Student Number: 168396 & Kaggle Team Name: podge

1. Introduction

This machine learning task was about discovering how one can train their machine to differentiate between whether a picture is sunny or not sunny. The training data was a range of photographs taken on a beach however the testing data is taken in urban city areas, making the task slightly more complex to tackle.

1.1. Approach

The approaches taken in solving this task both used a Random Forest Classifier with slightly different parameters. This classifier works by using a vast range of decision trees to train, delivering a result showing the prediction. Decision trees aid in classification by using their branches representing the observations about the data and the leaves representing the conclusion. The conclusions here are the labels of the data and the branches are the features that lead to them. Random Forests use these tactics and at the same time combat the issue of decision trees overfitting to their training data; having low bias but very high variance. Random Forests reduce this variance by averaging numerous decision trees trained on different parts of a set of training data, consequently increasing the accuracy [1].

2. Methodology

2.1. Pre-Processing

Normalisation was implemented using the Normalizer from sklearn.preprocessing in the Python library. Normalisation refers to the process of rescaling real valued numeric attributes into the range 0 and 1. This is useful for models that rely on the magnitude of values and in the preparation of coefficients in regression.

Principle Component Analysis (PCA) is an effective method of reducing high dimensionality in data which is often a cause of inaccuracy in many machine learning techniques. This was considered when deliberating which technique to use for this project but after Random Forests became the technique of choice it was clear that PCA would not be necessary and it would not need to be reduced. Feature importance is already integrated into the functionality of Random Forests, used in the tree splitting, so does not require this information from PCA. Additionally, decision trees deal with this dimensionality well.

2.2. Class Imbalance

The training data provided begins with a majority leaning towards 1s, which could have been dealt with by either balancing them out or making them imbalanced in the other direction. Both of these techniques were used and this was the key difference in the two approaches. In the first approach the data was skewed to reflect the reality of the ratio of the testing data, having more 0s than 1s (specifically 67.67% 0s and 32.33% 1s). In order to achieve this reverse in imbalance and, all 1 values with a confidence of 0.66 were removed. These were the best to remove as the pieces of data are as not as valuable since they do not provide a reliable enough insight into the data. The other approach used a balanced training set achieved by counting the difference in 1s and 0s and removing that exact amount of 1s (again with 0.66 confidence) to equate their amount.

2.3 NaN values

The training data contains many missing values, therefore in order to make the data more useful a mean value across each feature was used to replace them. This was achieved by performing the fillna and nanmean method of NumPy provided by SciPy in the Python library.

2.4 Optimising the Hyper Parameters

Both the approaches utilised a random search in order to configure the optimal parameters to use in the Random Forest Classifier. The parameters that can be modified are the number of estimators 'n_estimators', 'min_samples_split','min_samples_leaf', 'max_features', 'max_depth' and 'random_state'. Tuning the number of estimators is essential as if their quantity is too large it can overfit to the training data which reintroduces the decision tree's overfitting problem that the Random Forest is attempting to solve. As a result the random search is run with a wide range of n_estimators values ranging from 10 to 810 increasing by 20 each time. This range was chosen as 10 would likely underfit the model and anything around a 1000 would overfit it. Using this range in a random search meant that the best value that fit within this range was found. The random searches provided alternate suggestions for n_estimators as they were both run with a differently skewed training set, suggesting a lower number of estimators for the balanced training data. This suggests that getting a distinction between the classes required more data when the training data was skewed. Given that more estimators are used alongside the imbalance in the favour of 0s, one may expect this model to perform better than the balanced model.

3. Results and Discussion

3.1 Results

The model with the highest accuracy was the one which skewed the data to match the testing set, with a score of 72.017% and a cross validation score of 75%. The model with the balanced classes achieved a score of 65.340% with a cross validation score of 72%. As you can see from the learning curves in Figure 1 and Figure 2, neither of them obviously over or underfit the training data. As a result these two models were chosen for the final submission.

Although it is best practice to balance a training data set when performing a classification task, from the learning curves and cross validation testing the model that skewed the data appears to have performed better. This could be a result of the random forest attaining more value out of more features when the data is reflective of the testing data.

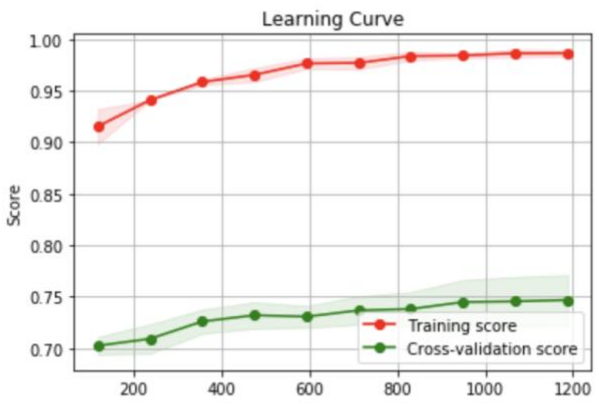

Figure 1-Learning Curve for balanced model

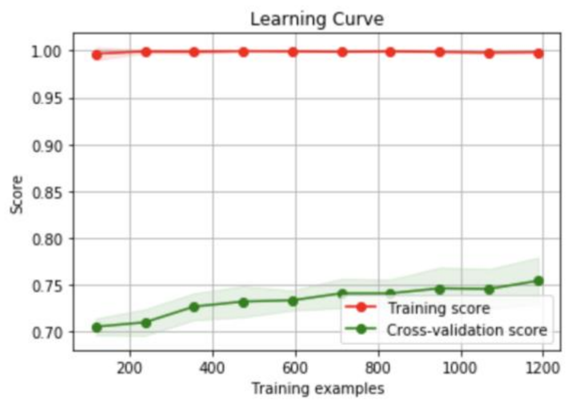

Figure 2-Learning Curve for imbalanced model

3.2 Criticisms

The skewed training data is suitable for the task at hand as the proportions of the data are already known, however given that these proportions have been used the model is overfitted to this testing data set. If one were to use this model to classify a different set of images that did not contain these proportions the best model may not perform as well. Conversely, the model of the balanced data may prove to be a more robust classifier in the above example.

Additionally, the images in the testing data were taken in an urban environment in contrast to the training data which were taken by the seaside. As a result there may be a slight difference between the source and target domain as these environments are likely to have an effect on which features could determine something is sunny. Therefore to improve both of these models some effort could be made into understanding where these domains differ and if any preprocessing could be performed to map them both into a single domain which may bring more accurate classification.

In terms of preprocessing the NaN values were replaced just using the mean and normalization was the only method attempted to bring the data into a comparable range. There may have been some value in trying other techniques such as scaling or binarization in bringing the data into a comparable format. Similarly, one could have attempted to replace the NaN values using some sort of prediction, perhaps using linear regression, which may have added more value.

References

[1] R. T. J. F. Trevor Hastie, The Elements of Statistical Learning: Data Mining, Inference and Prediction, Springer, 2009.