RANDOM FOREST

STAT598 FIRST REPORT

Wenjie Li
Department of Statistics
li3549@purdue.edu

Evan Li
Department of Statistics
li3550@purdue.edu

Jiajun Liang
Department of Statistics
liang333@purdue.edu

1 Preliminaries and Definitions

An important aspect of machine learning is classification: Given an observation, what class does this observation belong to? This has many real life applications; two of which include predicting the onset of a certain disease in a given patient and predicting whether a particular user will buy a certain product. There exist many classification algorithms, but the one we will work with in this project is the random decision forests algorithm.

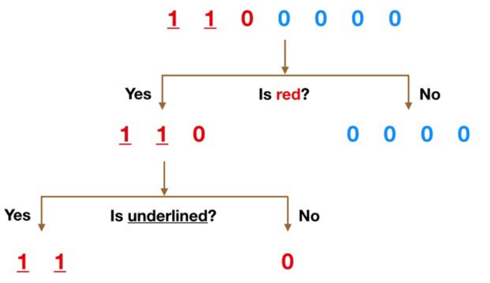


Figure 1: A simple decision tree example

There have been a few algorithms proposed prior to random forests which address classification similarly through an ensemble of trees. Random split selection, proposed by Dietterich in 1998, considers the K best splits at each node and randomly picks one of these splits[1]. Ho in 1998 proposed the "random subspace" method, which randomly selects from a subset of variables/features to use for each tree[2]. Finally, Breiman in the same year proposed a randomization of outputs in the training set, which would generate new training sets[3].

The random forests algorithm, proposed by Breiman in 2001, is an important ensemble machine learning method used for classification, regression, and other tasks, by constructing a number of decision trees at training time and outputting the prediction that is the mode of all predictions (classification) or mean of the individual trees (regression)[4]. For this project, we focus on the classification aspect of the random forests algorithm. In order to discuss the algorithm, we must discuss important preliminaries required to understand the algorithm.

First, we introduce the definition of a decision tree. A decision tree is a relatively intuitive model for data classification, and as to reinforce this intuition, a simple example of a decision tree is shown in Figure 1[5], where numbers with different colors and underlines are classified into two categories in each layer of the tree.

Given any data set D, a decision tree aims to separate the data by looking at one variable or feature of the observations. We formulate the testing of each variable as a binary question/test (T/F or Y/N), and each pair of branches represents one of the results. For example, at the top of the tree we examine the color of each observation in the root node and ask the question, "Is it red?" Any observations which are red then follow the path labeled "YES", and the other observations which are not red follow the other path labeled "NO." A decision tree repeats this process until each set of leaf nodes cannot be split any further or has reached a desired minimum number N_{min}. Note that we do not want to examine the variables arbitrarily, but rather we want to split the data in a way such that *the resulting subgroups after a split are as different from each other as possible, but the observations within any subgroup are as similar to each other as possible.*

How do we go about splitting the data in a way such that the starred condition is satisfied? Given D, we pick the test that results in the highest information (weighted entropy) gain. We then repeat with the newly generated children nodes, until the tree consists only of leaf/terminal nodes. The information gain process is summarized as follows:

Let the decision node have n observations, and suppose a variable test splits the observations into children nodes with n_1 and n_2 observations, respectively, such that $n_1 + n_2 = n$. For the children nodes and the parent node, calculate the entropy of the discrete distribution of observations, and let these entropy values be E_1, E_2, E respectively. We pick the variable test F_{jk} that maximizes

$$E - (\frac{n_1}{n}E_1 + \frac{n_2}{n}E_2).$$

Algorithm 1 Decision Tree Algorithm

```
1: Input: Dataset $D = \{X_i = (\mathbf{F}_i, C_i)\}_{i\in\mathbb{N}}$, where each feature $\mathbf{F}_i = \{(F_{1i}, F_{2i}, \cdots, F_{ni})\}$, minimum size $N_{min}$
2: while node sizes larger than $N_{min}$ do
3:     Calculate the weighted entropy gain of splitting the dataset at each value of each feature.
4:     Get the best split $F_{jk}$ that maximizes the weighted entropy gain.
5:     if the split generates children nodes of valid sizes and it hasn't been chosen then
6:         Accept the split and continue the loop
7: for each leaf node $L_i$ in the tree do
8:     Classify all the data points in $L_i$ to class $C_i = \text{argmax}_{C_k}\{\sum_{j=1}^{|L_i|}\delta(C_j = C_k), X_j \in L_i\}$
```

Above is the pseudocode for generating a decision tree. In order for this algorithm to always work, we must also show that information gain is always nonnegative, and this result is proven in the next section.

