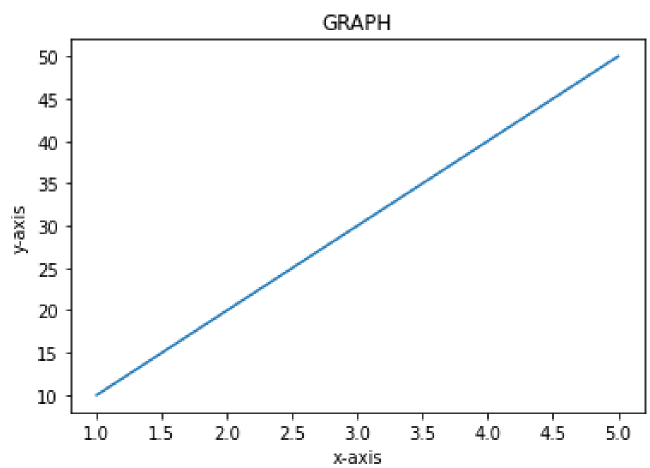
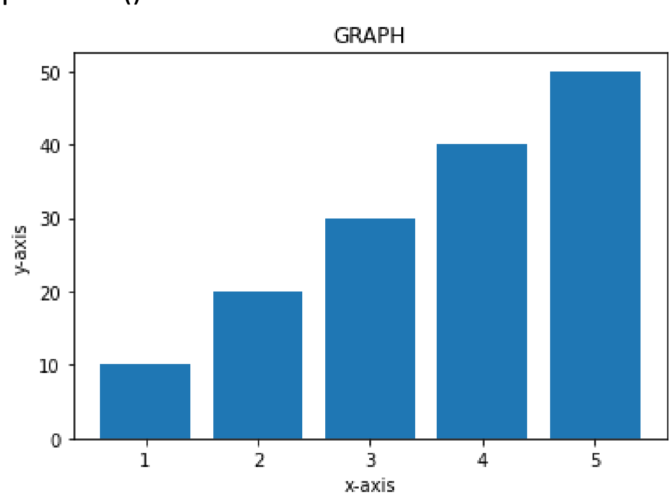
To study about data visualization tools using matplotlib library

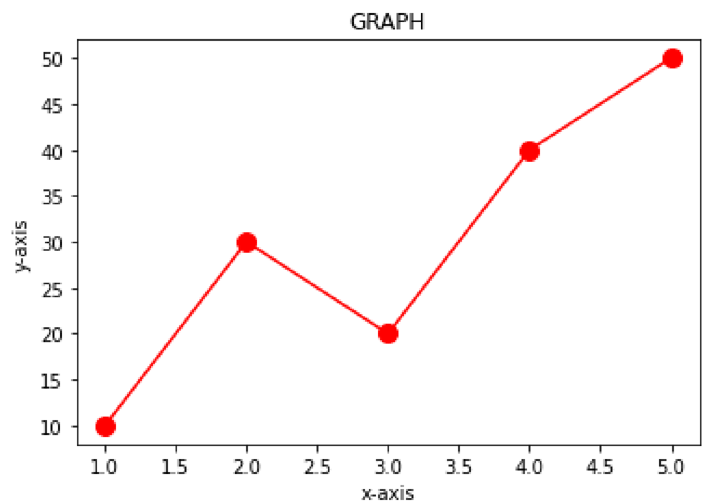
```
import matplotlib.pyplot as plt
x=[1,2,3,4,5]
y=[10,20,30,40,50]
plt.plot(x,y)
plt.xlabel("x-axis")
plt.ylabel("y-axis")
plt.title("GRAPH")
plt.show()
```



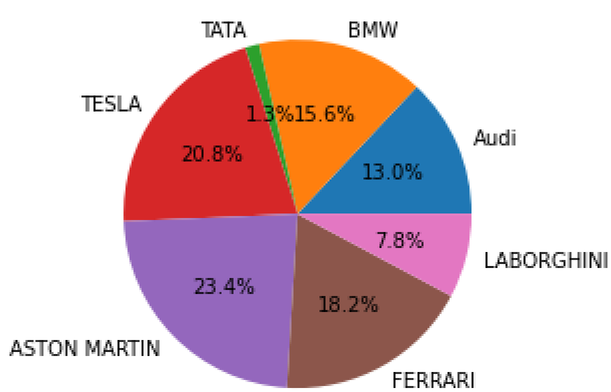

```
x=[1,2,3,4,5]
y=[10,20,30,40,50]
plt.bar(x,y)
plt.xlabel("x-axis")
plt.ylabel("y-axis")
plt.title("GRAPH")
plt.show()
```



```
x=[1,2,3,4,5]
y=[10,30,20,40,50]
plt.plot(x,y,marker = 'o', ms="10", color="red")
plt.xlabel("x-axis")
plt.ylabel("y-axis")
plt.title("GRAPH")
plt.show()
```

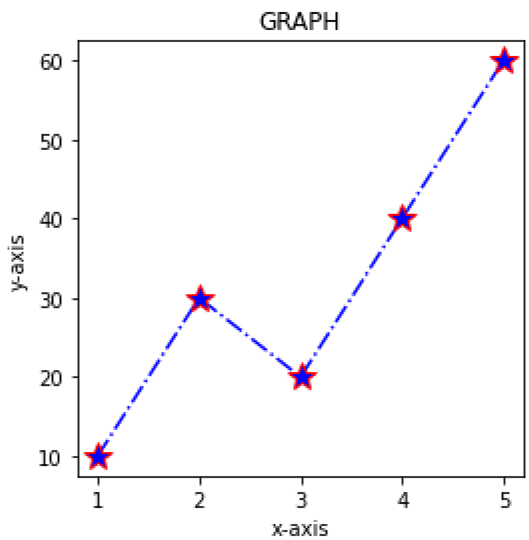



```
cars=["Audi","BMW","TATA","TESLA","ASTON MARTIN","FERRARI","LABORGHINI"]
price=[50,60,5,80,90,70,30]
plt.pie(price,labels=cars  ,autopct='%1.1f%%')
plt.show()
```

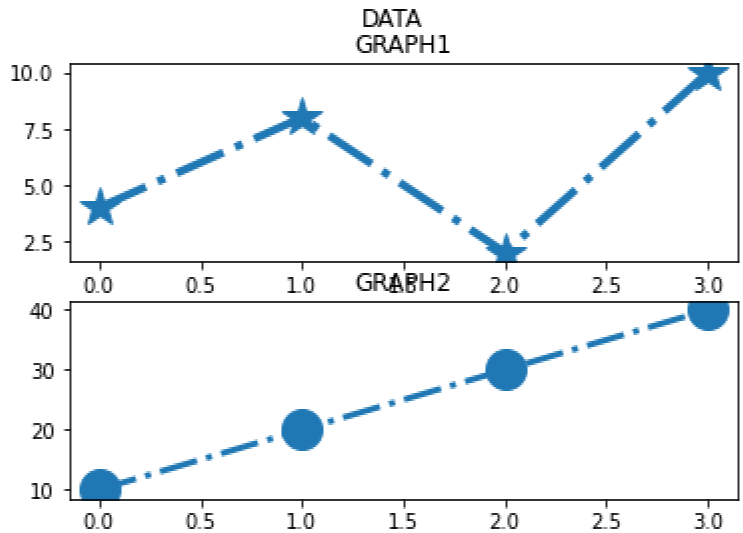



```
x=[1,2,3,4,5]
y=[10,30,20,40,60]
plt.figure(figsize=(4,4))
plt.plot(x,y,marker="*", ms="15", mec="red" ,linestyle="dashdot",color="blue")
plt.tick_params()
```

```
plt.xlabel("x-axis")
plt.ylabel("y-axis")
plt.title("GRAPH")
plt.show()
```




```
import numpy as np
x = np.array([0, 1, 2, 3])
y = np.array([4, 8, 2, 10])
plt.subplot(2, 1, 1)
plt.title("GRAPH1")
plt.plot(x,y, linestyle="dashdot" ,marker="*",ms="20",linewidth="4")
x = np.array([0, 1, 2, 3])
y = np.array([10, 20, 30, 40])
plt.subplot(2, 1, 2)
plt.plot(x,y, linestyle="dashdot" ,marker="o",ms="20",linewidth="3")
plt.title("GRAPH2")
plt.suptitle("DATA")
plt.show()
```




```
import numpy as np
import matplotlib.pyplot as plt
x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])
plt.subplot(1, 2, 1)
plt.plot(x,y)
plt.title("SALES")
x = np.array([0, 1, 2, 3])
y = np.array([10, 20, 30, 40])
plt.subplot(1, 2, 2)
plt.plot(x,y)
plt.title("INCOME")
plt.suptitle("MY SHOP")
plt.show()
```

