**Prediction of Mobile price range using decision tree**[**¶**](#gjdgxs)

Bob has started his own mobile company. He wants to give tough fight to big companies like Apple,Samsung etc.

He does not know how to estimate price of mobiles his company creates. In this competitive mobile phone market you cannot simply assume things. To solve this problem he collects sales data of mobile phones of various companies.

Bob wants to find out some relation between features of a mobile phone(eg:- RAM,Internal Memory etc) and its selling price. But he is not so good at Machine Learning. So he needs your help to solve this problem.

In this problem you do not have to predict actual price but a price range indicating how high the price is.

0 - Very low 1 - Low 2 - High 3 - Very high

In [1]:

*#IMPORTING THE LIBRARIES*  
**import** **pandas** **as** **pd**  
**import** **numpy** **as** **np**  
**import** **seaborn** **as** **sns**  
**import** **matplotlib.pyplot** **as** **plt**  
%**matplotlib** inline  
**from** **sklearn.model\_selection** **import** train\_test\_split, cross\_val\_score  
**from** **sklearn.metrics** **import** accuracy\_score, roc\_auc\_score, roc\_curve  
**import** **warnings**  
warnings.filterwarnings("ignore", category=**DeprecationWarning**)

**1) Acquire the Data**[**¶**](#30j0zll)

In [2]:

*#LOADING THE DATASET*  
df = pd.read\_csv('mobile.csv')  
df.head()

Out[2]:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **battery\_power** | **blue** | **clock\_speed** | **dual\_sim** | **fc** | **four\_g** | **int\_memory** | **m\_dep** | **mobile\_wt** | **n\_cores** | **...** | **px\_height** | **px\_width** | **ram** | **sc\_h** | **sc\_w** | **talk\_time** | **three\_g** | **touch\_screen** | **wifi** | **price\_range** |
| **0** | 842 | 0 | 2.2 | 0 | 1 | 0 | 7 | 0.6 | 188 | 2 | ... | 20 | 756 | 2549 | 9 | 7 | 19 | 0 | 0 | 1 | 1 |
| **1** | 1021 | 1 | 0.5 | 1 | 0 | 1 | 53 | 0.7 | 136 | 3 | ... | 905 | 1988 | 2631 | 17 | 3 | 7 | 1 | 1 | 0 | 2 |
| **2** | 563 | 1 | 0.5 | 1 | 2 | 1 | 41 | 0.9 | 145 | 5 | ... | 1263 | 1716 | 2603 | 11 | 2 | 9 | 1 | 1 | 0 | 2 |
| **3** | 615 | 1 | 2.5 | 0 | 0 | 0 | 10 | 0.8 | 131 | 6 | ... | 1216 | 1786 | 2769 | 16 | 8 | 11 | 1 | 0 | 0 | 2 |
| **4** | 1821 | 1 | 1.2 | 0 | 13 | 1 | 44 | 0.6 | 141 | 2 | ... | 1208 | 1212 | 1411 | 8 | 2 | 15 | 1 | 1 | 0 | 1 |

5 rows × 21 columns

In [3]:

X = df.drop(["price\_range"],axis = 1)  
y = df['price\_range']

In [4]:

*#DISTRIBUTION OF DATASET INTO TRAINING AND TESTING SETS*  
X\_train,X\_test,y\_train,y\_test = train\_test\_split(X,y,test\_size=0.2,random\_state=0)

**2) Train the Model**[**¶**](#1fob9te)

In [5]:

**from** **sklearn.tree** **import** DecisionTreeClassifier

In [6]:

my\_model = DecisionTreeClassifier(random\_state=0)  
result = my\_model.fit(X\_train,y\_train)

**3) Test the Model**[**¶**](#3znysh7)

In [7]:

predictions = result.predict(X\_test)  
predictions

Out[7]:

array([3, 0, 2, 2, 3, 0, 0, 3, 3, 1, 0, 3, 0, 2, 3, 0, 3, 2, 2, 1, 0, 0,  
 3, 1, 2, 2, 3, 0, 3, 1, 1, 0, 2, 0, 2, 3, 0, 0, 3, 3, 3, 1, 3, 2,  
 1, 3, 0, 1, 3, 1, 0, 3, 0, 3, 0, 3, 3, 1, 0, 3, 3, 1, 3, 2, 2, 2,  
 3, 2, 2, 2, 3, 2, 1, 0, 1, 3, 2, 2, 2, 2, 3, 3, 2, 0, 0, 0, 2, 1,  
 2, 3, 1, 2, 3, 0, 0, 2, 3, 3, 0, 3, 2, 1, 3, 1, 3, 2, 2, 2, 2, 3,  
 3, 0, 0, 1, 2, 3, 0, 0, 0, 0, 1, 3, 2, 1, 2, 2, 1, 1, 0, 2, 0, 3,  
 2, 3, 3, 3, 3, 1, 0, 1, 1, 2, 2, 3, 0, 2, 0, 0, 1, 0, 2, 1, 1, 1,  
 3, 0, 0, 3, 1, 3, 2, 1, 3, 1, 2, 3, 3, 2, 1, 0, 3, 2, 2, 3, 3, 0,  
 2, 2, 3, 0, 3, 1, 0, 1, 3, 1, 3, 0, 2, 3, 1, 1, 0, 3, 3, 0, 1, 3,  
 1, 0, 3, 3, 3, 1, 2, 3, 2, 3, 0, 0, 0, 2, 3, 3, 0, 0, 1, 3, 2, 3,  
 3, 3, 0, 0, 2, 1, 3, 1, 0, 2, 0, 0, 0, 3, 3, 0, 2, 2, 0, 1, 0, 2,  
 3, 3, 0, 0, 1, 3, 3, 1, 3, 0, 3, 1, 1, 0, 2, 3, 3, 2, 0, 0, 1, 2,  
 3, 2, 2, 3, 2, 1, 1, 3, 3, 3, 2, 3, 3, 2, 2, 1, 0, 2, 1, 1, 0, 0,  
 2, 2, 2, 2, 0, 2, 3, 0, 1, 3, 3, 0, 2, 0, 1, 1, 3, 0, 0, 1, 3, 1,  
 1, 0, 2, 0, 3, 0, 3, 3, 2, 2, 1, 2, 2, 0, 1, 2, 0, 1, 0, 3, 1, 0,  
 3, 1, 0, 1, 2, 0, 3, 1, 2, 1, 1, 3, 0, 2, 1, 1, 1, 2, 2, 0, 2, 0,  
 0, 3, 1, 2, 3, 2, 3, 0, 3, 2, 1, 1, 2, 2, 3, 3, 3, 0, 2, 0, 2, 0,  
 1, 1, 2, 2, 2, 3, 1, 2, 0, 1, 2, 3, 0, 0, 1, 3, 0, 2, 0, 2, 2, 1,  
 1, 0, 2, 1], dtype=int64)

In [8]:

**from** **sklearn.metrics** **import** mean\_absolute\_error,accuracy\_score

In [9]:

mean\_absolute\_error(y\_test, predictions)

Out[9]:

0.175

In [10]:

accuracy\_score(y\_test,predictions)

Out[10]:

0.8275

In [11]:

**from** **sklearn.metrics** **import** classification\_report  
matrix = classification\_report(y\_test,predictions,labels=[1,0])  
matrix

Out[11]:

' precision recall f1-score support\n\n 1 0.80 0.73 0.76 92\n 0 0.91 0.96 0.93 95\n\n micro avg 0.86 0.84 0.85 187\n macro avg 0.85 0.84 0.85 187\nweighted avg 0.85 0.84 0.85 187\n'

In [12]:

**from** **sklearn.metrics** **import** confusion\_matrix  
**import** **seaborn** **as** **sn**  
matrix=confusion\_matrix(y\_test,predictions)  
print(matrix)

[[ 91 4 0 0]  
 [ 9 67 16 0]  
 [ 0 12 73 14]  
 [ 0 1 13 100]]

In [14]:

confusion\_df = pd.DataFrame(matrix, index=['Actual Label 0','Actual Label 1','Actual Label 2','Actual Label 3'], columns=['Predicted Label 0','Predicted Label 1','Predicted Label 2','Predicted Label 3'])  
confusion\_df

Out[14]:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Predicted Label 0** | **Predicted Label 1** | **Predicted Label 2** | **Predicted Label 3** |
| **Actual Label 0** | 91 | 4 | 0 | 0 |
| **Actual Label 1** | 9 | 67 | 16 | 0 |
| **Actual Label 2** | 0 | 12 | 73 | 14 |
| **Actual Label 3** | 0 | 1 | 13 | 100 |

In [16]:

**from** **sklearn** **import** metrics  
print('**\n**\*\*Classification Report:**\n**',metrics.classification\_report(y\_test,predictions))

\*\*Classification Report:  
 precision recall f1-score support  
  
 0 0.91 0.96 0.93 95  
 1 0.80 0.73 0.76 92  
 2 0.72 0.74 0.73 99  
 3 0.88 0.88 0.88 114  
  
 micro avg 0.83 0.83 0.83 400  
 macro avg 0.83 0.83 0.82 400  
weighted avg 0.83 0.83 0.83 400

**4) Deploy the Model**[**¶**](#2et92p0)

In [17]:

pred\_new=list(result.predict([[842,0,2.2,0,1,0,7,0.6,188,2,2,20,756,2549,9,7,19,0,0,1]]))  
pred\_new

Out[17]:

[1]

In [18]:

pred\_new=list(result.predict([[1821,0,1.7,4,1,10,0.8,139,8,10,381,1018,3220,13,8,18,1,0,1,1]]))  
pred\_new

Out[18]:

[0]

In [ ]:

In [ ]: