Homework 1: Decision Tree and Random Forest

Stefanela Stevanović
MLDS1 23/24, FRI, UL
63220492

I. Introduction

The goal of this homework is to implement decision tree and random forest with 100 trees algorithms for the classification problem, report their performance and compute feature importances. The results obtained from the experiment are presented and discussed in this paper. Data used for this task was the FTIR spectral data set, that consists of 188 samples (rows) that belong to two different classes, and 198 features (columns). First 130 rows of the FTIR data set were always used as the training set and the remainder as the testing set.

II. Decision Tree and Random Forest Implementation

The decision tree algorithm is implemented using two classes: *Tree* class, whose purpose is to build a tree model, and the *TreeNode* class, whose purpose is to make predictions with the built tree model. The tree is built by recursively splitting a parent node into two child nodes based on the values of the input features. Two stopping criteria are defined:

1) Pure node: all labels at the node are the same
2) The number of samples at the node is less than min_samples. Parameter min_samples is 2 for the classification tree in this homework.

When these criteria are met, further recursive splitting of the node is stopped and it is considered a leaf node, which represents a class label. To find the best split, at each node, the algorithm considers all possible splits on all features and selects the one that leads to the largest reduction in Gini impurity. For each feature, the values were sorted and based on these indices, the training set was also sorted in order to enable greater efficiency of the algorithm. Then, the algorithm loops through each unique value and creates a split at that value, separating the training data into two groups: those with feature values lower than the current value(threshold) and those with feature values greater than or equal to the threshold. The algorithm computes weighed Gini impurity of each split and chooses feature value that resulted in the lowest Gini impurity, which will be used as the split for the current node in the decision tree. This procedure is repeated recursively for each subsequent node until one of the stopping criteria is reached.

The predict method of the *TreeNode* class is used to predict the labels of new samples by traversing the tree. Traversing a tree involves starting at the root node of the tree and following a path down to a leaf node based on the values of the input features.

To build the random forest model with 100 trees in the *RandomForest* class, two levels of randomness were introduced:

1) Bagging (bootstrap aggregating): In each tree, a random subset of the training data is selected with replacement to grow a tree. Each random subset has the same size as the training set.
2) Random feature selection: In each tree, at each node, a random subset of features is considered for splitting the data. This random subset of features has size equal to square root of total number of features, which in this case is 14 random features.

Predict method of *RFModel* class uses majority voting technique to give predictions of the random forest model. It collects predictions made by each tree in random forest and chooses the most common prediction for each data point as the final one.

III. Performance

Misclassification rate, which is simply the proportion of data points that are misclassified by the model, was used as a performance measure of the decision tree and random forest models. The uncertainty of both the decision tree and the random forest was estimated in order to understand the reliability of the model's predictions. Standard deviation was used to quantify the variability of a model's performance metrics across 1000 data subsets. In order to generate these subsets, the already mentioned bootstrap resampling technique was used on model predictions. The misclassification rate was calculated for each of the 1000 subsets and the standard deviation was computed. Misclassification rates and uncertainty estimates of the implemented algorithms on train and test data are shown in the table 1.

From the table 1 it's obvious that random forest model achieves higher accuracy than the single decision tree and reduces variance. This happens due to randomness introduced when building the random forest. Bootstrapping the training data in the random forest model helps to reduce the variance by reducing the impact of individual outliers or noisy samples in the training data. Furthermore, using random subset of features

TABLE I
DECISION TREE AND RANDOM FOREST WITH 100 TREES MISCLASSIFICATION RATES AND STANDARD DEVIATION.

Model	Data	Misclassification Rate [%]	Standard Deviation [%]
Decision Tree	train	0.00	0.00
	test	24.14	5.47
Random Forest	train	0.00	0.00
	test	3.45	2.36

at each split, prevents the model from relying too heavily on any single feature and makes model more robust to noise and irrelevant features. Via performing majority voting on the predictions of these diverse trees, the errors of individual trees cancel out, which leads to more accurate final prediction of the random forest. Also, it is important to note that the misclassification rate of random forest varies significantly for different random seeds.

Figure 1 shows that the misclassification rates of the random forest generally decrease with the increase in the number of trees. Having more trees in the forest increases the chances that the algorithm will capture the underlying patterns in the data, an thus produce more accurate predictions. The best performance of the model is observed with around 75 and 85 trees. After 90 trees, the curve flattens out to a slightly higher misclassification rate. This may be caused by accuracy oscillations due to the randomness of the model, or it may indicate overfitting.

