1-data-acquisition

April 8, 2024

[1]:
```
# Aim : To perform Data Acquisition of given data set using Pandas
```

[2]:
```
# Name: Samiksha Badhe
# Class: 3rd Year
# Sec: B
# Roll No. : 05
```

[3]:
```
import os
import numpy as np
import pandas as pd
from sklearn.datasets import load_iris
```

[4]:
```
data=pd.read_csv("C:\\Users\\hp\\Desktop\\IRIS.csv")
```

[5]:
```
data.head()
```

[5]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

[6]:
```
data.tail()
```

[6]:

	sepal_length	sepal_width	petal_length	petal_width	species
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

[7]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
```

```
---  ------        --------------  -----
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

[8]:

```
data.describe()
```

[8]:

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

[9]:

```
data.ndim
```

[9]: 2

[10]:

```
data.shape
```

[10]: (150, 5)

[11]:

```
data.size
```

[11]: 750

[12]:

```
data.isnull()
```

[12]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	False	False	False	False	False
1	False	False	False	False	False
2	False	False	False	False	False
3	False	False	False	False	False
4	False	False	False	False	False
..	…	…	…	…	…
145	False	False	False	False	False
146	False	False	False	False	False
147	False	False	False	False	False
148	False	False	False	False	False
149	False	False	False	False	False

```
[150 rows x 5 columns]
```

[13]:
```python
data.isnull().sum()
```

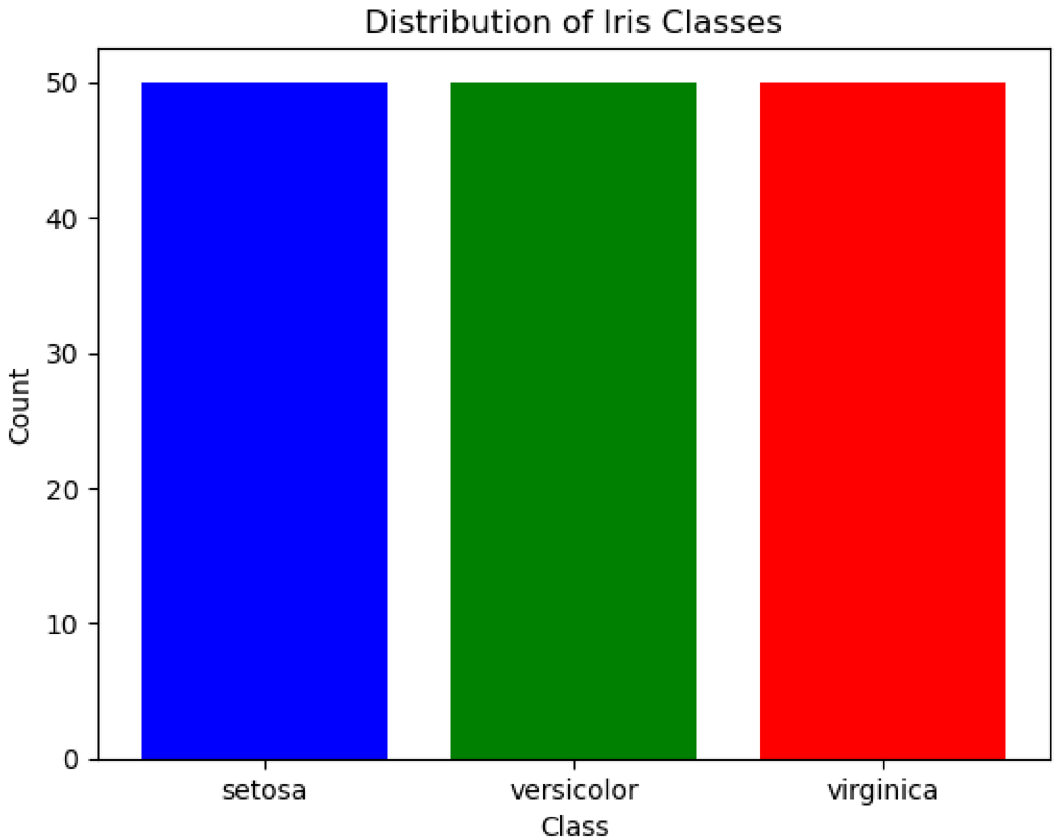
[13]:
```
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
```

[14]:
```python
df = pd.DataFrame(data)

# Count unique values in 'Column1'
value_counts = df['species'].value_counts()
print(value_counts)
```

```
species
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: count, dtype: int64
```

[15]:
```python
from matplotlib import pyplot as plt
```

[16]:
```python
iris = load_iris()
target_names = iris.target_names
y = iris.target

# Count occurrences of each class
class_counts = np.bincount(y)

# Plot the bar chart
plt.bar(target_names, class_counts, color=['blue', 'green', 'red'])
plt.xlabel('Class')
plt.ylabel('Count')
plt.title('Distribution of Iris Classes')
plt.show()
```


Distribution of Iris Classes
Count
50
40
30
20
10
0
setosa
versicolor
virginica
Class

```
[ ]:
```

2-linear-regression

April 8, 2024

[1]:
```
# Aim: To perform Simple Linear Regression and find out the Coefficient of it.
```

[2]:
```
# Name: Samiksha Badhe
# Class: 3rd Year
# Section : B
# Roll no: 05
```

[3]:
```
import os
```

[4]:
```
import pandas as pd
```

[5]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import seaborn as sns
import matplotlib.pyplot as plt
```

[6]:
```
from sklearn.linear_model import LogisticRegression  # for Logistic Regression␣
 ↪algorithm
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris
from sklearn.linear_model import LinearRegression
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error, r2_score
```

[7]:
```
iris=load_iris()
X = iris.data  # Features
y = iris.target
dir(iris)
```

[7]:
```
['DESCR',
 'data',
 'data_module',
 'feature_names',
 'filename',
 'frame',
 'target',
 'target_names']
```

[8]:
```python
os.getcwd()
```

[8]: 'C:\\Users\\hp\\Desktop\\BDA practicals(ET-2)'

[9]:
```python
df=pd.read_csv("C://Users//hp//Desktop//IRIS.csv")
```

[10]:
```python
df.head()
```

[10]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

[11]:
```python
df.tail()
```

[11]:

	sepal_length	sepal_width	petal_length	petal_width	species
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

[12]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

[13]:
```python
df.describe()
```

[13]:

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000

```
50%        5.800000     3.000000      4.350000     1.300000
75%        6.400000     3.300000      5.100000     1.800000
max        7.900000     4.400000      6.900000     2.500000
```

[14]:
```python
df.isnull()
```

[14]:
```
     sepal_length  sepal_width  petal_length  petal_width  species
0           False        False         False        False    False
1           False        False         False        False    False
2           False        False         False        False    False
3           False        False         False        False    False
4           False        False         False        False    False
..            …            …            …           …        …
145         False        False         False        False    False
146         False        False         False        False    False
147         False        False         False        False    False
148         False        False         False        False    False
149         False        False         False        False    False

[150 rows x 5 columns]
```

[15]:
```python
df.isna().sum()
```

[15]:
```
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
```

[16]:
```python
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
↪random_state=40)
```

[17]:
```python
model = LinearRegression()

# Train the model using the training sets
model.fit(X_train, y_train)

# Make predictions using the testing set
y_pred = model.predict(X_test)

# Coefficients
print('Coefficients:', model.coef_)
```

```
Coefficients: [-0.1502982  -0.04339123  0.25345042  0.58205165]
```

[18]:
```python
mse = mean_squared_error(y_test, y_pred)
print('Mean squared error: %.2f' % mse)

# Calculate coefficient of determination (R^2 score)
r2 = r2_score(y_test, y_pred)
print('Coefficient of determination (R^2 score): %.2f' % r2)
```

```
Mean squared error: 0.04
Coefficient of determination (R^2 score): 0.94
```

[ ]:
```python

```

3-logistic-regression

April 8, 2024

[1]:
```
# Aim: To perform and find the accuracy of Logistic Regression
```

[2]:
```
# Name: Samiksha Badhe
# Class: 3rd Year
# Section : B
# Roll no: 05
```

[3]:
```
import pandas as pd
```

[4]:
```
from sklearn.model_selection import train_test_split
```

[5]:
```
from sklearn.linear_model import LogisticRegression
```

[6]:
```
from sklearn.metrics import accuracy_score
```

[7]:
```
df=pd.read_csv('C:\\Users\\hp\\Desktop\\CHD_preprocessed.csv')
```

[8]:
```
df.head()
```

[8]:
```
   male  age  education  currentSmoker  cigsPerDay  BPMeds  prevalentStroke  \
0     1   39          1              0         0.0     0.0                0
1     0   46          0              0         0.0     0.0                0
2     1   48          0              1        20.0     0.0                0
3     0   61          1              1        30.0     0.0                0
4     0   46          1              1        23.0     0.0                0

   prevalentHyp  diabetes  totChol  sysBP  diaBP    BMI  heartRate  glucose  \
0             0         0    195.0  106.0   70.0  26.97       80.0     77.0
1             0         0    250.0  121.0   81.0  28.73       95.0     76.0
2             0         0    245.0  127.5   80.0  25.34       75.0     70.0
3             1         0    225.0  150.0   95.0  28.58       65.0    103.0
4             0         0    285.0  130.0   84.0  23.10       85.0     85.0

   TenYearCHD
0           0
1           0
2           0
```

```
3            1
4            0
```

[9]:
```
df.tail()
```

[9]:
```
      male  age  education  currentSmoker  cigsPerDay  BPMeds  \
4128     1   50          0              1         1.0     0.0
4129     1   51          1              1        43.0     0.0
4130     0   48          0              1        20.0     0.0
4131     0   44          0              1        15.0     0.0
4132     0   52          0              0         0.0     0.0

      prevalentStroke  prevalentHyp  diabetes  totChol  sysBP  diaBP    BMI  \
4128                0             1         0    313.0  179.0   92.0  25.97
4129                0             0         0    207.0  126.5   80.0  19.71
4130                0             0         0    248.0  131.0   72.0  22.00
4131                0             0         0    210.0  126.5   87.0  19.16
4132                0             0         0    269.0  133.5   83.0  21.47

      heartRate  glucose  TenYearCHD
4128       66.0     86.0           1
4129       65.0     68.0           0
4130       84.0     86.0           0
4131       86.0     82.0           0
4132       80.0    107.0           0
```

[10]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4133 entries, 0 to 4132
Data columns (total 16 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   male             4133 non-null   int64
 1   age              4133 non-null   int64
 2   education        4133 non-null   int64
 3   currentSmoker    4133 non-null   int64
 4   cigsPerDay       4133 non-null   float64
 5   BPMeds           4133 non-null   float64
 6   prevalentStroke  4133 non-null   int64
 7   prevalentHyp     4133 non-null   int64
 8   diabetes         4133 non-null   int64
 9   totChol          4133 non-null   float64
 10  sysBP            4133 non-null   float64
 11  diaBP            4133 non-null   float64
 12  BMI              4133 non-null   float64
 13  heartRate        4133 non-null   float64
```

```
 14  glucose          4133 non-null   float64
 15  TenYearCHD       4133 non-null   int64
dtypes: float64(8), int64(8)
memory usage: 516.8 KB
```

