MINI DATA SCIENCE REPORT

MOBILE PHONE PRICE CLASSIFICATION MODELLING

USING DATA SOURCED FROM KAGGLE

JOEL DOMINGO

joeldomingo117@gmail.com

ABSTRACT

Using machine learning tools, we developed a classification model which utilizes support vector classification to produce an accurate pricing model for a hypothetically soon-to-be released mobile phone. We further experimented with increasing these scores using bagging and boosting ensemble methods. However, most successful model produced very high accuracy and ROC AUC scores without the aid of such techniques. We then applied the model to our hypothetical phone, which has the same specs as the OPPO A53s to compare our model performance to real world data. Our findings indicated that the model itself is practical, however, it requires up-to-date data due to the fast-paced nature of phone technology in order to be used to train a useful model.

PROBLEM STATEMENT

Bob has started his own mobile company. He wants to give tough fight to big companies like Apple, Samsung etc. He does not know how to estimate price of mobiles his company creates.

In this competitive mobile phone market, you cannot simply assume things. To solve this problem, he collects sales data of mobile phones of various companies.

Bob wants to find out some relation between features of a mobile phone (e.g.: - RAM, Internal Memory etc) and its selling price. But he is not so good at Machine Learning. So, he needs your help to solve this problem.

Introduction

Mobile phone prices vary wildly depending on the specifications, import and software regulations, and the current market share of the company. In such a challenging market, one of the main challenges presented to tech companies is to specify appropriate pricing models to align with the future release of a mobile phone.

Goal

Using data science techniques, we want to be able to develop a hypothetical model which can price a company's new mobile phone range using an appropriate pricing model, based on the specifications of the phone.

Method

To achieve such a model, we will use machine learning techniques on a dataset provided by Abhishek Sharma from www.kaggle.com. Model types will include classifier types such as Logistic Regression, K-Neighbors Classifier. Using baseline models as our target, we will experiment on whether ensemble modelling methods such as bagging, and boosting will be able to **improve those models** to achieve a higher classification accuracy.

THE DATA

Source

The data can be found at: https://www.kaggle.com/iabhishekofficial/mobile-price-classification. It contains specification data on mobile phones of various companies.

Size and Volume

The dataset contains **2000 records** of various mobile phone specifications, each with **21 features**.

Data Dictionary

Feature Name	Description	Type	Example
ID	Record ID.	Index	117
Battery power	Total energy a battery can store in one time measured in mAh.	Integer	1043
Blue	Has Bluetooth or not.	Boolean	1
Clock speed	Speed at which microprocessor executes instructions.	Float	1.8
Dual sim	Whether phone has dual sim support or not.	Boolean	0
Fc	Front camera mega pixels.	Integer	14
Four g	Has 4G or not.	Boolean	1
Int memory	Internal memory in gigabytes.	Integer	64
M dep	Mobile depth in cm.	Integer	0.7
Mobile wt	Weight of mobile phone.	Integer	107
N cores	Number of cores in processor.	Integer	8
Pc	Primary camera mega pixels.	Integer	20
Px height	Pixel resolution height.	Integer	720
Px width	Pixel resolution width.	Integer	1280
Ram	Random Access Memory in megabytes.	Integer	3840
Sc h	Screen height of mobile in cm.	Integer	19
Sc w	Screen width of mobile in cm.	Integer	10
Talk time	Longest time that a single battery charge will last when you are on call, in hours.	Integer	20
Three g	Has 3G or not.	Boolean	1
Touch screen	Has touch screen or not.	Boolean	0
WIFI	Has Wi-Fi capabilities or not.	Boolean	1

ANALYSIS

Target Variable – Distribution

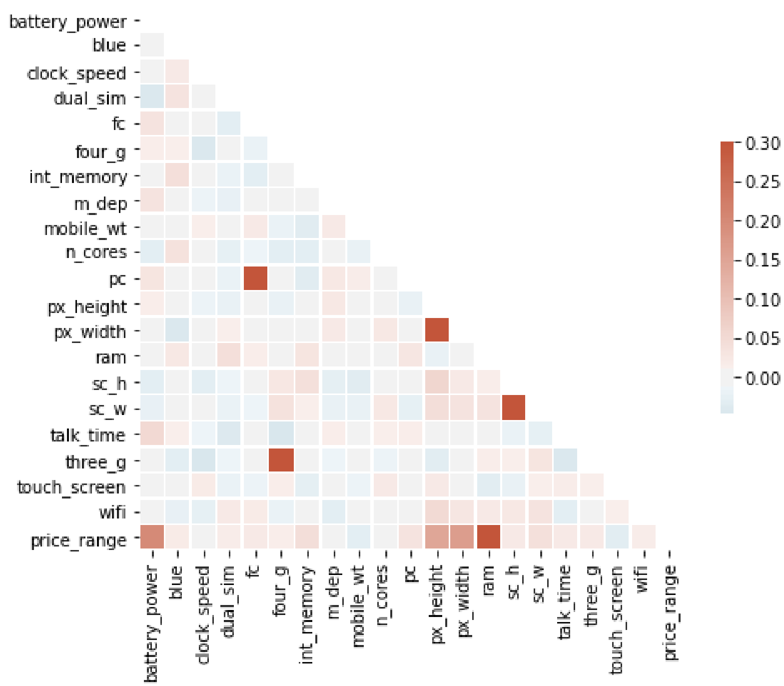
Our target variable for the dataset will be the price-range feature. It has the following class ranges and subsequent distribution in the dataset:

Class	Value Counts
Low cost	500
Medium cost	500
High cost	500
Very high cost	500
Total	**2000**

From these counts, we can see that our data is evenly distributed. Therefore, our target classification accuracy to beat for our model is **25%.** That is to say that if our model were to predict any one class 100% of the time, it would be 25% accurate.

Correlation

Looking at the correlation heatmap below, we can see that mobile phone prices are strongly driven by its battery power and ram capacity features.

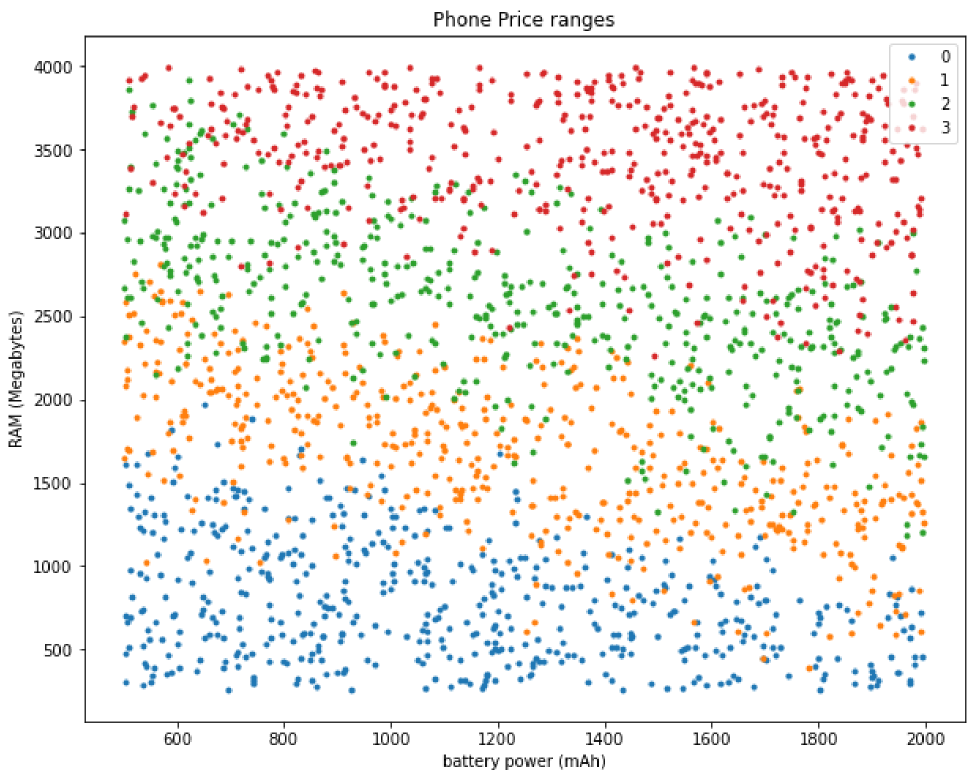


ANALYSIS

Visualizing the Relationship

Observations reveal that all classes can have a large range of battery power specifications. The main distinguisher between classes appears to be the amount of RAM within the phone.

