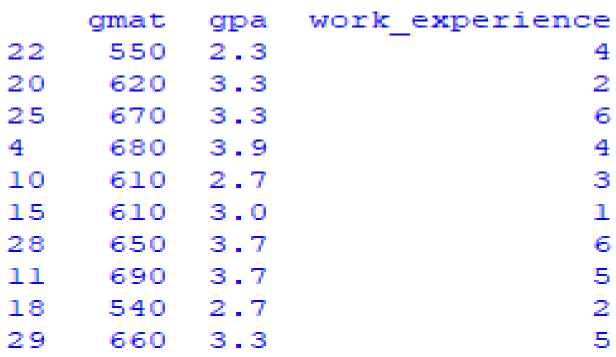
The accuracy is therefore 80% for the test set.

6. Print test data and predicted data Predictions on the test set

Diving Deeper into the Results -> print two components in the python code:

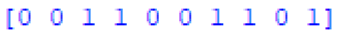
```
print (x_test)
print (y_pred)
```

Recall that our original dataset (from step 1) had 40 observations. Since we set the test size to 0.25, then the confusion matrix displayed the results for 10 records (=40*0.25). These are the 10 test records:



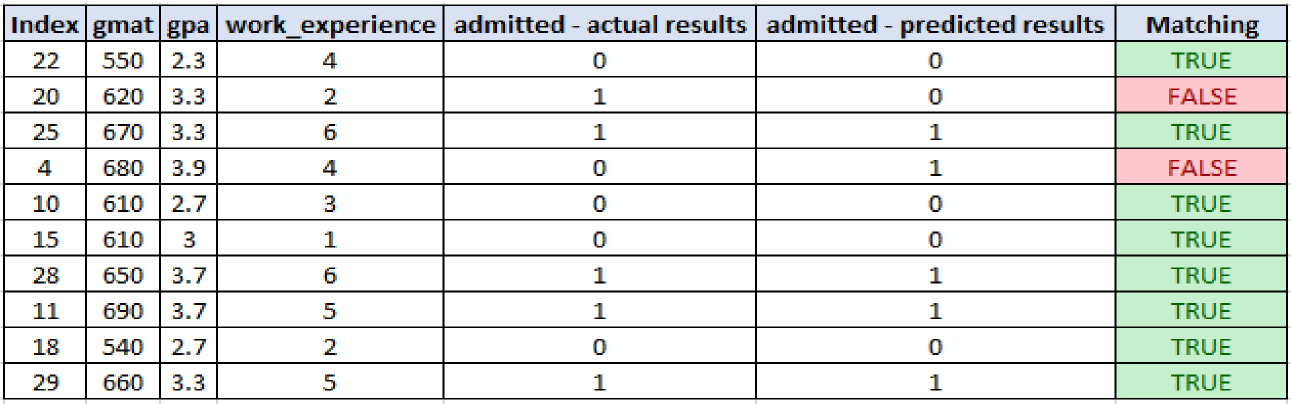
```
    gmat  gpa  work_experience
22   550  2.3                4
20   620  3.3                2
25   670  3.3                6
4    680  3.9                4
10   610  2.7                3
15   610  3.0                1
28   650  3.7                6
11   690  3.7                5
18   540  2.7                2
29   660  3.3                5
```

The prediction was also made for those 10 records (where 1 = admitted, while 0 = rejected):



```
[0 0 1 1 0 0 1 1 0 1]
```

In the actual dataset (from step-1), you'll see that for the test data, we got the correct results 8 out of 10 times:



Index	gmat	gpa	work_experience	admitted - actual results	admitted - predicted results	Matching
22	550	2.3	4	0	0	TRUE
20	620	3.3	2	1	0	FALSE
25	670	3.3	6	1	1	TRUE
4	680	3.9	4	0	1	FALSE
10	610	2.7	3	0	0	TRUE
15	610	3	1	0	0	TRUE
28	650	3.7	6	1	1	TRUE
11	690	3.7	5	1	1	TRUE
18	540	2.7	2	0	0	TRUE
29	660	3.3	5	1	1	TRUE

This is matching with the accuracy level of 80%

Lab Assignments

SET A

1. Create 'sales' Data set having 5 columns namely: ID, TV, Radio, Newspaper and Sales.(random 500 entries) Build a linear regression model by identifying independent and target variable. Split the variables into training and testing sets. then divide the training and testing sets into a 7:3 ratio, respectively and print them. Build a simple linear regression model.
2. Create 'realestate' Data set having 4 columns namely: ID,flat, houses and purchases (random 500 entries). Build a linear regression model by identifying independent and target variable. Split the variables into training and testing sets and print them. Build a simple linear regression model for predicting purchases.
3. Create 'User' Data set having 5 columns namely: User ID, Gender, Age, EstimatedSalary and Purchased. Build a logistic regression model that can predict whether on the given parameter a person will buy a car or not.

SET B

1. Build a simple linear regression model for Fish Species Weight Prediction. (download dataset https://www.kaggle.com/aungpyaeap/fish-market?select=Fish.csv)

2. Use the iris dataset. Write a Python program to view some basic statistical details like percentile, mean, std etc. of the species of 'Iris-setosa', 'Iris-versicolor' and 'Iris-virginica'. Apply logistic regression on the dataset to identify different species (setosa, versicolor, verginica) of Iris flowers given just 4 features: sepal and petal lengths and widths.. Find the accuracy of the model.

Signature of the instructor □ **Date** □

Assignment Evaluation

0:Not done	□	**2:Late Complete**	□	**4:Complete**	□
1:Incomplete	□	**3:Needs improvement**	□	**5:Well Done**	□

Assignment 2: Frequent itemset and Association rule mining

No. of slots: 02

Objectives

- To understand the impact of finding frequent patterns from large datasets.
- To learn the Apriori Algorithm which is used for frequent itemsets mining.
- To understand Association Rule Mining.
- To write and learn implementation of such concepts with Python.

Reading

You should read the following topics before starting this exercise:

- Why Pre-processing is must before analysis of data.
- What is support, confidence and lift.
- Learn definitions such as frequent itemsets, association between things, Apriori Property of sets.
- Basic understanding of libraries supported in Python for performing these tasks.

Ready Reference

Frequent Itemset Mining: Finding frequent patterns, associations, correlations, or causal structures among sets of items or objects in transaction databases, relational databases, and other information repositories.

Association Mining searches for frequent items in the data-set. In frequent mining usually the interesting associations and correlations between item sets in transactional and relational databases are found.
If there are 2 items X and Y purchased frequently then it is good to put them together in stores or provide some discount offer on one item on purchase of other item. This can really increase the sales. For example it is likely to find that if a customer buys Milk and bread he/she also buys Butter. So the association rule is [‘milk]^[‘bread’]=>[‘butter’].

Applications: Market Basket Analysis is one of the key techniques used by large retailers to uncover associations between item, catalog design, loss-leader analysis, clustering, classification, recommendation systems, etc.

Consider the following Transaction database:
D= {{butter, bread, milk, sugar};
    {butter, flour, milk, sugar};
    {butter, eggs, milk, salt};
    {eggs};
    {butter, flour, milk, salt, sugar}}

Question of interest: Which items are bought together frequently?

Apriori is an algorithm for frequent item set mining and association rule learning over relational databases. The name of the algorithm is based on the fact that the algorithm uses *prior knowledge* of frequent itemset properties. Apriori employs an iterative approach known as a *levelwise* search, where k-itemsets are used to explore (*k*+1)-itemsets

To construct association rules between items, the algorithm considers 3 important factors which are, support, confidence and lift. Each of these factors is explained as follows:

Support:

The support of item I is defined as the ratio between the number of transactions containing the item I by the total number of transactions expressed as :

$$support(I) = \frac{Number\ of\ transactions\ containing\ I}{Total\ number\ of\ transactions}$$

Confidence:

This is measured by the proportion of transactions with item I1, in which item I2 also appears.

$$confidence(I1 \rightarrow I2) = \frac{Number\ of\ transactions\ containing\ items\ I1\ and\ I2}{Total\ number\ of\ transactions\ containing\ I1}$$

Given that the item on the left hand side (**antecedent**) is purchased then the item on the right hand side(**consequent**) would also be purchased.

Lift:

Lift is the ratio between the confidence and support expressed as :

$$lift(I1 \rightarrow I2) = \frac{confidence(I1 \rightarrow I2)}{support(I2)}$$

Lift (antecedent => consequent) = 1 means that there is no correlation within the itemset, > 1 means that there is a positive correlation within the itemset, i.e., products in the itemset, antecedent, and consequent, are more likely to be bought together, < 1 means that there is a negative correlation within the itemset, i.e., products in itemset, antecedent, and consequent, are unlikely to be bought together.

The steps of the apriori algorithm can be given as:-

1. Define the minimum support and confidence for the association rule
2. Take all the subsets in the transactions with higher support than the minimum support
3. Take all the rules of these subsets with higher confidence than minimum confidence
4. Sort the association rules in the decreasing order of lift.
5. Visualize the rules along with confidence and support.

In this assignment you will analyze collections of market baskets and will determine frequent itemsets and association rules present in the collections.

Python libraries

Python has many libraries for apriori implementation.

i. Mlxtend (apriori)
ii. Apyori (apriori)
iii. pypi (efficient_apriori)

The apriori module from mlxtend library provides fast and efficient apriori implementation.

Usage:
apriori(df, min_support=0.5, use_colnames=False, max_len=None, verbose=0, low_memory=False)

Parameters

- df : One-Hot-Encoded DataFrame or DataFrame that has 0 and 1 or True and False as values
- min_support : Floating point value between 0 and 1 that indicates the minimum support required for an itemset to be selected.
  # of observation with item / total observation# of observation with item / total observation
- use_colnames : This allows to preserve column names for itemset making it more readable.
- max_len : Max length of itemset generated. If not set, all possible lengths are evaluated.
- verbose : Shows the number of iterations if >= 1 and low_memory is True. If =1 and low_memory is False , shows the number of combinations.
- low_memory :
- If True, uses an iterator to search for combinations above min_support. Note that while low_memory=True should only be used for large dataset if memory resources are limited, because this implementation is approx. 3–6x slower than the default.

The function returns a pandas DataFrame with columns ['support', 'itemsets'] of all itemsets that are >= min_support and < than max_len (if max_len is not None).

Mining Association Rules
Frequent if-then associations called association rules which consists of an antecedent (if) and a consequent (then). The following function returns the association rules from the frequent itemsets which satisfy the given metric threshold. Metric can be set to confidence, lift, support, leverage and conviction.

association_rules(frequent_items, metric=’confidence’, min_threshold=0.5, support_only=False)

Leverage computes the difference between the observed frequency of A and C appearing together and the frequency that would be expected if A and C were independent. A leverage value of 0 indicates independence.
A high conviction value means that the consequent is highly depending on the antecedent.

Self-Activity

Step 1: Install the libraries

```
pip install mlxtend
```

Step 2: Import the libraries

```
import pandas as pd
from mlxtend.frequent_patterns import apriori, association_rules
```

Step 3: Read the data, encode the data

Create the sample dataset

```
transactions = [['eggs', 'milk','bread'],
                ['eggs', 'apple'],
                ['milk', 'bread'],
                ['apple', 'milk'],
                ['milk', 'apple', 'bread']]
```

The dataset contains a set of transactions with a set of text items. This needs to be converted into numerical form to be analyzed. The label encoding process is used to convert textual labels into numeric form in order to prepare it to be used in a machine-readable form. We can transform it into the right format via the TransactionEncoder as follows:

```
from mlxtend.preprocessing import TransactionEncoder
te=TransactionEncoder()
te_array=te.fit(transactions).transform(transactions)
df=pd.DataFrame(te_array, columns=te.columns_)
df
```

You should see something like this

	apple	**bread**	**Eggs**	**milk**
0	False	True	True	True
1	True	False	True	False
2	False	True	False	True
3	True	False	False	True
4	True	True	False	True

Step 4: Find the frequent itemsets

Generate frequent itemsets that have a support value of at least 50%. By default, apriori returns the column indices of the items, For better readability, we can set use_colnames=True to convert these integer values into the respective item names:

```
freq_items = apriori(df, min_support = 0.5, use_colnames = True)
print(freq_items)
```

You will get

```
support      itemsets
0      0.6        (apple)
1      0.6        (bread)
2      0.8         (milk)
3      0.6  (milk, bread)
```

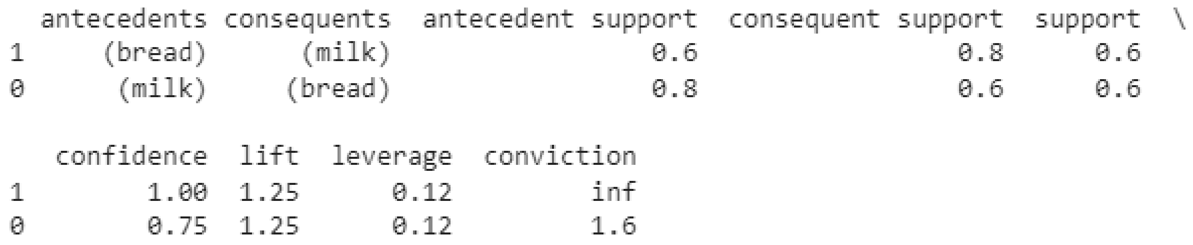
Change the value of the min_support and see the output

Step 5: Generate the association rules

Generate association rules that have a support value of at least 5%

```
rules = association_rules(freq_items, metric ='support', min_threshold=0.05
)
rules = rules.sort_values(['support', 'confidence'], ascending =[False,Fals
e])
print(rules)
```

The output will be:



```
  antecedents consequents  antecedent support  consequent support  support  \
1     (bread)      (milk)                 0.6                 0.8      0.6   
0      (milk)     (bread)                 0.8                 0.6      0.6   

   confidence  lift  leverage  conviction  
1        1.00  1.25      0.12         inf  
0        0.75  1.25      0.12         1.6  
```

To perform the above on a standard dataset, apply the following steps:

1. Download the csv file

```
from google.colab import files
data = files.upload()
```

2. Read the data from the csv file

```
import io
import pandas as pd
df = pd.read_csv(io.BytesIO(data['Market_Basket_Optimisation.csv']))
```

3. View the contents, info, columns and other details
4. Now, Convert Pandas DataFrame into a list of lists for encoding

```
transactions = []
for i in range(0, len(df)):
    transactions.append([str(df.values[i,j]) for j in range(0, len(df.c
olumns))])
```

5. Apply TransformEncoder to the transactions list
6. Apply the apriori algorithm

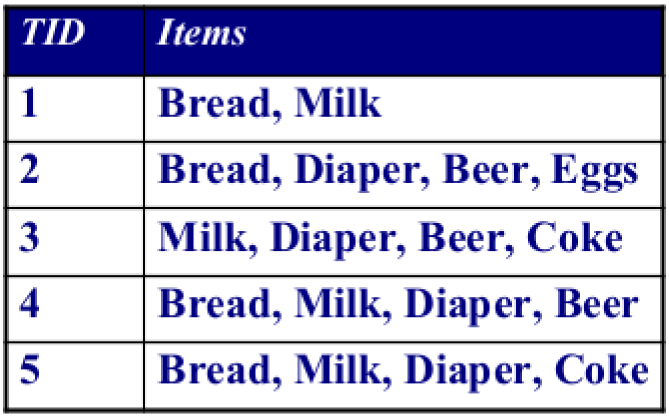
Dataset Sources

https://www.kaggle.com/datasets/sivaram1987/association-rule-learningapriori
https://github.com/shivang98/Market-Basket-Optimization
https://www.kaggle.com/datasets/hemanthkumar05/market-basket-optimization
https://www.kaggle.com/datasets/irfanasrullah/groceries

Lab Assignments

SET A:

1. Create the following dataset in python



TID	*Items*
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Convert the categorical values into numeric format.

Apply the apriori algorithm on the above dataset to generate the frequent itemsets and association rules. Repeat the process with different min_sup values.

2. Create your own transactions dataset and apply the above process on your dataset.

SET B:

1. Download the Market basket dataset.
   Write a python program to read the dataset and display its information.
   Preprocess the data (drop null values etc.)
   Convert the categorical values into numeric format.
   Apply the apriori algorithm on the above dataset to generate the frequent itemsets and association rules.
2. Download the groceries dataset.
   Write a python program to read the dataset and display its information.
   Preprocess the data (drop null values etc.)
   Convert the categorical values into numeric format.
   Apply the apriori algorithm on the above dataset to generate the frequent itemsets and association rules.

SET C:

Write a python code to implement the apriori algorithm. Test the code on any standard dataset.

Signature of the instructor □ Date □

Assignment Evaluation

0: Not done □	2: Late Complete □	4: Complete □
1: Incomplete □	3: Needs improvement □	5: Well Done □

Assignment 3 : Text and Social Media Analytics

No. of slots: 03

Objectives

- To understand the concept of sentiment analysis.
- To learn various methodologies for analysis on text including text analytics, tokenization, frequency distribution, stopwords, stemming, lemmatization, part-of-speech tagging.
- To write the Python scripts using various libraries for sentiment analysis using natural language processing toolkit and classifying emotions on basis of labels i.e. Positive, Negative and Neutral. Also to use wordcloud package for words comparison.
- To perform analysis on social media data such as Facebook, Twitter, YouTube.
- To graphically represent the analyzed data.

Reading

You should read the following topics before starting the exercise :
What is the need of doing data analysis using natural language processing. Basics of Python libraries such as pandas, matplotlib, numpy, scikit-learn, nltk, VADER tool to perform the data analysis.

Ready Reference

Python Libraries for performing text and Sentiment Analysis :

Natural Language Toolkit (NLTK) :
NLTK is a Python Package for performing Natural Language Processing on human language data which is mostly unstructured. It mainly focuses on analyzing textual data. It supports different natural language processing algorithms such as Tokenization, Frequency Distribution, Stopwords, Lexicon Normalization, Stemming, Lemmatization, POS Tagging. These are considered as pre-processing steps to perform text analytics.

Installation of NLTK : You can use any IDE to perform Python programming for the following tasks. Here Spyder IDE is used.

✓ To install NLTK, use pip as follows :

```
pip install nltk
```

✓ Then download the supportable NLTK packages using :

```
import nltk
nltk.download()
```

✓ After running the above script, a screen will come to download the packages. Here click on download to download all the supporting NLTK packages.

You can also download all NLTK packages using Python statement :

```
nltk.download(‘all’)
```

✓ If all the packages are not needed, then individual packages can also be installed by passing its name in nltk.download().

