Smart Mobile Phone Price Prediction Using Machine Learning

K. Mahesh
Department of Emerging Technologies
Malla Reddy College of Engineering
&
Technology
Hyderabad, Telangana
20N31A6732@mrcet.ac.in

Submission Date:14-07-2023

***Abstract*: The objective of this study is to develop a predictive model that can estimate the price of a mobile phone based on its features and specifications. Dataset is collected from the website www.kaggle.com. Different algorithms are used to identify and remove irrelevant and redundant features and have minimum computational complexity. Different classifiers are used to achieve as higher accuracy as possible. Results are compared in terms of highest accuracy achieved and minimum features selected. Conclusion is made on the basic of feature selection algorithm and classifier for the given dataset. This work can be used in any type of marketing and business to find optimal price of the product. This work can also assist consumers, retailers, and manufacturers in making informed decisions related to pricing, purchasing, and market strategies.**

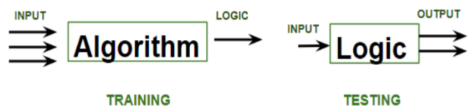
Keywords:- Machine Learning, Prediction, feature selection, Logistic Regression, Decision Tree, KNN and Random Forest Algorithms.

I. INTRODUCTION

Mobile phone price prediction is of significant importance in the dynamic mobile phone industry, where accurate price estimation is crucial for manufacturers, retailers, and consumers. This research aims to propose a machine learning-based model that leverages diverse features, such as phone specifications, customer reviews, market trends, and economic indicators, to accurately forecast mobile phone prices. The proposed model has the potential to enhance pricing strategies for manufacturers, enable data-driven pricing decisions for retailers, and assist consumers in making informed purchase choices. By exploring various machine learning algorithms and conducting comprehensive evaluations, this research seeks to provide valuable insights and contribute to the field of mobile phone price prediction.
To choose just the best characteristics and reduce the dataset, many types of algorithms are required.
The computational complexity of the issue will be reduced as a result. Because this is an optimization issue, a variety of optimization techniques are frequently employed to lower the datasets dimensionality. Mobile is currently one of the most popular apps for sales and transactions. Every day, new mobile phones with new versions and additional apps are introduced. Every day, hundreds of thousands of cell phones are sold and purchased. As a result, the prediction of the mobile pricing class is a case study for the given issue type, namely, identifying the best product. The same method may be used to determine the true cost of any item, including cars, motorbikes, generators, motors, food, medication, and so on. Mobile Processor, for example, is one of the most essential programmes for calculating mobile costs. The time of batteries is also very important in today's hectic human existence. Size and thickness of the mobile device are other essential factors to consider when making a selection. Internal memory, camera pixels, and video consistency must all be recalled. Internet browsing is also one of the most important technical constraints of the twenty-first century. Also, the list of various features is determined by the size of the mobile device. As a result, we'll utilise all of the aforementioned characteristics to decide if the smartphone will be very-costly, economical, pricey, or very-costly.

II. RESEARCH METHODOLOGY

The research was carried out in Google Colab's Python kernel. The general workflow diagram of supervised ML tasks is as follows:



The dataset is portioned into two – train for training the model and test for its evaluation. The computer tries to comprehend the logic behind the pricing of a mobile based on its features and uses it to forecast future instances as correctly as possible.

III. UNDERSTANDING THE DATASET

The Mobile Price Class dataset sourced from the Kaggle data science community website

(https://www.kaggle.com/code/vikramb/mobile-price-prediction/input?select=train.csv)

The dataset contains 21 attributes in total – 20 features and a class label which is the price range. The features include battery capacity, RAM, weight, camera pixels, etc. The class label is the price range. It has 4 kinds of values – 0,1,2 and 3 which are of ordinal data type representing the increasing degree of price. Higher the value, higher is the price range the mobile falls under. These 4 values can be interpreted as economical, mid-range, flagship and premium.

