PREDICTION OF HANDSET TYPE

"Prediction of Price Range of Mobile Handset according to its specifications by Machine Learning using Python3"

Presented by :-

Monishita Ghosh (Roll. No. :- 11100118037)

Golam Kibria (Roll. No. :- 11100118041)

Amit Baskey (Roll. No. :- 11100118051)

Raktim Midya (Roll. No. :- 11100118030)

Apratim Sarkar (Roll. No. :- 11100118048)

Date :- 21st August,2019

Table of Contents

“Acknowledgements”

We have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. We would like to extend our sincere thanks to all of them. Also, we would like to thank our project guide ***Titas Roy Chowdhury Sir*** *for his kind efforts.*

We are highly indebted to ***Globsyn Finishing School*** *for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.*

We would like to express our gratitude towards our college teachers and friends for their kind co-operation and encouragement which help us in completion of this project.

We would like to express our special gratitude and thanks to industry persons for giving us such attention and time.

Our thanks and appreciations also go to my colleague in developing the project and people who have willingly helped us out with their abilities.

Abstract

To predict “If the mobile with given features will be Economical or Expensive” is the main motive of this research work.

Real Dataset is collected from our project guide. Different feature selection algorithms are used to identify and remove less important and redundant features and have minimum computational complexity.

Different classifiers are used to achieve as higher accuracy as possible.

Results are compared in terms of highest accuracy achieved and minimum features selected.

Conclusion is made on the base of best feature selection algorithm and best classifier for the given dataset.

This work can be used in any type of marketing and business to find optimal product (with minimum cost and maximum features).

Future work is suggested to extend this research and find more sophisticated solution to the given problem and more accurate tool for price estimation.

1.1 Introduction :-

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 (e.g.: - 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 we do not have to predict actual price but a price range indicating how high the price should be according to the phone's specification.

Price is the most effective attribute of marketing and business. The very first question of costumer

is about the price of items. All the costumers are first worried and thinks “If he would be able to purchase something with given specifications or not”.

So, to estimate price at home is the basic purpose of the work. This paper is only the first step toward the above-mentioned destination. Artificial Intelligence-which makes machine capable to answer the questions intelligently- now a days is very vast engineering field.

Machine learning provides us best techniques for artificial intelligence like classification, regression, supervised learning and unsupervised learning and many more. Different tools are available for machine learning tasks like MATLAB, Python, Cygwin, WEKA etc. We can use any of classifiers like Decision tree, Naïve Bayes and many more. Different type of feature selection algorithms are available to select only best features and minimize dataset.

This will reduce computational complexity of the problem. As this is optimization problem so many optimization techniques are also used to reduce dimensionality of the dataset.

1.2 Objective :-

Mobile now a days is one of the most selling and purchasing device. Every day new mobiles with new version and more features are launched. Hundreds and thousands of mobiles are sold and purchased on daily basis. So here the mobile price class prediction is a case study for the given type of problem i.e. finding optimal product. The same work can be done to estimate real price of all products like cars, bikes, generators, motors, food items, medicine etc.

Many features are very important to be considered to estimate price of mobile. For example, Processor of the mobile. Battery timing is also very important in today's busy schedule of human being. Size and thickness of the mobile are also important decision factors. Internal memory, Camera pixels, and video quality must be under consideration. Internet browsing is also one of the most important constraints in this technological era of 21st century. And so is the list of many features

based upon those, mobile price is decided. So, we will use many of above-mentioned features to classify whether the mobile would be very economical, economical, expensive or very_ expensive.

As a Machine Learning Enthusiasts, we have studied this case and we are going to tell you our vision about this problem and also, we will suggest you that what is the best price range value for your Smartphone Handset according to its specifications and features.

2.1 A brief history :-

A mobile phone, cell phone, cell phone, or hand phone, sometimes shortened to simply mobile, cell or just phone, is a portable telephone that can make and receive calls over a radio frequency link while the user is moving within a telephone service area.

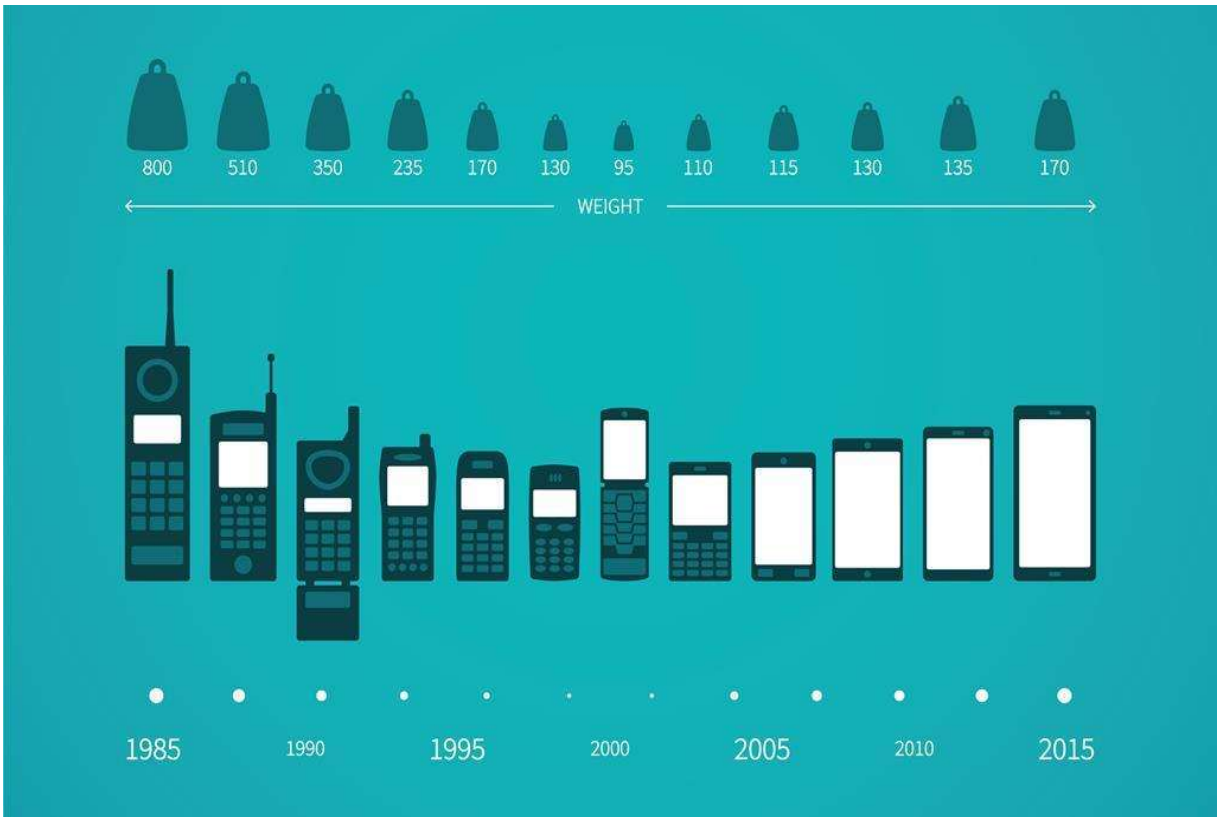
The first handheld mobile phone was demonstrated by John F. Mitchell and Martin Cooper of Motorola in 1973, using a handset weighing c. 2 kilograms (4.4 lbs).

In 1979, Nippon Telegraph and Telephone (NTT) launched the world's first cellular network in Japan. In 1983, the DynaTAC 8000x was the first commercially available handheld mobile phone. From 1983 to 2014, worldwide mobile phone subscriptions grew to over seven billion—enough to provide one for every person on Earth.

In first quarter of 2016, the top smartphone developers worldwide were Samsung, Apple, and Huawei, and smartphone sales represented 78 percent of total mobile phone sales. For feature phones (or "dumbphones") as of 2016, the largest were Samsung, Nokia, and Alcatel.

source:https://en.wikipedia.org/wiki/Mobile_phone

Evolution of Mobile Phone :-



2.2 PREVIOUS WORK :-

Using previous data to predict price of available and new launching product is an interesting research background for machine learning researchers. Sameerchand-Pudaruth predict the prices of second hand cars in Mauritius. He implemented many techniques like Multiple linear regression, k-nearest neighbors (KNN), Decision Tree, and Naïve Bayes to predict the prices. Sameerchand-Pudaruth got Comparable results from all these techniques. During research it was

found that most popular algorithms i.e. Decision Tree and Naïve Bayes are unable to handle, classify and predict Numerical values. Number of instances for his research was only 97(47 Toyota+38 Nissan+12 Honda). Due to a smaller number of instances used, very poor prediction accuracies were recorded. Shonda Kuiper has also worked in the same field. Kuiper used multivariate regression model to predict price of 2005 General Motor cars. He collected the data from available online source www.pakwheels.com. The main part of this research work is "Introduction of suitable variable selection techniques, which helped to find that which variables are more suitable and relevant for inclusion in model. This (His research) helps students and future researchers in many fields to understand the conditions under which studies should be conducted and gives them the knowledge to discern when appropriate techniques should be used. Support Vector Machine (SVM) concept is used by one another researcher Mariana Listiani for the same work. Listiani predicted prices of leased cars using above mentioned technique. It was found in this

research that SVM technique is far better and accurate for price prediction as compared to other like multiple linear regression when a very large data set is available. The researcher also showed that SVM also handles high dimensional data better and avoids both the under-fitting and over-fitting issues. To find important features for SVM Listiani used Genetic Algorithm. However, the technique failed to show in terms of variance and mean standard deviation why SVM is better than simple multiple regression. Neural Networks (NN) are better in estimating price of house, this was concluded in the research of Limsombunchai. By comparing with hedonic method his method was more accurate. Operation of both the methods are same, but in NN the model is trained first and then tested for prediction. Using both the methods NN produced higher R-sq. and smaller root mean square error (RMSE), while hedonic produced lower values. This research was limited because the actual house price was missing and only estimated prices were used for the research work. K Noor and Saddaqat J also worked to predict the price of Vehicles using different techniques. The

researchers achieved highest accuracy using multiple linear regression. This paper proposes a system where price is dependent variable which is predicted, and this price is derived from factors like vehicle's model, make, city, version, color, mileage, alloy rims and power steering.

2.3 The Future :-

What's new is really just incremental: This is one of the most common observations made by analysts at about the future of smartphones during the past few years, when faster processors, more storage, improved cameras, new colours, and so on have been the latest and relatively greatest. Yes, price/performance almost always improves with each new model, and about a billion and a half smartphones get sold worldwide each year regardless. Apple is really just an iPhone company now, with the remainder of its product line--such as it is--seemingly a footnote.

But with smartphones now the preferred access and communications device for almost everyone everywhere, it’s fair to ask if there is--or even could be--anything really big on the horizon that might start a new handset revolution. We don't expect that to happen in the near term. When it comes to the future of smartphones, consider it more of an ongoing evolution. Given that advances in Wi-Fi (802.11ac Wave 2, the upcoming 802.11ax, and perhaps even 802.11ad and .11ay) and 5G will absolutely require new hardware in the future, it would seem that that the upgrade market will be intact for now, even absent any other meaningful user-visible enhancements.

We’re definitely not expecting any major changes in the form factor, internal hardware architecture (still a computer), operating systems, or the end-user look and feel of smartphones for the foreseeable future. The old adage “form follows function” always applies in industrial design, and smartphones are basically touchscreens with the batteries and electronics required to make them go.

Mobile operating systems will evolve incrementally with a few new features, but will mostly remain the same so as not to upset the massive installed base of apps, the developer community or even end-users: Wholesale change to user interfaces for the sake of wholesale change is always a bad idea, although some cleverness here is invited.

So, in future we can expect that many features will be added to the smartphone and as features will gradually increase the price will also increase and as a result Machine Learning Predictions will become more and more complex.

3.1 Essential Libraries and Tools :-

We have to import some libraries to work various operations on the dataset. Here is the list of libraries mentioned below that are used for predictions :-

NumPy Library :-

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data.

Arbitrary data-types can be defined using Numpy which allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

Pandas Library :-

Python is a great language for doing data analysis, primarily because of the fantastic ecosystem of

data-centric python packages. Pandas is one of those packages and makes importing and analysing data much easier.

Pandas dataframe.info () function is used to get a concise summary of the dataframe. It comes really handy when doing exploratory analysis of the data. To get a quick overview of the dataset we use the dataframe.info () function.

Matplotlib Library :-

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+. There is also a procedural "pylab" interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB, though its use is discouraged.[3] SciPy makes use of Matplotlib.

Seaborn Library :-

Seaborn is a Python data visualization library based on matplotlib. It provides a high-level

interface for drawing attractive and informative statistical graphics.

For a brief introduction to the ideas behind the library, you can read the introductory notes. Visit the installation page to see how you can download the package. You can browse the example gallery to see what you can do with seaborn, and then check out the tutorial and API reference to find out how.

To see the code or report a bug, please visit the github repository. General support issues are most at home on stackoverflow, where there is a seaborn tag.

Sklearn Library :-

Scikit-learn (Sklearn) is a library in Python that provides many unsupervised and supervised learning algorithms. It's built upon some of the technology you might already be familiar with, like NumPy, pandas, and Matplotlib!

The functionality that scikit-learn provides include:

- Regression, including Linear and Logistic Regression
- Classification, including K-Nearest Neighbors
- Clustering, including K-Means and K-Means++
- Model selection
- Pre-processing, including Min-Max Normalization

As we move through our Machine Learning content, you will become familiar with many of these terms. You will also see scikit-learn (in Python, sklearn) modules being used. For example:

sklearn.linear_model.LinearRegression() is a Linear Regression model inside the linear_model module of sklearn.

The power of scikit-learn will greatly aid your creation of robust Machine Learning programs.

These are the libraries we are going to use in python3.

The Tools that are used are given below:-

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import neighbors
from sklearn import preprocessing
from sklearn import tree
from sklearn import metrics
from sklearn import model_selection
from sklearn import linear_model
from sklearn import feature_selection
from sklearn import tree
from sklearn import ensemble
from sklearn import preprocessing
from sklearn import neighbors
from sklearn import naive_bayes
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import confusion_matrix,classification_report
from tempfile import TemporaryFile
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import log_loss
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.preprocessing import LabelBinarizer
from sklearn.metrics import roc_auc_score
from sklearn.multiclass import OneVsRestClassifier
from scipy import interp
from sklearn.preprocessing import label_binarize
from sklearn import svm
from itertools import cycle
from sklearn.metrics import roc_curve, auc
from sklearn.model_selection import train_test_split
```

4. METHODOLOGY :-

The experiment is performed using WEKA (Waikato Environment for Knowledge Analysis). The main steps of machine learning are as follows

4.1 Dataset Collection :-

We have been given the train dataset by our project guide Titas Roy Chowdhury Sir.

The dataset has 2000 rows and 21 columns.

Reading the dataset :-

```
df=pd.read_csv("C:/Users/Apratim/Desktop/C/ml/datasets/handset_segmentation/train.csv")
```

Data Study & Analysis

In [3]:
```
def brief_info():
    print(df.info())
    print(df.shape)
    print(df.head())
    print(df.tail())
    print('Discrete columns are:')
    for cols in df.columns:
        if (df[cols].dtypes)=='int64':
            print(cols,'|',end=' ')
    print('\n_____________________________')
    print('Continous Columns are:')
    for cols in df.columns:
        if (df[cols].dtypes)!='int64':
            print(cols)
    return df.describe()
brief_info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 21 columns):
battery_power    2000 non-null int64
blue             2000 non-null int64
clock_speed      2000 non-null float64
dual_sim         2000 non-null int64
fc               2000 non-null int64
four_g           2000 non-null int64
int_memory       2000 non-null int64
m_dep            2000 non-null float64
mobile_wt        2000 non-null int64
n_cores          2000 non-null int64
pc               2000 non-null int64
px_height        2000 non-null int64
px_width         2000 non-null int64
ram              2000 non-null int64
sc_h             2000 non-null int64
sc_w             2000 non-null int64
talk_time        2000 non-null int64
three_g          2000 non-null int64
touch_screen     2000 non-null int64
wifi             2000 non-null int64
price_range      2000 non-null int64
dtypes: float64(2), int64(19)
memory usage: 328.2 KB
None
(2000, 21)
```

Understanding The Columns :-

battery_power: Total energy a battery can store in one time measured in mAh

blue: Has bluetooth or not

clock_speed: speed at which microprocessor executes instructions

dual_sim: Has dual sim support or not

fc: Front Camera mega pixels

four_g: Has 4G or not

int_memory: Internal Memory in Gigabytes

m_dep: Mobile Depth in cm

mobile_wt: Weight of mobile phone

n_cores: Number of cores of processor

pc: Primary Camera mega pixels

px_height: Pixel Resolution Height

px_width: Pixel Resolution Width

ram: Random Access Memory in Mega Bytes

sc_h: Screen Height of mobile in cm

sc_w: Screen Width of mobile in cm

talk_time: longest time that a single battery charge will last when you are

three_g: Has 3G or not

touch_screen: Has touch screen or not

wifi: Has wifi or not

price_range: This is the target variable with value of 0(low cost), 1(medium cost), 2(high cost) and 3(very high cost).

```
Discrete columns are:
battery_power | blue | dual_sim | fc | four_g | int_memory | mobile_wt | n_cores | pc | px_height | px_width | ram | sc_h | sc_
w | talk_time | three_g | touch_screen | wifi | price_range |

_____________________

Continous Columns are:
clock_speed
m_dep
```

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	...	px_height
count	2000.000000	2000.0000	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
std	439.418206	0.5001	0.816004	0.500035	4.341444	0.499662	18.145715	0.288416	35.399655	2.287837	...	443.780811
min	501.000000	0.0000	0.500000	0.000000	0.000000	0.000000	2.000000	0.100000	80.000000	1.000000	...	0.000000
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	...	564.000000
75%	1615.250000	1.0000	2.200000	1.000000	7.000000	1.000000	48.000000	0.800000	170.000000	7.000000	...	947.250000
max	1998.000000	1.0000	3.000000	1.000000	19.000000	1.000000	64.000000	1.000000	200.000000	8.000000	...	1960.000000

8 rows × 21 columns

px_width	ram	sc_h	sc_w	talk_time	three_g	touch_screen	wifi	price_range
2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000
1251.515500	2124.213000	12.306500	5.767000	11.011000	0.761500	0.503000	0.507000	1.500000
432.199447	1084.732044	4.213245	4.356398	5.463955	0.426273	0.500116	0.500076	1.118314
500.000000	256.000000	5.000000	0.000000	2.000000	0.000000	0.000000	0.000000	0.000000
874.750000	1207.500000	9.000000	2.000000	6.000000	1.000000	0.000000	0.000000	0.750000
1247.000000	2146.500000	12.000000	5.000000	11.000000	1.000000	1.000000	1.000000	1.500000
1633.000000	3064.500000	16.000000	9.000000	16.000000	1.000000	1.000000	1.000000	2.250000
1998.000000	3998.000000	19.000000	18.000000	20.000000	1.000000	1.000000	1.000000	3.000000

Data Analysis :-

In [4]:
```
l4=[]
def seperator():
    l1=[]
    m=df.columns
    for i in range(len(m)):
        l1.append(np.array(df[m[i]]))
    l2=[]
    for j in range(len(l1)):
        l2.append(np.unique(l1[j]))
    l3=[]
    for i in range(len(l2)):
        if len(l2[i])==2:
            l3.append(i)
    for i in range(len(l3)):
        l4.append(m[l3[i]])
    l5=list(set(df.columns)-set(l4))
    print("Boolean valued columns are:",l4)
    print("_____________________")
    print("Rest of the columns are:",l5)
    print("_____________________")
    for item in l4:
        print(df[item].value_counts())
seperator()
```

```
Boolean valued columns are: ['blue', 'dual_sim', 'four_g', 'three_g', 'touch_screen', 'wifi']
_____
Rest of the columns are: ['n_cores', 'battery_power', 'pc', 'fc', 'ram', 'px_width', 'sc_h', 'm_dep', 'mobile_wt', 'talk_time',
'int_memory', 'price_range', 'clock_speed', 'sc_w', 'px_height']
_____
0    1010
1     990
Name: blue, dtype: int64
1    1019
0     981
Name: dual_sim, dtype: int64
1    1043
0     957
Name: four_g, dtype: int64
1    1523
0     477
Name: three_g, dtype: int64
1    1006
0     994
Name: touch_screen, dtype: int64
1    1014
0     986
Name: wifi, dtype: int64
```

4.2 Data Cleaning :-

No Data Cleaning is required for this dataset. All Columns are already enough précised.

Also, there is no null value in the dataset.

4.3 Visualization & Graphical Analysis :-

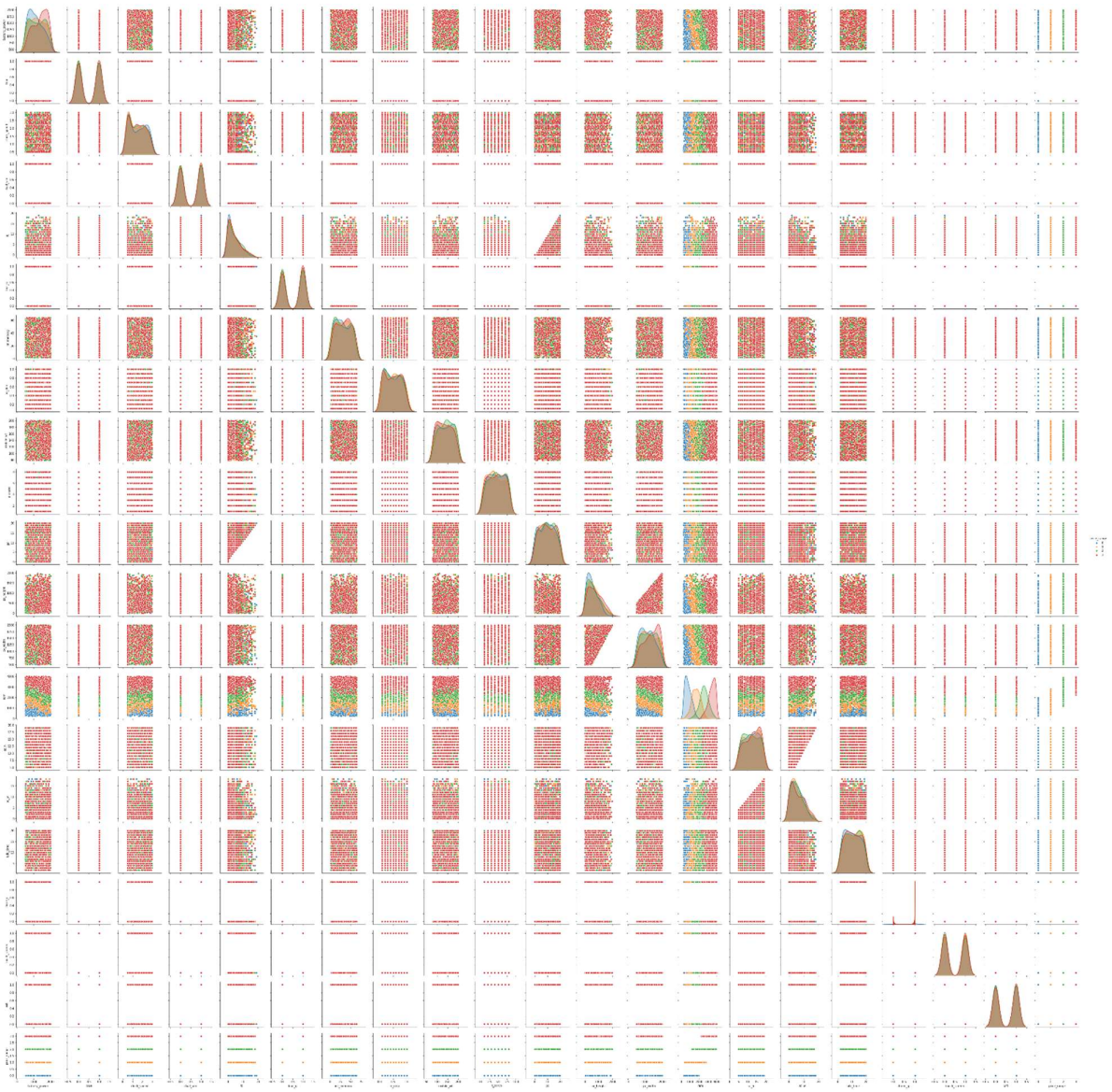
Pairplot() :-

When you generalize joint plots to datasets of larger dimensions, you end up with *pair plots*. This is very useful for exploring correlations between multidimensional data, when you'd like to plot all pairs of values against each other.

```
In [6]: sns.pairplot(data=df,hue='price_range')
```

```
C:\Users\Apratim\Anaconda3\desktop\lib\site-packages\statsmodels\nonparametric\kde.py:488: RuntimeWarning: invalid value encoun
tered in true_divide
  binned = fast_linbin(X, a, b, gridsize) / (delta * nobs)
C:\Users\Apratim\Anaconda3\desktop\lib\site-packages\statsmodels\nonparametric\kdetools.py:34: RuntimeWarning: invalid value en
countered in double_scalars
  FAC1 = 2*(np.pi*bw/RANGE)**2
```

```
Out[6]: <seaborn.axisgrid.PairGrid at 0x1eaa6fb3d68>
```



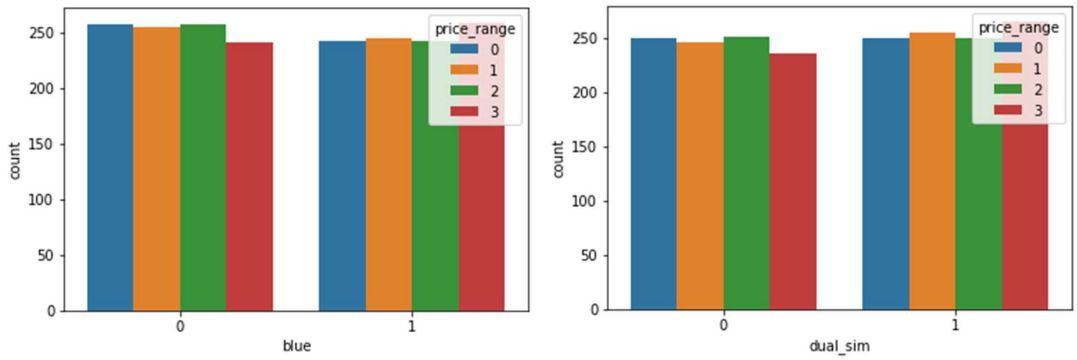
From this pairplot we can guess some important features are ram, battery_power, int_memory, px_width and px_height.

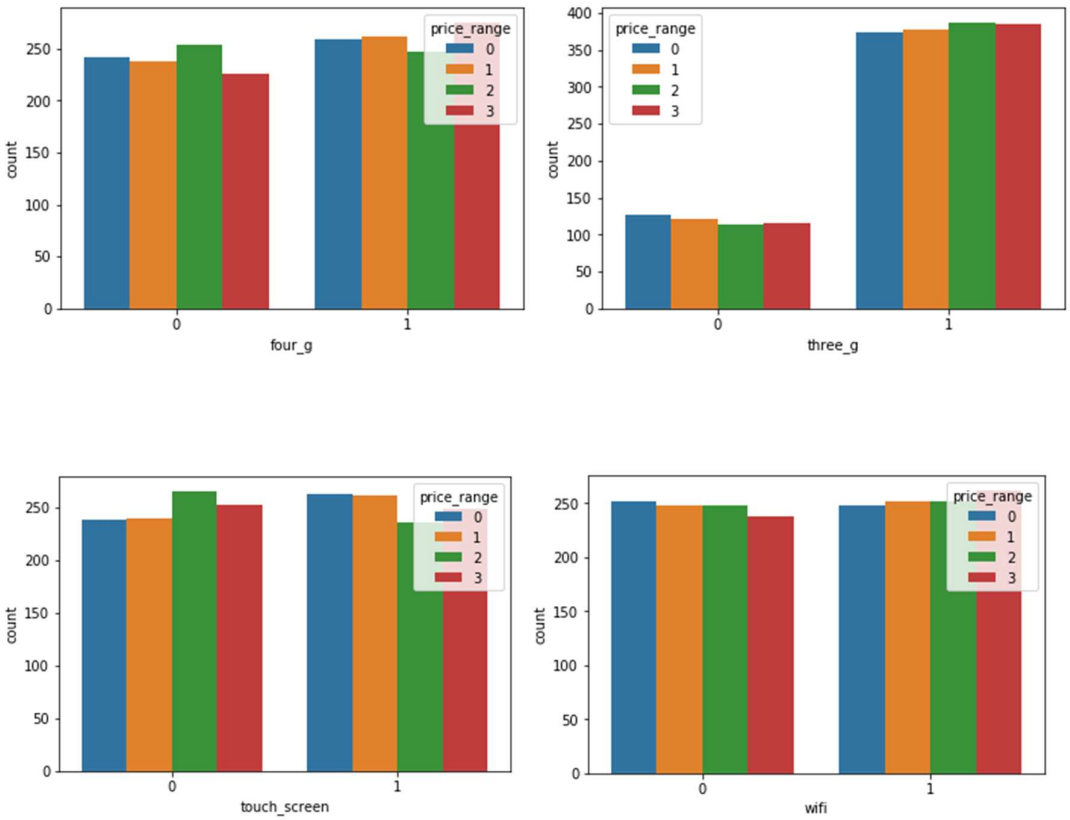
Countplot() :-

A count plot can be thought of as a histogram across a categorical, instead of quantitative, variable. The basic API and options are identical to those for barplot(), so you can compare counts across nested variables.

```
def plot():
    for i in range(len(l4)):
        sns.countplot(x=l4[i],hue='price_range',data=df)
        plt.show()
plot()
```

Here we have plotted some categorical value with price range to find how these values are related with price range.

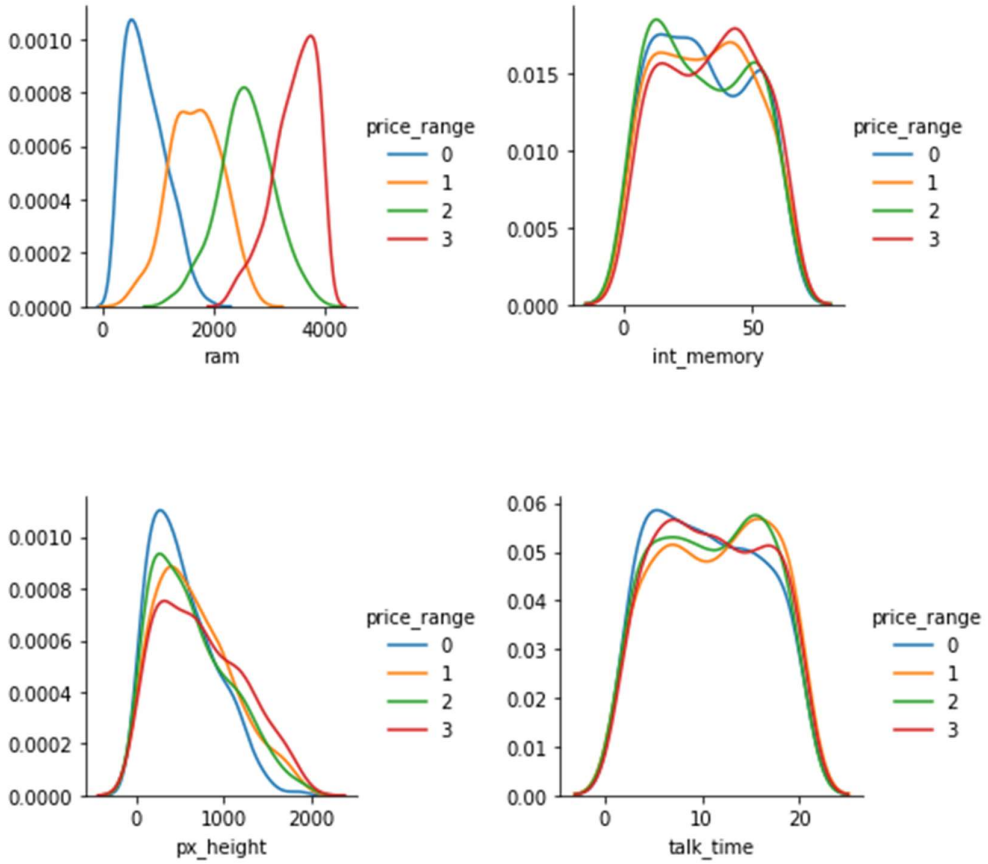




After plotting we can see that blue, dual_sim, four_g, three_g, touch_screen, and wifi has not that much impact on price_range so from this we can tell these values are not needed for predictions.

Showplot() :-

```
def showplot():
    l=['ram','int_memory','px_height','talk_time']
    for i in l:
        fig=sns.FacetGrid(data=df,hue="price_range")
        fig.map(sns.kdeplot,i)
        fig.add_legend()
        plt.show()
showplot()
```

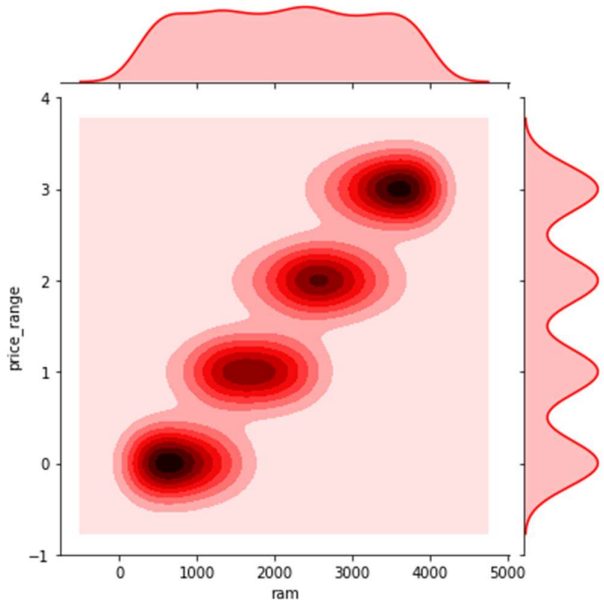


Jointplot() :-

Draw a plot of two variables with bivariate and univariate graphs.

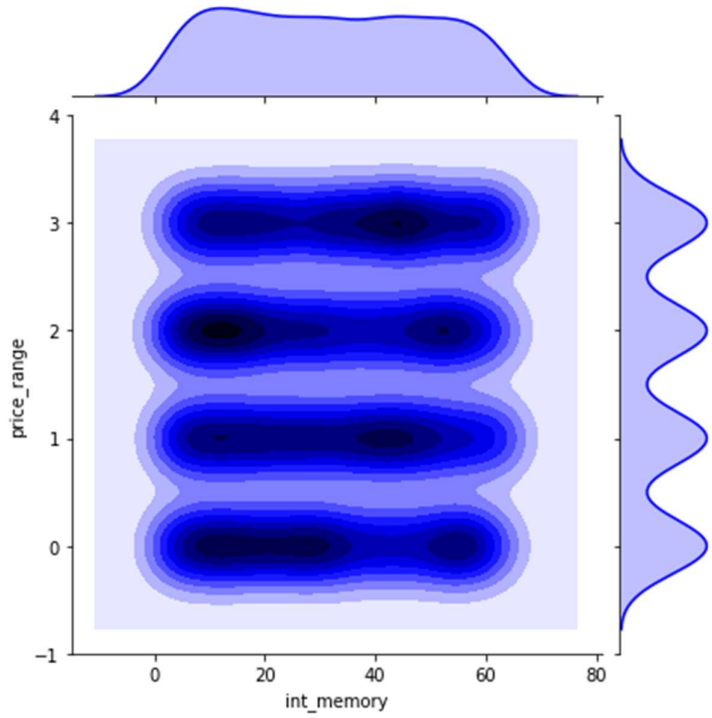
This function provides a convenient interface to the JointGrid class, with several canned plot kinds. This is intended to be a fairly lightweight wrapper; if you need more flexibility, you should use JointGrid directly.

```
sns.jointplot(x='ram',y='price_range',data=df,color='red',kind='kde')
```



From this jointplot between ram and price_range, we can see that price_range is gradually increasing as ram increases. So, we can say ram has a huge impact towards the prediction of price.

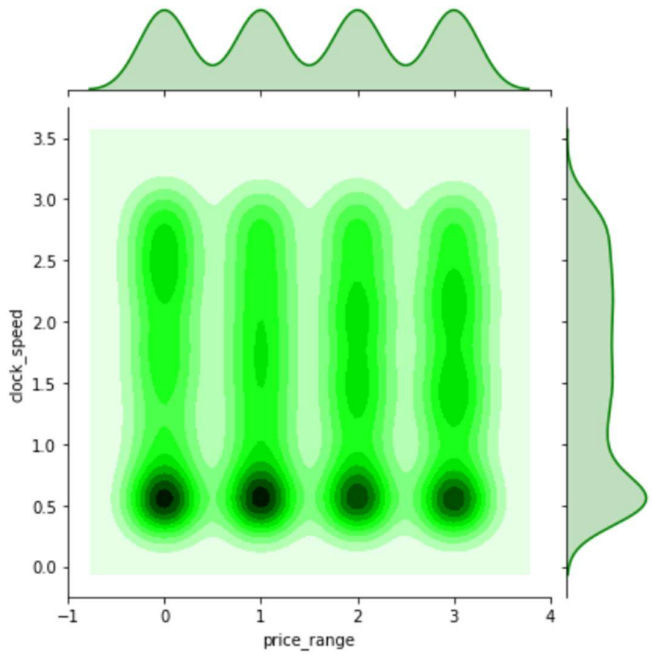
```
sns.jointplot(x='int_memory',y='price_range',data=df,color='blue',kind='kde')
```



From this jointplot between int_memory and price_range, we can see that price_range is not gradually increasing as ram increases. So, we can say int_memory doesn't have a huge impact towards the prediction.

```
sns.jointplot(x='price_range',y='clock_speed',data=df,color='green',kind='kde')
```

```
<seaborn.axisgrid.JointGrid at 0x1eac7dcbc50>
```



From this jointplot between clock_speed and price_range, we can see that price_range is not gradually increasing as clock_speed increases. So, we can say clock_speed have not that much impact towards the prediction.

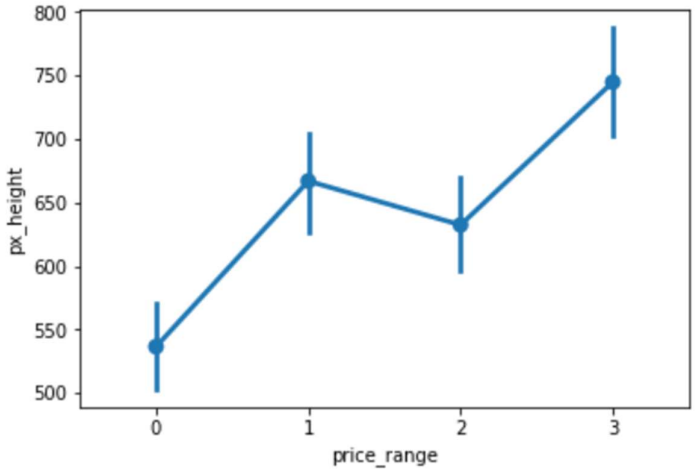
Pointplot() :-

Show point estimates and confidence intervals using scatter plot glyphs.

A point plot represents an estimate of central tendency for a numeric variable by the position of scatter plot points and provides some indication of the uncertainty around that estimate using error bars.

```
sns.pointplot(y='px_height',x='price_range',data=df)
```

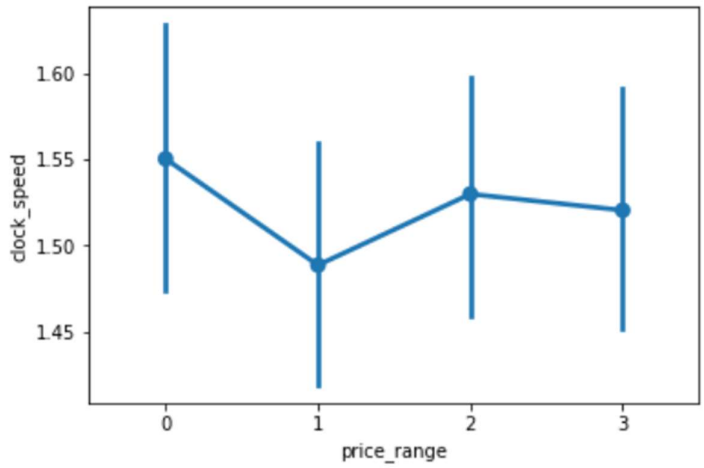
```
<matplotlib.axes._subplots.AxesSubplot at 0x1eac7d516d8>
```



From this Pointplot between px_height and price_range, we can see that price_range is gradually increasing as ram increases. So, we can say int_memory have a huge impact towards the prediction.

```
sns.pointplot(y='clock_speed',x='price_range',data=df)
```

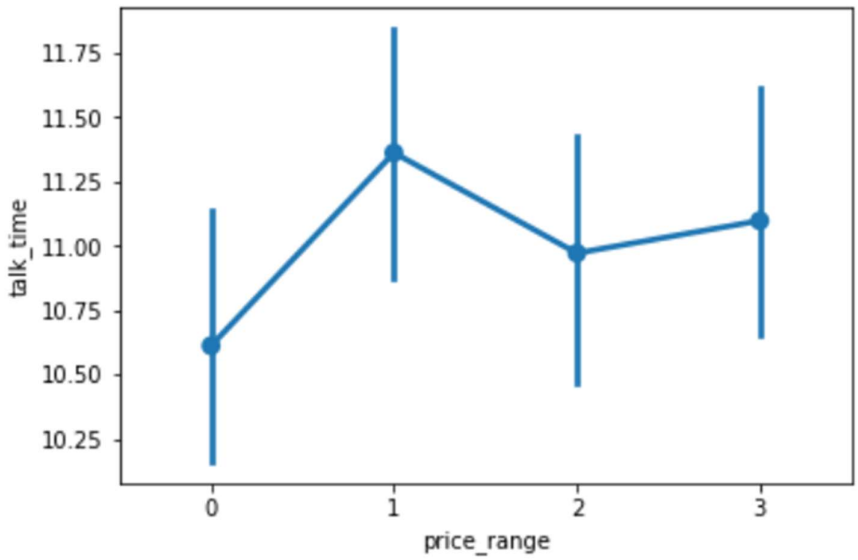
```
<matplotlib.axes._subplots.AxesSubplot at 0x1eac7d926a0>
```



From this Pointplot between clock_speed and price_range, we can see that price_range is not gradually increasing as clock_speed increases. So, we can say clock_speed have not that much impact towards the prediction.

```
sns.pointplot(y='talk_time',x='price_range',data=df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1eac7e6c550>
```

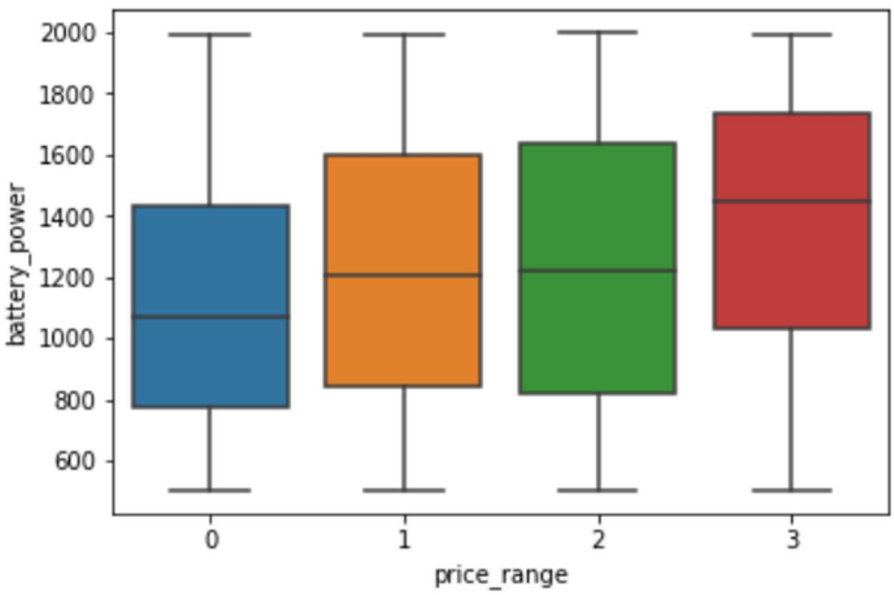


From this Pointplot between talk_time and price_range, we can see that price_range is not gradually increasing as talk_time increases. So, we can say talk_time have not that much impact towards the prediction.

Boxplot() :-

A box plot (or box-and-whisker plot) shows the distribution of quantitative data in a way that facilitates comparisons between variables or across levels of a categorical variable. The box shows the quartiles of the dataset while the whiskers extend to show the rest of the distribution, except for points that are determined to be “outliers” using a method that is a function of the inter-quartile range.

```
sns.boxplot(x="price_range",y="battery_power",data=df)
plt.show()
```



From this Boxplot between battry_power and price_range, we can see that price_range is gradually increasing as battery_power increases. So, we can say battery_power has a good impact towards the prediction.

Pieplot() :-

A pie plot is a proportional representation of the numerical data in a column. This function wraps matplotlib.pyplot.pie() for the specified column. If no column reference is passed and subplots=True a pie plot is drawn for each numerical column independently.

```
labels4g=["4g_supported","Not_supported"]
values4g=df["four_g"].value_counts().values
fig1,ax1=plt.subplots()
ax1.pie(values4g,labels=labels4g,shadow=True,startangle=90)
plt.show()
```

In this pieplot we plotted number of 4g supported and non supported devices. Through this plot we can see that the total number of 4g supported and non supported device is equally divided which means we can not predict price from this feature.

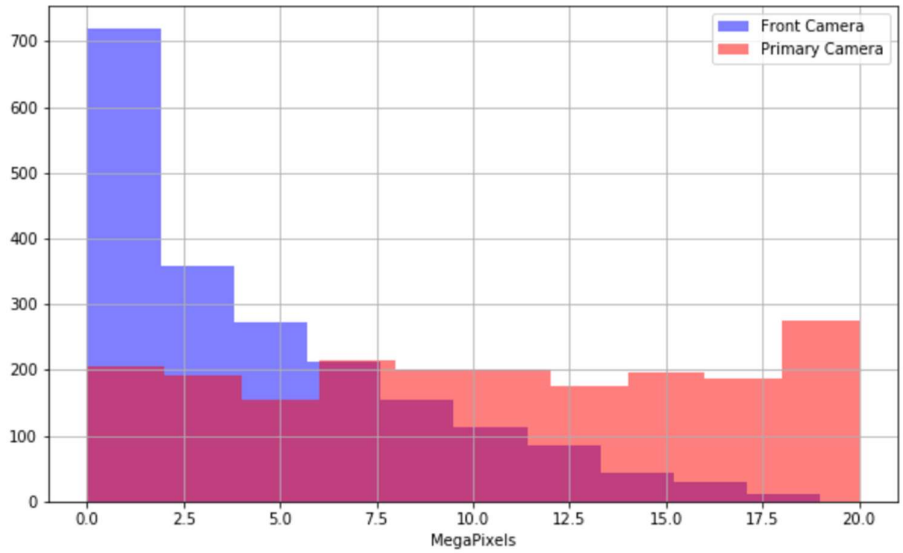
Histogram :-

The histogram of a categorical variable shows the frequency distribution of a that variable. By coloring the bars, you can visualize the distribution in connection with another categorical variable representing the colors.

Histogram shows the frequency distribution of a given variable. The below representation groups the frequency bars based on a categorical variable giving a greater insight about the continuous variable and the categorical variable in tandem.

```
plt.figure(figsize=(10,6))
df["fc"].hist(alpha=0.5,color="blue",label="Front Camera")
df["pc"].hist(alpha=0.5,color="red",label="Primary Camera")
plt.legend()
plt.xlabel("MegaPixels")
```

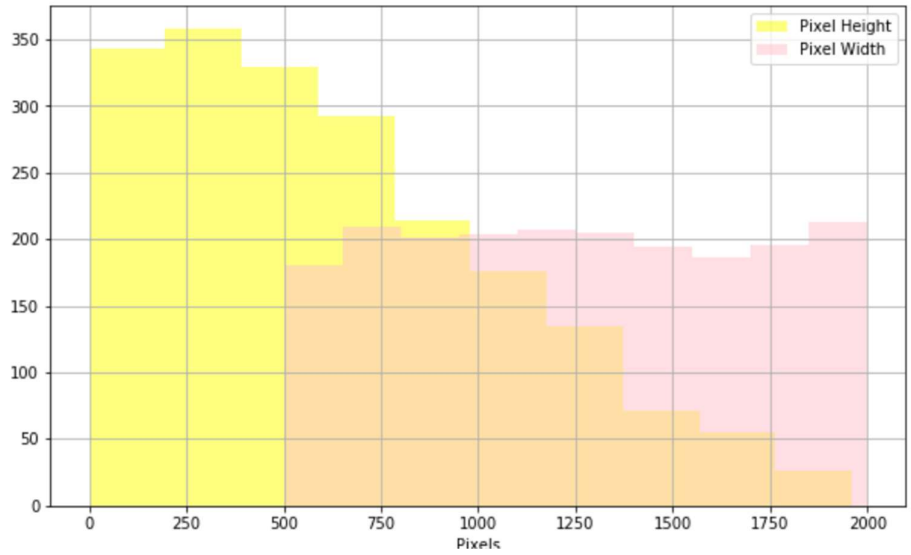
```
Text(0.5, 0, 'MegaPixels')
```



From this histogram we can see that front camera is more important feature than primary camera.

```
plt.figure(figsize=(10,6))
df["px_height"].hist(alpha=0.5,color="yellow",label="Pixel Height")
df["px_width"].hist(alpha=0.5,color="pink",label="Pixel Width")
plt.legend()
plt.xlabel("Pixels")
```

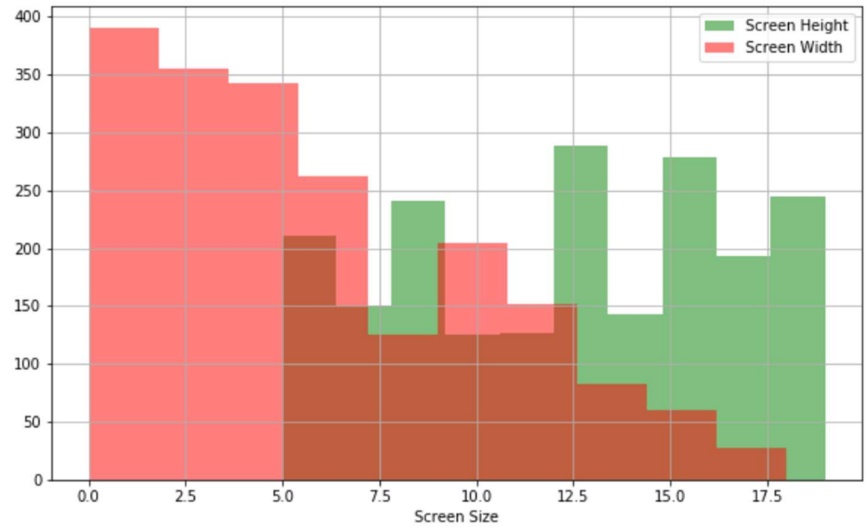
```
Text(0.5, 0, 'Pixels')
```



From this histogram we can see that pixel_height and pixel_width has near about same importance towards model building for prediction.

```
plt.figure(figsize=(10,6))
df["sc_h"].hist(alpha=0.5,color="green",label="Screen Height")
df["sc_w"].hist(alpha=0.5,color="red",label="Screen Width")
plt.legend()
plt.xlabel("Screen Size")
```

```
Text(0.5, 0, 'Screen Size')
```



From this histogram we can see that screen_width has slightly more importance than screen_height.

5.1 Dimensionality Reduction :-

Dimensionality reduction is the process of reducing the number of random variables (Features) under consideration, by obtaining a set of principal variables. The higher the number of features, the harder it gets to visualize the training set and then work on it. Sometimes, most of these features are correlated, and hence redundant. This is where dimensionality reduction algorithms come into play.

Two types of Dimensionality reduction algorithms are there i.e. Feature selection, Feature extraction.

Feature Selection:-

In feature selection we are interested in finding k of the d dimensions that give us the most information, and we discard the other (d − k) dimensions.

Feature Extraction:-

In feature extraction we are interested in finding a new set of k dimensions that are combinations of the original d dimensions for example Principal

Component Analysis . Here feature selection algorithms are used. There are two approaches: Forward selection and backward selection.

- ## Forward Selection

  In forward selection, we start with no variables and add them one by one, at each step adding the one that decreases the error the most, until any further addition does not decrease the error (or decreases it only slightly).

- ## Backward Selection

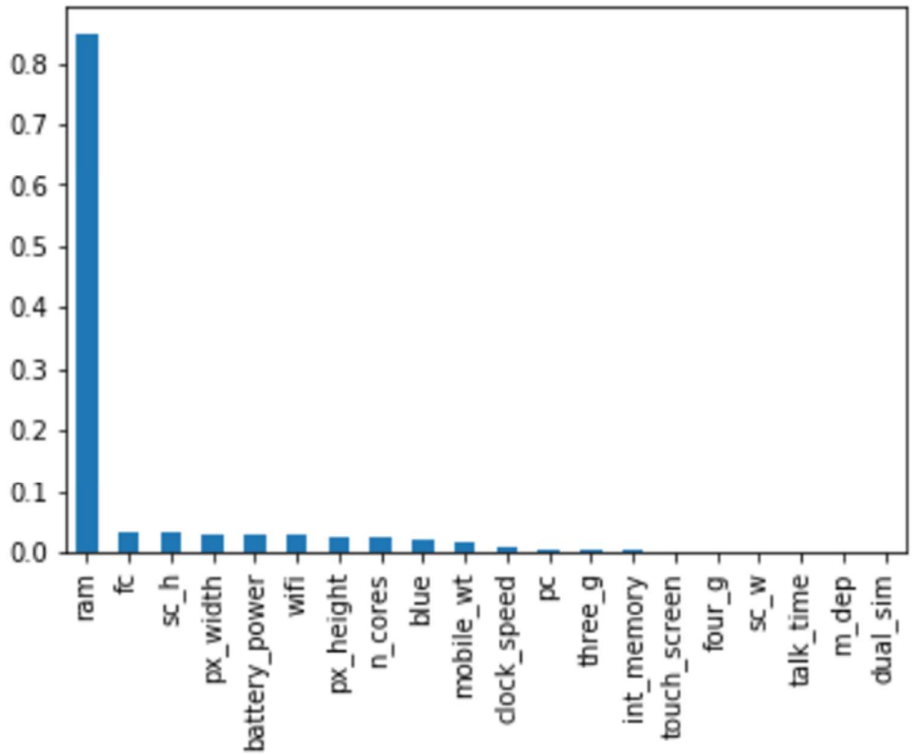
  In backward selection we start with all variables and remove them one by one, at each step removing the one that decreases the error the most (or increases it only slightly), until any further removal increases the error significantly.

Two feature selection algorithms are used InfoGainEval and WrapperattributEval. InfoGainAttributeEval evaluates the worth of an attribute by measuring the information gain with respect to the class. It gives us Ranked list from maximum important feature to minimum important features. While WrapperattributEval where the

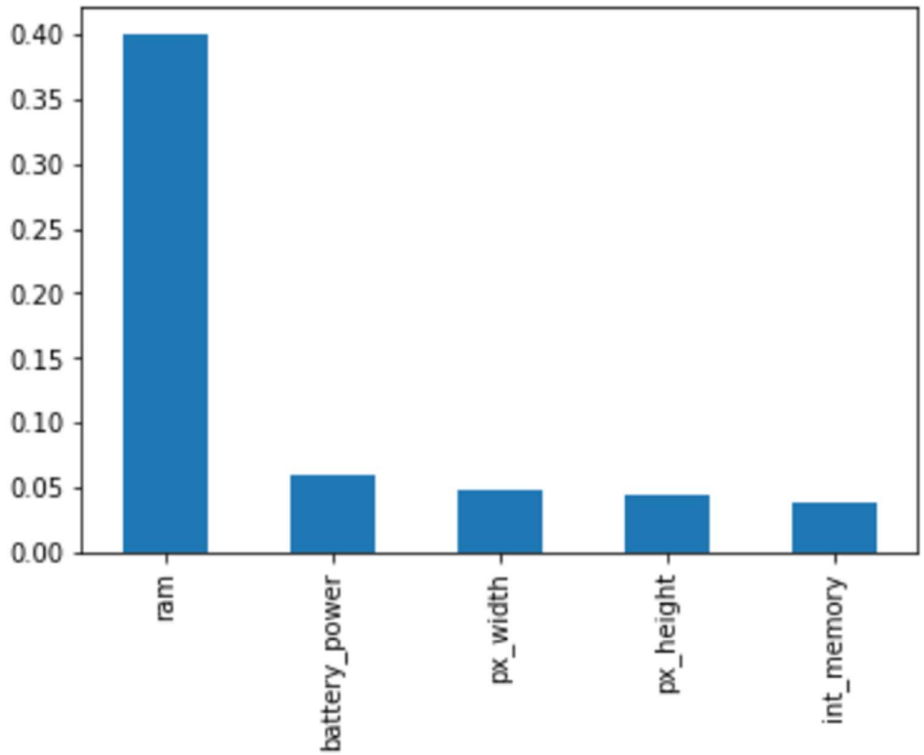
process of feature extraction is thought to “wrap” around the learner it uses as a subroutine. It gives us only list of important features.

Feature Selection

```
def mutual(X,y):
    mu=feature_selection.mutual_info_classif(X,y)
    muser=pd.Series(mu)
    muser.index=df1.columns.values
    muser.sort_values(ascending=False).plot.bar()
mutual(X2,y)
```



```
list1=[]
def feature(n):
    bestfeatures=SelectKBest(k=n)
    fit = bestfeatures.fit(X2,y)
    dfscores = pd.DataFrame(fit.scores_)
    dfcolumns = pd.DataFrame(df1.columns)
    #concat two dataframes for better visualization
    featureScores = pd.concat([dfcolumns,dfscores],axis=1)
    featureScores.columns = ['Specs','Score']  #naming the dataframe columns
    f1=featureScores.nlargest(n,'Score')
    f1.plot(kind='bar')
    plt.show()
    set1=set(f1["Specs"])
    #Feature importance method
    print("============================================")
    model1 = ExtraTreesClassifier()
    model1.fit(X2,y)
    #use inbuilt class feature_importances of tree based classifiers
    #plot graph of feature importances for better visualization
    feat_importances = pd.Series(model1.feature_importances_,index=df1.columns)
    f2=feat_importances.nlargest(n)
    set2=set(f2.index)
    f2.plot(kind='bar')
    plt.show()
    set3=set1.intersection(set2)
    set4=set(df.columns)
    lset=list(set4.difference(set3))
    for item in lset:
        list1.append(item)

feature(5)
```



So, as we guessed previously that the top important features are ram, battery_power, int_memory, px_width and px_height. From feature selection we see that our guessing is right.

So, after feature selection the top features are :-

- ram
- battery_power
- int_memory
- px_width
- px_height

Creating & training models :-

```
X1=preprocessing.scale(X,with_mean=True,with_std=True,copy=True)
df1=pd.DataFrame(X1)
df1.columns=X.columns
X2=df1
```

Train_test_split

