MACHINE LEARNING AND DATA ANALYTICS LAB [SCSA2601]

CYCLE 1.

1. Simulating a simple calculator
2. Armstrong Series
3. Fibonacci Series
4. Modules and Functions
5. Working with Strings
    a) From the string input count the special characters, alphabets, digits, lowercase and uppercase characters
    b) Print the String "Welcome". Matrix size must be N X M. (N is an odd natural number, and M is 3 times N.). The design should have 'WELCOME' written in the center. The design pattern should only use |, .and - characters.

CYCLE-II

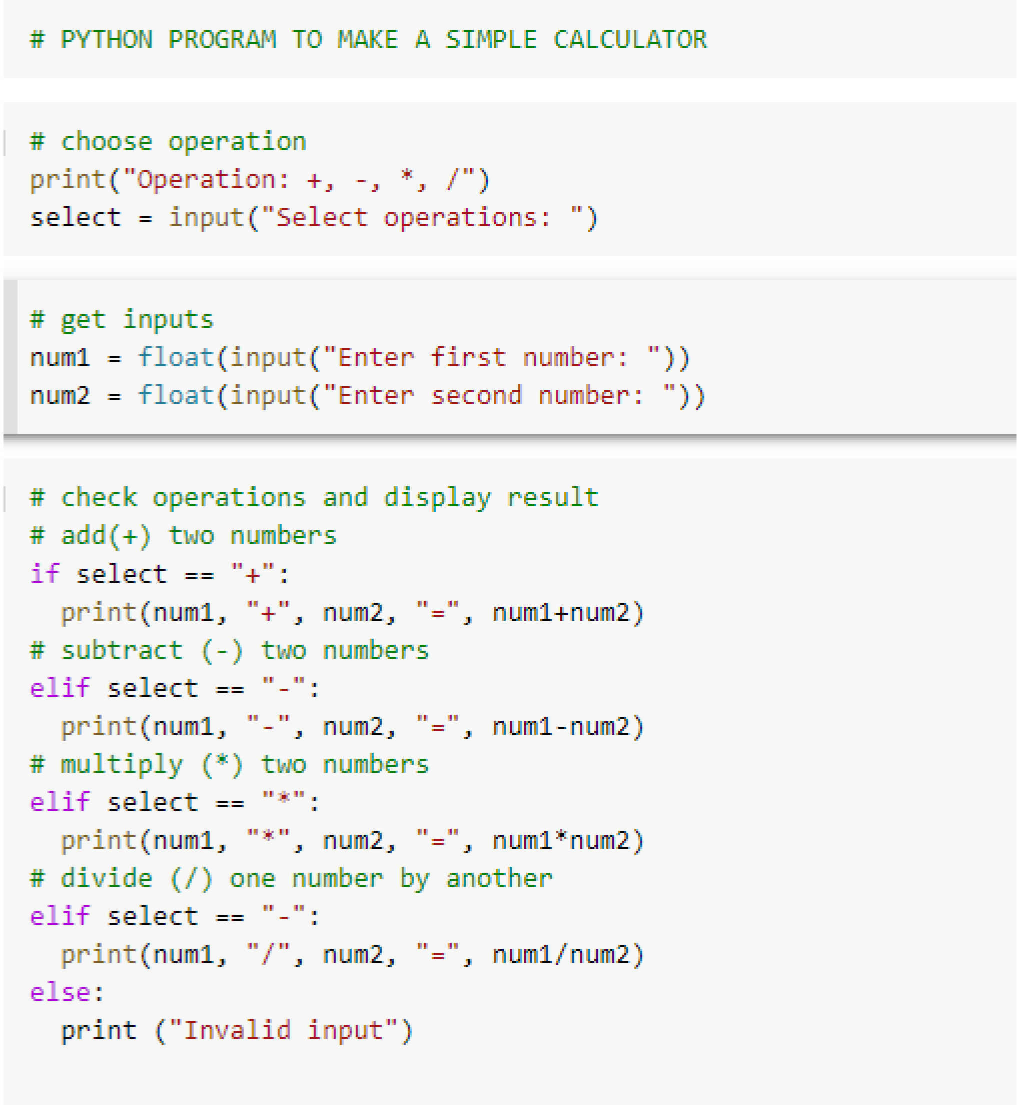
6. Data Preprocessing : Building Good Training sets
    1. Describe the dataset
    2. In the given dataset, count the rows that are having no value from each column
    3. Replace the value 0 with NaN
    4. Remove the rows with the missing values
    5. Impute the missing data with the mean values
    6. Split the dataset into training and testing sets (split training and testing in 80:20 ratio)
7. Manipulating the twitter Dataset
8. Evaluating the Results of Machine Learning
9. Implementing Linear Regression
10. Implementing Classification Algorithm
11. Implementing Clustering using K-Means clustering algorithm

CYCLE -III

12. Study of NoSQL, Hadoop, HDFS, YARN, Pig and Hive
13. Data Visualization using Tableau

1. SIMULATING A SIMPLE CALCULATOR

```
# PYTHON PROGRAM TO MAKE A SIMPLE CALCULATOR

# choose operation

      print("Operation: +, -, *, /")
      select = input("Select operations: ")

      #get inputs

      num1 = float(input("Enter first number: "))
      num2 = float(input("Enter second number: "))


      # check operations and display result
      # add(+) two numbers
      if select == "+":
      print(num1, "+", num2, "=", num1+num2)
      # subtract (-) two numbers
      elif select == "-":
      print(num1, "-", num2, "=", num1-num2)
      # multiply (*) two numbers
      elif select == "*":
        print(num1, "*", num2, "=", num1*num2)
      # divide (/) one number by another
      elif select == "-":
       print(num1, "/", num2, "=", num1/num2)
      else:
       print ("Invalid input")
```



```
# PYTHON PROGRAM TO MAKE A SIMPLE CALCULATOR
```

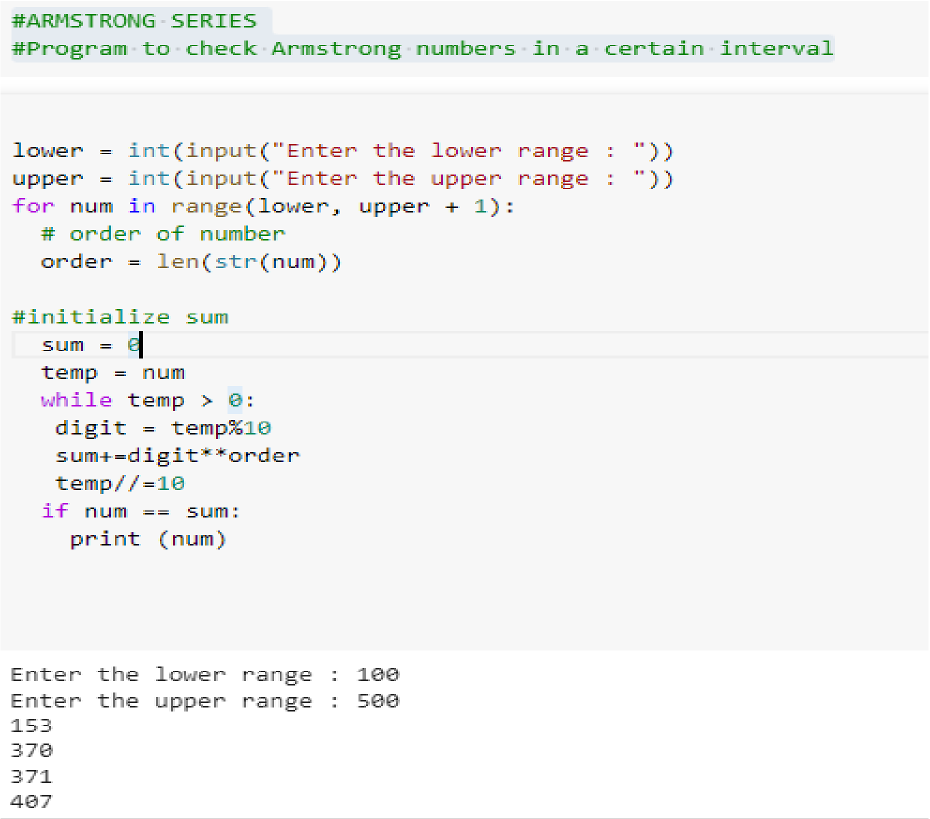
```
# choose operation
print("Operation: +, -, *, /")
select = input("Select operations: ")
```

```
# get inputs
num1 = float(input("Enter first number: "))
num2 = float(input("Enter second number: "))
```

```
# check operations and display result
# add(+) two numbers
if select == "+":
  print(num1, "+", num2, "=", num1+num2)
# subtract (-) two numbers
elif select == "-":
  print(num1, "-", num2, "=", num1-num2)
# multiply (*) two numbers
elif select == "*":
  print(num1, "*", num2, "=", num1*num2)
# divide (/) one number by another
elif select == "-":
  print(num1, "/", num2, "=", num1/num2)
else:
  print ("Invalid input")
```

2. ARMSTRONG SERIES

```
#ARMSTRONG SERIES
#Program to check Armstrong numbers in a certain interval
lower = int(input("Enter the lower range : "))
upper = int(input("Enter the upper range : "))
for num in range(lower, upper + 1):
  # order of number
  order = len(str(num))

#initialize sum
  sum = 0
  temp = num
  while temp > 0:
   digit = temp%10
   sum+=digit**order
   temp//=10
  if num == sum:
    print (num)
```



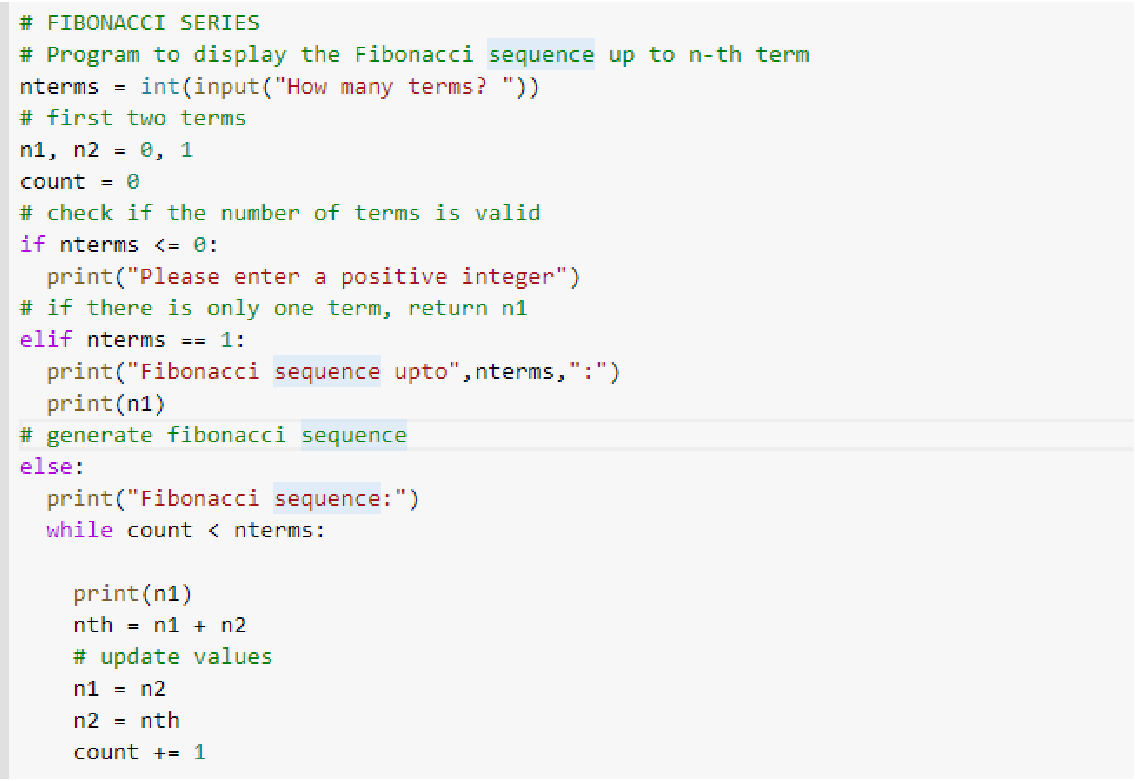
```
#ARMSTRONG SERIES
#Program to check Armstrong numbers in a certain interval


lower = int(input("Enter the lower range : "))
upper = int(input("Enter the upper range : "))
for num in range(lower, upper + 1):
  # order of number
  order = len(str(num))

#initialize sum
  sum = 0
  temp = num
  while temp > 0:
   digit = temp%10
   sum+=digit**order
   temp//=10
  if num == sum:
    print (num)
```

```
Enter the lower range : 100
Enter the upper range : 500
153
370
371
407
```

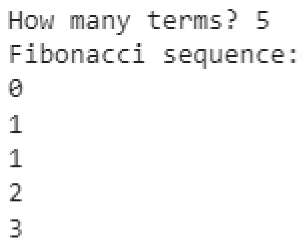
3. FIBONACCI SERIES

```
# FIBONACCI SERIES
# Program to display the Fibonacci sequence up to n-th term
nterms = int(input("How many terms? "))
# first two terms
n1, n2 = 0, 1
count = 0
# check if the number of terms is valid
```

```
if nterms <= 0:
  print("Please enter a positive integer")
# if there is only one term, return n1
elif nterms == 1:
  print("Fibonacci sequence upto",nterms,":")
  print(n1)
# generate fibonacci sequence
else:
  print("Fibonacci sequence:")
  while count < nterms:

    print(n1)
    nth = n1 + n2
    # update values
    n1 = n2
    n2 = nth
    count += 1
```



```
# FIBONACCI SERIES
# Program to display the Fibonacci sequence up to n-th term
nterms = int(input("How many terms? "))
# first two terms
n1, n2 = 0, 1
count = 0
# check if the number of terms is valid
if nterms <= 0:
  print("Please enter a positive integer")
# if there is only one term, return n1
elif nterms == 1:
  print("Fibonacci sequence upto",nterms,":")
  print(n1)
# generate fibonacci sequence
else:
  print("Fibonacci sequence:")
  while count < nterms:

    print(n1)
    nth = n1 + n2
    # update values
    n1 = n2
    n2 = nth
    count += 1
```

