# THW: Number Classification Demostration

Nathan Walter

(Dated: May 4, 2016)

## I. DATA GENERATION

For this assignment, we study various methods as they are applied to a classification problem. The classification problem is to determine a model for predicting a handwritten 3 or 8. The 3 or 8 is placed into a box and pixilated such that predictor values are the gray scale value of each pixel in the box. The goal is to predict if a number is a 3 or 8 based on the gray scale scores in a box. The training data is composed of 1200 samples and the test set is composed of 332 samples. There is one response variable, either of class "3" or class "8". The data is composed of 456 predictors, the first 256 are the gray scale scores of the pixels in the 16 x 16 box. The next 100 predictors are created by randomly choosing 2 of the first 256 predictors and multiplying the values. The remaining 100 predictors are a repeat of one of the first 356 predictors randomly chosen.

## II. LDA

The goal of a predictive model is to predict the value of $\Pr(Y = k|X = x)$. Linear Discriminant Analysis (LDA) attempts to model this by using Bayes theorem for conditional probability. Thus, LDA solves

$$\Pr(Y = k|X = x) = \frac{\pi_k \Pr(X = x|Y = k)}{\sum_{l=1}^{K} \pi_l \Pr(X = x|Y = l))} \tag{1}$$

where K is the number of classifiers, $\pi_k$ is the probability a observation belongs to class k. Next, LDA assumes that the values for $\pi$ are equivalent to those in the training data. Further, LDA approximates the conditional probabilities as multivariate gaussian distributions for each class k, this takes the form

$$\Pr(X = x|Y = k) = \frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_k)^T\Sigma^{-1}(x-\mu_k)\right) \tag{2}$$

where p is the number of predictors, $\mu_k$ is a vector of means for class $k$ of length p, and $\Sigma$ is the covariance matrix which is the same for all classes. Thus, the means is determined from the training data for each class, which the covariance matrix is determined from all the training data. Thus, LDA classifies an observation x based on which class it most likely belongs to.

For this assignment, I use the package MASS and the function lda to perform linear discriminant analysis.

## III. NAIVE BAYES

Naive Bayes is similar to LDA in that it attempts to approximate the probability of an observation belonging to a class k with Bayes Theorem.

$$\Pr(Y = k|X = x) = \frac{\pi_k \Pr(X = x|Y = k)}{\sum_{l=1}^{K} \pi_l \Pr(X = x|Y = l))} \tag{3}$$

However, Naive Bayes assumes that the predictors are independent of one another, or that the covariance matrix, $\Sigma$, is diagonal, full rank, and is unique to the class. Thus, Naive Bayes makes the following assumption

$$\Pr(X = x|Y = k) = \Pi_i^p \Pr(X_i = x_i|Y = k) \tag{4}$$

There is two ways to perform Naive Bayes, parametric and non-parametric. Parametric Naive Bayes makes assumptions about the shapes of the probabilities of each predictor. For this assignment we assume that the probabilities for each predictor are univariate normal distributions.

$$\Pr(X = x|Y = k) = \Pi_i^p \frac{1}{(2\pi\sigma_{ik}^2)^{1/2}} \exp\left(-\frac{(x_i-\mu_{ik})^2}{2\sigma_{ik}^2}\right) \tag{5}$$

where the mean and variance is unique to the predictor and class. Thus, for this assignment, since we have 2 classes, when we program the parametric naive bayes I used the following equation to determine the class

$$\frac{\Pr(X = x|Y = 3)}{\Pr(X = x|Y = 8)} > 1 \tag{6}$$

If this statement is true, the class is evaluated as apart of class 3. The previous equation can be further simplified by taking the log of both sides and simplifying to

$$\begin{aligned}&\frac{1}{2}\sum_i^p \log\left(\frac{\sigma_{i2}^2}{\sigma_{i1}^2}\right) + \log(\frac{\pi_1}{1-\pi_1})\\ &-\sum_i^p \frac{(x_i-\mu_{i1})^2}{2\sigma_{i1}^2} + \sum_i^p \frac{(x_i-\mu_{i2})^2}{2\sigma_{i2}^2} > 0\end{aligned} \tag{7}$$

if this is true, then the observation is apart of class 3. Parametric naive bayes is implemented both by my own code and by the function naiveBayes from the package e1071.

