Student Number: 198735 & Kaggle Team Name: jc812

1. Approach

The approach adopted was the Extra Trees Classifier [3], after attempting the Random Forest [4] and Decision Trees [1], and understanding that Extra Trees provided the best solution to the Machine Learning problem. These types of classifiers were intially considered as the high dimensions of the dataset, suggested that each of these dimensions should be sorted as a tree, and therefore have a higher amount of decision making before deciding on a classification.

Extra Trees or to give the proper name Extremely Randomised Trees, extends upon the logic of Random Forests.

Random Forests are an ensemble of Trees, the trees being a set of Nodes that test a single feature, in this case this is one of the features in the image. The splits in the tree look for the most discriminative thresholds and are calculated using the whole training set or a random subset of the size 'max_features' which the user can set [4]. Each of these trees are different from every other tree. This is because they are randomised in two ways: by taking a random sample subset of the training data; followed by a random subset of the data features (these being the image features that each of the training data images has). Predictions are then made by majority voting across all trees, the most likely prediction being the most frequent. By combining the trees together, the predictions are improved and there is less overfitting overall in the classifier [6] .

Extra Trees calculate the splits differently, the thresholds are calculated by picking a threshold at random for each feature, and the best of these thresholds are picked as the splitting rule; this causes a reduction in variance but does increase the bias of the model [2].

2. Methodology

Due to the limited amount of training data and the fact that this was a domain adaptation problem, the method used was model selection in order to choose the appropriate complexity and hyper-parameters of the given classifier. Using visual inspection combinedwith trial and error. A single hyper-parameter was altered in the classifier and submitted to the Kaggle competition in order for it to show a score. Based upon this score and previous scores, the parameters of the classifier were adjusted accordingly [7][8].

The feature pre-processing used was standardisation, by collecting each of the features into lists of that feature, using Scikit's scale on the lists, then re-adding them to the original rows [9].

Regarding feature selection, after trial and error it was discovered that using the whole training dataset was unsuitable as the classifier was too well versed on the amount of 1 classifications and it would return limited proportions of 0 classifications. When selecting the subset in question, as there were minimal 0 classifications (326), using all of them would be the best call. As for 1 classifications, choosing the first 1's in the list turned out to be an easy and efficient way to select a subset for training the classifier. The ratio of class 0 to 1 was made to be 50/50, and finding that the percentage of 1's classified was lower than required, the ratio was changed to 43/57 [12].

Both CNNs and GIST features are equally important, as they added equal amounts of information needed to classify these images, and therefore the more information given about the images the better the classifier would perform.

To increase the accuracy of the classifier the additional training data was used. An assumption was made that the missing features, the nulls in the data, are not present in their corosponding image and therefore the NaNs were changed to 0s. As this training data had additional 0 classifications it was useful as the classifer would understand what makes a 0 classification clearer [5].

The test label proportions were inputted into the classifier using the parameter 'class_weights' [3]. It was found that the classifier would remain closer to these proportions after adding them to the parameters, and as the proportion of classifications were accounted for, this would ensure there wasn't a disproportionate amount of one classification. Additionally, knowing the proportions the test data is meant to be, may enable a quicker insight into how well this classifier would do when testing.

When fitting the classifier, it transpired that (after looking on the Scikit website) it had a third parameter which was for training label confidence. After adding the list of confidences, the classifier should be more accurate, as it takes into account given the features, how likely one classification could be [3].

On the subject of the domain adaptation problem, the approach used to combat this was the use of re-weighting the classifier by using the test proportions in order to make the training data appear like the testing data. Furthermore, the classifier is being taught on different domain data, not the same type of data, and therefore won't be oblivious when looking at the testing data as it would understand what each feature means in an image, and what it means when coming to classify it [10].

3. Results and Discussion

3.1 Results

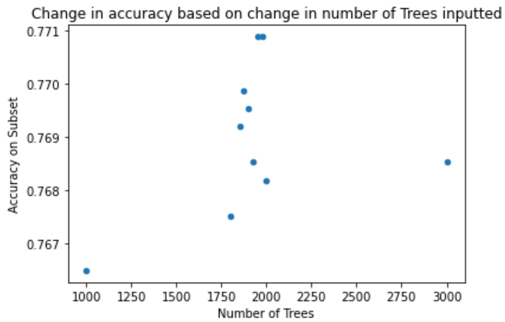

Figure 1 – A graph showing the change in accuracy dependant upon the number of trees inputted as a parameter

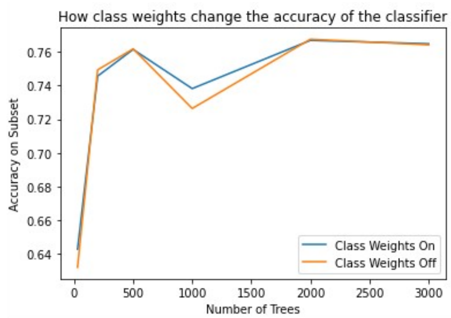

Figure 2 – A graph showing the change in accuracy dependant upon whether class_weights were used

