Tree Guided Personalized Machine Learning Prediction With Applications To Precision Diagnostics

by

Nishtha Shah

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved October 2023 by the
Graduate Supervisory Committee:

Yunro Chung, Co-Chair
Kookjin Lee, Co-Chair
Hassan Ghasemzadeh

ARIZONA STATE UNIVERSITY

December 2023

ABSTRACT

The proposed research is motivated by the colon cancer bio-marker study, which recruited case (or colon cancer) and healthy control samples and quantified their large number of candidate bio-markers using a high-throughput technology, called nucleic acid-programmable protein array (NAPPA). The study aimed to identify a panel of biomarkers to accurately distinguish between the cases and controls. A major challenge in analyzing this study was the bio-marker heterogeneity, where bio-marker responses differ from sample to sample.

The goal of this research is to improve prediction accuracy for motivating or similar studies. Most machine learning (ML) algorithms, developed under the one-size-fits-all strategy, were not able to analyze the above-mentioned heterogeneous data. Failing to capture the individuality of each subject, several standard ML algorithms tested against this dataset performed poorly resulting in 55-61% accuracy. Alternatively, the proposed personalized ML (PML) strategy aims at tailoring the optimal ML models for each subject according to their individual characteristics yielding best highest accuracy of 72%.

ACKNOWLEDGMENTS

I would like to thank my mentor, who is also one of the co-chairs for my defense committee Dr. Yunro Chung for all the support, guidance, and motivation. Also, thank you for having me as a part of your team and introducing me to this study which became motivation for my thesis.

I would also like to thank Dr. Kookjin Lee and Dr. Hassan for agreeing to serve as co-chair and committee member for my thesis defense respectively and for believing in my work.

A huge thanks to Arizona State University, for providing me this chance to work on such a project and providing me a platform where I can defend my research. Also, this research would not have been possible without the computational resources made available by the University.

Lastly, a huge thanks to my family and friends who have constantly supported and motivated me and have had faith in me.

TABLE OF CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

Chapter 1

INTRODUCTION

1.1 Motivation

Our research is motivated by the colon cancer bio-marker study, which recruited 599 case (or colon cancer) and 599 healthy control samples and quantified their large number of candidate biomarkers using a high-throughput technology, called nucleic acid-programmable protein array (NAPPA) (Díez, 2015). The study aimed to identify a panel of biomarkers to distinguish between the cases and controls accurately. A major challenge in analyzing this study was the bio-marker heterogeneity, where bio-marker responses differ from sample to sample. As displayed in fig. 1.1, the top one bio-marker had a low area under the receiver operating characteristics (ROC) curve (AUC) value of 0.61 because most of the case and control samples had similar values between 0 and 10, even if the other few cases had values substantially higher than the others. The other biomarkers showed similar patterns with different responders. We used several (supervised) machine learning (ML) algorithms to combine these biomarkers linearly or non-linearly but failed to improve classification accuracy.

The goal of this paper is to improve prediction accuracy for motivating or similar studies. Most ML algorithms, including those used in fig. 1.1, have been developed under the one-size-fits-all strategy, assuming there is a common set of variables that can accurately predict outcomes for all subjects. However, as demonstrated in the data analysis above, this strategy is limited to analyzing heterogeneous data because there are no such one-size-fits-all set of variables. Rather, each subject may have a unique set of variables that can be used to predict his or her outcome, implying personal-

izing or customizing the ML algorithm would improve prediction accuracy. In order to achieve that, inspiration was drawn from the medicinal concepts of personalized medicine (PM) and personalized diagnostics (PD) (McAlister, 2017). personalized medicine aims to improve health outcomes by leveraging patient heterogeneity and tailoring treatment to individual patients, e.g. selecting treatment A versus B, based on patients' profiles, and personalized diagnostics is a sub-category of PM that aims to improve disease diagnostics by tailoring biomarkers (medical tests or risk scores) to each individual ideally before any treatment or clinical symptoms begin. With the combination of this concept with standard ML algorithms, the proposed tree-guided personalized ML (PML) strategy aims at tailoring the optimal ML algorithms to each subject according to their individual characteristics, including, but not limited to, lifestyle, genetic, and environmental factors. Fig. 1.2 is an example of the PML strategy, where subjects are separated into three subgroups and ML algorithms are tailored for each subgroup.

1.2 Relation To Other Techniques

There are various ML algorithms for personalization and recommendation systems (McAuley, 2022). Some of these methods could be used for the PML strategy, but they were primarily based on unsupervised learning techniques and not necessarily to provide optimal solutions. For example, a clustering algorithm is used to find some subgroups, and the best ML algorithm is selected for each of the subgroups. However, because the two algorithms are performed independently, the selected ML algorithm may not be optimal, even if the clustering algorithm performs perfectly. To tackle the challenge, we propose a novel tree-guided PML by adding a specific ML algorithm at each terminal node. Using ideas similar to the classification and regression tree (CART) method (Breiman, 2017), we use the recursive approach to estimate the optimal decision tree and terminal-node specific ML algorithm simultaneously.

The closest related work was the treed regression model (Alexander and Grimshaw, 1996) by fitting a simple linear regression model at each node, and the model was further extended to use piece-wise or kernel regressions with multiple covariates (Wang and Witten, 1996; Torgo, 1997) or Bayesian linear regression with probabilistic tree structure (Chipman *et al.*, 2002). However, all these models had two limitations: (a) a small number of covariates were considered at each node, and (b) the regression model had exact same structure, e.g. linear regression, across all nodes. The proposed method is more flexible because it allows high-dimensional variables without any linearity or underlying modeling assumptions.