Syntax : `nltk.download(‘package_name’)`

For example : `nltk.download(‘punkt’)`

Pre-processing steps for text analytics using NLTK :

- **Tokenization :** It is the first step to perform text analytics. Tokenization means breaking down a textual paragraph into small chunks such as words or sentences. It is classified into two sections :

✓ **Sentence Tokenization and Word Tokenization :** Sentence Tokenization breaks the text into sentences whereas Word Tokenization breaks the text into words.

Example :

```
# Import sent_tokenize and word_tokenize package belonging to nltk
from nltk.tokenize import sent_tokenize
from nltk.tokenize import word_tokenize

# Take some textual content to tokenize it in sentences and words.
paragraph_text="""Hello all, Welcome to Python Programming Academy. Python
Programming Academy is a nice platform to learn new programming skills. It is
difficult to get enrolled in this Academy."""

# Supply textual content to sent_tokenize() and word_tokenize()
tokenized_text_data=sent_tokenize(paragraph_text)
tokenized_words=word_tokenize(paragraph_text)
print("Tokenized Sentences : \n", tokenized_text_data, "\n")
print("Tokenized Words : \n",tokenized_words, "\n")
```

Output :

```
Tokenized Sentences :
 ['Hello all, Welcome to Python Programming Academy.', 'Python Programming Acade
my is a nice platform to learn new programming skills.', 'It is difficult to get
enrolled in this Academy.']

Tokenized Words :
 ['Hello', 'all', ',', 'Welcome', 'to', 'Python', 'Programming', 'Academy', '.',
'Python', 'Programming', 'Academy', 'is', 'a', 'nice', 'platform', 'to', 'learn'
, 'new', 'programming', 'skills', '.', 'It', 'is', 'difficult', 'to', 'get', 'en
rolled', 'in', 'this', 'Academy', '.']
```

- **Frequency Distribution :**

✓ The frequency distribution helps to understand how many words have occurred how many times in the given textual data.

Example :

```
# Import word_tokenize
from nltk.tokenize import word_tokenize
# Import FreqDist package belonging to nltk.probability
from nltk.probability import FreqDist
# Textual data for word tokenization
```

```
paragraph_text="""Hello all, Welcome to Python Programming Academy. Python
Programming Academy is a nice platform to learn new programming skills. It is
difficult to get enrolled in this Academy."""
# Word Tokenization
tokenized_words=word_tokenize(paragraph_text)
# Pass the tokenized words to FreqDist
frequency_distribution=FreqDist(tokenized_words)
print(frequency_distribution)
```

Output :

```
<FreqDist with 24 samples and 32 outcomes>
```

✓ To find most common words using Frequency Distribution, add the following lines in above code :

```
print(frequency_distribution.most_common(2))
```

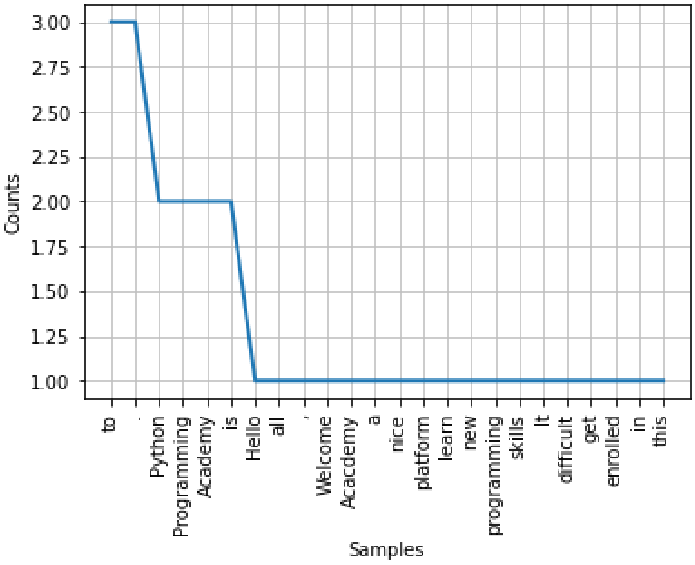
Output :

```
[('to', 3), ('.', 3)]
```

✓ To plot the Frequency Distribution, add the following lines of code :

```
import matplotlib.pyplot as plt
frequency_distribution.plot(32,cumulative=False)
plt.show()
```

Output :



➢ **Stopwords :** Stopwords are considered as Noise in textual data. For example if text is containing words such as is, are, am, a, this, the, an etc. then they are treated as stopwords.

✓ These stopwords needs to be removed from actual text for further processing. Using NLTK, first identify and create a list of stopwords in given text. Then remove it from the original content. Before working with stopwords, make sure to download it by using following :

```
import nltk
nltk.download('stopwords')
```

To check list all Stopwords :

```
from nltk.corpus import stopwords
# It will find the stowords in English language.
stop_words_data=set(stopwords.words("english"))
print(stop_words_data)
```

Output :

```
{'wouldn', 'down', 'was', 'any', 'themselves', 'on', 'how', 'y', 'them', 'do',
'as', "couldn't", 'wasn', 'can', 'yourself', "mightn't", 'm', "wasn't", 'yours',
"haven't", 'have', 'their', 'from', 'with', 'through', 'been', 'couldn', 'here',
'your', 'above', 'same', 'ours', 'now', 'isn', 'that', 'just', 'further',
'only', "won't", 'having', 'these', 'won', 'himself', 'ourselves', 'which',
"you're", 'while', 'of', "doesn't", "should've", "mustn't", 'hadn', 'are',
'not', 'he', 'she', 'am', 'an', 'most', 'whom', 'where', 'than', 'didn',
"isn't", 'shouldn', 'what', 'mustn', 'some', 'very', 'should', 'ain', "you'd",
'yourselves', 'own', 'but', 'we', 't', 'out', 'such', 'in', 've', 'this',
'shan', 'about', 'over', 'both', 'all', 'why', 'i', 'being', "wouldn't", 'll',
'myself', 'between', 'has', "didn't", 'hers', 'hasn', "she's", 'other', 'if',
'itself', 'below', "aren't", 'too', 'under', 'herself', 'be', 'after', 'off',
're', 'during', 'until', 'our', "shouldn't", 'into', 'don', 'again', 'nor',
'needn', "that'll", "weren't", 'no', 'so', 'then', 'before', 'his', 'its',
'few', 'doing', "don't", "you'll", "hadn't", 'because', 'there', 'did', 'my',
"needn't", "it's", 'they', 'for', 'does', 'is', 'a', 'against', 'who', 'and',
"shan't", 'o', 'weren', 'him', 'or', 'theirs', 'were', 'had', 'doesn', 'you',
'haven', 'those', 'me', 'when', 's', 'd', 'it', 'up', 'by', 'each', 'once',
'aren', "you've", 'her', "hasn't", 'to', 'more', 'will', 'mightn', 'the', 'at',
'ma'}
```

Removing Stopwords :

The above words in the output are predefined stopwords in English Language. If either of these words occur in a user-defined textual data, then it can be removed as follows :

```
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
# Textual data to remove stopwords
paragraph_text="""Hello all, Welcome to Python Programming Academy. Python
Programming Academy is a nice platform to learn new programming skills. It is
difficult to get enrolled in this Academy."""
# Word Tokenization
tokenized_words=word_tokenize(paragraph_text)
# It will find the stowords in English language.
stop_words_data=set(stopwords.words("english"))
# Create a stopwords list to filter it from original text
filtered_words_list=[]
for words in tokenized_words:
    if words not in stop_words_data:
        filtered_words_list.append(words)
print("Tokenized Words : \n",tokenized_words,"\n")
print("Filtered Words : \n",filtered_words_list,"\n")
```

Output :

```
Tokenized Words :

 ['Hello', 'all', ',', 'Welcome', 'to', 'Python', 'Programming', 'Academy', '.', 'Python', 'Programming', 'Academy', 'is', 'a', 'nice', 'platform', 'to', 'learn', 'new', 'programming', 'skills', '.', 'It', 'is', 'difficult', 'to', 'get', 'enrolled', 'in', 'this', 'Academy', '.']

Filtered Words :

 ['Hello', ',', 'Welcome', 'Python', 'Programming', 'Academy', '.', 'Python', 'Programming', 'Academy', 'nice', 'platform', 'learn', 'new', 'programming', 'skills', '.', 'It', 'difficult', 'get', 'enrolled', 'Academy', '.']
```

- **Stemming :** Stemming is a process of linguistics normalization to reduce words to their word root or divide the derivational affixes. For example : writing, wrote, written can stemmed or reduced as write.

Example :

```python
# Same code as previous example to remove stop words from tokenized words
from nltk.stem import PorterStemmer
porter_stemmer=PorterStemmer()
stemmed_text_words=[]
for words in filtered_words_list:
    stemmed_text_words.append(porter_stemmer.stem(words))
print("Filtered Words : \n",tokenized_words,"\n")
print("Stemmed Words : \n",stemmed_text_words,"\n")
```

- **Lemmatization :** Lemmatization is a process of removing words to their base words which is linguistically correct lemmas. For example : “Running” word will be lemmatized to “run”. Before that download the package “wordnet” belonging to nltk as follows :

```python
import nltk
nltk.download('wordnet')
# Lemmatization
from nltk.stem.wordnet import WordNetLemmatizer
lemmatizer=WordNetLemmatizer()
word_text="running"
print("Lemmatized Word : ",lemmatizer.lemmatize(word_text,"v"))
```

Output :

```
Lemmatized Word :  run
```

- **POS Tagging :** The POS (Part-of-Speech) tagging is basically used to identify the grammatical group of given words i.e. Noun, Pronoun, Verb, Adjective, Adverbs etc. on the basis of its context.
Before that download the package “averaged_perceptron_tagger” belonging to nltk as follows :

```python
import nltk
nltk.download('averaged_perceptron_tagger')
# Part-of-Speech Tagging
import nltk
```

```
from nltk.tokenize import word_tokenize
text_data="Hello all, Welcome to Python programming"
tokenized_data=word_tokenize(text_data)
print(nltk.pos_tag(tokenized_data))
```

Output :

```
[('Hello', 'NNP'), ('all', 'DT'), (',', ','), ('Welcome', 'NNP'), ('to', 'TO'),
('Python', 'NNP'), ('programming', 'NN')]
```

Text Summarization :

Text summarization is an NLP technique that extracts text from a large amount of data. It is the process of identifying the most important meaningful information in a document and compressing it into a shorter version by preserving its meaning. Types: Extractive summarization and Abstractive summarization

To perform extractive summarization, we calculate the sentence weights and choose the first ‘n’ sentences with maximum weight. The weights are calculated on the basis of the word frequencies

Steps:

1. Preprocess the text
2. Create the word frequency table
3. Tokenize the sentence
4. Score the sentences: Term frequency
5. Generate the summary

Sample code

```
import nltk
nltk.download('all')
#Preprocessing
import re
text="""

 Large paragraph of text
"""
text = re.sub(r'[[0-9]*]', ' ', text)
text = re.sub(r's+', ' ', text)
# Removing Square Brackets, digits, special symbols
import re
text = re.sub(r'[[0-9]{}*]', ' ', text)
# Removing special characters and digits
formatted_text = re.sub('[^a-zA-Z]', ' ', text)
#Here the formatted_article_text contains the formatted article.
#We will use this object to calculate the weighted frequencies and we will replace the weighted
#frequencies with words in the text object.
#Calculate the frequency of occurrence of each word. To find the weighted frequency, we divide the
#number of occurrences of all the words by the frequency of the most occurring word.
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize, sent_tokenize
stopWords = set(stopwords.words("english"))
words = word_tokenize(formatted_text)
```

```
# Creating a frequency table of words
wordfreq = {}
for word in words:
    if word in stopWords:
        continue
    if word in wordfreq:
        wordfreq[word] += 1
    else:
        wordfreq[word] = 1
#Compute the weighted frequencies
maximum_frequency = max(wordfreq.values())
for word in wordfreq.keys():
    wordfreq[word] = (wordfreq[word]/maximum_frequency)
# Creating a dictionary to keep the score # of each sentence
sentences = sent_tokenize(text)
sentenceValue = {}
for sentence in sentences:
    for word, freq in wordfreq.items():
        if word in sentence.lower():
            if sentence in sentenceValue:
                sentenceValue[sentence] += freq
            else:
                sentenceValue[sentence] = freq
import heapq
summary = ''
summary_sentences = heapq.nlargest(4, sentenceValue, key=sentenceValue.get)
summary = ' '.join(summary_sentences)
print(summary)
```

Sentiment Analysis using NLTK :

Sentiment analysis is a technique which detects the underlying sentiment on specific textual content. It is considered as a process of classifying text on the basis of labels i.e. positive, negative or neutral. To perform Sentiment Analysis using NLTK, it has a supportable package named as VADER. VADER stands for **Valence Aware Dictionary and Sentiment Reasoner**. It is a lexicon and rule-based sentiment analysis tool which is specifically used to identify the expressed sentiments. It is beneficial to use VADER on social media data since it not only gives the analysis as whether the text is positive or negative, but it also tells about the intensity of text i.e. how much positive or negative the text is. To use VADER, first download “vader_lexicon” package belonging to nltk.

```
import nltk
nltk.download('vader_lexicon')
```

Examples : Let’s consider some text statements expressing different emotions and analyzing them using VADER.

Example 1 :

```
# Import SentimentIntensityAnalyzer from vader package
from nltk.sentiment.vader import SentimentIntensityAnalyzer
# Create an object of SentimentIntensityAnalyzer()
```

```
vader_analyzer=SentimentIntensityAnalyzer()
text1="I am feeling good" # The text is positive.
print(vader_analyzer.polarity_scores(text1))
```

Output :

```
{'neg': 0.0, 'neu': 0.185, 'pos': 0.815, 'compound': 0.5267}
```

Here the output is giving some labels :

- neg : Negative
- neu : Neutral
- pos : Positive

It has given 'pos' value as 0.815 which is maximum of all the other values since the statement is positive. Similarly, we can check it on other emotions as well.

Example 2 :

```
text1="I hate tea"
print(vader_analyzer.polarity_scores(text1))
```

Output :

```
{'neg': 0.787, 'neu': 0.213, 'pos': 0.0, 'compound': -0.5719}
```

Example 3 : Consider the following example to get the overall rating about a statement i.e. overall whether it is positive, negative or neutral.

```
text1="I like Python"
result1=vader_analyzer.polarity_scores(text1)

# To find percentage of ratings
print("The sentence is rated as ",result1['pos']*100,"% Positive")
print("The sentence is rated as ",result1['neg']*100,"% Negative")
print("The sentence is rated as ",result1['neu']*100,"% Neutral")
if result1['compound']>=0.05:
    print("Overall rating for sentence is Positive")
elif result1['compound']<=-0.05:
    print("Overall rating for sentence is Negative")
else:
    print("Overall rating for sentence is neutral")
```

```
Output  :

The sentence is rated as  71.39 % Positive
The sentence is rated as  0.0 % Negative
The sentence is rated as  28.59 % Neutral
Overall rating for sentence is Positive
```

Word Cloud for Sentiment Analysis :

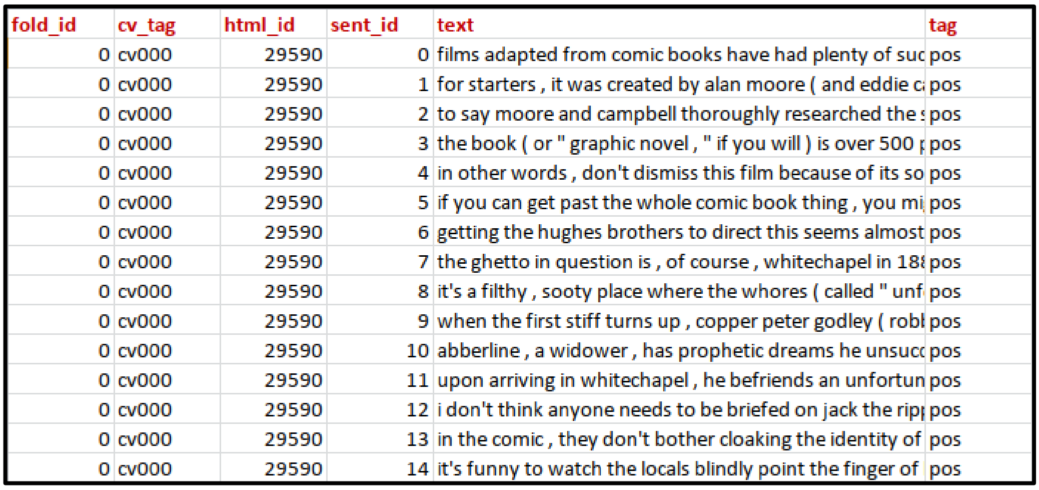
Word cloud is basically a data visualization technique to represent the textual content where the size of each visualized word implies its importance, frequency and intensity. It is a good tool to visualize the text and perform sentiment analysis to find the frequency of words having positive, negative or neutral emotions.

