TREE-BASED ENSEMBLE CLASSIFICATION ALGORITHMS FOR GENOMIC DATA

A THESIS

Presented to the Department of Mathematics and Statistics

California State University, Long Beach

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Applied Statistics

Committee Members:

Hojin Moon, Ph.D. (Chair)
Kagba Suaray, Ph.D.
Tianni Zhou, Ph.D.

College Designee:

Tangan Gao, Ph.D.

By Quoc A. Doan

B.S., 2009, University of California, Riverside

May 2020

ABSTRACT

Random Forest is one of the widely used tree-based ensemble classification algorithms. Many aspects of building tree ensembles are introduced to reduce correlation among decision trees within the forest. Bootstrap is used in Random Forest to reduce bias decision tree and to decide splits in every decision tree. Classification by Ensembles from Random Partitions (CERP) is a different algorithm to create an ensemble. CERP randomly partitions the data instead of using bootstrap and creates multiple ensembles instead of one. A forest consists of several decision trees, an ensemble of trees. While Random Forest builds a forest, CERP builds an ensemble of forests. A base classifier in Random Forest uses an exhaustive search to find a split. On the other hand, the Generalized, Unbiased, Interaction Detection and Estimation (GUIDE) algorithm uses statistical hypothesis testing, which is faster than exhaustively search algorithms and is able to detect interaction using a statistical method. This thesis investigated tree-based ensemble classification algorithms that include the CERP, GUIDE, and Random Forest for genetic data.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

BACKGROUND

A microarray is a tool used to parallelly detect gene expression of thousands of genes. The microarray would contain gene chips which are microscope slides with known genes. The DNA within these gene chips act to detect gene expression. Humans in general contain at least 46,831 genes. One individual or observation can produce 46,831 possible gene expressions or predictors (Saey 2018). Microarray experiments generate a large number of features all of which are thousands to millions of genes but very few observations. Data sets having more features than observations is said to be high dimensional data. We used a prostate cancer microarray data set having 20 patients with 22284 probe sets (Best, Gillespie, Yi, et al. 2005).

Microarray data are a challenge for classification. Having a large number of predictors means that logistic regression and many traditional regression algorithms can only use a subset of features to avoid problems such as multicollinearity (where the features are correlated among themselves arises). Tree-based ensemble algorithms may be better at handling large numbers of features. Random subspace method is a method that uses bootstrap, and each decision tree is built on a subset of features (Ho 1998).

Traditional classification algorithms, especially parametric ones, tend to have strict assumptions. It is not feasible to assume normality or some kind of distribution with high dimensional data nor does it make sense; there is just not enough data to assume a distribution. Nonparametric classification algorithms may be more preferable. This type of algorithm require less assumptions and is more flexible with high dimensional data.

With logistic regression and many other regression algorithms, the process of selecting predictors is tedious and somewhat error prone. The Stepwise method for selecting predictors may have disadvantages in a regression model for classification (Flom and Cassell 2007). An alternative to Stepwise may be model averaging or tree-based ensemble algorithms (Flom and Cassell 2007). Tree-based algorithms can handle different types of predictors without introducing a selection bias. Categorical predictors in traditional regression models require dummy variables, in contrast with tree-based predictors do not require any transformation.

Another challenge for high dimensional dataset is the "curse of dimensionality," which is a problem that arises when a dataset has more features than cases (Trunk 1979). The more features a dataset has, the larger the volume of spaces, which requires more observations to have reliable results. As the features increases the volume of space between, these observations grow exponentially therefore requiring a large amount of observations. A problem in high dimensional data is that there are more features than there are observations. The "curse of dimensionality" problem comes into play. A way to handle this "curse of dimensionality" problem is by using random subspace method (Ho 1998).

The data included in this thesis are prostate cancer and myeloma cancer. Prostate cancer is the second most common cases of cancer for men; skin cancer is the most common. The prostate cancer data used in this paper is based on an experiment which is trying to help shed light on a rare and aggressive form of prostate cancer, androgen-independent prostate cancer (Best, Gillespie, Yi, et al. 2005). Prostate cancer is treated using androgen ablation therapy; the form of prostate cancer that recurs after therapy is called androgen-independent prostate cancer.

It is rare for prostate cancer to spread away from prostate organ. Androgen-independent prostate cancer form of cancer rare because it spread toward other regions.

Myeloma cancer is known as multiple myeloma, which is a form of cancer that affects white blood cell in bone marrow. This cancer causes growth of cancerous plasma cells within the bone marrow, which causes tumors and disrupts the production of red blood cell. This form of cancer is hard to detect because symptoms do not appear until the cancer has spread from bone marrow into the neighboring regions.

The myeloma cancer data used in this thesis is based on an experiment by Tian E and et al (Tian and et al. 2003). The team used plasma cells from bone marrow of patients that were diagnosed with myeloma cancer, and also a control group for microarray profiling. The main purpose of their study was to study a specific gene that causes bone lytic lesion. Bone lytic lesion is the softening of bone due to a disease, in this case the disease is multiple myeloma. Myeloma cancer that causes bone lytic lesion has a specific elevated gene known as *dickkopf1*. Myeloma cancer where bone lytic is not present does not exhibit elevation of *dickkopf1* gene.

The results of this thesis may contribute in classifying high dimensional assay data. It may be a better alternative for classifying high-dimensional assay data than Random Forest or XGboost. The results of this thesis may further extend to high-dimensional datasets in a different application instead of high-dimensional assay data set. The results may be expected to give researchers new possible directions in research for tree-based ensemble classification algorithms for high dimensional data. With highly accurate model, the progress may help optimize and improve medical methods, which would lead to a better quality of life for patients.

CHAPTER 2

STATISTICAL THEORY

Random Forest Introduction to CART Decision Tree

A tree can be either used for regression or classification. A decision tree starts with a root node which then splits into other nodes based on some split criteria. From the root node a decision tree can split into several nodes. The nodes that are split from the root node are called the child nodes, and the root node is the parent node of the child nodes. This split process repeats for the root node's child nodes and continues on recursively for the child's child nodes until certain criteria are met. For the purpose of this thesis, we will only discuss the binary decision tree that does classification. Classification is the process of deciding which subpopulation a given new observation belongs to.

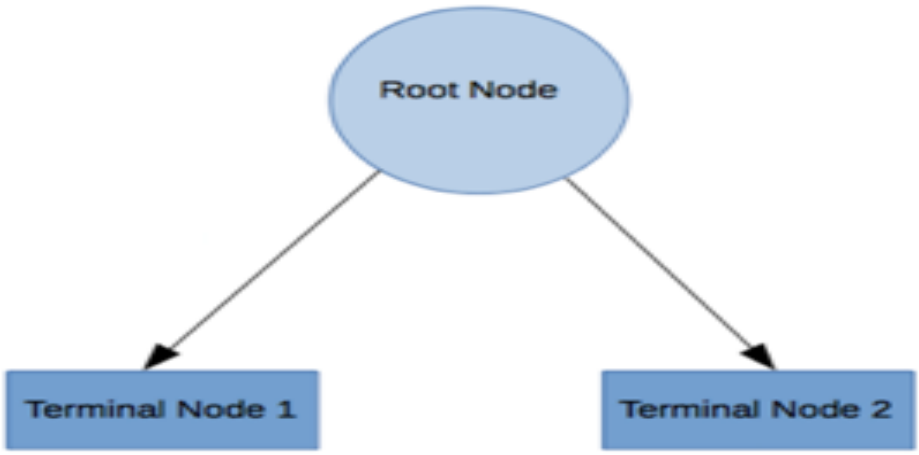

FIGURE 1. An example of a binary decision tree. All nonterminal nodes are represented as circles and terminal nodes are represented as rectangles.

Binary decision trees split into two child nodes by finding a split point within a feature. A split point means that when a node in the binary decision tree splits, it is based on a value of a chosen feature. This split point partitions the data into two partitions and can be seen as two sub

datasets. At the root node we have the entire data set, as the binary decision tree split, the root node will end up with two child nodes. The process of splitting into two child nodes from the root node will partition the original data set into two new partitions, each partitions represents a new separate datasets. The splitting process partitions the given dataset into two sub datasets. One child node will have one of two partitioned datasets and the other child node will have the other partitioned dataset. Taken together the two nodes represent the original dataset from the parent node. Terminal nodes are nodes that have no splits and no child nodes. These terminal nodes will belong to a class. Whatever the most prominent class within the partitioned sub dataset, represented by the terminal node, will be what the binary decision tree will classify. What if the dataset has more than two classes to classify? If the dataset has more than two classes, then the data partition is between one class and the rest of the other classes.

The node split process can be based on a criterion or several criteria. These split criteria are based on an algorithm. Every time a binary classification decision tree splits, the dataset is partitioned into two sub datasets of the previous node dataset. The partitions will be more homogeneous in terms of having more of the same class. The goal of the split criteria algorithm is to partition the data into two partitions representing subsets of data, with each subset having the majority of a particular class. The algorithm chosen to figure out the split criteria will determine the type of decision tree.

Gini impurity is a split criteria in Classification and Regression Tree (CART). The Gini impurity criteria is based on the data impurity. Data impurity measures the homogeneity of a dataset within a tree node. If the data set in a chosen node has only one class of observation, then the Gini impurity score will be zero. An example is a data set in a node that have an evenly

mixed observations of different classes. This example will have a higher Gini score compared to a dataset in a node that contain only observations belonging to one class. Gini impurity measures how homogeneous a dataset in a node is in regard to classification. Gini impurity uses criteria to determine what is the best feature. The feature's value to split the dataset which represent child nodes splitting from the parent node.

Classification and Regression Trees search exhaustively for every combination of features and possible values of features to figure out what is the best split. The best value to split means that the subsets of data are the most homogenous out of all other possible values for chosen feature. The Gini impurity is computed by adding all the probability p_i of an item with label i being chosen, times the probability of misclassifying item $1 - p_i$. The Gini impurity equation is

$$I(p) = \sum_{i=1}^{j} p_i(1 - p_i) = 1 - \sum_{j} p_i^2,$$

where j is the total number of labels that an item can be classified as, p_i is the proportion of items in a dataset with labeled as class i. An example of calculating Gini impurity with two class is shown below.

