Predicting and Classifying the Mobile Prices using ML

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 Introduction

1.1 Overview

In the present day everywhere, smartphones are the most important part of human lives. The people are claiming and purchasing the different features of the mobiles based on the display, ram, networking (4G/5G), camera, and some other features. Hence the study focuses on utilising the features from the dataset downloaded from Kaggle and attempting machine learning algorithms such that it can find its applications in fields like mobile price prediction, and price prediction through data collected from review and apply sentimental analysis to find the similar features and apply the similar context. However, based on the features the mobile process varies from each other. There are a lot of existing methods that are developed by the various authors for predicting prices for houses and other objects but only a few contexts are based on predicting the mobile prices. Hence, here from the machine learning algorithms, in this work and comparison is displayed between the single classifier, ensemble techniques and Boosting algorithms. Finally, the comparison of the classification metrics is shown between the Logistic regression, Decision tree, random forest and XGboost algorithm. The comparisons are concluded by performing the enhancement techniques like the feature selection method and information gain method. Hence, mobile phone prices and classifying. This proposed work develops machine learning models which predict and classify mobile phones easily.

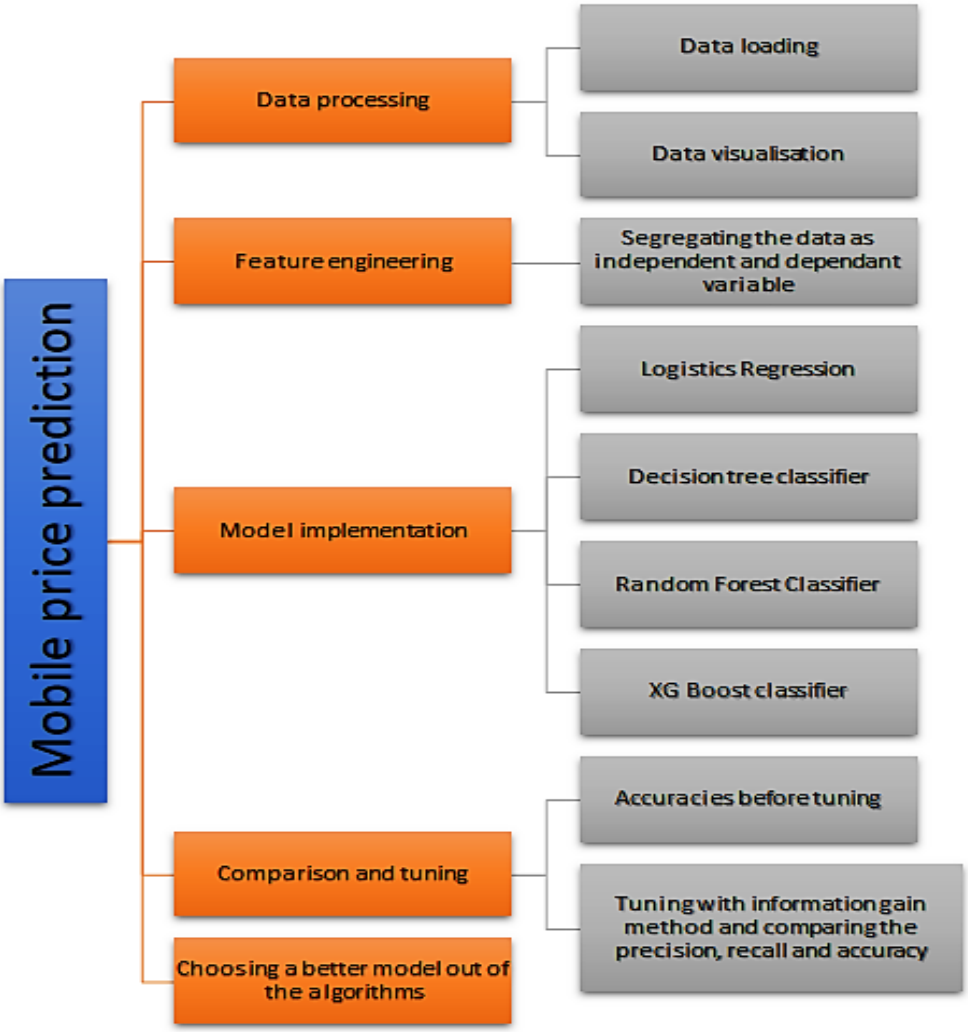
1.2 Aim and objectives

A. Aim

Predicting and Classifying the Mobile Prices using Machine Learning. The main goal of this project is to make a comparison among the classification algorithm like Logistic regression, Decision tree, random forest and XGboost algorithm to classify the prices of different mobiles based on their independent features. The study also aims to apply the information gain method of feature selection to improve the precision, recall and accuracy of the models.

B. Objectives

- To find the appropriate dataset related to mobile prices.to understand the features in the dataset.
- To understand the obtained dataset and perform the pre-processing techniques.
- To explore and research the different Machine Learning algorithms in the methodology chapter and make justifications based on the individual algorithms' advantages as well as disadvantages.


Mobile price prediction
Data processing
Data loading
Data visualisation
Feature engineering
Segregating the data as independent and dependant variable
Model implementation
Logistics Regression
Decision tree classifier
Random Forest Classifier
XG Boost classifier
Comparison and tuning
Accuracies before tuning
Tuning with information gain method and comparing the precision, recall and accuracy
Choosing a better model out of the algorithms

1.5 Organisation of the thesis

The overall process is divided into chapters and the chapters are divided into different sections and subsections. The overall chapters are simple and the **first chapter** is the introduction to the thesis which provides information like the aim and objectives of the study along with the purpose of the study and also about the methodology to be implemented in the study with a better-quality project flow diagram. Chapter **2** which is the literature review of the chapter provides the comparisons of the author's point of view on the different studies conducted online in the same context and the other contacts which can be related to the study. Finally, based on the conclusions from the chapter a better methodology is described in **chapter three** where different types of methods implemented here are discussed about their working and the author's examples are added in this chapter to better explain the working of these methods and algorithms used in the study. The chapter also consists of the justification the author has made on the implementation and selection of these machine learning algorithms which will provide a better view of the thesis. Finally, in **chapter 4** results are discussed along with the various features which are developed as an output from the code implemented and finally in **chapter 5** the conclusions are made along with the recommendations provided by the author to conclude that overall, the references are noted at the end in the Harvard style.

2 Literature Review

2.1 Overview

A lot of research is carried out on the proposed work for getting insights into the dataset and the different machine learning algorithms. Before directly working on this project analyse the model building and the results accuracy part that are obtained for each model. Now comparing the different authors' aim for undertaking this work and the methods that are implemented for obtaining the better results in predicting the prices of the mobile phones are detailly discussed below.

2.2 Recent studies in the work

The author Asim and Khan, (2018), has considered a real-time data set from a website by scraping it and applied feature selection methods as the present work has done but in the present study information gain type of feature selection method is used the results are compared based through the experiment that it is a reliable machine learning algorithm which can provide a

solution to predict the price of Smartphones when compared to that of the other models. The author has utilised three machine learning models which were support vector regression, backpropagation neural network as well as a Linear Regression model these results from the model were compared based on its efficiency however the data used by the author was based on the recordings from 2018 and compared to that of this present study the authors have also utilise 20% of data for testing. The inputs from the data used by the author have shown that they have used different types of brands of mobile and their pricing ranges however in this present study brands of the mobiles are not mentioned but the specifications of the mobile are used.

Table 2.1 Comparison and discussions on the related work considered here

Author	Aim	Methods	Result
Asim and Khan, 2018	To predict /classify the prices of mobile phones and real-time data set from a website by scraping it and applying feature selection methods	Decision Tree and Naïve Bayes with information gain type of feature selection method	Here in the study, the author has evaluated two of the classifiers which are the Decision tree and the Naïve Bayes classifier. The total accuracy of the models that were built, namely decision tree and Naive Bayes after feature selection, dimensionality, and improvement approaches, was 71.4 percent. In terms of the naive Bayes classifier, it was discovered to be 60.7 percent after the author used the information gain kind of feature selection approach. However, here Information gain technique is used and the highest recorded accuracy is 95%
Chandrashekhara *et al.* 2019			

Çetın, and Koç, 2021			

Hence it was decided that the information gain type of feature selection method will be applied and some of the features will be deducted and again the models will be implemented the models were also compared based on other evaluation metrics like Precision and recall such that the reliability of the machine learning model will be discussed and put forth. Hence, it was found that before and after applying the Information gain method the model accuracies, as well as precision and recall and improved and overall Logistic regression, have been recorded with the highest accuracy.

3 Methodology

In this chapter, the various machine learning models that can be implemented on the data has discussed. The workings of various models and their mathematical and probabilistic abilities of the models are analyzed and scrutinized. The subsections of the chapter are divided into sections which discuss the working of each algorithm based on the research articles, and later sections contain information regarding the justification of each algorithm.

3.1 Logistic Regression

According to **(**Patel *et al.* 2021; Cvitić *et al.* 2021)**,** the Logistic regression, shortened as LR, is considered a classification model and not a regression model. This algorithm is a type of supervised learning method which is powerful and is specially used only when the target variable is categorical. To predict the categorical independent variable, it utilizes a set of independent variables. The output is either of the forms like discrete value or the categorical value. Schober and Vetter (2021), state that the output may be of a form like Yes or No, 0 or 1, true or false, and so on. Instead of giving the exact values like 0 or 1, it gives probabilistic values which may range between 0 and 1.

