Mobile Price Prediction using SVM

Name:	**Dheeraj Dev C D**
Registration No./Roll No.:	20105
Institute/University Name:	IISER Bhopal
Program/Stream:	EECS
Problem Release date:	January 12, 2023
Date of Submission:	16/04/2023

1 Introduction

This project seeks to predict a price range for mobile phones given the varied specs of various phones. The price range can be divided into four groups: cheap, moderate, economical, and expensive, or, accordingly, 0, 1, 2, and 3. We want to estimate the price range for the mobile devices listed in the test data. There are 20 feature vectors in the data set. The dataset has no missing values.

2 Methods

2.1 Train data, Test Data split

The training dataset is split into train data and test data in 60:40 ratio.

2.2 Hyperparameter Tuning

Hyperparameters are specified parameters that can be used to tune the behaviour of a machine-learning algorithm. These are initialized before the training and supplied to the model. To perform better and improve on the evaluation metric, hyperparameters are tuned by selecting the ideal values. To tune models, all feasible permutations of the hyperparameters for a specific model are used, and the best-performing ones are chosen.

2.3 Classification methods used

Since this is a classification problem, I have used some existing classification methods by tuning their parameters using hyperparameter tuninng. The methods used are:

- k-Nearest Neighbour:
  Here F-measure is found for different values of k from 3 to 100, and the value of K, which gave the maximum value of f-measure, is chosen.
- Decision tree modelling:
  Hyperparameters used are:
  - $'criterion'$: ('gini', 'entropy')
  - $'max\_features'$: ('auto', 'sqrt', 'log2', None)
  - $'max\_depth'$: (15, 30, 45, 60)
  - $'ccp\_alpha'$: $(0.009, 0.005, 0.05)$

  Values for criterion, max_feature, and max_depth are found through grid search, and keeping these values constant, the f-measure 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 f-measure is chosen.

- random forest modelling:
  Hyperparameters used are:
  - $'criterion'$: ('gini', 'entropy')
  - $'n\_estimators'$: [int(x) for x in np.linspace (start = 200, stop = 2000, num = 100)]
  - $'max\_depth'$: $(10, 20, 30, 50, 100, 200)$

  Values for criterion, max_feature, and max_depth are found through grid search. The values for these parameters, which gave the maximum value of f measure, is chosen.

- Gaussian Naive Bayes:
  Hyperparameter used is:
  - $'var\_smoothing'$:np.logspace(0, -13, num = 100)

  Value for var_smoothing is found through grid search, and the value which gave the maximum value of f measure is chosen.

- support vector machine:
  Hyperparameters used are:
  - $'C'$: [0.01,0.1,1,10,100]
  - 'gamma': [1,0.1,0.01,0.001]
  - $'kernel'$:('linear', 'rbf', 'polynomial', 'sigmoid')

  Values for 'C', 'gamma', and 'kernel' are found through grid search. The values for these parameters, which gave the maximum value of f measure are chosen.

- logistic regression:
  Hyperparameters used are:
  - $'C'$: np.logspace(-6, 6, num = 50, base = 2)
  - $'penalty'$:["l1", "l2", 'elasticnet']
  - $'solver'$:['newton-cg', 'lbfgs', 'liblinear']

  Values for 'C', 'penalty', and 'solver' are found through grid search. The values for these parameters, which gave the maximum value of f-measure, are chosen.

2.4 Improving Hyperparameter tuning

The main drawback of hyperparameter tuning is that it can only test for the parameters' values we entered. The time it takes to run the program dramatically increases if we supply additional values for a parameter. To improve performance without adding more values to the grid search, I varied a parameter around its value I obtained from the grid search while fixing other parameters same to get a local maximum. I applied this methodology in the decision tree, SVM, and logistic regression models. A pseudo-code for the same:

- perform grid search to get the best parameters among all the combinations of the inputted parameters
- Select one parameter (which must be numeric) to vary without changing other parameters.
- Plot a graph between the chosen parameter and the model's performance (here, I have used f_measure) around the best value of that parameter we got from the grid search.
- Find the graph's peak; the corresponding parameter value is considered for that model.

```
prediction precision = 0.9735443852539412
prediction accuracy = 0.97375
prediction f measure = 0.9729397024141158
[[213   3   0   0]
 [  1 174   3   0]
 [  0   7 187   1]
 [  0   0   6 205]]
```

