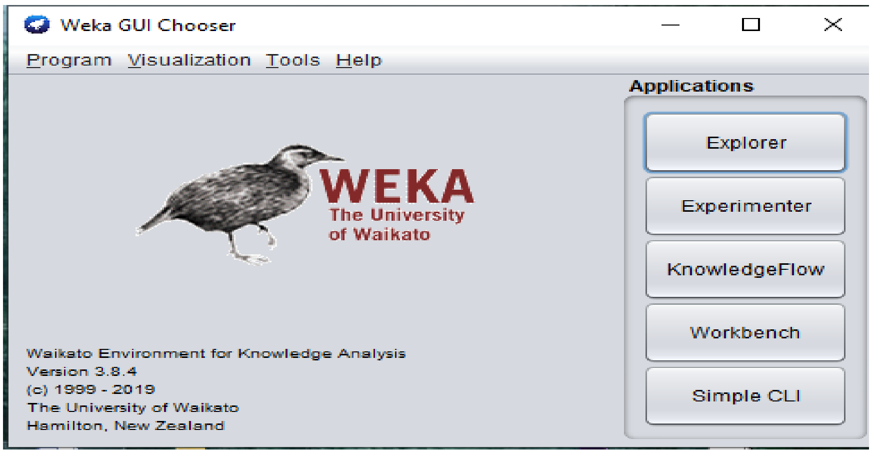
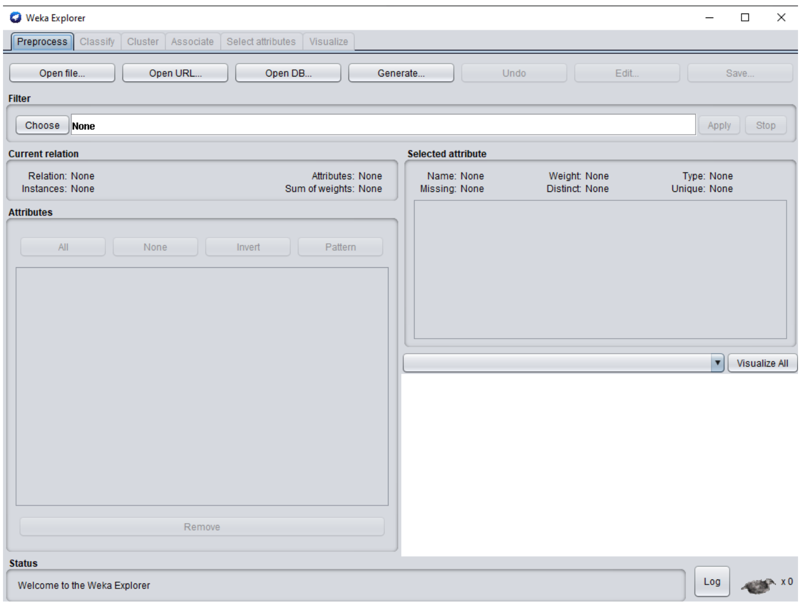
**Week – 2 Explore machine learning tool "WEKA" Study the arff file format Explore the available data sets in WEKA. Load a data set (ex. Weather dataset, Iris dataset, etc.)**
**Load each dataset and observe the following:**
**1. List the attribute names and they types**
**2. Number of records in each dataset**
**3. Identify the class attribute (if any)**
**4. Plot Histogram**
**5. Determine the number of records for each class.**
**6. Visualize the data in various dimensions**

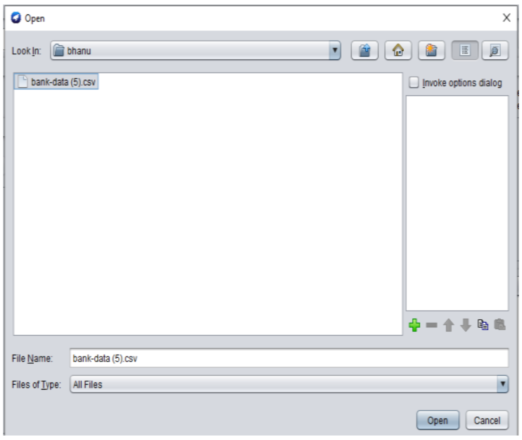
**Introduction to weka:**

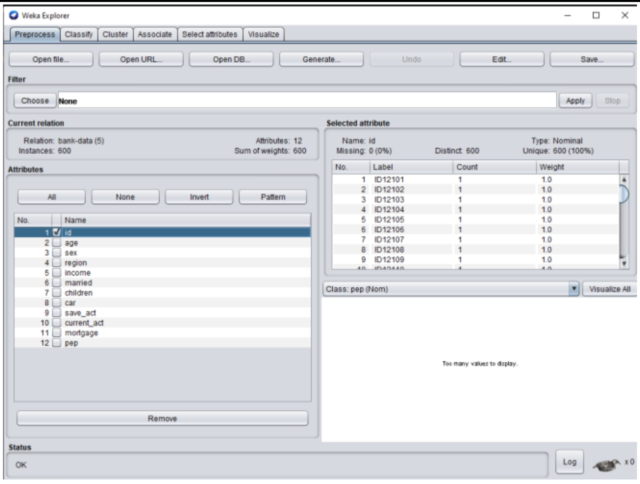
WEKA - an open source software provides tools for data preprocessing, implementation of several Machine Learning algorithms, and visualization tools so that you can develop machine learning techniques and apply them to real-world data mining problems.

features:






**Available data sets in weka:**

**1.airline**
**2.breast cancer**
**3.contact lenses**
**4.cpu**
**5.cpu with vendor**
**6.iris**
**7.weather.nominal**
**8.weather.numeric**
**9.diabetes**
**10.glass**

<u>WEATHER.ARFF:</u>

```
@relation weather
@attribute outlook {sunny, overcast, rainy}
@attribute temperature numeric
@attribute humidity numeric
@attribute windy {TRUE, FALSE}
@attribute play {yes, no}
@data
sunny,85,85,FALSE,no
sunny,80,90,TRUE,no
overcast,83,86,FALSE,yes
rainy,70,96,FALSE,yes
rainy,68,80,FALSE,yes
rainy,65,70,TRUE,no
overcast,64,65,TRUE,yes
sunny,72,95,FALSE,no
sunny,69,70,FALSE,yes
rainy,75,80,FALSE,yes
sunny,75,70,TRUE,yes
overcast,72,90,TRUE,yes
overcast,81,75,FALSE,yes
rainy,71,91,TRUE,no
```

<u>IRIS.ARFF:</u>

```
% 1. Title: Iris Plants Database
%
% 2. Sources:
%      (a) Creator: R.A. Fisher
%      (b) Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
%      (c) Date: July, 1988
```

```
%
% 3. Past Usage:
%    - Publications: too many to mention!!!  Here are a few.
%    1. Fisher,R.A. "The use of multiple measurements in taxonomic problems"
%       Annual Eugenics, 7, Part II, 179-188 (1936); also in "Contributions
%       to Mathematical Statistics" (John Wiley, NY, 1950).
%    2. Duda,R.O., & Hart,P.E. (1973) Pattern Classification and Scene Analysis.
%       (Q327.D83) John Wiley & Sons.  ISBN 0-471-22361-1.  See page 218.
5. Number of Instances: 150 (50 in each of three classes)
%
% 6. Number of Attributes: 4 numeric, predictive attributes and the class
%
% 7. Attribute Information:
%    1. sepal length in cm
%    2. sepal width in cm
%    3. petal length in cm
%    4. petal width in cm
%    5. class:
%       -- Iris Setosa
%       -- Iris Versicolour
%       -- Iris Virginica
%
% 8. Missing Attribute Values: None
%
% Summary Statistics:
%                 Min  Max   Mean    SD   Class Correlation
%    sepal length: 4.3  7.9   5.84  0.83    0.7826
%     sepal width: 2.0  4.4   3.05  0.43   -0.4194
%    petal length: 1.0  6.9   3.76  1.76    0.9490  (high!)
%     petal width: 0.1  2.5   1.20  0.76    0.9565  (high!)
%
% 9. Class Distribution: 33.3% for each of 3 classes.
@RELATION iris
@ATTRIBUTE sepallength      REAL
@ATTRIBUTE sepalwidth       REAL
@ATTRIBUTE petallength      REAL
@ATTRIBUTE petalwidth       REAL
@ATTRIBUTE class    {Iris-setosa,Iris-versicolor,Iris-virginica}
@DATA
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
7.0,3.2,4.7,1.4,Iris-versicolor
6.4,3.2,4.5,1.5,Iris-versicolor
6.9,3.1,4.9,1.5,Iris-versicolor
5.5,2.3,4.0,1.3,Iris-versicolor
6.3,3.3,6.0,2.5,Iris-virginica
5.8,2.7,5.1,1.9,Iris-virginica
7.1,3.0,5.9,2.1,Iris-virginica
6.3,2.9,5.6,1.8,Iris-virginica
6.5,3.0,5.8,2.2,Iris-virginica
```

**AIRLINE.ARFF:**

```
%% Monthly totals of international airline passengers (in thousands) for
%% 1949-1960.

@relation airline_passengers
@attribute passenger_numbers numeric
@attribute Date date 'yyyy-MM-dd'

@data
112,1949-01-01
118,1949-02-01
132,1949-03-01
129,1949-04-01
121,1949-05-01
135,1949-06-01
148,1949-07-01
148,1949-08-01
136,1949-09-01
119,1949-10-01
104,1949-11-01
118,1949-12-01
115,1950-01-01
126,1950-02-01
141,1950-03-01
135,1950-04-01
125,1950-05-01
149,1950-06-01
170,1950-07-01
170,1950-08-01
158,1950-09-01
133,1950-10-01
```

**CPU.ARFF:**

```
%
% As used by Kilpatrick, D. & Cameron-Jones, M. (1998). Numeric prediction
% using instance-based learning with encoding length selection. In Progress
% in Connectionist-Based Information Systems. Singapore: Springer-Verlag.
%
% Deleted "vendor" attribute to make data consistent with with what we
% used in the data mining book.
%
@relation 'cpu'
@attribute MYCT numeric
@attribute MMIN numeric
@attribute MMAX numeric
@attribute CACH numeric
@attribute CHMIN numeric
@attribute CHMAX numeric
@attribute class numeric
@data
125,256,6000,256,16,128,198
29,8000,32000,32,8,32,269
29,8000,32000,32,8,32,220
29,8000,32000,32,8,32,172
```

29,8000,16000,32,8,16,132
26,8000,32000,64,8,32,318
23,16000,32000,64,16,32,367
23,16000,32000,64,16,32,489
23,16000,64000,64,16,32,636
23,32000,64000,128,32,64,1144
400,1000,3000,0,1,2,38

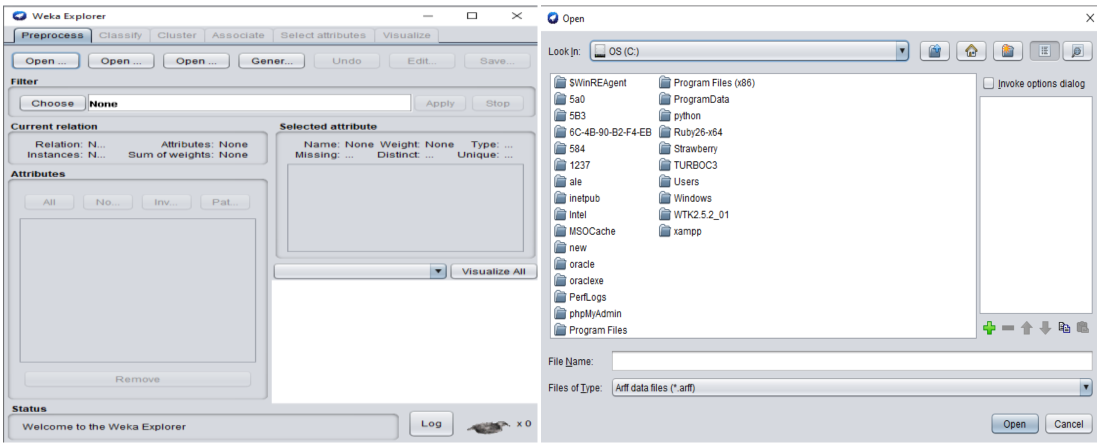
**CONTACT-LENSES.ARFF**:

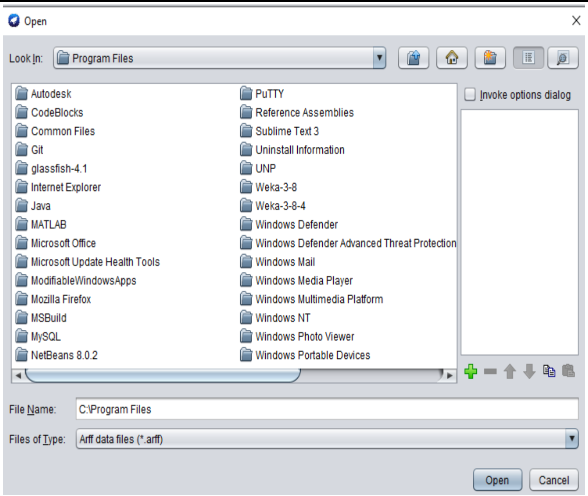
```
@relation contact-lenses

@attribute age                  {young, pre-presbyopic, presbyopic}
@attribute spectacle-prescrip   {myope, hypermetrope}
@attribute astigmatism          {no, yes}
@attribute tear-prod-rate       {reduced, normal}
@attribute contact-lenses       {soft, hard, none}

@data
%
% 24 instances
%
young,myope,no,reduced,none
young,myope,no,normal,soft
young,myope,yes,reduced,none
young,myope,yes,normal,hard
young,hypermetrope,no,reduced,none
young,hypermetrope,no,normal,soft
```

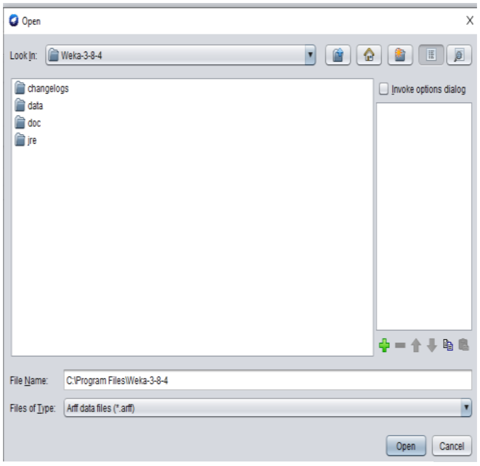
**LOAD DATA SETS IN WEKA**
**DESCRIPTION:**

