ISSN No:-2456-2165

Prediction of Mobile Phone Price Class using Supervised Machine Learning Techniques

A Varun Kiran and Dr. Jebakumar R.
SRM Institute of Science and Technology, Kattankulathur Campus,
Chennai – 603203, Tamil Nadu, India.

Abstract:- The aim of this research is to develop a model to predict the price of a mobile when the specifications of a mobile are given and to find the ML algorithm that predicts the price most accurately. The usage of archival data to accurately forecast forthcoming instances is the essence of Predictive Analytics. One of the ways Predictive Analytics can be performed is by using Machine Learning. Predictive Machine Learning works by taking in data as input to develop and train a prediction model and the trained model is used to predict the outcome of future data instances. Supervised Machine Learning algorithms make use of data that contains a pre-defined class label, which is the attribute that needs to be predicted. The class label is the price of a mobile in our case. The Mobile Price Class dataset sourced from the Kaggle data science community website (https://www.kaggle.com/iabhishekofficial/mobile-price-classification) that categorizes mobiles into price ranges was used to train the prediction model. Python is used due to its readily accessible ML libraries. Various classification algorithms were used to train the model to try and find the algorithm that is able to predict the mobile price class most accurately. Metrics like accuracy score, confusion matrix, etc. are were used to evaluate the trained model to determine the algorithm most suitable among the ones used.

Keywords:- *Machine Learning, Predictive Analytics, Supervised Machine Learning, Python.*

I. INTRODUCTION

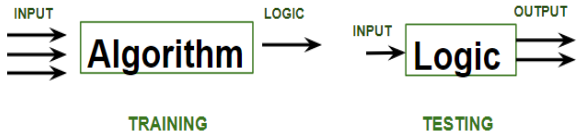
Price is the most important component in the marketing of any product and is often the definitive factor in its sale to a consumer. In a constantly evolving and volatile market, the price is often the factor that makes or breaks a product. Setting an optimal price before the release of a product is imperative for any company. A tool that gives the estimated price of a product after weighing in the features it provides can come in handy and can help the company in making an informed decision while setting the market price for a product. Such a tool can also be used by a consumer to get an estimated price based on the features they are looking for in the product.

Machine learning algorithms can perform various tasks which are to be chosen with respect to the data to be worked with and the motive of the task. Various tools and languages like Python, MATLAB, Java, WEKA, Cygwin, Octave, etc are available to perform machine learning tasks. Examples of some frequently used algorithms include Naïve Bayes, K-NN, etc. Feature selection algorithms can be used to select and extract only the best parameters to train a model to optimize the accuracy and lessen the computational time of the model. Any of these methods can be used to perform the task of predicting the price of a product depending on the type of data available to train the model.

Nowadays, a cellphone is an essential accessory of a person. It is the fastest evolving and moving product in the technology market space. New mobiles with updated versions and new features are introduced into the market at a rapid pace. Thousands of mobiles are sold each day. In such a fast-paced and volatile market, a mobile company needs to set optimal prices to compete with its rivals. The first step in fixing a price is to estimate the price based on the features. The objective this research is to develop an ML model capable of estimating the price of a mobile phone based on its features. A potential buyer can also make use of the model to estimate the price of a mobile by inputting just the features they require into the tool. The same approach to create a prediction model can be used to develop a price estimation model for most products that have similar independent variable parameters. The price of a mobile is dependent on many features for example, the processor, battery capacity, camera quality, display size and thickness, etc. These features can be used to classify phones into various categories like entry-level, mid-range, flagship, premium, etc. Supervised ML algorithms are used in this paper as the dataset used has a definitive class label for price range.

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.

ISSN No:-2456-2165

III. UNDERSTANDING THE DATASET

The Mobile Price Class dataset sourced from the Kaggle data science community website (https://www.kaggle.com/iabhishekofficial/mobile-price-classification) that categorizes mobiles into price ranges was used to train the prediction model.

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.

```
dataset = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/archive (5)/train.csv')
dataset.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	t
0	842	0	2.2	0	1	0	7	0.6	188	2	2	20	756	2549	9	7	
1	1021	1	0.5	1	0	1	53	0.7	136	3	6	905	1988	2631	17	3	
2	563	1	0.5	1	2	1	41	0.9	145	5	6	1263	1716	2603	11	2	
3	615	1	2.5	0	0	0	10	0.8	131	6	9	1216	1786	2769	16	8	
4	1821	1	1.2	0	13	1	44	0.6	141	2	14	1208	1212	1411	8	2	

The dataset contains 2000 records in total.