[11]:
```
df.describe()
```

[11]:
```
              male          age    education  currentSmoker   cigsPerDay  \
count  4133.000000  4133.000000  4133.000000    4133.000000  4133.000000
mean      0.427293    49.557222     0.280668       0.494798     9.101621
std       0.494745     8.561628     0.449380       0.500033    11.918440
min       0.000000    32.000000     0.000000       0.000000     0.000000
25%       0.000000    42.000000     0.000000       0.000000     0.000000
50%       0.000000    49.000000     0.000000       0.000000     0.000000
75%       1.000000    56.000000     1.000000       1.000000    20.000000
max       1.000000    70.000000     1.000000       1.000000    70.000000

            BPMeds  prevalentStroke  prevalentHyp     diabetes      totChol  \
count  4133.000000      4133.000000   4133.000000  4133.000000  4133.000000
mean      0.034358         0.006049      0.311154     0.025647   236.664408
std       0.182168         0.077548      0.463022     0.158100    43.909188
min       0.000000         0.000000      0.000000     0.000000   107.000000
25%       0.000000         0.000000      0.000000     0.000000   206.000000
50%       0.000000         0.000000      0.000000     0.000000   234.000000
75%       0.000000         0.000000      1.000000     0.000000   262.000000
max       1.000000         1.000000      1.000000     1.000000   600.000000

             sysBP        diaBP          BMI    heartRate      glucose  \
count  4133.000000  4133.000000  4133.000000  4133.000000  4133.000000
mean    132.367046    82.872248    25.778571    75.925236    81.946528
std      22.080332    11.952654     4.074360    12.049188    22.860954
min      83.500000    48.000000    15.540000    44.000000    40.000000
25%     117.000000    75.000000    23.060000    68.000000    72.000000
50%     128.000000    82.000000    25.380000    75.000000    80.000000
75%     144.000000    89.500000    27.990000    83.000000    85.000000
max     295.000000   142.500000    56.800000   143.000000   394.000000

        TenYearCHD
count  4133.000000
mean      0.151948
std       0.359014
min       0.000000
25%       0.000000
50%       0.000000
75%       0.000000
max       1.000000
```

[12]:
```python
df.isnull()
```

[12]:
```
       male    age  education  currentSmoker  cigsPerDay  BPMeds  \
0     False  False      False          False       False   False
1     False  False      False          False       False   False
2     False  False      False          False       False   False
3     False  False      False          False       False   False
4     False  False      False          False       False   False
…       …     …        …             …           …      …
4128  False  False      False          False       False   False
4129  False  False      False          False       False   False
4130  False  False      False          False       False   False
4131  False  False      False          False       False   False
4132  False  False      False          False       False   False

      prevalentStroke  prevalentHyp  diabetes  totChol  sysBP  diaBP    BMI  \
0               False         False     False    False  False  False  False
1               False         False     False    False  False  False  False
2               False         False     False    False  False  False  False
3               False         False     False    False  False  False  False
4               False         False     False    False  False  False  False
…                 …            …       …         …     …     …      …
4128            False         False     False    False  False  False  False
4129            False         False     False    False  False  False  False
4130            False         False     False    False  False  False  False
4131            False         False     False    False  False  False  False
4132            False         False     False    False  False  False  False

      heartRate  glucose  TenYearCHD
0         False    False       False
1         False    False       False
2         False    False       False
3         False    False       False
4         False    False       False
…          …      …           …
4128      False    False       False
4129      False    False       False
4130      False    False       False
4131      False    False       False
4132      False    False       False

[4133 rows x 16 columns]
```

[13]:
```python
df.isna().sum()
```

[13]:
```
male               0
age                0
```

```
education           0
currentSmoker       0
cigsPerDay          0
BPMeds              0
prevalentStroke     0
prevalentHyp        0
diabetes            0
totChol             0
sysBP               0
diaBP               0
BMI                 0
heartRate           0
glucose             0
TenYearCHD          0
dtype: int64
```

[14]:
```
x=df.drop("TenYearCHD",axis=1)
y=df['TenYearCHD']
```

[15]:
```
x
```

[15]:
```
      male  age  education  currentSmoker  cigsPerDay  BPMeds  \
0        1   39          1              0         0.0     0.0
1        0   46          0              0         0.0     0.0
2        1   48          0              1        20.0     0.0
3        0   61          1              1        30.0     0.0
4        0   46          1              1        23.0     0.0
…     …   …         …              …          …       …
4128     1   50          0              1         1.0     0.0
4129     1   51          1              1        43.0     0.0
4130     0   48          0              1        20.0     0.0
4131     0   44          0              1        15.0     0.0
4132     0   52          0              0         0.0     0.0

      prevalentStroke  prevalentHyp  diabetes  totChol  sysBP  diaBP    BMI  \
0                   0             0         0    195.0  106.0   70.0  26.97
1                   0             0         0    250.0  121.0   81.0  28.73
2                   0             0         0    245.0  127.5   80.0  25.34
3                   0             1         0    225.0  150.0   95.0  28.58
4                   0             0         0    285.0  130.0   84.0  23.10
…                …             …        …       …      …     …      …
4128                0             1         0    313.0  179.0   92.0  25.97
4129                0             0         0    207.0  126.5   80.0  19.71
4130                0             0         0    248.0  131.0   72.0  22.00
4131                0             0         0    210.0  126.5   87.0  19.16
4132                0             0         0    269.0  133.5   83.0  21.47
```

```
      heartRate  glucose
0          80.0     77.0
1          95.0     76.0
2          75.0     70.0
3          65.0    103.0
4          85.0     85.0
…           …       …
4128       66.0     86.0
4129       65.0     68.0
4130       84.0     86.0
4131       86.0     82.0
4132       80.0    107.0

[4133 rows x 15 columns]
```

[16]:
```python
y
```

[16]:
```
0       0
1       0
2       0
3       1
4       0
       ..
4128    1
4129    0
4130    0
4131    0
4132    0
Name: TenYearCHD, Length: 4133, dtype: int64
```

[17]:
```python
x_train,x_test,y_train,y_test= train_test_split(x,y,test_size=0.
 ↪2,random_state=42)
```

[18]:
```python
model = LogisticRegression(max_iter=1600)
```

[19]:
```python
model.fit(x_train,y_train)
model.score(x_train, y_train)
```

[19]:
```
0.8623714458560193
```

description-using-numpy-and-scipy

April 8, 2024

[1]:
```
# Aim: To perform finding Stastical mean, median, mode, standard deviation,␣
↪Variance using Numpy and Scipy
```

[2]:
```
# Name: Samiksha Badhe
# Class: 3rd Year
# Sec: B
# Roll No. : 05
```

[3]:
```
import numpy as np
from scipy import stats
```

[4]:
```
x=np.array([1,2,3,4,5,6,7,2,6,2,1,4,2,2,6])
```

[5]:
```
x
```

[5]:
```
array([1, 2, 3, 4, 5, 6, 7, 2, 6, 2, 1, 4, 2, 2, 6])
```

[6]:
```
print(np.mean(x))
```

```
3.533333333333333
```

[7]:
```
print(np.median(x))
```

```
3.0
```

[8]:
```
print(stats.mode(x))
```

```
ModeResult(mode=2, count=5)
```

[9]:
```
from scipy import stats
```

[10]:
```
print(stats.mode(x))
```

```
ModeResult(mode=2, count=5)
```

[11]:
```
print(np.std(x))
```

```
1.9618585292749546
```

[12]:
```
print(np.var(x))
```

3.8488888888888884

[13]:
```
import numpy as np
x=np.array([1,100,200,300,4000,5000])
y=np.array([2,4,6,8,10])
```

