VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

Winter Semester 2022-2023

CSE3506 – Essentials of Data Analytics

Mobile Price Classification

Faculty- Dr.N.M Elango

Batch Members

1. Vishnupriya S -19MIA1018
2. Kalyani G -19MIA1064
3. Diya Harish - 19MIA1107

Table of Contents

Abstract

Mobile price prediction is a process of using machine learning algorithms to predict the price of mobile devices based on various features such as hardware specifications, brand, model, and market trends. In this task, historical data is used to train a predictive model that can then be used to make predictions on new, unseen data.

The process of mobile price prediction typically involves several steps. First, the data is collected and preprocessed to ensure that it is clean and ready for analysis. Next, a feature selection process is performed to identify the most relevant features for predicting mobile prices. Once the features are selected, a machine learning model is trained on the data to make predictions.

There are several different machine learning algorithms that can be used for mobile price prediction, including linear regression, decision trees, random forests, and neural networks. Each algorithm has its own strengths and weaknesses, and the choice of algorithm depends on the specific requirements of the problem.

Mobile price prediction has a wide range of applications, including in the mobile industry where it can help manufacturers and retailers to set prices for their products, as well as in the finance industry where it can be used for risk assessment and investment analysis. Additionally, mobile price prediction can be used by consumers to make informed decisions about which mobile devices to purchase based on their budget and desired features.

Introduction

Mobile price classification using machine learning is an emerging field that involves the use of various algorithms to classify mobile phones into different price categories based on their features. It can be difficult for customers to evaluate and select the best option for their budget due to the market's growing variety of mobile phone models. This project is aimed at predicting the price range of a mobile device based on its features and specifications. With the increasing popularity of mobile devices and their diverse price ranges, such a project can provide valuable insights for both consumers and manufacturers. By analyzing factors such as processor speed, camera quality, storage capacity, and brand reputation, the project can help consumers make informed purchasing decisions and allow manufacturers to understand the key factors that affect mobile pricing.

Manufacturers are able to develop products that fall within a variety of price ranges by analyzing the trends that are occurring in the market and the features that consumers are willing to pay for. This assists with guaranteeing that their items address the issues and inclinations of various shoppers. Makers can set creation focuses for their items by understanding the interest for various cost ranges,. This helps them make the most of their resources by ensuring that they produce the appropriate number of devices for each price range.

Manufacturers are also able to set competitive prices for their products by evaluating the features and prices of competing devices. This makes it easier for them to keep their products appealing to customers and competitive in the market. Manufacturers can gain insight into consumer behavior by analyzing historical data on trends and prices for mobile devices. New products, consumer preferences, and market trends can all be predicted with this information.

This project involves the use of machine learning algorithms and techniques to analyze and classify mobile devices into different price categories. The ultimate goal is to develop an accurate and efficient model that can predict the price range of a mobile device with a high level of accuracy.

Literature Review

"Mobile Phone Price Class Prediction Using Different Classification Algorithms with Feature Selection and Parameter Optimization" by *Mustafa Çetın; Yunus Koç*

The article "Mobile Phone Price Class Prediction Using Different Classification Algorithms with Feature Selection and Parameter Optimization" suggests a technique for determining the price class of mobile phones by utilizing machine learning techniques. K-Nearest Neighbor's (KNN), Decision Tree (DT), Random Forest (RF), and Support Vector Machine (SVM) are four different categorization techniques used in the study. 20 features are present in each of the 2,000 instances that together constitute the study's dataset. In order to increase the precision of the prediction models, the study use feature selection approaches that reduce the number of features. Chi-Squared (CHI), Information Gain (IG), and Recursive Feature Elimination (RFE) are the feature selection techniques used. The performance of the models is additionally enhanced by the authors' use of parameter optimization. This research contrasts the effectiveness of feature selection and parameter optimization-based classification systems. The outcomes demonstrate that the Random Forest method with CHI-based feature selection and parameter optimization offers the best performance. The RF method has a 90.2% accuracy rate after feature selection and parameter optimization. According to the study's results, parameter optimization and feature selection are efficient methods for boosting the precision of machine learning algorithms' predictions of mobile phone pricing classes. The study also emphasizes how crucial it is to choose the best feature selection method and classification algorithm for a particular dataset.

"Mobile Price Class prediction using Machine Learning Techniques" by *Muhammad Asim*

The primary goal of this research work is to predict "If the mobile with given features will be Economical or Expensive." Real Dataset is collected from www.GSMArena.com website. Various feature selection techniques are used to identify and eliminate redundant and less important characteristics with the least amount of computational complexity. In order to attain the highest accuracy possible, a variety of classifiers are used. Results are contrasted based on the smallest features chosen and the maximum accuracy attained. The optimal feature selection technique and classifier for the given dataset are used to draw conclusions. This research can be applied to any sort of marketing or business to locate the best product (with the smallest amount of money spent and the most features). It is recommended that this research be expanded upon in the future in order to identify more sophisticated solutions to the problem at hand and more precise tools for pricing calculation.

"Prediction of Mobile Phone Price Class using Supervised Machine Learning Techniques" by *A Varun Kiran and Dr. Jebakumar R.*

The objective of this research is to create a model that predicts a mobile's prices given its specifications and to identify the ML algorithm that does so most accurately. The basic concept of predictive analytics is the use of historical data to precisely estimate future occurrences. Applying machine learning is one method of performing predictive analytics.

In order to create and train a prediction model for predictive machine learning, data is used as input. The trained model is then applied to forecast the results of upcoming data instances. Algorithms for supervised machine learning use data that includes a class label for the attribute that has to be predicted. In this paper, the class label represents the cost of a mobile. The prediction model is trained using the Mobile Price Class dataset from the Kaggle data science community website (https://www.kaggle.com/iabhishekofficial/mobileprice-classification), which classifies mobiles into price ranges. Python is chosen because its ML libraries are easily accessible. The model is trained using a variety of classification algorithms in an effort to identify the one that can accurately predict the pricing class for mobile devices. The trained model is evaluated using metrics like accuracy score, confusion matrix, etc. to choose the best method out of those being used.

"Getting Smart About Phones: New Price Indexes and the Allocation of Spending Between Devices and Services Plans in Personal Consumption Expenditures" by *Ana Aizcorbe, David M. Byrne & Daniel E. Sichel*

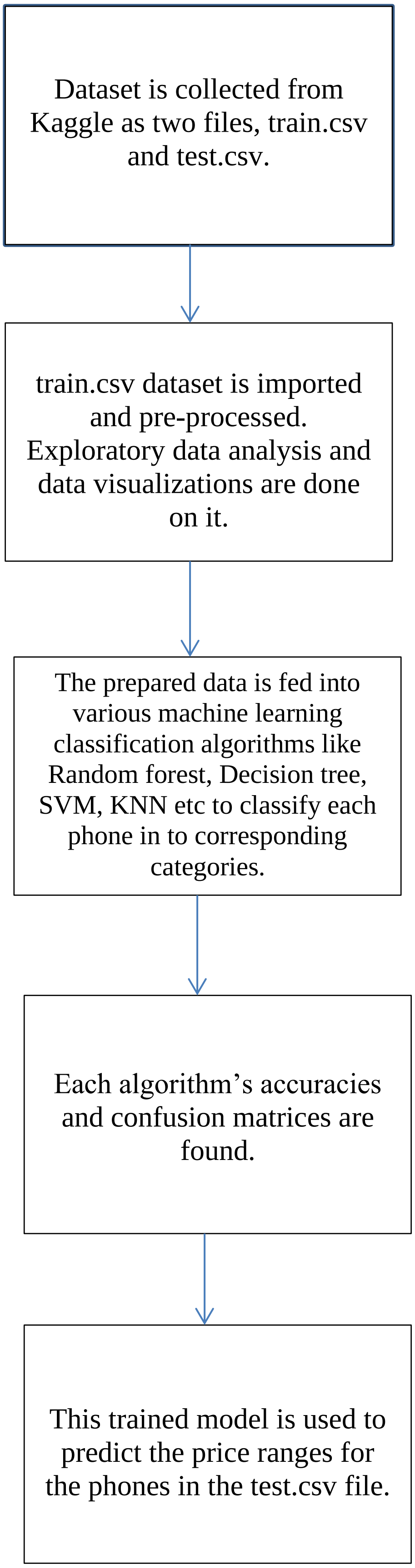
This paper addresses two mobile phone measurement-related challenges. First, we create a new mobile phone price index utilizing matched-model indices for feature phones and hedonic quality-adjusted pricing for smartphones. Between 2010 and 2018, the reduction of the index was 17 percent per year on average, which is comparable to the pace of decline in the GDP Accounts' price index. The indicator indicates a significant improvement in quality given the relatively stable average pricing throughout this time. Second, we provide a mechanism to separate wireless service and phone purchases when they are combined as a part of a long-term service agreement. Real PCE requires accurate allocation because the price deflators for wireless services and phones show significantly distinct patterns. According to the corrected estimates, the category of cellular phone services currently includes real PCE spending that grew 4 percentage points faster than indicated by the data that has been publicly released.

Problem Description

If someone launches their own mobile business, he intends to battle hard against large corporations like Samsung, Apple, and others. He is unable to estimate the cost of the mobile devices that his business makes. It is impossible to make assumptions in this cutthroat mobile phone market. He gathers sales information on mobile phones from numerous vendors to address this issue. He is not very skilled at machine learning, but he wants to determine a relationship between mobile phones' features (such as RAM, internal memory, etc.) and its selling price. There comes the application of this project, Mobile price classification using Machine Learning

In this project instead of predicting the actual price, we just have to provide a price range showing how high the price is.

Workflow



Dataset is collected from Kaggle as two files, train.csv and test.csv.

train.csv dataset is imported and pre-processed. Exploratory data analysis and data visualizations are done on it.

The prepared data is fed into various machine learning classification algorithms like Random forest, Decision tree, SVM, KNN etc to classify each phone in to corresponding categories.

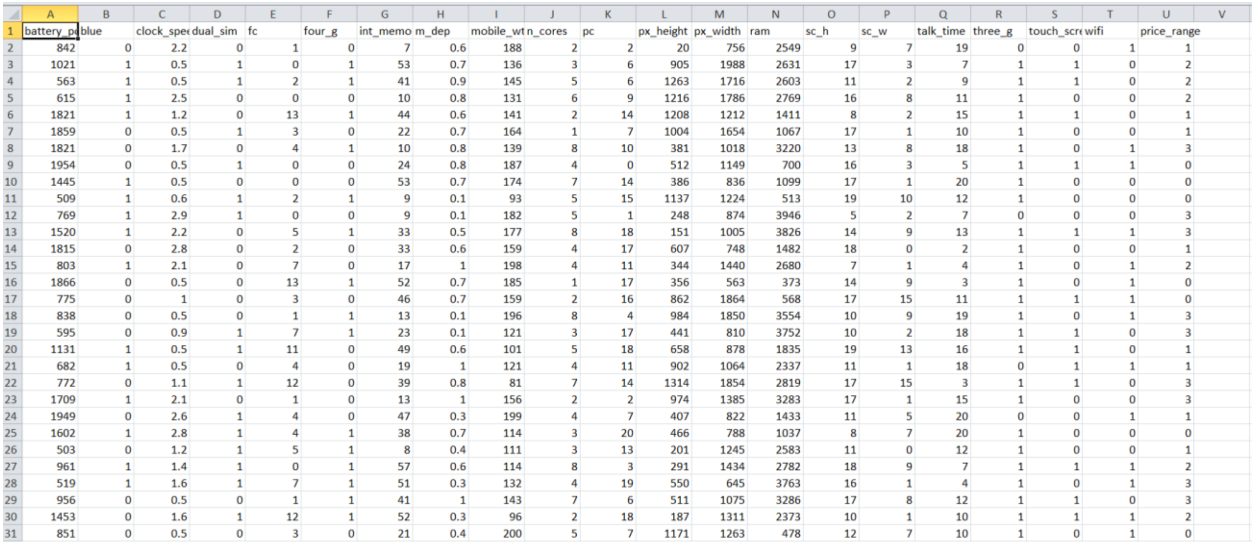
Each algorithm’s accuracies and confusion matrices are found.

This trained model is used to predict the price ranges for the phones in the test.csv file.

