SVKM's NMIMS

Mukesh Patel School of Technology Management &

EngineeringA PROJECT REPORT ON

Prediction of Social Network Advertising using Naïve Bayes Classification

Faculty mentor:

MR. SURAJ PATIL

(Assistant Professor)

Submitted by:

AMOGH PRADEEP

(B240)

NISHIT SARAF

(B243)

ADITYA SHARMA

(B248)

MPSTME, NMIMS

Shirpur Campus

Course: B TECH CS

Batch: 2019-2023

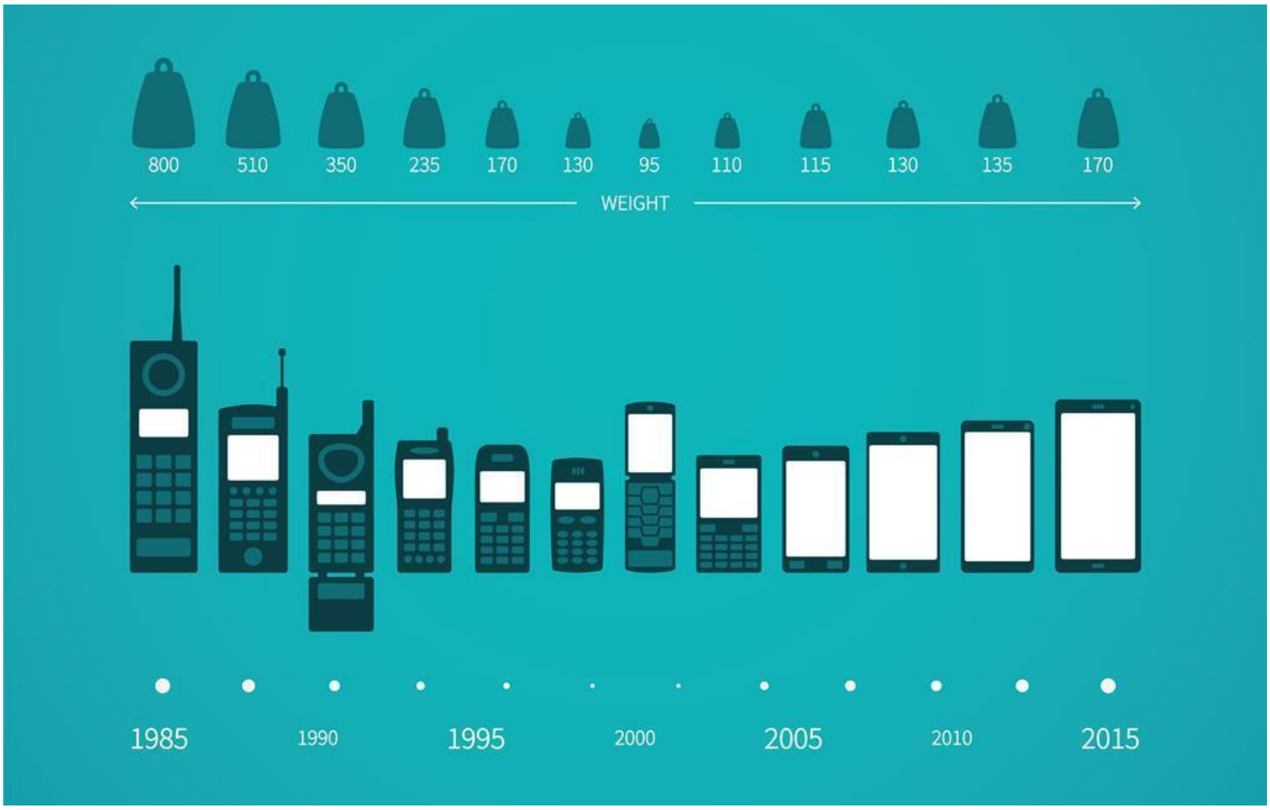
Introduction:

A mobile phone, cell phone, cellphone, 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.



Problem Statement:

In this report we will try to use the Mobile Price Classification dataset to create a model which can predict the price range of a mobile phone. We will test different algorithms like ID3, Naive Bayesian, and K-Means and compare their performance.

About the Data:

Dataset has 2000 rows and 21 columns.

Columns

Column Name	Description
battery_power	Total energy a battery can store in one charge 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 megapixels
pc	Primary Camera megapixels
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
px_height	Pixel Resolution Height
px_width	Pixel Resolution Width
ram	Random Access Memory in Megabytes
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

There are no categorical values in the dataset. Every column is numeric.

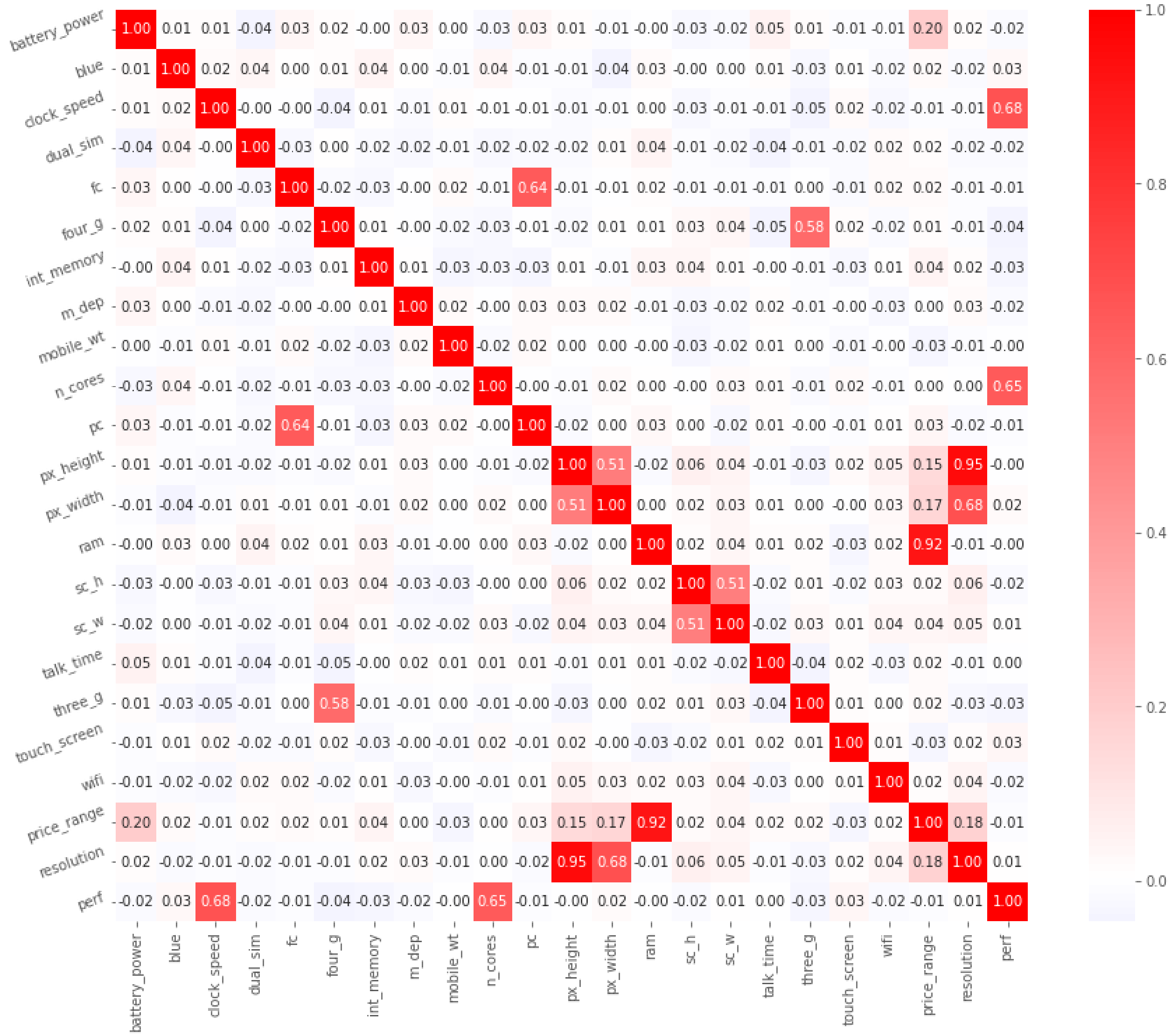
	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	pc
1	842	0	2.2	0	1	0	7	0.6	188	2	2
2	1021	1	0.5	1	0	1	53	0.7	136	3	6
3	563	1	0.5	1	2	1	41	0.9	145	5	6
4	615	1	2.5	0	0	0	10	0.8	131	6	9
5	1821	1	1.2	0	13	1	44	0.6	141	2	14
6	1859	0	0.5	1	3	0	22	0.7	164	1	7

px_height	px_width	ram	sc_h	sc_w	talk_time	three_g	touch_screen	wifi	price_range
20	756	2549	9	7	19	0	0	1	1
905	1988	2631	17	3	7	1	1	0	2
1263	1716	2603	11	2	9	1	1	0	2
1216	1786	2769	16	8	11	1	0	0	2
1208	1212	1411	8	2	15	1	1	0	1
1004	1654	1067	17	1	10	1	0	0	1

Overview of the Dataset

<u>Data Visualization:</u>

Let's start by plotting the correlation matrix for our dataset (or the heat map)

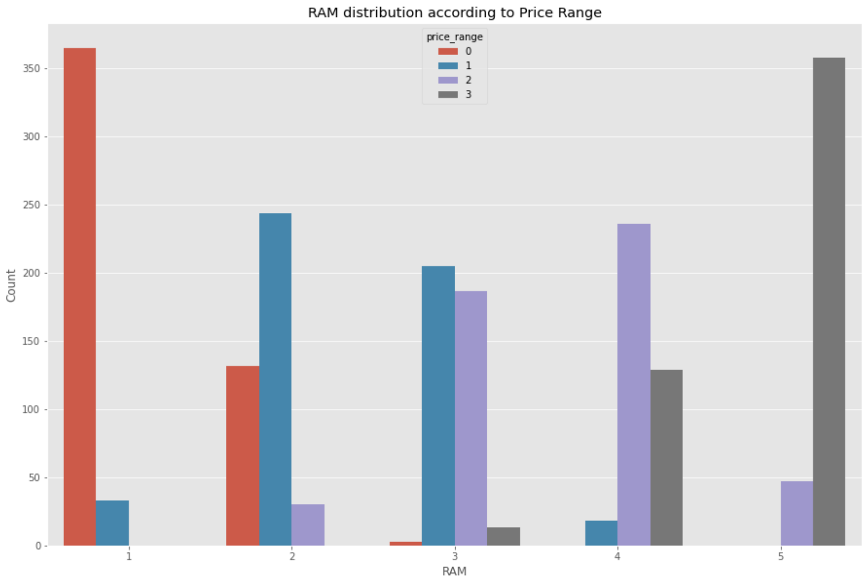


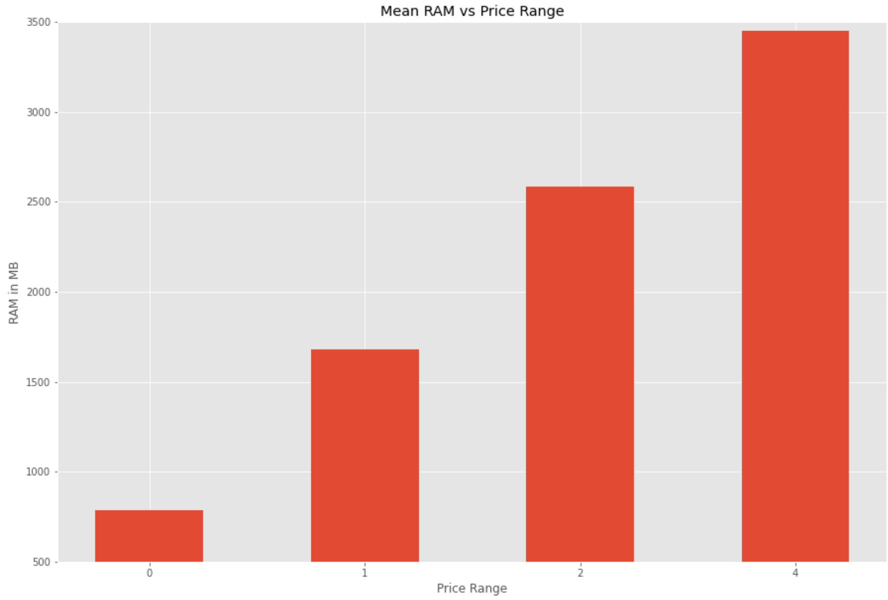
Note : I have added `perf` and `resolution` columns which are `(n_cores * clock_speed)` and `(px_height * px_width)` respectively

Strong correlations exist between the following variables :

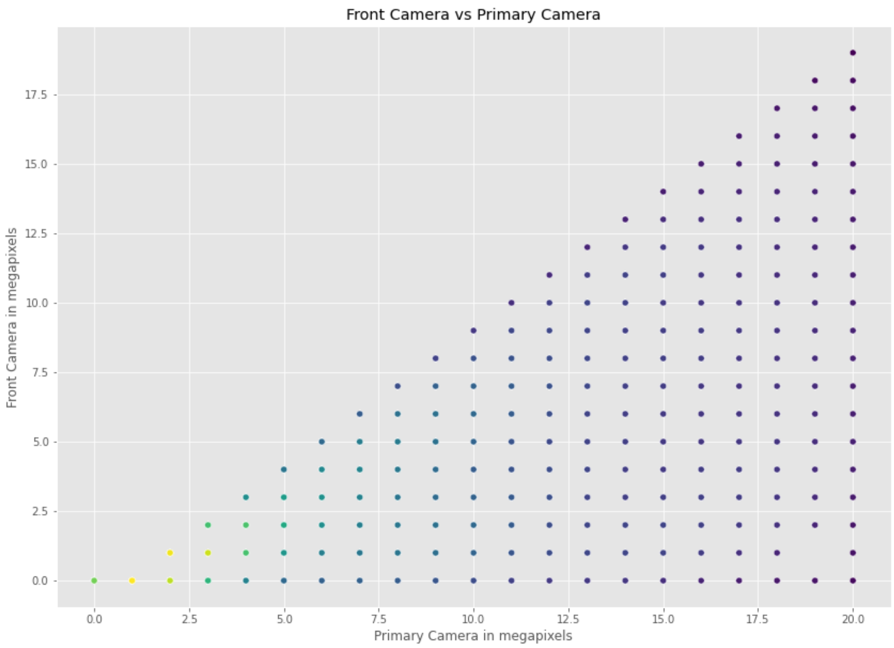
- `price_range` and `ram` have strong positive correlation of 0.92
- `fc` and `pc` have strong positive correlation of 0.64

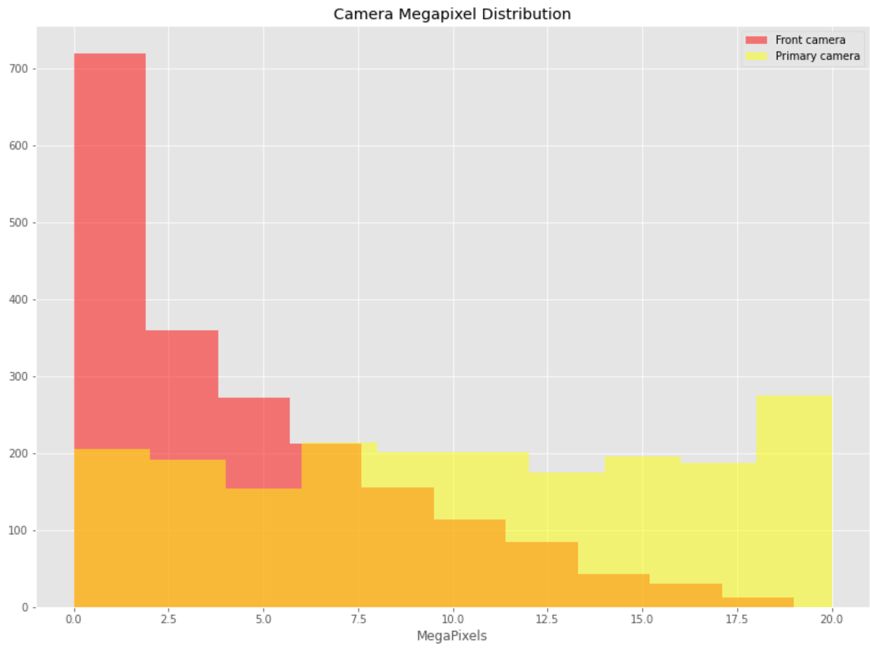
Let's observe the relationship between `ram` and `price_range`



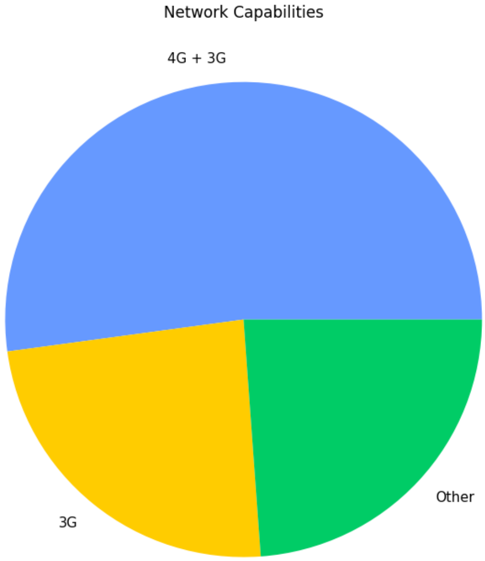


Relationship between `front camera` and `primary camera`

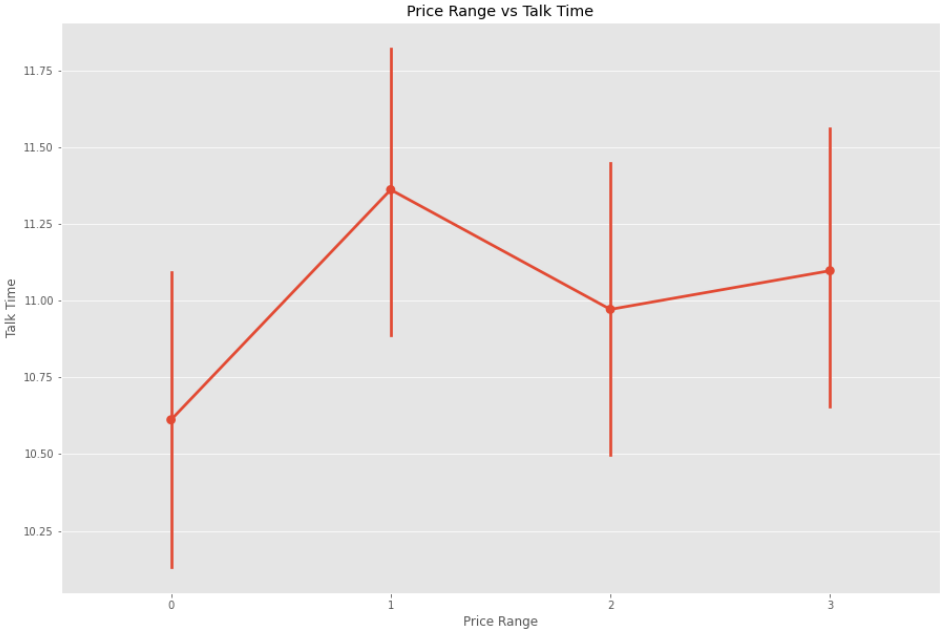




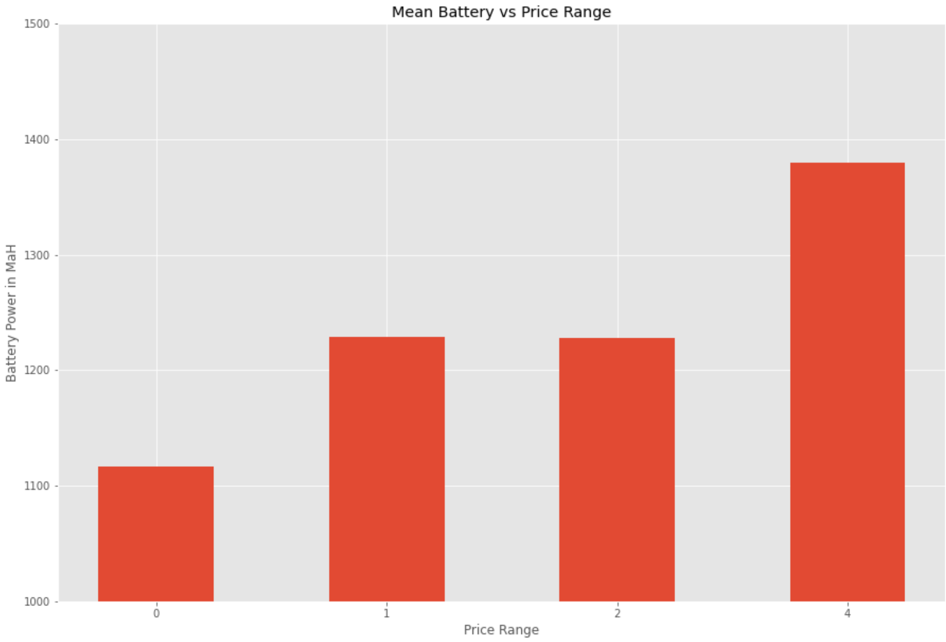
Network capabilities of smartphones



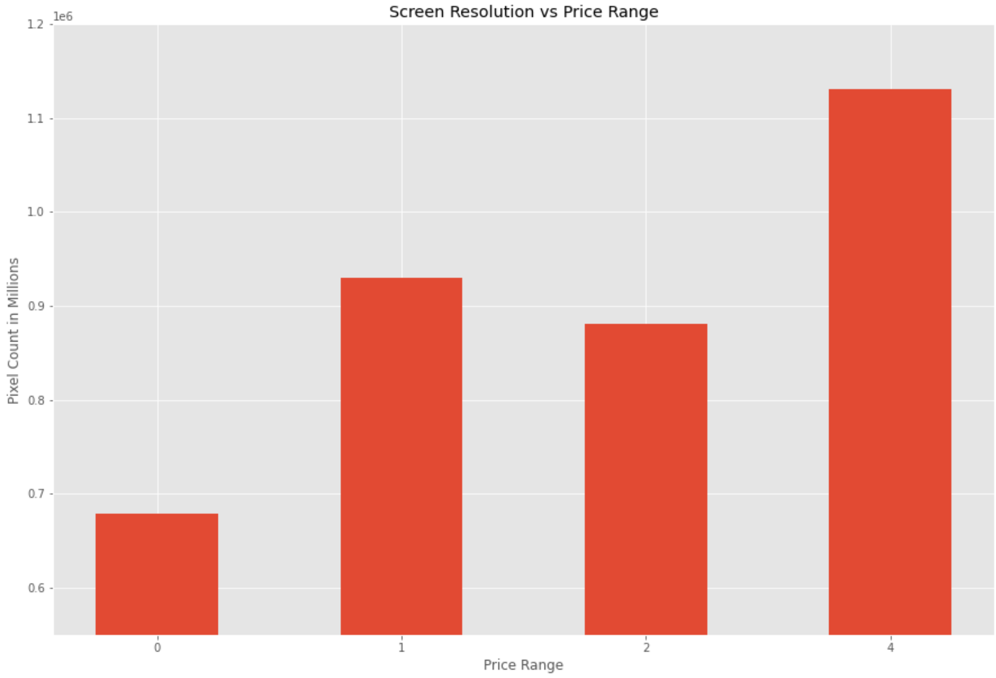
Let's see how `talk time` changes with `price range`



Let’s see how `battery power` changes with `price range`



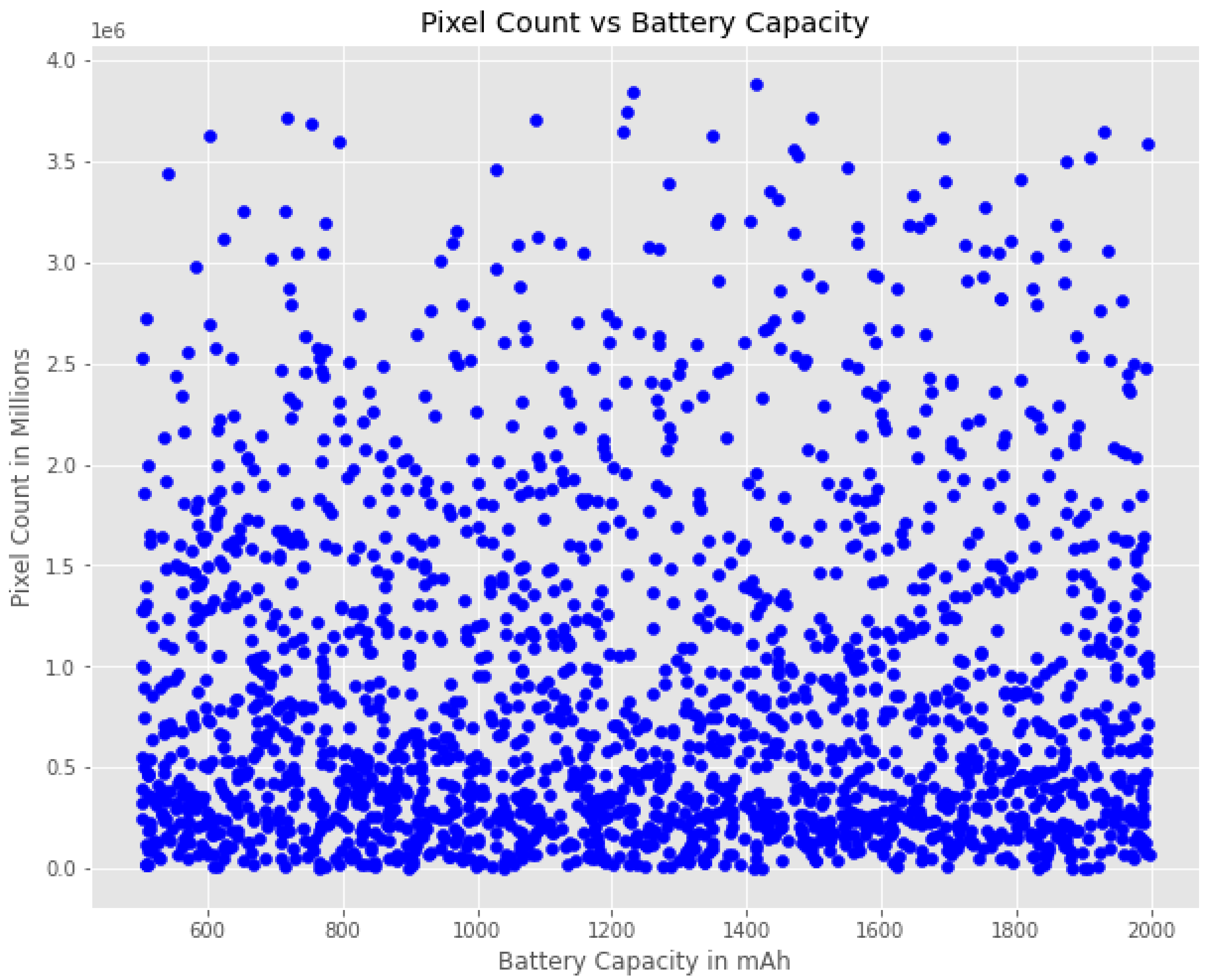
Let’s see how `screen resolution` changes with the `price range`.


Screen Resolution vs Price Range
1e6
1.2
1.1
1.0
0.9
0.8
0.7
0.6
Pixel Count in Millions
0
1
2
4
Price Range

Case Study

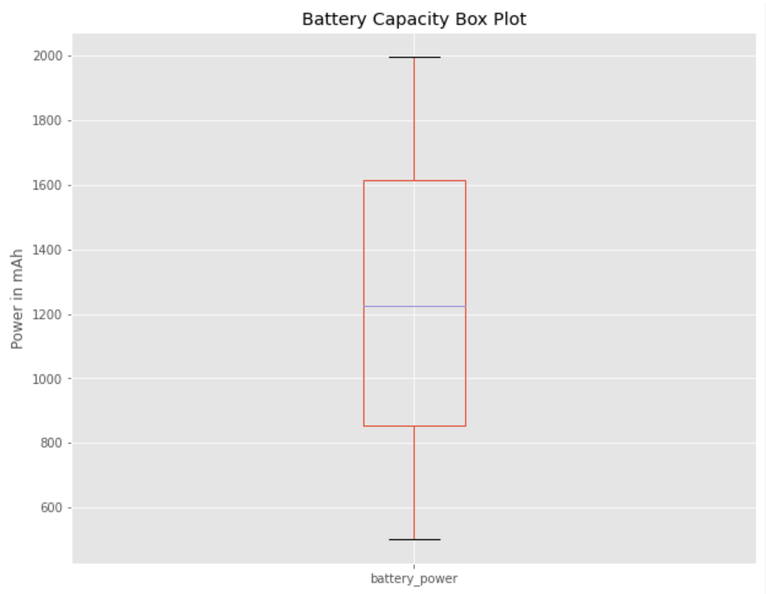
Hypothesis : Battery capacity should show a positive correlation with screen resolution. In modern smartphones, the screen is the major source of power draw accounting for almost 50% of total power use. To counter this smartphone makers should equip larger screens with bigger batteries.

Let's first draw a scatterplot between `battery_power` and `px_count`

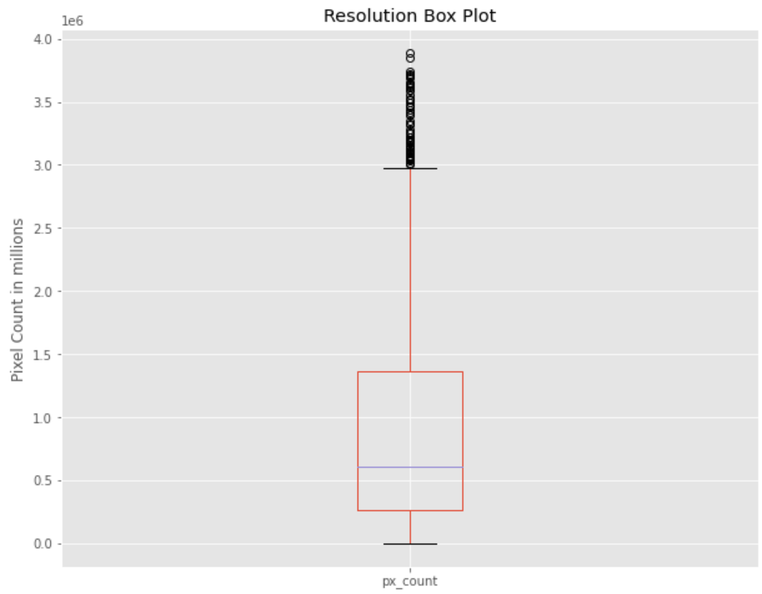


No relation can be seen.

This can be due to outliers in data. Let's check for outliers in `px_count` and `battery_power`



Battery Power variable is distributed normally, and has no outliers. Let's check `px_count`



Large Number of outliers present in `px_count`

Let's remove outliers, by eliminating rows which are not in the interquartile range (IQR).

Code

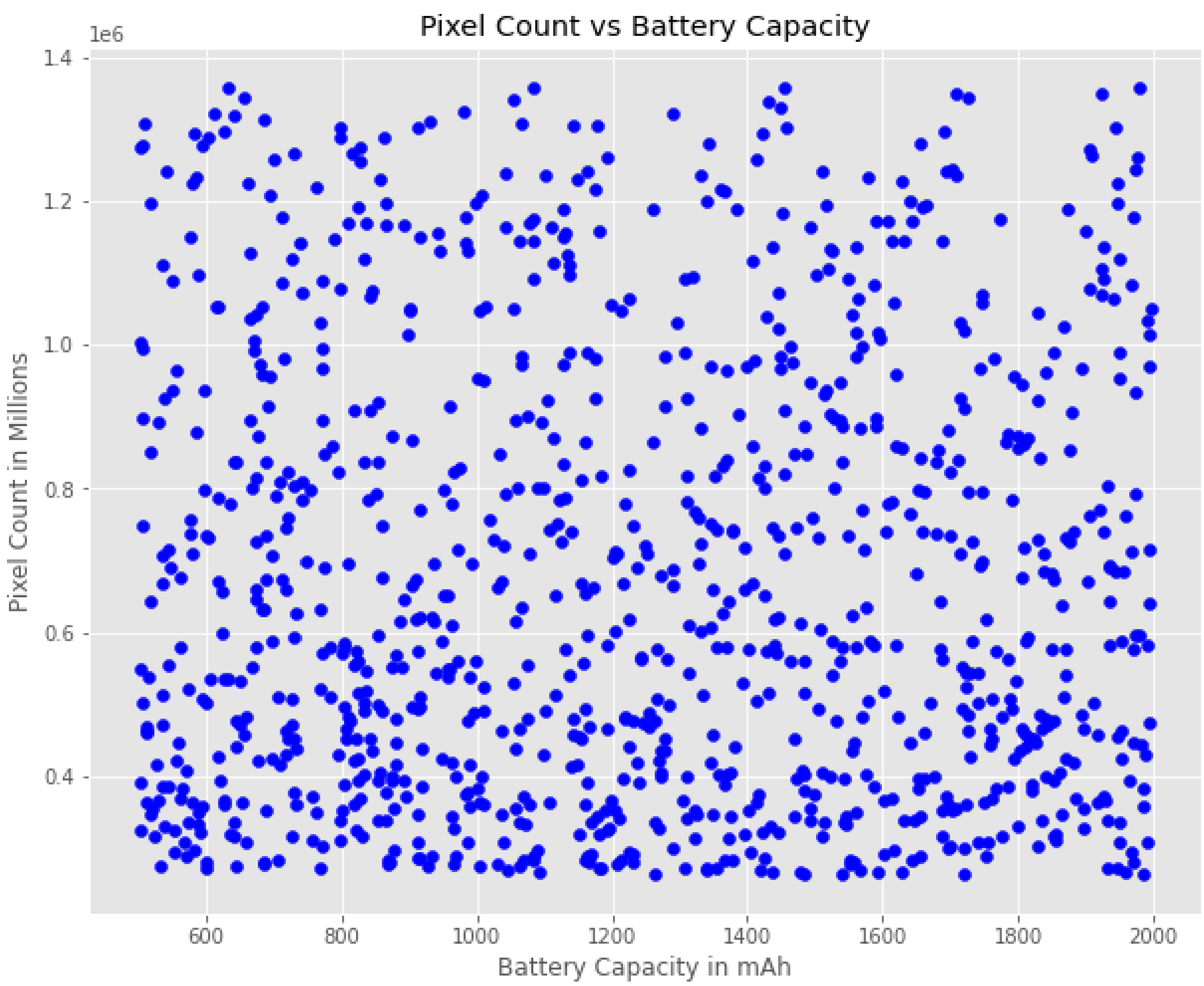
```
per_25 = train["px_count"].quantile(0.25)
per_75 = train["px_count"].quantile(0.75)

iqr = per_75 - per_25
lower_fence = per_25
upper_fence = per_75

print("lower boundary {lower} and upper boundary {upper}\nIQR =
{iqr}".format(lower = lower_fence, upper = upper_fence, iqr = iqr))
```

```
pixels = []
battery_cap = []

# removing rows which are not in the interquartile range.
for i in range (len(train)):
    if (train.iloc[i]["px_count"] > lower_fence and
train.iloc[i]["px_count"] < upper_fence):
        pixels.append(train.iloc[i]["px_count"])
        battery_cap.append(train.iloc[i]["battery_power"])

# creating new dataframe with IQR values
pixels_vs_battery = pd.DataFrame()
pixels_vs_battery["px_count"] = pixels
pixels_vs_battery["battery_power"] = battery_cap
```

Output

```
lower boundary 263200.5 and upper boundary 1359027.25
IQR = 1095826.75
```

Let's draw the scatter plot again.