After establishing how decision trees are created, the next step is describing in general how forest works. More specifically, how a collection of trees collectively works together to arrive at a consensus in classifying a given data.

label 1	0
label 2	10

$$P(label\ 1) = \frac{1}{10}, \qquad P(label\ 2) = \frac{10}{10}$$

$$Gini = 1 - P(label\ 1)^2 - P(label\ 2)^2 = 1 - \frac{0^2}{10^2} - \frac{10^2}{10^2} = 0$$

FIGURE 2. This shows a Gini impurity score of 0. This means that this partition is fully homogeneous. There is no point to partition the data beyond this partition.

label 1	1
label 2	9

$$P(label\ 1) = \frac{1}{10}, \qquad P(label\ 2) = \frac{9}{10}$$

$$Gini = 1 - P(label\ 1)^2 - P(label\ 2)^2 = 1 - \frac{1^2}{10^2} - \frac{9^2}{10^2} = 0.18$$

FIGURE 3. This shows a Gini impurity score of 0.18. This means that this partition is mostly homogeneous. This means that one class, label 2, is overwhelmingly the majority class in this partition.

label 1	5
label 2	5

$$P(label\ 1) = \frac{5}{10}, \qquad P(label\ 2) = \frac{5}{10}$$

$$Gini = 1 - P(label\ 1)^2 - P(label\ 2)^2 = 1 - \frac{5^2}{10^2} - \frac{5^2}{10^2} = 0.5$$

FIGURE 4. This shows a Gini impurity score of 0.50. This means that this partition is not homogeneous. This means that there may be a need to partition this data into sub partitions.

Bootstrap Method

The bootstrap method is a nonparametric statistical data driven method for approximating the sampling distribution that was introduced by Dr. Bradley Efron (Efron 1979). The data driven part is the resampling that bootstrap does. It resamples the original sample to get an approximate of the sampling distribution, which is then used to make statistical inferences. With bootstrap method, resampling is done with replacement. The algorithm for bootstrap method is the following:

1. Choose the sample size for resampling sample.

2. Choose the number of times to resample.

3. For each resampled sample:

    1. Resample with replacement with chosen sample size.

    2. Calculate the statistic on that resample sample.

4. Take all the calculated statistics from the resample samples and calculated the average. Standard error is also calculated and a confidence interval is built around the calculated averaged statistic.

The advantage of bootstrap is that it does not assume a given sample has a specific distribution. An example to highlight this advantage is when we need to infer the population mean from a sample. The one sample *t*-test is often used to do this, inferring the population mean with a confidence interval. The null hypothesis for the *t*-test will be the population mean is equal to some constant. We then find the *t*-value using $t = \frac{\bar{x}-\mu_0}{s/\sqrt{n}}$ where $\bar{x}$ is the sample mean, μ_0 is the

population mean, s is the sample standard deviation, and n is the sample size. We can continue to go on how to build a confidence interval with one sample *t*-test, but this is a good place to stop and highlight why the bootstrap method is better than one-sample *t*-test. The *t*-test assumption about the sample distribution is restrictive, it assume that the sample distribution is standard normal (as seen in the *t*-value equation). If the population true distribution is log normal then the *t*-test assumption that the sample are standard normal does not make sense. This is where the bootstrap method is better. Using the bootstrap method, one can create the sample distribution through resampling, and from there infer the statistic and build the confidence interval. The bootstrap method would better approximate the true population distribution of log normal than the one sample *t*-test.

The bootstrap method is not without flaws. There are two major flaws to the bootstrap method. The first one is that bootstrap make a strong assumption toward what values the sample can have. Since it resample from the original sample, the values in the original sample are the only values that the sample distribution created by bootstrap method can have. Another flaw is that bootstrap method is dependent on the sample size, if the sample size is unusually small the bootstrap method will not work well.

Bagging (Bootstrap Aggregation)

Introduction to Bagging

Bagging is also called bootstrap aggregation and was introduced by Dr. Leo Breiman in 1994 to improve classification (Breiman 1994). The reason to use bagging is to reduce overfitting and to reduce bias. The algorithm for bagging is the following:

1. Take a random sample of the original training set.

2. Train a classifier using the resampled sample.
3. Repeat step 1 and step 2 for a fixed number of times.
4. With a collection of classifiers created from step 3, one can now aggregate all predictions from each classifier and choose the class label that is predicted the most. This process is often called majority voting. Each prediction from a classifier is a vote and the majority vote win.

In step 2 of the bagging algorithm the classifier in this thesis will be decision tree. For Step 4 of the bagging algorithm, often times to break a tie from majority voting, with two labels for classification, an odd number of classifiers are chosen.

Introduction to Random Subspace

The random subspace method was proposed by Dr. Tin Kam Ho (Ho 1998). The method is an attempt to create an ensemble with classifiers having little to no correlation among each other. The main point of the random subspace method is each classifier within an ensemble has its own individual random sample of predictors. The motivation for the creation of this method is to have a perfect accuracy in training data sets and at the same time also have good accuracy for untrained data sets.

Introduction to Random Forest

Random Forest employ bagging and random subspace when building a forest which consists of CART decision trees (Breiman 2001). While bagging is used to reduce bias there are still problems with it. The random sample of training sets in the long run may look very similar and may not be sufficient to reduce correlation among similar classifiers. Random Forest solved this problem by using random subspace on top of bagging in order to make the decision tree

more heterogenous from each other. Let N be the size of training data. Pseudo-code for the random forest,

1. Take a random sample the training data with sample size N.
2. Take a random sample of the predictors without replacement with sample size m, for example $m = \sqrt{p}$, where p represents to total number of predictors in the training data.
3. Create the first decision tree split using CART, partition the data via Gini impurity, for example.
4. The observations that are not used to train because of bootstrap in step 1 are call "out-of-bag" data.
5. Step 1 to step 4 are steps to build decision trees. Repeat this step a number of times 500 to 1000 or more to build a forest.

Generalized, Unbiased, Interaction Detection and Estimation (GUIDE)

Introduction to GUIDE's Decision Tree

For the Random Forest ensemble, it may have a drawback: selection bias. Selection bias is when the tree is biased toward certain types of predictors to split when building a decision tree. Dr. Wei-Yin Loh's findings have shown that CART is biased toward variables with more splits, this means the decision trees are often bigger and require more partitions of data (meaning more nodes in the decision tree); they are biased toward categorical variables (Loh 2010). On the same paper, Dr. Loh also shown that GUIDE trees do not have selection bias. Selection bias can be overcome with many observations. Unfortunately for genetic data and for many medical datasets,

it is generally not the case where there is a large amount of observations to overcome selection bias.

GUIDE decision trees fixed this problem where all predictors have equal probabilities to be selected for tree split. What GUIDE does is to separate the process of finding the best split. GUIDE first select the predictor using two chi-square tests. The chi-square tests are used on main effects and interaction effects between pairs of predictors. From the chi-square tests a predictor is chosen. After a predictor is chosen, GUIDE chooses the best value for that predictor to split using quadratic discriminant analysis. GUIDE stops building decision trees by pruning, using 10-folds cross-validation with 0.5 standard error cutoff as the default value. For a more detailed mathematical implementation interested readers are referred to, "Improving the precision of classification trees" by Dr Wei-Yin Loh (Loh 2009) which describes in detail two modified chi-squared tests, one for main effects and one for interaction effects. The paper then goes into details of how a variable selection and split works based on continuous predictors and categorical predictors.

Classification by Ensembles from Random Partitions (CERP)

Introduction to CERP

Classification by Ensembles from Random Partitions (CERP) is a classification method using random partitions by Dr. Ahn, Dr. Hojin Moon, and et al (Ahn, Moon, Fazzari, et al. 2006). Random partitions here refer to the shuffling of predictors and then partitioning the predictors into disjoint subsets. For each partition a classifier is created. The number of classifiers is dictated by the number of partitions. How the number of partitions is chosen for this paper is through cross validation, the one with the best predicting accuracy from cross validation

is chosen. Majority vote is used for these classifiers for classification. Another way of seeing this is shuffling the order of predictors in the data and randomly sampling the predictors without replacement. Classification by Ensembles from Random Partitions enables the creation of an ensemble of ensembles which is illustrated in the CERP-GUIDE section. The proposed algorithm below will further illustrate how CERP works.

Proposed Algorithm (CERP-GUIDE)

Introduction to CERP-GUIDE

Classification-Tree CERP (C-T CERP) is Dr. Moon's previous work which is an ensemble of ensembles. While CERP is a method for any classifier, C-T CERP was developed specifically for decision trees and forests. C-T CERP is an ensemble of forests where each forest contains CART decision trees with split criteria based on GINI impurity and 10-folds cross validation. Dr. Moon showed that as the value of positive correlation increases among the decision trees, the accuracy of the classifiers decreases (Moon, Ahn, Kodell, et al. 2007). Classification by Ensembles from Random Partitions (CERP) enabled the creation of decision trees that are less correlated compared to other methods such as bagging, boosting, and random subspace.

While C-T CERP addresses the problem of correlation among the decision trees in a forest, it may not need to address the selection bias problem when predictors are chosen to split due to the structure of the dataset. To extend the CERP idea to a general dataset having multiple types of features, we investigated an algorithm that uses GUIDE decision trees as classifiers in the ensemble and uses CERP for data partitions and forest ensembles. Proposed algorithm in thesis will be refer as CERP-GUIDE.

The CERP-GUIDE algorithm first uses CERP to randomize the predictors and then partitions the data set into disjoint subsets. The process of finding the optimal number of partitions will be explained later. From each data partition a GUIDE decision tree is created. This is just one forest. The CERP-GUIDE algorithm repeats this process to create multiple forests. The process of finding the optimal number of forests will be explained later. Classification is carried out by the use of majority voting among the forests. Note that the process of creating GUIDE trees does not employ random subspace, there is no bootstrapping of predictors at every split. What is done here is that the predictors are partitioned into disjoint subsets in CERP.