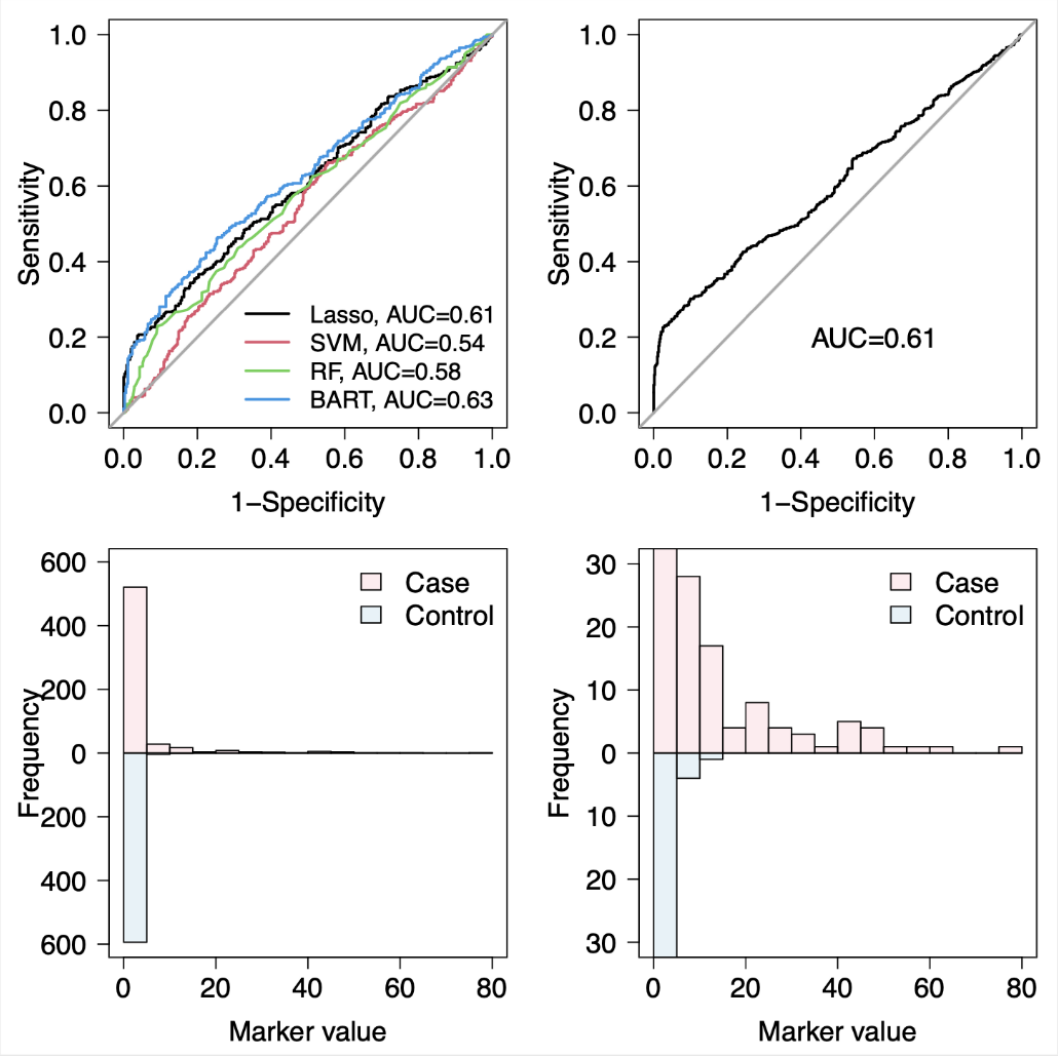


Figure 1.1: Colon Cancer Data Analysis. Upper: Test ROC Curve For Lasso Regression (Tibshirani (1996)), Support Vector Machine (Hearst *et al.* (1998)), Random Forest (Breiman (2001)), And Bayesian Additive Regression Tree Algorithms (Chipman *et al.* (2010)) (Left), And Fitted ROC Curve For The Top One Bio-marker (Right). Below: Histogram For The Top Bio-marker With The Entire Range Of Y Value (Left) And Y Value Between 0 And 30 (Right).

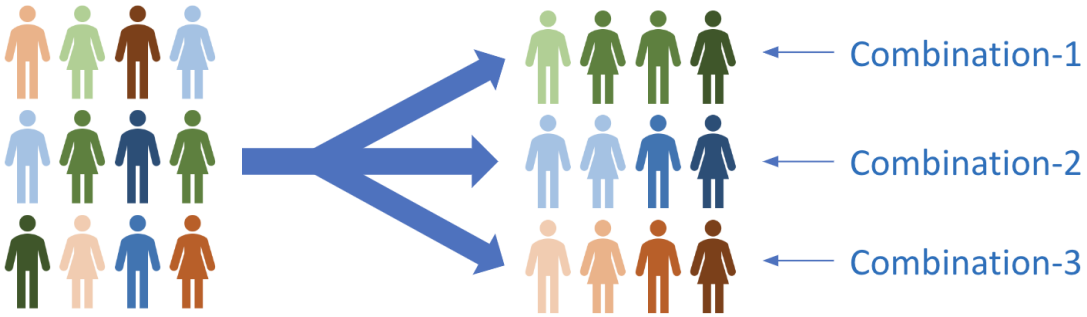


Figure 1.2: Tailoring ML Algorithms To Groups With Similar Patterns

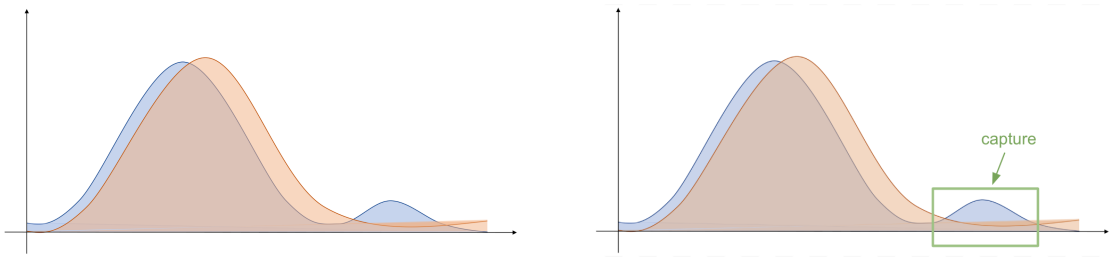
Chapter 2

PROPOSED METHODOLOGY

2.1 Limitation Of The One-Size-Fits-All Strategy

As mentioned earlier, most ML algorithms have been developed under the one-size-fits-all strategy, assuming there is a common set of co-variables that can accurately predict outcomes for all subjects. However, this strategy is limited to analyzing heterogeneous data because there are no such common co-variables.

For instance, fig. 2.1a shows a sample distribution of a binary dependent variable, e.g. case or control, over a single independent variable. As observed, the majority of case and control distributions are the same, but a small portion of the high value in the case group implies high heterogeneity. The proposed tree-guided PML prediction algorithm aims to find a value of the independent variable which groupifies the data in such a way that certain functions can fit to the groups to optimize the prediction as shown in fig. 2.1b. The working of the algorithm and a simulated scenario where the tree-guided PML strategy will outperform the one-size-fits-all are explained in the next sections.



(a) (b)

Figure 2.1: Distribution Of A Biomarker For Case And Control

2.2 Proposed Tree-Guided PML Algorithm

2.2.1 Overview

The tree-guided PML algorithm is presented to address the problem of heterogeneity in any given dataset. Given a data set consisting of N observations on P independent variables and a single dependent variable Y, tree-guided PML creates a binary tree with an ML model tailored to each leaf node. Similar to the standard CART algorithm, tree-guided PML is also a greedy algorithm where each node of the tree consists of an inequality condition on one of the independent variables, and the tree is generated by a recursive partitioning algorithm.

