Prediction of Mobile Phone Pricing Groups

# Introduction

In today’s world, people are more reliant on mobile phones than ever. Many of us no longer have landline phones due to the reliability that cellphones have gained over the last few years. The mobile devices themselves have become more and more sophisticated as well. We manage our Financials, Calendars, play games, use them for museum tours, order food, and even as the access mechanism for transit in some larger cities. They have replaced cameras and video capture devices for all but enthusiasts.

This expansion of features has led the price mobile phones to steadily rise, as cramming all these features and more that are demanded today by even the most casual of user. Mobile phones are now major a purchase most people are making every two or three years on average. Correctly pricing them for features is more important for mobile phone companies as more companies continue to get into the game for designing a better phone. So, what features become the most important predictors of price?

For this project I am looking at the Bob’s mobile phone data set on Kaggle and looking to utilize a model that can be used to predict which price bucket to which 2000 mobile phones belong. There categories are as follows:

0 representing the end of the market, these might be entry level phones for children or often offered by mobile phone companies.

1 representing the medium grade phone, there is cost the consumer but lower in comparison to major flag ship model. These can also be lower models of more expensive phones. Example would be iPhone C.

2 represents high-cost phones. Examples would the traditional iPhone or Samsung Galaxy.

3 represents the highest level offered. These are commonly business use phones but are still available for heavy phone users. iPhone Max would sit here.

# Preprocessing

As mentioned in the pervious section the data set was collected from Kaggle. It includes 20 variables about 2000 mobile phones. It also includes a column that includes the pricing categories as defined in the introduction.

The data was provided in the numeric format for all variables including the categories. First step was to turn the categories from a numeric variable to factors. Classification algorithms prefer nominal non-numeric values for the classification tags. By transforming to factors we have provided them with the data in a format that the algorithms prefer.

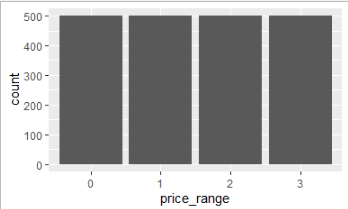
In addition to transforming the classification tag, I also converted several attributes that were 0 or 1 to from numeric values to nominal as well. 0 In these cases represents that the feature was absent from the device. While 1 represented that the phone had the feature.

These features were as follows: is the phone blue, does it have a dual sim card, does the phone have 4g capabilities, does the phone have 3g capabilities, does the phone have a touch screen, does the phone have a wifi receiver.

# Dataset exploration

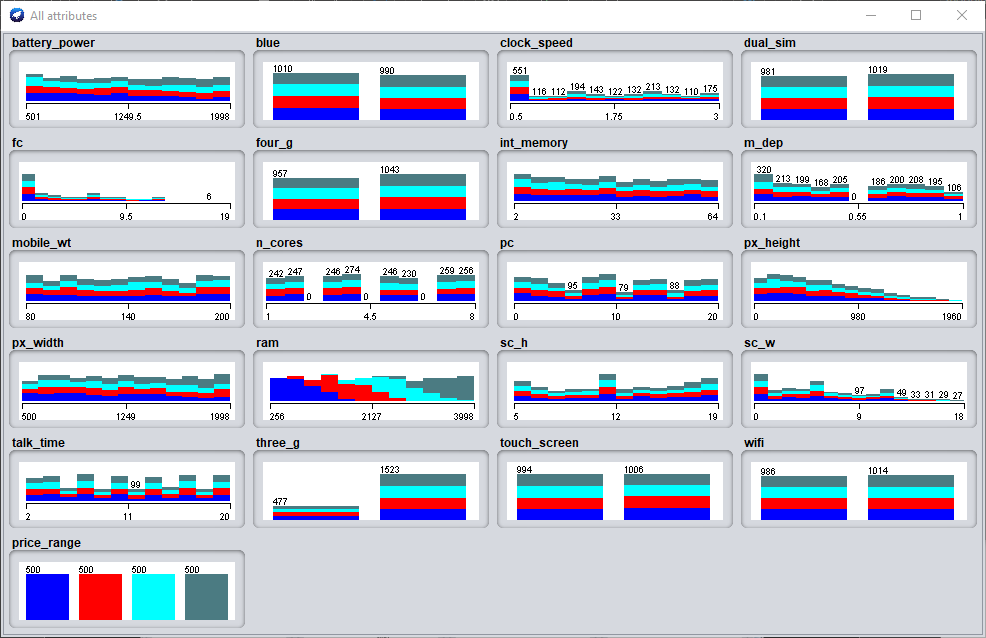
As a second step before training the models, it to look at data that is contained in the dataset and make sure that variables are balanced.

Starting with the classification; it is quickly seen that this dataset is quite balanced as 500 samples for each classification level.



It is important for the algorithm to have enough of each type for it to be able to distinguish between each of the categories and make predictions. This may be a bit over kill as I would expect that in the real world, we would probably see more of price range 1 and 2 out in the world. This would give the companies products that could be easily marketed to mass public. With price range 3 being maybe 1 model a year per company as they would not want to spend time producing phones that they do not have the demand to sell through. Similarly, price range 0 would also likely be rarer as people in this category tend be either those who are less acclimated to technology products or people who cannot afford those in price ranges 1 or 2. For the algorithms though having these equally balanced distributions should help the models learn overall.

In addition to making sure the balance of classification tags, looking across the features to see if there is anything that is likely to be of help in the models. Weka helped with this quick visualization of the different features against their price tag.



Notice that ram will likely be a major factor in what helps some of the models determine the price range. In addition, although slight, there is a visual increase in battery power amount the more expensive models and a decrease in the lower models. Both make intuitive sense as the phones with more features will need more battery power to power the feature expanse and memory to handle all the different tasks at hands.

# Training the Models

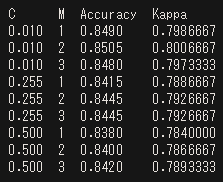
For this data set, I will be looking at 4 different classification models. Starting with the Decision Tree Model, Random Forest, Support Vector Machines, and lastly Naïve Bayes to determine which model works best for classifying to which price range each phone belongs.

Additionally, for all the models I will be using cross validation fold of 10. This means that while the model is training it breaks the training set into 10 even groups. 9 of which are used to train the model, and 1 is used to test. The model will run through each set and test and the training accuracy to figure out how well the model is working. It will score both in terms of percentage accuracy and kappa (or how well the model reperforms).

## Decision Trees

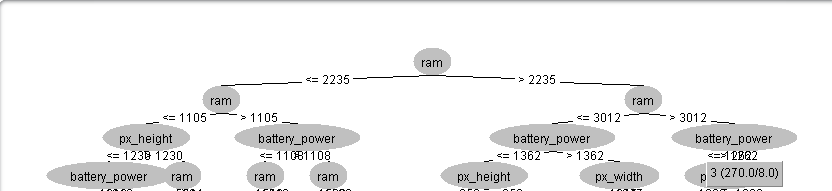
Decision trees are one of the easiest to understand machine learning models. Named for their appearance when drawn to look like the upper portion of a tree. A person can quickly follow from the root node to the child leaf node to see how model is classify a specific instance. While they are simple to follow, they tend to preform poorly when introduced to new data.

I will be utilizing the J48 decision tree algorithm defined in Weka. The Algorithm allows for two tuning measures. Confidence, or how aggressively prune off lower branches of the tree. Minimum Number of instances, meaning how many branches is a node required to have.



The results show that for the best results a confidence level .01 and Minimum split of 2 branches per nodes results 85% accuracy and a recall of 80%.

Here is a graphical representation mentioned earlier of the upper part of this tree in generated using Weka. As we see the root node starts with the RAM and moves down to battery power or more surprising the pixel height feature of the phone.

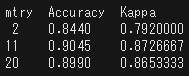


Overall, the decision tree provided a surprisingly high level of accuracy.