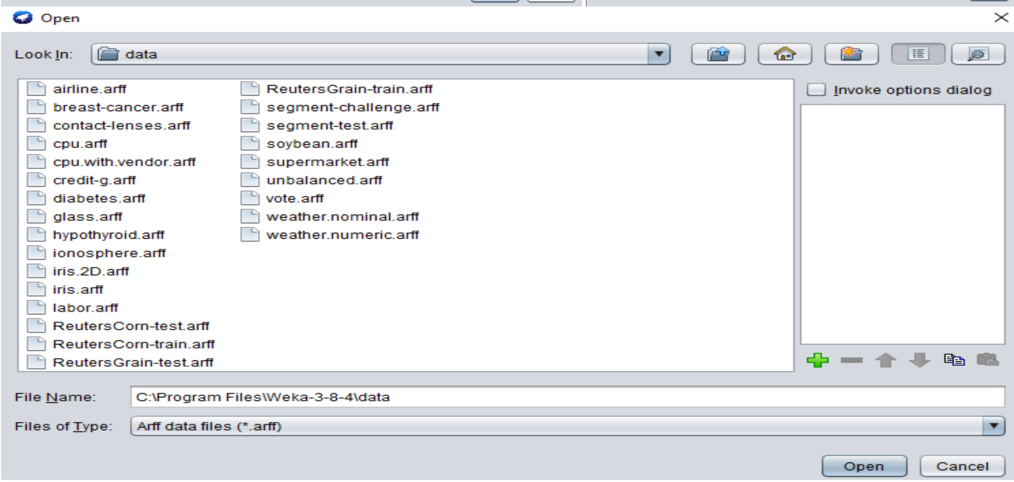
**step1**: open weka
**step2**:Go to file explorer
**step3**:select open file under preprocess
**step4**:select the folder where the arff file is located
**step5**:Open the file
**step6**:observe attributes names,types,class attribute

1.WEATHER.ARFF:

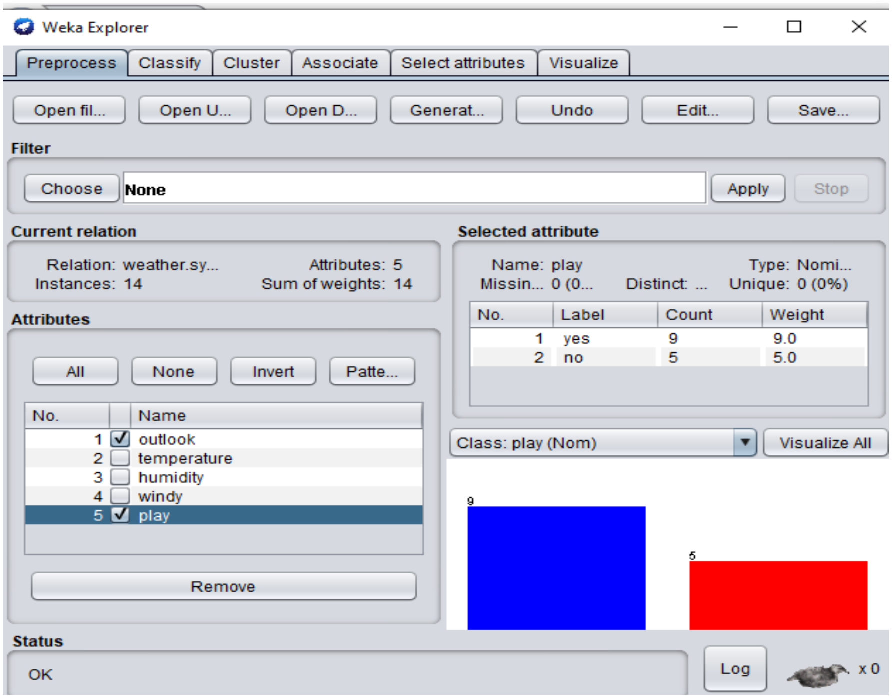





UPLOADING WEATHER.ARFF FILE:

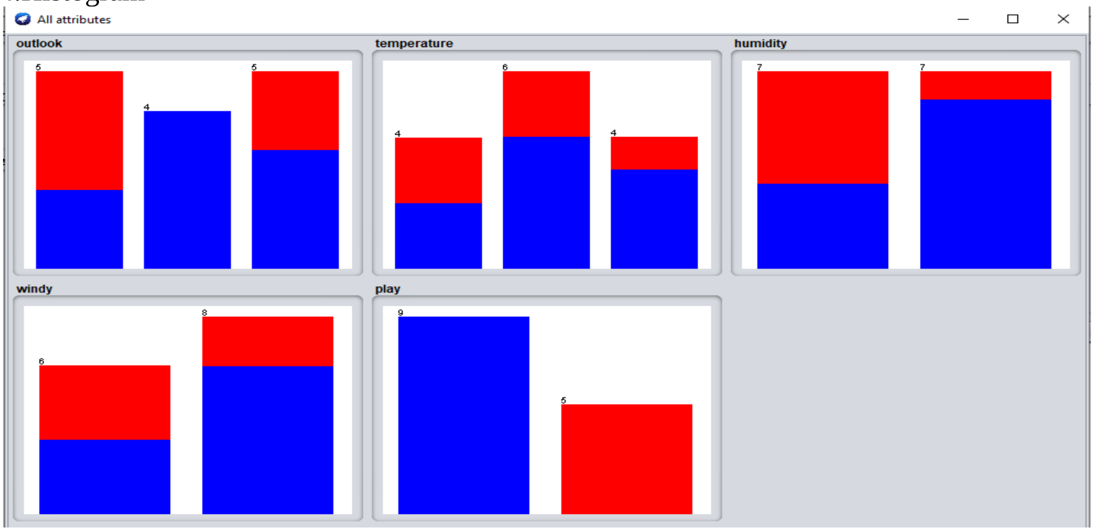



1. List the attribute names and they types

Attributes are outlook,temperature,humidity,windy,play

2. Number of records in each dataset
   number of records are 14
3. Identify the class attribute (if any)
   class attribute is play

4.Histogram

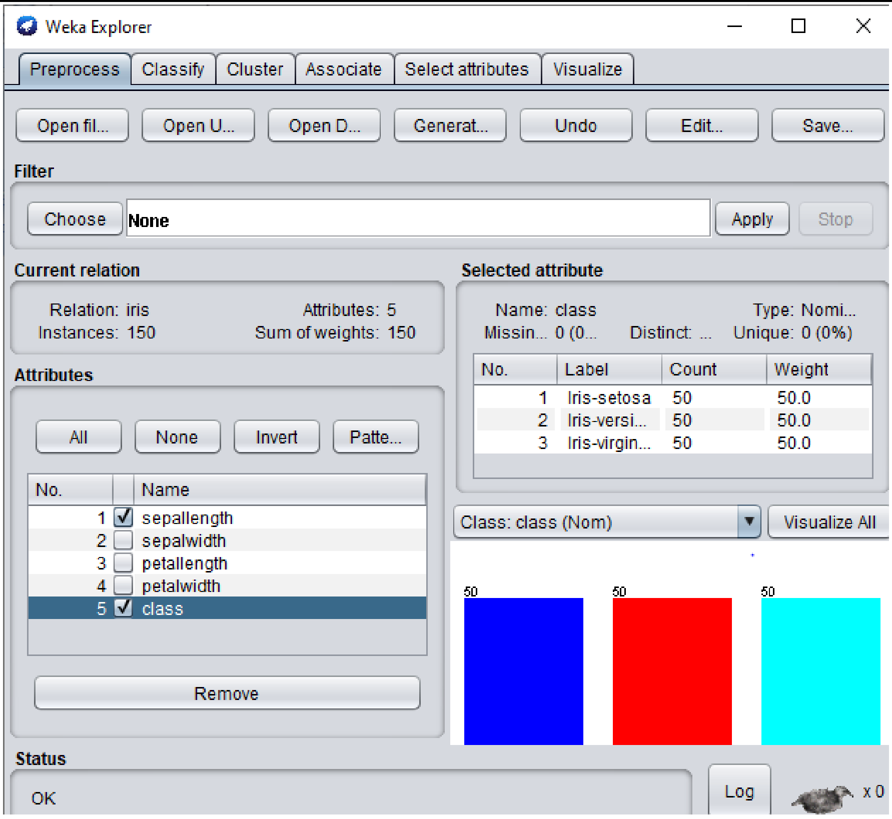



python program:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
data=pd.read_csv("/Iris.csv")
print(data.head(10))
data.info()
plt.figure(figsize
```

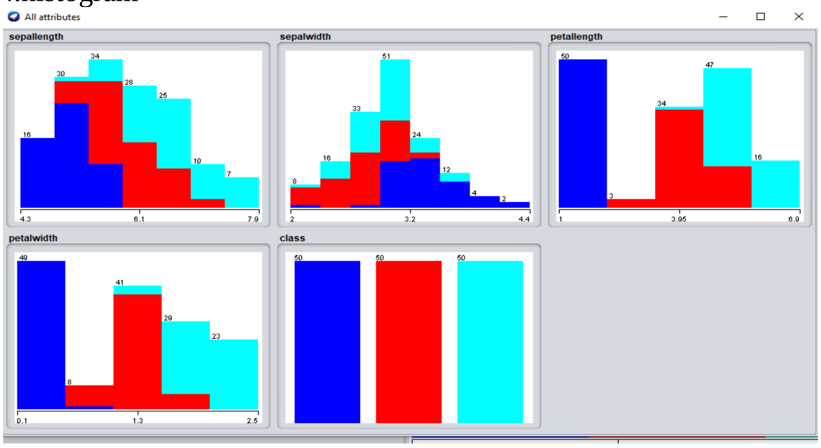
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
data=pd.read_csv("/Iris.csv")
print(data.head(10))

data.info()
plt.figure(figsize=(10,7))
x=data["SepalLengthCm"]
plt.hist(x,bins=20,color="green")
plt.title("Sepal length in cm")
plt.xlabel("Sepal_Lengh_cm")
plt.ylabel("Count")
```

IRIS.ARFF:

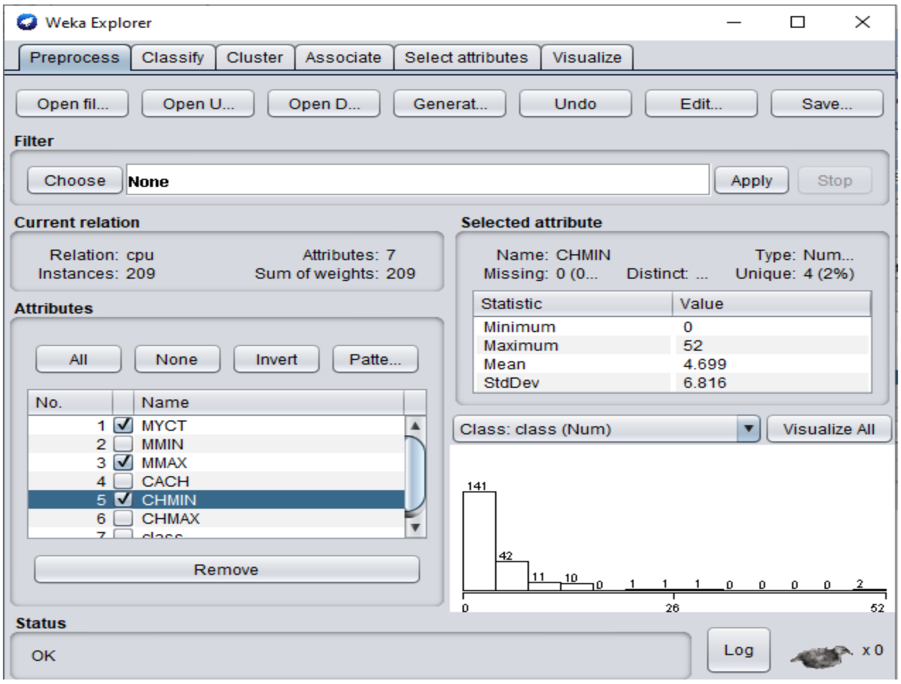



1. List the attribute names and they types
   attribute names and they types :
   sepallength REAL
   sepalwidth REAL
   petallength REAL
   petalwidth REAL
   class {Iris-setosa,Iris-versicolor,Iris-virginica}

2. Number of records in each dataset
   number of records=150
   3. Identify the class attribute (if any)
   class {Iris-setosa,Iris-versicolor,Iris-virginica}
   4.histogram



CPU.ARFF:



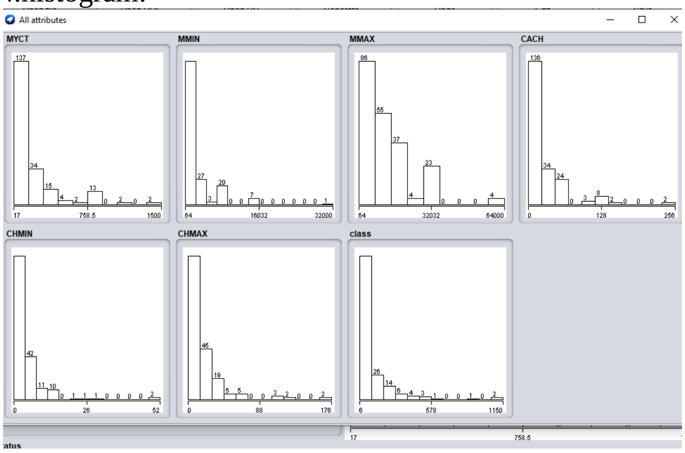

1. List the attribute names and they types
   attribute names and they types:

attribute MYCT numeric
MMIN numeric
MMAX numeric
@attribute CACH numeric
@attribute CHMIN numeric
@attribute CHMAX numeric
@attribute class numeric

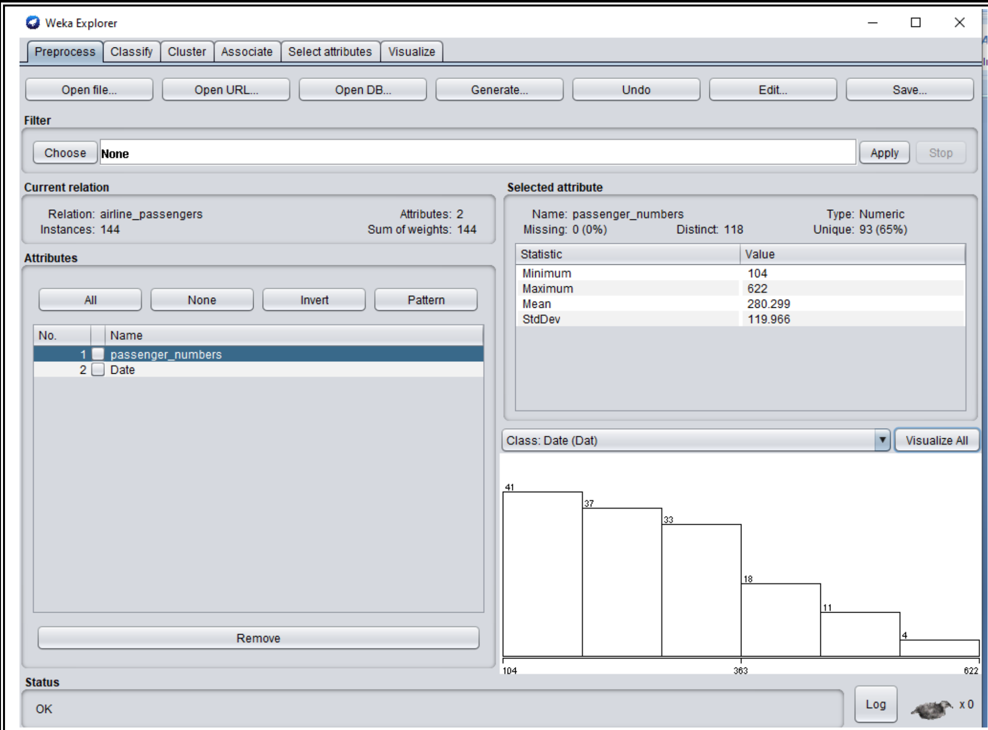
2. Number of records in each dataset:11
3. Identify the class attribute (if any)

class numeric

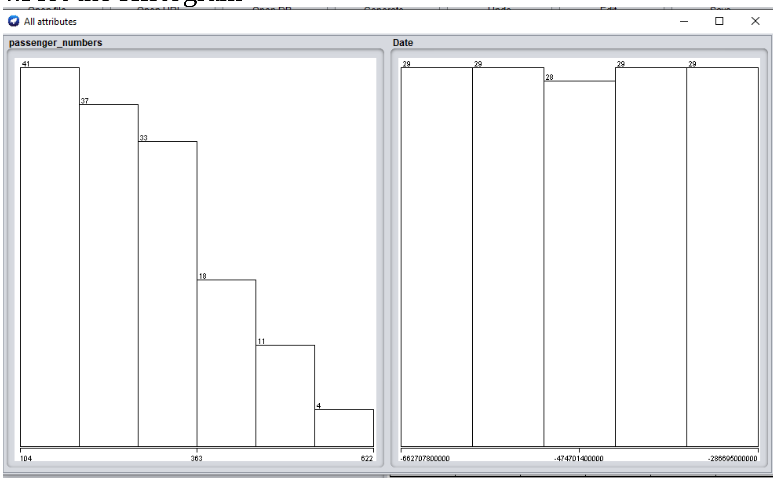
4.histogram:



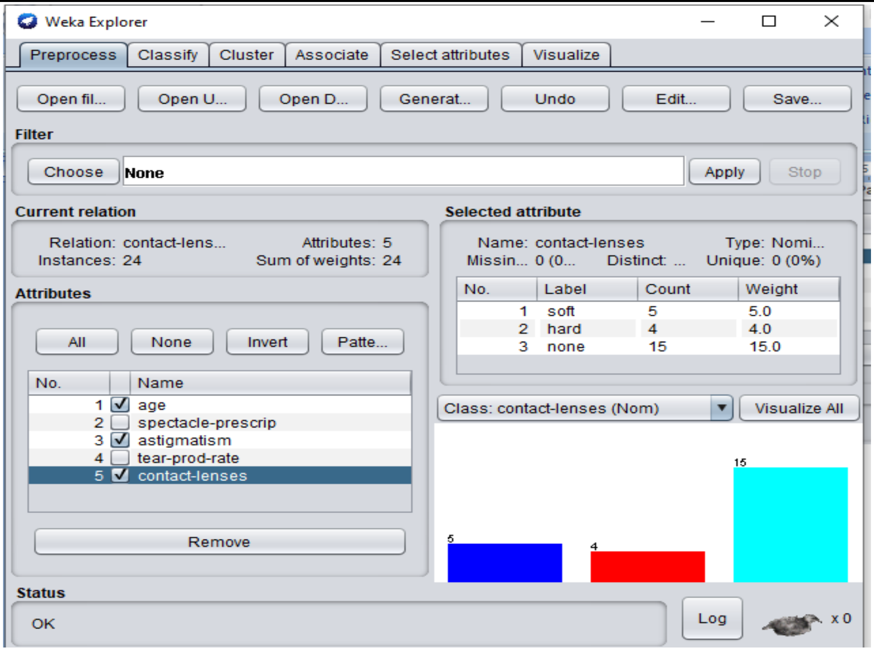

AIRLINE.ARFF File:



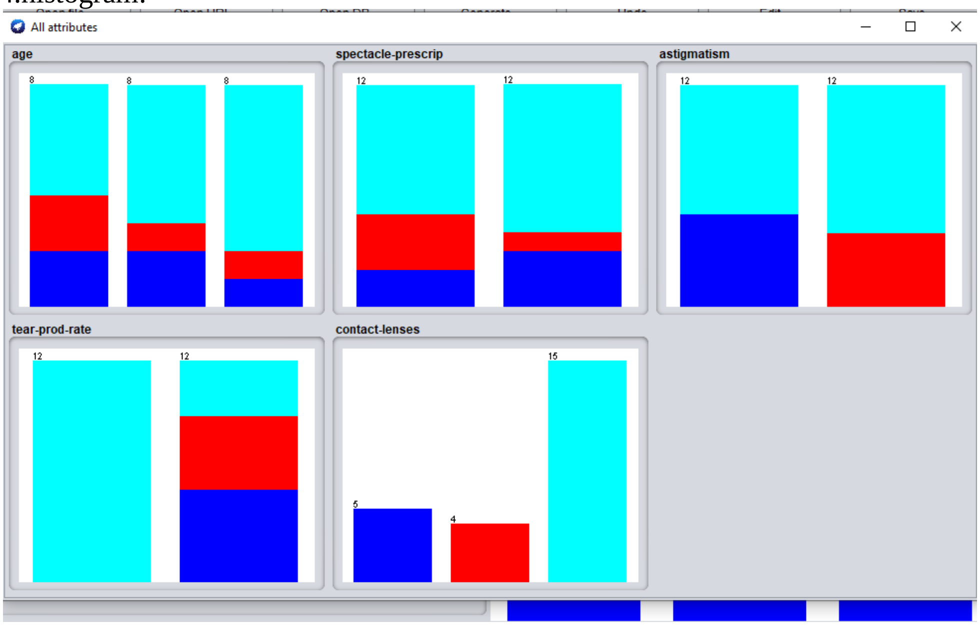

1.List of attributes and types
passengers number
date
2.Number of data recors:22
3/No class attribute
4.Plot the Histogram




**contact-lenses.arff:**




4.histogram:




program:

**numpy:**

NumPy is a Python library used for working with arrays.

It also has functions for working in domain of linear algebra, fourier transform, and matrices.

NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely.

NumPy stands for Numerical Python.

**pandas:**

pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labeled” data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real-world data analysis in Python

**matplotlib:**

Matplotlib is a cross-platform, data visualization and graphical plotting library for Python and its numerical extension NumPy. As such, it offers a viable open source alternative to MATLAB. Developers can also use matplotlib's APIs (Application Programming Interfaces) to embed plots in GUI applications

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

```
[ ] data=pd.read_csv("/content/Iris.csv")
```

```
[ ] print(data.head(10))

   Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm      Species
0   1            5.1           3.5            1.4           0.2  Iris-setosa
1   2            4.9           3.0            1.4           0.2  Iris-setosa
2   3            4.7           3.2            1.3           0.2  Iris-setosa
3   4            4.6           3.1            1.5           0.2  Iris-setosa
4   5            5.0           3.6            1.4           0.2  Iris-setosa
5   6            5.4           3.9            1.7           0.4  Iris-setosa
6   7            4.6           3.4            1.4           0.3  Iris-setosa
7   8            5.0           3.4            1.5           0.2  Iris-setosa
8   9            4.4           2.9            1.4           0.2  Iris-setosa
9  10            4.9           3.1            1.5           0.1  Iris-setosa
```

```
[ ] data.describe()
```

```
[17] data.describe()
```

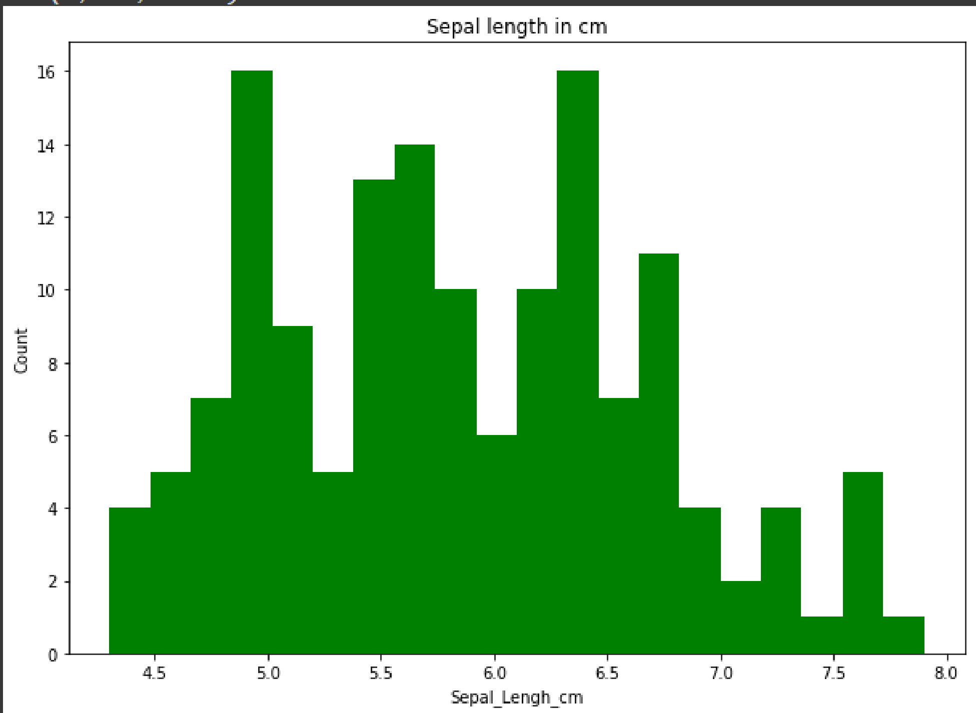
| | Id | SepalLengthCm | SepalWidthCm | PetalLengthCm | PetalWidthCm |
|---|---|---|---|---|---|
| count | 150.000000 | 150.000000 | 150.000000 | 150.000000 | 150.000000 |
| mean | 75.500000 | 5.843333 | 3.054000 | 3.758667 | 1.198667 |
| std | 43.445368 | 0.828066 | 0.433594 | 1.764420 | 0.763161 |
| min | 1.000000 | 4.300000 | 2.000000 | 1.000000 | 0.100000 |
| 25% | 38.250000 | 5.100000 | 2.800000 | 1.600000 | 0.300000 |
| 50% | 75.500000 | 5.800000 | 3.000000 | 4.350000 | 1.300000 |
| 75% | 112.750000 | 6.400000 | 3.300000 | 5.100000 | 1.800000 |
| max | 150.000000 | 7.900000 | 4.400000 | 6.900000 | 2.500000 |

```
[18] data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
```

```
 0   Id             150 non-null    int64
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

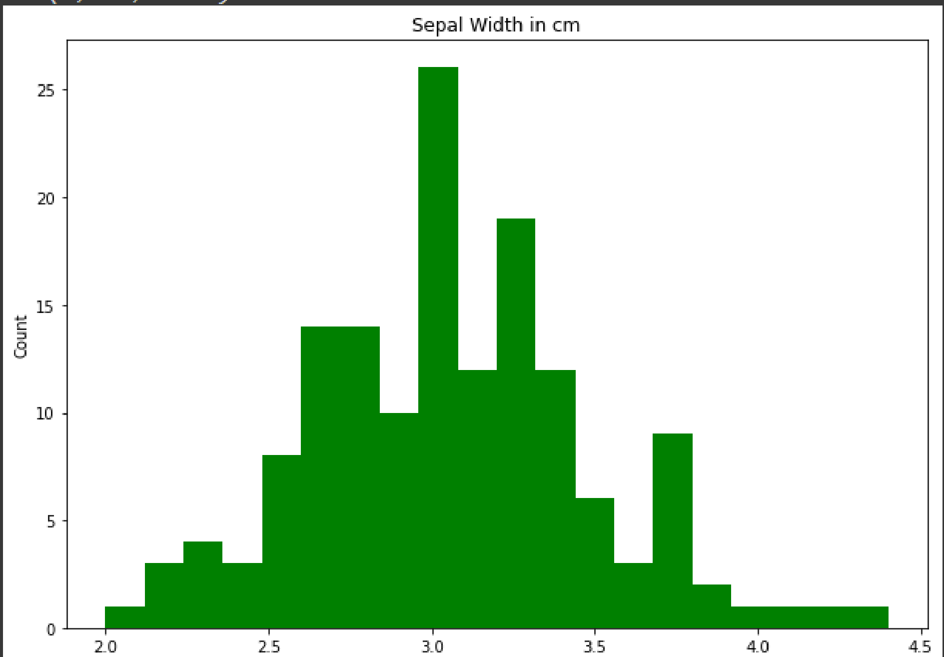
```
[19] plt.figure(figsize=(10,7))
     x=data["SepalLengthCm"]
     plt.hist(x,bins=20,color="green")
     plt.title("Sepal length in cm")
     plt.xlabel("Sepal_Lengh_cm")
     plt.ylabel("Count")
```

```
Text(0, 0.5, 'Count')
```



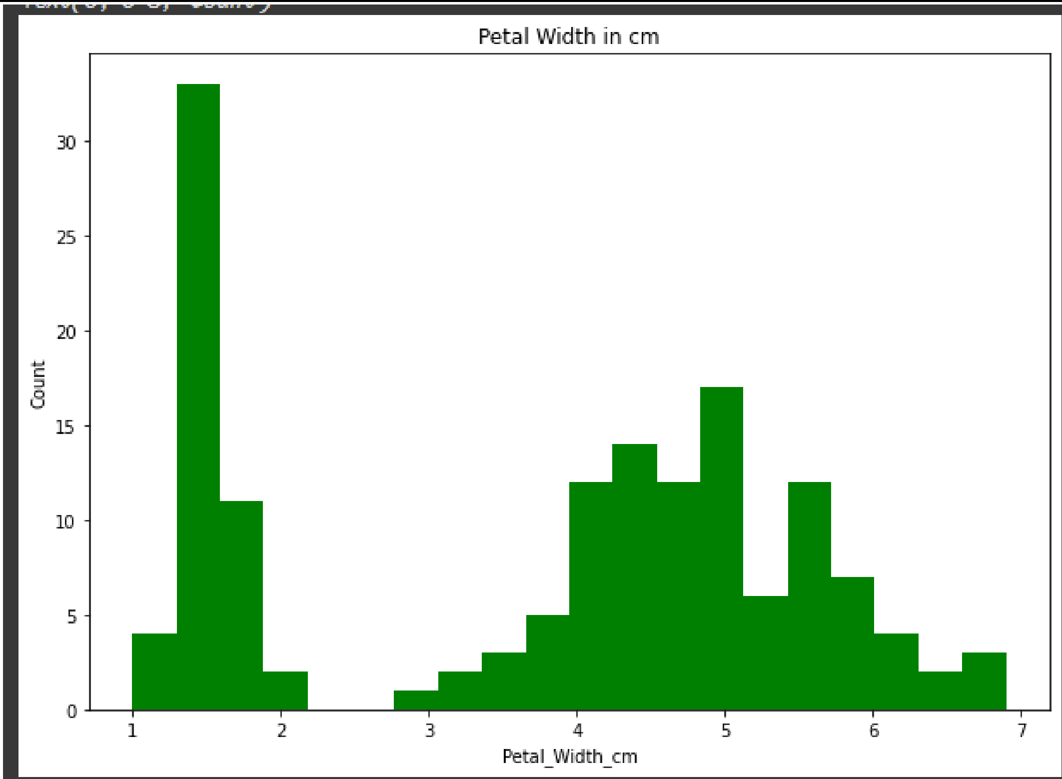
```
[20] plt.figure(figsize=(10,7))
     x=data["SepalWidthCm"]
     plt.hist(x,bins=20,color="green")
     plt.title("Sepal Width in cm")
     plt.xlabel("Sepal_Width_cm")
     plt.ylabel("Count")
```

Text(0, 0.5, 'Count')



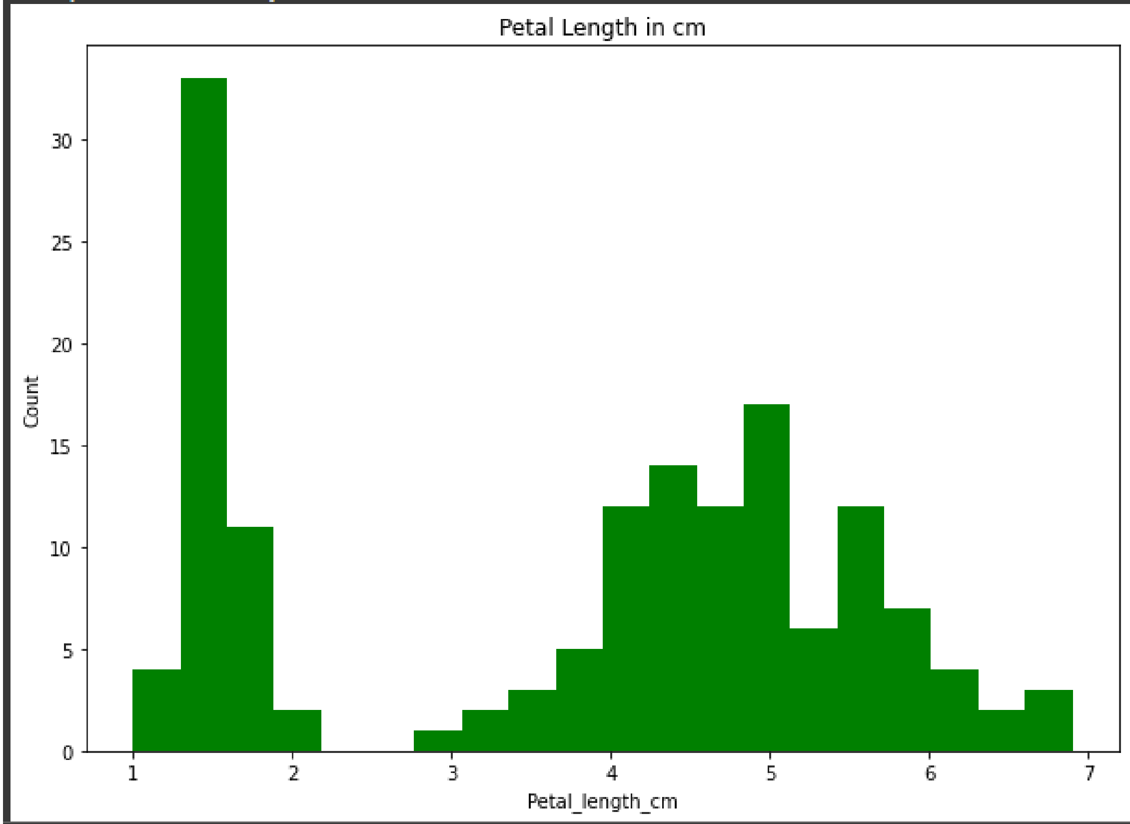

✓ 0s completed at 4:15 PM

```
plt.figure(figsize=(10,7))
x=data["PetalWidthCm"]
plt.hist(x,bins=20,color="green")
plt.title("Petal Width in cm")
plt.xlabel("Petal_Width_cm")
plt.ylabel("Count")
```

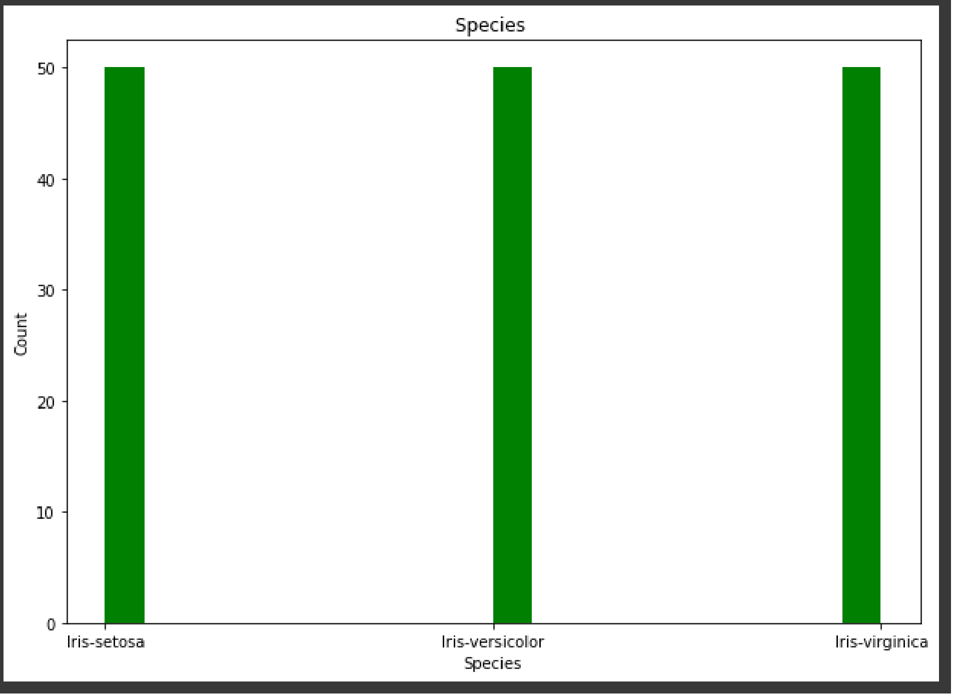



```
plt.figure(figsize=(10,7))
x=data["PetalLengthCm"]
plt.hist(x,bins=20,color="green")
plt.title("Petal Length in cm")
plt.xlabel("Petal_length_cm")
plt.ylabel("Count")
```

Text(0, 0.5, 'Count')




```
plt.figure(figsize=(10,7))
x=data.Species
plt.hist(x,bins=20,color="green")
plt.title("Species ")
plt.xlabel("Species")
plt.ylabel("Count")
plt.show()
```



**5) Determine the number of records for each class.**

**Iris-data set**

Iris-setosa 50 records are there

iris-versicolor 50 records are there

Iris virginica 50 records are there

**Weather.nominal set**

9 records are yes and 5 records are no

total 14 records

**Diabetes:**

500 tested_negative diabetes

268 tested_positive diabetes

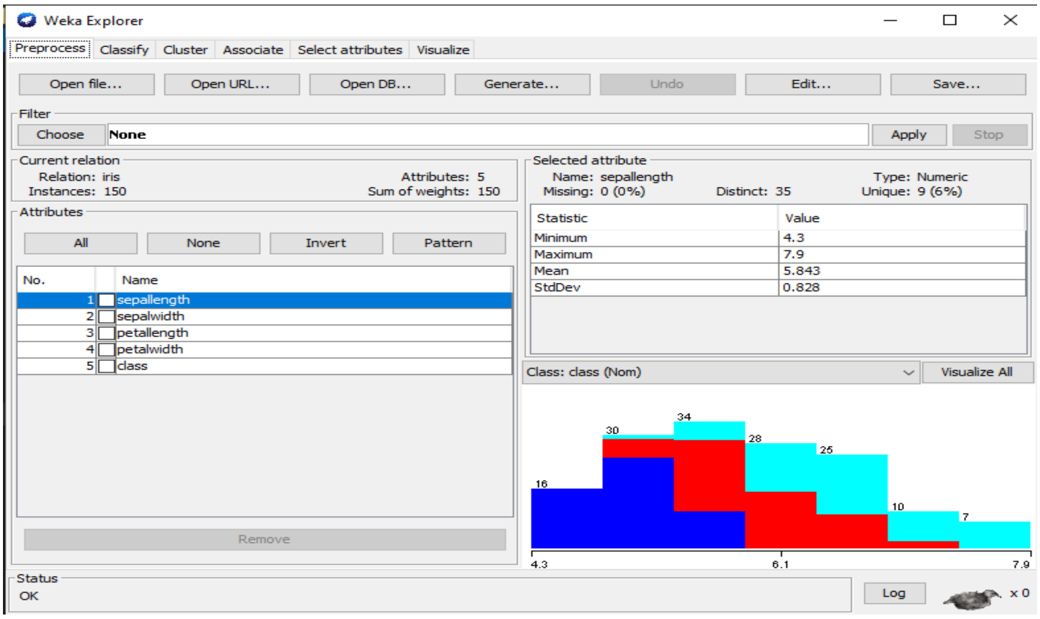
**Breast Cancer**

201 records –no recurrence events

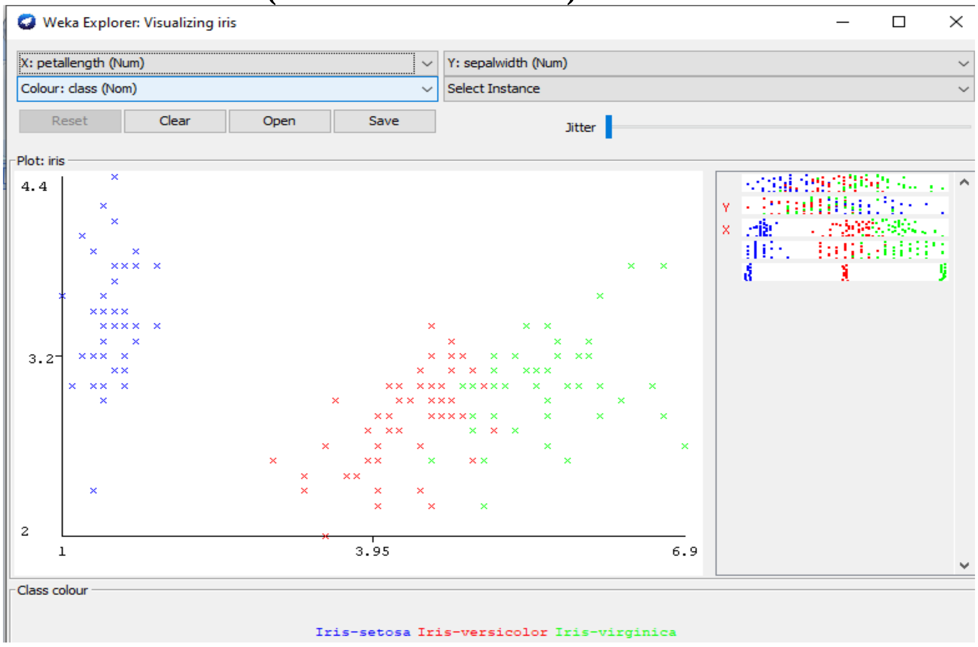
85 records recurrence events

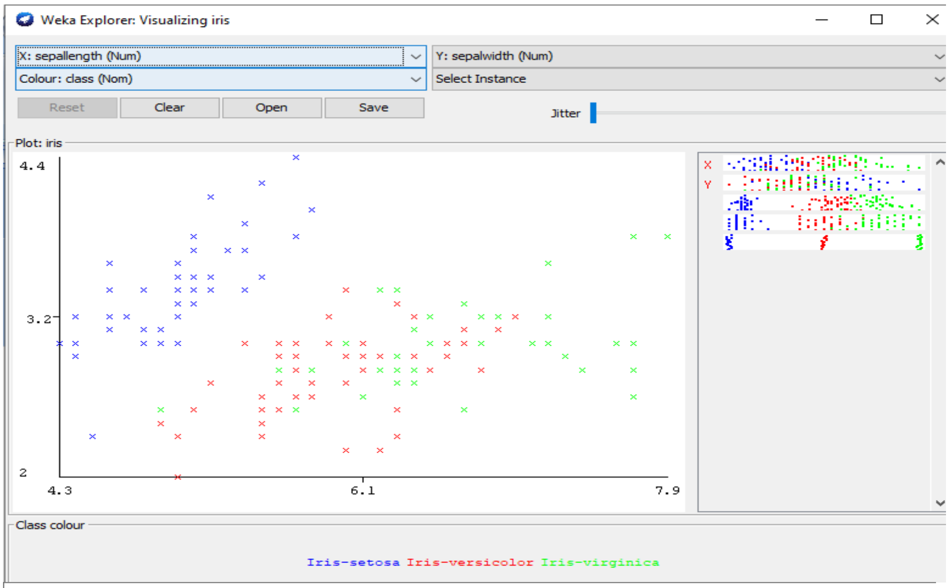
**6)Visualize the data in various dimensions**

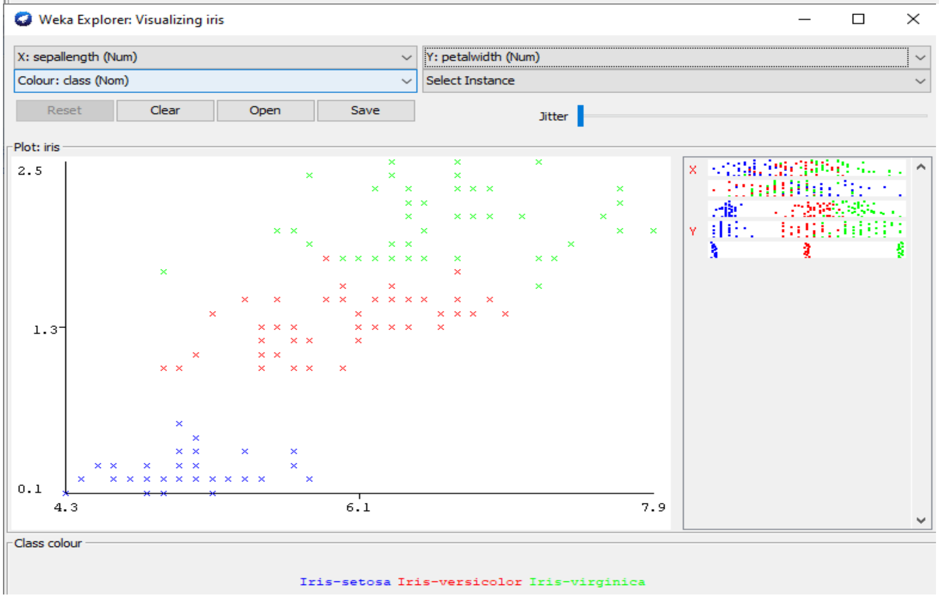
**Load iris data set into weka**




**click on visualize tab (next to select attributes)**




Weka Explorer: Visualizing iris
X: sepallength (Num)
Y: sepalwidth (Num)
Colour: class (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: iris
4.4
3.2
2
4.3
6.1
7.9
Class colour
Iris-setosa Iris-versicolor Iris-virginica


Weka Explorer: Visualizing iris
X: sepallength (Num)
Y: petalwidth (Num)
Colour: class (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: iris
2.5
1.3
0.1
4.3
6.1
7.9
Class colour
Iris-setosa Iris-versicolor Iris-virginica


Weka Explorer: Visualizing iris
X: petallength (Num)
Y: sepallength (Num)
Colour: class (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: iris
7.9
6.1
4.3
1
3.95
6.9
Class colour
Iris-setosa Iris-versicolor Iris-virginica


Weka Explorer: Visualizing iris
X: petallength (Num)
Y: petalwidth (Num)
Colour: class (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: iris
2.5
1.3
0.1
1
3.95
6.9
Class colour
Iris-setosa Iris-versicolor Iris-virginica

**Week – 3 Perform following data preprocessing tasks using Python**
**Rescale Data**
**Binarize Data**
**Standardize Data**

**AIM:**To Perform following data preprocessing tasks using Python i) Rescale Data ii) Binarize Data iii)Standardize Data

**Normalization:**

` Normalization is used to scale the data of an attribute so that it falls in a smaller range, such as -1.0 to 1.0 or 0.0 to 1.0. It is generally useful for classification algorithms.

Min-Max Normalization :

In this technique of knowledge normalization, a linear transformation is performed on the first data. Minimum and maximum value from data is fetched and each value is replaced according to the following formula. Min-Max Normalization preserves the relationships among the original data values. It will encounter an out-of-bounds error if a future input case for normalization falls outside the first data range for A. The formula is given below

$$V' = V - min(A)|max(A) - min(A)(new_max(A) - new_min(A)) + new_min(A)$$

Where A is the attribute data represent as follows.
Min(A) - It is the minimum absolute value A.
Max(A) - It is maximum absolute value of A.
v' - It is the new value of each attribute data.
v - It is the old value of each attribute data.
new_max(A), new_min(A) is the max and min value within the range
(i.e boundary value of range required) respectively.

Example :
Here, we will discuss an example as follows.
Normalize the following group of data –
1000,2000,3000,9000
using min-max normalization by setting min:0 and max:1
Solution –
here,new_max(A)=1 , as given in question- max=1
new_min(A)=0,as given in question- min=0
max(A)=9000,as the maximum data among 1000,2000,3000,9000 is 9000
min(A)=1000,as the minimum data among 1000,2000,3000,9000 is 1000

Case-1: normalizing 1000 –
v = 1000 , putting all values in the formula,we get