On the other hand, Naive Bayes can also be non-parametric, which means no assumptions are made about the shape of the distribution for each predictor. Instead, the non-parametric naive bayes histograms all of the observations and uses a kernel to smooth the histogram over all space. Then, from this smoother histogram, the observation is given a class. For non-parametric Naive Bayes, I used the package klaR and the function Naive Bayes.

For parametric Naive Bayes, the method suffers from problems where a predictor has a zero variance. This is due to the fact that if the variance is zero, the method will have a division by zero, resulting in NaN values. Thus, in order to avoid this problem, I preprocessed the data to determine which class and predictor has a zero variance and I excluded that variable from the calculation. When I looked into the predictors that had a zero variance, it appeared to mostly be the corner pixels that had zero variance.

## IV. PCA

PCA, principle component analysis works by using singular value decomposition to determine the principle components in the data set. Then, based on these principle components, perform linear least squares regression with these components, then transform the model back to the original predictors. This method thus works by only studying the predictors that are not heavily correlated with one another. This method is similar to noise reduction in signal processing and file compression. PCA can be used to determine components that explain the data very well. Thus, PCA is used to eliminate predictors that are not strong explainers of the data. For this portion, the function princomp from the package stats is used. From this function, I determine the principle components that explain 95% of the data, and transform the training and test data into these components.

## V. LOGISTIC REGRESSION

Logistic Regression attempts to model the classification problem with a function that is restricted to the bounds 0 to 1. The function used is

$$\Pr(Y=1|X=x) = \frac{\exp(\beta_o + \sum_i^p \beta_i X_i)}{1+\exp(\beta_o + \sum_i^p \beta_i X_i)} \tag{8}$$

where $\beta$ is the coefficients that will be estimated to fit the equation to the training data based on MLE. Thus, this provides a continuous value between the values 0 and 1. Then, one can choose a threshold value $0 < T < 1$ to determine which class the observation belongs to.

$$\begin{array}{ll} \Pr(Y=1|X=x) < T & Y=0 \\ \Pr(Y=1|X=x) > T & Y=1 \end{array} \tag{9}$$

This is the foundation for logistic regression. For this assignment we study the logistic regression model with all predictors (full model), a model created with forward AIC, a model created with forward BIC, and a model created with Lasso. For this assignment I use the package glmnet. For logistic regression of the full model I use the function glm with family = binomial, for AIC I use step and glm with family = binomial and direction = "forward", for BIC I use the same functions but with k = log(n), and for lasso I use cv.glment with alpha = 1 with 10 fold cv. To determine the value of $\lambda$, I use cv.glmnet which uses 10-fold cv to determine the error for various values of $\lambda$. I then create the model based on the value of $\lambda.min$ returned

## VI. AIC/BIC/LASSO

For this report, AIC and BIC is often used to determine the best selected model. These two values are calculated for a certain model, and the best model is selected based on the model that results in the lowest value for AIC or BIC. The AIC is calculated as

$$AIC = 2d - 2\ln\hat{L} \tag{10}$$

where $d$ is the number of predictors in the model, and $\hat{L}$ is the likelihood estimator for the model. If we assume the errors are distributed by a gaussian profile then the likelihood estimator is equivalently

$$\ln\hat{L} = -\frac{n}{2}\ln(2\pi) - \frac{n}{2}\ln(\hat{\sigma}) - \frac{1}{2\hat{\sigma^2}}\sum_{i=1}^{n}(y_i - \hat{\mu})^2 \tag{11}$$

where n is the number of observations and $\hat{\sigma}$ and $\hat{\mu}$ are the estimated mean and standard deviation, and $y$ is the response variable. Ignoring terms that are mutual to all models, the AIC is

$$AIC = 2d - n\ln(\hat{\sigma}) \tag{12}$$

where $\hat{\sigma}$ can be estimated from the model. Similarly, the BIC differs from the AIC only slightly

$$BIC = 2\ln(n) - 2\ln\hat{L} = 2\ln(n) - n\ln(\hat{\sigma}) \tag{13}$$

Thus, in this report, when a model is selected by AIC or BIC, this is how the values are computed, and the best model is selected as the model that minimizes these values.