The LR is almost similar to that of the linear regression and is usually based on how it is used. LR is used especially for solving the problems related to classification and linear regression is used for the purpose of solving problems that are related to the regression. However, (Abdulhafedh, 2022, and Shouval *et al.* 2021), explained that in this algorithm sigmoid function ("S" shaped) is utilized which is a logistic function that is usually fitted, and helps in determining the values like 0 and 1 rather than a regression line that is suitable. The possibility of something is given by the logistic function, which helps in obtaining the curve. It is an important ML algorithm since it can obtain the probabilities and to categorize the upcoming data into datasets like discrete or continuous. LR possesses some of the assumptions, that the dependent variable that is categorical in nature is important; and also, the independent variable should include multi-collinearity. Uddin *et al.* (2019), explained that in order to make use of this algorithm as a binary classifier, the threshold is necessary to split into separate classes. Consider an example in which the probability value is more than 0.50 for an input instance it classifies into two classes (independent) like class A and Class B. The most common version of logistic regression is known as multinomial LR. Consider the graph that identifies the probability of any disease that uses the LR method is shown in Fig 3.1.

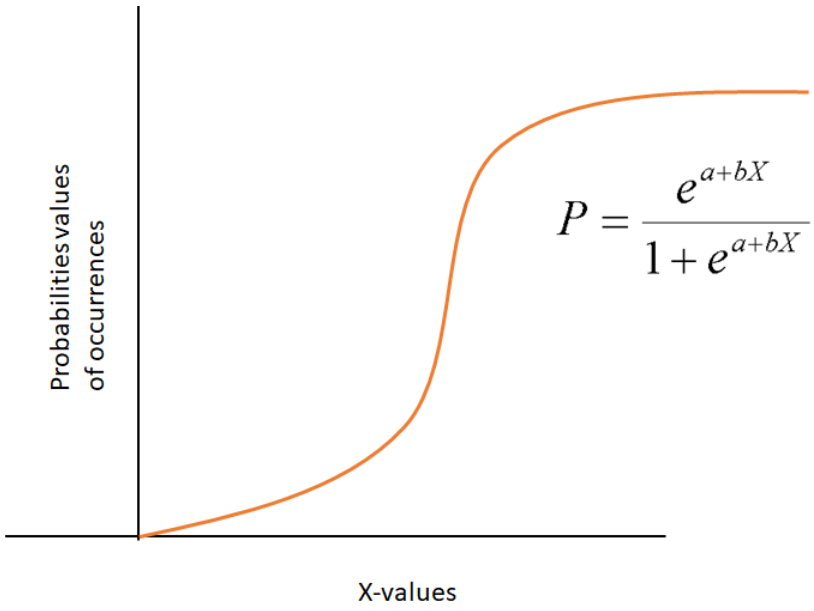


Figure 3.1Representation of Logistic regression

Source: Made by the author

A. Working of Decision Tree

The decision tree is the algorithm that comes under the supervised learning technique. It is suitable for both classification and regression types of problems; but it is mainly used for the classification problems (Charbuty and Abdulazeez, 2021; Rakhra *et al.* 2021). It is a classifier, which is in tree structure in which dataset features are provided by internal nodes and the decision rules are provided by the branches. Here each of the leaf nodes indicates the result. The two nodes in the decision tree are the decision node and the leaf node. The decision nodes are used for making decisions and consist of multiple branches; and the leaf nodes indicate the output of those decisions and further does not include the branches (Dridi, 2021). Usually, the decisions are done based on the features of the dataset provided. A decision tree is a form of graphical representation to get the solutions (possible) to a given problem based on the criteria provided. It is named a decision tree; since it looks like a tree, it begins with the root node, which helps in expanding the further branches and builds a tree. For building the trees, the classification and regression tree techniques are employed (McClarren, 2021). A working of it is based on: it asks a question and depending on the answer like Yes or No, it tries to divide the tree into the subtrees. It can contain both categorical data and numerical data. The working procedure of decision tree: To build a tree, start with the root node that includes in the total dataset. In the second step, identify the attribute that is best suitable in the dataset by using a measure called attribute selection. Then split the root node into the subsets that exhibit possible

values for the best attribute, and later the decision tree node is generated that includes the best attribute. At last, make new decision trees to work recursively by using dataset subsets and continue this until the stage is obtained where it finds difficult to further classify the nodes. The final node is called the leaf node. However, Mndawe *et al.* (2022) explained some of the advantages of a decision tree like it is very simple to understand, helpful for solving problems based on decisions, and it tries to get possible results for a problem given. The possible representation of the decision tree while solving the classification problem in the mobile price dataset is shown in Fig. 3.2.

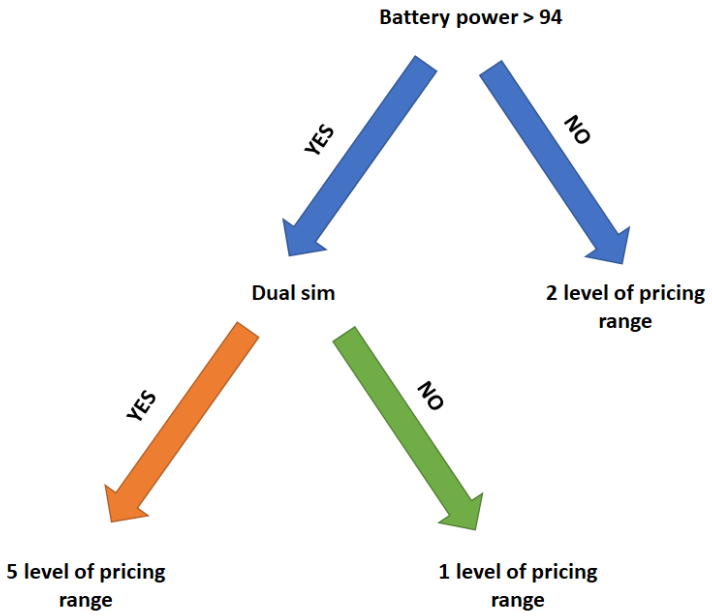


Figure 3.2 Interpretation of working of Decision tree

However, this method is quite popular to use as helps get more deep structured trees, the more advantages are seen in Section 3.2.

B. Working on XGBoost

categorical data variables. And also handles the missing data automatically. In this method also feature scaling is not necessary. The non-linear parameters are also handled much more efficiently also with missing values. RF automatically handles the robust values of outliers. This algorithm is more stable than that of other methods. It involves a longer period of training and also the complexity of the algorithm is high. However, based on the dataset used in our research, decision trees provide more accuracy when compared to XGB algorithm and other algorithms like RF and LR. After decision trees, XGB provides good accuracy compared to RF and LR. And LR provides the least accuracy compared to other algorithms used.

4 Results and Discussions

4.1 Overview

For the research, the data was obtained from the publicly available resource. After loading the data, the data frame was checked for null values and after ensuring the data set does not contain null values, the author went ahead with the data visualization steps. After understanding the data and further step to decide which machine learning algorithms are capable of handling the data is made. Upon deciding, the author performed Logistic regression, Decision Trees, XGBoost, and Random Forests were performed on the data.

According to Garcia *et al.* (2015), for any machine learning task it is essential to include steps such as data gathering and preparation, and pre-processing and transformation of the data. From this, the author of this work opines that including the essential steps in the machine learning task allow the data scientist to discover insights about the data and help attain statistical understanding of the data and help tune the machine learning algorithm such that the algorithm can provide insights about the data.

4.2 Dataset Description

The dataset information can be seen in Fig. 4.5, There are 2000 rows in the data set and 21 columns in the data set which also includes a column which was excluded, the ID of the mobile phone. Hence, the ID of the mobile phone is said to be the unique identifier for the mobile phone described in the data set. The other features are the battery power, Teacher describes the

```
[ ]  #Required Libraries
     import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import seaborn as sns
     from sklearn.model_selection import train_test_split
     from sklearn.preprocessing import StandardScaler
     from sklearn.linear_model import LogisticRegression
     from sklearn.tree import DecisionTreeClassifier
     from sklearn.metrics import accuracy_score,classification_report,plot_confusion_matrix
     from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
```

```
[ ] train_path="/content/drive/MyDrive/Mobile price prediction/archive (20).zip (Unzipped Files)/train.csv"
    train_data=pd.read_csv(train_path)
    train_data
```

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	...	px_height	px_width	ram	sc_h	sc_w	talk_time	three_g	touch_screen	wifi	price_range
0	842	0	2.2	0	1	0	7	0.6	188	2	...	20	756	2549	9	7	19	0	0	1	1
1	1021	1	0.5	1	0	1	53	0.7	136	3	...	905	1988	2631	17	3	7	1	1	0	2
2	563	1	0.5	1	2	1	41	0.9	145	5	...	1263	1716	2603	11	2	9	1	1	0	2
3	615	1	2.5	0	0	0	10	0.8	131	6	...	1216	1786	2769	16	8	11	1	0	0	2
4	1821	1	1.2	0	13	1	44	0.6	141	2	...	1208	1212	1411	8	2	15	1	1	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1995	794	1	0.5	1	0	1	2	0.8	106	6	...	1222	1890	668	13	4	19	1	1	0	0
1996	1965	1	2.6	1	0	0	39	0.2	187	4	...	915	1965	2032	11	10	16	1	1	1	2
1997	1911	0	0.9	1	1	1	36	0.7	108	8	...	868	1632	3057	9	1	5	1	1	0	3
1998	1512	0	0.9	0	4	1	46	0.1	145	5	...	336	670	869	18	10	19	1	1	1	0
1999	510	1	2.0	1	5	1	45	0.9	168	6	...	483	754	3919	19	4	2	1	1	1	3

```
test_path="/content/drive/MyDrive/Mobile price prediction/archive (20).zip (Unzipped Files)/test.csv"
test_data=pd.read_csv(test_path)
test_data
```

	id	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	...	pc	px_height	px_width	ram	sc_h	sc_w	talk_time	three_g	touch_screen	wifi
0	1	1043	1	1.8	1	14	0	5	0.1	193	...	16	226	1412	3476	12	7	2	0	1	0
1	2	841	1	0.5	1	4	1	61	0.8	191	...	12	746	857	3895	6	0	7	1	0	0
2	3	1807	1	2.8	0	1	0	27	0.9	186	...	4	1270	1366	2396	17	10	10	0	1	1
3	4	1546	0	0.5	1	18	1	25	0.5	96	...	20	295	1752	3893	10	0	7	1	1	0
4	5	1434	0	1.4	0	11	1	49	0.5	108	...	18	749	810	1773	15	8	7	1	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
995	996	1700	1	1.9	0	0	1	54	0.5	170	...	17	644	913	2121	14	8	15	1	1	0
996	997	609	0	1.8	1	0	0	13	0.9	186	...	2	1152	1632	1933	8	1	19	0	1	1
997	998	1185	0	1.4	0	1	1	8	0.5	80	...	12	477	825	1223	5	0	14	1	0	0
998	999	1533	1	0.5	1	0	0	50	0.4	171	...	12	38	832	2509	15	11	6	0	1	0
999	1000	1270	1	0.5	0	4	1	35	0.1	140	...	19	457	608	2828	9	2	3	1	0	1

```
test=test_data.drop(['id'],axis=1)
test.head()
```

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	pc	px_height	px_width	ram	sc_h	sc_w	talk_time	three_g	touch_screen	wifi
0	1043	1	1.8	1	14	0	5	0.1	193	3	16	226	1412	3476	12	7	2	0	1	0
1	841	1	0.5	1	4	1	61	0.8	191	5	12	746	857	3895	6	0	7	1	0	0
2	1807	1	2.8	0	1	0	27	0.9	186	3	4	1270	1366	2396	17	10	10	0	1	1
3	1546	0	0.5	1	18	1	25	0.5	96	8	20	295	1752	3893	10	0	7	1	1	0
4	1434	0	1.4	0	11	1	49	0.5	108	6	18	749	810	1773	15	8	7	1	0	1

```
[ ]
Train dataset (2000, 21)
Test dataset (1000, 20)
```

```
train_data.info()
```

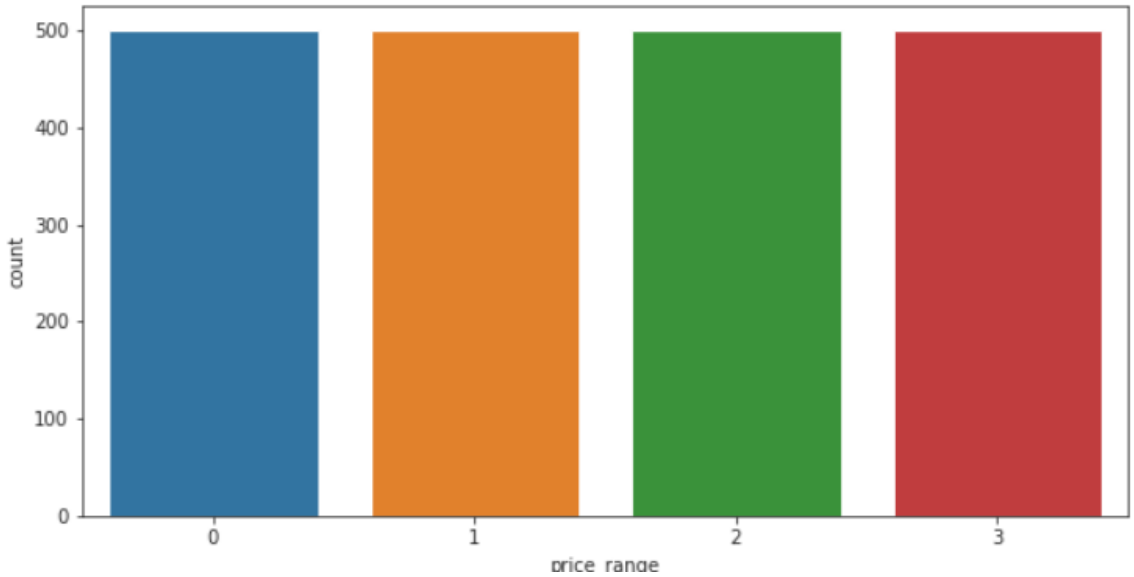
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   battery_power  2000 non-null   int64
 1   blue           2000 non-null   int64
 2   clock_speed    2000 non-null   float64
 3   dual_sim       2000 non-null   int64
 4   fc             2000 non-null   int64
 5   four_g         2000 non-null   int64
 6   int_memory     2000 non-null   int64
 7   m_dep          2000 non-null   float64
 8   mobile_wt      2000 non-null   int64
 9   n_cores        2000 non-null   int64
 10  pc             2000 non-null   int64
 11  px_height      2000 non-null   int64
 12  px_width       2000 non-null   int64
 13  ram            2000 non-null   int64
 14  sc_h           2000 non-null   int64
 15  sc_w           2000 non-null   int64
 16  talk_time      2000 non-null   int64
 17  three_g        2000 non-null   int64
 18  touch_screen   2000 non-null   int64
 19  wifi           2000 non-null   int64
 20  price_range    2000 non-null   int64
dtypes: float64(2), int64(19)
memory usage: 328.2 KB
```

```
train_data.isnull().sum()
```

```
battery_power    0
blue             0
clock_speed      0
dual_sim         0
fc               0
four_g           0
int_memory       0
m_dep            0
mobile_wt        0
n_cores          0
pc               0
px_height        0
px_width         0
ram              0
sc_h             0
sc_w             0
talk_time        0
three_g          0
touch_screen     0
wifi             0
price_range      0
dtype: int64
```

```
plt.figure(figsize=(10,5))
sns.set_style('ticks')
sns.countplot(x='price_range',data=train_data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f46ef75d0d0>
```



values in one column tend to decrease with an increase in the other column. This correlation gives information about the relationship between the variables. The heatmap is obtained using the Seaborn library, it is a plot of the rectangular colour-encoded matrix as seen in Fig. 4.8. This is one of the easiest ways to visualize the correlation within the data. From the correlation heatmap, it can be said that most of the columns are not strongly correlated with each other. Therefore, the author opines that there is no need to drop any column because of correlation.

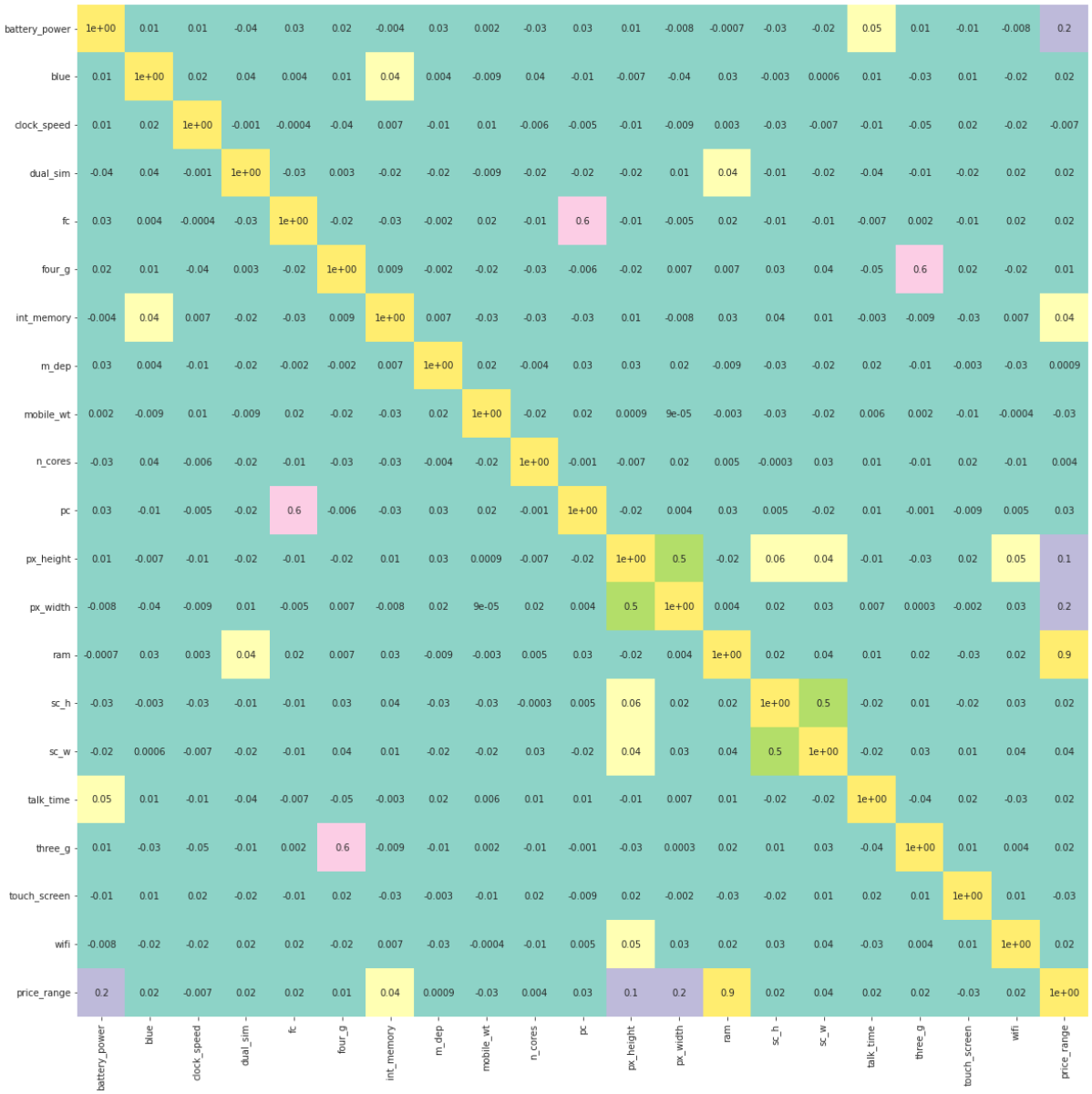


Figure 4.8 Heatmap of the correlation in the train data

The scatterplot is used to determine if the two columns of the data have a relationship. The scatterplot shown in Fig. 4.9 was obtained by using the scatterplot in the seaborn library. The scatterplot thus obtained contains values of the price range column in comparison with the "FC" column of the data frame. This shows the numbers in comparison.