✓ To install wordcloud, use the following command :

```
pip install wordcloud
```

✓ **Example :** Download the movie_review.csv dataset from Kaggle by using the following link : https://www.kaggle.com/nltkdata/movie-review/version/3?select=movie_review.csv

Sample Data from movie_review.csv



fold_id	cv_tag	html_id	sent_id	text	tag
0	cv000	29590	0	films adapted from comic books have had plenty of suc	pos
0	cv000	29590	1	for starters , it was created by alan moore (and eddie c	pos
0	cv000	29590	2	to say moore and campbell thoroughly researched the	pos
0	cv000	29590	3	the book (or " graphic novel , " if you will) is over 500	pos
0	cv000	29590	4	in other words , don't dismiss this film because of its so	pos
0	cv000	29590	5	if you can get past the whole comic book thing , you mi	pos
0	cv000	29590	6	getting the hughes brothers to direct this seems almost	pos
0	cv000	29590	7	the ghetto in question is , of course , whitechapel in 18	pos
0	cv000	29590	8	it's a filthy , sooty place where the whores (called " unf	pos
0	cv000	29590	9	when the first stiff turns up , copper peter godley (robl	pos
0	cv000	29590	10	abberline , a widower , has prophetic dreams he unsucc	pos
0	cv000	29590	11	upon arriving in whitechapel , he befriends an unfortun	pos
0	cv000	29590	12	i don't think anyone needs to be briefed on jack the rip	pos
0	cv000	29590	13	in the comic , they don't bother cloaking the identity of	pos
0	cv000	29590	14	it's funny to watch the locals blindly point the finger of	pos

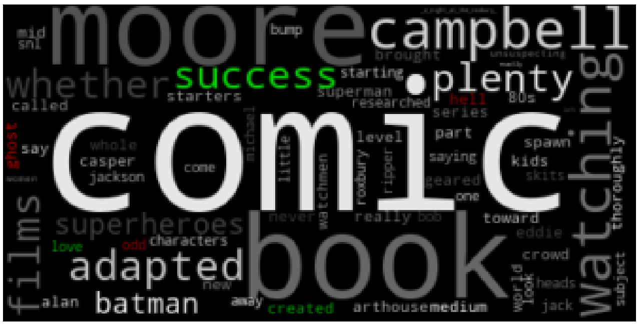
Now to perform sentiment analysis on above dataset and creating a wordcloud, consider the following code : (Here, we will represent Positive words with green color, Negative words with red color and Neutral words with white color)

```
# Import the necessary packages
from nltk.corpus import stopwords
from nltk.sentiment.vader import SentimentIntensityAnalyzer
from wordcloud import WordCloud, get_single_color_func
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

# Read Movies Reviews Data CSV file.
movies_data=pd.read_csv('movie_review.csv')
# Convert the column data type from ndarray to string.
movies_reviews=movies_data['text'].values.astype(str)
movies_reviews1=np.array_str(movies_reviews)
# It will find the stowords in English language.
stop_words_data=set(stopwords.words("english"))
words=movies_reviews1.split()

# Remove Duplicate Values
final_data=[]
for w in words:
    if w not in final_data:
        final_data.append(w)
```

```
# Create dictionaries to store positive and negative words with polarity.
positive_words=dict()
negative_words=dict()

# Create lists to store positive and negative words without polarity.
positive=[]
negative=[]

# Sentiment Analysis
sentiment_analyzer=SentimentIntensityAnalyzer()
for i in words:
    if not i.lower() in stop_words_data: # It will remove stopwords.
        polarity=sentiment_analyzer.polarity_scores(i)
        if polarity['compound']>=0.05: # Positive Sentiment
            positive_words[i]=polarity['compound']
        if polarity['compound']<=-0.05: # Negative Sentiment
            negative_words[i]=polarity['compound']

# Append the positive and negative words from dictionaries to lists i.e.
positive[] and negative[]
for key,value in positive_words.items():
    positive.append(key)
for key,value in negative_words.items():
    negative.append(key)

# Create a dictionary to mention the colors : green for positive and red for
negative
coloured_words={"green":positive,"red":negative}

# Implement separate colour assignments
class ColourAssignment(object):
    # Functions to give different colours on the basis of sentiments.
    def __init__(self,coloured_words,default):
        self.coloured_words=[
            (get_single_color_func(colour),set(words))
            for (colour,words) in coloured_words.items()]
        self.default=get_single_color_func(default)

    def get_colour(self,word):
        try:
            colour=next(
                colour for (colour,words) in self.coloured_words
                if word in words)
        except StopIteration:
            colour=self.default
        return colour

    def __call__(self,word, **kwargs):
        return self.get_colour(word) (word, **kwargs)
```

```
# To print the plot
word_cloud=WordCloud(collocations=False,background_color='black').generate(movies_reviews1)
# Neutral words will be visible as black
group_color=ColourAssignment(coloured_words, 'white')
word_cloud.recolor(color_func=group_color)
plt.figure()
plt.imshow(word_cloud, interpolation="bilinear")
plt.axis("off")
plt.show()
```

Output :



Social Media Data Analytics :

When it comes to social media such as Twitter, Facebook, YouTube etc., bulk of data is available which is needed to be examined or analyzed for interpreting the opinions of people conveyed in different formats. It basically puts the subjective information in the form emotions.

Twitter Data Analysis :

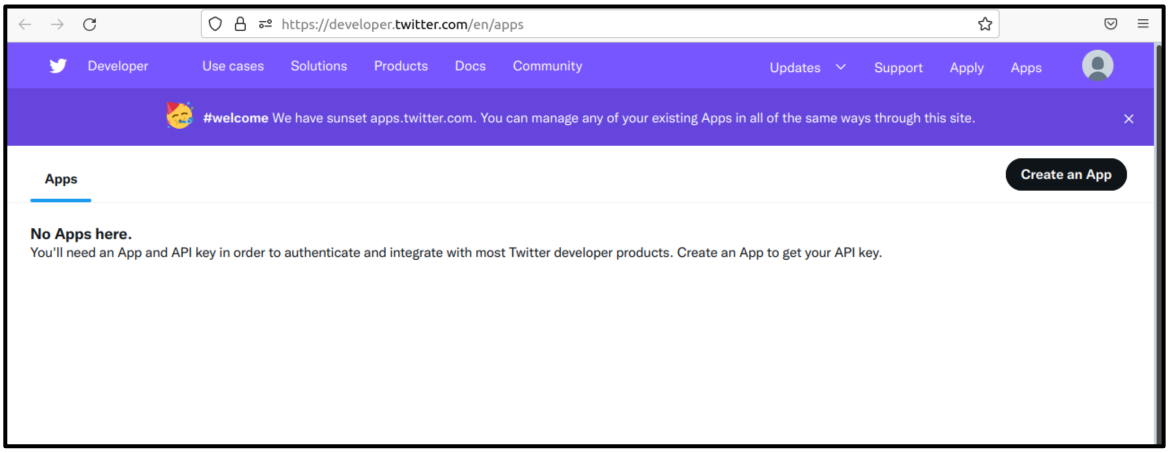
✓ To perform the analysis on twitter data, first we need to get the data.
✓ There are multiple ways to get Twitter data. Some are :
  - Getting tweets through twitter API and analyzing them.
  - Downloading the Twitter datasets online (Example : Kaggle)

Getting tweets through twitter API and analyzing the tweets :

To get the tweets through twitter API, Twitter account is needed and App is to be registered. Follow the below steps :

❖ First create a Twitter account if you do not have one. Visit to https://twitter.com/i/flow/signup and create an account. Existing account can also be used.

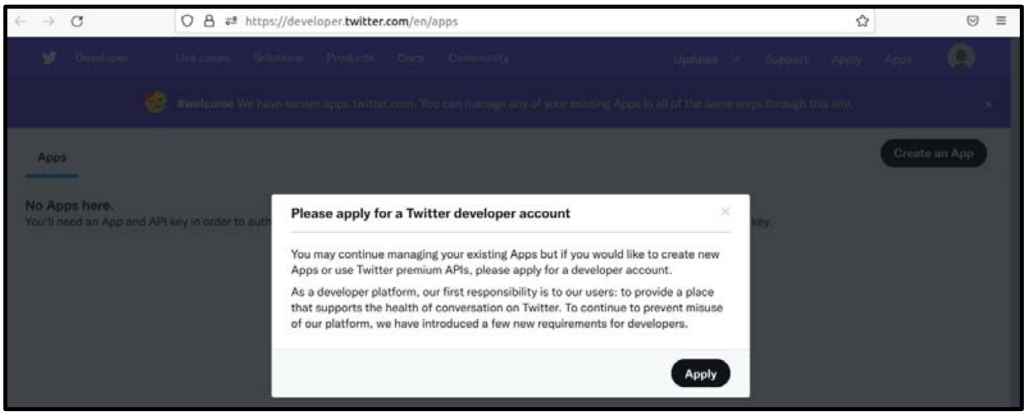
❖ Now create an App on Twitter Developer using following link : https://developer.twitter.com/en/apps



#welcome We have sunset apps.twitter.com. You can manage any of your existing Apps in all of the same ways through this site.

No Apps here.
You'll need an App and API key in order to authenticate and integrate with most Twitter developer products. Create an App to get your API key.

❖ Now click on "Create an App" button to create an application to get the API key for credentials. It will ask to apply for a Developer Account.

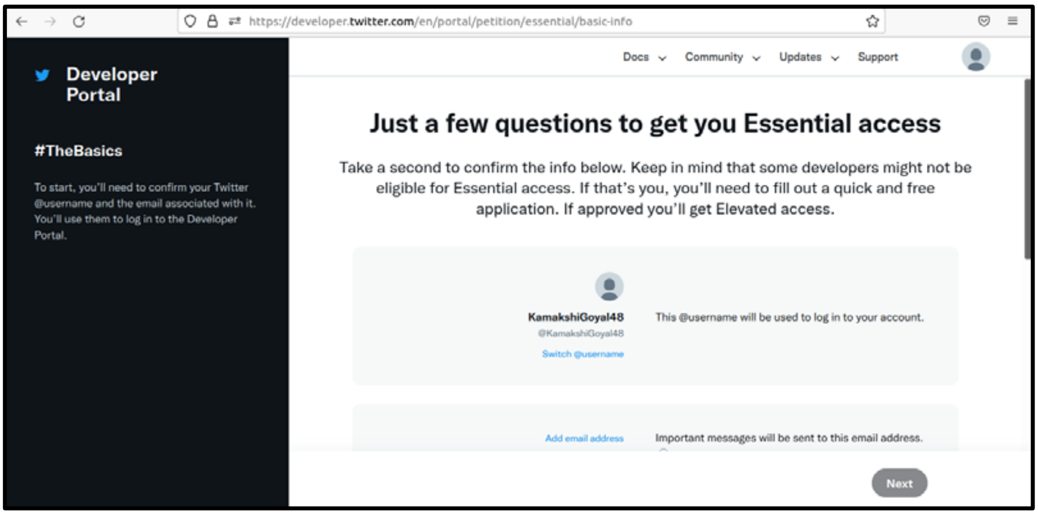


Please apply for a Twitter developer account

You may continue managing your existing Apps but if you would like to create new Apps or use Twitter premium APIs, please apply for a developer account.

As a developer platform, our first responsibility is to our users: to provide a place that supports the health of conversation on Twitter. To continue to prevent misuse of our platform, we have introduced a few new requirements for developers.

❖ Click on Apply and continue. And then answer the questions visible on the screen.



https://developer.twitter.com/en/portal/petition/essential/basic-info
Developer Portal
Docs
Community
Updates
Support

#TheBasics

To start, you'll need to confirm your Twitter @username and the email associated with it. You'll use them to log in to the Developer Portal.

Just a few questions to get you Essential access

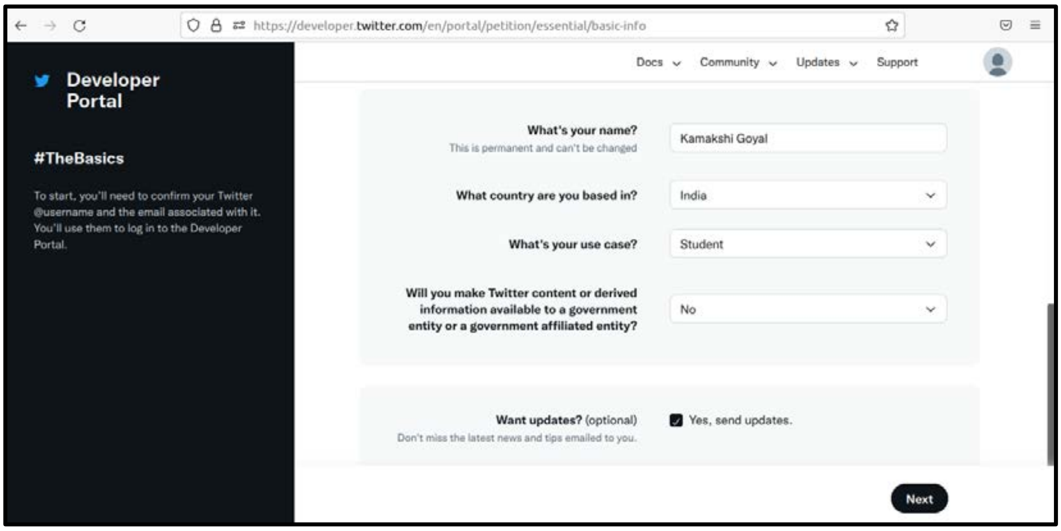
Take a second to confirm the info below. Keep in mind that some developers might not be eligible for Essential access. If that's you, you'll need to fill out a quick and free application. If approved you'll get Elevated access.

KamakshiGoyal48
@KamakshiGoyal48
Switch @username

This @username will be used to log in to your account.

Add email address

Important messages will be sent to this email address.



https://developer.twitter.com/en/portal/petition/essential/basic-info

Developer Portal

#TheBasics

To start, you'll need to confirm your Twitter @username and the email associated with it. You'll use them to log in to the Developer Portal.

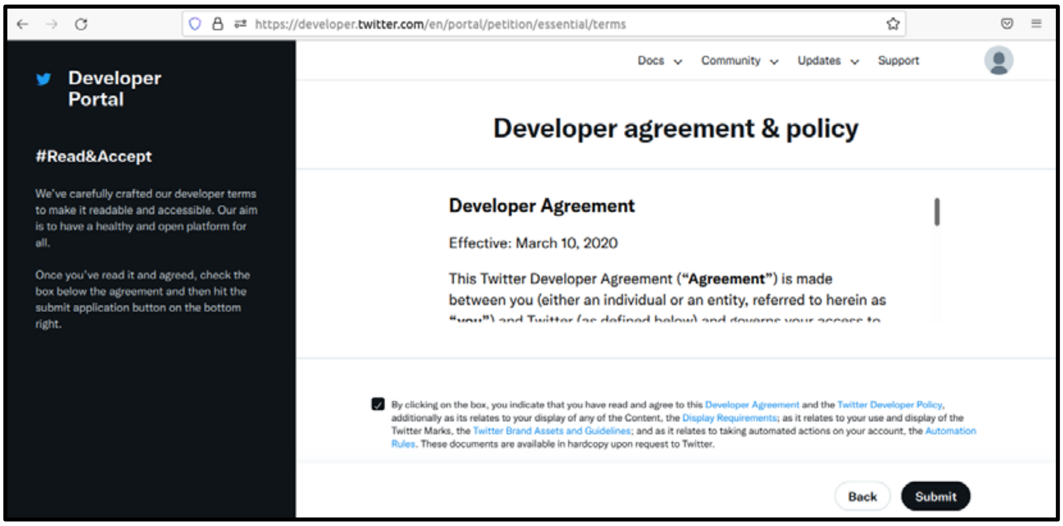
What's your name? This is permanent and can't be changed: Kamakshi Goyal

What country are you based in?: India

What's your use case?: Student

Will you make Twitter content or derived information available to a government entity or a government affiliated entity?: No

Want updates? (optional) Don't miss the latest news and tips emailed to you. ☑ Yes, send updates.



https://developer.twitter.com/en/portal/petition/essential/terms

Developer Portal

#Read&Accept

We've carefully crafted our developer terms to make it readable and accessible. Our aim is to have a healthy and open platform for all.

Once you've read it and agreed, check the box below the agreement and then hit the submit application button on the bottom right.

Developer agreement & policy

Developer Agreement

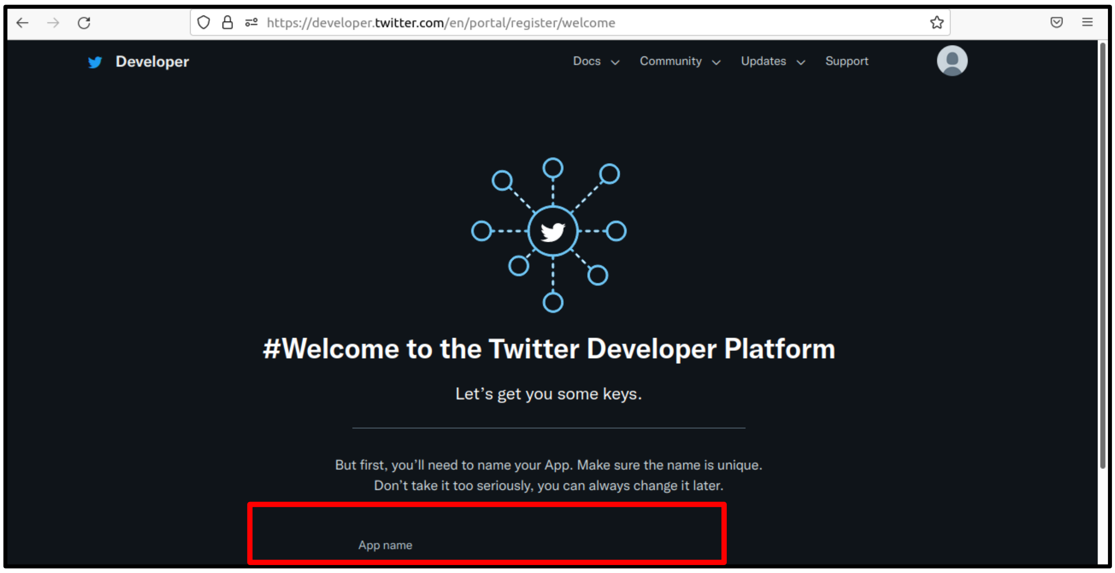
Effective: March 10, 2020

This Twitter Developer Agreement (**"Agreement"**) is made between you (either an individual or an entity, referred to herein as **"you"**) and Twitter (as defined below) and governs your access to

☑ By clicking on the box, you indicate that you have read and agree to this Developer Agreement and the Twitter Developer Policy, additionally as its relates to your display of any of the Content, the Display Requirements; as it relates to your use and display of the Twitter Marks, the Twitter Brand Assets and Guidelines; and as it relates to taking automated actions on your account, the Automation Rules. These documents are available in hardcopy upon request to Twitter.

❖ After submitting the request, you will receive a message from twitter to get the Email confirmation. Then we can get the keys. Visit the following link for App creation.

https://developer.twitter.com/en/portal/register/welcome



https://developer.twitter.com/en/portal/register/welcome

Developer

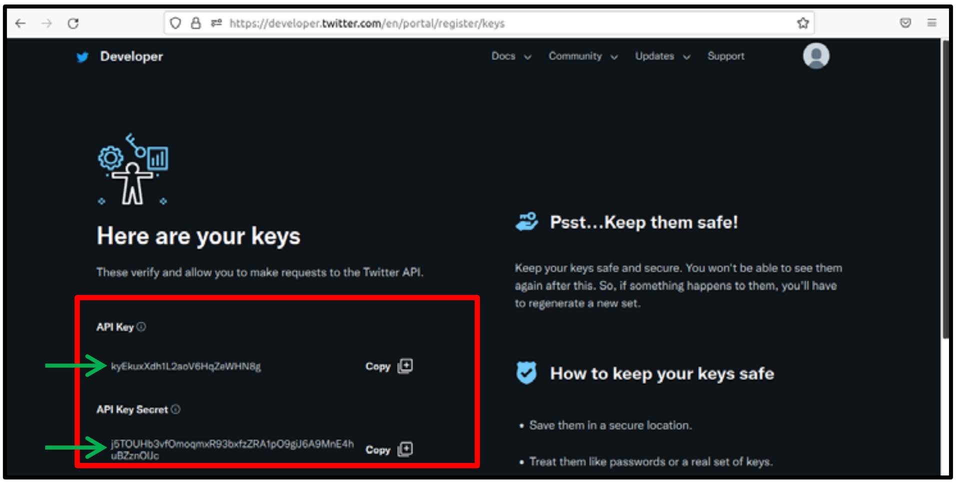
#Welcome to the Twitter Developer Platform

Let's get you some keys.