Class	Color
Low cost	Blue
Medium cost	Orange
High cost	Green
Very high cost	Red



MODELLING

For model performance summary, proceed to page 20.

Baseline Model – Logistic Regression

Model	Accuracy	ROC AUC	Precision	Recall	F1-Score
Logistic Regression	0.65	0.89	0.67	0.65	0.65

```
Model: logistic regression
Model score: 0.65
----------------------------------------------------------
ROC_AUC Score: 0.89
----------------------------------------------------------
Classification Report:

              precision    recall  f1-score   support

           0       0.93      0.72      0.81       109
           1       0.52      0.64      0.57        89
           2       0.55      0.45      0.49       106
           3       0.64      0.78      0.70        96

    accuracy                           0.65       400
   macro avg       0.66      0.65      0.65       400
weighted avg       0.67      0.65      0.65       400
```

Figure 1 Classification Report Logistic Regression

Observations

- Considerably higher than target accuracy.
- High area under curve value.
- Model struggled with medium and high price classes.
- Did not perform well across all classes for both precision and recall.
- Not a very high overall model score to beat.

MODELLING

K Neighbors Classifier – Parameters

For this model, we need to find the optimal number of k (neighbors). To do this, we performed parameter testing using GridSearchCV, with a cross validation of 5. This revealed the optimal parameters as follows:

n_neighbors = 15

Baseline Model – K Neighbors Classifier (cv = 5, n_neighbors = 15)

Model	Accuracy	ROC AUC	Precision	Recall	F1-Score
KNN	0.95	1.0	0.95	0.95	0.95

```
Model: knn
Model score: 0.93
-----------------------------------------------------------
ROC_AUC Score: 1.0
-----------------------------------------------------------
Classification Report:

              precision    recall  f1-score   support

           0       0.98      0.96      0.97       109
           1       0.91      0.98      0.94        89
           2       0.90      0.90      0.90       106
           3       0.93      0.90      0.91        96

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

Figure 2: Classification Report KNN

Observations

- Perfect AUC score.
- High accuracy score of 0.93.
- Balanced high values across all classes for precision and recall.

MODELLING

Baseline Model – Support Vector Classifier (probability = True)

Model	Accuracy	ROC AUC	Precision	Recall	F1-Score
SVC	0.96	1.0	0.96	0.95	0.95

```
Model: SVC
Model score: 0.96
---------------------------------------------------------
ROC_AUC Score: 1.0
---------------------------------------------------------
Classification Report:

              precision    recall  f1-score   support

           0       0.99      0.97      0.98       109
           1       0.90      0.99      0.94        89
           2       0.97      0.90      0.93       106
           3       0.96      0.97      0.96        96

    accuracy                           0.95       400
   macro avg       0.95      0.96      0.95       400
weighted avg       0.96      0.95      0.95       400
```

Figure 3: Classification Report Support Vector Classifier

Observations

- Perfect AUC score.
- Very high accuracy of 0.96
- Balanced high values across all classes for precision and recall.

MODELLING

Decision Tree – Parameters

For this model, we need to find the optimal number of max leaf nodes. To do this, we performed parameter testing using GridSearchCV, with a cross validation of 5. This revealed the optimal parameters as follows:

Max_leaf_nodes = 51

Baseline Model – Decision Tree Classifier (max_leaf_nodes = 51)

Model	Accuracy	ROC AUC	Precision	Recall	F1-Score
Decision Tree	0.86	0.95	0.87	0.86	0.87

```
Model: DecisionTree
Model score: 0.86
-----------------------------------------------------------
ROC_AUC Score: 0.95
-----------------------------------------------------------
Classification Report:

              precision    recall  f1-score   support

           0       0.94      0.90      0.92       109
           1       0.79      0.84      0.82        89
           2       0.81      0.85      0.83       106
           3       0.92      0.86      0.89        96

    accuracy                           0.86       400
   macro avg       0.87      0.86      0.86       400
weighted avg       0.87      0.86      0.87       400
```

Figure 4: Classification Report Decision Tree Classifier

Observations

- High AUC Score.
- High accuracy score, but not as high as previous results.
- Balanced values across all classes for precision and recall.

MODELLING

Summary – Baseline Results

Looking back at our baseline models, we observe that K-Neighbors and Support Vector classifiers were our optimal models at a base level. Our next steps are to attempt to attain an observable improvement on any of our models using bagging and boosting techniques. We will be using the sklearn Bagging Classifier, as well as their AdaBoost Classifier.

Baseline Model	Base Accuracy Score	Base ROC_AUC Score
Logistic Regression	0.65	0.89
KNN	0.93	1.0
SVC	0.96	1.0
Decision Tree	0.86	0.95

MODELLING

Bagging Classifier (Base estimator = Logistic Regression)

Model	Accuracy	ROC AUC	Precision	Recall	F1-Score
Bagging (logistic regression)	0.65	0.89	0.67	0.65	0.65

```
Classification Report:

              precision    recall  f1-score   support

           0       0.94      0.72      0.82       109
           1       0.52      0.66      0.58        89
           2       0.55      0.43      0.48       106
           3       0.64      0.78      0.70        96

    accuracy                           0.65       400
   macro avg       0.66      0.65      0.65       400
weighted avg       0.67      0.65      0.65       400
```

Figure 5: Classification Report Bagging Classifier (logistic regression)

Observations

- No noticeable improvement on baseline results across all scores.

MODELLING

Bagging Classifier (Base estimator = KNeighborsClassifier (n_neighbors = 15))

Model	Accuracy	ROC AUC	Precision	Recall	F1-Score
Bagging (KNN)	0.94	1.0	0.94	0.94	0.94

```
Classification Report:

              precision    recall  f1-score   support

           0       0.99      0.95      0.97       109
           1       0.90      0.99      0.94        89
           2       0.91      0.91      0.91       106
           3       0.95      0.90      0.92        96

    accuracy                           0.94       400
   macro avg       0.93      0.94      0.93       400
weighted avg       0.94      0.94      0.94       400
```

Figure 6: Bagging Classifier KNN

Observations

- 1% improvement on baseline accuracy, including all other classification evaluation scores.
- Maintained stability across all classes.

MODELLING

Bagging Classifier (Base estimator = SVC (probability = True))

Model	Accuracy	ROC AUC	Precision	Recall	F1-Score
Bagging (SVC)	0.96	1.0	0.96	0.96	0.96

```
Classification Report:

              precision    recall  f1-score   support

           0       0.99      0.96      0.98       109
           1       0.92      0.99      0.95        89
           2       0.97      0.92      0.94       106
           3       0.95      0.97      0.96        96

    accuracy                           0.96       400
   macro avg       0.96      0.96      0.96       400
weighted avg       0.96      0.96      0.96       400
```

Figure 5: Bagging Classifier SVC

Observations

- No noticeable improvement. Accuracy score remains the same.
- Slight improvements in f1-score by 1%.
- Slight improvements in precision for medium and very high price classes.
- Slight improvements in recall for low, and high price classes.

MODELLING

It should be noted that using a bagging classifier on a decision tree is the same concept as using a Random Forest Classifier.

Random Forest Classifier (criterion = ‘entropy’)

Model	Accuracy	ROC AUC	Precision	Recall	F1-Score
Random Forest	0.88	0.98	0.88	0.88	0.88

```
Classification Report:

              precision    recall  f1-score   support

           0       0.97      0.95      0.96       109
           1       0.80      0.88      0.84        89
           2       0.82      0.79      0.80       106
           3       0.91      0.89      0.90        96

