NITTE EDUCATION TRUST | N.M.A.M. INSTITUTE OF TECHNOLOGY
(An Autonomous Institution affiliated to Visvesvaraya Technological University, Belagavi)
Nitte – 574 110, Karnataka, India
Department of Computer Science and Engineering
(B.E. Computer Science & Engineering Program Accredited by NBA from 2018-19 to 2021-25)

Report on Mini Project

Data Analysis On Mobile Phone Price Classification

Course Code : 21CSA31

Course Name : R Programming

Semester: III SEM Section: A

Submitted To:

Dr. Anisha P Rodrigues
Associate Professor
Department of Computer Science
and Engineering

Submitted By:
Avisha V Shetty (4NM21CS041)
Deepthi(4NM21CS053)
Devishree Bangera(4NM21CS054)

Date of submission:
17 January 2023

Anisha 17/1/23
Signature of Course Instructor

ABSTRACT

The project includes various types of data visualization and data analysis on mobile price classification. Data visualization is a technique used for graphical representation of data. Visualization of any data through charts and graphs gives a clear image and idea of the data set. When there is a large data set it becomes a difficult task to understand the data. Data visualization and analysis makes it easy to view and comprehend the patterns in data. It is more understandable and effective. Hence we have come up with the following data visualization and data analysis techniques in this project.

- Structure (str())
- Summary
- Correlation Matrix : A correlation matrix is simple table which displays the correlation coefficients for different variables.
- Scatter Plot : Scatter plot is type of plot used to display the relationship between two numerical variables, and plots one dot for each observation. It requires two vectors of same length (for X and Y axis).
- Line Plot : A line chart is a graph that connects a series of points by drawing line segments between them. These points are ordered in one of their coordinate value. Line charts are usually used in identifying the trends in data.
- Box Plot : Boxplots is a measure of how well data is distributed across a data set. It is also useful in comparing the distribution of data across data sets by drawing boxplots of each of them.
- Histogram : Histogram is similar to bar chart but the difference is that it groups the values into continuous ranges. Each bar in histogram represents the height of the number of values present in that range.

This project also includes the animations of the above mentioned plots. Animation brings any concept to life with ease. Animation deepens visual understanding much more than traditional diagrams and make the concepts more interesting.

TABLE OF CONTENTS

INTRODUCTION

R is a data science-oriented programming language that offers more than 18,000 data science packages. R allows us to perform feature selection, to perform all type machine learning and deep learning tasks, to apply various machine learning methods, such as classification, regression, clustering. They can be used to quickly identify outliers, compare different groups or variables, and communicate findings to others. Additionally, R has a wide variety of graphical packages and functions, such as ggplot2, lattice, and base R graphics, that makes it easy to create high-quality, customizable visualizations. R is widely used for data analysis, data visualization, and statistical modeling. R is open-source software, which means that it is free to use and distribute. It has a large and active community of users and developers, which contributes to its ongoing development and improvement. R is particularly useful for working with large and complex data sets, as well as for creating advanced visualizations and statistical models.

R contains functionality for many plot types including graphic maps, mosaic plots, biplots, and the list goes on. Probability distribution play a vital role in statistics and by using R we can easily handle various types of probability distribution such as Binomial Distribution, Normal Distribution, Chi-squared Distribution and many more. It provides a large, coherent and integrated collection of tools for data analysis.

Mobile phones have become an integral part of our daily lives and the market for these devices is growing rapidly. As a result, it is important for manufacturers, retailers, and consumers to be able to accurately classify and price of mobile phones based on their features and specifications. In this project we are using a dataset called mobile price classification, which is a collection of data that is used to classify mobile phones based on their prices. The dataset typically includes a number of different features or attributes that describes each mobile phone, such as the id, battery_power, blue, clock_speed, dual sim, fc, four_g, int memory, m depth, m weight, n cores, pc, px height, px width, ram, sc_h, sc_w, talk_time, 3g, touch screen, wifi, price range. In R, graphs are used to visualize data and make it easier to understand patterns, trends, and relationships within the data.

Mobile phones are an essential part of our daily lives, and their price vary greatly depending on various factors such as brand, features and storage capacity. Data visualization is an effective tool for understanding and analyzing the complex relationships between these factors and the price of mobile phones. In this analysis , we will explore different visualizations to classify mobile prices and identify patterns in the data.

PROBLEM STATEMENT

Aim of this project is to perform analysis on the “mobile price classification” data set using R programming and also to use various visualizing concepts to represent the contents of the dataset.

OBJECTIVES

The objective of data analysis on mobile phone price classification is to use analysis and visualization techniques to classify mobile phones into different price ranges based on their characteristics and features. The features include mobile weight, height, width, Bluetooth, etc.

The aim of this data analysis is to identify patterns, trends, and relationships in the data, such as correlations between variables or clustering of observations. Large datasets can create problems in understanding the data. Also the process of understanding the large dataset is time consuming and may not be accurate. So to overcome these complications we have come up with different types of data analysis and data visualization.

The goal of the project is to plot varieties of graphs along with its animations so as to make analyzing of data simple and easy and also to create interactive visualizations that clearly communicates the key factors affecting the prices of the mobile phones and to identify any patterns or trends in the data. This involves using various tools and techniques such as histograms, scatter plots, box plots, etc. to create visual representations of the data, which can help identify patterns and trends that might not be immediately found from the raw data.

METHODOLOGY

Following Packages are used in the analysis:

1. ***'ggplot2'*** : The 'ggplot2' package in R Programming Language also termed as Grammar of Graphics is a free, open-source, and easy-to-use visualization package widely used in R. It includes several layers on which it is governed.

   The layers are as follows**:**

   - Data: Dataset used to plot the graph.
   - Aesthetics: The data is to map onto the Aesthetics attributes such as x-axis, y-axis, colour, fill, size, labels, alpha, shape, line width, line type.
   - Geometrics: How our data being displayed using point, line, histogram, bar, boxplot.
   - Facets: It displays the subset of the data using Columns and rows.
   - Statistics: Binning, smoothing, descriptive, intermediate.
   - Coordinates: The space between data and display using Cartesian, fixed, polar, limits.
   - Themes: Non-data link used to customize the appearance of the various elements of a plot, such as the background, axis labels, and legend.