So, despite price traditionally being a numeric problem, the type of ML is classification (not regression) since there are discrete values in the class label. This is advantageous when using algorithms like Naïve Bayes and Decision Tree as they normally don't work well with numeric data.

```
[2] #importing and loading datasets
    train_data=pd.read_csv("train.csv")
    test_data=pd.read_csv("test.csv")

[3] train_data
```

	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	price_range
0	842	0	2.2	0	1	0	7	0.6	188	2	2	20	756	2549	9	7	1
1	1021	1	0.5	1	0	1	53	0.7	136	3	6	905	1988	2631	17	3	2
2	563	1	0.5	1	2	1	41	0.9	145	5	6	1263	1716	2603	11	2	2
3	615	1	2.5	0	0	0	10	0.8	131	6	9	1216	1786	2769	16	8	2
4	1821	1	1.2	0	13	1	44	0.6	141	2	14	1208	1212	1411	8	2	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1995	794	1	0.5	1	0	1	2	0.8	106	6	14	1222	1890	668	13	4	0
1996	1965	1	2.6	1	0	0	39	0.2	187	4	3	915	1965	2032	11	10	2
1997	1911	0	0.9	1	1	1	36	0.7	108	8	3	868	1632	3057	9	1	3
1998	1512	0	0.9	0	4	1	46	0.1	145	5	5	336	670	869	18	10	0
1999	510	1	2.0	1	5	1	45	0.9	168	6	16	483	754	3919	19	4	3

2000 rows × 17 columns

Train dataset of mobile price prediction

Exploratory Data Analysis

```
[5] train_data.shape
(2000, 17)

test_data.shape
(1000, 17)
```

train_data.describe()

```
train_data.describe()
```

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	pc	px_height	px_width	ram
count	2000.000000	2000.0000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000
mean	1238.518500	0.4950	1.522250	0.509500	4.309500	0.521500	32.046500	0.501750	140.249000	4.520500	9.916500	645.108000	1251.515500	2124.213000
std	439.418206	0.5001	0.816004	0.500035	4.341444	0.499662	18.145715	0.288416	35.399655	2.287837	6.064315	443.780811	432.199447	1084.732044
min	501.000000	0.0000	0.500000	0.000000	0.000000	0.000000	2.000000	0.100000	80.000000	1.000000	0.000000	0.000000	500.000000	256.000000
25%	851.750000	0.0000	0.700000	0.000000	1.000000	0.000000	16.000000	0.200000	109.000000	3.000000	5.000000	282.750000	874.750000	1207.500000
50%	1226.000000	0.0000	1.500000	1.000000	3.000000	1.000000	32.000000	0.500000	141.000000	4.000000	10.000000	564.000000	1247.000000	2146.500000
75%	1615.250000	1.0000	2.200000	1.000000	7.000000	1.000000	48.000000	0.800000	170.000000	7.000000	15.000000	947.250000	1633.000000	3064.500000
max	1998.000000	1.0000	3.000000	1.000000	19.000000	1.000000	64.000000	1.000000	200.000000	8.000000	20.000000	1960.000000	1998.000000	3998.000000

test_data.describe()

```
test_data.describe()
```

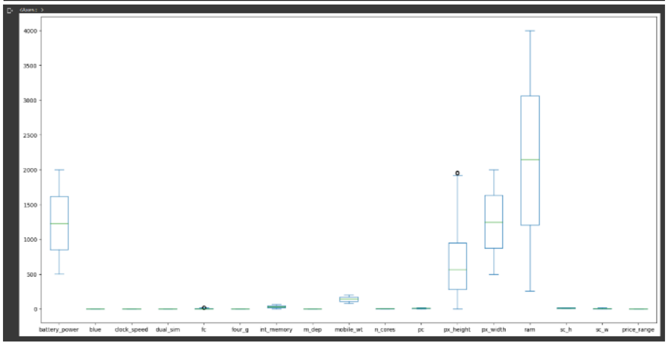
	id	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	pc	px_height	px_width
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.00000	1000.000000	1000.000000	1000.000000	1000.000000
mean	500.500000	1248.510000	0.516000	1.540900	0.517000	4.593000	0.487000	33.652000	0.517500	139.51100	4.328000	10.054000	627.121000	1239.774000
std	288.819436	432.458227	0.499994	0.829268	0.499961	4.463325	0.500081	18.128694	0.280861	34.85155	2.288155	6.095099	432.929699	439.670981
min	1.000000	500.000000	0.000000	0.500000	0.000000	0.000000	0.000000	2.000000	0.100000	80.00000	1.000000	0.000000	0.000000	501.000000
25%	250.750000	895.000000	0.000000	0.700000	0.000000	1.000000	0.000000	18.000000	0.300000	109.75000	2.000000	5.000000	263.750000	831.750000
50%	500.500000	1246.500000	1.000000	1.500000	1.000000	3.000000	0.000000	34.500000	0.500000	139.00000	4.000000	10.000000	564.500000	1250.000000
75%	750.250000	1629.250000	1.000000	2.300000	1.000000	7.000000	1.000000	49.000000	0.800000	170.00000	6.000000	16.000000	903.000000	1637.750000
max	1000.000000	1999.000000	1.000000	3.000000	1.000000	18.000000	1.000000	64.000000	1.000000	200.00000	8.000000	20.000000	1907.000000	1998.000000

Comparing train and test datasets

```
[13] #compare train and test dataset
     compare= {'features':['battery_power', 'clock_speed', 'fc', 'int_memory', 'mobile_wt', 'pc', 'px_height', 'px_width', 'ram', 'sc_h', 'sc_w'],
          'train_mean':[train_data.battery_power.mean(),train_data.clock_speed.mean(), train_data.fc.mean()
                       ,train_data.int_memory.mean(),train_data.mobile_wt.mean(),train_data.pc.mean(),
                       train_data.px_height.mean(),train_data.px_width.mean(),train_data.ram.mean(),train_data.sc_h.mean(),
                       train_data.sc_w.mean()],
          'test_mean': [test_data.battery_power.mean(),test_data.clock_speed.mean(), test_data.fc.mean()
                       ,test_data.int_memory.mean(),test_data.mobile_wt.mean(),test_data.pc.mean(),
                       test_data.px_height.mean(),test_data.px_width.mean(),test_data.ram.mean(),test_data.sc_h.mean(),
                       test_data.sc_w.mean()]}
     comparison = pd.DataFrame(compare)
     comparison
```

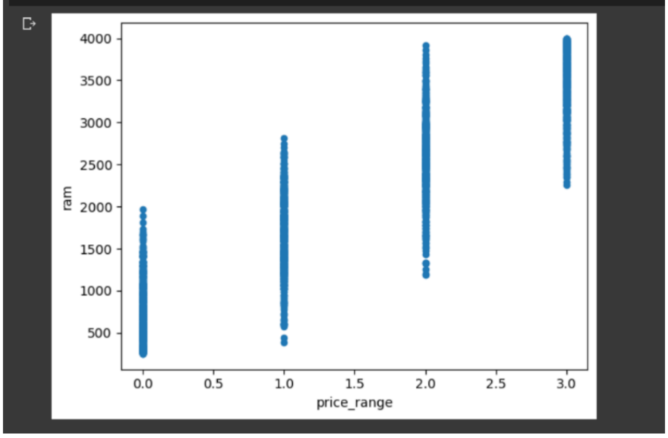
	features	train_mean	test_mean
0	battery_power	1238.51850	1248.5100
1	clock_speed	1.52225	1.5409
2	fc	4.30950	4.5930
3	int_memory	32.04650	33.6520
4	mobile_wt	140.24900	139.5110
5	pc	9.91650	10.0540
6	px_height	645.10800	627.1210
7	px_width	1251.51550	1239.7740
8	ram	2124.21300	2138.9980
9	sc_h	12.30650	11.9950
10	sc_w	5.76700	5.3160

```
[14] #visualization categorical and numerical data
     train_data.plot(kind='box',figsize=(20,10))
```