```
sns.scatterplot(x="price_range", y="fc", data=train_data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f46ec609bd0>
```

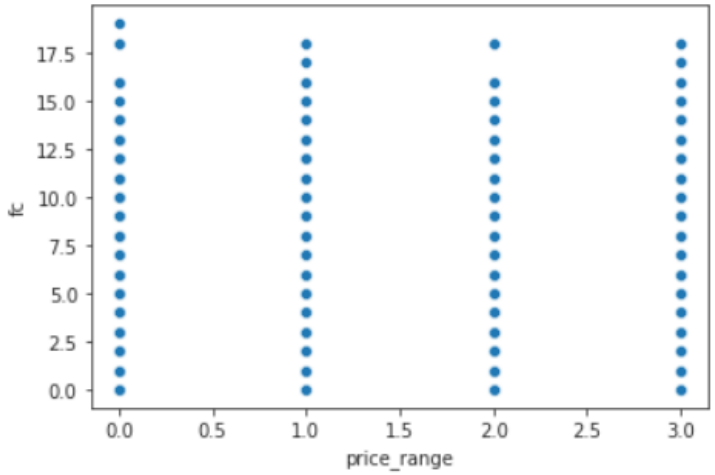


Figure 4.9 Scatterplot between the price range and FC columns in the dataset

The value counts of each column are used to depict the number of each specific value in the column of the data frame. The value counts are obtained using the "data frame.value_counts()". The value counts of the dataset are shown in the Fig. 4.10.

```
train_data.value_counts()
```

battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	pc	px_height	px_width	ram	sc_h	sc_w	talk_time	three_g	touch_screen	wifi	price_range	
501	0	2.3	0	12	1	54	0.3	131	4	19	504	1089	2346	13	12	2	1	0	1	1	1
1494	0	1.9	0	2	1	47	0.4	129	2	9	207	1177	3202	13	1	14	1	0	0	3	1
1504	0	1.9	1	7	1	25	0.3	152	4	9	395	1853	1788	13	11	13	1	0	1	1	1
		1.5	0	12	1	34	0.7	142	8	15	507	973	2806	7	2	7	1	0	1	2	1
1503	1	0.7	0	10	1	43	0.2	166	4	14	841	1304	2122	15	1	2	1	0	0	2	1
																					..
972	1	1.7	0	0	1	54	0.7	107	3	4	681	1050	1900	16	8	19	1	0	1	1	1
		1.3	0	8	0	44	0.7	82	7	10	297	589	3925	7	6	7	0	1	0	3	1
		0.5	1	5	0	13	0.4	193	5	8	1392	1795	1615	11	6	13	0	1	0	1	1
970	1	1.3	1	5	0	59	0.2	84	7	19	1765	1786	3675	12	0	12	0	0	0	3	1
1998	1	1.2	0	8	0	31	0.6	196	8	19	50	1356	2311	17	9	18	1	1	0	2	1

```
Length: 2000, dtype: int64
```

Figure 4.10 Checking value counts in the dataset

Using the below block of code as shown in the Fig. 4.11 the count plot of five different columns is obtained using the seaborn library. Column such as "blue", "dual_sim", "four_g", "three_g", "touch_screen", "wifi", and "int_memory" was used and the count plot was obtained.

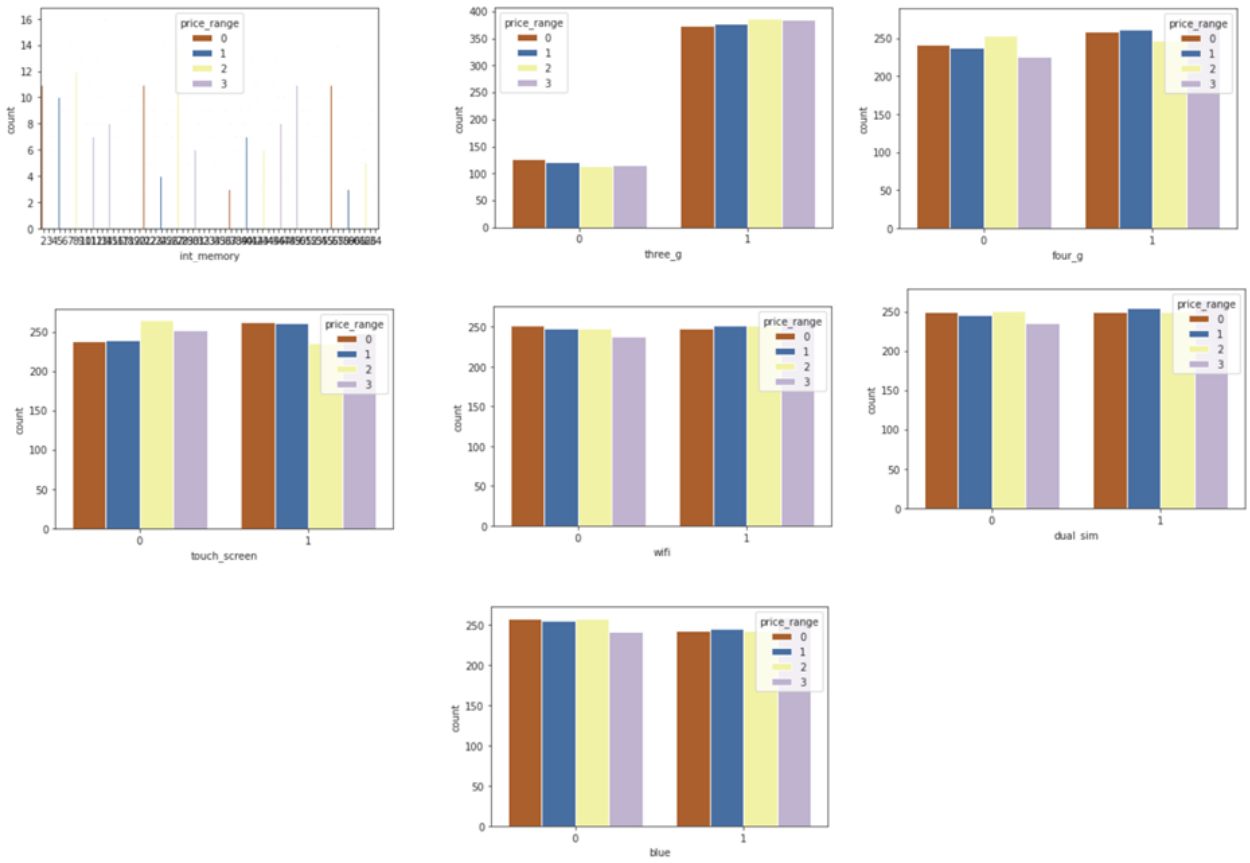


Figure 4.11 Checking the total number of Records and the list of labels in the dataset

The distplot is used to depict the variation in the data. The distribution plot is used to show the overall distribution of the continuous data variables. In Fig. 4.12, the distplot of the price range of the data is obtained. It is observed that labels 1 and 2 are normally distributed and the labels 0 and 3 are skewed toward the left and right respectively.

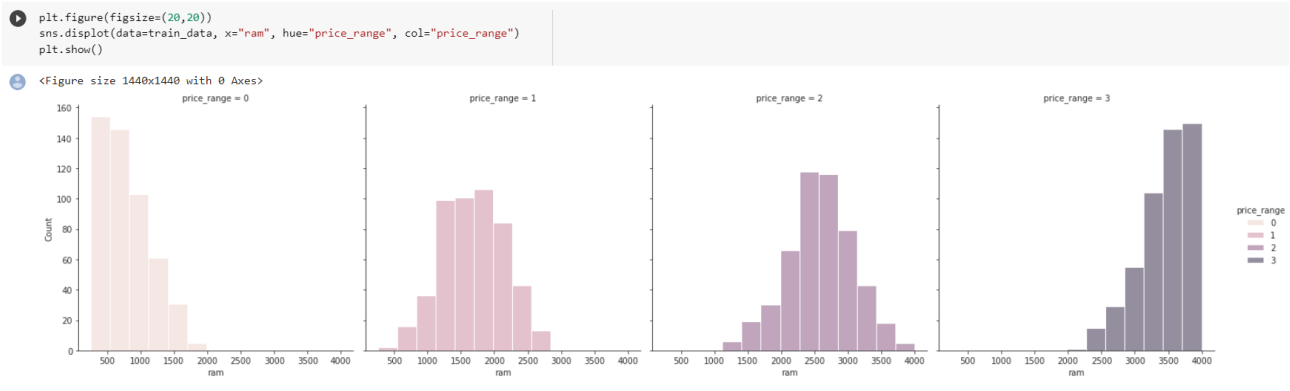


Figure 4.12.Distplot of the classes in the target column in the dataset

The boxplot is used to display the distribution of the data based on the summary of the minimum, maximum, first quartile, second quartile, and third quartile. This is used to identify the outliers in the data. The boxplot of the data is shown in Fig. 4.13.