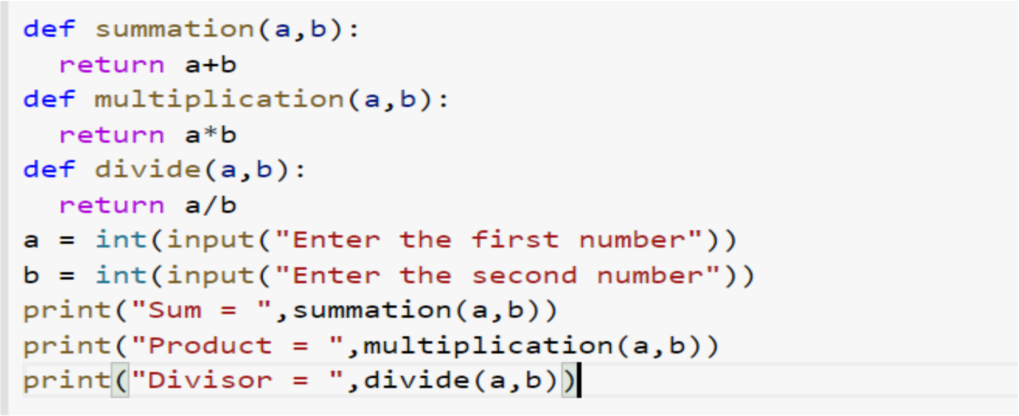


```
How many terms? 5
Fibonacci sequence:
0
1
1
2
3
```

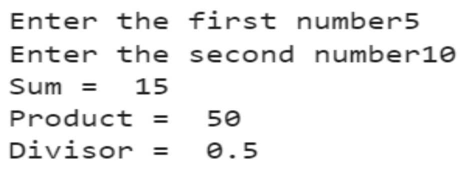
4. MODULES AND FUNCTIONS

Creating functions and importing those functions as modules

```
def summation(a,b):
  return a+b
def multiplication(a,b):
  return a*b
def divide(a,b):
  return a/b
a = int(input("Enter the first number"))
b = int(input("Enter the second number"))
print("Sum = ",summation(a,b))
print("Product = ",multiplication(a,b))
print("Divisor = ",divide(a,b))
```



```
def summation(a,b):
  return a+b
def multiplication(a,b):
  return a*b
def divide(a,b):
  return a/b
a = int(input("Enter the first number"))
b = int(input("Enter the second number"))
print("Sum = ",summation(a,b))
print("Product = ",multiplication(a,b))
print("Divisor = ",divide(a,b))
```

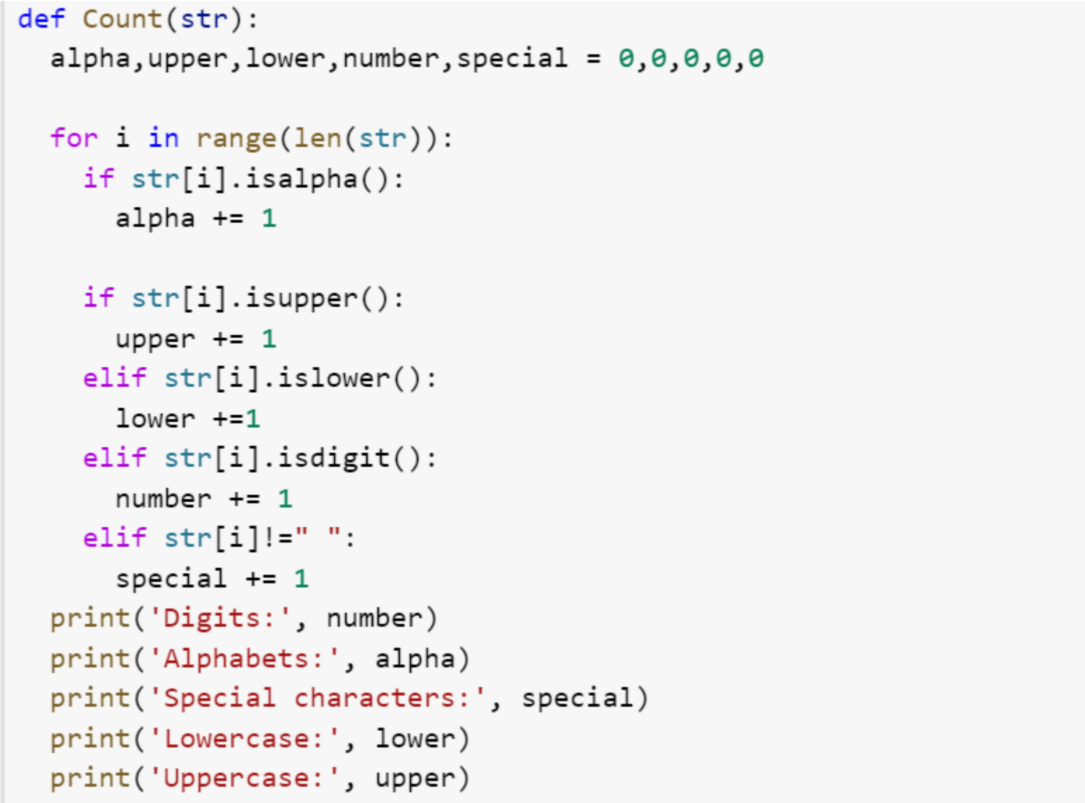


```
Enter the first number5
Enter the second number10
Sum =  15
Product =  50
Divisor =  0.5
```

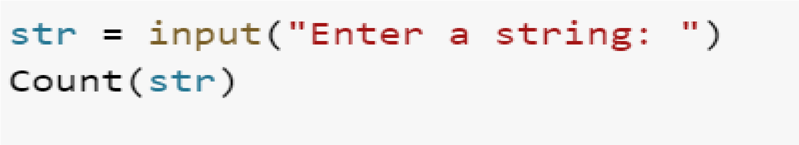
5. WORKING WITH STRINGS

a) From the string input count the special characters, alphabets, digits, lowercase and uppercase characters

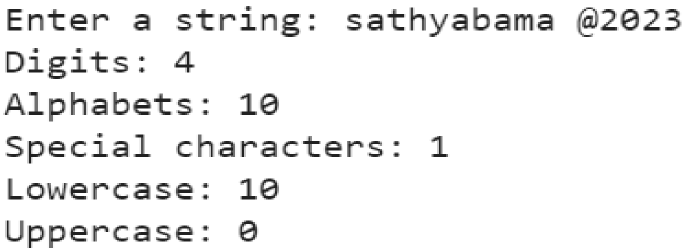
```
def Count(str):
  alpha,upper,lower,number,special = 0,0,0,0,0

  for i in range(len(str)):
    if str[i].isalpha():
      alpha += 1

    if str[i].isupper():
      upper += 1
    elif str[i].islower():
      lower +=1
    elif str[i].isdigit():
      number += 1
    elif str[i]!=" ":
      special += 1
  print('Digits:', number)
  print('Alphabets:', alpha)
  print('Special characters:', special)
  print('Lowercase:', lower)
  print('Uppercase:', upper)

str = input("Enter a string: ")
Count(str)
```



```
def Count(str):
  alpha,upper,lower,number,special = 0,0,0,0,0

  for i in range(len(str)):
    if str[i].isalpha():
      alpha += 1

    if str[i].isupper():
      upper += 1
    elif str[i].islower():
      lower +=1
    elif str[i].isdigit():
      number += 1
    elif str[i]!=" ":
      special += 1
  print('Digits:', number)
  print('Alphabets:', alpha)
  print('Special characters:', special)
  print('Lowercase:', lower)
  print('Uppercase:', upper)
```



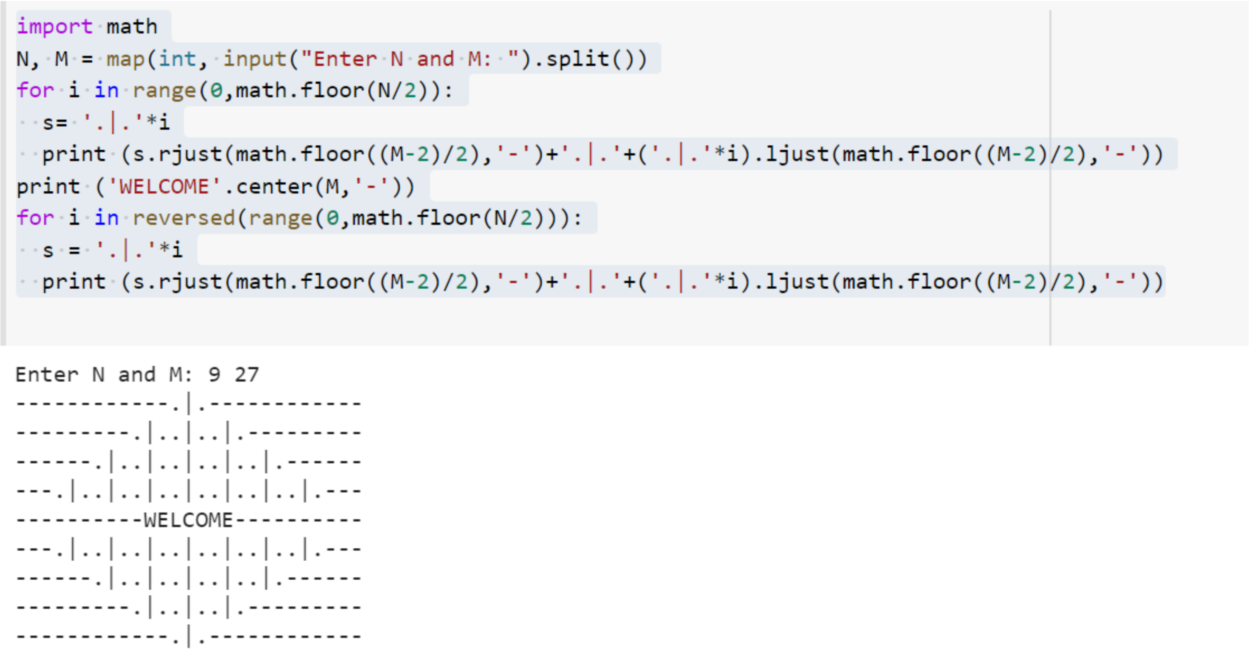
```
str = input("Enter a string: ")
Count(str)
```



```
Enter a string: sathyabama @2023
Digits: 4
Alphabets: 10
Special characters: 1
Lowercase: 10
Uppercase: 0
```

b) Print the String "Welcome". Matrix size must be N X M. (N is an odd natural number, and M is 3 times N.). The design should have 'WELCOME' written in the center. The design pattern should only use |, .and - characters.

```
import math
N, M = map(int, input("Enter N and M: ").split())
for i in range(0,math.floor(N/2)):
  s= '.|.'*i
  print (s.rjust(math.floor((M-2)/2),'-
')+'.|.'+('.|.'*i).ljust(math.floor((M-2)/2),'-'))
print ('WELCOME'.center(M,'-'))
for i in reversed(range(0,math.floor(N/2))):
  s = '.|.'*i
  print (s.rjust(math.floor((M-2)/2),'-
')+'.|.'+('.|.'*i).ljust(math.floor((M-2)/2),'-'))
```



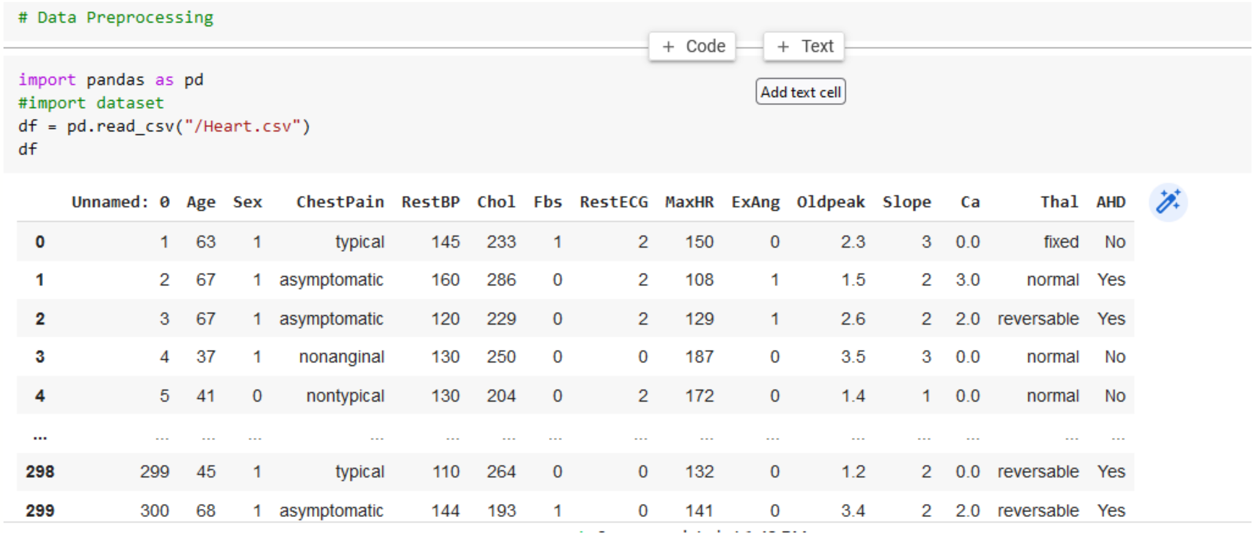
```
import math
N, M = map(int, input("Enter N and M: ").split())
for i in range(0,math.floor(N/2)):
  s= '.|.'*i
  print (s.rjust(math.floor((M-2)/2),'-')+'.|.'+('.|.'*i).ljust(math.floor((M-2)/2),'-'))
print ('WELCOME'.center(M,'-'))
for i in reversed(range(0,math.floor(N/2))):
  s = '.|.'*i
  print (s.rjust(math.floor((M-2)/2),'-')+'.|.'+('.|.'*i).ljust(math.floor((M-2)/2),'-'))
```

```
Enter N and M: 9 27
------------.|.------------
---------.|..|..|.---------
------.|..|..|..|..|.------
---.|..|..|..|..|..|..|.---
----------WELCOME----------
---.|..|..|..|..|..|..|.---
------.|..|..|..|..|.------
---------.|..|..|.---------
------------.|.------------
```