Lasso works similarly as full least squares except that they penalize the least square regression. Rather then simply minimize the residual sum of squares, ridge regression adds a term based on the L2 norm. Thus,

$$\hat{\beta} = (\mathbf{X^T X} + \lambda\mathbf{I})^{\mathbf{-1}}\mathbf{X^T Y} \tag{14}$$

The added term penalizes the least squares. $\lambda$ is a varying parameter that determines how strong the regularization is, a value of 0 is the normal least squares. Lasso works similar to ridge but instead of penalizing the L2 norm, Lasso penalizes based on the L1 norm.

For this report, the values of lambda used will be lambda.min and lambda.1se. Lambda.min is the lambda that results in the least cross validated mean error, and lambda.1se is the largest value of lambda within one standard error of the error of lambda.min. Thus, lambda.1se is adds more regularization at the cost of more error.

## VII. SVM

SVM or support Vector Machines attempts to use a hyperplane to separate the two classes. This plan attempts to place all of one class to side of the plane and all of the other class to the other side of the plane. However, the issue is that there often an infinite number of planes that will satisfy this goal, so the problem becomes how to chose the best hyperplane. In mathematic terms, this model is

$$\begin{aligned}\beta_o + \sum_{i}^{p} \beta_i X_i > 0 \qquad Y = 1 \\ \beta_o + \sum_{i}^{p} \beta_i X_i < 0 \qquad Y = 0\end{aligned} \tag{15}$$

In order to eliminate the infinite number of potential hyperplanes to one, the idea of margins is introduced. Thus, the hyperplane that maximizes the distance between the data points and the margins is used. However, in order to account for problems when a perfect separation cannot occur, the idea of soft margins is introduced. The idea is that some points may cross the margin or the hyperplane but if a point is on the wrong side a cost is applied. Lastly, for this method, different types of hyperplanes can be applied. Basically, depending on the type of hyperplane shape desired a different kernel is used to represent the inner product and is used to estimate the coefficients for the hyperplane. For a linear case, a linear hyperplane uses the kernel

$$K(x_i, x_{i'}) = \sum_{j=1}^{p} x_{ij} x_{i'j} \tag{16}$$

For a polynomial hyperplane, the kernel is

$$K(x_i, x_{i'}) = (1 + \sum_{j=1}^{p} x_{ij} x_{i'j})^d \tag{17}$$

where d is the polynomial degree. Lastly, for a gaussian hyperplane the kernel is

$$K(x_i, x_{i'}) = \exp(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2) \tag{18}$$

For this assignment, the package e1071 is used and the function svm is used. For a linear kernel, svm is used with kernel = "linear". For a polynomial kernel, the function svm is used with kernel "polynomial". Both functions are used with the cost equal to of 1. For the gaussian kernel, the function tune is used first to determine the optimal value of $\gamma$ and cost, however, I found that the defaults actually still worked best for me.

## VIII. TREE

The tree method starts with all possible predictors and divides the space into two binary section based on a value of one predictor, based on which predictor split results in the best reduction in the deviance. A decision tree then picks a new predictor to make the next split on. However, the trees can easily result in over fitting, thus it is often beneficial to prune the tree based either on the deviance or the number of misclasses. For this assignment I use the package tree and the functions tree, cv.tree, and prune. tree. Tree is used to build a full tree, the parameters used are mindev = 0.0005 and minsize = 1, and the split based on deviance. These parameters were used because it would result in a very large tree before pruning. Then, I used prune.tree with both deviance and misclass to prune the tree and determine the optimal tree. cv.tree was used to determine the best tree based on 10-fold cross-validiation.