[14]:
```
print(np.std(x))
```

2072.711623024829

[15]:
```
print(np.std(y))
```

2.8284271247461903

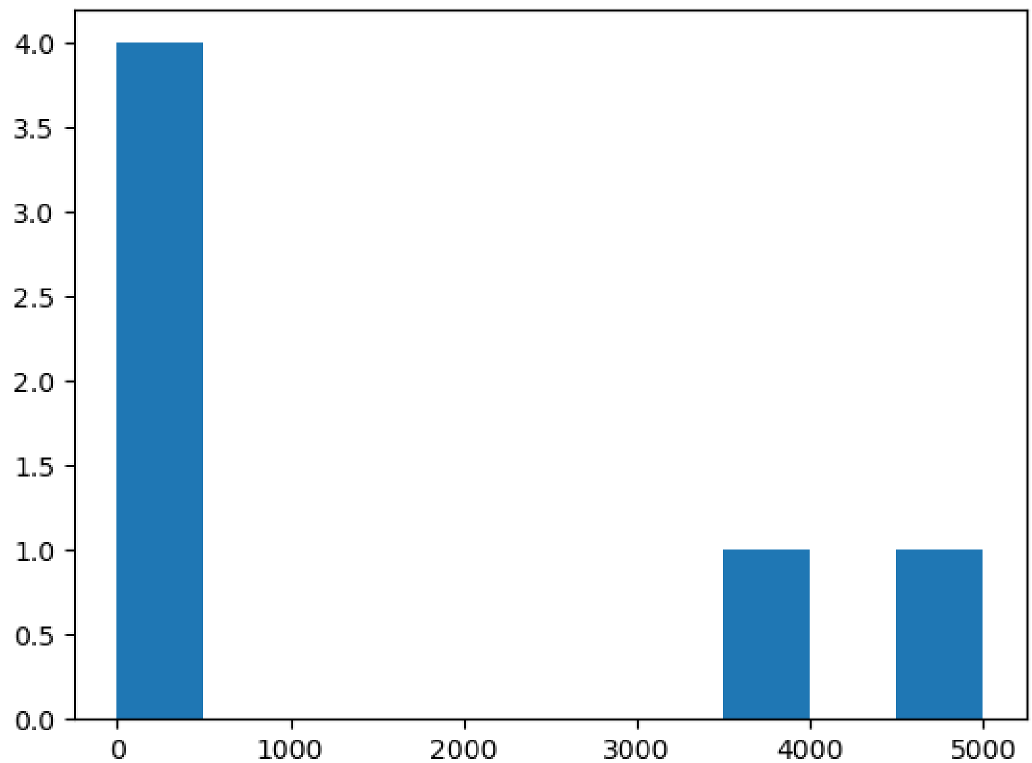
[16]:
```
print(np.var(x))
```

4296133.472222221

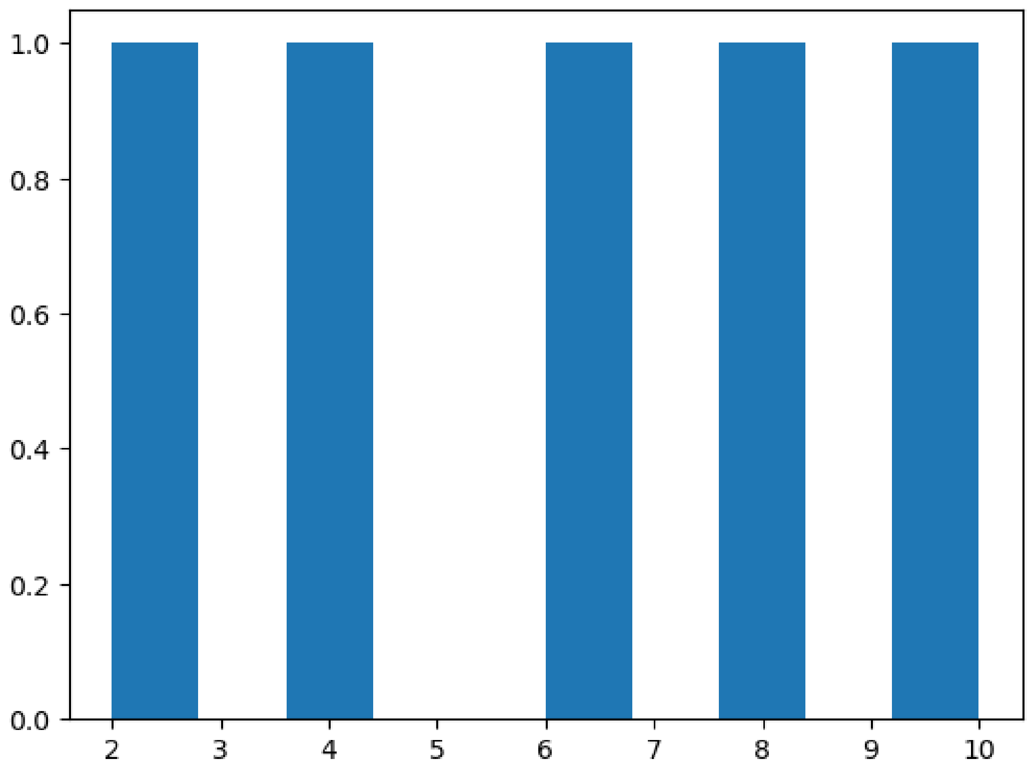
[17]:
```
print(np.var(y))
```

8.0

[18]:
```
from matplotlib import pyplot as plt
plt.hist(x)
plt.show()
```



[19]:
```
from matplotlib import pyplot as plt
plt.hist(y)
plt.show()
```



[20]:
```
from statsmodels.stats.weightstats import ztest as ztest
#enter IQ levels for 20 patients
data = [88, 92, 94, 94, 96, 97, 97, 97, 99, 99,
 105, 109, 109, 109, 110, 112, 112, 113, 114, 115]
#perform one sample z-test
ztest(data)
(1.5976240527147705, 0.1101266701438426)
```

[20]:
```
(1.5976240527147705, 0.1101266701438426)
```

5-f-test-anova

April 8, 2024

1 F-Test

[1]:
```
# Aim : To perform hypothesis testing using ANOVA (F-TEST) One-Way F-Test(Anova).
```

[2]:
```
# Name: Samiksha Badhe
# Class: 3rd Year
# Sec: B
# Roll No. : 05
```

[3]:
```
ages=[10,20,35,50,28,40,55,18,16,55,30,25,43,18,30,28,14,24,16,17,32,35,26,27,65,18,43,23,21,2
```

[4]:
```
len(ages)
```

[4]:
```
56
```

[5]:
```
import numpy as np
```

[6]:
```
sample_size=10
age_sample=np.random.choice(ages,sample_size)
```

[7]:
```
import scipy.stats
import numpy as np
```

[8]:
```
data1 = [0.0842, 0.0368, 0.0847, 0.0935, 0.0376, 0.0963, 0.0684,
0.0758, 0.0854, 0.0855]
data2 = [0.0785, 0.0845, 0.0758, 0.0853, 0.0946, 0.0785, 0.0853,
0.0685]
data3 = [0.0864, 0.2522, 0.0894, 0.2724, 0.0853, 0.1367, 0.853]
```

[9]:
```
# Performing the F-Test
f_test, p_val = scipy.stats.f_oneway(data1, data2, data3)
print("p-value is: ", p_val)
```

```
p-value is:  0.04043792126789144
```

[10]:
```
# taking the threshold value as 0.05 or 5%
if p_val < 0.05:
    print(" We can reject the null hypothesis")
else:
    print("We can accept the null hypothesis")
```

```
 We can reject the null hypothesis
```

[11]:
```
variance1 = np.var(data1)
```

[12]:
```
print(variance1)
```

```
0.00040949560000000005
```

[13]:
```
variance2 = np.var(data2)
```

[14]:
```
print(variance2)
```

```
5.3606874999999995e-05
```

[15]:
```
variance3 = np.var(data3)
```

[16]:
```
print(variance3)
```

```
0.06522053346938775
```

[ ]:
```

```

6-t-test

April 8, 2024

1 T Test

[1]:
```
# Aim : To perform hypothesis testing using T test.
```

[2]:
```
# Name: Samiksha Badhe
# Class: 3rd Year
# Sec: B
# Roll No. : 05
```

T Test A t-test is a type of inferential statistic which is used to determine if there is a significant difference between the means of two groups which may be related in certain features

[3]:
```
ages=[10,20,35,50,28,40,55,18,16,55,30,25,43,18,30,28,14,24,16,17,32,35,26,27,65,18,43,23,21,2
```

[4]:
```
len(ages)
```

[4]:
```
32
```

[5]:
```
import numpy as np
ages_mean=np.mean(ages)
print(ages_mean)
```

```
30.34375
```

[6]:
```
sample_size=10
age_sample=np.random.choice(ages,sample_size)
```

[7]:
```
age_sample
```

[7]:
```
array([18, 14, 70, 21, 50, 18, 18, 18, 50, 55])
```

[8]:
```
from scipy.stats import ttest_1samp
```

[9]:
```
ttest,p_value=ttest_1samp(age_sample,30)
```

[10]:
```
print(p_value)
```

```
0.6357349574999751
```

```
[11]: if p_value < 0.05:    # alpha value is 0.05 or 5%
          print(" we are rejecting null hypothesis")
      else:
          print("we are accepting null hypothesis")
```

```
we are accepting null hypothesis
```

7-z-test

April 8, 2024

1 Z Test

[1]:
```
# Aim : To perform hypothesis testing using Z test.
```

[2]:
```
# Name: Samiksha Badhe
# Class: 3rd Year
# Sec: B
# Roll No. : 05
```

[3]:
```
ages=[10,20,35,50,28,40,55,18,16,55,30,25,43,18,30,28,14,24,16,17,32,35,26,27,65,18,43,23,21,2
```

[4]:
```
len(ages)
```

[4]:
```
32
```

[5]:
```
import numpy as np
ages_mean=np.mean(ages)
print(ages_mean)
```

```
30.34375
```

[6]:
```
## Lets take sample

sample_size=31
age_sample=np.random.choice(ages,sample_size)
```

[7]:
```
age_sample
```

[7]:
```
array([24, 24, 55, 16, 65, 55, 35, 14, 65, 20, 28, 43, 10, 10, 30, 30, 55,
       32, 20, 17, 23, 32, 17, 70, 27, 16, 16, 18, 20, 19, 35])
```

[8]:
```
# from scipy.stats import ztest_1samp
```

[9]:
```
from statsmodels.stats import weightstats as stests

# Perform one-sample z-test
ztest, p_value = stests.ztest(age_sample)
```

```python
# Print the results
print("ztest", ztest)
print("P-value:", p_value)
```

```
ztest 9.851701971870249
P-value: 6.73929402110435e-23
```

[10]:
```python
if p_value < 0.05:    # alpha value is 0.05 or 5%    (Level of significance)
    print(" we are rejecting null hypothesis")
else:
    print("we are accepting null hypothesis")
```

```
 we are rejecting null hypothesis
```

[ ]:
```python

```

8-knn-classifier

April 8, 2024

[1]:
```
# Aim: To perform and find the accuracy of K-Nearest Neighbors Algorithm i.e.␣
↪KNN Classifier
```

[2]:
```
# Name: Samiksha Badhe
# Class: 3rd Year
# Sec: B
# Roll No. : 05
```

[3]:
```
import pandas as pd
import os
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from sklearn.model_selection import train_test_split
import warnings
warnings.filterwarnings('ignore')
```

[4]:
```
df=pd.read_csv('C:\\Users\\hp\\Desktop\\CHD_preprocessed.csv')
```

[5]:
```
df.head()
```

[5]:
```
   male  age  education  currentSmoker  cigsPerDay  BPMeds  prevalentStroke  \
0     1   39          1              0         0.0     0.0                0
1     0   46          0              0         0.0     0.0                0
2     1   48          0              1        20.0     0.0                0
3     0   61          1              1        30.0     0.0                0
4     0   46          1              1        23.0     0.0                0

   prevalentHyp  diabetes  totChol  sysBP  diaBP    BMI  heartRate  glucose  \
0             0         0    195.0  106.0   70.0  26.97       80.0     77.0
1             0         0    250.0  121.0   81.0  28.73       95.0     76.0
2             0         0    245.0  127.5   80.0  25.34       75.0     70.0
3             1         0    225.0  150.0   95.0  28.58       65.0    103.0
4             0         0    285.0  130.0   84.0  23.10       85.0     85.0

   TenYearCHD
0           0
```

```
1          0
2          0
3          1
4          0
```

[6]:
```python
df.tail()
```

[6]:
```
      male  age  education  currentSmoker  cigsPerDay  BPMeds  \
4128     1   50          0              1         1.0     0.0
4129     1   51          1              1        43.0     0.0
4130     0   48          0              1        20.0     0.0
4131     0   44          0              1        15.0     0.0
4132     0   52          0              0         0.0     0.0

      prevalentStroke  prevalentHyp  diabetes  totChol  sysBP  diaBP    BMI  \
4128                0             1         0    313.0  179.0   92.0  25.97
4129                0             0         0    207.0  126.5   80.0  19.71
4130                0             0         0    248.0  131.0   72.0  22.00
4131                0             0         0    210.0  126.5   87.0  19.16
4132                0             0         0    269.0  133.5   83.0  21.47

      heartRate  glucose  TenYearCHD
4128       66.0     86.0           1
4129       65.0     68.0           0
4130       84.0     86.0           0
4131       86.0     82.0           0
4132       80.0    107.0           0
```

[7]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4133 entries, 0 to 4132
Data columns (total 16 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   male             4133 non-null   int64
 1   age              4133 non-null   int64
 2   education        4133 non-null   int64
 3   currentSmoker    4133 non-null   int64
 4   cigsPerDay       4133 non-null   float64
 5   BPMeds           4133 non-null   float64
 6   prevalentStroke  4133 non-null   int64
 7   prevalentHyp     4133 non-null   int64
 8   diabetes         4133 non-null   int64
 9   totChol          4133 non-null   float64
 10  sysBP            4133 non-null   float64
 11  diaBP            4133 non-null   float64
```