6.DATA PREPROCESSING: BUILDING GOOD TRAINING SETS

```
import pandas as pd
#import dataset
df = pd.read_csv("/Heart.csv")
df


# Find the description of data in the data frame
# Count the number of rows that are having no value from each column
df.describe()
```

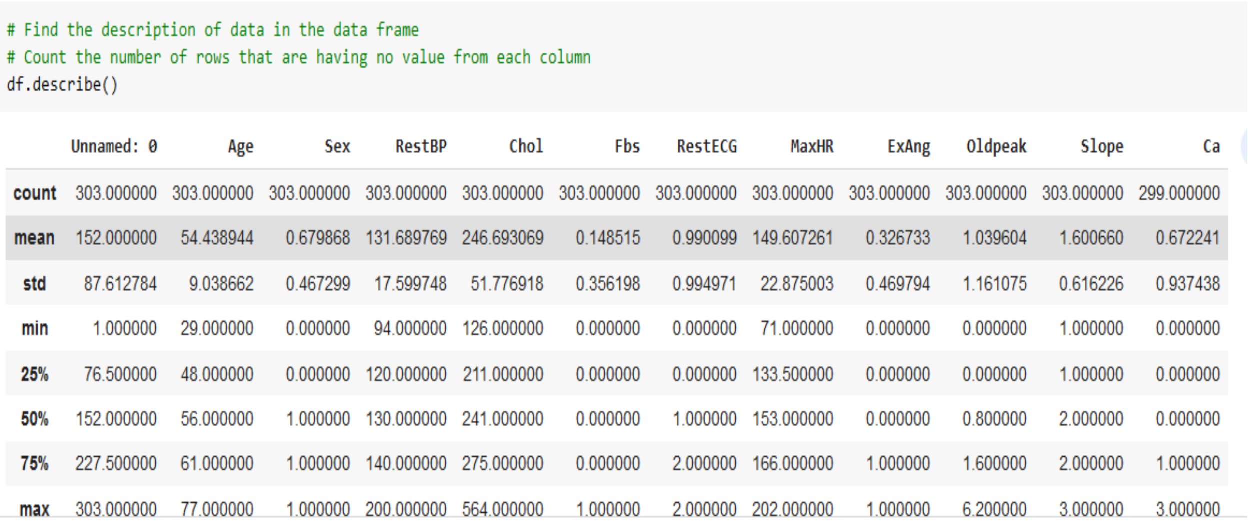
```
#  Print the number of columns, column labels, column data types from the
data frame
df.info()

# Replace the value 0 with NAN
df.replace(0,'NAN')

# Remove the rows with the missing values
df.dropna()

# Impute the missing data with mean values
df.fillna(df.mean())

# Assign values to x excepting the last column
x=df.iloc[:,0:14].values
x

#Assign values to y
y=df.iloc[:,14].values
y

# Split the dataset into Training : Testing (80:20)

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test  = train_test_split(x,y,test_size = 0.2, ran
dom_state=0)
print (x_train.shape)
print (x_test.shape)
```



```
# Data Preprocessing
```

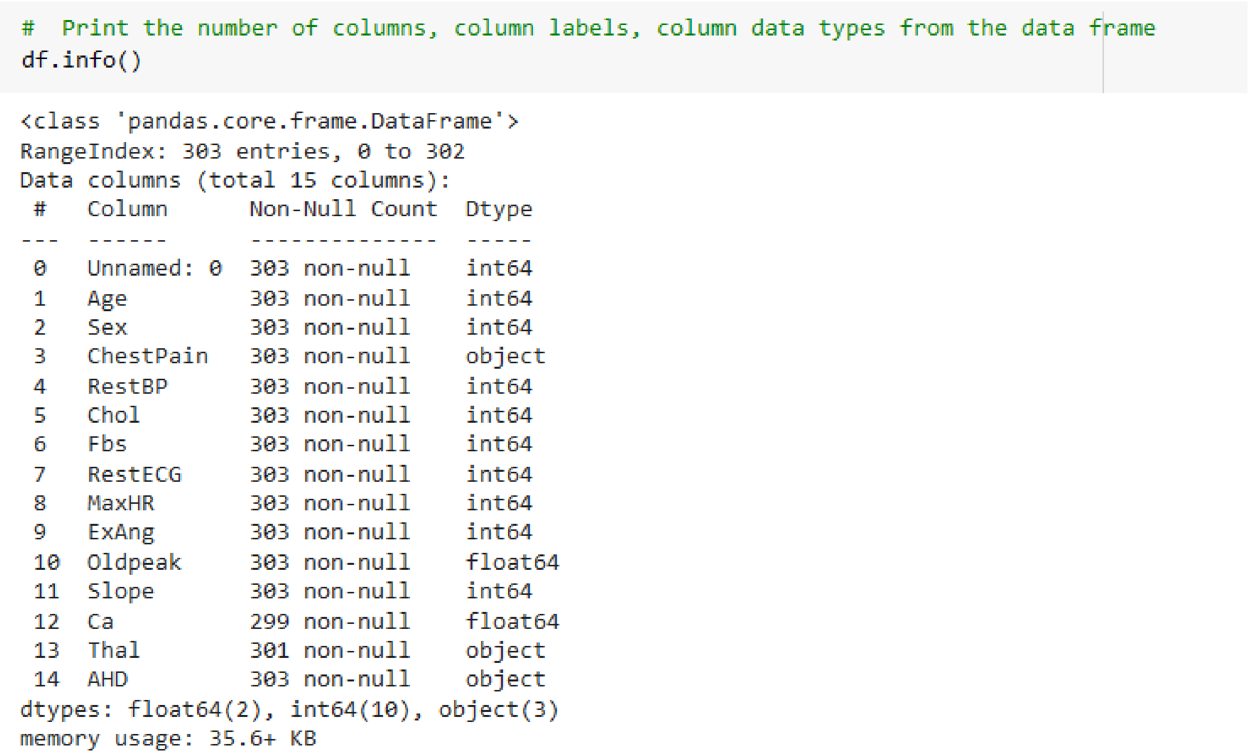
```
import pandas as pd
#import dataset
df = pd.read_csv("/Heart.csv")
df
```

	Unnamed: 0	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca	Thal	AHD
0	1	63	1	typical	145	233	1	2	150	0	2.3	3	0.0	fixed	No
1	2	67	1	asymptomatic	160	286	0	2	108	1	1.5	2	3.0	normal	Yes
2	3	67	1	asymptomatic	120	229	0	2	129	1	2.6	2	2.0	reversable	Yes
3	4	37	1	nonanginal	130	250	0	0	187	0	3.5	3	0.0	normal	No
4	5	41	0	nontypical	130	204	0	2	172	0	1.4	1	0.0	normal	No
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	299	45	1	typical	110	264	0	0	132	0	1.2	2	0.0	reversable	Yes
299	300	68	1	asymptomatic	144	193	1	0	141	0	3.4	2	2.0	reversable	Yes



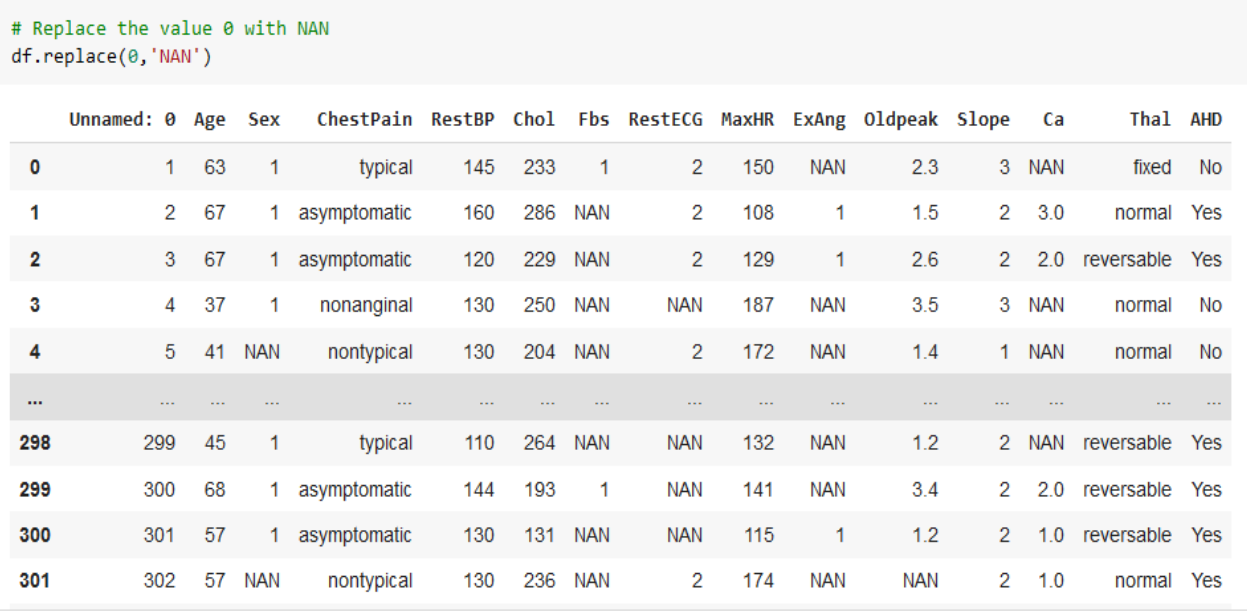
```
# Find the description of data in the data frame
# Count the number of rows that are having no value from each column
df.describe()
```

	Unnamed: 0	Age	Sex	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	299.000000
mean	152.000000	54.438944	0.679868	131.689769	246.693069	0.148515	0.990099	149.607261	0.326733	1.039604	1.600660	0.672241
std	87.612784	9.038662	0.467299	17.599748	51.776918	0.356198	0.994971	22.875003	0.469794	1.161075	0.616226	0.937438
min	1.000000	29.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	1.000000	0.000000
25%	76.500000	48.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.000000	0.000000	1.000000	0.000000
50%	152.000000	56.000000	1.000000	130.000000	241.000000	0.000000	1.000000	153.000000	0.000000	0.800000	2.000000	0.000000
75%	227.500000	61.000000	1.000000	140.000000	275.000000	0.000000	2.000000	166.000000	1.000000	1.600000	2.000000	1.000000
max	303.000000	77.000000	1.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	3.000000	3.000000



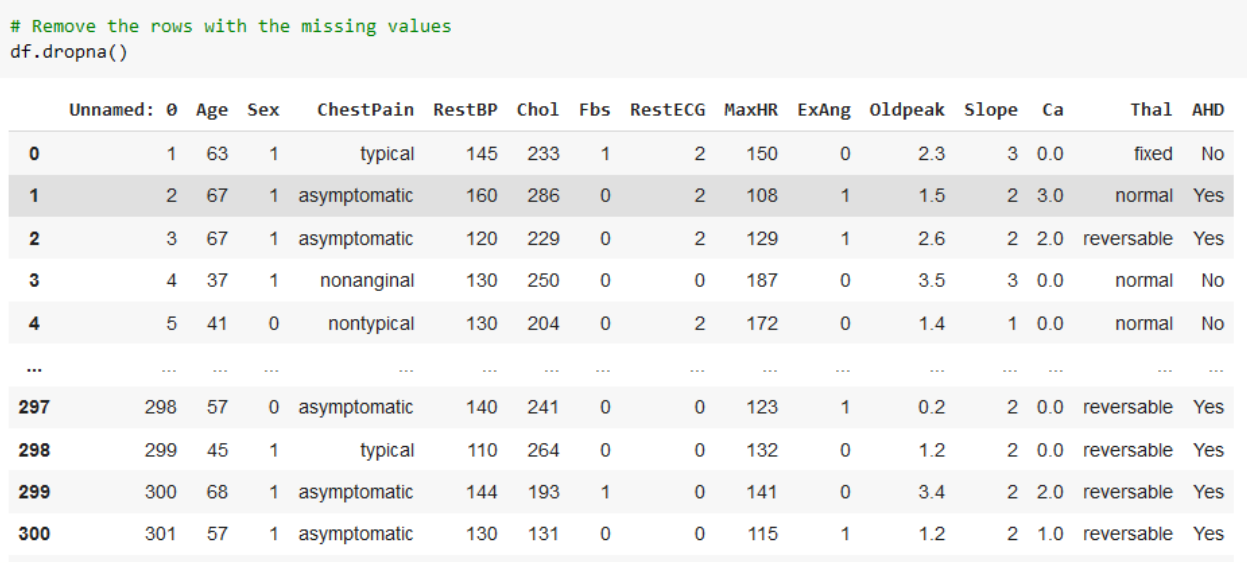
```
#  Print the number of columns, column labels, column data types from the data frame
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 15 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Unnamed: 0  303 non-null    int64
 1   Age         303 non-null    int64
 2   Sex         303 non-null    int64
 3   ChestPain   303 non-null    object
 4   RestBP      303 non-null    int64
 5   Chol        303 non-null    int64
 6   Fbs         303 non-null    int64
 7   RestECG     303 non-null    int64
 8   MaxHR       303 non-null    int64
 9   ExAng       303 non-null    int64
 10  Oldpeak     303 non-null    float64
 11  Slope       303 non-null    int64
 12  Ca          299 non-null    float64
 13  Thal        301 non-null    object
 14  AHD         303 non-null    object
dtypes: float64(2), int64(10), object(3)
memory usage: 35.6+ KB
```