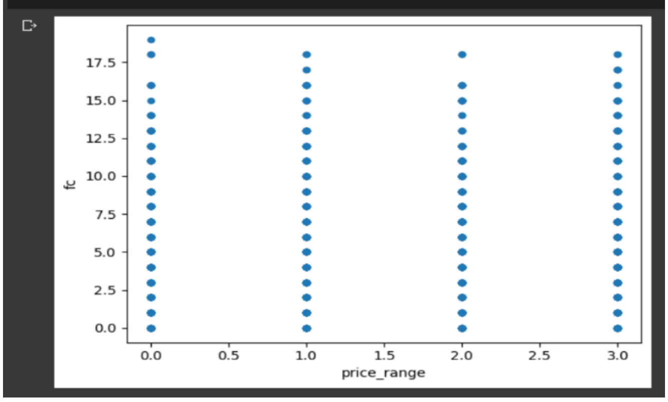
Price range vs RAM

```
train_data.plot(x='price_range',y='ram',kind='scatter')
plt.show()
```



Price range vs front camera

```
train_data.plot(x='price_range',y='fc',kind='scatter')
plt.show()
```



IV. TRAINING THE PREDICTION MODEL

The first step in creating a model is to extract the required features for training from the dataset and assigning the parameter that is to be the class label.

```
X=train_data.drop('price_range',axis=1)
Y=train_data['price_range']
```

In this code snippet, the first 20 attributes are being extracted to serve as the training parameters and the final attribute (price_range) is used as the class label.

```
[48] from sklearn.model_selection import train_test_split
     from sklearn.metrics import mean_squared_error
     from sklearn.metrics import mean_absolute_error
     from sklearn.linear_model import LinearRegression
     from sklearn.model_selection import train_test_split
     X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,random_state=0)
```

The data is then portioned into two for the purpose of training the model and testing it.

1) Decision Tree Algorithm

```
[70] from sklearn.tree import DecisionTreeClassifier
     DT=DecisionTreeClassifier()
```

```
[71] DT.fit(X_train,Y_train)
     Y_pred=DT.predict(X_test)
     Y_pred
```

Decision Tree was used here to train the prediction model.

2) KNN Algorithm

```
[59] #We need to scale the data using Standard Scaler because of distance matter here
```

```
[60] from sklearn.preprocessing import StandardScaler
     std=StandardScaler()
     X_train_std=std.fit_transform(X_train)
     X_test_std=std.transform(X_test)
     X_test_std
```

```
[61] from sklearn.neighbors import KNeighborsClassifier
     knn=KNeighborsClassifier()
     knn.fit(X_train_std,Y_train)
     KNeighborsClassifier()
     Y_pred=knn.predict(X_test_std)
     Y_pred
```

KNN was used to train the model here

3) Logistic Regression

```
[63] from sklearn.linear_model import LogisticRegression
     lr=LogisticRegression()
     lr.fit(X_train_std,Y_train)
     LogisticRegression()
     Y_pred=lr.predict(X_test_std)
     Y_pred
```

Logistic Regression was used to train the model here

4) Random Forest Algorithm

```
[65] from sklearn.ensemble import RandomForestClassifier
     rfc = RandomForestClassifier()
     rfc.fit(X_train, Y_train)
     y_pred = rfc.predict(X_test)
     accuracy = accuracy_score(Y_test, y_pred)
     print("Random Forest Accuracy:", accuracy)
```

Random Forest was used to train the model here

V. RESULTS AND DISCUSSION

Metrics used to evaluate the algorithms in this paper are classification report and accuracy score. Accuracy score gives the accuracy of the trained model after evaluating it using test data, for which we have sampled 20% of the dataset

```
[58] from sklearn.metrics import accuracy_score
     DT_ACCRY=accuracy_score(Y_test,Y_pred)
     print("Decision Tree Accuracy:",DT_ACCRY)

     Decision Tree Accuracy: 0.845
```