1)On the fruit dataset, compare the performance of Logistic Regression, SVM, KNN on the basis of their accuracy.

```
import pandas as pd   # to load dataset
import matplotlib.pyplot as plt
import seaborn as sns
import pylab as pl
from sklearn.model_selection import train_test_split   # for splitting dataset
from sklearn.preprocessing import MinMaxScaler  # for scaling
from sklearn.linear_model import LogisticRegression   # machine learning lib/model, # get accuracy by Logistic regression
from sklearn.tree import DecisionTreeClassifier  # get accuracy by Decision Tree classifier
from sklearn.neighbors import KNeighborsClassifier  # get accuracy by KNN classifier
from sklearn.naive_bayes import GaussianNB # get accuracy by GNB classifier
df=pd.read_csv('fruit_data.csv')
df.shape
df.describe()
```

	fruit_label	mass	width	height	color_score
count	59.000000	59.000000	59.000000	59.000000	59.000000
mean	2.542373	163.118644	7.105085	7.693220	0.762881
std	1.208048	55.018832	0.816938	1.361017	0.076857
min	1.000000	76.000000	5.800000	4.000000	0.550000
25%	1.000000	140.000000	6.600000	7.200000	0.720000
50%	3.000000	158.000000	7.200000	7.600000	0.750000
75%	4.000000	177.000000	7.500000	8.200000	0.810000
max	4.000000	362.000000	9.600000	10.500000	0.930000

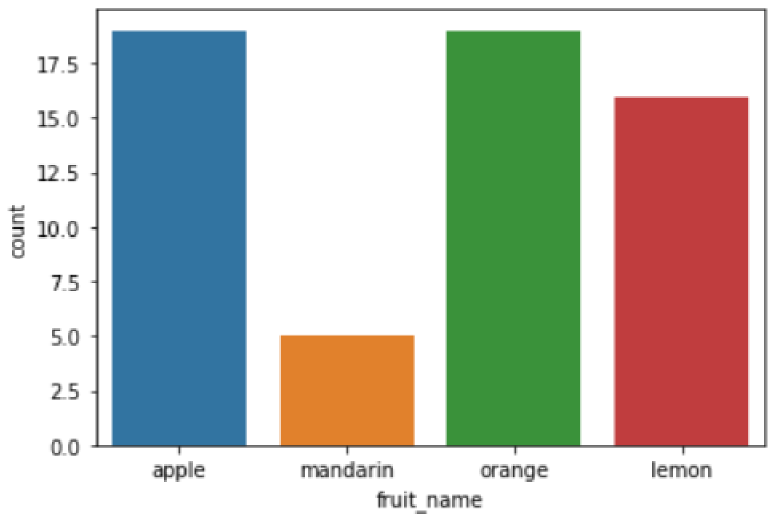
```
print(df['fruit_name'].unique()) # unique fruits name
```

```
['apple' 'mandarin' 'orange' 'lemon']
```

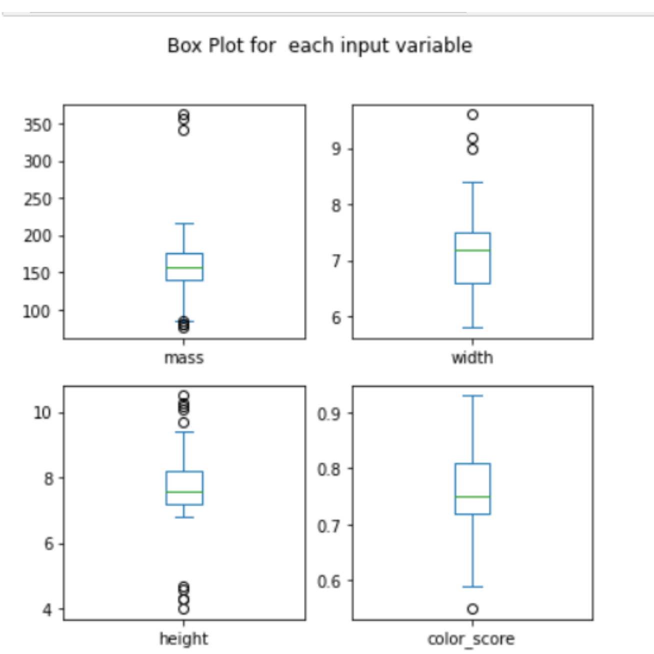
```
print(df['fruit_subtype'].unique()) # unique fruit subtype
```

```
['granny_smith' 'mandarin' 'braeburn' 'golden_delicious' 'cripps_pink'
 'spanish_jumbo' 'selected_seconds' 'turkey_navel' 'spanish_belsan'
 'unknown']
```

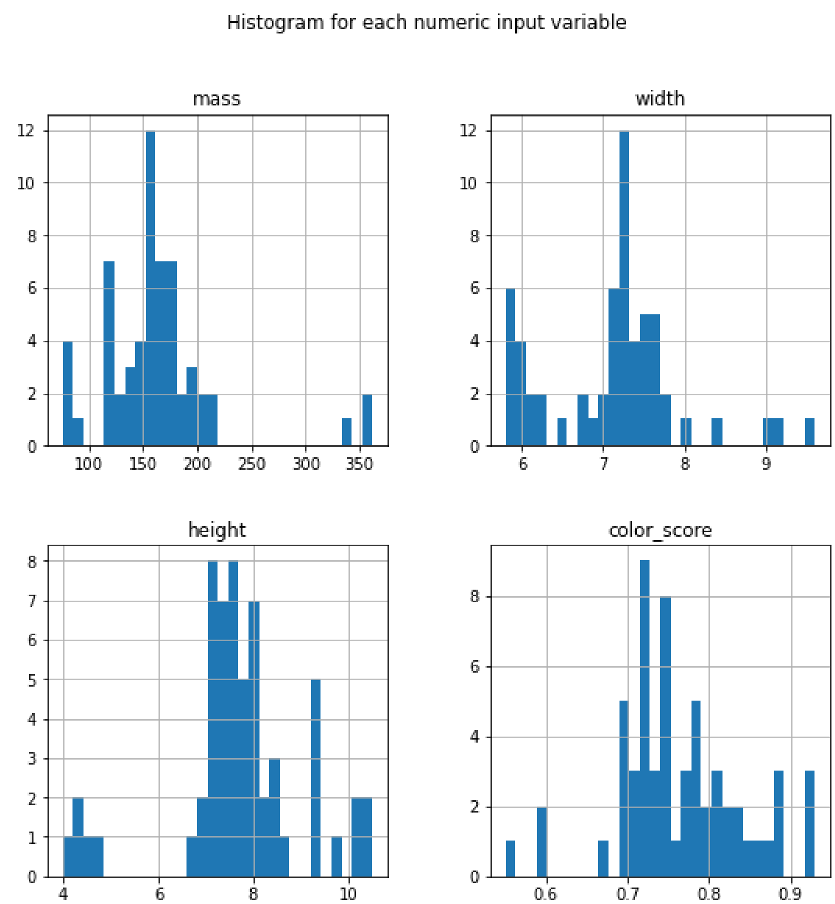
```
sns.countplot(df['fruit_name'],label='Count') # count plot
plt.show()
```



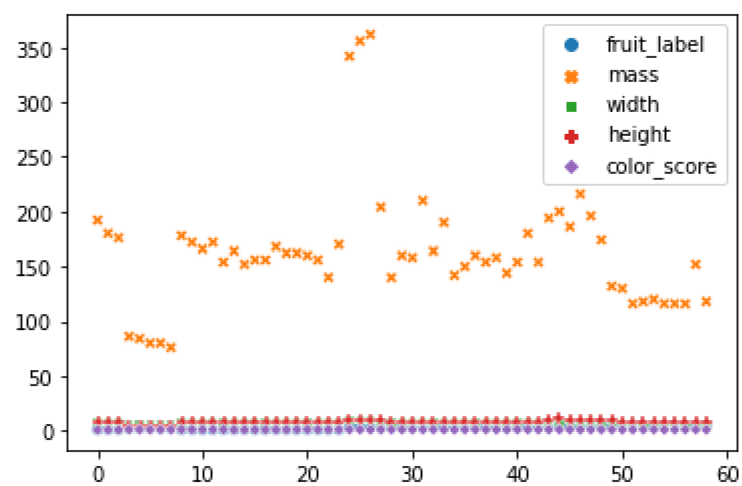

```
df.drop('fruit_label',axis=1).plot(kind='box', subplots=True, layout=(2,2), sharex=False, sharey=False, figsize=(6,6), title='Box Plot for  each input variable')
plt.savefig('fruits_box')
plt.show()
```




```
import pylab as pl
df.drop('fruit_label', axis=1).hist(bins=30, figsize=(9,9))
pl.suptitle("Histogram for each numeric input variable")
plt.savefig('fruits_hist')
```




```
#scaterplot
sns.scatterplot(data=df)
```




```
#preparing data with scaling
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
feature_names = ['mass', 'width', 'height', 'color_score']
x=df[feature_names]
```

```
y=df['fruit_label']
x_train, x_test, y_train, y_test = train_test_split(x,y, random_state=0)
print(x_train[:3]) # to check output
scaler = MinMaxScaler()
x_train=scaler.fit_transform(x_train)
x_test= scaler.transform(x_test)
print("\nAfter scaling\n")
print(x_train[:3]) # to check output
```

```
    mass  width  height  color_score
42   154    7.2     7.2         0.82
48   174    7.3    10.1         0.72
7     76    5.8     4.0         0.81

After scaling

[[0.27857143 0.41176471 0.49230769 0.72972973]
 [0.35       0.44117647 0.93846154 0.45945946]
 [0.         0.         0.         0.7027027 ]]
```

```
from sklearn.linear_model import LogisticRegression   # machine learning lib/model
feature_names = ['mass', 'width', 'height', 'color_score']
x=df[feature_names]
y=df['fruit_label']
x_train, x_test, y_train, y_test = train_test_split(x,y, random_state=0)

scaler = MinMaxScaler()
x_train=scaler.fit_transform(x_train)
x_test= scaler.transform(x_test)
#logistic regression
logreg = LogisticRegression()  # machine learning algorithm
logreg.fit(x_train, y_train)
#print score of train data
print('Accuracy of Logistic regression classifier on training set:{:.2f}'
    .format(logreg.score(x_train, y_train)))
#print score of test data
print('Accuracy of Logistic regression classifier on test set:{:.2f}'
    .format(logreg.score(x_test, y_test)))
```

```
Accuracy of Logistic regression classifier on training set:0.75
Accuracy of Logistic regression classifier on test set:0.47
```

```
from sklearn.neighbors import KNeighborsClassifier
# KNN method
knn = KNeighborsClassifier()
knn.fit(x_train, y_train)
#print score of train data
print('Accuracy of KNN classifier on training set:{:.2f}'
    .format(knn.score(x_train, y_train)))
#print score of test data
print('Accuracy of KNN Classifier on test set:{:.2f}'
    .format(knn.score(x_test, y_test)))
```

```
Accuracy of KNN classifier on training set:0.95
Accuracy of KNN Classifier on test set:1.00
```

```
from sklearn.svm import SVC
# SVM classifier
svm = SVC()
svm.fit(x_train, y_train)
#print score of train data
print('Accuracy of SVM classifier on training set:{:.2f}'
    .format(svm.score(x_train, y_train)))
#print score of test data
print('Accuracy of SVM Classifier on test set:{:.2f}'
    .format(svm.score(x_test, y_test)))
```

```
Accuracy of SVM classifier on training set:0.91
Accuracy of SVM Classifier on test set:0.80
```

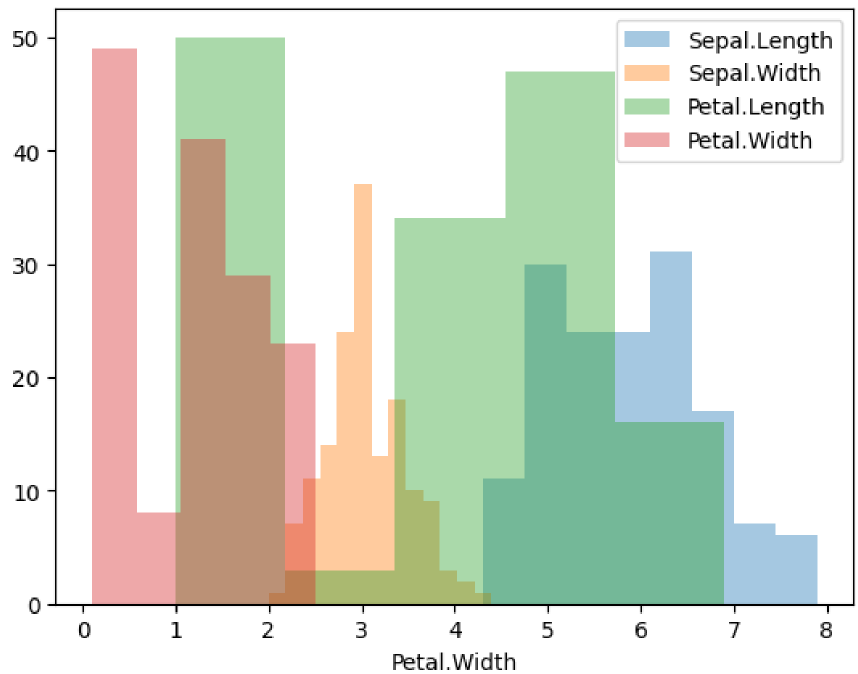
```
data = {'Training Accuracy (in %)':[75,95,91],'Testing Accuracy (in %)':[47,100,80]}
df1 = pd.DataFrame(data, index =['Logistic Regression','K-Nearest Neighbour
(KNN)','Support Vector Machine (SVM)'])
df1
```

Out[19]:

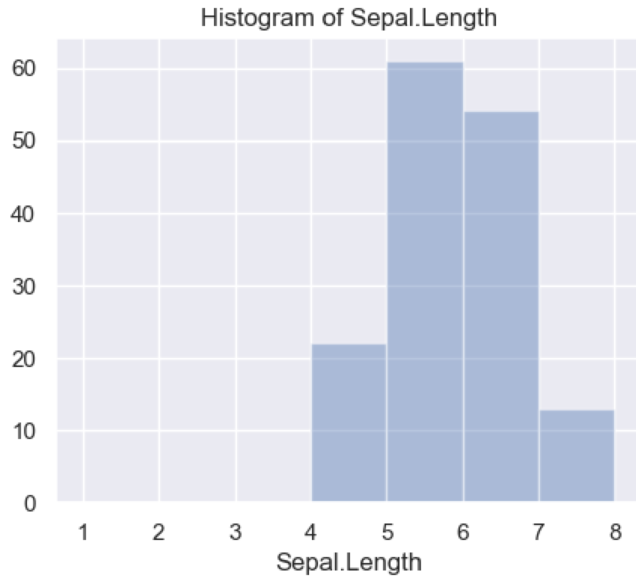
	Training Accuracy (in %)	Testing Accuracy (in %)
Logistic Regression	75	47
K-Nearest Neighbour (KNN)	95	100
Support Vector Machine (SVM)	91	80

2) On the iris dataset, perform KNN algorithm and discuss result

```
import pandas as pd
iris = pd.read_csv("iris.csv")
import matplotlib.pyplot as plt # mostly used for visualization purposes
import numpy as np
import seaborn as sns
sns.distplot(iris['Sepal.Length'], kde=False,label='Sepal.Length')
sns.distplot(iris['Sepal.Width'], kde=False,label='Sepal.Width')
sns.distplot(iris['Petal.Length'], kde=False,label='Petal.Length')
sns.distplot(iris['Petal.Width'], kde=False,label='Petal.Width')
plt.legend()
```



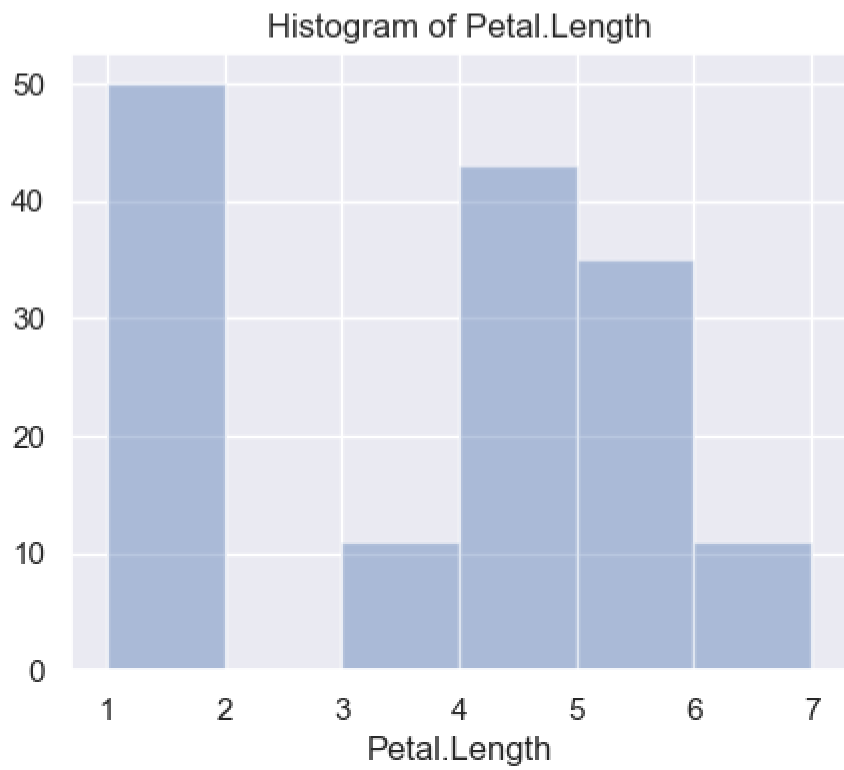

```
# 'Sepal.Length'
bins = [1,2,3,4,5,6,7,8]
plt.figure(figsize=(5,4))
sns.set() # light color background
sns.distplot(iris["Sepal.Length"],bins = bins, kde=False)
plt.xticks(bins) # x-axis (1-8)
plt.title("Histogram of Sepal.Length")
plt.show()
iris["Sepal.Length"].value_counts()
```



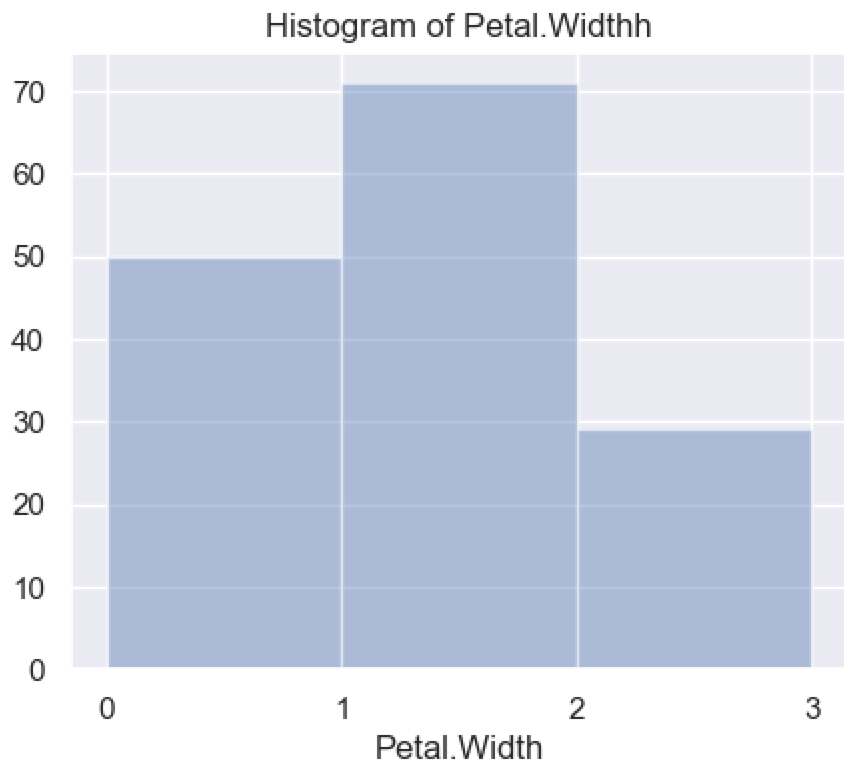

```
# 'Sepal.Width'
bins = [1,2,3,4,5]
plt.figure(figsize=(5,4))
sns.set() # light color background
sns.distplot(iris["Sepal.Width"],bins = bins, kde=False)
plt.xticks(bins) # x-axis (1-8)
plt.title("Histogram of Sepal.Width")
plt.show()
iris["Sepal.Width"].value_counts()
```


```
# 'Petal.Length'
bins = [1,2,3,4,5,6,7]
```

```
plt.figure(figsize=(5,4))
sns.set() # light color background
sns.distplot(iris["Petal.Length"],bins = bins, kde=False)
plt.xticks(bins) # x-axis (1-8)
plt.title("Histogram of Petal.Length")
plt.show()
iris["Petal.Length"].value_counts()
```




```
#'Petal.Width'
bins = [0,1,2,3]
plt.figure(figsize=(5,4))
sns.set() # light color background
sns.distplot(iris["Petal.Width"],bins = bins, kde=False)
plt.xticks(bins) # x-axis (1-8)
plt.title("Histogram of Petal.Widthh")
plt.show()
iris["Petal.Width"].value_counts()
```




```
#preparing data with scaling
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
feature_names = ['Sepal.Length', 'Sepal.Width', 'Petal.Length', 'Petal.Width']
x=iris[feature_names]
y=iris['Species']
x_train, x_test, y_train, y_test = train_test_split(x,y, random_state=0)
print(x_train[:3]) # to check output
scaler = MinMaxScaler()
x_train=scaler.fit_transform(x_train)
x_test= scaler.transform(x_test)
print("\nAfter scaling\n")
print(x_train[:3]) # to check output
```

```
     Sepal.Length  Sepal.Width  Petal.Length  Petal.Width
61            5.9          3.0           4.2          1.5
92            5.8          2.6           4.0          1.2
112           6.8          3.0           5.5          2.1

After scaling

[[0.44444444 0.41666667 0.53448276 0.58333333]
 [0.41666667 0.25       0.5        0.45833333]
 [0.69444444 0.41666667 0.75862069 0.83333333]]
```

```
from sklearn.neighbors import KNeighborsClassifier
# KNN method
knn = KNeighborsClassifier()
knn.fit(x_train, y_train)
#print score of train data
print('Accuracy of KNN classifier on training set:{:.2f}'
   .format(knn.score(x_train, y_train)))
#print score of test data
print('Accuracy of KNN Classifier on test set:{:.2f}'
   .format(knn.score(x_test, y_test)))
```

```
Accuracy of KNN classifier on training set:0.96
Accuracy of KNN Classifier on test set:0.97
```

3) implement apriori algorithm on online retail datset ans discuss result

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib as mlp
import matplotlib.pyplot as plt
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
transaction_df= pd.read_csv("../input/online-retail-ii-uci/online_retail_II.csv")
transaction_df

transaction_df = transaction_df[transaction_df.Country=='France']
transaction_filtered = transaction_df[['Invoice','Description','Quantity']].copy()
transaction_filtered

transaction_filtered.sort_values(by='Quantity', ascending=True)
transaction_filtered = transaction_filtered[transaction_filtered.Quantity > 0 ]

transaction_filtered.sort_values(by='Quantity', ascending=True)
transaction_filtered['Quantity']= [1]*len(transaction_filtered)
invoice = list(transaction_filtered.Invoice)

index_no = [invoice[index] for index in np.arange(len(invoice)) if not
invoice[index].isnumeric()]
transaction_filtered[transaction_filtered['Invoice'].isin(index_no)]
```

Out[10]:

Invoice	Description	Quantity

```
transaction_filtered= transaction_filtered[~transaction_filtered['Invoice'].isin(index_no)]
invoice = list(transaction_filtered.Invoice)

index_no = [index for index in np.arange(len(invoice)) if not invoice[index].isnumeric()]
transaction_filtered.iloc[index_no,:]
```