```
#checking the outliers present in the dataset
train_data.plot(kind='box',figsize=(20,10))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f46ef7b0210>
```

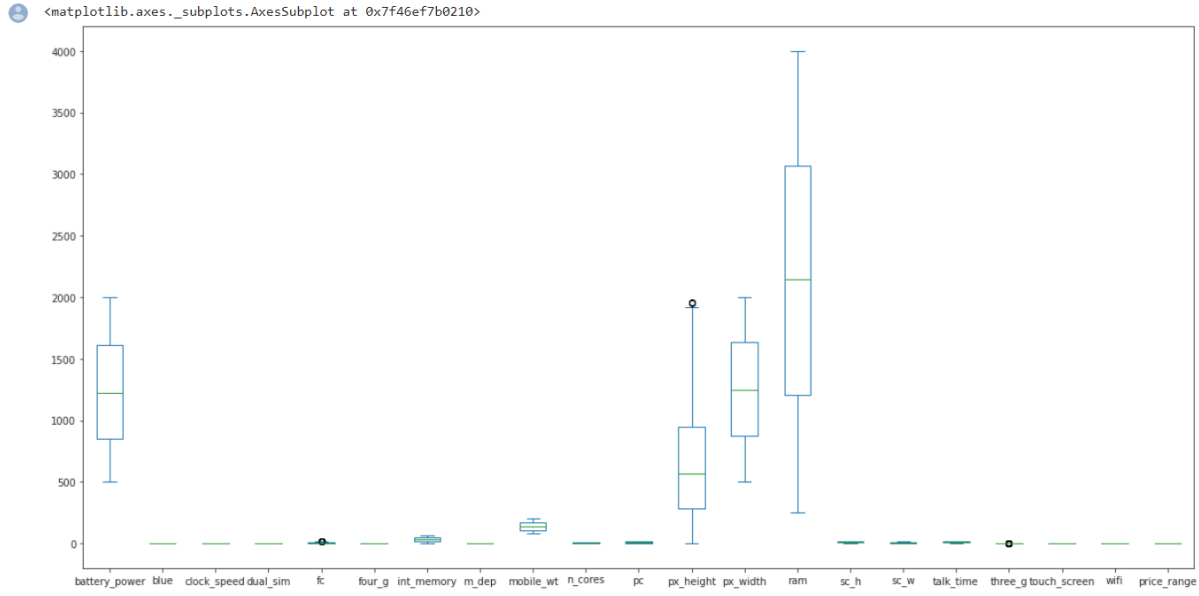


Figure 4.13 Boxplot of the columns of the dataset

4.6 Feature Engineering

The train data frame contains 21 columns and 2000 rows and the test data frame contains 20 columns and 1000 rows are combined using the PD. concat ( ), and the resultant data frame is stored in the data frame "df". The process of concatenation and the resultant data frame is shown in Fig. 4.14.

```
df=pd.concat([train_data,test])
df.head()
```

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	...	px_height	px_width	ram	sc_h	sc_w	talk_time	three_g	touch_screen	wifi	price_range
0	842	0	2.2	0	1	0	7	0.6	188	2	...	20	756	2549	9	7	19	0	0	1	1.0
1	1021	1	0.5	1	0	1	53	0.7	136	3	...	905	1988	2631	17	3	7	1	1	0	2.0
2	563	1	0.5	1	2	1	41	0.9	145	5	...	1263	1716	2603	11	2	9	1	1	0	2.0
3	615	1	2.5	0	0	0	10	0.8	131	6	...	1216	1786	2769	16	8	11	1	0	0	2.0
4	1821	1	1.2	0	13	1	44	0.6	141	2	...	1208	1212	1411	8	2	15	1	1	0	1.0

Figure 4.14 Concatenating the training and testing datasets

```
x=train_data.drop('price_range',axis=1)
x.head(5)
```

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	pc	px_height	px_width	ram	sc_h	sc_w	talk_time	three_g	touch_screen	wifi
0	842	0	2.2	0	1	0	7	0.6	188	2	2	20	756	2549	9	7	19	0	0	1
1	1021	1	0.5	1	0	1	53	0.7	136	3	6	905	1988	2631	17	3	7	1	1	0
2	563	1	0.5	1	2	1	41	0.9	145	5	6	1263	1716	2603	11	2	9	1	1	0
3	615	1	2.5	0	0	0	10	0.8	131	6	9	1216	1786	2769	16	8	11	1	0	0
4	1821	1	1.2	0	13	1	44	0.6	141	2	14	1208	1212	1411	8	2	15	1	1	0

```
y=train_data['price_range']
y.head(5)
```

```
0    1
1    2
2    2
3    2
4    1
Name: price_range, dtype: int64
```

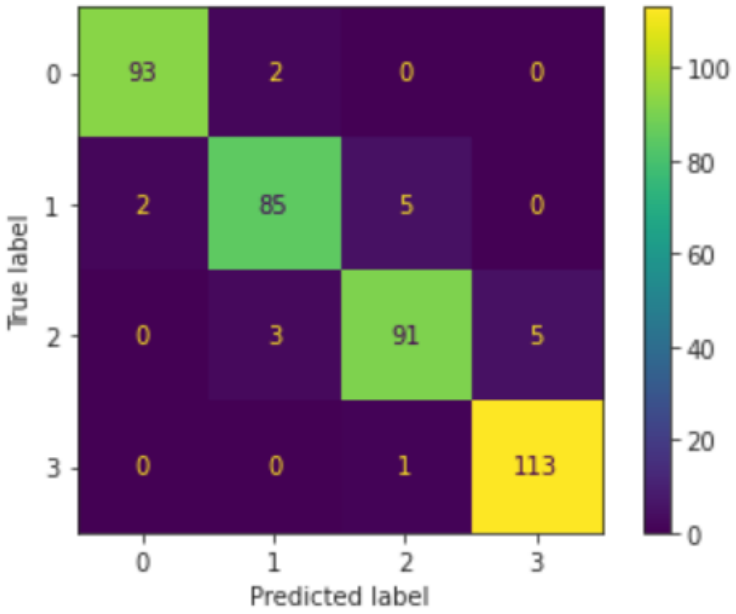
```
[ ]  print(classification_report(log_pred,y_test))
```

```
              precision    recall  f1-score   support

           0       0.98      0.98      0.98        95
           1       0.92      0.94      0.93        90
           2       0.92      0.94      0.93        97
           3       0.99      0.96      0.97       118

    accuracy                           0.95       400
   macro avg       0.95      0.95      0.95       400
weighted avg       0.96      0.95      0.96       400
```

```
cm_log=confusion_matrix(log_pred,y_test)
print(cm_log)
plot_confusion_matrix(log,x_test,y_test)
plt.show()
```

```
[[ 93   2   0   0]
 [  2  85   3   0]
 [  0   5  91   1]
 [  0   0   5 113]]
```

/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: Function plot_confusion_matrix is deprecated; Function `plot_confusion_matrix` is deprecated in 1.0 and will be removed in 1.2. Use one of the class methods: ConfusionMatrixDisplay.from_predictions or ConfusionMatrixDisplay.from_estimator.
warnings.warn(msg, category=FutureWarning)



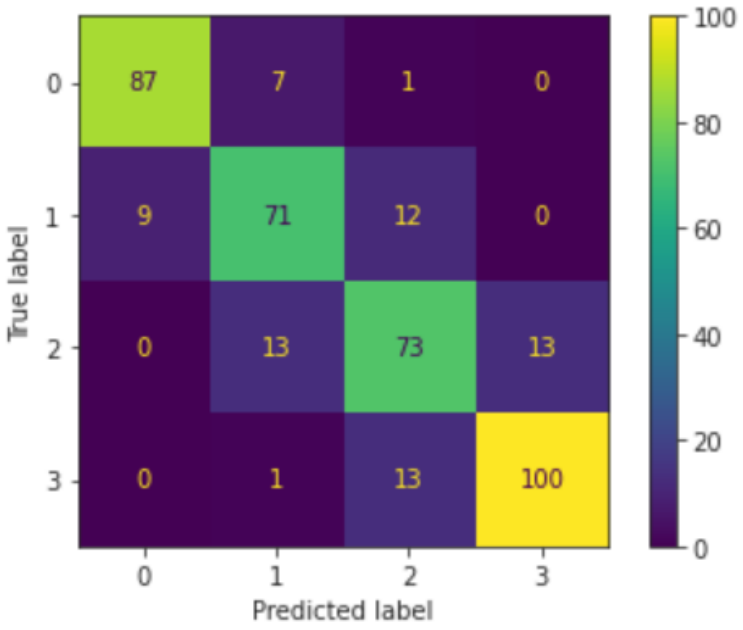
```
[ ]  print(classification_report(Dtc_pred,y_test))
```

```
                  precision    recall  f1-score   support

               0       0.88      0.90      0.89        93
               1       0.73      0.74      0.74        90
               2       0.76      0.72      0.74       104
               3       0.88      0.88      0.88       113

        accuracy                           0.81       400
       macro avg       0.81      0.81      0.81       400
    weighted avg       0.81      0.81      0.81       400
```

