Mobile Price Prediction using Machine Learning

Name:	**Suraj Kanu**
Registration No./Roll No.:	2311007
Institute/University Name:	IISER Bhopal
Program/Stream:	PhD DSE
Problem Release date:	August 17, 2023
Date of Submission:	November 19, 2023

1 Introduction

Mobile phones have become integral to our daily lives in today's technologically driven world. With a wide range of mobile devices available in the market, each offering various features and specifications, determining the price of a mobile phone can be a complex task. However, we can develop models that predict mobile phone prices based on their attributes by using machine learning [1]. Mobile price prediction using machine learning involves analyzing historical data on mobile phones, including their features and corresponding prices. By training a machine learning model on this data, we can uncover patterns and relationships between the features of a mobile phone and its price. This predictive model can then be used to estimate the cost of a new mobile phone based on its specifications.

1.1 Objective

My objective is to predict the price range of a mobile phone by building a model that considers various features provided in the dataset.

1.2 Data Set Description

The data set used in this project is training data to train the model and test data to test the model performance in the training data.

- There are 20 numerical features, and the number of training and test instances are 2000 and 1000, respectively. There are four classes classified as cheap, moderate, economical and expensive. Each category has been assigned a numeric label: cheap - 0, moderate - 1, economical - 2, expensive - 3.
- 500 training instances belong to individual classes.
- As the target variables are discrete, it is considered a classification problem.

1.3 Data Exploration

There aren't any missing values in the test dataset. In the correlation matrix, I have observed that features like RAM(91.70%), pixel height(14.89%)/width(16.58%), and battery power(20.07%) are correlated with the price range. As the RAM increases, the mobile price range also increases.

2 Methods

2.1 Dataset Division

The training dataset is divided into two main subsets: the training set and the testing set in an 80:20 ratio [2].

2.2 Hyperparameter Tuning

Hyperparameter tuning is vital in optimizing the performance of machine learning models, particularly in classification problems. It involves selecting the optimal values for the hyperparameters, which are parameters not learned from the data but set by the user before training the model. By effectively tuning the hyperparameters, one can enhance the model's predictive capabilities and improve the overall classification accuracy.

2.3 Model Selection

Since this is a classification problem, I have employed various existing classification methods by fine-tuning their parameters through hyperparameter tuning. The following methods have been utilized.

2.3.1 Random Forest Classifier

Hyperparameters are used

- criterion:('entropy','gini'), estimators:(30,50,100), max_depth:(10,20,30,50,100,200)

The values for the criterion, max feature, and max depth parameters were determined through grid search, a technique used in hyperparameter tuning. The grid search involved systematically exploring different combinations of values for these parameters. The goal was to identify the values that resulted in the maximum value of the accuracy, a performance metric commonly used in classification problems.

2.3.2 Decision Tree Classifier

Hyperparameters are used

- criterion:('entropy','gini'), max features:('auto', 'sqrt', 'log2'), estimators: (30,50,100), max depth: (10,40,45,60), alpha:(0.009,0.01,0.05,0.1)

The values for criterion, max feature, and max depth are found through grid search and keeping these values constant, the accuracy is measured for different values of ccp alpha from 0 to 0.1, and the value of ccp alpha, which gave the maximum value of Accuracy, is selected.

2.3.3 Multinomial Naive Bayes Classifier

Hyperparameters are used

- alpha: (0,1)

The value of ccp alpha is found through grid search, which gave the maximum value of Accuracy selected.

2.3.4 Logistic Regression Classifier

Hyperparameters are used

- random_state:(0,10)

The value of the random state is found through grid search, which gives the maximum Accuracy value selected. The value of C is found through grid search, which gave the maximum value of Accuracy selected.

2.3.5 Support Vector Machine

Hyperparameters are used

- C:(0.1,1,100), kernel:('linear','rbf','poly','sigmoid'), class weight='balanced',probability=True

The value of C and kernel are found through grid search, which gave the maximum value of Accuracy, is selected.

2.3.6 AdaBoost

Hyperparameters are used

- base estimator:(be1,be2,be3), random state:(0,10)

The value of base_estimator and random state are found through grid search, which gave the maximum value of Accuracy is selected.

2.3.7 K-Nearest Neighbors

Hyperparameters are used

- neighbors: [3, 5, 7,8,11], weights: ['uniform', 'distance'],

The value of neighbours and weights are found through grid search, which gave the maximum value of Accuracy, is selected.

3 Experimental Setup

3.1 Evaluation Criteria

In this project, the performance of different models are measured by the following evaluation criteria:
Accuracy: The accuracy of a classification model is defined as the ratio of correctly predicted instances to the total instances in the dataset.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Where: TP is True Positives,, TN is True Negatives,, FP is False Positives, and, FN is False Negatives.
Precision: Precision measures the proportion of correctly identified positive instances (True Positives) out of all instances predicted as positive.

$$\text{Precision} = \frac{TP}{TP + FP}$$

Recall: Recall, also known as Sensitivity or True Positive Rate, measures the proportion of correctly identified positive instances (True Positives) out of all actual positive instances.

$$\text{Recall} = \frac{TP}{TP + FN}$$

As the dataset is balanced, I choose accuracy as the performance measure.