But first, you'll need to name your App. Make sure the name is unique. Don't take it too seriously, you can always change it later.

App name

❖ Now give the App name and click on Get Keys.



Here are your keys

These verify and allow you to make requests to the Twitter API.

API Key

kyEkuxXdh1L2aoV6HqZeWHN8g

API Key Secret

j5TOUHb3vfOmoqmxR93bxfzZRA1pO9gJ6A9MnE4h uBZznOlJc

Psst...Keep them safe!

Keep your keys safe and secure. You won't be able to see them again after this. So, if something happens to them, you'll have to regenerate a new set.

How to keep your keys safe

- Save them in a secure location.
- Treat them like passwords or a real set of keys.

❖ Libraries used for twitter data analysis :

1. **tweepy :** It is a Python library which is used to access the Twitter API. To install tweepy, use the following command :

```
pip install tweepy
```

Import the necessary library as :

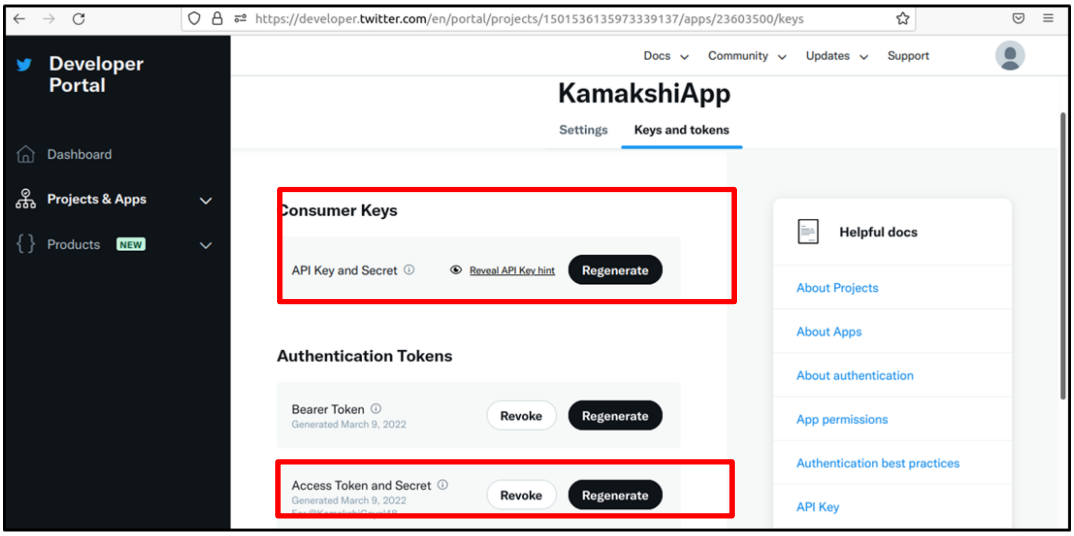
```
import tweepy
```

❖ Twitter API Authentication :

To perform Twitter API authentication, we have multiple options :

We can perform authentication using :

- Consumer Key
- Consumer Secret Key
- Access Token
- Access Token Secret


https://developer.twitter.com/en/portal/projects/1501536135973339137/apps/23603500/keys
Developer Portal
Dashboard
Projects & Apps
Products NEW
Docs
Community
Updates
Support

KamakshiApp

Consumer Keys

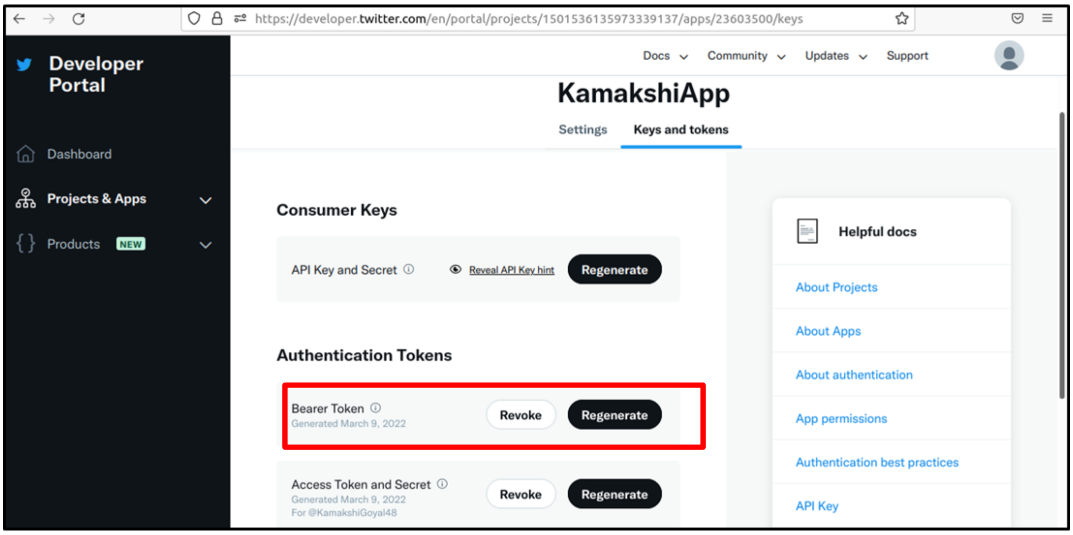
API Key and Secret

Authentication Tokens

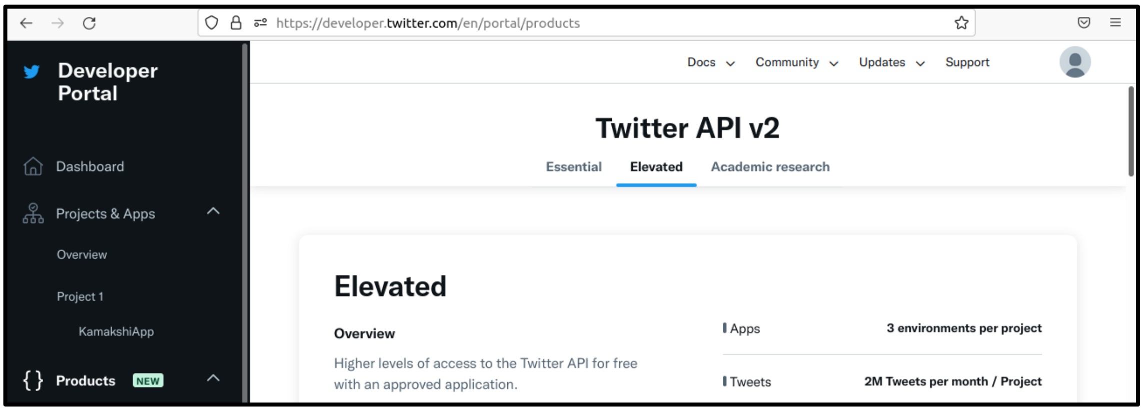
Bearer Token
Generated March 9, 2022

Access Token and Secret
Generated March 9, 2022

Or you can also use the “Bearer Token” to perform authentication. For this code, Bearer Token is used. If you want to use another approach, refer this : https://docs.tweepy.org/en/stable/authentication.html. You can find Bearer Token here :



Before using Bearer Token, make sure to use "Elevated" section of App. When first time app gets created, it comes with "Essential". But to use Bearer Token directly, "Elevated" is to be used.

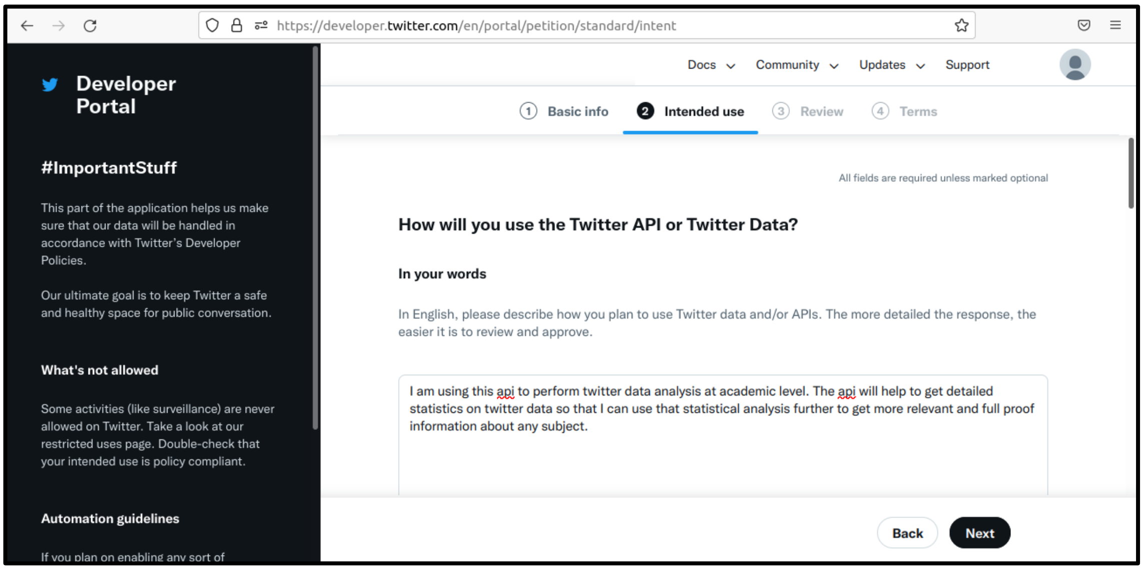


Elevated

Overview

Higher levels of access to the Twitter API for free with an approved application.

Apps	3 environments per project
Tweets	2M Tweets per month / Project



#ImportantStuff

This part of the application helps us make sure that our data will be handled in accordance with Twitter's Developer Policies.

Our ultimate goal is to keep Twitter a safe and healthy space for public conversation.

What's not allowed

Some activities (like surveillance) are never allowed on Twitter. Take a look at our restricted uses page. Double-check that your intended use is policy compliant.

Automation guidelines

If you plan on enabling any sort of

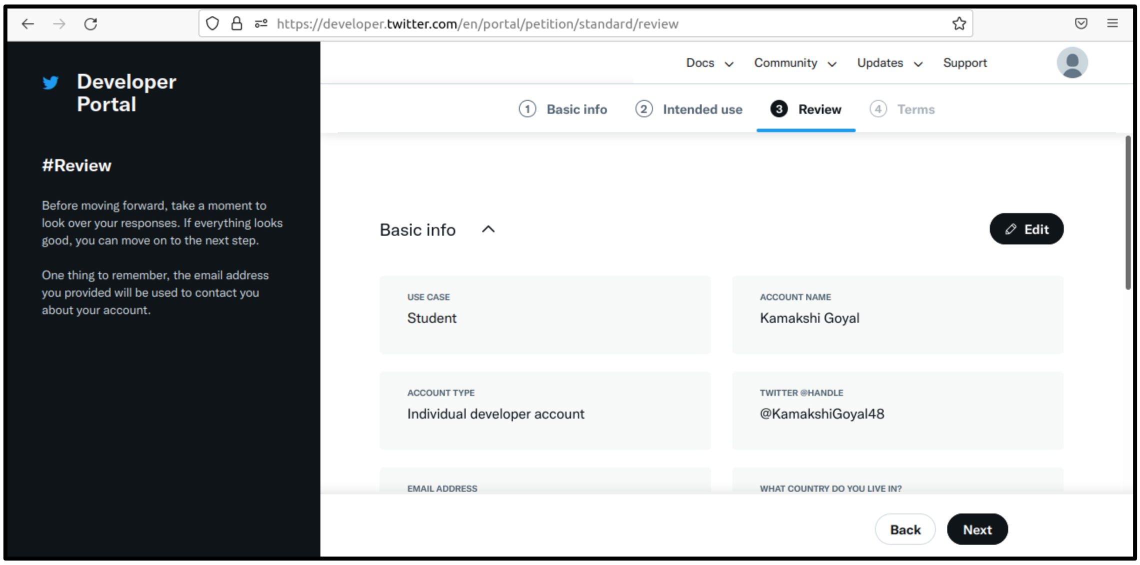
All fields are required unless marked optional

How will you use the Twitter API or Twitter Data?

In your words

In English, please describe how you plan to use Twitter data and/or APIs. The more detailed the response, the easier it is to review and approve.

I am using this api to perform twitter data analysis at academic level. The api will help to get detailed statistics on twitter data so that I can use that statistical analysis further to get more relevant and full proof information about any subject.



#Review

Before moving forward, take a moment to look over your responses. If everything looks good, you can move on to the next step.

One thing to remember, the email address you provided will be used to contact you about your account.

Basic info

USE CASE
Student

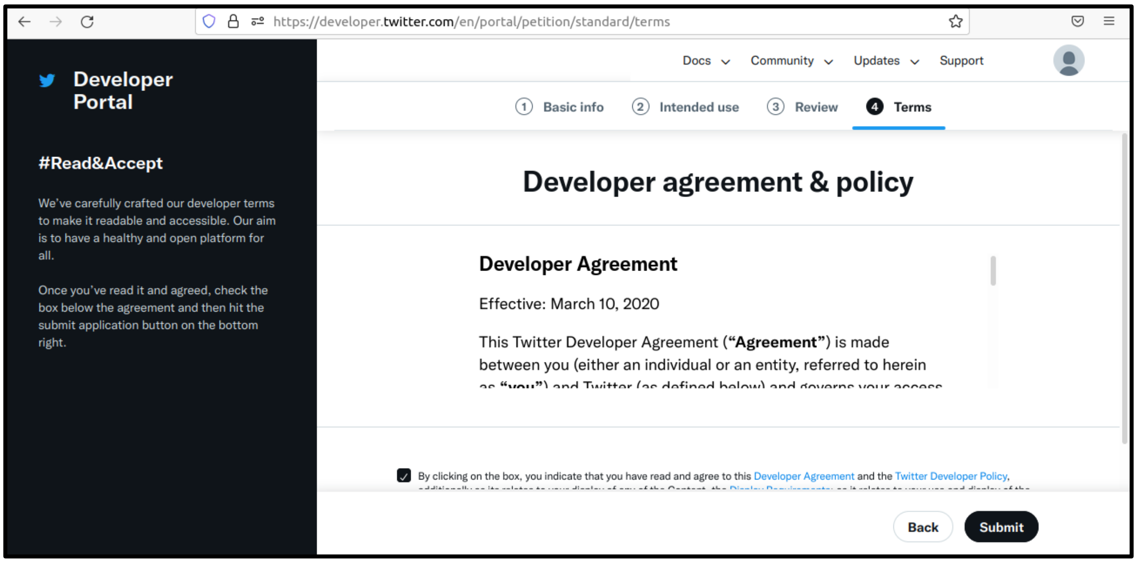
ACCOUNT NAME
Kamakshi Goyal

ACCOUNT TYPE
Individual developer account

TWITTER @HANDLE
@KamakshiGoyal48

EMAIL ADDRESS

WHAT COUNTRY DO YOU LIVE IN?



#Read&Accept

We've carefully crafted our developer terms to make it readable and accessible. Our aim is to have a healthy and open platform for all.

Once you've read it and agreed, check the box below the agreement and then hit the submit application button on the bottom right.

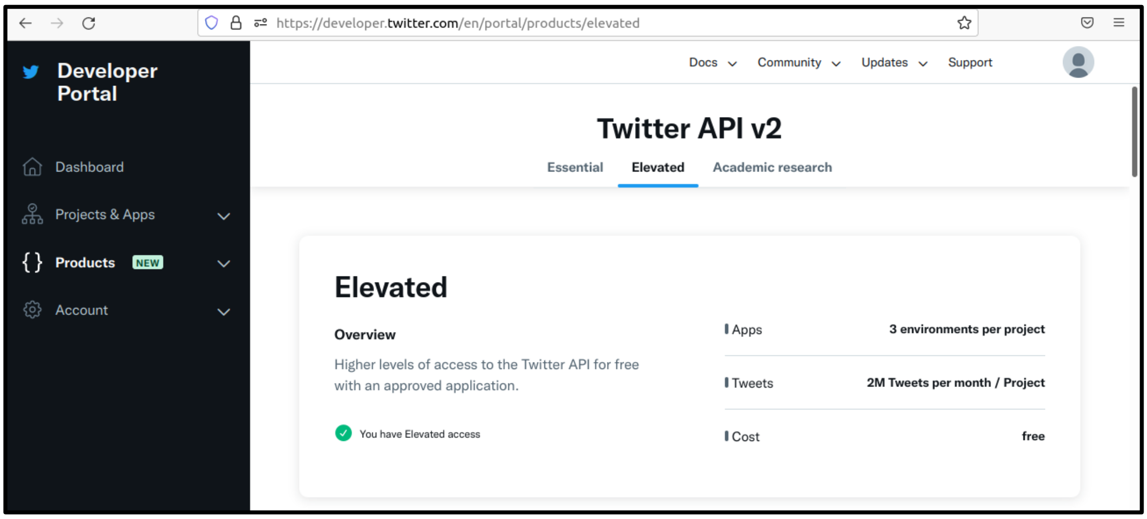
Developer agreement & policy

Developer Agreement

Effective: March 10, 2020

This Twitter Developer Agreement (**"Agreement"**) is made between you (either an individual or an entity, referred to herein as **"you"**) and Twitter (as defined below) and governs your access

☑ By clicking on the box, you indicate that you have read and agree to this Developer Agreement and the Twitter Developer Policy,



Twitter API v2

Elevated

Overview

Higher levels of access to the Twitter API for free with an approved application.