A random forest consists of many individual decision trees that act together (thus an ensemble learning method) to obtain better predictive performance when compared to any single decision tree. In other words, a large number of relatively uncorrelated decision trees acting together will be more accurate than any single tree in the group, and this intuitively makes sense as it reduces variance. The trees must have low correlation, so that if any error is made, this error will not be made by all trees in the group. Each individual tree will return a prediction, and the mode of all predictions in the forest is the overall prediction for any unclassified observation. Evidently, with a higher number of trees in the group, the chance of making a wrong classification will decrease provided that the probability of erring is less than $\frac{1}{2}$.

To ensure the trees in an ensemble have low correlation with one another, the bootstrap aggregating (bagging) technique, introduced by Breiman in 1996, is used[6]. The description of the technique is as follows: Given a training set of data T with n observations, a random selection of b observations, with replacement, is made from T. The random samples of the training data will lead to a set of trees that have a low correlation with one another. In addition, because decision trees are very sensitive to the given training set, small perturbations in the training set can lead to a drastic change in the final construction of the tree. The bagging technique also solves this instability and can give major gains in accuracy by reducing variance and avoiding overfitting. The following pseudocode defines the random forest algorithm we implement in this project.

Algorithm 2 Random Forest Algorithm

```
1: Input: Dataset $D = \{X_i = (\mathbf{F}_i, C_i)\}_{i\in\mathbb{N}}$, Training Data $T = \{T_i = (\mathbf{F}_i, C_i)\}_{i\in\mathbb{N}}$, number of trees $N_{tree}$, feature
   fraction $q$
2: for ($i$ in 1:$N_{tree}$) do
3:     Randomly sample $\lceil nq \rceil$ features from the $n$ features
4:     Randomly sample training data with replacement from $T$ to get a $T_{new}$ of size $b$
5:     Generate a new tree by Algorithm 1 using selected features and $T_{new}$
6: for any one data point $X_i$ in $D$ do
7:     Classify $X_i \in C_j$ to be in the class which most trees classify it into
```

2 Basic Analysis and Variants of Random Forests

The first three results can be found in much more detail on the original 2001 Breiman paper on random forests.

- Random forests always converge. Define a margin function m which measures how much the average number of votes for the right class exceeds the average vote for any other class. Thus for a larger m, there will be more confidence in the classification. We next define the generalization error G as the probability that the margin is less than 0, or $G = P(m < 0)$. Then for a large number of trees, for almost surely all trees, G converges to a finite limiting value by the Strong Law of Large Numbers. Due to this result, adding more trees will not cause overfitting.
- An upper bound for G can be derived from two quantities that measure how accurate the individual classifiers are and the dependence between classifiers. More specifically, these quantities are the expectation of the margin function s and the mean correlation $\bar{p}$, and the upper bound is given by

$$G \leq \frac{\bar{p}(1 - s^2)}{1 - s^2}.$$

- Through empirical analysis on data with weak variables (no single variable or small group of variables can distinguish between classes, common in medical diagnosis, etc.), random forests seem to have the ability to work with very weak variables as long as their correlation is low.