```
for invoice in list(temp_df.Invoice):
   if len(transaction_filtered[transaction_filtered.Invoice == invoice]) > 1:
      print((str)(invoice))
      temp = transaction_filtered[transaction_filtered.Invoice == 
invoice].groupby(['Invoice']).agg({'Description':lambda x: list(x)})
      if len(list(set(temp)))>0 :
         print(temp)
transaction_filtered.dropna(axis=0, inplace=True)

def return_one(x):
   return 1

table = pd.pivot_table(transaction_filtered, values='Quantity', index=['Invoice'],
               columns=['Description'], aggfunc=return_one, fill_value=0)
table
```

Out[19]:

Description	50'S CHRISTMAS GIFT BAG LARGE	DOLLY GIRL BEAKER	FLAMINGO LIGHTS	I LOVE LONDON MINI BACKPACK	LARGE SKULL WINDMILL	NINE DRAWER OFFICE TIDY	RED/WHITE DOT MINI CASES	SET 2 TEA TOWELS I LOVE LONDON	SPACEBOY BABY GIFT SET	TRELLIS COAT RACK	...	YELLOW RED FLOWER PIGGY BANK	YELLOW SHARK HELICOPTER
Invoice													
489439	0	0	0	0	0	0	0	0	0	0	...	0	0
489557	0	0	0	0	0	0	0	0	0	0	...	0	0
489883	0	0	0	0	0	0	1	0	0	0	...	0	0
490139	0	0	0	0	0	0	0	0	0	0	...	0	0
490152	0	0	0	0	0	0	1	0	0	0	...	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
580986	0	0	0	0	0	0	0	0	0	0	...	0	0
581001	0	0	0	0	0	0	0	0	0	0	...	0	0
581171	0	0	0	0	0	0	0	0	0	0	...	0	0
581279	0	0	0	0	0	0	0	0	0	0	...	0	0
581587	0	0	0	0	0	0	0	0	0	0	...	0	0

622 rows × 2211 columns

```
frequent_itemsets = apriori(table, min_support=0.01, use_colnames=True)
frequent_itemsets
```

Out[20]:

	support	itemsets
0	0.014469	(DOLLY GIRL BEAKER)
1	0.027331	(RED/WHITE DOT MINI CASES)
2	0.025723	(SET 2 TEA TOWELS I LOVE LONDON)
3	0.025723	(SPACEBOY BABY GIFT SET)
4	0.020900	(10 COLOUR SPACEBOY PEN)
...	...	...
8174	0.011254	(SET/6 RED SPOTTY PAPER PLATES, POSTAGE, PACK ...
8175	0.017685	(SET/6 RED SPOTTY PAPER PLATES, POSTAGE, PACK ...
8176	0.011254	(SET/6 RED SPOTTY PAPER PLATES, PACK OF 20 SKU...
8177	0.011254	(SET/6 RED SPOTTY PAPER PLATES, POSTAGE, PACK ...
8178	0.011254	(SET/6 RED SPOTTY PAPER PLATES, POSTAGE, SET O...

8179 rows × 2 columns

```
rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1)
rules
```

Out[21]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(DOLLY GIRL BEAKER)	(DOLLY GIRL CHILDRENS BOWL)	0.014469	0.028939	0.011254	0.777778	26.876543	0.010835	4.369775
1	(DOLLY GIRL CHILDRENS BOWL)	(DOLLY GIRL BEAKER)	0.028939	0.014469	0.011254	0.388889	26.876543	0.010835	1.612686
2	(DOLLY GIRL BEAKER)	(POSTAGE)	0.014469	0.749196	0.011254	0.777778	1.038150	0.000414	1.128617
3	(POSTAGE)	(DOLLY GIRL BEAKER)	0.749196	0.014469	0.011254	0.015021	1.038150	0.000414	1.000560
4	(DOLLY GIRL BEAKER)	(SPACEBOY CHILDRENS BOWL)	0.014469	0.032154	0.011254	0.777778	24.188889	0.010789	4.355305
...	...	...	...	...	...	...	...	...	...
68519	(SET OF 9 BLACK SKULL BALLOONS)	(SET/6 RED SPOTTY PAPER PLATES, POSTAGE, SET/2...	0.057878	0.019293	0.011254	0.194444	10.078704	0.010137	1.217430
68520	(SET/20 RED RETROSPOT PAPER NAPKINS)	(SET/6 RED SPOTTY PAPER PLATES, POSTAGE, SET O...	0.094855	0.016077	0.011254	0.118644	7.379661	0.009729	1.116374
68521	(PACK OF 6 SKULL PAPER PLATES)	(SET/6 RED SPOTTY PAPER PLATES, POSTAGE, SET O...	0.048232	0.011254	0.011254	0.233333	20.733333	0.010711	1.289669
68522	(PACK OF 6 SKULL PAPER CUPS)	(SET/6 RED SPOTTY PAPER PLATES, POSTAGE, SET O...	0.057878	0.011254	0.011254	0.194444	17.277778	0.010603	1.227409
68523	(SET/6 RED SPOTTY PAPER CUPS)	(SET/6 RED SPOTTY PAPER PLATES, POSTAGE, SET O...	0.136656	0.011254	0.011254	0.082353	7.317647	0.009716	1.077480

68524 rows × 9 columns

```
rules.sort_values(by=['support','confidence'], ascending=False)
```

In [22]:
```
rules.sort_values(by=['support','confidence'], ascending=False)
```

Out[22]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
3654	(RED TOADSTOOL LED NIGHT LIGHT)	(POSTAGE)	0.212219	0.749196	0.184887	0.871212	1.162863	0.025894	1.947418
3655	(POSTAGE)	(RED TOADSTOOL LED NIGHT LIGHT)	0.749196	0.212219	0.184887	0.246781	1.162863	0.025894	1.045886
3684	(ROUND SNACK BOXES SET OF4 WOODLAND)	(POSTAGE)	0.173633	0.749196	0.157556	0.907407	1.211175	0.027471	2.708682
3685	(POSTAGE)	(ROUND SNACK BOXES SET OF4 WOODLAND)	0.749196	0.173633	0.157556	0.210300	1.211175	0.027471	1.046432
3258	(PLASTERS IN TIN CIRCUS PARADE)	(POSTAGE)	0.167203	0.749196	0.136656	0.817308	1.090913	0.011388	1.372821
...	...	...	...	...	...	...	...	...	...
67513	(POSTAGE)	(PLASTERS IN TIN STRONGMAN, STRAWBERRY LUNCH B...	0.749196	0.011254	0.011254	0.015021	1.334764	0.002823	1.003825
67636	(POSTAGE)	(RED RETROSPOT MINI CASES, RED TOADSTOOL LED N...	0.749196	0.011254	0.011254	0.015021	1.334764	0.002823	1.003825
68014	(POSTAGE)	(SET/6 RED SPOTTY PAPER PLATES, PACK OF 20 SKU...	0.749196	0.011254	0.011254	0.015021	1.334764	0.002823	1.003825
68392	(POSTAGE)	(SET/6 RED SPOTTY PAPER PLATES, PACK OF 20 SKU...	0.749196	0.011254	0.011254	0.015021	1.334764	0.002823	1.003825
68518	(POSTAGE)	(SET/6 RED SPOTTY PAPER PLATES, SET OF 9 BLACK...	0.749196	0.012862	0.011254	0.015021	1.167918	0.001618	1.002193

68524 rows × 9 columns

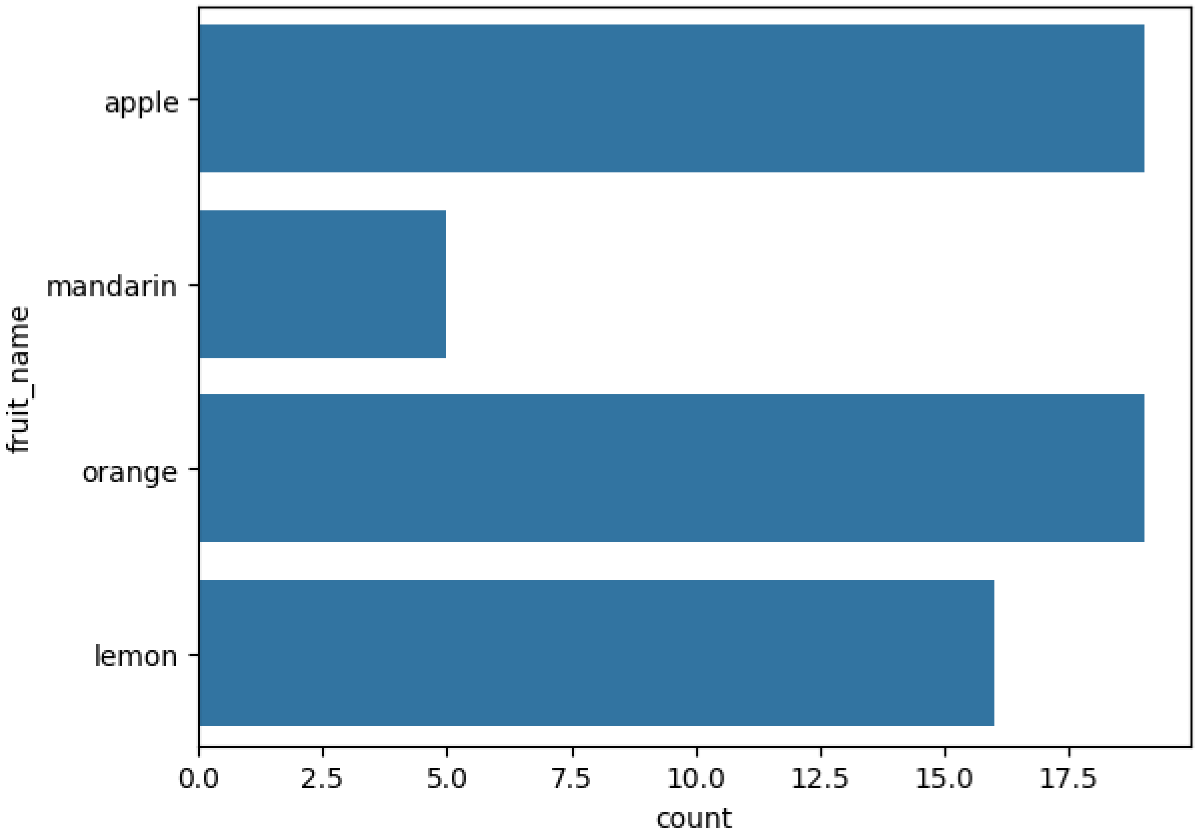
4)Implement Naïve Bayes Classifier and K-Nearest Neighbor Classifier on Data set of your choice. Test and Compare for Accuracy and Precision.