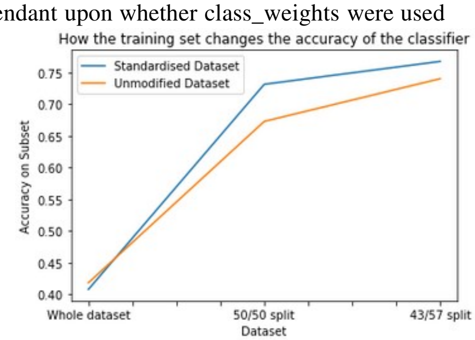

Figure 3 – A graph showing the change in accuracy based on the training set size changing

3.2 Discussion

The graphs above represent the key hyper-parameters of the Extra Trees classifier, and the change in how the dataset it is trained on effects the accuracy. Hyper-parameters not shown in the graphs include: instances to split an internal node, which were kept at a default of 2, as this allowed the most amount of nodes and an increase in decision making and overall accuracy; and features to consider when splitting, after testing using the whole training data was optimal as it was too small to be effective when taking a subset of the data [11].

As seen in Figure 1, the optimum amount of trees and the highest score achieved was with 1950 trees. The graph shows a constant trend other than this spike in accuracy. The amount of trees is the hyper-parameter which changed the most when testing the classifier.

Figure 2 shows a close relationship between having class_weights on and off. By deciding to keep them on it was found that using the test proportions increased the reliability of the classifier due to it learning more about the testing data.

Figure 3 shows a dramatic increase in accuracy when changing what the classifier learns (its learning curve). The graph shows, that when standardising the training and testing data, greater accuracy overall was achieved.

The additional training data was very useful as it provided extra information for the classifier to be trained on. The test label proportions were very useful, as they enabled better judgement on how well a submission would be graded based on the percentage of class 1s to 0s. Finally, the training label confidence was useful as it allowed the classifier to judge how useful each training classification was, and made it more accurate in the process.

The domain adaptation problem here meant that no classifier would be 100% accurate or anywhere close, as the classifier doesn't understand the data it is being tested on. Therefore when training the classifier, it needs all the information it can get that is relevant in a general context of that subject so that it can predict accurately [10].

To improve on the work produced, the dataset could be standardised earlier, in order to hit higher accuracies quicker. Further, by changing the NaNs to 1s in the additional training data an improvement in accuracy may have been achieved [5]. Improving the accuracy of the classifier would need a way of training the classifier with more relavent data, whether that be greater amount of data in general as the testing set was much greater than the training set, or getting data images about London to train with and therefore forgoing the domain adaption problem. Additionally, having a way to test the classifier on a subset of the testing data not on Kaggle would have been beneficial as it would have allowed unlimited testing on the dataset in a single day.

In evaluation of the classifier an increase in data shown in the graphs would indicate the classifier going through a smoother learning curve. On reflection, when submitting to the Kaggle competition the selected submissions should have been the last submissions not the best public data submissions, as the best public data submissions did not standardise the data. Furthermore, starting with the Extra trees classifier as opposed to Decision trees would improve earlier results. However, at its peak the classifier achieved 77% accuracy, and this being within the top 100 of year.

References

[1] scikit-developers (2007) Decision Trees. Available at: https://scikit-learn.org/stable/modules/tree.html

[2] scikit-developers (2007) Ensemble Methods. Available at: https://scikit-learn.org/stable/modules/ensemble.html

[3] scikit-developers (2007) sklearn.ensemble.ExtraTreesClassifier. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html

[4] scikit-developers (2007) sklearn.ensemble.RandomForestClassifier. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[5] Păpăluță, V. (2020) What's the best way to handle NaN values? Available at: https://towardsdatascience.com/whats-the-best-way-to-handle-nan-values-62d50f738fc

[6] Simpson I. (2020) 'Decision Trees and Random Forests' [PowerPoint] G6061:Fundementals of Machine Learning. Available at: https://canvas.sussex.ac.uk/courses/8739/files/1306660?module_item_id=713473

[7] Brownlee, J. (2019) A Gentle Introduction to Model Selection for Machine Learning. Available at: https://machinelearningmastery.com/a-gentle-introduction-to-model-selection-for-machine-learning/

[8] Simpson I. (2020) 'Model Selection and the Bias-Variance Decomposition' [PowerPoint] G6061:Fundementals of Machine Learning. Available at: https://canvas.sussex.ac.uk/courses/8739/files/1248156?module_item_id=704690

[9] scikit-developers (2007) Preprocessing data. Available at: https://scikit-learn.org/stable/modules/preprocessing.html#standardization-or-mean-removal-and-variance-scaling

[10] Multiple Contributors (2020) Domain Adaption. Available at: https://en.wikipedia.org/wiki/Domain_adaptation

[11] Fraj M.B. (2017) In Depth: Parameter tuning for Random Forest. Available at: https://medium.com/all-things-ai/in-depth-parameter-tuning-for-random-forest-d67bb7e920d

[12] Multiple Contributors (2020) Feature Selection. Available at: https://en.wikipedia.org/wiki/Feature_selection