The process of choosing optimal number of partitions (trees) is as follows:

1. Shuffle the dataset predictor columns once.
2. Let p be the number of partitions. Let p start at 3. 3 is chosen because that is the minimum number for an ensemble.
3. Create GUIDE decision trees out of the partitions according to p, cross validate the forest, and record the performance confusion matrix. An example is $p = 3$, means that that there are only three GUIDE decision trees created since there are three partitions.
4. Increment p by two and repeat step 3 to 4. The forest needs to be odd number in case of tie. Stop this process when p is equal the number of predictors if odds else stop at the largest odd number that is just below the total number of predictors in the dataset.
5. The optimal number of partitions, p, is the one with the best accuracy from cross validation using the training set.

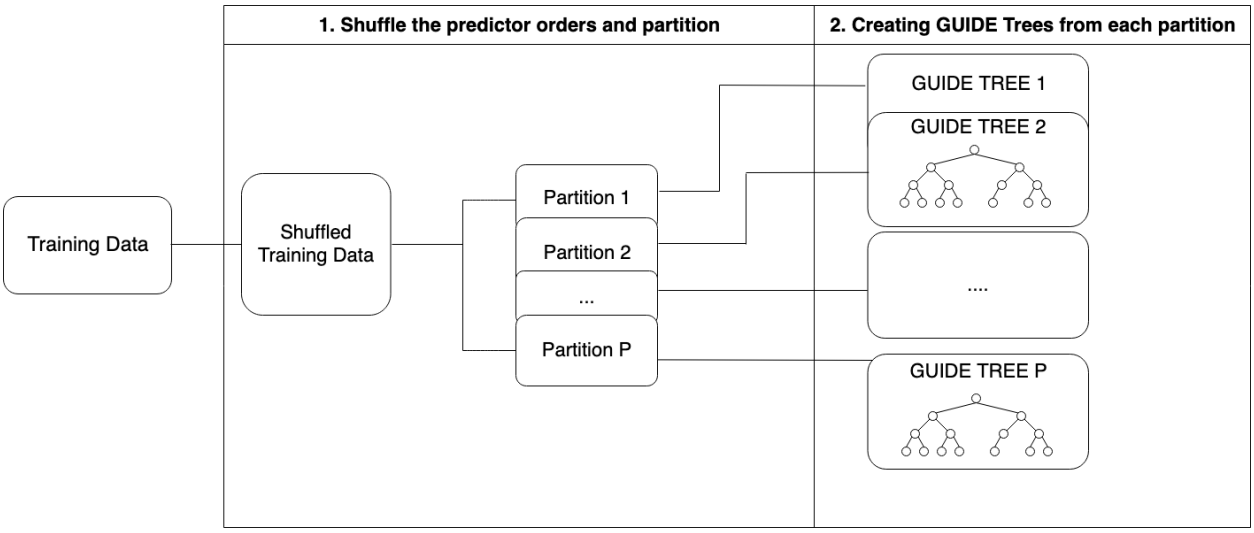

FIGURE 5. CERP-GUIDE process of creating one forest with one shuffled training set.

Once the optimal number of partitions is chosen, the process of finding the optimal number of forests begins as follows:

1. Let op be the optimal number of partitions and f be the current number of forests.
2. The f parameter starts from 1.
3. Build the forest. The number of forests built is equal to f. The forest consists of GUIDE decision trees. The number of GUIDE decision trees in a forest is dictated by op (optimal number of partitions).
4. Record performance of the model using the train set and record the results.
5. Increment f by 2 (we want odd numbers to break any tie voting) and repeat step 3 to step 5. The stop criterion is empirical, from previous works, the suggested is around 19 forests.

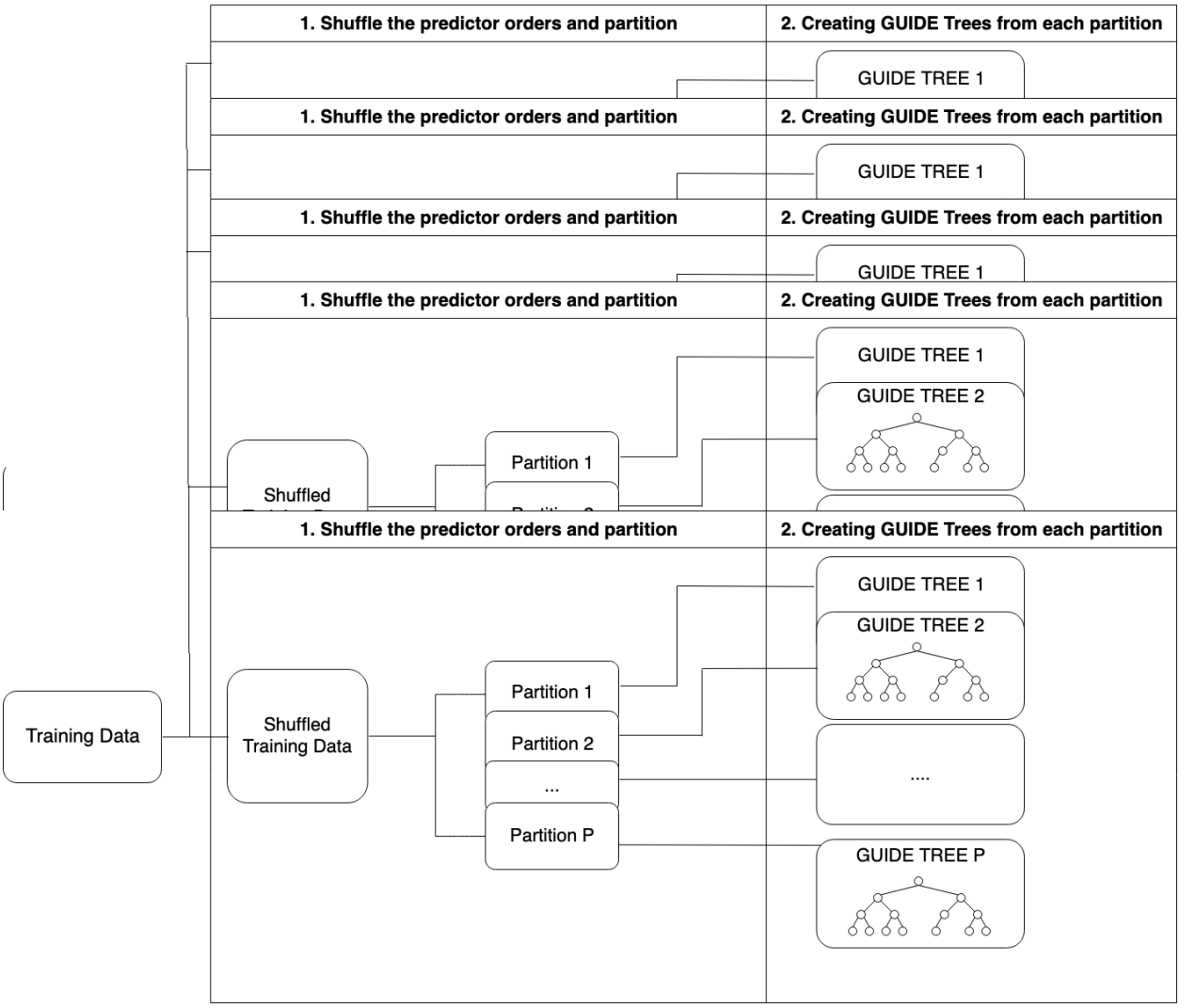

FIGURE 6. CERP-GUIDE process of creating multiple forest with multiple shuffled training sets (one for each forest).

Boosting

Introduction to Boosting

Boosting is an alternative to bootstrapping predictors at every split. Boosting takes weak learning algorithms and adds weights to them to create strong learning algorithms. In the context of decision trees and forest learning algorithms, boosting adds weight to decision

trees according to their performance to increase the overall accuracy of the forest. Boosted trees are trees that have weight added by boosting method.

A classic example of boosting is Adaboost, here the pseudocode is given from Richard A. Berk (Berk 2008. 259).

1. Initialize the observation weights $w_i = 1/N,\ i = 1, 2, \ldots, N$.
2. For $m = 1$ to M:
    a. Fit a classifier $G_m(x)$ to the training data using the weights w_i.
    b. Compute: $err_m = \frac{\sum_{i=1}^{N} w_i\, I(y_i \neq G_m(x_i))}{\sum_{i=1}^{n} w_i}$.
    c. Compute: $\alpha_m = log[(1 - err_m)/err_m]$.
    d. Set $w_i \leftarrow w_i \cdot exp\left[\alpha_m \cdot I\left(y_i \neq G_m(x_i)\right)\right], i = 1, 2, \ldots, N$.
3. Output $G(x) = sign\ [\sum_{m=1}^{M} \alpha_m\, G_m(x_i)]$.

Here N represents the total number of observations or rows in the data. M represents the number rounds of boosting. The $I\left(y_i \neq G_m(x_i)\right)$ function is an indicator function whether the classifier incorrectly predicted or not. If a classifier correctly predicts then the indicator function is 0 and if the classifier incorrectly predicts then the indicator function will set to 1. This means that the weight penalty will be added. The $sign$ in the third step defines the binary class. If the output is larger than 0 then the class will be assigned to the first class and if the output is less than 0 then the class assignment will be the second class.