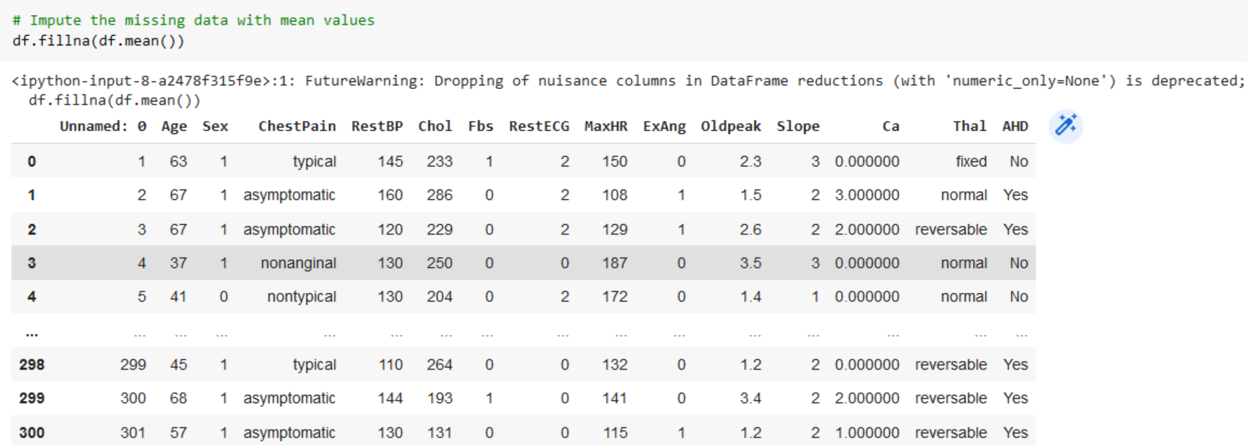
```python
# Replace the value 0 with NAN
df.replace(0,'NAN')
```

	Unnamed: 0	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca	Thal	AHD
0	1	63	1	typical	145	233	1	2	150	NAN	2.3	3	NAN	fixed	No
1	2	67	1	asymptomatic	160	286	NAN	2	108	1	1.5	2	3.0	normal	Yes
2	3	67	1	asymptomatic	120	229	NAN	2	129	1	2.6	2	2.0	reversable	Yes
3	4	37	1	nonanginal	130	250	NAN	NAN	187	NAN	3.5	3	NAN	normal	No
4	5	41	NAN	nontypical	130	204	NAN	2	172	NAN	1.4	1	NAN	normal	No
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	299	45	1	typical	110	264	NAN	NAN	132	NAN	1.2	2	NAN	reversable	Yes
299	300	68	1	asymptomatic	144	193	1	NAN	141	NAN	3.4	2	2.0	reversable	Yes
300	301	57	1	asymptomatic	130	131	NAN	NAN	115	1	1.2	2	1.0	reversable	Yes
301	302	57	NAN	nontypical	130	236	NAN	2	174	NAN	NAN	2	1.0	normal	Yes



```python
# Remove the rows with the missing values
df.dropna()
```

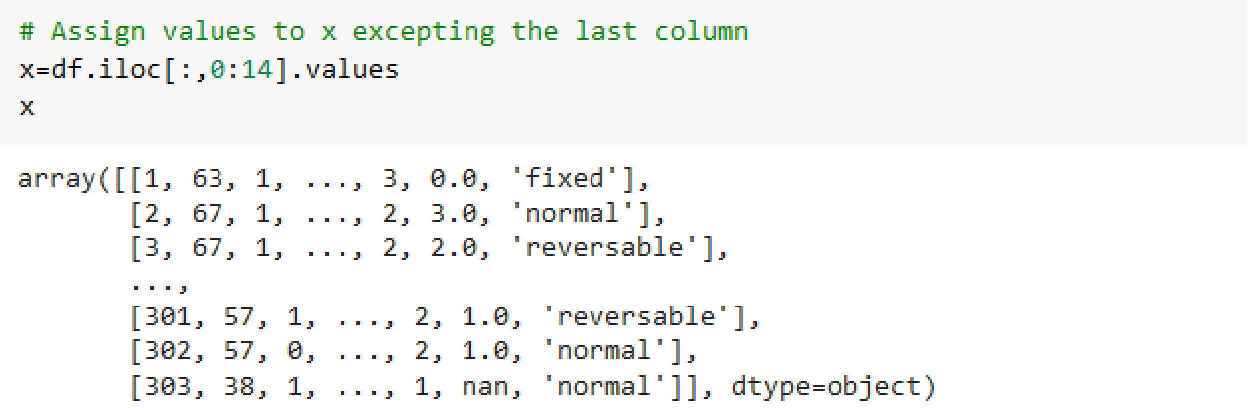
	Unnamed: 0	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca	Thal	AHD
0	1	63	1	typical	145	233	1	2	150	0	2.3	3	0.0	fixed	No
1	2	67	1	asymptomatic	160	286	0	2	108	1	1.5	2	3.0	normal	Yes
2	3	67	1	asymptomatic	120	229	0	2	129	1	2.6	2	2.0	reversable	Yes
3	4	37	1	nonanginal	130	250	0	0	187	0	3.5	3	0.0	normal	No
4	5	41	0	nontypical	130	204	0	2	172	0	1.4	1	0.0	normal	No
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
297	298	57	0	asymptomatic	140	241	0	0	123	1	0.2	2	0.0	reversable	Yes
298	299	45	1	typical	110	264	0	0	132	0	1.2	2	0.0	reversable	Yes
299	300	68	1	asymptomatic	144	193	1	0	141	0	3.4	2	2.0	reversable	Yes
300	301	57	1	asymptomatic	130	131	0	0	115	1	1.2	2	1.0	reversable	Yes



```python
# Impute the missing data with mean values
df.fillna(df.mean())
```

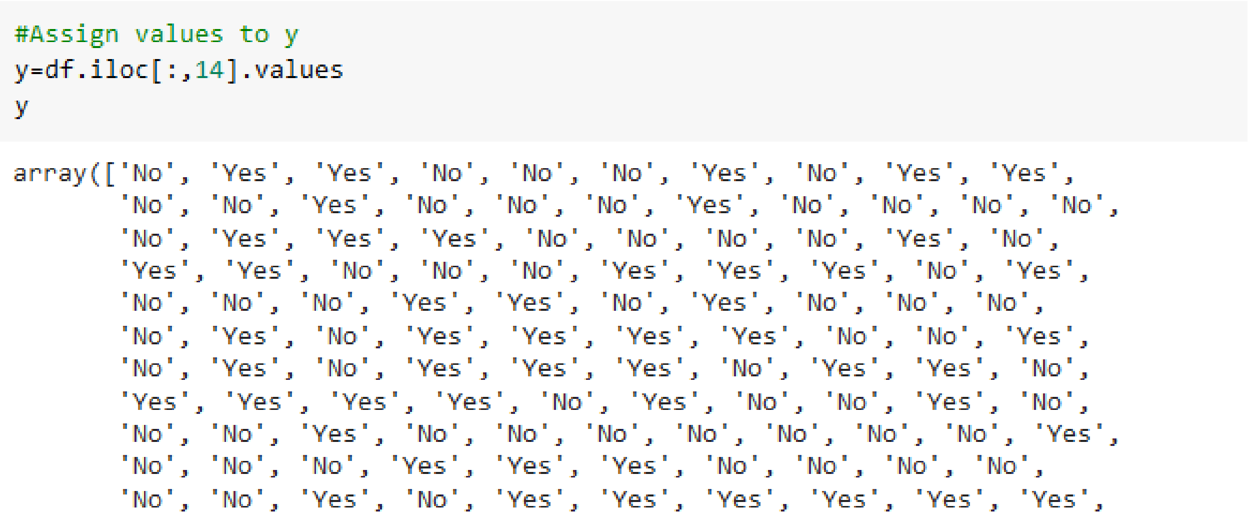
```
<ipython-input-8-a2478f315f9e>:1: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated;
  df.fillna(df.mean())
```

	Unnamed: 0	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca	Thal	AHD
0	1	63	1	typical	145	233	1	2	150	0	2.3	3	0.000000	fixed	No
1	2	67	1	asymptomatic	160	286	0	2	108	1	1.5	2	3.000000	normal	Yes
2	3	67	1	asymptomatic	120	229	0	2	129	1	2.6	2	2.000000	reversable	Yes
3	4	37	1	nonanginal	130	250	0	0	187	0	3.5	3	0.000000	normal	No
4	5	41	0	nontypical	130	204	0	2	172	0	1.4	1	0.000000	normal	No
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	299	45	1	typical	110	264	0	0	132	0	1.2	2	0.000000	reversable	Yes
299	300	68	1	asymptomatic	144	193	1	0	141	0	3.4	2	2.000000	reversable	Yes
300	301	57	1	asymptomatic	130	131	0	0	115	1	1.2	2	1.000000	reversable	Yes



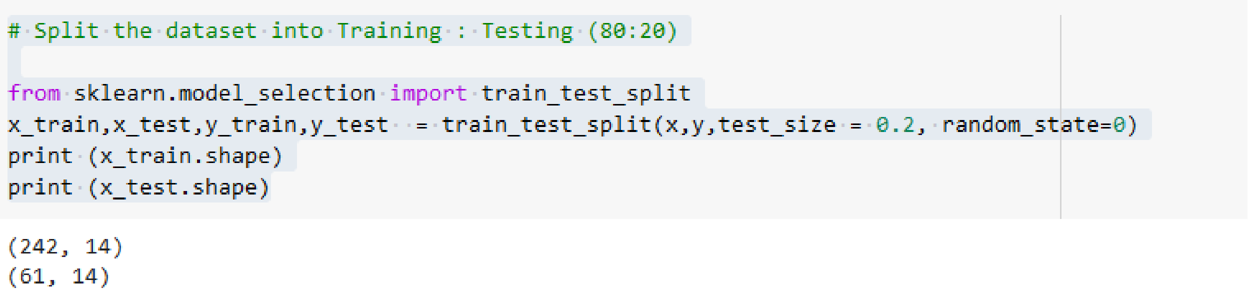
```
# Assign values to x excepting the last column
x=df.iloc[:,0:14].values
x
```

```
array([[1, 63, 1, ..., 3, 0.0, 'fixed'],
       [2, 67, 1, ..., 2, 3.0, 'normal'],
       [3, 67, 1, ..., 2, 2.0, 'reversable'],
       ...,
       [301, 57, 1, ..., 2, 1.0, 'reversable'],
       [302, 57, 0, ..., 2, 1.0, 'normal'],
       [303, 38, 1, ..., 1, nan, 'normal']], dtype=object)
```



```
#Assign values to y
y=df.iloc[:,14].values
y
```

```
array(['No', 'Yes', 'Yes', 'No', 'No', 'No', 'Yes', 'No', 'Yes', 'Yes',
       'No', 'No', 'Yes', 'No', 'No', 'No', 'Yes', 'No', 'No', 'No', 'No',
       'No', 'Yes', 'Yes', 'Yes', 'No', 'No', 'No', 'No', 'Yes', 'No',
       'Yes', 'Yes', 'No', 'No', 'No', 'Yes', 'Yes', 'Yes', 'No', 'Yes',
       'No', 'No', 'No', 'Yes', 'Yes', 'No', 'Yes', 'No', 'No', 'No',
       'No', 'Yes', 'No', 'Yes', 'Yes', 'Yes', 'Yes', 'No', 'No', 'Yes',
       'No', 'Yes', 'No', 'Yes', 'Yes', 'Yes', 'No', 'Yes', 'Yes', 'No',
       'Yes', 'Yes', 'Yes', 'Yes', 'No', 'Yes', 'No', 'No', 'Yes', 'No',
       'No', 'No', 'Yes', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'Yes',
       'No', 'No', 'No', 'Yes', 'Yes', 'Yes', 'No', 'No', 'No', 'No',
       'No', 'No', 'Yes', 'No', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes',
```



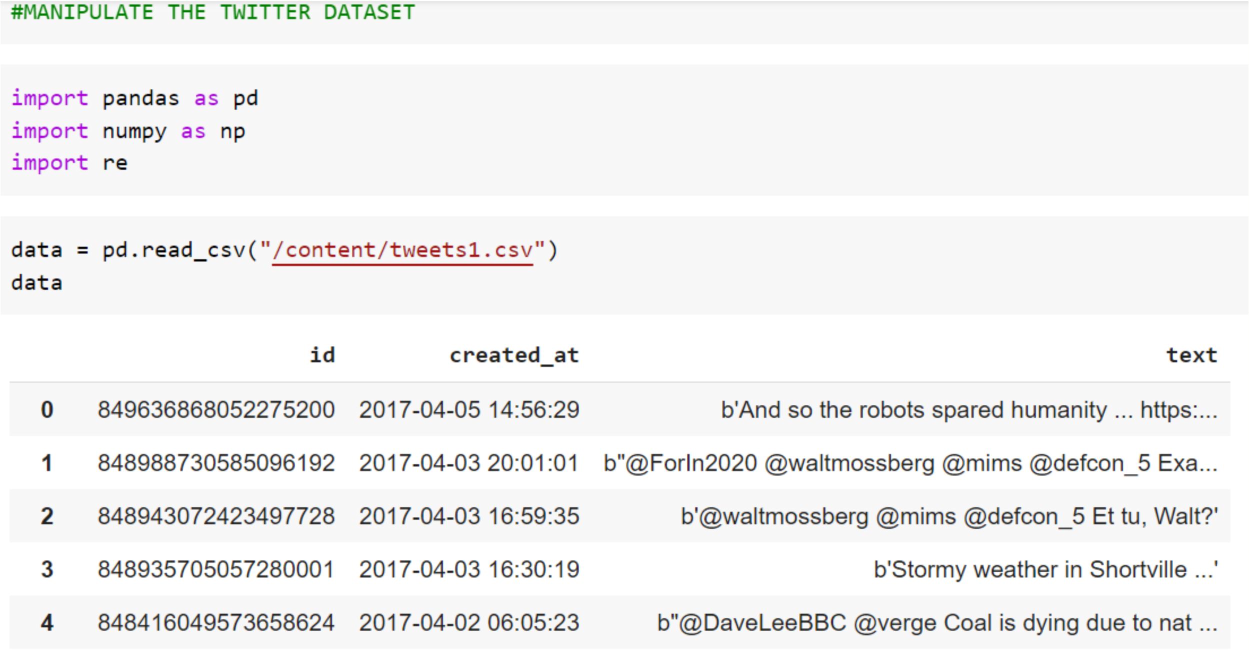
```
# Split the dataset into Training : Testing (80:20)

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test  = train_test_split(x,y,test_size = 0.2, random_state=0)
print (x_train.shape)
print (x_test.shape)
```

```
(242, 14)
(61, 14)
```

