Predicting the Price range of Mobile Phones

Mohd Asif Siddique

September 24, 2020

1. Introduction/Business Problem:

Mobile Phones or more precisely smartphones are becoming more and more popular. There are around 5.5 billion unique smartphone users in the world, vary from a very low prices to very high (expensive) ones. And these numbers are increasing day by day. So, lots and lots of new companies are being created, with different categories of users in their mind. Each brands have distinct stories and motives, they all want to survive and stay as long as possible in the market. Because, the more they stick the more lucrative they become. But, a mobile phone has many features (or precisely components) to look at, it's performance, battery, camera, ram, price, etc., But for users perspective it’s a price, and from brands it’s features and price. The price of the phone will let the companies to stay competitive and survive for long. Now, how will the brands decide and select the price that is what our model will do. Based, on the prices of thousands of models it'll predict the competitive price range.

2. Data Description:

The Data that I'll be using has many features such as battery, bluetooth, performance, ram, and much more, which we'll be looking at soon. I got this data from Kaggle, datasets. So, as I've mentioned the dataset has many features, let's look at them more closely, the dataset has 20 columns, each containing different features. Most of the features are integers with few as floats, so our doesn't have object values (we don't have to perform encoding here). Of 20 features the dataset has 2000 instances with different features. Out of 20 columns our target variable is price range which takes values from (0: Low cost, 1: Medium cost, 2: High cost, 3: Very high cost).
Let's look at an example suppose the mobile has a following values(battery:1021, bluetooth: 1, clock_speed: 0.2, four_g: 1, ram: 2631, and so on, then it'll have a price range of 2).
In test set of the there isn’t a target variable, so I split the train data into training and validations.

3. Methodology

3.1 Gaining insights

In this we’ll going to perform an exploratory analysis on our data, the analysis will give us the relations, insights, and similarities in our data. First let’s look at the few instances of our data.

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	pc	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	2	20	756	2549	9	7	19	0	0	1	1
1	1021	1	0.5	1	0	1	53	0.7	136	3	6	905	1988	2631	17	3	7	1	1	0	2
2	563	1	0.5	1	2	1	41	0.9	145	5	6	1263	1716	2603	11	2	9	1	1	0	2
3	615	1	2.5	0	0	0	10	0.8	131	6	9	1216	1786	2769	16	8	11	1	0	0	2
4	1821	1	1.2	0	13	1	44	0.6	141	2	14	1208	1212	1411	8	2	15	1	1	0	1

As we can see our data dozens of features, before going for any graphical representations, we'll gain more insights.

Now, we'll have look at the columns and their data types.

```
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   battery_power   2000 non-null   int64
 1   blue            2000 non-null   int64
 2   clock_speed     2000 non-null   float64
 3   dual_sim        2000 non-null   int64
 4   fc              2000 non-null   int64
 5   four_g          2000 non-null   int64
 6   int_memory      2000 non-null   int64
 7   m_dep           2000 non-null   float64
 8   mobile_wt       2000 non-null   int64
 9   n_cores         2000 non-null   int64
 10  pc              2000 non-null   int64
 11  px_height       2000 non-null   int64
 12  px_width        2000 non-null   int64
 13  ram             2000 non-null   int64
 14  sc_h            2000 non-null   int64
 15  sc_w            2000 non-null   int64
 16  talk_time       2000 non-null   int64
 17  three_g         2000 non-null   int64
 18  touch_screen    2000 non-null   int64
 19  wifi            2000 non-null   int64
 20  price_range     2000 non-null   int64
dtypes: float64(2), int64(19)
```

There are 20 columns with 2000 instances each, two of the following columns i.e., dual_sum, and m_dep are float values others are integers.

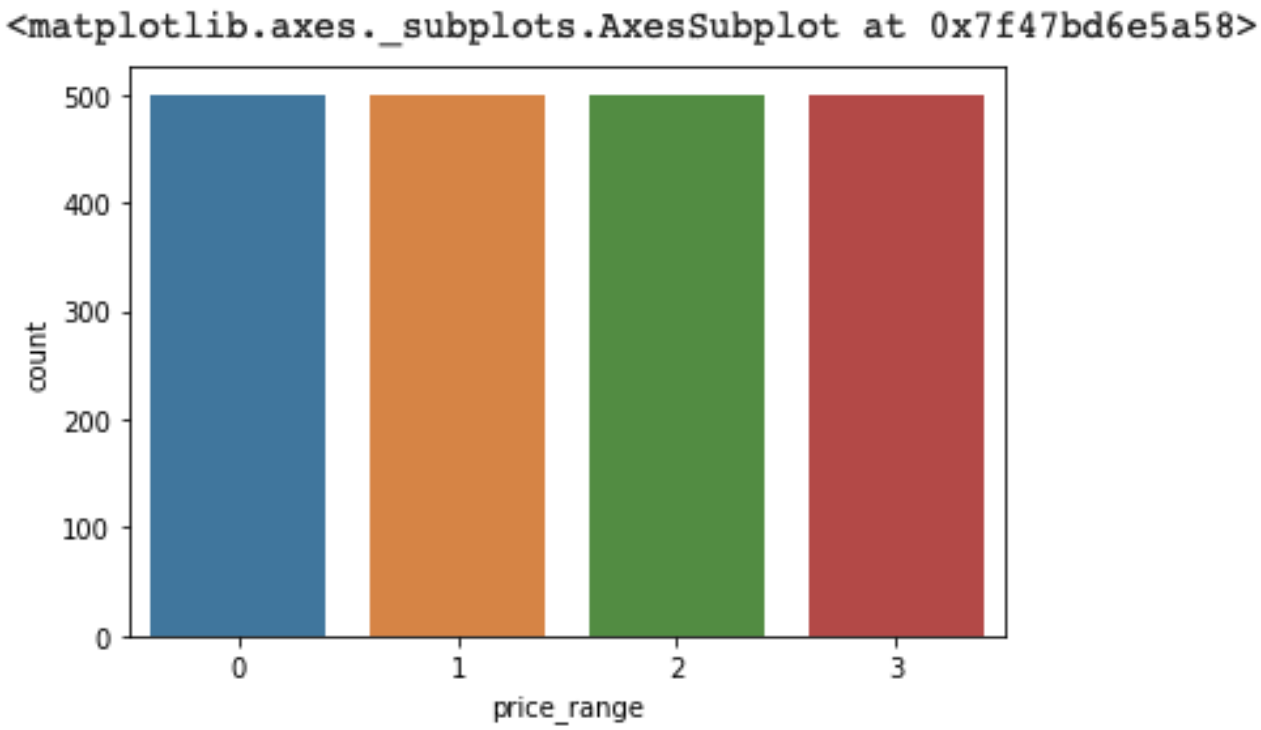
	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	pc	px_height	px_width	ram	sc_h	sc_w	talk_time	three_g	touch_screen	wifi	price_range
count	2000.000000	2000.0000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	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	9.916500	645.108000	1251.515500	2124.213000	12.306500	5.767000	11.011000	0.761500	0.503000	0.507000	1.500000
std	439.418206	0.5001	0.816004	0.500035	4.341444	0.499662	18.145715	0.288416	35.399655	2.287837	6.064315	443.780811	432.199447	1084.732044	4.213245	4.356398	5.463955	0.426273	0.500116	0.500076	1.118314
min	501.000000	0.0000	0.500000	0.000000	0.000000	0.000000	2.000000	0.100000	80.000000	1.000000	0.000000	0.000000	500.000000	256.000000	5.000000	0.000000	2.000000	0.000000	0.000000	0.000000	0.000000
25%	851.750000	0.0000	0.700000	0.000000	1.000000	0.000000	16.000000	0.200000	109.000000	3.000000	5.000000	282.750000	874.750000	1207.500000	9.000000	2.000000	6.000000	1.000000	0.000000	0.000000	0.750000
50%	1226.000000	0.0000	1.500000	1.000000	3.000000	1.000000	32.000000	0.500000	141.000000	4.000000	10.000000	564.000000	1247.000000	2146.500000	12.000000	5.000000	11.000000	1.000000	1.000000	1.000000	1.500000
75%	1615.250000	1.0000	2.200000	1.000000	7.000000	1.000000	48.000000	0.800000	170.000000	7.000000	15.000000	947.250000	1633.000000	3064.500000	16.000000	9.000000	16.000000	1.000000	1.000000	1.000000	2.250000
max	1998.000000	1.0000	3.000000	1.000000	19.000000	1.000000	64.000000	1.000000	200.000000	8.000000	20.000000	1960.000000	1998.000000	3998.000000	19.000000	18.000000	20.000000	1.000000	1.000000	1.000000	3.000000

Above table describes our data by giving mean, median, quartiles, deviations, and etc.

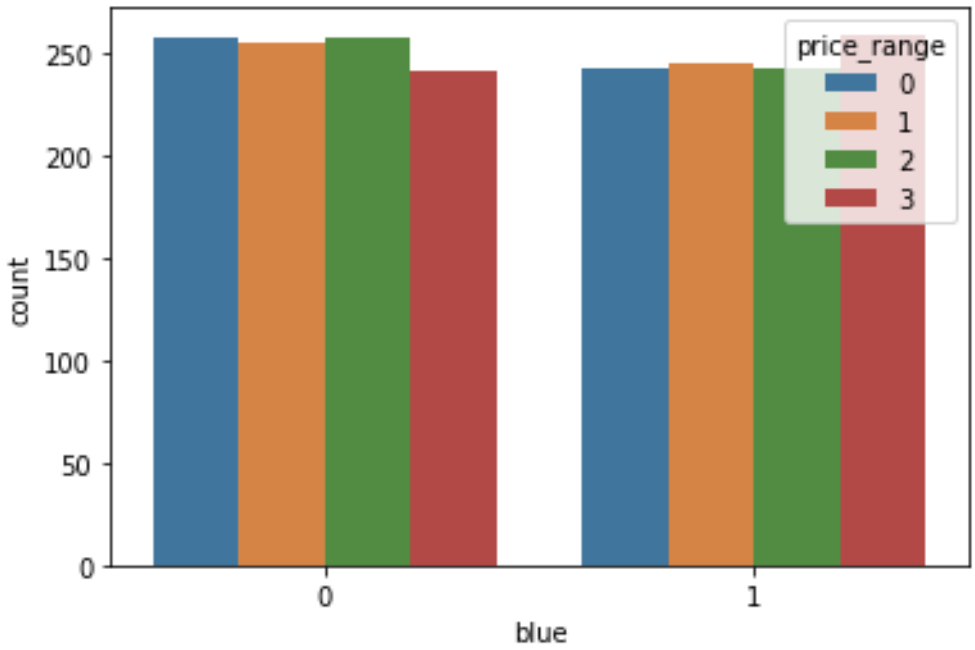
3.2 Exploratory Data Analysis

First, we'll look at the count plots:

A - count plot of price_range:



<matplotlib.axes._subplots.AxesSubplot at 0x7f47bee985c0>



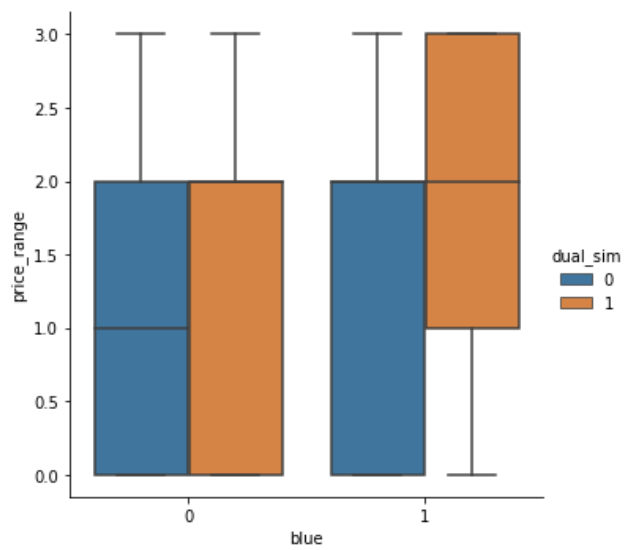
B - count plot of bluetooth (price_range as hue):

Aforementioned plots show the counts, in our data we have 500 instances in each price range.

C - Relationship of bluetooth vs Price_range (dual_sim as hue)

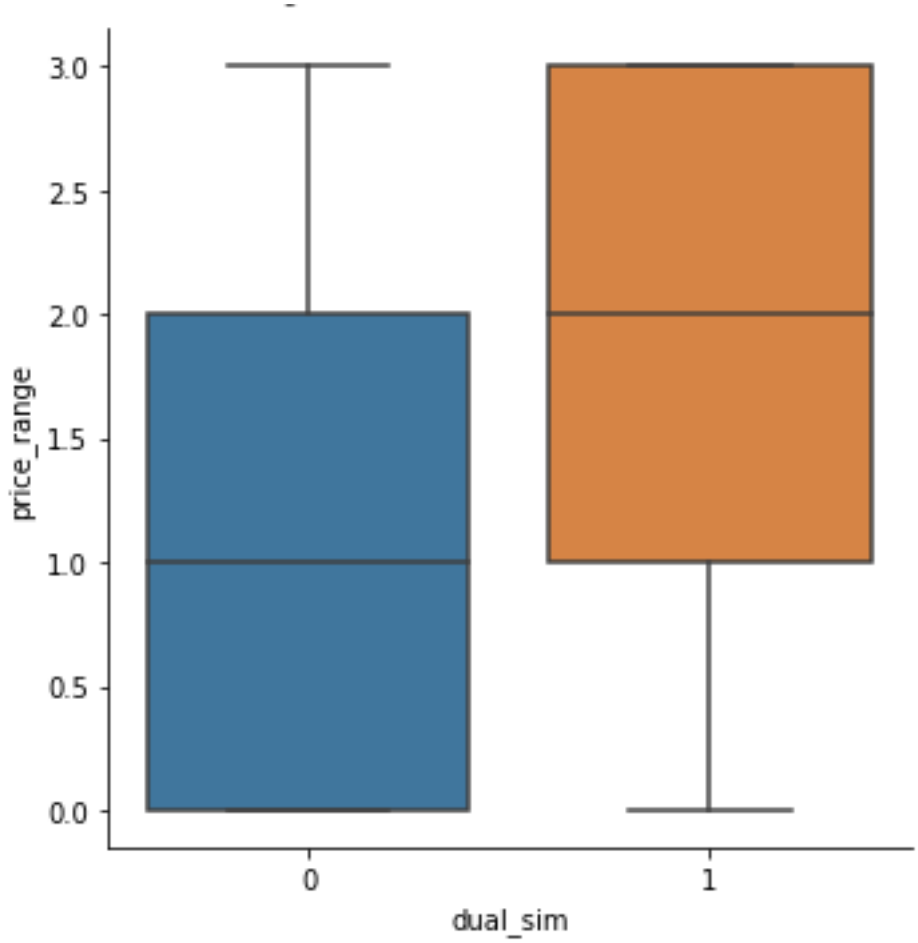
Below plot shows the relationship between the bluetooth, price_range, and dual_sim. So if a phone have a bluetooth and dual_sim it'll sell for more than others.