## IX. RANDOM FOREST

Random Forest works by creating many trees and averaging the results. Each tree is created by randomly sampling m¡p predictors and creating a split, rather then considering all p predictors. For this assignment we use a value of $m = \sqrt{p}$ because the book and Internet claims it is the optimal value to use in random forest for a classification problem. Random forest randomly selects m predictors in order because if there is one predictor that is dominate, then the tree method with bagging will almost always choose this predictor for the first split, thus random forest by potentially not considering this predictor will create more unique trees to average over. For this assignment, the package randomForest with the function randomForest is used. The number of trees made is 500.

## X. BOOSTING

Boosting iteratively builds more trees but builds the new tree with the knowledge of the previous trees built. The new tree considers how much of the data is explained by the previous tree, and thus attempts to build a new tree to explain the remaining unexplained data. There is a tuning parameter of number of iterations and the shrinkage parameter. For this assignment the package gbm is used and I use the function gbm and gbm.pref. gbm will determine a serious of iteratively created trees, and gbm.perf will estimate the optimal number of iterations. To determine the optimal value of the shrinkage parameter I perform a for loop over various values of the shrinkage value.

## XI. RESULTS

For problem one, the data is modeled with LDA, Naive Bayes parametric, my own Naive Bayes parametric, and Naive Bayes non-parametric. The first thing to note, is that the results are the same for my Naive Bayes and the

Naive Bayes from package e1071, this is expected. It is important to note that e1071 can perform naive bayes even with zero variance predictors included. However, once these predictors are removed, the model predicted by e1071, klar, and my naive bayes parametric is the same model. Also, the test error is greater for the parametric naive bayes when the zero variance predictors are included. The next important result is that LDA outperformed all bayes methods, and non-parametric out performs parametric bayes. This is expected since LDA has the greatest amount of flexibility. Naive Bayes assumes the predictors are independent which results in an increase in test error. Further, the parametric methods assumes the shape of the predictors which results in more error compared to non-parametric.

Next, pca is performed reducing the number of predictors from 456 to 114. With only the 114 principle components considered (which explain 95% of the data), the test error goes down significantly for LDA and naive bayes. However, it is interesting to note that the test error for parametric naive bayes is now less than non parametric naive bayes. LDA is still the best of the three methods.

Next, the data is modeled with logistic regression full, forward AIC, forward BIC, and lasso. For lasso, the value for lambda used was lambda.min reported by 10-fold cross validation. For this the best method was lasso, and the worst was the full model, while AIC and BIC actually predicted the same model. This makes sense since the full model most likely over predicts the data. AIC and BIC created models with 21 predictors. Of the 21 predictors used, none of the predictors included were the noise variables. Logistic regression with lasso predicted a model with 116 terms, many of which are the noise predictors. Additionally, all of the predictors used in AIC and BIC are included in the predictors for lasso, lasso just predicted a much larger model.

Next, I tried logistic regression again but with pca performed first. All of the models had the test error reduced, however, now the full model has a lower test error then AIC or BIC. Lasso still had the lowest test error. Further, AIC and BIC had the same model again, and again the predictors were all in the lasso prediction but lasso again predicted a much larger model of 74 instead of 18.

Next, I performed SVM with linear, quadratic, and gaussian kernels on the whole set of predictors. From this, SVM linear was the best and gaussian was the worst in terms of test error. However, once SVM was repeated on the pca components, the SVM linear was the worst of the models in terms of test error and SVM Quad and SVM Gaus had the same test error. However, when checking the exact predictions made by SVM Gaus and SVM Quad, the predictions were not the same, the test error just turned out the same. Also, interesting is that pca improved the svm gaus and quad but worsened the SVM linear.