```
list1.remove("price_range")
X2=df1.drop(columns=list1)
##Train,test split
Xtrain,Xtest,ytrain,ytest=model_selection.train_test_split(X2,y,test_size=.40,random_state=42)
```

5.2 Model Building Algorithm :-

To find the best model we are going to fit these train and test datasets in some models.

Logistic_Regression :-

Logistic regression is a classification algorithm used to assign observations to a discrete set of classes. Some of the examples of classification problems are Email spam or not spam, Online transactions Fraud or not Fraud, Tumour Malignant or Benign. We can call a Logistic Regression a Linear Regression model but the Logistic Regression uses a more complex cost function, this cost function can be defined as the ‘Sigmoid function’ or also known as the ‘logistic function’ instead of a linear function.
The hypothesis of logistic regression tends it to limit the cost function between 0 and 1. Therefore linear functions fail to represent it as it can have a value greater than 1 or less than 0 which is not possible as per the hypothesis of logistic regression.

Making predictions with a logistic regression model is as simple as plugging in numbers into the logistic regression equation and calculating a result.

Decision Tree Classifier :-

In computer science, Decision tree learning uses a decision tree (as a predictive model) to go from

observations about an item (represented in the branches) to conclusions about the item's target value (represented in the leaves). It is one of the predictive modelling approaches used in statistics, data mining and machine learning. Tree models where the target variable can take a discrete set of values are called classification trees; in these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels. Decision trees where the target variable can take continuous values (typically real numbers) are called regression trees. In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making. In data mining, a decision tree describes data (but the resulting classification tree can be an input for decision making). This page deals with decision trees in data mining.

Decision Tree Classifier, repetitively divides the working area(plot) into sub part by identifying lines. Decision tree at every stage selects the one that gives best information gain.

k-nearest neighbors algorithm :-

In pattern recognition, the *k*-nearest neighbors algorithm (*k*-NN) is a non-parametric method used for classification and regression.[1] In both cases,

the input consists of the *k* closest training examples in the feature space. The output depends on whether *k*-NN is used for classification or regression:

- In *k-NN classification*, the output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its *k* nearest neighbors (*k* is a positive integer, typically small). If *k* = 1, then the object is simply assigned to the class of that single nearest neighbor.
- In *k-NN regression*, the output is the property value for the object. This value is the average of the values of *k* nearest neighbors.

k-NN is a type of instance-based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until classification.

Both for classification and regression, a useful technique can be to assign weights to the contributions of the neighbors, so that the nearer neighbors contribute more to the average than the more distant ones. For example, a common weighting scheme consists in giving each neighbor a weight of 1/*d*, where *d* is the distance to the neighbor.[2]

The neighbors are taken from a set of objects for which the class (for *k*-NN classification) or the object property value (for *k*-NN regression) is known. This can be thought of as the training set for the algorithm, though no explicit training step is required.

A peculiarity of the *k*-NN algorithm is that it is sensitive to the local structure of the data.

Gaussian Naive Bayes :-

In machine learning, naive Bayes classifiers are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (naive) independence assumptions between the features.

Naive Bayes has been studied extensively since the 1960s. It was introduced (though not under that name) into the text retrieval community in the early 1960s,[1]and remains a popular (baseline) method for text categorization, the problem of judging documents as belonging to one category or the other (such as spam or legitimate, sports or politics, etc.) with word frequencies as the features. With appropriate pre-processing, it is competitive in this domain with more advanced methods including support vector machines.[2] It also finds application in automatic medical diagnosis.

Naive Bayes classifiers are highly scalable, requiring a number of parameters linear in the number of variables (features/predictors) in a learning problem. Maximum-likelihood training can be done by evaluating a closed-form expression, which takes linear time, rather than by expensive iterative approximation as used for many other types of classifiers.

In the statistics and computer science literature, naive Bayes models are known under a variety of names, including simple Bayes and independence Bayes. All these names reference the use of Bayes' theorem in the classifier's decision rule, but naive Bayes is not (necessarily)
a Bayesian method.

Random forest :-

Random forests or random decision forests are an ensemble learning method
for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set.

The first algorithm for random decision forests was created by Tin Kam Ho using the random subspace method, which, in Ho's formulation, is a

way to implement the "stochastic discrimination" approach to classification proposed by Eugene Kleinberg.

An extension of the algorithm was developed by Leo Breiman and Adele Cutler, who registered "Random Forests" as a trademark (as of 2019, owned by Minitab, Inc.). The extension combines Breiman's "bagging" idea and random selection of features, introduced first by Ho[1] and later independently by Amit and Geman in order to construct a collection of decision trees with controlled variance.

5.3 Model Development :-

Train_test_split

In [25]:
```
list1.remove("price_range")
X2=df1.drop(columns=list1)
##Train,test split
Xtrain,Xtest,ytrain,ytest=model_selection.train_test_split(X2,y,test_size=.40,random_state=42)
model=linear_model.LogisticRegression()
treemodel=tree.DecisionTreeClassifier(max_depth=2)
gnbobj=naive_bayes.GaussianNB()
```

In [32]:
```
rfmodel=ensemble.RandomForestClassifier()
b=[]
def best_n(n):
    param_grid={'n_estimators': [200,n]}
    CV_rfmodel=GridSearchCV(estimator=rfmodel,param_grid=param_grid,cv=5)
    CV_rfmodel.fit(Xtrain, ytrain)
    b.append(CV_rfmodel.best_params_['n_estimators'])
best_n(500)
rfmodel=ensemble.RandomForestClassifier(n_estimators=b[0])
```

Model - fit - Predict

In [27]:
```
a=[]
def best_k(n):
    k=[l for l in range(5,n,2)]
    knnobj=neighbors.KNeighborsClassifier(n_neighbors=k)
    grid={"n_neighbors":k}
    grid_obj = GridSearchCV(estimator=knnobj,param_grid=grid)
    grid_fit =grid_obj.fit(Xtrain,ytrain)
    knnobj = grid_fit.best_estimator_
    knnobj.fit(Xtrain,ytrain)
    a.append(grid_fit.best_params_['n_neighbors'])
best_k(17)
knnobj=neighbors.KNeighborsClassifier(n_neighbors=a[0])
```

GridSearchCV :-

Grid search is the process of performing hyper parameter tuning in order to determine the optimal values for a given model. This is significant as the performance of the entire model is based on the hyper parameter values specified.

If you work with ML, you know what a nightmare it is to stipulate values for hyper parameters. There are libraries that have been implemented, such as GridSearchCV of the sklearn library, in order to automate this process and make life a little bit easier for ML enthusiasts.

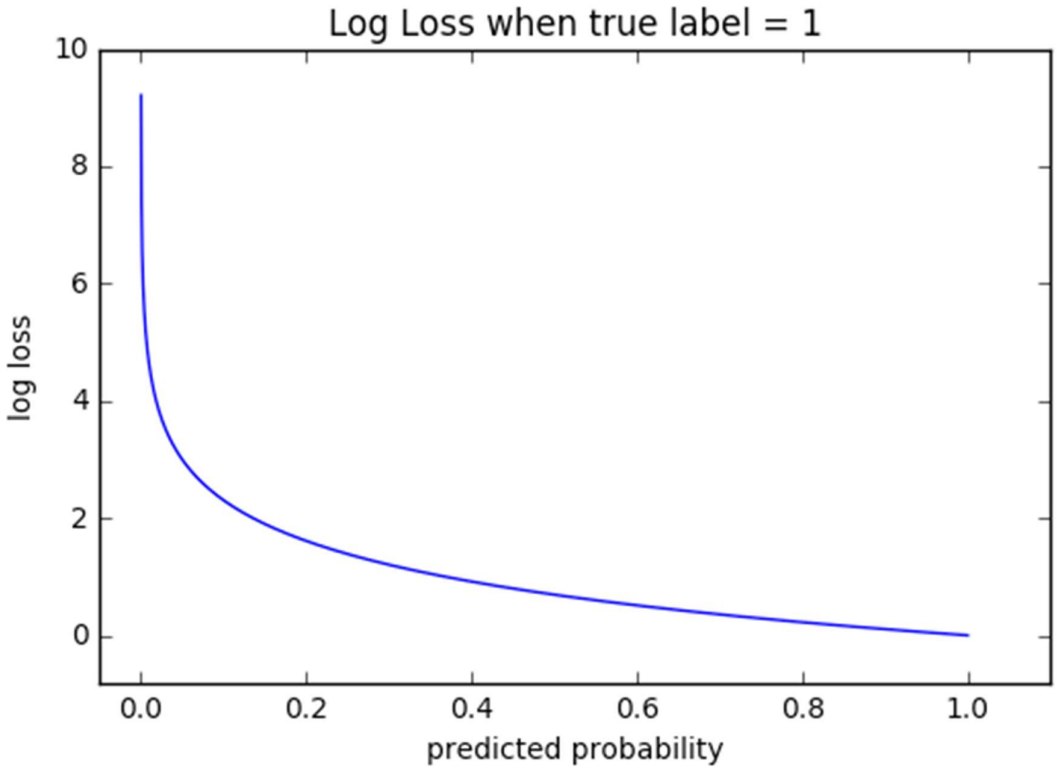
Using GridSearchCV is easy. You just need to import GridSearchCV from sklearn.grid_search, setup a parameter grid (using multiples of 10's is a good place to start) and then pass the algorithm, parameter grid and number of cross validations to the GridSearchCV method.

GridSearchCV implements a "fit" method and a "predict" method like any classifier except that the parameters of the classifier used to predict is optimized by cross-validation.

Cross-entropy :-

Cross-entropy loss, or log loss, measures the performance of a classification model whose output is a probability value between 0 and 1. Cross-entropy loss increases as the predicted probability diverges from the actual label. So, predicting a probability of .012 when the actual observation label is 1 would be bad and result in a high loss value. A perfect model would have a log loss of 0.

Cross entropy indicates the distance between what the model believes the output distribution should be, and what the original distribution really is. It is defined as, H(y,p)=−∑iyilog(pi)H(y,p)=−∑iyilog(pi) Cross entropy measure is a widely used alternative of squared error. It is used when node activations can be understood as representing the probability that each hypothesis might be true, i.e. when the output is a probability distribution. Thus, it is used as a loss function in neural networks which have softmax activations in the output layer.



The graph above shows the range of possible loss values given a true observation (isDog = 1). As the predicted probability approaches 1, log loss slowly decreases. As the predicted probability decreases, however, the log loss increases rapidly. Log loss penalizes both types of errors, but especially those predictions that are confident and wrong!

Cross-entropy and log loss are slightly different depending on context, but in machine learning when calculating error rates between 0 and 1 they resolve to the same thing.

Let's see the programs to select the best model among these 5 models.

In [28]:
```
def cross_entropy(y,p):
    m = y.shape[0]
    log_likelihood = -np.log(p[range(m),y])
    loss = np.sum(log_likelihood)/m
    return loss
```

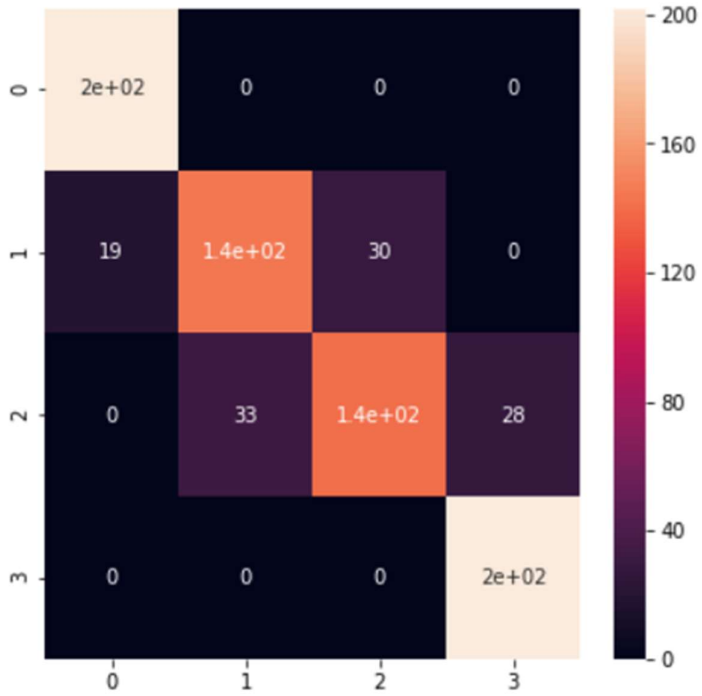
In [29]:
```
def multiclass_roc_auc_score(ytest,testp, average="macro"):
    lb = LabelBinarizer()
    lb.fit(ytest)
    ytest = lb.transform(ytest)
    testp = lb.transform(testp)
    return roc_auc_score(ytest,testp,average=average)
```

In [30]:
```
li=[]
li1=[]
li2=[]
def model_built(m):
    m.fit(Xtrain,ytrain)
    testp=m.predict(Xtest)
    print('Accuracy:',metrics.accuracy_score(ytest,testp))
    li2.append(metrics.accuracy_score(ytest,testp))
    print(classification_report(ytest,testp))
    m2=metrics.confusion_matrix(ytest,testp)
    print(m2)
    p=m.predict_proba(Xtest)
    print("Loss:",cross_entropy(ytest,p))
    print("\nroc_auc_score:",multiclass_roc_auc_score(ytest,testp))
    print("                                                ")
    li.append(cross_entropy(ytest,p))
    li1.append(multiclass_roc_auc_score(ytest,testp))
    print("========================================")
    plt.figure(figsize=(6,6))
    sns.heatmap(m2,annot=True)
    plt.show()
    print(m)
```

In [31]:
```
def fit():
    l6=[model,treemodel,knnobj,gnbobj,rfmodel]
    for item in l6:
        print("                                        ")
        model_built(item)
        print("========================================")
    dff=pd.DataFrame({"Cross Entropy Loss":li,"Roc_Auc":li1,"Accuracy":li2},index=['model','treemodel','knnobj','gnbobj','rfmode
l'])
    print("                                                ")
    print("The Entropy Loss of the best model is :",dff["Cross Entropy Loss"].min(),dff[["Cross Entropy Loss"]].idxmin())
    print("The max auc score:",dff["Roc_Auc"].max(),dff[["Roc_Auc"]].idxmax())
    print("The max accuracy :",dff["Accuracy"].max(),dff[["Accuracy"]].idxmax())
    li.clear()
    li1.clear()
    li2.clear()
    return dff

fit()
```

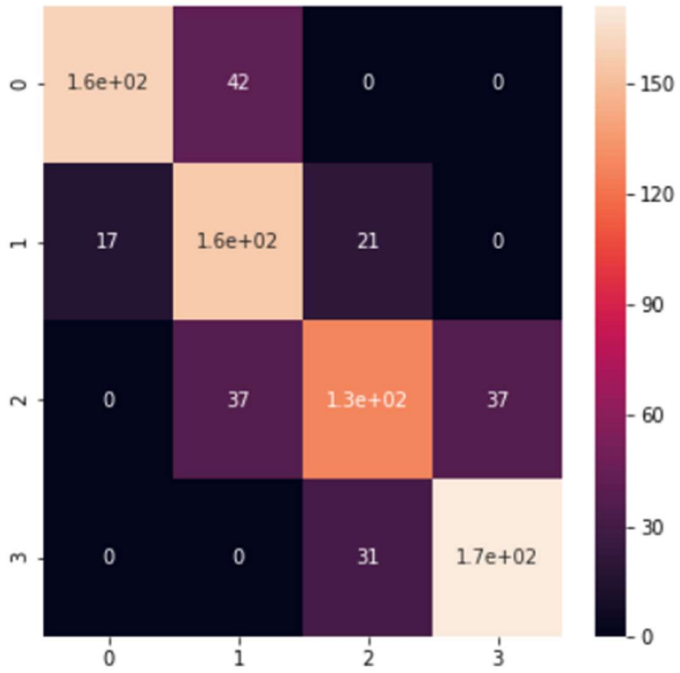
Now Let's see the outputs...