    accuracy                           0.88       400
   macro avg       0.88      0.88      0.88       400
weighted avg       0.88      0.88      0.88       400
```

Figure 8: Bagging Classifier SVC

Observations

- 2% increase in baseline decision tree accuracy.
- 3% increase in ROC AUC score.

MODELLING

AdaBoost Classifier (base estimator = Logistic Regression(), n_estimators = 10)

Model	Accuracy	ROC AUC	Precision	Recall	F1-Score
AdaBoost (logistic regression)	0.61	0.88	0.60	0.61	0.58

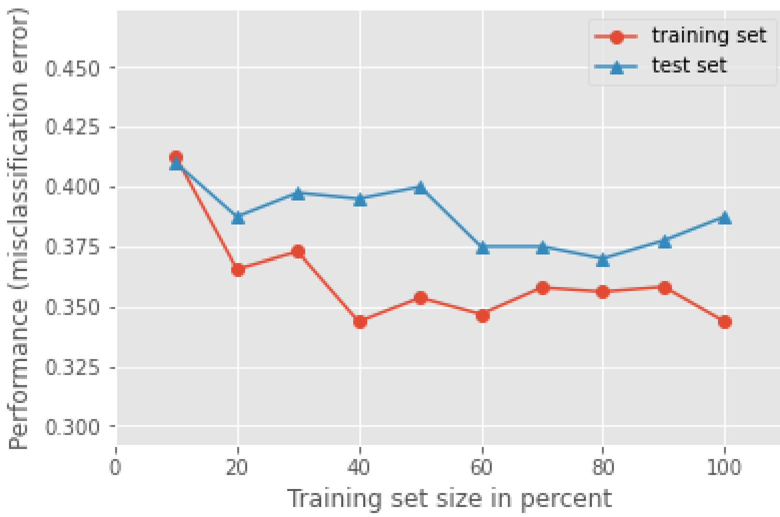


Figure 9: Learning Curve AdaBoost (LogReg)

```
Classification Report:

              precision    recall  f1-score   support

           0       0.77      0.85      0.81       109
           1       0.49      0.47      0.48        89
           2       0.55      0.22      0.31       106
           3       0.57      0.91      0.70        96

    accuracy                           0.61       400
   macro avg       0.60      0.61      0.58       400
weighted avg       0.60      0.61      0.58       400
```

Figure 10: Classification Report AdaBoost LogReg

Observations

- -4% decrease in accuracy on baseline score, -1% in ROC AUC.
- Performed poorly with too much bias in the training and test set.

MODELLING

AdaBoost Classifier (base estimator = SVC (probability = True), n_estimators = 10)

Model	Accuracy	ROC AUC	Precision	Recall	F1-Score
AdaBoost (SVC)	0.94	1.0	0.94	0.94	0.94

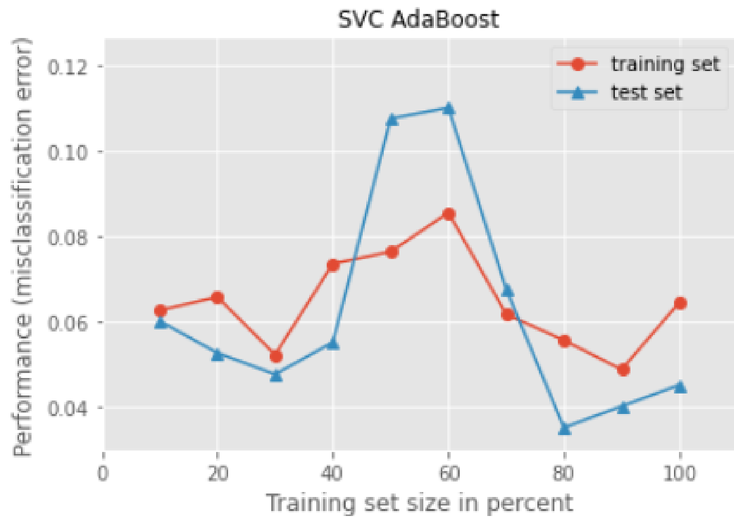


Figure 11: Learning Curve AdaBoost (SVC)

```
Classification Report:

              precision    recall  f1-score   support

           0       0.96      0.99      0.97       109
           1       0.98      0.90      0.94        89
           2       0.92      0.92      0.92       106
           3       0.92      0.96      0.94        96