```
v' =  (1000-1000) X (1-0)
     -----------------   + 0  =0
      9000-1000
```

Case-2: normalizing 2000 –
v = 2000, putting all values in the formula,we get

```
v  '= (2000-1000) X (1-0)
         -----------------       + 0 =0 .125
           9000-1000
```

Case-3: normalizing 3000 –
v=3000, putting all values in the formula,we get

```
v'=(3000-1000) X (1-0)
     -----------------       + 0 =0 .25
       9000-1000
```

Case-4: normalizing 9000 –
v=9000, putting all values in the formula, we get

v'=(9000-1000) X (1-0)
 -----------------        + 0 =1
 9000-1000
Outcome :
Hence, the normalized values of 1000,2000,3000,9000 are 0, 0.125, .25, 1.

**PROGRAM:**

```
from numpy import asarray
from sklearn.preprocessing import MinMaxScaler
#define data
data=asarray([[100,0.001],
        [8,0.05],
        [50,0.005],
        [88,0.07],
        [4,0.1]])
print(data)
#define min max scaler
scaler=MinMaxScaler()
#transform data
scaled=scaler.fit_transform(data)
print(scaled)
```

**OUTPUT:**

```
[[1.0e+02 1.0e-03]

 [8.0e+00 5.0e-02]
[5.0e+01 5.0e-03]
 [8.8e+01 7.0e-02]
 [4.0e+00 1.0e-01]]
[[1.         0.        ]
 [0.04166667 0.49494949]
 [0.47916667 0.04040404]
[0.875      0.6969697 ]
 [0.         1.        ]]
```

```
from numpy import asarray
from sklearn.preprocessing import MinMaxScaler
#define data
data=asarray([[200,0.001],
              [800,0.05],
              [500,0.005],
              [570,0.07],
              [400,0.1]])
print(data)
#define min max scaler
scaler=MinMaxScaler()
#transform data
scaled=scaler.fit_transform(data)
print(scaled)
```

```
[[2.0e+02 1.0e-03]
 [8.0e+02 5.0e-02]
 [5.0e+02 5.0e-03]
 [5.7e+02 7.0e-02]
 [4.0e+02 1.0e-01]]
[[0.         0.        ]
 [1.         0.49494949]
 [0.5        0.04040404]
 [0.61666667 0.6969697 ]
 [0.33333333 1.        ]]
```

```
from numpy import asarray
from sklearn.preprocessing import MinMaxScaler
#define data
data=asarray([[1000,0.001],
              [2000,0.05],
              [5000,0.005],
              [9070,0.07],
              [40,0.1]])
print(data)
#define min max scaler
scaler=MinMaxScaler()
#transform data
scaled=scaler.fit_transform(data)
print(scaled)
```

```
[[1.00e+03 1.00e-03]
 [2.00e+03 5.00e-02]
 [5.00e+03 5.00e-03]
 [9.07e+03 7.00e-02]
 [4.00e+01 1.00e-01]]
[[0.10631229 0.        ]
 [0.21705426 0.49494949]
 [0.54928018 0.04040404]
 [1.         0.6969697 ]
 [0.         1.        ]]
```

Features always comes under X

y is a class variable

**PROGRAM:**

```
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler

iris=datasets.load_iris()
X=iris.data
y=iris.target

print(X)
print(y)

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=1,stratify=y)

mmscaler = MinMaxScaler()
X_train_norm=mmscaler.fit_transform(X_train)
X_test_norm = mmscaler. transform(X_test)
print(X_train_norm)
print(X _test_norm)
```

**OUTPUT:**

```
[5.1 3.3 1.7 0.5]
[4.8 3.4 1.9 0.2]
[5.  3.  1.6 0.2]
[5.  3.4 1.6 0.4]
[5.2 3.5 1.5 0.2]
[5.2 3.4 1.4 0.2]
[4.7 3.2 1.6 0.2]
[4.8 3.1 1.6 0.2]
[5.4 3.4 1.5 0.4]
[5.2 4.1 1.5 0.1]
[5.5 4.2 1.4 0.2]
[4.9 3.1 1.5 0.2]
[5.  3.2 1.2 0.2]
[5.5 3.5 1.3 0.2]
[4.9 3.6 1.4 0.1]
[4.4 3.  1.3 0.2]
[5.1 3.4 1.5 0.2]
[5.  3.5 1.3 0.3]
[4.5 2.3 1.3 0.3]
[4.4 3.2 1.3 0.2]
[5.  3.5 1.6 0.6]
[5.1 3.8 1.9 0.4]
[4.8 3.  1.4 0.3]
[5.1 3.8 1.6 0.2]
[4.6 3.2 1.4 0.2]
[5.3 3.7 1.5 0.2]
[5.  3.3 1.4 0.2]
[7.  3.2 4.7 1.4]
[6.4 3.2 4.5 1.5]
[6.9 3.1 4.9 1.5]
[5.5 2.3 4.  1.3]
[6.5 2.8 4.6 1.5]
[5.7 2.8 4.5 1.3]
```

```
[[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]
 [5.4 3.9 1.7 0.4]
 [4.6 3.4 1.4 0.3]
 [5.  3.4 1.5 0.2]
 [4.4 2.9 1.4 0.2]
 [4.9 3.1 1.5 0.1]
 [5.4 3.7 1.5 0.2]
 [4.8 3.4 1.6 0.2]
 [4.8 3.  1.4 0.1]
 [4.3 3.  1.1 0.1]
 [5.8 4.  1.2 0.2]
 [5.7 4.4 1.5 0.4]
 [5.4 3.9 1.3 0.4]
 [5.1 3.5 1.4 0.3]
 [5.7 3.8 1.7 0.3]
 [5.1 3.8 1.5 0.3]
 [5.4 3.4 1.7 0.2]
 [5.1 3.7 1.5 0.4]
 [4.6 3.6 1.  0.2]
 [5.1 3.3 1.7 0.5]
 [4.8 3.4 1.9 0.2]
 [5.  3.  1.6 0.2]
 [5.  3.4 1.6 0.4]
 [5.2 3.5 1.5 0.2]
 [5.2 3.4 1.4 0.2]
 [4.7 3.2 1.6 0.2]
 [4.8 3.1 1.6 0.2]
 [5.4 3.4 1.5 0.4]
 [5.2 4.1 1.5 0.1]
```

```
[5.5 2.4 3.7 1. ]
[5.8 2.7 3.9 1.2]
[6.  2.7 5.1 1.6]
[5.4 3.  4.5 1.5]
[6.  3.4 4.5 1.6]
[6.7 3.1 4.7 1.5]
[6.3 2.3 4.4 1.3]
[5.6 3.  4.1 1.3]
[5.5 2.5 4.  1.3]
[5.5 2.6 4.4 1.2]
[6.1 3.  4.6 1.4]
[5.8 2.6 4.  1.2]
[5.  2.3 3.3 1. ]
[5.6 2.7 4.2 1.3]
[5.7 3.  4.2 1.2]
[5.7 2.9 4.2 1.3]
[6.2 2.9 4.3 1.3]
[5.1 2.5 3.  1.1]
[5.7 2.8 4.1 1.3]
[6.3 3.3 6.  2.5]
[5.8 2.7 5.1 1.9]
[7.1 3.  5.9 2.1]
[6.3 2.9 5.6 1.8]
[6.5 3.  5.8 2.2]
[7.6 3.  6.6 2.1]
[4.9 2.5 4.5 1.7]
[7.3 2.9 6.3 1.8]
[6.7 2.5 5.8 1.8]
[7.2 3.6 6.1 2.5]
[6.5 3.2 5.1 2. ]
[6.4 2.7 5.3 1.9]
[6.8 3.  5.5 2.1]
```

```
[6.3 3.3 4.7 1.6]
[4.9 2.4 3.3 1. ]
[6.6 2.9 4.6 1.3]
[5.2 2.7 3.9 1.4]
[5.  2.  3.5 1. ]
[5.9 3.  4.2 1.5]
[6.  2.2 4.  1. ]
[6.1 2.9 4.7 1.4]
[5.6 2.9 3.6 1.3]
[6.7 3.1 4.4 1.4]
[5.6 3.  4.5 1.5]
[5.8 2.7 4.1 1. ]
[6.2 2.2 4.5 1.5]
[5.6 2.5 3.9 1.1]
[5.9 3.2 4.8 1.8]
[6.1 2.8 4.  1.3]
[6.3 2.5 4.9 1.5]
[6.1 2.8 4.7 1.2]
[6.4 2.9 4.3 1.3]
[6.6 3.  4.4 1.4]
[6.8 2.8 4.8 1.4]
[6.7 3.  5.  1.7]
[6.  2.9 4.5 1.5]
[5.7 2.6 3.5 1. ]
[5.5 2.4 3.8 1.1]
[5.5 2.4 3.7 1. ]
[5.8 2.7 3.9 1.2]
[6.  2.7 5.1 1.6]
[5.4 3.  4.5 1.5]
```