Next, I performed a tree analysis on the data. The parameters i used, listed above, resulted in a very large tree, as shown in Figure 1. Figure 1 shows a very complex tree. In fact, after pruning, one would note that the tree over predicts the data based on the test error rate and is more complex then need be. The full tree used 27 nodes. The tree includes many of the same predictors used in the lasso logistic regression and AIC/BIC, but each method also has many of its own unique predictors included. Next, the tree was pruned with deviance as a criteria, which resulted in Figure 2, which has 10 nodes. Figure 2 shows the tree once it is pruned based on deviance. Clearly, the resulting tree is far less complex, however, the test error rate is the exact same. The tree includes many of the same predictors used in the lasso logistic regression and AIC/BIC, but each method also has many of its own unique predictors included. Next, the tree was pruned based on misclassification rate, this is shown in Figure 3. Figure 3 shows the tree resulting from pruning the full tree with misclassification as a criteria. This resulted in a tree with 12 nodes. However, the test error rate went down compared to the full or deviance tree. The tree includes many of the same predictors used in the lasso logistic regression and AIC/BIC, but each method also has many of its own unique predictors included. Next, a random forest of size 500 trees with depth of 4 was created. The error rates determined from the random forest is shown in Figure 4. Figure 4 shows the error as the number of trees in the random forest is increased. Clearly from the figure, the number of trees chosen, 500, was unnecessarily large. From the plot,

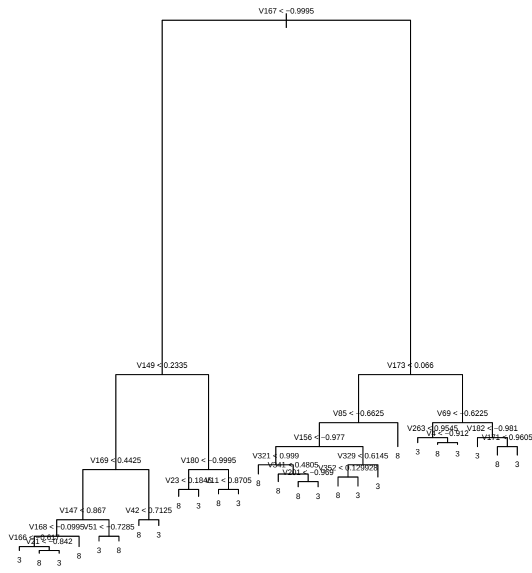


FIG. 1. Full tree created by tree function, 27 nodes were created

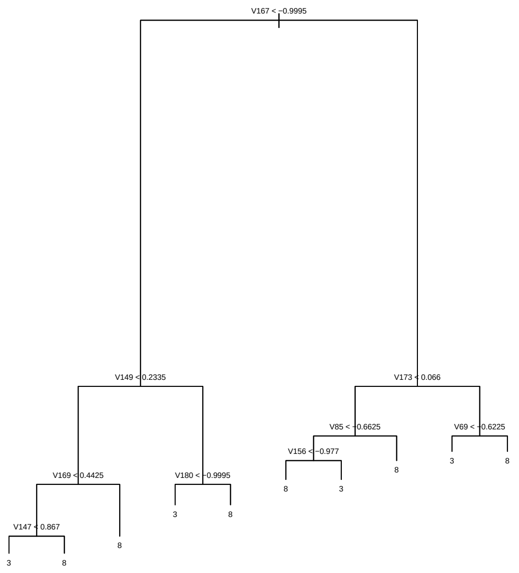


FIG. 2. Tree after pruning with deviance as criteria

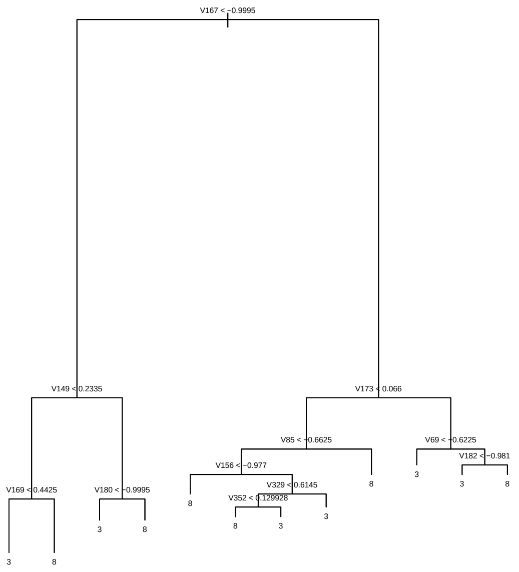


FIG. 3. Tree after pruning with misclass as a criteria

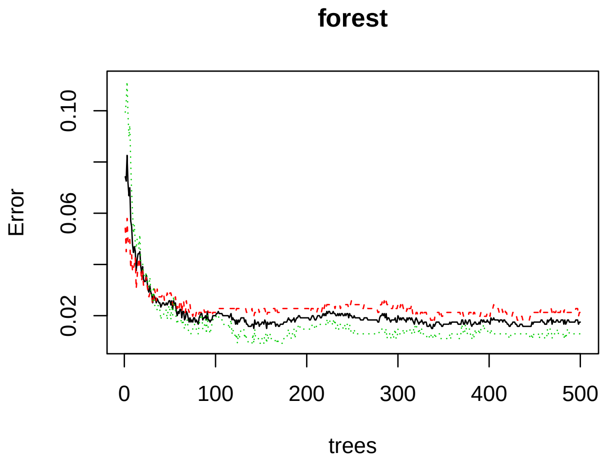


FIG. 4. random forest error plot. This shows the three errors as a function of number of trees created

it is clear that around 100 trees would have sufficed. The test error from the random forest was the lowest of all the methods other then the methods with the same error rate, such as lasso, SVM linear, PCA SVM quad, and others. The number of parameters chosen at each split was the square root of p. However, I checked the predictions, and they merely have the same test error rate, the actual predictions differ across the methods. I checked this to see if any of the methods created similar models, but this was not the case. Next, the importance plots were created for the random forest, as shown in Figure 5. Figure 5 shows the importance plots for the random forest. The importance plot shows how much the accuracy decreases of the trees when certain predictors are included. The first thing to note, is that most of the top predictors listed in the importance plot are apart of the models predicted by lasso and AIC/BIC. However, some of the top predictors are not included in the lasso model, such as V423. Further, many of the nodes in the pruned trees are listed as top predictors in the importance plot, but some nodes are not. There is great overlap between these models, but each one has some deviation from one another, resulting in the difference in predictions of the models.

Lastly, gbm was used to study boosting of the trees. Initially, I use 1000 boosting iterations. I also loop over a few different shrinkage values, (1, .5, .1, .05, .01, .001). The shrinkage value determines how influential the previous trees results are considered in the next iteration of trees. I tried both bootstrapping and not bootstrapping, however, I did not see much difference in the two, so for the final results I only reported no bootstrapping. To choose the optimal number of iterations I use gbm.perf. There is a clear relation between the optimal number of iterations and the shrinkage parameter. The smaller the shrinkage parameter, the higher the number of iterations,

