Mobile Price Prediction

Name:	**Aditya Prakash Singh**
Roll No.:	19016
Institute Name:	IISER Bhopal
Stream:	EECS
Problem Release date:	January 13, 2022
Date of Submission:	April 24, 2022

Introduction & Motivation

Mobile phones,[1] also known as cellular phones have been one of mankind's greatest successes towards achieving a complete revolution of communication technology. From simple voicemails & audio conversations, today's smartphones can perform more tasks than full-fledged computers could, just a few years ago.
As smartphones have slowly started turning into a necessity for everyone, the industry has become highly competitive for companies, while their multiple product options often confuse buyers. Hence, a simple tool that can predict the prices of a phone on the basis of its key specifications can be helpful for companies aiming to launch a new product as well as for customers trying to optimise their budget, requirements & expectations from a new mobile phone.

Data

train_data.csv

This file comprises of 2000 rows, each describing a specific mobile phone model through 21 different features, which are as follows:

Feature	Data Type
battery_power	int64
blue	int64
clock_speed	float64
dual_sim	int64
fc	int64
four_g	int64
int_memory	int64

Feature	Data Type
m_dep	float64
mobile_wt	int64
n_cores	int64
pc	int64
px_height	int64
px_width	int64

Feature	Data Type
ram	int64
sc_h	int64
sc_w	int64
talk_time	int64
three_g	int64
touch_screen	int64
wifi	int64

traindata_classlabels.csv

It contains the price_range labels (data type: int64) for each of the mobile phones listed in the *train_data.csv* file, which has the following possible values:
0 - Basic mobile phones, with bare minimum features, having the lowest prices.
1 - Slightly better than class "0", providing better specifications at moderate prices.
2 - Mobile phones with nearly all the necessary features, priced in a medium-high range.
3 - Premium devices, with latest & advanced features along with good design & build quality.

testdata.csv

A file with the exact same columns as test_data.csv, with 1000 rows of data on which the Machine Learning models will make predictions after training as a result of this project.

Methods

The entire project & associated files can be found here.

Visualisation

Upon plotting various graphs involving the different features with respect to their price_range labels, we see a consistent trend of values being directly proportional to the price, irrespective of the column containing continuous values like *battery_power, clock_speed, int_memory*, etc. or discrete ones such as *four_g, wifi, touch_screen* and so on.

Pre-Processing

Fortunately, the provided dataset had no missing values or unnecessary columns that could be eliminated owing to their insignificance. Hence, it could directly be processed for training & testing.

Train-Test Split

The given dataset is randomly split into training & test segments by a factor of 0.75, i.e. 1500 rows (75%) & 500 rows (25%) respectively.

Models Used

As can be seen from the data, this is a classification problem with the various columns acting as feature vectors while the price_range data are our class labels, comprising of 4 classes (0, 1, 2, 3) which are as explained above. Hence, the following classification algorithms were used:

- K-Nearest Neighbors
- Logistic Regression
- Decision Tree Classifier
- Random Forest Classifier
- Gaussian Naive Bayes
- Support Vector Classifier

Hyperparameter Tuning

Hyperparameters are special parameters that are used to tune the behaviour of a machine learning algorithm. These are initialized before the training & supplied to the model, while normal parameters are values that the algorithm learns during training.

GridSearchCV by Scikit-learn[2] is one of the most widely-used & basic hyper parameter tuning techniques in which all feasible permutations of the hyperparameters for a specific model are used. The performance of the model is evaluated on all the combinations of the hyper-parameters & the best performing ones in terms of accuracy are chosen. The hyperparameters for our models are:

Model	**Hyperparameters**
KNeighborsClassifier()	n_neighbors, weights
LogisticRegression()	solver, random_state
DecisionTreeClassifier()	criterion, max_features, max_depth, ccp_alpha
RandomForestClassifier()	criterion, max_depth
GaussianNB()	priors, var_smoothing
SVC()	kernel, C

Experimental Analysis

Firstly, the baseline versions of all the models are trained on the dataset. Then, the model(s) which gave a very high accuracy score were further tuned using GridSearchCV & eventually the algorithm with the highest acccuracy is used to make predictions on the test data.