Extreme Gradient Boost (XGBoost)

Introduction to XGBoost

Before going into detail with XGBoost, we need to note that the Adaboost algorithm cannot be parallelized when building a forest, because the weight for the classifier is updated sequentially. This process is known as "boosting." Random Forest and other forests that use the random subspace method for building trees require no knowledge of how the other decision trees are built. The decision trees can be built independently. For boosting methods such as Adaboost requires knowledge of which tree predicted incorrectly is required to add a weight. This make the process of building trees concurrently is impossible during the boosting phase. However, XGBoost may be parallelized so that XGBoost makes the process faster. In every round of boosting each decision trees may be tested via cross validation before weights are added to individual trees. What is different is how XGBoost calculates the weight and also it adds regularization to penalized complex model. The weights may be updated by training the loss and regularization functions. It may be stated as follows, for example:

$$\mathcal{L}(\phi) = \sum_i l(\hat{y}_i, y_i) \; + \; \sum_k \Omega(f_k)$$

$$\text{where } \Omega(f) = \; \gamma T + \; \frac{1}{2}\lambda \|w\|^2$$

The $\mathcal{L}(\phi)$ is called the regularized learning objective function. The regularization part is $\sum_k \Omega(f_k)$ because it penalizes the complexity of the model. This regularization is for trees and is explained in "Learning nonlinear functions using regularized greedy forest" by T. Zhang and R. Johnson (Zhang and Johnson 2014). The f_k part represents an independent decision tree with k total number of decision trees. The loss function is $\sum_i l(\hat{y}_i, y_i)$, it calculates the difference

between the predicted value of the model, $\hat{y}_i$, with the actual value y_i. Using this equation XGBoost is based on additive training as follows:

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i{}^{(t-1)} + f_t(x_i)\right) + \Omega(f_k)$$

The t here represents the current boosting round. In boosting, the concept of round is where the algorithm adds weight to each decision tree based on which ones predicted correctly and incorrectly. After training the decision trees in the forest, weights may be updated using cross validation. If there is another boosting round after that, the new forest with all the weights added to each decision tree in the previous round will be updated and this process will be repeated.

So $\mathcal{L}^{(t)}$ represents the regularized learning objective for the t-th boosting round. Like previously stated we have the loss function $\sum_{i=1}^{n} l\left(y_i, \hat{y}_i{}^{(t-1)} + f_t(x_i)\right)$ and the regularize penalty function $\Omega(f_k)$. One thing to notice is that XGboost added $f_t(x_i)$ to the predicted value $\hat{y}_i$, this is the function that determines how much weight we need. Recall that $\hat{y}_i$ is the predicted value from the forest containing all of the decision trees. The point of the weights is that we add to the predicted value so we can minimize the loss function. To put it another way, the algorithm is minimizing the distance or differences between the actual value and the predicted value by adding weight to the predicted value.

Using the above equation, the XGBoost algorithm derives the weight for each terminal leaf for each decision tree and the algorithm also derives an equation for the optimal learning objective function. XGBoost does not exhaustively look for every possible combination of split but instead looks at a subset of combinations of split. The split algorithm instead looks at candidate splitting points based on percentiles of feature distribution. For more details, interested

readers are referred to “XGBoost: A scalable Tree Boosting System” by Tianqi Chen and Carlos Guestrin (Chen, Tianqi, and Guestrin 2016). One point to note is that at every split point XGBoost does not use random subspace method, which means there is no random sampling of predictors. Only the commercial version has the random subspace method available under the name TreeNet (www.salford-systems.com/products/treenet). XGBoost also employs several computer optimizations to run the algorithm faster but that is beyond the scope and focus of this paper.

CHAPTER 3

DATA PROCESS

Data Access

There are two cancer data sets which were made accessible through BRB-ArrayTools Data Archive by the National Institutes of Health (NIH). The first is the prostate cancer data set which was made available for general study through the experiment design conducted in the "Molecular alterations in Primary prostate cancer after androgen ablation therapy" paper (Best, Gillespie, Yi, et al. 2005). This data set is available at https://brb.nci.nih.gov/data_archive/GDS1390-Project.zip. The prostate cancer data set has 22,283 genes, 20 observations where 10 observations have androgen-dependent prostate cancer, and the other 10 observations having androgen-independent prostate cancer. The most common prostate cancer treatment is androgen ablation therapy. If the prostate cancer recurs after androgen ablation therapy, then this form of prostate cancer is labeled as "androgen-independent" prostate cancer. Prostate cancer that does not recur after androgen ablation therapy is labeled as "androgen-dependent" prostate cancer.

The second data set is myeloma cancer which was made available for general study through the experiment design conducted in "The Role of the Wnt-Signaling Antagonist DKK1 in the Development of Osteolytic Lesions in Multiple Myeloma" paper (Tian, Zhan, Walker, et al. 2003). This data set is available at https://brb.nci.nih.gov/data_archive/GDS531-Project.zip. The myeloma cancer data set has 12,625 genes, 173 observations. Of the 173 observations, 137 observations classified as "with bone lytic lesion" and only 36 observations classified as "without bone lytic lesion". Bone lytic lesion is a symptom and is the softening of the bone

caused by a variety of diseases. Myeloma cancer is one of the diseases that can cause bone lytic lesion. The high level of dickkopf1 (DKK1) gene with myeloma cancer indicates that the patient most likely has bone lytic lesion problem.

Visualization & Cleaning the Data

All visualizations were done in R using the ggplot2 package (Wickham 2016). All cleaning was done in R using the data frame data structure. There were no missing data in the prostate cancer data set but there were three columns/features which contain missing values in the myeloma cancer data set. Using *t*-test those three columns with missing data were not significant so they were dropped. Both data sets were preprocessed using *t*-test between the response and the features to reduce the number of columns/features using 0.05 significant level. The prostate cancer dataset had the number of features reduced from 22,283 to 1,932. The myeloma cancer data set had the number of features reduced from 12,625 to 1,660.

CHAPTER 4

METHODS

Training Set and Test Set Data

For Prostate data, the data is balanced with 20 observations, with 10 of those observations labeled as “androgen-dependent” and the other 10 observations labeled as “androgen-independent”. The data was split into train set and test set, where 14 observations were set aside for the train set and 6 observations set aside for the test set. Of the 14 observations in the train set, 7 observations were labeled as “androgen-dependent” and 7 other observations labeled as “androgen-independent”. Of the 6 observations in the test set, there were 3 observations labeled as “androgen-dependent” and 3 other observations labeled as “androgen-independent”.

For myeloma data, the data is unbalanced. Of the 173 observations there are 137 observations classified as “with bone lytic lesion” and only 36 observations classified as “without bone lytic lesion”. The original data was split into a train set and a test set. The train set have 40 observations; 20 of those observations are labeled with “with bone lytic lesion” and the other 20 observations labeled “without bone lytic lesion”. The reason why the train dataset is small compared to the test set is that the smallest class, “without bone lytic lesion”, have 36 observations. It was decided that 20 of those observations will be part of the training set and another 20 observations with the class “with bone lytic lesion” will be part of the training set. With this decision there is an equal chance of detecting each class. The test set has 133 observations. Of the 133 observations in the test set, there are 117 classified as “with bone lytic lesion” and 16 observations classified as “without bone lytic lesion”.

GUIDE Decision Tree

There were no parameters of interest for the GUIDE decision tree. The GUIDE decision tree is a single tree so there is no need to find the optimal number of trees in a forest. A single decision tree does not have the problem of correlations among predictors since there is only one predictor, the single decision tree. This means no algorithms are required to reduce correlations among the predictors so random subspace or CERP were not needed. The GUIDE decision tree was built using the train set and the performance was based on the test set. The GUIDE software used was provided by the creator of the GUIDE decision tree Dr. Loh (Loh 2009). The results for the prostate cancer dataset using the GUIDE decision tree was an accuracy of 0.6667, with sensitivity of 0.6667, specificity of 0.6667, positive predictive value of 0.6667, and a negative predictive value of 0.6667. The results for the myeloma cancer dataset using the GUIDE decision tree was an accuracy of 0.3308, with sensitivity of 0.2479, specificity of 0.9375, positive predictive value of 0.9667, and a negative predictive value of 0.1456.

TABLE 1. The Prostate Cancer Confusion Matrix Test Results for the CERP Model

Prostate Cancer	D (Reference)	I (Reference)
D (Prediction)	2	1
I (Prediction)	1	2

TABLE 2. The Multiple Myeloma Cancer Confusion Matrix Test Results for the CERP Model

Myeloma Cancer	W (Reference)	WO (Reference)
W (Prediction)	29	1
WO (Prediction)	88	15

CERP-GUIDE Forest Ensemble

The parameters of interest in our proposed forest ensemble model are the number of partitions, which indictates the number of decision trees, and the other parameter of interest is the number of forests. Since there are two parameters of interest, the process of building the CERP-GUIDE ensemble of forests is broken into two steps. The first step requires finding the optimal number of partitions for one CERP-GUIDE forest. The second step is to find the optimal number of forests in an ensemble. The second step is dependent on the first step because the forests built in the second step use the optimal number of partitions found in the first step.

First Step – Finding the Optimal Number of Partitions for a CERP-GUIDE Forest

What is first needed to build the CERP-GUIDE ensemble of forests is finding the optimal number of partitions for one forest before creating an ensemble of forests. Before we can create an ensemble of forests, multiple forests, we need to figure how to best build one forest. For this thesis the best way to build a forest is by finding the best number of decision trees that a forest should have. Recall that the number of partitions determines the number of decision trees in a CERP-GUIDE forest, this means to build the best forest we need to find the optimal number of partitions for a forest. The optimal number of partitions will dictate how best to build a forest.

The process of finding the optimal number of partitions is brute force. This process builds a CERP-GUIDE forest for all possible odd partitions from a training set. All the forests built from this process will be tested individually against a train set and their performance will be recorded. The optimal partition number will be based on the forest that performs the best on the train set. An odd number of partitions is chosen to break ties if there is a tie between decision trees within a forest.

Since we need to determine the optimal number of partitions, we need to determine how many possible partitions a data set can have. What are the conditions that are imposed upon when building a CERP-GUIDE forest? The first one is to break ties: all CERP-GUIDE forests need to contain an odd number of GUIDE decision trees. The second condition is that the minimum number of decision trees to be consider a forest is 3 trees. The third condition is the maximum number of decision trees that a CERP-GUIDE forest can have.

For many forest algorithms, the maximum number of decision trees a forest can have is not clear and is more empirical based. For the CERP-GUIDE forest, the CERP algorithm gives us a hard number of how many decision trees a CERP-GUIDE forest can have. Recall that CERP algorithm partitions a data set by predictors. This means that the maximum number of partitions for a dataset is equal to the number of predictors. Our third condition is the largest number of trees a forest can have is the largest odd number in the total number of predictors. To clarify, if the dataset has a total of 1986 predictors and the largest odd number of predictors is 1985, then the largest CERP-GUIDE forest will have 1985 decision trees.