    accuracy                           0.94       400
   macro avg       0.94      0.94      0.94       400
weighted avg       0.94      0.94      0.94       400
```

Figure 12: Classification Report AdaBoost (SVC)

Observations

- -2% decrease on baseline accuracy, zero change in ROC AUC.
- Balanced performance across classes.

MODELLING

AdaBoost Classifier (base estimator = Decision Tree Classifier (max_leaf_nodes = 51), n_estimators = 10)

Model	Accuracy	ROC AUC	Precision	Recall	F1-Score
AdaBoost (Decision Tree)	0.84	0.97	0.85	0.84	0.84

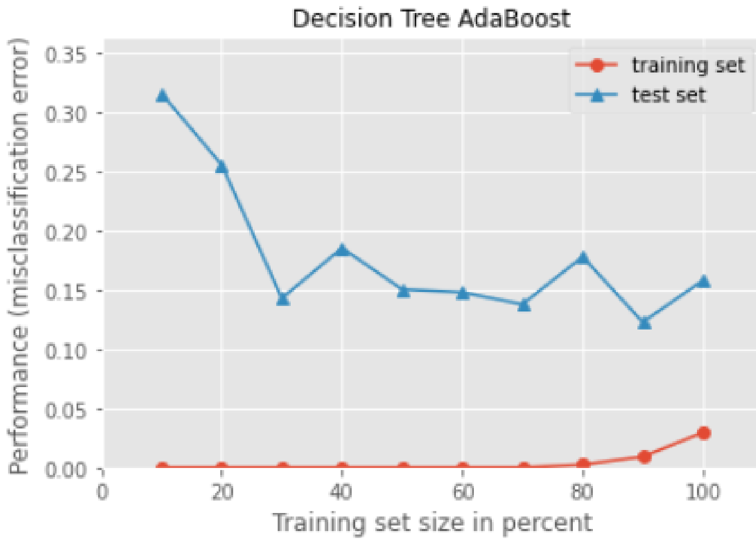


Figure 13: Learning Curve AdaBoost (SVC)

```
Classification Report:

              precision    recall  f1-score   support

           0       0.98      0.90      0.94       109
           1       0.75      0.87      0.80        89
           2       0.75      0.78      0.76       106
           3       0.91      0.81      0.86        96

    accuracy                           0.84       400
   macro avg       0.85      0.84      0.84       400
weighted avg       0.85      0.84      0.84       400
```

Figure 14: Classification Report AdaBoost (SVC)

Observations

- -2% decrease on baseline accuracy, -1% in ROC AUC.
- Learning curve shows the training had high amounts of bias, and was not able to generalize as well as it could on the test set.

MODELLING

AdaBoost Classifier (base estimator = Random Forest (criterion = ‘entropy’), n_estimators = 10)

Model	Accuracy	ROC AUC	Precision	Recall	F1-Score
AdaBoost (Decision Tree)	0.84	0.97	0.85	0.84	0.84

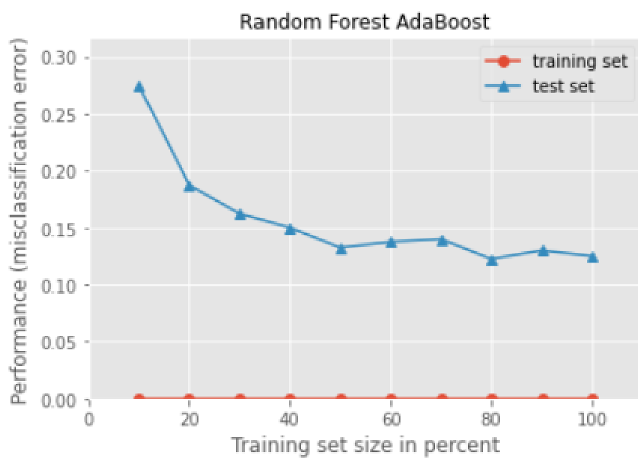


Figure 15: Learning Curve AdaBoost (SVC)

```
Classification Report:

              precision    recall  f1-score   support

           0       0.97      0.94      0.96       109
           1       0.78      0.89      0.83        89
           2       0.84      0.75      0.80       106
           3       0.90      0.92      0.91        96

