Ensemble Methods

Assignment:

Using Pima Indians diabetes data set predict the class of diabetes.

First, I prepared the environment, loaded required libraries and the dataset:

```
> ### Assignment:   Pima Indians Diabetes Dataset
>
> rm(list=ls()) #Clear the environment
> setwd("YOUR_PATH") #Set working directory for the assignment
> getwd() #Check working directory
[1] "YOUR_PATH"
>
> ###Load packages
> library(mlbench)
> library(caret)
Loading required package: lattice
Loading required package: ggplot2
> library(adabag)
Loading required package: rpart
Loading required package: foreach
Loading required package: doParallel
Loading required package: iterators
Loading required package: parallel
> library(fastAdaboost)

>
> ####Load data
>
> data(PimaIndiansDiabetes)
> #Save under a different name to keep separate from the original data file
> diabetes <- PimaIndiansDiabetes
```

The dataset contains 768 observations of 9 variables (number of times pregnant, plasma glucose concentration a 2 hours in an oral glucose tolerance test, diastolic blood pressure (mm Hg), triceps skinfold thickness (mm), 2-hour serum insulin (mu U/ml), body mass index (weight in kg/(height in m)^2), diabetes pedigree function, age (years), and class variable (0 or 1)). All variables are numeric, except for the diabetes column, which is a factor with two levels – negative and positive.

```
> str(diabetes) #df structure
'data.frame':      768 obs. of  9 variables:
 $ pregnant: num  6 1 8 1 0 5 3 10 2 8 ...
```

```
 $ glucose : num  148 85 183 89 137 116 78 115 197 125 ...
 $ pressure: num  72 66 64 66 40 74 50 0 70 96 ...
 $ triceps : num  35 29 0 23 35 0 32 0 45 0 ...
 $ insulin : num  0 0 0 94 168 0 88 0 543 0 ...
 $ mass    : num  33.6 26.6 23.3 28.1 43.1 25.6 31 35.3 30.5 0 ...
 $ pedigree: num  0.627 0.351 0.672 0.167 2.288 ...
 $ age     : num  50 31 32 21 33 30 26 29 53 54 ...
 $ diabetes: Factor w/ 2 levels "neg","pos": 2 1 2 1 2 1 2 1 2 2 ...
```

Diabetes column is going to be the target variable for modeling, and I renamed it to class:

```
> colnames(diabetes)[9] <- "class" #change name for the target variable to class
```

Before proceeding with ensemble learning, I performed some exploratory data analysis to familiarize myself with the data.

I checked the summary statistics for all variables:

```
> summary(diabetes) #summary for all variables
    pregnant         glucose         pressure         triceps         insulin           mass      
 Min.   : 0.000   Min.   :  0.0   Min.   :  0.00   Min.   : 0.00   Min.   :  0.0   Min.   : 0.00  
 1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 62.00   1st Qu.: 0.00   1st Qu.:  0.0   1st Qu.:27.30  
 Median : 3.000   Median :117.0   Median : 72.00   Median :23.00   Median : 30.5   Median :32.00  
 Mean   : 3.845   Mean   :120.9   Mean   : 69.11   Mean   :20.54   Mean   : 79.8   Mean   :31.99  
 3rd Qu.: 6.000   3rd Qu.:140.2   3rd Qu.: 80.00   3rd Qu.:32.00   3rd Qu.:127.2   3rd Qu.:36.60  
 Max.   :17.000   Max.   :199.0   Max.   :122.00   Max.   :99.00   Max.   :846.0   Max.   :67.10  
    pedigree           age        class   
 Min.   :0.0780   Min.   :21.00   neg:500  
 1st Qu.:0.2437   1st Qu.:24.00   pos:268  
 Median :0.3725   Median :29.00            
 Mean   :0.4719   Mean   :33.24            
 3rd Qu.:0.6262   3rd Qu.:41.00            
 Max.   :2.4200   Max.   :81.00            
> 
```

I looked at the first few rows of data:

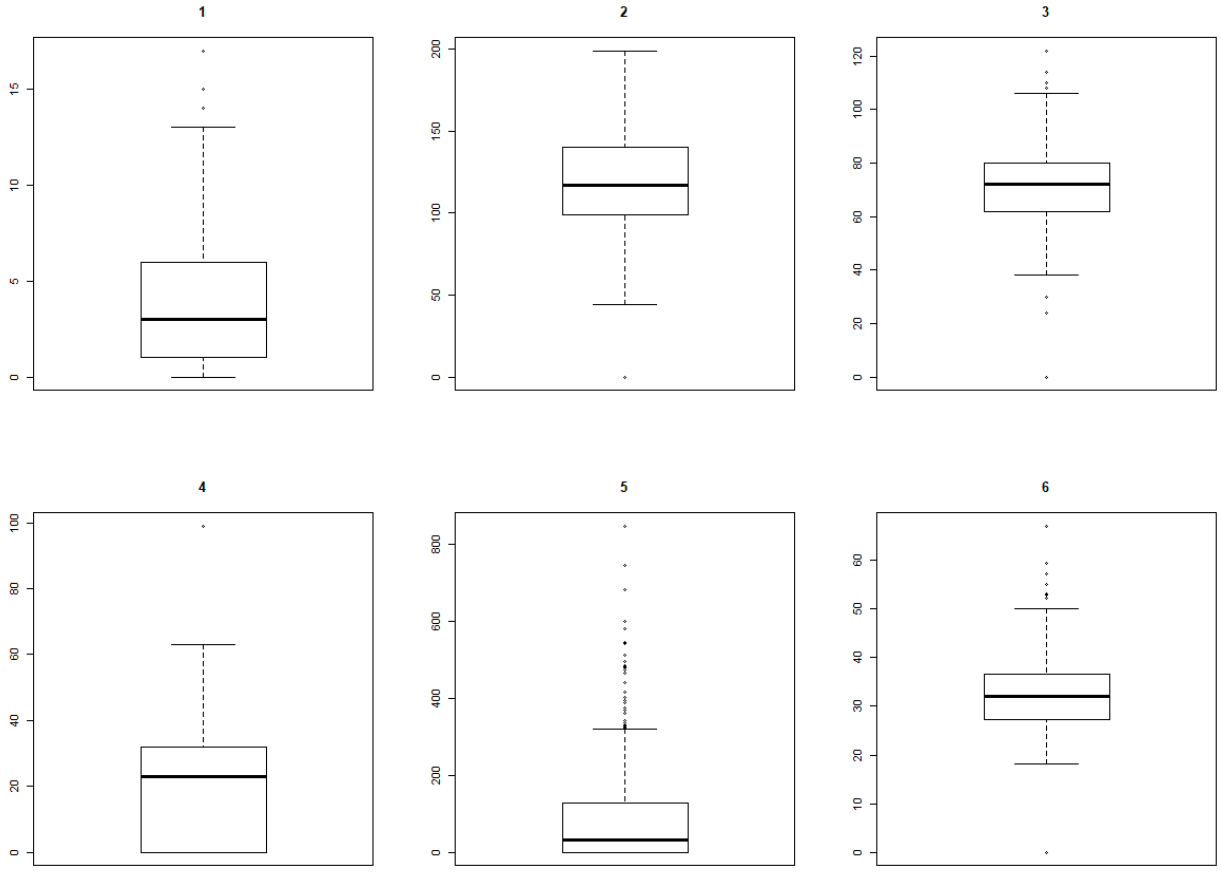
```
> head(diabetes) #First few rows of data
  pregnant glucose pressure triceps insulin mass pedigree age class
1        6     148       72      35       0 33.6    0.627  50   pos
2        1      85       66      29       0 26.6    0.351  31   neg
3        8     183       64       0       0 23.3    0.672  32   pos
4        1      89       66      23      94 28.1    0.167  21   neg
5        0     137       40      35     168 43.1    2.288  33   pos
6        5     116       74       0       0 25.6    0.201  30   neg
```

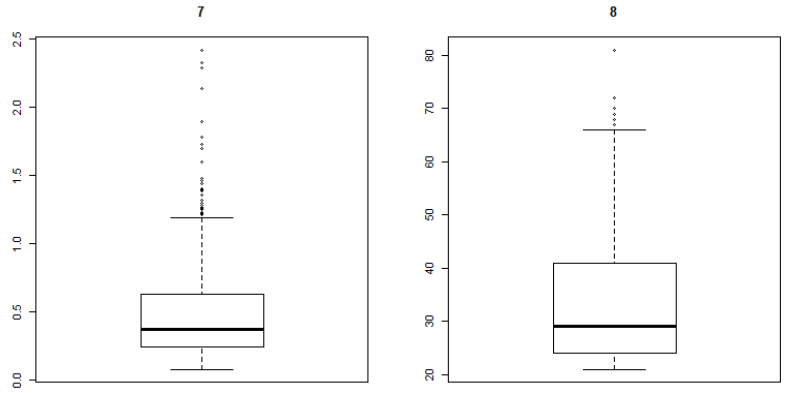
I checked for missing values, and found none:

```
> sum(is.na(diabetes)) #Missing values?
[1] 0
```