The set of all possible CERP-GUIDE forests can be created based on those three previous stated conditions in a forest containing three trees, a forest containing the largest odd number of total number of predictors, and all odd number between that. The first CERP-GUIDE forest to be built consists of three GUIDE decision trees, which means the dataset is partitioned into 3 partitions. The next forest to be built is the next odd number of decision trees, a CERP-GUIDE forest consisting for 5 GUIDE decision trees, which means that the data set is partitioned into 5 partitions. This process continue until the last forest is built.

The CERP algorithm requires the order of predictors to be randomized before partitioning but to find the optimal number of partitions, the same order has to be maintained. The reason why the order of predictors is shuffled before partitioning is to enable an ensemble of forests, which makes each forest within an ensemble different. If the predictors are not shuffled before every partition and before a forest is built, then every forest in an ensemble of forests will be the same and there would be no point of an ensemble.

For the prostate cancer data, there are 1,932 predictors in the prostate cancer data set which means there are 966 ways of partitioning those 1,932 predictors into odd number of partitions. A partition number of one is an odd number and will be excluded since it is not a forest; there are only 965 ways to build a CERP-GUIDE forest for the prostate cancer data with the conditions of this thesis. The CERP-GUIDE forests were built for 965 different odd partitions and their performance measures were recorded for comparison. The best performance measure from the 965 CERP-GUIDE forests will determine the best optimal number of odd partitions. The optimum number of partitions for one forest using the prostate cancer data set is 7 partitions, this means that the optimal forest is one with 7 decision trees with each tree having 276 predictors. The optimal number of partitions is 7 with an accuracy of 1.000, with sensitivity of 1.0000, specificity of 1.0000, positive predictive value of 1.0000, and a negative predictive value of 1.0000.

The same process of finding the optimal number of partitions for prostate cancer was used for the myeloma cancer data set. The myeloma cancer data set have 1660 predictors, so the maximum number of odd partitions is 1659, which means there are 830 ways of partitioning the myeloma data into odd partitions. A partition of one is considered an odd number of partition but

because of the condition set for building a CERP-GUIDE forest it is excluded. There is only 829 ways of building CERP-GUIDE forests. The CERP-GUIDE forests were created from the myeloma train set and all were tested using the train set. The optimal number of partitions chosen was 9. Using 9 partitions as the optimal number of partitions for myeloma cancer gave accuracy of 1.0000 with sensitivity of 1.0000, specificity of 1.0000, positive predictive value of 1.0000, and a negative predictive value of 1.0000.

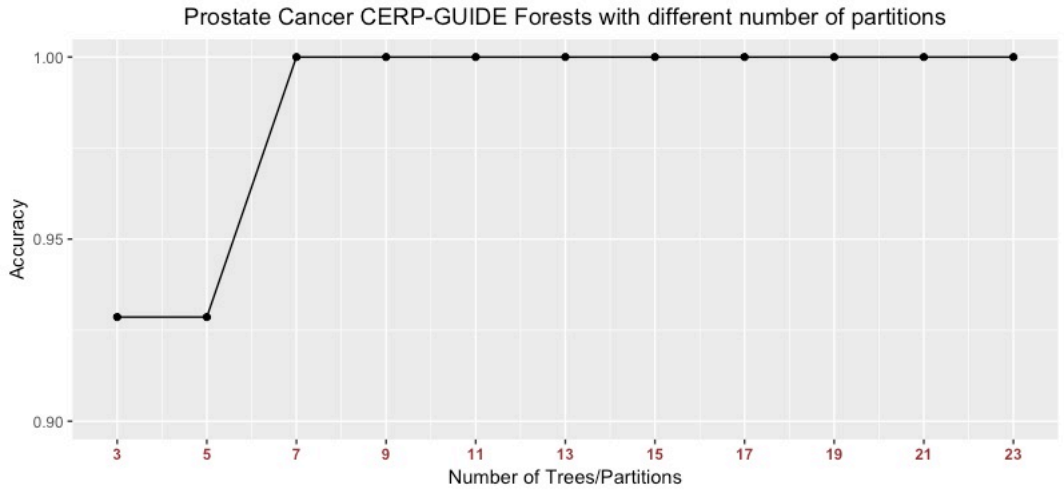

FIGURE 7. This graph shows a subset of the 965 CERP-GUIDE forests to find the optimal number of partitions for prostate cancer. Each point represents a forest, the x-axis shows the number of decision trees within that forest which also represent the number of partitions, and the y-axis represent the accuracy of that forest.

The second step, after finding the optimal way to build a CERP-GUIDE forest, is to find the optimal way to build an ensemble of CERP-GUIDE forests. The optimal way to build an ensemble of CERP-GUIDE forests is to figure out the optimal number forests in an ensemble of CERP-GUIDE forests. The criteria for the optimal number of forests in an ensemble of forests will be based on the performance of a train set.

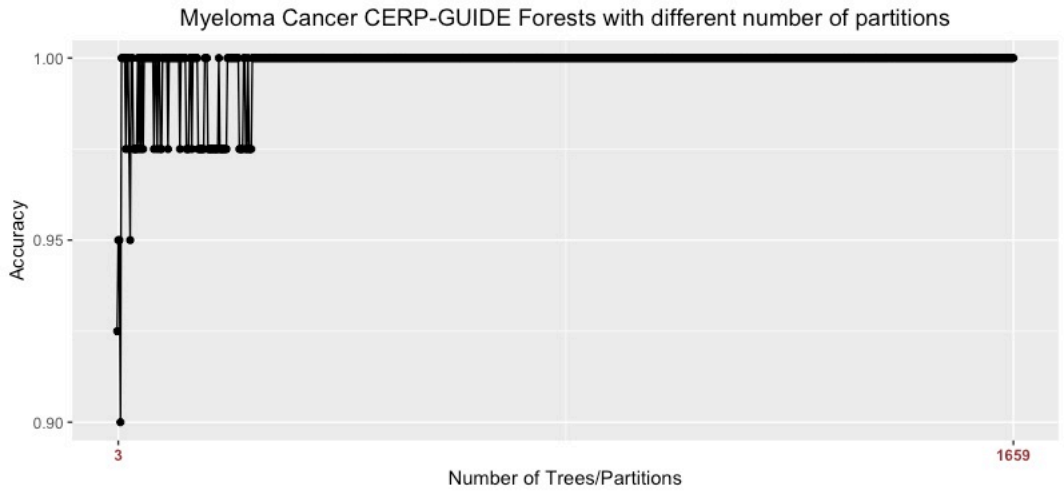

FIGURE 8. This graph shows all 829 CERP-GUIDE forests to find the optimal number of partitions for myeloma cancer. Each point represents a forest, the x-axis shows the number of decision trees within that forest which also represent the number of partitions, and the y-axis represents the accuracy of that forest.

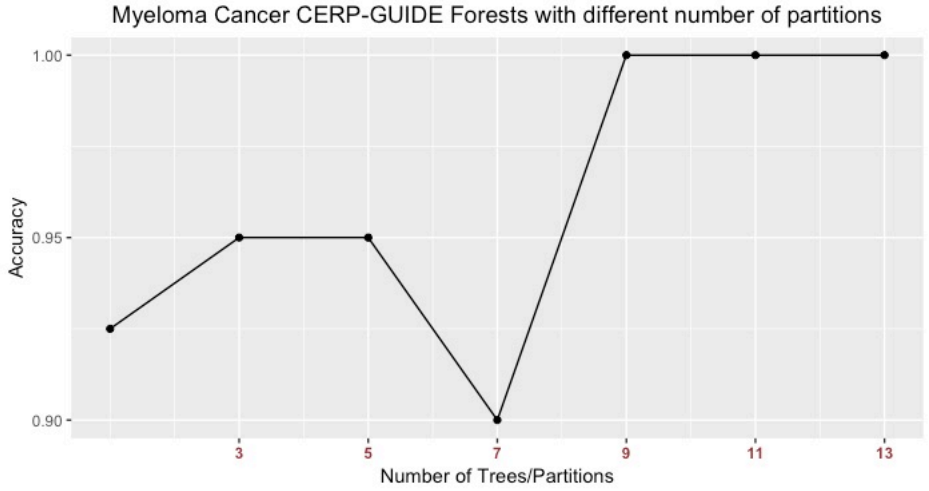

FIGURE 9. This graph shows a subset of 829 CERP-GUIDE forests to find the optimal number of partitions for myeloma cancer. It emphasized on the forests around optimal number of partitions which is 9 partitions.

Second Step – Finding the Optimal Number of Forests in an Ensemble of Forests

The forests that are built in the ensemble of CERP-GUIDE forests uses the optimal way to build a forest in the first step. This means each forest within an ensemble of CERP-GUIDE

forests will be built using the same number of decision trees. This number of decision trees for a forest is the optimal number of decision trees found in step one. Before a forest is built for an ensemble of forests, the order of the predictors is shuffled and then partitioned based on the optimal number of partitions found in step one.

What are the conditions imposed on when building an ensemble of CERP-GUIDE forests? The first condition is there should be an odd number of forests to break ties among the forests. The second condition is an ensemble of forests should have more than one forest but because of the first condition it cannot be two so the minimum number of forests an ensemble of forests can have for our case is three forests. Unlike step one, there is no hard limit on the maximum number of forests an ensemble of forests can have. The maximum number of forests that an ensemble of forests will have is empirically driven.

The ensemble of forests was built starting from three forests using the training set data and the result of the training set was recorded. The next odd number of forests is three, keeping the same forest, two more forests were added to make three forests within an ensemble of forests. The results from the test set were recorded. This process continued, keeping the last three forests, two more forests were added to the ensemble of forests, and the test set results were recorded. The maximum amount of forests for an ensemble of forests was decided based on Dr. Moon's previous works which is around fifteen forests within an ensemble of forests (Moon, Ahn, Kodell, et al. 2007).

For both data sets, prostate cancer and myeloma cancer, there were no advantages in the ensemble of forests. With prostate cancer, regardless of the number of forests in the ensemble of forests, it all had subpar accuracy performances compared to just one CERP-GUIDE forest. For

myeloma cancer, the ensemble of forests does as good as just one CERP-GUIDE forest in accuracy performance.

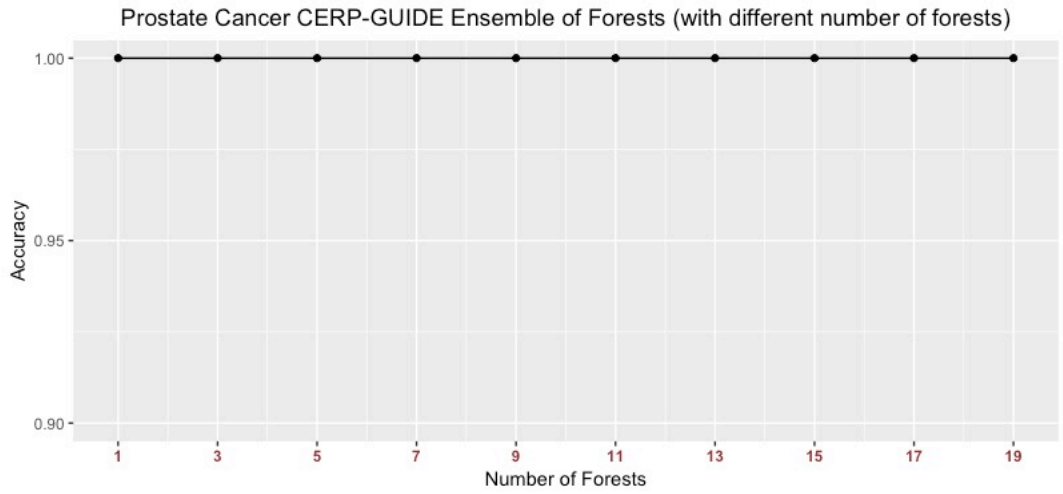

FIGURE 10. This graph shows 19 trained ensembles of CERP-GUIDE forests to find the optimal number of forests for prostate cancer. Each point represents an ensemble of forests, the x-axis shows the number of forests within that ensemble, and the y-axis represents the accuracy of that ensemble of forests.

With the optimal number of partitions found and the optimal number of ensembles found, we now test the models against the test data. For the prostate data set, with optimal number of partitions of 7 and optimal number of ensembles of 1, the performance using the test data is an accuracy of 0.8333 with sensitivity of 1.0000, specificity of 0.6667, positive predictive value of 0.7500, and a negative predictive value of 1.0000. For the myeloma data, set with optimal number of partitions of 9 and optimal number of ensembles of 1, the performance using the test data is an accuracy of 0.7519 with sensitivity of 0.7863, specificity of 0.5000, positive predictive value of 0.9200, and a negative predictive value of 0.2424.

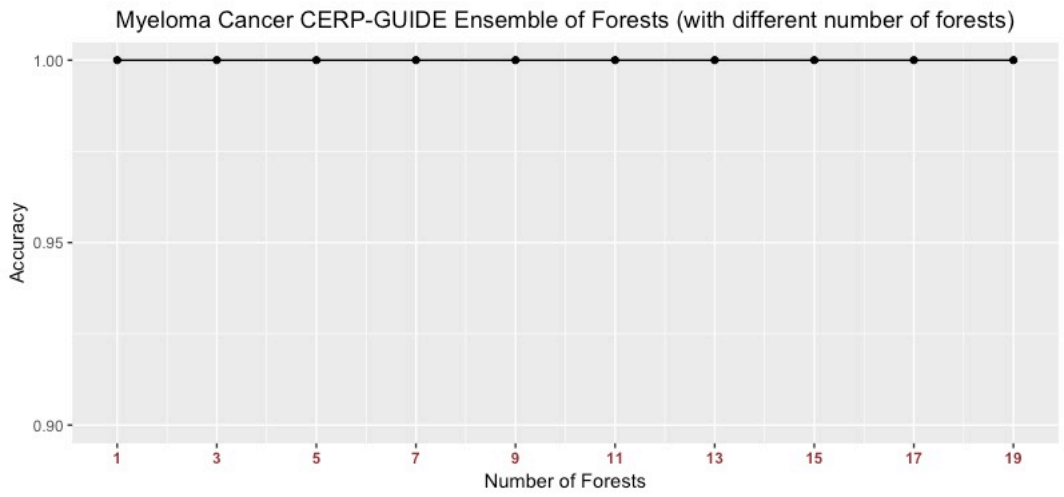

FIGURE 11. This graph shows 19 trained ensembles of CERP-GUIDE forests to find the optimal number of forests for myeloma cancer. Each point represents an ensemble of forests, the x-axis shows the number of forests within that ensemble, and the y-axis represents the accuracy of that ensemble of forests.

TABLE 3. The Prostate Cancer Confusion Matrix Test Results for the CERP-GUIDE Model

Prostate Cancer	D (Reference)	I (Reference)
D (Prediction)	3	1
I (Prediction)	0	2

TABLE 4. The Multiple Myeloma Cancer Confusion Matrix Test Results for the CERP-GUIDE Model

Myeloma Cancer	W (Reference)	WO (Reference)
W (Prediction)	92	8
WO (Prediction)	25	8

CART Decision Tree

There were no parameters of interest in the CART decision tree. The reasons were the same as for the GUIDE decision tree, it is a single predictor so there was no need to be concerned with correlations among predictors, like in algorithm that are an ensemble of

predictors. The package used for creating the CART decision tree is an R package named rpart (Therneau and Atkinson 2018). The results for the prostate cancer dataset using CART decision tree was an accuracy of 0.5000, with sensitivity of 0.5000, specificity of 0.5000, positive predictive value of 0.5000, and a negative predictive value of 0.5000. The results for the myeloma cancer dataset using CART decision tree was an accuracy of 0.3308, with sensitivity of 0.2479, specificity of 0.9375, positive predictive value of 0.9667, and a negative predictive value of 0.1456.

TABLE 5. The Prostate Cancer Confusion Matrix Test Results for the CART Model

Prostate Cancer	D (Reference)	I (Reference)
D (Prediction)	3	3
I (Prediction)	0	0

TABLE 6. The Multiple Myeloma Cancer Confusion Matrix Test Results for the CART Model

Myeloma Cancer	W (Reference)	WO (Reference)
W (Prediction)	29	1
WO (Prediction)	88	15

Random Forest

The parameter of interest in Random Forest is the number of decision trees in the Random Forest. The R package "randomForest" was chosen to build the Random Forest models. All other options are left as default, as a side note, the default number of decision trees for the R package is 500 decision trees (Liaw and Wiener 2002). Since the default for the package is 500 decision trees, it was decided to build Random Forests containing from 3 decision trees to 1000 decision trees and all odd number of trees in between.

Since there is only one parameter of interest in Random Forest, the number of decision trees within a forest, this means there is only one step in finding the best Random Forest. We just need to find the best number of decision trees for one Random Forest. What are the conditions imposed on building a Random Forest? The first condition is a Random Forest can only contain an odd number of decision trees because we do not want any ties between decision trees within the forest. The second is the minimum number of decision trees a Random Forest can have. A forest, at minimum, has more than one decision trees but because of the first condition, the minimum number of decision trees will be three instead of two. With the proposed algorithm CERP-GUIDE ensemble of forests, there is a maximum number of decision trees a forest can have because of the CERP algorithm. In contrast, Random Forest does not have a maximum number of decision trees in a Random Forest, it is empirically driven.

For the prostate cancer data, 2500 forests were trained from the training set containing 3 decision trees to 4999 decision trees. From those 2500 trained Random Forests, the best forest was chosen based on the train set performance. The best Random Forest contained three decision trees. The optimal Random Forest test set performance is an accuracy of 0.6667 with three decision trees, sensitivity of 1.0000, specificity of 0.3333, positive predictive value of 0.6000, and a negative predictive value of 1.0000.

For the myeloma cancer, 1000 Random Forests were built starting from 3 trained Random Forest with one decision trees, all the way up to one Random Forest with 1999 decision trees. Only odd numbers of decision trees were chosen in case of a tie among the decision trees within the Random Forest. All Random Forests were built using the myeloma cancer training data set, and the performance recorded was based on the train set. The best Random Forest was

the one that had the best predictive accuracy using the train set. The best Random Forest had three decision trees with the best accuracy of 0.2782, with sensitivity of 0.1795, specificity of 1.0000, positive predictive value of 1.0000, and a negative predictive value of 0.1429.

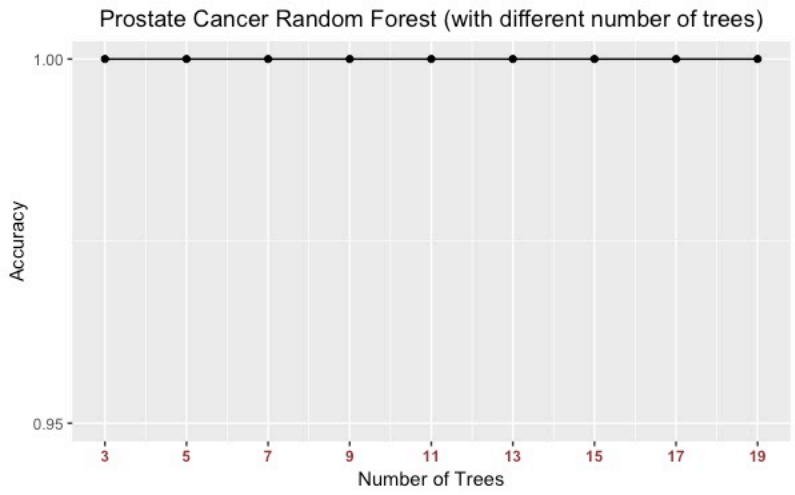

FIGURE 12. This graph only shows 9 trained Random Forests to find the optimal number of trees in a Random Forest for prostate cancer. Each point represents one Random Forest, the x-axis shows the number of trees within that Random Forest, and the y-axis represents the accuracy of that Random Forest.