```
cm_Dtc=confusion_matrix(Dtc_pred,y_test)
print(cm_Dtc)
plot_confusion_matrix(Dtc,x_test,y_test)
plt.show()
```

```
[[ 87   9   0   0]
 [  7  71  13   1]
 [  1  12  73  13]
 [  0   0  13 100]]
```

/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: Function plot_confusion_matrix is deprecated; Function `plot_confusion_matrix` is deprecated in 1.0 and will be removed in 1.2. Use one of the class methods: ConfusionMatrixDisplay.from_predictions or ConfusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)


87
7
1
0
9
71
12
0
0
13
73
13
0
1
13
100
0
1
2
3
True label
Predicted label
100
80
60
40
20
0

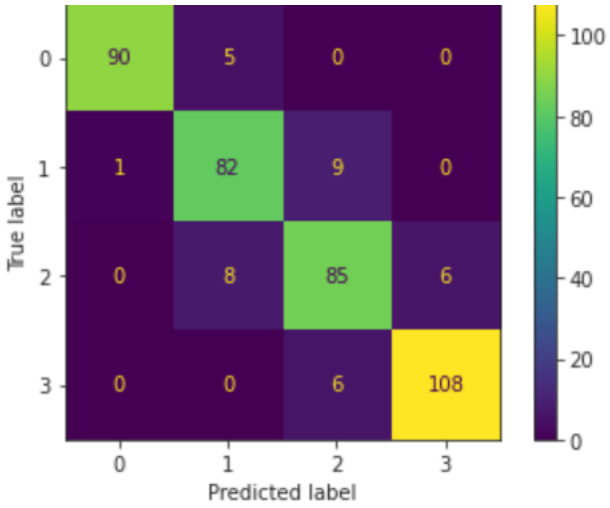
```
print(classification_report(xgb_pred,y_test))
```

```
              precision    recall  f1-score   support

           0       0.95      0.99      0.97        91
           1       0.89      0.86      0.88        95
           2       0.86      0.85      0.85       100
           3       0.95      0.95      0.95       114

    accuracy                           0.91       400
   macro avg       0.91      0.91      0.91       400
weighted avg       0.91      0.91      0.91       400
```

```
[ ] cm_xgb=confusion_matrix(xgb_pred,y_test)
    print(cm_xgb)
    plot_confusion_matrix(xgb,x_test,y_test)
    plt.show()
```

```
[[ 90   1   0   0]
 [  5  82   8   0]
 [  0   9  85   6]
 [  0   0   6 108]]
```

/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: Function plot_confusion_matrix is deprecated; Function `plot_confusion_matrix` is deprecated in 1.0 and will be removed in 1.2. Use one of the class methods: ConfusionMatrixDisplay.from_predictions or ConfusionMatrixDisplay.from_estimator.
warnings.warn(msg, category=FutureWarning)



```
RFC = RandomForestClassifier(n_estimators=30)
RFC.fit(x_train, y_train)
print("training accuracy of RF",RFC.score(x_train,y_train)*100)
print("testing accuracy of RF",RFC.score(x_test,y_test)*100)
RFC_pred = RFC.predict(x_test)
RFC_pred
```

```
training accuracy of RF 100.0
testing accuracy of RF 86.5
array([3, 0, 2, 1, 3, 0, 0, 2, 3, 1, 0, 3, 0, 2, 3, 0, 3, 2, 2, 1, 0, 0,
       3, 1, 1, 2, 3, 1, 3, 1, 1, 0, 2, 0, 2, 3, 0, 0, 3, 3, 2, 1, 3, 3,
       1, 3, 0, 1, 3, 1, 1, 3, 0, 3, 0, 2, 2, 1, 0, 3, 3, 1, 3, 2, 1, 2,
       3, 3, 2, 2, 3, 2, 1, 0, 1, 3, 2, 1, 2, 2, 3, 3, 3, 0, 0, 0, 2, 0,
       1, 3, 1, 3, 2, 1, 0, 2, 3, 3, 0, 3, 1, 1, 3, 1, 3, 2, 2, 3, 2, 3,
       3, 0, 0, 1, 3, 3, 0, 0, 1, 0, 1, 3, 2, 2, 1, 2, 1, 1, 0, 2, 1, 3,
       3, 3, 3, 3, 3, 2, 0, 1, 1, 2, 2, 3, 0, 3, 0, 0, 2, 0, 1, 1, 1, 1,
       3, 0, 0, 3, 1, 3, 2, 1, 3, 1, 3, 3, 3, 2, 1, 0, 3, 2, 2, 3, 3, 0,
       1, 2, 3, 0, 2, 1, 0, 2, 3, 1, 2, 0, 2, 3, 1, 1, 0, 2, 3, 0, 1, 3,
       2, 0, 3, 3, 2, 1, 2, 3, 3, 3, 0, 0, 0, 2, 3, 3, 0, 0, 1, 3, 2, 3,
       3, 3, 0, 0, 2, 2, 3, 1, 0, 2, 0, 0, 0, 3, 3, 0, 2, 1, 0, 1, 0, 2,
       3, 3, 0, 0, 1, 3, 3, 1, 3, 0, 3, 1, 1, 0, 1, 3, 2, 2, 0, 0, 1, 2,
       3, 2, 2, 3, 1, 1, 0, 3, 3, 2, 1, 3, 3, 2, 2, 1, 0, 2, 2, 1, 0, 0,
       2, 2, 2, 2, 0, 1, 3, 0, 1, 2, 3, 0, 2, 0, 1, 1, 3, 0, 0, 2, 3, 1,
       1, 0, 2, 0, 3, 0, 3, 3, 2, 3, 1, 2, 2, 1, 1, 1, 0, 1, 0, 3, 1, 0,
       3, 1, 0, 1, 2, 0, 3, 2, 2, 0, 1, 3, 0, 2, 1, 1, 2, 1, 1, 0, 2, 0,
       0, 3, 1, 2, 3, 2, 3, 0, 3, 2, 2, 1, 2, 2, 3, 3, 3, 0, 2, 0, 3, 0,
       1, 1, 2, 3, 1, 2, 1, 1, 0, 1, 2, 3, 0, 1, 1, 3, 0, 3, 0, 1, 2, 1,
       1, 0, 3, 0])
```

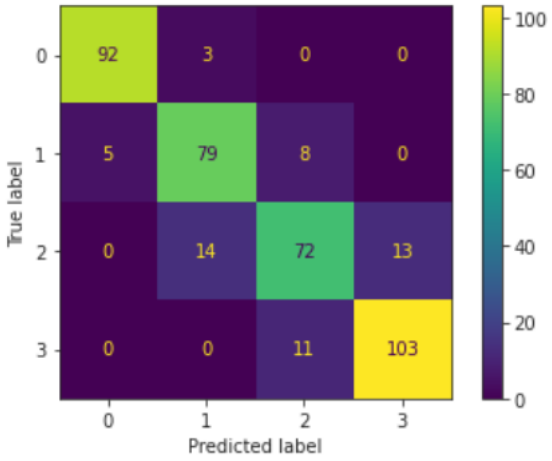
```
print(classification_report(RFC_pred,y_test))
```

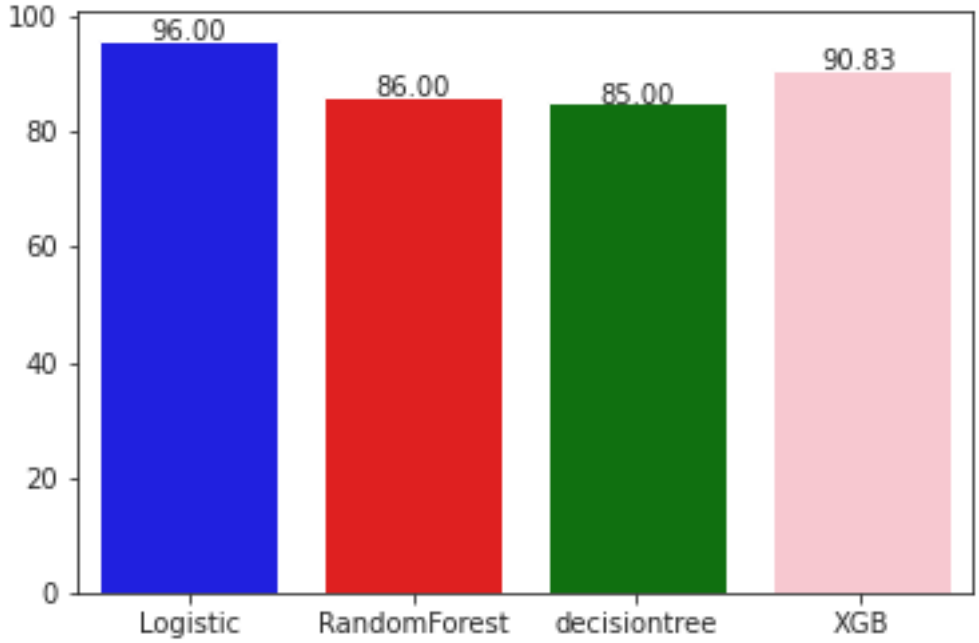
```
              precision    recall  f1-score   support

