In [3]:

Out[3]:

3

price\_range

Out[7]:

In [8]:

Out[8]:

1000

In [7]: #battery power vs price range

2000 1800

1600 1400

ے 1200 ا 超 1000

800

600

Ó

#checking nan values

dataset.isna().sum()

0

0 0

0

0

plt.figure(figsize=(15,15)) sns.heatmap(corr\_mat,annot=True)

battery\_power

clock\_speed dual\_sim

px\_height px\_width

sc\_h SC\_W talk\_time three\_g touch\_screen

wifi

Out[9]:

price\_range dtype: int64

In [9]: corr\_mat=dataset.corr()

<AxesSubplot:>

battery\_power - 1

dock\_speed -0.011 0.021

In [10]: x = dataset.iloc[:,:-1].values

In [11]: x[0]

In [12]: y[0]

In [14]: x\_train.shape

(1500, 20)

x\_test.shape

sc=StandardScaler()

(500, 20)

Out[12]: 1

Out[14]:

Out[16]:

Out[41]:

Out[22]:

Out[23]:

In [22]: y\_pred1

y = dataset.iloc[: , -1].values

0.000e+00, 1.000e+00])

In [13]: from sklearn.model\_selection import train\_test\_split

In [15]: **from** sklearn.preprocessing **import** StandardScaler

In [41]: **from** sklearn.ensemble **import** RandomForestClassifier

from sklearn.metrics import confusion\_matrix

0

14

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy\_score

0

1.3e+02

ż

classifier2=KNeighborsClassifier(n\_neighbors=39, metric='minkowski')

array([3, 0, 2, 1, 1, 0, 0, 2, 3, 0, 1, 3, 1, 2, 2, 1, 3, 2, 1,

 $0,\ 3,\ 1,\ 2,\ 2,\ 1,\ 1,\ 2,\ 0,\ 3,\ 0,\ 2,\ 2,\ 2,\ 0,\ 2,\ 2,\ 1,\ 1,$ 

2, 3, 3, 3, 2, 1, 0, 1, 0, 2, 1, 2, 0, 3, 0, 0, 2, 0, 1,

2, 0, 3, 3, 3, 1, 3, 3, 3, 0, 0, 0, 2, 2, 3, 0, 0, 1, 2, 3, 0, 0, 2, 3, 2, 0, 1, 1, 1, 1, 0, 3, 2, 1, 2, 1, 1,

3, 1, 2, 2, 2, 1, 0, 2, 3, 2, 0, 3, 2, 3, 2, 1, 1, 2, 1,

1, 0, 1, 1, 2, 0, 2, 3, 2, 2, 0, 3, 2, 2, 1, 2, 1, 2, 1,

requirements.

3, 2, 1, 2, 3, 2, 2, 0, 0, 0, 1, 0, 1, 3, 0, 1, 3, 3, 2, 2, 3, 2,

2, 2, 2, 1, 3, 1, 1, 0, 2, 2, 2, 2, 2, 1, 2, 3, 2, 0, 1, 0, 2, 1, 3, 1, 2, 1, 0, 2, 3, 3, 3, 0, 3, 0, 2, 2, 1, 2, 2, 2, 2, 2, 3, 2, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 3, 2, 2, 1, 2, 0, 1, 0, 3, 0, 3,

3, 1, 0, 3, 1, 2, 2, 0, 2, 2, 1, 3, 3, 1, 2, 0, 3, 1, 1, 3, 3, 0, 2, 1, 3, 1, 2, 2, 0, 1, 3, 1, 2, 0, 2, 3, 1, 1, 0, 2, 2, 0, 0, 2,

2, 3, 1, 0, 1, 3, 3, 1, 3, 0, 3, 1, 0, 0, 2, 3, 3, 1, 0, 1, 1, 1,

1, 2, 1, 2, 0, 1, 3, 0, 3, 2, 3, 0, 1, 0, 1, 1, 3, 0, 0, 2, 3, 0,

3, 2, 1, 1, 2, 0, 3, 2, 1, 1, 2, 3, 0, 1, 1, 1, 1, 1, 1, 0, 2, 0,  $0,\ 3,\ 1,\ 3,\ 3,\ 3,\ 0,\ 3,\ 2,\ 2,\ 1,\ 3,\ 1,\ 3,\ 3,\ 3,\ 0,\ 2,\ 0,\ 1,\ 1,$ 0, 0, 1, 2, 1, 2, 0, 1, 0, 1, 1, 3, 0, 0, 0, 3, 0, 3, 0, 3, 1, 1, 0, 0, 2, 2, 0, 1, 3, 0, 3, 3, 0, 3, 0, 2, 2, 2, 3, 2, 1, 3, 2, 2, 0, 0, 2, 1, 0, 1, 0, 3, 2, 2, 1, 0, 2, 0, 1, 1, 3, 0, 0, 2, 1, 2, 2, 1, 0, 2, 1, 2, 0, 2, 0, 2, 3, 0, 2, 2, 0, 2, 2, 0, 0, 0, 0, 2, 0, 3, 2, 2, 0, 1, 0, 2, 2, 2, 1, 3, 1, 1, 0, 0, 1, 2, 1, 2, 1, 0, 2, 3, 0, 2, 3, 1, 1, 3, 1, 1, 0, 3, 0, 2, 2, 1], dtype=int64)