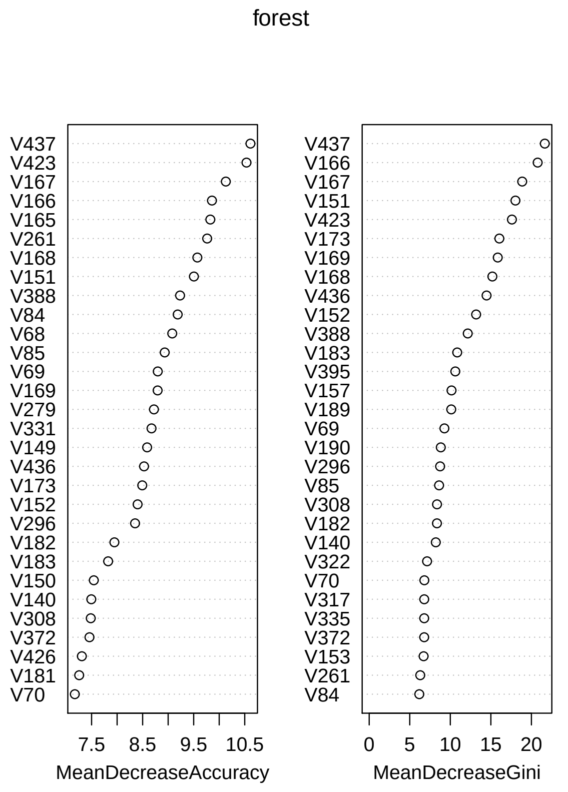
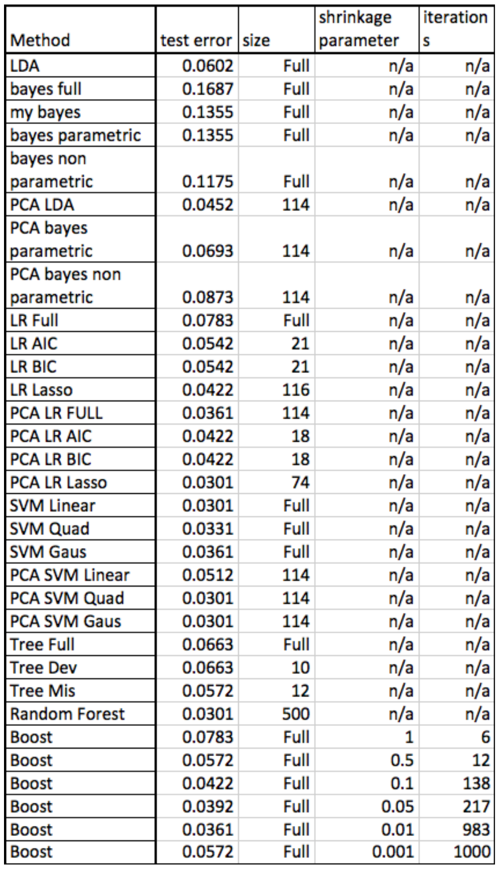


FIG. 5. Random Forest Importance Plot

and the smaller the test error, until the optimal number of iterations exceeds the initial number of iterations provided.

To summarize Table 6 is included.

FIG. 6. table to summarize the errors and tuning parameters of the different methods use



| Method | test error | size | shrinkage parameter | iterations |
|---|---|---|---|---|
| LDA | 0.0602 | Full | n/a | n/a |
| bayes full | 0.1687 | Full | n/a | n/a |
| my bayes | 0.1355 | Full | n/a | n/a |
| bayes parametric | 0.1355 | Full | n/a | n/a |
| bayes non parametric | 0.1175 | Full | n/a | n/a |
| PCA LDA | 0.0452 | 114 | n/a | n/a |
| PCA bayes parametric | 0.0693 | 114 | n/a | n/a |
| PCA bayes non parametric | 0.0873 | 114 | n/a | n/a |
| LR Full | 0.0783 | Full | n/a | n/a |
| LR AIC | 0.0542 | 21 | n/a | n/a |
| LR BIC | 0.0542 | 21 | n/a | n/a |
| LR Lasso | 0.0422 | 116 | n/a | n/a |
| PCA LR FULL | 0.0361 | 114 | n/a | n/a |
| PCA LR AIC | 0.0422 | 18 | n/a | n/a |
| PCA LR BIC | 0.0422 | 18 | n/a | n/a |
| PCA LR Lasso | 0.0301 | 74 | n/a | n/a |
| SVM Linear | 0.0301 | Full | n/a | n/a |
| SVM Quad | 0.0331 | Full | n/a | n/a |
| SVM Gaus | 0.0361 | Full | n/a | n/a |
| PCA SVM Linear | 0.0512 | 114 | n/a | n/a |
| PCA SVM Quad | 0.0301 | 114 | n/a | n/a |
| PCA SVM Gaus | 0.0301 | 114 | n/a | n/a |
| Tree Full | 0.0663 | Full | n/a | n/a |
| Tree Dev | 0.0663 | 10 | n/a | n/a |
| Tree Mis | 0.0572 | 12 | n/a | n/a |
| Random Forest | 0.0301 | 500 | n/a | n/a |
| Boost | 0.0783 | Full | 1 | 6 |
| Boost | 0.0572 | Full | 0.5 | 12 |
| Boost | 0.0422 | Full | 0.1 | 138 |
| Boost | 0.0392 | Full | 0.05 | 217 |
| Boost | 0.0361 | Full | 0.01 | 983 |
| Boost | 0.0572 | Full | 0.001 | 1000 |