```
 12  BMI              4133 non-null   float64
 13  heartRate        4133 non-null   float64
 14  glucose          4133 non-null   float64
 15  TenYearCHD       4133 non-null   int64
dtypes: float64(8), int64(8)
memory usage: 516.8 KB
```

[8]:
```
df.describe()
```

[8]:
```
              male          age    education  currentSmoker   cigsPerDay  \
count  4133.000000  4133.000000  4133.000000    4133.000000  4133.000000
mean      0.427293    49.557222     0.280668       0.494798     9.101621
std       0.494745     8.561628     0.449380       0.500033    11.918440
min       0.000000    32.000000     0.000000       0.000000     0.000000
25%       0.000000    42.000000     0.000000       0.000000     0.000000
50%       0.000000    49.000000     0.000000       0.000000     0.000000
75%       1.000000    56.000000     1.000000       1.000000    20.000000
max       1.000000    70.000000     1.000000       1.000000    70.000000

            BPMeds  prevalentStroke  prevalentHyp     diabetes      totChol  \
count  4133.000000      4133.000000   4133.000000  4133.000000  4133.000000
mean      0.034358         0.006049      0.311154     0.025647   236.664408
std       0.182168         0.077548      0.463022     0.158100    43.909188
min       0.000000         0.000000      0.000000     0.000000   107.000000
25%       0.000000         0.000000      0.000000     0.000000   206.000000
50%       0.000000         0.000000      0.000000     0.000000   234.000000
75%       0.000000         0.000000      1.000000     0.000000   262.000000
max       1.000000         1.000000      1.000000     1.000000   600.000000

             sysBP        diaBP          BMI    heartRate      glucose  \
count  4133.000000  4133.000000  4133.000000  4133.000000  4133.000000
mean    132.367046    82.872248    25.778571    75.925236    81.946528
std      22.080332    11.952654     4.074360    12.049188    22.860954
min      83.500000    48.000000    15.540000    44.000000    40.000000
25%     117.000000    75.000000    23.060000    68.000000    72.000000
50%     128.000000    82.000000    25.380000    75.000000    80.000000
75%     144.000000    89.500000    27.990000    83.000000    85.000000
max     295.000000   142.500000    56.800000   143.000000   394.000000

        TenYearCHD
count  4133.000000
mean      0.151948
std       0.359014
min       0.000000
25%       0.000000
50%       0.000000
75%       0.000000
```

```
max        1.000000
```

[9]:
```python
df.isna().sum()
```

[9]:
```
male               0
age                0
education          0
currentSmoker      0
cigsPerDay         0
BPMeds             0
prevalentStroke    0
prevalentHyp       0
diabetes           0
totChol            0
sysBP              0
diaBP              0
BMI                0
heartRate          0
glucose            0
TenYearCHD         0
dtype: int64
```

[18]:
```python
# Splitting the dependent and independent variables
x = df.drop('TenYearCHD',axis=1)
y = df['TenYearCHD']
```

[19]:
```python
x #Checking the features
```

[19]:
```
      male  age  education  currentSmoker  cigsPerDay  BPMeds  \
0        1   39          1              0         0.0     0.0
1        0   46          0              0         0.0     0.0
2        1   48          0              1        20.0     0.0
3        0   61          1              1        30.0     0.0
4        0   46          1              1        23.0     0.0
…     …   …          …              …           …       …
4128     1   50          0              1         1.0     0.0
4129     1   51          1              1        43.0     0.0
4130     0   48          0              1        20.0     0.0
4131     0   44          0              1        15.0     0.0
4132     0   52          0              0         0.0     0.0

      prevalentStroke  prevalentHyp  diabetes  totChol  sysBP  diaBP    BMI  \
0                   0             0         0    195.0  106.0   70.0  26.97
1                   0             0         0    250.0  121.0   81.0  28.73
2                   0             0         0    245.0  127.5   80.0  25.34
3                   0             1         0    225.0  150.0   95.0  28.58
4                   0             0         0    285.0  130.0   84.0  23.10
```

```
…                …              …        …       …     …     …      …
4128              0              1        0     313.0  179.0    92.0  25.97
4129              0              0        0     207.0  126.5    80.0  19.71
4130              0              0        0     248.0  131.0    72.0  22.00
4131              0              0        0     210.0  126.5    87.0  19.16
4132              0              0        0     269.0  133.5    83.0  21.47

      heartRate  glucose
0          80.0     77.0
1          95.0     76.0
2          75.0     70.0
3          65.0    103.0
4          85.0     85.0
…           …      …
4128       66.0     86.0
4129       65.0     68.0
4130       84.0     86.0
4131       86.0     82.0
4132       80.0    107.0

[4133 rows x 15 columns]
```

1 Train Test Split

[20]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.
↪2,random_state=42)
```

[21]:
```
y_train
```

[21]:
```
173     1
1022    0
3182    0
331     1
2222    0
       ..
3444    0
466     0
3092    0
3772    0
860     0
Name: TenYearCHD, Length: 3306, dtype: int64
```

[22]:
```
y_test
```

[22]:
```
1864    0
1210    0
```

```
1924    0
1752    0
1095    0
       ..
881     0
25      1
3256    0
2269    0
1074    0
Name: TenYearCHD, Length: 827, dtype: int64
```

[23]:

```
x_train
```

[23]:

```
      male  age  education  currentSmoker  cigsPerDay  BPMeds  \
173      0   60          1              0         0.0     0.0
1022     1   42          1              1        20.0     0.0
3182     1   58          0              0         0.0     0.0
331      0   58          0              0         0.0     0.0
2222     1   39          1              0         0.0     0.0
…     …   …         …            …          …      …
3444     0   49          0              0         0.0     0.0
466      1   50          0              0         0.0     0.0
3092     0   36          0              0         0.0     0.0
3772     0   64          0              0         0.0     0.0
860      0   47          0              0         0.0     0.0

      prevalentStroke  prevalentHyp  diabetes  totChol  sysBP  diaBP    BMI  \
173                 0             1         0    325.0  182.0  106.0  27.61
1022                0             0         0    270.0  112.0   77.0  24.77
3182                0             0         0    225.0  105.5   74.0  25.68
331                 0             1         0    200.0  158.0  101.0  23.06
2222                0             1         0    208.0  146.0   92.0  25.91
…                …             …        …        …      …      …      …
3444                0             1         0    233.0  149.0   91.5  26.03
466                 0             1         0    219.0  145.0  100.0  26.26
3092                0             0         0    209.0  107.0   73.5  21.59
3772                0             1         0    279.0  172.0   87.0  24.01
860                 0             0         0    232.0  113.5   73.0  28.78

      heartRate  glucose
173        80.0     77.0
1022       73.0     85.0
3182       50.0     93.0
331        85.0     77.0
2222       69.0     74.0
…           …       …
3444       68.0     82.0
```

```
466         78.0    108.0
3092        75.0     73.0
3772        80.0     70.0
860         75.0     77.0

[3306 rows x 15 columns]
```

[24]:
```
x_test
```

[24]:
```
      male  age  education  currentSmoker  cigsPerDay  BPMeds  \
1864     1   40          0              1         5.0     0.0
1210     0   50          0              1        10.0     0.0
1924     0   64          1              0         0.0     0.0
1752     0   55          0              0         0.0     1.0
1095     0   46          1              0         0.0     0.0
…    …   …       …         …        …      …
881      0   44          0              1         1.0     0.0
25       1   47          1              1        20.0     0.0
3256     0   63          1              0         0.0     0.0
2269     1   40          0              1        20.0     0.0
1074     1   57          1              0         0.0     1.0

      prevalentStroke  prevalentHyp  diabetes  totChol  sysBP  diaBP    BMI  \
1864                0             0         0    282.0  120.0   87.0  22.98
1210                0             1         0    298.0  156.0   90.0  24.24
1924                0             0         0    330.0  108.0   82.0  23.09
1752                0             1         0    285.0  158.0   98.0  30.23
1095                0             1         0    259.0  173.0  102.0  27.22
…               …            …        …       …      …     …     …
881                 0             0         0    217.0  124.5   82.0  22.36
25                  0             0         0    294.0  102.0   68.0  24.18
3256                0             0         0    297.0  133.5   92.0  25.09
2269                0             0         0    193.0  122.0   78.0  28.40
1074                0             1         0    195.0  162.0  108.0  32.65

      heartRate  glucose
1864       60.0     82.0
1210       75.0    100.0
1924       85.0     80.0
1752       70.0     88.0
1095       85.0     75.0
…          …       …
881        87.0     68.0
25         62.0     66.0
3256       80.0     74.0
2269       70.0     93.0
1074       85.0     73.0
```

[827 rows x 15 columns]

[25]:
```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=5, p=2, metric='minkowski')
knn.fit(x_train, y_train)
acc = knn.score(x_test, y_test)*100
print(acc)
```

81.62031438935912

[ ]:
```
```

9-k-means

April 8, 2024

[1]:
```
#Aim: To perform and find the accuracy of K means algorithm
```

[ ]:
```
# Name: Samiksha Badhe
# Class: 3rd Year
# Sec: B
# Roll No. : 05
```

```
Running cells with 'c:\Users\ASUS-PC\AppData\Local\Microsoft\WindowsApps\python3.
↪11.exe' requires the ipykernel package.

Run the following command to install 'ipykernel' into the Python environment.

Command: 'c:/Users/ASUS-PC/AppData/Local/Microsoft/WindowsApps/python3.11.exe -m
↪pip install ipykernel -U --user --force-reinstall'
```

[2]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt # for data visualization
import seaborn as sns # for statistical data visualization
%matplotlib inline
from sklearn.cluster import KMeans
from sklearn.metrics import adjusted_rand_score
from sklearn.cluster import KMeans
import warnings
warnings.filterwarnings('ignore')
```

[3]:
```
df=pd.read_csv('C:\\Users\\hp\\Desktop\\CHD_preprocessed.csv')
```

[4]:
```
df.head()
```

[4]:
```
   male  age  education  currentSmoker  cigsPerDay  BPMeds  prevalentStroke  \
0     1   39          1              0         0.0     0.0                0
1     0   46          0              0         0.0     0.0                0
2     1   48          0              1        20.0     0.0                0
3     0   61          1              1        30.0     0.0                0
4     0   46          1              1        23.0     0.0                0
```

```
   prevalentHyp  diabetes  totChol  sysBP  diaBP    BMI  heartRate  glucose  \
0             0         0    195.0  106.0   70.0  26.97       80.0     77.0
1             0         0    250.0  121.0   81.0  28.73       95.0     76.0
2             0         0    245.0  127.5   80.0  25.34       75.0     70.0
3             1         0    225.0  150.0   95.0  28.58       65.0    103.0
4             0         0    285.0  130.0   84.0  23.10       85.0     85.0