Impact of Project on Market

The aim of this project is to forecast the mobile prices so as to minimize the problems faced by the customer. The important reason for predicting Mobile Prices is the link between mobile costs and the Users Budget. This model will help Customers and Sellers to invest in an appropriate devices according to their mentioned

cm=confusion\_matrix(y\_test,y\_pred1)

sns.heatmap(cm, annot=True)

0

accuracy\_score(y\_test,y\_pred1)

classifier2.fit(x\_train,y\_train)

KNeighborsClassifier(n\_neighbors=39)

y\_pred2=classifier2.predict(x\_test)

from sklearn.metrics import accuracy\_score

accuracy\_score(y\_test,y\_pred2)

Conclusion

<AxesSubplot:>

1.2e+02

0.854

y\_pred2

Out[25]:

In [38]:

Out[38]:

In [39]:

In [42]:

Out[42]:

RandomForestClassifier(criterion='entropy', n\_estimators=15)

x\_train=sc.fit\_transform(x\_train)

classifier1.fit(x\_train , y\_train)

x\_test=sc.transform(x\_test)

In [21]: y\_pred1=classifier1.predict(x\_test)

blue

fc four\_g int\_memory m\_dep mobile\_wt n\_cores

2000

ram

3000

sns.boxplot(x='price\_range',y='battery\_power',data=dataset)

<AxesSubplot:xlabel='price\_range', ylabel='battery\_power'>

price\_range

0.011 0.011 -0.042 0.033 0.016 -0.004 0.034 0.0018 -0.03 0.031 0.015-0.0084.000650.03 -0.021 0.053 0.012 -0.011-0.0083 0.2

dual\_sim\_-0.042\_0.035-0.0013 1 -0.0290.0032-0.016-0.022-0.009-0.025-0.017-0.021\_0.014\_0.041-0.012-0.017-0.039-0.014-0.017\_0.023\_0.017

fc -0.0330.00360.000430.029 1 -0.017-0.0290.00180.024-0.013 0.64 -0.01-0.00520.015-0.011-0.0120.00680.0018-0.015 0.02 0.022

four g -0.016 0.013 -0.0430.0032-0.017 1 0.00870.00180.017 -0.03-0.00560.0190.00740.00730.027 0.037-0.047 0.58 0.017 -0.018 0.015

m dep -0.034 0.004-0.014-0.0220.00180.00180.0069 1 0.022-0.00350.026 0.025 0.024-0.00940.025-0.018 0.017-0.0120.00260.0280.0008

int\_memory -0.004 0.041 0.0065-0.016-0.0290.0087 1 0.0069-0.034-0.028-0.033 0.01-0.00830.033 0.038 0.012-0.00280.00940.027 0.007 0.044

mobile wt -0.00180.00860.012-0.009 0.024-0.017-0.034 0.022 1 -0.019 0.0190.000949e-05-0.00260.034-0.0210.00620.0016-0.0140.000410.03

px\_height -0.015-0.00690.015-0.021-0.01-0.019-0.01-0.0250.00094.00690.018

px width -0.00840.0420.00950.014-0.00520.00740.00830.024 9e-05 0.0240.0042 0.51 1

array([8.420e+02, 0.000e+00, 2.200e+00, 0.000e+00, 1.000e+00, 0.000e+00,

7.000e+00, 6.000e-01, 1.880e+02, 2.000e+00, 2.000e+00, 2.000e+01, 7.560e+02, 2.549e+03, 9.000e+00, 7.000e+00, 1.900e+01, 0.000e+00,