   The basic syntax of ggplot is:

   ```
   ggplot(data, mapping=aes()) + geometric object
   ```

   Here geometric object is the type of plot you want to show.

   The most common object are:

   - Point: `` `geom_point()` ``
   - Bar: `` `geom_bar()` ``
   - Line: `` `geom_line()` ``
   - Histogram: `` `geom_histogram()` ``

2. ***'gganimate'*** : The 'gganimate' package in R is an extension of the 'ggplot2' package, which allows you to create animations from static ggplot2 plots. It allows us to easily create animations by defining a transition between different states of the plot, such as different time periods, groups, or sub-plots. The gganimate package also allows us to control the animation using various arguments such as the frame rate, duration, and easing functions.

   The `animate()` function in the gganimate package is used to create animations from a ggplot2 object. It allows us to easily create animations by defining a transition between different states of the plot, such as different time periods, groups, or sub-plots. The `animate()` function also allows us to control the animation using various arguments such as the frame rate, duration, and easing functions.

3. ***'corrplot'*** : 'corrplot' is a R package that is used to visualize the correlation matrix of a dataset. The package provides a variety of plot types and customization options to help you effectively communicate the patterns in the correlation matrix.

4. ***'plotrix'*** : The 'plotrix' package in R programming is a collection of various plotting functions and tools that extend the functionality of the base R plotting functions. It provides additional functionality for creating plots such as 3D scatterplots, dot plots, parallel coordinates plots, and more. The package consists of several functions. One of them is pie3D().

5. '***reshape2'*** : The 'reshape2' package is a popular package in R for reshaping and reconstructing data. It provides several functions for reshaping data, including 'melt()', 'cast()', and 'dcast()'. The 'reshape2' package is very useful when working with data that needs to be transformed or reshaped to fit a specific format or structure, or when working with data that is in a long format that needs to be transformed into a wide format.

The code snippet to load the packages is as follows:

```
library(ggplot2)
library(gganimate)
library(corrplot)
library(plotrix)
library(reshape2)
```

Before starting with the analysis, set the working directory using the `getwd()` and `setwd()` functions so that we can access the file from the directory that contains the dataset.

Code snippets:

Reading the csv file:

The code snippet used to read the csv file is :

```
a=read.csv("data.csv", header=TRUE, sep=",")
```

The `read.csv()` function in R is used to read a csv file and create a data frame from it.
The code reads a csv file named "test.csv" and creates a data frame named "a" with the data contained in the file.
`'header=TRUE'` indicates that the first row contains the column names and `'sep=","'` indicates that the file uses a comma as the separator between values.

Rows and Columns of the dataframe:

`nrow(a)` function is used to get the number of rows in the data frame 'a'.
Output:

```
[1] 1000
```

`ncol(a)` function is used to get the number of columns in the data frame 'a'.
Output:

```
[1] 22
```

`colnames(a)` function is used to get the column names in the data frame 'a'.
These functions are useful to get a quick overview of the data frame and to check if the data is loaded correctly.
Output:

```
 [1] "id"            "battery_power" "blue"          "clock_speed"
 [5] "dual_sim"      "fc"            "four_g"        "int_memory"
 [9] "m_dep"         "mobile_wt"     "n_cores"       "pc"
```

```
[13] "px_height"     "px_width"      "ram"           "sc_h"
[17] "sc_w"          "talk_time"     "three_g"       "touch_screen"
[21] "wifi"          "price_range
```

Structure:

`str(a)` function is used to display the internal structure of the data frame ‘a’.
The structure function returns a concise summary of the object's properties, including the number of observations and variables, the names of the variables, and the data types of the variables. It can also be used to check for missing data or other issues that may need to be addressed before analysing the data.

Summary:

`summary(a)` is used to generate a brief summary of the key statistics of the data frame ‘a’.
The function produces a summary of the distribution of the data, including measures of central tendency (such as the mean, median, and mode) and measures of variability (such as the standard deviation and range). It also shows the number of missing values and the number of distinct values in a variable.

Correlation Matrix:

```
cord=cor(head(a,100))
```

The `cor()` function is used to compute the correlation matrix for the dataframe ‘a’.
`head(a,100)` is used to select first 100 rows of the data frame ‘a’ , and the correlation matrix of this is found using `cor()` and the result is stored in ‘cord’.

```
png("corrplot.png",width = 800,height=800)
corrplot(cord, method="square",bg="black",type="full",tl.col="black",
         title="Correlation Plot" ,tl.cex=0.7)
dev.off()
```

`png()` function is used to create a png image of name ‘corrplot.png’ and set the width and height of the image.
`corrplot()` function is used to create a correlation plot using the correlation matrix.
The `corrplot()` function takes in several arguments such as ‘cord’ which is the correlation matrix, ‘method’ which is set to ‘square’ to display the correlation matrix as squares, ’bg’ which sets the background colour as ‘black’, ‘type’ which it is set to ‘full’ to display all the correlations, ‘tl.col’ sets the text colour as ‘black’, ‘title’ gives the title to the plot as "Correlation Matrix" and ‘tl.cex’ sets the text label size to 0.7.
`dev.off()` function is used to close the image device created by the `png()` function.

Occurrence of each price range :

```
freq = table(a$price_range)
```

The `table()` function is used to create a frequency table of the values in data frame. The above code gives the table that contains the frequency of each value in the column "price_range" of the dataframe "a". The result is stored in the variable ‘freq’.
The output of the code is:

```
  0   1   2   3
242 234 248 276
```

Pie Chart:

```
colors = c("gray52","gray65","black","gray90")
```

'colors' is a vector of colors "gray52","gray65","black","gray90" which are used to color the different sections of the pie chart.

```
png("pie_chart1.png")
```

`png()` functions creates png image of name 'pie_chart1.png'

```
pie = pie(freq, col=colors,main="Pie chart representing Price Range")
```

The above code creates a pie chart of the frequency table stored in the variable 'freq' using the `pie()` function.

The `pie()` function contains arguments such as 'freq' which is a variable that contains the frequency table , the 'colors' vector is assigned to 'col' and main' is used to provide the title , "Pie chart Representing Price Range".

```
legend("bottomright",c("0(Low Cost)","1(Medium Cost)","2(High
Cost)","3(Very High Cost)"),cex=0.8,fill=colors,bg="grey")
```

`legend()` function is used to create legend to the pie chart. It takes in arguments that specifies the position, label, size, colour and background colour of the legend.

