Data Science with Python Minor Project

Gudala Swapna
b182507@rgukt.ac.in
7660032577
Computer Science and Engineering

Project Statement:

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

Code:

In [3]:
```python
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()
```

Out[3]:

	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 [4]:
```python
dataset.describe()
```

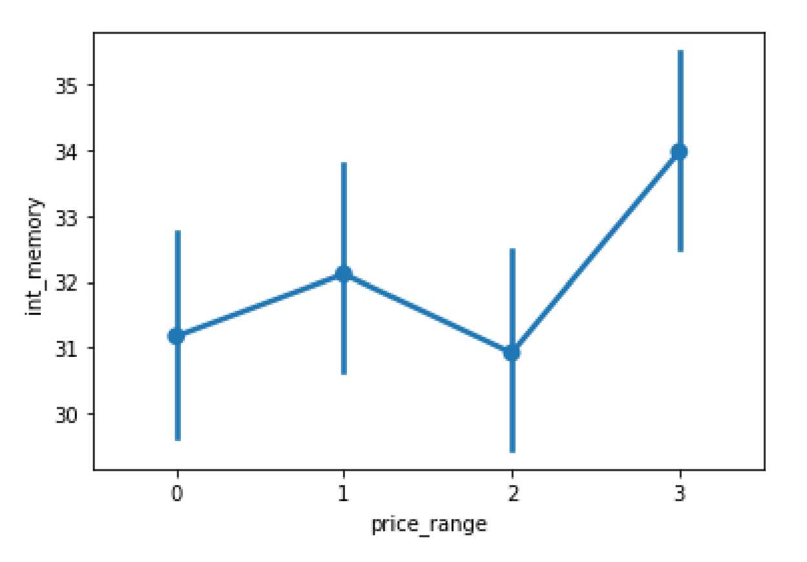
Out[4]:

	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
count	2000.000000	2000.0000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	...	2000.000000	2000.000000	2000.000000	2000.000000
mean	1238.518500	0.4950	1.522250	0.509500	4.309500	0.521500	32.046500	0.501750	140.249000	4.520500	...	645.108000	1251.515500	2124.213000	12.306500
std	439.418206	0.5001	0.816004	0.500035	4.341444	0.499662	18.145715	0.288416	35.399655	2.287837	...	443.780811	432.199447	1084.732044	4.213245
min	501.000000	0.0000	0.500000	0.000000	0.000000	0.000000	2.000000	0.100000	80.000000	1.000000	...	0.000000	500.000000	256.000000	5.000000
25%	851.750000	0.0000	0.700000	0.000000	1.000000	0.000000	16.000000	0.200000	109.000000	3.000000	...	282.750000	874.750000	1207.500000	9.000000
50%	1226.000000	0.0000	1.500000	1.000000	3.000000	1.000000	32.000000	0.500000	141.000000	4.000000	...	564.000000	1247.000000	2146.500000	12.000000
75%	1615.250000	1.0000	2.200000	1.000000	7.000000	1.000000	48.000000	0.800000	170.000000	7.000000	...	947.250000	1633.000000	3064.500000	16.000000
max	1998.000000	1.0000	3.000000	1.000000	19.000000	1.000000	64.000000	1.000000	200.000000	8.000000	...	1960.000000	1998.000000	3998.000000	19.000000

8 rows × 21 columns

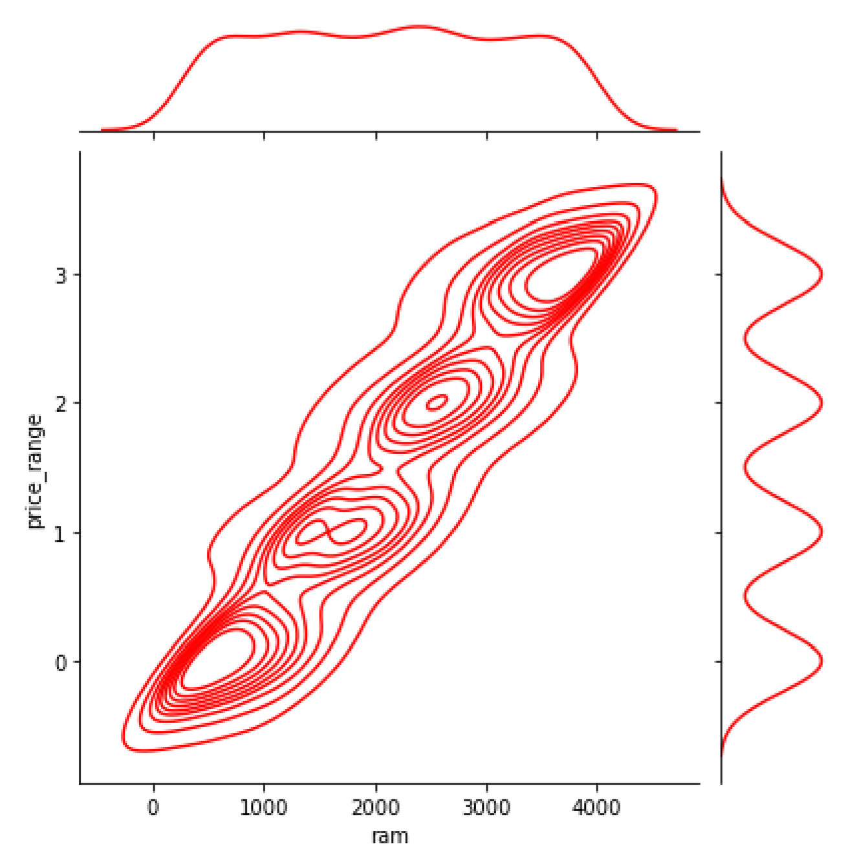
In [5]:
```python
#internal memory vs price range
sns.pointplot(y='int_memory',x='price_range',data=dataset)
```

Out[5]: <AxesSubplot:xlabel='price_range', ylabel='int_memory'>



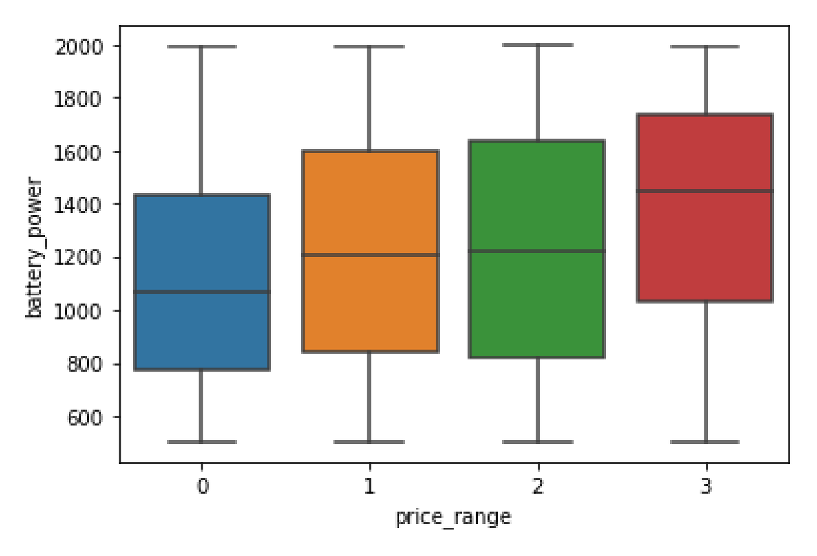
In [6]:
```python
#ram vs price range
sns.jointplot(x='ram',y='price_range',data=dataset,color='red',kind='kde')
```

Out[6]: <seaborn.axisgrid.JointGrid at 0x275fd4b26a0>



In [7]:
```python
#battery power vs price range
sns.boxplot(x='price_range',y='battery_power',data=dataset)
```

Out[7]: <AxesSubplot:xlabel='price_range', ylabel='battery_power'>

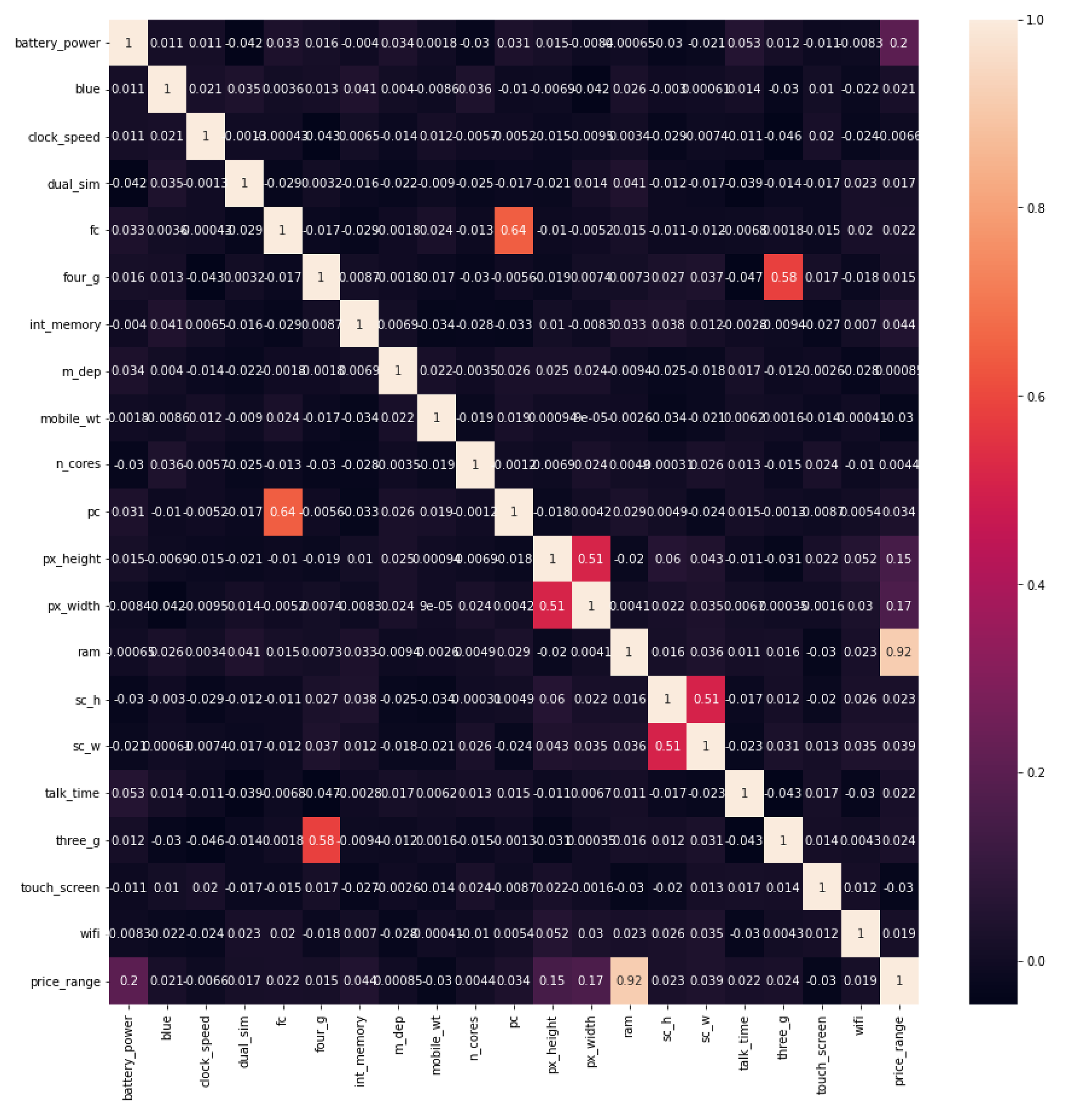


In [8]:
```python
#checking nan values
dataset.isna().sum()
```

Out[8]:
```
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
```

In [9]:
```python
corr_mat=dataset.corr()
plt.figure(figsize=(15,15))
sns.heatmap(corr_mat,annot=True)
```

Out[9]: <AxesSubplot:>



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

In [11]:
```python
x[0]
```

Out[11]:
```
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,
       0.000e+00, 1.000e+00])
```

In [12]:
```python
y[0]
```

Out[12]: 1

In [13]:
```python
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=0)
```

In [14]:
```python
x_train.shape
```

Out[14]: (1500, 20)

In [16]:
```python
x_test.shape
```

Out[16]: (500, 20)

In [15]:
```python
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x_train=sc.fit_transform(x_train)
x_test=sc.transform(x_test)
```

In [41]:
```python
from sklearn.ensemble import RandomForestClassifier
classifier1 = RandomForestClassifier(n_estimators = 15 , criterion = 'entropy')
classifier1.fit(x_train , y_train)
```

Out[41]: RandomForestClassifier(criterion='entropy', n_estimators=15)

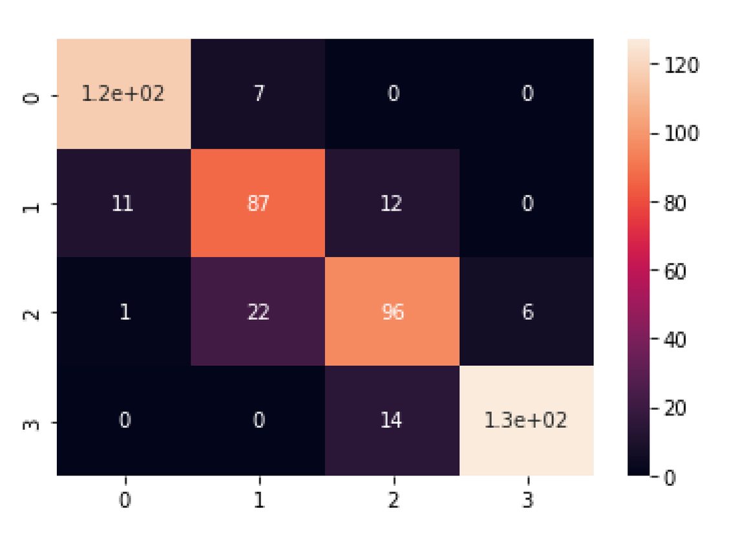
In [21]:
```python
y_pred1=classifier1.predict(x_test)
```

In [22]:
```python
y_pred1
```

Out[22]:
```
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, 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)
```

In [23]:
```python
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_pred1)
sns.heatmap(cm,annot=True)
```

Out[23]: <AxesSubplot:>



In [25]:
```python
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred1)
```

Out[25]: 0.854

In [38]:
```python
from sklearn.neighbors import KNeighborsClassifier
classifier2=KNeighborsClassifier(n_neighbors=39,metric='minkowski')
classifier2.fit(x_train,y_train)
```

Out[38]: KNeighborsClassifier(n_neighbors=39)

In [27]:
```python
y_pred2=classifier2.predict(x_test)
```

In [39]:
```python
y_pred2
```

Out[39]:
```
array([3, 0, 2, 1, 1, 0, 0, 2, 3, 0, 1, 3, 1, 2, 2, 1, 3, 2, 1, 0, 1, 0,
       3, 2, 1, 2, 3, 2, 2, 0, 0, 0, 1, 0, 1, 3, 0, 1, 3, 3, 2, 2, 3, 2,
       0, 3, 1, 2, 2, 1, 1, 2, 0, 3, 0, 2, 2, 2, 0, 2, 2, 1, 1, 2, 2, 2,
       2, 2, 2, 1, 3, 1, 1, 0, 2, 2, 2, 2, 2, 1, 2, 3, 2, 0, 1, 0, 2, 1,
       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,
       2, 3, 3, 3, 2, 1, 0, 1, 0, 2, 1, 2, 0, 3, 0, 0, 2, 0, 1, 1, 1, 0,
       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, 0, 3, 3, 3, 1, 3, 3, 3, 3, 0, 0, 0, 2, 2, 3, 0, 0, 1, 2, 1, 3,
       2, 3, 0, 0, 2, 3, 2, 0, 1, 1, 1, 1, 0, 3, 2, 1, 2, 1, 1, 1, 0, 2,
       2, 3, 1, 0, 1, 3, 3, 1, 3, 0, 3, 1, 0, 0, 2, 3, 3, 1, 0, 1, 1, 1,
       3, 1, 2, 2, 2, 1, 0, 2, 3, 2, 0, 3, 2, 3, 2, 1, 1, 2, 1, 1, 0, 0,
       1, 2, 1, 2, 0, 1, 3, 0, 3, 2, 3, 0, 1, 0, 1, 1, 3, 0, 0, 2, 3, 0,
       1, 0, 1, 1, 2, 0, 2, 3, 2, 2, 0, 3, 2, 2, 1, 2, 1, 2, 1, 3, 1, 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, 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, 2, 0,
       3, 2, 2, 0, 1, 0, 2, 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)
```

In [42]:
```python
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred2)
```

Out[42]: 0.61

Conclusion

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 requirements.