Student Number: 215865

1. Introduction

This report will explore the use of a Random Forest Classifier to solve a binary classification task, while considering the impact of model selection, feature selection, imputation and confidence scores on the accuracy of the model.

2. Approach

Random Forest Classifiers (RFC) are an ensemble machine learning method where a number of decision trees are constructed for the training set, and the result is averaged over them. This makes them effective at generalizing predictions, rather than simply learning the pattern of the training data.

Random Forests use bootstrapping, the process of randomly sampling a subset of the training data to construct each decision tree. This also allows them to use 'OOB Score', which is similar to cross validation, as it allows the user to test the accuracy of the classifier on unseen data.

RFC use sample weights as a parameter which allowed us to explore incorporating the confidence scores for the labels into the model. Feature importance is automatically calculated by RFCs, which also made them useful for feature selection

3. Methods

3.1. Pre-Procesing

Initially we used K-Nearest Neighbours Imputation, in order to resolve the missing values. However, KNN Imputation was a time consuming process, and so we attempted using the row mean as an alternative. This produced no noticeable difference in the accuracy of the resulting model but significantly reduced the speed of the model, and so for future runs we used the row mean to resolve the missing values. Upon further investigation, we found that replacing the missing values simply with 0 was a better approach, and this is what was used for the final model.

We did not rescale the data, as Random Forests and decision trees do not require it to produce accurate predictions.

3.2. Model Selection

In order to ensure that the model was of a suitable complexity to produce accurate predictions but not overfit, tuned the following parameters of the model, using GridSearchCV. To do this, we set aside a validation data set, explored cross-validation and used the Out-of-bag score for testing the model, as this is similar to cross-validation in that it calculates accuracy for an unseen dataset.

- Number of trees in the forest
- Maximum depth of each tree
- Maximum features to consider at each split
- Minimum samples to split

The maximum depth of the tree is important in controlling the complexity of a machine learning model. The default value in sci-kit is None, meaning that the model can grow out as far as possible and therefore overfit. Although Random Forest's already average out the predictions to reduce overfitting, controlling the maximum depth of the tree proved to be significant in accuracy and running time. We chose a value of 5 after experimenting with GridSearchCV.

Max features has the effect that the lower it is, the greater the reduction of variance, but also the greater the increase in bias. On the advice of [4] and the results of Grid Search, the square root of the number of features was chosen. It refers to the number of features to consider when looking for the best split at a node.

3.3. Feature Selection

Initially, we used Principal Component Analysis (PCA) in order the reduce the number of dimensions in the training data, however after experimenting with PCA we wanted an approach which considered the features importance, as there may be many which do not contribute to the predictions of the model.

The Random Forest Classifier also allows you to inspect the impurity based importance of the different features, which we investigated. The wrapper class 'SelectFromModel' in Sklearn allowed us to easily transform the dataset into a smaller dimensionality. You can see the effects of feature selection in the Figure. We tested the model over the following numbers of features: 10, 50, 100, 500, 1000, 2000, 4000 to decide.

3.4. Predictions

Since the test data also had missing values, we performed imputation using 0 as done in the training set. The test data was also transformed using the SelectFromModel package in the same way as the training and validation data.

4. Results

The average accuracy of our final classifier is 0.79, using both training data sets with the second data set imputed with 0 for all missing values, and only using data when the confidence for the label was 1. We also selected the top 500 features for the model, as seen in Figure 2, this produced reasonable results and reduced the complexity of the model.

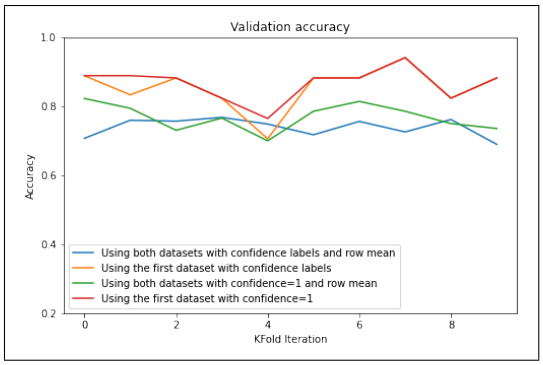

Figure 1. Comparison between models that don't use feature selection, on different training sets.