3D Pie Chart:

Creating a 3D pie chart in R programming requires the package called plotrix.

```
png("3Dpie_chart.png")
```

`png()` creates a png image of name '3Dpie_chart.png'.

```
piepercent = round(100*freq/sum(freq))
```

The above line of code is used to calculate the percentage of each value in the frequency table.

```
pie3D(freq,labels=piepercent,col=colors,    main="3D    Pie    Chart
Representing   Price Range",explode=0.05)
```

The `pie3D()` function takes several arguments, such as 'freq' which is a variable that contains the frequency table, 'labels' which is used to show label on the pie chart (in this case, it's set to 'piepercent' which is the percentage of each value in the frequency table), 'col' contains the vector 'colors' which is used to colour different section of the pie chart, 'main' sets the title to the pie chart as '3D Pie Chart Representing Price Range' and 'explode' sets the size of the gap between the different sections of the pie chart.

```
legend("bottomright",c("0(Low Cost)","1(Medium Cost)","2(High
Cost)","3(Very High Cost)"),cex=0.8,fill=colors,bg="grey")
```

`legend()` function is used to create legend to the pie chart. It takes in arguments that specifies the position, label, size, colour and background colour of the legend.

Scatter plot:

```
s2=ggplot(a,aes(battery_power,talk_time,size=mobile_wt,
   colour=price_range ))+
   geom_point(alpha =0.7,show.legend=TRUE)+facet_wrap(~price_range)+
   theme(plot.background=element_rect(fill="grey"))+
   labs(title = "Scatter Plot",x = 'Battery power ', y = 'Talk Time')
```

The `ggplot()` function is used for creating plots using the 'ggplot2' R package. The first input argument defines the input data that is stored in the variable 'a'. The `aes()` function allows aesthetic mapping of the input variables by defining the internal_memory to be displayed on the X axis and ram to be displayed on the Y axis. The size of each data point will now be dependent

on the mobile_weight variable .Finally, the colour of the data points will depend on the price range it belongs.

`geom_point()` function is used to define the alpha transparency (i.e. the data point will be translucent as defined by the alpha parameter of 0.7; the lower the value the more translucent they become) of each data point . `'show.legend=TRUE'` is used to display the legend to the scatter plot.

`labs()` function defines the plot title, X axis title and the Y axis title.

`theme_bw()` is a function in the 'ggplot2' package for R that is used to change the appearance of a ggplot plot to a black and white theme. This function sets the background color, grid lines, and text color to shades of gray.

The `theme()` function is used to customize the appearance of a plot. The `theme()` function can be added to a ggplot plot to change various elements of the plot such as the background color, axis labels, and text size. Here the 'plot.background' element controls the appearance of the background of a plot. It includes properties such as the background color, grid lines, and borders.Here the plot background colour is set to 'grey'.

Above plot is animated using the following code:

```
q1=ggplot(a,aes(int_memory,ram,size=mobile_wt,colour=price_range))+
   geom_point(alpha = 0.7, show.legend = TRUE)  +
  labs(title='Price Range={frame_time}',x='Internal Memory',y='Ram')+
  transition_time(price_range)+ ease_aes('linear')+ theme_bw()
animate(q1)
```

The `transition_time()` function from the gganimate package is used to control the animation of a plot over time. The `ease_aes()` function is part of the 'gganimate' package and it is used to control the easing of the animation of a plot.

The `animate()` function will create an animation of the plot, where price_range is the variable in the data that represents the time and the animation will change linearly over time. Linear easing is a simple linear transition, which means the animation will progress at a constant rate. Similarly another scatter plot is plotted with X axis displaying Battery Power and Y axis displaying Talk Time.

```
s2=ggplot(a,aes(battery_power,talk_time,size=mobile_wt,
   colour=price_range ))+
   geom_point(alpha =0.7,show.legend=TRUE)+facet_wrap(~price_range)+
   theme(plot.background=element_rect(fill="grey"))+
   labs(title = "Scatter Plot",x = 'Battery power ', y = 'Talk Time')
```

The extra function here is the `facet_wrap()` function. The `facet_wrap()` function splits the plot to multiple sub-plots.

```
facet_wrap(~price_range)
```

This will create a grid of scatter plots, where each panel shows the data for a different values of price_range.

The second scatter plot is also animated using the following code where touch_screen is given as the transition time,

```
q2=ggplot(a,aes(battery_power,talk_time,size=mobile_wt,
   colour=price_range))+
   geom_point(alpha = 0.7, show.legend = TRUE)+
   facet_wrap(~price_range)+
   labs(title='Touch screen= {frame_time}',x ='Battery Power',
   y ='Talk Time')+ transition_time(touch_screen)+ ease_aes('linear')
animate(q2)

ggsave('scatter1.png', width=8, height=8)
ggsave('scatter2.png', width=8, height=8)
```

`ggsave()` function is used to save the scatter plot to a file in png format.

Box plot:

Boxplots are a measure of how well data is distributed across a data set. This divides the data into three quartiles.

```
w1=data.frame(a$fc,a$pc,a$sc_w,a$sc_h,a$price_range)
dmelt1<-melt(w1,id="a.price_range",variable="Features")
print(dmelt1)
```

First, a new data frame ‘w1’ is created by selecting the columns fc (front cam mega pixels) , pc (primary camera mega pixels), sc_w (screen width of mobile), sc_h (screen height of mobile), and price_range from the original data frame ‘a’.

Then, it uses the `melt()` function from the ‘reshape2’ package to reshape the data frame ‘w1’ into a long format with two columns: 'Features' and 'value' which makes the data ready for plotting. The result of melt function is stored in ‘dmelt1’.

```
b1=ggplot(dmelt1, aes(x= Features,y= value,
   fill=as.factor(a.price_range)))+ geom_boxplot()+
   theme(axis.text.x=element_text(angle=45,hjust=1),
   plot.background=element_rect(fill="grey"))+ labs(title="Box Plot")
```

The `ggplot()` function is used for creating plots using the ‘ggplot2’ R package. . The first input argument defines the input data that is stored in the variable ‘dmelt1’. The `aes()` function allows aesthetic mapping of the input variables by defining the in ‘Features’ to be displayed on the X axis and ‘value’ to be displayed on the Y axis. ‘fill’ represents the variable 'a.price_range' as factor variable.