Figure 1: Performance after grid search for logistic regression

```
prediction precision = 0.9747533376646539
prediction accuracy = 0.975
prediction f measure = 0.9741960594539361
[[213   3   0   0]
 [  1 174   3   0]
 [  0   7 188   1]
 [  0   0   5 205]]
```

Figure 2: performance after tuning around a parameter for logistic regression

Here, instead of getting the model's best possible parameters, we get one that is better than the parameter obtained from the grid search. Giving these values as input to grid search may produce better results than fine-tuning close to the parameter, but the program will take a lot more time to run. In the case of the decision tree, f_measure was increased from 0.8528 to 0.8600, and for the SVM, the f_measure was increased from 0.97430 to 0.97559

2.5 GitHub link

`https://github.com/dh33rajd/Mobile-Price-Prediction.git`

3 Evaluation Criteria

In this project, the performance is measured by the following evaluation criteria:

- Precision is defined as the ratio of correctly classified positive samples to the total number of classified positive samples.

$$\text{precision} = \frac{\text{No of correctly predicted positive points}}{\text{total predicted positive points}} = \frac{\text{TP}}{\text{TP + FP}}$$

- Recall is the ratio of correctly classified positive samples to the total number of actual positive samples.

$$\text{Recall} = \frac{\text{No of correctly predicted positive points}}{\text{total actual positive points}} = \frac{\text{TP}}{\text{TP + FN}}$$

- Accuracy is defined as the number of samples correctly predicted to the desired classes divided by the total number of samples in the dataset.

$$\text{Accuracy} = \frac{\text{No of correctly predicted data points}}{\text{total number of data points}} = \frac{\text{TP + TN}}{\text{TP + FP + FN + TN}}$$

- F1 Score is the harmonic mean between Precision and Recall.

$$\text{f1\_score} = \frac{2 * \text{precision} * \text{recall}}{\text{precision + recall}}$$

The model giving the best f_measure was used for evaluation.

4 Analysis of Results

The best parameters obtained after tuning are:

- kNN : Value of k = 16.
- Decision tree : ccp_alpha = 0.0045, criterion = 'entropy', max_depth = 15, max_features = None

- Random forest : criterion = 'entropy', max_depth = 30, n_estimators = 222
- Gaussian Naive Bayes: var_smoothing=3.430469286314912e-05
- SVM :C=0.0055, gamma=1, kernel='linear'
- logistic regression : C = 0.005, multi_class = 'multinomial', solver = 'newton-cg'

Table 1 shows the precision, accuracy and f measure for the models used in this project

Table 1: Performance Of Different Classifiers Using All Terms

Classifier	Precision	Accuracy	F-measure	Perfomance ranking
K-NN	0.93295	0.93375	0.93288	3
Decision tree	0.86092	0.8625	0.86002	5
Random forest	0.88149	0.88375	0.88142	4
Gaussian Naive Bayes	0.80025	0.80375	0.80004	6
logistic regression	0.9747	0.975	0.9741	2
SVM	0.97609	0.97625	0.97559	1

5 Discussions and Conclusion

For this particular selection of train and test data, we have found that SVM gives maximum f_measure. Logistic regression is also giving a good f_measure. I will be using the SVM method to classify the test data.

5.1 Improving the methods

- Feature engineering can be implemented, which may increase efficiency and make it easier for the algorithm to detect patterns in the data.
- More the tuning of the parameters, the more efficient the methods will be. Grid search with more and more parameter tuning will improve the models' performance.
- The more parameters chosen for hyperparameter tuning, the more will be the performance.
- Empirical studies show that the best results are obtained if we use 20-30 % of the data for training. Splitting datasets in this ratio may increase efficiency.

5.2 Future scope

- With growing technology, more and more features are rolling out in mobile phones. Adding those features to the dataset, this model can be implemented and predict the prices of mobiles in the future also.
- Same feature may be used with different technology. So the price can vary with technology. These factors can be considered to get accurate results.
- Artificial neural networks can also be used for classification.

6 References

- Tutorial codes provided by the Tutors
- `https://scikit-learn.org/stable/`
- `https://www.kaggle.com/datasets/iabhishekofficial/mobile-price-classification`
- `https://scholarworks.utep.edu/cs_techrep/1209/`