✓ You have Elevated access

Apps	3 environments per project
Tweets	2M Tweets per month / Project
Cost	free

In case if the token gets expired, then it can be regenerated as well. Add the following lines of code :

```
auth=tweepy.OAuth2BearerHandler("Your Bearer Token")
api=tweepy.API(auth)
```

❖ Getting tweets having Hash Tags or by using Keywords :

```
# Get the tweets on the basis of Hash Tags or Keywords.
search_tag=input("Enter the Hash Tag or Keyword for which you want to get the
tweets : ")
no_of_tweets=int(input("How many tweets you want ? "))
# Iterate over the tweets.
tweets=tweepy.Cursor(api.search_tweets, q=search_tag).items(no_of_tweets)
# Create a list to store all the tweets.
tweet_list=[]
for tweet in tweets:
    tweet_list.append(tweet.text)

print(tweet_list)
```

Output : (Example)

```
Enter the Hash Tag or Keyword for which you want to get the tweets : #sadhguru

How many tweets you want ? 5

['RT @gauravsingh_ss: 2- @SadhguruJV &amp; @cpsavesoil now "ST. XAVIER\'S HIGH
SCHOOL" (Chapra, Bihar) have also taken responsibility of #SaveSoil…', 'RT
@TUndercoverMonk: Get ready for the BIGGEST Environment movement on the planet.
\nThe intention is to make #Soil as the MAIN talking poin…', 'RT @SouvikMitra94:
@SadhguruJV @ivivianrichards @BeefyBotham @cpsavesoil #Sadhguru and
#SirVivRichards in same frame to #SaveSoil □□ https:…', "RT @PenguinIndia: Read
this #exclusive excerpt from @MansinghVivek's latest release, where @SadhguruJV
talks about the role of the mind, bo…", 'RT @AntiguaOpm: #InPhotos Today, PM
@gastonbrowne met with @SadhguruJV and @machelmontano ahead of the launch of a
global ecological campai…']
```

❖ **More Analysis on Twitter Data :** We can further perform different analysis on gathered data as follows :

- ✓ First select the user ID on which analysis is to be done.
- ✓ Then we can find various information related to tweets such as 'created_at', 'id', 'id_str', 'text', 'truncated', 'entities', 'metadata', 'source', 'in_reply_to_status_id', 'in_reply_to_status_id_str', 'in_reply_to_user_id', 'in_reply_to_user_id_str', 'in_reply_to_screen_name', 'user', 'geo', 'coordinates', 'place', 'contributors', 'retweeted_status', 'is_quote_status', 'retweet_count', 'favorite_count', 'favorited', 'retweeted', 'lang', 'possibly_sensitive'.

Example :

```
# Select a specific user by using a twitter user ID.
user_id=input("Enter a Twitter user ID : ")
no_of_tweets=int(input("How many tweets you want ? "))

# To get tweets details such as tweet ID,
tweets=api.user_timeline(screen_name=user_id, count=no_of_tweets, include_rts=False,
tweet_mode="extended")
```

```
for tweet_info in tweets:
    print("Tweet ID : ",tweet_info.id) # Tweet ID
    print("Created at : ",tweet_info.created_at) # Date on which tweet is created.
    print("Tweet : ",tweet_info.full_text) # Tweet
    print("Retweet count : ",tweet_info.retweet_count) # Number of retweets on each
tweet.
    print("\n")
```

Output : (Example)

```
Enter a Twitter user ID : SadhguruJV

How many tweets you want ? 5
Tweet ID :  1502151438419464196
Created at :  2022-03-11 05:16:26+00:00
Tweet :  Sir Vivian Richards &amp; Lord Ian Botham - a joy to meet you during my
Antigua visit for the #SaveSoil movement. Your achievements in cricket &amp;
beyond are commendable. Please join me in restoring our world’s Soil, the basis
of all Life on Earth. -Sg @ivivianrichards @BeefyBotham https://t.co/M53Ckhu0Lg
Retweet count :  1736


Tweet ID :  1502113329103487012
Created at :  2022-03-11 02:45:00+00:00
Tweet :  Kriya Yoga requires nothing but dedication towards the practice. As you
refine your energies, there is no way you can remain untransformed.
#SadhguruQuotes https://t.co/byjrSIld2u
Retweet count :  1696


Tweet ID :  1501771371562471426
Created at :  2022-03-10 04:06:11+00:00
Tweet :  Congratulations @CISFHQrs for your courageous &amp; committed
contribution to Nation Building for more than five decades. Bharat is proud
&amp; grateful for your stellar service. May you continue to inspire Peace &amp;
Prosperity. Best Wishes. –Sg #CISFRaisingDay2022
Retweet count :  1595


Tweet ID :  1501750941187551232
Created at :  2022-03-10 02:45:00+00:00
Tweet :  You cannot change the past. You can only experience the present moment.
The future must be crafted the way you want. #SadhguruQuotes
https://t.co/eTCAmU3gOl
Retweet count :  2510


Tweet ID :  1501624364889825281
Created at :  2022-03-09 18:22:02+00:00
Tweet :  Machel, #VelliangiriMountains are a Cascade of Grace. Their Power has
empowered millions &amp; will continue to empower future populations. Wonderful
your #Sadhanapada culminated here; it was beautiful to have you &amp; Renee.
Journey on- sing, dance, also transform lives. Blessings. –Sg
https://t.co/y2qV6EBM2k
Retweet count :  1483
```

❖ **Visualizing Twitter Data :** We can visualize the twitter data in multiple ways on the basis of attributes returned by Twitter API.

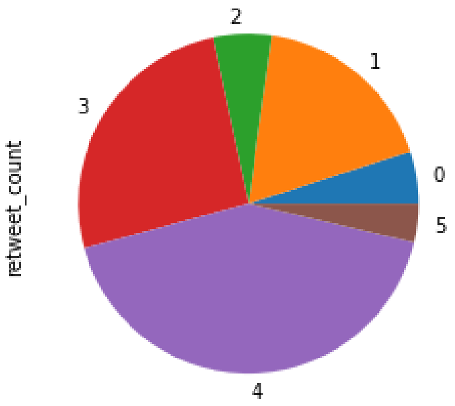
Example : To visualize the number of re-tweets on each tweet :

First create a DataFrame so that it will become easy to get the attributes of Twitter API. Then create a plot (e.g. Pie Plot) to get the number of re-tweets on each tweet.

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

auth=tweepy.OAuth2BearerHandler("Your Bearer Token")
api=tweepy.API(auth)

# Select a specific user by using a twitter user ID.
user_id=input("Enter a Twitter user ID : ")
no_of_tweets=int(input("How many tweets you want ? "))

# To get tweets details such as tweet ID,
tweets=api.user_timeline(screen_name=user_id, count=no_of_tweets, include_rts=False,
tweet_mode="extended")

# Take lists to store different data to create a DataFrame.
tweet_id=[]
tweet_created_at=[]
tweet_full_text=[]
tweet_retweet_count=[]
tweet_favorite_count=[]

for tweet_info in tweets:
    tweet_id.append(tweet_info.id)
    tweet_created_at.append(tweet_info.created_at)
    tweet_full_text.append(tweet_info.full_text)
    tweet_retweet_count.append(tweet_info.retweet_count)
    tweet_favorite_count.append(tweet_info.favorite_count)

twitter_data={'id':tweet_id,'created_at':tweet_created_at,'full_text':tweet_full_text,'retweet_count':tweet_retweet_count,'favorite_count':tweet_favorite_count}
# DataFrame
twitter_dataframe=pd.DataFrame(twitter_data)
# Plotting Pie Graph for retweets on each tweet.
twitter_dataframe['retweet_count'].plot.pie()
plt.show()
```

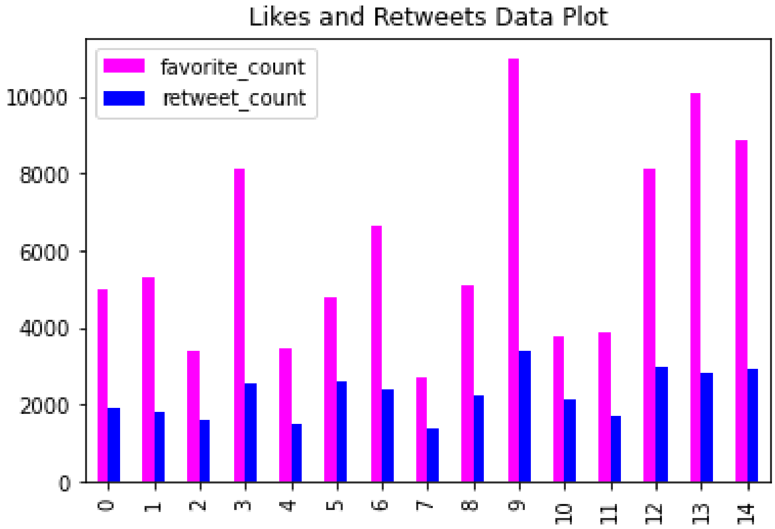
Output :

```
Enter a Twitter user ID : Tesla
How many tweets you want ? 10
```



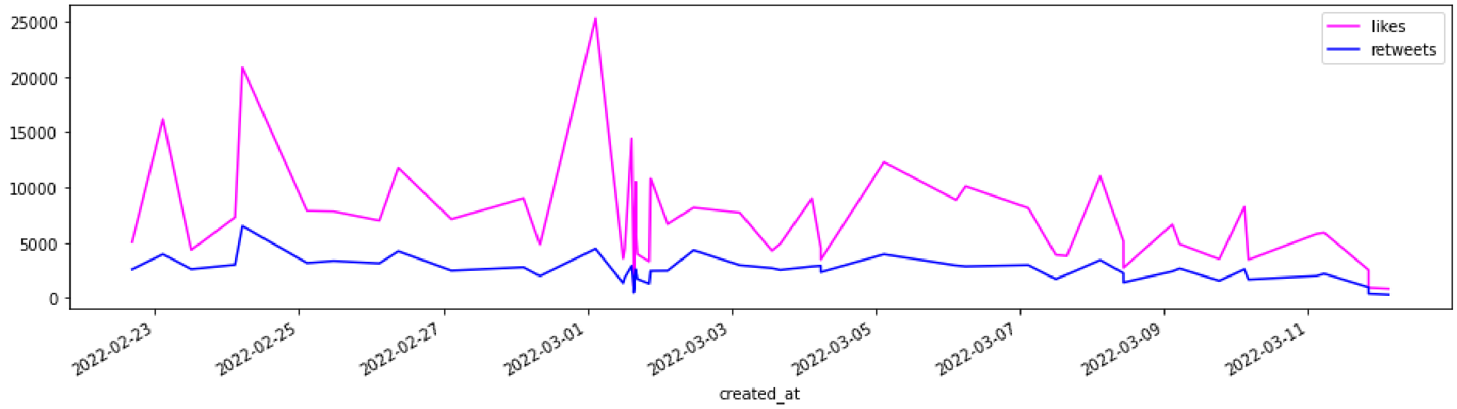
Now to plot the likes and re-tweets received on each tweet, add the following script :

```
# Plot for likes and retweets.
twitter_dataframe.plot.bar(y=['favorite_count','retweet_count'],color=['magenta','blue']
)
plt.show()
```



Consider the following script to plot the Time Series for likes and re-tweets along with dates on which the tweets were published.

```
# Time Series
time_likes=pd.Series(data=twitter_dataframe['favorite_count'].values,index=twitter_dataf
rame['created_at'])
time_likes.plot(figsize=(16,4),label="likes",legend=True,color="magenta")

time_retweets=pd.Series(data=twitter_dataframe['retweet_count'].values,index=twitter_dat
aframe['created_at'])
time_retweets.plot(figsize=(16,4),label="retweets",legend=True,color="blue")

plt.show()
```



✓ You can also get information regarding tweets as total number of likes and re-tweets on each tweet, which tweet has maximum count of likes and got maximum re-tweets.

```
for tweet_info in tweets:
    print("Tweet ID : ",tweet_info.id) # Tweet ID
    print("Created at : ",tweet_info.created_at) # Date on which tweet is created.
    print("Tweet : ",tweet_info.full_text) # Tweet
    print("Retweet count : ",tweet_info.retweet_count) # Number of retweets on each 
tweet.
    print("Favorite count : ",tweet_info.favorite_count)
    print("\n")

# To find total number of tweets, likes and retweets on all tweets.
print("Total number of tweets : ",no_of_tweets)
print("Total number of likes on each tweet : 
",twitter_dataframe['favorite_count'].sum())
print("Total number of retweets on each tweet : 
",twitter_dataframe['retweet_count'].sum())

# To find the number of likes for the most liked tweet.
max_liked_tweet=twitter_dataframe['favorite_count'].max()
print("Number of likes for most liked tweet : ",max_liked_tweet)

# To find the number of retweets for the most retweeted tweet.
max_retweeted_tweet=twitter_dataframe['retweet_count'].max()
print("Number of retweets for the most retweeted tweet : ",max_retweeted_tweet)

# Most liked tweet text.
most_liked_tweet=twitter_dataframe[twitter_dataframe['favorite_count']==twitter_datafram
e['favorite_count'].max()]
print("Most Liked Tweet : ")
print(most_liked_tweet['full_text'])

# Most retweeted tweet text.
most_retweeted_tweet=twitter_dataframe[twitter_dataframe['retweet_count']==twitter_dataf
rame['retweet_count'].max()]
print("Most Retweeted Tweet : ")
print(most_retweeted_tweet['full_text'])
```

Output : (Example)

```
Enter a Twitter user ID : SadhguruJV
How many tweets you want ? 5
Total number of tweets :  5
Total number of likes on each tweet :  16863
Total number of retweets on each tweet :  6180
Number of likes for most liked tweet :  5964
Number of retweets for the most retweeted tweet :  2207

Tweet ID :  1502475716935442442
Created at :  2022-03-12 02:45:00+00:00
Tweet :  Only if you invest your emotions in what matters to you, will life become
powerful and really meaningful. #SadhguruQuotes https://t.co/EWJ2Aneqps
Retweet count :  447
Favorite count :  1247

Tweet ID :  1502375448885432326
Created at :  2022-03-11 20:06:34+00:00
Tweet :  #SaveSoil #MoU #CARICOM
@GastonBrowne @AntiguaOpm @SkerritR @PhilipJPierreLC @pmharriskn @antiguagov
@SaintLuciaGov @skngov @molwynjoseph @SamMarshallMP @machelmontano @armandarton
@GlobalCitizenFo @cpsavesoil @PMOIndia https://t.co/RMXpcgW12d
Retweet count :  461
Favorite count :  1026

Tweet ID :  1502375423451164672
Created at :  2022-03-11 20:06:28+00:00
Tweet :  A historic moment marked by the first #SaveSoil MoUs signed by the pearls of
the ocean. Governments of Antigua &amp; Barbuda, Dominica, St Lucia, and St Kitts &amp;
Nevis — may your commitment to soil revitalization be an inspiration to the rest of the
world. -Sg  @CARICOMorg #CARICOM https://t.co/0glWuMlFBy
Retweet count :  1074
Favorite count :  2806

Tweet ID :  1502151438419464196
Created at :  2022-03-11 05:16:26+00:00
Tweet :  Sir Vivian Richards &amp; Lord Ian Botham - a joy to meet you during my Antigua
visit for the #SaveSoil movement. Your achievements in cricket &amp; beyond are
commendable. Please join me in restoring our world’s Soil, the basis of all Life on
Earth. -Sg @ivivianrichards @BeefyBotham https://t.co/M53Ckhu0Lg
Retweet count :  2207
Favorite count :  5964

Tweet ID :  1502113329103487012
Created at :  2022-03-11 02:45:00+00:00
Tweet :  Kriya Yoga requires nothing but dedication towards the practice. As you refine
your energies, there is no way you can remain untransformed. #SadhguruQuotes
https://t.co/byjrSIld2u
Retweet count :  1991
Favorite count :  5820

Most Liked Tweet :
3    Sir Vivian Richards &amp; Lord Ian Botham - a ...
Name: full_text, dtype: object

Most Retweeted Tweet :
3    Sir Vivian Richards &amp; Lord Ian Botham - a ...
Name: full_text, dtype: object
```

❖ **Sentiment Analysis on Twitter Data :** We can also perform sentiment analysis on gathered twitter data. Here two libraries will be needed, i.e. TextBlob and Vader.

1. **textblob :** It is a Python library which is used for processing textual data. It is built on top of NLTK module and offers a simple API to access its methods to perform basic Natural Language Processing tasks. To install textblob, use the following command :

   pip install textblob

2. **VADER** description has already been given in previous topic for Sentiment Analysis using NLTK.

Import the necessary libraries as :

```
import tweepy
import pandas as pd
import matplotlib.pyplot as plt
import textblob
from nltk.sentiment.vader import SentimentIntensityAnalyzer
```

Now perform Twitter API authentication.

```
#Twitter API Authentication.
auth=tweepy.OAuth2BearerHandler("Your Bearer Token")
api=tweepy.API(auth)
```

Perform Sentiment Analysis on Twitter data.

```
# Sentiment Analysis
def sentiment_percentage(tweet_emotion_part,num_tweets):
    return 100*float(tweet_emotion_part)/float(num_tweets)

# Get the tweets on the basis of Hash Tags or Keywords.
search_tag=input("Enter the Hash Tag or Keyword for which you want to get the tweets : ")
no_of_tweets=int(input("How many tweets you want ? "))

# Iterate over the tweets.
tweets=tweepy.Cursor(api.search_tweets, q=search_tag).items(no_of_tweets)

positive_tweets=0
negative_tweets=0
neutral_tweets=0
polarity_of_tweets=0

# Lists to store positive, negative and neutral tweets.
tweets_list=[]
positive_tweets_list=[]
negative_tweets_list=[]
neutral_tweets_list=[]

for tweet in tweets:
    tweets_list.append(tweet.text)
    analysis=textblob.TextBlob(tweet.text)
    polarity_score=SentimentIntensityAnalyzer().polarity_scores(tweet.text)
    negative_score=polarity_score['neg']
    positive_score=polarity_score['pos']
    neutral_score=polarity_score['neu']
    compound_score=polarity_score['compound']

    polarity_of_tweets+=analysis.sentiment.polarity

    if negative_score>positive_score:
        negative_tweets_list.append(tweet.text)
        negative_tweets+=1
    elif positive_score>negative_score:
        positive_tweets_list.append(tweet.text)
        positive_tweets+=1
    elif positive_score==negative_score:
        neutral_tweets_list.append(tweet.text)
        neutral_tweets+=1

positive_tweets=sentiment_percentage(positive_tweets,no_of_tweets)
negative_tweets=sentiment_percentage(negative_tweets,no_of_tweets)
neutral_tweets=sentiment_percentage(neutral_tweets,no_of_tweets)
polarity_of_tweets=sentiment_percentage(polarity_of_tweets,no_of_tweets)

positive_tweets=format(positive_tweets,'.1f')
negative_tweets=format(negative_tweets,'.1f')
neutral_tweets=format(neutral_tweets,'.1f')
```

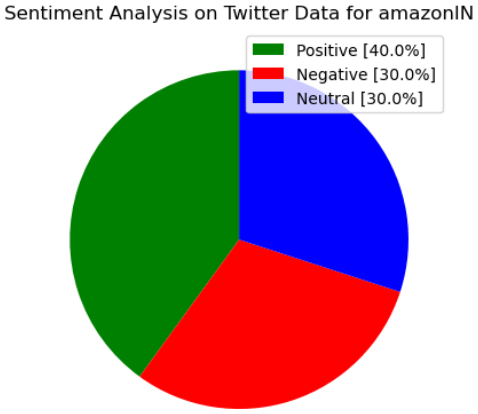
Plot the analyzed data.