```
import pandas as pd   # to load dataset
import matplotlib.pyplot as plt
import seaborn as sns
import pylab as pl
from sklearn.model_selection import train_test_split   # for splitting dataset
from sklearn.preprocessing import MinMaxScaler  # for scaling
from sklearn.linear_model import LogisticRegression   # machine learning lib/model, # get accuracy by Logistic regression
from sklearn.tree import DecisionTreeClassifier  # get accuracy by Decision Tree classifier
from sklearn.neighbors import KNeighborsClassifier  # get accuracy by KNN classifier
from sklearn.naive_bayes import GaussianNB # get accuracy by GNB classifier
df=pd.read_csv('fruit_data.csv')
df.shape
df.describe()
```

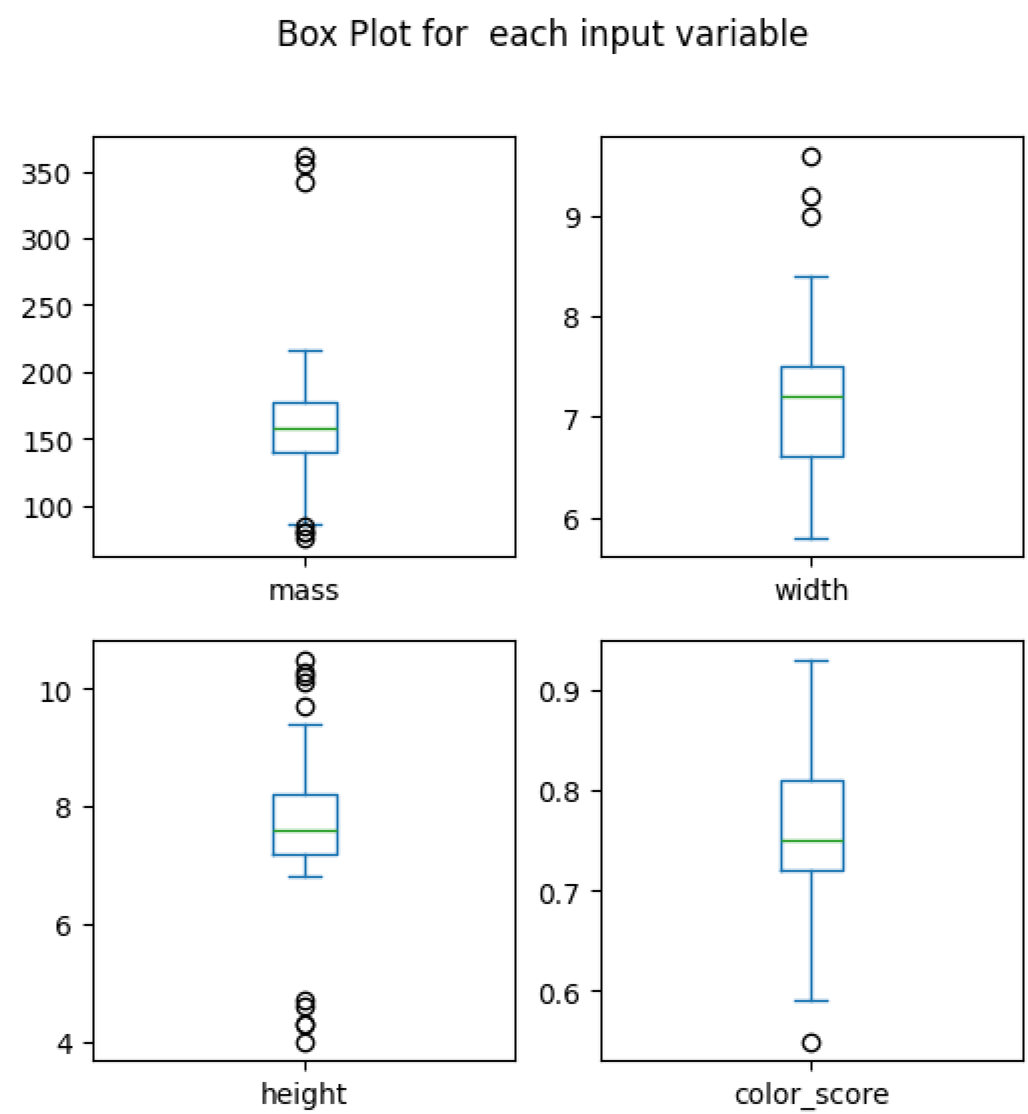
Out[13]:

	fruit_label	mass	width	height	color_score
count	59.000000	59.000000	59.000000	59.000000	59.000000
mean	2.542373	163.118644	7.105085	7.693220	0.762881
std	1.208048	55.018832	0.816938	1.361017	0.076857
min	1.000000	76.000000	5.800000	4.000000	0.550000
25%	1.000000	140.000000	6.600000	7.200000	0.720000
50%	3.000000	158.000000	7.200000	7.600000	0.750000
75%	4.000000	177.000000	7.500000	8.200000	0.810000
max	4.000000	362.000000	9.600000	10.500000	0.930000

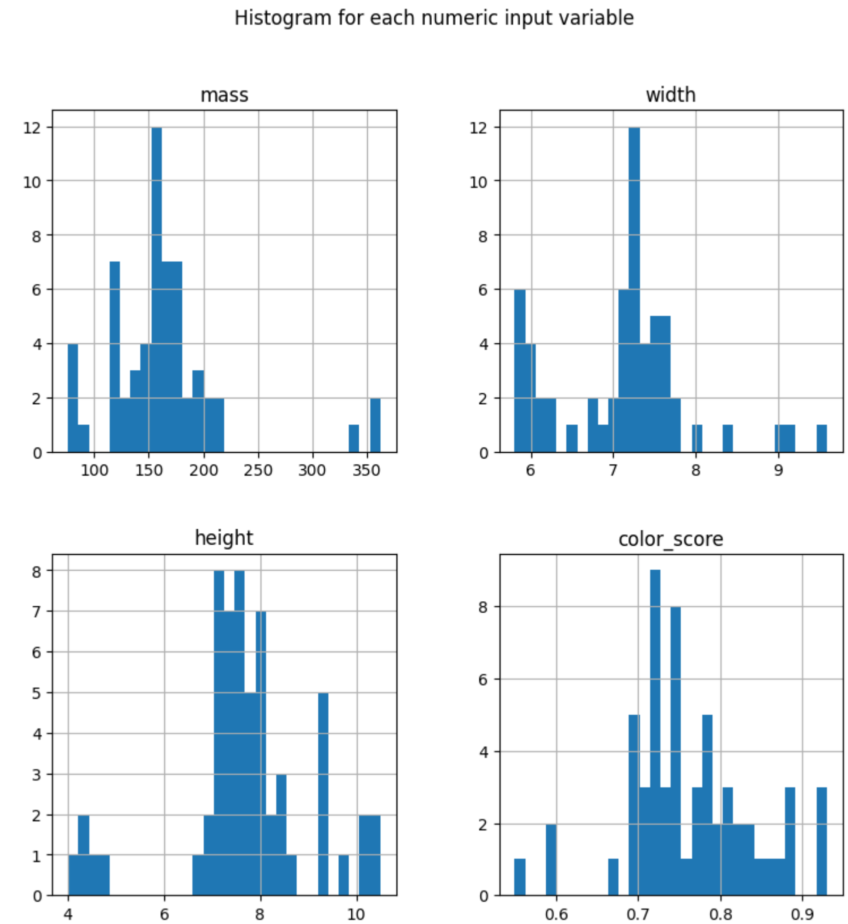
```
print(df['fruit_name'].unique()) # unique fruits name
print(df['fruit_subtype'].unique()) # unique fruit subtype
sns.countplot(df['fruit_name'],label='Count') # count plot
plt.show()
```



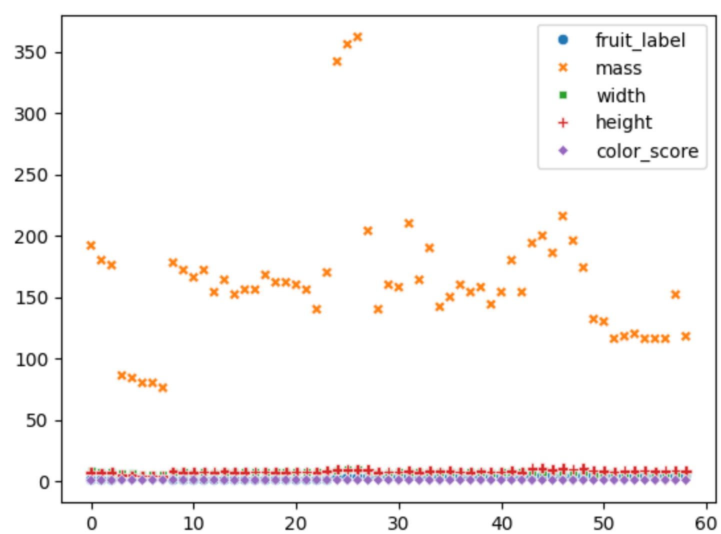
df.drop('fruit_label',axis=1).plot(kind='box', subplots=True, layout=(2,2), sharex=False, sharey=False, figsize=(6,6), title='Box Plot for each input variable')
plt.savefig('fruits_box')
plt.show()