   TenYearCHD
0           0
1           0
2           0
3           1
4           0
```

[5]:
```
df.tail()
```

[5]:
```
      male  age  education  currentSmoker  cigsPerDay  BPMeds  \
4128     1   50          0              1         1.0     0.0
4129     1   51          1              1        43.0     0.0
4130     0   48          0              1        20.0     0.0
4131     0   44          0              1        15.0     0.0
4132     0   52          0              0         0.0     0.0

      prevalentStroke  prevalentHyp  diabetes  totChol  sysBP  diaBP    BMI  \
4128                0             1         0    313.0  179.0   92.0  25.97
4129                0             0         0    207.0  126.5   80.0  19.71
4130                0             0         0    248.0  131.0   72.0  22.00
4131                0             0         0    210.0  126.5   87.0  19.16
4132                0             0         0    269.0  133.5   83.0  21.47

      heartRate  glucose  TenYearCHD
4128       66.0     86.0           1
4129       65.0     68.0           0
4130       84.0     86.0           0
4131       86.0     82.0           0
4132       80.0    107.0           0
```

[6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4133 entries, 0 to 4132
Data columns (total 16 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   male             4133 non-null   int64
 1   age              4133 non-null   int64
```

```
 2   education        4133 non-null   int64  
 3   currentSmoker    4133 non-null   int64  
 4   cigsPerDay       4133 non-null   float64
 5   BPMeds           4133 non-null   float64
 6   prevalentStroke  4133 non-null   int64  
 7   prevalentHyp     4133 non-null   int64  
 8   diabetes         4133 non-null   int64  
 9   totChol          4133 non-null   float64
 10  sysBP            4133 non-null   float64
 11  diaBP            4133 non-null   float64
 12  BMI              4133 non-null   float64
 13  heartRate        4133 non-null   float64
 14  glucose          4133 non-null   float64
 15  TenYearCHD       4133 non-null   int64  
dtypes: float64(8), int64(8)
memory usage: 516.8 KB
```

[7]:
```
df.size
```

[7]: 66128

[8]:
```
df.shape
```

[8]: (4133, 16)

[9]:
```
df.describe()
```

[9]:
```
              male          age    education  currentSmoker   cigsPerDay  \
count  4133.000000  4133.000000  4133.000000    4133.000000  4133.000000   
mean      0.427293    49.557222     0.280668       0.494798     9.101621   
std       0.494745     8.561628     0.449380       0.500033    11.918440   
min       0.000000    32.000000     0.000000       0.000000     0.000000   
25%       0.000000    42.000000     0.000000       0.000000     0.000000   
50%       0.000000    49.000000     0.000000       0.000000     0.000000   
75%       1.000000    56.000000     1.000000       1.000000    20.000000   
max       1.000000    70.000000     1.000000       1.000000    70.000000   

            BPMeds  prevalentStroke  prevalentHyp     diabetes      totChol  \
count  4133.000000      4133.000000   4133.000000  4133.000000  4133.000000   
mean      0.034358         0.006049      0.311154     0.025647   236.664408   
std       0.182168         0.077548      0.463022     0.158100    43.909188   
min       0.000000         0.000000      0.000000     0.000000   107.000000   
25%       0.000000         0.000000      0.000000     0.000000   206.000000   
50%       0.000000         0.000000      0.000000     0.000000   234.000000   
75%       0.000000         0.000000      1.000000     0.000000   262.000000   
max       1.000000         1.000000      1.000000     1.000000   600.000000   
```

```
              sysBP        diaBP          BMI    heartRate      glucose  \
count  4133.000000  4133.000000  4133.000000  4133.000000  4133.000000
mean    132.367046    82.872248    25.778571    75.925236    81.946528
std      22.080332    11.952654     4.074360    12.049188    22.860954
min      83.500000    48.000000    15.540000    44.000000    40.000000
25%     117.000000    75.000000    23.060000    68.000000    72.000000
50%     128.000000    82.000000    25.380000    75.000000    80.000000
75%     144.000000    89.500000    27.990000    83.000000    85.000000
max     295.000000   142.500000    56.800000   143.000000   394.000000

        TenYearCHD
count  4133.000000
mean      0.151948
std       0.359014
min       0.000000
25%       0.000000
50%       0.000000
75%       0.000000
max       1.000000
```

[10]:
```
X = df.drop(columns=['TenYearCHD'])
kmeans = KMeans(n_clusters=2, random_state=0)
```

[11]:
```
kmeans.fit(X)
kmeans.cluster_centers_
kmeans.inertia_
```

[11]:
```
9282994.90372527
```

[ ]:
```
```

10-naive-bayes-classifier

April 8, 2024

[1]:
```
# Aim: To perform and find the accuracy of Naive bayes Classifier
```

[2]:
```
# Name: Samiksha Badhe
# Class: 3rd Year
# Sec: B
# Roll No. : 05
```

[3]:
```
import pandas as pd
import os
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
import warnings
warnings.filterwarnings('ignore')
```

[4]:
```
df=pd.read_csv('C:\\Users\\hp\\Desktop\\CHD_preprocessed.csv')
```

[5]:
```
df.head()
```

[5]:
```
   male  age  education  currentSmoker  cigsPerDay  BPMeds  prevalentStroke  \
0     1   39          1              0         0.0     0.0                0
1     0   46          0              0         0.0     0.0                0
2     1   48          0              1        20.0     0.0                0
3     0   61          1              1        30.0     0.0                0
4     0   46          1              1        23.0     0.0                0

   prevalentHyp  diabetes  totChol  sysBP  diaBP    BMI  heartRate  glucose  \
0             0         0    195.0  106.0   70.0  26.97       80.0     77.0
1             0         0    250.0  121.0   81.0  28.73       95.0     76.0
2             0         0    245.0  127.5   80.0  25.34       75.0     70.0
3             1         0    225.0  150.0   95.0  28.58       65.0    103.0
4             0         0    285.0  130.0   84.0  23.10       85.0     85.0

   TenYearCHD
0           0
```

```
1            0
2            0
3            1
4            0
```

[6]:
```
df.tail()
```

[6]:
```
      male  age  education  currentSmoker  cigsPerDay  BPMeds  \
4128     1   50          0              1         1.0     0.0
4129     1   51          1              1        43.0     0.0
4130     0   48          0              1        20.0     0.0
4131     0   44          0              1        15.0     0.0
4132     0   52          0              0         0.0     0.0

      prevalentStroke  prevalentHyp  diabetes  totChol  sysBP  diaBP    BMI  \
4128                0             1         0    313.0  179.0   92.0  25.97
4129                0             0         0    207.0  126.5   80.0  19.71
4130                0             0         0    248.0  131.0   72.0  22.00
4131                0             0         0    210.0  126.5   87.0  19.16
4132                0             0         0    269.0  133.5   83.0  21.47

      heartRate  glucose  TenYearCHD
4128       66.0     86.0           1
4129       65.0     68.0           0
4130       84.0     86.0           0
4131       86.0     82.0           0
4132       80.0    107.0           0
```

[7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4133 entries, 0 to 4132
Data columns (total 16 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   male             4133 non-null   int64
 1   age              4133 non-null   int64
 2   education        4133 non-null   int64
 3   currentSmoker    4133 non-null   int64
 4   cigsPerDay       4133 non-null   float64
 5   BPMeds           4133 non-null   float64
 6   prevalentStroke  4133 non-null   int64
 7   prevalentHyp     4133 non-null   int64
 8   diabetes         4133 non-null   int64
 9   totChol          4133 non-null   float64
 10  sysBP            4133 non-null   float64
 11  diaBP            4133 non-null   float64
```

```
 12  BMI              4133 non-null   float64
 13  heartRate        4133 non-null   float64
 14  glucose          4133 non-null   float64
 15  TenYearCHD       4133 non-null   int64
dtypes: float64(8), int64(8)
memory usage: 516.8 KB
```

[8]:
```
df.describe()
```

[8]:
```
              male          age    education  currentSmoker   cigsPerDay  \
count  4133.000000  4133.000000  4133.000000    4133.000000  4133.000000
mean      0.427293    49.557222     0.280668       0.494798     9.101621
std       0.494745     8.561628     0.449380       0.500033    11.918440
min       0.000000    32.000000     0.000000       0.000000     0.000000
25%       0.000000    42.000000     0.000000       0.000000     0.000000
50%       0.000000    49.000000     0.000000       0.000000     0.000000
75%       1.000000    56.000000     1.000000       1.000000    20.000000
max       1.000000    70.000000     1.000000       1.000000    70.000000

            BPMeds  prevalentStroke  prevalentHyp     diabetes      totChol  \
count  4133.000000      4133.000000   4133.000000  4133.000000  4133.000000
mean      0.034358         0.006049      0.311154     0.025647   236.664408
std       0.182168         0.077548      0.463022     0.158100    43.909188
min       0.000000         0.000000      0.000000     0.000000   107.000000
25%       0.000000         0.000000      0.000000     0.000000   206.000000
50%       0.000000         0.000000      0.000000     0.000000   234.000000
75%       0.000000         0.000000      1.000000     0.000000   262.000000
max       1.000000         1.000000      1.000000     1.000000   600.000000

             sysBP        diaBP          BMI    heartRate      glucose  \
count  4133.000000  4133.000000  4133.000000  4133.000000  4133.000000
mean    132.367046    82.872248    25.778571    75.925236    81.946528
std      22.080332    11.952654     4.074360    12.049188    22.860954
min      83.500000    48.000000    15.540000    44.000000    40.000000
25%     117.000000    75.000000    23.060000    68.000000    72.000000
50%     128.000000    82.000000    25.380000    75.000000    80.000000
75%     144.000000    89.500000    27.990000    83.000000    85.000000
max     295.000000   142.500000    56.800000   143.000000   394.000000

        TenYearCHD
count  4133.000000
mean      0.151948
std       0.359014
min       0.000000
25%       0.000000
50%       0.000000
75%       0.000000
```

```
max        1.000000
```

[9]:
```python
df.size
```

[9]:
```
66128
```

[10]:
```python
df.shape
```

[10]:
```
(4133, 16)
```

[11]:
```python
df.isna().sum()
```

[11]:
```
male               0
age                0
education          0
currentSmoker      0
cigsPerDay         0
BPMeds             0
prevalentStroke    0
prevalentHyp       0
diabetes           0
totChol            0
sysBP              0
diaBP              0
BMI                0
heartRate          0
glucose            0
TenYearCHD         0
dtype: int64
```

[12]:
```python
x = df.drop("TenYearCHD",axis=1)
y = df['TenYearCHD']
```

[13]:
```python
x
```

[13]:
```
      male  age  education  currentSmoker  cigsPerDay  BPMeds  \
0        1   39          1              0         0.0     0.0
1        0   46          0              0         0.0     0.0
2        1   48          0              1        20.0     0.0
3        0   61          1              1        30.0     0.0
4        0   46          1              1        23.0     0.0
…     …   …          …              …           …      …
4128     1   50          0              1         1.0     0.0
4129     1   51          1              1        43.0     0.0
4130     0   48          0              1        20.0     0.0
4131     0   44          0              1        15.0     0.0
4132     0   52          0              0         0.0     0.0
```