`geom_boxplot()` is a function that belongs to ‘ggplot2’ package and creates a box plot, which is a standardized way of displaying the distribution of a dataset.

The `theme()` function is used to customize the appearance of the plot. In this case, it rotates the x-axis text by 45 degrees and set the background color as grey.

`labs()` function is used to add title to the box plot as “Box Plot”.

```
ggsave('box1.png', width=8, height=8)
```
`ggsave()` function is used to save the box plot to a file in png format.

Above plot is animated using the following code:

```
animb1 = b1 + transition_time(a.price_range)+
         labs(title="price_range:{frame_time}")+ ease_aes('linear')
animate(animb1)
```
The `transition_time()` function from the 'gganimate' package is used to control the animation of a plot over time. The `ease_aes()` function is part of the 'gganimate' package and it is used to control the easing of the animation of a plot. The `animate()` function will create an animation of the plot, where price_range is the variable in the data that represents the time and the animation will change linearly over time.

Line plot:

A line plot is a type of graph that displays data as a series of points connected by lines. It is often used to visualize changes in a variable over time or to compare the values of multiple variable.

```
new=data.frame(a$m_dep,a$mobile_wt,a$px_height,a$px_width,a$id)
new1=head(new,50)
m=melt(new1,id="a.id",variable= "Features")
print(m)
```
First, a new data frame 'new' is created by selecting the columns m_dep(mobile depth),mobile_weigt,px_height(pixel resolution height),px_width(pixel resolution height),and id from the original dataframe 'a'.

Then, it uses the `melt()` function from the 'reshape2' package to reshape the data frame 'new' into a long format with two columns: 'Features' and 'value' which makes the data ready for plotting. The result of melt function is stored in 'm'.

```
plot=ggplot(m,aes(x=a.id,y=value,colour=Features))+ geom_line()+
     labs(title="Line Plot")+theme_classic()+
     theme(panel.background=element_rect(fill="#333333"),
     plot.background = element_rect(fill="#CCCCCC"))
```

The `ggplot()` function is used for creating plots using the 'ggplot2' R package. The first input argument defines the input data that is stored in the variable 'm'. The `aes()` function allows aesthetic mapping of the input variables by defining the in 'a.id' to be displayed on the X axis and 'value' to be displayed on the Y axis. 'colour' represent the 'Features' column.

`geom_line()` is a function in the ggpplot2 library, used to create line plot from a dataset.

The `theme()` function is used to customize the appearance of the plot. In this case, it is used to set the panel and plot background colour.

`labs()` function is used to add title to the box plot as "Line Plot".

```
ggsave('line11.png', width=8, height=8)
```
`ggsave()` function is used to save the box plot to a file in png format.

Above plot is animated using the following code:

```
animline=plot+transition_reveal(a.id)+labs(title='id={frame_along}')+
         ease_aes('linear')
animate(animline)
```

creates an animation of the plot with the `transition_reveal()` function. This function takes the variable 'a.id' as its argument, which means that the animation will reveal the data points one by one according to the distinct values of 'a.id'.

`labs()` function is used to add the title to the animation with the 'id' number of the current frame. The `ease_aes('linear')` function is used to make the animation a linear transition. The animate() function will create an animation of the plot.

Histogram:

A histogram is a graphical representation of the distribution of a dataset. It is an estimate of the probability distribution of a continuous variable. It is a way to summarize large amount of data by grouping data into ranges. It is a bar chart showing the frequency of values in each range of values.

```
hist1=ggplot(a,aes(x=px_height,fill=as.factor(blue)))+
      geom_histogram(aes(y=..density..),position='identity',
      alpha=0.5,bins=50)+
      labs(title="Pixel Resolution Height Histogram Plot",
      x="Pixel Height",y="Density")+
      theme_dark()+theme(panel.background=element_rect(fill="black"),
      plot.background = element_rect(fill="#CCCCCC"))
```

The `ggplot()` creates the basic ggplot object using the dataframe 'a' and mapping the X-axis to the variable 'px_height' and fill to the variable 'blue' as factor variable.

`geom_histogram()` adds the histogram geometry to the plot.

`aes(y=..density..)`: This argument maps the y-axis to the density of the variable 'px_height'.
`position='identity'`: This argument is used to stack the bars of the histogram.
`alpha=0.5`: This argument is used to specify the transparency of the bars in the histogram.
`bins=50`: This argument is used to specify the number of bins in the histogram.

`labs()` function defines the plot title as "Pixel Resolution Height Histogram Plot", X axis title as 'Pixel Height' and the Y axis title as 'Density'.

The `theme()` function is used to customize the appearance of the plot. In this case, it is used to set the panel and plot background colour. `theme_dark()` changes the theme of the plot to dark.

Similarly, another histogram is plotted,which creates the basic ggplot object using the data frame 'a' and mapping the x-axis to the variable 'px_width and fill to the variable 'wifi' as factor variable.

`geom_histogram()` adds the histogram geometry to the plot .

`geom_density(alpha=0.4)`: This line adds the density curve on top of histogram.

`facet_grid(price_range~.)`: This line create a facet grid by price_range variable.

`labs()` function defines the plot title as "Pixel Resolution Width Histogram Plot", X axis title as 'Pixel Width' and the Y axis title as 'Density'.