We claim that information gain must be nonnegative in the first section. Information gain is defined to be the amount of information gained about a random variable T given some new information from a random variable A. Let $P(T)$ and $P(A)$ denote the marginal PDFs of T and A, and $P(T, A)$ denote the joint PDF of T and A. Let $H(T)$ denote the entropy of T and $I(T, A)$ denote the information gained about T given A. Then, by definition,

$$\begin{aligned}
I(T,A) &= H(T) - H(T|A) \\
&= -\sum_T P(T)\log(P(T)) + \sum_A P(A)\sum_T P(T|A)\log(P(T|A)) \\
&= \sum_A\sum_T P(A)P(T|A)\log(P(T|A)) - \sum_A\sum_T P(T,A)\log(P(T)) \\
&= \sum_A\sum_T P(T,A)\log(\frac{P(T|A)}{P(T)}) = -\sum_A\sum_T P(T,A)\log(\frac{P(T)}{P(T|A)}) \qquad (1) \\
&= -\sum_A P(A)\sum_T P(T|A)\log(\frac{P(T)}{P(T|A)}) \\
&\geq -\log(\sum_A\sum_T(\frac{P(A)P(T|A)P(T)}{P(T|A)} = -\log(\sum_A\sum_T P(A)P(T)) = -\log(1) = 0
\end{aligned}$$

The last line follows from the concavity of the logarithm function and Jensen's inequality. In addition, another way to show information gain is nonnegative is to claim that information gain in the context of machine learning is Kullback-Lieblier divergence, which we have shown to be always non-negative previously in the course. More precisely, let $I(T, a)$ denote the information gained from T given an attribute $A = a$ and $D_{KL}(X||Y)$ denote the KL divergence of the random variable Y from the random variable X. Then the information gain is the KL divergence of the prior distribution from the posterior distribution given a, namely

$$I(T,a) = D_{KL}(P(T|a)||P(T)) \geq 0.$$

Breiman also suggests many variants of the initial algorithm.

- Using random variables/features
  The randomness has to minimize $\bar{p}$ and maintain strength.
- Using random input selection
  At each node, randomly select a small group of variables to split on.
- Using linear combinations of input variables
  Consider the case where there are only a few variables to split on. Let this number be M. If we then take a small group of variables to split on, this fraction may be too large and yield highly correlated trees. In this case, we "define" more variables by taking random linear combinations of a subset of variables. To do so, specify a number $L \leq M$ for the number of random variables to be combined. We add these L variables together, each with a random coefficient uniformly distributed on $[-1, 1]$. We generate F such linear combinations, and then use the best linear combination as the split.
- Categorical variables
  To deal with these, assign numbers 0 or 1 to them depending on a condition on the variable. Breiman suggests that when we need to split on a categorical variable C, we select a random subset of the categories of C, and if the observation has a C value in the subset, we assign it to be 1, and otherwise 0. We employ this strategy in our data set as well (i.e. survival = 1, death = 0), but note this can also be applied to a more complex example (i.e. green or red colors = 1, other colors = 0 for some color variable).

3 Experiments

3.1 Empirical Results

In this section, we implement and test our own decision tree and random forest algorithm on the Titanic dataset. The sinking of the Titanic is one of the most infamous shipwrecks in history. The Titanic, a supposedly unsinkable ocean liner, sank in the early morning of April 15, 1912, and was ill-equipped to save all the lives of the onboard crew and passengers. Obviously, there is some luck involved with survival in this situation, but it does seem that certain groups of people were more likely to survive than others. We use passenger data characteristics such as age, gender, socio-economic class, etc. to predict the survival of the passengers. We first provide a summary of the dataset and the variables we use in the classification analysis in Figure 2a.

```
  PassengerId     Survived Pclass                                     Name         Sex          Age
 Min.   :  1.0   0:549    1:216   Abbing, Mr. Anthony                  :  1   female:314   Min.   : 0.42
 1st Qu.:223.5   1:342    2:184   Abbott, Mr. Rossmore Edward          :  1   male  :577   1st Qu.:20.12
 Median :446.0            3:491   Abbott, Mrs. Stanton (Rosa Hunt)     :  1                Median :28.00
 Mean   :446.0                    Abelson, Mr. Samuel                  :  1                Mean   :29.70
 3rd Qu.:668.5                    Abelson, Mrs. Samuel (Hannah Wizosky):  1                3rd Qu.:38.00
 Max.   :891.0                    Adahl, Mr. Mauritz Nils Martin       :  1                Max.   :80.00
                                  (Other)                              :885                NA's   :177
     SibSp           Parch            Ticket        Fare                Cabin      Embarked
 Min.   :0.000   Min.   :0.0000   1601    :  7   Min.   :  0.00                :687    :  2
 1st Qu.:0.000   1st Qu.:0.0000   347082  :  7   1st Qu.:  7.91   B96 B98    :  4   C:168
 Median :0.000   Median :0.0000   CA. 2343:  7   Median : 14.45   C23 C25 C27:  4   Q: 77
 Mean   :0.523   Mean   :0.3816   3101295 :  6   Mean   : 32.20   G6         :  4   S:644
 3rd Qu.:1.000   3rd Qu.:0.0000   347088  :  6   3rd Qu.: 31.00   C22 C26    :  3
 Max.   :8.000   Max.   :6.0000   CA 2144 :  6   Max.   :512.33   D          :  3
                                  (Other) :852                    (Other)    :186
```