The tree-guided PML algorithm is developed with modifications made to a standard CART algorithm to achieve optimized grouping of data. Dividing data into groups and sub-groups for tailoring personalized ML functions is a crucial part of tree-guided PML. For developing the tree-guided PML algorithm, a significant addition made to the base CART is the intrication of some candidate ML algorithms that can be tailored to the resulting groups. Let, $f = (f_1, f_2, ..., f_q)$ be a set of Q candidate ML algorithms, where $f_j : X \rightarrow Y$, where $X \in (N \times P)$ and $Y \in \{0, 1\}$ (binary dependent variable). Here, $X = (X_1, X_2, ..., X_P)$ is a P dimensional vector of predictors (independent variables), and observed data is represented as $(Y_i, X_{i1}, X_{i2}, ..., X_{iP})$ for $i = 1, 2, ..., N$

2.2.2 Proposed Tree-guided PML Algorithm

Following is the grow-tree function algorithm for tree-guided PML:

Grow-tree():

1. Continue only if current depth $<= max\_depth$.

2. For $j = 1, 2, ..., p$:

   (a) Sort the X_i's into the unique, ascending values $(X_1, X_2, ..., X_N)$ (training data) as per the j.

   (b) Calculate K cutoff points $(c_{1j}, c_{2j}, ..., c_{Kj})$, where c_k for a j is calculated using eq. 2.1 for a standard CART algorithm. (However, it will be updated to quantile/percentile-based cut-offs for optimized tree-guided PML)

$$c_k = \frac{(X_i + X_{i+1})}{2} \tag{2.1}$$

   (c) For $k = 1, 2, ..., K$:

      i. Divide the dataset into two mutually exclusive sub-groups based on $X_{ij} <= c_{kj}$ or $X_{ij} > c_{kj}$ (training and validation).

      ii. Make sure, the no. of samples in the divided data $>= min\_samples$

      iii. Fit Q candidate ML algorithms for each subgroup of training data.

      iv. Calculate loss for each combination of ML algorithms on sub-groups of validation data. Let $(f_{left,kj}, f_{right,kj})$ be the selected combination of ML algorithms for subgroups that minimize loss among Q^2 combinations.

      v. If $loss < min\_loss$, retain $X_{(j)}$, c_{kj} and $(f_{left,kj}, f_{right,kj})$.

      vi. Continue only if $\Delta Loss >= {}_{min}\Delta Loss$, where $\Delta Loss =$ |loss w/o split – loss with split|.

3. For the retained $(X_{(j)}, c_{kj})$, the tree is split into 2 parts, left child and right child.

4. Grow-tree() for left child and Grow-tree() for right child.

Here, training-validation data splits are optimizations (explained in the next section), and for Step-2.c.4., the loss is maximized AUC or minimized Brier Score (or minimize MSE if Y is continuous).

Fig. 2.2 shows how a sample tree output from the tree-guided PML model tailoring one of the candidate ML algorithms to each leaf node. Here, multiple leaves can have the same ML algorithm but with different model parameters.

2.2.3 Simulated Scenario

Consider the following distribution of data as shown in f fig. 2.3a. Any standard one-size-fits-all ML algorithm would fail to fit this type of distribution. This distribution behaves uniquely for certain sections which cannot be analyzed by any single ML algorithm. It is a combination of both linear and non-linear distribution and hence, it needs to be handled group-wise. For similar distributions, fitting the proposed tree-guided PML algorithm will yield higher accuracy as it is designed to optimize the overall performance by analyzing data in groups. Fig. 2.3b shows how tree-guided PML will treat and analyze this distribution while fig. 2.4 shows what would the outputted result look like, where:

- TN-1 represents A-group of the data and the fitted ML algorithm $f_(1)$ is Lasso Linear Regression
- TN-2 represents B-group of the data and the fitted ML algorithm $f_(2)$ is Support Vector Classifier
- TN-3 represents C-group of the data and the fitted ML algorithm $f_(3)$ is Random Forest

2.3 Challenges And Proposed Alternatives

2.3.1 Over-Fitting

As mentioned in the above section, we have used validation data to find the best split. Usually, the validation dataset is used for hyper-parameter tuning, but it has been used to serve a different purpose here. In the motivation dataset for a large number of features (independent variables) there are very few samples in the data, that is, $p >> n$, which resulted in extreme over-fitting (Ying, 2019). During grid search, the most complex ML algorithm is selected for left and right child nodes at each split and thus for all terminal nodes too even though data is separated into training and test datasets. When trying to fit using tree-guided PML, the resulting tree had a very variance and thus performed poorly on the test dataset. To produce more generalized predictions it was necessary not to split the tree just using the training dataset. Thus, we used validation data to avoid the over-fitting problem. The entire data is divided into 3 datasets:

- Training Data: ML models are trained on this data (for training $f(X_{ij})$, $X_i \in$ training data).
- Validation Data: Model and its hyper-parameters are tuned based on the model's performance on this data (for loss $min(Loss(f(X_{ij} <= c_{kj}), f(X_{ij} > c_{kj})))$, $X_i \in$ validation data).
- Testing Data: This is completely unseen to the tree-guided PML model and is used to evaluate the model's performance.

The f_{left} and f_{right} are trained on the training data, but for splitting, loss calculated on the validation data is used.

Another parameter taken into consideration to avoid over-fitting was the Loss function. Initially, the tree was being split based on the AUC of the ROC. The combination of models yielding the highest AUC scores was tailored to the nodes. Finally, the tree-guided PML model was evaluated using the AUC score of the test data. However, the output tree-guided PML tree showed some variance and thus, was over-fitting. Later, the loss function was changed to Brier score (Rufibach, 2010) which boosted the performance of the algorithm and optimized its overall accuracy. The reason for this was that the AUC score is calculated based on binary hit-or-miss. Whereas when the Brier score was used as the loss function, the algorithm started focusing more on optimizing the probability of samples being classified correctly. Thus, the AUC score of the test data calculated for the tree-guided PML model developed using the Brier Score was higher than for the tree developed using the AUC score itself. The results for the same are discussed in the next chapter.

2.3.2 Time And Computation Complexity

Another major challenge while developing the tree-guided PML tree was the huge time complexity and high computational expense. Training of any single ML algorithm by itself is very expensive. While constructing a tree-guided PML tree, considering a single predictor $N \times Q^2$ ML algorithms are trained and evaluated, where N is the number of samples and Q is the number of candidate algorithms and this is repeated for all P predictors. Thus, the total complexity for a tree-guided PML tree with $depth = D$ becomes $O(N \times P \times Q(N,P)^2 \times D)$ because even Q depends on (N, P). In order to compare complexity for various different optimizations, we fixed constant depth which made the complexity proportional to $(N, P)^k$ as per eq. 2.2 which is very large, especially when the number of parameters (independent variables) P is huge.

$$O(N \times P \times Q(N,P)^2) \approx O((N \times P)^k) \tag{2.2}$$

In order to address this, two different methods were tested.