```
[5] dataset.columns
```

```
Index(['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'],
      dtype='object')
```

```
[7] dataset.shape
```

```
(2000, 21)
```

This is the numerical breakdown of the dataset:

```
[9] dataset.describe()
```

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_
count	2000.000000	2000.0000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.0000
mean	1238.518500	0.4950	1.522250	0.509500	4.309500	0.521500	32.046500	0.501750	140.2490
std	439.418206	0.5001	0.816004	0.500035	4.341444	0.499662	18.145715	0.288416	35.3996
min	501.000000	0.0000	0.500000	0.000000	0.000000	0.000000	2.000000	0.100000	80.0000
25%	851.750000	0.0000	0.700000	0.000000	1.000000	0.000000	16.000000	0.200000	109.0000
50%	1226.000000	0.0000	1.500000	1.000000	3.000000	1.000000	32.000000	0.500000	141.0000
75%	1615.250000	1.0000	2.200000	1.000000	7.000000	1.000000	48.000000	0.800000	170.0000
max	1998.000000	1.0000	3.000000	1.000000	19.000000	1.000000	64.000000	1.000000	200.0000

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.

```
[12] X = dataset.iloc[:,:20].values
     y = dataset.iloc[:,20:21].values
```

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.

```
[ ] 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. A test size of 0.2 implies that 80% of the data is assigned to train the prediction model and the rest is utilized to measure the quality of the developed model.

Decision Tree

```
[10] from sklearn.tree import DecisionTreeClassifier
     from sklearn import metrics
     mod_dt = DecisionTreeClassifier(max_depth = 3, random_state = 1)
     mod_dt.fit(X_train,y_train)
     prediction=mod_dt.predict(X_test)
```

Decision Tree was used here to train the prediction model.

Linear Discriminant Analysis

```
[11] from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
     clf = LDA()
     clf.fit(X_train, y_train)
     prediction1 = clf.predict(X_test)
```

Here, LDA was enforced to train the model.

Naive Bayes Classifier

```
[12] from sklearn.naive_bayes import MultinomialNB
     model_nb=MultinomialNB()
     model_nb.fit(X_train,y_train)
     prediction2 = model_nb.predict(X_test)
```

Naïve Bayes algorithm was applied here.

▾ KNN

```
[13] from sklearn.neighbors import KNeighborsClassifier
     knn = KNeighborsClassifier(n_neighbors=5)
     knn.fit(X_train, y_train)
     prediction3 = knn.predict(X_test)
```

KNN was used to train the model here.

▾ Random Forest Classifier

```
[34] from sklearn.ensemble import RandomForestClassifier
     classifier = RandomForestClassifier(n_estimators=1000, random_state=0)
     classifier.fit(X_train, y_train)
     prediction4 = classifier.predict(X_test)
```

Random Forest was used to make the prediction model here. All these algorithms are evaluated using various metrics to find the algorithm most suited for the problem.

V. RESULTS AND DISCUSSION

Metrics used to evaluate the algorithms in this paper are confusion matrix, classification report and accuracy score.

A confusion matrix has the total count of the accurately grouped occurrences along its cross and the count of the incorrectly classified instances in the rest of the matrix. We have used 4 class values; so, the matrix generated is a 4*4 matrix.

A classification report gives the full report of the classification with parameters like recall, precision, f1-score, etc.

Accuracy score gives the accuracy of the trained model after evaluating it using test data, for which we have sampled 20% of the dataset.

```
[ ] from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

[ ] print(confusion_matrix(y_test, prediction))
    print(classification_report(y_test, prediction))
    print('The accuracy of the Decision Tree is :', "{:.3f}".format(metrics.accuracy_score(prediction,y_test)*100),'%.')
```

```
[[77 18  0  0]
 [ 4 73 15  0]
 [ 0 25 60 14]
 [ 0  0 21 93]]
              precision    recall  f1-score   support

           0       0.95      0.81      0.87        95
           1       0.63      0.79      0.70        92
           2       0.62      0.61      0.62        99
           3       0.87      0.82      0.84       114

    accuracy                           0.76       400
   macro avg       0.77      0.76      0.76       400
weighted avg       0.77      0.76      0.76       400

The accuracy of the Decision Tree is : 75.750 %.
```

Decision Tree was found to be able to correctly forecast the classes with a certainty of 75.75%. The reason for the average level of accuracy obtained is that the Decision Tree is not suited for handling numeric data.

```
[31] print(confusion_matrix(y_test, prediction1))
     print(classification_report(y_test, prediction1))
     print('The accuracy of LDA is :',"{:.3f}".format(metrics.accuracy_score(prediction1,y_test)*100),'%.')
```

```
[[ 93   2   0   0]
 [  2  85   5   0]
 [  0   5  93   1]
 [  0   0   5 109]]
              precision    recall  f1-score   support

           0       0.98      0.98      0.98        95
           1       0.92      0.92      0.92        92
           2       0.90      0.94      0.92        99
           3       0.99      0.96      0.97       114

    accuracy                           0.95       400
   macro avg       0.95      0.95      0.95       400
weighted avg       0.95      0.95      0.95       400

The accuracy of LDA is : 95.000 %.
```