(a) Dataset Summary.

```
##    node observations                 filter  state
## 1     1          891                   <NA> parent
## 2     2           55            Fare >= 93.5   leaf
## 3     3          836             Fare < 93.5 parent
## 4     4          599 Fare < 93.5 & Age >= 12   leaf
## 5     5           65  Fare < 93.5 & Age < 12   leaf
```

(b) Decision Tree Example.

Figure 2: (a). Dataset Summary Information. (b). The Structure of Decision Tree. *Survived* $\sim$ *Age* $+$ *Fare*

As shown in Figure 2a, there are some characteristics that could generate very interesting classification results. For example, do female passengers have a higher chance to survive? Do people who pay more also have a higher chance to survive? To examine the accuracy of these hypotheses, we first implement our own decision tree algorithm as in **Algorithm** 1. We choose entropy as the loss function to minimize, and set the stopping criteria to be $N_{min} \leq \frac{1}{50}N$, where N is the size of the dataset. Using only one decision tree and 2 features "Age" and "Fare", we are able to reach classification accuracy of 0.6425, with the decision tree as in Figure 2b.

```
  node observations                                     filter  state
1    1          891                                       <NA> parent
2    2          271                               Fare >= 27.75 parent
3    3          620                                Fare < 27.75   leaf
4    4          217               Fare >= 27.75 & Fare >= 32.3208   leaf
5    5           54                Fare >= 27.75 & Fare < 32.3208 parent
6    6           34 Fare >= 27.75 & Fare < 32.3208 & Fare >= 30   leaf
7    7           20  Fare >= 27.75 & Fare < 32.3208 & Fare < 30   leaf
```

(a) One Example Tree in the Random Forest.

```
   node observations                                                                    filter  state
1     1          891                                                                      <NA> parent
2     2          540                                                                  Sex >= 2 parent
3     3          351                                                                   Sex < 2   leaf
4     4           22                                                       Sex >= 2 & Parch >= 2   leaf
5     5          518                                                        Sex >= 2 & Parch < 2 parent
6     6           68                                            Sex >= 2 & Parch < 2 & Parch >= 1   leaf
7     7          450                                             Sex >= 2 & Parch < 2 & Parch < 1 parent
8     8          353                              Sex >= 2 & Parch < 2 & Parch < 1 & Embarked >= 4 parent
9     9           97                               Sex >= 2 & Parch < 2 & Parch < 1 & Embarked < 4   leaf
10   10          212                Sex >= 2 & Parch < 2 & Parch < 1 & Embarked >= 4 & Pclass >= 3 parent
11   11          141                 Sex >= 2 & Parch < 2 & Parch < 1 & Embarked >= 4 & Pclass < 3 parent
12   12           22 Sex >= 2 & Parch < 2 & Parch < 1 & Embarked >= 4 & Pclass >= 3 & SibSp >= 1   leaf
13   13          190  Sex >= 2 & Parch < 2 & Parch < 1 & Embarked >= 4 & Pclass >= 3 & SibSp < 1   leaf
14   14           71 Sex >= 2 & Parch < 2 & Parch < 1 & Embarked >= 4 & Pclass < 3 & Pclass >= 2   leaf
15   15           70  Sex >= 2 & Parch < 2 & Parch < 1 & Embarked >= 4 & Pclass < 3 & Pclass < 2   leaf
```