Dataset

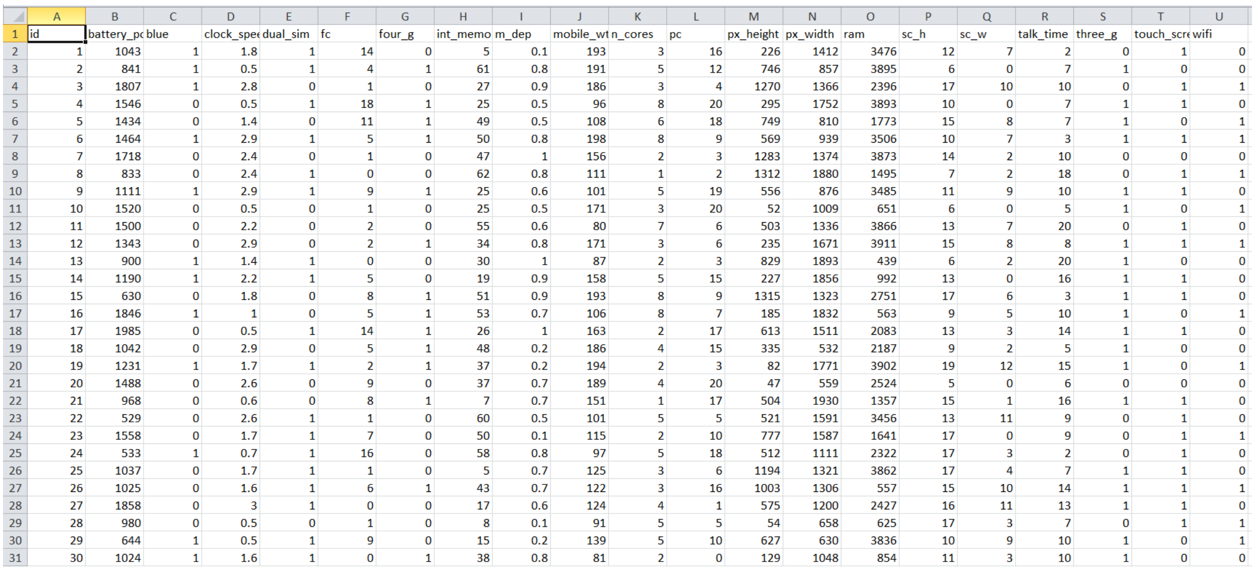
https://www.kaggle.com/datasets/iabhishekofficial/mobile-price-classification

train.csv



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1	battery_pc	blue	clock_spee	dual_sim	fc	four_g	int_memo	m_dep	mobile_wt	n_cores	pc	px_height	px_width	ram	sc_h	sc_w	talk_time	three_g	touch_scre	wifi	price_range	
2	842	0	2.2	0	1	0	7	0.6	188	2	2	20	756	2549	9	7	19	0	0	1	1	
3	1021	1	0.5	1	0	1	53	0.7	136	3	6	905	1988	2631	17	3	7	1	1	0	2	
4	563	1	0.5	1	2	1	41	0.9	145	5	6	1263	1716	2603	11	2	9	1	1	0	2	
5	615	1	2.5	0	0	0	10	0.8	131	6	9	1216	1786	2769	16	8	11	1	0	0	2	
6	1821	1	1.2	0	13	1	44	0.6	141	2	14	1208	1212	1411	8	2	15	1	1	0	1	
7	1859	0	0.5	1	3	0	22	0.7	164	1	7	1004	1654	1067	17	1	10	1	0	0	1	
8	1821	0	1.7	0	4	1	10	0.8	139	8	10	381	1018	3220	13	8	18	1	0	1	3	
9	1954	0	0.5	1	0	0	24	0.8	187	4	0	512	1149	700	16	3	5	1	1	1	0	
10	1445	1	0.5	0	0	0	53	0.7	174	7	14	386	836	1099	17	1	20	1	0	0	0	
11	509	1	0.6	1	2	1	9	0.1	93	5	15	1137	1224	513	19	10	12	1	0	0	0	
12	769	1	2.9	1	0	0	9	0.1	182	5	1	248	874	3946	5	2	7	0	0	0	3	
13	1520	1	2.2	0	5	1	33	0.5	177	8	18	151	1005	3826	14	9	13	1	1	1	3	
14	1815	0	2.8	0	2	0	33	0.6	159	4	17	607	748	1482	18	0	2	1	0	0	1	
15	803	1	2.1	0	7	0	17	1	198	4	11	344	1440	2680	7	1	4	1	0	1	2	
16	1866	0	0.5	0	13	1	52	0.7	185	1	17	356	563	373	14	9	3	1	0	1	0	
17	775	0	1	0	3	0	46	0.7	159	2	16	862	1864	568	17	15	11	1	1	1	0	
18	838	0	0.5	0	1	1	13	0.1	196	8	4	984	1850	3554	10	9	19	1	0	1	3	
19	595	0	0.9	1	7	1	23	0.1	121	3	17	441	810	3752	10	2	18	1	1	0	3	
20	1131	1	0.5	1	11	0	49	0.6	101	5	18	658	878	1835	19	13	16	1	1	0	1	
21	682	1	0.5	0	4	0	19	1	121	4	11	902	1064	2337	11	1	18	0	1	1	1	
22	772	0	1.1	1	12	0	39	0.8	81	7	14	1314	1854	2819	17	15	3	1	1	0	3	
23	1709	1	2.1	0	1	0	13	1	156	2	2	974	1385	3283	17	1	15	1	0	0	3	
24	1949	0	2.6	1	4	0	47	0.3	199	4	7	407	822	1433	11	5	20	0	0	1	1	
25	1602	1	2.8	1	4	1	38	0.7	114	3	20	466	788	1037	8	7	20	1	0	0	0	
26	503	0	1.2	1	5	1	8	0.4	111	3	13	201	1245	2583	11	0	12	1	0	0	1	
27	961	1	1.4	1	0	1	57	0.6	114	8	3	291	1434	2782	18	9	7	1	1	1	2	
28	519	1	1.6	1	7	1	51	0.3	132	4	19	550	645	3763	16	1	4	1	0	1	3	
29	956	0	0.5	0	1	1	41	1	143	7	6	511	1075	3286	17	8	12	1	1	0	3	
30	1453	0	1.6	1	12	1	52	0.3	96	2	18	187	1311	2373	10	1	10	1	1	1	2	
31	851	0	0.5	0	3	0	21	0.4	200	5	7	1171	1263	478	12	7	10	1	0	1	0	

test.csv



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	id	battery_pc	blue	clock_spee	dual_sim	fc	four_g	int_memo	m_dep	mobile_wt	n_cores	pc	px_height	px_width	ram	sc_h	sc_w	talk_time	three_g	touch_scre	wifi
2	1	1043	1	1.8	1	14	0	5	0.1	193	3	16	226	1412	3476	12	7	2	0	1	0
3	2	841	1	0.5	1	4	1	61	0.8	191	5	12	746	857	3895	6	0	7	1	0	0
4	3	1807	1	2.8	0	1	0	27	0.9	186	3	4	1270	1366	2396	17	10	10	0	1	1
5	4	1546	0	0.5	1	18	1	25	0.5	96	8	20	295	1752	3893	10	0	7	1	1	0
6	5	1434	0	1.4	0	11	1	49	0.5	108	6	18	749	810	1773	15	8	7	1	0	1
7	6	1464	1	2.9	1	5	1	50	0.8	198	8	9	569	939	3506	10	7	3	1	1	1
8	7	1718	0	2.4	0	1	0	47	1	156	2	3	1283	1374	3873	14	2	10	0	0	0
9	8	833	0	2.4	1	0	0	62	0.8	111	1	2	1312	1880	1495	7	2	18	0	1	1
10	9	1111	1	2.9	1	9	1	25	0.6	101	5	19	556	876	3485	11	9	10	1	1	0
11	10	1520	0	0.5	0	1	0	25	0.5	171	3	20	52	1009	651	6	0	5	1	0	1
12	11	1500	0	2.2	0	2	0	55	0.6	80	7	6	503	1336	3866	13	7	20	0	1	0
13	12	1343	0	2.9	0	2	1	34	0.8	171	3	6	235	1671	3911	15	8	8	1	1	1
14	13	900	1	1.4	1	0	0	30	1	87	2	3	829	1893	439	6	2	20	1	0	0
15	14	1190	1	2.2	1	5	0	19	0.9	158	5	15	227	1856	992	13	0	16	1	1	0
16	15	630	0	1.8	0	8	1	51	0.9	193	8	9	1315	1323	2751	17	6	3	1	1	0
17	16	1846	1	1	0	5	1	53	0.7	106	8	7	185	1832	563	9	5	10	1	0	1
18	17	1985	0	0.5	1	14	1	26	1	163	2	17	613	1511	2083	13	3	14	1	1	0
19	18	1042	0	2.9	0	5	1	48	0.2	186	4	15	335	532	2187	9	2	5	1	0	0
20	19	1231	1	1.7	1	2	1	37	0.2	194	2	3	82	1771	3902	19	12	15	1	0	1
21	20	1488	0	2.6	0	9	0	37	0.7	189	4	20	47	559	2524	5	0	6	0	0	0
22	21	968	0	0.6	0	8	1	7	0.7	151	1	17	504	1930	1357	15	1	16	1	1	0
23	22	529	0	2.6	1	1	0	60	0.5	101	5	5	521	1591	3456	13	11	9	0	1	0
24	23	1558	0	1.7	1	7	0	50	0.1	115	2	10	777	1587	1641	17	0	9	0	1	1
25	24	533	1	0.7	1	16	0	58	0.8	97	5	18	512	1111	2322	17	3	2	0	1	0
26	25	1037	0	1.7	1	1	0	5	0.7	125	3	6	1194	1321	3862	17	4	7	1	1	0
27	26	1025	0	1.6	1	6	1	43	0.7	122	3	16	1003	1306	557	15	10	14	1	1	1
28	27	1858	0	3	1	0	0	17	0.6	124	4	1	575	1200	2427	16	11	13	1	1	0
29	28	980	0	0.5	0	1	0	8	0.1	91	5	5	54	658	625	17	3	7	0	1	1
30	29	644	1	0.5	1	9	0	15	0.2	139	5	10	627	630	3836	10	9	10	1	0	1
31	30	1024	1	1.6	1	0	1	38	0.8	81	2	0	129	1048	854	11	3	10	1	0	0

This dataset contains information on various mobile devices and their features, as well as their price range.

The dataset contains the following features:

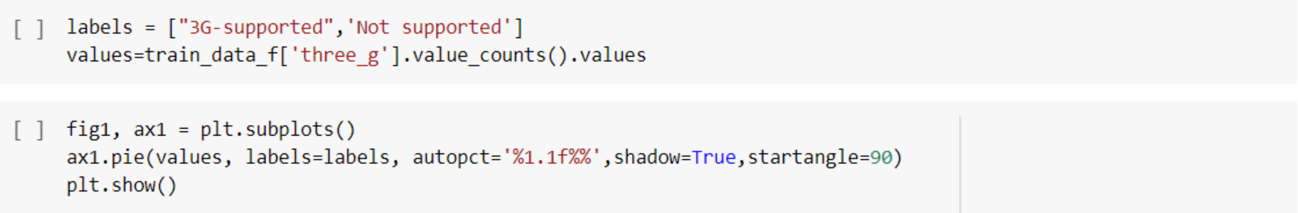
battery_power: The total energy a battery can store in one time measured in mAh.
blue: Has Bluetooth or not.
clock_speed: The speed at which the microprocessor executes instructions.
dual_sim: Has dual sim support or not.
fc: Front Camera mega pixels.
four_g: Has 4G or not.

int_memory: Internal Memory in Gigabytes.
m_dep: Mobile Depth in cm.
mobile_wt: Weight of the mobile phone.
n_cores: Number of cores of the processor.
pc: Primary Camera mega pixels.
px_height: Pixel Resolution Height.
px_width: Pixel Resolution Width.
ram: Random Access Memory in Megabytes.
sc_h: Screen Height of the mobile in cm.
sc_w: Screen Width of the mobile in cm.
talk_time: The longest time that a single battery charge will last when you are on a call.
three_g: Has 3G or not.
touch_screen: Has touch screen or not.
wifi: Has wifi or not.

The target variable in this dataset is the price range, which is divided into four categories: low-cost, medium-cost, high-cost, and very high-cost.

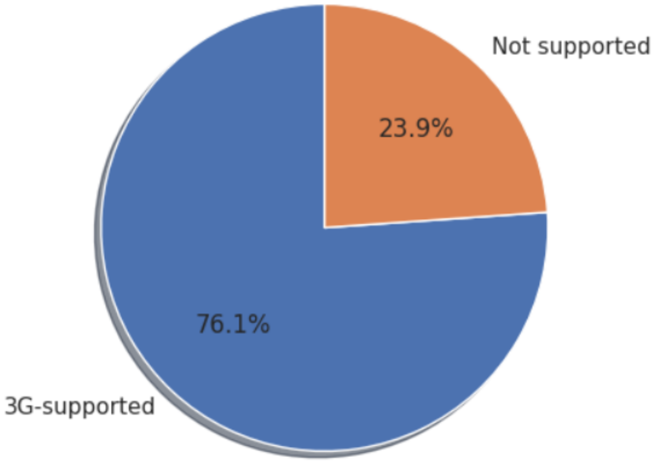
Visualizations

Pie chart of percentage of phones that support 3G

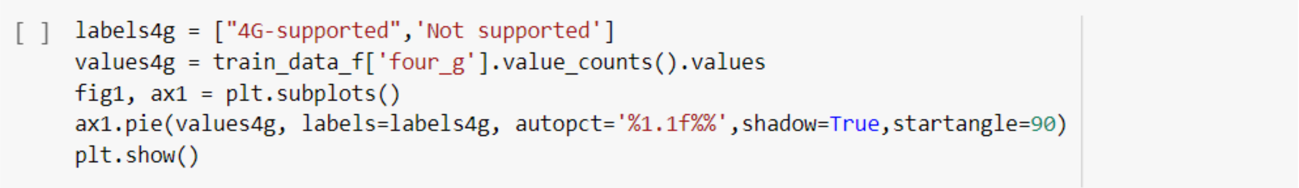


```
[ ] labels = ["3G-supported",'Not supported']
    values=train_data_f['three_g'].value_counts().values
```

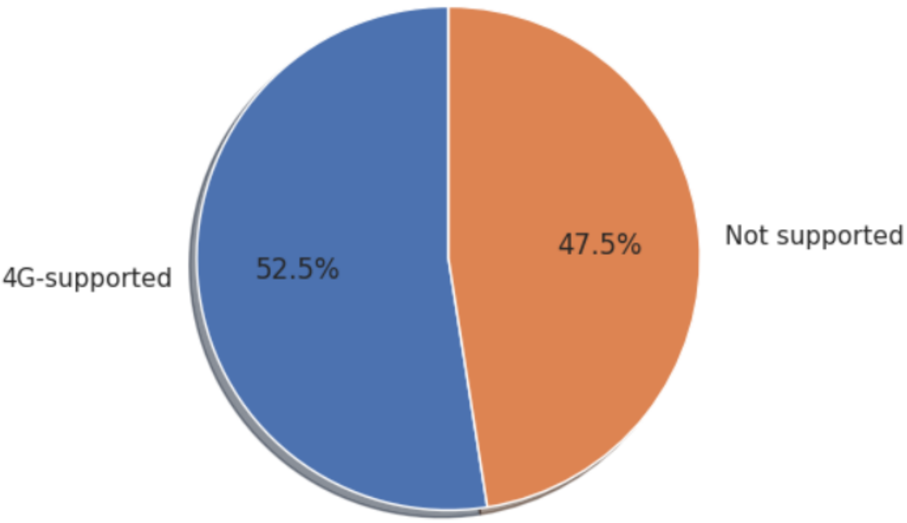
```
[ ] fig1, ax1 = plt.subplots()
    ax1.pie(values, labels=labels, autopct='%1.1f%%',shadow=True,startangle=90)
    plt.show()
```