```
import pylab as pl
df.drop('fruit_label', axis=1).hist(bins=30, figsize=(9,9))
pl.suptitle("Histogram for each numeric input variable")
plt.savefig('fruits_hist')
```




```
#scaterplot
sns.scatterplot(data=df)
```




```
#preparing data with scaling
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
feature_names = ['mass', 'width', 'height', 'color_score']
```

```
x=df[feature_names]
y=df['fruit_label']
x_train, x_test, y_train, y_test = train_test_split(x,y, random_state=0)
print(x_train[:3]) # to check output
scaler = MinMaxScaler()
x_train=scaler.fit_transform(x_train)
x_test= scaler.transform(x_test)
print("\nAfter scaling\n")
print(x_train[:3]) # to check output
```

```
    mass  width  height  color_score
42   154    7.2     7.2         0.82
48   174    7.3    10.1         0.72
7     76    5.8     4.0         0.81

After scaling

[[0.27857143 0.41176471 0.49230769 0.72972973]
 [0.35       0.44117647 0.93846154 0.45945946]
 [0.         0.         0.         0.7027027 ]]
```

```
from sklearn.neighbors import KNeighborsClassifier
# KNN method
knn = KNeighborsClassifier()
knn.fit(x_train, y_train)
#print score of train data
print('Accuracy of KNN classifier on training set:{:.2f}'
    .format(knn.score(x_train, y_train)))
#print score of test data
print('Accuracy of KNN Classifier on test set:{:.2f}'
    .format(knn.score(x_test, y_test)))
```

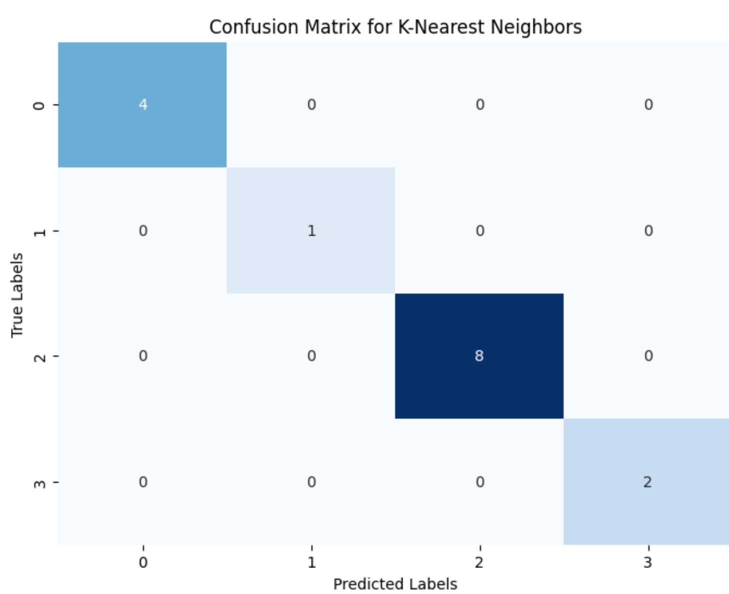
```
Accuracy of KNN classifier on training set:0.95
Accuracy of KNN Classifier on test set:1.00
```

```
from sklearn.naive_bayes import GaussianNB
# Gaussian Naive bayes
gnb = GaussianNB()
gnb.fit(x_train, y_train)
#print score of train data
print('Accuracy of GNB classifier on training set:{:.2f}'
```

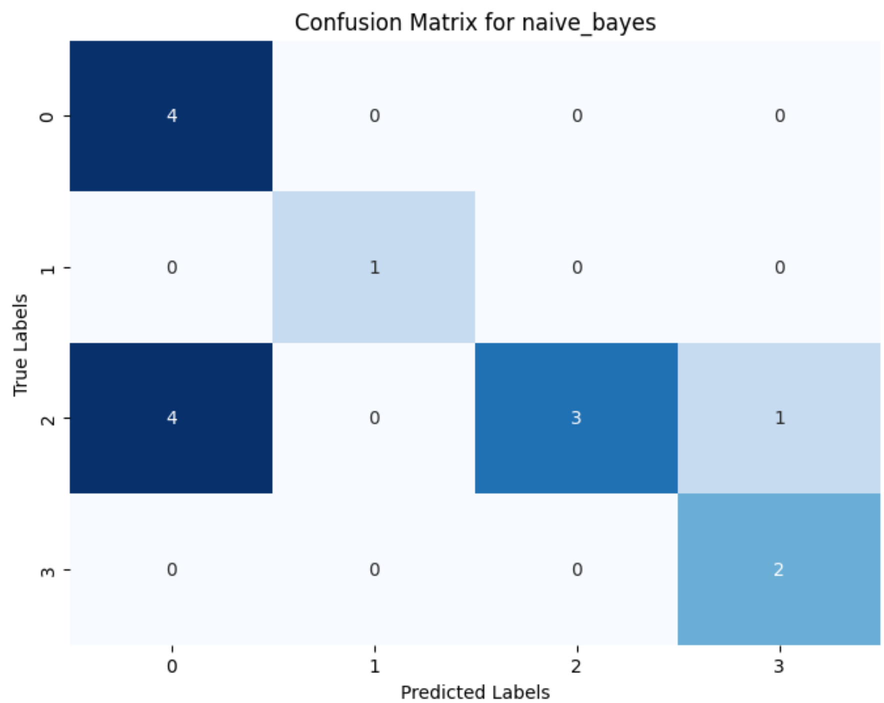
```
    .format(gnb.score(x_train, y_train)))
#print score of test data
print('Accuracy of GNB Classifier on test set:{:.2f}'
    .format(gnb.score(x_test, y_test)))
```

```
Accuracy of GNB classifier on training set:0.86
Accuracy of GNB Classifier on test set:0.67
```

```
#confusion matrix for KNN
import sklearn
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

# Assuming you have already trained your classification models, e.g., K-Nearest Neighbors (knn)
y_pred_knn = knn.predict(x_test)  # Make predictions on the test data
# Create the confusion matrix for K-Nearest Neighbors
confusion_knn = confusion_matrix(y_test, y_pred_knn)
# Create a heatmap of the confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(confusion_knn, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix for K-Nearest Neighbors')
plt.show()
```



