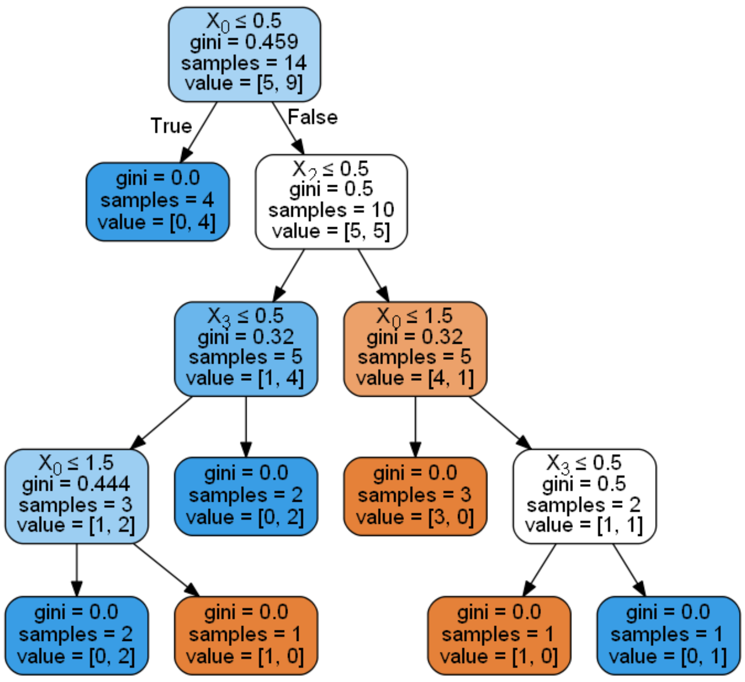
```
import pandas as pd
import numpy as np
from sklearn.externals.six import StringIO
from IPython.display import Image
from sklearn.tree import export_graphviz
import pydotplus

dataset=pd.read_csv("data.csv")
X=dataset.iloc[:,:-1]
y=dataset.iloc[:,5]

from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()

X=X.apply(le.fit_transform)
print("X")

from sklearn.tree import DecisionTreeClassifier
regressor=DecisionTreeClassifier()
regressor.fit(X.iloc[:,1:5],y)

X_in=np.array([1,1,0,0])
y_pred=regressor.predict([X_in])
print("Prediction:", y_pred)


dot_data=StringIO()
export_graphviz(regressor,out_file=dot_data,filled=True,rounded=True,special_characters=True)
graph=pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('tree.png')
```