```
Accuracy: 0.8625
              precision    recall  f1-score   support

           0       0.91      1.00      0.96       202
           1       0.81      0.75      0.78       194
           2       0.82      0.70      0.76       202
           3       0.88      1.00      0.94       202

   micro avg       0.86      0.86      0.86       800
   macro avg       0.86      0.86      0.86       800
weighted avg       0.86      0.86      0.86       800

[[202   0   0   0]
 [ 19 145  30   0]
 [  0  33 141  28]
 [  0   0   0 202]]
Loss: 0.6044832181396965

roc_auc_score: 0.9077780618802493
```



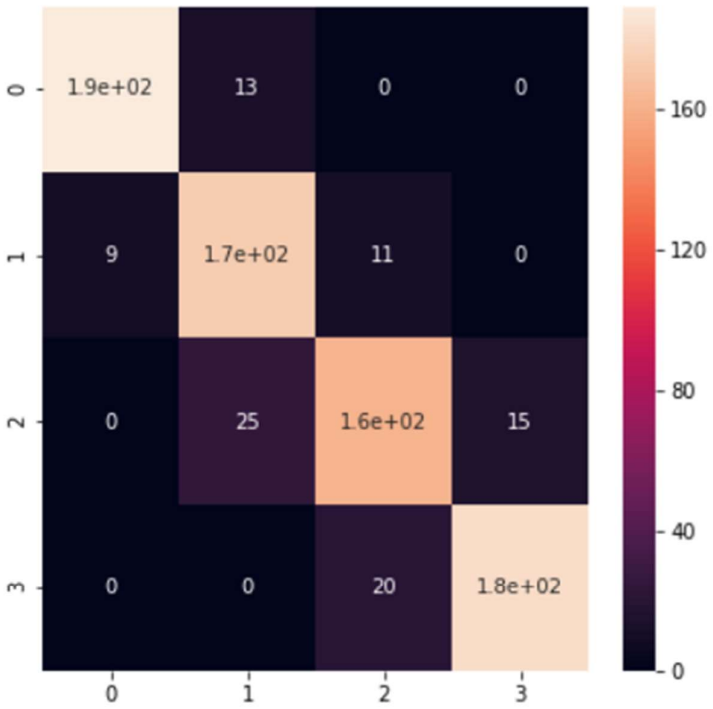

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='warn',
          n_jobs=None, penalty='l2', random_state=None, solver='warn',
          tol=0.0001, verbose=0, warm_start=False)
========================================
```

```
Accuracy: 0.76875
              precision    recall  f1-score   support

           0       0.90      0.79      0.84       202
           1       0.66      0.80      0.73       194
           2       0.71      0.63      0.67       202
           3       0.82      0.85      0.83       202

   micro avg       0.77      0.77      0.77       800
   macro avg       0.78      0.77      0.77       800
weighted avg       0.78      0.77      0.77       800

[[160  42   0   0]
 [ 17 156  21   0]
 [  0  37 128  37]
 [  0   0  31 171]]
Loss: 0.6305865529357235

roc_auc_score: 0.8460975472094671

========================================
```



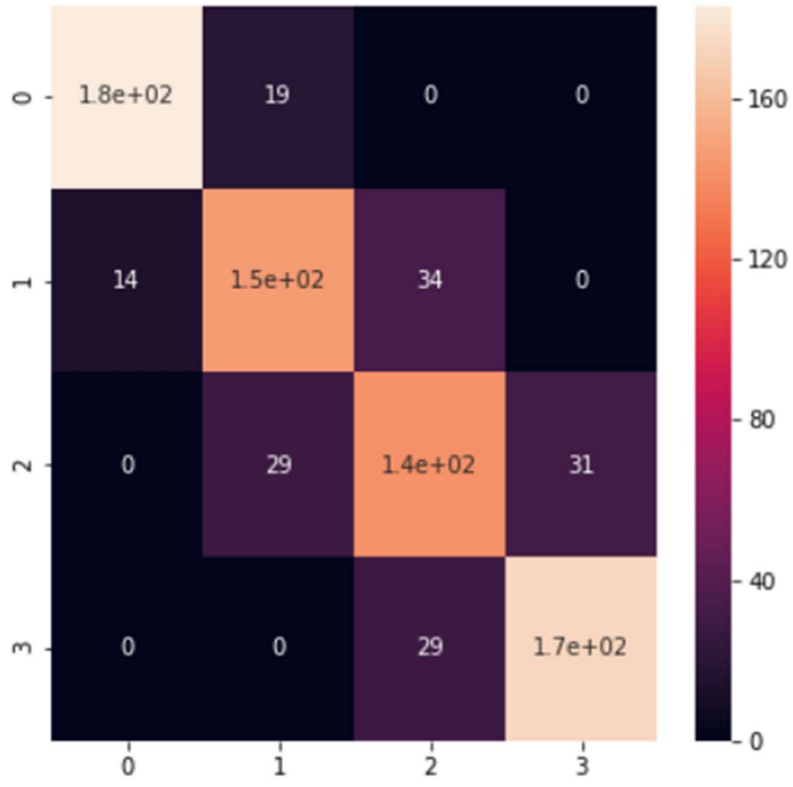

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=2,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best')
========================================
```

```
Accuracy: 0.88375
              precision    recall  f1-score   support

           0       0.95      0.94      0.95       202
           1       0.82      0.90      0.86       194
           2       0.84      0.80      0.82       202
           3       0.92      0.90      0.91       202

   micro avg       0.88      0.88      0.88       800
   macro avg       0.88      0.88      0.88       800
weighted avg       0.89      0.88      0.88       800

[[189  13   0   0]
 [  9 174  11   0]
 [  0  25 162  15]
 [  0   0  20 182]]
Loss: 0.3405525357380037

roc_auc_score: 0.9226051953883455

========================================
```



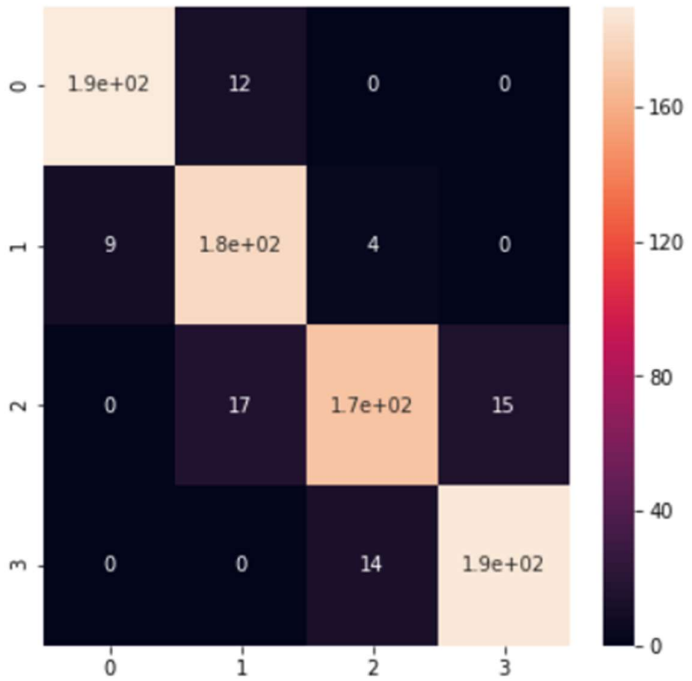

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=None, n_neighbors=15, p=2,
           weights='uniform')
========================================
```

```
Accuracy: 0.805
              precision    recall  f1-score   support

           0       0.93      0.91      0.92       202
           1       0.75      0.75      0.75       194
           2       0.69      0.70      0.70       202
           3       0.85      0.86      0.85       202

   micro avg       0.81      0.81      0.81       800
   macro avg       0.81      0.80      0.80       800
weighted avg       0.81      0.81      0.81       800

[[183  19   0   0]
 [ 14 146  34   0]
 [  0  29 142  31]
 [  0   0  29 173]]
Loss: 0.4476171337910448

roc_auc_score: 0.8697642408450064

========================================
```




```
GaussianNB(priors=None, var_smoothing=1e-09)
========================================
```

```
Accuracy: 0.91125
              precision    recall  f1-score   support

           0       0.95      0.94      0.95       202
           1       0.86      0.93      0.90       194
           2       0.90      0.84      0.87       202
           3       0.93      0.93      0.93       202

   micro avg       0.91      0.91      0.91       800
   macro avg       0.91      0.91      0.91       800
weighted avg       0.91      0.91      0.91       800

[[190  12   0   0]
 [  9 181   4   0]
 [  0  17 170  15]
 [  0   0  14 188]]
Loss: 0.24587803330632652

roc_auc_score: 0.94097150983243
```




```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=500, n_jobs=None,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
========================================
```

```
========================================

The Entropy Loss of the best model is : 0.24587803330632652 Cross Entropy Loss    rfmodel
dtype: object
The max auc score: 0.94097150983243 Roc_Auc    rfmodel
dtype: object
The max accuracy : 0.91125 Accuracy    rfmodel
dtype: object
```

Out[31]:

	Cross Entropy Loss	Roc_Auc	Accuracy
model	0.604483	0.907778	0.86250
treemodel	0.630587	0.846098	0.76875
knnobj	0.340553	0.922605	0.88375
gnbobj	0.447617	0.869764	0.80500
rfmodel	0.245878	0.940972	0.91125

6.1 Result Analysis :-

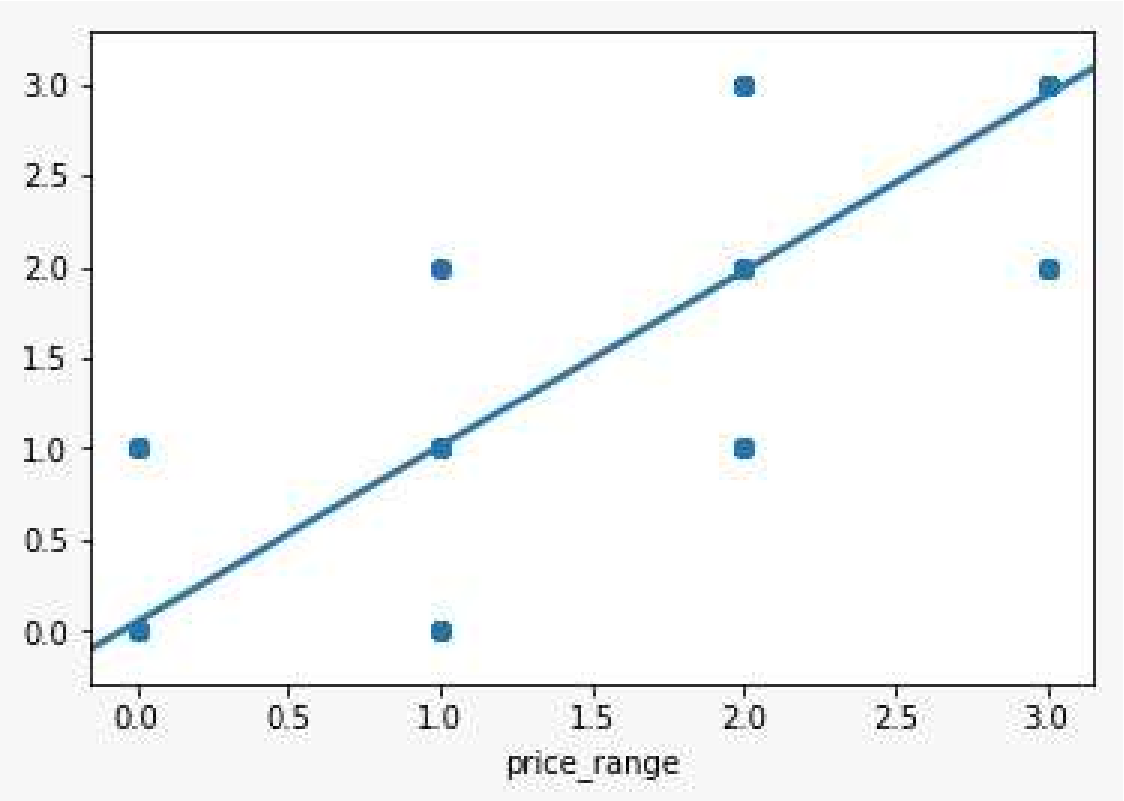
Model Finalization :-

After predicting these 5 models we can see the last chart where Random Forest Classifier is giving the lowest Cross Entropy Loss. According to the theory the lowest Cross Entropy Loss model is the best model for prediction so Random Forest is the best model. Also, we can see the Roc_Auc and Accuracy score is highest for Random Forest Classifier so we are using Random Forest Classifier for visualization of our final prediction.

```
: Xtest.head()
```

:

	battery_power	px_height	px_width	ram
1860	0.927552	-0.978448	0.825021	-1.326201
353	-0.128653	-0.834197	-0.614489	0.404613
1333	1.669628	-0.793626	-0.693176	-0.745267
905	-0.567980	-0.877022	0.329755	1.630107
1289	-1.419319	0.847234	1.635035	-0.201218

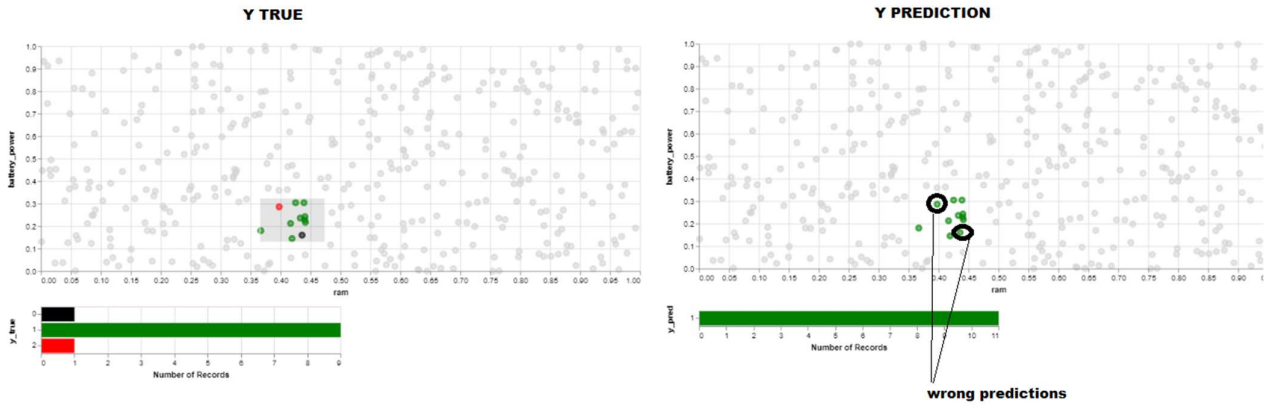


From this graph we can see each and every point is not that much scattered. That means our prediction is almost right.