The certainty of LDA is found to be in the ballpark of 95%.

```
[32] print(confusion_matrix(y_test, prediction2))
     print(classification_report(y_test, prediction2))
     print('The accuracy of Naive Bayes Classifier is :',"{:.3f}".format(metrics.accuracy_score(prediction2,y_test)*100),'%.')
```

```
[[74 14  7  0]
 [25 30 26 11]
 [ 4 23 25 47]
 [ 0  6 28 80]]
              precision    recall  f1-score   support

           0       0.72      0.78      0.75        95
           1       0.41      0.33      0.36        92
           2       0.29      0.25      0.27        99
           3       0.58      0.70      0.63       114

    accuracy                           0.52       400
   macro avg       0.50      0.51      0.50       400
weighted avg       0.50      0.52      0.51       400

The accuracy of Naive Bayes Classifier is : 52.250 %.
```

Naïve Bayes returned the correct classes with a veracity of only 52.25% as Naïve Bayes is a poor classifier when working with numeric data as input.

```
[33] print(confusion_matrix(y_test, prediction3))
     print(classification_report(y_test, prediction3))
     print('The accuracy of KNN is :',"{:.3f}".format(metrics.accuracy_score(prediction3,y_test)*100),'%.')
```

```
[[ 92   3   0   0]
 [  3  86   3   0]
 [  0   6  85   8]
 [  0   0   6 108]]
              precision    recall  f1-score   support

           0       0.97      0.97      0.97        95
           1       0.91      0.93      0.92        92
           2       0.90      0.86      0.88        99
           3       0.93      0.95      0.94       114

    accuracy                           0.93       400
   macro avg       0.93      0.93      0.93       400
weighted avg       0.93      0.93      0.93       400

The accuracy of KNN is : 92.750 %.
```

The efficiency of the model trained using the K-NN algorithm was found to be 92.75%.

```
[35] print(confusion_matrix(y_test, prediction4))
     print(classification_report(y_test, prediction4))
     print('The accuracy of the Random Forest Classifier is :', "{:.3f}".format(metrics.accuracy_score(prediction4,y_test)*100),'%.')
```

```
[[ 93   2   0   0]
 [  5  74  13   0]
 [  0  15  73  11]
 [  0   0   6 108]]
              precision    recall  f1-score   support

           0       0.95      0.98      0.96        95
           1       0.81      0.80      0.81        92
           2       0.79      0.74      0.76        99
           3       0.91      0.95      0.93       114

    accuracy                           0.87       400
   macro avg       0.87      0.87      0.87       400
weighted avg       0.87      0.87      0.87       400

The accuracy of the Random Forest Classifier is : 87.000 %.
```

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 LDA with an accuracy of 95%, followed closely by KNN that was able to predict instances with an accuracy of 92.75%. The Decision Tree classifier and the Naïve Bayes classifier failed to forecast the price range optimally.

VI. CONCLUSION

The model trained using LDA was found to predict mobile price classes most accurately (95%). 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.

REFERENCES

[1]. Mustafa Çetın, Yunus Koç, "Mobile Phone Price Class Prediction Using Different Classification Algorithms with Feature Selection and Parameter Optimization", IEEE, 2021, doi**:** 10.1109/ISMSIT52890.2021.9604550.
[2]. Muhammad Asim, Zafar Khan, "Mobile Price Class prediction using Machine Learning Techniques", International Journal of Computer Applications (0975 – 8887) Volume 179 – No.29, March 2018, doi: 10.5120/ijca2018916555.
[3]. P. Arora, S. Srivastava and B. Garg, "MOBILE PRICE PREDICTION USING WEKA", 2020.
[4]. P. Durganjali and M.V. Pujitha, "House Resale Price Prediction Using Classification Algorithms", 2019 International Conference on Smart Structures and Systems (ICSSS), pp. 1-4, 2019.
[5]. D. Banerjee and S. Dutta, "Predicting the housing price direction using machine learning techniques", 2017 IEEE International Conference on Power Control Signals and Instrumentation Engineering (ICPCSI), pp. 2998-3000, 2017.
[6]. Sameerchand 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. 753-764.
[7]. Kanwal Noor and Sadaqat Jan, "Vehicle Price Prediction System using Machine Learning Techniques", International Journal of Computer Applications (0975 – 8887) Volume 167 – No.9, June 2017.
[8]. R. Gareta, L.M. Romeo and A. Gil, "Forecasting of electricity prices with neural networks", Energy Conversion and Management, vol. 47, pp. 1770-1778, 2006.
[9]. https://www.kaggle.com/iabhishekofficial/mobile-price-classification
[10]. https://www.gsmarena.com/