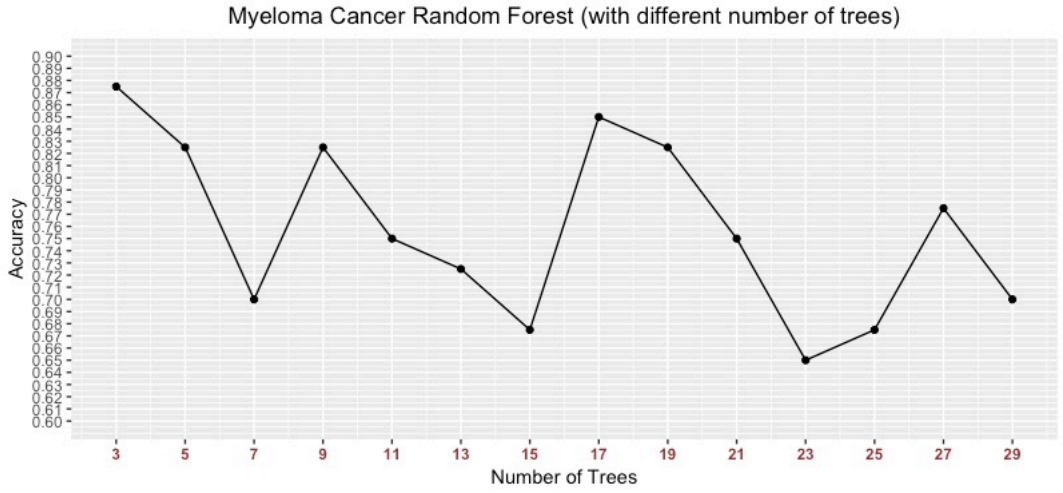

FIGURE 13. This graph only shows 14 trained Random Forests out of 2499 Random Forests to find the optimal number of trees in a Random Forest for myeloma cancer. Each

point represents one Random Forest, the x-axis shows the number of trees within that Random Forest, and the y-axis represents the accuracy of that Random Forest.

TABLE 7. The Prostate Cancer Confusion Matrix Test Results for the Random Forest Model

Prostate Cancer	D (Reference)	I (Reference)
D (Prediction)	3	2
I (Prediction)	0	1

TABLE 8. The Multiple Myeloma Cancer Confusion Matrix Test Results for the Random Forest Model

Myeloma Cancer	W (Reference)	WO (Reference)
W (Prediction)	21	0
WO (Prediction)	96	16

XGBoost

The parameter of interest in XGBoost is the number of boosting rounds while everything else was left at the default. The "xgboost" package in R was used (Tianqi Chen, Tong He, Michael Benesty et al. 2019). It was decided to train twenty models from one to twenty boosting rounds; afterwards, the best performing model was pick from those twenty models. There is no hard number when to stop boosting, rather it is more empirically driven when to stop.

For the prostate cancer data, twenty different XGboost models were created based on the training set. The best XGboost model chosen was based on the best performing one using the prostate train set. The best XGboost model was one with one round of boosting having the best accuracy of 0.6667 with one rounds of boosting, with sensitivity of 1.0000, specificity of 0.3333, positive predictive value of 0.6000, and a negative predictive value of 1.0000.

For the myeloma cancer, twenty XGboost models were created with the first model having one boosting round and the last XGboost model had twenty boosting rounds. All XGboost models were trained using the myeloma cancer training set and performance was based

on the myeloma cancer test set. The best XGboost model was the one with four rounds of boosting with the best accuracy of 0.3383, sensitivity of 0.2650, specificity of 0.8750, positive predictive value of 0.9394, and a negative predictive value of 0.1400.

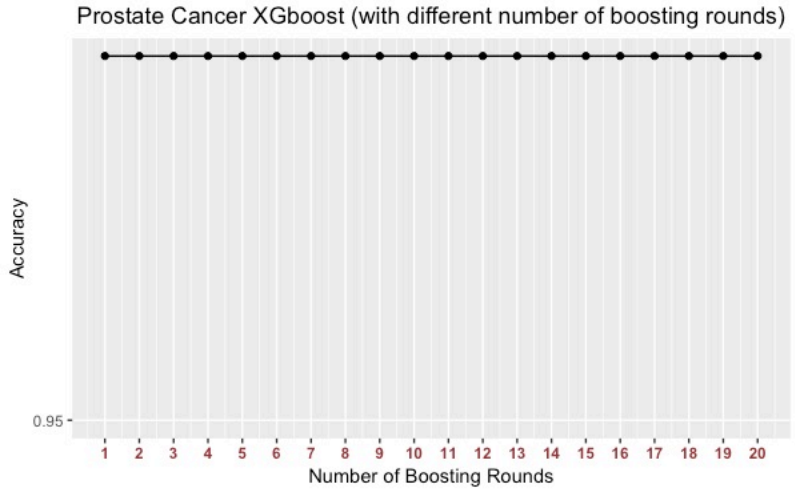

FIGURE 14. This graph shows all 20 trained XGboost forests. This is to find the optimal number of boosting rounds in a XGboost forest for prostate cancer. Each point represents one XGboost forest, the x-axis shows the number of boosting rounds within that XGboost forest, and the y-axis represents the accuracy of that XGboost forest.

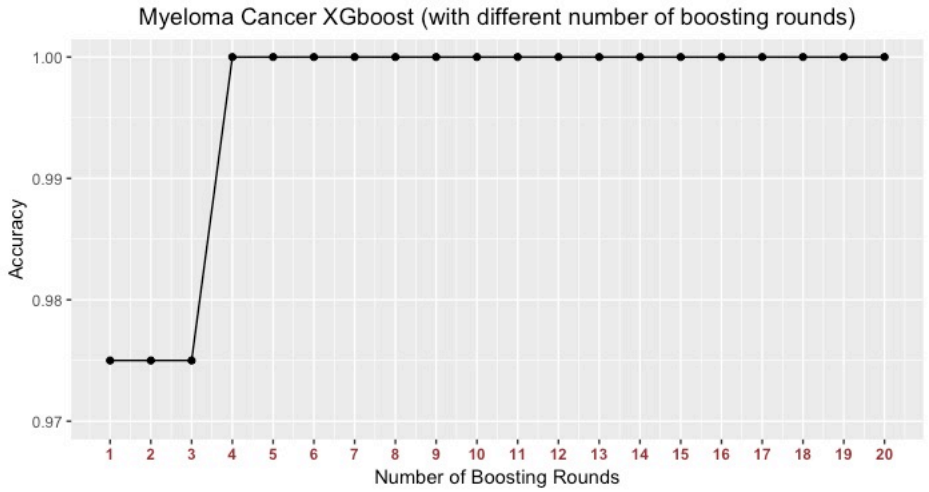

FIGURE 15. This graph shows all 20 trained XGboost forests. This is to find the optimal number of boosting rounds in a XGboost forest for myeloma cancer. Each point represents one XGboost forest, the x-axis shows the number of boosting rounds within that XGboost forest, and the y-axis represents the accuracy of that XGboost forest.

TABLE 9. The Prostate Cancer Confusion Matrix Test Results for the XGboost Model

Prostate Cancer	D (Reference)	I (Reference)
D (Prediction)	3	2
I (Prediction)	0	1

TABLE 10. The Multiple Myeloma Cancer Confusion Matrix Test Results for the XGboost Model

Myeloma Cancer	W (Reference)	WO (Reference)
W (Prediction)	31	2
WO (Prediction)	86	14

CHAPTER 5

COMPARISON

GUIDE Decision Tree vs. CERP-GUIDE Forest

The GUIDE decision tree does much worse than CERP-GUIDE forest for both cancer datasets. For prostate cancer, GUIDE decision tree has an accuracy of 0.6667 compared to CERP-GUIDE forest best accuracy of 0.8333. For myeloma cancer, GUIDE decision tree has an accuracy of 0.3308 compared to CERP-GUIDE forest's accuracy of 0.7519.

CERP-GUIDE Forest vs. Random Forest

The CERP-GUIDE forest does better than Random Forest in accuracy. For prostate cancer, CERP-GUIDE forest's best accuracy is 0.8333 compared to Random Forest's best accuracy of 0.6667. For the myeloma cancer data, CERP-GUIDE forest's best accuracy is 0.7519 compared to Random Forest's best accuracy of 0.2782. The best CERP-GUIDE forests in both datasets have more decision trees compared to best Random Forests for the two datasets.

TABLE 11. Highlighting the Total Number of Decision Trees in the Random Forest Compared to the CERP-GUIDE Forest Requires to Achieve Highest Accuracy for Prostate Cancer

Number of Trees	Random Forest's Accuracy	CERP-GUIDE Forest's Accuracy
3	1.0000	0.9286
5	1.0000	0.9286
7	1.0000	1.0000

The advantages of CERP-GUIDE over Random Forest is that the GUIDE decision trees are free from selection bias (Loh 2010) and that the trees have a much lower correlation among compared to Random Forest. The GUIDE decision tree also looks at interactions to decide how a node within a decision tree splits. In contrast, Random Forest decision trees employ random

subspace to reduce correlation among the trees which is dependent on randomness. This implies that there is a chance that a tree within Random Forest may bootstrap the same set of predictors. The CERP algorithm is not susceptible to such problems, hence the decision trees in CERP-GUIDE are more heterogenous and not as correlated as the decision trees in Random Forest.

TABLE 12. Highlighting The Total Number of Decision Trees in the Random Forest Compared to the CERP-GUIDE Forest Requires to Achieve Highest Accuracy for Myeloma Cancer

Number of Trees	Random Forest's Accuracy	CERP-GUIDE Forest's Accuracy
3	0.8750	0.9250
5	0.8250	0.9500
7	0.7000	0.9500
9	0.8250	1.0000

Another advantage of CERP-GUIDE is the CERP algorithm imposes the maximum number of decision trees a forest can have. With Random Forest, there is no maximum number of trees a forest can have. Recall that in a CERP-GUIDE forest the maximum number of decision trees is dictated by the number of partitions, the maximum number of partitions is when it is equal to the total number of predictors in the data set. In the CERP-GUIDE forest, you cannot partition past the total number of predictors available in a dataset, meaning there is a hard ceiling to the number of decision trees a CERP-GUIDE forest can have. An algorithm can exhaustively train all forests with all possible numbers of trees to find the best number trees in a forest.