7.MANIPULATING THE TWITTER DATASET

```
import pandas as pd
import numpy as np
import re

data = pd.read_csv("/content/tweets1.csv")
data
```

```
def remove_pattern(input_txt, pattern):
  r = re.findall(pattern,input_txt)
  for i in r:
    input_txt = re.sub(i,'',input_txt)
  return input_txt
print(data)

data['new'] = np.vectorize(remove_pattern)(data ['text'],"@[\w]*")
print(data)

data['new'] = data['new'].str.replace("[^a-zA-Z#]"," ")
print(data)

data['new'] = data['new'].apply(lambda x:' '.join([w for w in x.split() if
 len(w) > 3]))
print (data)

tokenized_tweet = data['new'].apply (lambda x:x.split())
print (tokenized_tweet.head())

from nltk.stem import PorterStemmer
stemmer = PorterStemmer()
tokenized_tweet = tokenized_tweet.apply(lambda x:[stemmer.stem(i) for i in
 x])
print (tokenized_tweet.head())
```

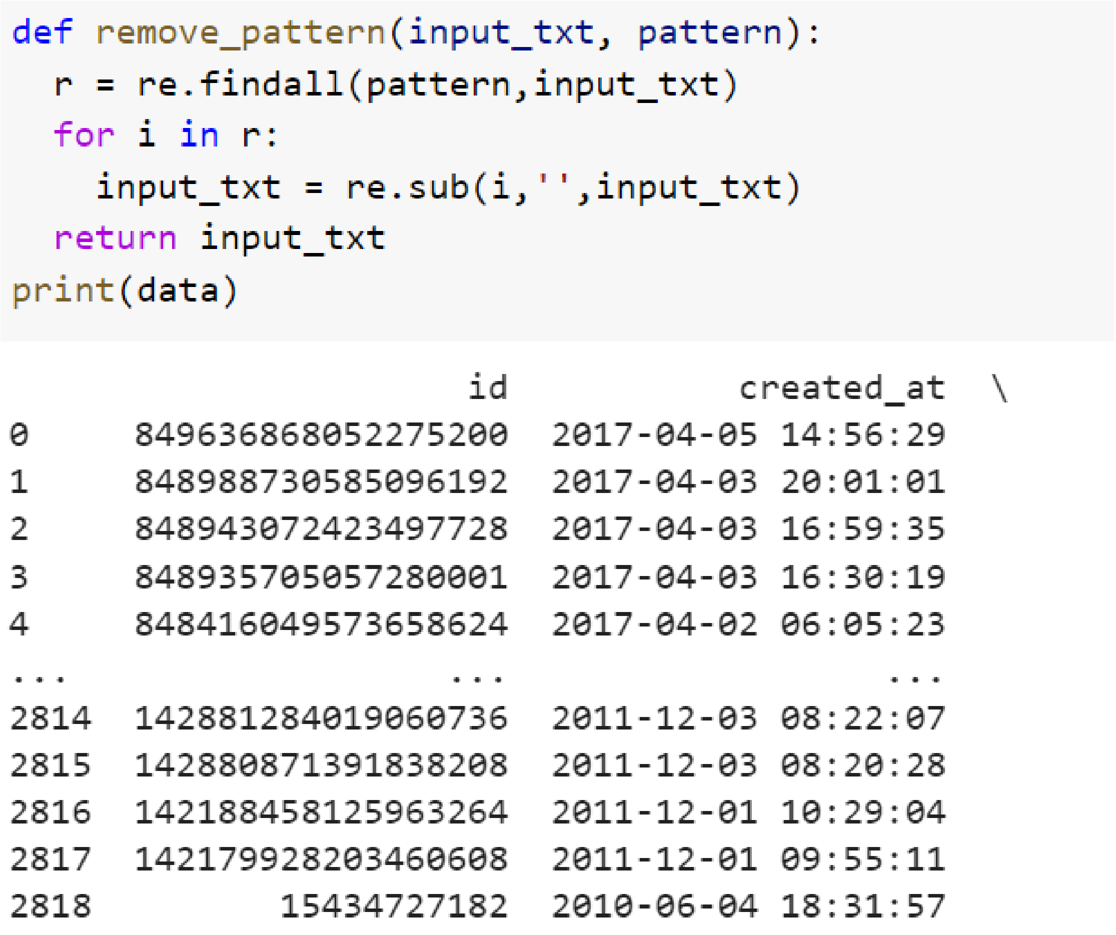


```
#MANIPULATE THE TWITTER DATASET
```

```
import pandas as pd
import numpy as np
import re
```

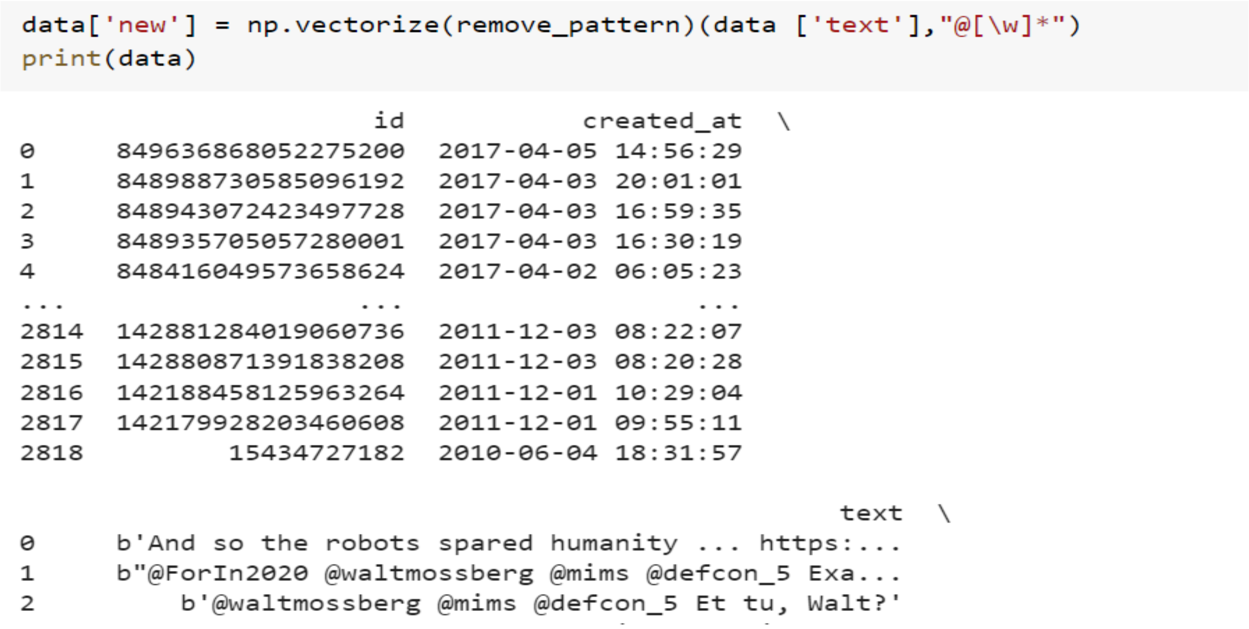
```
data = pd.read_csv("/content/tweets1.csv")
data
```

	id	created_at	text
0	849636868052275200	2017-04-05 14:56:29	b'And so the robots spared humanity ... https:...
1	848988730585096192	2017-04-03 20:01:01	b"@ForIn2020 @waltmossberg @mims @defcon_5 Exa...
2	848943072423497728	2017-04-03 16:59:35	b'@waltmossberg @mims @defcon_5 Et tu, Walt?'
3	848935705057280001	2017-04-03 16:30:19	b'Stormy weather in Shortville ...'
4	848416049573658624	2017-04-02 06:05:23	b"@DaveLeeBBC @verge Coal is dying due to nat ...



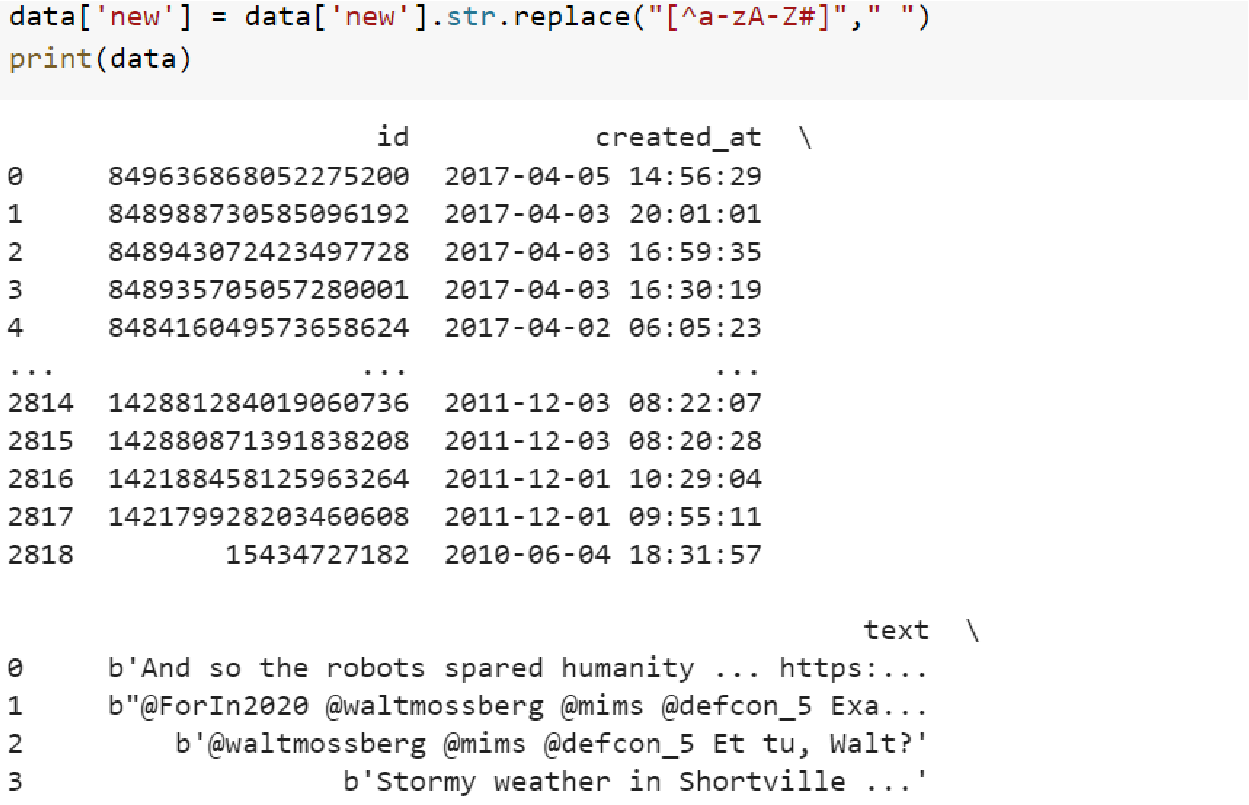
```
def remove_pattern(input_txt, pattern):
  r = re.findall(pattern,input_txt)
  for i in r:
    input_txt = re.sub(i,'',input_txt)
  return input_txt
print(data)
```

```
                        id            created_at  \
0     849636868052275200   2017-04-05 14:56:29
1     848988730585096192   2017-04-03 20:01:01
2     848943072423497728   2017-04-03 16:59:35
3     848935705057280001   2017-04-03 16:30:19
4     848416049573658624   2017-04-02 06:05:23
...                  ...                   ...
2814  142881284019060736   2011-12-03 08:22:07
2815  142880871391838208   2011-12-03 08:20:28
2816  142188458125963264   2011-12-01 10:29:04
2817  142179928203460608   2011-12-01 09:55:11
2818         15434727182   2010-06-04 18:31:57
```



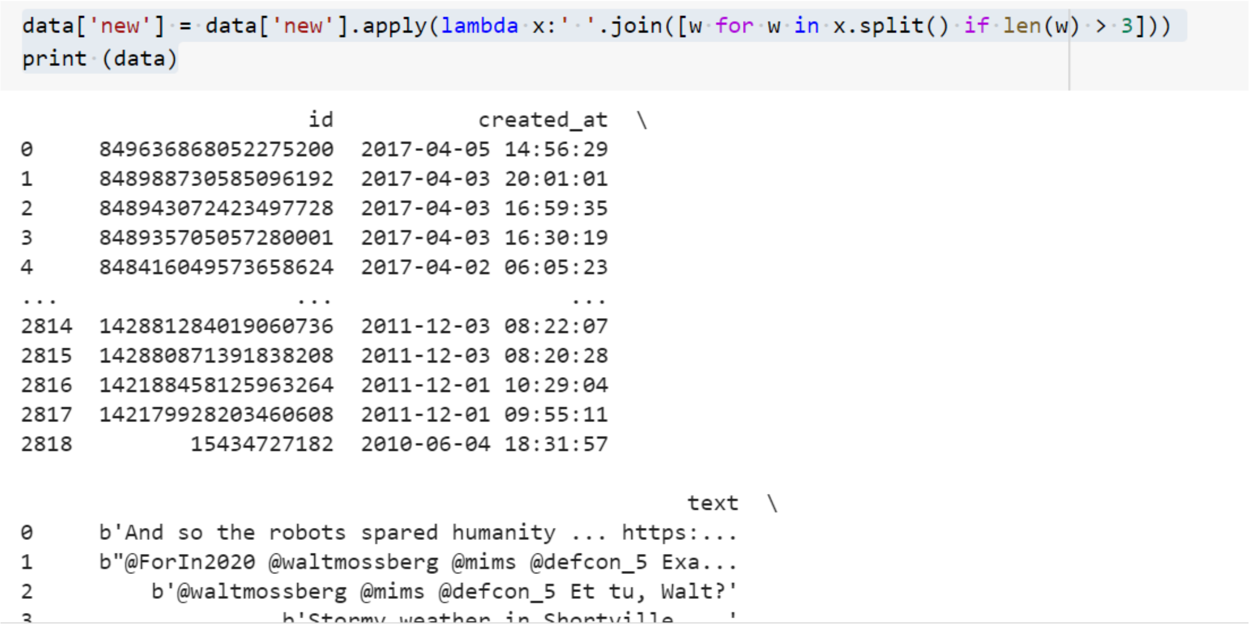
```
data['new'] = np.vectorize(remove_pattern)(data ['text'],"@[\w]*")
print(data)
```

```
                        id            created_at  \
0     849636868052275200   2017-04-05 14:56:29
1     848988730585096192   2017-04-03 20:01:01
2     848943072423497728   2017-04-03 16:59:35
3     848935705057280001   2017-04-03 16:30:19
4     848416049573658624   2017-04-02 06:05:23
...                  ...                   ...
2814  142881284019060736   2011-12-03 08:22:07
2815  142880871391838208   2011-12-03 08:20:28
2816  142188458125963264   2011-12-01 10:29:04
2817  142179928203460608   2011-12-01 09:55:11
2818         15434727182   2010-06-04 18:31:57

                                                  text  \
0     b'And so the robots spared humanity ... https:...
1     b"@ForIn2020 @waltmossberg @mims @defcon_5 Exa...
2         b'@waltmossberg @mims @defcon_5 Et tu, Walt?'
```