Output

```
X
Prediction: ['Yes']
```

Decision Tree generated-

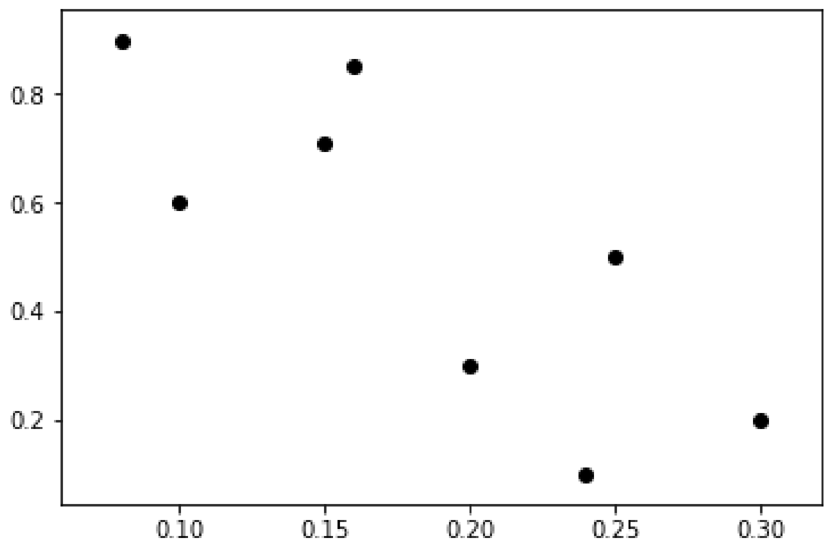


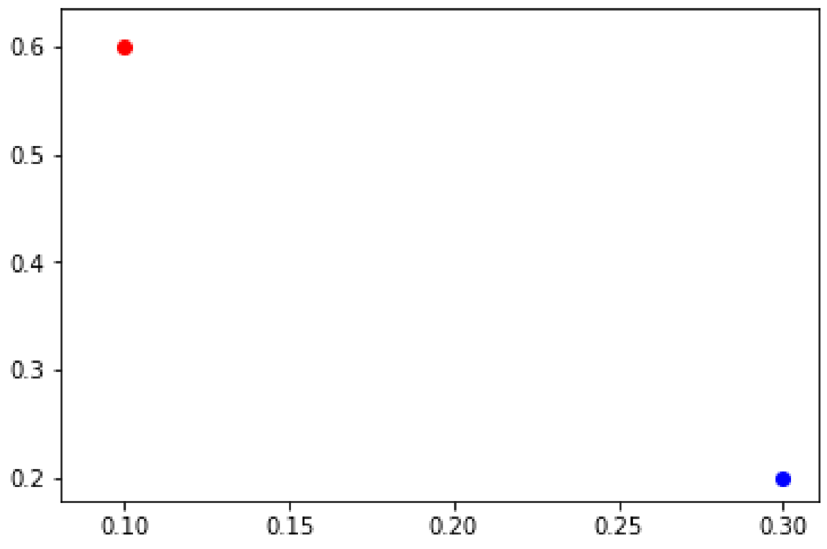
$X_0 \leq 0.5$
gini = 0.459
samples = 14
value = [5, 9]
True
False
gini = 0.0
samples = 4
value = [0, 4]
$X_2 \leq 0.5$
gini = 0.5
samples = 10
value = [5, 5]
$X_3 \leq 0.5$
gini = 0.32
samples = 5
value = [1, 4]
$X_0 \leq 1.5$
gini = 0.32
samples = 5
value = [4, 1]
$X_0 \leq 1.5$
gini = 0.444
samples = 3
value = [1, 2]
gini = 0.0
samples = 2
value = [0, 2]
gini = 0.0
samples = 3
value = [3, 0]
$X_3 \leq 0.5$
gini = 0.5
samples = 2
value = [1, 1]
gini = 0.0
samples = 2
value = [0, 2]
gini = 0.0
samples = 1
value = [1, 0]
gini = 0.0
samples = 1
value = [1, 0]
gini = 0.0
samples = 1
value = [0, 1]

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

df=pd.DataFrame({'X':[0.1,0.15,0.08,0.16,0.2,0.25,0.24,0.3],
                'y':[0.6,0.71,0.9,0.85,0.3,0.5,0.1,0.2]})
f1 = df['X'].values
f2 = df['y'].values
X = np.array(list(zip(f1, f2)))
print(X)

C_x=np.array([0.1,0.3])
C_y=np.array([0.6,0.2])
centroids=C_x,C_y

colmap = {1: 'r', 2: 'b'}
plt.scatter(f1, f2, color='k')
plt.show()

plt.scatter(C_x[0],C_y[0], color=colmap[1])
plt.scatter(C_x[1],C_y[1], color=colmap[2])
plt.show()

C = np.array(list((C_x, C_y)), dtype=np.float32)
print (C)

plt.scatter(f1, f2, c='#050505')
plt.scatter(C_x[0], C_y[0], marker='*', s=200, c='r')
plt.scatter(C_x[1], C_y[1], marker='*', s=200, c='b')
plt.show()

from sklearn.cluster import KMeans
model=KMeans(n_clusters=2,random_state=0)
model.fit(X)
labels=model.labels_
print(labels)

#using labels find population around centroid
count=0
for i in range(len(labels)):
    if (labels[i]==1):
        count=count+1

print('No of population around cluster 2:',count-1)
new_centroids = model.cluster_centers_
print('Previous value of m1 and m2 is:')
print('M1==',centroids[0])
print('M1==',centroids[1])
print('updated value of m1 and m2 is:')
print('M1==',new_centroids[0])
print('M1==',new_centroids[1])
```

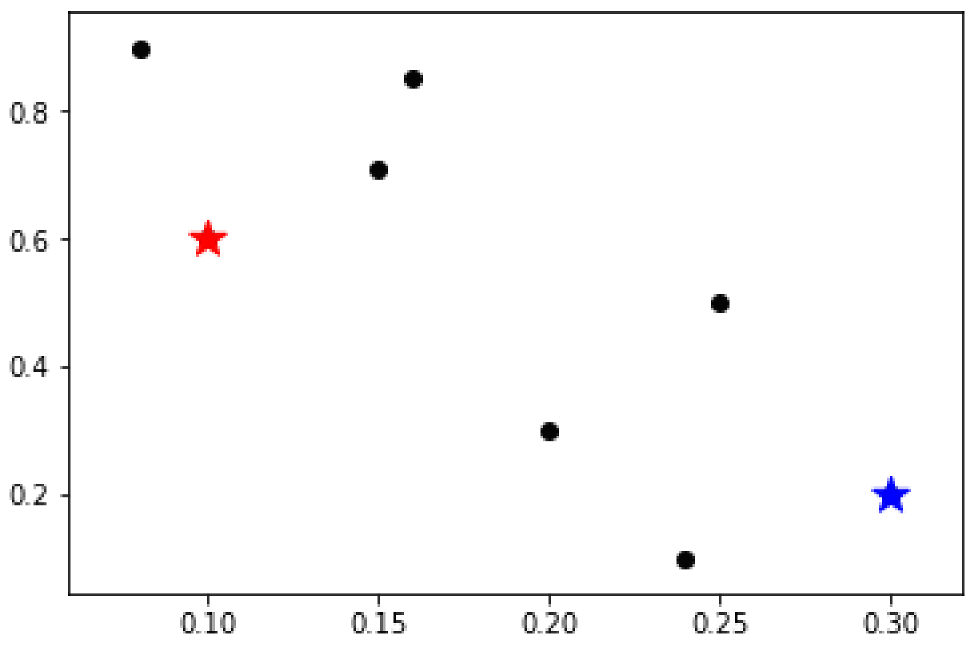
Output