The `theme()` function is used to customize the appearance of the plot. In this case, it is used to set the panel and plot background colour. `theme_bw()` changes the theme of the plot to black and white.

```
hist2=ggplot(a,aes(x=px_width,fill=as.factor(wifi)))+
      geom_histogram(aes(y=..density..),position='identity',
      alpha=0.5,bins = 50)+
      geom_density(alpha=0.4)+facet_grid(price_range~.)+
      labs(title="Pixel Resolution Width Histogram Plot",
      x="Pixel Width",y="Density")+theme_bw()+
      theme(panel.background element_rect(fill="lightblue"),
      plot.background = element_rect(fill="#CCCCCC"))
```

`ggsave()` function is used to save the histogram plot to a file in png format.

```
ggsave('hist1.png', width=8, height=8)
ggsave('hist2.png', width=8, height=8)
```

IMPLEMENTATION

```
getwd()
setwd("C:/Users/shett/Documents/R")

install.packages("ggplot2")
install.packages("gganimate")
install.packages("corrplot")
install.packages("plotrix")
install.packages("reshape2")
install.packages("caret")
library(ggplot2)
library(gganimate)
library(corrplot)
library(plotrix)
library(reshape2)

#read csv file
a=read.csv("data.csv", header=TRUE, sep=",")
#print contents of csv file
print(a)

#number of rows
nrow(a)
#number of columns
ncol(a)
#get column names
colnames(a)

#Structure
str(a)
#summary
summary(a)

#correlation matrix
cord=cor(head(a,100))
print(cord)
#corrplot using correlation matrix
png("corrplot.png",width = 800,height=800)
corrplot(cord, method="square",bg="black",type="full",tl.col="black",
         title="Correlation Plot" ,tl.cex=0.7)
dev.off()

#occurance of each price range
freq=table(a$price_range)
freq
```

#pie chart
```
colors = c("gray52","gray65","black","gray90")
png("pie_chart1.png")
pie = pie(freq, col=colors,main="Pie chart representing Price Range")
legend("bottomright",c("0(Low Cost)","1(Medium Cost)","2(High
Cost)","3(Very High Cost)"),cex=0.8,fill=colors,bg="grey")
dev.off()
```

#3D pie chart
```
png("3Dpie_chart.png")
piepercent = round(100*freq/sum(freq))
pie3D(freq,labels=piepercent,col=colors, main="3D Pie Chart
Representing   Price Range",explode=0.05)
legend("topright", c("0(Low Cost)","1(Medium Cost)","2(High
Cost)","3(Very High Cost)"), cex=0.8, fill=colors, bg="grey")
dev.off()
```

#scatter plot1
```
s1=ggplot(a,aes(int_memory,ram,size=mobile_wt,colour=price_range))+
   geom_point(alpha = 0.7, show.legend = TRUE)+
   labs(title = "Scatter Plot",x = 'Internal Memory', y = 'Ram')+
   theme_bw()+theme(plot.background=element_rect(fill="grey"))
ggsave('scatter1.png', width=8, height=8)
s1
```
#animated scatter plot1
```
q1=ggplot(a,aes(int_memory,ram,size=mobile_wt,colour=price_range))+
   geom_point(alpha = 0.7, show.legend = TRUE)  +
  labs(title='Price Range={frame_time}',x='Internal Memory',y='Ram')+
  transition_time(price_range)+ ease_aes('linear')+ theme_bw()
animate(q1)
```

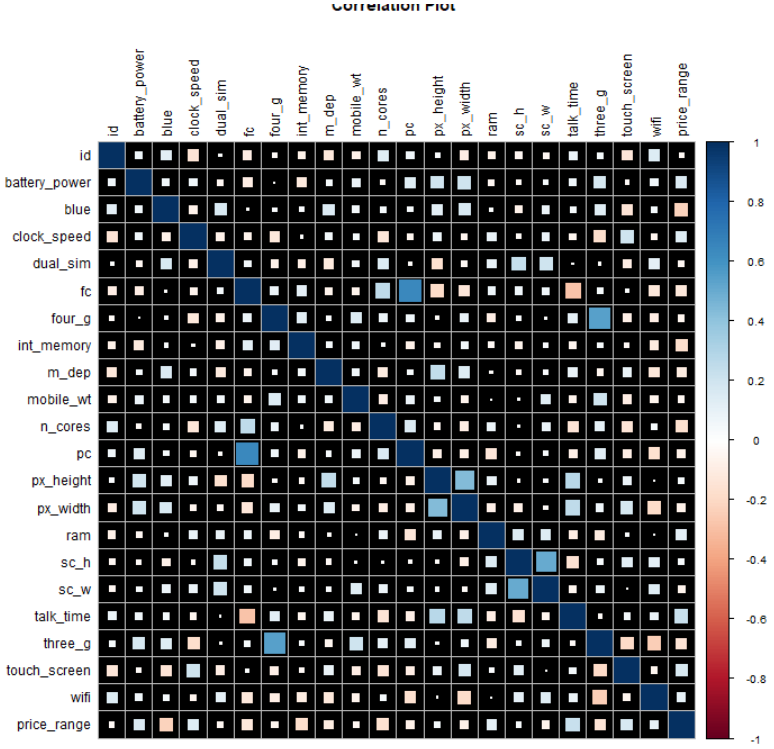
#scatterplot 2
```
s2=ggplot(a,aes(battery_power,talk_time,size=mobile_wt,
   colour=price_range ))+
   geom_point(alpha =0.7,show.legend=TRUE)+facet_wrap(~price_range)+
   theme(plot.background=element_rect(fill="grey"))+
   labs(title = "Scatter Plot",x = 'Battery power ', y = 'Talk Time')
ggsave('scatter2.png', width=8, height=8)
s2
```
#animate scatter plot 2
```
q2=ggplot(a,aes(battery_power,talk_time,size=mobile_wt,
   colour=price_range))+
   geom_point(alpha = 0.7, show.legend = TRUE)+
   facet_wrap(~price_range)+
   labs(title='Touch screen= {frame_time}',x ='Battery Power',
   y ='Talk Time')+ transition_time(touch_screen)+ ease_aes('linear')
animate(q2)
```

#box plot1
```
w1=data.frame(a$fc,a$pc,a$sc_w,a$sc_h,a$price_range)
dmelt1<-melt(w1,id="a.price_range",variable="Features")
```

```
print(dmelt1)
b1=ggplot(dmelt1, aes(x= Features,y= value,
   fill=as.factor(a.price_range)))+ geom_boxplot()+
   theme(axis.text.x=element_text(angle=45,hjust=1),
   plot.background=element_rect(fill="grey"))+ labs(title="Box Plot")
ggsave('box1.png', width=8, height=8)
b1
#animated box plot1
animb1 = b1 + transition_time(a.price_range)+
         labs(title="price_range:{frame_time}")+ ease_aes('linear')
animate(animb1)

#lineplot1
new=data.frame(a$m_dep,a$mobile_wt,a$px_height,a$px_width,a$id)
new1=head(new,50)
m=melt(new1,id="a.id",variable= "Features")
print(m)
plot=ggplot(m,aes(x=a.id,y=value,colour=Features))+ geom_line()+
     labs(title="Line Plot")+theme_classic()+
     theme(panel.background=element_rect(fill="#333333"),
     plot.background = element_rect(fill="#CCCCCC"))
ggsave('line1.png', width=8, height=8)
plot
#animate line plot
animline=plot+transition_reveal(a.id)+labs(title='id={frame_along}')+
         ease_aes('linear')
animate(animline)

#histogram1
hist1=ggplot(a,aes(x=px_height,fill=as.factor(blue)))+
      geom_histogram(aes(y=..density..),position='identity',
      alpha=0.5,bins=50)+
      labs(title="Pixel Resolution Height Histogram Plot",
      x="Pixel Height",y="Density")+
      theme_dark()+theme(panel.background=element_rect(fill="black"),
      plot.background = element_rect(fill="#CCCCCC"))
ggsave('hist1.png', width=8, height=8)
hist1

#histogram2
hist2=ggplot(a,aes(x=px_width,fill=as.factor(wifi)))+
      geom_histogram(aes(y=..density..),position='identity',
      alpha=0.5,bins = 50)+
      geom_density(alpha=0.4)+facet_grid(price_range~.)+
      labs(title="Pixel Resolution Width Histogram Plot",
      x="Pixel Width",y="Density")+theme_bw()+
      theme(panel.background element_rect(fill="lightblue"),
      plot.background = element_rect(fill="#CCCCCC"))
ggsave('hist2.png', width=8, height=8)
hist2
```

RESULTS AND DISCUSSIONS

Correlation Plot:

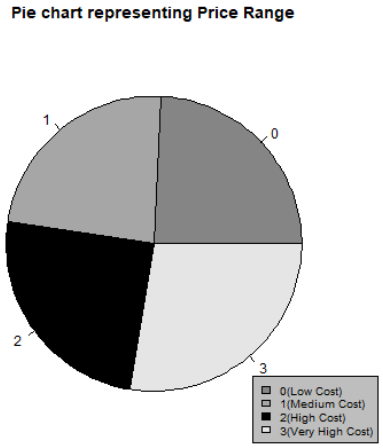


The obtained correlation matrix of the first 100 rows of the data frame is represented in the form of corrplot. Each square in a correlation plot represents the correlation between two different variables. The colour of the square indicates the strength of the correlation, with darker colours indicating a stronger correlation and lighter colours indicating a weaker correlation.

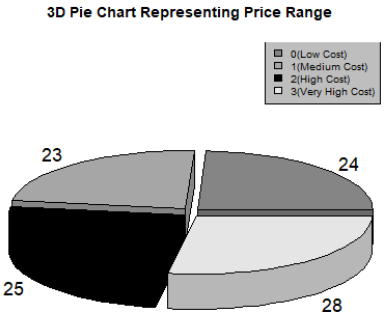
The value of the correlation coefficient ranges between -1 and 1, with -1 indicating a perfect negative correlation, 0 indicating no correlation, and 1 indicating a perfect positive correlation. When the correlation coefficient is close to 1 or -1, it indicates a strong correlation, while when the correlation coefficient is closer to 0, it indicates a weak correlation.

In addition to colour, the correlation plot also uses the size and shape of the square to indicate the correlation coefficient, so that it is easier to identify the strength of the correlation by just looking at the plot.

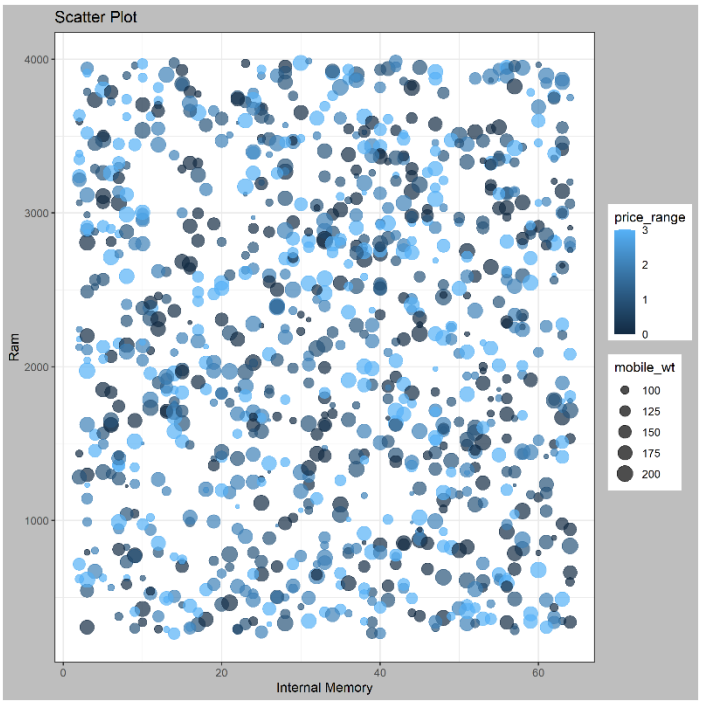
Pie chart and 3D pie :



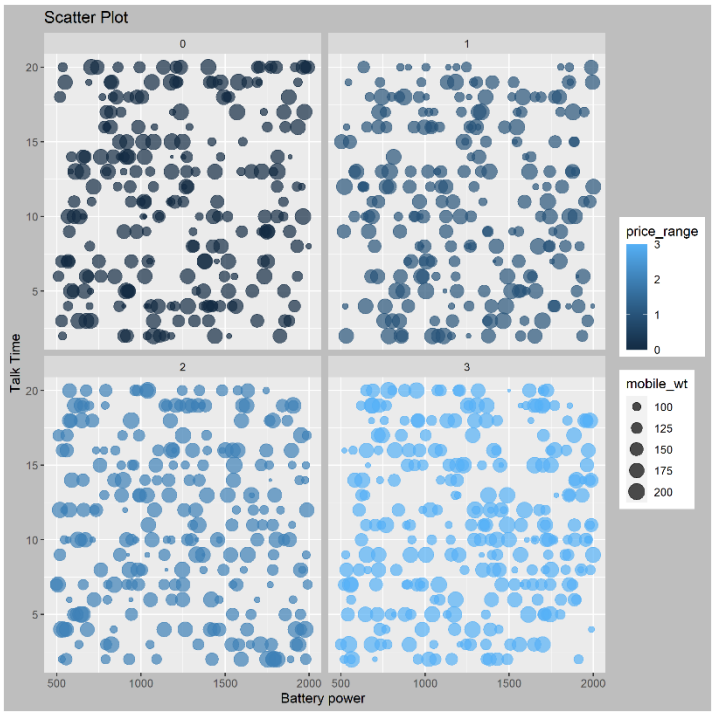
Both pie and 3D pie charts are used to represent the occurences of each price range, for the proper understanding of data. Price range 0 indicates Low Cost,1 indicates Medium Cost,2 indicates High Cost and 3 indicates Very High Cost.