The third advantage is that the CERP algorithm is not constrained to one forest, it can be an ensemble of forests whereas Random Forest is generally used as one forest. Multiple Random Forests can be ensembled together, but Random Subspace and the bootstrapping of observations are more prone to correlation among the decision trees and forests compared to an ensemble of CERP-GUIDE forests (Loh 2010) (Moon, Ahn, Kodell, et al. 2007). Empirically, the advantage

of an ensemble of forests does not show up in any of the two data sets. None of the ensembles of CERP-GUIDE forests did better than a single CERP-GUIDE forest.

CERP-GUIDE Forest vs. XGBoost

For both cancer datasets, CERP-GUIDE does better than XGBoost. The XGBoost's accuracy of 0.6667 for prostate cancer is the same as Random Forest's accuracy and is worse than CERP-GUIDE's accuracy of 0.8333. For the myeloma cancer data set XGBoost's accuracy is 0.3383 which is just a bit better than Random Forest but worse than CERP-GUIDE's accuracy of 0.7519. The number of trees in XGBoost is set at the default value of 100 decision trees compared to CERP-GUIDE decision trees.

The CERP-GUIDE forest has several advantages compared to XGBoost. The first being that CERP-GUIDE forest is not a boosting algorithm so in theory the algorithm should be more interpretable. There is no real interpretation for the weights that are added to each tree that XGBoost employed. The XGBoost forest only interpretation is that the decision tree is this far away from correctly predicting. There are several papers on the theory of Random Forest, all of them have to restrain Random Forest's greedy algorithm and bootstrapping for interpretation (Denil, Matheson, and Freitas 2014). For CERP-GUIDE there is no random element in the bootstrapping predictor at every split level which in theory would make it easier to interpret. The only random element would be the shuffling of predictors in the CERP algorithm part which is employed once.

Another advantage of CERP-GUIDE forest is the parameter of interest of CERP-GUIDE is predictable, there is a limit on how many decision trees there are. The XGBoost's parameter of interest is the number of boosting rounds which is arbitrary, there is no ceiling on when to stop.

A potential drawback is that XGBoost requires categorical response to be converted to continuous response.

Final Results for Prostate Cancer

TABLE 13. Summary of Accuracy, Sensitivity, and Specificity Performance Among the Different Tree-Based Algorithms for the Prostate Cancer Dataset

Model	Parameter	Accuracy	Sensitivity	Specificity
CERP-GUIDE	Number of Partition (Number of Trees)	0.8333	1.0000	0.6667
Random Forest	Number of Trees	0.6667	1.0000	0.3333
XGBoost	Number of Boosting Rounds	0.6667	1.0000	0.3333
GUIDE		0.6667	0.6667	0.6667
CART		0.5000	0.5000	0.5000

TABLE 14. Summary of Positive Predictive Values, Negative Predictive Values, and Area Under the Curve Performance Among the Different Tree-Based Algorithms for the Prostate Cancer Dataset

Model	Parameter	PPV	NPV	AUC
CERP-GUIDE	Number of Partition (Number of Trees)	0.7500	1.0000	0.8333
Random Forest	Number of Trees	0.6000	1.0000	0.6667
XGBoost	Number of Boosting Rounds	0.6000	1.0000	0.6667
GUIDE		0.6667	0.6667	0.6667
CART		0.5000	0.5000	0.5000

Final Results for Myeloma Cancer

TABLE 15. Summary of Accuracy, Sensitivity, and Specificity Performance Among the Different Tree-Based Algorithms for the Myeloma Cancer Dataset

Model	Parameter	Accuracy	Sensitivity	Specificity
CERP-GUIDE	Number of Partition (Number of Trees)	0.7519	0.7863	0.5000
Random Forest	Number of Trees	0.2782	0.1795	1.0000
XGBoost	Number of Boosting Rounds	0.3383	0.2650	0.8750
GUIDE		0.3308	0.2479	0.9375
CART		0.3308	0.2479	0.9375

TABLE 16. Summary of Positive Predictive Values, Negative Predictive Values, and Area Under the Curve Performance Among the Different Tree-Based Algorithms for the Myeloma Cancer Dataset

Model	Parameter	PPV	NPV	AUC
CERP-GUIDE	Number of Partition (Number of Trees)	0.9200	0.2424	0.7519
Random Forest	Number of Trees	1.0000	0.1429	0.2782
XGBoost	Number of Boosting Rounds	0.9394	0.1400	0.3383
GUIDE		0.9667	0.1456	0.3308
CART		0.9667	0.1456	0.3308

Chapter 6

CONCLUSION

Empirically, the CERP-GUIDE forest seems to work well for high-dimensional data. It is a compelling choice since it does not have selection bias when creating decision tree. The trees within the CERP-GUIDE forest are less correlated among each other compared to other tree-based algorithms. For future work, the proposed algorithm can adapt a faster prune algorithm. The creators of XGboost have put in a lot of work to optimize for faster run times. The XGboost forest takes into account how the data are stored in computer memory and optimize for memory access speed to decrease the time it takes to train a XGboost model. It also takes into account CPU cache to further reduce the time it takes to train a XGboost model. For future work, perhaps CERP-GUIDE can adopt some of those techniques to reduce the time it takes to train a model. One thing that CERP-GUIDE cannot leverage that XGboost uses is tree regularization because it depends on additive model building which boosting can utilize. Additive model building in this context means that the trees in XGboost are added sequentially, where as CERP-GUIDE are creating the trees concurrently, all at once. Tree regularization fits well in the boosting model framework.

REFERENCES

REFERENCES

Ahn, Hongshik, Hojin Moon, Melissa J. Fazzari, Noha Lim, James J. Chen, Ralph L. Kodell. 2007. “Classification by Ensembles from Random Partitions of High-dimensional Data.” *Computational Statistcs and Data Analysis* 51, no. 12: 6166-179.

Berk, Richard A. 2008. Statistical Learning From a Regression Perspective. Springer Series in Statistics. New York, NY: Springer New York.

Best, Carolyn J. M., John W. Gillespie, Yajun Yi, Gadisetti V.R. Chandramouli, Mark A. Perlmutter, Yvonne Gathright, Heidi S. Erickson, *et al*. 2005. "Molecular Alterations in Primary Prostate Cancer after Androgen Ablation Therapy." *Clinical Cancer Research* 11, no. 19 (October): 6823-34.

Breiman, Leo. 1996. “Bagging predictors.” *Machine Learning* 24, no. 2 (1996): 123-40.

Breiman, Leo. 2001. “Random forests.” *Machine Learning* 45, no. 1 (2001): 5-32.

Chen, Tianqi and Carlos Guestrin. 2016. “XGBoost: A Scalable Tree Boosting System.” Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 13-17, 2016.

Chen, Tianqi, Tong He, Michael Benesty, Vadim Khotilovich, Yuan Tang, Hyunsu Cho, Kailong Chen, et al. 2019. "xgboost: Extreme Gradient Boosting." *R package* version 0.81.0.1. https://CRAN.R-project.org/package=xgboost

Denil, Misha, David Matheson, and Nando De Freitas. 2014. "Narrowing the Gap: Random Forests In Theory and In Practice." Paper presented at the 31st meeting of the International Conference on Machine Learning, Beijing, China, June 21-26, 2014.

Efron, Bradley. 1979. “Bootstrap Methods: Another Look at the Jackknife.” *The Annals of Statistics*, 7, no. 1, 1--26. doi:10.1214/aos/1176344552.

Flom, Peter L. and David L. Cassell. 2007. “Stopping Stepwise: Why Stepwise and Similar Selection Methods are Bad, and What You Should Use.” Paper presented at the 2007 meeting of the North East SAS Users Group, Baltimore, November 11-14, 2007. https://support.sas.com/resources/papers/proceedings/pdfs/sgf2008/361-2008.pdf

Ho,Tin Kam. 1998. "The Random Subspace Method for Constructing Decision Forests." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20, no. 8: 832-44.

Johnson, Rei and Tong Zhang. 2014. "Learning Nonlinear Functions Using Regularized Greedy Forest." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36, no. 5: 942-54.

Kleinberg, Eugene. M. 1990. “Stochastic Discrimination.” *Annals of Mathematics and Artificial Intelligence* 1:207-39

Liaw, Andy and Matthew Wiener. 2002. "Classification and Regression by randomForest." *R News* 2, no. 3: 18-22. https://www.r-project.org/doc/Rnews/Rnews_2002-3.pdf

Loh, Wei-Yin. 2009. “Improving the Precision of Classification Trees.” *The Annals of Applied Statistics* 3, no. 4:1710-37.

Loh, Wei-Yin. 2010. “Tree-structured Classifiers.” *Wiley Interdisciplinary Reviews: Computational Statistics* 2, no. 3:364–69.

Moon, Hojin, Hongshik Ahn, Ralph L. Kodell, Songjoon Baek, Cheil Jedang Lin, James J. Chen. 2007. “Ensemble Methods for Classification of Patients for Personalized Medicine with High-Dimensional Data.” *Artificial Intelligence in Medicine* 41:197-207.

Saey, Tina Hesman. 2018. “A Recount of Human Genes Ups the Number to at Least 46,831.” *Science News*, October 13, 2018. https://www.sciencenews.org/article/recount-human-genes-ups-number-least-46831

Therneau, Terry and Beth Atkinson. 2018. “rpart: Recursive Partitioning and Regression Trees.” *R package* version 4.1-13. https://CRAN.R-project.org/package=rpart

Tian Erming, Fenghuang Zhan, Ronald Walker, Emilie Rasmussen, Yupo Ma, Bart Barlogie, John D. Shaughnessy, Jr. 2003. “The Role of Wnt-signaling Antagonist DKK1 in the Development of Osteolytic Lesions in Multiple Myeloma.” *The New England Journal of Medicine* 349*,* no. 26:2483-494.

Trunk, G. V. 1979. "A Problem of Dimensionality: A Simple Example." *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-1, no. 3 (July): 306-307. doi:10.1109/TPAMI.1979.4766926

Wickham, Hadley. 2009. *Ggplot2: Elegant Graphics for Data Analysis.* Springer. New York, NY: Springer New York.