```
# Printing Positive, Negative and Neutral Tweets.
print("Positive Tweets : ")
print(positive_tweets_list)

print("Negative Tweeets : ")
print(negative_tweets_list)

print("Neutral Tweets : ")
print(neutral_tweets_list)

# Plotting the data of Sentiment Alalysis.
labels=['Positive ['+str(positive_tweets)+'%]','Negative
['+str(negative_tweets)+'%]','Neutral ['+str(neutral_tweets)+'%]']
size=[positive_tweets,negative_tweets,neutral_tweets]
section_colors=['green','red','blue']
path,text=plt.pie(size,colors=section_colors,startangle=90)
plt.style.use('default')
plt.legend(labels)
plt.title("Sentiment Analysis on Twitter Data for "+search_tag)
plt.show()
```

Output :

```
Enter the Hash Tag or Keyword for which you want to get the tweets : amazonIN

How many tweets you want ? 10

Positive Tweets :
```

```
["Back to my weekend's favourite activity. Some insights\n\n56 days,\n12
emails\n&gt;30 calls.\n\nAnd the amazing collaboratio… https://t.co/38JoPHM0aw",
'@amazonIN  i am unable to rest my Amazon password and i am trying to call
180030001593 on this number but every tim… https://t.co/QNEeqtA1cm', 'RT
@amazonIN: Soundbar Days is back with exciting offers &amp; great discount from
popular brands! Get up to 55% off on bestselling soundbars,…', '@amazonIN
@amazon @AmitAgarwal \nAmazon app is not performing well like add to cart, save
later , move to cart  obse… https://t.co/DPnt2Zg2pL']

Negative Tweeets :

["RT @AnandVe82668274: I'm unable to reapply and cancel my application. please
solve this error ASPS. @amazonIN @AmazonHelp @ICICIBank @ICICI…", "I'm unable to
reapply and cancel my application. please solve this error ASPS. @amazonIN
@AmazonHelp @ICICIBank… https://t.co/4P6fjkTuTE", 'RT @IqooInd: Switching back
to black for the extraordinary and unmatched tempting looks.\nAnd that’s just
the starting point when it comes t…']

Neutral Tweets :

['@indusos @amazonIN □□□□\n App is More Than an App for
Me.\n□□□□□□□□□□□\n\n#Giveaway\n#AppSeBhiZyada #Giveaways #Contest…
https://t.co/LkIcqke37I', '@IqooInd -&gt; Qualcomm® Snapdragon™ 888+ 5G Mobile
Platform. \n#iQOORaidNights \n@IqooInd @iqooesports \n@amazonIN',
'@motorolaindia The #MotorolaEdge30Pro have the     indias beast&amp;fastest
snapdragon 8 Gen1 processor… https://t.co/yrhrQNLUFY']
```



Downloading the twitter datasets online : The online available datasets containing Twitter data can be downloaded and different analytics can be performed on it.

Example : https://www.kaggle.com/crowdflower/twitter-user-gender-classification

Self-Activity : Download and analyze the data using above link and apply different analytics techniques on it.

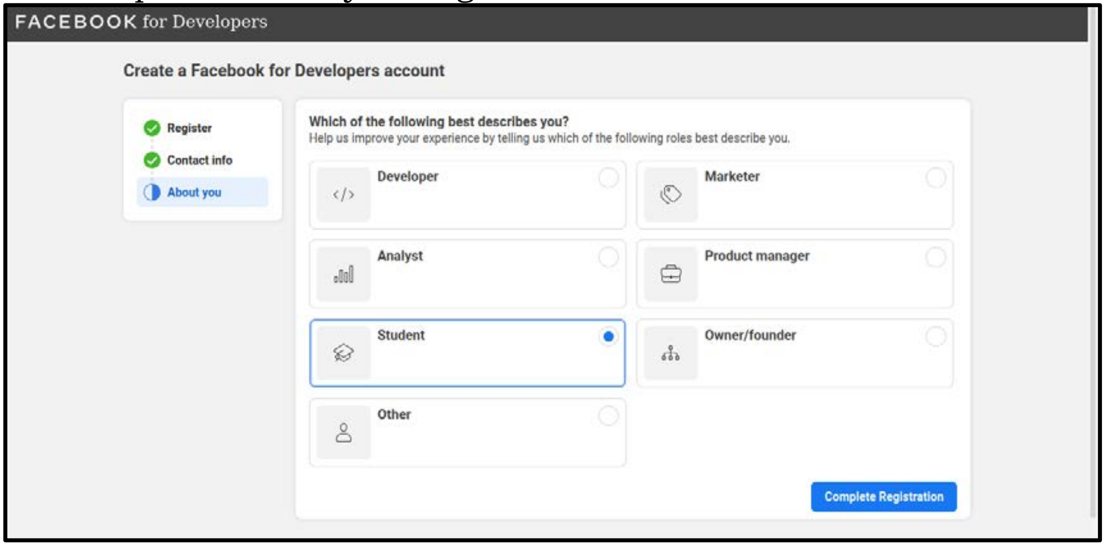
Facebook Data Analysis :

✓ To get Facebook data to perform analysis, there are multiple ways. Some of these are :

- o Getting data using Facebook Access Token.
- o Downloading data directly from a Facebook Account.
- o Getting Facebook data from online available datasets on Kaggle.

Getting data using Facebook Access Token

✓ For this, Facebook's developer account is needed.
✓ Visit the link : https://developers.facebook.com/
✓ Click on **Get Started**.
✓ Then provide the Email ID you want to associate with your Facebook Developer account and click on **Send Verification Email**.
✓ Or you can also proceed with your registered Email ID with Facebook.



FACEBOOK for Developers

Create a Facebook for Developers account

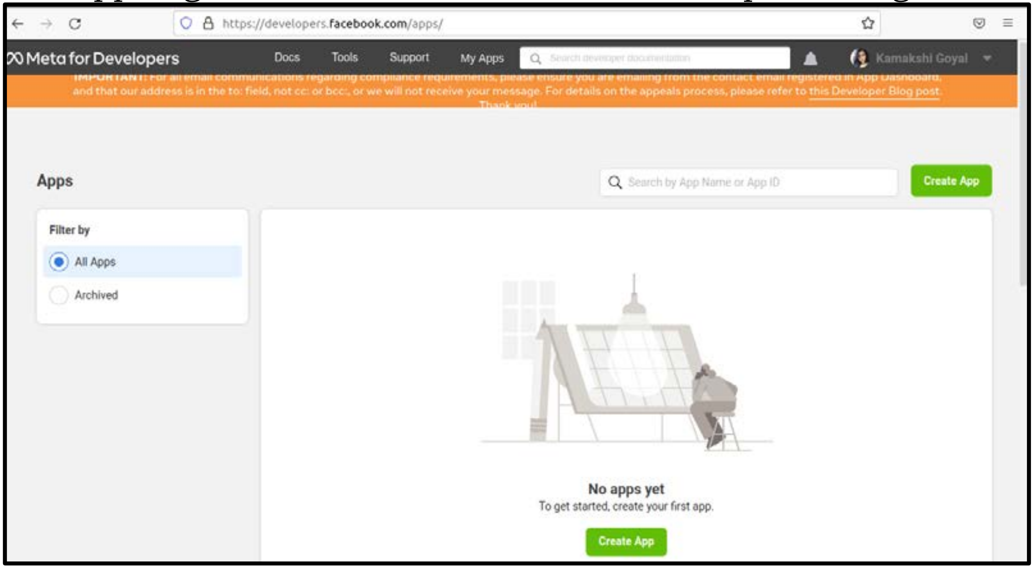
✓ Register
✓ Contact info
About you

Which of the following best describes you?
Help us improve your experience by telling us which of the following roles best describe you.

○ Developer
○ Marketer
○ Analyst
○ Product manager
● Student
○ Owner/founder
○ Other

Complete Registration

✓ Now create an app to get the token to be used for further processing.



https://developers.facebook.com/apps/

Meta for Developers — Docs — Tools — Support — My Apps — Search developer documentation — Kamakshi Goyal

IMPORTANT: For all email communications regarding compliance requirements, please ensure you are emailing from the contact email registered in App Dashboard, and that our address is in the to: field, not cc: or bcc:, or we will not receive your message. For details on the appeals process, please refer to this Developer Blog post. Thank you!

Apps

Search by App Name or App ID

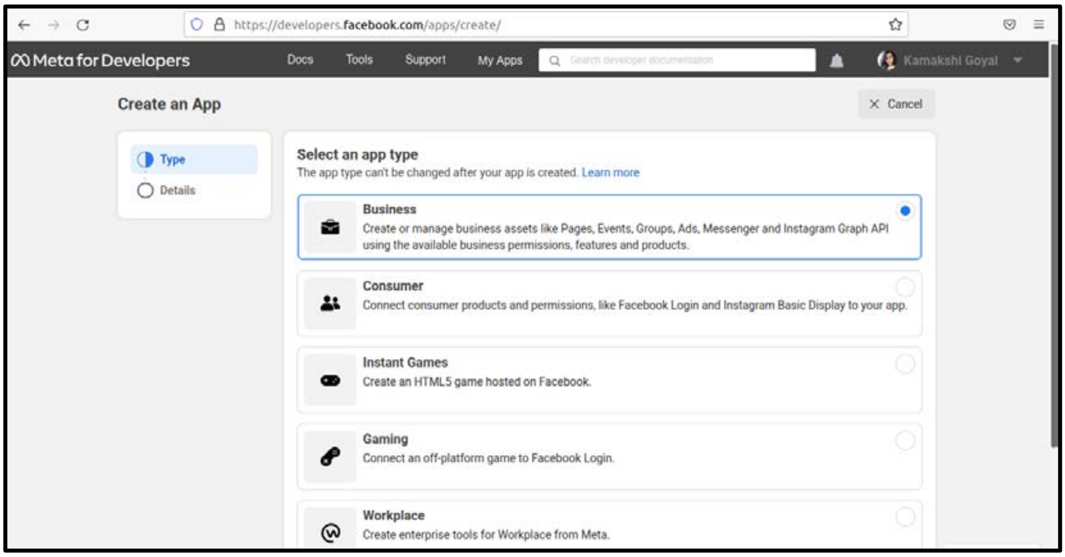
Create App

Filter by
● All Apps
○ Archived

No apps yet
To get started, create your first app.

Create App

✓ Click on **Create App** and then select the type of app to be created. Multiple options will be available i.e. Business, Consumer, Instant Games, Gaming, Workplace, None. You can read the details and select an option. If **Business** option available in list is selected, then it creates an app which manages business assets like Pages, Events, Groups, Ads, Messenger and Instagram Graph API using the available business permissions, features and products.


https://developers.facebook.com/apps/create/
Meta for Developers
Docs
Tools
Support
My Apps
Search developer documentation
Kamakshi Goyal
Cancel
Type
Details

Create an App

Select an app type

The app type can't be changed after your app is created. Learn more

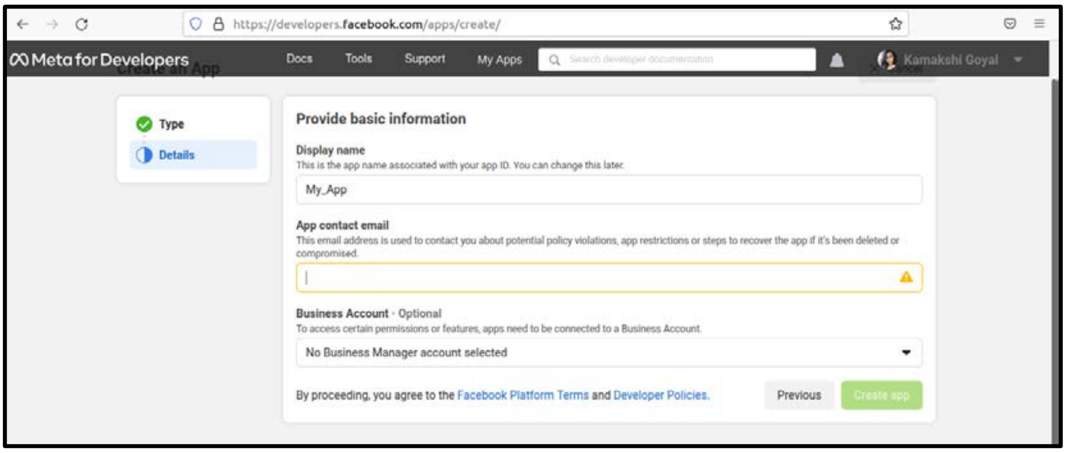
● Business
Create or manage business assets like Pages, Events, Groups, Ads, Messenger and Instagram Graph API using the available business permissions, features and products.

○ Consumer
Connect consumer products and permissions, like Facebook Login and Instagram Basic Display to your app.

○ Instant Games
Create an HTML5 game hosted on Facebook.

○ Gaming
Connect an off-platform game to Facebook Login.

○ Workplace
Create enterprise tools for Workplace from Meta.



Provide basic information

Display name
This is the app name associated with your app ID. You can change this later.
My_App

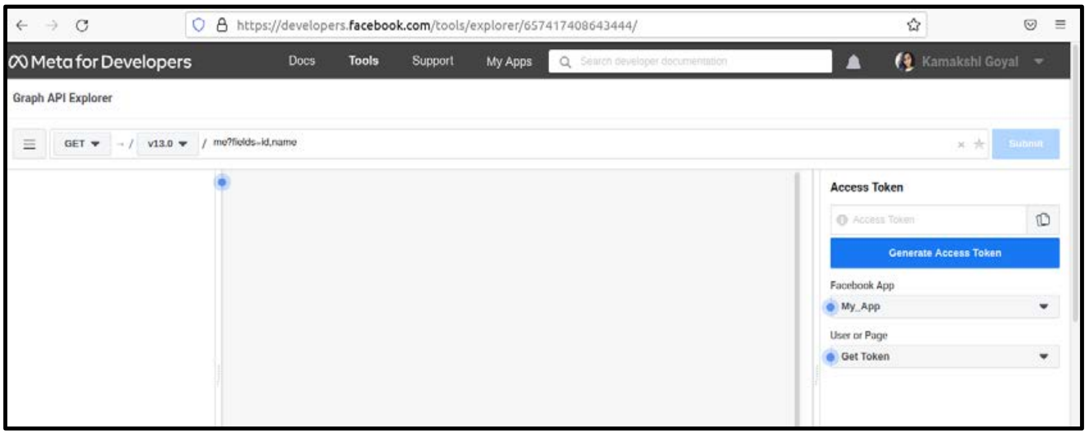
App contact email
This email address is used to contact you about potential policy violations, app restrictions or steps to recover the app if it's been deleted or compromised.

Business Account · Optional
To access certain permissions or features, apps need to be connected to a Business Account.
No Business Manager account selected

By proceeding, you agree to the Facebook Platform Terms and Developer Policies.

✓ After an app gets created, you can get the Access Token as follows :

- Go to : https://developers.facebook.com/tools/explorer/



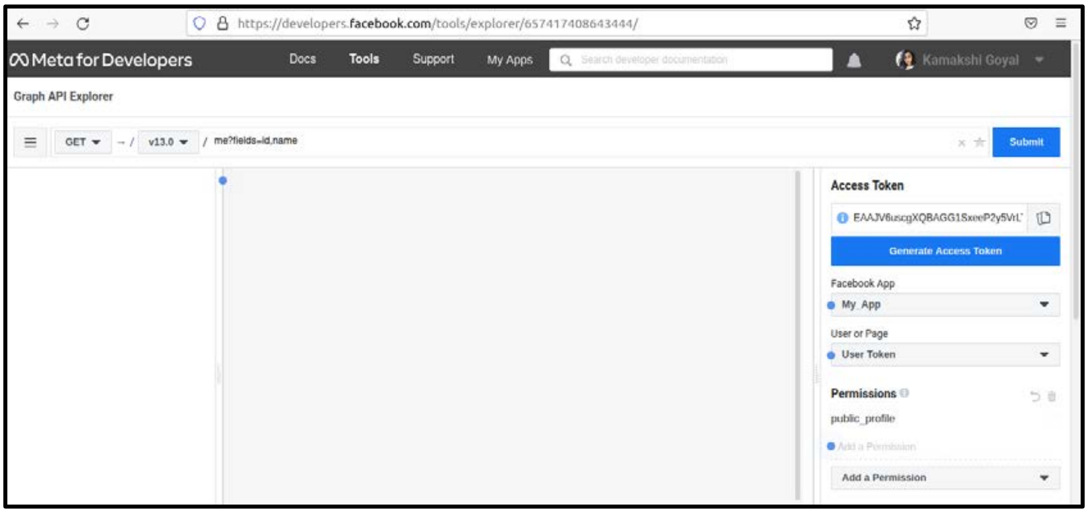
me?fields=id,name

Access Token

Facebook App: My_App

User or Page: Get Token

- Now click on **Generate Access Token**. Proceed to the next step by clicking on **Continue**.
- The **Access Token** will be visible in Access Token input box.



https://developers.facebook.com/tools/explorer/657417408643444/

Graph API Explorer

GET ▾ → / v13.0 ▾ / me?fields=id,name

Access Token: EAAJV6uscgXQBAGG1SxeeP2y5VrL

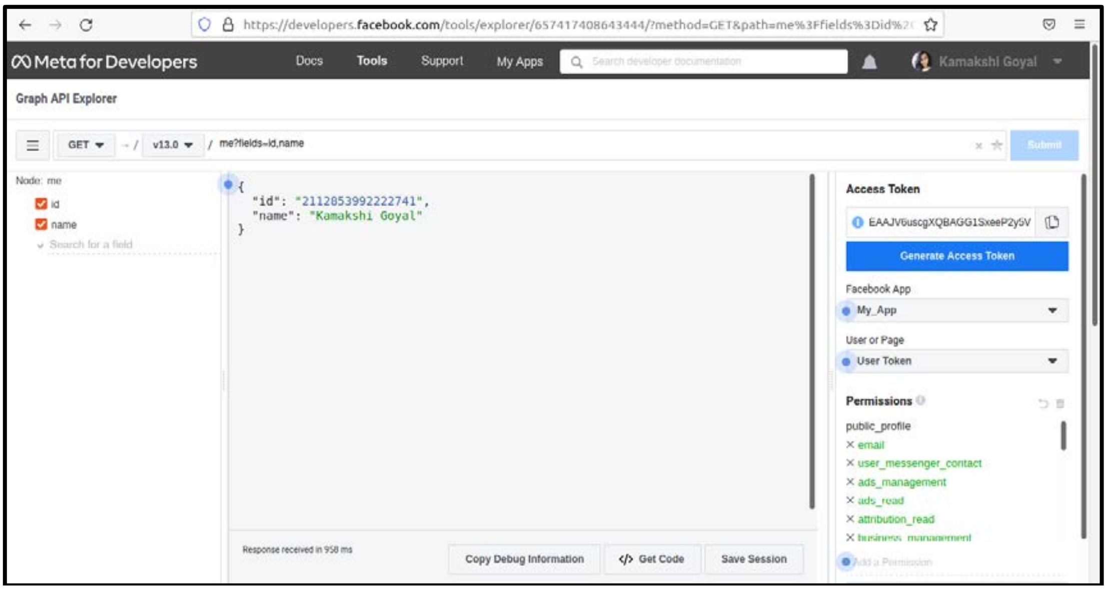
Facebook App: My_App

User or Page: User Token

Permissions: public_profile

Add a Permission

- Now allow the necessary permissions to access the Facebook pages as well.
- Click on “Add a Permission” dropdown and select the permissions from it.



https://developers.facebook.com/tools/explorer/657417408643444/?method=GET&path=me%3Ffields%3Did%2

Graph API Explorer

GET ▾ → / v13.0 ▾ / me?fields=id,name

Node: me

☑ id

☑ name

