Beer Classification Report

Morgan Reilly - 20235398

October 31, 2020

Research what open-source machine learning packages are available, and select one that you think will be appropriate for this task. Your report should include a brief justification for your choice and an overview of the main features of the package you have chosen.

Upon analysis of this I found that there are many available open-source machine learning packages available for this task. These consist of the likes of TensorFlow, Keras, Scikit-learn and Theano. Through past experience and projects I have used TensorFlow and Keras, and for this project I had initially opted to use Keras as it is relatively user-friendly and I thought should work well for a classification task such as this. However upon reviewing this in a QA session for this module I found out that this task was better suited to SciKit-Learn or WEKA as Tensorflow and Keras are more so aimed at Deep Learning projects. For this task I have opted to use SciKit-Learn as it has quite a robust feature set.
The main features of this application, outside of the necessary imports such as Pandas, Numpy, or CSV, consist of the likes of the 'tree' library, which allows me to use the various 'tree' models with that library, but mainly the Classification And Regression Tree (CART) algorithm which is necessary for this assignment; the 'preprocessing' library, which allows me to normalise the feature and sample set for this task; and the 'ensemble' library, allows me to use a Random Forest, which is required for the second part of this assignment as comparison. I have also included a handful other packages such as 'model_selection' which can be used for cross-validation and for getting the cross-validation score; the 'metrics' library, which is used to generate and plot a confusion matrix; and the 'ensemble' library, which allows me to implement a Random Forest Classifier.

A data-set is supplied below. You will probably need to do some work to prepare it for input into the ML package, depending on the package's requirements. Document any data preparation steps in your report.

To handle the data being read I made a function to read from a text file to save and return a csv file, which takes two parameters: 'file_in', the directory of the file to be read in; and 'file_out' the directory of the file to be stored. It digests the given text file and stores it into a Pandas data-frame. It declares the column headers as per project outline. It saves as a '.csv' file and returns a data-frame of the csv values.
Once the data-frame is set from both the testing and training .csv files, I then call a function I wrote which generates the samples and features for both the training and testing data. Sample generation involves looping over the data and storing each iteration into as a row in a list, apart from the classification label, which gets ignored. Feature generation does the inverse of sample generation, where it only stores the classification label. From there the lists are put through a normaliser. For the sample set I normalise between 0 and 1, and for the feature set it converts the string classification label to a from 0-2. These are then stored as numpy arrays for the model to handle.

In the ML package, select two different classification algorithms that you will apply to the data-set to learn two different classification models. One of these should be one that we have already covered in lectures (ID3/C4.5 or kNN) while the other should be completely different from those we have already covered. In your report, include a clear description of both algorithms. Ensure that you acknowledge all of your sources of information.

CART Algorithm

The Classification And Regression Tree (CART) algorithm is fundamentally based off of Ross Quinlan's ID3 algorithm, which is a Decision Tree based algorithm. Decision tree algorithms are normally quite fast due and have a comprehensive nature to them which is quite important for decision making. CART is represented as a binary tree, where each root node represents an input and leaf nodes represent output. Because of the binary nature of the tree, decision making is essentially an if-else statement. Data preparation for the CART model is also fairly straightforward where the only necessary steps being is that there is a clear problem to be addressed. This algorithm generates a tree by selecting input variables and splitting points on the variables. To do this it uses a greedy algorithm to minimise the cost function. The algorithm functions as a tree builder and weights each node with a value to compare the prediction to. The CART tree is constructed as a binary decision tree by repeatedly splitting two child nodes for the whole learning sample, beginning at with the root node.

Random Forest

Random forests are a class of ensemble learning where you essentially create multiple individual decision trees. Being a part of the ensemble learning class means that they utilise multiple learning algorithms to produce an optimal performance than what would be obtained from a single learning algorithm. With this each individual tree outputs a prediction for the class, which is then voted on. The class with the most votes becomes the prediction for the model.
A tree is grown in the following manner:

1. If number of cases in training set is N, random cases from sample N
2. If there are variables of M input, use $m << M$ to split the node. Value of m is constant during growth.
3. Grow each tree to maximum extent. Do not prune.

In the ML package, train each of your chosen algorithms using the training set provided in beer_training.txt. You should then test your trained models using the test set provided in beer_test.txt. Report on the results, including the classification accuracy for each model on the training set and on the test set. Also include details of the classification models constructed – these may include graphics if appropriate.

In terms of loading and training the data I'm unsure if I have done it correctly. I fear I may have missed a step in the process of it. The score for both models subsequently was relatively low at the start. On introduction of cross-validation it caused an serious over-fitting to both models. The scores for both models went from a dominant high on the training score to a dominant high on the testing score.