I also looked at boxplots for all numerical variables in the dataset:

```
> #Boxplots for all numerical variables in the dataset
> par(mfrow=c(2,3))
> for (i in 1:8){
+   boxplot(diabetes[i], main=i)
+ }
```





This analysis did not yield any unusual findings (e.g., values outside of logical boundaries), or missing data. None of the variables is following the classical normal distribution, and they all use different scales and contain a considerable number of outliers. However, since I am planning on applying tree-based classifications methods, this should not negatively impact the results. Overall, the dataset does not require any additional preprocessing and it is ready for the next step.

I checked proportions for the target class in the entire dataset:

```
> #check proortions for the target variable
> prop.table(table(diabetes$class))

      neg       pos
0.6510417 0.3489583
```

It contains about 65,10% observations negative to diabetes, and about 34.90% belong to people who do have diabetes. Then, I split the dataset into training and testing parts with 80:20 ration, and checked the proportion of negative and positive observations in the resulting subsets:

```
> ###Split the dataset into training and testing data
> set.seed(123)
> ind <- sample(2,nrow(diabetes),replace=TRUE,prob=c(0.7,0.3))
> train_set <- diabetes[ind==1,]
> test_set <- diabetes[ind==2,]
>
> #check proportions in the training and testing sets
> prop.table(table(train_set$class))

      neg       pos
```

```
0.6586271 0.3413729
> prop.table(table(test_set$class))

      neg       pos
0.6331878 0.3668122
```

The resulting training set contains 539 observations, 65.86% of which are negative to diabetes and 34.14% are positive to diabetes. The resulting test dataset is composed of 229 observations, split 63.32% negative to diabetes and 36.68% positive to diabetes. These proportions in the training and testing datasets are close enough to the class distribution in the initial dataset and suitable for model building.

Next, I moved to the main step of applying ensemble techniques for predicting diabetes status using ensemble methods.

First, I applied the bagging method using the Adabag package.

I created a bagging ensemble with 20 iterations using the following code:

```
> #create the bagging ensemble  with 20 iterations
> diabetes_bagging <- bagging(class~ ., data=train_set, mfinal=20)
```

As the output below shows, the two most important factors in this model are glucose and BMI, followed by age, diabetes pedigree, number of pregnancies, blood pressure, insulin and finally triceps skin thickness:

```
> diabetes_bagging$importance
      age   glucose   insulin      mass  pedigree  pregnant  pressure   triceps
 7.845284 53.233877  2.218421 21.279858  5.893884  4.321245  3.868461  1.338969
```

Next, I used the model to predict the diabetes status on test data set:

```
> #evaluate the model on the test set using predict()
> diabetes_predict.baggig <- predict(diabetes_bagging, newdata = test_set)
```

With the following results:

```
> #classification table using predicted results
> diabetes_predict.baggig$confusion
               Observed Class
Predicted Class neg pos
            neg 119  38
            pos  26  46
```

The bagging method correctly identified 119 negative observations and 46 positive observations in the dataset. A more convenient way to evaluate the results is by looking at the confusion matrix and additional statistics:

```
> #cross-tabulation of observed and predicted classes
> diabetes_predict.bagcfm <- confusionMatrix(table(test_set$class, diabetes_predict.baggig$class)
)
> #Display results
> diabetes_predict.bagcfm
Confusion Matrix and Statistics


      neg pos
  neg 119  26
  pos  38  46

               Accuracy : 0.7205
                 95% CI : (0.6576, 0.7776)
    No Information Rate : 0.6856
    P-Value [Acc > NIR] : 0.1426

                  Kappa : 0.3797
 Mcnemar's Test P-Value : 0.1691

            Sensitivity : 0.7580
            Specificity : 0.6389
         Pos Pred Value : 0.8207
         Neg Pred Value : 0.5476
             Prevalence : 0.6856
         Detection Rate : 0.5197
   Detection Prevalence : 0.6332
      Balanced Accuracy : 0.6984

       'Positive' Class : neg
```

As the output above shows, the overall accuracy of the bagging approach on the test dataset was 72.05 %, however, there is a significant difference between the positive predictive value (82.07%) and the negative predictive value (54.76%). Note, these statistics assumed that positive class is the absence of diabetes, and the negative class is the presence of diabetes.