```
{
  "id": "2112853992222741",
  "name": "Kamakshi Goyal"
}
```

Response received in 958 ms

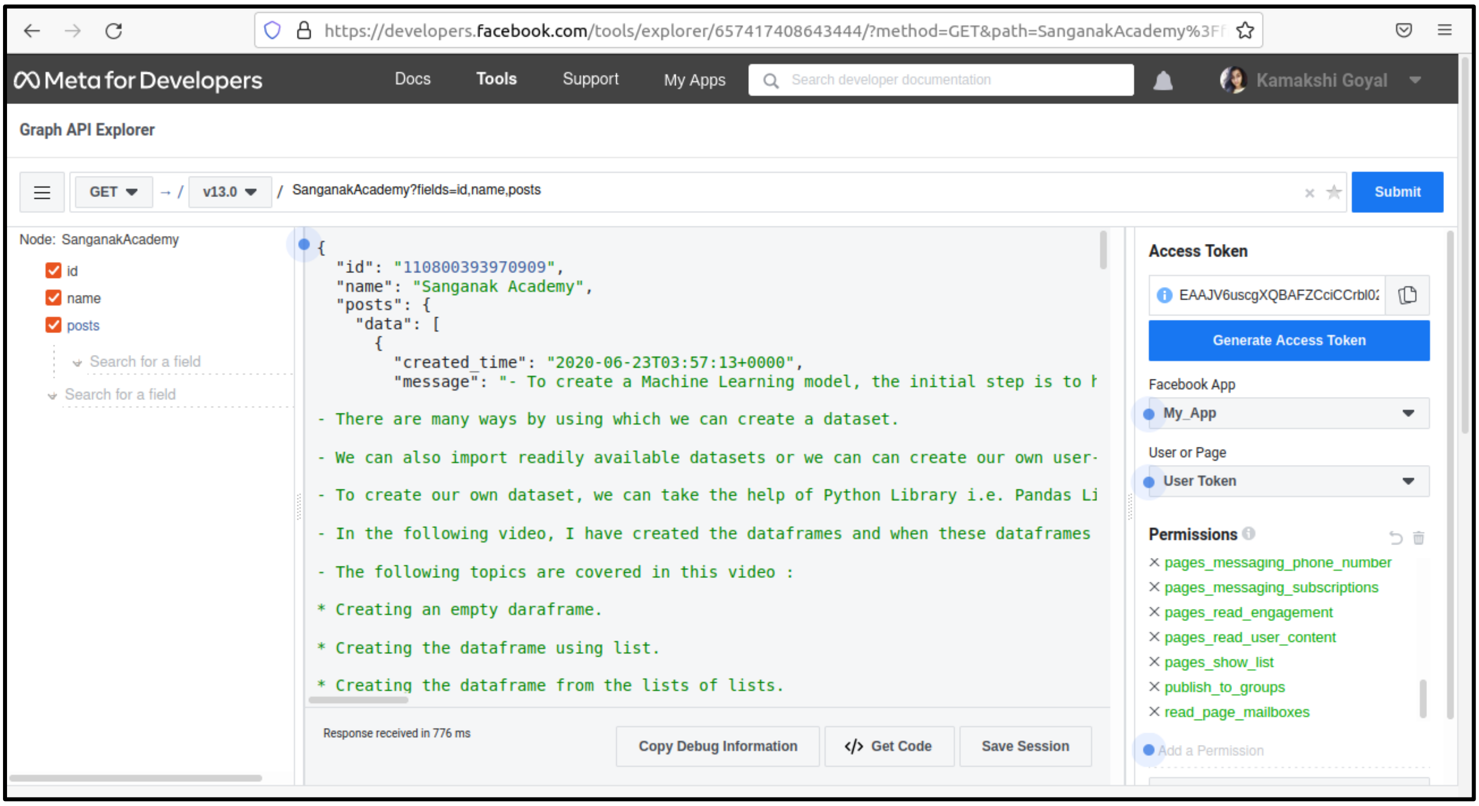
Access Token: EAAJV6uscgXQBAGG1SxeeP2y5V

Facebook App: My_App

User or Page: User Token

Permissions: public_profile, ✕ email, ✕ user_messenger_contact, ✕ ads_management, ✕ ads_read, ✕ attribution_read, ✕ business_management

- From “User or Pages” dropdown select “Get User Token” again to get the Token with revised permissions.
- Now if we want to see the details of publically available Facebook users or pages, change the request in the request url box.
  Example : If we want to get the details of Facebook Page “Sanganak Academy” then change the name of page as : **SanganakAcademy?fields=id,name**. Before that, make sure to allow the permissions for accessing that page as well.
- You can see the posts on this page as well by changing the url as : **SanganakAcademy?fields=id,name,posts**. Same you can do to access other information as well.



GET → / v13.0 / SanganakAcademy?fields=id,name,posts

Node: SanganakAcademy

☑ id
☑ name
☑ posts

```
{
  "id": "110800393970909",
  "name": "Sanganak Academy",
  "posts": {
    "data": [
      {
        "created_time": "2020-06-23T03:57:13+0000",
        "message": "- To create a Machine Learning model, the initial step is to h

- There are many ways by using which we can create a dataset.

- We can also import readily available datasets or we can can create our own user-

- To create our own dataset, we can take the help of Python Library i.e. Pandas Li

- In the following video, I have created the dataframes and when these dataframes

- The following topics are covered in this video :

* Creating an empty daraframe.

* Creating the dataframe using list.

* Creating the dataframe from the lists of lists.
```

Response received in 776 ms

Access Token: EAAJV6uscgXQBAFZCciCCrbl0

Facebook App: My_App

User or Page: User Token

Permissions
× pages_messaging_phone_number
× pages_messaging_subscriptions
× pages_read_engagement
× pages_read_user_content
× pages_show_list
× publish_to_groups
× read_page_mailboxes

✓ The Facebook data can be accessed using Python script as follows :

❖ Import the necessary libraries :

```python
import requests
import time
import pickle
import random
```

❖ Provide Access Token and get the URL to access the data :

```python
# Access Token
access_token="Your Access Token"
# In Graph URL, provide the correct version of Graph API. Here currently I am using v13.0
graphURL="https://graph.facebook.com/v13.0/"
# Request URL to get the relevant data of Facebook Page Sanganak Academy.
# You can access any other page by using its ID as well.
requestURL="SanganakAcademy?fields=id,name,posts{message,created_time,comments.limit(0).summary(true), likes.limit(0).summary(true)}"
actual_url=graphURL+requestURL
```

❖ Call the Graph API using get method of requests library.

```python
result=requests.get(actualURL,{'access_token':access_token})
```

❖ Getting posts from a Facebook page.

```python
# To get all posts.
result=result.json()['posts']
print(result['data'])
# To get individual post
print(result['data'][0])
```

❖ Create a DataFrame using Pandas library.

```python
# Create a dataframe using pandas library
import pandas as pd
fb_dataframe=pd.DataFrame(received_data)
fb_dataframe=pd.json_normalize(received_data)
print(fb_dataframe)
# Columns
print("Columns in Dataframe : ")
for col in fb_dataframe.columns:
     print(col)
```

Here columns in the dataframe are :

- message
- created_time
- id
- comments.data
- comments.summary.order
- comments.summary.total_count
- comments.summary.can_comment
- likes.data
- likes.summary.total_count
- likes.summary.can_like
- likes.summary.has_liked

❖ To find Top Liked Post.

```python
# To find top liked post.
top_liked_post=fb_dataframe[fb_dataframe['likes.summary.total_count']==fb_dataframe['likes.summary.total_count'].max()]
print("Top Liked Post : ")
print(top_liked_post['message'])
```

Similarly we can find the top commented post as well.

❖ Grouping Facebook posts by date on which they are created.

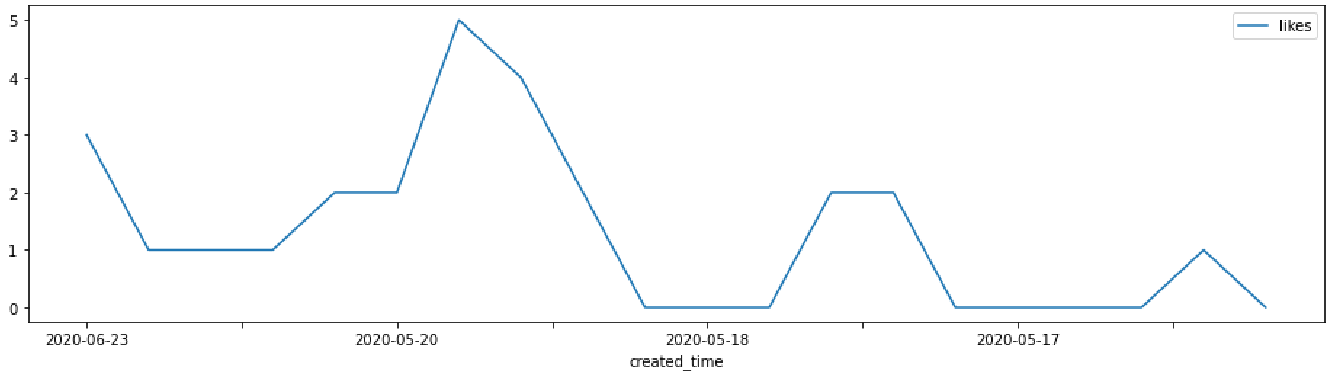
For this, use the dataframe column “created_time”. It contains datetime format data. So date can be obtained by splitting it in two sections i.e. date and time.

```python
# Grouping facebook posts by date.
fbdata_with_dates=pd.DataFrame(fb_dataframe[['created_time','message','likes.summary.total_count','comments.summary.total_count']])
# Split date and time.
date=fbdata_with_dates['created_time'].str.split('T')
fbdata_with_dates['created_time']=date.str[0]
print(fbdata_with_dates)
grouped_data=fbdata_with_dates.groupby('created_time').sum()
```

Visualize the data :

```python
# Visualizing the data.
import matplotlib.pyplot as plt
# Time Series
```

```
time_likes=pd.Series(data=fbdata_with_dates['likes.summary.total_count'].values,index
=fbdata_with_dates['created_time'])
time_likes.plot(figsize=(16,4),label="likes",legend=True)
plt.show()
```

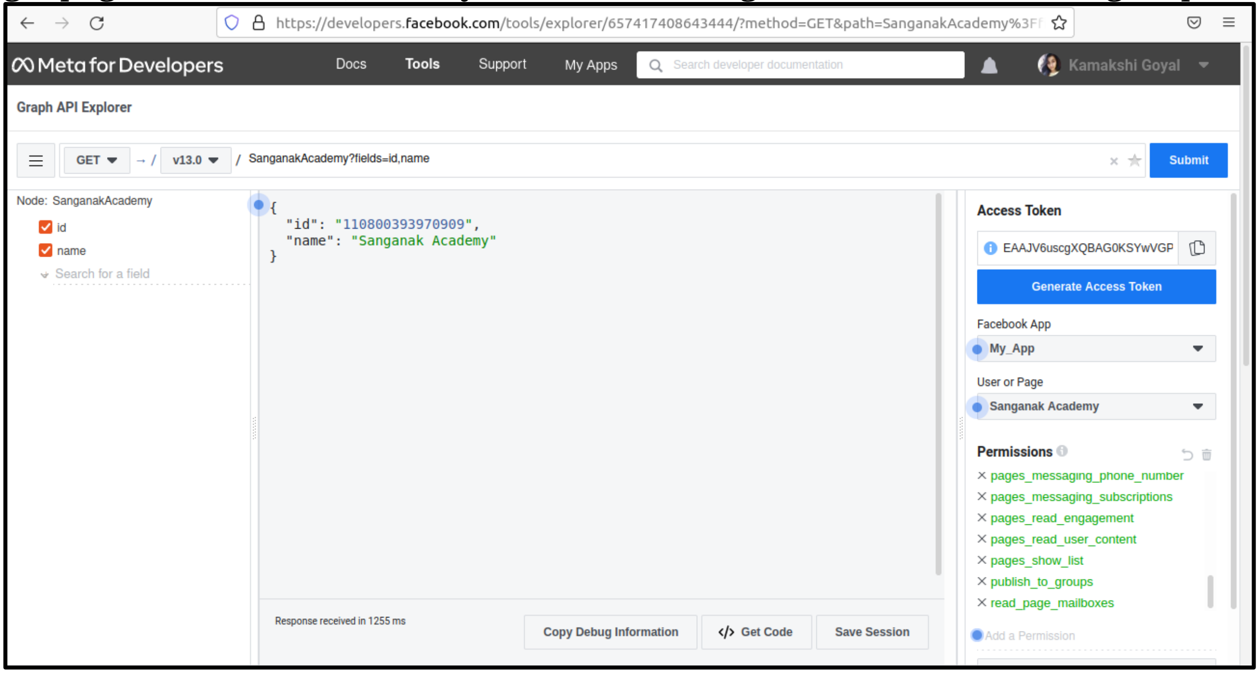


Automating Facebook Programmatically

(**Note :** To analyze the Facebook data, we can use User Access Token, but to add, update or delete something on live Facebook Page, we need Page Access Token. To get page access token, make sure there is a Facebook Page associated with your Facebook Account.)

Steps to get Page Access Token :

✓ To get page access token, select your Facebook Page name on “User or Page” option.



SanganakAcademy?fields=id,name

Node: SanganakAcademy

☑ id

☑ name

```
{
  "id": "110800393970909",
  "name": "Sanganak Academy"
}
```

Response received in 1255 ms

Access Token: EAAJV6uscgXQBAG0KSYwVGP

Facebook App: My_App

User or Page: Sanganak Academy

Permissions

- × pages_messaging_phone_number
- × pages_messaging_subscriptions
- × pages_read_engagement
- × pages_read_user_content
- × pages_show_list
- × publish_to_groups
- × read_page_mailboxes

✓ And then click on “Generate Access Token”. Now copy the generated access token and use it for further analysis.

❖ **Creating a Facebook post and Commenting on it :** Consider the following steps to create a post and comment on a specific Facebook Page on your own account :

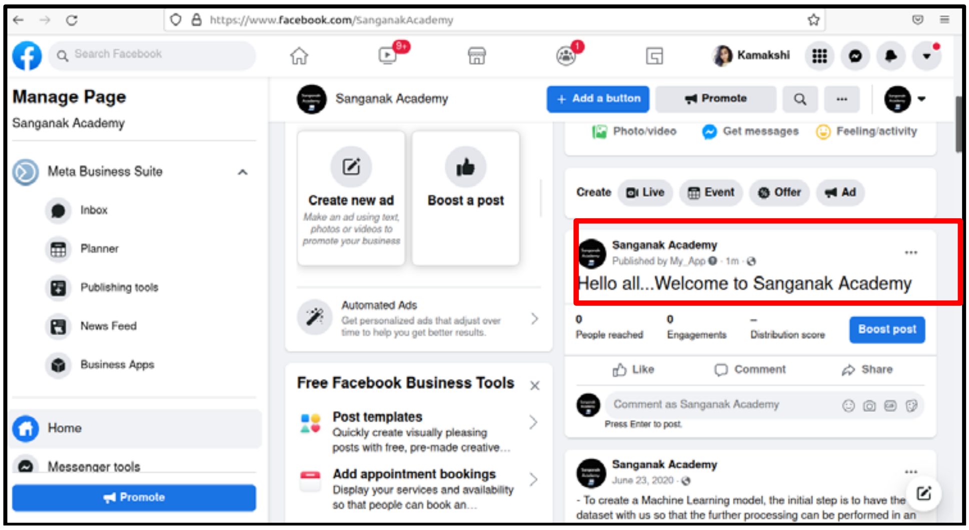
✓ For this, we need **facebook-sdk** library. To install facebook library, use :

```
pip install facebook-sdk
```

✓ **Creating a Facebook post :** To post something on Facebook Page wall, use put_object() method of Facebook Graph API.