Decision Tree was found to be able to correctly forecast the classes with a certainty of 84.5%.

```
[62] KNN_ACCRY=accuracy_score(Y_test,Y_pred)
     print("KNearestNeighbors Accuracy:",KNN_ACCRY)

     KNearestNeighbors Accuracy: 0.565
```

KNN is a poor classifier when working with numeric data as input.

```
[64] LR_ACCRY=accuracy_score(Y_test,Y_pred)
     print("Logistic regression Accuracy:",LR_ACCRY)

     Logistic regression Accuracy: 0.955
```

The certainty of Logistic Regression is found to be 95.5%.

```
[65] from sklearn.ensemble import RandomForestClassifier
     rfc = RandomForestClassifier()
     rfc.fit(X_train, Y_train)
     y_pred = rfc.predict(X_test)
     accuracy = accuracy_score(Y_test, y_pred)
     print("Random Forest Accuracy:", accuracy)

     Random Forest Accuracy: 0.87
```

A veracity of 87% was achieved using Random Forest.

The algorithm that is found to be able to classify instances the most accurately among the ones tested is Logistic Regression with an accuracy of 95.5%, followed closely by Random Forest that was able to predict instances with an accuracy of 87% and DT with an accuracy of 84%. The KNN classifier failed to forecast the price range optimally

Real Time Analysis

```
[44] index=int(input("Enter any index position to predict the price for the features: "))
     print("\n")
     if index < len(selected_features):
         row = selected_features.iloc[index]
         print(dict(row),"\n")
         input_row = pd.DataFrame([row])
         prediction = dt.predict(input_row)
         pre = prediction[0]*10000
         print('Predicted price range of the Mobile is:',pre-2500,'-',pre+2500)
     else:
       print("Index value sholud be in between 0 to 2000")

     Enter any index position to predict the price for the features: 10

     {'battery_power': 769, 'blue': 1, 'fc': 0, 'four_g': 0, 'int_memory': 9, 'mobile_wt': 182, 'pc': 1, 'ram': 3946}

     Predicted price range of the Mobile is: 27500 - 32500
```

```
[45] # Creating a new dataset frame for the predicted price range data.
     predicted_price=[pre-2500,pre+2500]
     predicted_df = pd.DataFrame({'predicted_price_range':predicted_price})
     predicted_df
```

	predicted_price_range
0	27500
1	32500

VI. CONCLUSION

The model trained using Logistic Regression was found to predict mobile price classes most accurately (95.5%). The accuracy of the models can be improved by doing some data preprocessing steps like normalization and standardization. Feature selection and extraction algorithms can be used to remove unsuitable and duplicative features to get better results. The same procedure used in this paper can be applied to predict the prices of other products like cars, bikes, houses, etc. using the archival data containing features like cost, specifications, etc. This would help organizations and consumers alike to make more educated decisions when it comes to price.

VII. REFERENCES

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

[2] U. Arul & Dr. S. Prakash, ‘Towards Fault Handling in B2B Collaboration using Orchestration based Web Services Composition’, International Journal of Emerging Technology and Advanced Engineering (IJETAE), Vol. 3, Issue 1, pp. 388-394, 2013

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

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

[5] U. Arul & S. Prakash, ‘Toward Automatic Web Service Composition based on Multilevel Workflow Orchestration and Semantic Web Service Discovery’, International Journal of Business Information Systems, Inderscience Publishers, Vol. 34, Issue 1, pp. 128–156, April 2020.

[6] https://github.com/vikram-bhati, Classification classify mobile price range.

[7] Introduction to dimensionality reduction, A computer science portal for Geeks. https://www.geeksforgeeks.org/dimensionalityreduction

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