CART Algorithm

To train this model I initially set the model by creating an instance of DecisionTreeClassifier. I then fit the model with the previously set X_train (feature set for training) and y_train (sample set for training) data. This will train the model using the normalised data, X_test (feature set for testing), and y_test(sample set for testing). I follow the same procedure for testing the model. For this model on the initial accuracy output gave a train score of 100% and a test score of 66%. To try and improve the accuracy of this model I apply

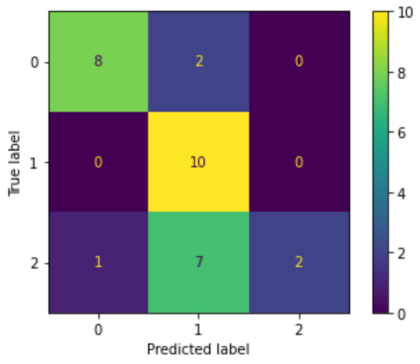

Figure 1: CART Confusion Matrix

a 5 fold cross validation to the model. Cross validation is a re-sampling process which is used to assess the performance of the predictions for the model. When cross-validation was applied it resulted in an output of 89% on training accuracy, and an output of 86% on test accuracy. There could definitely be improvements made to this model to increase accuracy but out of the box with cross validation it performed reasonably well. This model also utilises a metric known as the Gini index as a was of assessing and selecting the best attribute. This is used to measure the impurity of a set instead of using entropy. The Gini values of each node in the tree can be viewed in Figure 2 and is calculated as: $Gini(S) = 1 - \sum_{i=i}^{n} p_i^2$

Random Forest

To train this model I initially set the model by creating an instance of RandomForestClassifier. I then fit the model with the previously set X_train and y_train data. This will train the model using the normalised data, X_test (feature set for testing) and y_test(sample set for testing). I follow the same procedure for testing the random forest model. This model seemed to perform better than the CART model. It's initial train score was also 100%, but had a test score of 73% which was much better than the CART model. To also try and improve the accuracy of this model I apply a 5 fold cross validation to the model. After introducing Cross Validation the testing and training accuracy both dropped and rose. It's training accuracy got worse but the testing accuracy got much better. I also believe this model may also be over-fitting because on some runs the accuracy when testing was almost at 100% whereas the training accuracy would have been around the 90% mark. As discussed in the CART, this model also uses Gini as a way of measuring the impurity of a set, but I was unable to generate a decent diagram successfully.

Discuss in your report whether the two models give very similar or significantly different results, and why.

I think both models performed in a similar way since they are both Tree based. The Random Forest performed a bit better than the CART model tho due to the nature of taking the best tree as an output. If I were to make any changes I would have used the kNN algorithm instead of the CART algorithm, in hopes to get a more balanced view on the differences between the two.

References

[1] https://machinelearningmastery.com/classification-and-regression-trees-for-machine-learning/

[2] ftp://ftp.boulder.ibm.com/software/analytics/spss/support/Stats/Docs/Statistics/Algorithms/14.0/TREE-CART.pdf

[3] https://www.stat.berkeley.edu/ breiman/RandomForests/cc_home.htm

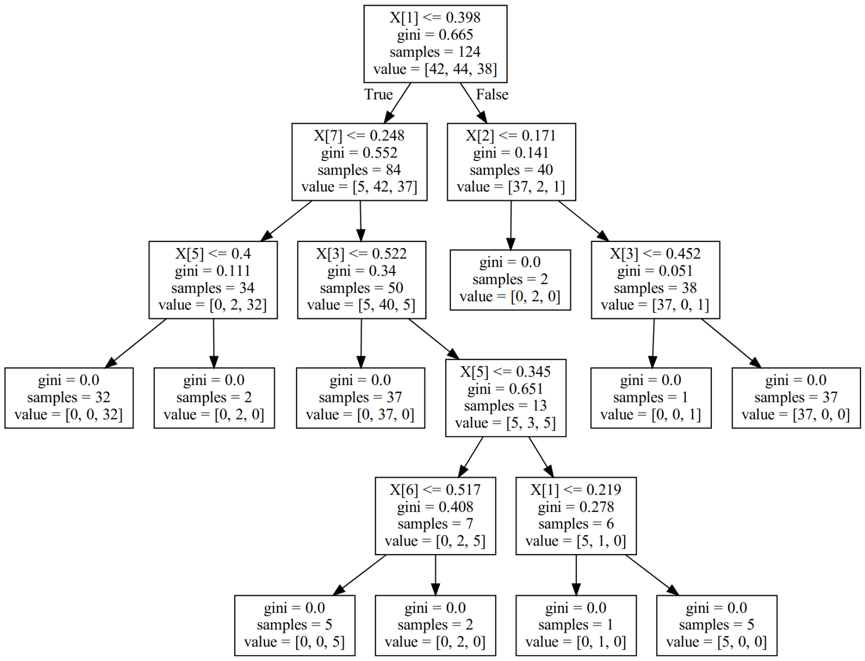

Figure 2: CART Tree

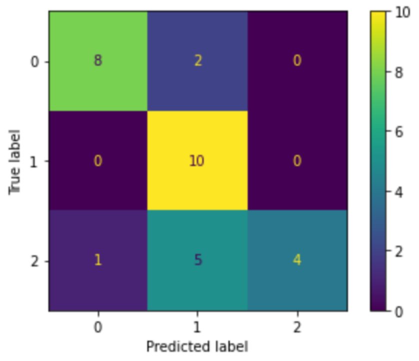

Figure 3: Random Forest Confusion Matrix