```
      prevalentStroke  prevalentHyp  diabetes  totChol  sysBP  diaBP    BMI  \
0                   0             0         0    195.0  106.0   70.0  26.97
1                   0             0         0    250.0  121.0   81.0  28.73
2                   0             0         0    245.0  127.5   80.0  25.34
3                   0             1         0    225.0  150.0   95.0  28.58
4                   0             0         0    285.0  130.0   84.0  23.10
…                 …             …       …       …     …     …     …
4128                0             1         0    313.0  179.0   92.0  25.97
4129                0             0         0    207.0  126.5   80.0  19.71
4130                0             0         0    248.0  131.0   72.0  22.00
4131                0             0         0    210.0  126.5   87.0  19.16
4132                0             0         0    269.0  133.5   83.0  21.47

      heartRate  glucose
0          80.0     77.0
1          95.0     76.0
2          75.0     70.0
3          65.0    103.0
4          85.0     85.0
…           …       …
4128       66.0     86.0
4129       65.0     68.0
4130       84.0     86.0
4131       86.0     82.0
4132       80.0    107.0

[4133 rows x 15 columns]
```

[14]:
```
y
```

[14]:
```
0       0
1       0
2       0
3       1
4       0
       ..
4128    1
4129    0
4130    0
4131    0
4132    0
Name: TenYearCHD, Length: 4133, dtype: int64
```

[15]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.
↪2,random_state=42)
```

[16]:
```
y_train
```

[16]:
```
173     1
1022    0
3182    0
331     1
2222    0
       ..
3444    0
466     0
3092    0
3772    0
860     0
Name: TenYearCHD, Length: 3306, dtype: int64
```

[17]:
```
y_test
```

[17]:
```
1864    0
1210    0
1924    0
1752    0
1095    0
       ..
881     0
25      1
3256    0
2269    0
1074    0
Name: TenYearCHD, Length: 827, dtype: int64
```

[18]:
```
nb_model = GaussianNB()
nb_model.fit(x_train, y_train)
```

[18]:
```
GaussianNB()
```

[19]:
```
# Evaluate the model
train_accuracy = nb_model.score(x_train, y_train)
test_accuracy = nb_model.score(x_test, y_test)
```

[20]:
```
print("Training Accuracy:", train_accuracy)
print("Testing Accuracy:", test_accuracy)
```

```
Training Accuracy: 0.8236539624924379
Testing Accuracy: 0.8101571946795647
```

11-data-visualization

April 8, 2024

[1]:
```
import numpy as np
from matplotlib import pyplot as plt
```

[2]:
```
# Name: Samiksha Badhe
# Class: 3rd Year
# Sec: B
# Roll No. : 05
```

[3]:
```
x=np.arange(1,11)
```

[4]:
```
x
```

[4]:
```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10])
```

[5]:
```
y=x*2
```

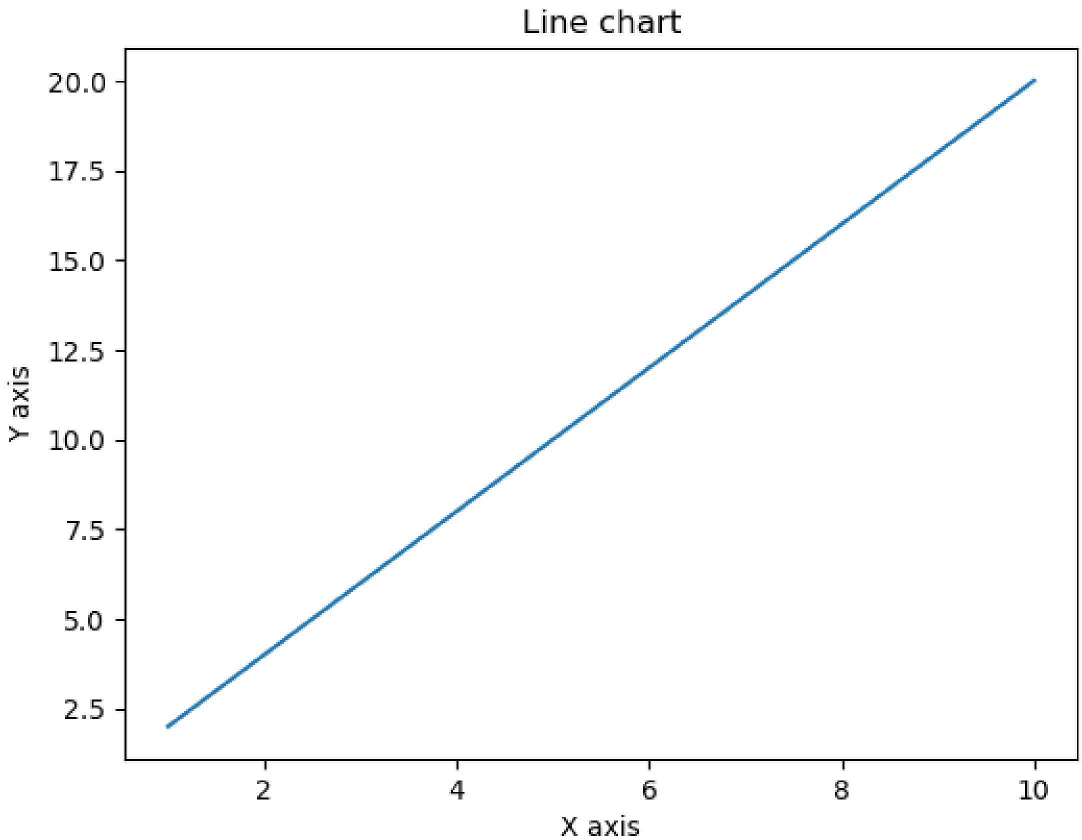
[6]:
```
x
```

[6]:
```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10])
```

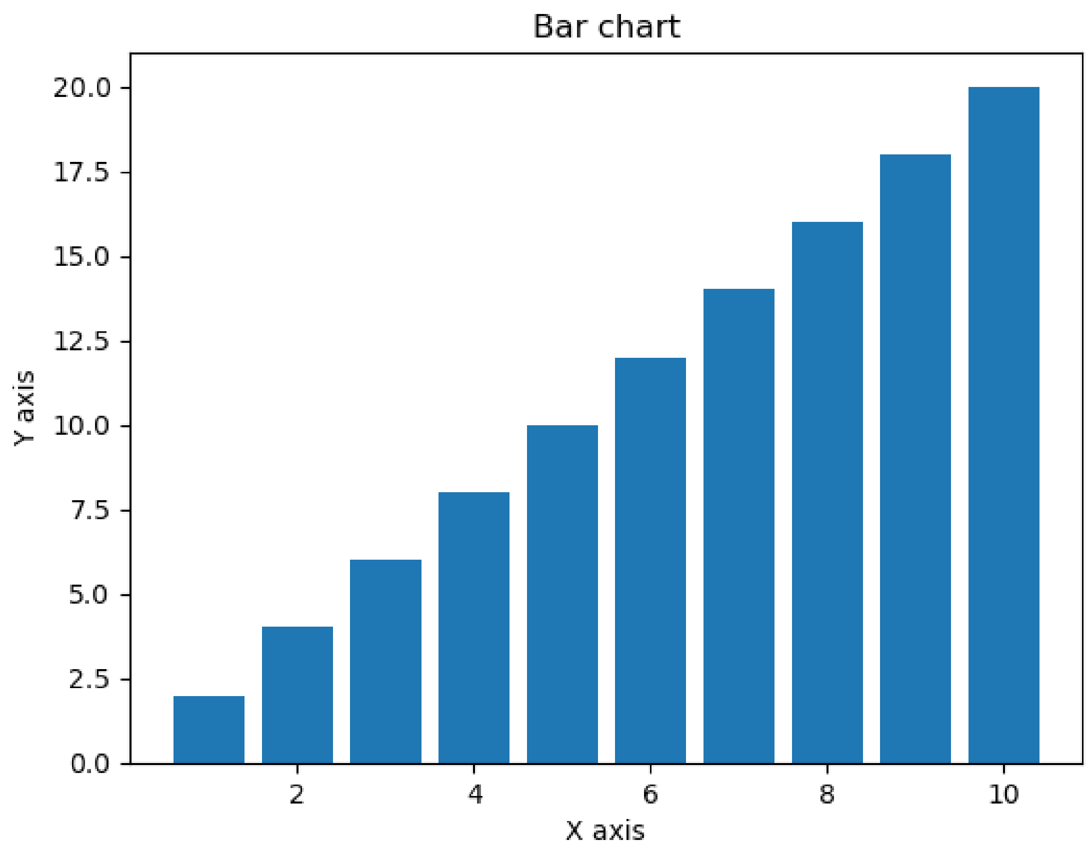
[7]:
```
y
```

[7]:
```
array([ 2,  4,  6,  8, 10, 12, 14, 16, 18, 20])
```

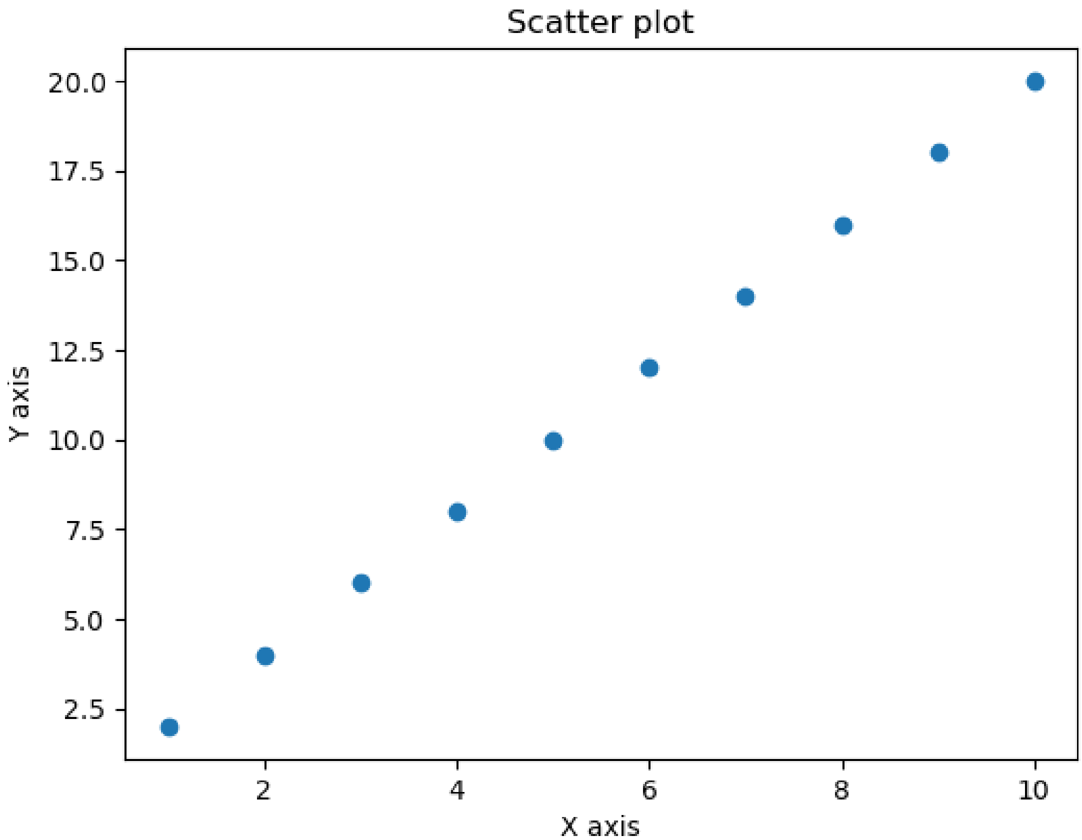
[8]:
```
#Line chart
plt.plot(x,y)
plt.title("Line chart")
plt.xlabel("X axis")
plt.ylabel("Y axis")
plt.show()
```