1. **Stopping Criteria:** Firstly, we restrict the growth of the tree-guided PML tree to avoid exploding the complexity. This was done by defining some hyper-parameters that serve as stopping criteria for the growth of the tree. Stopping criteria include fixing the following three hyper-parameters:

   (a) Max Depth: maximum depth to which the tree can grow.

   (b) Min Samples: minimum number of samples of each category required for a node to be considered for splitting further.

   (c) $_{min}\Delta Loss$: the loss after splitting should be more optimized than without, by a certain extent. Thus, for a node to split it should the criterion shown in eq. 2.3.

$$\begin{aligned} \Delta Loss &= |loss\, w/o\, split - loss\, with\, split| \\ \Delta Loss &>= {}_{min}\Delta Loss,\ {}_{min}\Delta Loss \in [0, \infty) \end{aligned} \tag{2.3}$$

2. **Dimensionality Reduction Techniques:**

   (a) **Principal Component Analysis (PCA):** In order to reduce dimensionality a very popular dimensionality reduction technique PCA (Howley *et al.*, 2005) was used. PCA reduces the dimensionality of the data while retaining as much of the original information as possible. It finds a new set of uncorrelated variables, called principal components, that capture the maximum variance in the data. From the found principal pompo-nents, components representing the highest variance are selected descend-ingly and they are now new parameters that can be used for training and testing ML models. Thus, by selected top p components, P is reduced to p in eq. 2.2 and complexity reduces to $O((N \times p)^k)$. However, when the

data is highly heterogeneous, even selecting some/many of the principal components would fail to capture much information. Thus, using PCA for developing a tree-guided PML tree with such a study only worsens the performance.

(b) **Quantile/Percentile:** For the construction of a normal CART/Decision-Tree with continuous parameters, a rolling average for each of two consecutive samples ordered ascending (individual parameter-wise) is taken into consideration for splitting the node. Thus, count $n(c_k) \approx N$ for each P, resulting in high complexity of eq. 2.2. Constructing the tree-guided PML tree with this high complexity was not feasible and hence it became of utmost importance to fix this issue. Unlike PCA which reduces P, we implemented a technique that focused on reducing N thus, not losing much parametric information. This technique is based on using quantiles/percentiles (Redivo, 2023) values rather than the rolling average for splitting the node. We can fix the number of partitions to be considered and thus, N becomes n in eq. 2.2 which can be $4, 10, 100, 200$, etc., reducing complexity to $O((n \times P)^k)$. This technique yielded a performance comparable to considering all N samples for the rolling average.

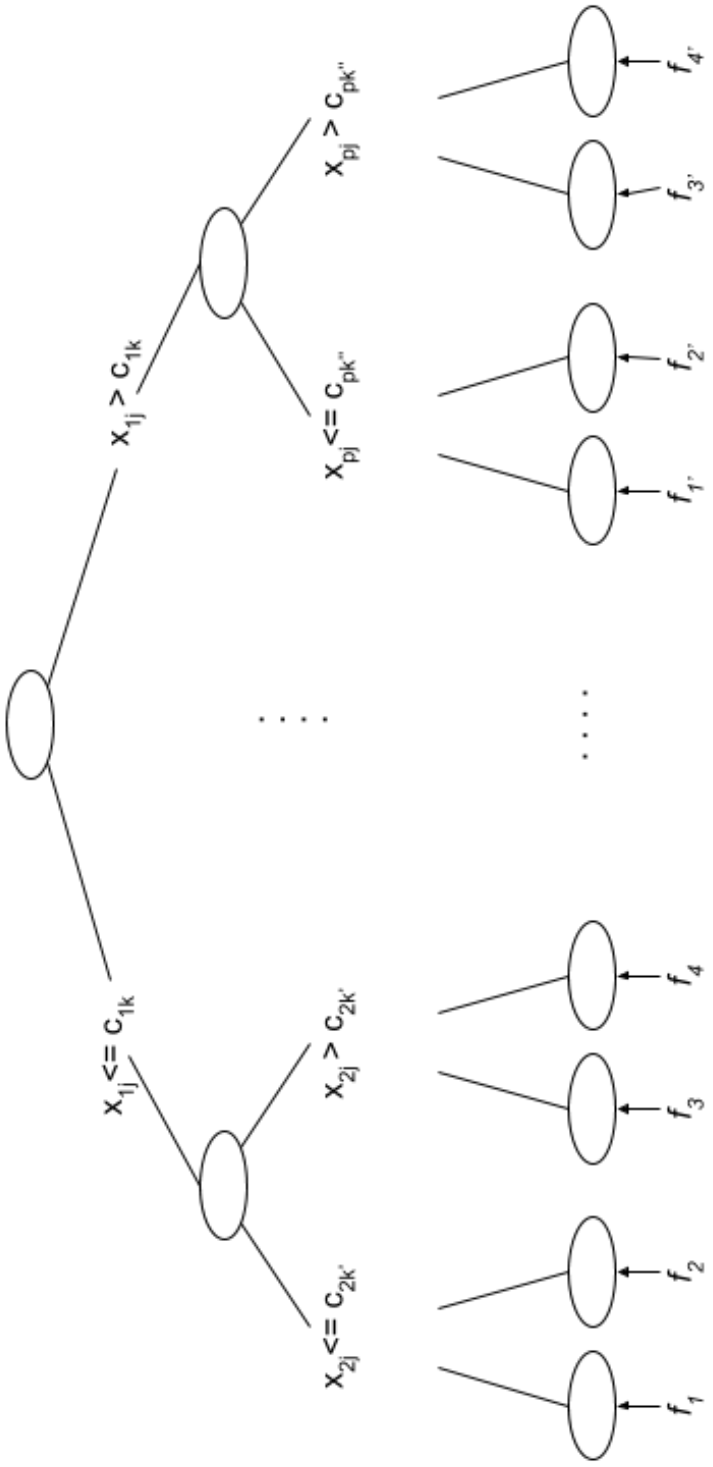
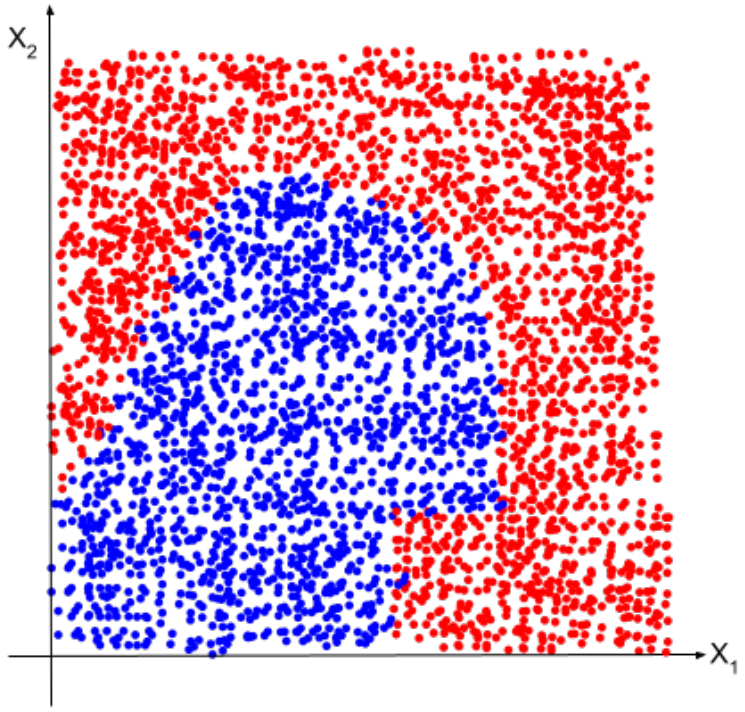
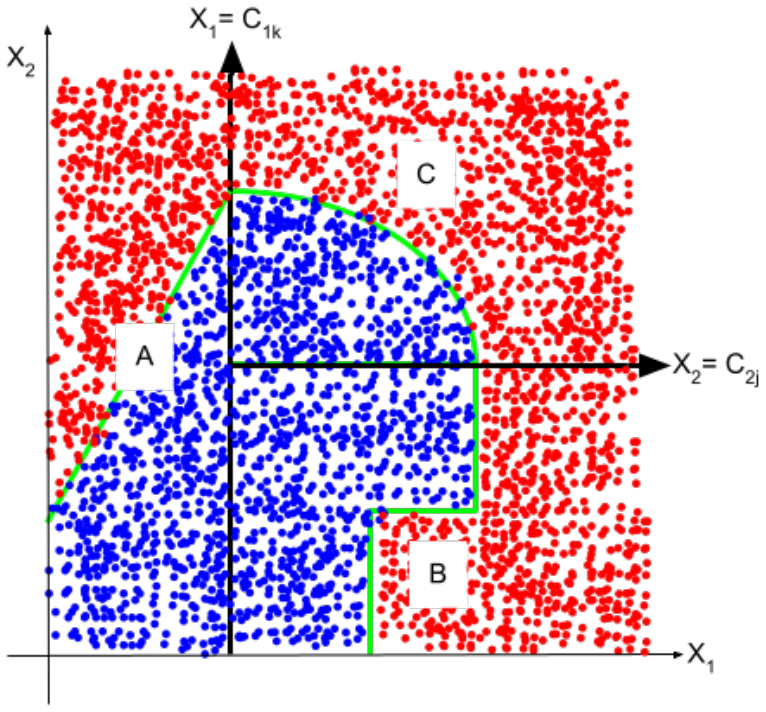


Figure 2.2: Tree-guided PML Model With f_i Tailored To Each Leaf



(a)



(b)

Figure 2.3: Binary Data Distribution For The Simulated Scenario (A) And Its Analysis Using Tree-guided PML (B)

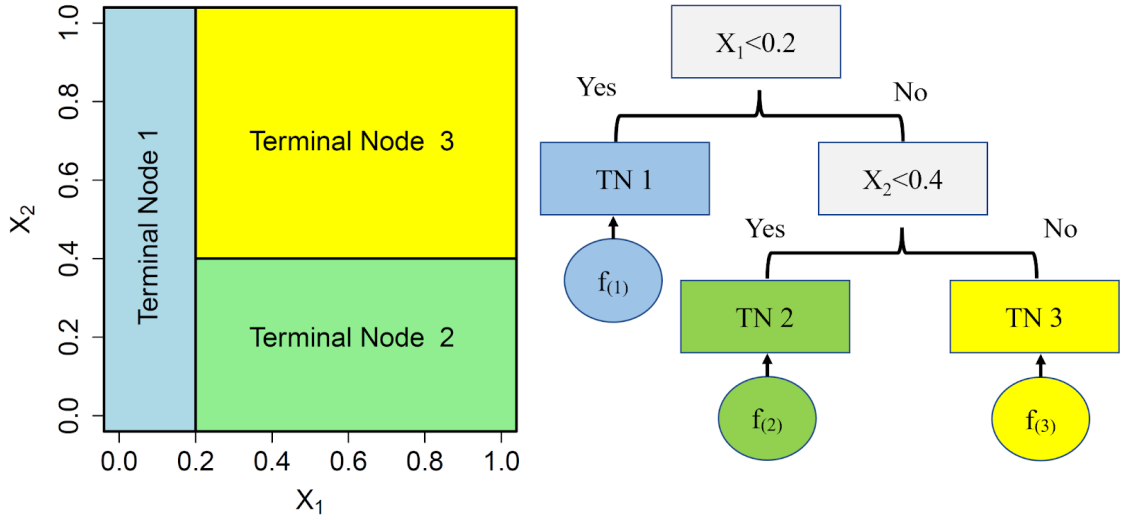


Figure 2.4: Developed Tree-guided PML With f_i Tailored To Each Terminal Node Representing Each Group Of Entire Data. The Data Is Divided Into Groups Such That It Can Be Analyzed Using Any Of The Candidate ML Algorithms

Chapter 3

REAL DATA ANALYSIS

3.1 Dataset Description

Arizona State University in collaboration with Mayo Clinic conducted the colon cancer biomarker study from 2016 to 2020 which was funded by National Cancer Institute. The data acquired from this study is the main motivation for our research. Under the cross-sectional case-control study design, the colon cancer biomarker study collected 599 colon cancer and 599 healthy control samples and quantified their 1654 candidate immunoglobulin-G biomarkers using a high-throughput technology, NAPPA. The aim of this study was to identify a panel of biomarkers to accurately distinguish between the cases and controls given that the biomarker showcased high heterogeneity, where they respond differently for each sample as displayed in fig. 1.1. The proposed tree-guided PML strategy accommodates this heterogeneity by tailoring the optimal ML models to each sample according to their individual characteristics.

3.2 Baseline Data & Hyper-parameters

Before starting with training and developing the tree-guided PML prediction tree, there were a few parameters that were fixed. Firstly, the entire dataset was normalized and shuffled, and then the total of 1198 samples (599 for each case and control) were divided into train, validation, and test datasets as follows.

1. Training Data (N): 700
2. Validation Data (M): 300
3. Test Data (T): 198

Then, the following stopping criteria were selected to examine the overall performance of the algorithm (however, they were updated as per requirements later).