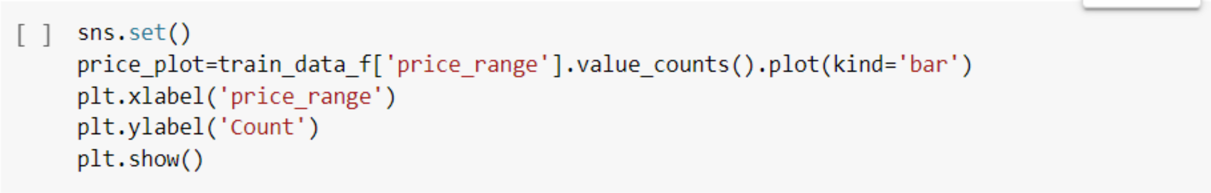
Pie chart of percentage of phones that support 4G



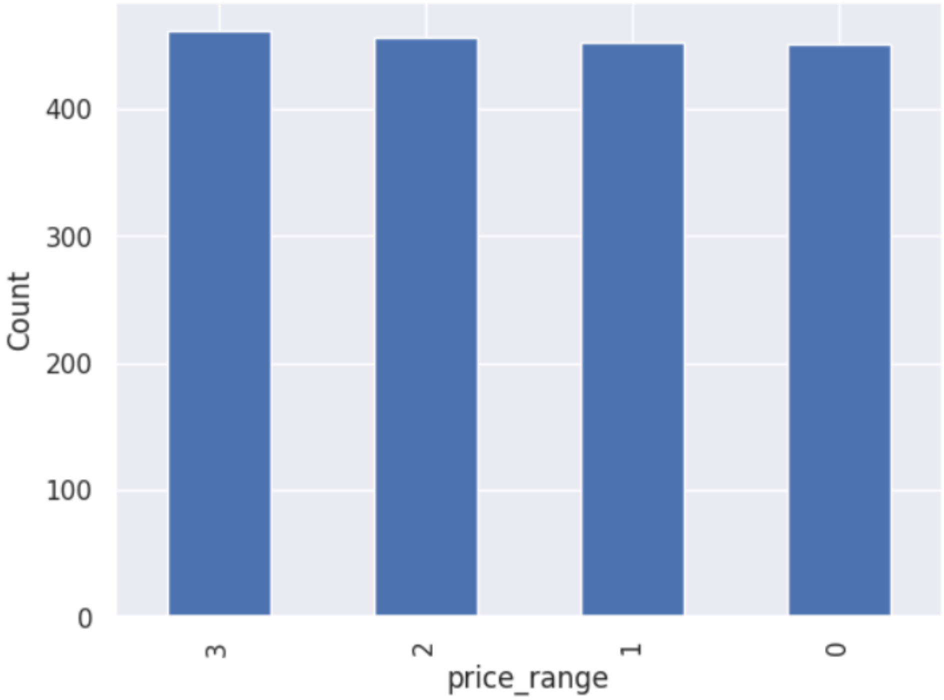
```
[ ]  labels4g = ["4G-supported",'Not supported']
     values4g = train_data_f['four_g'].value_counts().values
     fig1, ax1 = plt.subplots()
     ax1.pie(values4g, labels=labels4g, autopct='%1.1f%%',shadow=True,startangle=90)
     plt.show()
```



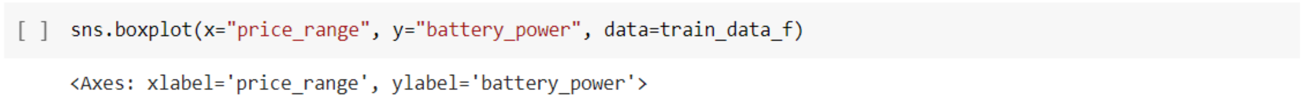
Bar plot showing count of phones in each category



```
[ ]  sns.set()
     price_plot=train_data_f['price_range'].value_counts().plot(kind='bar')
     plt.xlabel('price_range')
     plt.ylabel('Count')
     plt.show()
```

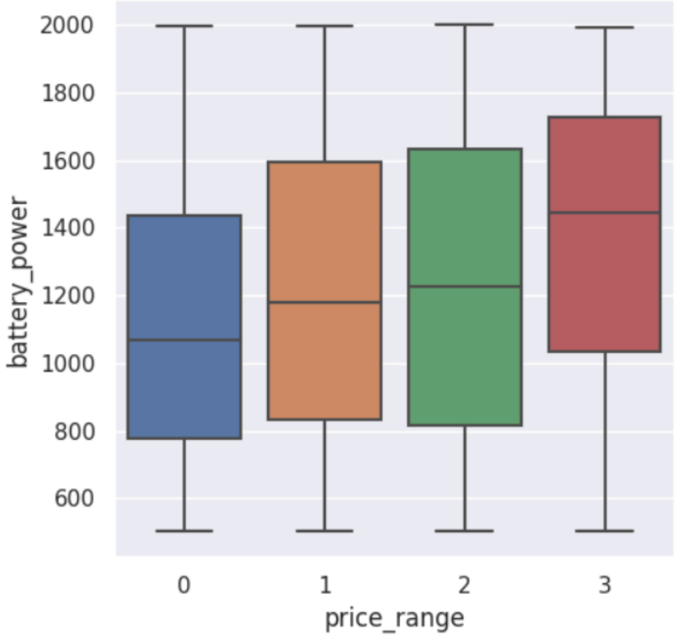


Boxplot between battery power and price range

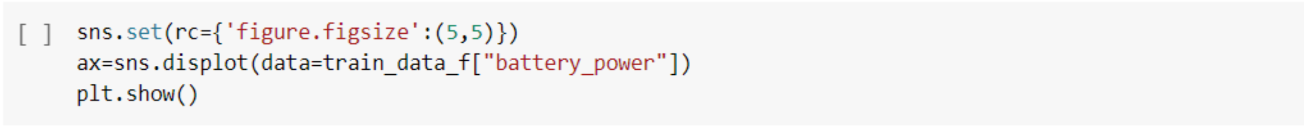


```
[ ] sns.boxplot(x="price_range", y="battery_power", data=train_data_f)
```

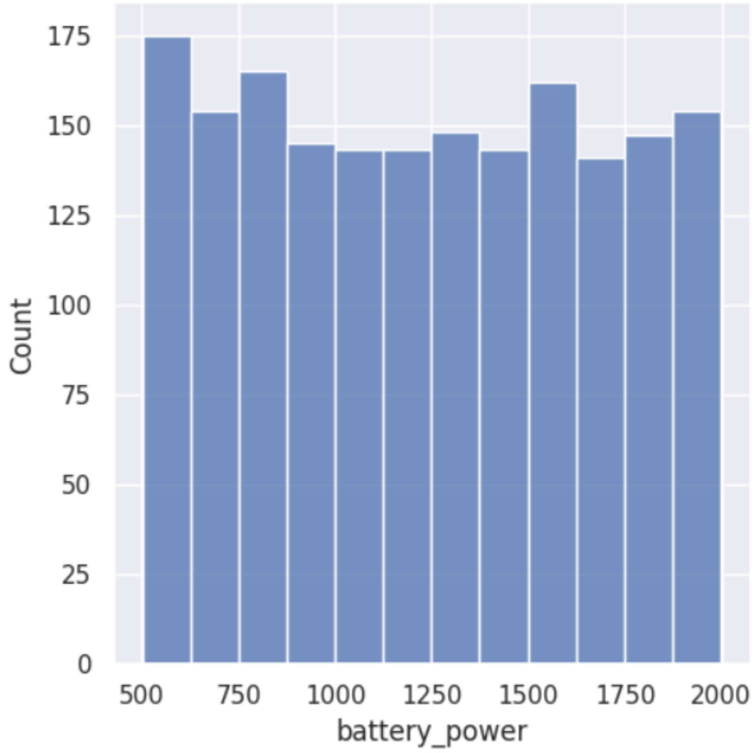
```
<Axes: xlabel='price_range', ylabel='battery_power'>
```



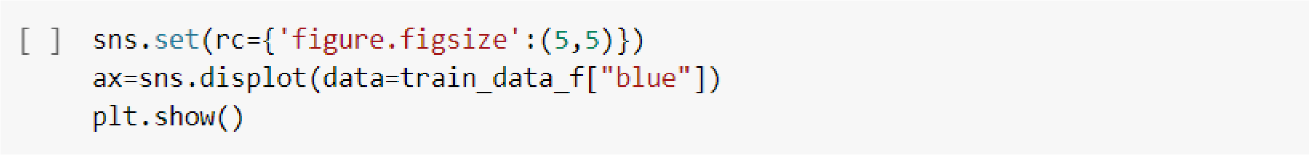
Histogram showing frequency of each bucket created for the battery power attribute.



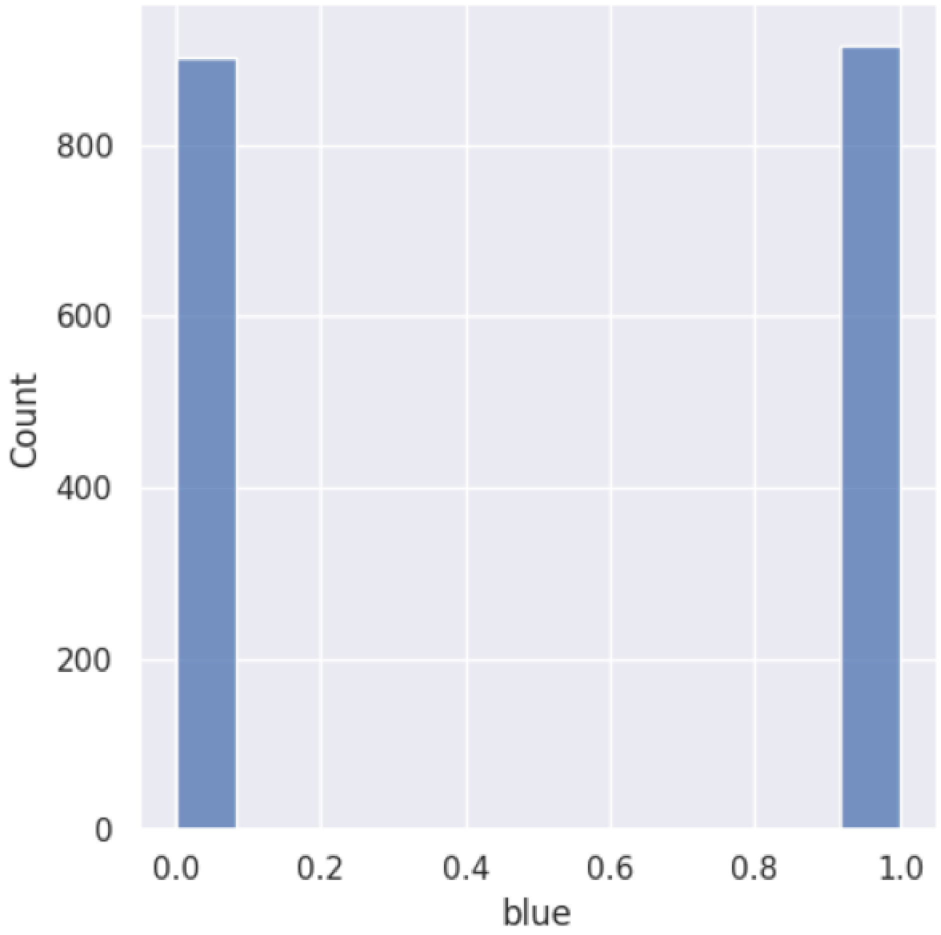
```
[ ] sns.set(rc={'figure.figsize':(5,5)})
    ax=sns.displot(data=train_data_f["battery_power"])
    plt.show()
```



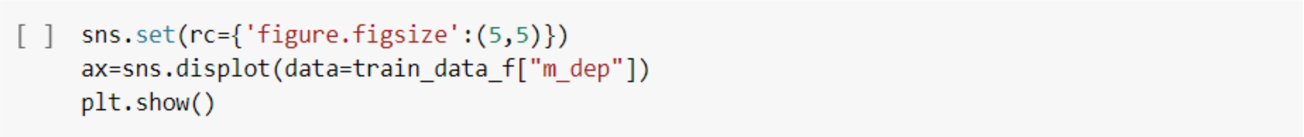
Histogram showing frequency of each bucket created for the blue attribute which indicates if the phone has Bluetooth or not.



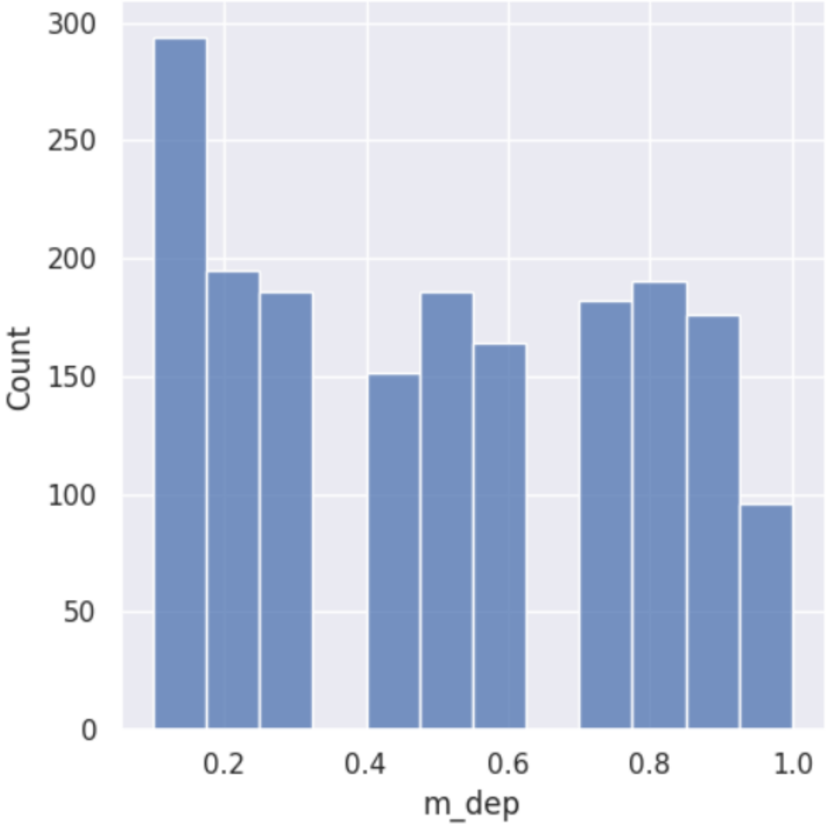
```
[ ]  sns.set(rc={'figure.figsize':(5,5)})
     ax=sns.displot(data=train_data_f["blue"])
     plt.show()
```



Histogram showing frequency of each bucket created for the mobile depth attribute.

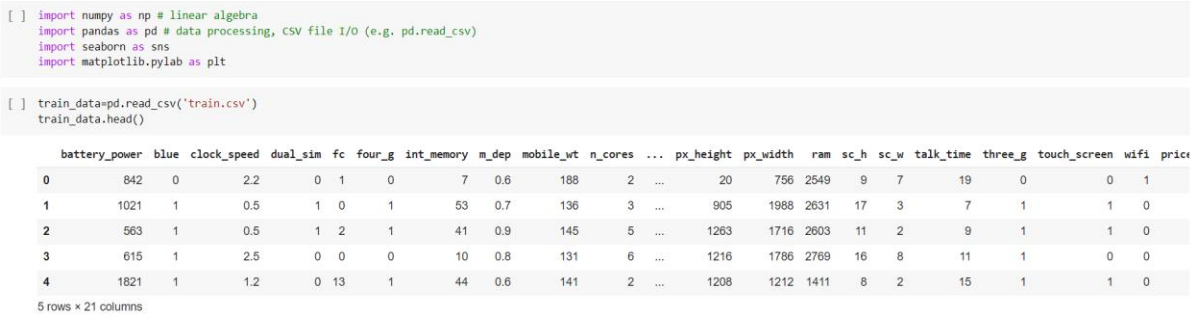


```
[ ]  sns.set(rc={'figure.figsize':(5,5)})
     ax=sns.displot(data=train_data_f["m_dep"])
     plt.show()
```



Methodology

The dataset train.csv is imported and displayed.



```
[ ]  import numpy as np # linear algebra
     import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
     import seaborn as sns
     import matplotlib.pylab as plt
```

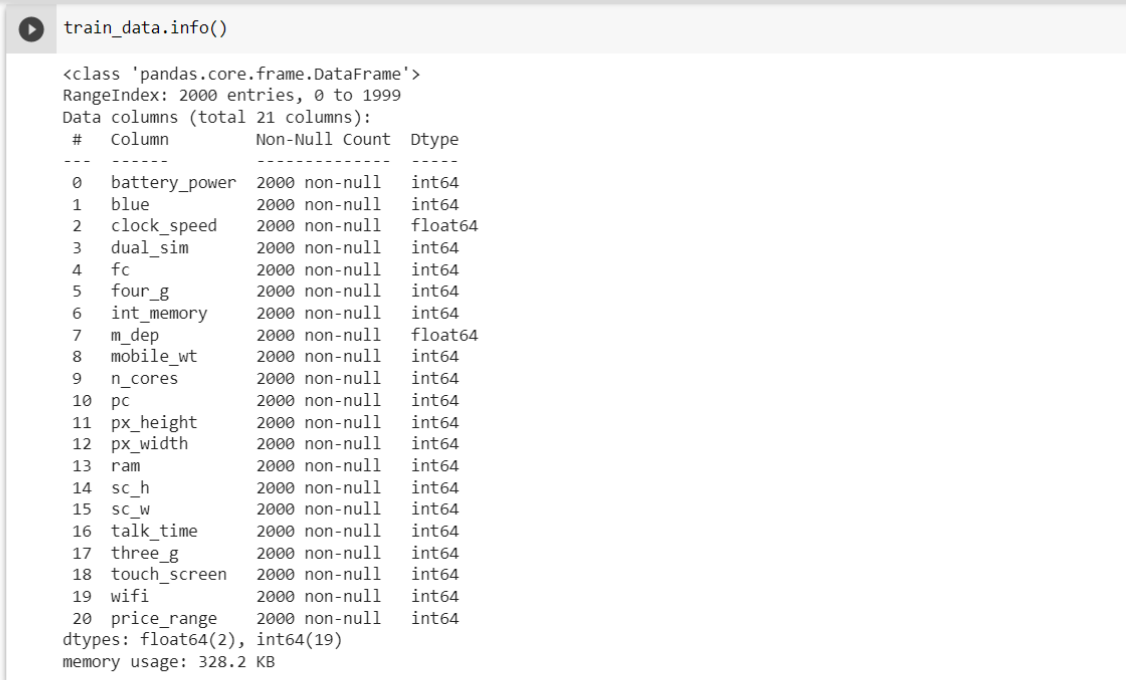
```
[ ]  train_data=pd.read_csv('train.csv')
     train_data.head()
```

	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
0	842	0	2.2	0	1	0	7	0.6	188	2	...	20	756	2549	9	7	19	0	0	1	
1	1021	1	0.5	1	0	1	53	0.7	136	3	...	905	1988	2631	17	3	7	1	1	0	
2	563	1	0.5	1	2	1	41	0.9	145	5	...	1263	1716	2603	11	2	9	1	1	0	
3	615	1	2.5	0	0	0	10	0.8	131	6	...	1216	1786	2769	16	8	11	1	0	0	
4	1821	1	1.2	0	13	1	44	0.6	141	2	...	1208	1212	1411	8	2	15	1	1	0	

5 rows × 21 columns

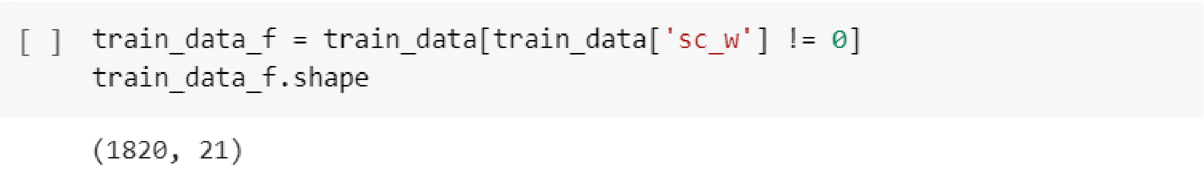
The info() function is used to display the different columns present in our dataset and their corresponding data types.

The shape() function showed that the dataset has 1820 rows and 21 columns.



```
train_data.info()
```

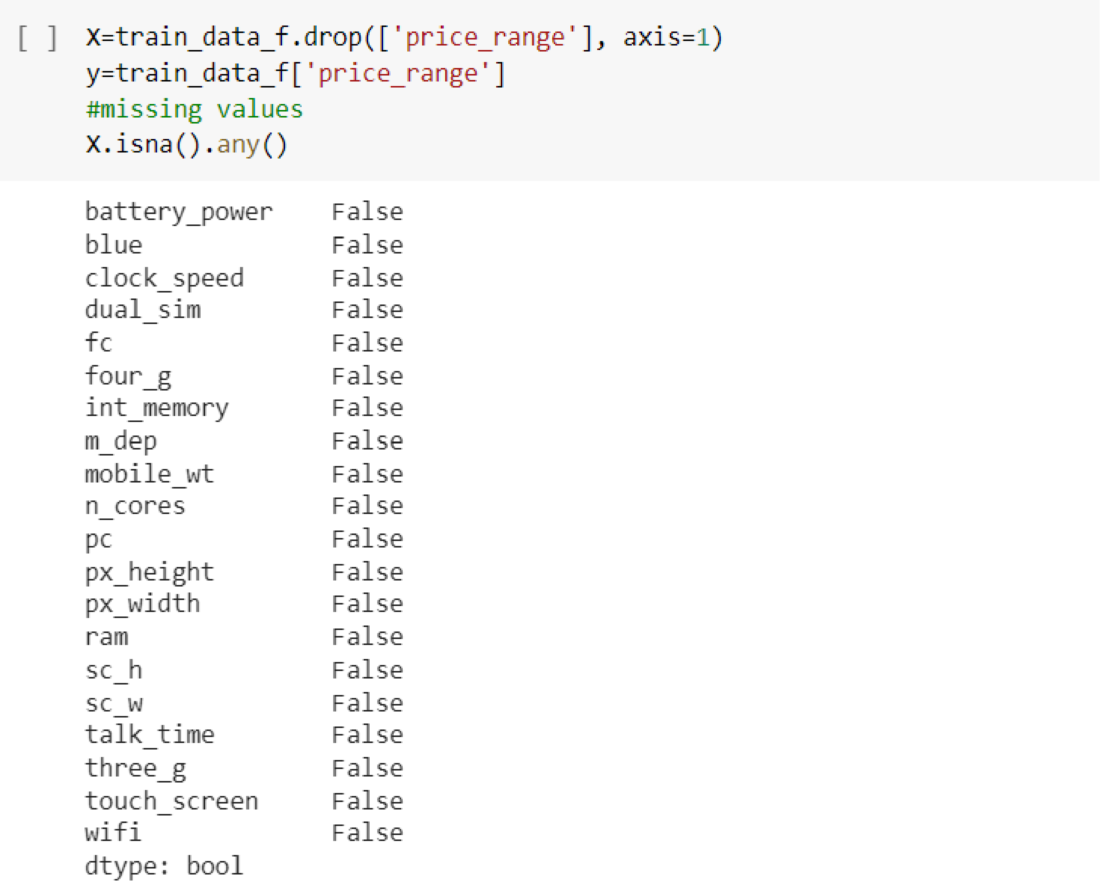
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   battery_power  2000 non-null   int64
 1   blue           2000 non-null   int64
 2   clock_speed    2000 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_f = train_data[train_data['sc_w'] != 0]
     train_data_f.shape
```

```
(1820, 21)
```

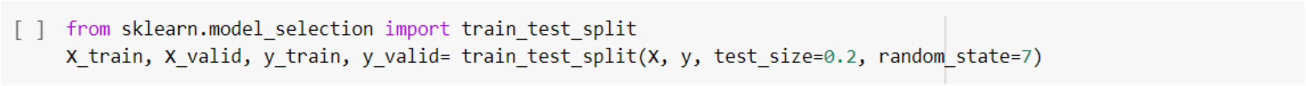
The dataset is divided into training and testing sets. The train data consists of all columns except the price range and the test data is only the target variable which is the price range category.



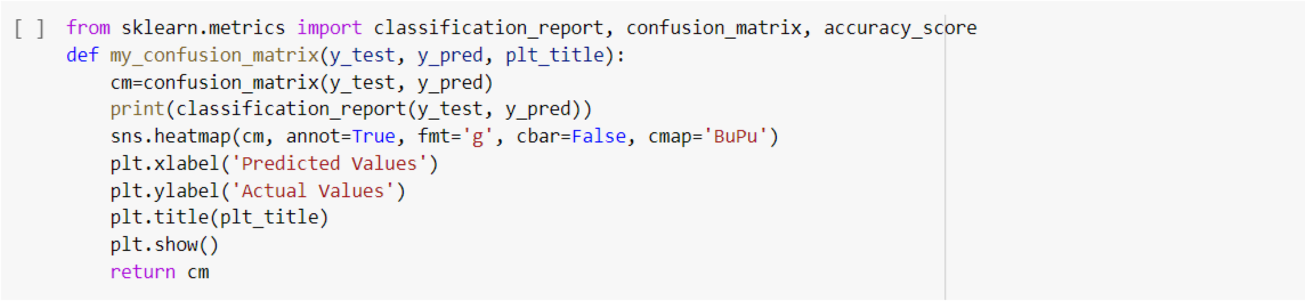
```
[ ]  X=train_data_f.drop(['price_range'], axis=1)
     y=train_data_f['price_range']
     #missing values
     X.isna().any()
```

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

The data is split into training and validation sets in the ratio 80:20. The function to generate the confusion matrix is defined.



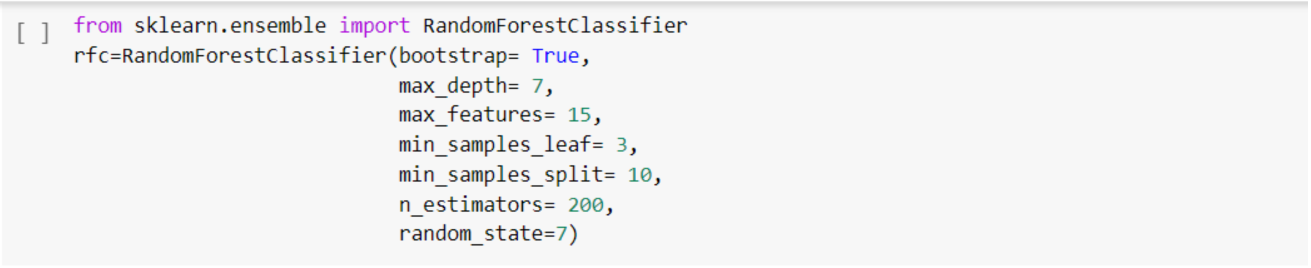
```
[ ]  from sklearn.model_selection import train_test_split
     X_train, X_valid, y_train, y_valid= train_test_split(X, y, test_size=0.2, random_state=7)
```



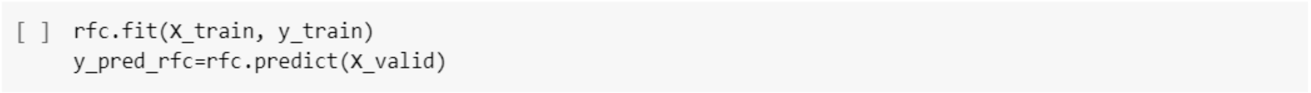
```
[ ]  from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
     def my_confusion_matrix(y_test, y_pred, plt_title):
         cm=confusion_matrix(y_test, y_pred)
         print(classification_report(y_test, y_pred))
         sns.heatmap(cm, annot=True, fmt='g', cbar=False, cmap='BuPu')
         plt.xlabel('Predicted Values')
         plt.ylabel('Actual Values')
         plt.title(plt_title)
         plt.show()
         return cm
```

Machine learning algorithms:

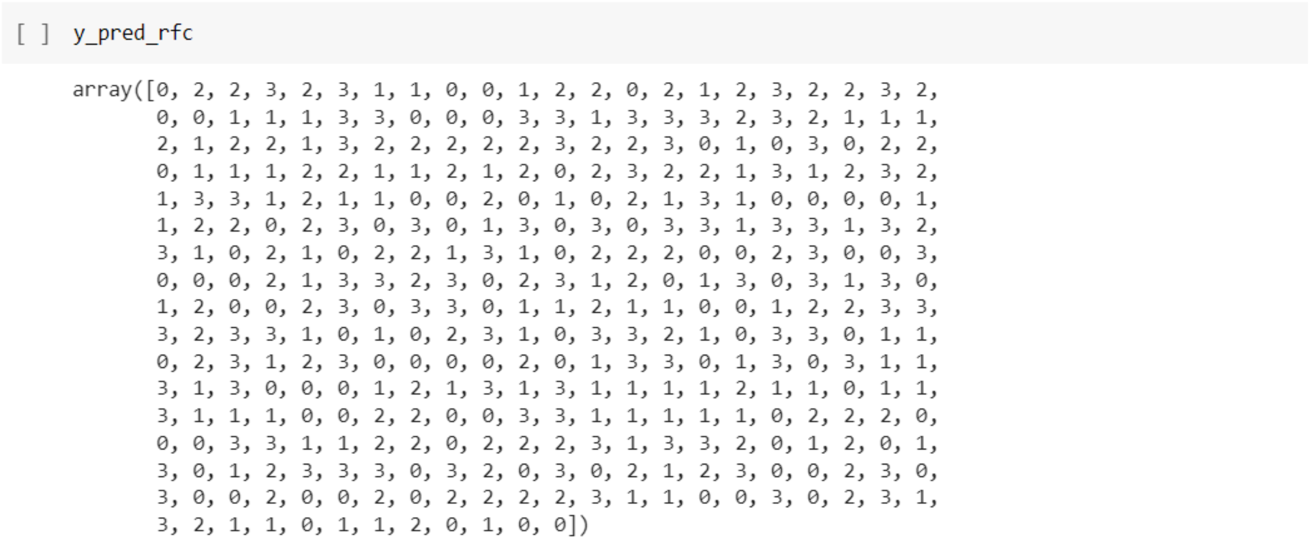
Random Forest: Training data is fit into the random forest classifier model and x_valid is used to predict the classes. Y_pred_rfc gives the classes into which each record in our dataset belong based on this model.



```
[ ]  from sklearn.ensemble import RandomForestClassifier
     rfc=RandomForestClassifier(bootstrap= True,
                                max_depth= 7,
                                max_features= 15,
                                min_samples_leaf= 3,
                                min_samples_split= 10,
                                n_estimators= 200,
                                random_state=7)
```

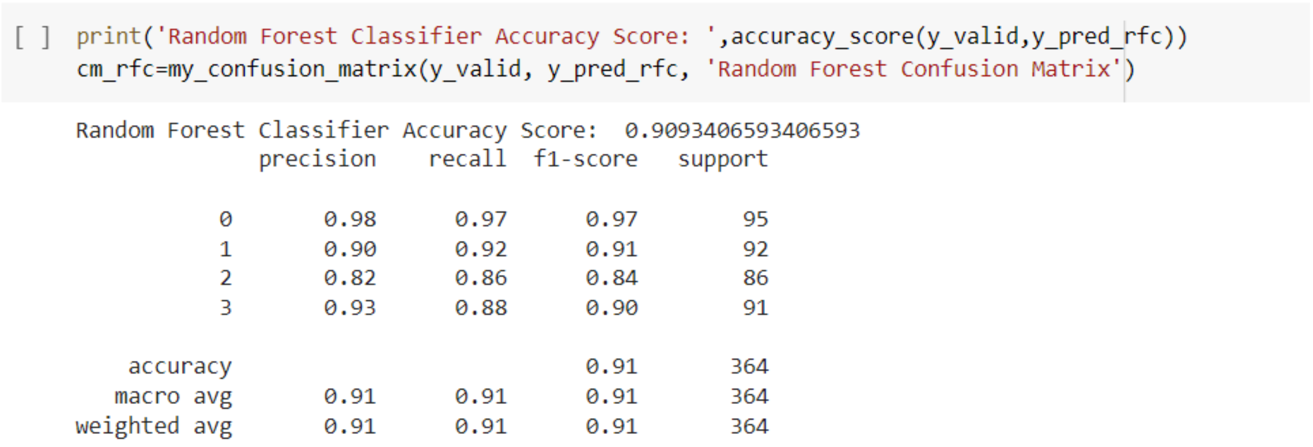


```
[ ]  rfc.fit(X_train, y_train)
     y_pred_rfc=rfc.predict(X_valid)
```



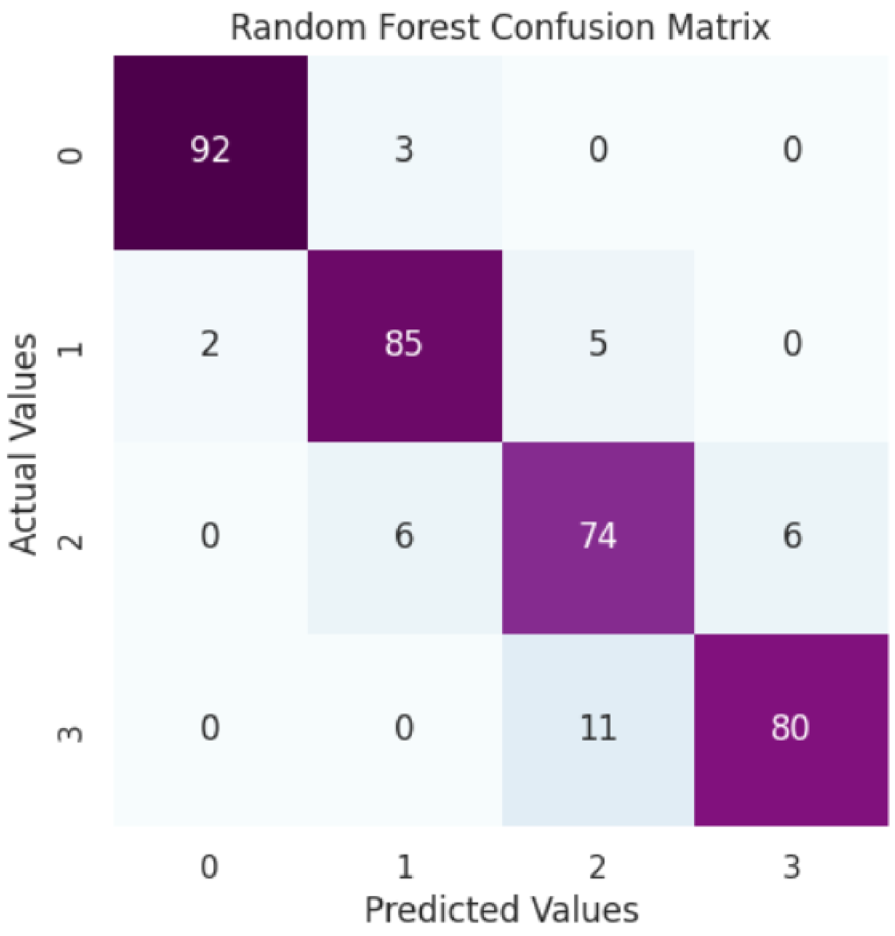
```
[ ]  y_pred_rfc

     array([0, 2, 2, 3, 2, 3, 1, 1, 0, 0, 1, 2, 2, 0, 2, 1, 2, 3, 2, 2, 3, 2,
            0, 0, 1, 1, 1, 3, 3, 0, 0, 0, 3, 3, 1, 3, 3, 3, 2, 3, 2, 1, 1, 1,
            2, 1, 2, 2, 1, 3, 2, 2, 2, 2, 2, 3, 2, 2, 3, 0, 1, 0, 3, 0, 2, 2,
            0, 1, 1, 1, 2, 2, 1, 1, 2, 1, 2, 0, 2, 3, 2, 2, 1, 3, 1, 2, 3, 2,
            1, 3, 3, 1, 2, 1, 1, 0, 0, 2, 0, 1, 0, 2, 1, 3, 1, 0, 0, 0, 0, 1,
            1, 2, 2, 0, 2, 3, 0, 3, 0, 1, 3, 0, 3, 0, 3, 3, 1, 3, 3, 1, 3, 2,
            3, 1, 0, 2, 1, 0, 2, 2, 1, 3, 1, 0, 2, 2, 2, 0, 0, 2, 3, 0, 0, 3,
            0, 0, 0, 2, 1, 3, 3, 2, 3, 0, 2, 3, 1, 2, 0, 1, 3, 0, 3, 1, 3, 0,
            1, 2, 0, 0, 2, 3, 0, 3, 3, 0, 1, 1, 2, 1, 1, 0, 0, 1, 2, 2, 3, 3,
            3, 2, 3, 3, 1, 0, 1, 0, 2, 3, 1, 0, 3, 3, 2, 1, 0, 3, 3, 0, 1, 1,
            0, 2, 3, 1, 2, 3, 0, 0, 0, 0, 2, 0, 1, 3, 3, 0, 1, 3, 0, 3, 1, 1,
            3, 1, 3, 0, 0, 0, 1, 2, 1, 3, 1, 3, 1, 1, 1, 1, 2, 1, 1, 0, 1, 1,
            3, 1, 1, 1, 0, 0, 2, 2, 0, 0, 3, 3, 1, 1, 1, 1, 1, 0, 2, 2, 2, 0,
            0, 0, 3, 3, 1, 1, 2, 2, 0, 2, 2, 2, 3, 1, 3, 3, 2, 0, 1, 2, 0, 1,
            3, 0, 1, 2, 3, 3, 3, 0, 3, 2, 0, 3, 0, 2, 1, 2, 3, 0, 0, 2, 3, 0,
            3, 0, 0, 2, 0, 0, 2, 0, 2, 2, 2, 2, 3, 1, 1, 0, 0, 3, 0, 2, 3, 1,
            3, 2, 1, 1, 0, 1, 1, 2, 0, 1, 0, 0])
```

Accuracy of this random forest model is found to be 90.93%

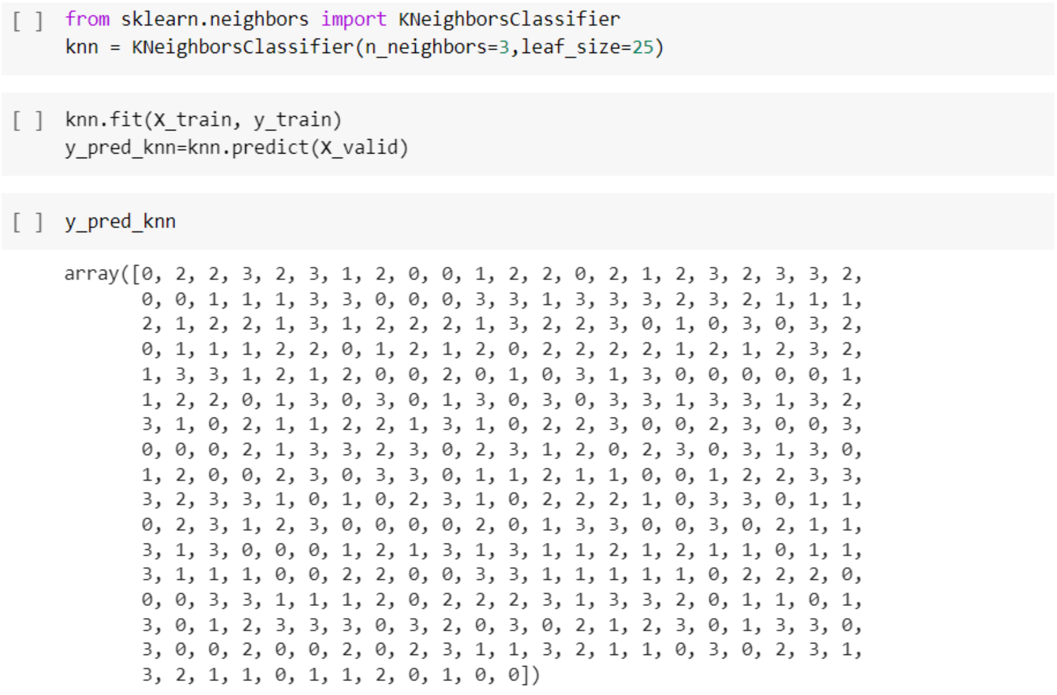


```
[ ]  print('Random Forest Classifier Accuracy Score: ',accuracy_score(y_valid,y_pred_rfc))
     cm_rfc=my_confusion_matrix(y_valid, y_pred_rfc, 'Random Forest Confusion Matrix')

     Random Forest Classifier Accuracy Score:  0.9093406593406593
                   precision    recall  f1-score   support

                0       0.98      0.97      0.97        95
                1       0.90      0.92      0.91        92
                2       0.82      0.86      0.84        86
                3       0.93      0.88      0.90        91

         accuracy                           0.91       364
        macro avg       0.91      0.91      0.91       364
     weighted avg       0.91      0.91      0.91       364
```

The confusion matrix below indicates that the actual and predicted values are found to be same for almost all the records, say, out of 95 records in 0 category, only 3 were misclassified as 1. The very few misclassifications indicate that the model has good accuracy.



KNN: Training data is fit into the KNN classifier model and x_valid is used to predict the classes. Y_pred_knn gives the classes into which each record in our dataset belong based on this model

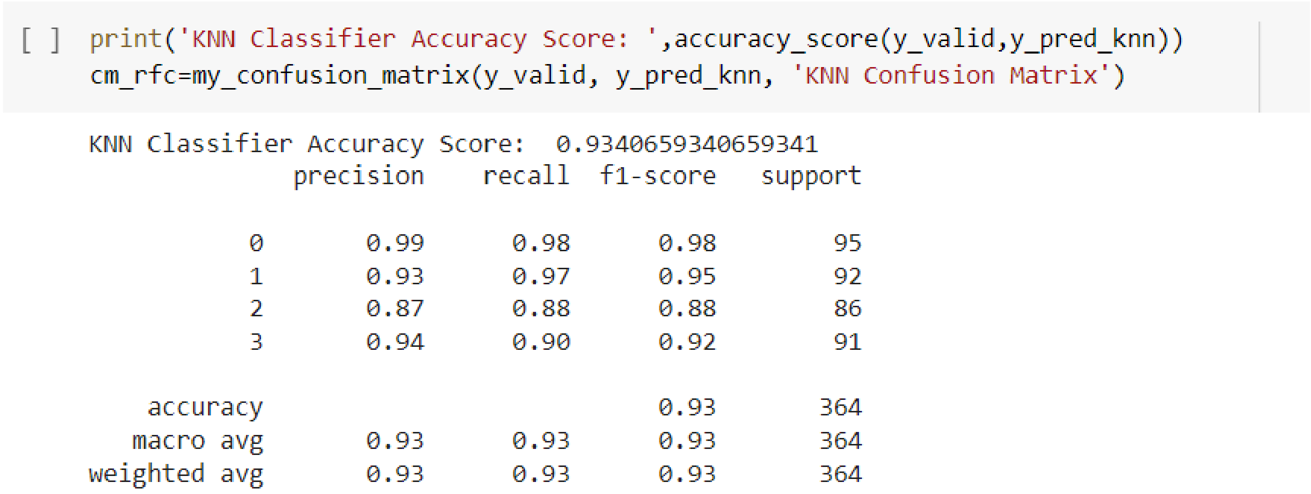


```
[ ]  from sklearn.neighbors import KNeighborsClassifier
     knn = KNeighborsClassifier(n_neighbors=3,leaf_size=25)
```

```
[ ]  knn.fit(X_train, y_train)
     y_pred_knn=knn.predict(X_valid)
```

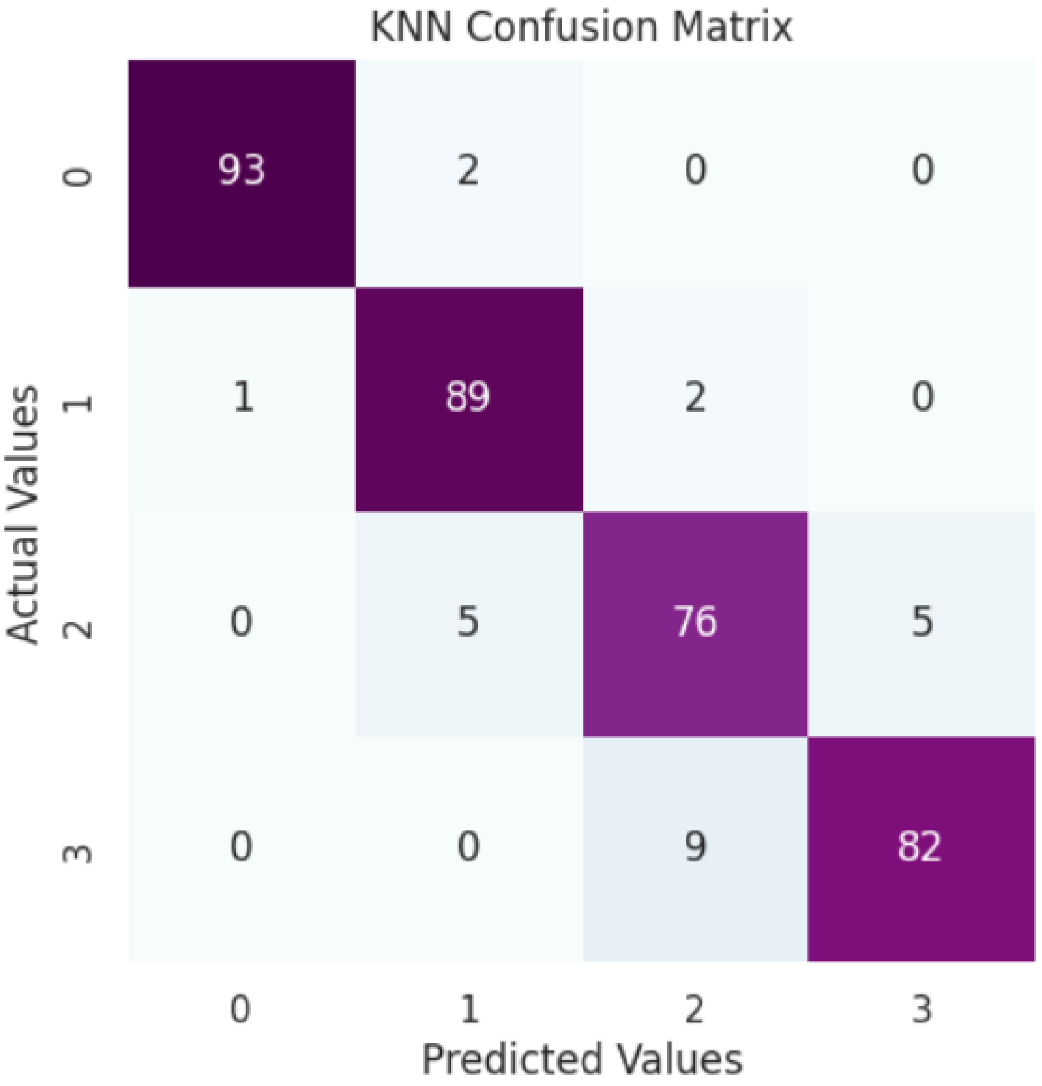
```
[ ]  y_pred_knn

     array([0, 2, 2, 3, 2, 3, 1, 2, 0, 0, 1, 2, 2, 0, 2, 1, 2, 3, 2, 3, 3, 2,
            0, 0, 1, 1, 1, 3, 3, 0, 0, 0, 3, 3, 1, 3, 3, 3, 2, 3, 2, 1, 1, 1,
            2, 1, 2, 2, 1, 3, 1, 2, 2, 2, 1, 3, 2, 2, 3, 0, 1, 0, 3, 0, 3, 2,
            0, 1, 1, 1, 2, 2, 0, 1, 2, 1, 2, 0, 2, 2, 2, 2, 1, 2, 1, 2, 3, 2,
            1, 3, 3, 1, 2, 1, 2, 0, 0, 2, 0, 1, 0, 3, 1, 3, 0, 0, 0, 0, 0, 1,
            1, 2, 2, 0, 1, 3, 0, 3, 0, 1, 3, 0, 3, 0, 3, 3, 1, 3, 3, 1, 3, 2,
            3, 1, 0, 2, 1, 1, 2, 2, 1, 3, 1, 0, 2, 2, 3, 0, 0, 2, 3, 0, 0, 3,
            0, 0, 0, 2, 1, 3, 3, 2, 3, 0, 2, 3, 1, 2, 0, 2, 3, 0, 3, 1, 3, 0,
            1, 2, 0, 0, 2, 3, 0, 3, 3, 0, 1, 1, 2, 1, 1, 0, 0, 1, 2, 2, 3, 3,
            3, 2, 3, 3, 1, 0, 1, 0, 2, 3, 1, 0, 2, 2, 2, 1, 0, 3, 3, 0, 1, 1,
            0, 2, 3, 1, 2, 3, 0, 0, 0, 0, 2, 0, 1, 3, 3, 0, 0, 3, 0, 2, 1, 1,
            3, 1, 3, 0, 0, 0, 1, 2, 1, 3, 1, 3, 1, 1, 2, 1, 2, 1, 1, 0, 1, 1,
            3, 1, 1, 1, 0, 0, 2, 2, 0, 0, 3, 3, 1, 1, 1, 1, 1, 0, 2, 2, 2, 0,
            0, 0, 3, 3, 1, 1, 1, 2, 0, 2, 2, 2, 3, 1, 3, 3, 2, 0, 1, 1, 0, 1,
            3, 0, 1, 2, 3, 3, 3, 0, 3, 2, 0, 3, 0, 2, 1, 2, 3, 0, 1, 3, 3, 0,
            3, 0, 0, 2, 0, 0, 2, 0, 2, 3, 1, 1, 3, 2, 1, 1, 0, 3, 0, 2, 3, 1,
            3, 2, 1, 1, 0, 1, 1, 2, 0, 1, 0, 0])
```

Accuracy of the KNN classifier is found to be 93.4%.

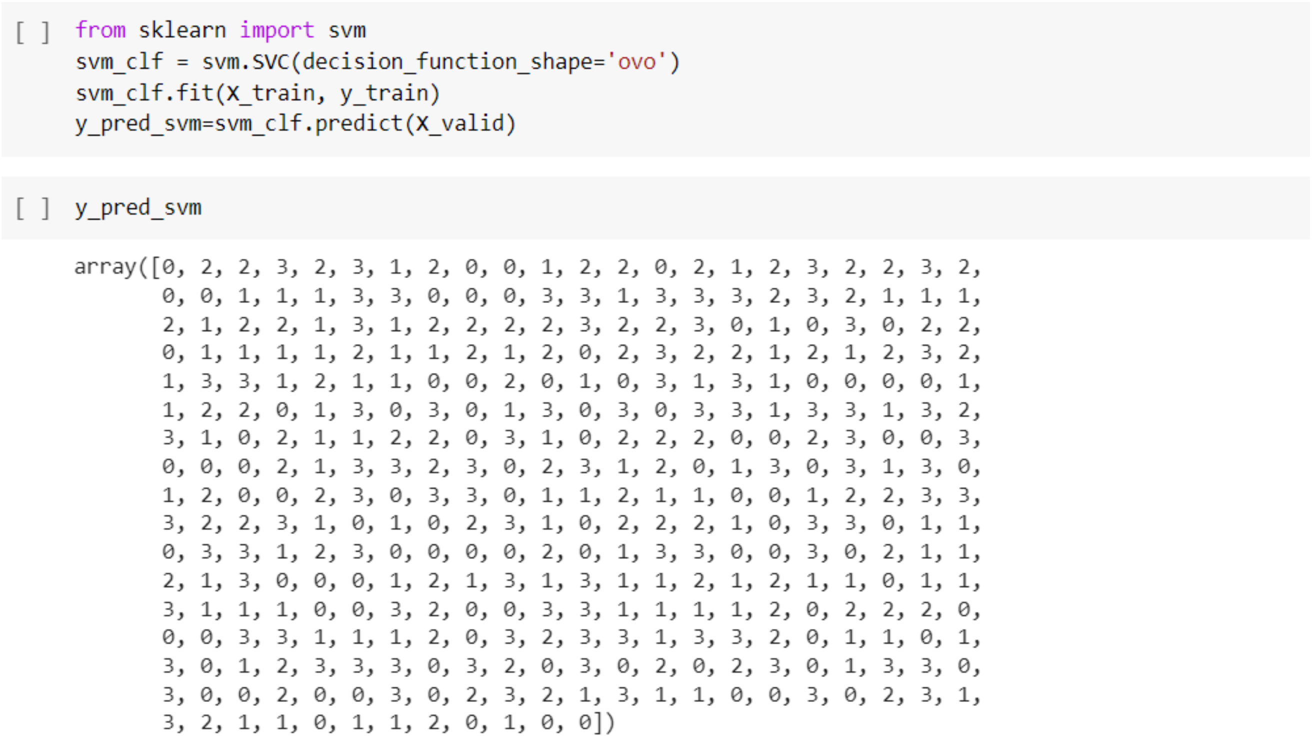


```
[ ]  print('KNN Classifier Accuracy Score: ',accuracy_score(y_valid,y_pred_knn))
     cm_rfc=my_confusion_matrix(y_valid, y_pred_knn, 'KNN Confusion Matrix')

     KNN Classifier Accuracy Score:  0.9340659340659341
                   precision    recall  f1-score   support

                0       0.99      0.98      0.98        95
                1       0.93      0.97      0.95        92
                2       0.87      0.88      0.88        86
                3       0.94      0.90      0.92        91

         accuracy                           0.93       364
        macro avg       0.93      0.93      0.93       364
     weighted avg       0.93      0.93      0.93       364
```

The confusion matrix below indicates that the actual and predicted values are found to be same for almost all the records, say, out of 95 records in 0 category, only 2 were misclassified as 1. The very few misclassifications indicate that the model has good accuracy.



SVM: Training data is fit into the SVM classifier model and x_valid is used to predict the classes. Y_pred_svm gives the classes into which each record in our dataset belong based on this model

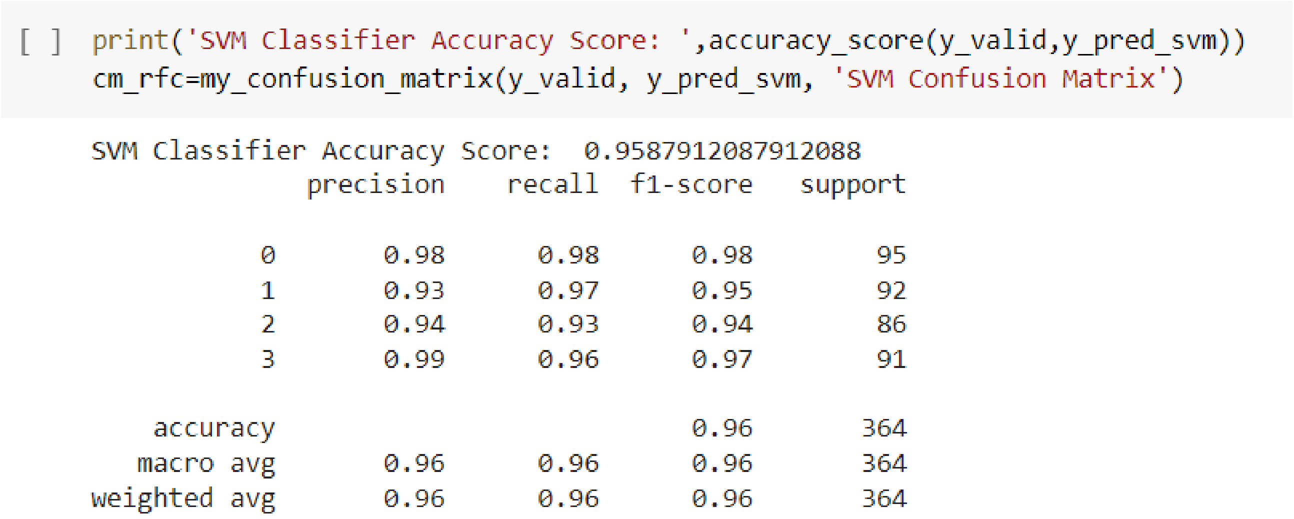


