Data Science Minor Project

Problem statement

Create a classification model to predict whether price range of mobile based on certain specifications

THE GIVEN DATASET

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

```
df = pd.read_csv('Downloads/Ajitha/mobile_price_range_data.csv')
df.head()
```

battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	...	px_height	px_width	ram	sc_
842	0	2.2	0	1	0	7	0.6	188	2	...	20	756	2549	
1021	1	0.5	1	0	1	53	0.7	136	3	...	905	1988	2631	
563	1	0.5	1	2	1	41	0.9	145	5	...	1263	1716	2603	
615	1	2.5	0	0	0	10	0.8	131	6	...	1216	1786	2769	
1821	1	1.2	0	13	1	44	0.6	141	2	...	1208	1212	1411	

ws × 21 columns

1) DATA PREPROCESSING

Handling null values and duplicate entries

```
df.isnull().sum()
```

```
battery_power    0
blue             0
```

```
clock_speed      0
dual_sim         0
fc               0
four_g           0
int_memory       0
m_dep            0
mobile_wt        0
n_cores          0
pc               0
px_height        0
px_width         0
ram              0
sc_h             0
sc_w             0
talk_time        0
three_g          0
touch_screen     0
wifi             0
price_range      0
dtype: int64
```

```
df.duplicated()
```

```
0       False
1       False
2       False
3       False
4       False
        ...
1995    False
1996    False
1997    False
1998    False
1999    False
Length: 2000, dtype: bool
```

Checking the target variable

```
df['price_range'].value_counts()
```

```
1    500
2    500
3    500
0    500
Name: price_range, dtype: int64
```

2) SPLITING DATA INTO TRAIN AND TEST

```

```

```
x = df.drop('price_range',axis = 1)
y = df['price_range']
print(type(x))
print(type(y))
print(x.shape)
```

```
<class 'pandas.core.frame.DataFrame'>
<class 'pandas.core.series.Series'>
(2000, 20)
```

```
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=42)
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(1500, 20)
(500, 20)
(1500,)
(500,)
```

3) APPLYING THE MODELS

a) Logistic Regression

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix,classification_report
```

```
m1 = LogisticRegression()
m1.fit(x_train,y_train)
```

```
LogisticRegression()
```

```
print('Train Score',m1.score(x_train,y_train))
print('Test Score',m1.score(x_test,y_test))
```

```
Train Score 0.6453333333333333
Test Score 0.616
```

PREDICTED PRICE RANGE FOR TEST DATA

```
ypred_m1 = m1.predict(x_test)
print(ypred_m1)
```

```
[0 2 0 3 1 2 3 0 3 3 0 1 2 3 3 2 2 2 1 0 0 1 0 2 1 1 3 3 3 0 1 0 3 0 2 3 2
 1 3 0 1 2 3 0 3 3 3 1 3 1 3 2 0 0 2 0 1 2 0 0 1 3 3 2 2 0 3 3 1 1 2 1 0 1
 2 0 0 3 2 1 3 2 1 0 1 3 3 3 3 0 3 3 3 0 3 2 2 3 2 1 0 1 0 0 1 3 3 0 0 1 0
 0 3 3 2 1 3 3 0 2 1 3 2 2 3 3 0 3 0 2 3 0 2 2 0 2 1 1 0 2 3 1 3 3 0 0 1 2
 1 2 3 1 1 0 2 3 0 1 0 1 3 3 1 2 1 0 0 2 1 3 3 1 0 0 3 1 1 2 0 1 0 0 0 1 3
 2 0 2 0 0 0 0 1 3 3 1 0 1 1 1 1 2 1 2 3 3 1 3 0 1 1 1 1 1 3 1 1 3 1 1 3 2
 3 0 0 3 0 2 0 0 1 0 2 3 2 1 0 2 3 1 3 3 2 3 0 3 2 2 2 3 3 1 1 3 2 1 2 3 3
 3 3 0 2 2 2 2 3 0 3 3 2 2 2 0 1 3 0 2 3 1 3 1 1 2 0 3 0 0 3 0 1 2 3 2 2 0
 1 0 0 3 3 0 1 1 2 0 3 3 3 3 1 3 2 0 3 2 3 2 0 0 1 3 1 3 1 1 2 0 3 3 2 0 2
 2 2 1 3 1 0 3 1 2 1 1 1 1 2 2 3 3 1 1 1 2 2 0 3 0 0 2 0 0 2 2 2 3 0 1 2 3
 3 3 2 3 1 2 0 2 1 3 3 0 1 3 1 3 2 3 1 0 3 2 0 0 3 3 1 2 3 2 0 3 0 2 2 2 0
 1 1 1 0 0 1 0 3 3 2 1 2 1 3 1 0 3 1 0 0 3 0 3 0 1 1 2 3 0 2 0 2 1 3 3 1 3
 1 2 1 0 3 2 0 2 2 2 2 2 1 1 2 3 1 0 3 1 1 1 3 3 3 2 0 2 2 0 1 2 3 1 2 0 0
 0 2 3 0 1 2 2 2 3 1 2 2 3 0 0 0 2 3 1]
```

COMPUTING CONFUSION MATRIX AND CLASSIFICATION REPORT