1. $MaxDepth = 5$
2. $MinSamples = 50$ (25 of each category)
3. ${}_{min}\Delta Loss = 0$

Also, the set of candidate ML algorithms was kept limited to [Lasso Logistic Regression, Random Forest Classifier (tunned), Support Vector Machine Classifier] and the evaluation metric selected was ROC-AUC score as the final results of the model needed to be evaluated on the same.

With no other updates made with respect to PCA, quantile-zation, ${}_{min}\Delta Loss$, or evaluation metrics, the algorithm was started to output a tree-guided PML prediction tree. However, as mentioned in the previous chapter, trying to run the algorithm without any optimization was not a good idea. It required more than 10+ days even to reach to depth of 3. Thus, various experiments were conducted with various

individuals and combinations of optimizations. The details of these experimentations are discussed in the next section.

3.3 Experiments And Results

As discussed the first thing that needed to be optimized was the number of dimensions and hence, the main focus was on experimenting with various dimensionality reduction techniques so that the algorithm can be executed in a feasible amount of time.

The first technique we experimented with is one of the most popular methods used to efficiently reduce the number number of dimensions which is PCA. On applying PCA to a normalized dataset, it returns linearly transformed components of the dataset distribution in descending order of their capability to explain the variance in the data. Meaning, that the first returned component called the principal component will constitute the highest percentage of the variance of the dataset that any of its linear transformations can explain. Thus, to reduce the time complexity, we reduce the number of training parameters (independent variables) by selecting the top p components calculated by applying PCA for our dataset. Keeping all the other hyper-parameters constant in eq. 2.2, we tested the proposed algorithm with various values of p. However, as mentioned in the previous chapter, when PCA is applied to a heterogeneous dataset, even the principal component can only comprehend a very small percentage of the variance. A lot of information will be lost when only a few of them are selected for dimensionality reduction and thus, this strategy fails to enhance the overall performance of the tree-guided PML algorithm. Table 3.1 shows the results of the tree-guided PML algorithm after applying PCA with various ps proving the same.

As PCA did not work for our dataset, the next thing we tried to reduce the time complexity was to just decrease the number of training samples, i.e. n. By randomly selecting a smaller set of samples for training, we can still make an accurate prediction

p **Components**	**% Variance**	**Days to Execute**	**Test AUC Score**
10	40.17%	1	0.281
50	41.02%	3	0.315
100	42.89%	5.5	0.348
200	46.66%	7	0.394
500	62.52%	9+	- - -

Table 3.1: PCA Results Table

and yet decrease the complexity. This might result in another problem, over-fitting, but it was important to optimize the time complexity for the successful execution of the algorithm. Hence, we tried randomly selecting samples varying the value of n which according to eq. 2.2 made the time complexity $O((n \times P)^k)$. The results are as shown in Table 3.2.

n **Samples**	**Days to Execute**	**Test AUC Score**
100	5	0.467
200	7	0.583
300	9	0.624
500	10+	- - -
700	10+	- - -

Table 3.2: Random Sampling with n-Samples Results Table

The next we tried was implementing a technique that optimizes complexity but not at the cost of over-fitting, a technique that optimizes both the potential challenges. For this, instead of performing random sampling to reduce n, we quantization to select values to be considered for the development of the tree-guided PML tree. With this, we neither lose parametric information as we do in PCA nor reduce the number of training samples causing over-fitting as we had to. This technique changes the

way the tree is being split at each node, thus affecting the complexity. Here, instead of considering the rolling average of all samples sorted ascendingly for the selected parameter, we consider their quantile values. Quantiles essentially mean 25, 50, 75th% percentiles of the samples values, but here, any partitioning values can be used, for example, percentiles with values at 1, 2, ..., 99, 100th% percentiles of the samples can be used too. Thus, n becomes the number of quantiles q. The results with various values of qs are as shown in Table 3.3.

q **Quantiles**	**Days to Execute**	**Test AUC Score**
4 (Quantiles)	1	0.378
100 (Percentiles)	4	0.544
200	7	0.668
400	10	0.671

Table 3.3: Quantiles Results Table

We tried controlling the growth of the tree in order to reduce the running time by regulating ${}_{min}\Delta Loss$ as per eq. 2.3. However, stopped the immediate growth of the tree because, at an earlier stage, the loss might not decrease that significantly (it might even increase), but eventually as the tree grows it would optimize the final results. Thus, this technique did not prove to be effective.

As can be seen from the above results, the algorithm still showed some variance and hence, we tried changing the loss function to see if it enhances the performance or not by avoiding this variance. For this, we changed the splitting criteria from maximum AUC to minimum Brier score. That is, till now, the combination of ML functions yielding the highest AUC scores was being tailored to the nodes, but now, ML functions with the lowest Brier score will be considered. However, the overall performance of the tree-guided PML tree and the algorithm will still be evaluated

using the AUC score of the test dataset. Brier score will be considered during the development of the tree where the node is being split. Replacing the intermediate AUC loss function with the Brier score boosted the performance of the algorithm. As discussed in the previous chapter, when the Brier score was used as the loss function, the algorithm started focusing more on optimizing the probability of samples being classified correctly. Thus, as can be seen in Table 3.4, the AUC score of the test data calculated for the tree-guided PML tree developed using the Brier score was higher than for the tree developed using the AUC score itself.

Technique	**Hyper-parameter**	**Test Brier Score**	**Test AUC Score**
PCA	$p = 200$	0.2703	0.402
Sampling	$n = 300$	0.2388	0.650
Quantiles	$q = 200$	0.2187	0.729

Table 3.4: Brier Score Results Table

Finally, we propose an optimized and best-performing tree-guided PML tree that is able to successfully analyze heterogeneous data. This model uses 200 quantile values of the training samples during the splitting of tree nodes, splitting loss is based on the Brier score loss function and all 700 training data samples are used for training ML algorithms. For the divided data and fixed hyper-paraments, its performance with comparison to standard ML algorithms and top-one biomarker is as described in Table 3.5. The comparison of their AUC score on the test data is shown in fig. 3.1.

After this entire process, the tree-guided PML tree generated that yielded the best results is as shown in fig. 3.2. The names of biomarkers are encoded for confidentiality reasons. In PD, the features contributing to successful analysis are of significant importance. The tree-guided PML tree and the features obtained from the Ml algorithms tailored to terminal nodes (according to their importance) are

Technique	Test AUC Score
Tree-guided PML	0.720
Lasso Regression	0.519
Random Forest	0.560
Support Vector	0.416
Top Biomarker	(fitted AUC) 0.637

Table 3.5: Results Comparison With Classical ML Algorithms

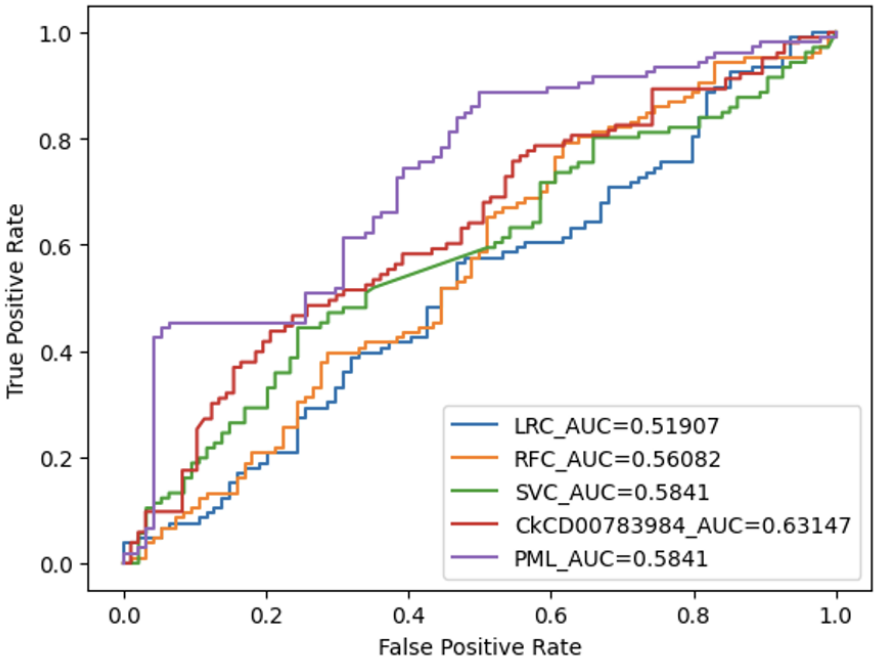


Figure 3.1: Test AUC Scores Of PML Tree With Comparison To Standard ML Algorithms And Top Biomarker.

studied further to improve diagnostics. The top-5 encoded features according to their importance for the tailored ML models are described in fig. 3.3. Thus, even though it is a time-consuming technique, the results and insights obtained as the results are worthwhile.

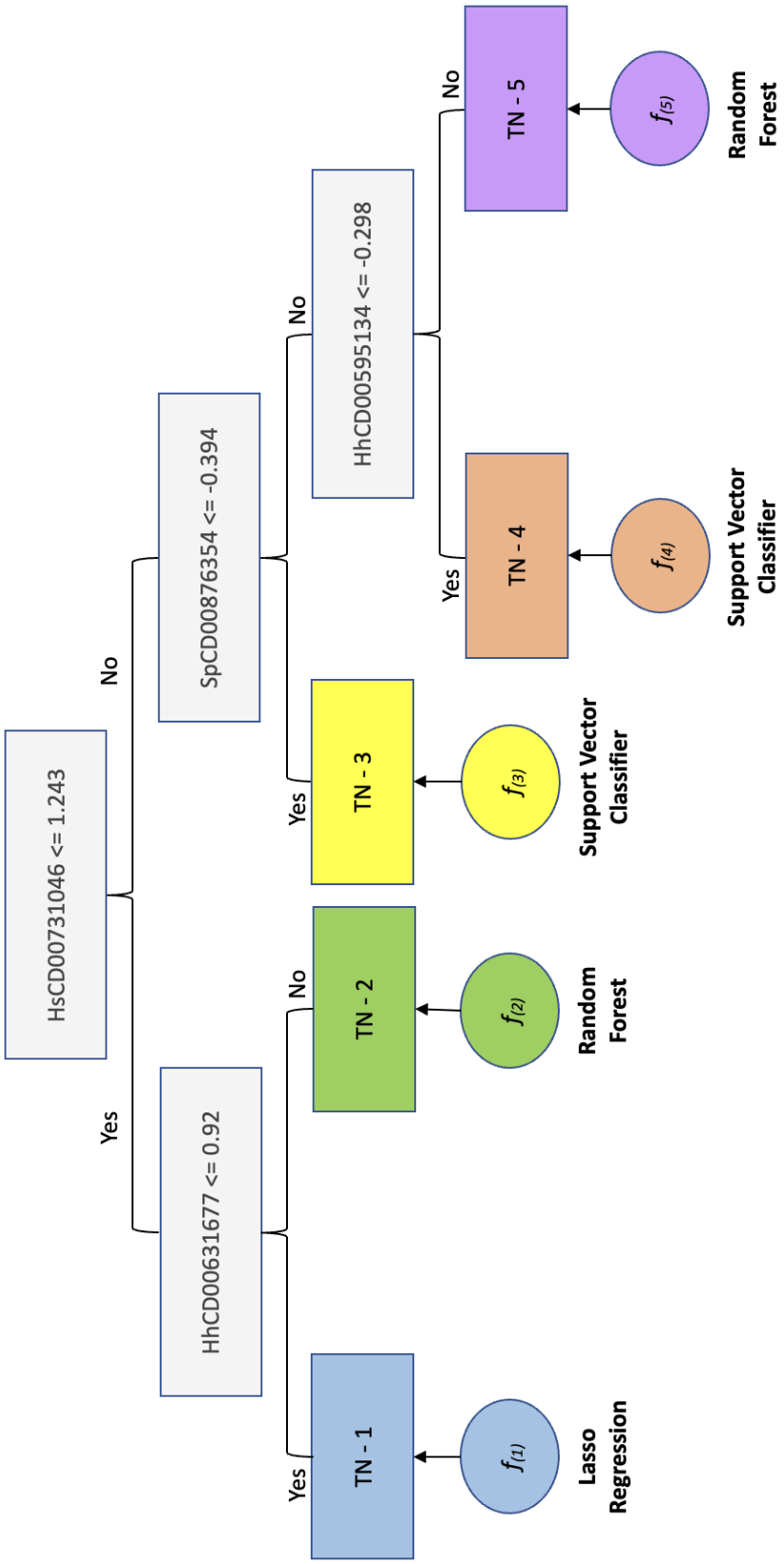


Figure 3.2: Best Performing Generated Tree-guided PML After All Optimizations.