```
[5.6 2.8 4.9 2. ]
[7.7 2.8 6.7 2. ]
[6.3 2.7 4.9 1.8]
[6.7 3.3 5.7 2.1]
[7.2 3.2 6.  1.8]
[6.2 2.8 4.8 1.8]
[6.1 3.  4.9 1.8]
[6.4 2.8 5.6 2.1]
[7.2 3.  5.8 1.6]
[7.4 2.8 6.1 1.9]
[7.9 3.8 6.4 2. ]
[6.4 2.8 5.6 2.2]
[6.3 2.8 5.1 1.5]
[6.1 2.6 5.6 1.4]
[7.7 3.  6.1 2.3]
[6.3 3.4 5.6 2.4]
[6.4 3.1 5.5 1.8]
[6.  3.  4.8 1.8]
[6.9 3.1 5.4 2.1]
[6.7 3.1 5.6 2.4]
[6.9 3.1 5.1 2.3]
[5.8 2.7 5.1 1.9]
[6.8 3.2 5.9 2.3]
[6.7 3.3 5.7 2.5]
[6.7 3.  5.2 2.3]
[6.3 2.5 5.  1.9]
[6.5 3.  5.2 2. ]
[6.2 3.4 5.4 2.3]
[5.9 3.  5.1 1.8]]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2]
```

```
[[0.33333333 1.         0.06779661 0.04166667]
 [0.30555556 0.63636364 0.11864407 0.04166667]
 [0.58333333 0.54545455 0.72881356 0.91666667]
 [0.66666667 0.59090909 0.79661017 0.83333333]
 [0.19444444 0.54545455 0.03389831 0.04166667]
 [0.66666667 0.5        0.77966102 0.95833333]
 [0.91666667 0.45454545 0.94915254 0.83333333]
 [0.41666667 0.90909091 0.03389831 0.04166667]
 [0.80555556 0.45454545 0.81355932 0.625     ]
 [0.63888889 0.40909091 0.61016949 0.5       ]
 [0.19444444 0.13636364 0.38983051 0.375     ]
 [0.25       0.31818182 0.49152542 0.54166667]
 [0.11111111 0.54545455 0.05084746 0.04166667]
 [0.5        0.36363636 0.62711864 0.45833333]
 [0.33333333 0.22727273 0.50847458 0.5       ]
 [0.41666667 0.31818182 0.69491525 0.75      ]
 [0.13888889 0.63636364 0.15254237 0.04166667]
 [0.19444444 0.         0.42372881 0.375     ]
 [0.41666667 0.31818182 0.49152542 0.45833333]
 [0.11111111 0.54545455 0.10169492 0.04166667]
 [0.52777778 0.36363636 0.6440678  0.70833333]
 [0.94444444 0.27272727 1.         0.91666667]
 [0.58333333 0.54545455 0.59322034 0.58333333]
 [0.38888889 0.36363636 0.59322034 0.5       ]
 [0.33333333 0.18181818 0.47457627 0.41666667]
 [0.55555556 0.63636364 0.77966102 0.95833333]
 [0.5        0.40909091 0.62711864 0.54166667]
 [0.5        0.27272727 0.77966102 0.54166667]
 [0.61111111 0.45454545 0.81355932 0.875     ]
 [0.41666667 0.36363636 0.69491525 0.95833333]
 [0.38888889 0.36363636 0.52542373 0.5       ]
 [0.22222222 0.77272727 0.08474576 0.125     ]
 [0.94444444 0.81818182 0.96610169 0.875     ]
```

```
 [0.72222222 0.5        0.69491525 0.91666667]
 [0.36111111 0.45454545 0.59322034 0.58333333]
 [0.58333333 0.40909091 0.55932203 0.5       ]
 [0.61111111 0.45454545 0.71186441 0.79166667]
 [0.77777778 0.45454545 0.83050847 0.83333333]
 [0.13888889 0.45454545 0.06779661 0.        ]
 [0.66666667 0.5        0.57627119 0.54166667]
 [0.36111111 0.36363636 0.66101695 0.79166667]
 [0.36111111 0.31818182 0.54237288 0.5       ]
 [0.55555556 0.59090909 0.62711864 0.625     ]
 [0.22222222 0.22727273 0.33898305 0.41666667]
 [0.66666667 0.59090909 0.79661017 1.        ]
 [0.19444444 0.68181818 0.10169492 0.20833333]
 [0.38888889 0.22727273 0.6779661  0.79166667]
 [0.13888889 0.63636364 0.10169492 0.04166667]
 [0.22222222 0.68181818 0.06779661 0.04166667]
 [0.47222222 0.09090909 0.50847458 0.375     ]
 [0.33333333 0.13636364 0.50847458 0.5       ]
 [0.83333333 0.40909091 0.89830508 0.70833333]
 [0.69444444 0.45454545 0.76271186 0.83333333]
 [0.16666667 0.72727273 0.06779661 0.        ]
 [0.19444444 0.45454545 0.10169492 0.04166667]
 [0.16666667 0.45454545 0.06779661 0.04166667]
 [0.75       0.54545455 0.62711864 0.54166667]
 [0.72222222 0.54545455 0.79661017 0.91666667]
 [0.44444444 0.45454545 0.69491525 0.70833333]
 [0.61111111 0.36363636 0.61016949 0.58333333]
 [0.19444444 0.63636364 0.08474576 0.04166667]
 [0.16666667 0.5        0.08474576 0.        ]]
```

**ii)Binarize data**

sklearn.preprocessing

Binarizer() is a method which belongs to preprocessing module.It plays a key role in the discretization of continuous feature values

**Example #1:**
A continuous data of pixels values of an 8-bit grayscale image have values ranging between 0 (black) and 255 (white) and one needs it to be black and white. So, using Binarizer() one can set a threshold converting pixel values from 0 – 127 to 0 and 128 – 255 as 1.

**Syntax:**
sklearn.preprocessing.Binarizer(threshold, copy)

**Parameters :**
threshold :[float, optional] Values less than or equal to threshold is mapped to 0, else to 1. By default threshold value is 0.0.
copy :[boolean, optional] If set to False, it avoids a copy. By default it is True.

```
import pandas as pd
data=pd.read_csv("/content/Iris (1).csv")
data.head(10)
```

| | Id | SepalLengthCm | SepalWidthCm | PetalLengthCm | PetalWidthCm | Species |
|---|---|---|---|---|---|---|
| 0 | 1 | 5.1 | 3.5 | 1.4 | 0.2 | Iris-setosa |
| 1 | 2 | 4.9 | 3.0 | 1.4 | 0.2 | Iris-setosa |
| 2 | 3 | 4.7 | 3.2 | 1.3 | 0.2 | Iris-setosa |
| 3 | 4 | 4.6 | 3.1 | 1.5 | 0.2 | Iris-setosa |
| 4 | 5 | 5.0 | 3.6 | 1.4 | 0.2 | Iris-setosa |
| 5 | 6 | 5.4 | 3.9 | 1.7 | 0.4 | Iris-setosa |
| 6 | 7 | 4.6 | 3.4 | 1.4 | 0.3 | Iris-setosa |
| 7 | 8 | 5.0 | 3.4 | 1.5 | 0.2 | Iris-setosa |
| 8 | 9 | 4.4 | 2.9 | 1.4 | 0.2 | Iris-setosa |
| 9 | 10 | 4.9 | 3.1 | 1.5 | 0.1 | Iris-setosa |

```
[6] data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             150 non-null    int64
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

```
[7] data.describe()
```

| | Id | SepalLengthCm | SepalWidthCm | PetalLengthCm | PetalWidthCm |
|---|---|---|---|---|---|
| count | 150.000000 | 150.000000 | 150.000000 | 150.000000 | 150.000000 |
| mean | 75.500000 | 5.843333 | 3.054000 | 3.758667 | 1.198667 |
| std | 43.445368 | 0.828066 | 0.433594 | 1.764420 | 0.763161 |
| min | 1.000000 | 4.300000 | 2.000000 | 1.000000 | 0.100000 |

```
[8] data.sample(10)
```

| | Id | SepalLengthCm | SepalWidthCm | PetalLengthCm | PetalWidthCm | Species |
|---|---|---|---|---|---|---|
| 97 | 98 | 6.2 | 2.9 | 4.3 | 1.3 | Iris-versicolor |
| 37 | 38 | 4.9 | 3.1 | 1.5 | 0.1 | Iris-setosa |
| 123 | 124 | 6.3 | 2.7 | 4.9 | 1.8 | Iris-virginica |
| 5 | 6 | 5.4 | 3.9 | 1.7 | 0.4 | Iris-setosa |
| 110 | 111 | 6.5 | 3.2 | 5.1 | 2.0 | Iris-virginica |
| 69 | 70 | 5.6 | 2.5 | 3.9 | 1.1 | Iris-versicolor |
| 6 | 7 | 4.6 | 3.4 | 1.4 | 0.3 | Iris-setosa |
| 99 | 100 | 5.7 | 2.8 | 4.1 | 1.3 | Iris-versicolor |
| 23 | 24 | 5.1 | 3.3 | 1.7 | 0.5 | Iris-setosa |
| 109 | 110 | 7.2 | 3.6 | 6.1 | 2.5 | Iris-virginica |

```
[9] data.columns

Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',
       'Species'],
      dtype='object')
```

```
data.shape

(150, 6)
```

```
#data[start:end]
print(data[10:21])

    Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm      Species
10  11            5.4           3.7            1.5           0.2  Iris-setosa
11  12            4.8           3.4            1.6           0.2  Iris-setosa
12  13            4.8           3.0            1.4           0.1  Iris-setosa
13  14            4.3           3.0            1.1           0.1  Iris-setosa
14  15            5.8           4.0            1.2           0.2  Iris-setosa
15  16            5.7           4.4            1.5           0.4  Iris-setosa
16  17            5.4           3.9            1.3           0.4  Iris-setosa
17  18            5.1           3.5            1.4           0.3  Iris-setosa
18  19            5.7           3.8            1.7           0.3  Iris-setosa
19  20            5.1           3.8            1.5           0.3  Iris-setosa
20  21            5.4           3.4            1.7           0.2  Iris-setosa
```

You can also save it in a variable for further use in analysis
**sliced_data=data[10:21]**

```
sliced_data=data[10:21]
print(sliced_data)
```

```
    Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm      Species
10  11            5.4           3.7            1.5           0.2  Iris-setosa
11  12            4.8           3.4            1.6           0.2  Iris-setosa
12  13            4.8           3.0            1.4           0.1  Iris-setosa
13  14            4.3           3.0            1.1           0.1  Iris-setosa
14  15            5.8           4.0            1.2           0.2  Iris-setosa
15  16            5.7           4.4            1.5           0.4  Iris-setosa
16  17            5.4           3.9            1.3           0.4  Iris-setosa
17  18            5.1           3.5            1.4           0.3  Iris-setosa
18  19            5.7           3.8            1.7           0.3  Iris-setosa
19  20            5.1           3.8            1.5           0.3  Iris-setosa
20  21            5.4           3.4            1.7           0.2  Iris-setosa
```

```
specific_data=data[["Id","Species"]]
print(specific_data)
```

```
      Id         Species
0      1     Iris-setosa
1      2     Iris-setosa
2      3     Iris-setosa
3      4     Iris-setosa
4      5     Iris-setosa
..   ...             ...
145  146  Iris-virginica
146  147  Iris-virginica
147  148  Iris-virginica
148  149  Iris-virginica
149  150  Iris-virginica

[150 rows x 2 columns]
```

```
data.loc[data["Species"]=="Iris-setosa"]
```

| | Id | SepalLengthCm | SepalWidthCm | PetalLengthCm | PetalWidthCm | Species |
|---|---|---|---|---|---|---|
| 0 | 1 | 5.1 | 3.5 | 1.4 | 0.2 | Iris-setosa |
| 1 | 2 | 4.9 | 3.0 | 1.4 | 0.2 | Iris-setosa |
| 2 | 3 | 4.7 | 3.2 | 1.3 | 0.2 | Iris-setosa |
| 3 | 4 | 4.6 | 3.1 | 1.5 | 0.2 | Iris-setosa |
| 4 | 5 | 5.0 | 3.6 | 1.4 | 0.2 | Iris-setosa |
| 5 | 6 | 5.4 | 3.9 | 1.7 | 0.4 | Iris-setosa |
| 6 | 7 | 4.6 | 3.4 | 1.4 | 0.3 | Iris-setosa |
| 7 | 8 | 5.0 | 3.4 | 1.5 | 0.2 | Iris-setosa |
| 8 | 9 | 4.4 | 2.9 | 1.4 | 0.2 | Iris-setosa |
| 9 | 10 | 4.9 | 3.1 | 1.5 | 0.1 | Iris-setosa |
| 10 | 11 | 5.4 | 3.7 | 1.5 | 0.2 | Iris-setosa |
| 11 | 12 | 4.8 | 3.4 | 1.6 | 0.2 | Iris-setosa |

```
data.loc[data["Species"]=="Iris-virginica"]
```

| | Id | SepalLengthCm | SepalWidthCm | PetalLengthCm | PetalWidthCm | Species |
|---|---|---|---|---|---|---|
| 100 | 101 | 6.3 | 3.3 | 6.0 | 2.5 | Iris-virginica |
| 101 | 102 | 5.8 | 2.7 | 5.1 | 1.9 | Iris-virginica |
| 102 | 103 | 7.1 | 3.0 | 5.9 | 2.1 | Iris-virginica |
| 103 | 104 | 6.3 | 2.9 | 5.6 | 1.8 | Iris-virginica |
| 104 | 105 | 6.5 | 3.0 | 5.8 | 2.2 | Iris-virginica |

```
data.iloc[2]
```

```
Id                        3
SepalLengthCm           4.7
SepalWidthCm            3.2
PetalLengthCm           1.3
PetalWidthCm            0.2
Species         Iris-setosa
Name: 2, dtype: object
```

```
data["Species"].value_counts()
```