4 Results and Discussion

I have modified the parameters using the sklearn documentation [3] and conducted experiments. The best results are shown below:

Classifier	Precision	Recall	Accuracy	Best Parameters
Adaptive Boosting	0.965921	0.96500	0.96	class weight= balanced, kernel= linear
Support Vector Machine	0.977668	0.9775	0.98	$C = 0.1$, class weight= balanced, kernel=linear
Decision Tree	0.475164	0.4725	0.47	$\alpha = 0.009$, criterion= entropy, maxdepth= 40, maxfeatures= sqrt, randomstate= 40
Multinomial Naive Bayes	0.496182	0.505	0.51	$\alpha = 0$
Logistic Regression	0.80	0.80	0.76	classweight= balanced, randomstate= 0,solver= liblinear
Random Forest	0.930304	0.93	0.93	classweight= balanced, criterion= entropy, maxdepth=30, maxfeatures= None
KNN	0.954695	0.95500	0.95	neighbors=11, weights='distance'

Table 1: Performance Of Different Classifiers Using All Features

Classifier	Precision	Recall	Accuracy	Best Parameters
Adaptive Boosting	0.9503068	0.952499	0.95	class weight= balanced, kernel= linear
Support Vector Machine	0.967386	0.9675	0.97	$C = 100$, class weight= balanced, kernel=linear, probability=True
Decision Tree	0.833537	0.83	0.83	$\alpha = 0.009$, criterion= entropy, maxdepth= 40, maxfeatures= sqrt, randomstate= 40
Multinomial Naive Bayes	0.477490	0.48750	0.49	$\alpha = 0$
Logistic Regression	0.770908	0.78	0.78	classweight= balanced, randomstate= 0,solver= liblinear
Random Forest	0.9406847	0.94	0.94	classweight= balanced, criterion= entropy, maxdepth=30, maxfeatures= None
KNN	0.957299	0.9575	0.96	neighbors=11, weights='distance'

Table 2: Performance Of Different Classifiers Taking Highly Correlated Features with the Price Range

Classifier	Precision	Recall	Accuracy	Best Parameters
Adaptive Boosting	0.953068	0.952499	0.95	class weight= balanced, kernel= linear
Support Vector Machine	0.967368	0.9675	0.97	$C = 100$, classweight=balanced, kernel=linear probability
Decision Tree	0.789481	0.79	0.79	criterion='entropy', maxdepth=10
Multinomial Naive Bayes	0.477490	0.48750	0.49	$\alpha = 0$
Logistic Regression	0.770908	0.78	0.78	class_weight=balanced, random_state=0 solver=liblinear
Random Forest	0.924735	0.92499	0.93	criterion='entropy', maxdepth=100
KNN	0.957299	0.9575	0.96	neighbors=11, weights='distance'

Table 3: Performance Of Different Classifiers using Feature Selection SelectKBest(k=4)

Classifier	Precision	Recall	Accuracy	Best Parameters
Adaptive Boosting	0.953068	0.952499	0.95	classweight=balanced , kernel=linear, probability= True
Support Vector Machine	0.967368	0.9675	0.97	$C = 100$, classweight=balanced, kernel=linear, probability
Decision Tree	0.597854	0.585	0.58	$\alpha = 0.009$, criterion= entropy, maxdepth=40, maxfeatures='sqrt', randomstate=40
Multinomial Naive Bayes	0.496182	0.505	0.51	$\alpha = 0$
Logistic Regression	0.811686	0.817490	0.82	classweight= balanced, randomstate=0, solver=liblinear
Random Forest	0.922676	0.92250	0.92	classweight= balanced, criterion= entropy, maxdepth=10, maxfeatures=None
KNN	0.954695	0.95500	0.95	neighbors=11, weights='distance'

Table 4: Performance Of Different Classifiers Using All Features and Applying One Hot Encoding

5 Conclusion

From the above performance table, I concluded that Support Vector Machine (SVM) is the best model for this classification problem as it has the highest Accuracy, so I choose SVM for predicting the labels of given test data.

5.1 Improvements

- We can choose other values for hyperparameter tunning to achieve better accuracy.
- We can choose different train test ratio, this may increase efficiency.

5.2 Future Scope

- We can apply Artificial Neural Network for the classification.
- We can incorporate temporal and spatial data if available, considering how mobile prices change over time or in different geographical regions.

6 GitHub Link

https://github.com/skgithub23/mlproject

References

[1] Muhammad Asim and Zafar Khan. Mobile price class prediction using machine learning techniques. *International Journal of Computer Applications*, 179(29):6–11, 2018.

[2] Jimin Tan, Jianan Yang, Sai Wu, Gang Chen, and Jake Zhao. A critical look at the current train/test split in machine learning. *arXiv preprint arXiv:2106.04525*, 2021.

[3] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.