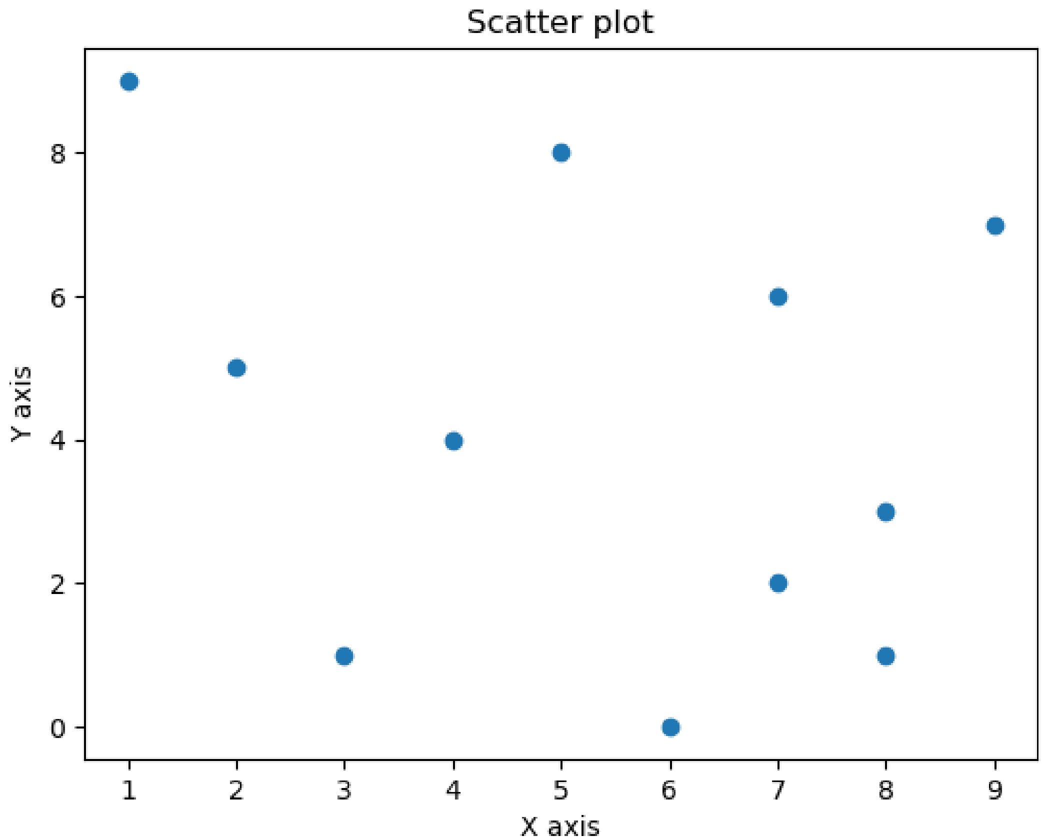
[9]:
```
plt.bar(x,y)
plt.title("Bar chart")
plt.xlabel("X axis")
plt.ylabel("Y axis")
plt.show()
```



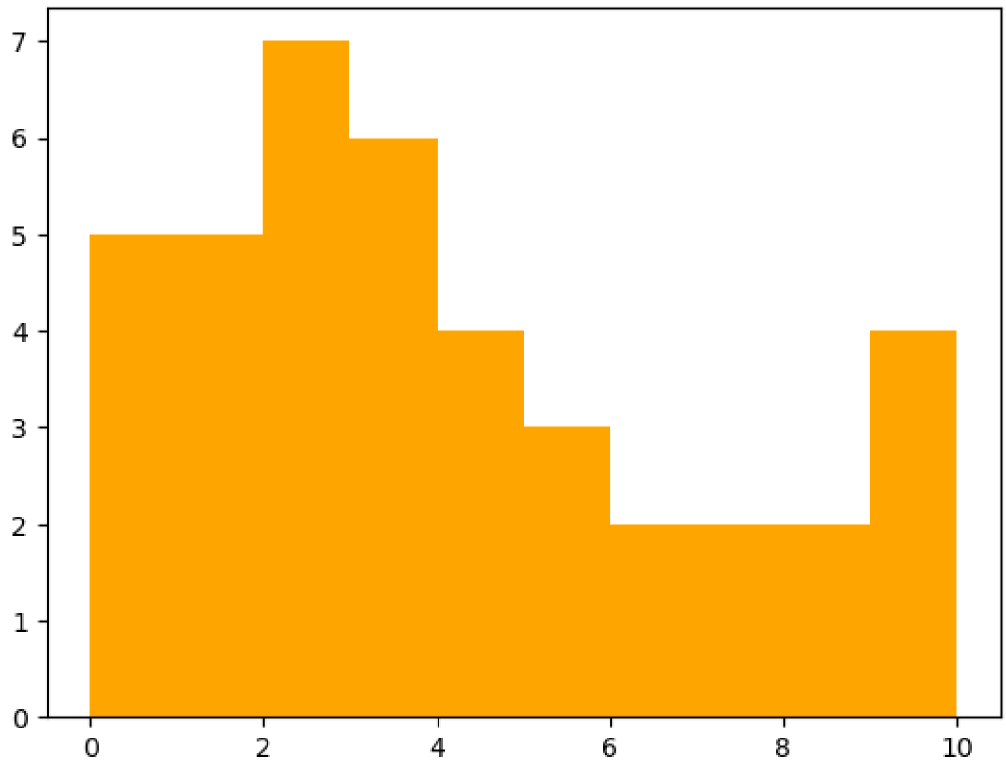
[10]:
```
plt.scatter(x,y)
plt.title("Scatter plot")
plt.xlabel("X axis")
plt.ylabel("Y axis")
plt.show()
```



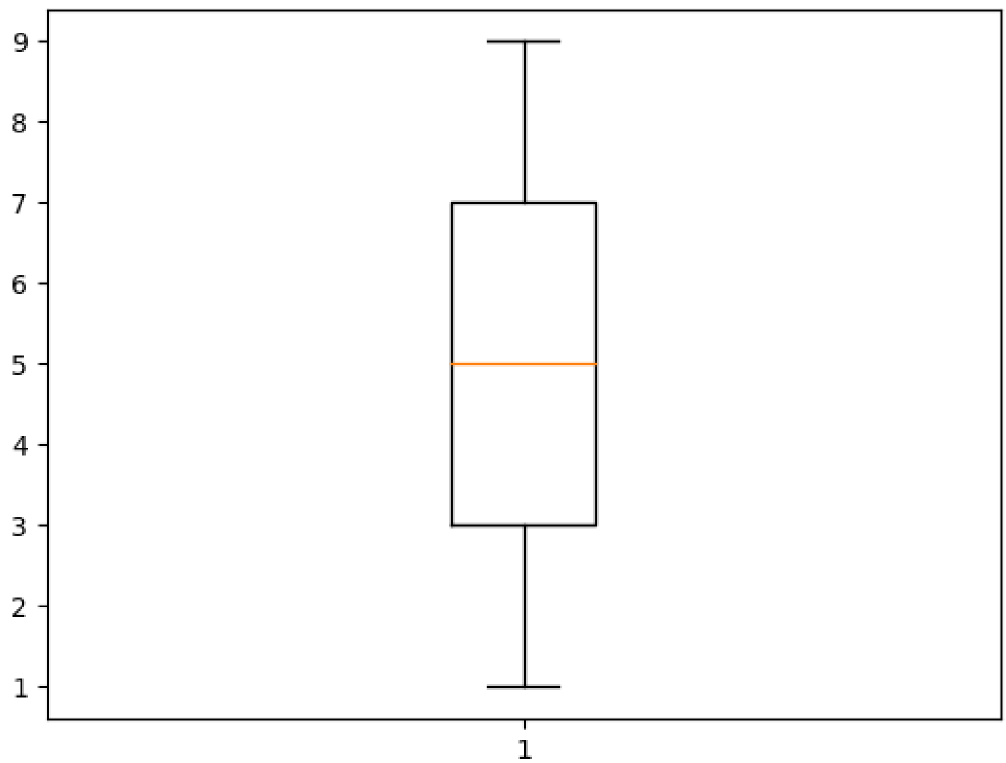
[11]:
```
X=(1,9,2,8,3,7,4,7,5,6,8)
Y=(9,7,5,3,1,2,4,6,8,0,1)
plt.scatter(X,Y)
plt.title("Scatter plot")
plt.xlabel("X axis")
plt.ylabel("Y axis")
plt.show()
```



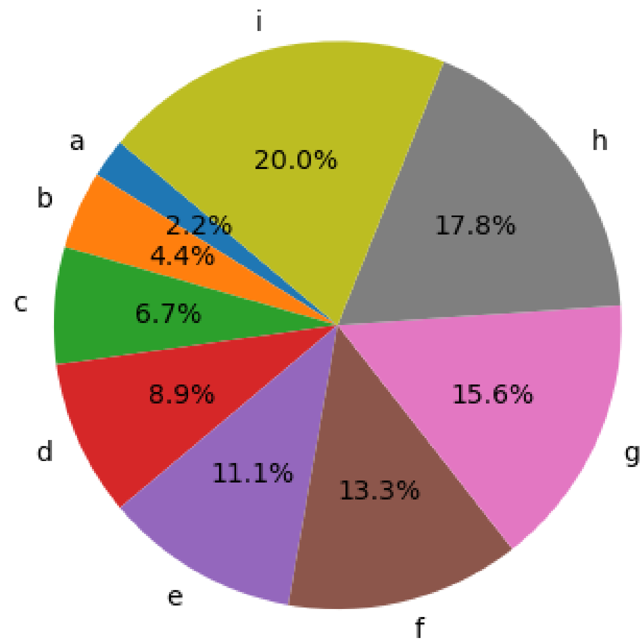
[12]:
```
#Histogram

H=[1,1,1,1,1,2,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,6,6,7,7,8,8,9,9,10,10,2,2,3,3,0,0,0,0,0]

plt.hist(H,color="orange")
plt.show()
```



[13]:
```
# Box plot
n=[1,2,3,4,5,6,7,8,9]

plt.boxplot(n)
plt.show()
```



[14]:
```
#pie chart
n=[1,2,3,4,5,6,7,8,9]
labels=['a','b','c','d','e','f','g','h','i']
plt.pie(n, labels=labels, autopct='%1.1f%%', startangle=140)
plt.show()
```


i
h
a
20.0%
b
17.8%
2.2%
4.4%
c
6.7%
15.6%
8.9%
g
d
11.1%
13.3%
e
f

[ ]:

Practical No. 12

Aim : To study of Hadoop ecosystem in detail
Theory :
Apache Hadoop is an open source framework intended to make interaction with **big data** easier, However, for those who are not acquainted with this technology, one question arises that what is big data ? Big data is a term given to the data sets which can't be processed in an efficient manner with the help of traditional methodology such as RDBMS. Hadoop has made its place in the industries and companies that need to work on large data sets which are sensitive and needs efficient handling. Hadoop is a framework that enables processing of large data sets which reside in the form of clusters. Being a framework, Hadoop is made up of several modules that are supported by a large ecosystem of technologies.