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

**PROGRAM**

```
[1] from sklearn.preprocessing  import Binarizer
    import pandas
    import numpy as np
    url="https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.csv"
```

```
[8] colnames=['preg','plas','pres','skin','test','mass','pedi','age',
    'class']
```

```
[10] print(colnames)

    ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
```

```
[9] data=pandas.read_csv(url,names=colnames)
```

```
[6] print(data)

         preg  plas  pres  skin  test  mass   pedi  age  class
    0       6   148    72    35     0  33.6  0.627   50      1
    1       1    85    66    29     0  26.6  0.351   31      0
    2       8   183    64     0     0  23.3  0.672   32      1
    3       1    89    66    23    94  28.1  0.167   21      0
    4       0   137    40    35   168  43.1  2.288   33      1
    ..    ...   ...   ...   ...   ...   ...    ...  ...    ...
    763    10   101    76    48   180  32.9  0.171   63      0
    764     2   122    70    27     0  36.8  0.340   27      0
    765     5   121    72    23   112  26.2  0.245   30      0
    766     1   126    60     0     0  30.1  0.349   47      1
    767     1    93    70    31     0  30.4  0.315   23      0

    [768 rows x 9 columns]
```

```
[11] array=data.values
```

```
[12] array

    array([[  6.   , 148.   ,  72.   , ...,   0.627,  50.   ,   1.   ],
           [  1.   ,  85.   ,  66.   , ...,   0.351,  31.   ,   0.   ],
           [  8.   , 183.   ,  64.   , ...,   0.672,  32.   ,   1.   ],
           ...,
           [  5.   , 121.   ,  72.   , ...,   0.245,  30.   ,   0.   ],
           [  1.   , 126.   ,  60.   , ...,   0.349,  47.   ,   1.   ],
           [  1.   ,  93.   ,  70.   , ...,   0.315,  23.   ,   0.   ]])
```

```
[13] X=array[:,0:8]
     Y=array[:,8]
```

```
print(X)

[[  6.    148.     72.    ...  33.6     0.627  50.   ]
 [  1.     85.     66.    ...  26.6     0.351  31.   ]
 [  8.    183.     64.    ...  23.3     0.672  32.   ]
 ...
 [  5.    121.     72.    ...  26.2     0.245  30.   ]
 [  1.    126.     60.    ...  30.1     0.349  47.   ]
 [  1.     93.     70.    ...  30.4     0.315  23.   ]]
```

```
print(Y)
```

```
[1. 0. 1. 0. 1. 0. 1. 0. 1. 1. 0. 1. 0. 1. 1. 1. 1. 1. 0. 1. 0. 0. 1. 1.
 1. 1. 1. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 1. 1. 1. 0. 0. 0. 1. 0. 1. 0. 0.
 1. 0. 0. 0. 0. 1. 0. 0. 1. 0. 0. 0. 0. 1. 0. 0. 1. 0. 1. 0. 0. 0. 1. 0.
 1. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 1. 0. 0. 0. 1. 0. 0. 0. 0. 1. 0. 0.
 0. 0. 0. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 0. 0. 1. 1. 1. 0. 0. 0.
 1. 0. 0. 0. 1. 1. 0. 0. 1. 1. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.
 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 1. 1. 0. 0. 0. 1. 0. 0. 0. 0. 1. 1. 0. 0.
 0. 0. 1. 1. 0. 0. 0. 1. 0. 1. 0. 1. 0. 0. 0. 0. 0. 1. 1. 1. 1. 1. 0. 0.
 1. 1. 0. 1. 0. 1. 1. 1. 0. 0. 0. 0. 0. 0. 1. 1. 0. 1. 0. 0. 0. 1. 1. 1.
 1. 0. 1. 1. 1. 1. 0. 0. 0. 0. 0. 1. 0. 0. 1. 1. 0. 0. 0. 1. 1. 1. 1. 0.
 0. 0. 1. 1. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 0. 0. 0. 1. 0. 1. 0. 0.
 1. 0. 1. 0. 0. 1. 1. 0. 0. 0. 0. 0. 1. 0. 0. 0. 1. 0. 0. 1. 1. 0. 0. 1.
 0. 0. 0. 1. 1. 1. 0. 0. 1. 0. 1. 0. 1. 1. 0. 1. 0. 0. 1. 0. 1. 1. 0. 0.
 1. 0. 1. 0. 0. 1. 0. 1. 0. 1. 1. 1. 0. 0. 1. 0. 1. 0. 0. 0. 1. 0. 0. 0.
 0. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 1. 1. 1. 0. 1.
 1. 0. 0. 1. 0. 0. 1. 0. 0. 1. 1. 0. 0. 0. 0. 1. 0. 0. 1. 0. 0. 0. 0. 0.
 0. 0. 1. 1. 1. 0. 0. 1. 0. 0. 1. 0. 0. 1. 0. 1. 1. 0. 1. 0. 1. 0. 1. 0.
 1. 1. 0. 0. 0. 0. 1. 1. 0. 1. 0. 1. 0. 0. 0. 0. 1. 1. 0. 1. 0. 1. 0. 0.
 0. 0. 0. 1. 0. 0. 0. 0. 1. 0. 0. 1. 1. 1. 0. 0. 1. 0. 0. 1. 0. 0. 0. 1.
 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0.
```

```
[17] binarizer=Binarizer(threshold=0.0).fit(X)
     binaryX=binarizer.transform(X)
```

```
print(binaryX[0:10,:])
```

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

**iii)STANDARDIZE DATA:**

Data standardization is **the process of rescaling the attributes so that they have mean as 0 and variance as 1**. The ultimate goal to perform standardization is to bring down all the features to a common scale without distorting the differences in the range of the values.

```
[2] from sklearn.preprocessing import StandardScaler
    import pandas
    import numpy as np
    url="https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.csv"
```

```
[3] colnames=['preg','plas','pres','skin','test','mass','pedi','age',
    'class']
```

```
[4] print(colnames)
```

```
['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
```

```
[5] data=pandas.read_csv(url,names=colnames)
```

```
[6] print(data)
```

```
     preg  plas  pres  skin  test  mass   pedi  age  class
0       6   148    72    35     0  33.6  0.627   50      1
1       1    85    66    29     0  26.6  0.351   31      0
2       8   183    64     0     0  23.3  0.672   32      1
3       1    89    66    23    94  28.1  0.167   21      0
4       0   137    40    35   168  43.1  2.288   33      1
..    ...   ...   ...   ...   ...   ...    ...  ...    ...
763    10   101    76    48   180  32.9  0.171   63      0
764     2   122    70    27     0  36.8  0.340   27      0
765     5   121    72    23   112  26.2  0.245   30      0
766     1   126    60     0     0  30.1  0.349   47      1
767     1    93    70    31     0  30.4  0.315   23      0
```

```
[7]  array=data.values
```

```
[8]  array
```

```
array([[  6.   , 148.   ,  72.   , ...,   0.627,  50.   ,   1.   ],
       [  1.   ,  85.   ,  66.   , ...,   0.351,  31.   ,   0.   ],
       [  8.   , 183.   ,  64.   , ...,   0.672,  32.   ,   1.   ],
       ...,
       [  5.   , 121.   ,  72.   , ...,   0.245,  30.   ,   0.   ],
       [  1.   , 126.   ,  60.   , ...,   0.349,  47.   ,   1.   ],
       [  1.   ,  93.   ,  70.   , ...,   0.315,  23.   ,   0.   ]])
```

```
[10] X=array[:,0:8]
     Y=array[:,8]
```

```
[11] print(X)
```

```
[[  6.    148.     72.    ...  33.6     0.627  50.   ]
 [  1.     85.     66.    ...  26.6     0.351  31.   ]
 [  8.    183.     64.    ...  23.3     0.672  32.   ]
 ...
 [  5.    121.     72.    ...  26.2     0.245  30.   ]
 [  1.    126.     60.    ...  30.1     0.349  47.   ]
 [  1.     93.     70.    ...  30.4     0.315  23.   ]]
```

```
[12] print(Y)
```

```
[1. 0. 1. 0. 1. 0. 1. 0. 1. 1. 0. 1. 0. 1. 1. 1. 1. 1. 0. 1. 0. 0. 1. 1.
 1. 1. 1. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 1. 1. 1. 0. 0. 0. 1. 0. 1. 0. 0.
 1. 0. 0. 0. 0. 1. 0. 0. 1. 0. 0. 0. 0. 1. 0. 0. 1. 0. 1. 0. 0. 0. 1. 0.
```

```
[13] scaler=StandardScaler().fit(X)
     rescaledX=scaler.transform(X)
```

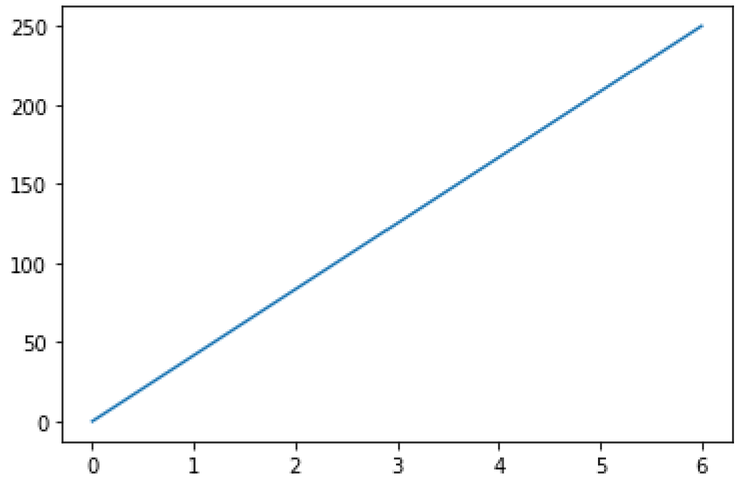
```
print(rescaledX)
```

```
[[ 0.63994726  0.84832379  0.14964075 ...  0.20401277  0.46849198
   1.4259954 ]
 [-0.84488505 -1.12339636 -0.16054575 ... -0.68442195 -0.36506078
  -0.19067191]
 [ 1.23388019  1.94372388 -0.26394125 ... -1.10325546  0.60439732
  -0.10558415]
 ...
 [ 0.3429808   0.00330087  0.14964075 ... -0.73518964 -0.68519336
  -0.27575966]
 [-0.84488505  0.1597866  -0.47073225 ... -0.24020459 -0.37110101
   1.17073215]
 [-0.84488505 -0.8730192   0.04624525 ... -0.20212881 -0.47378505
  -0.87137393]]
```

**Week – 12**

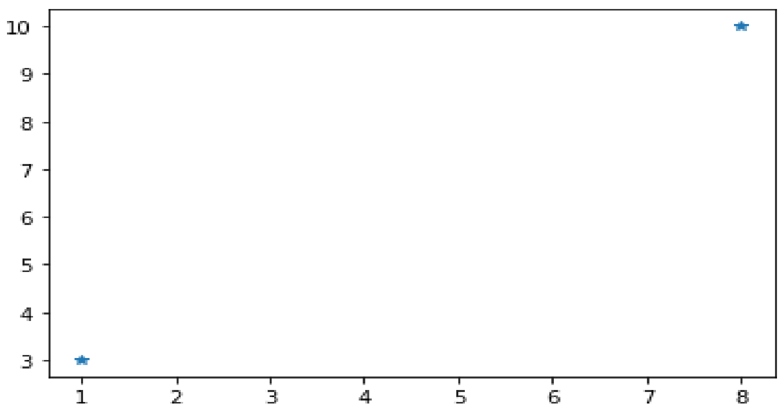
**Visualize the datasets using matplotlib in python.(Histogram, Box plot, Bar chart, Pie chart etc.,)**

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

xpoints=np.array([0,6])
ypoints=np.array([0,250])

plt.plot(xpoints,ypoints)
plt.show()
```

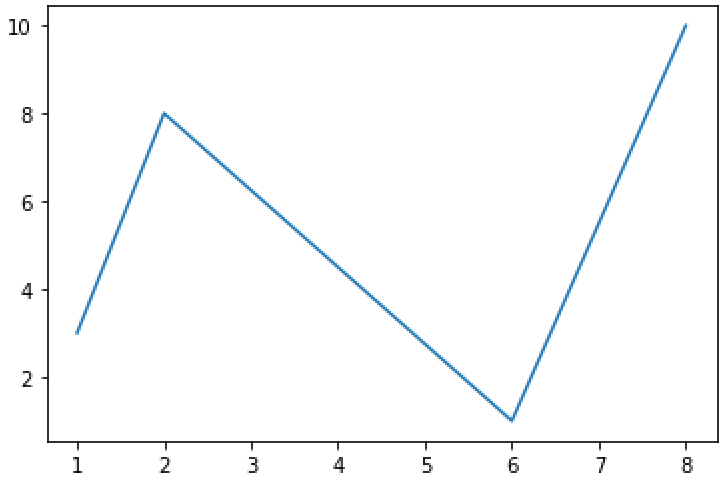



The plot() function is used to draw points (markers)in a diagram.By default,the plot() function Draws a line from point to point.The function takes a parameters for specifying points in the diagram.Parameter1 is an array containing the points on the x-axis.
Parameter 2 is an array containing the points on the y-axis.

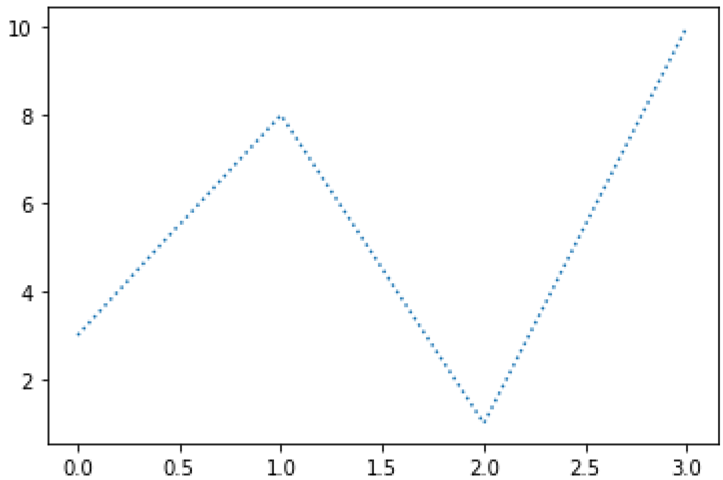
```
#plotting with out a line
import matplotlib.pyplot as plt
import numpy as np

xpoints=np.array([1,8])
ypoints=np.array([3,10])