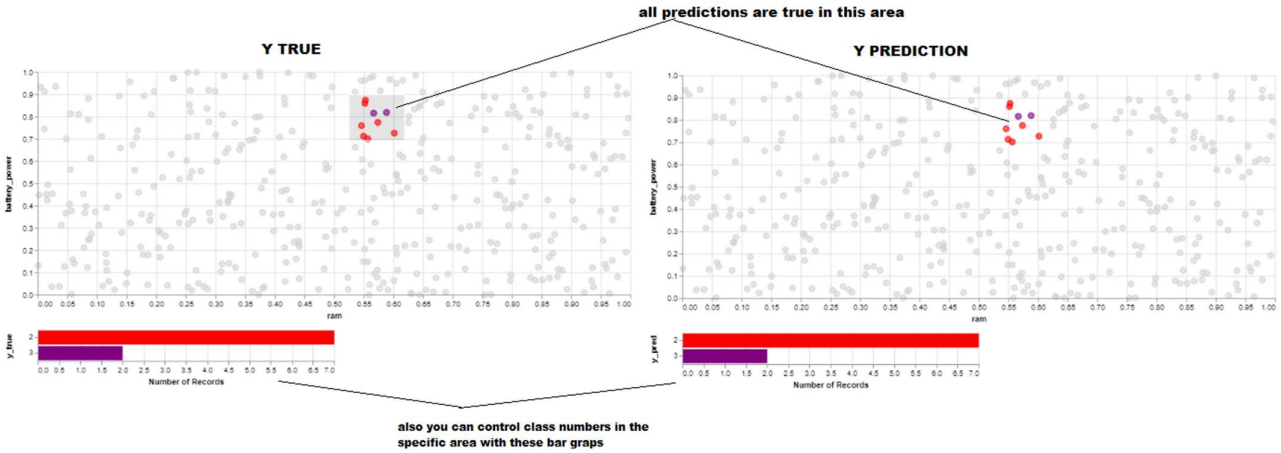
6.2 Predicted Model Visualization :-

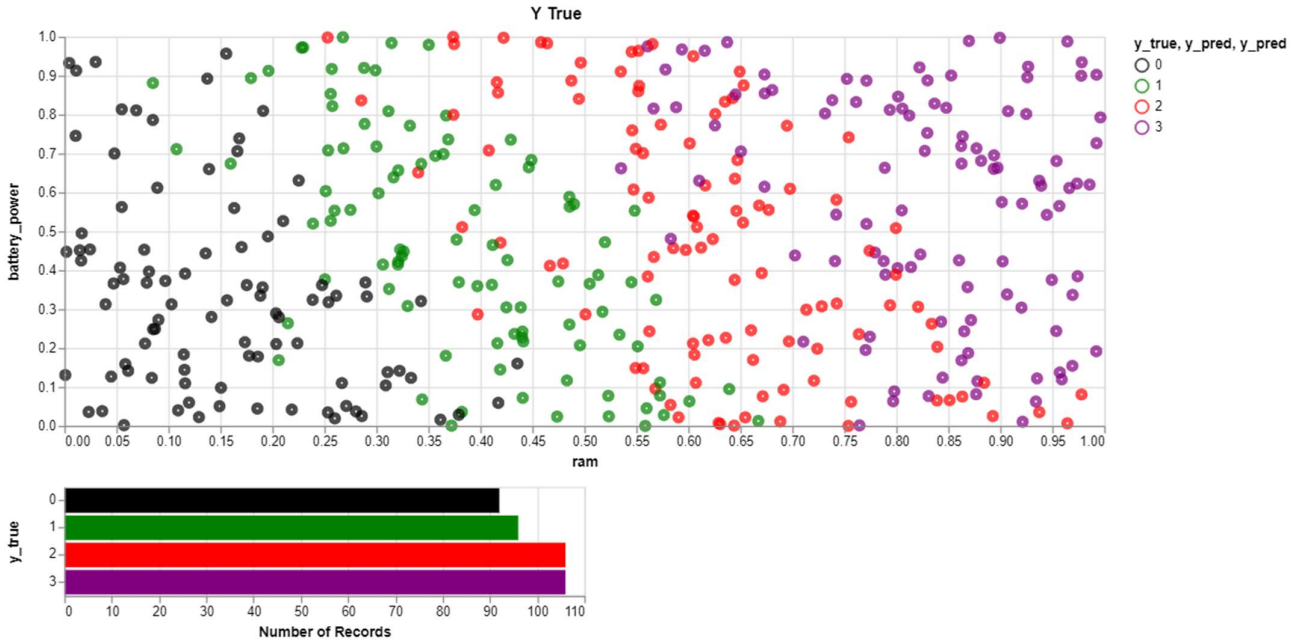
Following graphs shows ram and battery power features according to true values and predicted values. If you select an area you can see the differences. For example:

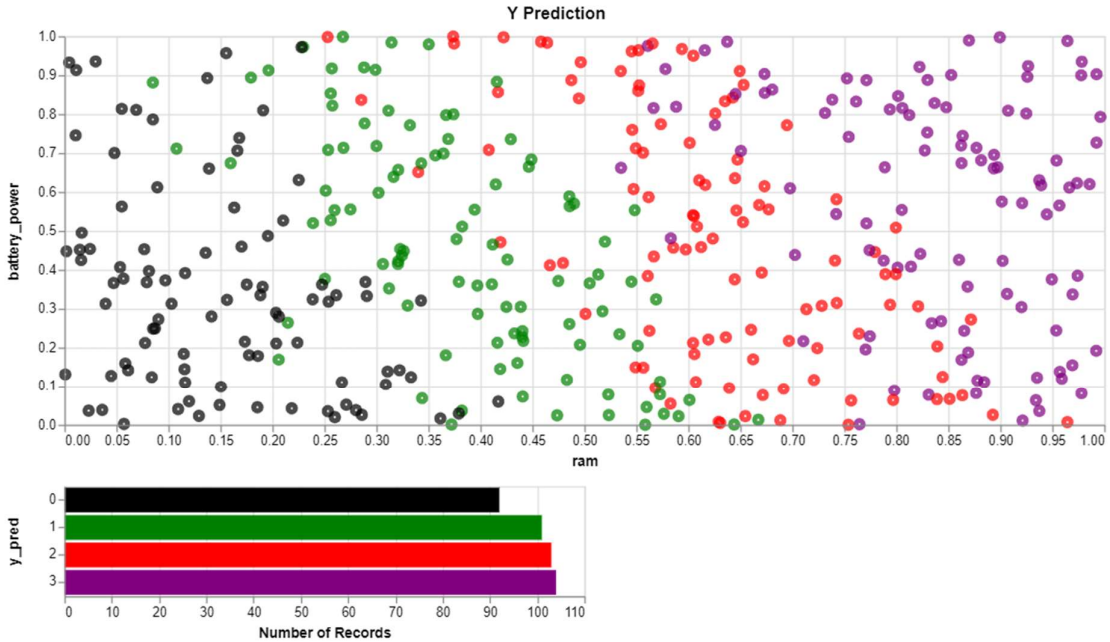
Example 1 :-



Example 2 :-




Y True
battery_power
ram
y_true, y_pred, y_pred
0
1
2
3
y_true
Number of Records


Y Prediction
battery_power
ram
y_pred
Number of Records

7. Conclusion & Future Scope :-

This work can be concluded with the comparable results of both Feature selection algorithms and Random Forest Classifier. This combination has achieved maximum accuracy and selected minimum but most appropriate features. It is important to note that in Forward selection by adding irrelevant or redundant features to the data set decreases the efficiency of both classifiers. While in backward selection if we remove any important feature from the data set, its efficiency decreases. The main reason of low accuracy rate is low number of instances in the data set. One more thing should also be considered while working that converting a regression problem into classification problem introduces more error.

30.

Future work is suggested to extend this research and find more sophisticated solution to the given problem and more accurate tool for price estimation.

Also, in future we can use other types of Models to predict more accurately. If this dataset contains more data then prediction will become more accurate. We can add more feature in smartphone to gain more complex situation and more pricing.

Then we can achieve more better results by changing our models according to conditions.

Outcomes of the Work :-

1 Cost prediction is the very important factor of marketing and business. To predict the cost same procedure can be performed for all types of products for example Cars, Foods, Medicine, Laptops etc.

2 Best marketing strategy is to find optimal product (with minimum cost and maximum specifications). So, products can be compared in terms of their specifications, cost, manufacturing company etc.

3 By specifying economic range a good product can be suggested to a costumer.

Future Work Extension :-

1 More sophisticated artificial intelligence techniques can be used to maximized the accuracy and predict the accurate price of the products.

2 Software or Mobile app can be developed that will predict the market price of any new launched product.

3 To achieve maximum accuracy and predict more accurate, more and more instances should be

added to the data set. And selecting more appropriate features can also increase the accuracy. So, data set should be large and more appropriate features should be selected to achieve higher accuracy.

References :-

[1] Sameerchand Pudaruth . “Predicting the Price of Used Cars using Machine Learning Techniques”, International Journal of Information & Computation Technology. ISSN 0974-2239 Volume 4, Number 7 (2014), pp. 753764

[2] Shonda Kuiper, “Introduction to Multiple Regression: How Much Is Your Car Worth? ” , Journal of Statistics Education · November 2008

[3] Mariana Listiani , 2009. “Support Vector Regression Analysis for Price Prediction in a Car Leasing Application”. Master Thesis. Hamburg University of Technology.

[4] Limsombunchai, V. 2004. “House Price Prediction: Hedonic Price Model vs. Artificial Neural Network”, New Zealand Agricultural and Resource Economics Society Conference, New Zealand, pp. 25-26. 2004

[5] Kanwal Noor and Sadaqat Jan, “Vehicle Price Prediction System using Machine Learning

Techniques” , International Journal of Computer Applications (0975 – 8887) Volume 167 – No.9, June 2017.

[6] Mobile data and specifications online available from https://www.gsmarena.com/ (Last Accessed on Friday, December 22, 2017, 6:14:54 PM)

[7] Introduction to dimensionality reduction, A computer science portal for Geeks. https://www.geeksforgeeks.org/dimensionality-reduction/ (Last Accessed on Monday , Jan 2018 22, 3 PM)

[8] Ethem Alpaydın, 2004. Introduction to Machine Learning, Third Edition. The MIT Press Cambridge, Massachusetts London, England

[9] InfoGainAttributeEval-Weka Online available from http://weka.WrapperattributEval/doc.dev/weka/attributeS election/InfoGainAttributeEval.html (Last Accessed in Jan 2018)

[10] Thu Zar Phyu, Nyein Nyein Oo. Performance Comparison of Feature Selection Methods. MATEC Web of Conferences42, (2016).

❖ *CERTIFICATE* ❖

This is to certify that **MS. MONISHITA GHOSH** of Government College of Engineering and Textile Technology, Berhampore, registration number: 181110110017 of 2018-2019, has successfully completed a project on "**PREDICTION OF PRICE RANGE TYPE OF MOBILE HANDSET**" **using Machine Learning (Python3)** under the guidance of **Mr. Titas Roy Chowdhury**.

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

Globsyn Finishing School

❖ *CERTIFICATE* ❖

This is to certify that **MR. GOLAM KIBRIA** of Government College of Engineering and Textile Technology, Berhampore, registration number: 181110110013 of 2018-2019, has successfully completed a project on "**PREDICTION OF PRICE RANGE TYPE OF MOBILE HANDSET**" **using Machine Learning (Python3)** under the guidance of **Mr. Titas Roy Chowdhury**.

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

Globsyn Finishing School

❖ *CERTIFICATE* ❖

This is to certify that **MR. AMIT BASKEY** of Government College of Engineering and Textile Technology, Berhampore, registration number: 181110110003 of 2018-2019, has successfully completed a project on "**PREDICTION OF PRICE RANGE TYPE OF MOBILE HANDSET**" **using Machine Learning (Python3)** under the guidance of **Mr. Titas Roy Chowdhury**.

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

Globsyn Finishing School

❖ *CERTIFICATE* ❖

This is to certify that **MR. RAKTIM MIDYA** of Government College of Engineering and Textile Technology, Berhampore, registration number: 181110110024 of 2018-2019, has successfully completed a project on "**PREDICTION OF PRICE RANGE TYPE OF MOBILE HANDSET**" **using Machine Learning (Python3)** under the guidance of **Mr. Titas Roy Chowdhury**.

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

Globsyn Finishing School

❖ *CERTIFICATE* ❖

This is to certify that **MR. APRATIM SARKAR** of Government College of Engineering and Textile Technology, Berhampore, registration number: 181110110006 of 2018-2019, has successfully completed a project on "**PREDICTION OF PRICE RANGE TYPE OF MOBILE HANDSET**" **using Machine Learning (Python3)** under the guidance of **Mr. Titas Roy Chowdhury**.

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

Globsyn Finishing School