```
[ ]  from sklearn import svm
     svm_clf = svm.SVC(decision_function_shape='ovo')
     svm_clf.fit(X_train, y_train)
     y_pred_svm=svm_clf.predict(X_valid)
```

```
[ ]  y_pred_svm
```

```
     array([0, 2, 2, 3, 2, 3, 1, 2, 0, 0, 1, 2, 2, 0, 2, 1, 2, 3, 2, 2, 3, 2,
            0, 0, 1, 1, 1, 3, 3, 0, 0, 0, 3, 3, 1, 3, 3, 3, 2, 3, 2, 1, 1, 1,
            2, 1, 2, 2, 1, 3, 1, 2, 2, 2, 2, 3, 2, 2, 3, 0, 1, 0, 3, 0, 2, 2,
            0, 1, 1, 1, 1, 2, 1, 1, 2, 1, 2, 0, 2, 3, 2, 2, 1, 2, 1, 2, 3, 2,
            1, 3, 3, 1, 2, 1, 1, 0, 0, 2, 0, 1, 0, 3, 1, 3, 1, 0, 0, 0, 0, 1,
            1, 2, 2, 0, 1, 3, 0, 3, 0, 1, 3, 0, 3, 0, 3, 3, 1, 3, 3, 1, 3, 2,
            3, 1, 0, 2, 1, 1, 2, 2, 0, 3, 1, 0, 2, 2, 2, 0, 0, 2, 3, 0, 0, 3,
            0, 0, 0, 2, 1, 3, 3, 2, 3, 0, 2, 3, 1, 2, 0, 1, 3, 0, 3, 1, 3, 0,
            1, 2, 0, 0, 2, 3, 0, 3, 3, 0, 1, 1, 2, 1, 1, 0, 0, 1, 2, 2, 3, 3,
            3, 2, 2, 3, 1, 0, 1, 0, 2, 3, 1, 0, 2, 2, 2, 1, 0, 3, 3, 0, 1, 1,
            0, 3, 3, 1, 2, 3, 0, 0, 0, 0, 2, 0, 1, 3, 3, 0, 0, 3, 0, 2, 1, 1,
            2, 1, 3, 0, 0, 0, 1, 2, 1, 3, 1, 3, 1, 1, 2, 1, 2, 1, 1, 0, 1, 1,
            3, 1, 1, 1, 0, 0, 3, 2, 0, 0, 3, 3, 1, 1, 1, 1, 2, 0, 2, 2, 2, 0,
            0, 0, 3, 3, 1, 1, 1, 2, 0, 3, 2, 3, 3, 1, 3, 3, 2, 0, 1, 1, 0, 1,
            3, 0, 1, 2, 3, 3, 3, 0, 3, 2, 0, 3, 0, 2, 0, 2, 3, 0, 1, 3, 3, 0,
            3, 0, 0, 2, 0, 0, 3, 0, 2, 3, 2, 1, 3, 1, 1, 0, 0, 3, 0, 2, 3, 1,
            3, 2, 1, 1, 0, 1, 1, 2, 0, 1, 0, 0])
```

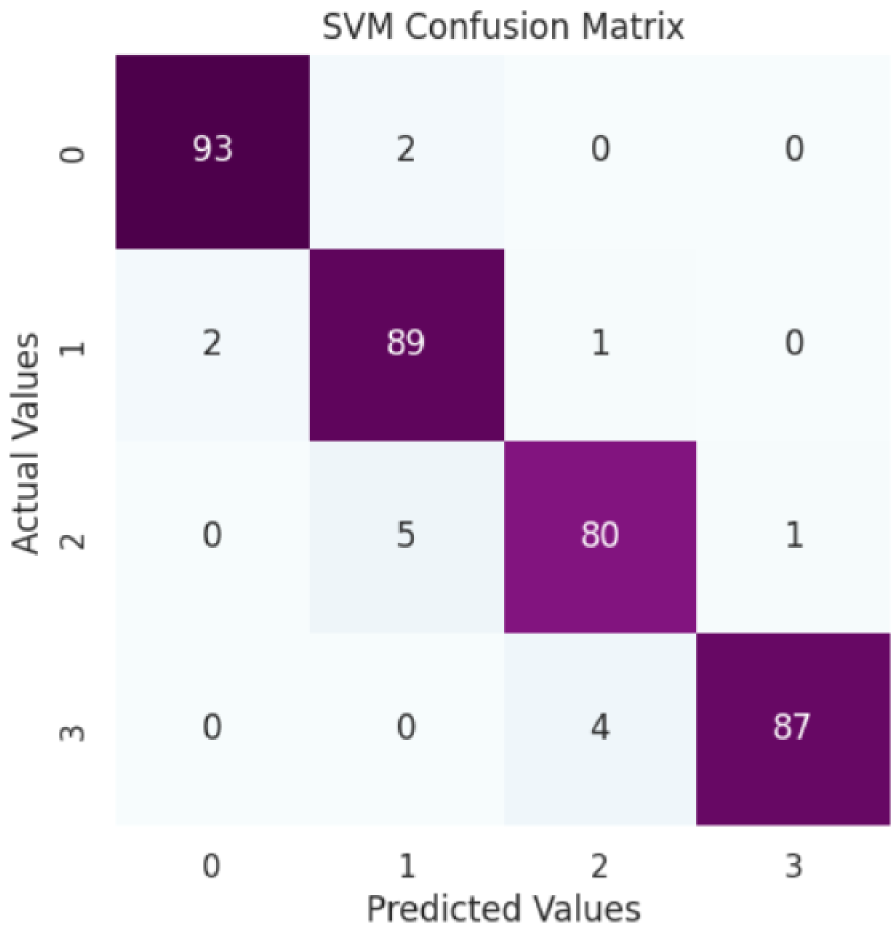
Accuracy of this SVM classifier is found to be 95.87%.



```
[ ]  print('SVM Classifier Accuracy Score: ',accuracy_score(y_valid,y_pred_svm))
     cm_rfc=my_confusion_matrix(y_valid, y_pred_svm, 'SVM Confusion Matrix')
```

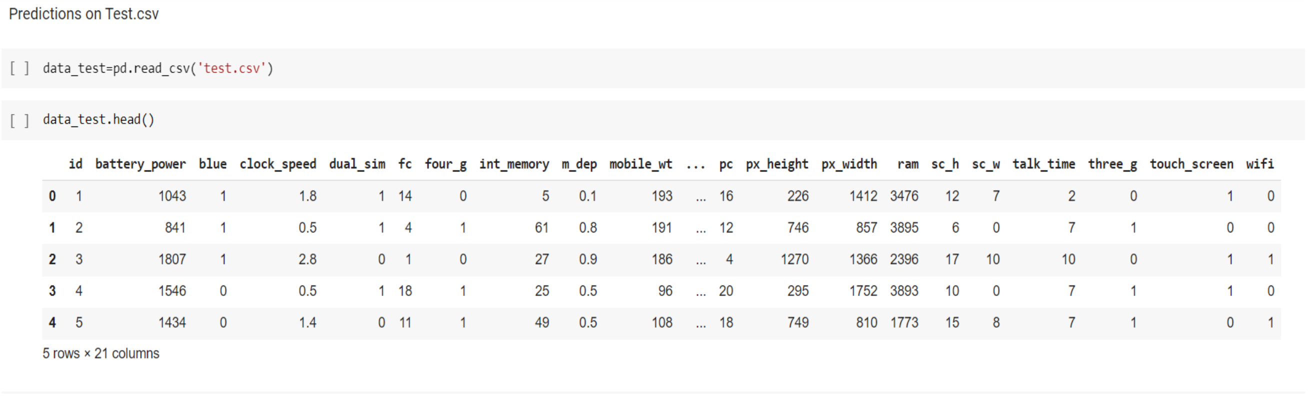
```
SVM Classifier Accuracy Score:  0.9587912087912088
              precision    recall  f1-score   support

           0       0.98      0.98      0.98        95
           1       0.93      0.97      0.95        92
           2       0.94      0.93      0.94        86
           3       0.99      0.96      0.97        91

    accuracy                           0.96       364
   macro avg       0.96      0.96      0.96       364
weighted avg       0.96      0.96      0.96       364
```

The confusion matrix below indicates that the actual and predicted values are found to be same for almost all the records, say, out of 95 records in 0 category, only 2 were misclassified as 1. The very few misclassifications indicate that the model has good accuracy



	0	1	2	3
0	93	2	0	0
1	2	89	1	0
2	0	5	80	1
3	0	0	4	87

The test.csv dataset is used to make predictions of the price range category using the above trained models. It consists of all the attributes in train.csv except the ‘price_range’ column.


Predictions on Test.csv

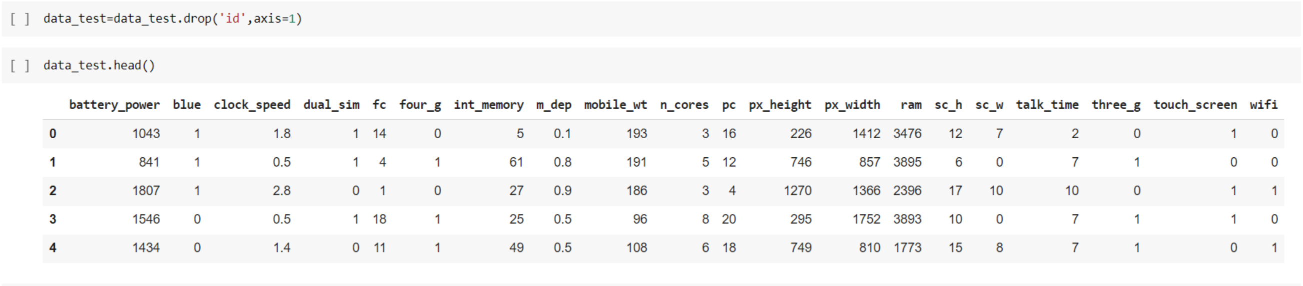
```
[ ] data_test=pd.read_csv('test.csv')
```

```
[ ] data_test.head()
```

	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

5 rows × 21 columns

The ‘id’ column is dropped as it is not relevant to the prediction.

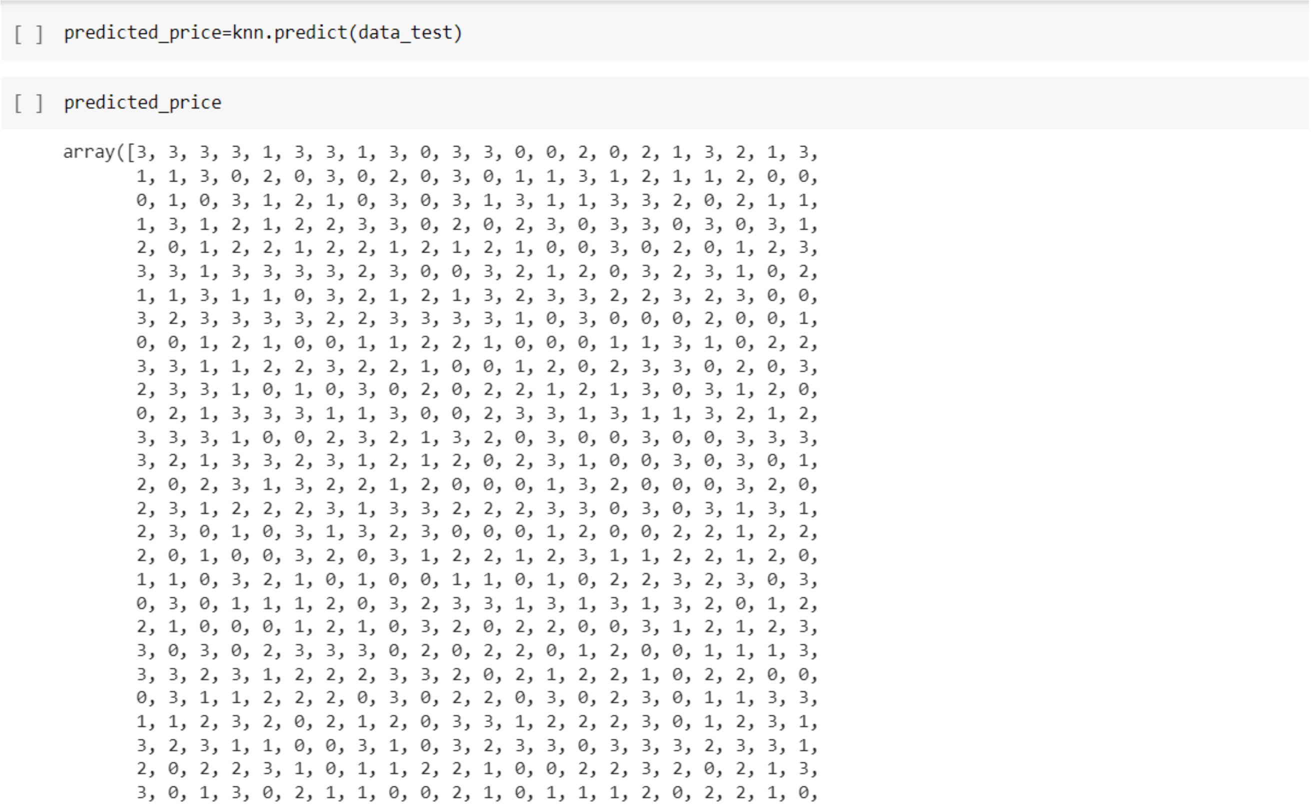


```
[ ] data_test=data_test.drop('id',axis=1)
```

```
[ ] data_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

The KNN model trained before is used to make predictions on the dataset. ‘predicted_price’ predicts the price range category of the phones present in the test.csv dataset.