plt.plot(xpoints,ypoints,'*')
plt.show()
```



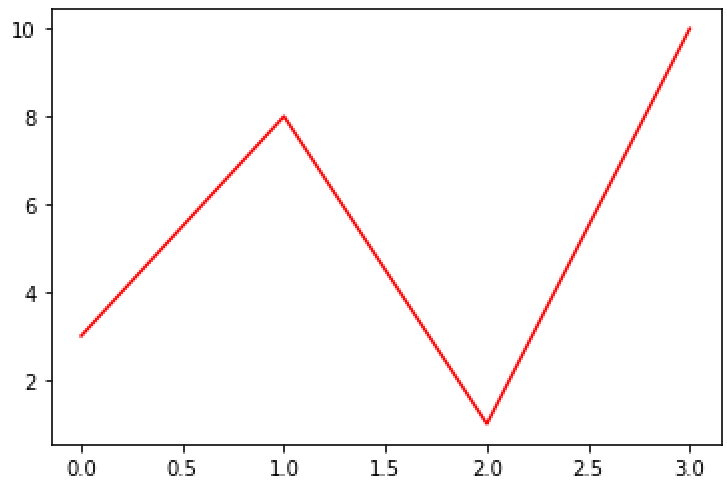
```
#multiple points
import matplotlib.pyplot as plt
import numpy as np

xpoints=np.array([1,2,6,8])
ypoints=np.array([3,8,1,10])

plt.plot(xpoints,ypoints)
plt.show()
```



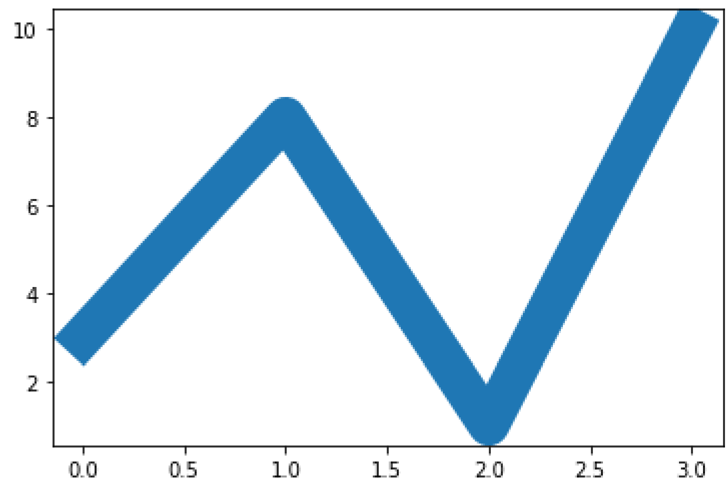
```
#Linestyle:You can use keyword argument
#linestyle,or shorter is,to change the style of the plotted line
import matplotlib.pyplot as plt
import numpy as np

ypoints=np.array([3,8,1,10])

plt.plot(ypoints,linestyle='dotted')
plt.show()
```



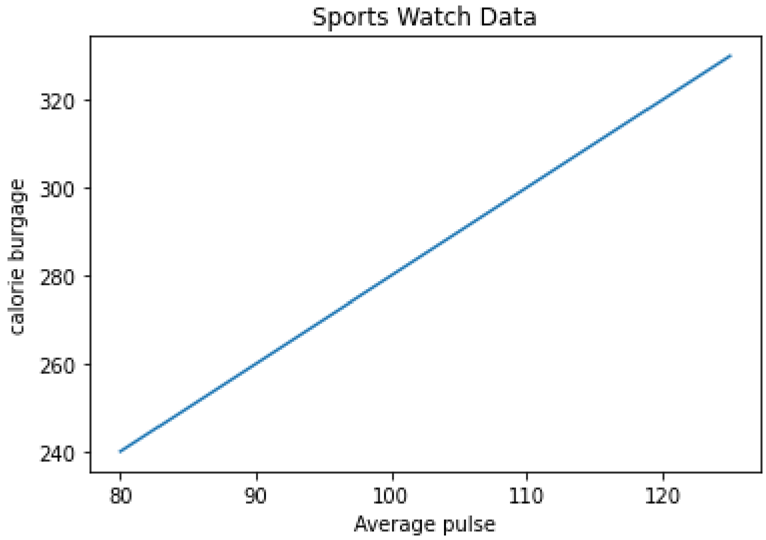
```
import matplotlib.pyplot as plt
import numpy as np

ypoints=np.array([3,8,1,10])

plt.plot(ypoints,color='r')
plt.show()
```




```
#Line width
import matplotlib.pyplot as plt
import numpy as np

ypoints=np.array([3,8,1,10])

plt.plot(ypoints,linewidth='20.5')
plt.show()
```



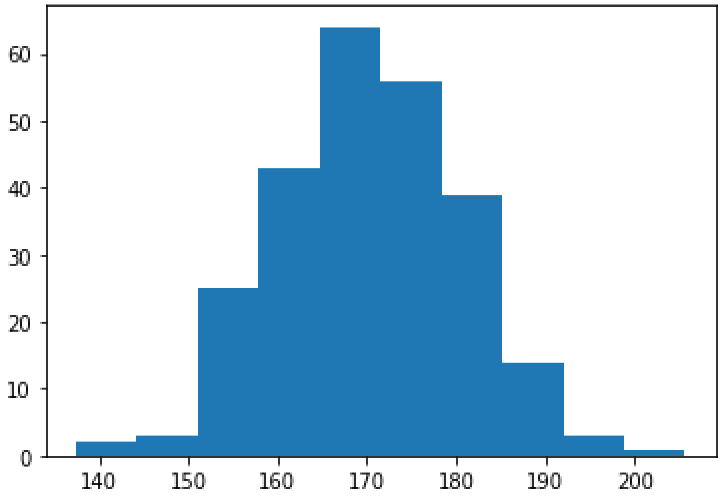
```
import matplotlib.pyplot as plt

x=np.array([80,85,90,95,100,105,110,115,120,125])
y=np.array([240,250,260,270,280,290,300,310,320,330])

plt.plot(x,y)

plt.title("Sports Watch Data")
plt.xlabel("Average pulse")
plt.ylabel("calorie burgage")
plt.show()
```




```
#A normal data distribution by numpy
import numpy as np

x=np.random.normal(170,10,250)

print(x)
```

```
[162.06219045 168.89655805 178.02696963 173.43754038 168.10975633
 167.257875   160.94525091 171.33103894 174.07918171 170.8098853
 176.02079345 182.46538275 178.28976614 175.821064   171.2425441
 180.34654865 167.67320374 179.9604204  163.11327908 167.64301519
 156.65948856 181.27839547 162.84200082 171.8238321  175.71627964
 164.50983203 170.4828843  154.55699173 169.39975546 163.27211912
 168.03416565 149.03720244 167.31908601 161.96142092 173.05782436
 172.27035122 170.52218219 183.26859979 151.07846246 168.80611226
 163.90645598 180.21534805 166.31669544 176.74855211 177.53949826
 177.9365034  166.76187359 168.26793957 165.03540585 161.48454623
 187.65374431 183.84429794 190.83578551 168.30900982 150.95911974
```

```
#The hist() function will read the array and produce a histogram:
import matplotlib.pyplot as plt
import numpy as np

x=np.random.normal(170,10,250)

plt.hist(x)
plt.show()
```




**Box Plot**: It is a type of chart that depicts a group of numerical data through their quartiles. It is a simple way to visualize the shape of our data. It makes comparing characteristics of data between categories very easy

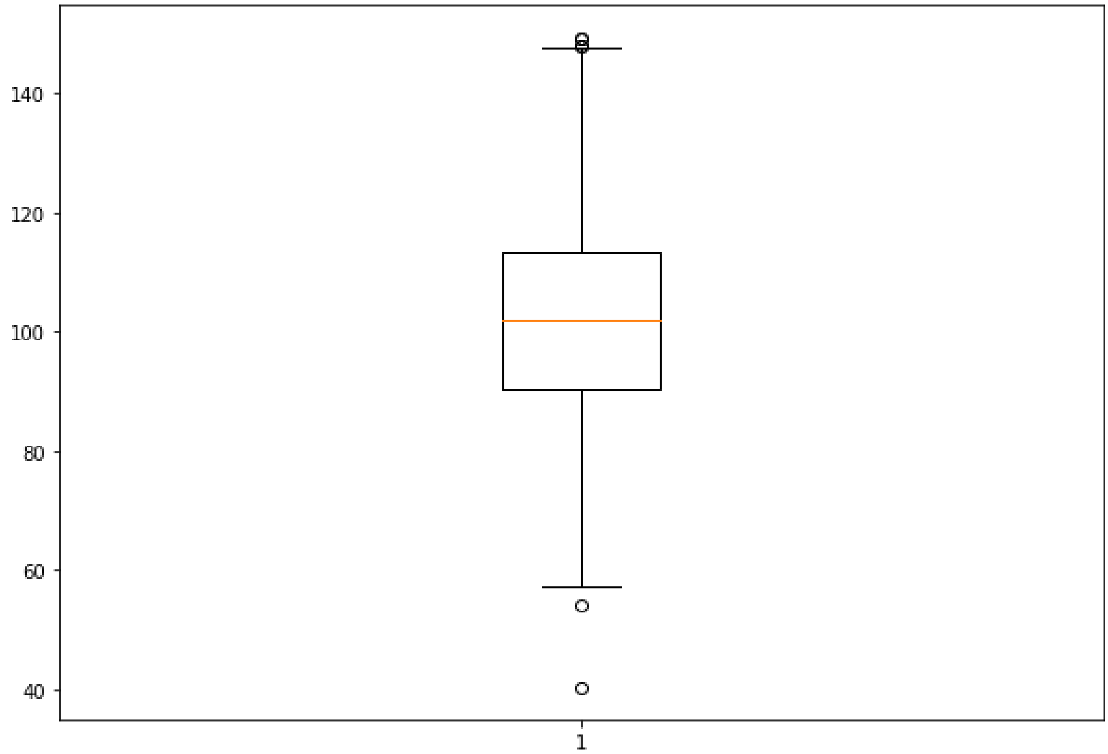
**Uses of a Box Plot**

Box plots provide a visual summary of the data with which we can quickly identify the average value of the data, how dispersed the data is, whether the data is skewed or not (skewness).

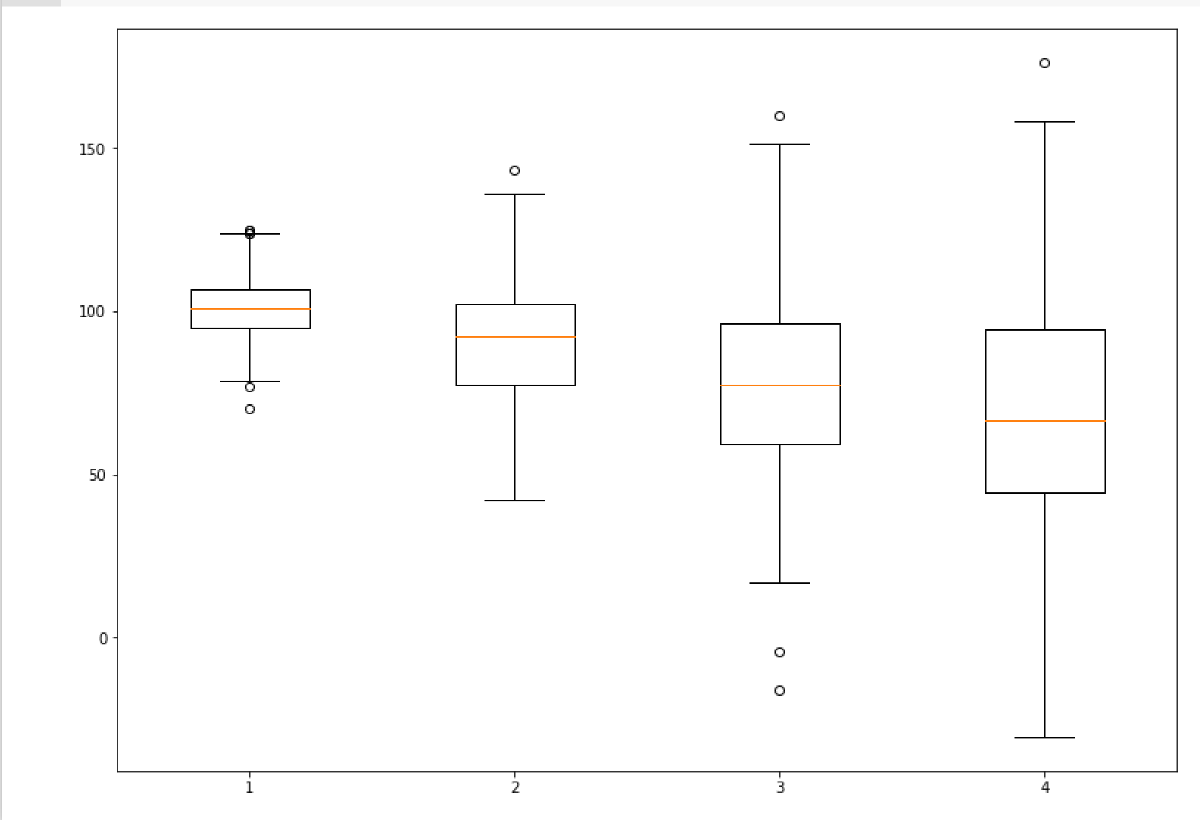
The Median gives you the average value of the data.

Box Plots shows Skewness of the data-

```
import numpy as np
#creating dataset
np.random.seed(10)
data=np.random.normal(100,20,200)

fig=plt.figure(figsize=(10,7))

#creating plot

plt.boxplot(data)

#show plot
plt.show()
```




```
import matplotlib.pyplot as plt
import numpy as np
#creating dataset
np.random.seed(10)

data_1=np.random.normal(100,10,200)
data_2=np.random.normal(90,20,200)
data_3=np.random.normal(80,30,200)
data_4=np.random.normal(70,40,200)
data=[data_1,data_2,data_3,data_4]

fig=plt.figure(figsize=(10,7))
#creating axes instance
ax=fig.add_axes([0,0,1,1])\

#creating plot
bp=ax.boxplot(data)

#show plot
plt.show()
```




**Bar Chart:**

A Bar graph or a Histogram is the diagrammatic representation of data in statistics. In bar graphs or histograms, the use of graphs, charts, and tabular data makes it very easy to understand the concept and relationships among data.

The pictorial representation of data in groups, either in horizontal or vertical bars where the length of the bar represents the value of the data present on axis. They (bar graphs) are usually used to display or impart the information belonging to ‘categorical data’ i.e; data that fit in some category.

```python
import matplotlib.pyplot as plt
fig=plt.figure()
ax=fig.add_axes([0,0,1,1])
langs=['C','C++','Java','Python','PHP']
students=[23,17,35,29,12]
ax.bar(langs,students)
plt.show()
```