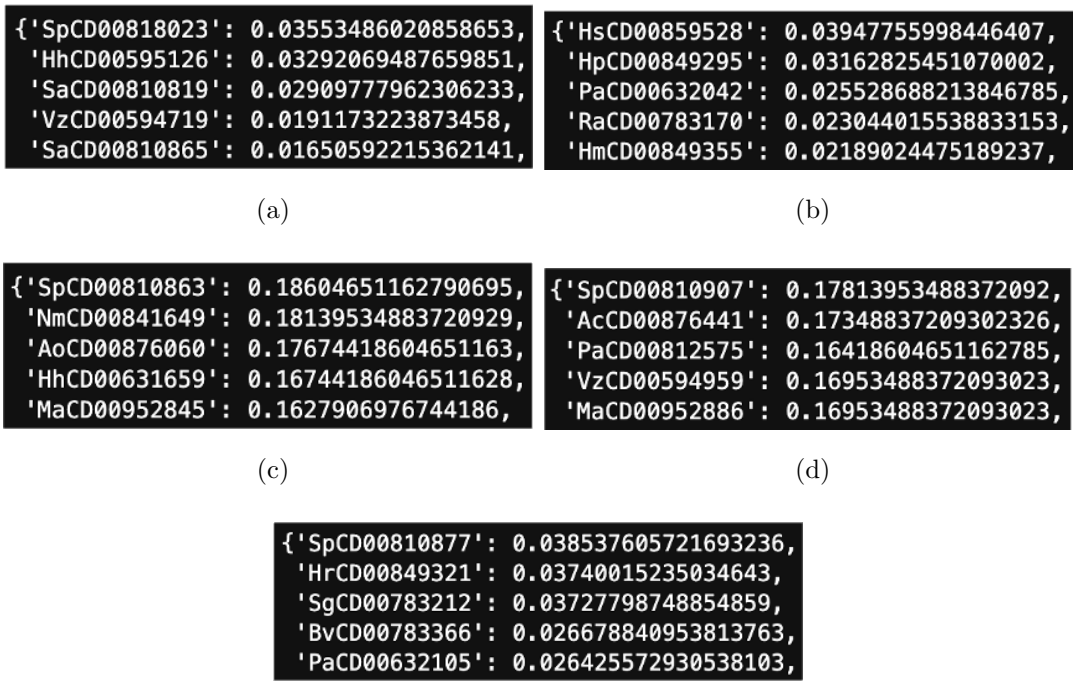


Figure 3.3: ML Model-wise Top-5 Features And Their Importance Score With (A) For Lasso Regression $f_{(1)}$, (B) For Random Forest $f_{(2)}$, (C) For Support Vector $f_{(3)}$, (D) For Support Vector $f_{(4)}$, And (E) For Random Forest $f_{(5)}$

Chapter 4

DISCUSSION

One of the biggest limitations of the proposed algorithm is generalizing the output. Because of the high time complexity, it was difficult to develop a relativity matrix for all the important biomarkers surfacing for an appropriate number of experiments. One such extension for this could be developing a heatmap-like matrix highlighting the importance of features for the fixed number of experiments. This would provide confidence value for the feature importance score and thus, help to evaluate the consistency of the algorithm. Other techniques like cross-validation and pruning can be appended to the current algorithm to achieve higher-level generalization (and reduce over-fitting).

As discussed in earlier chapters, time and computation complexity pose a serious challenge for developing the tree-guided PML output, developing a more efficient and optimized solution to this problem can still be researched further. One such solution can be parallel computing. For the development of any ML Tree, for each feature, approximately N (for each sample) or q (if using quantile/percentiles) iterations are performed to find the value c_k that minimizes the loss for that particular feature. This process is independent for each feature and hence, the minimum possible loss for each feature can be calculated parallelly and stored in memory so that the minimum of them can be selected for splitting the node. By doing this, the number of computations can be decreased by n times, where n is the number of cores available for parallel training.

The dependent variable of the motivation case study was binary i.e., $Y \in \{0, 1\}$. The tree-guided PML algorithm was developed and tested keeping this study in mind.

Thus, the loss function and the set of considered candidate ML algorithms were both defined to perform the binary classification task. However, the proposed tree-guided PML algorithm can be used for multi-variate and continuous dependent variables too. The only updates needed to be made to the current algorithm would be to use a different set of candidate algorithms and the loss function customized accordingly to the problem at hand. The code for generalized tree-guided PML strategy supporting these outputs has also been developed as an extension of this research. The further extension that can be made is to develop a Python package for the developed Python functionalities.

REFERENCES

Alexander, W. P. and S. D. Grimshaw, "Treed regression", Journal of Computational and Graphical Statistics **5**, 2, 156–175 (1996).

Breiman, L., "Random forests", Machine Learning **45**, 5–32 (2001).

Breiman, L., *Classification and regression trees* (Routledge, 2017).

Chipman, H., E. George and R. McCulloch, "Bayesian treed models", Machine Learning **48**, 299–320 (2002).

Chipman, H. A., E. I. George and R. E. McCulloch, "BART: Bayesian additive regression trees", The Annals of Applied Statistics **4**, 1, 266–298, URL `https://doi.org/10.1214/09-AOAS285` (2010).

Díez, P., ""nappa as a real new method for protein microarray generation."", Microarrays (Basel, Switzerland) **4**, 2, 214–217 (2015).

Hearst, M., S. Dumais, E. Osuna, J. Platt and B. Scholkopf, "Support vector machines", IEEE Intelligent Systems and their Applications **13**, 4, 18–28 (1998).

Howley, T., M. Madden, M.-L. O'Connell and A. Ryder, "The effect of principal component analysis on machine learning accuracy with high dimensional spectral data", pp. 209–222 (2005).

McAlister, F. A. e. a., "Finding the right balance between precision medicine and personalized care", CMAJ : Canadian Medical Association journal = journal de l'Association medicale canadienne **189**, 33, 1065–E1068 (2017).

McAuley, J., *Personalized machine learning* (Cambridge University Press, 2022).

Redivo, e. a., Edoardo, "Quantile-distribution functions and their use for classification, with application to naïve bayes classifiers", Statistics and Computing **32**, 2 (2023).

Rufibach, K., "Use of brier score to assess binary predictions", Journal of clinical epidemiology **63**, 8, 938–939 (2010).

Tibshirani, R., "Regression shrinkage and selection via the lasso", Journal of the Royal Statistical Society. Series B (Methodological) **58**, 1, 267–288 (1996).

Torgo, L., "Functional models for regression tree leaves", pp. 385–393 (1997), proceedings of the Fourteenth International Conference on Machine Learning.

Wang, Y. and I. H. Witten, "Induction of model trees for predicting continuous classes", Proceedings of the poster papers of the European Conference on Machine Learning (1996).

Ying, X., "An overview of overfitting and its solutions", Journal of Physics: Conference Series **1168**, 022022 (2019).

APPENDIX A

RESOURCES

Computational Resource

"Days to Execute" is considered on the basis of the time taken for the algorithm to complete on one node with a single core of ASU's Agave Cluster. It was running sequentially.

Permission

The research has been conducted under the guidance of my mentor and co-chair, Dr. Yunro Chung. His permission has been obtained to include his background work in this document.

Github Repository

All the codes developed during the course of the work can be found at `https://github.com/nsshah15/CPD.git`

Presentation Link

The link for the presentation is as follows: `https://drive.google.com/file/d/1NwxkRUzclpe1jmm8kKOMg8piNyHdIoGx/view?usp=sharing`. Please use only the ASU email ID to access it.