```
[[0.1  0.6 ]
 [0.15 0.71]
 [0.08 0.9 ]
 [0.16 0.85]
 [0.2  0.3 ]
 [0.25 0.5 ]
 [0.24 0.1 ]
 [0.3  0.2 ]]
```





```
[[0.1 0.3]
 [0.6 0.2]]
```



```
[1 1 1 1 0 0 0 0]
No of population around cluster 2: 3
Previous value of m1 and m2 is:
M1== [0.1 0.3]
M1== [0.6 0.2]
updated value of m1 and m2 is:
M1== [0.2475 0.275 ]
M1== [0.1225 0.765 ]
```

```
import pandas as pd
import numpy as np

dataset=pd.read_csv("kdata.csv")
X=dataset.iloc[:,:-1].values
y=dataset.iloc[:,2].values

from sklearn.neighbors import KNeighborsClassifier
classifier=KNeighborsClassifier(n_neighbors=3)
classifier.fit(X,y)

X_test=np.array([6,6])
y_pred=classifier.predict([X_test])
print('General KNN',y_pred)

classifier=KNeighborsClassifier(n_neighbors=3,weights='distance')
classifier.fit(X,y)

X_test=np.array([6,2])
y_pred=classifier.predict([X_test])
print('Distance Weighted KNN',y_pred)
```

Output

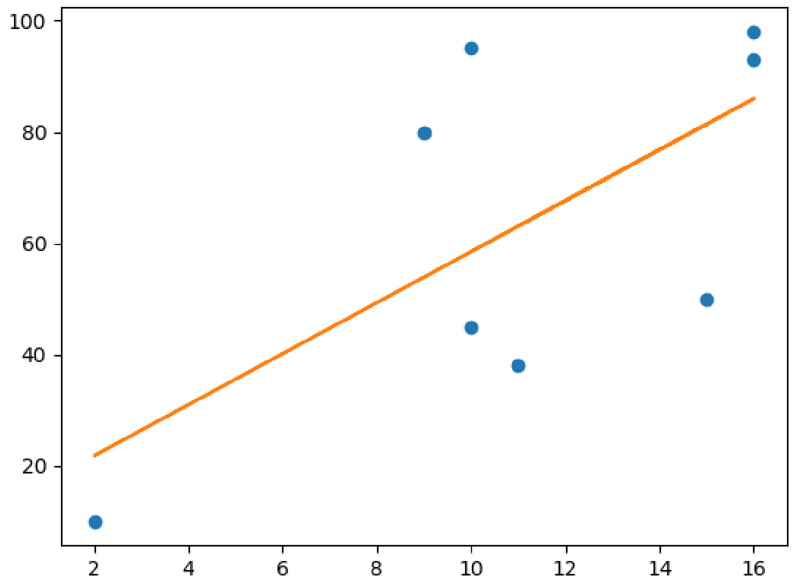
```
General KNN ['negative']
Distance Weighted KNN ['positive']
```

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

dataset=pd.read_csv("hours.csv")
X=dataset.iloc[:,:-1].values
y=dataset.iloc[:,1].values

from sklearn.linear_model import LinearRegression
regressor=LinearRegression()
regressor.fit(X,y)
Accuracy=regressor.score(X, y)*100
print("Accuracy :")
print(Accuracy)

y_pred=regressor.predict([[10]])
print(y_pred)
hours=int(input('Enter the no of hours'))
eq=regressor.coef_*hours+regressor.intercept_
y='%f*%f+%f' %(regressor.coef_,hours,regressor.intercept_)
print("y :")
print(y)
print("Risk Score : ", eq[0])
plt.plot(X,y,'o')
plt.plot(X,regressor.predict(X));
plt.show()
```

Output

```
Accuracy :
43.709481451010035
[58.46361406]

Enter the no of hours 10
y :
4.587899*10.000000+12.584628
Risk Score :  58.4636140637776
```

Graph


100
80
60
40
20
2
4
6
8
10
12
14
16