```
print(confusion_matrix(y_test,ypred_m1))
print(classification_report(y_test,ypred_m1))
```

```
[[95 36  1  0]
 [24 59 26  9]
 [ 1 23 59 37]
 [ 0  3 32 95]]
              precision    recall  f1-score   support

           0       0.79      0.72      0.75       132
           1       0.49      0.50      0.49       118
           2       0.50      0.49      0.50       120
           3       0.67      0.73      0.70       130

    accuracy                           0.62       500
   macro avg       0.61      0.61      0.61       500
weighted avg       0.62      0.62      0.62       500
```

ACCURACY

```
from sklearn.metrics import accuracy_score
```

```
accuracy_lr = accuracy_score(y_test, ypred_m1)
print("Accuracy with Logistic regression:", accuracy_lr)
```

```
Accuracy with Logistic regression: 0.616
```

b) KNN Classification

```
from sklearn.neighbors import KNeighborsClassifier
```

```
knn1 = KNeighborsClassifier(n_neighbors=11)
knn1.fit(x_train,y_train)
```

```
KNeighborsClassifier(n_neighbors=11)
```

```
print('Train Score', knn1.score(x_train,y_train))
print('Test Score', knn1.score(x_test,y_test))
```

```
Train Score 0.952
Test Score 0.938
```

PREDICTED PRICE RANGE FOR TEST DATA

```
ypred_knn1= knn1.predict(x_test)
print(ypred_knn1)
```

```
[0 2 1 3 1 1 2 0 3 1 0 1 2 3 2 2 3 3 1 0 0 1 1 2 0 1 3 2 2 0 0 0 3 0 1 1 2
 0 3 0 2 2 2 0 3 2 2 1 3 1 3 1 0 0 0 1 1 3 0 0 1 3 3 1 0 0 3 3 1 2 2 2 0 1
 2 0 0 3 2 1 3 2 1 0 1 3 1 3 3 0 3 3 2 1 3 2 2 3 1 1 0 0 1 0 0 3 2 0 1 1 0
 0 3 1 3 2 3 2 0 2 1 3 2 1 3 3 0 2 0 2 3 0 2 2 0 3 1 0 0 2 2 1 2 2 0 0 0 1
 1 2 3 1 1 0 2 2 0 1 0 2 2 3 3 2 1 0 1 2 2 3 3 0 1 0 3 1 1 2 1 0 0 0 0 0 3
 2 0 3 0 0 0 0 1 3 3 1 0 1 1 1 1 1 2 3 3 3 1 2 0 0 0 2 1 1 3 1 0 2 1 1 3 2
 3 0 0 2 1 3 0 1 2 0 2 3 2 0 1 3 3 0 1 3 2 3 0 3 1 2 3 3 2 1 1 3 3 1 3 3 3
 3 3 0 2 2 2 1 3 0 1 3 2 2 2 1 0 1 0 3 3 1 3 1 0 3 1 2 0 0 3 0 1 2 3 3 3 1
 1 0 1 3 3 0 1 2 2 0 3 3 2 3 2 3 2 0 2 1 1 1 0 0 0 3 2 3 1 0 1 0 1 2 3 0 3
 3 2 1 2 0 0 2 1 3 2 0 1 1 1 0 1 3 2 0 0 3 3 0 3 0 0 2 0 1 2 2 2 3 0 3 2 2
 3 3 3 2 1 1 0 3 1 3 3 0 2 3 2 3 3 3 0 0 2 3 0 0 2 3 2 1 1 2 1 2 1 3 1 2 0
 0 1 0 1 0 1 0 2 2 3 2 1 1 3 1 0 3 1 0 0 3 0 1 0 0 1 3 3 0 2 0 1 1 3 3 1 2
 0 2 0 0 3 3 0 2 2 1 3 1 2 0 1 3 1 0 3 1 0 0 3 2 3 2 0 2 1 0 1 2 3 2 1 1 0
 1 2 2 1 1 1 3 1 2 0 2 2 3 1 0 1 2 3 1]
```

COMPUTING CONFUSION MATRIX AND CLASSIFICATION REPORT

```
print(confusion_matrix(y_test,ypred_knn1))
print(classification_report(y_test,ypred_knn1))
```

```
[[127   5   0   0]
 [  4 113   1   0]
 [  0  11 106   3]
 [  0   0   7 123]]
              precision    recall  f1-score   support
```

```
              0       0.97      0.96      0.97       132
              1       0.88      0.96      0.91       118
              2       0.93      0.88      0.91       120
              3       0.98      0.95      0.96       130

       accuracy                           0.94       500
      macro avg       0.94      0.94      0.94       500
   weighted avg       0.94      0.94      0.94       500
```

ACCURACY

```
accuracy_knn = accuracy_score(y_test, ypred_knn1)
print("Accuracy with KNN Classification:", accuracy_knn)
```

```
Accuracy with KNN Classification: 0.938
```

c) i) SVM Classifier with linear kernel

```
from sklearn.svm import SVC
```

```
svc_m1 = SVC(kernel = 'linear', C=10)
svc_m1.fit(x_train,y_train)
```

```
SVC(C=10, kernel='linear')
```

```
print('Train score', svc_m1.score(x_train,y_train))
print('Test score', svc_m1.score(x_test,y_test))
```

```
Train score 0.9833333333333333
Test score 0.964
```

PREDICTED PRICE RANGE FOR TEST DATA

```
ypred_svc_m1 = svc_m1.predict(x_test)
print(ypred_svc_m1)
```

```
[0 2 1 3 1 1 2 0 3 1 0 1 2 3 3 2 3 3 1 0 0 2 1 2 0 1 3 3 2 0 0 0 3 0 1 1 2
 0 3 0 2 3 2 0 2 3 2 1 3 1 3 1 0 0 1 1 1 3 0 0 1 3 3 1 0 0 3 3 1 2 2 2 0 1
 2 0 1 3 2 2 3 2 1 0 1 3 1 3 3 0 3 3 2 1 3 2 2 3 1 1 0 0 1 0 1 3 2 0 1 1 0
 0 3 1 3 2 3 2 0 2 1 3 2 1 3 3 0 2 0 2 3 0 2 2 0 3 1 0 0 2 2 1 2 2 0 0 0 1
 1 2 3 1 1 0 2 2 0 1 0 2 2 3 3 3 1 0 1 2 2 3 3 0 1 0 3 1 1 2 1 0 0 0 0 0 3
 2 0 3 0 0 0 0 1 3 3 1 0 1 1 1 1 2 2 3 3 3 1 2 0 0 0 2 1 1 3 1 1 2 1 1 3 2
 3 0 0 2 1 3 0 1 2 0 2 3 2 0 1 3 3 0 1 3 3 3 0 3 1 2 3 3 2 1 0 3 3 1 3 3 3
```

```
 3 3 0 1 2 2 1 3 0 2 3 2 2 2 1 0 1 0 2 3 1 3 1 1 3 1 2 0 0 3 0 1 2 3 3 3 1
 1 0 1 3 3 0 1 2 2 0 3 3 2 3 2 3 2 0 2 1 1 1 0 0 0 3 3 3 1 0 1 0 1 2 3 0 3
 3 2 1 3 0 0 2 1 3 2 0 1 1 1 1 1 3 2 0 0 3 3 0 3 0 0 2 0 1 2 2 2 3 0 3 2 3
 3 3 3 2 1 1 0 3 1 3 3 0 2 3 2 3 3 3 0 0 2 3 0 0 2 3 2 1 1 2 1 3 0 3 1 2 0
 0 1 0 1 0 1 0 2 2 3 2 1 1 2 1 1 3 1 0 0 3 0 1 0 0 2 3 3 0 2 0 1 1 3 3 1 2
 0 2 0 0 3 3 0 2 2 2 3 1 2 0 1 3 1 0 3 1 0 0 3 2 3 2 0 2 1 0 1 2 3 2 1 1 0
 1 2 2 1 0 1 3 1 2 0 2 2 3 0 0 1 2 3 1]
```

COMPUTING CONFUSION MATRIX AND CLASSIFICATION REPORT

```
cm_svc1 = confusion_matrix(y_test,ypred_svc_m1)
print(cm_svc1)
```

```
[[127   5   0   0]
 [  1 117   0   0]
 [  0   5 110   5]
 [  0   0   2 128]]
```

```
cls_rep_svc1 = classification_report(y_test,ypred_svc_m1)
print(cls_rep_svc1)
```

```
                precision    recall  f1-score   support

             0       0.99      0.96      0.98       132
             1       0.92      0.99      0.96       118
             2       0.98      0.92      0.95       120
             3       0.96      0.98      0.97       130

      accuracy                           0.96       500
     macro avg       0.96      0.96      0.96       500
  weighted avg       0.97      0.96      0.96       500
```

ACCURACY

```
accuracy_svc1 = accuracy_score(y_test, ypred_svc_m1)
print("Accuracy with SVM Classification-Linear kernel:", accuracy_svc1)
```

```
Accuracy with SVM Classification-Linear kernel: 0.964
```

c) ii) SVM Classifier with rbf kernel

```
clf_rbf = SVC(kernel = 'rbf')
clf_rbf.fit(x_train, y_train)
```

```
SVC()
```

```
print('Train Score',clf_rbf.score(x_train,y_train))
print('Test Score',clf_rbf.score(x_test,y_test))
```

```
Train Score 0.9553333333333334
Test Score 0.952
```

PREDICTED PRICE RANGE FOR TEST DATA

```
ypred_svc_m2 =clf_rbf.predict(x_test)
print(ypred_svc_m2)
```

```
[0 2 1 3 1 1 2 0 3 1 0 0 2 3 2 2 3 3 1 0 0 1 1 2 0 1 3 2 2 0 0 0 3 0 1 1 2
 0 3 0 2 3 2 0 2 2 2 1 3 1 3 1 0 0 1 1 1 3 0 0 1 3 3 1 0 0 3 3 1 2 2 2 0 1
 2 0 0 3 2 2 3 2 1 0 1 3 1 3 3 0 3 3 2 1 3 2 2 3 1 1 0 0 1 0 0 3 2 0 1 1 0
 0 3 1 3 2 3 2 0 2 1 3 2 1 3 3 0 3 0 2 3 0 2 2 0 3 1 0 0 2 2 1 2 2 0 0 0 1
 1 2 3 1 1 0 2 2 0 1 0 2 2 3 3 2 1 0 1 2 2 3 3 0 1 0 3 1 1 2 1 0 0 0 0 0 3
 2 0 3 0 0 0 0 1 3 3 1 0 1 1 1 1 2 2 3 3 3 1 2 0 0 0 2 1 1 3 1 0 2 1 1 3 2
 3 0 0 2 1 3 0 1 2 0 2 3 2 0 1 3 3 0 1 3 3 3 0 3 1 2 3 3 2 1 1 3 3 1 3 3 3
 3 3 0 1 2 2 2 3 0 2 3 2 2 2 1 0 1 0 3 3 1 3 1 0 3 1 2 0 0 3 0 1 2 3 3 3 0
 1 0 1 3 3 0 1 2 2 0 3 3 2 3 2 3 2 0 2 1 1 1 0 0 0 2 2 3 1 0 1 0 1 2 3 0 3
 3 2 1 3 0 0 2 1 3 2 0 1 1 1 1 1 3 2 0 0 3 3 0 3 0 0 2 0 1 2 2 2 3 0 3 2 2
 3 3 3 2 1 1 0 3 1 3 3 0 2 3 2 3 3 3 0 0 2 3 0 0 2 3 2 1 1 2 1 3 1 3 1 2 0
 0 1 0 1 0 1 0 2 2 3 2 1 1 2 1 1 3 1 0 0 3 0 1 0 0 1 3 3 0 2 0 1 1 3 3 0 2
 0 2 0 0 3 3 0 2 2 2 3 1 2 0 1 3 1 0 3 1 0 0 3 2 3 2 0 2 1 0 1 2 3 2 1 1 0
 1 2 2 1 1 1 3 1 2 0 3 2 3 1 0 1 2 3 1]
```

COMPUTING CONFUSION MATRIX AND CLASSIFICATION REPORT

```
cm_svc2 = confusion_matrix(y_test,ypred_svc_m2)
print(cm_svc2)
```

```
[[128   4   0   0]
 [  3 115   0   0]
 [  0   6 109   5]
 [  0   0   6 124]]
```

```
cls_rep_rbf = classification_report(y_test,ypred_svc_m2)
print(cls_rep_rbf)
```

```
              precision    recall  f1-score   support

           0       0.98      0.97      0.97       132
           1       0.92      0.97      0.95       118
           2       0.95      0.91      0.93       120
           3       0.96      0.95      0.96       130
```

```
          accuracy                           0.95       500
         macro avg       0.95      0.95      0.95       500
      weighted avg       0.95      0.95      0.95       500
```

ACCURACY

```
accuracy_svc2 = accuracy_score(y_test, ypred_svc_m2)
print("Accuracy with SVM Classification-rbf kernel:", accuracy_svc2)
```

```
Accuracy with SVM Classification-rbf kernel: 0.952
```

d) Decision Tree Classifier

```
from sklearn.tree import DecisionTreeClassifier
```

```
clf = DecisionTreeClassifier()
clf.fit(x_train, y_train)
```

```
DecisionTreeClassifier()
```

```
train_score = clf.score(x_train, y_train)
print("Train score:", train_score)

test_score = clf.score(x_test, y_test)
print("Test score:", test_score)
```

```
Train score: 1.0
Test score: 0.804
```

PREDICTED PRICE RANGE FOR TEST DATA

```
y_pred_dt = clf.predict(x_test)
print(y_pred_dt)
```

```
[0 2 1 3 1 2 2 0 2 1 0 1 3 2 2 2 3 3 1 0 0 1 1 1 0 1 2 2 2 0 0 0 3 0 2 1 2
 0 3 0 2 3 2 0 2 2 1 1 3 1 3 1 0 0 0 0 1 3 0 0 1 3 3 1 0 0 3 3 1 2 2 2 0 1
 3 0 0 3 2 2 3 2 1 0 1 3 2 3 3 0 3 3 2 0 3 2 2 3 1 1 0 0 1 0 0 3 2 0 1 1 0
 0 2 1 2 2 2 2 0 2 1 3 2 1 3 3 0 3 1 2 3 1 2 2 0 3 1 0 0 2 3 1 3 3 0 0 0 1
 2 2 3 1 1 0 2 2 0 1 0 1 2 3 3 3 1 0 1 2 2 3 3 1 1 0 3 2 2 2 1 1 0 0 0 0 3
 2 0 3 0 1 0 0 1 3 3 2 0 1 1 1 1 1 2 2 3 3 1 2 0 0 0 2 1 1 3 1 1 3 1 1 3 2
 3 0 0 1 1 3 0 1 2 0 2 3 2 1 1 3 3 0 2 3 3 3 0 3 1 2 3 3 3 1 1 3 3 0 3 3 3
 3 3 0 1 2 3 1 3 0 1 3 2 3 2 1 0 2 0 2 3 1 3 1 0 3 1 2 0 0 3 0 1 3 3 3 3 0
 0 0 1 3 3 0 1 1 2 0 3 3 2 3 2 3 2 0 2 1 1 1 0 0 1 3 2 3 1 0 1 1 1 3 3 1 3
```

```
 3 2 1 2 0 0 3 1 3 3 0 2 1 2 0 2 3 1 0 0 3 3 0 3 0 0 2 0 0 1 2 2 3 0 2 2 2
 3 3 3 3 1 2 0 3 2 3 3 0 2 3 2 3 3 3 1 0 2 3 0 0 3 3 1 1 1 2 1 2 1 3 1 2 0
 0 1 0 1 0 1 1 2 3 3 2 1 1 3 1 0 3 1 0 1 2 0 1 0 1 1 3 3 0 2 0 1 1 3 3 0 2
 0 2 0 0 3 3 0 2 2 1 3 1 2 0 0 3 1 0 3 1 1 0 3 2 3 2 0 3 0 0 1 2 3 2 1 1 0
 1 2 2 1 1 1 3 2 2 0 2 2 3 1 0 1 2 3 1]
```

COMPUTING CONFUSION MATRIX AND CLASSIFICATION REPORT

```
cm_dt = confusion_matrix(y_test,y_pred_dt)
print(cm_dt)
```

```
[[113  19   0   0]
 [ 12  92  14   0]
 [  0  17  82  21]
 [  0   0  15 115]]
```

```
cls_rep_dt = classification_report(y_test,y_pred_dt)
print(cls_rep_dt)
```

```
                   precision    recall  f1-score   support

               0       0.90      0.86      0.88       132
               1       0.72      0.78      0.75       118
               2       0.74      0.68      0.71       120
               3       0.85      0.88      0.86       130

        accuracy                           0.80       500
       macro avg       0.80      0.80      0.80       500
    weighted avg       0.81      0.80      0.80       500
```

ACCURACY

```
accuracy_dt = accuracy_score(y_test, y_pred_dt)
print("Accuracy with Decision Tree is:", accuracy_dt)
```

```
Accuracy with Decision Tree is: 0.804
```

e) Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier
```

```
clf_rfc = RandomForestClassifier()
clf_rfc.fit(x_train, y_train)
```

```
RandomForestClassifier()
```

```
train_score_rfc = clf_rfc.score(x_train, y_train)
print("Train score:", train_score_rfc)

test_score_rfc = clf.score(x_test, y_test)
print("Test score:", test_score_rfc)
```

```
Train score: 1.0
Test score: 0.804
```

PREDICTED PRICE RANGE FOR TEST DATA

```
y_pred_rfc = clf_rfc.predict(x_test)
print(y_pred_rfc)
```

```
[0 2 1 3 1 2 2 0 3 1 0 1 2 3 2 2 3 3 1 0 0 1 1 2 0 1 3 2 2 0 0 0 3 0 1 1 2
 0 3 0 2 3 2 0 2 2 2 1 3 1 3 1 0 0 1 1 1 2 0 0 0 3 3 1 0 0 3 3 2 2 2 3 0 1
 2 0 0 3 2 2 3 2 1 0 1 3 2 3 3 0 3 3 2 1 3 2 2 3 1 1 0 0 1 0 0 3 2 0 1 1 0
 0 3 1 3 2 3 2 0 2 1 3 2 1 3 3 0 3 0 2 3 0 2 2 0 3 1 0 0 2 3 0 2 2 0 0 0 1
 1 2 3 1 1 0 2 2 0 1 0 2 2 2 2 2 1 0 0 2 2 3 3 1 1 0 3 1 2 2 1 0 0 0 0 0 3
 2 0 3 0 0 0 0 1 3 3 1 0 1 2 1 1 2 2 2 3 3 1 2 0 0 0 2 1 1 3 1 0 2 1 1 3 1
 2 0 0 2 1 2 0 0 2 0 1 3 2 0 1 3 3 0 1 3 2 3 0 3 1 2 3 3 2 1 1 3 3 1 3 3 3
 3 3 0 1 2 2 2 2 0 2 3 2 2 2 1 0 1 0 3 3 1 3 1 0 3 1 2 0 0 3 0 1 2 3 3 3 1
 1 0 1 3 3 0 1 1 2 0 3 3 2 3 1 3 2 0 2 1 1 1 0 0 1 3 2 3 1 0 1 0 1 3 2 0 3
 3 2 1 3 0 0 3 1 3 2 0 1 1 2 1 1 3 1 0 0 3 3 0 3 0 0 2 0 0 2 2 2 3 0 3 2 2
 3 3 3 2 1 2 0 3 1 3 3 0 2 3 2 3 3 3 0 0 2 3 0 0 2 3 2 1 1 2 1 3 1 3 1 2 0
 0 1 0 1 0 2 0 2 2 3 2 1 1 3 1 0 3 1 0 0 3 0 1 0 0 1 3 3 0 2 1 1 1 3 3 1 2
 0 2 0 0 3 3 0 2 2 1 3 1 1 0 1 3 1 0 3 1 0 0 3 2 3 2 0 2 1 0 1 2 3 2 1 0 0
 1 2 2 1 1 1 3 1 2 0 3 2 3 0 0 1 2 3 1]
```

COMPUTING CONFUSION MATRIX AND CLASSIFICATION REPORT

```
cm_rfc = confusion_matrix(y_test,y_pred_rfc)
print(cm_rfc)
```

```
[[124   8   0   0]
 [  9 101   8   0]
 [  0  13  98   9]
 [  0   0  13 117]]
```

```
cls_rep_rfc = classification_report(y_test,y_pred_rfc)
print(cls_rep_rfc)
```

```
                   precision    recall  f1-score   support

               0       0.93      0.94      0.94       132
```

```
              1       0.83      0.86      0.84       118
              2       0.82      0.82      0.82       120
              3       0.93      0.90      0.91       130

       accuracy                           0.88       500
      macro avg       0.88      0.88      0.88       500
   weighted avg       0.88      0.88      0.88       500
```

ACCURACY

```
accuracy_rfc = accuracy_score(y_test, y_pred_rfc)
print("Accuracy with Random forest Classifier is:", accuracy_rfc)
```

```
Accuracy with Random forest Classifier is: 0.88
```

RESULT

```
df1 = pd.read_csv('Downloads/Ajitha/result_minor_project.csv')
df1.head(6)
```

	MODEL	ACCURACY
0	Logistic Regression	61.60%
1	KNN Classification	93.80%
2	SVM Classifier - Linear kernel	96.40%
3	SVM Classifier - rbf kernel	95.20%
4	Decision Tree Classifier	80.40%
5	Random Forest Classifier	88.00%

Hence, SVM Classifier - Linear kernel gives out the most accurate price range for the mobiles from the above tested models with 96.4% accuracy.

```

```