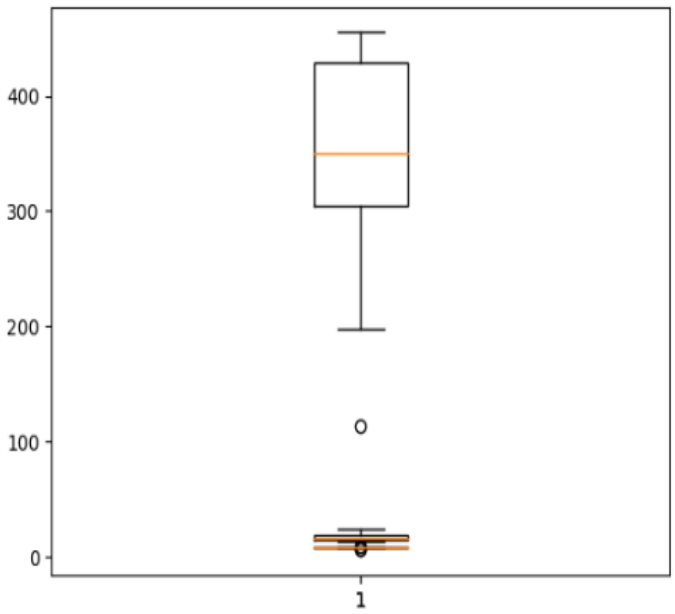
1.Structure Data in ML

```
import numpy as np

import statistics as s

import matplotlib.pyplot as mb

mpg=np.array([18,15,18,16,17,15,14,13,13,15,14,14,24,22,18,17])

cylinder=np.array([8,8,8,8,8,8,8,8,8,8,8,8,8,6,6])

displacement=np.array([307,350,318,304,302,429,454,440,455,390,383,340,400,455,113,198,9])

modelyear=np.array([70,70,70,70,70,70,70,70,70,70,70,70,70,70,70])

origin=np.array([1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1])

carname=np.array(['chevrolet','skylark','tornio','polo','k10','innova','vento','vento','k10','polo','polo','polo','cias','in
nova','wagonr'])

print("mean_msg",s.mean(mpg))

print("mean_cylinder",s.mean(cylinder))

print("mean_displacement",s.mean(displacement))

print("mean_modelyear",s.mean(modelyear))

print("median_mpg",s.median(mpg))

print("median_cylinder",s.median(cylinder))

print("median_displacement",s.median(displacement))

print("median_modelyear",s.median(modelyear))

print("mode_carname",s.mode(carname))

print("BOX PLOT_mpg")

mb.boxplot(mpg)

print("Box PLOT_cylinder")

mb.boxplot(cylinder)

print("BOX PLOT_displacement")

mb.boxplot(displacement)

print("Box PLOT_modelyear")

mb.boxplot(modelyear)

print("HISTORICAL_mpg")

mb.hist(mpg)
```

```
print("HISTORICAL_cylinder")
mb.hist(cylinder)
print("HISTORICAL_displacement")
mb.hist(displacement)
print("HISTORICAL_modelyear")
mb.hist(modelyear)
mb.scatter(mpg,cylinder)
import pandas as p
p.crosstab(mpg,cylinder,
        rownames=['mpg'],
        colnames=['cylinder'])
```

```
mean_msg 16
mean_cylinder 7
mean_displacement 332
mean_modelyear 70
median_mpg 15.5
median_cylinder 8.0
median_displacement 350
median_modelyear 70
mode_carname polo
BOX PLOT_mpg
Box PLOT_cylinder
BOX PLOT_displacement
```

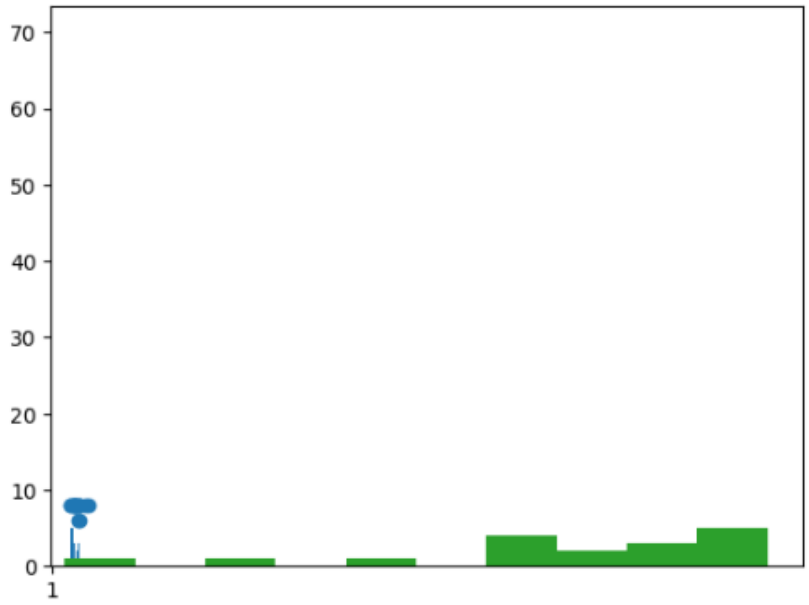
Out[1]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x1d707c46d10>,
  <matplotlib.lines.Line2D at 0x1d707c47990>],
 'caps': [<matplotlib.lines.Line2D at 0x1d707c545d0>,
  <matplotlib.lines.Line2D at 0x1d707c55110>],
 'boxes': [<matplotlib.lines.Line2D at 0x1d707c1c890>],
 'medians': [<matplotlib.lines.Line2D at 0x1d707c55c50>],
 'fliers': [<matplotlib.lines.Line2D at 0x1d707c565d0>],
 'means': []}
```



```
Box PLOT_modelyear
HISTORICAL_mpg
HISTORICAL_cylinder
HISTORICAL_displacement
HISTORICAL_modelyear
```

Out[2]:

cylinder	6	8
mpg		
13	0	2
14	0	3
15	0	3
16	0	1
17	1	1
18	1	2
22	0	1
24	0	1



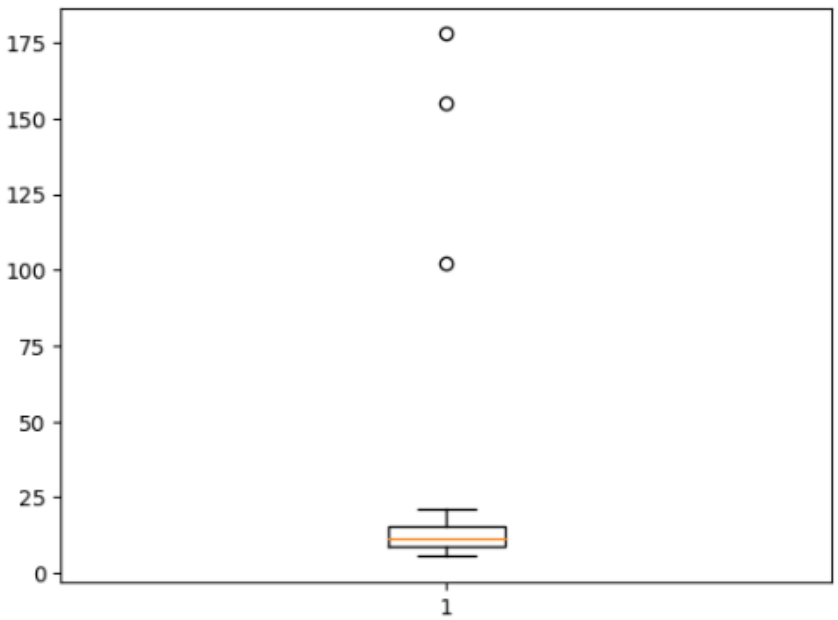
```
import numpy as np
import matplotlib.pyplot as mb
import pandas as pd
df=pd.read_csv("D:\\Merlin\\MACHINE LEARNING\\weather2.csv")
print(df.head())
print(df.info())
print(df.to_string())
df.describe()
print(df.shape)
print(df.isnull())
print(df.isnull().sum())
df.dropna(subset=['Summary'],inplace=True)
df.dropna(subset=['Precip Type'],inplace=True)
df.dropna (subset=['Apparent Temperature (C)'],inplace=True)
df.dropna(subset=['Wind Speed (km/h)'],inplace=True)
df.dropna(subset=['Pressure (millibars)'],inplace=True)
df.dropna(subset=['Daily Summary'],inplace=True)
print(df.isnull().sum())
import statistics as s
mode=s.mode(df.Temperatur)
print(mode)
df.fillna(value = {'Temperatur':mode}, inplace=True)
df.isnull().sum()
mb.boxplot(df['Temperatur'])
q1=df['Temperatur'].quantile(.25)
q3=df['Temperatur'].quantile(.75)
iqr=q3-q1
upper =(q3+1.5*iqr)
lower=(q1-1.5*iqr)
print(df.shape)
```

```
df1=df[(df.Temperatur < upper) & (df.Temperatur >lower)]

print(df1)

print(df1.shape)

mb.boxplot(df1['Temperatur'])
```

```
                 Formatted Date        Summary Precip Type  Temperatur  \
0  2006-04-01 00:00:00.000 +0200  Partly Cloudy        rain    9.472222
1  2006-04-01 01:00:00.000 +0200  Partly Cloudy        rain         NaN
2  2006-04-01 02:00:00.000 +0200  Mostly Cloudy         NaN    9.377778
3  2006-04-01 03:00:00.000 +0200  Partly Cloudy        rain    8.288889
4  2006-04-01 04:00:00.000 +0200  Mostly Cloudy        rain    8.755556

   Apparent Temperature (C)  Humidity  Wind Speed (km/h)  \
0                  7.388889      0.89            14.1197
1                  7.227778      0.86            14.2646
2                  9.377778      0.89             3.9284
3                       NaN      0.83            14.1036
4                  6.977778      0.83            11.0446

   Wind Bearing (degrees)  Visibility (km)  Loud Cover  Pressure (millibars)  \
0                     251          15.8263           0               1015.13
1                     259          15.8263           0               1015.63
2                     204          14.9569           0               1015.94
3                     269          15.8263           0               1016.41
4                     259          15.8263           0               1016.51

                    Daily Summary
0  Partly cloudy throughout the day.
1  Partly cloudy throughout the day.
2  Partly cloudy throughout the day.
3  Partly cloudy throughout the day.
4  Partly cloudy throughout the day.
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 99 entries, 0 to 98
Data columns (total 12 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Formatted Date            99 non-null     object
 1   Summary                   94 non-null     object
 2   Precip Type               94 non-null     object
 3   Temperatur                96 non-null     float64
 4   Apparent Temperature (C)  96 non-null     float64
 5   Humidity                  99 non-null     float64
 6   Wind Speed (km/h)         98 non-null     float64
 7   Wind Bearing (degrees)    99 non-null     int64
 8   Visibility (km)           99 non-null     float64
 9   Loud Cover                99 non-null     int64
 10  Pressure (millibars)      97 non-null     float64
 11  Daily Summary             91 non-null     object
dtypes: float64(6), int64(2), object(4)
memory usage: 9.4+ KB
None
    Formatted Date  Summary  Precip Type  Temperatur  \
0            False    False        False       False
1            False    False        False        True
2            False    False         True       False
3            False    False        False       False
4            False    False        False       False
..             ...      ...          ...         ...
94           False    False        False       False
95           False    False        False       False
96           False     True        False       False
97           False    False        False       False
98           False    False        False       False