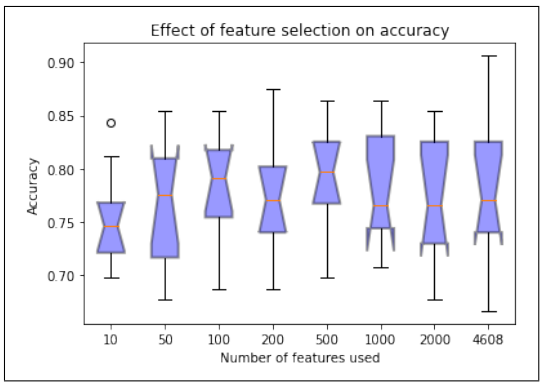

Figure 2. Comparison between the number of features selected for the model

As seen in Figure 1, the highest accuracy used only the first data set, however this was a data set of only 600, and if we only consider confidence scores of 1, 172 samples, which is insufficient to train an image classifier.

Figure 2 demonstrates how we chose the number of selected features, 500, as the median was higher than any other number of features, and reducing the number of features by this much successfully sped up the classifier. Figure 3 indicates the importance of choosing the maximum depth of the decision trees, as higher depths encouraged over-fitting on the training accuracy, but do not increase the validation accuracy.

5. Discussion

Although we experimented using KNN Imputation and filling the missing values by row mean or 0, a greater accuracy could be achieved by using the most common value by class, as our imputation method does not take the class into account.

The running times for the GridSearchCV algorithm were very high so we were unable to explore all of the possible combinations to find the best. It is highly likely that there

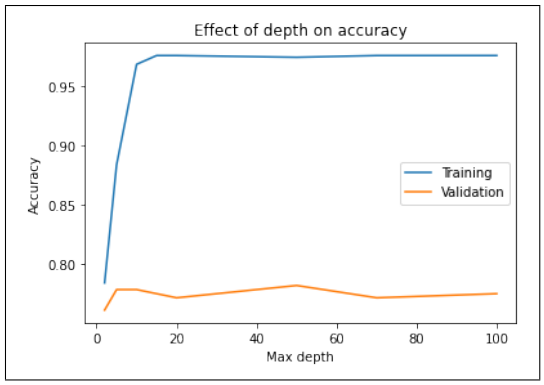

Figure 3. Plot showing the effect of the maximum depth of each decision tree in the random forest on the accuracy and overfitting.

are alternate combinations for this model which would improve the performance.

Although we have used accuracy, F1 score and confusion matrix to evaluate the performance of the model, these are all quantitative measures. Access to the images would have allowed us to perform qualitative evaluation of the model, and potentially better identify the sources of uncertainty in the model to better tune the hyper-parameters.

The most time-consuming process during the building of this classifier was in tuning the hyper-parameters, which has demonstrated the value of this tuning process, and shown that there is no "one-size fits all" approach to binary classification. We have also discovered the importance of the order of pre-processing steps. Initially, our pre-processing involved using KNN Imputation before separating the data into training and validation steps, and the validation data was unusually high (¿0.9). We later found that there was another source of information leakage in the pre-processing steps, but through our research we found that KNN Imputation should be performed after train/validation split in order to prevent any information leaking which could impact the performance of the model on validation sets.

References

[1] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011

[2] Rogers J., Gunn S. (2006) Identifying Feature Relevance Using a Random Forest. In: Saunders C., Grobelnik M., Gunn S., Shawe-Taylor J. (eds) Subspace, Latent Structure and Feature Selection. SLSFS 2005. Lecture Notes in Computer Science, vol 3940. Springer, Berlin, Heidelberg. https://doi.org/10.1007/11752790$_1$2

[3] L. Breiman, "Random Forests", Machine Learning, 45(1), 5-32, 2001

[4]Simon Bernard, Laurent Heutte, Sébastien Adam. Influence of Hyperparameters on Random For- est Accuracy. International Workshop on Multiple Classifier Systems (MCS), Jun 2009, Reykjavik, Iceland. pp.171-180, 10.1007/978-3-642-02326-2$_1$8.hal –

00436358.