I also retrieved the average error of the bagging results, which was equal to 0.2789476.

```
> #average error of bagging results
> diabetes_predict.baggig$error
[1] 0.279476
```

Next, I performed 10-fold cross validation using bagging and bagging.cv() function from the adabag package. This function divided the observation into v=10 non-overlapping subsets of roughly equal size and applied bagging to 9 of the subsets. Then if made predication based on the left out subsets. The process is repeated for each of the 10 subsets. So, training dataset (and not the test dataset) is used in the following code:

```
> ###10-fold cross validation using bagging
> diabetes_baggingcv <- bagging.cv(class ~., v=10, data = train_set, mfinal =
20)
```

This method allowed to correctly detect 311 negative observations and 115 positive observations, while misclassifying 113 observations. In order to be able to compare accuracy to the previous model, I constructed a full confusion matrix:

```
> #confusion matrix
> confusionMatrix(diabetes_baggingcv$confusion)
Confusion Matrix and Statistics

               Observed Class
Predicted Class neg pos
            neg 311  69
            pos  44 115

               Accuracy : 0.7904
                 95% CI : (0.7535, 0.824)
    No Information Rate : 0.6586
    P-Value [Acc > NIR] : 1.336e-11

                  Kappa : 0.518
 Mcnemar's Test P-Value : 0.02396

            Sensitivity : 0.8761
            Specificity : 0.6250
         Pos Pred Value : 0.8184
         Neg Pred Value : 0.7233
             Prevalence : 0.6586
         Detection Rate : 0.5770
   Detection Prevalence : 0.7050
```

```
Balanced Accuracy : 0.7505

 'Positive' Class : neg
```

10-fold cross-validation allowed to improve overall classification accuracy to 79.04 % (compare to 72.05% with bagging). The negative predictive value increased considerably from 54.76% to 72.33% with positive predictive value remaining close to the previous indicator (81.84% compared to 82.07%). Balanced accuracy, which talks into consideration probability distribution between classes, improved from 69.84% for bagging to 75.05% with 10-fold cross-validation.

```
> #estimation error from the cross-validation results (train_set)
> diabetes_baggingcv$error
[1] 0.2096475
```

Similarly, estimation error with 10-fold cross validation is 0.02096475, with is lower than 0.279476 in case of bagging.

I also tried an alternative approach to 10-fold cross validation using the caret package in order to be able to apply it to the testing data set.

First, I configured training parameters and fit the predictive model over the tuning parameters using the following code:

```
> ###Using caret package to perform 10-fold cross validation
>
> #Perform cross-validation using caret package
> trainctrl <- trainControl(method="cv", number=10) #10-fold cross-validation
> #fit predictive model over the tuning parameters
> bag_caretcv <- train(class~., data=train_set, method="treebag", trControl=trainctrl)
```

Then, I applied it to the testing data set with the following results presented as the confusion matrix:

```
> #apply predictive model to the test set
> prediction_caretbagcv <-predict(bag_caretcv, newdata = test_set)
> #confusion matrix for 10-fold cross-validation using caret
```

```
> confusionMatrix(table(test_set$class, prediction_caretbagcv))
Confusion Matrix and Statistics

     prediction_caretbagcv
      neg pos
  neg 117  28
  pos  38  46

               Accuracy : 0.7118
                 95% CI : (0.6485, 0.7695)
    No Information Rate : 0.6769
    P-Value [Acc > NIR] : 0.1444

                  Kappa : 0.3636
 Mcnemar's Test P-Value : 0.2679

            Sensitivity : 0.7548
            Specificity : 0.6216
         Pos Pred Value : 0.8069
         Neg Pred Value : 0.5476
             Prevalence : 0.6769
         Detection Rate : 0.5109
   Detection Prevalence : 0.6332
      Balanced Accuracy : 0.6882

       'Positive' Class : neg
```

Surprisingly, this approach demonstrated slightly worse results on the testing dataset than bagging. The 10-fold cross-validation with caret had 71,18 % overall accuracy in detecting presence or absence of diabetes on the testing observations compared to 72.05% accuracy of bagging. The positive predictive values decreased from 82.07% to 80.69% , and the negative predictive value remained the same at the 54.76%. The balanced accuracy taking into consideration probability distribution between classes for 10-fold cross-validation with caret on testing dataset was 68.82% which is about one percent less than the bagging balanced accuracy of 69.84%.