```
#confusion matrix for naive bayes
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
# Assuming you have already trained your classification models
y_pred_gnb = gnb.predict(x_test)  # Make predictions on the test data
# Create the confusion matrix
confusion_gnb = confusion_matrix(y_test, y_pred_gnb)
# Create a heatmap of the confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(confusion_gnb, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix for naive_bayes')
plt.show()
```



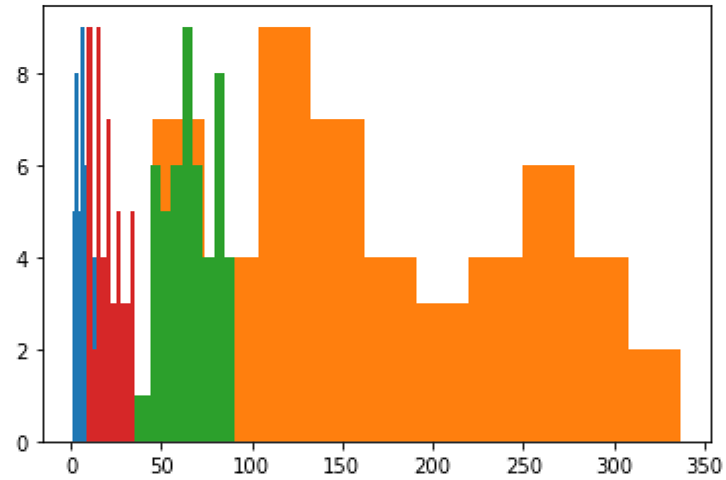
5) Implement K-means Clustering on a proper dataset of your choice¶

```
import pandas as pd              # for Data Manipulation
import matplotlib.pyplot as plt   # for Visualization
import numpy as np               #for Mathematical calculations
import seaborn as sns            #for Advanced visualizations
crime = pd.read_csv("crime_data.csv")
crime.head()
```

Out[2]:

	Unnamed: 0	Murder	Assault	UrbanPop	Rape
0	Alabama	13.2	236	58	21.2
1	Alaska	10.0	263	48	44.5
2	Arizona	8.1	294	80	31.0
3	Arkansas	8.8	190	50	19.5
4	California	9.0	276	91	40.6

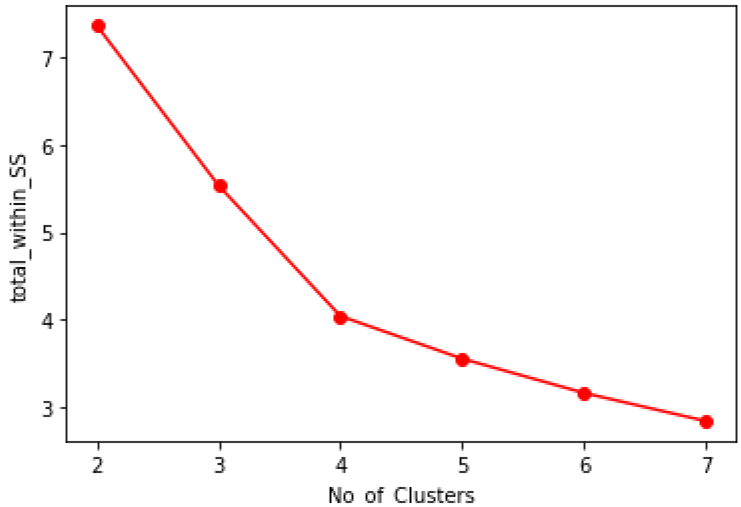
```
# We see the columns in the dataset
crime['State'] = crime.iloc[:,0]
crime = crime.iloc[:, [5,1,2,3,4]]
def norm_func(i):
   x = (i - i.min())       / (i.max() - i.min())
   return (x)

# Normalized data frame (considering the numerical part of data)
df_norm = norm_func(crime.iloc[:,1:])

###########################Univariate, Bivariate################
plt.hist(crime["Murder"])   #Univariate
plt.hist(crime["Assault"])
plt.hist(crime["UrbanPop"])
plt.hist(crime["Rape"])
crime.skew(axis = 0, skipna = True)
crime.kurtosis(axis = 0, skipna = True)
```




```
# calculating TWSS - Total within SS using different cluster range
from sklearn.cluster import KMeans
TWSS = []
k = list(range(2, 8))
for i in k:
    kmeans = KMeans(n_clusters = i)
    kmeans.fit(df_norm)
    TWSS.append(kmeans.inertia_)
TWSS
```

```
Out[16]: [7.358376498536079,
          5.532071995078602,
          4.0407678952238815,
          3.5539811127025747,
          3.1628651131109455,
          2.8417637970747243]
```

```
# Plotting the Scree plot using the TWSS from above defined function
plt.plot(k, TWSS, 'ro-');plt.xlabel("No_of_Clusters");plt.ylabel("total_within_SS")
```




```
model = KMeans(n_clusters = 4)
model.fit(df_norm)
model.labels_ # getting the labels of clusters assigned to each row
crime['clust'] = mb # creating a  new column and assigning it to new column
crime = crime.iloc[:,[5,0,1,2,3,4]]
crime.head()
crime.iloc[:, 1:6].groupby(crime.clust).mean()
```

Out[26]:

clust	Murder	Assault	UrbanPop	Rape
0	3.600000	78.538462	52.076923	12.446154
1	10.815385	257.384615	76.000000	30.930769
2	13.937500	243.625000	53.750000	21.412500
3	5.656250	138.875000	73.875000	18.843750

6) Design and implement SVM for classification with the proper dataset of your choice commenton design and implementation for linearly non sepearble datsset

```
#importing required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

train= pd.read_csv("SalaryData_Train.csv")
test= pd.read_csv("SalaryData_Test.csv")

from sklearn.preprocessing import LabelEncoder
lb = LabelEncoder()
train.education = lb.fit_transform(train.education)
test.education = lb.fit_transform(test.education)

train=
pd.get_dummies(train,columns=["workclass","maritalstatus","occupation","relationship
","race","sex","native"])
test =
pd.get_dummies(test,columns=["workclass","maritalstatus","occupation","relationship"
,"race","sex","native"])
train.Salary.value_counts()
```

```
Out[17]:  <=50K    22653
           >50K     7508
          Name: Salary, dtype: int64
```

```
x_train = train.drop("Salary",axis=1)
x_test = test.drop("Salary",axis = 1)
y_train = train.Salary
y_test = test.Salary

#Linear model
from sklearn.svm import SVC
model1 = SVC(kernel="linear",max_iter=100000)
model1.fit(x_train,y_train)
test_pred = model1.predict(x_test)
```

```
linear_accuracy = np.mean(y_test == test_pred)
linear_accuracy
```

```
Out[56]: 0.2353253652058433
```

```
# rgf Model
model2 = SVC(kernel="rbf",max_iter=150000)
model2.fit(x_train,y_train)
rbf_pred=model2.predict(x_test)
rbf_accuracy = np.mean(y_test == rbf_pred)
rbf_accuracy
```

```
Out[54]: 0.7964143426294821
```

```
#Poly Model
model3 = SVC(kernel="poly",max_iter=100000)
model3.fit(x_train,y_train)
poly_pred = model3.predict(x_test)
poly_pred = model3.predict(x_test)
poly_accuracy = np.mean(y_test == poly_pred)
poly_accuracy
```

```
Out[43]: 0.7795484727755644
```

```
# Sigmoid Model
model4 = SVC(kernel="sigmoid",max_iter=100000)
model4.fit(x_train,y_train)
sigmoid_pred=model4.predict(x_test)
sig_accuracy = np.mean(y_test == sigmoid_pred)
sig_accuracy
```

```
Out[46]: 0.7567729083665339
```

results = pd.DataFrame({"linear_model": linear_accuracy,"rbf_model": rbf_accuracy,"poly_accuracy":poly_accuracy,"sigmoid_accuracy":sig_accuracy},index=[" Accuracy"])

Out[58]:

	linear_model	rbf_model	poly_accuracy	sigmoid_accuracy
Accuracy	0.235325	0.796414	0.779548	0.756773

7) Implement a basic not gate using perceptron

```
# importing Python library
import numpy as np
# define Unit Step Function
def unitStep(v):
    if v >= 0:
        return 1
    else:
        return 0
# design Perceptron Model
def perceptronModel(x, w, b):
    v = np.dot(w, x) + b
    y = unitStep(v)
    return y
# NOT Logic Function
# w = -1, b = 0.5
def NOT_logicFunction(x):
    w = -1
    b = 0.5
    return perceptronModel(x, w, b)
# testing the Perceptron Model
test1 = np.array(1)
test2 = np.array(0)
print("NOT({}) = {}".format(1, NOT_logicFunction(test1)))
print("NOT({}) = {}".format(0, NOT_logicFunction(test2)))
```

```
NOT(1) = 0
NOT(0) = 1
```