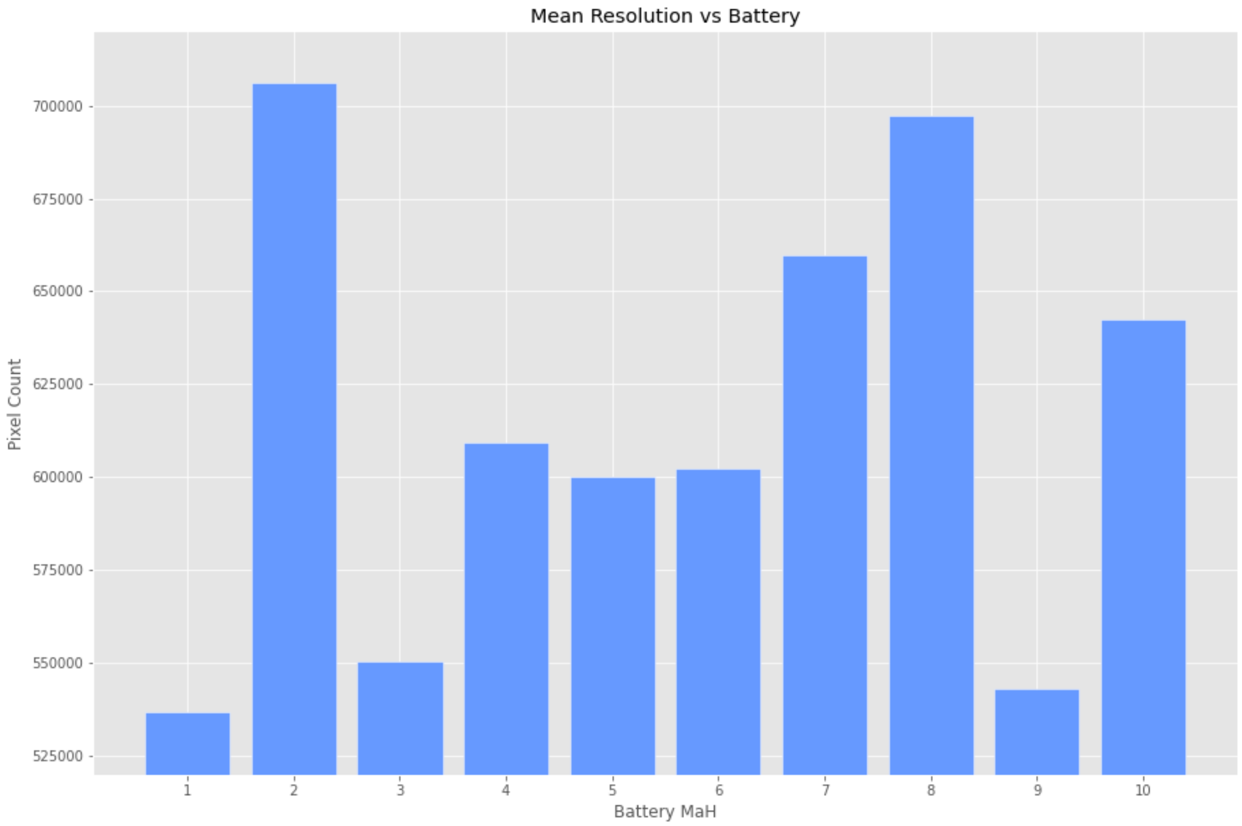
Still no pattern visible.

Finally let's try to plot the median `px_count` per `battery_power` group.

Code

```
pixels_vs_battery['battery_power_category'] =
pd.cut(pixels_vs_battery['battery_power'], bins = 10, labels = [var for
var in range (10)])
# calculating median per battery_power_category
median_res =
pixels_vs_battery.groupby("battery_power_category")["px_count"].median(
)
```

Barplot



A slight increasing pattern can be observed, but many exceptions such as battery group 2 and group 9 are present.

Data doesn't support the hypothesis.

Hypothesis rejected.

Data Preprocessing

- ### Missing Values

  The dataset is complete. No missing values present.

```
> print(nrow(df) - sum(complete.cases(df)))
[1] 0
```

- ### Encoding Categorical Data

  Not needed as no categorical data is present in the dataset.

- ### Splitting dataset into Train and Test

  We have opted for a 80-20 split between train and test.

```
train_idx <- sample(1 : nrow(df), size = floor(0.8 * nrow(df)), replace = FALSE)
train <- df[train_idx, ]
test <- df[-train_idx, ]
```

  Number of rows in train : 1600

  Number of rows in test : 400

- ### Feature Scaling

Although feature scaling can be applied in our dataset, we are not planning to use models which require feature scaling like KNN (K-Nearest Neighbors), Neural Networks, Linear Regression, and Logistic Regression.

Model Training

K-Means Clustering

Intuition behind K-Means

The k-means algorithm searches for a pre-determined number of clusters within an unlabeled multidimensional dataset. It accomplishes this using a simple conception of what the optimal clustering looks like:

- The "cluster center" is the arithmetic mean of all the points belonging to the cluster.
- Each point is closer to its own cluster center than to other cluster centers.

Those two assumptions are the basis of the k-means model. We will soon dive into exactly how the algorithm reaches this solution, but for now let's take a look at a simple dataset and see the k-means result.

Creating Model

K-means doesn't require the dataset to be split into test and train. So we'll work with a complete dataset df.

```
km<-kmeans(df,4)
```

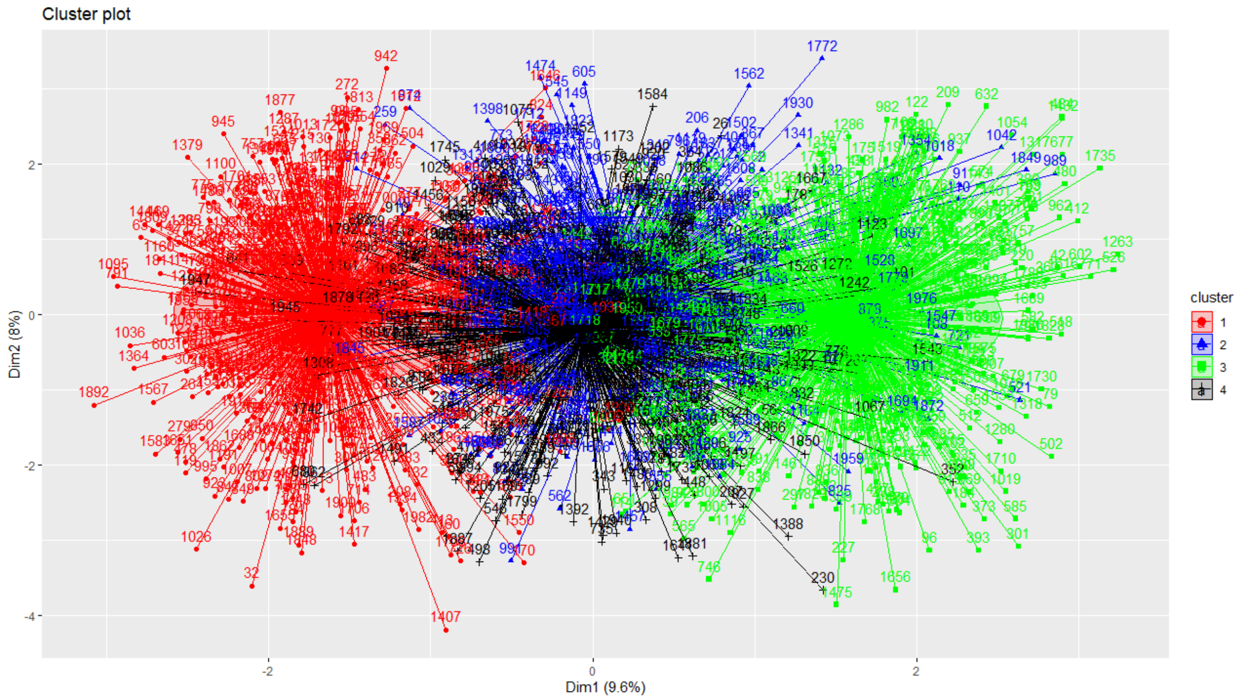
Predicting model and checking accuracy

```
df$price_range<-as.factor(km$cluster)
confusionMatrix(table(price_range,df$price_range))
```