## Random Forest

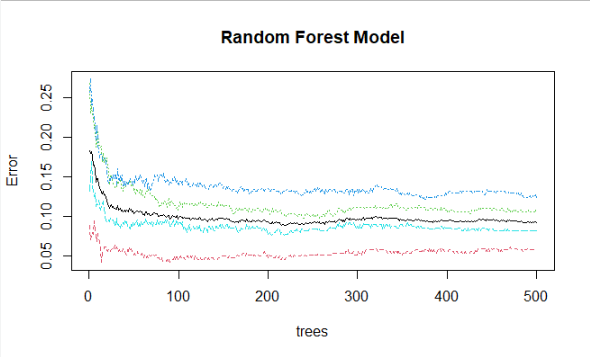
Random Forest is the next model I tested on the data set. It is a relative of the decision tree. It works by randomly starting with different variables and growing decision trees. The major tuning factor with this algorithm is how many trees are needed in this forest.

When the data is ready to be classified, the entry is run down each of the decision trees and the outcome is scored. When the data has been run down all the trees, the result which has the highest score is the one which wins out.



I grew this forest using R, which allowed me to test for the optimum number of trees automatically. The optimum number of trees for the forest was 11 with a 90.45% accuracy and 87% recall.

This can be seen more easily in the graph below, the error (inverse of accuracy) starts high and quickly level off around 11, we after that point the model runs pretty much pretty much horizontal to the x-axis meaning more trees are not reducing the error.



This model performed better than the decision tree model both in terms of accuracy with the training data, and its ability to perform when being introduced to new data. This is not surprising as noise in the data that is modeled in one decision tree can be filtered out other decisions.

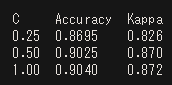
## Support Vector Machines

Support Vector Machines is the third model type I tried. At a base level what it does is looks at data in multiple dimensions and tries to find the place where the two data points of different classes and be separated by the most distance.

Once that point is found, a patrician is created at the midpoint between these two points. As new data is introduced to the model they are classified based on where the points fall in relation to this barrier. The model is surer of points that fall further way than the training points from the median, and less sure about those that fall in the margin between them. Because of the type of math that it utilized to look at the data set, it can classify nonlinear data and creating boundaries that look like nonlinear shapes on 2d graphs.

The tuning concept for this model is called cost. It looks at how to get the most accurate results when points in the training data may be misclassified. By allowing cost we can reduce the noise training data that can cause the model to over fit.

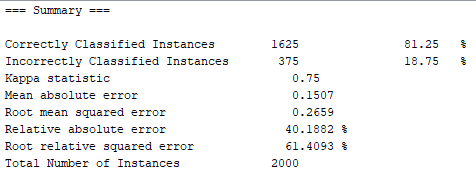
To test the model, I ran varying cost allowance and found the greatest accuracy by allow cost in the model.



The Model accuracy here was five hundredth of a percent less accurate than the random forest, it is important to note when introduced to new data through the cross fold validation that the model was able to preform at the same level as the random forest.

## Naïve Bayes

Last model is the Naïve Bayes Model. The Naïve Bayes model uses a logistic curve to predict a yes or no value for a classifier. The NB Model also had the added ability of providing how sure it is of a certain classifier. For the phone which may have features putting it towards to the top or bottom of a category, this added information can be helpful if trying between 1 category or another when making predictions on close classifiers.



The model; however, performed the worst on the data set. Providing an overall accuracy of only 81% and performing around 75% when being introduced to new data.

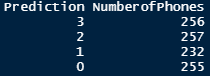
# Conclusion

Based on the review of the various algorithm, the recommendation for the Phones Data Set on Kaggle would be to utilize a Random Forest Algorithm on the test set to predict the price range classes for the various mobile phones.

This algorithm produced the highest accuracy on 10-fold cross validation with 11 trees grown. The model was also tied with support vector machines with the highest Kappa score. As such, my confidence in the model to preform well when introduced to new data is quite high.

The other discoveries for the hypothetical client to focus in on for the client are Ram, Pixel Height, and not surprisingly battery life if they wish to command top dollar for their devices.

As the data from Kaggle comes with a test set that that client in the prompt wanted reviewed for the best outcome to figure out which phones get priced, I ran the results of the test. There are 1000 phones here is the breakout by category:



The prediction shows the set of phones is well stratified. Since the customer in the prompt is looking for which category to set the price for each phone, I would provide back the IDs with their prediction categorization buck and the breakdown.