Next, I used the boosting method. I trained a boosting classification model on the training data set using the following code:

```
> ###Boosting
> set.seed(123)
```

```
> #train boosting classification model
> diabetes.boost <- boosting(class ~., data=train_set, mfinal=20, coeflearn = "Freund", boos=FAL
SE, control=rpart.control(maxdepth=3))
```

Then, I used it to make predictions on the testing data set:

```
> #make predictions on the test set using the boosted model
> diabetes.boost.pred <- predict.boosting(diabetes.boost, newdata = test_set)
```

This approach allowed to correctly classify 118 negative observations and 53 positive observations, while 58 observations were misclassified:

```
> #retrieve classification table with prediction results
> diabetes.boost.pred$confusion
               Observed Class
Predicted Class neg pos
            neg 118  31
            pos  27  53
```

The following confusion matrix demonstrates more detailed statistics:

```
> #confusion matrix and other statistics
> confusionMatrix(diabetes.boost.pred$confusion)
Confusion Matrix and Statistics

               Observed Class
Predicted Class neg pos
            neg 118  31
            pos  27  53

               Accuracy : 0.7467
                 95% CI : (0.6852, 0.8017)
    No Information Rate : 0.6332
    P-Value [Acc > NIR] : 0.000168

                  Kappa : 0.4492
 Mcnemar's Test P-Value : 0.693641

            Sensitivity : 0.8138
            Specificity : 0.6310
         Pos Pred Value : 0.7919
         Neg Pred Value : 0.6625
             Prevalence : 0.6332
         Detection Rate : 0.5153
   Detection Prevalence : 0.6507
      Balanced Accuracy : 0.7224

       'Positive' Class : neg
```

The overall accuracy of the boosting method demonstrated on the testing dataset was 74.67% which is higher than for bagging and 10-fold cross validation using caret, but lower than results obtained with 10-fold cross validation using the adabag package on the training dataset.

Compared to bagging, boosting showed slightly lower positive predictive value 79.19% compared to 82.07%, but demonstrated improvements in the negative predictive value (66.25% compared to 54.76% for bagging). The balanced accuracy for boosting was 72.24% which is higher than for bagging or 10-fold cross-validation with caret.

The average error for predicted results with boosting was 0.2532751, which is lower than in case of bagging (0.278476).

Next, I performed 10-fold cross-validation with boosting.

10-fold cross-validation applied to tee training dataset produced the following results:

```
> #cross-validate training dataset
> diabetes.boostcv <- boosting.cv(class ~ ., v=10, data = train_set, mfinal = 5, control = rpart.
control(cp=0.01))
i:  1 Sun Nov 18 12:55:41 2018
i:  2 Sun Nov 18 12:55:43 2018
i:  3 Sun Nov 18 12:55:44 2018
i:  4 Sun Nov 18 12:55:45 2018
i:  5 Sun Nov 18 12:55:47 2018
i:  6 Sun Nov 18 12:55:48 2018
i:  7 Sun Nov 18 12:55:50 2018
i:  8 Sun Nov 18 12:55:51 2018
i:  9 Sun Nov 18 12:55:52 2018
i:  10 Sun Nov 18 12:55:54 2018
> #obtain classification results
> diabetes.boostcv$confusion
                Observed Class
Predicted Class neg pos
            neg 286  78
            pos  69 106
```

286 negative observations and 106 positive observation from the training set were classified correctly, and 147 observations were misclassified.

```
> #confusion matrix and other statistics
```

```
> confusionMatrix(diabetes.boostcv$confusion)
Confusion Matrix and Statistics

               Observed Class
Predicted Class neg pos
            neg 286  78
            pos  69 106

               Accuracy : 0.7273
                 95% CI : (0.6876, 0.7645)
    No Information Rate : 0.6586
    P-Value [Acc > NIR] : 0.0003738

                  Kappa : 0.3863
 Mcnemar's Test P-Value : 0.5093636

            Sensitivity : 0.8056
            Specificity : 0.5761
         Pos Pred Value : 0.7857
         Neg Pred Value : 0.6057
             Prevalence : 0.6586
         Detection Rate : 0.5306
   Detection Prevalence : 0.6753
      Balanced Accuracy : 0.6909

       'Positive' Class : neg
```

As the confusion matrix above shows, the overall accuracy, in this case, was 72.73%, which is lower than boosting on the testing set (74.67%). All other indicators, including sensitivity, specificity, positive and negative predictive values for 10-fold cross validation with boosting on the train set were worse than in case of previous boosting results on the test dataset. The balanced accuracy also decreased to 69.09% from 72.24%. At the same time the average error went up to 0.2727273 (from 0.2532751):

```
> #average errors
> diabetes.boostcv$error
[1] 0.2727273
```

If we compare these results to the 10-fold cross validation and bagging using adabag package previously applied to the training dataset, boosting cross validation model showed about 5% worse accuracy than bagging.

In order to be able to compare results on the testing data set, I also used an alternative approach to 10-fold cross validation with boosting using the caret package.

```
> #Perform cross-validation using caret package
> trainctrl2 <- trainControl(method="cv", number=10) #10-fold cross-validation
> #fit predictive model over the tuning parameters
> boost_caretcv <- train(class~., data=train_set, method="adaboost", trControl=trainctrl2)
```

I applied the model to the testing set:

```
> #apply predictive model to the test set
> prediction_caretboostcv <-predict(boost_caretcv, newdata = test_set)
```

With the following results presented as a confusion matrix:

```
> #confusion matrix for 10-fold cross-validation using caret
> confusionMatrix(table(test_set$class, prediction_caretboostcv))
Confusion Matrix and Statistics

     prediction_caretboostcv
      neg pos
  neg 127  18
  pos  36  48

               Accuracy : 0.7642
                 95% CI : (0.7038, 0.8176)
    No Information Rate : 0.7118
    P-Value [Acc > NIR] : 0.04467

                  Kappa : 0.4684
 Mcnemar's Test P-Value : 0.02070

            Sensitivity : 0.7791
            Specificity : 0.7273
         Pos Pred Value : 0.8759
         Neg Pred Value : 0.5714
             Prevalence : 0.7118
         Detection Rate : 0.5546
   Detection Prevalence : 0.6332
      Balanced Accuracy : 0.7532

       'Positive' Class : neg
```

The overall accuracy of this approach on the testing dataset turned out to be 769.42% which is the best result among all methods applied to the testing dataset. The positive predictive value was 87.59%, however the negative predictive value was only 57,14% which is lower than

in the case of boosting 66.25%. Taking into consideration class distribution in the dataset, the balanced accuracy was 75.32% which is higher than for any other approach.

Comparing bagging and boosting methods, when tested on the unseen dataset, boosting showed about 2.5% higher accuracy than bagging (74.67% against 75.05%). Boosting was better at detecting both observations with and without the diabetes diagnosis with lower average prediction error. Applying 10-fold cross-validation using caret package to the bagging method, surprisingly, did not improve its accuracy. In fact, it decreased overall accuracy for about one percent. On the other hand, 10-fold cross validation using the caret package allowed to further improve boosting results. The accuracy of predictions in the testing dataset increased to 76.42% with improvements especially in detection for the positive class (in our case it was the absence of diabetes). Overall, the boosting with 10-fold cross-validation showed the most accurate results on the testing dataset.

I would also like to note, that I preferred using caret package for 10-fold cross validation as opposed to cross-validation options of the adabag package because the former allowed to compare predictive accuracy on the testing dataset (not training as with adabag option) and make it directly comparable to the results obtained for bagging and boosting.

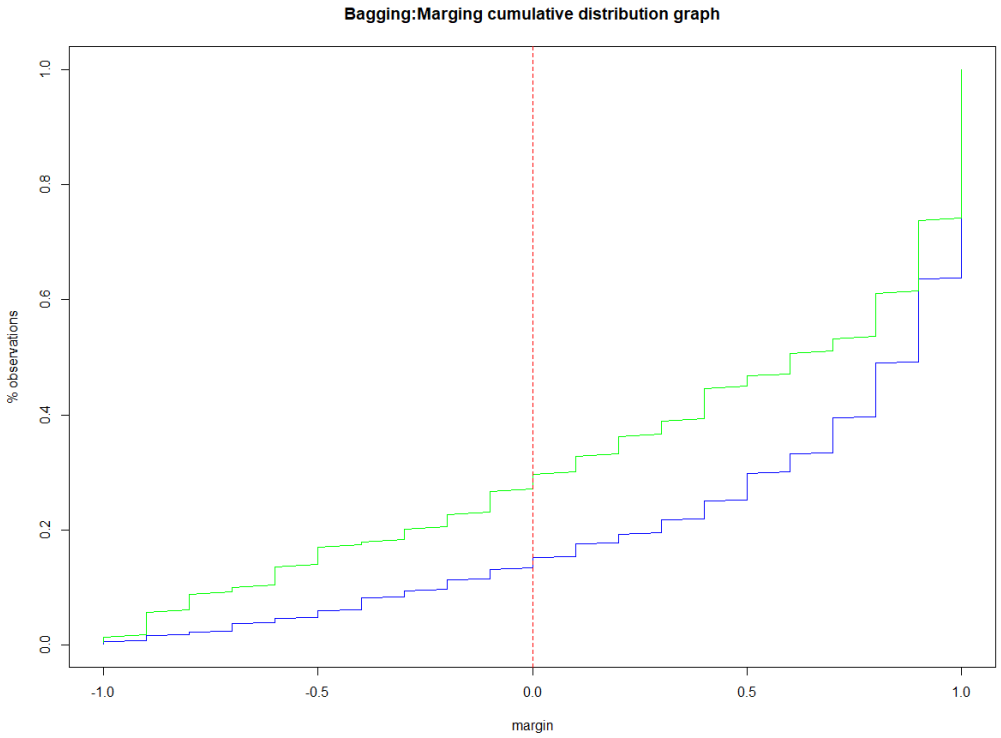
A more formal approach to compare these ensemble techniques is to look at the margin as the measurement of the certainty of the classification. The margin is computed as a difference between the support of correctly classified observations and the maximum support of incorrectly classified observations. If the margin value is close to one, then it means that the correctly

classified examples have a high degree of confidence. If all observations were classified correctly, the graph would be a vertical line equal to one.

The following calculations are based on the suggestion by Chiu (2015).

First, I calculated and plotted margins for the bagging classifier:

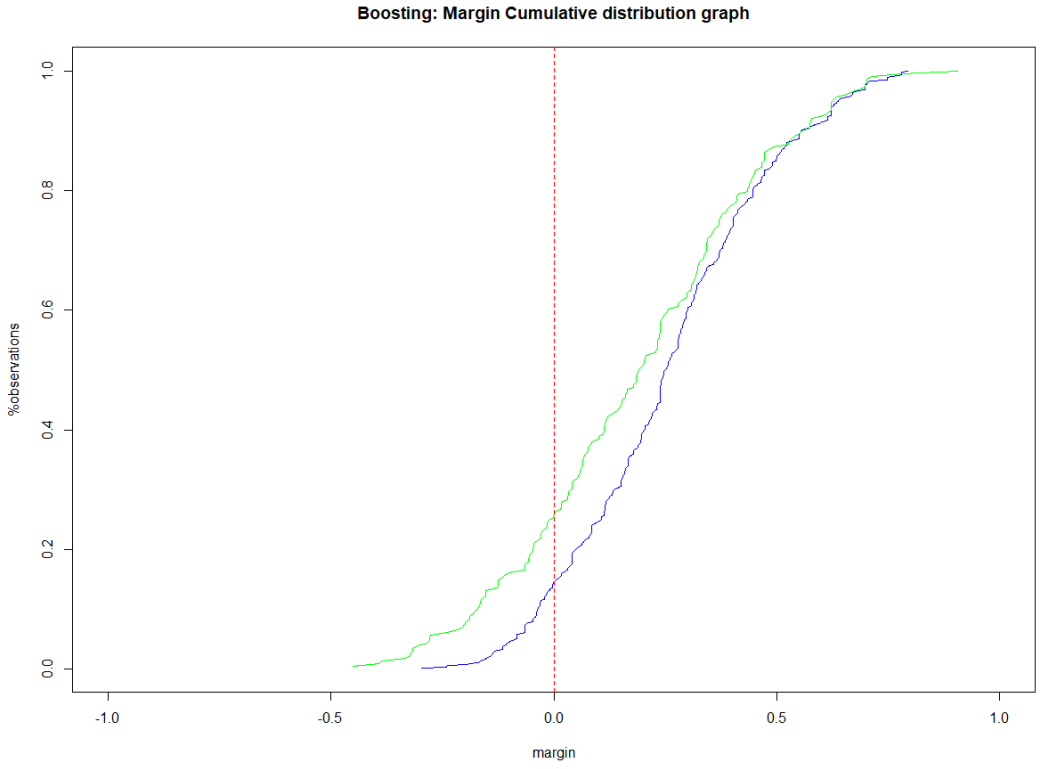
```
> ###Calculating margins of the bagging classifier
>
> bagging_margins <- margins(diabetes_bagging, train_set)
> bagging_predict_margins <- margins(diabetes_predict.baggig, test_set)
> #plot a marginh cumulative distribution graph od the bagging classifiers
> plot(sort(bagging_margins[[1]]), (1:length(bagging_margins[[1]]))/length(bagging_margins[[1]]),
type="l", xlim = c(-1,1), main="Bagging:Marging cumulative distribution graph", xlab="margin", yl
ab="% observations", col="blue")
> lines(sort(bagging_predict_margins[[1]]), (1:length(bagging_predict_margins[[1]]))/length(baggi
ng_predict_margins[[1]]), type="l", col="green")
> abline(v=0, col="red", lty=2)
```



Then, I followed the same procedure for calculating margins for the boosting classifier.

```
> #Calculating the margins of the boosting classifier
> boost_margins <-margins(diabetes.boost, train_set)
> boost_predict_margins <- margins(diabetes.boost.pred, test_set)
> boost_predict_margins <- margins(diabetes.boost.pred, test_set)
> #plot marginal cumulative distribution for the boosting classifier
> par(mfrow=c(1,1))
```

```
> #plot marginal cumulative distribution for the boosting classifier
> par(mfrow=c(1,1))
> plot(sort(boost_margins[[1]]), (1:length(boost_margins[[1]]))/length(boost_margins[[1]]), type
="l", xlim=c(-1,1), main="Boosting: Margin Cumulative distribution graph", xlab="margin", ylab="%
observations", col="blue")
> lines(sort(boost_predict_margins[[1]]), (1:length(boost_predict_margins[[1]]))/length(boost_pre
dict_margins[[1]]), type ="l", col="green")
> abline(v=0, col="red", lty=2)
```



Comparison of these two graphs shows, that both classifiers have slightly higher margins when applied to test datasets (plotted in green) compared to training sets (plotted in blue), while theoretically, they should be similar. It means that in both cases we could expect performance on the real-life datasets to be slightly different from the indicators calculated during training and testing. In addition, graphs imply slightly higher certainty levels for the boosting classifier. These results are confirmed by the calculation of the percentages of the negative margins for bagging and boosting.

Calculations for the bagging example:

```
> #calculate percentage of negative margin matches training errors
> bagging_training_margin <- table(bagging_margins[[1]]>0)
> bagging_negative_training <- as.numeric(bagging_training_margin[1]/bagging_training_margin[2])
> bagging_negative_training
[1] 0.1794311


> #calculating percentage of negative margin matches testing errors
> bagging_testing_margin <- table(bagging_predict_margins[[1]] >0)
> bagging_negative_testing <- as.numeric(bagging_testing_margin[1]/bagging_testing_margin[2])
> bagging_negative_testing
[1] 0.4223602
```

Calculations for the boosting example:

```
> #percentage of negative margin matches training errors
> boosting_training_margin <- table(boost_margins[[1]] >0)
> boosting_negatvie_training <- as.numeric(boosting_training_margin[1]/boosting_training_margin[2
])
> boosting_negatvie_training
[1] 0.1691974
> #percentage of negative margin matches testing errors
> boosting_testing_margin <- table(boost_predict_margins[[1]] >0)
> boosting_negatvie_testing <- as.numeric(boosting_testing_margin[1]/boosting_testing_margin[2])
> boosting_negatvie_testing
[1] 0.3391813
```

As the output above shows, boosting approach in this assignment demonstrated lower percentages of the negative margins (33.91%) on the testing dataset, compared to 42.36% results for bagging on the testing dataset,

In summary, for predicting diabetes diagnosis boosting approach demonstrated more accurate and reliable results compared to bagging. The best results were obtained when combining boosting with 10-fold cross-validation.

References

Chiu, Y. (2015). Machine Learning with R cookbook. Packt Publishing. Retrieved from http://lumen.regis.edu/record=b1659405~S3.