    Apparent Temperature (C)  Humidity  Wind Speed (km/h)  \
0                      False     False              False
1                      False     False              False
2                      False     False              False
3                       True     False              False
4                      False     False              False
..                       ...       ...                ...
94                     False     False              False
95                     False     False              False
96                     False     False              False
97                     False     False              False
98                     False     False              False

    Wind Bearing (degrees)  Visibility (km)  Loud Cover  Pressure (millibars)  \
0                    False            False       False                 False
1                    False            False       False                 False
2                    False            False       False                 False
3                    False            False       False                 False
4                    False            False       False                 False
..                     ...              ...         ...                   ...
94                   False            False       False                 False
95                   False            False       False                 False
96                   False            False       False                 False
97                   False            False       False                 False
98                   False            False       False                 False
```

```
    Daily Summary
0           False
1           False
2           False
3           False
4           False
..            ...
94           True
95          False
96          False
97          False
98          False

[99 rows x 12 columns]
Formatted Date               0
Summary                      5
Precip Type                  5
Temperatur                   3
Apparent Temperature (C)     3
Humidity                     0
Wind Speed (km/h)            1
Wind Bearing (degrees)       0
Visibility (km)              0
Loud Cover                   0
Pressure (millibars)         2
Daily Summary                8
dtype: int64
Formatted Date               0
Summary                      0
Precip Type                  0
Temperatur                   3
Apparent Temperature (C)     0
Humidity                     0
Wind Speed (km/h)            0
Wind Bearing (degrees)       0
Visibility (km)              0
Loud Cover                   0
Pressure (millibars)         0
Daily Summary                0
dtype: int64
11.18333333
```

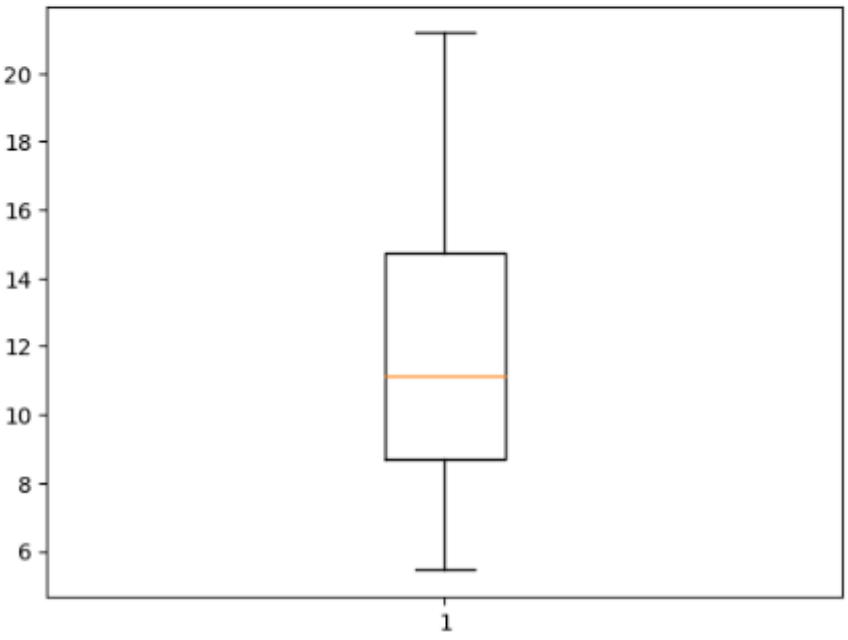
Out[9]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x1d70c05f3d0>,
  <matplotlib.lines.Line2D at 0x1d70c05fe50>],
 'caps': [<matplotlib.lines.Line2D at 0x1d70c0702d0>,
  <matplotlib.lines.Line2D at 0x1d70c070f10>],
 'boxes': [<matplotlib.lines.Line2D at 0x1d70c05ead0>],
 'medians': [<matplotlib.lines.Line2D at 0x1d70c0716d0>],
 'fliers': [<matplotlib.lines.Line2D at 0x1d70bf2b450>],
 'means': []}
```



```
(78, 12)
                  Formatted Date        Summary Precip Type  Temperatur  \
0   2006-04-01 00:00:00.000 +0200  Partly Cloudy        rain    9.472222
1   2006-04-01 01:00:00.000 +0200  Partly Cloudy        rain   11.183333
4   2006-04-01 04:00:00.000 +0200  Mostly Cloudy        rain    8.755556
5   2006-04-01 05:00:00.000 +0200  Partly Cloudy        rain    9.222222
6   2006-04-01 06:00:00.000 +0200  Partly Cloudy        rain    7.733333
..                            ...           ...         ...         ...
92  2006-04-12 20:00:00.000 +0200  Mostly Cloudy        rain    9.900000
93  2006-04-12 21:00:00.000 +0200       Overcast        rain    8.794444
95  2006-04-12 23:00:00.000 +0200       Overcast        rain    7.855556
97  2006-04-13 01:00:00.000 +0200       Overcast        rain    7.244444
98  2006-04-13 02:00:00.000 +0200  Partly Cloudy        rain    5.438889

    Apparent Temperature (C)  Humidity  Wind Speed (km/h)  \
0                   7.388889      0.89            14.1197
1                   7.227778      0.86            14.2646
4                   6.977778      0.83            11.0446
5                   7.111111      0.85            13.9587
6                   5.522222      0.95            12.3648
..                       ...       ...                ...
92                  7.716667      0.66            15.7297
93                  6.816667      0.71            12.3648
95                  6.122222      0.72             9.8049
97                  6.005556      0.75             7.1162
98                  5.438889      0.88             3.7191

    Wind Bearing (degrees)  Visibility (km)  Loud Cover  Pressure (millibars)  \
0                      251          15.8263           0               1015.13
1                      259          15.8263           0               1015.63
4                      259          15.8263           0               1016.51
5                      258          14.9569           0               1016.66
6                      259           9.9820           0               1016.72
..                     ...              ...         ...                   ...
92                     348          11.0285           0               1005.48
93                      20           9.9820           0               1005.89
95                      11          15.0052           0               1006.56
97                     309          15.8746           0               1007.37
98                     193           9.9820           0               1012.23
```

```
                                Daily Summary
0            Partly cloudy throughout the day.
1            Partly cloudy throughout the day.
4            Partly cloudy throughout the day.
5            Partly cloudy throughout the day.
6            Partly cloudy throughout the day.
..                                         ...
92  Foggy overnight and breezy in the morning.
93  Foggy overnight and breezy in the morning.
95  Foggy overnight and breezy in the morning.
97                Overcast throughout the day.
98                Overcast throughout the day.

[75 rows x 12 columns]
(75, 12)
```

Out[10]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x1d70bdaf5d0>,
  <matplotlib.lines.Line2D at 0x1d70b555390>],
 'caps': [<matplotlib.lines.Line2D at 0x1d70bd6c3d0>,
  <matplotlib.lines.Line2D at 0x1d70bd6f850>],
 'boxes': [<matplotlib.lines.Line2D at 0x1d70bdaed90>],
 'medians': [<matplotlib.lines.Line2D at 0x1d70bd6c850>],
 'fliers': [<matplotlib.lines.Line2D at 0x1d70b554ad0>],
 'means': []}
```



```
from pandas import read_csv
from numpy import set_printoptions
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_classif
filename ="D:\\Merlin\\MACHINE LEARNING\\pima-indians-diabetes (1).csv"
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
dataframe = read_csv(filename, names=names)
array = dataframe.values
X = array[:,0:8]
Y = array[:,8]
test = SelectKBest(score_func=f_classif, k=4)
fit = test.fit(X, Y)
set_printoptions(precision=3)
print(fit.scores_)
features = fit.transform(X)
print(features[0:5,:])
import urllib.request
import numpy
from pandas import read_csv
from sklearn.decomposition import PCA
url = "D:\\Merlin\\MACHINE LEARNING\\pima-indians-diabetes (1).csv"
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
dataframe = read_csv(url, names=names)
array = dataframe.values
X = array[:,0:8]
Y = array[:,8]
pca = PCA(n_components=3)
fit = pca.fit(X)
print("Explained Variance: %s" % fit.explained_variance_ratio_)
print(fit.components_)
```

```
[ 39.67  213.162   3.257   4.304  13.281  71.772  23.871  46.141]
[[  6.  148.   33.6  50. ]
 [  1.   85.   26.6  31. ]
 [  8.  183.   23.3  32. ]
 [  1.   89.   28.1  21. ]
 [  0.  137.   43.1  33. ]]
```

```
Explained Variance: [0.889 0.062 0.026]
[[-2.022e-03  9.781e-02  1.609e-02  6.076e-02  9.931e-01  1.401e-02
   5.372e-04 -3.565e-03]
 [-2.265e-02 -9.722e-01 -1.419e-01  5.786e-02  9.463e-02 -4.697e-02
  -8.168e-04 -1.402e-01]
 [-2.246e-02  1.434e-01 -9.225e-01 -3.070e-01  2.098e-02 -1.324e-01
  -6.400e-04 -1.255e-01]]
```

```
from sklearn.metrics import confusion_matrix

X_actual = [1, 1, 0, 1, 0, 0, 1, 0, 0, 1]

Y_predic = [1, 0, 1, 1, 1, 0, 1, 1, 1, 0]

results = confusion_matrix(X_actual, Y_predic)

print ('Confusion Matrix :')

print(results)

TP = results[0][0]

TN = results[0][1]

FP = results[1][0]

FN = results[1][1]

Total = TP + TN + FP + FN

error_rate = (FP + FN)/Total

print( "Error rate ",error_rate)

P_a = (TP + FP)/Total

P_Pr= (((TP + FP)/Total ) * ((TP + FN)/Total))+(((TN + FP)/Total ) * ((TN + FN)/Total))

kappa_value = (P_a - P_Pr)/(1-P_Pr)

print("Kappa value ",kappa_value)

sensitivity = TP/(TP+FP)

Specificity = TN/(TN + FP)

Precision = TP/(TP+FP)

Recall =TP/(TP + FP)

print("sensitivity",sensitivity)

print("Specificity",Specificity)

print("Precision ",Precision)

print("Recall ",Recall)

fmeasure = (2 * Precision * Recall)/(Precision + Recall)

print("F MEASURE", fmeasure)
```

```
Confusion Matrix :
[[1 4]
 [2 3]]
Error rate  0.5
Kappa value  -0.5217391304347827
sensitivity 0.3333333333333333
Specificity 0.6666666666666666
Precision  0.3333333333333333
Recall  0.3333333333333333
F MEASURE 0.3333333333333333
```