Output

```
Confusion Matrix and Statistics


price_range   1   2   3   4
          1 478   9   0  13
          2 113 175   0 212
          3   0 116 127 257
          4   0  23 459  18

Overall Statistics

               Accuracy : 0.399
                 95% CI : (0.3775, 0.4208)
    No Information Rate : 0.2955
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.1987

 Mcnemar's Test P-Value : NA
```

Visualizing Clusters



Observation

K-means Clustering has an accuracy of only ~40%. We'll try to improve this by testing out new models based on ID3 and Naive Bayesian classifier.

Decision Tree

Intuition

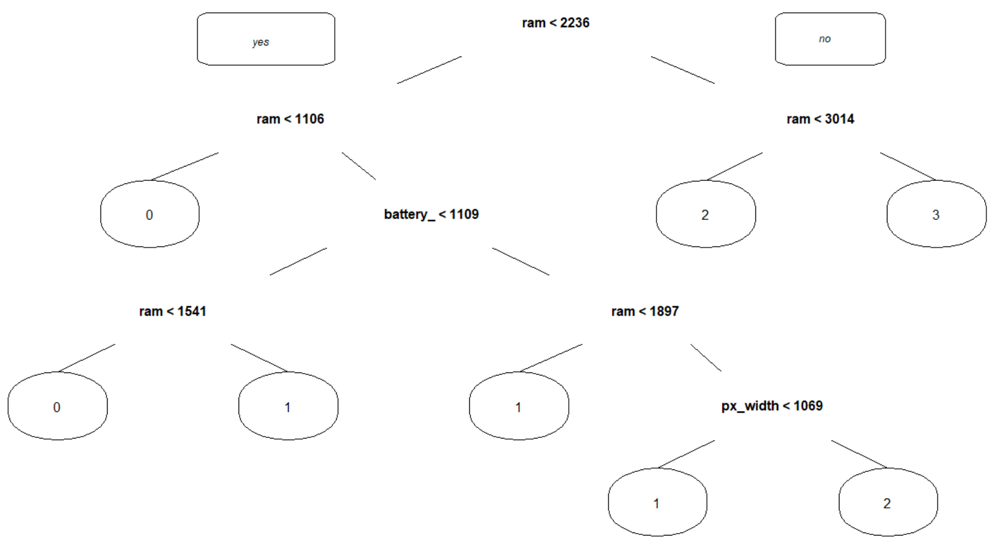
A decision tree is a type of supervised machine learning used to categorize or make predictions based on how a previous set of questions were answered. The model is a form of supervised learning, meaning that the model is trained and tested on a set of data that contains the desired categorization.

- Root node: The base of the decision tree.
- Splitting: The process of dividing a node into multiple sub-nodes.
- Decision node: When a sub-node is further split into additional sub-nodes.
- Leaf node: When a sub-node does not further split into additional sub-nodes; represents possible outcomes.
- Pruning: The process of removing sub-nodes of a decision tree.
- Branch: A subsection of the decision tree consisting of multiple nodes.

Creating Model

Creating a decision tree through rpart function.

```
tree<-rpart(price_range~.,data=df)
summary(tree)
```



Predicting model and checking accuracy

Code

```r
p<-predict(object = tree,test,type="class")

confusionMatrix(table(p,test$price_range))
```

Output

```
Confusion Matrix and Statistics


p     0   1   2   3
  0 102  13   0   0
  1   7  72  19   0
  2   0  12  64  16
  3   0   0  20  75

Overall Statistics

               Accuracy : 0.7825
                 95% CI : (0.7388, 0.822)
    No Information Rate : 0.2725
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7096

 Mcnemar's Test P-Value : NA
```

Observation

Decision Tree gives us an accuracy of ~79%. This is a lot better than K-means clustering.

Naive Bayesian

Intuition

It is a theorem that works on conditional probability. Conditional probability is the probability that something will happen, given that something else has already occurred. The conditional probability can give us the probability of an event using its prior knowledge.

Conditional probability:

$$P(A \mid B) = \frac{P(B \mid A) \cdot P(A)}{P(B)}$$

Conditional Probability

Where,

P(A): The probability of hypothesis H being true. This is known as the prior probability.

P(B): The probability of the evidence.

P(A|B): The probability of the evidence given that hypothesis is true.

P(B|A): The probability of the hypothesis given that the evidence is true.

Creating Model

```
library(e1071)
model<-naiveBayes(price_range~., train)
```

Predicting model and checking accuracy

Code

```
p<-predict(model,test)
confusionMatrix(table(p,test$price_range))
```

Output

```
Confusion Matrix and Statistics


p     0  1  2  3
  0 98  8  0  0
  1 11 74 17  0
  2  0 15 78  7
  3  0  0  8 84

Overall Statistics

               Accuracy : 0.835
                 95% CI : (0.7949, 0.87)
    No Information Rate : 0.2725
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7798

 Mcnemar's Test P-Value : NA
```

Observation

Naive Bayesian gives us the best accuracy of 83.5%.

Conclusion

Using independent variables such as ram, camera quality, screen resolution, battery life, and network capabilities we successfully predicted the price range of a smartphone.

We also visualized trends between variables such as ram, battery capacity, network, talk time, screen resolution, and camera quality.

We used classification techniques such as Naive Bayesian and Decision Tree algorithm, and clustering techniques such as K-Means to build an accurate model.

Out of all these techniques we conclude that Naive Bayesian is the most accurate on this dataset with an accuracy of 83.5% .