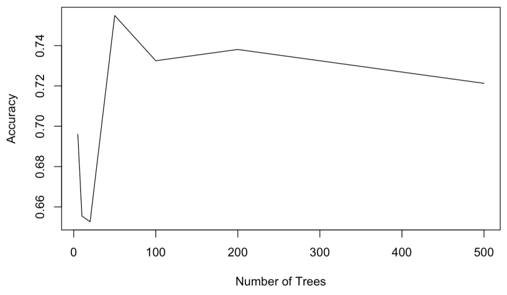
(b) Another Example Tree in the Random Forest.

Figure 3: Decision Trees in the Random Forest

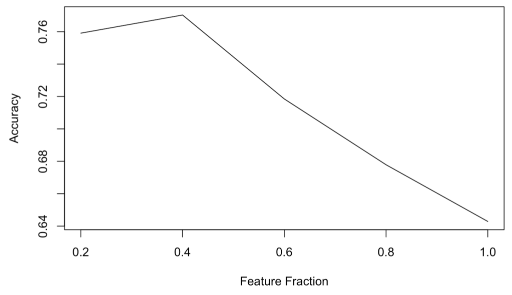
The confusion matrix of our decision tree is provided in Table 1a. Based on the decision tree implementation, we are able to implement our own random forest algorithm as in **Algorithm** 2. We set the number of trees to be 500 as in the R built-in random forest algorithm, and the feature fraction $q = 0.6$. The features which are available in the random selection are "Age", "Fare", "PClass", "SibSp", "Parch", "Sex", "Embarked" in the experiments. Using the bagging and feature selection techniques, we are able to boost the performance of decision tree up to 0.7339 when applying the random forest algorithm. We show two example trees generated by our algorithm in Figures 3a and 3b. Note that in tree 3a, the splits are only values of "Fare" while in tree 3b, multiple features are selected to split the dataset, which proves the existence of random feature selection.

Next, we show how the prediction accuracy changes with different values of the hyper-parameters N_{tree} and q. We first examine the effect of the number of trees in the forest on the prediction accuracy. As can be observed from Figure 4a, although increasing the number of trees does result in higher classification accuracy, its marginal effect becomes smaller when the number becomes larger. Also, a large number of trees makes the calculation process much slower and induces a higher computational cost.

We also investigate the best value of the feature fraction parameter q. By a grid search on the possible values of $q \in (0, 1]$, we can see that middle values $[0.4, 0.6]$ generate the highest accuracy in Figure 4b, which implies that feature selection is very necessary and effective in the random forest algorithm.



(a) Classification Accuracy with Different N_{tree}.



(b) Classification Accuracy with Different q.

Figure 4: Classification Accuracy with Different Hyper-Parameter values.

(a) Confusion Matrix of our Decision Tree Algorithm

	Predicted 0	Predicted 1	Error
True 0	384	42	0.0986
True 1	215	78	0.7338
Total			0.3575

(b) Confusion Matrix of our Random Forest Algorithm

	Predicted 0	Predicted 1	Error
True 0	415	9	**0.0212**
True 1	181	109	**0.6241**
Total			**0.2661**

Table 1: Classification Accuracy of the Decision Tree and the Random Forest Algorithm

3.2 Algorithm modifications

In the Titanic dataset, there exist both quantitative and categorical variables. Quantitative variables are variables which take numerical values and can be summarized by standard mathematical operations. Age and fare are examples of quantitative variables in this dataset, but one problem that may lead to greater inaccuracy in our algorithm is deciding on the number to split on. For example, should we split on arbitrary values such as Fare $= 25$, the mean fare, the mode of the fares, etc.? It follows that it is not efficient to check all the values in the domain of each quantitative variable. Thus whenever we need to check for a split on a quantitative variable Q, we refer to the practice of regression trees and find a critical value $Q = c$ such that when the parent node is divided into two sets of data consisting of observations with $Q < c$ and $Q \geq c$, the information gain for splitting on Q is maximized.

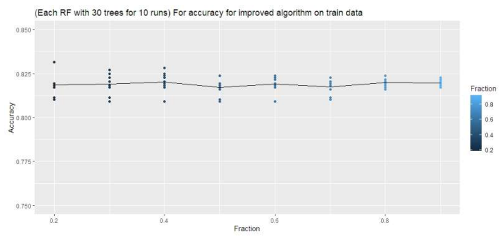
Categorical variables are ones which place an individual into one of several groups or categories, and an example of this variable in our dataset is sex, which only takes two types of values (M/F). Consider a general categorical variable C which takes k types of values $(x_1, x_2, \cdots, x_k)$. Then we calculate the entropy of the k tests or classifications of the form: $(C = x_1 | C \neq x_1), \cdots (C = x_k | C \neq x_k)$, and choose the test that maximizes information gain.

We plot the final accuracy of multiple independent runs with different values of feature fraction in Figure 5a. Using this type of structure involving the two types of variables above results in a >80% survival accuracy compared to the 70% accuracy of the original algorithm (both algorithms used 30 trees), and repeated trials show an 8-10% higher accuracy improvement over the initial unmodified algorithm.

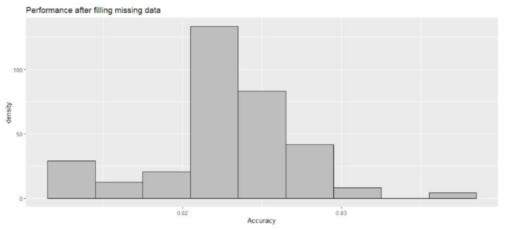
At the same time, we can see that the variance of the accuracy is relatively stable for different feature fractions. As the fraction increases (the proportion of predictors selected increases), the variance of accuracy decreases, which makes sense because the variance of the predictor variables also decreases. As a result, the trees in the forest have generally richer structures and the algorithm is more robust.

3.3 N.A. values

In this dataset, the existence of missing values (N.A.) causes a lot of trouble for programming and prediction. Although our algorithm avoids N.A.s by adding trees and neglecting them in binary classification, we want to see if other approaches are useful to utilize the data with missing values. In the Titanic dataset, there are 177 missing values in the "Age" predictor, and the remaining 714 are known. To use the data with missing values, one way is to modify the calculation of weighted entropy to take into consideration the presence of N.A. values. However, it doesn't make intuitive sense to classify samples with N.A.s into any category. Another method is to use algorithms such as EM and sampling methods to estimate the missing values. Note that the "Age" variable has little connection with other variables, so we can choose the sampling method to construct an empirical distribution of the "Age" variable. We randomly sample from the empirical distribution to fill the 177 missing values and then run the algorithm with the filled training data and evaluate its performance. As shown in Figure 5b, we run the algorithm on the augmented dataset multiple times and the histogram shows that the average accuracy again improves by about 4%.



(a) The improvement after algorithm modification



(b) The improvement after sampling missing values

Figure 5: The improvement after algorithm modification and N.A. value processing.

4 Conclusions

In this report, we briefly introduce the random forest algorithm in a classification task. We first talk about the decision tree algorithm and extend it to the idea of random forest. We then analyze how the algorithm works and what potential variants of the algorithm exist. Finally, we implement our own algorithms in R and evaluate its performance on the Titanic dataset. Our experiments show that the idea of randomization in feature selection and dataset boosting indeed improve the prediction ability of decision trees, and random forest has a strong ability of fitting the dataset and retrieving important information from the features. In the future, we will also apply the random forest algorithm on other different tasks and different datasets to see its generalization and transfer ability.

5 Differences from Proposal and Final Project

In the proposal, we stated these as our objectives, and we discuss below each objective whether we were able to make sufficient progress or if we had to digress due to unseen difficulties.

- Learn more about the background and motivation of random forests by analyzing problems faced in classification algorithms.

  This was covered throughout the preliminary section, where we discuss various problems that can be covered through bagging and random forests. Despite the relative simplicity of the random forests algorithm, it still serves as a surprisingly accurate classifier for the data we examined.
- Study the intuition and the basic ideas of random forests.

  This was also covered extensively in the first two sections.
- Implement random forests in specific data sets to examine its performance, namely the Titanic and MNIST datasets.

  We only had enough time to implement it on the Titanic dataset, but given our classification accuracy, we note random forests does a very good job. Using random forests on the MNIST dataset turned out to be much more complicated, and for the sake of time, we had to forgo analysis for this dataset. The intuition for classification on MNIST would have been to analyze the colors or shading of various regions of pixels, and determining these regions to form the basis for training data was difficult.
- Obtain an understanding of its classification accuracy, its stability when there is missing data or perturbations in the data, and its performance in different types of data. (Is random forests optimal given another type of data set?)

  The first point was discussed throughout the project and the second point was briefly mentioned in the analysis section. In addition, because we were personally not able to examine multiple data sets, we do not have direct results on performance in different types of data. However, in Breiman's 2001 paper, we were still able to gain an understanding of its performance in different types of data.
- Given there is enough time, implement parallel classification algorithms in the same data set to compare the advantages and disadvantages of random forests. Expand on problems with some of our own ideas. Read some of the latest papers about random forests to learn about later developments.

  We were able to come up with some modifications to the algorithm that touched on the problem of arbitrary test selection, and these modifications actually improved the accuracy of our random forests algorithm. Similarly, we were able to improve the accuracy of our algorithm by dealing with the problem of missing data. We also briefly discuss variants of random forests that were developments before Breiman's 2001 paper.

6 Self-Reflection

In the beginning of this project, I planned to participate in both theoretical analysis and empirical application of the random forest algorithm. However, because I didn't have enough time to do both of them, I gave up the theoretical part and let Jiajun and Evan took charge of it while I focused myself on the coding part of the random forest algorithm and successfully implemented our own decision tree and random forest. Even though I didn't fully understand the theoretical background of the algorithm, I believed that I had a good grasp of how decision trees and random forest worked in practice.

The biggest challenge in the final project was that at the beginning of the coding process, I wasn't aware of any data structure in R that could be used to store a tree. I had some experience with object oriented programming, but R seemed to be very different from other languages such as Java or Python. Therefore, I spent days looking for answers to what data structure should be used and how the code should be formatted. Another challenge that I faced was how to communicate and collaborate with my group members because this project was not easy and it involved a heavy amount of coding as well as very theoretical understanding of the algorithm. We were not sure how we could write code or read papers together and we made little progress in the beginning. However, after a thorough discussion, we managed to assign different duties to everyone and finished the project very efficiently.

From this project, I think I have learnt many things. The most important thing is that I have gained a lot of high-level R programming experience, which can be beneficial to my future R coding. Also, I have experienced the benefits and difficulties of scientific collaboration, and it really makes me wonder how I should conduct research and what kind of people I wish to work with in the future.

References

[1] Thomas Dietterich (1998). *An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting and randomization*, Machine Learning, 1–22.

[2] Tin Kam Ho (1998). *The random subspace method for constructing decision forests*. IEEE Trans. on Pattern Analysis and Machine Intelligence, 20(8), 832–844.

[3] Leo Breiman (1998). Randomizing outputs to increase prediction accuracy. Technical Report 518, May 1, 1998, Statistics Department, UCB (in press, Machine Learning).

[4] Leo Breiman (2001). *Random Forests*. Machine Learning (2001) 45: 5. Retrieved from https://doi.org/10.1023/A:1010933404324

[5] Tony Yiu (2001).*Understanding Random Forest: How the Algorithm Works and Why it Is So Effective* Towards Data Science. Retrieved from: https://towardsdatascience.com/understanding-random-forest-58381e0602d2

[6] Leo Breiman (1996). *Bagging Predictors*. Department of Statistics, University of California Berkeley. Technical Report No. 421. Retrieved from https://www.stat.berkeley.edu/ breiman/bagging.pdf