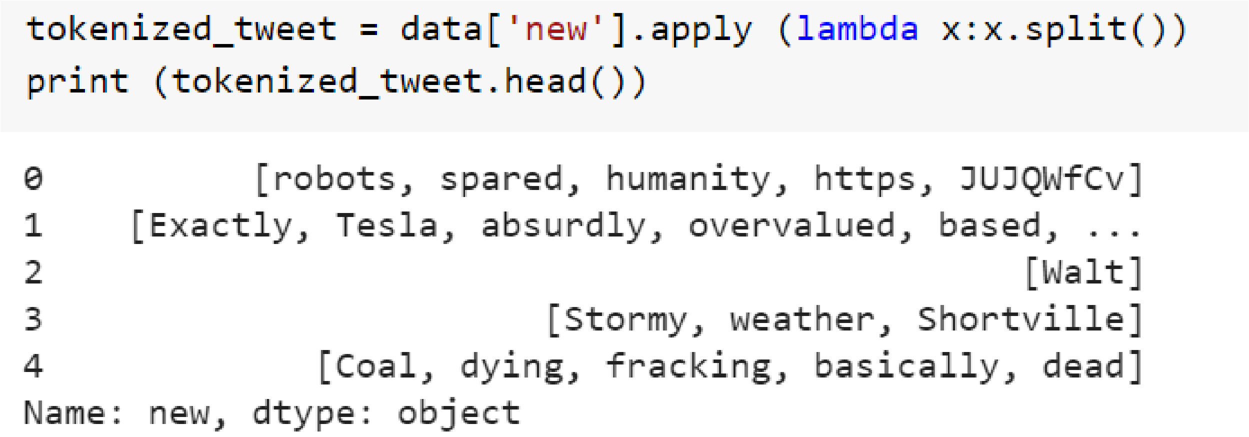
```
data['new'] = data['new'].str.replace("[^a-zA-Z#]"," ")
print(data)
```

```
                        id          created_at  \
0     849636868052275200 2017-04-05 14:56:29
1     848988730585096192 2017-04-03 20:01:01
2     848943072423497728 2017-04-03 16:59:35
3     848935705057280001 2017-04-03 16:30:19
4     848416049573658624 2017-04-02 06:05:23
...                  ...                 ...
2814  142881284019060736 2011-12-03 08:22:07
2815  142880871391838208 2011-12-03 08:20:28
2816  142188458125963264 2011-12-01 10:29:04
2817  142179928203460608 2011-12-01 09:55:11
2818         15434727182 2010-06-04 18:31:57

                                                   text  \
0     b'And so the robots spared humanity ... https:...
1     b"@ForIn2020 @waltmossberg @mims @defcon_5 Exa...
2         b'@waltmossberg @mims @defcon_5 Et tu, Walt?'
3                   b'Stormy weather in Shortville ...'
```



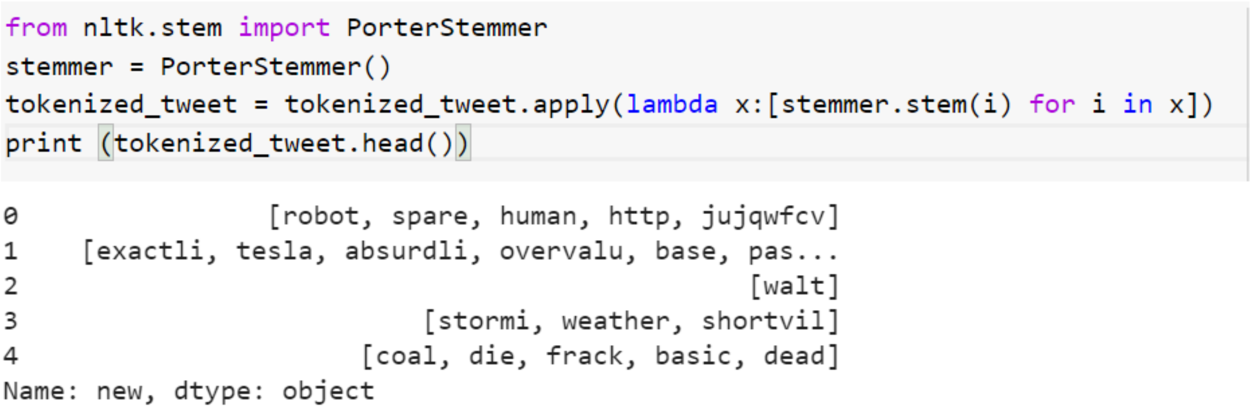
```
data['new'] = data['new'].apply(lambda x:' '.join([w for w in x.split() if len(w) > 3]))
print (data)
```

```
                        id          created_at  \
0     849636868052275200 2017-04-05 14:56:29
1     848988730585096192 2017-04-03 20:01:01
2     848943072423497728 2017-04-03 16:59:35
3     848935705057280001 2017-04-03 16:30:19
4     848416049573658624 2017-04-02 06:05:23
...                  ...                 ...
2814  142881284019060736 2011-12-03 08:22:07
2815  142880871391838208 2011-12-03 08:20:28
2816  142188458125963264 2011-12-01 10:29:04
2817  142179928203460608 2011-12-01 09:55:11
2818         15434727182 2010-06-04 18:31:57

                                                   text  \
0     b'And so the robots spared humanity ... https:...
1     b"@ForIn2020 @waltmossberg @mims @defcon_5 Exa...
2         b'@waltmossberg @mims @defcon_5 Et tu, Walt?'
3                   b'Stormy weather in Shortville ...'
```



```
tokenized_tweet = data['new'].apply (lambda x:x.split())
print (tokenized_tweet.head())
```

```
0          [robots, spared, humanity, https, JUJQWfCv]
1    [Exactly, Tesla, absurdly, overvalued, based, ...
2                                               [Walt]
3                         [Stormy, weather, Shortville]
4             [Coal, dying, fracking, basically, dead]
Name: new, dtype: object
```



```
from nltk.stem import PorterStemmer
stemmer = PorterStemmer()
tokenized_tweet = tokenized_tweet.apply(lambda x:[stemmer.stem(i) for i in x])
print (tokenized_tweet.head())
```

```
0                [robot, spare, human, http, jujqwfcv]
1    [exactli, tesla, absurdli, overvalu, base, pas...
2                                               [walt]
3                           [stormi, weather, shortvil]
4                      [coal, die, frack, basic, dead]
Name: new, dtype: object
```

8.EVALUATING THE RESULTS OF MACHINE LEARNING

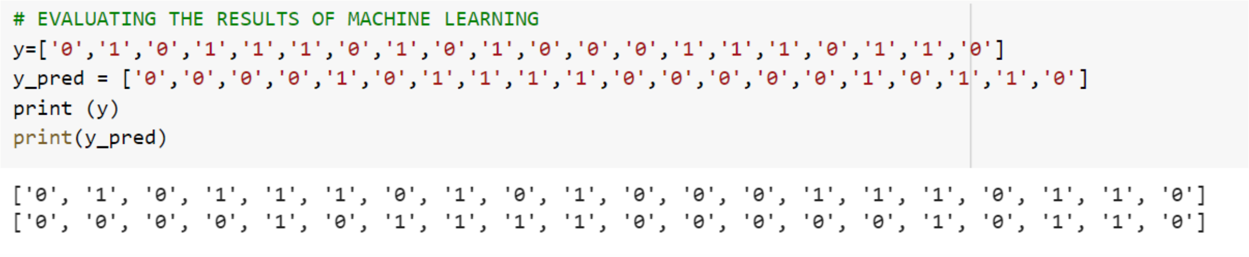
Algorithm:

1. Read actual values vs predicted values
2. Compute the following:
   - Confusion Matrix
   - Accuracy
   - Specificity
   - Sensitivity
   - Precision
   - Recall
   - Misclassification Error

Program :

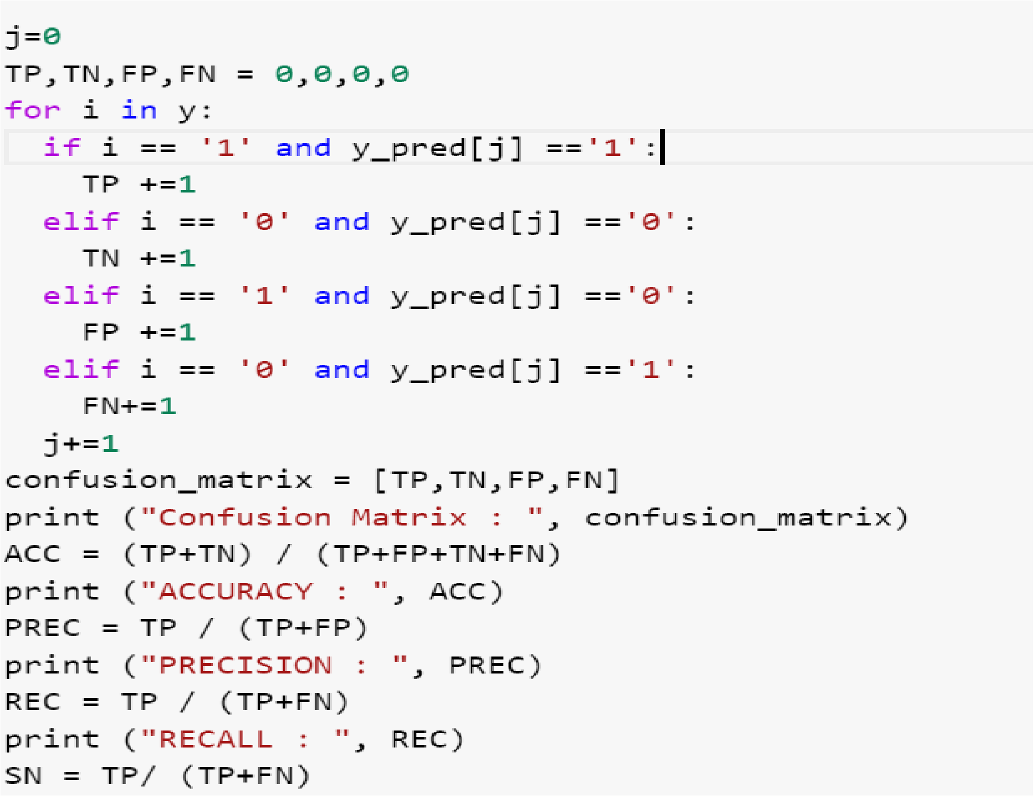
```
# EVALUATING THE RESULTS OF MACHINE LEARNING
y=['0','1','0','1','1','1','0','1','0','1','0','0','0','1','1','1','0','1'
,'1','0']
y_pred = ['0','0','0','0','1','0','1','1','1','1','0','0','0','0','0','1',
'0','1','1','0']
print (y)
print(y_pred)

j=0
TP,TN,FP,FN = 0,0,0,0
for i in y:
  if i == '1' and y_pred[j] =='1':
    TP +=1
  elif i == '0' and y_pred[j] =='0':
    TN +=1
  elif i == '1' and y_pred[j] =='0':
    FP +=1
  elif i == '0' and y_pred[j] =='1':
    FN+=1
  j+=1
confusion_matrix = [TP,TN,FP,FN]
```

```
print ("Confusion Matrix : ", confusion_matrix)
ACC = (TP+TN) / (TP+FP+TN+FN)
print ("ACCURACY : ", ACC)
PREC = TP / (TP+FP)
print ("PRECISION : ", PREC)
REC = TP / (TP+FN)
print ("RECALL : ", REC)
SN = TP/ (TP+FN)
print ("SENSITIVITY : ", SN)
SP = TN/ (TN+FP)
print ("SPECIFICITY : ", SP)
MCE = 1-ACC
print ("MISCLASSIFICATION ERROR : ", MCE)
```

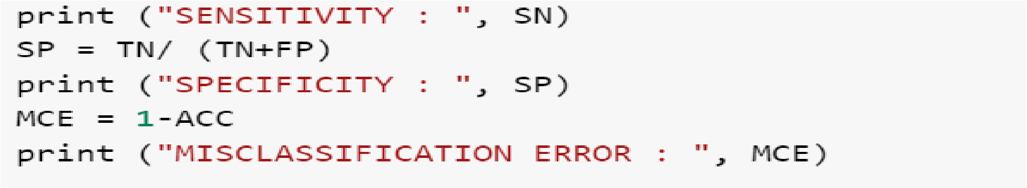


```
# EVALUATING THE RESULTS OF MACHINE LEARNING
y=['0','1','0','1','1','1','0','1','0','1','0','0','0','1','1','1','0','1','1','0']
y_pred = ['0','0','0','0','1','0','1','1','1','1','0','0','0','0','0','1','0','1','1','0']
print (y)
print(y_pred)
```