    accuracy                           0.88       400
   macro avg       0.87      0.88      0.87       400
weighted avg       0.88      0.88      0.87       400
```

Figure 16: Classification Report AdaBoost (SVC)

Observations

- ➢ +2% increase on baseline accuracy, +3% in ROC AUC.
- ➢ Learning curve shows the model had high amounts of bias and was unable to generalize well on unseen data (test set).

MODELLING EVALUATION SUMMARY

Performance Comparison

Baseline Model	Base Accuracy Score	Bagging Accuracy Score	AdaBoost Accuracy Score	Base ROC_AUC Score	Bagging ROC_AUC Score	AdaBoost ROC_AUC Score
Logistic Regression	0.65	0.65	0.61 (-4%)	0.89	0.89	0.88 (-1%)
KNN	0.93	0.94 (+1%)	n/a	1.0	1.0	n/a
SVC	0.96	0.96	0.94 (-2%)	1.0	1.0	1.0
Decision Tree (Bagger = RF)	0.86	0.88 (+2%)	0.84 (-2%)	0.95	0.98 (+3%)	0.97 (-1%)
Random Forest	0.86	0.86	0.88 (+2%)	0.95	0.95	0.98 (+3%)

Observations

- Overall, boosting methods did not increase, but rather decreased performance. The boosting models created an environment which increased bias, making the resulting model unable to generalize well on the test data. This means the model was vulnerable to overfitting unseen data.

- Upon observation, AdaBoost with Random Forest as a base estimator achieved **better results** than the bagging and baseline models.

- **KNN** was left out of boosting, as it did not have the attributes for supporting sample weighting, making it incompatible with AdaBoost.

- **Our best overall model** remains to be the **baseline** Support Vector Classifier. As seen in the relationship between RAM and Battery Power on page 6, if we were to strictly use these two attributes to distinguish our input, the classes are clearly separable. Therefore, it is expected that the SVC will perform well.

- Bagging and Boosting did little to overcome the base model enough to justify them over the baseline SVC model.

USE CASE

Objective

For our use case, we will be making a hypothetical classification on the OPPO A53s, which was released in October of 2020, to provide a demonstration on the model.

The Model

The model being used is the baseline Support Vector Classifier, which we achieved an accuracy score of 0.96, and an ROC AUC of 0.98.

The Data

- For our classification, we will be using the official specifications of the A53s gathered from the official OPPO website (https://www.oppo.com/au/smartphones/series-a/a53s/).
- It has the following specifications (mapped to our model features):

	Specs		Specs
Battery Power	2000 mAh	**Primary Camera**	13 MP
Bluetooth Capable	Yes	**Resolution Height**	1600 px
CPU Clock Speed	1.8	**Resolution Width**	720 px
Dual Sim	No	**RAM**	4000 MB
Front Camera MP	8 MP	**Screen Height**	164 mm
Has 4G	Yes	**Screen Width**	75 mm
Storage	64 GB	**Talk Time (charge)**	19.3
Mobile Depth	0.8 cm	**Has 3G**	Yes
Weight	186 g	**Has Touch Screen**	Yes
CPU Cores	8	**WIFI**	Yes

USE CASE (RESULTS)

Result

Inputting the features specified on the A53s, our SVC model predicts this should **follow a premium (3 – very high) pricing model.**

Problems

It should be noted that the data is outdated (last updated 3 years ago in 2018), and since then, phone, communications and computing technology in general has advanced by a large margin. In the real world (2021), the OPPO A53s' pricing model, as well as its technical specifications follow a budget-like standard.

In a somewhat subjective sense, the current real-world pricing scale would be as follows:

Price Range ($ AUD)	Price Class
100 – 399	Low
400 – 699	Medium
700 – 1000	High
1000 +	Very High

Our use case specifies a high price model, however in reality is impractical and inaccurate. This suggests that the model requires time-sensitive data, and the training dataset must be updated with current year technology for it to be effective in a business context.

CONCLUSION

Using the data provided, we were able to train multiple classification models which take the specifications of different mobile phones from several different companies. The training data was sufficiently diverse regarding both specifications and pricing classes, enough to make a clear distinction between them. It was observed that the strongest drivers for pricing a phone include the battery power and the RAM capacity. Out of the models trained, we found the best baseline model to be the Support Vector Classifier. However, we were unable to significantly improve on this model using bagging and boosting techniques as they indicated high amounts of bias when training the data. They could potentially be further optimized using parameter tuning, however the baseline model appeared to be sufficient at a 0.96 accuracy rate. Overall, the features which feed into the model were found to be sufficient for a well-performing model.

Recommendations

- Use well-maintained data. Model requires time-sensitive data to suggest a practical pricing model.

Sources

- **Dataset Source:** https://www.kaggle.com/iabhishekofficial/mobile-price-classification
- **GitHub link to code:** https://github.com/jdomingo117/Projects/tree/main/Mobile%20Phone%20Price%20Classification/Code
- https://scikit-learn.org/
- https://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretation-of-performance-measures/