<seaborn.axisgrid.FacetGrid at 0x7f47b6e191d0>



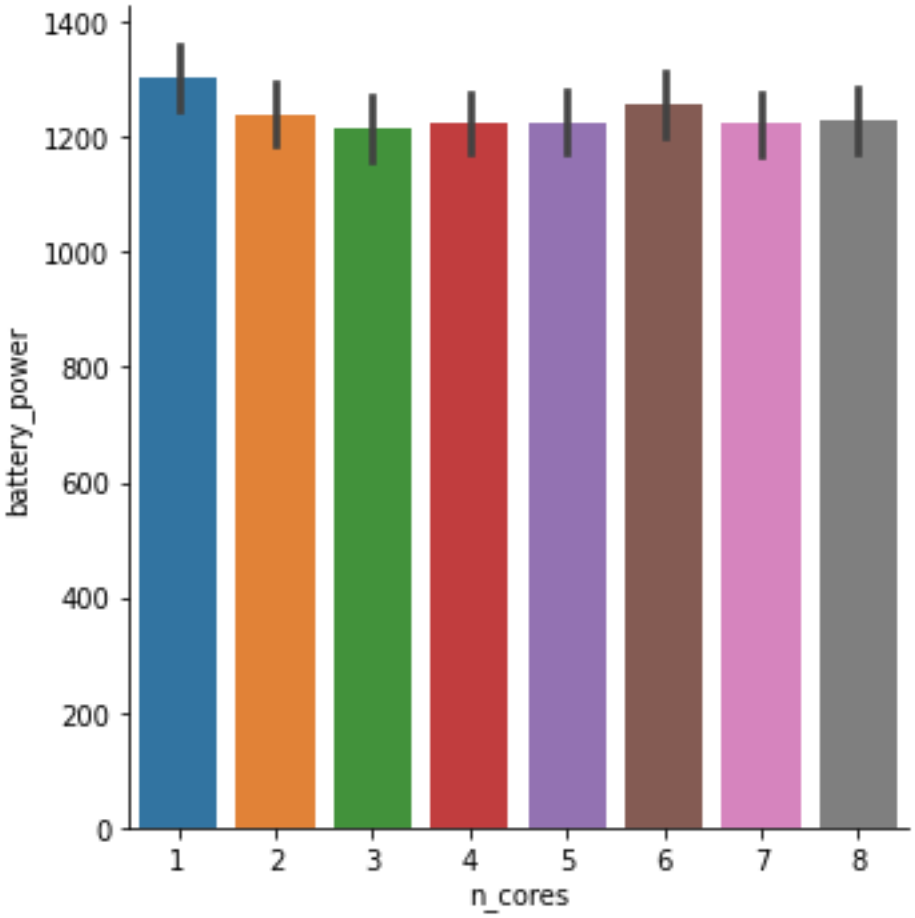
D - Relationship of dual_sim vs price_range:

Here we can see that the phones with dual_sim are expensive than the one with single sim.



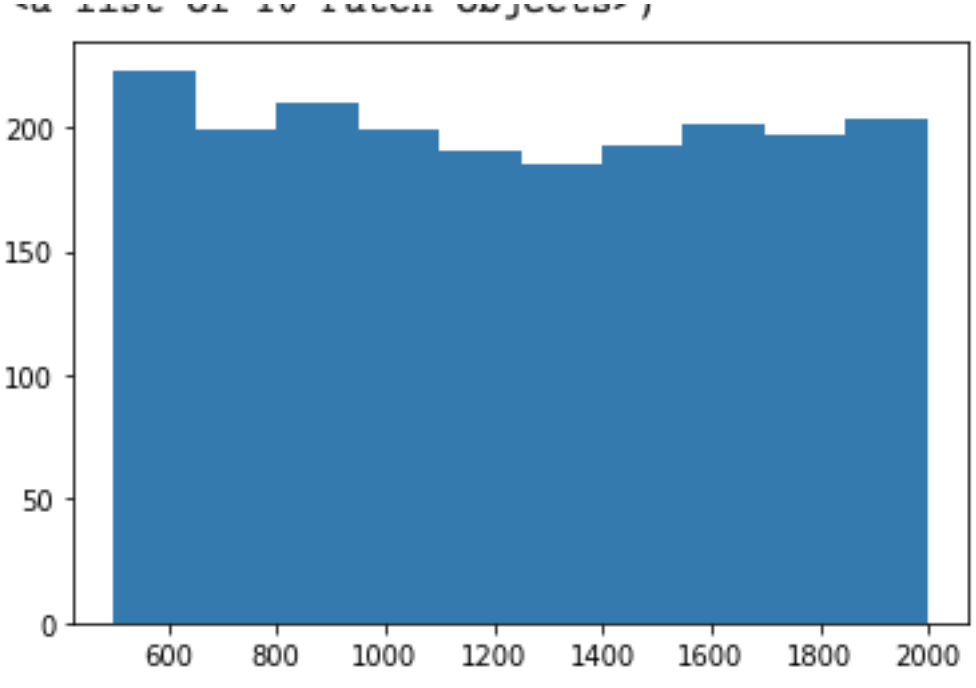
E - Bar graph n_cores vs battery_power:

The following Bar graph shows the number of cores vs the battery_power. We can see that it don’t vary much.



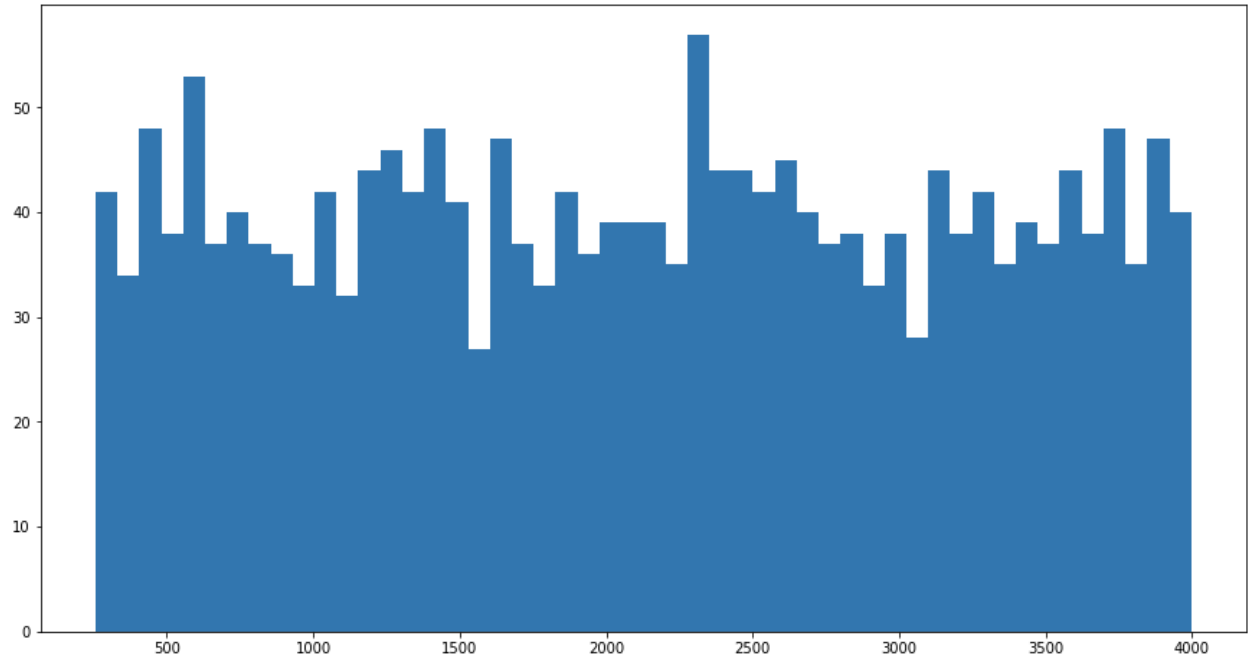
F - Histogram of battery_power:

Below histogram shows the numbers of phones within the specified range.



G - Histogram of ram:

Below histogram represent the number of phones with specified range.



H - Let’s look at the correlation graph which will show the correlation between the variables:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	pc	px_height	px_width	ram	sc_h	sc_w	talk_time	three_g	touch_screen	wifi	price_range
battery_power	1.000000	0.011252	0.011482	-0.041847	0.033334	0.015665	-0.004004	0.034085	0.001844	-0.029727	0.031441	0.014901	-0.008402	-0.000653	-0.029959	-0.021421	0.052510	0.011522	-0.010516	-0.008343	0.200723
blue	0.011252	1.000000	0.021419	0.035198	0.003593	0.013443	0.041177	0.004049	-0.008605	0.036161	-0.009952	-0.006872	-0.041533	0.026351	-0.002952	0.000613	0.013934	-0.030236	0.010061	-0.021863	0.020573
clock_speed	0.011482	0.021419	1.000000	-0.001315	-0.000434	-0.043073	0.006545	-0.014364	0.012350	-0.005724	-0.005245	-0.014523	-0.009476	0.003443	-0.029078	-0.007378	-0.011432	-0.046433	0.019756	-0.024471	-0.006606
dual_sim	-0.041847	0.035198	-0.001315	1.000000	-0.029123	0.003187	-0.015679	-0.022142	-0.008979	-0.024658	-0.017143	-0.020875	0.014291	0.041072	-0.011949	-0.016666	-0.039404	-0.014008	-0.017117	0.022740	0.017444
fc	0.033334	0.003593	-0.000434	-0.029123	1.000000	-0.016560	-0.029133	-0.001791	0.023618	-0.013356	0.644595	-0.009990	-0.005176	0.015099	-0.011014	-0.012373	-0.006829	0.001793	-0.014828	0.020085	0.021998
four_g	0.015665	0.013443	-0.043073	0.003187	-0.016560	1.000000	0.008690	-0.001823	-0.016537	-0.029706	-0.005598	-0.019236	0.007448	0.007313	0.027166	0.037005	-0.046628	0.584246	0.016758	-0.017620	0.014772
int_memory	-0.004004	0.041177	0.006545	-0.015679	-0.029133	0.008690	1.000000	0.006886	-0.034214	-0.028310	-0.033273	0.010441	-0.008335	0.032813	0.037771	0.011731	-0.002790	-0.009366	-0.026999	0.006993	0.044435
m_dep	0.034085	0.004049	-0.014364	-0.022142	-0.001791	-0.001823	0.006886	1.000000	0.021756	-0.003504	0.026282	0.025263	0.023566	-0.009434	-0.025348	-0.018388	0.017003	-0.012065	-0.002638	-0.028353	0.000853
mobile_wt	0.001844	-0.008605	0.012350	-0.008979	0.023618	-0.016537	-0.034214	0.021756	1.000000	-0.018989	0.018844	0.000939	0.000090	-0.002581	-0.033855	-0.020761	0.006209	0.001551	-0.014368	-0.000409	-0.030302
n_cores	-0.029727	0.036161	-0.005724	-0.024658	-0.013356	-0.029706	-0.028310	-0.003504	-0.018989	1.000000	-0.001193	-0.006872	0.024480	0.004868	-0.000315	0.025826	0.013148	-0.014733	0.023774	-0.009964	0.004399
pc	0.031441	-0.009952	-0.005245	-0.017143	0.644595	-0.005598	-0.033273	0.026282	0.018844	-0.001193	1.000000	-0.018465	0.004196	0.028984	0.004938	-0.023819	0.014657	-0.001322	-0.008742	0.005389	0.033599
px_height	0.014901	-0.006872	-0.014523	-0.020875	-0.009990	-0.019236	0.010441	0.025263	0.000939	-0.006872	-0.018465	1.000000	0.510664	-0.020352	0.059615	0.043038	-0.010645	-0.031174	0.021891	0.051824	0.148858
px_width	-0.008402	-0.041533	-0.009476	0.014291	-0.005176	0.007448	-0.008335	0.023566	0.000090	0.024480	0.004196	0.510664	1.000000	0.004105	0.021599	0.034699	0.006720	0.000350	-0.001628	0.030319	0.165818
ram	-0.000653	0.026351	0.003443	0.041072	0.015099	0.007313	0.032813	-0.009434	-0.002581	0.004868	0.028984	-0.020352	0.004105	1.000000	0.015996	0.035576	0.010820	0.015795	-0.030455	0.022669	0.917046
sc_h	-0.029959	-0.002952	-0.029078	-0.011949	-0.011014	0.027166	0.037771	-0.025348	-0.033855	-0.000315	0.004938	0.059615	0.021599	0.015996	1.000000	0.506144	-0.017335	0.012033	-0.020023	0.025929	0.022986
sc_w	-0.021421	0.000613	-0.007378	-0.016666	-0.012373	0.037005	0.011731	-0.018388	-0.020761	0.025826	-0.023819	0.043038	0.034699	0.035576	0.506144	1.000000	-0.022821	0.030941	0.012720	0.035423	0.038711
talk_time	0.052510	0.013934	-0.011432	-0.039404	-0.006829	-0.046628	-0.002790	0.017003	0.006209	0.013148	0.014657	-0.010645	0.006720	0.010820	-0.017335	-0.022821	1.000000	-0.042688	0.017196	-0.029504	0.021859
three_g	0.011522	-0.030236	-0.046433	-0.014008	0.001793	0.584246	-0.009366	-0.012065	0.001551	-0.014733	-0.001322	-0.031174	0.000350	0.015795	0.012033	0.030941	-0.042688	1.000000	0.013917	0.004316	0.023611
touch_screen	-0.010516	0.010061	0.019756	-0.017117	-0.014828	0.016758	-0.026999	-0.002638	-0.014368	0.023774	-0.008742	0.021891	-0.001628	-0.030455	-0.020023	0.012720	0.017196	0.013917	1.000000	0.011917	-0.030411
wifi	-0.008343	-0.021863	-0.024471	0.022740	0.020085	-0.017620	0.006993	-0.028353	-0.000409	-0.009964	0.005389	0.051824	0.030319	0.022669	0.025929	0.035423	-0.029504	0.004316	0.011917	1.000000	0.018785
price_range	0.200723	0.020573	-0.006606	0.017444	0.021998	0.014772	0.044435	0.000853	-0.030302	0.004399	0.033599	0.148858	0.165818	0.917046	0.022986	0.038711	0.021859	0.023611	-0.030411	0.018785	1.000000

I - Correlation of price_range:

Now, I’ll specifically look at the correlation of price_range variable with others. It’ll describe the similarities between price_range variable to other. Correlation score ranges between (-1 to 1) 1 means strongly related, -1 means opposite, and 0 mean not related.

	price_range
battery_power	0.200723
blue	0.020573
clock_speed	-0.006606
dual_sim	0.017444
fc	0.021998
four_g	0.014772
int_memory	0.044435
m_dep	0.000853
mobile_wt	-0.030302
n_cores	0.004399
pc	0.033599
px_height	0.148858
px_width	0.165818
ram	0.917046
sc_h	0.022986
sc_w	0.038711
talk_time	0.021859
three_g	0.023611
touch_screen	-0.030411
wifi	0.018785
price_range	1.000000

Above table contains the scores of correlation, and we can see not all variables are related to the price_range. This is quite a good metrics to not take into account irrelevant variables. We will remove the variables with negative score later.

3.3 Predictive Modelling

In supervised Machine Learning we have two types of models Classification and Regression. Here, I'll be using the Classification model, because the later one is used to predict the output on a continuous scale i.e., any real value. And in Classification we classify the outputs into classes.

The question comes here is, why am I using Classification? Well, I could have used the Regression if I've to predict the actual prices of the phones, rather the definition here is a bit distinct i.e., the prediction of output won't be an actual price, instead a range of classes. The classes are 0: low cost, 1: Medium cost, 2: High cost, 3: Very high cost. I hope the doubts have been cleared.

A - Preprocessing:

Before feeding the data to algorithm I'll remove the irrelevant data, and then normalize the data i.e., scale the data. And then assigning the values to the variables *X_train*, *X_val*, *y_train*. Fortunately, the dataset don't have any null values.

Removing the irrelevant variables is quite easy I've directly dropped the variables using pandas *dataframe.drop*() function. After dropping the unnecessary variables I assigned values to variables for training and evaluation. For that I've created variables *X_train*, and assign it the values of data frame except *price_range* which will be given to *y_train*. Then made another variable *X_val* and given it the values of validation. Then, further made *X_val,* and *y_val* for training set for validation purpose.

For scaling purpose I've had use of *Scikit-Learn's StandardScaler*() class. Then using that scaled the values of *X_train, and X_val.*

After going through the preprocessing phase now, we now have a data that is preprocessed and clean.

B - Feeding the data to Machine Learning models.

Here I've used the following Classification model with different parameters:

I. Decision Tree:

Decision Tree classifier divides the dataset into smaller and smaller subsets based on distinct criteria. Once the division completes then it predicts the class by following the tree.

II. Random Forest:

This is also called an Ensemble classifier, what id does is it uses the same hyper parameters as Decision Tree Classifier but adds an additional randomness to the model while growing.

III: Support Vector Machine:

Like above mentioned models this one as well is one the most famous classifier in Machine Learning. Support vector machines separates the classes by specifying a line between (which sometimes called as street).

IV: Neural Network:

A neural network is a network or circuit of neurons, or in a modern sense, an artificial neural network, composed of artificial neurons or nodes — WIKIPEDIA. Here's I used one offered by scikit_learn i.e., MLPClassifier.

I've used these four models of Machine Learning to get to the solution. And I've already speak briefly about the problem and why I'm not going towards the realm of Regression.

By tweaking and tuning the hyperparameters of the mentioned models, what accuracies I've got and the evaluations discussed in the next section. i.e., result.

4. Result

4.1 Accuracies:

So, what does accuracy tell us? It's the number of correct predictions over total number of samples.

A question comes here is, do accuracy score evaluate good here? Yes, because our dataset is not skewed (i.e. the distribution of the target is same on all the four classes).

Below is the accuracy table:

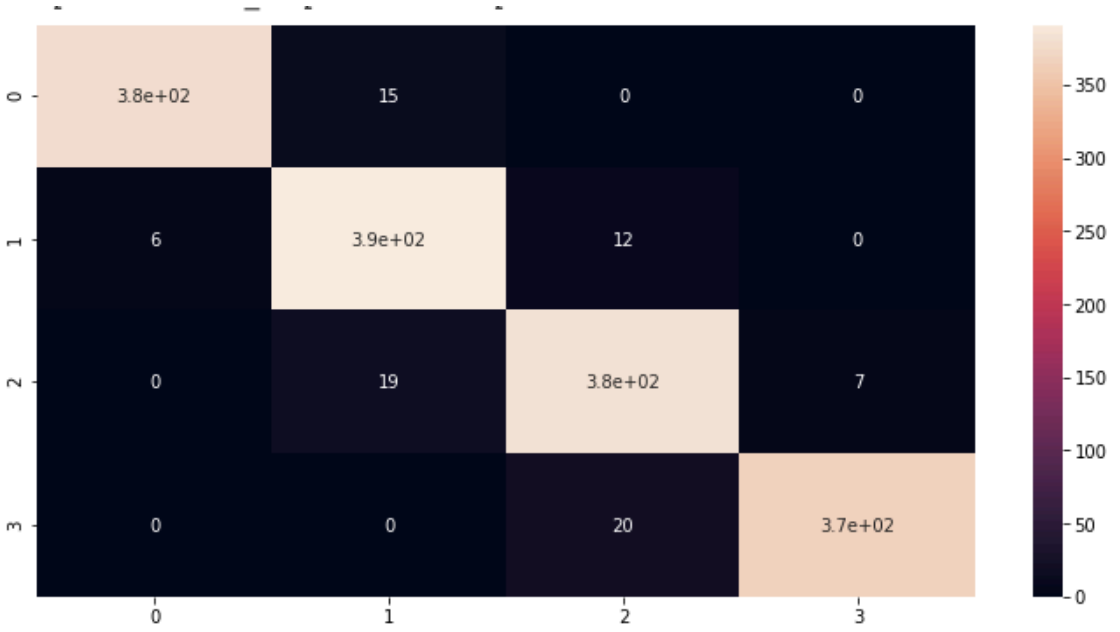
Models	Decision Tree Classifier	Random Forest Classifier	Support Vector Machine	Multi Perceptron Classifier
Training Accuracy	95.06	100.0	97.06	97.31
Val Accuracy	85.75	91.0	96.25	96.25

Accuracy table

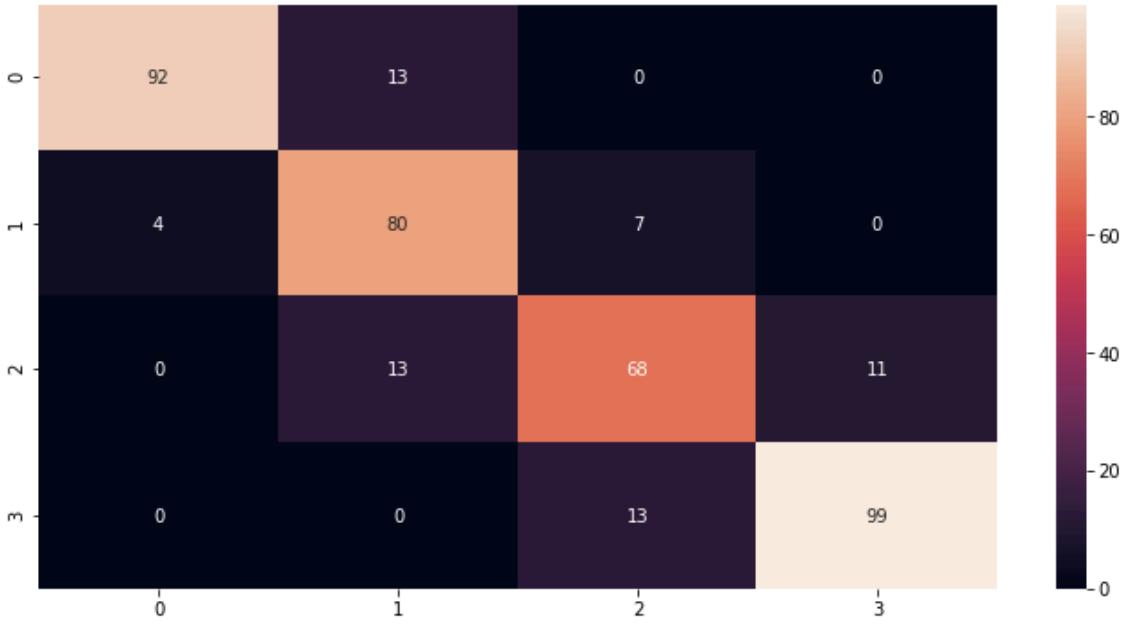
Above table shows the accuracies of our models on training and validation set. I've tuned the parameter using *GridSearchCV* and *RandomizedSearchCV* both function do the same task but differently. From the table conclusion is quite easy i.e., Support Vector Machine and Multi Perceptron Classifier would be the desired model to go for.

Now let's look at the confusion matrix, it's also quite a good way to see the models performance:

Decision Tree Classifier Confusion matrix:

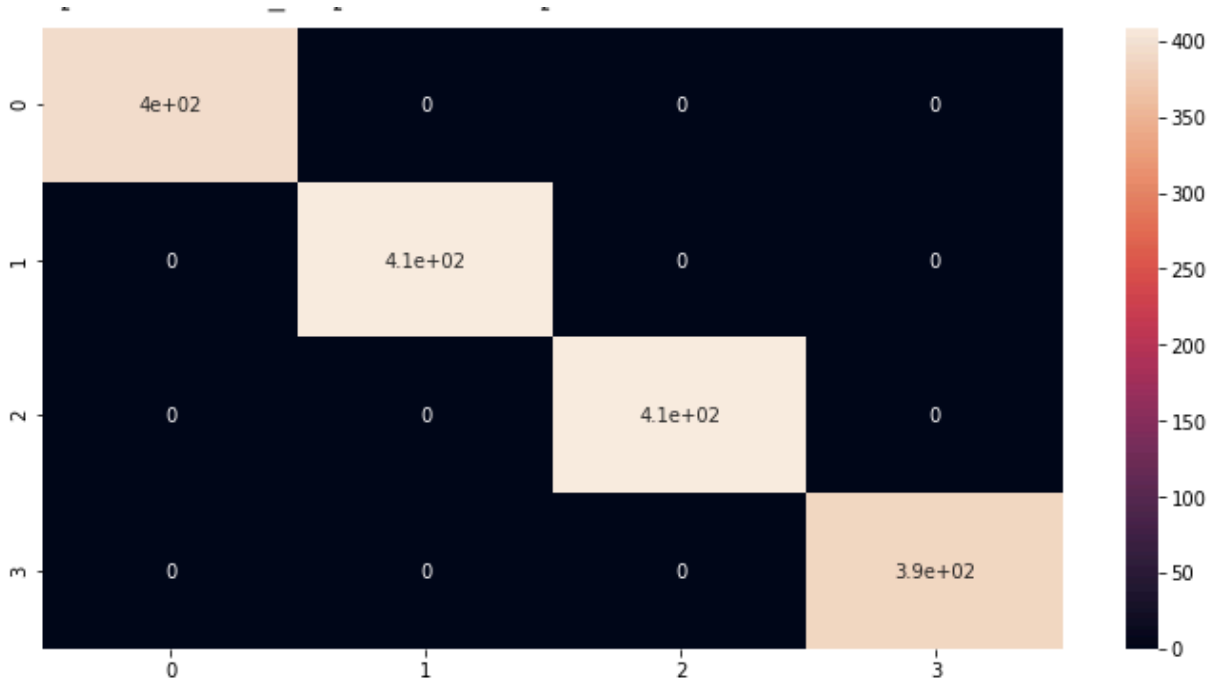


Decision Tree Training Data Predictions Matrix

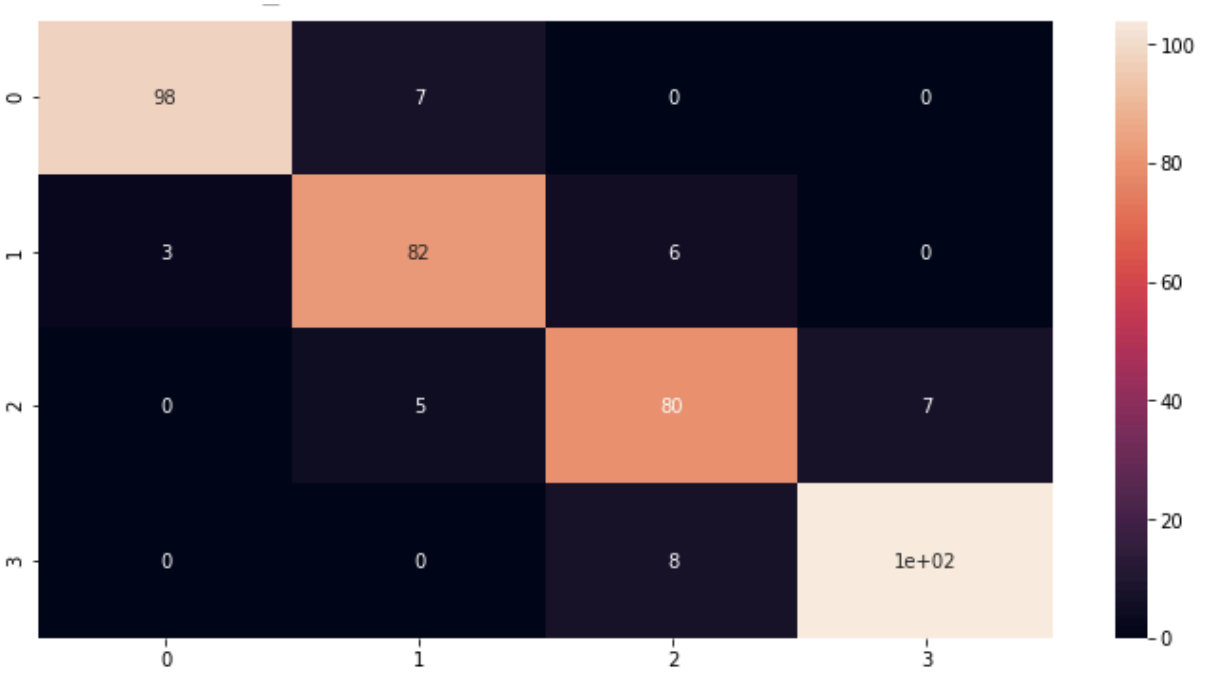


Decision Tree Validation Data Predictions Matrix

Random Forest Classifier Confusion matrix

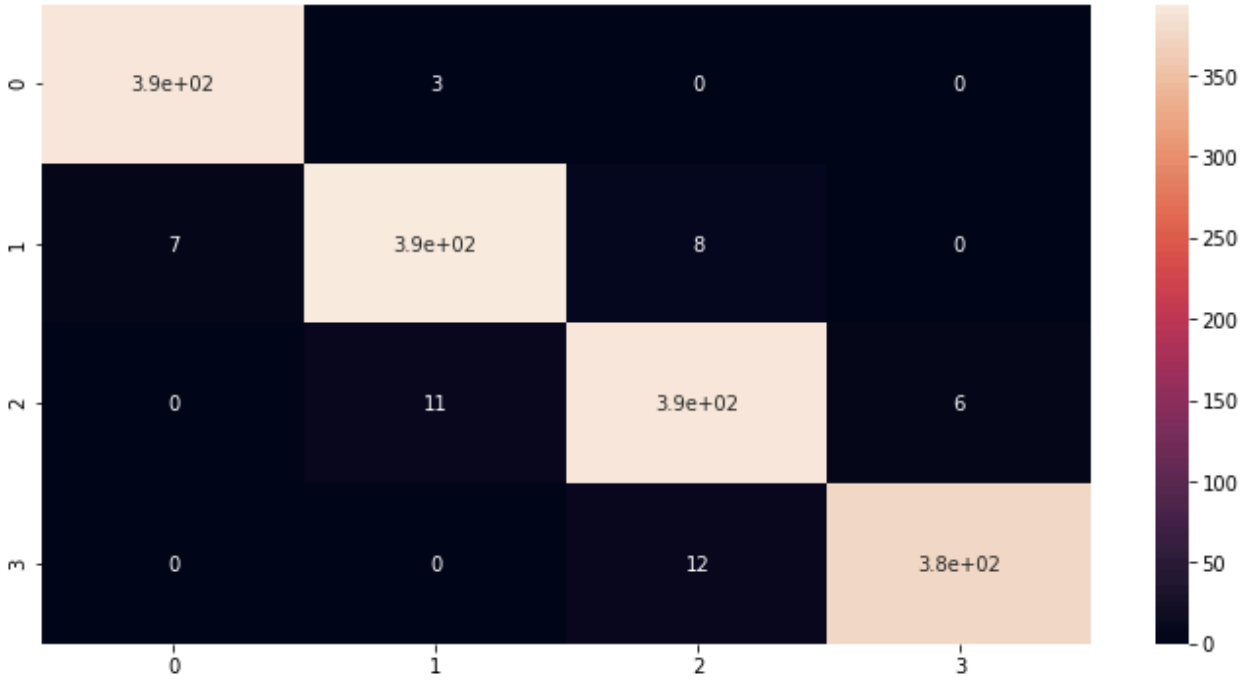


Random Forest Classifier Training Predictions Confusion Matrix

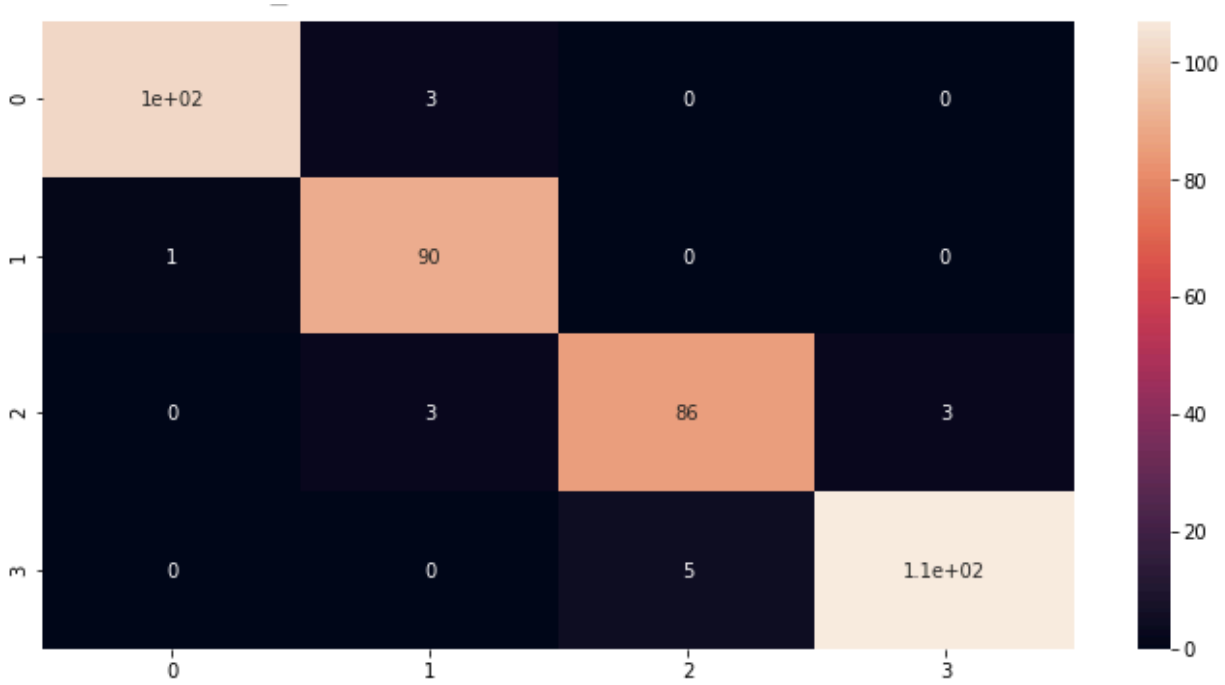


Random Forest Classifier Validation Predictions Confusion Matrix

Support Vector Machine Confusion Matrix

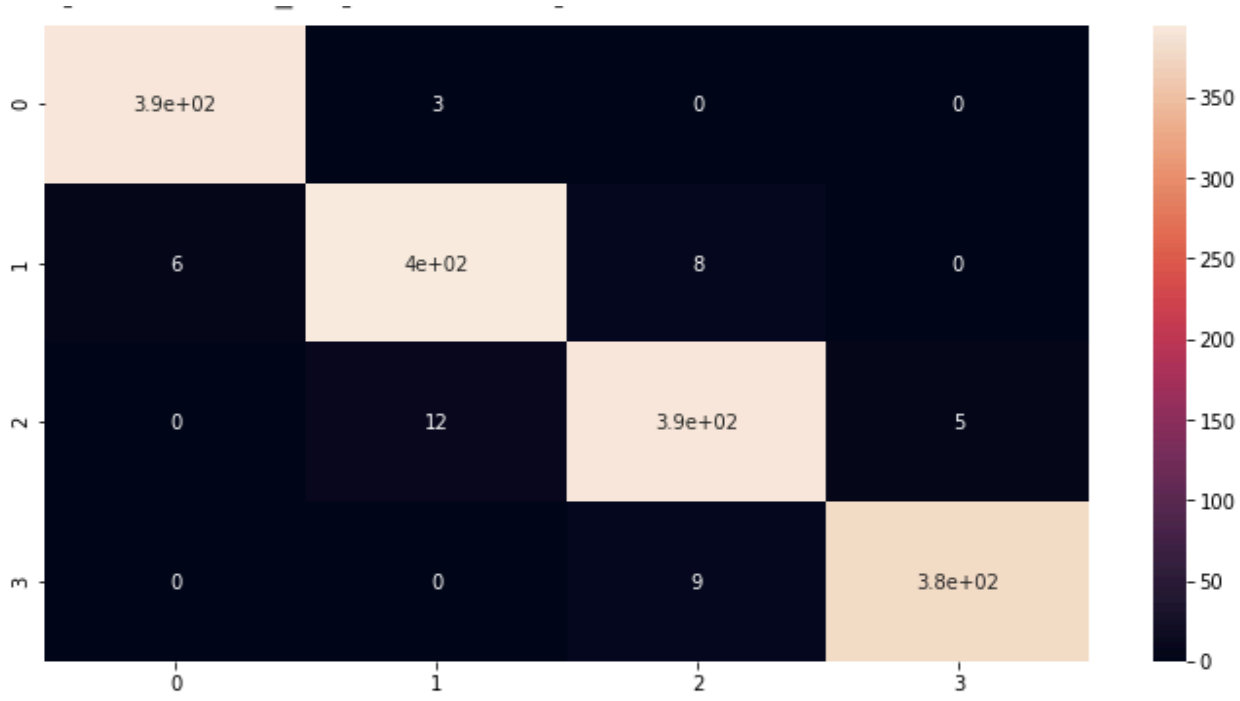


Support Vector Machine Training Predictions Confusion Matrix

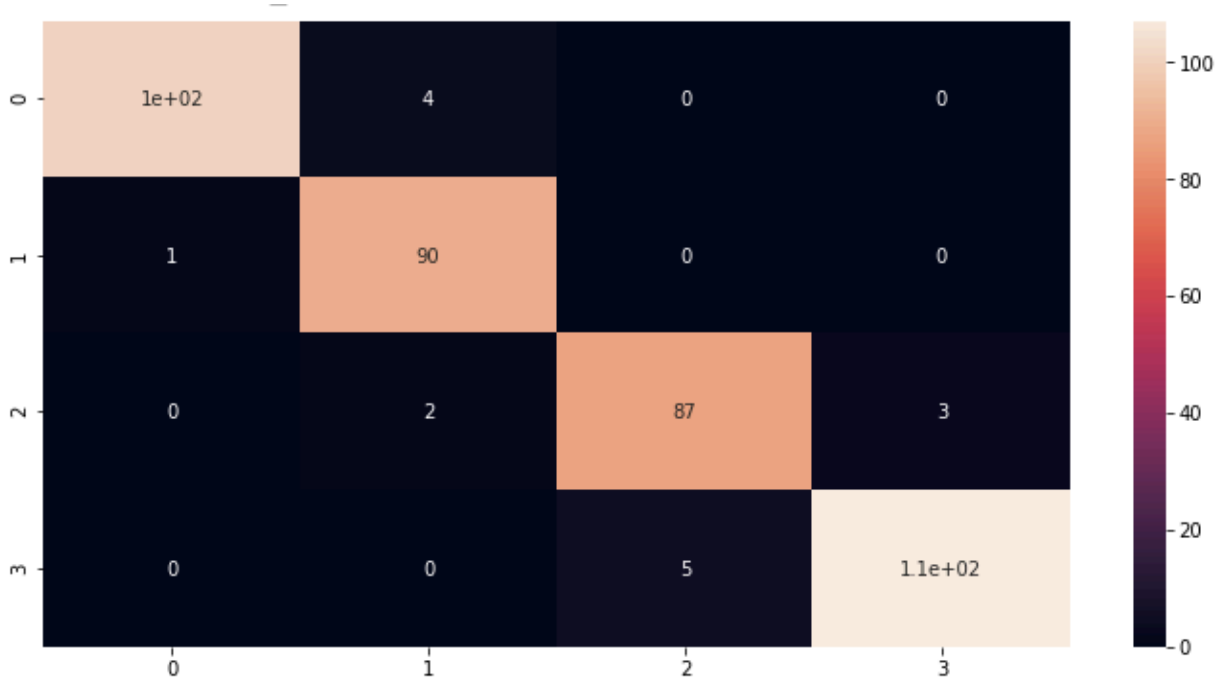


Support Vector Machine Validation Predictions Confusion Matrix

Multi Perceptron Classifier Confusion Matrix



Multi Perceptron Classifier Training Prediction Confusion Matrix



Multi Perceptron Classifier Validation Prediction Confusion Matrix

Confusion matrix is a great way to visualise the predictions done by our model, it gives the number of positives and negatives. In confusion matrix the diagonal values are the correct predictions. From above confusion matrix it's easy to interpret that I don't have many false predictions, i.e., what I've wanted.

So, The accuracies I've got are above 90% i.e., quite good. There's always room for an improvement and I'm sure here as well.

4.3 Discussion

Let's go through the whole process, first I got the data, then preprocessed it, later split and transformed the data, further applied Machine Learning models and fine tuned them, and finally visualised the confusion matrix and classification report. The problem I've got was to predict the Mobile Phones price range given the features, and there are several features whose impact are greater than others, as I've shown that in correlation table. The accuracies are above par and the predictions given by my model can be trusted.

5. Conclusion

I studied and analysed Mobile Phones price range given its features, I visualised the data and performed an exploratory analysis to gain insights. I've employed Classification model as the problem was better suited for that. Then fine tuned the models to get the best hyper parameters. These models can be very useful for brands at deciding the prices of their next smartphones to stay competitive in the market. And can help the companies to make the Mobile phones with features that matter the most.