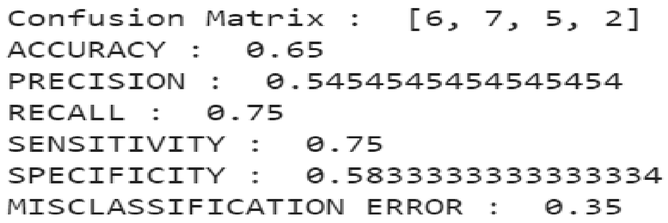
```
['0', '1', '0', '1', '1', '1', '0', '1', '0', '1', '0', '0', '0', '1', '1', '1', '0', '1', '1', '0']
['0', '0', '0', '0', '1', '0', '1', '1', '1', '1', '0', '0', '0', '0', '0', '1', '0', '1', '1', '0']
```



```
j=0
TP,TN,FP,FN = 0,0,0,0
for i in y:
  if i == '1' and y_pred[j] =='1':
    TP +=1
  elif i == '0' and y_pred[j] =='0':
    TN +=1
  elif i == '1' and y_pred[j] =='0':
    FP +=1
  elif i == '0' and y_pred[j] =='1':
    FN+=1
  j+=1
confusion_matrix = [TP,TN,FP,FN]
print ("Confusion Matrix : ", confusion_matrix)
ACC = (TP+TN) / (TP+FP+TN+FN)
print ("ACCURACY : ", ACC)
PREC = TP / (TP+FP)
print ("PRECISION : ", PREC)
REC = TP / (TP+FN)
print ("RECALL : ", REC)
SN = TP/ (TP+FN)
```



```
print ("SENSITIVITY : ", SN)
SP = TN/ (TN+FP)
print ("SPECIFICITY : ", SP)
MCE = 1-ACC
print ("MISCLASSIFICATION ERROR : ", MCE)
```



```
Confusion Matrix :  [6, 7, 5, 2]
ACCURACY :  0.65
PRECISION :  0.5454545454545454
RECALL :  0.75
SENSITIVITY :  0.75
SPECIFICITY :  0.5833333333333334
MISCLASSIFICATION ERROR :  0.35
```

9.IMPLEMENTING LINEAR REGRESSION

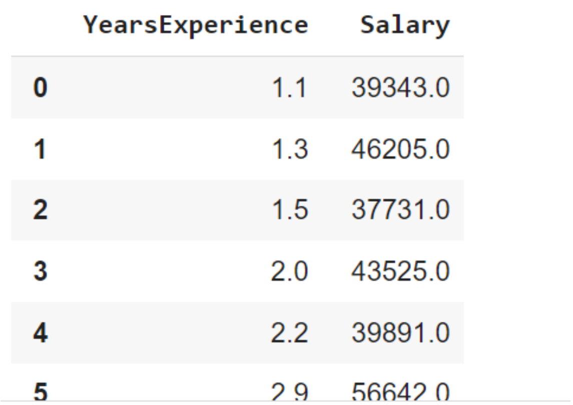
i) Input a Dataset and the X value to predict future Y
ii) Apply Regression algorithm

Output :

i) Scatter Plot and Best Regression Line
ii) Predicted Y value

Program :

```
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
import pandas as pd
import matplotlib.pyplot as plt
df=pd.read_csv(r"Salary_Data.csv")
x=list(df["YearsExperience"])
y=list(df["Salary"])
df
```

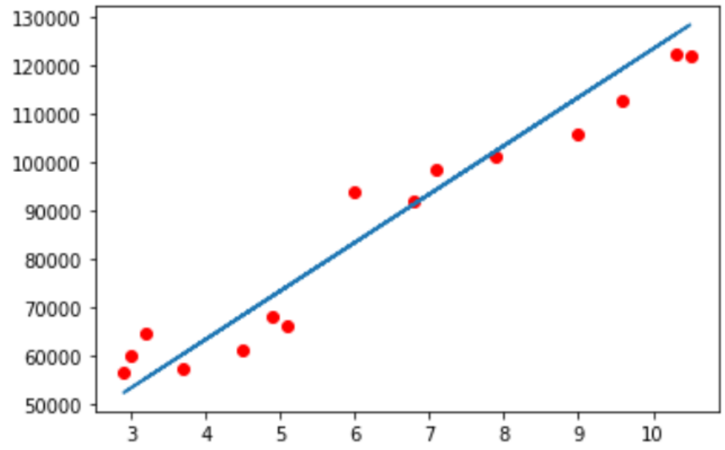


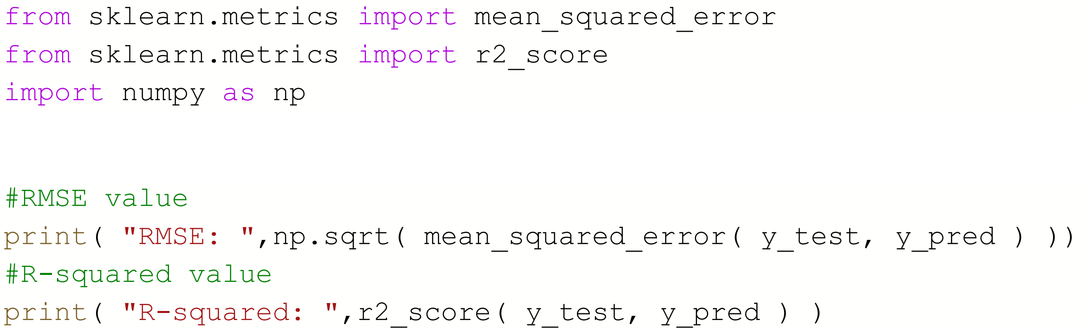
	YearsExperience	Salary
0	1.1	39343.0
1	1.3	46205.0
2	1.5	37731.0
3	2.0	43525.0
4	2.2	39891.0
5	2.9	56642.0

```
def LinearRegressor(x,y):
 sumX=sum(x)
 sumY=sum(y)
 xMean=sumX/len(x)
 yMean=sumY/len(y)
 x_minus_xmean=[val-xMean for val in x]
 y_minus_ymean=[val-yMean for val in y]
 zip_li=zip(x_minus_xmean,y_minus_ymean)
 val=[x*y for x,y in zip_li]
 b1=sum(val)/sum([x**2 for x in x_minus_xmean])
 b0=yMean-b1*xMean
 return b0,b1
```

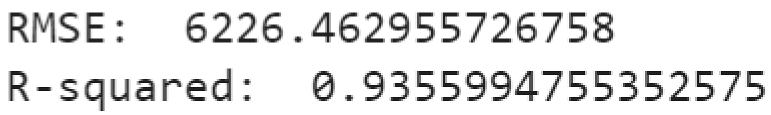
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=1/2,shuffle=True)
b=LinearRegressor(x_train,y_train)
y_pred=[b[0]+b[1]*val for val in x_test]

r2_score(y_test,y_pred)

plt.plot(x_test,y_pred)
plt.scatter(x_test, y_test,c="k")
```

```
<matplotlib.collections.PathCollection at 0x7f275ded5190>
```





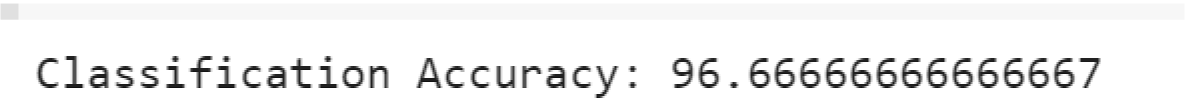
```
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
import numpy as np


#RMSE value
print( "RMSE: ",np.sqrt( mean_squared_error( y_test, y_pred ) ))
#R-squared value
print( "R-squared: ",r2_score( y_test, y_pred ) )
```



```
RMSE:  6226.462955726758
R-squared:  0.9355994755352575
```

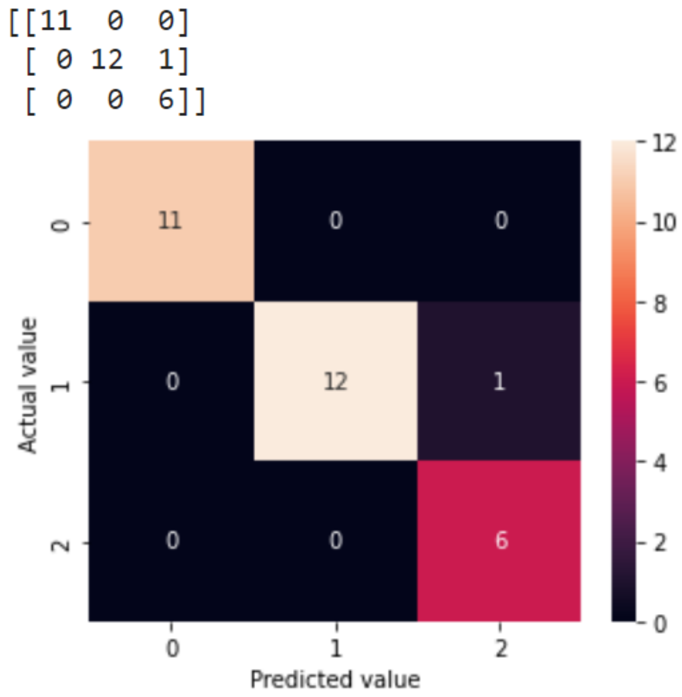
10. IMPLEMENTING CLASSIFICATION ALGORITHM

```
import pandas as pd
data=pd.read_csv("Iris.csv")
X =data.iloc[:,[1,2,3,4]].values
y =data.iloc[:,5].values
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(X_train, y_train)
y_pred=gnb.predict(X_test)
from sklearn import metrics
print("Classification Accuracy:", metrics.accuracy_score(y_test, y_pred)*100)
cm=metrics.confusion_matrix(y_test,y_pred)
```



```
Classification Accuracy: 96.66666666666667
```

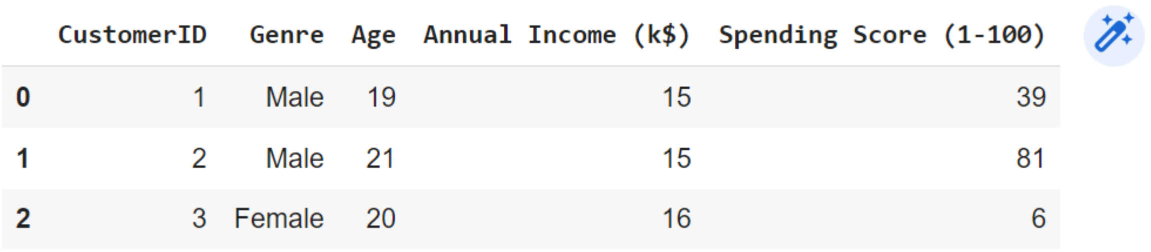
print(cm)
import seaborn as sn
from matplotlib import pyplot as plt
plt.figure(figsize=(5,4))
sn.heatmap(cm,annot=True)
plt.xlabel('Predicted value')
plt.ylabel('Actual value')
plt.show()



```
[[11  0  0]
 [ 0 12  1]
 [ 0  0  6]]
```

11. IMPLEMENTING CLUSTERING USING K-MEANS CLUSTERING ALGORITHM

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

df= pd.read_csv('/Mall_Customers.csv')
df.head(3)
```

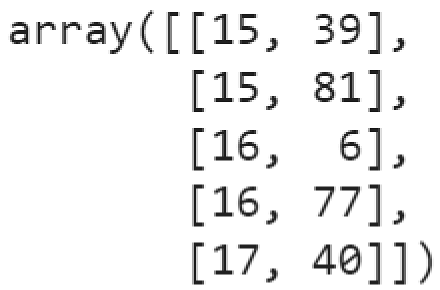


	CustomerID	Genre	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6

```
len(df)
```

```
200
```

```
X = df.iloc[:, [3,4]].values
X[0:5]
```



```
array([[15, 39],
       [15, 81],
       [16,  6],
       [16, 77],
       [17, 40]])
```

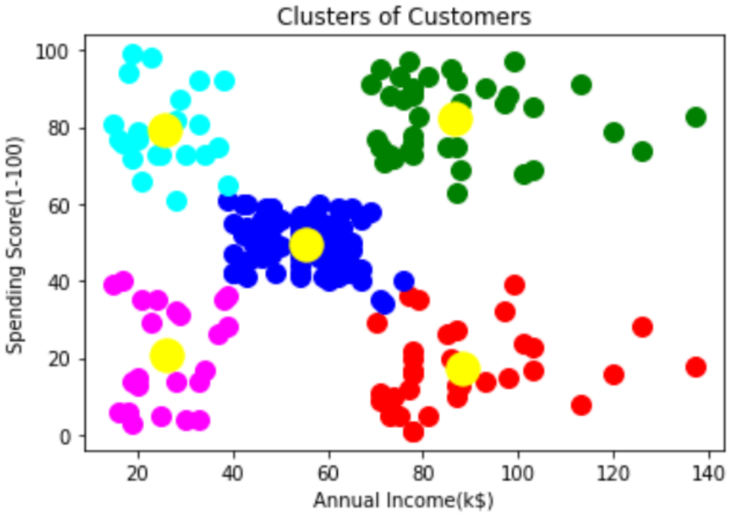
```
# KMeans class from the sklearn library.
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=5, init ='k-means++', max_iter=300, n_init=10, random_state=0 )
kmeans.n_clusters
```

```
5
```

```
y_kmeans = kmeans.fit_predict(X)

df['cluster'] = y_kmeans
print(y_kmeans.shape)
```

```
(200,)
```

```
# Visualising the clusters
plt.scatter(X[y_kmeans==0, 0], X[y_kmeans==0, 1], s=100, c='red', label =' Cluster 1')
plt.scatter(X[y_kmeans==1, 0], X[y_kmeans==1, 1], s=100, c='blue', label =
'Cluster 2')
plt.scatter(X[y_kmeans==2, 0], X[y_kmeans==2, 1], s=100, c='green', label
='Cluster 3')
plt.scatter(X[y_kmeans==3, 0], X[y_kmeans==3, 1], s=100, c='cyan', label =
'Cluster 4')
plt.scatter(X[y_kmeans==4, 0], X[y_kmeans==4, 1], s=100, c='magenta', labe
l ='Cluster 5')
#Plot the centroid.
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1],
s=300, c='yellow', label = 'Centroids')
plt.title('Clusters of Customers')
plt.xlabel('Annual Income(k$)')
plt.ylabel('Spending Score(1-100)')
plt.show()
```



CYCLE III

12. Study of NoSQL, Hadoop, HDFS, YARN, Pig and Hive

NoSQL databases (also named as not only SQL) are non-tabular databases and store data differently than relational tables. NoSQL databases come in a variety of types based on their data model. The main types are document, key-value, wide-column, and graph. They provide flexible schemas and scale easily with large amounts of data and high user loads. The most popular NoSQL database is MongoDB

NoSQL database features:

Each NoSQL database has its own unique features. At a high level, many NoSQL databases have the following features:

- Flexible schemas
- Horizontal scaling
- Fast queries due to the data model
- Ease of use for developers

For example:

Let us take an example of a client who needs a database design for his website. His website has the following requirements:

Every post is distinct (contains unique title, description and url).
Every post can have one or more tags.