           0       0.98      0.95      0.96        98
           1       0.85      0.80      0.82        98
           2       0.70      0.80      0.75        86
           3       0.93      0.90      0.91       118

    accuracy                           0.86       400
   macro avg       0.86      0.86      0.86       400
weighted avg       0.87      0.86      0.87       400
```

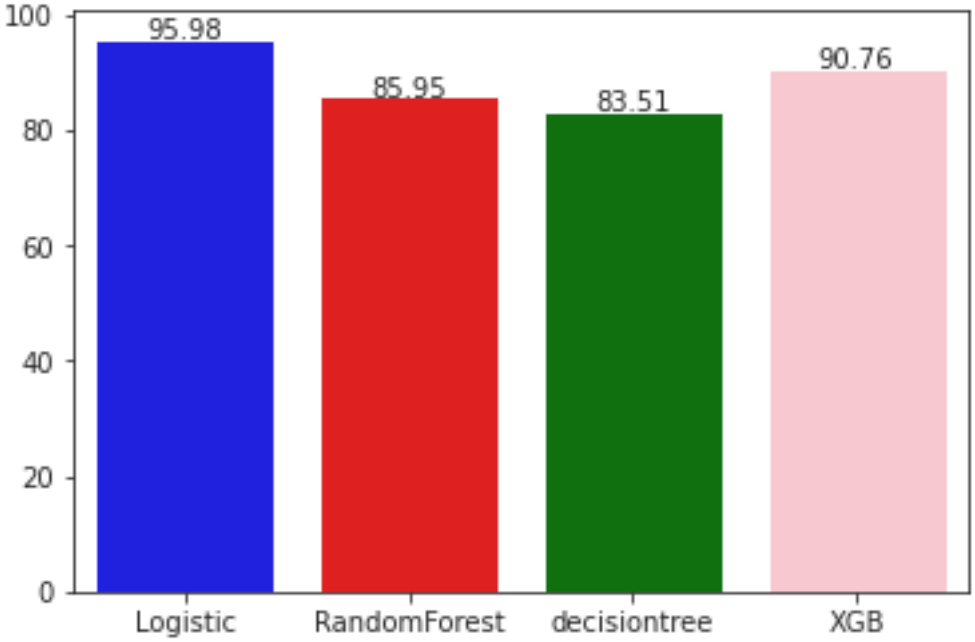
```
cm_RFC=confusion_matrix(RFC_pred,y_test)
print(cm_RFC)
plot_confusion_matrix(RFC,x_test,y_test)
plt.show()
```

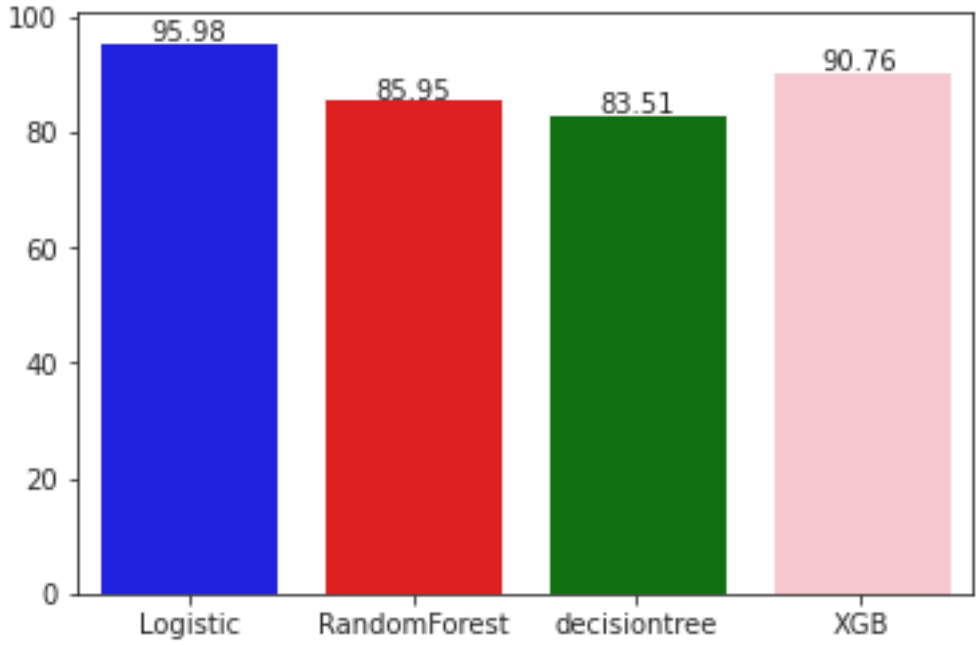
```
[[ 92   5   0   0]
 [  3  79  14   0]
 [  0   8  72  11]
 [  0   0  13 103]]
```

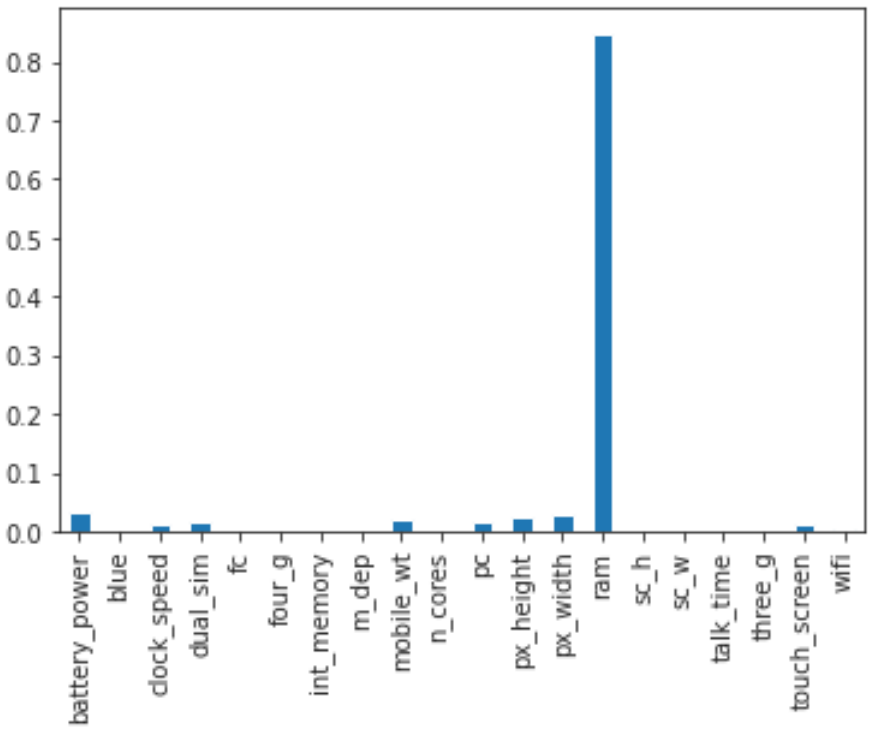
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: Function plot_confusion_matrix is deprecated; Function `plot_confusion_matrix` is deprecated in 1.0 and will be removed in 1.2. Use one of the class methods: ConfusionMatrixDisplay.from_predictions or ConfusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)




100
80
60
40
20
0
96.00
86.00
85.00
90.83
Logistic
RandomForest
decisiontree
XGB


100
80
60
40
20
0
95.98
85.95
83.51
90.76
Logistic
RandomForest
decisiontree
XGB


100
80
60
40
20
0
95.98
85.95
83.51
90.76
Logistic
RandomForest
decisiontree
XGB



Overall, the idea behind utilising the feature importance factor in the present study was that in the starting it was observed in the data set that there were no in balance issues but however when the random forest model and extra boost model was implemented both Precision and recall values started acting inefficient and provided a zero value which tells us that the model was not performing well and not predicting the positive as well as the negative samples of the data set. however overfitting issues were also seen in the models random forest and XGBoost be used to enhance this feature selection methods were adapted however feature importance technique was used here as overfitting was occurring and the number of features can be reduced using this course provided by feature importance and the number of samples will also be decreased such that the model concentrates on the tests is provided and provide us with better predictions of positive as well as negative labels. Hence, from the Fig. 4.31, it can be seen that Ram of the mobile contributes more towards the price of the mobile rather than WIFI specifications, Bluetooth, 4G specifications, microprocessor cores, screen height and width as well as the 3G specification. Hence, these features were reduced in the independent variables and models were implemented. Through the implementations there was a difference seen in accuracy, weighted precision as well as recall of the models.

5 Conclusions and discussions

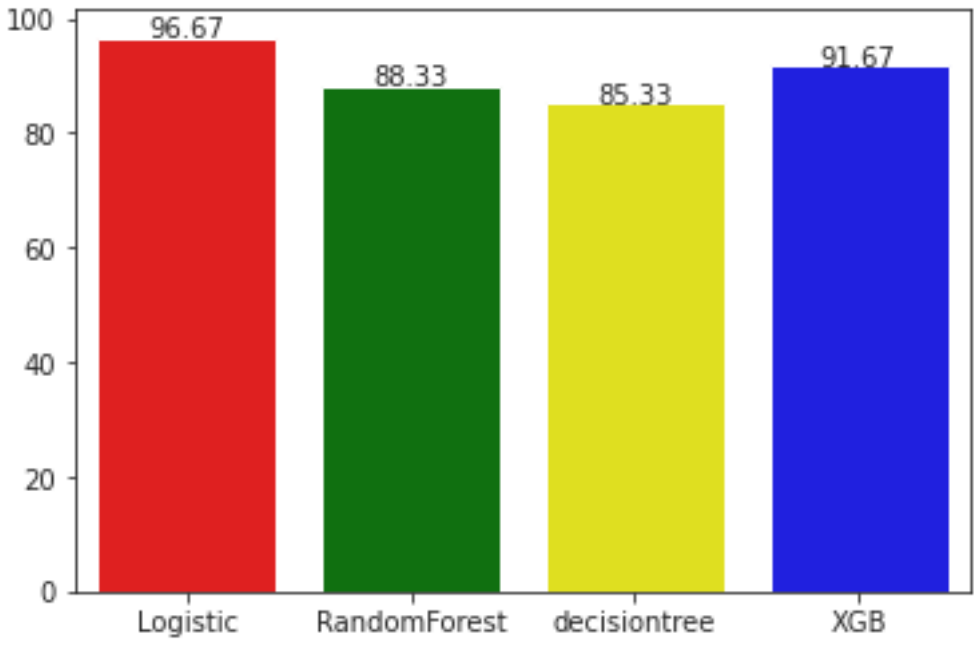
5.1 Conclusions

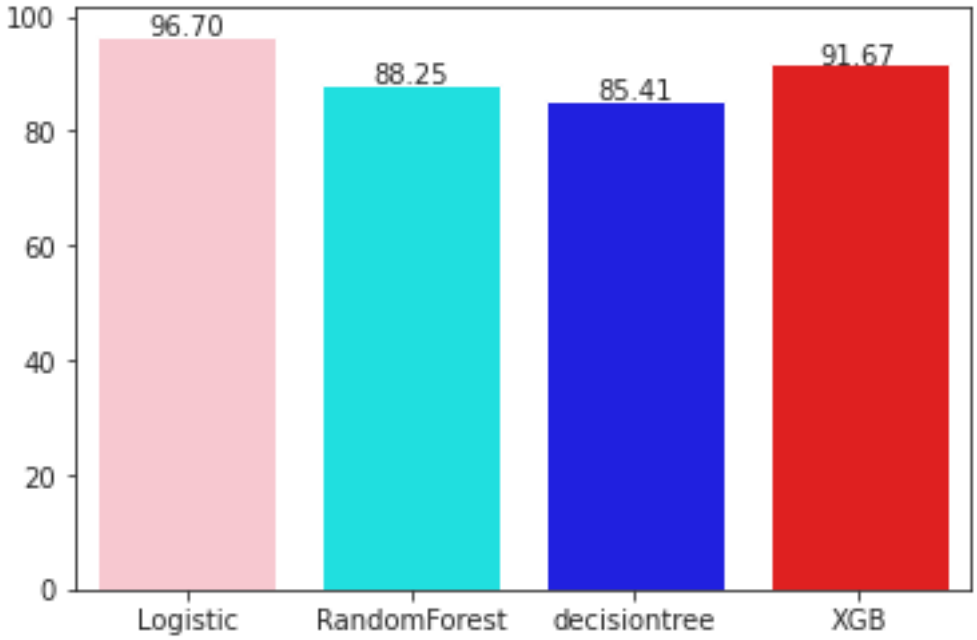
This work is intended to deliver the realizations of the work carried out in the current field of data science and machine learning. The intended objective of the work is discussed, and the clear intent of the work is organised in Chapter 1 as well. The machine learning algorithms such as the Decision Trees, Random Forest, XGB classifier and Logistic regression. The necessary evaluation of the model is discussed in Chapter 4 of the document. In this proposed work initially, the dataset is collected from Kaggle and trying to pre-process the data and extract the image features of the data. All these data pre-processing is done by using the python code. Here Goggle Colab notebook is used for writing the code and implementing the ML model using Python. Then developing a Logistic Regression, Decision tree, SVM, and Random Forest models which predicts and classifies the mobile prices. Finally compare the accuracies, loss, confusion matrix, and classification report for all the three models. The possible recommendations are given here in this section.

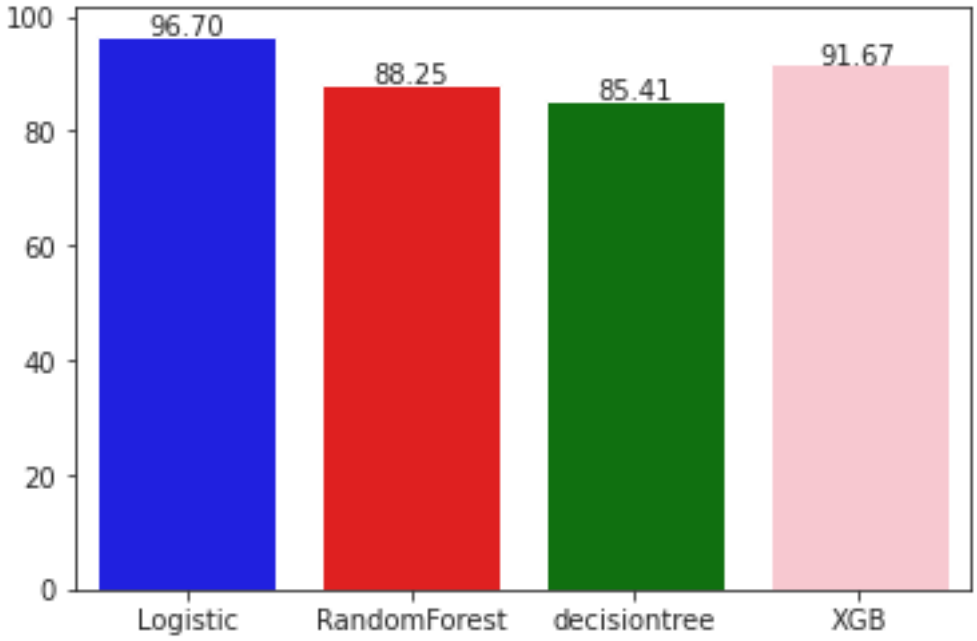
- The various factors affecting the mobile price is discussed to understand the basics of the business model of the mobile phone manufacturers. The aim is to initiate the necessary discussion about the price identification of the mobile phones and the factors affecting the final decision. Hence, it is important to understand the factors which influence the price of the mobile phone.
- The dataset contains four different classes and to classify them mathematically, machine learning algorithms such the Logistic regression, Decision trees, Random Forest, and XGB classifier were implemented, and the necessary evaluations were made.