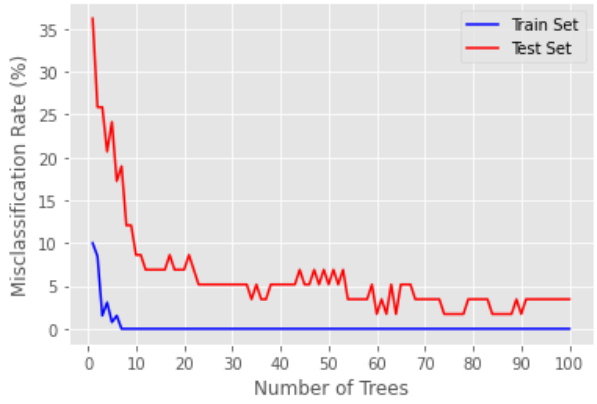

Fig. 1. Misclassification rate versus the number of trees. This figure shows how the misclassification rates of the random forest for the train and test set changes with an increase in the number of trees

IV. FEATURE IMPORTANCE

To compute the feature importance of the random forest with 100 trees, the permutation method was used, described in Breiman (2001) [1]. The idea behind this method is use out-of-bag samples, randomly shuffle the values of each feature and observe the impact on the model's performance. The feature with the highest impact on misclassification rate is considered the most important. For each tree in the forest, the baseline misclassification rate was computed by prediction the out-of-bag samples. Then, one by one feature was randomly shuffled and the percentage change in misclassification rate with the shuffled feature was computed. This percentage change is considered as the feature importance. Feature importances of each of 100 trees in random forest were summed and averaged in order to get feature importances of the random forest model. Furthermore, feature importance for the 100 Non-random trees (considering all features at each split) was computed using the described methodology. The results are shown in figure 2.

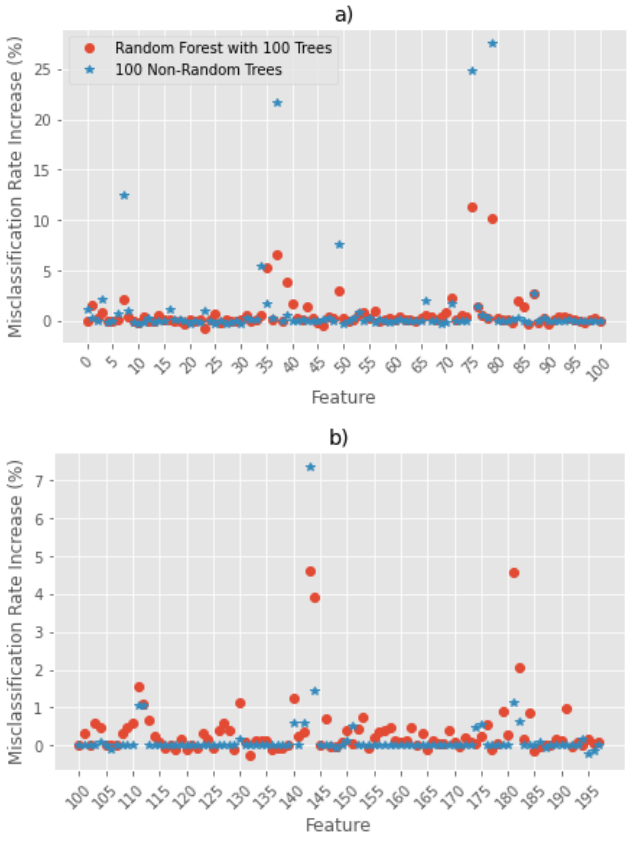

Fig. 2. Feature importance measured as misclassification rate increase for permuted feature of random forest with 100 trees and 100 non-random trees, for: a) features 0 to 100 b) features 101 to 198

Given that our input data set is FTIR spectra, it is expected that the most important features are peaks in this spectrum that appear in different places or have different intensity values for two classes of samples, and on the basis of which we can classify them. Examining feature importance in figure 2, we can see that distribution of important features for random forest better resembles peaks of FTIR spectra. In the case of non-random trees, cetrain features have very high importance in comparison to their importance in random forest. This indicates that the non-random trees are overfitting to those (possibly noisy) features and not considering other relevant features.

REFERENCES

[1] Breiman, L. (2001). Random forests. Machine learning, 45, 5-32.