Since this is a classification problem, the performance metrics used are accuracy, macro-averaged precision, recall & f-measure. These can be mathematically expressed as:

$$Accuracy = \frac{Correctly\ predicted\ data\ points}{Total\ number\ of\ data\ points}$$

$$Precision = \frac{Sum\ of\ true\ positives\ of\ all\ classes}{Sum\ of\ predicted\ positive\ points\ of\ all\ classes}$$

$$Recall = \frac{Sum\ of\ true\ positives\ of\ all\ classes}{Sum\ of\ actual\ positive\ points\ of\ all\ classes}$$

$$f - measure = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

All our selected ML models were baseline trained on the data & their predictions on the test-train split data had the following performance metrics:

	Accuracy	Precision	Recall	f-measure
KNeighborsClassifier()	0.92	0.92	0.92	0.92
LogisticRegression()	0.75	0.75	0.74	0.74
DecisionTreeClassifier()	0.83	0.83	0.83	0.83
RandomForestClassifier()	0.87	0.87	0.87	0.87
GaussianNB()	0.81	0.81	0.81	0.81
SVC()	0.96	0.96	0.96	0.96

Performance Metric Values Obtained from Baseline Training of Models

As we can see, two of our trained models, namely K-Nearest Neighbors (KNN) & Support Vector Classifier (SVC) have a very high rate of accuracy, i.e. above 90%.
Hence, we will proceed with these two for hyperparameter tuning, and attempt to take the resultant accuracy as close as possible to 100%.

K-Nearest Neighbors (kNN)

This method finds the k-nearest neighbors of the test data point from the training samples using a distance function & accordingly allocates it to a particular class. Therefore, in simpler terms, it assigns the test point to a class by taking a majority vote among the k-nearest neighbors.
Upon applying GridSearchCV on kNN, it suggested tuning the attributes *weight = 'distance'* & *n_neighbors = 10*, which enhanced the accuracy from 0.92 to 0.94, an increase of 2.17%.

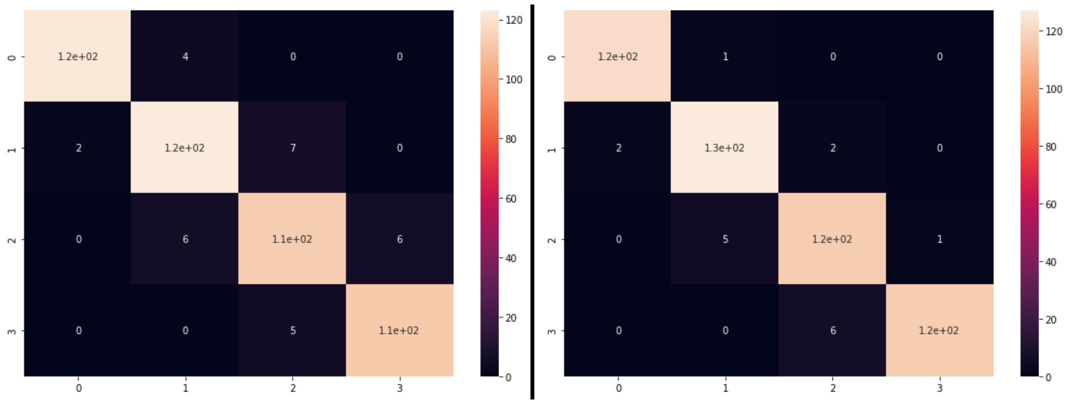
Support Vector Classifier (SVC)

This algorithm is based on libvsm & the fit time scales at least quadratically with the number of samples, and multiclass support handled according to a one-vs-one scheme.
Upon applying GridSearchCV on SVC, it suggested tuning the attributes *kernel = 'linear'* and *C = 0.1*, which enhanced the accuracy from 0.96 to 0.97, an increase of 1.04%.

Following are the results achieved from training both our selected models after hyperparameter tuning as suggested by GridSearchCV:

	Accuracy	Precision	Recall	f-measure
KNeighborsClassifier()	0.94	0.94	0.94	0.94
SVC()	0.97	0.97	0.97	0.97

Performance Metric Values Obtained after applying GridSearchCV on Selected Models



Confusion Matrices for KNN & SVC after Hyperparameter Tuning

Hence, we applied Support Vector Classifier (SVC) to predict the *price_range* labels in the form of an array & then appended them to the original *test_data.csv* file as a column at the end.

Discussions & Future Plans

A few possible improvements in this analysis could be done along these lines:

- Deeper visualization of features to build a mathematical relation of the variations to *price_range*.
- Finding an appropriate scale & currency to which the *price_range* labels can be anchored.
- Allocating weightages to features with respect to how much importance customers & companies place on that particular specification.

Moving forwards, this analysis can give rise to:

- Automated interactive tools that help customers get the best suggestions available within their specification & budget requirements.
- On the contrary, it can also act as a way for companies to collect user sentiment data for aligning their upcoming products accordingly.

References

[1] Wikipedia contributors. Mobile phone — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Mobile_phone&oldid=1072413092`, 2022. [Online; accessed 17-February-2022].

[2] 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.