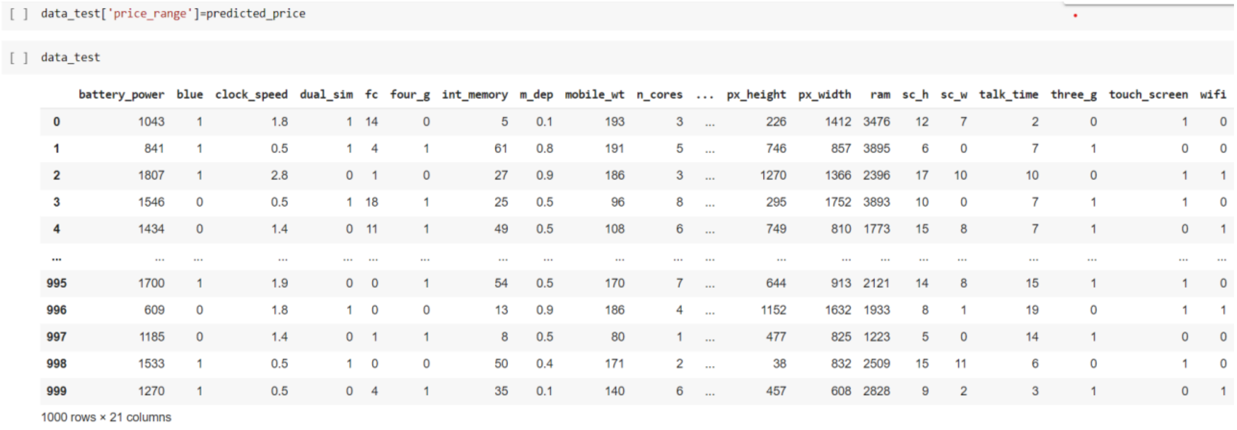


```
[ ]  predicted_price=knn.predict(data_test)
```

```
[ ]  predicted_price
```

```
array([3, 3, 3, 3, 1, 3, 3, 1, 3, 0, 3, 3, 0, 0, 2, 0, 2, 1, 3, 2, 1, 3,
       1, 1, 3, 0, 2, 0, 3, 0, 2, 0, 3, 0, 1, 1, 3, 1, 2, 1, 1, 2, 0, 0,
       0, 1, 0, 3, 1, 2, 1, 0, 3, 0, 3, 1, 3, 1, 1, 3, 3, 2, 0, 2, 1, 1,
       1, 3, 1, 2, 1, 2, 2, 3, 3, 0, 2, 0, 2, 3, 0, 3, 3, 0, 3, 0, 3, 1,
       2, 0, 1, 2, 2, 1, 2, 2, 1, 2, 1, 2, 1, 0, 0, 3, 0, 2, 0, 1, 2, 3,
       3, 3, 1, 3, 3, 3, 3, 2, 3, 0, 0, 3, 2, 1, 2, 0, 3, 2, 3, 1, 0, 2,
       1, 1, 3, 1, 1, 0, 3, 2, 1, 2, 1, 3, 2, 3, 3, 2, 2, 3, 2, 3, 0, 0,
       3, 2, 3, 3, 3, 3, 2, 2, 3, 3, 3, 3, 1, 0, 3, 0, 0, 0, 2, 0, 0, 1,
       0, 0, 1, 2, 1, 0, 0, 1, 1, 2, 2, 1, 0, 0, 0, 1, 1, 3, 1, 0, 2, 2,
       3, 3, 1, 1, 2, 2, 3, 2, 2, 1, 0, 0, 1, 2, 0, 2, 3, 3, 0, 2, 0, 3,
       2, 3, 3, 1, 0, 1, 0, 3, 0, 2, 0, 2, 2, 1, 2, 1, 3, 0, 3, 1, 2, 0,
       0, 2, 1, 3, 3, 3, 1, 1, 3, 0, 0, 2, 3, 3, 1, 3, 1, 1, 3, 2, 1, 2,
       3, 3, 3, 1, 0, 0, 2, 3, 2, 1, 3, 2, 0, 3, 0, 0, 3, 0, 0, 3, 3, 3,
       3, 2, 1, 3, 3, 2, 3, 1, 2, 1, 2, 0, 2, 3, 1, 0, 0, 3, 0, 3, 0, 1,
       2, 0, 2, 3, 1, 3, 2, 2, 1, 2, 0, 0, 0, 1, 3, 2, 0, 0, 0, 3, 2, 0,
       2, 3, 1, 2, 2, 2, 3, 1, 3, 3, 2, 2, 2, 3, 3, 0, 3, 0, 3, 1, 3, 1,
       2, 3, 0, 1, 0, 3, 1, 3, 2, 3, 0, 0, 0, 1, 2, 0, 0, 2, 2, 1, 2, 2,
       2, 0, 1, 0, 0, 3, 2, 0, 3, 1, 2, 2, 1, 2, 3, 1, 1, 2, 2, 1, 2, 0,
       1, 1, 0, 3, 2, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 2, 2, 3, 2, 3, 0, 3,
       0, 3, 0, 1, 1, 1, 2, 0, 3, 2, 3, 3, 1, 3, 1, 3, 1, 3, 2, 0, 1, 2,
       2, 1, 0, 0, 0, 1, 2, 1, 0, 3, 2, 0, 2, 2, 0, 0, 3, 1, 2, 1, 2, 3,
       3, 0, 3, 0, 2, 3, 3, 3, 0, 2, 0, 2, 2, 0, 1, 2, 0, 0, 1, 1, 1, 3,
       3, 3, 2, 3, 1, 2, 2, 2, 3, 3, 2, 0, 2, 1, 2, 2, 1, 0, 2, 2, 0, 0,
       0, 3, 1, 1, 2, 2, 2, 0, 3, 0, 2, 2, 0, 3, 0, 2, 3, 0, 1, 1, 3, 3,
       1, 1, 2, 3, 2, 0, 2, 1, 2, 0, 3, 3, 1, 2, 2, 2, 3, 0, 1, 2, 3, 1,
       3, 2, 3, 1, 1, 0, 0, 3, 1, 0, 3, 2, 3, 3, 0, 3, 3, 3, 2, 3, 3, 1,
       2, 0, 2, 2, 3, 1, 0, 1, 1, 2, 2, 1, 0, 0, 2, 2, 3, 2, 0, 2, 1, 3,
       3, 0, 1, 3, 0, 2, 1, 1, 0, 0, 2, 1, 0, 1, 1, 1, 2, 0, 2, 2, 1, 0,
```

This new column of price_range is added to the test.csv data.



```
[ ] data_test['price_range']=predicted_price
```

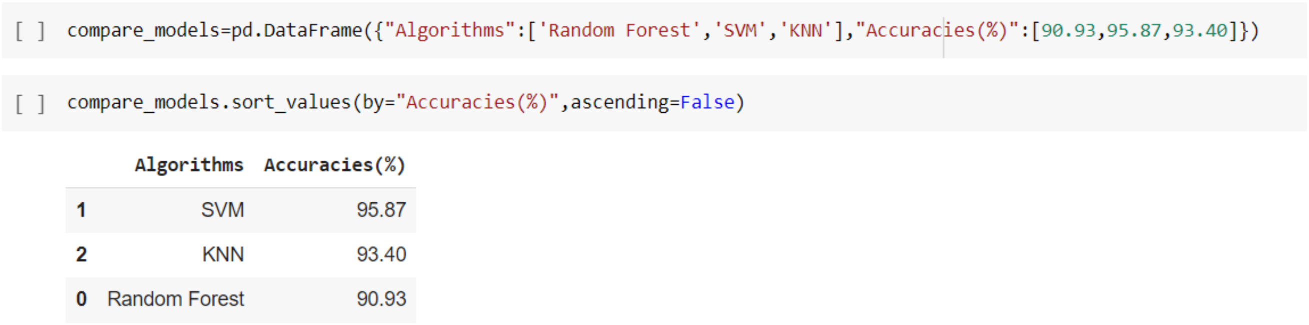
```
[ ] data_test
```

	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
0	1043	1	1.8	1	14	0	5	0.1	193	3	...	226	1412	3476	12	7	2	0	1	0
1	841	1	0.5	1	4	1	61	0.8	191	5	...	746	857	3895	6	0	7	1	0	0
2	1807	1	2.8	0	1	0	27	0.9	186	3	...	1270	1366	2396	17	10	10	0	1	1
3	1546	0	0.5	1	18	1	25	0.5	96	8	...	295	1752	3893	10	0	7	1	1	0
4	1434	0	1.4	0	11	1	49	0.5	108	6	...	749	810	1773	15	8	7	1	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
995	1700	1	1.9	0	0	1	54	0.5	170	7	...	644	913	2121	14	8	15	1	1	0
996	609	0	1.8	1	0	0	13	0.9	186	4	...	1152	1632	1933	8	1	19	0	1	1
997	1185	0	1.4	0	1	1	8	0.5	80	1	...	477	825	1223	5	0	14	1	0	0
998	1533	1	0.5	1	0	0	50	0.4	171	2	...	38	832	2509	15	11	6	0	1	0
999	1270	1	0.5	0	4	1	35	0.1	140	6	...	457	608	2828	9	2	3	1	0	1

1000 rows × 21 columns

Results

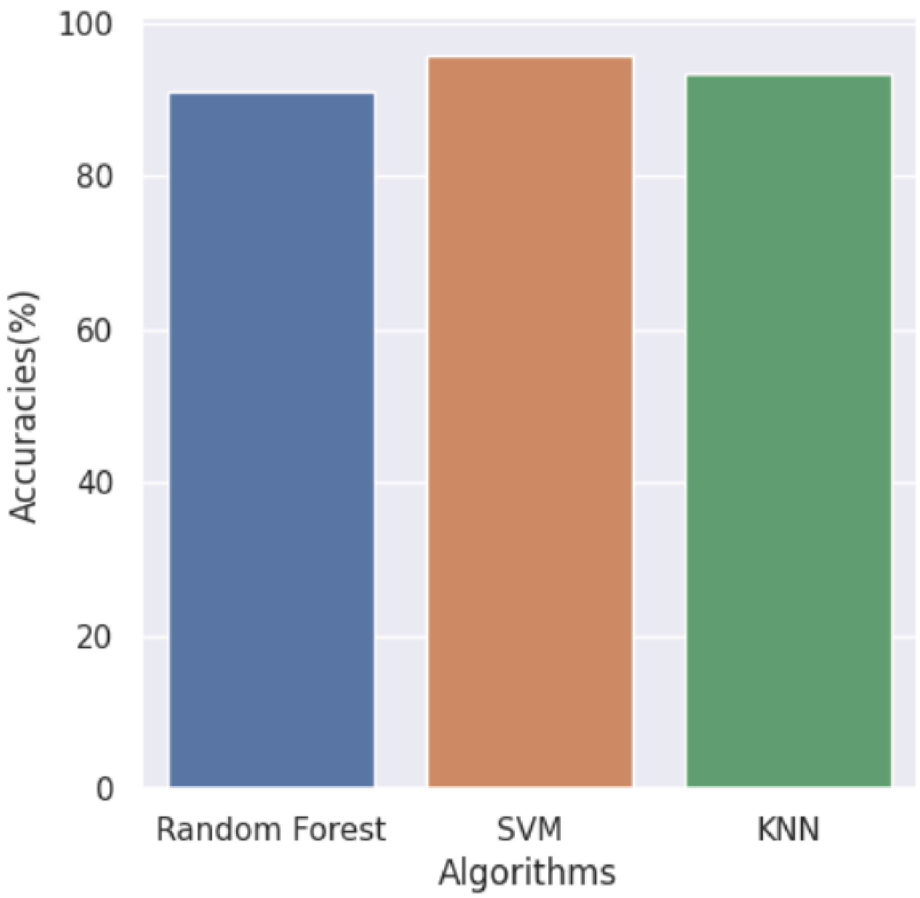
The different algorithms applied on the train.csv dataset are compared. Their names and accuracies and made into a data frame and plotted to find the highest accuracy model. Here, SVM gives the highest accuracy of 95.87% compared to random forest and KNN. This concludes that for the given mobile price dataset, SVM classifier works best to provide accurate classifications.



```
[ ]  compare_models=pd.DataFrame({"Algorithms":['Random Forest','SVM','KNN'],"Accuracies(%)":[90.93,95.87,93.40]})
```

```
[ ]  compare_models.sort_values(by="Accuracies(%)",ascending=False)
```

	Algorithms	Accuracies(%)
1	SVM	95.87
2	KNN	93.40
0	Random Forest	90.93



Conclusion

The mobile price prediction project, in conclusion, is a useful tool that may help estimate the cost of a mobile device based on a variety of criteria like brand, specifications, and market trends. The project has the ability to deliver accurate mobile phone pricing predictions using machine learning algorithms and historical data, which can be helpful for buyers, dealers, and other mobile industry stakeholders. More data can be incorporated, algorithms can be improved, and conclusions can be verified with real-world data. Overall, the research to anticipate mobile prices has shown encouraging results and has the potential to be a useful tool for mobile buyers and sellers in making decisions.

Future works

Our mobile price prediction project will evolve in the future with the addition of innovative machine learning models, improved feature engineering, real-time data integration, NLP approaches, an updated user interface, mobile price comparison, and price alerts. Our forecasts will become more accurate and user-friendly as a result of these enhancements, enabling users to choose mobile purchases wisely.

References

- https://www.nber.org/papers/w25645
- https://ieeexplore.ieee.org/document/9604550
- https://www.researchgate.net/publication/323994340_Mobile_Price_Class_prediction_using_Machine_Learning_Techniques
- https://ijisrt.com/assets/upload/files/IJISRT22JAN380.pdf