```
import pandas as pd
import numpy as np
import csv
from pgmpy.estimators import MaximumLikelihoodEstimator
from pgmpy.models import BayesianNetwork
from pgmpy.inference import VariableElimination
heartDisease = pd.read_csv("D:\\Merlin\\MACHINE LEARNING\\heart.csv")
heartDisease = heartDisease.replace('?', np.nan)
print('Sample instances from the dataset are given below:')
print(heartDisease.head())
print('\n Attributes and datatypes:')
print(heartDisease.dtypes)
model = BayesianNetwork([
    ('age', 'heartdisease'),
    ('gender', 'heartdisease'),
    ('cp', 'heartdisease'),
    ('exang', 'heartdisease'),
    ('heartdisease', 'restecg'),
    ('heartdisease', 'slope')
])
print('\nLearning CPD using Maximum likelihood estimators...')
model.fit(heartDisease, estimator=MaximumLikelihoodEstimator)
print('\nInferencing with Bayesian Network:')
HeartDiseasetest_infer = VariableElimination(model)
print('\n1. Probability of HeartDisease given evidence= exang:')
q1 = HeartDiseasetest_infer.query(variables=['heartdisease'], evidence={'exang': 1})
print(q1)
print('\n2. Probability of HeartDisease given evidence= cp:')
q2 = HeartDiseasetest_infer.query(variables=['heartdisease'], evidence={'cp': 2})
print(q2)
```

```
Sample instances from the dataset are given below:
   age  gender  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  \
0   63       1   1       145   233    1        2      150      0      2.3
1   67       1   4       160   286    0        2      108      1      1.5
2   67       1   4       120   229    0        2      129      1      2.6
3   41       0   2       130   204    0        2      172      0      1.4
4   56       1   2       120   236    0        0      178      0      0.8

   slope  ca  thal  heartdisease
0      3   0     6             0
1      2   3     3             2
2      2   2     7             1
3      1   0     3             0
4      1   0     3             0

 Attributes and datatypes:
age               int64
gender            int64
cp                int64
trestbps          int64
chol              int64
fbs               int64
restecg           int64
thalach           int64
exang             int64
oldpeak         float64
slope             int64
ca                int64
thal              int64
heartdisease      int64
dtype: object
```

```
Learning CPD using Maximum likelihood estimators...

Inferencing with Bayesian Network:

1. Probability of HeartDisease given evidence= exang:
+-----------------+---------------------+
| heartdisease    |   phi(heartdisease) |
+=================+=====================+
| heartdisease(0) |              0.2907 |
+-----------------+---------------------+
| heartdisease(1) |              0.3547 |
+-----------------+---------------------+
| heartdisease(2) |              0.3547 |
+-----------------+---------------------+

2. Probability of HeartDisease given evidence= cp:
+-----------------+---------------------+
| heartdisease    |   phi(heartdisease) |
+=================+=====================+
| heartdisease(0) |              0.4133 |
+-----------------+---------------------+
| heartdisease(1) |              0.2933 |
+-----------------+---------------------+
| heartdisease(2) |              0.2933 |
+-----------------+---------------------+
```

```
import pandas as pd
from sklearn import tree
from sklearn.preprocessing import LabelEncoder
from sklearn.naive_bayes import GaussianNB
data = pd.read_csv("D:\\Merlin\\MACHINE LEARNING\\tennisdata.csv")
print("The first 5 values of data is :\n",data.head())
X = data.iloc[:,:-1]
print("\nThe First 5 values of train data is\n",X.head())
y = data.iloc[:,-1]
print("\nThe first 5 values of Train output is\n",y.head())
le_outlook = LabelEncoder()
X.Outlook = le_outlook.fit_transform(X.Outlook)
le_Temperature = LabelEncoder()
X.Temperature = le_Temperature.fit_transform(X.Temperature)
le_Humidity = LabelEncoder()
X.Humidity = le_Humidity.fit_transform(X.Humidity)
le_Windy = LabelEncoder()
X.Windy = le_Windy.fit_transform(X.Windy)
print("\nNow the Train data is :\n",X.head())
le_PlayTennis = LabelEncoder()
y = le_PlayTennis.fit_transform(y)
print("\nNow the Train output is\n",y)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.20)
classifier = GaussianNB()
classifier.fit(X_train,y_train)
from sklearn.metrics import accuracy_score
print("Accuracy is:",accuracy_score(classifier.predict(X_test),y_test))
```

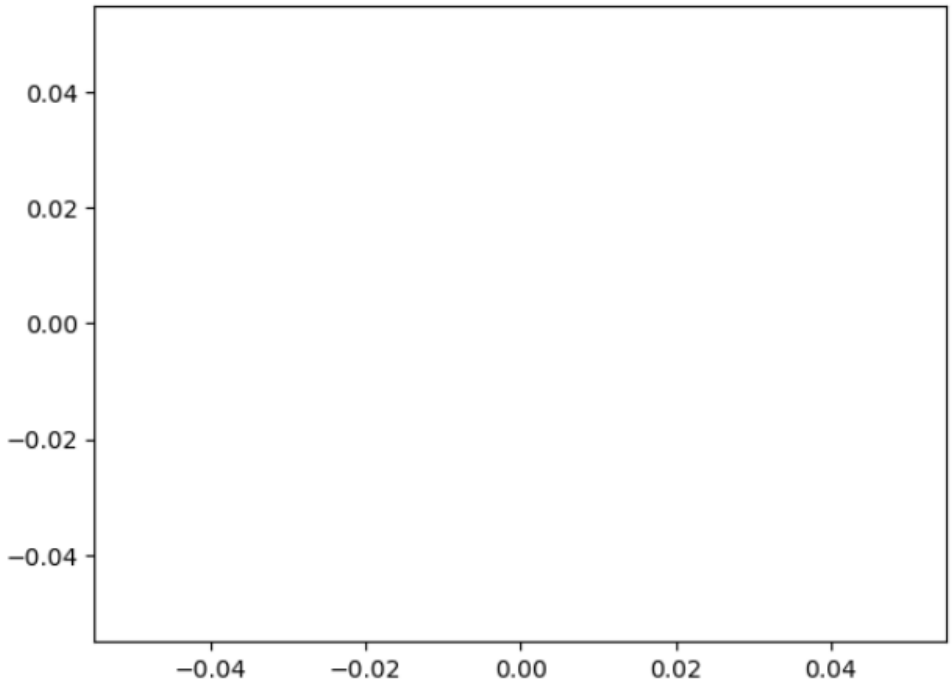
```
The first 5 values of data is :
     Outlook Temperature Humidity  Windy PlayTennis
0     Sunny         Hot     High  False         No
1     Sunny         Hot     High   True         No
2  Overcast         Hot     High  False        Yes
3     Rainy        Mild     High  False        Yes
4     Rainy        Cool   Normal  False        Yes

The First 5 values of train data is
     Outlook Temperature Humidity  Windy
0     Sunny         Hot     High  False
1     Sunny         Hot     High   True
2  Overcast         Hot     High  False
3     Rainy        Mild     High  False
4     Rainy        Cool   Normal  False

The first 5 values of Train output is
 0     No
1     No
2    Yes
3    Yes
4    Yes
Name: PlayTennis, dtype: object

Now the Train data is :
    Outlook  Temperature  Humidity  Windy
0        2            1         0      0
1        2            1         0      1
2        0            1         0      0
3        1            2         0      0
4        1            0         1      0

Now the Train output is
 [0 0 1 1 1 0 1 0 1 1 1 1 1 0]
Accuracy is: 0.6666666666666666
```

```
from sklearn.cluster import KMeans
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
data=pd.read_csv("D:\\Merlin\\MACHINE LEARNING\\em.csv")
df1=pd.DataFrame(data)
print(df1 )
f1 = df1['sepal.length'].values
f2 = df1 ['petal.length'].values
X=np.matrix(list(zip(f1,f2)))
plt.plot()
plt.xlim([0, 100])
plt.ylim([0, 50])
plt.title('Dataset')
plt.ylabel('sepal.length')
plt.xlabel('petal.length')
plt.scatter(f1,f2)
colors = ['b', 'g', 'r']
markers = ['o', 'v', 's']
X = np.asarray(list(zip(f1, f2)))
kmeans_model = KMeans(n_clusters=3).fit(X)
plt.plot()
for i, l in enumerate(kmeans_model.labels_):
    plt.plot(f1[i], f2[i], color=colors[l], marker=markers[1],ls='None')
plt.xlim([0, 10])
plt.ylim([0, 10])
plt.show()
```

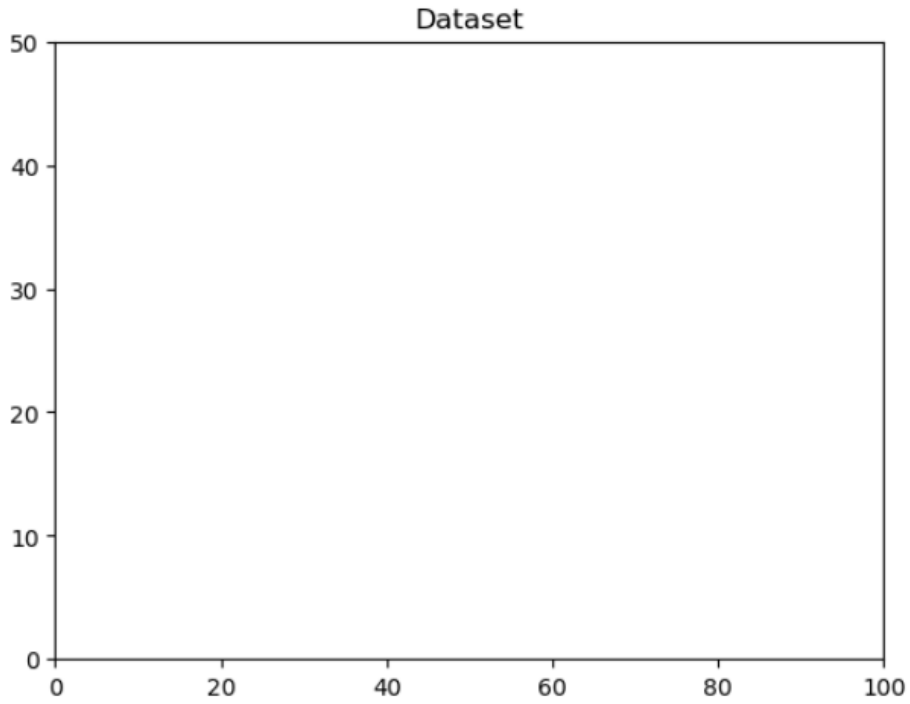
	sepal.length	sepal.width	petal.length	petal.width	variety
0	5.1	3.5	1.4	0.2	Setosa
1	4.9	3.0	1.4	0.2	Setosa
2	4.7	3.2	1.3	0.2	Setosa
3	4.6	3.1	1.5	0.2	Setosa
4	5.0	3.6	1.4	0.2	Setosa
..	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Virginica
146	6.3	2.5	5.0	1.9	Virginica
147	6.5	3.0	5.2	2.0	Virginica
148	6.2	3.4	5.4	2.3	Virginica
149	5.9	3.0	5.1	1.8	Virginica

[150 rows x 5 columns]

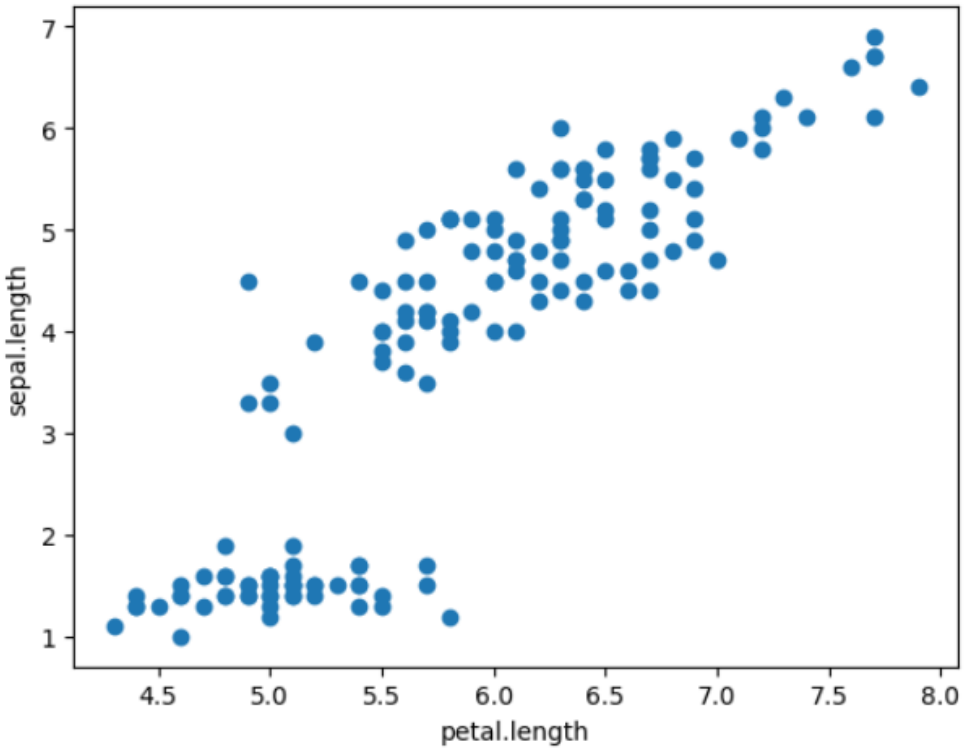
Out[8]: []



Out[9]: Text(0.5, 1.0, 'Dataset')

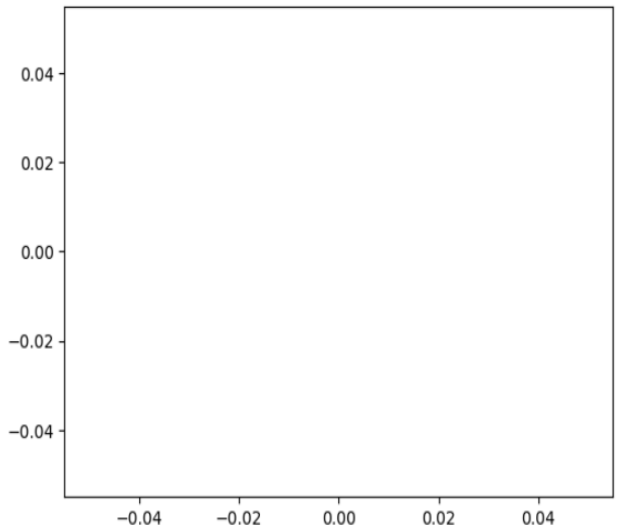


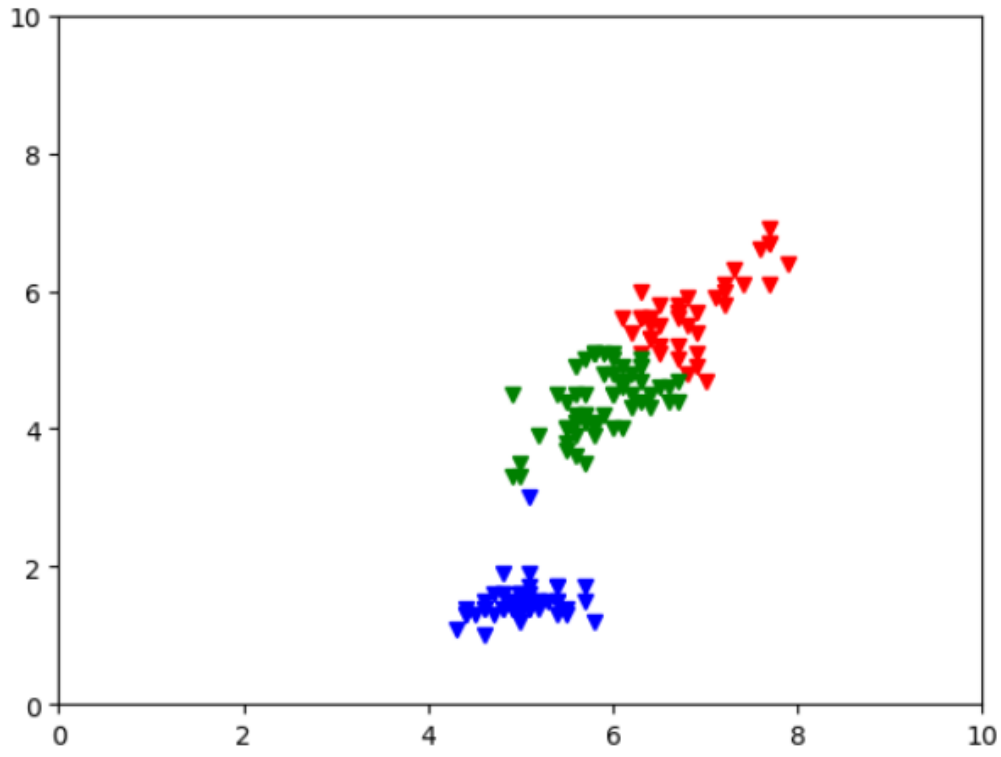
Out[10]: <matplotlib.collections.PathCollection at 0x12cc4d12f90>



```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will c
hange from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a memory leak
on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OM
P_NUM_THREADS=1.
  warnings.warn(
```

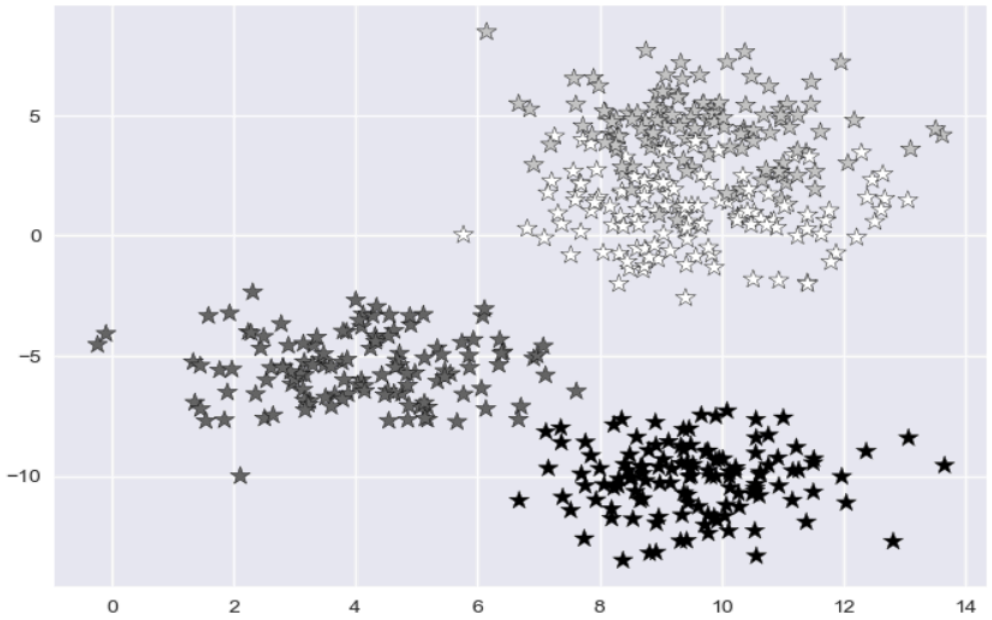
Out[44]: []





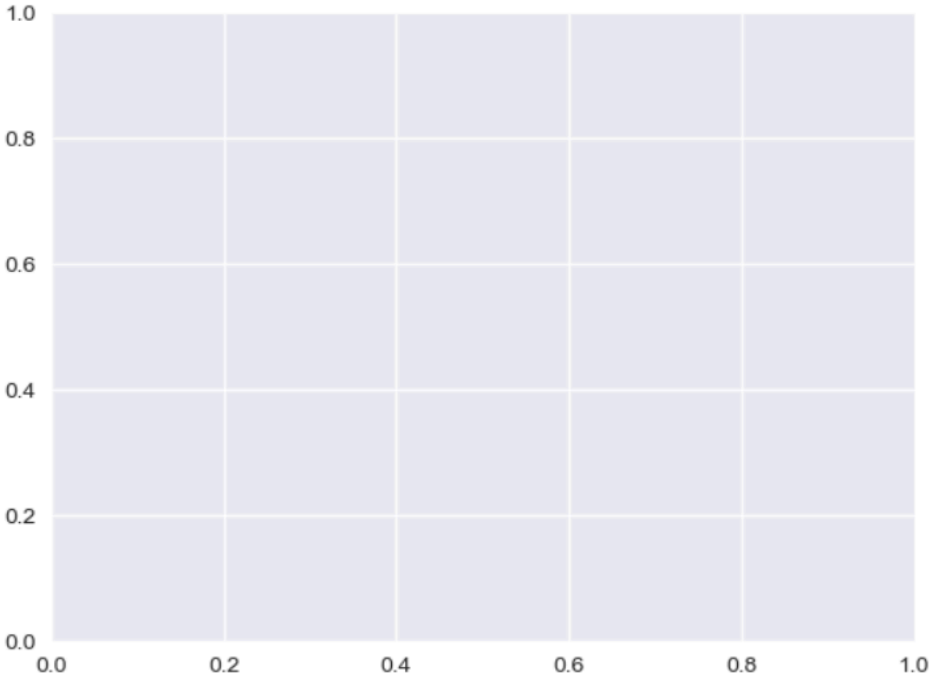
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
plt.style.use('seaborn')
X, y= make_blobs(n_samples = 500, n_features =2, centers = 4,cluster_std =
1.5, random_state = 4)
plt.figure(figsize = (10,10))
plt.show()
plt.scatter(X[:,0], X[:,1], c=y, marker='*',s=100,edgecolors='black')
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state = 0)
knn5 = KNeighborsClassifier(n_neighbors=5)
knnl = KNeighborsClassifier(n_neighbors=1)
knn5.fit(X_train, y_train)
knnl.fit(X_train, y_train)
y_pred_5= knn5.predict(X_test)
y_pred_1= knnl.predict(X_test)
from sklearn.metrics import accuracy_score
print("Accuracy with k=5", accuracy_score(y_test, y_pred_5)* 100)
print("Accuracy with k=1", accuracy_score(y_test, y_pred_1)* 100)
plt.figure(figsize = (15,5))
plt.subplot(1,2,1)
plt.scatter(X_test[:,0],X_test[:,1],c=y_pred_5,marker='*',s=100,edgecolors='bla
ck')
plt.title("Predicted values with k=5",fontsize=20)
plt.subplot(1,2,2)
plt.scatter(X_test[:,0],X_test[:,1],c=y_pred_1,marker='*',s=100,edgecolors='bla
ck')
plt.title("Predicted values with k=1",fontsize=20)
plt.show()
```