Scatter Plot:



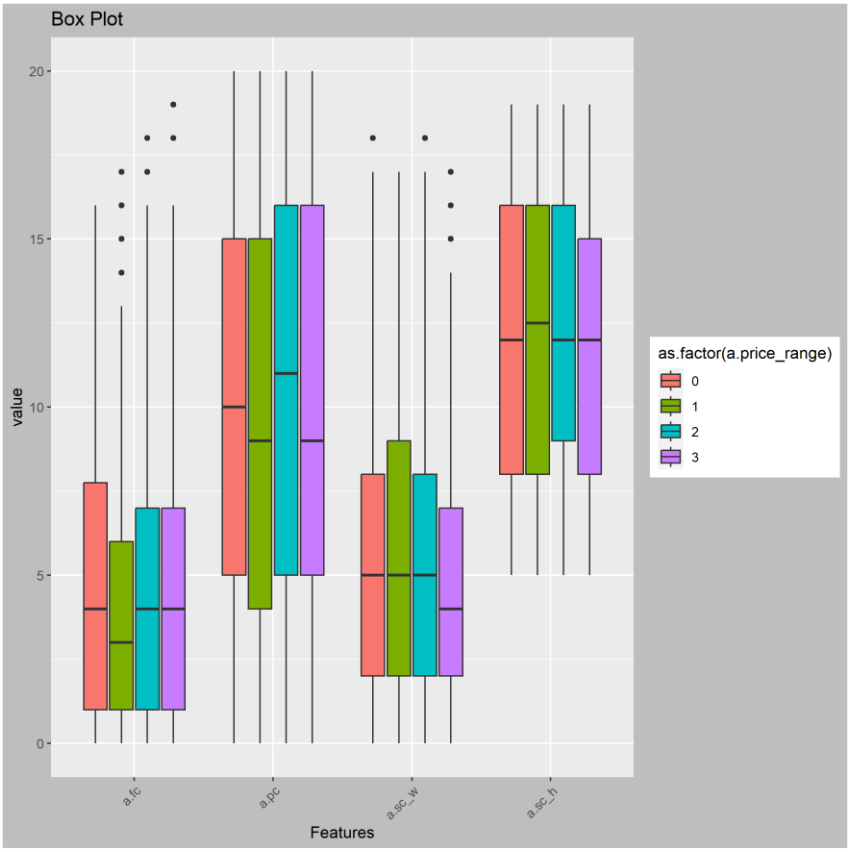
The scatter plot is used to plot the internal memory and ram in the x and y axis respectively. Size of each data point represents the mobile weight and the colour of each data point represents the price range.

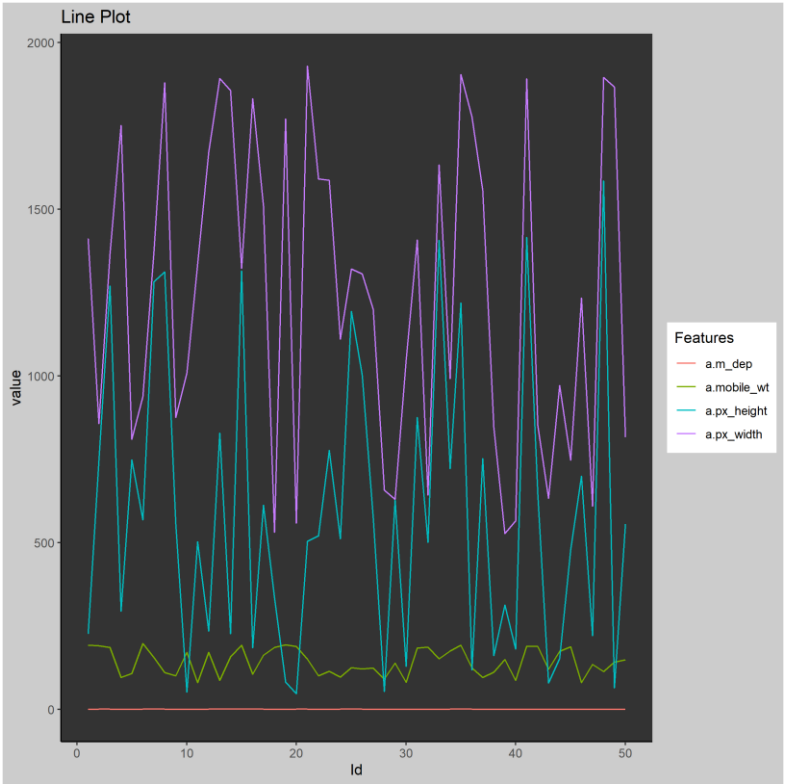


Similarly, another scatter plot is plotted which displays battery power and talk time in the x and y axis respectively. The `facet_wrap()` function is used to split the plot into subplots based on the price range .This gives the clear idea about the battery power and talk time of the different price ranges.

Box Plot:

Box plot is plotted for the selected columns fc (front cam mega pixels), pc (primary camera mega pixels), sc_w (screen width of mobile), sc_h (screen height of mobile), and price_range from the original data frame ‘a’. The selected columns are together considered as Features and are taken along the x axis, their corresponding values are taken in the y axis. Four different colours are used to represent the price range.