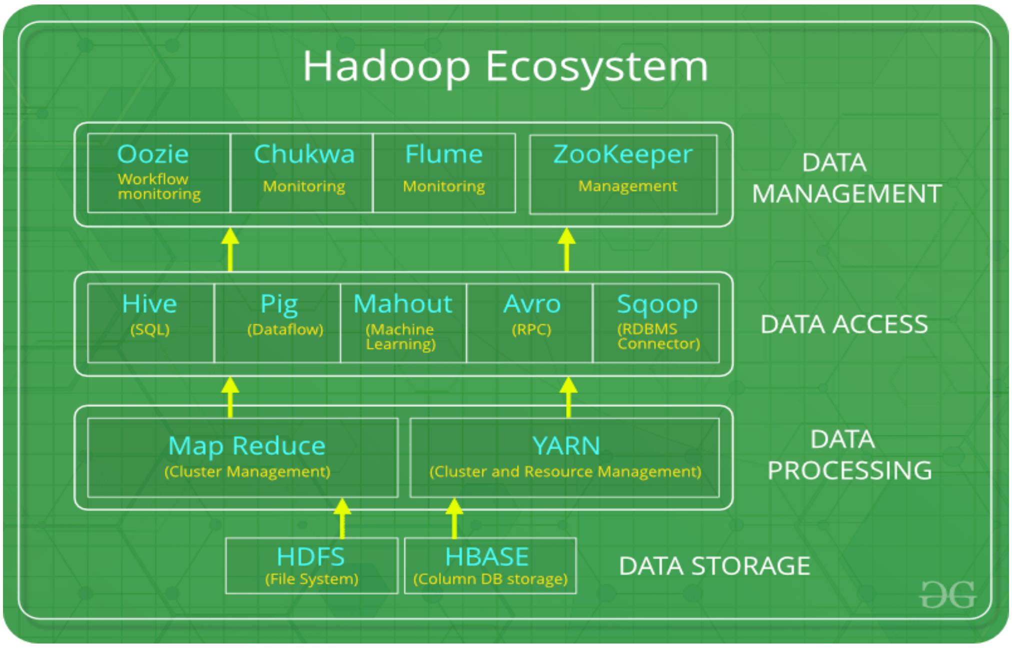
Hadoop Ecosystem is a platform or a suite which provides various services to solve the big data problems. It includes Apache projects and various commercial tools and solutions. There are *four major elements of Hadoop* i.e. **HDFS, MapReduce, YARN, and Hadoop Common Utilities**. Most of the tools or solutions are used to supplement or support these major elements. All these tools work collectively to provide services such as absorption, analysis, storage and maintenance of data etc.

Following are the components that collectively form a Hadoop ecosystem:

Following are the components that collectively form a Hadoop ecosystem:

- **HDFS:** Hadoop Distributed File System
- **YARN:** Yet Another Resource Negotiator
- **MapReduce:** Programming based Data Processing
- **Spark:** In-Memory data processing
- **PIG, HIVE:** Query based processing of data services
- **HBase:** NoSQL Database

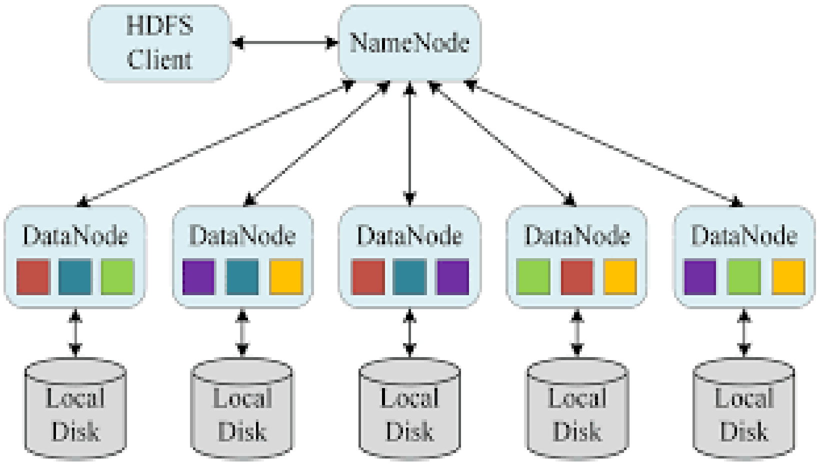
- **Mahout, Spark MLLib:** Machine Learning algorithm libraries
- **Solar, Lucene:** Searching and Indexing
- **Zookeeper:** Managing cluster
- **Oozie:** Job Scheduling



All these toolkits or components revolve around one term i.e. *Data*. That's the beauty of Hadoop that it revolves around data and hence making its synthesis easier.

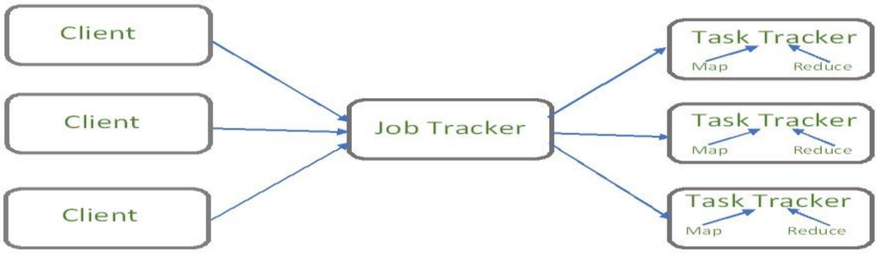
HDFS:

- HDFS is the primary or major component of Hadoop ecosystem and is responsible for storing large data sets of structured or unstructured data across various nodes and thereby maintaining the metadata in the form of log files.
- HDFS consists of two core components i.e.
  1.Name node
  2.Data node
- Name Node is the prime node which contains metadata (data about data) requiring comparatively fewer resources than the data nodes that stores the actual data. These data nodes are commodity hardware in the distributed environment. Undoubtedly, making Hadoop cost effective
- HDFS maintains all the coordination between the clusters and hardware, thus working at the heart of the system.



YARN:

- Yet Another Resource Negotiator, as the name implies, YARN is the one who helps to manage the resources across the clusters. In short, it performs scheduling and resource allocation for the Hadoop System.
- Consists of three major components i.e.
  1. Resource Manager
  2. Nodes Manager
  3. Application Manager
- Resource manager has the privilege of allocating resources for the applications in a system whereas Node managers work on the allocation of resources such as CPU, memory, bandwidth per machine and later on acknowledges the resource manager. Application manager works as an interface between the resource manager and node manager and performs negotiations as per the requirement of the two.



Hadoop 1.0 architecture

MapReduce:

- By making the use of distributed and parallel algorithms, MapReduce makes it possible to carry over the processing's logic and helps to write applications which transform big data sets into a manageable one.

- MapReduce makes the use of two functions i.e. Map() and Reduce() whose task is:

    1. *Map()* performs sorting and filtering of data and thereby organizing them in the form of group. Map generates a key-value pair based result which is later on processed by the Reduce() method.

    2. *Reduce()*, as the name suggests does the summarization by aggregating the mapped data. In simple, Reduce() takes the output generated by Map() as input and combines those tuples into smaller set of tuples.

PIG:

Pig was basically developed by Yahoo which works on a pig Latin language, which is Query based language similar to SQL.

- It is a platform for structuring the data flow, processing and analyzing huge data sets.

- Pig does the work of executing commands and in the background, all the activities of MapReduce are taken care of. After the processing, pig stores the result in HDFS.

- Pig Latin language is specially designed for this framework which runs on Pig Runtime. Just the way Java runs on the JVM.
- Pig helps to achieve ease of programming and optimization and hence is a major segment of the Hadoop Ecosystem.

HIVE:

- With the help of SQL methodology and interface, HIVE performs reading and writing of large data sets. However, its query language is called as HQL (Hive Query Language).
- It is highly scalable as it allows real-time processing and batch processing both. Also, all the SQL datatypes are supported by Hive thus, making the query processing easier.
- Similar to the Query Processing frameworks, HIVE too comes with two components: *JDBC Drivers* and *HIVE Command Line*.
- JDBC, along with ODBC drivers work on establishing the data storage permissions and connection whereas HIVE Command line helps in the processing of queries.

Mahout:

- Mahout, allows Machine Learnability to a system or application. Machine Learning, as the name suggests helps the system to develop itself based on some patterns, user/environmental interaction or on the basis of algorithms.
- It provides various libraries or functionalities such as collaborative filtering, clustering, and classification which are nothing but concepts of Machine learning. It allows invoking algorithms as per our need with the help of its own libraries.

Apache Spark:

- It's a platform that handles all the process consumptive tasks like batch processing, interactive or iterative real-time processing, graph conversions, and visualization, etc.

- It consumes in memory resources hence, thus being faster than the prior in terms of optimization.
- Spark is best suited for real-time data whereas Hadoop is best suited for structured data or batch processing, hence both are used in most of the companies interchangeably.

Apache HBase:

- It's a NoSQL database which supports all kinds of data and thus capable of handling anything of Hadoop Database. It provides capabilities of Google's BigTable, thus able to work on Big Data sets effectively.
- At times where we need to search or retrieve the occurrences of something small in a huge database, the request must be processed within a short quick span of time. At such times, HBase comes handy as it gives us a tolerant way of storing limited data

Other Components: Apart from all of these, there are some other components too that carry out a huge task in order to make Hadoop capable of processing large datasets. They are as follows:

- **Solr, Lucene:** These are the two services that perform the task of searching and indexing with the help of some java libraries, especially Lucene is based on Java which allows spell check mechanism, as well. However, Lucene is driven by Solr.
- **Zookeeper:** There was a huge issue of management of coordination and synchronization among the resources or the components of Hadoop which resulted in inconsistency, often. Zookeeper overcame all the problems by performing synchronization, inter-component based communication, grouping, and maintenance.
- **Oozie:** Oozie simply performs the task of a scheduler, thus scheduling jobs and binding them together as a single unit. There is two kinds of jobs .i.e Oozie workflow and Oozie coordinator jobs. Oozie workflow is the jobs that need to be executed in a sequentially ordered manner whereas Oozie Coordinator jobs are those that are triggered when some data or external stimulus is given to it.

Conclusion : In this way , we learn about the Hadoop ecosystem.