Out[13]: <matplotlib.collections.PathCollection at 0x1fe0038c650>

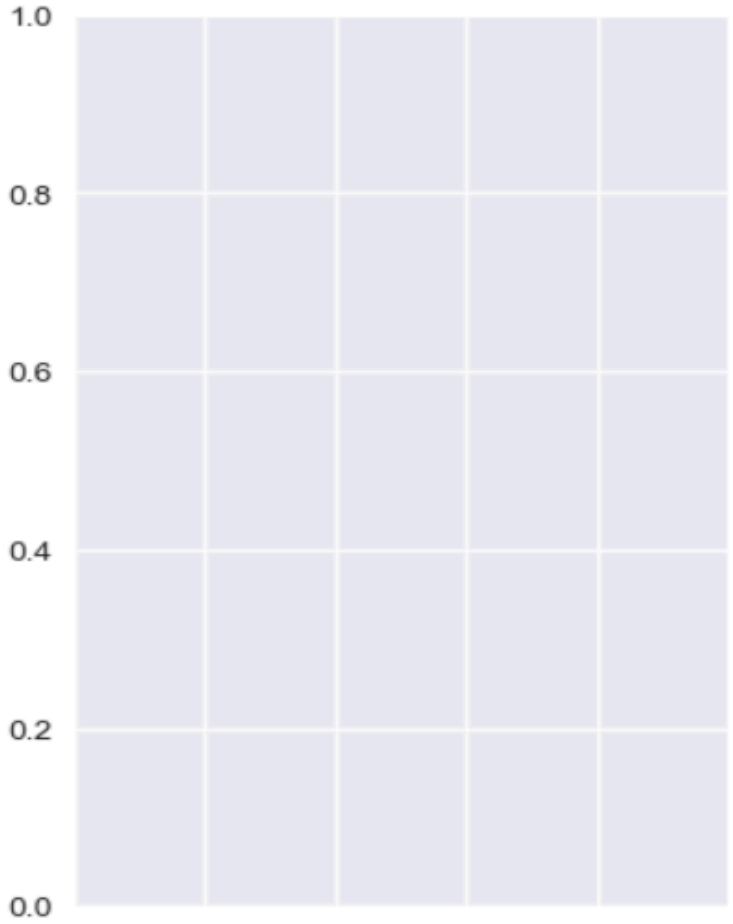



```
Accuracy with k=5 93.60000000000001
Accuracy with k=1 90.4
```

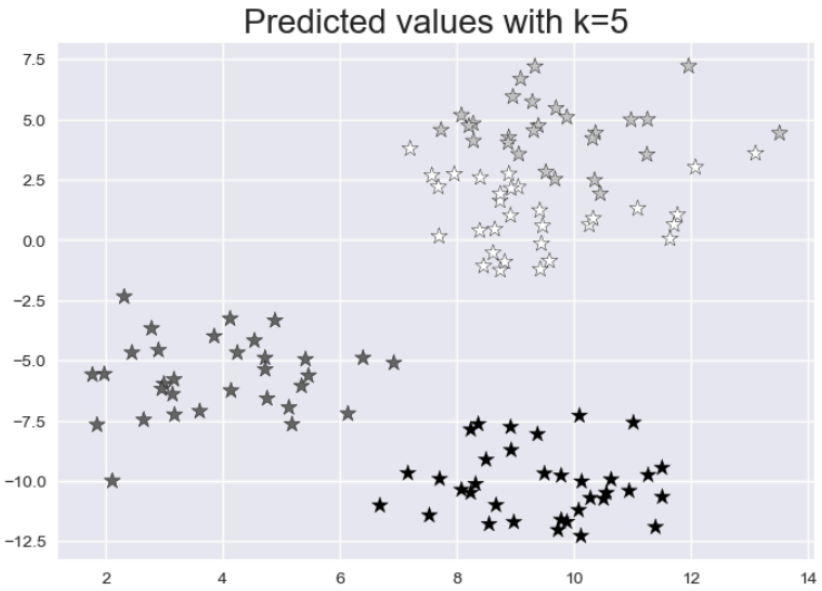
Out[10]: <Axes: >

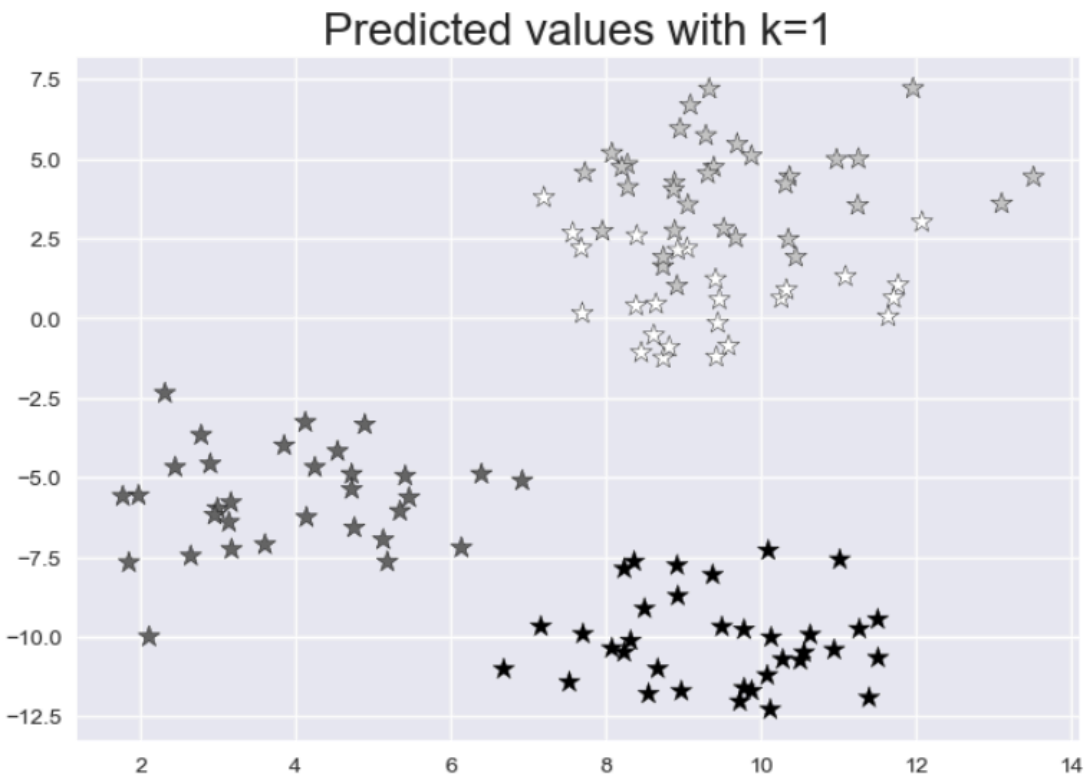


Out[8]: <Axes: >



Out[17]: <matplotlib.collections.PathCollection at 0x1fe00c4b910>




Predicted values with k=1
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
−12.5
2
4
6
8
10
12
14