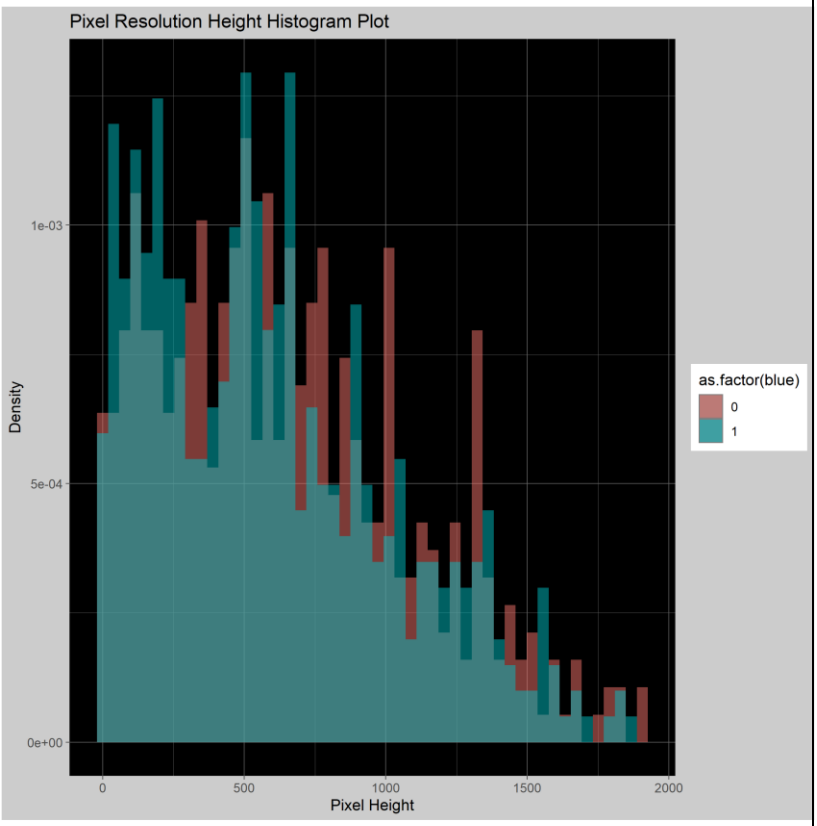


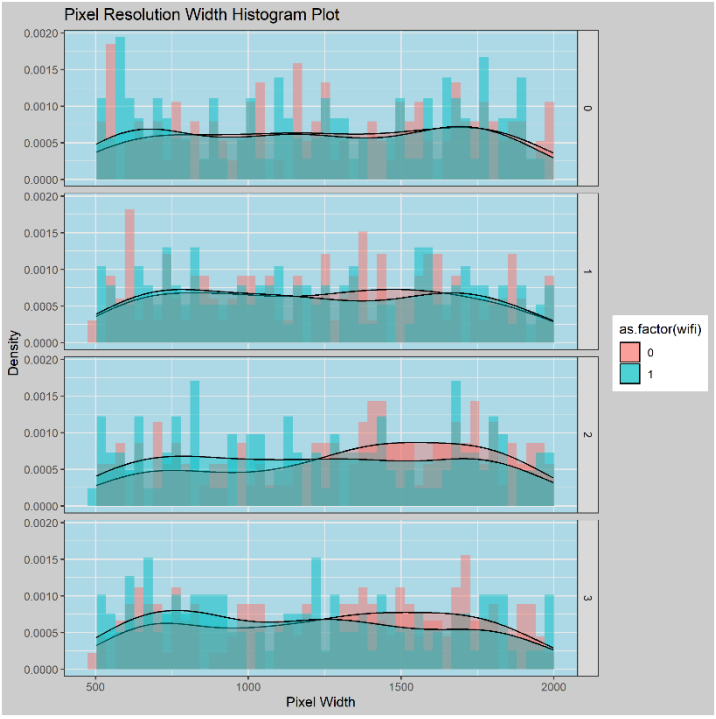
Line Plot:

Line plot is plotted using the selected columns m_dep (mobile depth), mobile_weight,
px_height (pixel resolution height), px_width (pixel resolution height), and id from the original data frame ‘a’. The selected columns are together called as Features .The Id is taken along the x axis and the corresponding values of each feature is taken along the y axis. Each Feature is being represented by different colours.

Histogram Plot:

In the plotted histogram the X-axis is mapped to the variable ‘px_height’ (pixel height) and the color of the bins represent the Bluetooth (whether the Bluetooth is present or not).The density of the ‘px_height’ is mapped to the y axis of the histogram.





Similarly, another histogram is plotted with pixel width in the x axis and its density in the y axis. The color of the bins specify whether the wi-fi is present or not. The density curve is plotted over the histogram and also is facetted based on the price range. The facetted histogram gives clear idea about the pixel width of different price range in addition to the wifi availability.

CONCLUSION AND FUTURE SCOPE

The different ways of data analysis and data visualization is done and we can conclude that graphs make the understanding easier and effective. Mobile phones are classified into different price ranges based on their characteristics and features and the graphs are plotted. These graphs have high accuracy and hence it gives clear and accurate information to the readers. The plotting of graph has helped in following the trends and pattern in mobile price classification.

Comparing various features becomes easier using the plots. The structure and summary of the dataset gives us the idea about the dataset we are dealing with so that it becomes easier for us to use the visualization concept.

Future Work :

R is a powerful programming language that is widely used in many different fields. As the language continues to evolve and new tools and technologies become available, it's possible to analyze and visualize data in various ways to get an effective result. Some of them are mentioned below.

- Ensemble methods: These methods combine multiple models to improve the overall performance, for example, Random Forest, Gradient Boosting, and Adaboost are ensemble methods that can be used for mobile price classification.

- Transfer learning: This technique uses a pre-trained model to extract features from the mobile phone data, which can then be used to train a new model for mobile price classification. This can be useful when there is limited data available for training.

- Multi-task learning: This approach trains a model to perform multiple tasks simultaneously, such as mobile price classification and image recognition. This can be useful when the mobile phone data includes images of the devices.

- Anomaly detection: This type of analysis is used to identify and handle outliers in the mobile phone data, which can be useful when the dataset contains noise or inconsistencies.

- Time series analysis: This type of analysis can be used to analyse the price trends of mobile phones over time, which can be useful for forecasting future prices.

REFERENCES

[1] Data set: https://www.kaggle.com/datasets/iabhishekofficial/mobile-price-classification

[2] https://www.geeksforgeeks.org

[3] https://towardsdatascience.com/how-to-create-animated-plots-in-r-adf53a775961

[4] http://www.sthda.com/english/wiki/ggplot2-histogram-plot-quick-start-guide-r-software-and-data-visualization#add-mean-line-and-density-plot-on-the-histogram