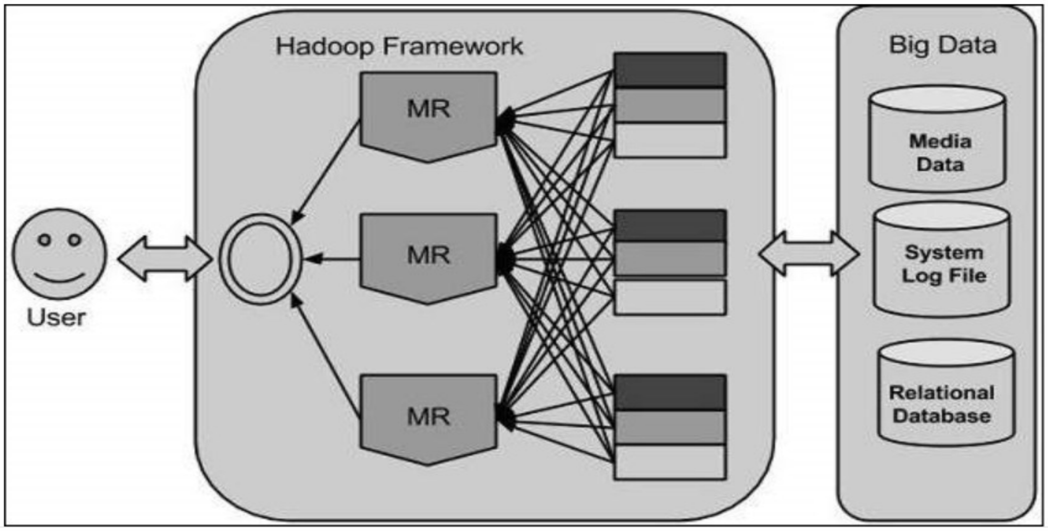
The data model we design for a NoSQL database will depend on the type of NoSQL database we choose. The schema design, if selected MongoDB will have one collection post and has the following structure:

```
{
 _id: POST_ID
 title: TITLE_OF_POST,
 description: POST_DESCRIPTION,
 by: POST_BY,
 url: URL_OF_POST,
 tags: [TAG1, TAG2, TAG3],
 }
```

In order to retrieve all of the information about the website requirements, a single document can be retrieved from the database. No joins are required, resulting in faster queries.

HADOOP, HDFS and YARN

Hadoop is an open-source framework that allows to store and process big data in a distributed environment across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Hadoop runs applications using the MapReduce algorithm, where the data is processed in parallel with others. In short, Hadoop is used to develop applications that could perform complete statistical analysis on huge amounts of data. The following figure depicts the Hadoop Framework.



At its core, Hadoop has two major layers namely −

- Processing/Computation layer (MapReduce), and
- Storage layer (Hadoop Distributed File System).

MAPREDUCE

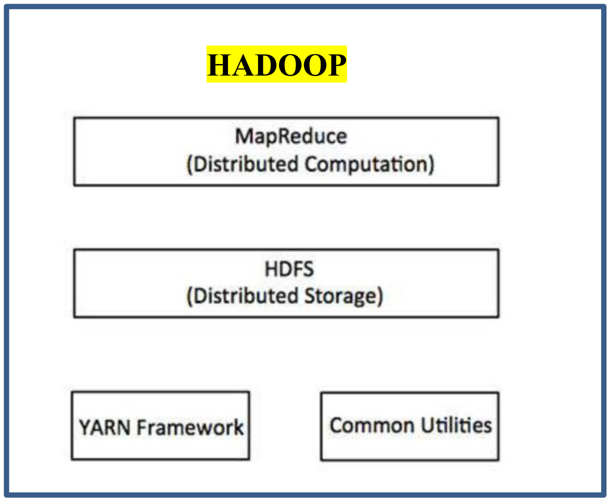
MapReduce is a parallel programming model for writing distributed applications devised at Google for efficient processing of large amounts of data (multi-terabyte data-sets), on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner. The MapReduce program runs on Hadoop which is an Apache open-source framework.

HADOOP DISTRIBUTED FILE SYSTEM (HDFS)

The Hadoop Distributed File System (HDFS) is based on the Google File System (GFS) and provides a distributed file system that is designed to run on commodity hardware. It is highly fault-tolerant. It provides high throughput access to application data and is suitable for applications having large datasets.

Hadoop framework also includes the following two modules −

- **Hadoop Common** − These are Java libraries and utilities required by other Hadoop modules.
- **Hadoop YARN** − This is a framework for job scheduling and cluster resource management.



How Does Hadoop Work?

It is quite expensive to build bigger servers with heavy configurations that handle large scale processing, but as an alternative, you can tie together many commodity computers with single-CPU, as a single functional distributed system and practically, the clustered machines can read the dataset in parallel and provide a much higher throughput.

- Hadoop framework allows the user to quickly write and test distributed systems. It utilizes the underlying parallelism of the CPU cores.
- Hadoop does not rely on hardware to provide fault-tolerance and high availability (FTHA), rather Hadoop library itself has been designed to detect and handle failures at the application layer.
- Servers can be added or removed from the cluster dynamically and Hadoop continues to operate without interruption.
- Another big advantage of Hadoop is that apart from being open source, it is compatible on all the platforms since it is Java based.

YARN (Yet Another Resource Manager)

Yet Another Resource Manager takes programming to the next level beyond Java , and makes it interactive to let another application Hbase, Spark etc. to work on it. Different Yarn applications can co-exist on the same cluster so MapReduce, Hbase, Spark all can run at the same time bringing great benefits for manageability and cluster utilization.

Components of YARN

- **Client:** For submitting MapReduce jobs.
- **Resource Manager:** To manage the use of resources across the cluster
- **Node Manager:**For launching and monitoring the computer containers on machines in the cluster.
- **Map Reduce Application Master:** Checks tasks running the MapReduce job. The application master and the MapReduce tasks run in containers that are scheduled by the resource manager, and managed by the node managers.

  Job tracker & Task tracker were were used in previous version of Hadoop, which were responsible for handling resources and checking progress management. However, Hadoop 2.0 has Resource manager and NodeManager to overcome the shortfall of Jobtracker & Tasktracker.

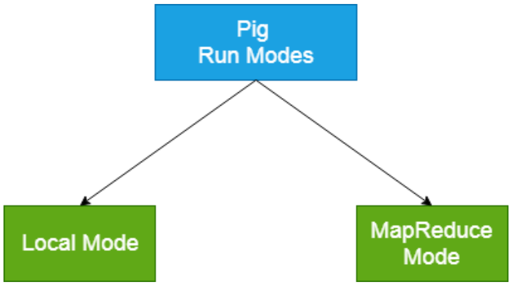
Benefits of YARN

Scalability
Utilization
Multitenancy

PIG

Pig is a high-level data flow platform for executing Map Reduce programs of Hadoop. It was developed by Yahoo. The language for Pig is pig Latin.

Apache Pig Run Modes



Local Mode

- It executes in a single JVM and is used for development experimenting and prototyping.
- Here, files are installed and run using localhost.
- The local mode works on a local file system. The input and output data stored in the local file system.

The command for local mode grunt shell:

```
$ pig-x local
```

MapReduce Mode

- The MapReduce mode is also known as Hadoop Mode.
- It is the default mode.
- In this Pig renders Pig Latin into MapReduce jobs and executes them on the cluster.
- It can be executed against semi-distributed or fully distributed Hadoop installation.
- Here, the input and output data are present on HDFS.

The command for Map reduce mode:

```
$ pig
```

HIVE :

Apache Hive is a data ware house system for Hadoop that runs SQL like queries called HQL (Hive query language) which gets internally converted to map reduce jobs. Hive was developed by Facebook. It supports Data definition Language, Data Manipulation Language and user defined functions.

Create Database Statement

Create Database is a statement used to create a database in Hive. A database in Hive is a **namespace** or a collection of tables.

The **syntax** for this statement is as follows:

```
CREATE DATABASE|SCHEMA [IF NOT EXISTS] <database name>
```

Here, IF NOT EXISTS is an optional clause, which notifies the user that a database with the same name already exists.

The following query is used to verify a databases list:

```
hive> SHOW DATABASES;
```

Create Table Statement

Create Table is a statement used to create a table in Hive.

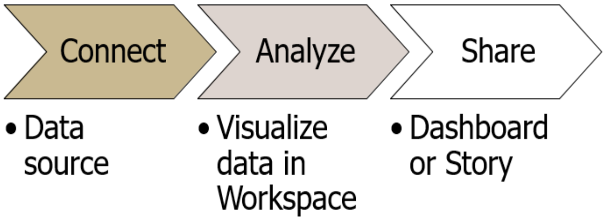
Syntax

```
CREATE [TEMPORARY] [EXTERNAL] TABLE [IF NOT EXISTS] [db_name.] table_name

[(col_name data_type [COMMENT col_comment], ...)]
[COMMENT table_comment]
[ROW FORMAT row_format]
[STORED AS file_format]
```

13. DATA VISUALIZATION USING TABLEAU

As a leading data visualization tool, Tableau has many desirable and unique features. Its powerful data discovery and exploration application allows you to answer important questions in seconds.

Tableau Work flow:

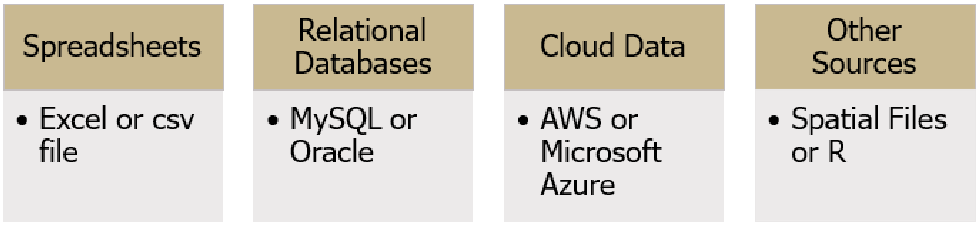


Connect
- Data source

Analyze
- Visualize data in Workspace

Share
- Dashboard or Story

Data Source Types:



Spreadsheets
- Excel or csv file

Relational Databases
- MySQL or Oracle

Cloud Data
- AWS or Microsoft Azure

Other Sources
- Spatial Files or R

Data Field :

A field, also known as a column, is a single piece of information from a record in a data set.

- Qualitative Field (Dimensions)
    - Describes or Categorizes Data
    - What, when or who
    - Slices the quantitative data
- Quantitative Field (Measures)
    - Numerical Data
    - Provides measurement for qualitative category

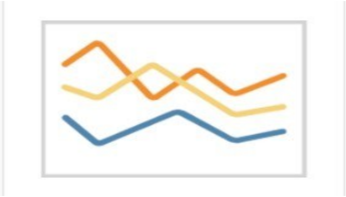
Can be used in calculations

Tableau supports the following data types:

1. **Boolean**: True and false can be stored in this data type.

2. **Date/Datetime**:
   This data type can help in leveraging Tableau’s default date hierarchy behavior when applied to valid date or DateTime fields.
3. **Number**: These are values that are numeric. Values can be integers or floating-point numbers (numbers with decimals).
4. **String**: This is a sequence of characters encased in single or double quotation marks.
5. **Geolocation:** These are values that we need to plot maps.

Chart Types



Line — View trends in data over time.

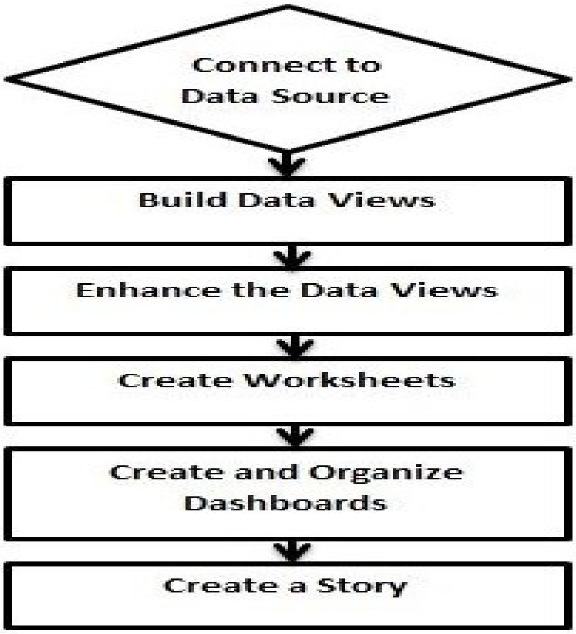
Bar — Compare data across categories.

Heat Map — Show the relationship between two factors.

Steps to Perform Visualization

1. Open Tableau tool
2. In the Connect panel at the left side of the Start page, click the Excel link under the **“To a File”** heading to the open file selection option.
3. Using the file selection box, select the Excel worksheet that you want to open, and then click the Open button to continue
4. Select the required Worksheet from the navigation menu on the left and drag it onto the Drag Sheets
5. After loading we can perform data cleaning, data preprocessing and feature extraction

Though the final outcome expected from a Tableau project is ideally a dashboard with story, there are many intermediate steps which needs to be completed to reach this goal. Following is a flow diagram of design steps that should be ideally followed to create effective dashboards.


Connect to Data Source
Build Data Views
Enhance the Data Views
Create Worksheets
Create and Organize Dashboards
Create a Story