```
import facebook
access_token="Your Page Access Token"
```

```
fb=facebook.GraphAPI(access_token)
fb.put_object(parent_object='me', connection_name='feed', message='Hello all...Welcome
to Sanganak Academy')
```



Sanganak Academy
Published by My_App · 1m ·

Hello all...Welcome to Sanganak Academy

Sanganak Academy
June 23, 2020 ·

- To create a Machine Learning model, the initial step is to have the dataset with us so that the further processing can be performed in an

✓ **Commenting on a Facebook post :** To comment on a specific post, "Post ID" and "Facebook Page ID" is needed. To get the Facebook Post ID, go to the post and click on the date on which post is created. Inside the website URL, you will see the post ID at last. Example : https://www.facebook.com/SanganakAcademy/posts/511268930590718

Here, 511268930590718 is the Facebook Post ID. To get "Facebook Page ID", Open the Facebook Page and in **About** section, you will find the Facebook Page ID.

Now combine Facebook Page ID and Facebook Post ID as pageid_postid. For example : If Page ID = 12345 and Post ID = 511268930590718, then combine it as 12345_511268930590718.

Syntax : graph_api_object.put_object(parent_object = 'post_id', connection_name= 'comments', message = 'Your comment')

Consider the following script to comment on a post.

Example :

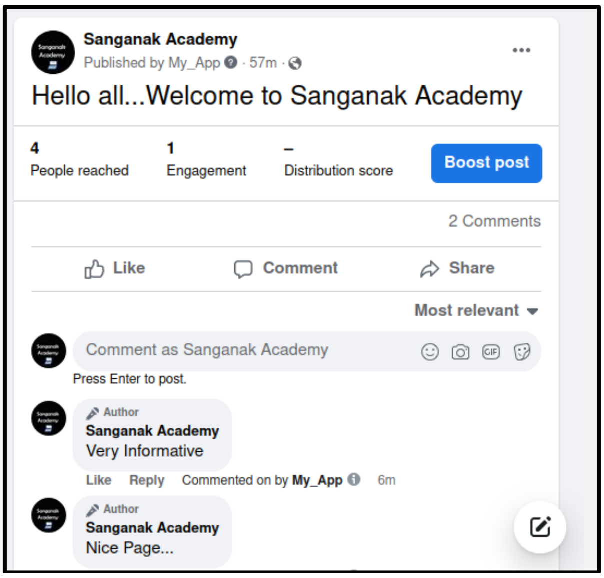
```
# Writing a comment on a facebook post.
fb.put_object(parent_object='12345_511268930590718',connection_name='comments',message='
Nice Page...')
```

You can also comment on a specific post using put_comment() method as :

Syntax : graph_api_object.put_comment(object_id = 'post_id', message = 'Your comment')

Example :

```
# Writing a comment on a facebook post.
fb.put_comment(object_id='12345_511268930590718', message='Very Informative')
```



Sanganak Academy
Published by My_App · 57m ·

Hello all...Welcome to Sanganak Academy

4 People reached
1 Engagement
– Distribution score

2 Comments

Comment as Sanganak Academy
Press Enter to post.

Author
Sanganak Academy
Very Informative
Like Reply Commented on by My_App 6m

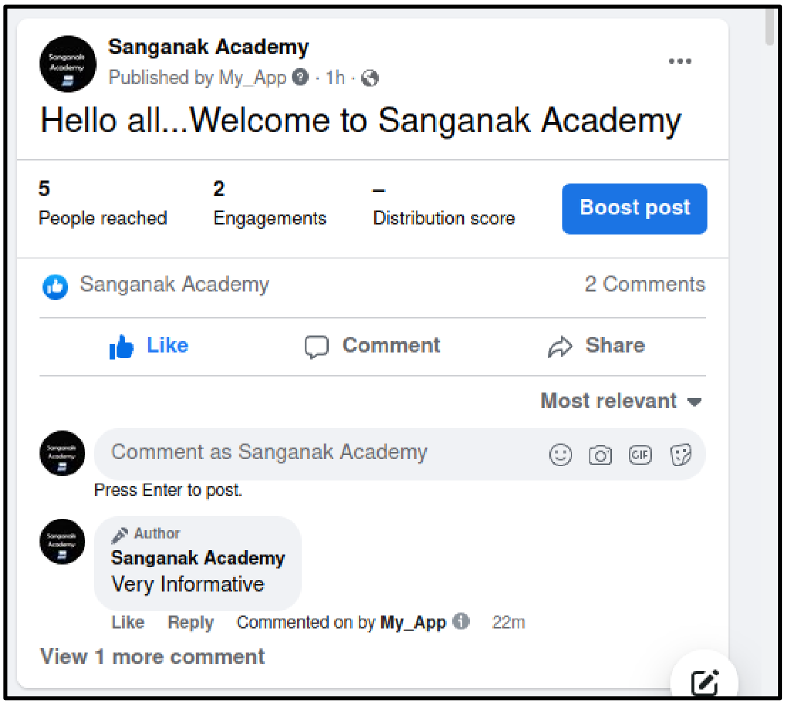
Author
Sanganak Academy
Nice Page...

❖ Liking a post :

Syntax : `graph_api_object.put_like(object_id = ‘post_id’)`

Example :

```
# Liking a facebook post.
fb.put_like(object_id='12345_511268930590718')
```



Sanganak Academy
Published by My_App · 1h ·

Hello all...Welcome to Sanganak Academy

5 People reached
2 Engagements
– Distribution score

Sanganak Academy
2 Comments

Comment as Sanganak Academy
Press Enter to post.

Author
Sanganak Academy
Very Informative
Like Reply Commented on by My_App 22m

View 1 more comment

❖ Deleting a Facebook post :

Syntax : `graph_api_object.delete_object(id = ‘post_id’)`

Example :

```
# Deleting a facebook post.
fb.delete_object(id='12345_511268930590718')
```

The Facebook post will be deleted.

❖ Getting all friends of an active user :

```
# Get the active user's friends.
friends = fb.get_connections(id='me', connection_name='friends')
```

(Refer this https://facebook-sdk.readthedocs.io/en/latest/api.html for more information about Facebook SDK.)

YouTube Data Analysis :

- ✓ YouTube data can be gathered from multiple sources such as :
  - o Using YouTube API.
  - o Downloading already available datasets on Kaggle.

Consider the dataset : https://www.kaggle.com/datasnaek/youtube-new?select=CAvideos.csv

❖ Read the dataset

```
import pandas as pd
youtube_data=pd.read_csv('CAvideos.csv')

# Get the columns in a dataset
print(youtube_data.columns)
```

❖ To find total views, likes, dislikes and comment count

```
# To find total views, likes, dislikes and comment count.
print(youtube_data[['views','likes','dislikes','comment_count']].sum())
```

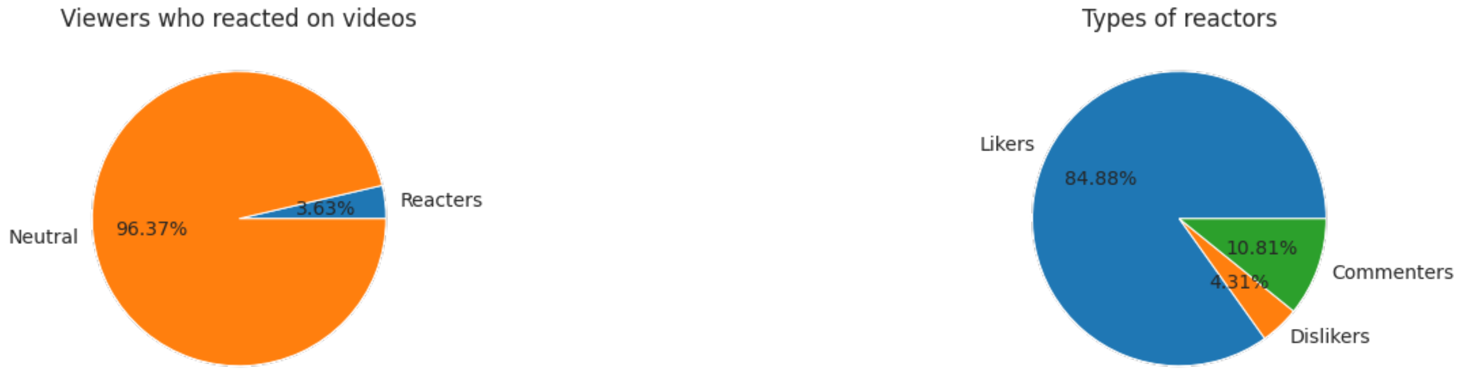
❖ Perform statistical analysis on YouTube data

It will plot the data for :

- Finding the viewers who reacted on videos.
- To classify the reactors on videos as “Likers”, “Dislikers” and “Commenters”.

```
# performing statistical analysis on youtube data.
import seaborn as sns
import matplotlib
import matplotlib.pyplot as plt

sns.set_style('darkgrid')
matplotlib.rcParams['font.size'] = 14
matplotlib.rcParams['figure.figsize'] = (12, 5)
matplotlib.rcParams['figure.facecolor'] = '#00000000'

fig = plt.figure()

axis1 = fig.add_axes([0, 0, 0.75, 0.75], aspect=1) # add_axes([left, bottom, width, height],aspect=1)

# To find viewers who reacted on videos.
pie_vars = ['Reacters','Neutral'];
pie_values =
[youtube_data['likes'].sum()+youtube_data['dislikes'].sum(),youtube_data['views'].sum()-(youtube_data['likes'].sum()+youtube_data['dislikes'].sum())]
axis1.pie(pie_values,labels=pie_vars,autopct='%1.2f%%')
axis1.set_title("Viewers who reacted on videos")
```

```
axis2 = fig.add_axes([0.8, 0, 0.75, 0.75], aspect=1)
# Pie chart for reactors
pie_vars = ['Likers','Dislikers','Commenters']
pie_values =
[youtube_data['likes'].sum(),youtube_data['dislikes'].sum(),youtube_data['comment_count'
].sum()]
axis2.pie(pie_values,labels=pie_vars,autopct='%1.2f%%')
axis2.set_title("Types of reactors")

plt.show()
```



❖ Find top most viewed videos

```
# To find top 5 most viewed videos.
print(youtube_data.sort_values(by='views',ascending=False).head(5))
```

❖ Find least viewed videos

```
# To find top 5 least viewed videos.
print(youtube_data.sort_values(by='views',ascending=True).head(5))
```

❖ Find top most liked videos

```
# To find top 5 most liked videos.
print(youtube_data.sort_values(by='likes',ascending=False).head(5))
```

❖ Find least liked videos

```
# To find top 5 least liked videos.
print(youtube_data.sort_values(by='likes',ascending=True).head(5))
```

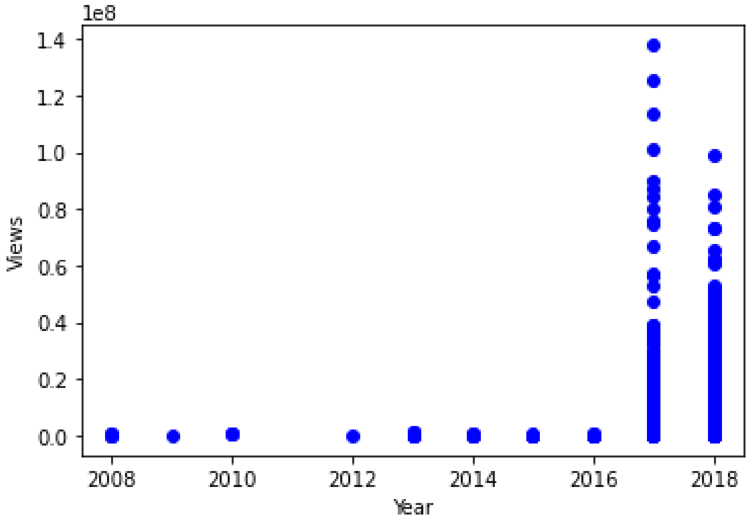
❖ Performing Year wise Statistics

First clean the column where we need to work with date. In above dataset, “publish_time” column contains data in datetime format. To perform statistics year wise or month wise, split the date in separate sections as year, month and day.

```
# Split the date into year, month and date.
import datetime

i=0

for i in range(youtube_data.shape[0]):
    date_time_obj = datetime.datetime.strptime(youtube_data['publish_time'].at[i],'%Y-
%m-%dT%H:%M:%S.000Z')
    youtube_data['publish_time'].at[i] = date_time_obj
    i = i+1

date=[]
year=[]
month=[]
day=[]

for i in range(youtube_data.shape[0]):
    d = youtube_data['publish_time'][i].date()
    y = youtube_data['publish_time'][i].date().year
    m = youtube_data['publish_time'][i].date().month
    days = youtube_data['publish_time'][i].date().day
    date.append(d) # Storing dates
    year.append(y) # Storing years
    month.append(m) # Storing months
    day.append(d) # Storing days
    i = i+1
youtube_data.drop(['publish_time'], inplace=True,axis=1)
youtube_data['publish_time']=date
youtube_data['year']=year
youtube_data['month'] = month
youtube_data['day'] = day
```

Year wise statistics of views :

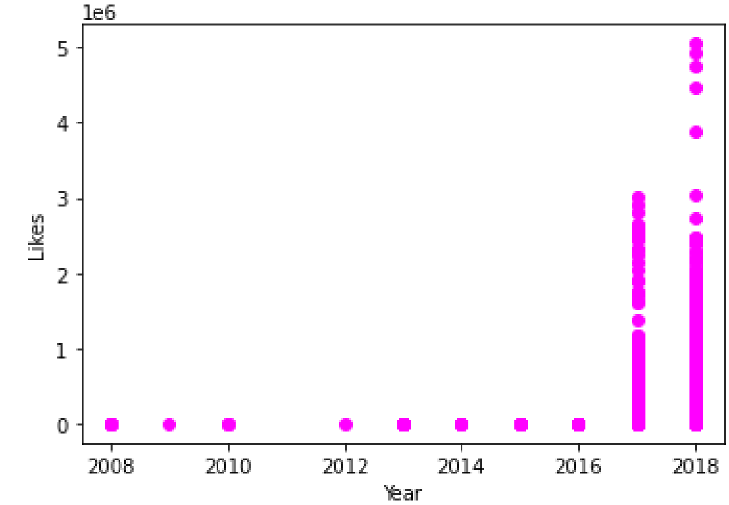
```
# Year wise statistics of views.
plt.scatter(youtube_data['year'], youtube_data['views'], c="red")
plt.xlabel("Year")
plt.ylabel("Views")
plt.show()
```

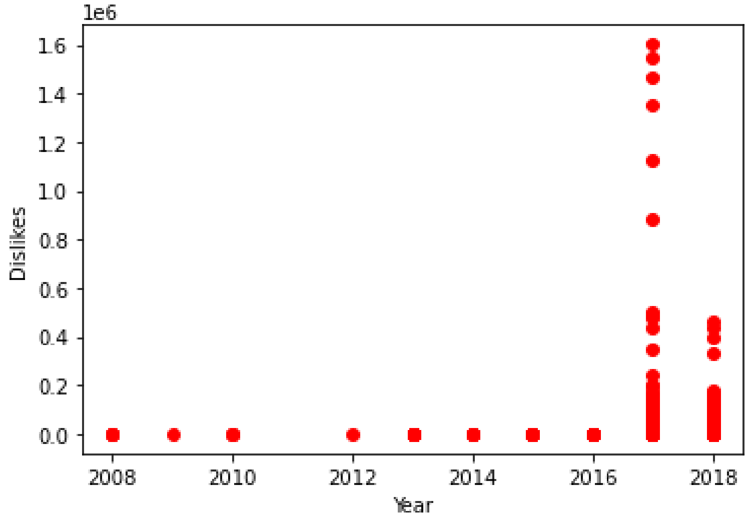


Year wise statistics of likes and dislikes

```
# Year wise statistics of likes.
plt.scatter(youtube_data['year'], youtube_data['likes'], c="magenta")
plt.xlabel("Year")
plt.ylabel("Likes")
plt.show()
```

```
# Year wise statistics of dislikes.
plt.scatter(youtube_data['year'], youtube_data['dislikes'], c="red")
plt.xlabel("Year")
plt.ylabel("Dislikes")
plt.show()
```





Lab Assignments

SET A

1. Consider any text paragraph. Preprocess the text to remove any special characters and digits. Generate the summary using extractive summarization process.
2. Consider any text paragraph. Remove the stopwords. Tokenize the paragraph to extract words and sentences. Calculate the word frequency distribution and plot the frequencies. Plot the wordcloud of the text.
3. Consider the following review messages. Perform sentiment analysis on the messages.

   i. I purchased headphones online. I am very happy with the product.

   ii. I saw the movie yesterday. The animation was really good but the script was ok.

iii. I enjoy listening to music

iv. I take a walk in the park everyday

4. Perform text analytics on WhatsApp data :

Write a Python script for the following :

i. First Export the WhatsApp chat of any group. Read the exported ".txt" file using open() and read() functions.

ii. Tokenize the read data into sentences and print it.

iii. Remove the stopwords from data and perform lemmatization.

iv. Plot the wordcloud for the given data.

Set B

1. Consider the following dataset :

https://www.kaggle.com/datasets/prasertk/top-1000-instagram-influencers

Write a Python script for the following :

i. Read the dataset and find the top 5 Instagram influencers from India.

ii. Find the Instagram account having least number of followers.

iii. Read the column "Category", remove stopwords and plot the wordcloud to find the keywords which will imply that in which category maximum accounts are created.

iv. Group the Instagram accounts category wise.

v. Visualize the dataset and plot the relationship between Followers and Authentic engagement columns.

2. Consider the following dataset :

https://www.kaggle.com/datasets/seungguini/youtube-comments-for-covid19-related-videos?select=covid_2021_1.csv

Write a Python script for the following :

i. Read the dataset and perform data cleaning operations on it.

ii. Tokenize the comments in words.

iii. Perform sentiment analysis and find the percentage of positive, negative and neutral comments.

3 Consider the following dataset :

https://www.kaggle.com/datasets/datasnaek/youtube-new?select=INvideos.csv

Write a Python script for the following :

i. Read the dataset and perform data cleaning operations on it.

ii. Find the total views, total likes, total dislikes and comment count.

iii. Find the least and topmost liked and commented videos.

iv. Perform year wise statistics for views and plot the analyzed data.

v. Plot the viewers who reacted on videos.

Set C

Q.2 Write a Python script to read the Tweets using Twitter API and tweepy library to perform the following tasks :

i. Authenticate Twitter API (Using Bearer Token)

ii. Get the tweets using Keywords or Hash Tags.

iii. Find the total number of likes and retweets on each tweet.

iv. Find the most liked tweet and print its text.

v. Visualize the tweets and plot the time series for likes and retweets along with dates on which tweets are published.

Download the data directly from a Facebook Account

- First log in to your Facebook account.
- Then go to **Settings** -> **Your Facebook information** -> **Download your information**
- Then click on View link to download the data in JSON format. Select what information you want to download.
- You can select multiple options available to download.
- For example, if we want to work with Posts on Facebook, then select posts option and then click on **Request a download** button. You can select any number of information and download it.
- The data will be available in separate folders in JSON format.

1. Import and format the data into a DataFrame using pandas library. Example : For working with Facebook Posts, read the JSON file available in “Posts” folder as :

```
import pandas as pd
facebook_dataframe=pd.read_json("your_posts.json")
```

Similarly you can work with other downloaded data and read the JSON files available in them.

2. Now perform data cleaning operation on created dataframe and remove unnecessary columns.

3. Perform multiple statistical analysis such as finding the posts by date, number of likes on a post, comments on a post.

4. Perform sentiment analysis to find the polarity scores and classify the posts text in three categories i.e. positive, negative and neutral posts.

Signature of the instructor □ Date □

Assignment Evaluation

0: Not done □ 2: Late Complete □ 4: Complete □

1: Incomplete □ 3: Needs improvement □ 5: Well Done □