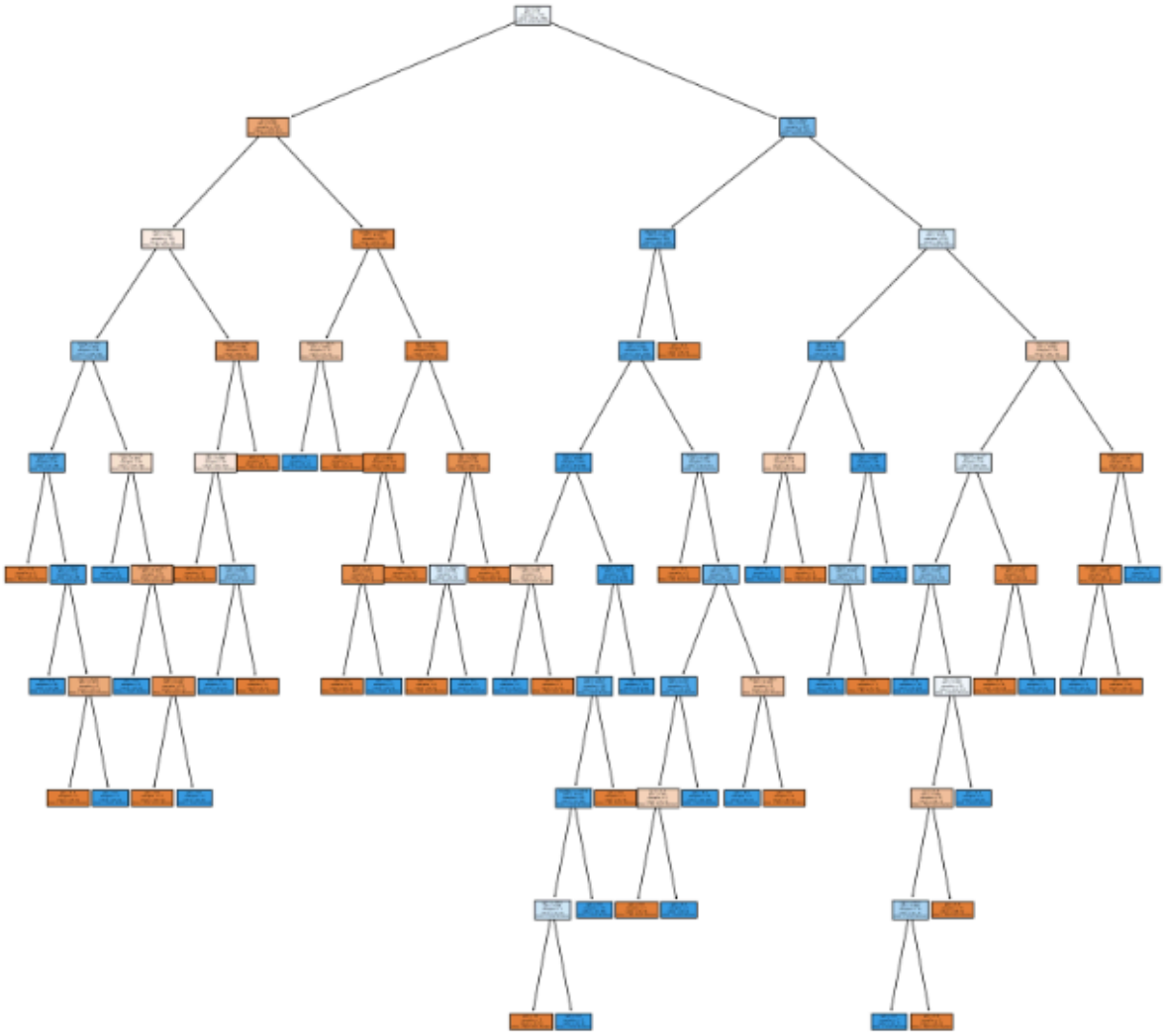
```
import numpy as np
import pandas as pd
import os
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn import tree
from sklearn.model_selection import RandomizedSearchCV, GridSearchCV
from sklearn.metrics import accuracy_score,confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
data = "C:\\Users\\DELL\\Downloads\\heart1.csv"
df= pd.read_csv(data)
df.head()
X = df.drop(columns=['target'])
y= df['target']
print(X.shape)
print(y.shape)
x_train,x_test,y_train,y_test=train_test_split(X,y,stratify=y)
print(x_train.shape)
print(x_test.shape)
clf= tree.DecisionTreeClassifier(random_state=0)
clf.fit(x_train,y_train)
y_train_pred = clf.predict(x_train)
y_test_pred = clf.predict(x_test)
plt.figure(figsize=(20,20))
features = df.columns.tolist()
classes = ['Not heart disease','heart disease']
tree.plot_tree(clf,feature_names=features,class_names=classes, filled=True)
plt.show()
def plot_confusionmatrix(y_train_pred,y_train,dom):
    print(f'{dom} Confusion matrix')
    cf= confusion_matrix(y_train_pred,y_train)
    sns.heatmap(cf,annot=True,yticklabels=classes, xticklabels=classes,cmap='Blues',
fmt='g')
    plt.tight_layout()
    plt.show()
    print(f'Train score {accuracy_score(y_train_pred,y_train)}')
print(f'Test score {accuracy_score(y_test_pred,y_test)}')
plot_confusionmatrix(y_train_pred,y_train,dom='Train')
plot_confusionmatrix(y_test_pred,y_test,dom='Test')
```

```
path = clf.cost_complexity_pruning_path(x_train, y_train)
ccp_alphas, impurities = path.ccp_alphas, path.impurities
print(ccp_alphas)
clfs =[]
for ccp_alpha in ccp_alphas:
    clf= tree.DecisionTreeClassifier(random_state=0, ccp_alpha=ccp_alpha)
    clf.fit(x_train, y_train)
    clfs.append(clf)
clf_= tree.DecisionTreeClassifier(random_state=0,ccp_alpha=0.020)
clf_.fit(x_train,y_train)
y_train_pred = clf_.predict(x_train)
y_test_pred = clf_.predict(x_test)
print(f'Train score {accuracy_score(y_train_pred,y_train)}')
print(f'Test score {accuracy_score(y_test_pred,y_test)}')
plot_confusionmatrix(y_train_pred,y_train,dom='Train')
plot_confusionmatrix(y_test_pred,y_test,dom='Test')
plt.figure(figsize=(20,20))
features = df.columns.tolist()
classes =['Not heart disease','heart disease']
tree.plot_tree(clf_,feature_names=features,class_names=classes, filled=True)
plt.show()
```

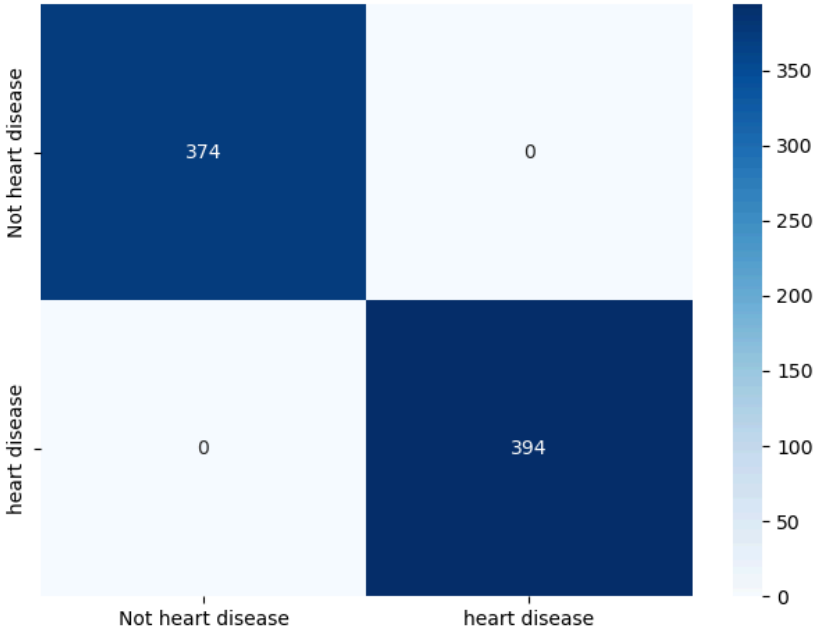
Out[51]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	0
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	0
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	0

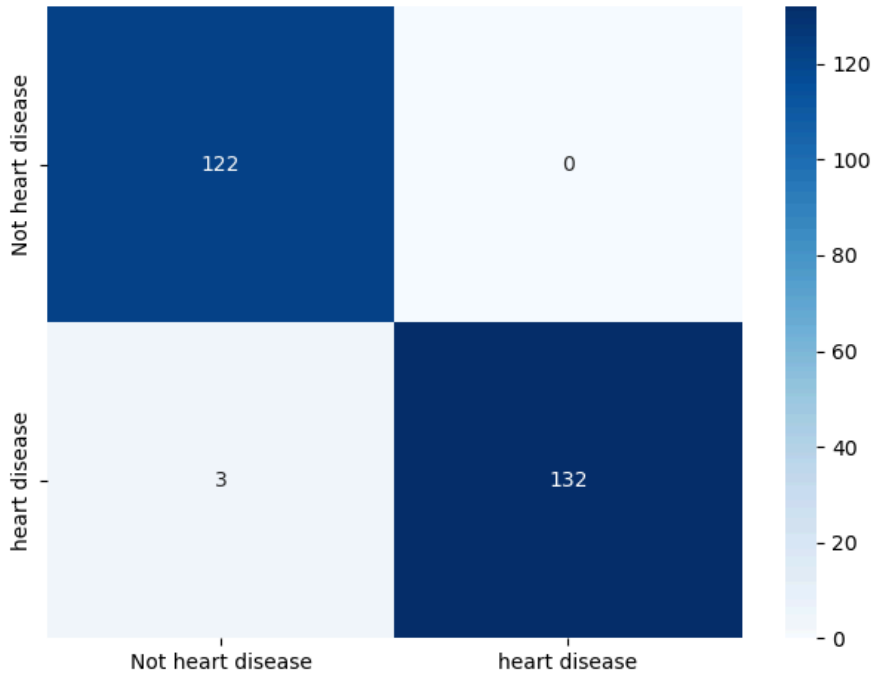
```
(1025, 13)
(1025,)
(768, 13)
(257, 13)
```



```
Test score 0.9883268482490273
Train Confusion matrix
```



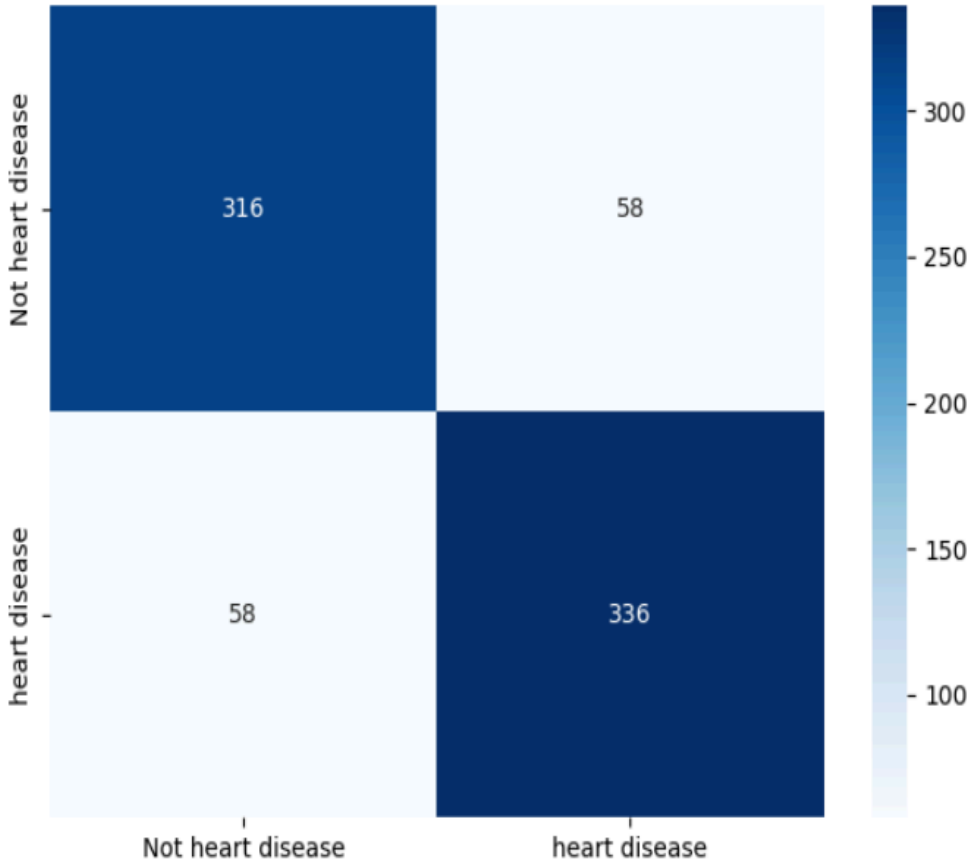

```
Train score 1.0
Test Confusion matrix
```



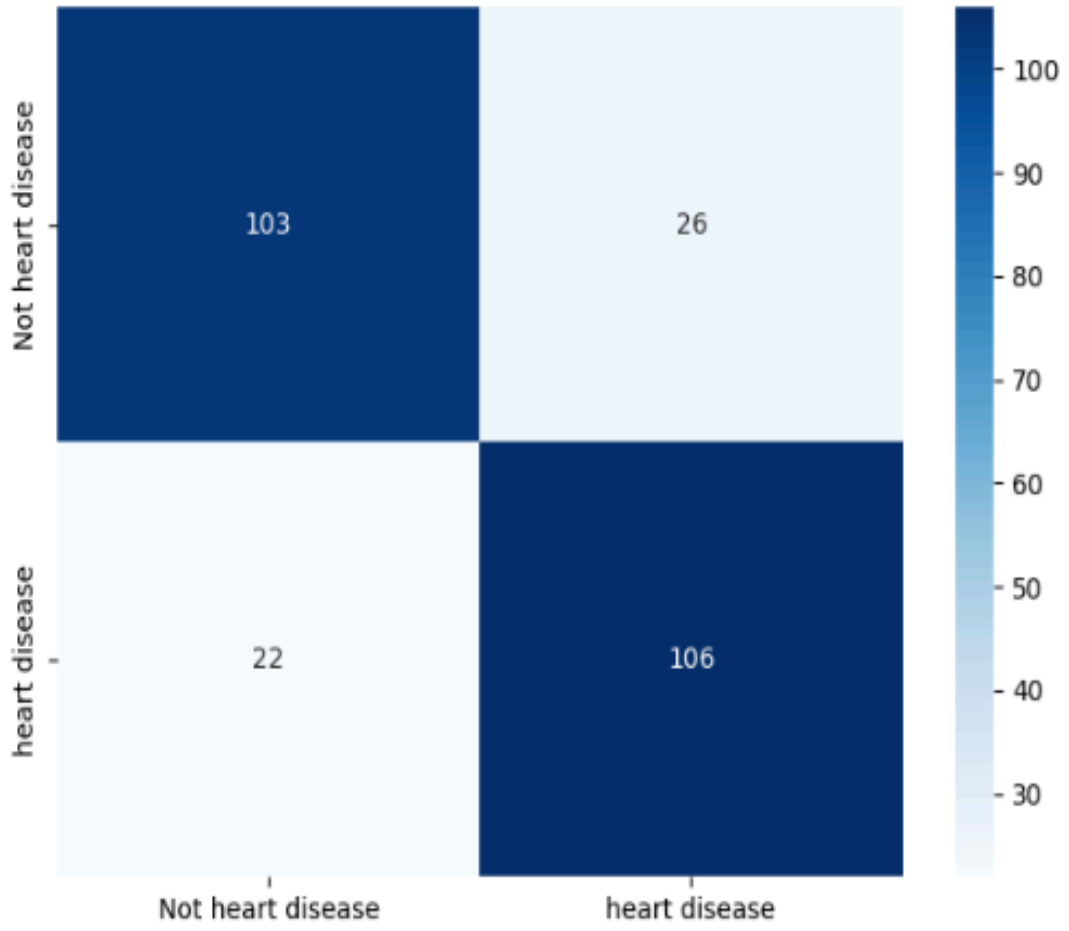

```
Train score 0.9883268482490273
```

```
[0.         0.00127378 0.00195312 0.00243056 0.00252976 0.00257035
 0.003125   0.003125   0.00317301 0.00393145 0.0041447  0.00419529
 0.00464929 0.00466009 0.00477071 0.00488281 0.00520833 0.00540147
 0.00540865 0.00589928 0.00607639 0.00645852 0.00670498 0.00674065
 0.00695168 0.00708129 0.00710227 0.00806297 0.00858874 0.00912247
 0.00992359 0.0119576  0.01266927 0.0131774  0.02414773 0.0277737
 0.03307757 0.03893083 0.14217872]
```

```
Train score 0.8489583333333334
Test score 0.8132295719844358
Train Confusion matrix
```

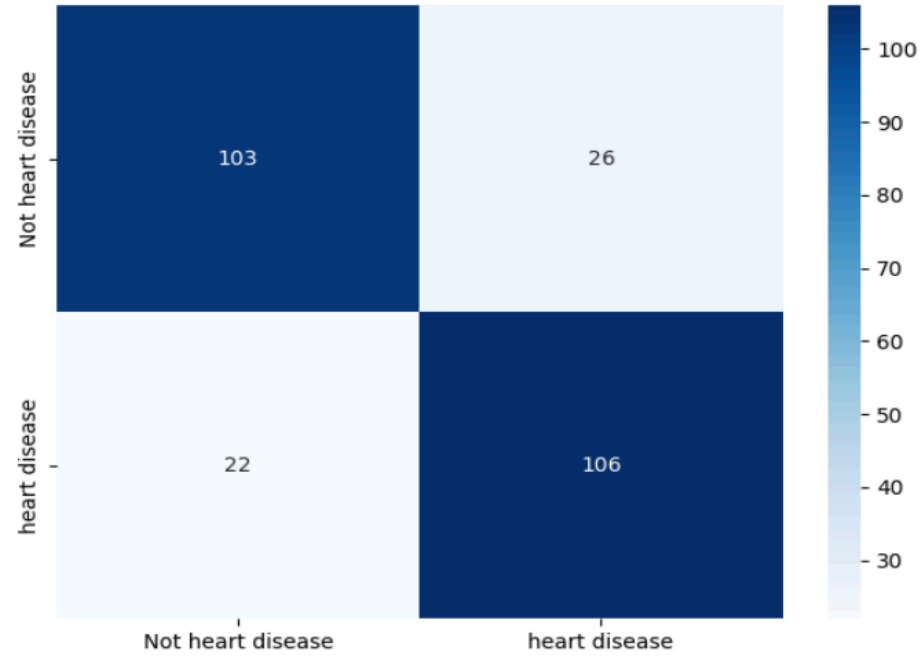


```
Train score 0.8489583333333334
Test Confusion matrix
```

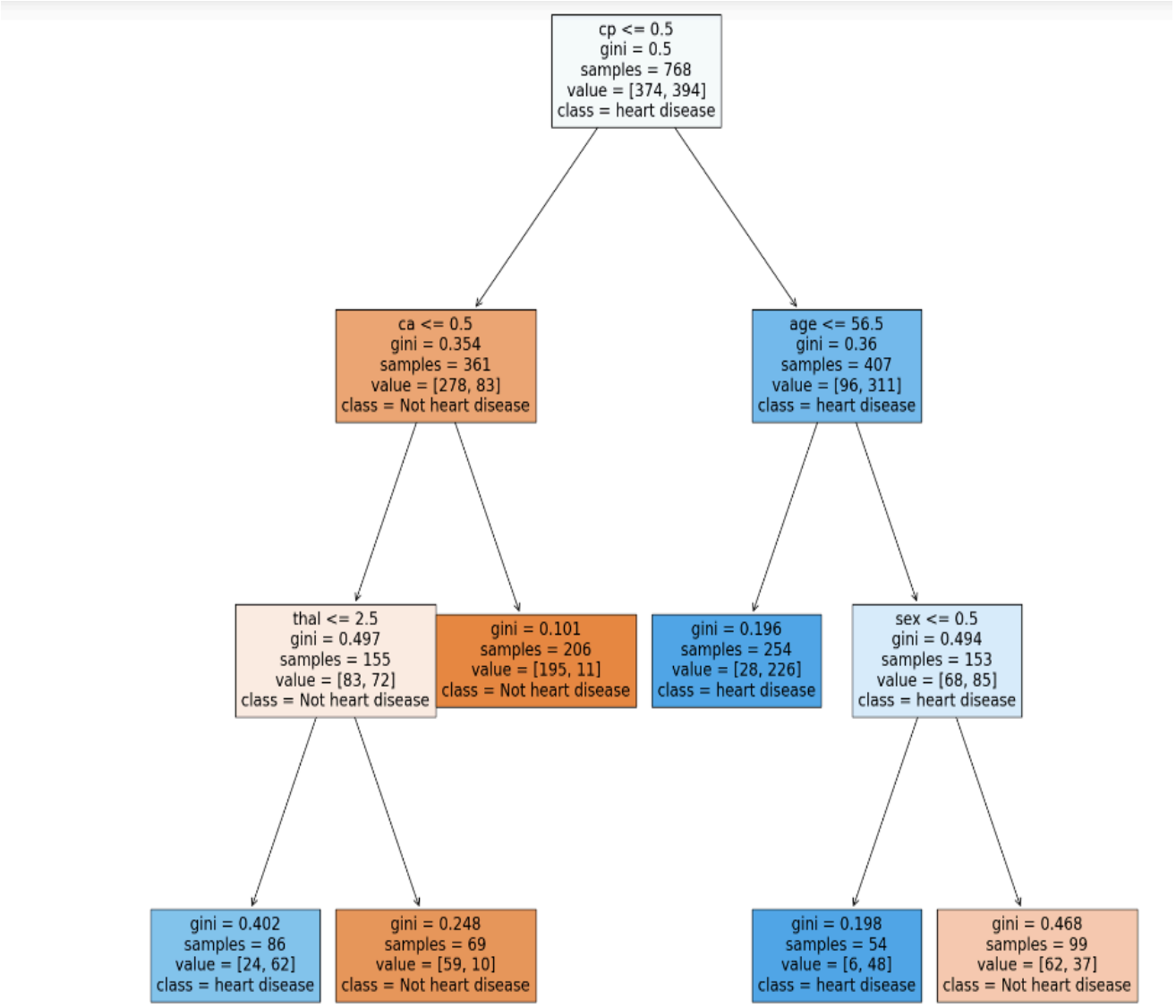



```
Train score 0.8132295719844358
```

```
Train score 0.8489583333333334
Test Confusion matrix
```




```
Train score 0.8132295719844358
```


cp <= 0.5
gini = 0.5
samples = 768
value = [374, 394]
class = heart disease
ca <= 0.5
gini = 0.354
samples = 361
value = [278, 83]
class = Not heart disease
age <= 56.5
gini = 0.36
samples = 407
value = [96, 311]
class = heart disease
thal <= 2.5
gini = 0.497
samples = 155
value = [83, 72]
class = Not heart disease
gini = 0.101
samples = 206
value = [195, 11]
class = Not heart disease
gini = 0.196
samples = 254
value = [28, 226]
class = heart disease
sex <= 0.5
gini = 0.494
samples = 153
value = [68, 85]
class = heart disease
gini = 0.402
samples = 86
value = [24, 62]
class = heart disease
gini = 0.248
samples = 69
value = [59, 10]
class = Not heart disease
gini = 0.198
samples = 54
value = [6, 48]
class = heart disease
gini = 0.468
samples = 99
value = [62, 37]
class = Not heart disease

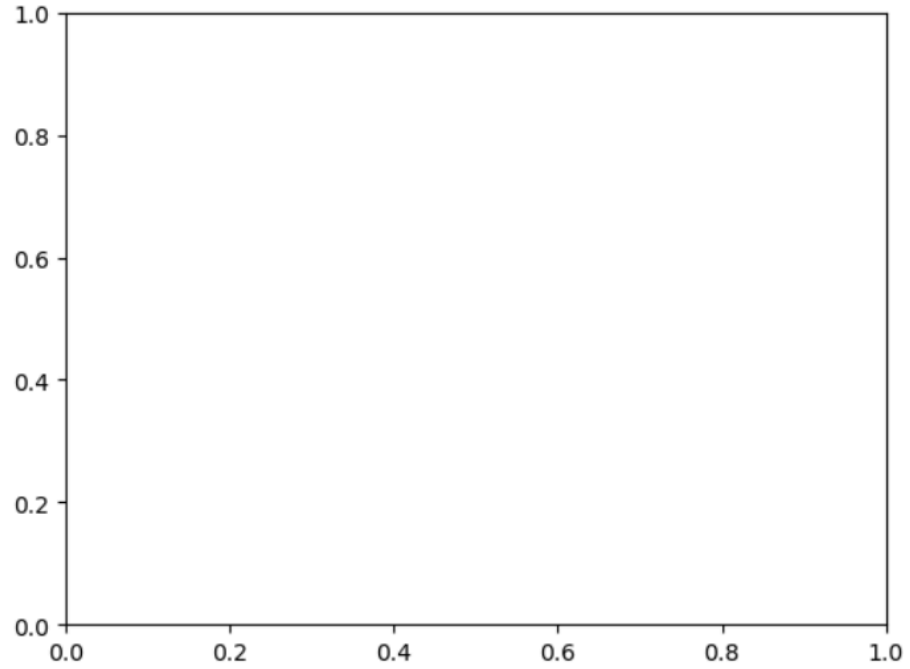
```
import numpy as np
X= np.array(([2, 9], [1, 5], [3, 6]), dtype=float)
y= np.array(([92], [86], [89]), dtype=float)
X= X/np.amax(X,axis=0)
y= y/100
def sigmoid (x):
    return 1/(1 + np.exp(-x))
def derivatives_sigmoid(x):
    return x * (1 -x)
epoch=5
Ir=0.1
inputlayer_neurons = 2
hiddenlayer_neurons =3
output_neurons =1
wh=np.random.uniform(size=(inputlayer_neurons,hiddenlayer_neurons))
bh=np.random.uniform(size=(1,hiddenlayer_neurons))
wout=np.random.uniform(size=(hiddenlayer_neurons,output_neurons))
bout=np.random.uniform(size=(1,output_neurons))
for i in range(epoch):
    hinpl=np.dot(X, wh)
    hinp=hinpl + bh
hlayer_act = sigmoid(hinp)
outinpl=np.dot(hlayer_act,wout)
outinp= outinpl+bout
output = sigmoid(outinp)
EO = y-output
outgrad = derivatives_sigmoid(output)
d_output = EO * outgrad
EH= d_output.dot(wout.T)
hiddengrad = derivatives_sigmoid(hlayer_act)
d_hiddenlayer = EH * hiddengrad
wout += hlayer_act.T.dot(d_output) *Ir
wh += X.T.dot(d_hiddenlayer) *Ir
print ("---Epoch-", i+1, "Starts--------.")
print("Input: \n" + str(X))
print("Actual Output:\n" + str(y))
print("Predicted Output:\n",output)
print ("----Epoch-", i+1, "Ends----------\n")
print("Input: \n" + str(X))
print("Actual Output: \n" + str(y))
print("Predicted Output: \n",output)
```

```
---Epoch- 5 Starts--------.
Input:
[[0.66666667 1.        ]
 [0.33333333 0.55555556]
 [1.         0.66666667]]
Actual Output:
[[0.92]
 [0.86]
 [0.89]]
Predicted Output:
 [[0.85966332]
 [0.84166838]
 [0.8614564 ]]
----Epoch- 5 Ends----------
```

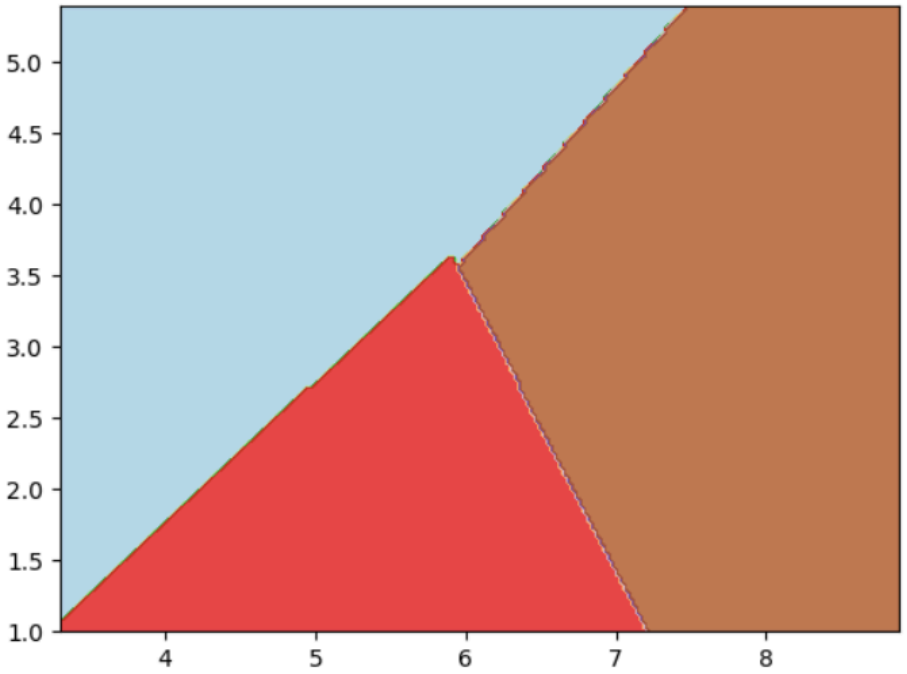
```
Input:
[[0.66666667 1.        ]
 [0.33333333 0.55555556]
 [1.         0.66666667]]
Actual Output:
[[0.92]
 [0.86]
 [0.89]]
Predicted Output:
 [[0.85966332]
 [0.84166838]
 [0.8614564 ]]
```

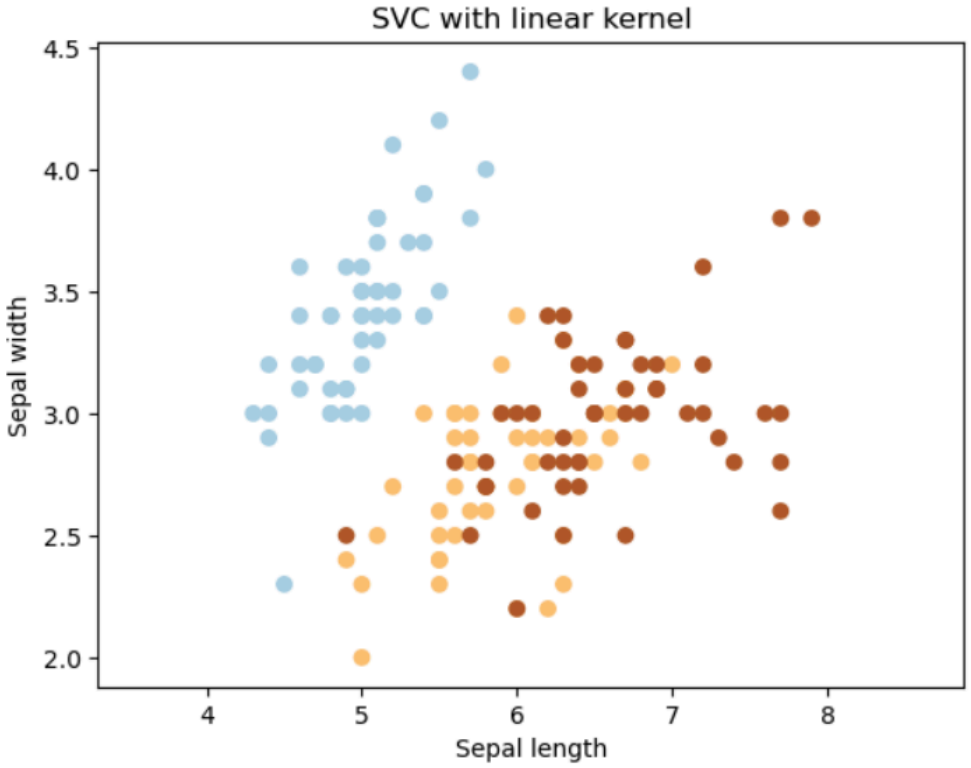
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm, datasets
iris = datasets.load_iris()
X= iris.data[:, :2]
y= iris.target
C=1.0
SvC = svm.SVC(kernel ='linear', C = 1).fit(X, y)
x_min, x_max = X[:, 0].min() - 1, X[:,0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:,1].max() + 1
h=(x_max /x_min)/100
XX, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max,h))

plt.subplot(1, 1, 1)

Z= SvC.predict(np.c_[XX.ravel(), yy.ravel()])
Z= Z.reshape(XX.shape)
plt.contourf(XX, yy, Z, cmap = plt.cm.Paired, alpha = 0.8)

plt.scatter(X[:, 0], X[:, 1], c = y,cmap=plt.cm.Paired)
plt.xlabel('Sepal length')
plt.ylabel('Sepal width')
plt.xlim(XX.min(), XX.max())
plt.title('SVC with linear kernel')
plt.show()
```

Out[10]: <Axes: >



Out[15]: <matplotlib.contour.QuadContourSet at 0x2c6b22e8fd0>




SVC with linear kernel
Sepal width
Sepal length
4.5
4.0
3.5
3.0
2.5
2.0
4
5
6
7
8

```
import pandas as pd
col_names=['pregnant','glucose','bp','skin','insulin','bmi','age','label']
pima=pd.read_csv("C:\\Users\\DELL\\Downloads\\pima-indians-diabetes.csv",header
=None,names=col_names)
pima.head()

feature_cols=['pregnant','insulin','bmi','age','glucose','bp']
X=pima[feature_cols]
y=pima.label
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.25,random_state=16)
from sklearn.linear_model import LogisticRegression
logreg=LogisticRegression(random_state=16)
logreg.fit(X_train,y_train)
y_pred=logreg.predict(X_test)
print(y_pred)
```

Out[9]:

	pregnant	glucose	bp	skin	insulin	bmi	age	label
6	148	72	35	0	33.6	0.627	50	1
1	85	66	29	0	26.6	0.351	31	0
8	183	64	0	0	23.3	0.672	32	1
1	89	66	23	94	28.1	0.167	21	0
0	137	40	35	168	43.1	2.288	33	1

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