- The mobile procedure differs depending on the characteristics. There are several current methods created by various authors for forecasting the values of houses and other items, but only a few situations are based on predicting the costs of mobile phones. As a result, from the machine learning algorithms, a comparison is shown in this study between the single classifier, ensemble approaches, and Boosting algorithms. Finally, the classification metrics of the Logistic regression, Decision tree, Random Forest, and XGboost algorithms are compared. The comparisons are completed by employing enhancing strategies such as feature selection and information acquisition. As a result, mobile phone pricing and categorization. This proposed study creates machine learning models that anticipate and categorise prices. The LR is virtually identical to do in and

is frequently used in the same way. LR is used to solve issues connected to classification, whereas regression analysis is used to solve problems related. Some of the authors have described some of the benefits of a decision tree, such as how it is easy to grasp, useful for addressing issues based on decisions, and attempts to find the best possible solution to a particular situation. XGBoost is a shared Gradient Boosting (GB) library that has been modified being more quick, accessible, and configurable. According to multiple publications, the XGBoost algorithm implements machine learning (ML) methods using the GB framework. To handle data science challenges, it employs parallel tree boosting, also known as gradient boosted decision trees (GBDT), GBM, and this approach produces findings in a timely and correct manner.

There were various studies that used feature selection methods and studies that did not use feature selection methods, but it was discovered that when the models were




100
80
60
40
20
0
96.70
88.25
85.41
91.67
Logistic
RandomForest
decisiontree
XGB


100
80
60
40
20
0
96.70
88.25
85.41
91.67
Logistic
RandomForest
decisiontree
XGB

- Hence in every sense and every metrics feature importance have improved the accuracy and Precision of these models, however Logistic regression is providing highest accuracy as well as procession which can be defined as a reliable method to be applied in real time. Finally, the conclusions can be made on the weighted recall values are counted after applying the feature importance where that it has been increased by 1%

for Logistic regression by 3% for random forest and decision tree as well as 1% for the XGB model (Fig. 5.3) which is also good sign and these results are better when compared to that of other studies in the context of mobile price prediction.

6 References

Abdulhafedh, A., 2022. Comparison between Common Statistical Modeling Techniques Used in Research, Including: Discriminant Analysis vs Logistic Regression, Ridge Regression vs LASSO, and Decision Tree vs Random Forest. Open Access Library Journal, 9(2), pp.1-19.

Al Hamad, M. and Zeki, A.M., 2018, November. Accuracy vs. cost in decision trees: A survey. In 2018 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT) (pp. 1-4). IEEE.

Ali, M.M., Paul, B.K., Ahmed, K., Bui, F.M., Quinn, J.M. and Moni, M.A., 2021. Heart disease

Sur, P. and Candès, E.J., 2019. A modern maximum-likelihood theory for high-dimensional logistic regression. Proceedings of the National Academy of Sciences, 116(29), pp.14516-14525.

Thakur, S., Choudhary, J. and Singh, D.P., 2021. A survey on missing values handling methods for time series data. In Intelligent Systems (pp. 435-443). Springer, Singapore.

Tutz, G., 2021. Ordinal trees and random forests: Score-free recursive partitioning and improved ensembles. Journal of Classification, pp.1-23.

Uddin, S., Khan, A., Hossain, M.E. and Moni, M.A., 2019. Comparing different supervised machine learning algorithms for disease prediction. BMC medical informatics and decision making, 19(1), pp.1-16.

Wei, Y., Ma, D. and Dong, J., 2021, December. Optimization for overfitting problems in spam email classification based on parameter adjusting. In International Conference on Algorithms, High Performance Computing, and Artificial Intelligence (AHPCAI 2021) (Vol. 12156, pp. 258-264). SPIE.

Zong, X., Li, R. and Ye, Z., 2021, September. An Intrusion Detection Model Based on Improved Whale Optimization Algorithm and XGBoost. In 2021 11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS) (Vol. 1, pp. 542-547). IEEE.