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x , y , test\_size=0.25 , random\_state=0)

classifier1 = RandomForestClassifier(n\_estimators = 15 , criterion = 'entropy')

array([3, 0, 2, 1, 2, 0, 0, 3, 3, 1, 0, 3, 0, 2, 3, 0, 3, 2, 2, 1, 0, 0,

3, 1, 2, 2, 3, 1, 3, 1, 1, 0, 1, 0, 2, 3, 0, 0, 3, 3, 3, 1, 3, 2, 1, 3, 0, 1, 3, 1, 1, 3, 0, 3, 0, 2, 1, 1, 0, 3, 3, 1, 3, 2, 1, 2, 3, 2, 1, 2, 3, 2, 1, 0, 0, 3, 2, 2, 2, 2, 2, 3, 3, 0, 0, 0, 2, 0, 2, 3, 1, 2, 2, 0, 0, 3, 3, 3, 0, 3, 1, 2, 2, 1, 3, 1, 2, 3, 2, 3, 3, 0, 0, 1, 3, 3, 0, 1, 1, 0, 1, 3, 2, 2, 1, 2, 1, 1, 0, 2, 1, 3, 2, 3, 3, 3, 1, 0, 1, 1, 2, 2, 3, 0, 3, 0, 0, 2, 0, 1, 1, 1, 1, 3, 0, 0, 3, 1, 3, 2, 1, 3, 1, 2, 3, 3, 2, 1, 0, 3, 2, 2, 2, 3, 0, 1, 2, 3, 0, 2, 1, 0, 1, 2, 1, 2, 0, 2, 3, 1, 1, 0, 2, 3, 0, 1, 2, 2, 0, 3, 3, 2, 1, 1, 3, 3, 3, 0, 0, 0, 2, 3, 3, 0, 0, 1, 3, 2, 3, 3, 3, 0, 0, 2, 2, 3, 2, 0, 2, 0, 0, 0, 3, 3, 0, 2, 2, 1, 1, 0, 2, 3, 3, 0, 0, 1, 3, 3, 1, 3, 0, 3, 1, 1, 0, 1, 3, 2, 2, 0, 0, 1, 2, 3, 2, 1, 3, 2, 1, 0, 3, 3, 2, 1, 3, 3, 2, 2, 1, 0, 2, 1, 1, 0, 0, 2, 2, 2, 2, 0, 1, 3, 0, 1, 2, 3, 0, 2, 0, 0, 1, 3, 0, 0, 2, 3, 1, 1, 0, 1, 0, 3, 0, 3, 3, 2, 3, 0, 1, 2, 0, 1, 2, 0, 1, 0, 3, 1, 1, 3, 1, 0, 1, 3, 0, 3, 2, 2, 0, 1, 2, 0, 2, 2, 1, 2, 2, 1, 0, 2, 0,  $0,\ 3,\ 1,\ 2,\ 3,\ 2,\ 2,\ 0,\ 3,\ 2,\ 2,\ 1,\ 2,\ 2,\ 3,\ 3,\ 3,\ 0,\ 2,\ 0,\ 3,\ 0,$ 1, 1, 2, 2, 1, 3, 1, 1, 0, 1, 0, 3, 0, 1, 1, 3, 0, 3, 0, 1, 2, 0, 1, 0, 3, 1, 0, 1, 3, 0, 3, 3, 0, 3, 1, 3, 2, 1, 3, 2, 0, 3, 3, 2, 0, 0, 3, 0, 1, 1, 1, 3, 2, 3, 2, 0, 3, 0, 0, 1, 2, 0, 0, 3, 2, 2, 2, 3, 0, 1, 1, 2, 0, 2, 0, 3, 3, 0, 2, 3, 0, 3, 1, 1, 1, 2, 2, 1, 3, 2, 2, 0, 2, 0, 3, 3, 2, 1, 0, 3, 1, 2, 0, 0, 1, 3, 0, 3, 0, 0, 1, 2, 0, 1, 2, 0, 2, 1, 1, 2, 0, 3, 0, 1, 3, 2], dtype=int64)

- 120

- 100

- 20

sc h - -0.03 -0.003-0.029-0.012-0.011 0.027 0.038 -0.025-0.0340.00030.0049 0.06 0.022 0.016 1

n\_cores - -0.03 0.036-0.00570.025-0.013 -0.03 -0.0280.00350.019 1 0.00120.00690.024 0.0049 0.00310.026 0.013 -0.015 0.024 -0.01 0.0044

pc -0.031 -0.01-0.00520.017 0.64 -0.00560.033 0.026 0.019-0.0012 1 -0.0180.00420.0290.0049-0.024 0.015-0.00130.00870.00540.034

sc w -0.0210.000640.00740.017-0.012 0.037 0.012 -0.018-0.021 0.026 -0.024 0.043 0.035 0.036 0.51 1 -0.023 0.031 0.013 0.035 0.039

talk time -0.053 0.014-0.011-0.0390.00680.0470.00280.0170.00620.013 0.015-0.0110.00670.011-0.017-0.023 1 -0.043 0.017 -0.03 0.022

three\_g -0.012 -0.03 -0.046-0.0140.0018 0.58-0.00940.0120.0016-0.0150.00130.0310.000350.016 0.012 0.031-0.043 1 0.0140.00430.024

wifi -0.00830.022-0.024 0.023 0.02 -0.018 0.007 -0.0280.000410.01 0.00540.052 0.03 0.023 0.026 0.035 -0.03 0.00430.012 1 0.019

0.2 0.021-0.00660.017 0.022 0.015 0.0440.00085-0.03 0.00440.034 0.15 0.17 0.92 0.023 0.039 0.022 0.024 -0.03 0.019

touch screen -0.011 0.01 0.02 -0.017-0.015 0.017-0.0270.00260.014 0.024-0.00870.022-0.0016-0.03 -0.02 0.013 0.017 0.014 1 0.012 -0.03

0.51 -0.02 0.06 0.043-0.011-0.031 0.022 0.052 0.15

0.00410.022 0.0350.00610.000349.0016 0.03 0.17

0.51 -0.017 0.012 -0.02 0.026 0.023

0.021 0.035 0.0036 0.013 0.041 0.004 0.0086 0.036 - 0.01 - 0.0069 0.042 0.026 - 0.00 0.006 10.014 - 0.03 0.01 - 0.022 0.021

1 0.0018.000430.0430.0065-0.0140.012-0.00570.00520.0150.00950.0034-0.0290.00740.011-0.046 0.02 -0.0240.0066

- 0.8

- 0.6

- 0.4

- 0.2

- 0.0

4000

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

import numpy as np import pandas as pd import matplotlib.pyplot as plt import seaborn as sns dataset=pd.read\_csv("C:\\Users\swapna\\Downloads\\mobile\_price.csv") dataset.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\_h sc\_w talk\_time three\_g touch\_screen wifi price\_range 0 2.2 0.6 0 842 0 1 0 7 188 2 ... 20 756 2549 9 19

Data Science with Python Minor Project

Gudala Swapna

7660032577

b182507@rgukt.ac.in

0

1

1

1

0

1

1

0

1 0

ram

256.000000

1207.500000

1

0

0

0

1

2

2

2

1

sc\_h

12.306500

4.213245

5.000000

9.000000

12.000000

16.000000

19.000000

3 ... 1 1021 1 0.5 1 0 53 0.7 136 905 1988 2631 17 3 7 1 2 1 1 2 5 ... 2 9 563 0.5 1 41 0.9 145 1263 1716 2603 11 6 ... 615 1 2.5 0 0 10 8.0 131 1216 1786 2769 16 8 11 4 1821 1 1.2 0 13 1 44 0.6 141 2 ... 1208 1212 1411 8 2 15 5 rows × 21 columns

dataset.describe() mobile wt px\_width blue clock\_speed dual\_sim fc m\_dep n\_cores ... px\_height battery\_power four\_g int\_memory 2000.000000 2000.0000 count 2000.000000 2000.000000 2000.000000 2000.000000 2000.000000 2000.000000 2000.000000 2000.000000 ... 1238.518500 0.4950 1.522250 0.509500 4.309500 0.521500 32.046500 0.501750 140.249000 4.520500 ... mean 439.418206 0.5001 0.816004 0.500035 4.341444 0.499662 18.145715 0.288416 35.399655 2.287837 ... 443.780811 std

Out[4]: 0.0000 501.000000 0.500000 0.000000 0.000000 0.000000 2.000000 0.100000 80.000000 1.000000 ... 0.000000 min **25**% 851.750000 0.0000 0.700000 0.000000 1.000000 0.000000 16.000000 0.200000 109.000000 3.000000 ... 282.750000 50%

1226.000000 0.0000 1.500000 1.000000 3.000000 1.000000 32.000000 0.500000 141.000000 4.000000 ...

1615.250000 1.000000 170.000000 75% 1.0000 2.200000 7.000000 1.000000 48.000000 0.800000 7.000000 ...

2000.000000 2000.000000 2000.000000 2000.000000 200 645.108000 1251.515500 2124.213000 432.199447 1084.732044 500.000000 874.750000 564.000000 1247.000000 2146.500000

1998.000000 1.0000 3.000000 1.000000 19.000000 1.000000 64.000000 1.000000 200.000000 max

947.250000 1633.000000 3064.500000

8 rows × 21 columns

sns.pointplot(y='int\_memory', x='price\_range', data=dataset)

<AxesSubplot:xlabel='price\_range', ylabel='int\_memory'>

 $8.000000 \dots 1960.000000 1998.000000 3998.000000$ In [5]: #internal memory vs price range

35

34

33 E₁32

31 30

0 price\_range In [6]: #ram vs price range sns.jointplot(x='ram', y='price\_range', data=dataset, color='red', kind='kde') <seaborn.axisgrid.JointGrid at 0x275fd4b26a0> Out[6]: