Feature Selection and the BSWiMS Method

José Gerardo Tamez-Peña

jose.tamezpena@tec.mx
Tecnológico de Monterrey, Escuela de Medicina,
Av. Morones Prieto No. 3000, Colonia Los Doctores CP 64710
Monterrey Nuevo León, México.

Table of Contents

Introduction 2
The BSWiMS procedure 4
    Data Normalization 7
    Model Selection via the BSWiMS method 8
    Diagnosing the BSWiMS Model 9
    Exploring the BSWiMS model 14
Univariate Feature Selection vs BSWiMS 18
References 23
Appendix A: Holdout Cross-Validation and the Ensemble Prediction 24
    Conclusion 27

Introduction

FRESA.CAD is an R package designed to aid health-related scientists in their data exploration efforts. It is specifically aimed to discover novel features or to design practical computer-aided diagnosis or decision support models. The package contains methods for data conditioning, data exploration, univariate filters, model building, model diagnosis, and model visualization as seen in Figure 1. Although there are many ways for a scientist to use the FRESA.CAD tools to explore their data, this vignette will focus on describing how to use the **B**ootstrapping **S**tage-**Wi**se **M**odel **S**election (BSWiMS) method to discover and visualize diagnosis logistic models for a specific disease.

The following sections will describe the method and the basic steps required to conditioning a database, run BSWiMS, display/visualize their outputs and finally how to compare the selected features to popular univariate methods. The following table shows the FRESA.CAD functions that will be showcased in this vignette:

Objective	FRESA.CAD Function	Description
Data Normalization	`FRESAScale(…)`	Case/Control-Based Data Normalization
Model Selection/FS	`BSWiMS.model(…)` `FRESA.Model(…)`	Fit/models the data selecting the best linear/logistic/Cox models.
Model Fitting	`baggedModel(…)`	Fit a set of linear models to the data using the bagging method.
Model Cross-Validation	`randomCV(…)`	Get the model test-performance using repeated holdout cross-validation.
Data Visualization	`heatMaps(…)`	Display the desired features in a heat map.
Model/Feature Description	`summary(…)`	Reports the model coefficient as well as their 95% Confidence intervals.
Model/Feature Description	`reportEquivalentVariables(…)`	Reports all variables that may be used as surrogated features in a model.
Model prediction	`predict(…)`	Predicts the model on the test data.
Model prediction	`ensemblePredict(…)`	Predicts the ensemble of models on the test data.
Performance Evaluation	`predictionStats_binary(…)`	Reports the predict/test performance by comparing the prediction to the ground truth.
Model diagnosis	`plot.bootstrapValidation(…)`	Plot and reports diagnose done by bootstrapping.
Model diagnosis	`plotModels.ROC(…)`	Plot the ROC of the given train/test sets as well as the confusion matrix.
Performance comparison	`barPlotCiError(…)`	Plots and compares the model evaluations using the 95%CI.
Data Analysis	`univariateRankVariables(…)`	Reports the basics descriptors of the data using univariate statistics.
Feature Selection	`univariate_Logit(…)`	Feature selection using univariate logistic models and NRI/IDI.
Feature Selection	`univariate_residual(…)`	Feature selection using univariate logistic or regression models.
Feature Selection	`univariate_tstudent(…)`	Feature selection by t-test.
Feature Selection	`univariate_Wilcoxon(…)`	Feature selection by Wilcoxon-test.
Feature Selection	`univariate_correlation(…)`	Feature selection by Outcome-data correlation.
Feature Selection	`mRMR.classic_FRESA(…)`	Feature selection using classic mRMR.

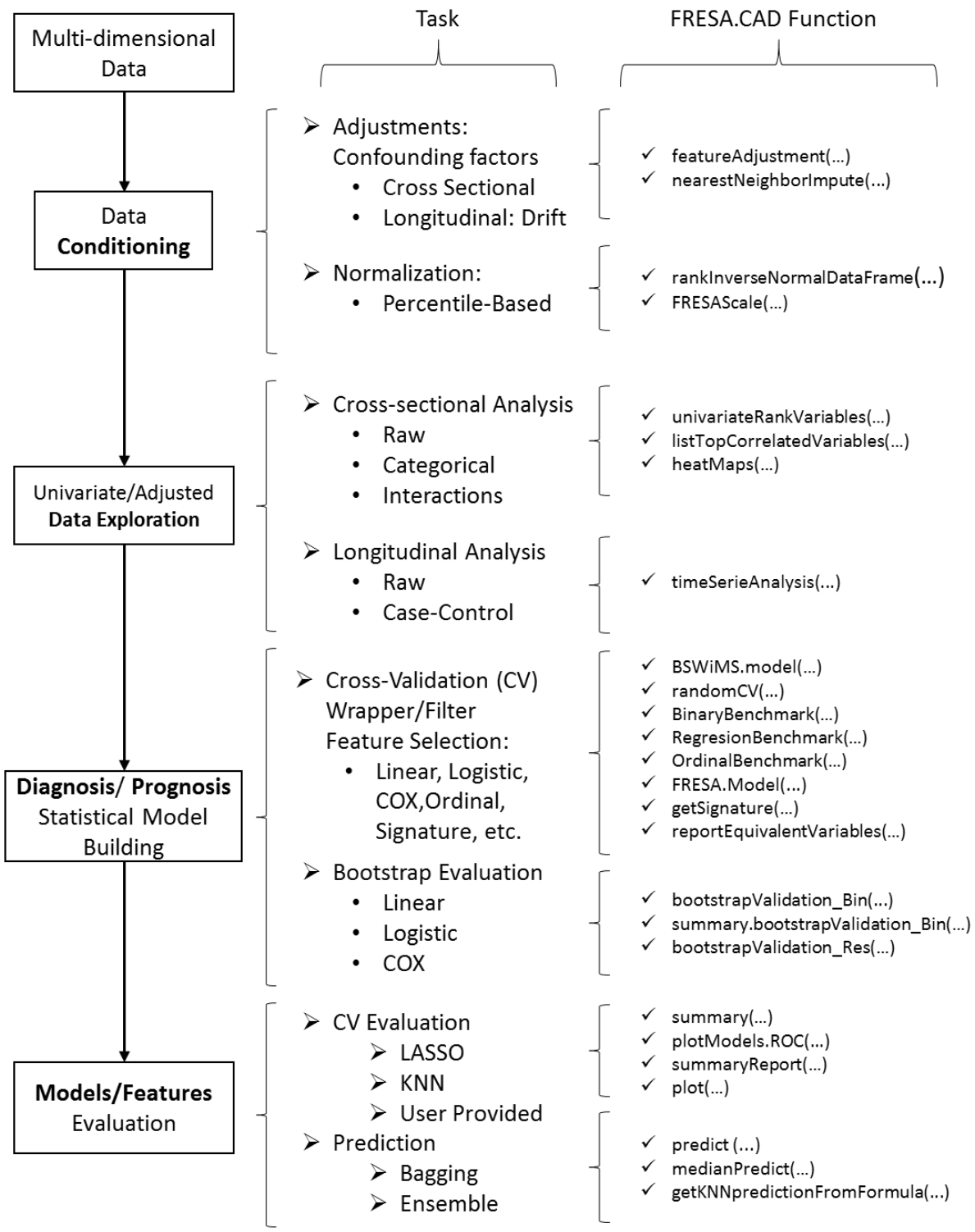


Figure 1: Summary of FRESA.CAD functions

The BSWiMS procedure

BSWiMS is a supervised model-selection method aimed to select a unique statistical model that predicts a user-specified outcome. The statistical model is constructed by bagging a set of compact linear models (model-nuggets), where each model-nugget is built by a unique set of model-wise statistically-significant features. To achieve this goal the algorithm is divided into five different stages: Univariate Filter, Bootstrapped Forward Selection, Frequency-based Forward Selection, Bootstrapped Backwards Elimination and Model Bagging as seen in Figure 2(a). Each stage of the process is designed to select features that are statistically relevant in explaining the desired outcome while trying to keep the false discovery rate (FDR) at the desired level.

The first stage of the BSWiMS is univariate filtering. This step computes the univariate association of each feature to the outcome. Then, the association p-value is FDR adjusted using the Benjamini–Hochberg procedure[1]. Only features above the desired q-value are selected as candidates to build linear models. The second stage builds a set of $\boldsymbol{B}$ linear models using a forward selection procedure, where $\boldsymbol{B}$ is the number of bootstraps samples. Figure 2(b) shows the steps of the approach and it is based on selecting $\boldsymbol{B}$ bootstrap training samples and their corresponding out of bag validation sets (OOB)[2]. For each train set, the algorithm starts by selecting the feature with the smallest OOB and train fitness improvement p-values. If the feature-size adjusted p-value is statistically significant, then the feature is added to the model and the selected feature is removed from the feature set. The procedure is repeated until no more significant features can be added to the model. The third stage is the frequency-based forward selection. To generate a single model from the set of bootstrapped formulas, the bootstrapped models' features are ranked from the highest frequency selection to the lowest. After ranking, the forward model is built by step-wise adding the ranked features if and only if the size-adjusted fitting p-value is statistically significant. The fourth stage is the backward elimination. This stage bootstraps the forward model and analyzes the bootstrap distribution of each model term. If the largest OOB size-adjusted p-value or the largest train p-value is not significant for a term member, then the feature is removed from the formula. Figure 2(c) shows the flowchart of the bootstrap stage. The resulting model after the backward elimination stage is a compact linear model; hence, a model-nugget. All terms of a model-nugget are model-wise statistically significant according to the user-selected statistical test that evaluated fitness improvement. In other words, each term of the model-nugget adds unique information, not shared by the other terms and each term is required to improve the model fitness in a statistically significant way. Once a model-nugget is found, the model-nugget features are removed and stages two to fourth are repeated until no more model-nuggets can be found or that the OOB performance of the last nugget is inferior to the first added nugget. After the last nugget is found, the set of nuggets extracted are stored. For selection ranking purposes, the process can be repeated several times. The fifth and last stage of the BSWiMS procedure is to bag all the extracted nuggets into a single statistical model. Model bagging consists are taking the performance-weighted average of the model-nugget coefficients. The bagged model is the BSWiMS model.

The main feature of a BSWiMS model is that every model term is statistically described by the average nugget-fitted fitness statistics and the feature selection frequency. Model coefficients with their 95%Ci are provided, the relevance of each feature is statistically tested and the statistics can be used to explore the relevance of each feature in describing the outcome. A second feature of the procedure is that ensemble predictions can be made using the set of nuggets and this prediction be compared to the ones done by the linear bagged model. This second feature may be useful if the nuggets are not collinear, hence the ensemble procedure may yield better performance than the bagged model.

The following sections will show how to:

- Generate BSWiMS models on control-normalized data sets.
- Diagnose the models.
- Visualize the selected features.
- Cross-validate the models using the cross-validation procedure as seen in Figure 1(d) and 1(e)
- Report the 95% confidence intervals via the bootstrapped bagging procedure.
- Inspect the returned features and

Compare those features to the ones returned by univariate feature selection methods

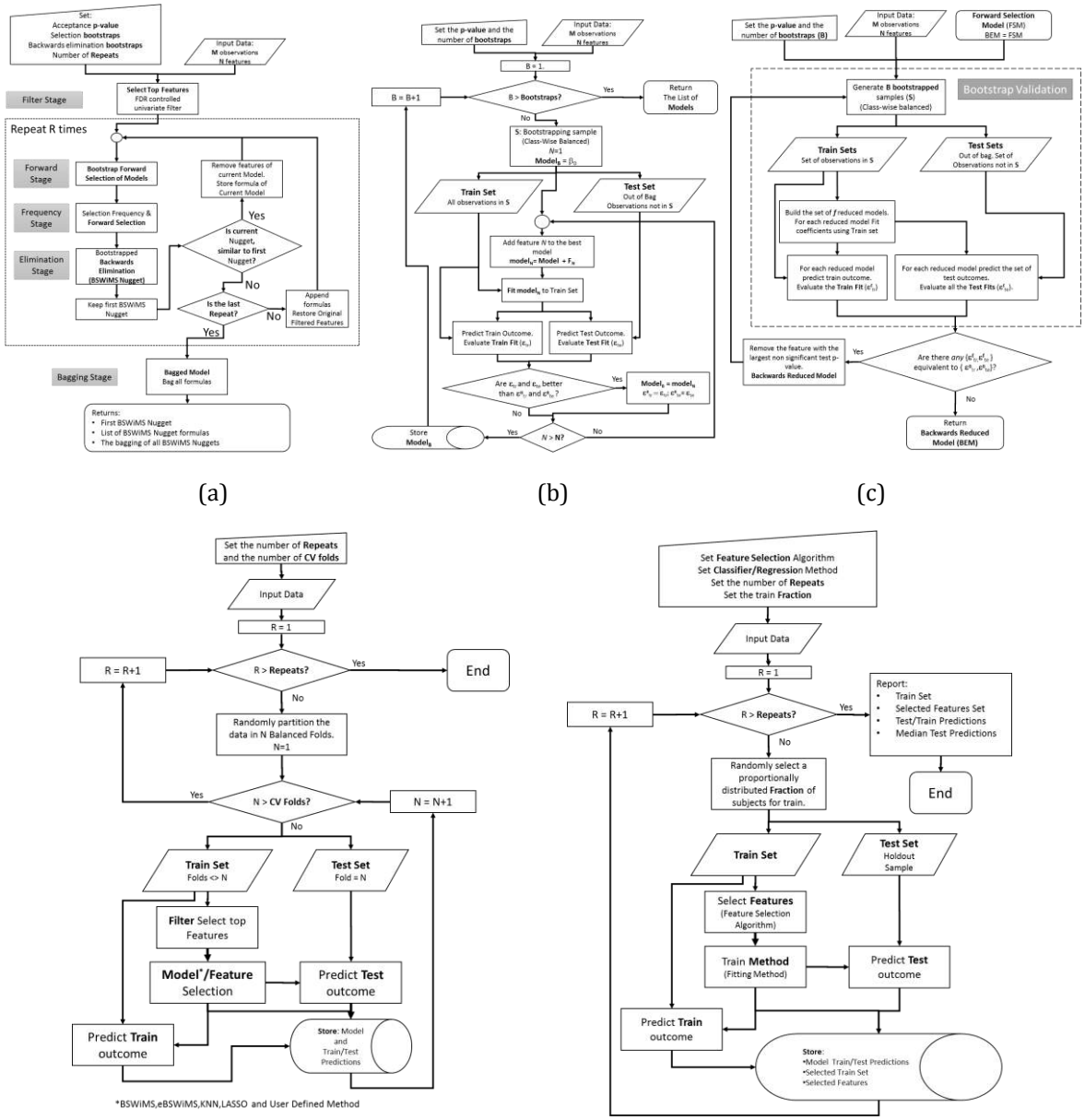


Figure 2: (a) the BSWiMS stages. (b) The bootstrap forward selection workflow. (c) The bootstrap backwards elimination. (d) FRESA.Model k-fold cross-validation. (e) Random holdout cross-validation.

Data Normalization

The data used in this vignette is the GlaucomaM data from the TH.data package[3]. The BSWiMS procedure requires numerical data frames where each row is a sample and each column is a feature. Therefore, the next step after loading the data is to transform it into a numeric data frame where the class levels are set to 0s for controls and 1s for glaucoma patients.

```
R> data(GlaucomaM,package = "TH.data")
R> GlaucomaM_mat <- as.data.frame(model.matrix(Class~.,GlaucomaM)[,-1])
R> GlaucomaM_mat$Class <- 1*(GlaucomaM$Class == "glaucoma")
```

An optional step in model discovery consists on normalize the data. FRESA.CAD has a normalization function that can be used to scale the data based on the observed distribution of the control strata. The `FRESAScale(x, control, method = "RankInv")` function with the "rankInv" method takes the empirical distribution of each feature in the control population to rank them to estimate the percentile of each subject and then it is used to get the corresponding z-value using the inverse transform of the normal distribution. The following code snippet will show how to rank features based on the normal population:

```
R> ALLcontrolSubjects <- subset(GlaucomaM_mat,Class == 0)
R> ALLCaseSubjects <- subset(GlaucomaM_mat,Class == 1)
R> ALLScaled <- FRESAScale(GlaucomaM_mat,refFrame = ALLcontrolSubjects,method = "RankInv")
```

To visualize the relevant features of the normalized data set we use the `univariate_Logit(…)` and the `heatMaps(…)` function. The first function computes the p-value of the logistic model fit, ranks the p-values, adjust them using the Benjamini–Hochberg FDR procedure, and finally return the features above the specified threshold. The second function wraps the `gplots::heatmap.2(…)` and tailors it to the needs of visualizing labeled data sets. Figure 2 shows the results of the following code:

```
R> q_valuesALL <- univariate_Logit(data = ALLScaled$scaledData,
                                   Outcome = "Class",
                                   pvalue = 1.0e-6)


R>  hm <- heatMaps(Outcome = "Class",
                   data = ALLScaled$scaledData[,c("Class",names(q_valuesALL))],
                   title = "Heat Map: Normal Inverse (ALL)",Scale = c(-2,2),
                   hCluster = "col",cexRow = 0.25,cexCol = 0.15,srtCol = 45)
```

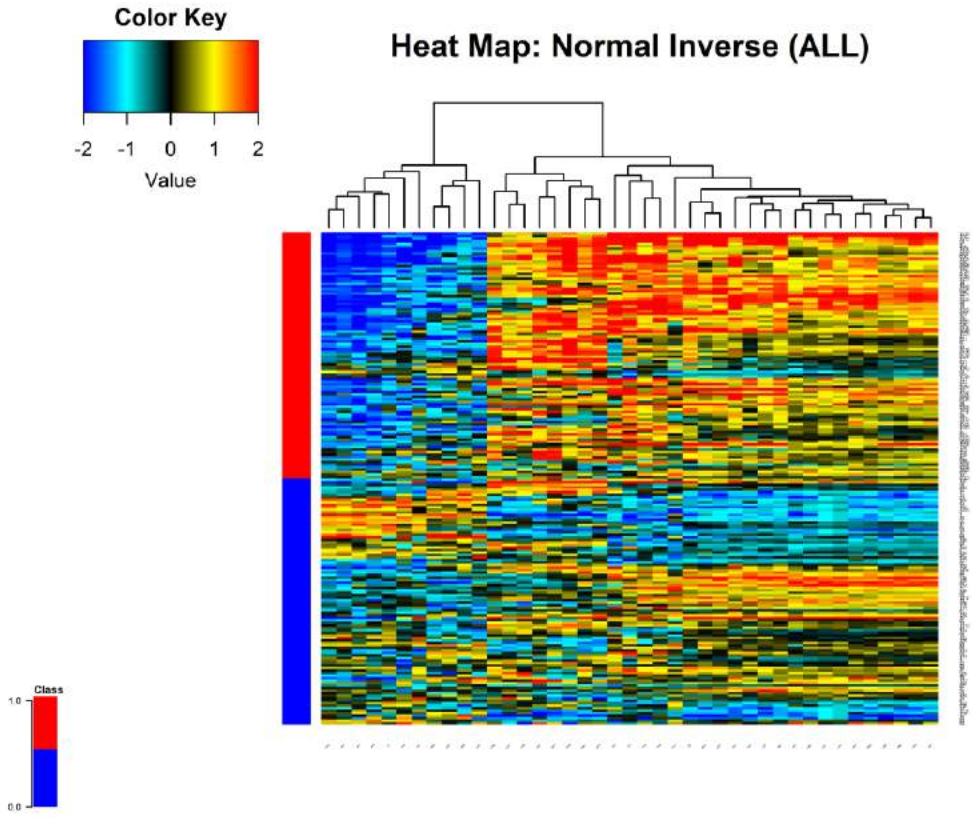


Figure 3: Heat map of relevant features of the z-normalized Glaucoma data set.

Model Selection via the BSWiMS method

Once the data is ready we can use the `BSWiMS.model(…)` function to generate a logistic model that discriminates Glaucoma and normal subjects, and we can evaluate the expected performance on an independent test set using the `randomCV(…)` function or we can perform a comprehensive k-fold cross-validation using the `FRESA.Model(…)` function[4]. All the methods used the integrated discriminant improvement (IDI) for the evaluation of the significance of each feature in improvement diagnosis performance[5]. The following code snippet shows the proper call of each option on the Glaucoma data set.

```
R> BSWiMSMODEL_ALL <- BSWiMS.model(formula = "Class ~ 1",data = ALLScaled$scaledData,NumberofRepeats = 25)
R> ALLcv <- randomCV(ALLScaled$scaledData,"Class", BSWiMS.model,trainFraction = 0.9, repetitions = 50)
R> FRESAMODEL_ALL <- FRESA.Model(formula = "Class ~ 1",data = ALLScaled$scaledData,CVfolds = 5,repeats =
10,equivalent = TRUE,usrFitFun = e1071::svm)
```

The first and third call requires the definition of the base formula: `formula = "Class ~ 1"`. This formula may contain covariates that will be included in all models. The scaled data frame is specified in all three calls (`data = ALLScaled$scaledData`). In the first call, we are specifying that the procedure is repeated 25 times before doing the last bagging step (`NumberofRepeats = 25`). By repeating the procedure 25 times we may infer the importance of each feature in separating cases from controls. The returned object will

contain all the information to diagnose and inspect the selected model/models and features. The second call evaluated BSWiMS on the Glaucoma data set by repeating 50 times a holdout cross-validation that used 80% for training and 20% for testing. The CV returned the expected behavior of the BSWiMS model on the test set. The third call performed a full validation and model-comparison of the BSWiMS method using a 5-fold cross-validation repeated 10 times. The function call also returned the CV of equivalent models, the CV of the LASSO model with lambda set to "lambda.1se", the CV of KNN modeling with k set to the square root of samples, and lastly it returned the CV performance of an SVM with a Gaussian kernel[6, 7].

The next section I will show you different aspects of the returned objects.

Diagnosing the BSWiMS Model

The `plot.bootstrapValidation_Bin(…)` function is used to verify that the top model-nugget is not overfitted. The plots output the bootstrapped validation tests, hence it may be used to get an idea of the expected model performance on a test set. The following code will return the set of bootstrapped validation plots of the top model:

```
R> pm <- plot(BSWiMSMODEL_ALL$BSWiMS.model$bootCV,main = "Bootstrap Validation (ALL)")
```

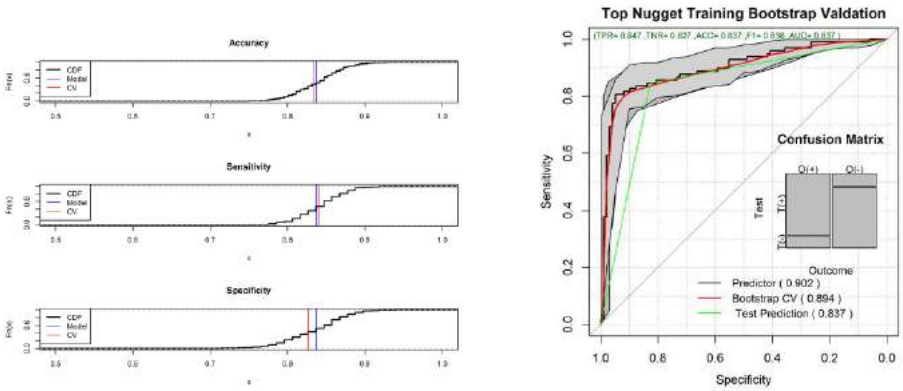


Figure 4: Left: Accuracy, Sensitivity, and Specificity bootstrapped distributions with the sample values of the model (Blue), and the bootstrapped cross-validated mean estimations (red). Right: the ROC plot of the model along with the Confusion matrix, where the red line is the bootstrapped estimation, the black line is the sample training curve, and the green line shows the operation point.

Figure 4 shows the output of the plot command. It takes the output of bootstrapped validation object (`BSWiMSMODEL_ALL$BSWiMS.model$bootCV`) and it plots the train distribution as well as the validation accuracy, sensitivity, specificity that the generated models. This plot does not show any signs of overfitting. The ROC plot of the model is the left element of figure 4. This plot also shows that the train and bootstrapped validation are equivalent.

The test performance is evaluated via the `randomCV(…)` the output can be analyzed and visualized using the following command:

```
R> ALLpsCV <- predictionStats_binary(ALLcv$medianTest,plotname = "50 Rep Random CV (All Subjects)"cex =
0.8)
```

Figure 5(a) shows the output of the above command. The plot shows the ROC curve of the predicted probabilities for both classes, and the operation point (p>0.5) as a green line. The confusion matrix of the operation point is shown on the left side of the plot, and common performance metrics are displayed on the top. The plot command also returns the evaluation of the confusion matrix and their evaluation

```
R> pander::pander(ALLpsFoldCV$CM.analysis$tab)
```

	Outcome +	Outcome -	Total
Test +	80	11	91
Test -	18	87	105
Total	98	98	196

The following instructions display the accuracy, specificity and the balanced error with their corresponding 95%CI.

```
R> pander::pander(ALLpsCV$accc)
```

est	lower	upper
0.852	0.7945	0.8986

```
R> pander::pander(ALLpsCV$sensitivity)
```

est	lower	upper
0.8163	0.7253	0.8874

```
R> pander::pander(ALLpsCV$specificity)
```

est	lower	upper
0.8878	0.808	0.9426

```
R> pander::pander(ALLpsCV$berror)
```

50%	2.5%	97.5%
0.1476	0.09913	0.1971

The bootstrapped estimation of the above statistics are the following:

```
R> pander::pander(ALLpsCV$ClassMetrics)
```

- **accci**:

50%	2.5%	97.5%
0.852	0.801	0.9031

- **senci**:

50%	2.5%	97.5%
0.8524	0.8029	0.9009

- **aucci**:

50%	2.5%	97.5%
0.8524	0.8029	0.9009

- **berci**:

50%	2.5%	97.5%
0.1476	0.09913	0.1971

- **preci**:

50%	2.5%	97.5%
0.8542	0.805	0.9031

- **F1ci**:

50%	2.5%	97.5%
0.8517	0.8008	0.9015

The full statistical analysis is stored in `ALLpsCV$CM.analysis`. The output comes from the `epiR::epi.tests(…)` I'll let the user explore the returned metrics.

The cross-validation output of the **FRESA.Model(…)** can visualize using the **plotModels.ROC**(…) function. The next line of commands produced the outputs of Figure 5(b-f).

```
R> pmeBSWiMS <- plotModels.ROC(FRESAMODEL_ALL$cvObject$Models.testPrediction,main = "Top Nugget eBSWiMS: 5-Fold CV (All Subjects)",theCVfolds = 5,predictor = "eB.SWiMS",cex = 0.8)
R> pmBSWiMS <- plotModels.ROC(FRESAMODEL_ALL$cvObject$Models.testPrediction,main = "BSWiMS: 5-Fold CV (All Subjects)",theCVfolds=5,predictor="Prediction",cex = 0.8)
R> pmLASSO <- plotModels.ROC(FRESAMODEL_ALL$cvObject$LASSO.testPredictions,main = "LASSO: 5-Fold CV (All Subjects)",theCVfolds=5,predictor="Prediction",cex = 0.8)
R> pmKNN <- plotModels.ROC(FRESAMODEL_ALL$cvObject$KNN.testPrediction,main = "KNN: 5-Fold CV (All Subjects)",theCVfolds=5,predictor="Prediction",cex = 0.8)
R> FRESAMODEL_ALL$cvObject$Models.testPrediction[,"usrFitFunction_Sel"] <- FRESAMODEL_ALL$cvObject$Models.testPrediction[,"usrFitFunction_Sel"] - 0.5
R> pmSVM <- plotModels.ROC(FRESAMODEL_ALL$cvObject$Models.testPrediction,main = "SVM: 5-Fold CV (All Subjects)",theCVfolds = 5,predictor="usrFitFunction_Sel",cex = 0.8)
```

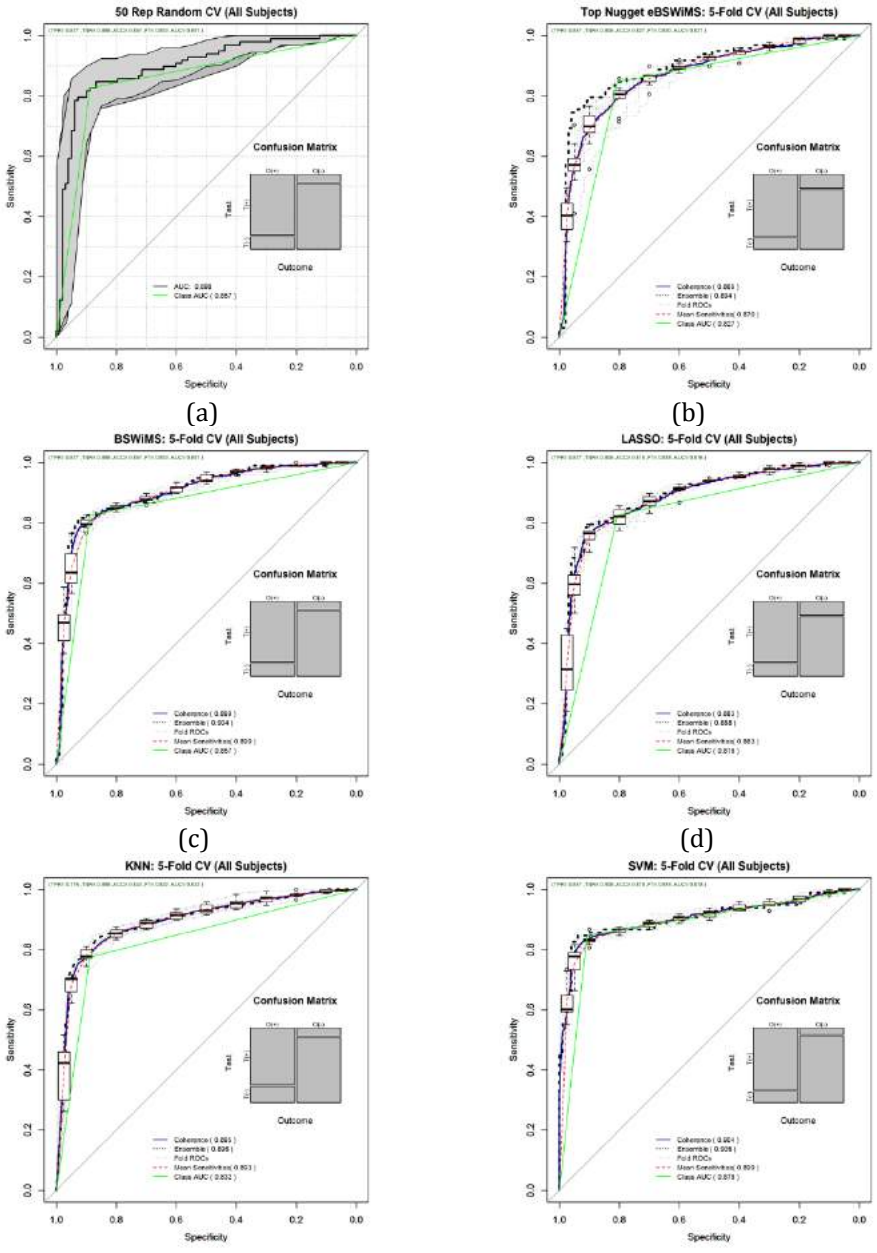


Figure 5: ROC plots of Cross-Validation. (a) Hold-out test results. (b) 5-fold CV test results for equivalent models. (c) 5-fold CV of BSWiMS model. (d) 5-fold CV of LASSO. (e) 5-fold CV of KNN. (f) 5-fold CV of SVM.

FRESA.CAD provides a simple function: `barPlotCiError(…)` that can be used compare the performance of the different classifiers. To create the outputs, the returned object of the `plotModels.ROC(…)` have to be analyzed by the `predictionStats_binary(…)` Then the output is passed to the `barPlotCiError` function. The next line of code was used to create the results shown in figure 6.

```
R> ALLpsFoldCV <- predictionStats_binary(pmBSWIM$ensemblePrediction,plotname = "10 Rep 5-Fold CV (All
Subjects)",cex = 0.8)
R> ALLLASSOFoldCV <- predictionStats_binary(pmLASSO$ensemblePrediction,plotname = "LASSO: 5-Fold CV (All
Subjects)",cex = 0.8)
R> ALLKNNFoldCV <- predictionStats_binary(pmKNN$ensemblePrediction,plotname = "KNN: 5-Fold CV (All
Subjects)",cex = 0.8)
R> ALLSVMFoldCV <- predictionStats_binary(pmSVM$ensemblePrediction,plotname = "SVM: 5-Fold CV (All
Subjects)",cex = 0.8)
R> ALLEquiFoldCV <- predictionStats_binary(pmeBSWiMS$ensemblePrediction,plotname = "eBSWiMS: 5-Fold CV
(All Subjects)",cex = 0.8)

R> AllbalancedError <- rbind(BSWiMS = ALLpsFoldCV$berror, LASSO = ALLLASSOFoldCV$berror,KNN =
ALLKNNFoldCV$berror,SVM = ALLSVMFoldCV$berror,eBSWiMS = ALLEquiFoldCV$berror)

R> bpCI <- barPlotCiError(as.matrix(AllbalancedError),metricname = "Balanced Error",
                        thesets =  c("Repeated 5-Fold CV Test Set"),
                        themethod = rownames(AllbalancedError),
                        main = "Balanced Error",
                        offsets = c(0.5,1),
                        scoreDirection = "<",ho = 0.5,
                        args.legend = list(bg = "white",x = "bottomright"),
                        col = terrain.colors(nrow(AllbalancedError)))

R> AllAUC <- rbind(BSWiMS = ALLpsFoldCV$aucs, LASSO = ALLLASSOFoldCV$aucs,KNN = ALLKNNFoldCV$aucs,SVM =
ALLSVMFoldCV$aucs,eBSWiMS = ALLEquiFoldCV$aucs)

R> bpCI <- barPlotCiError(as.matrix(AllAUC),metricname = "AUC",
                        thesets =  c("Repeated 5-Fold CV Test Set"),
                        themethod = rownames(AllAUC),
                        main = "ROC AUC",
                        offsets = c(0.5,1),
                        args.legend = list(bg = "white",x = "bottomright"),
                        col = terrain.colors(nrow(AllAUC)))
```

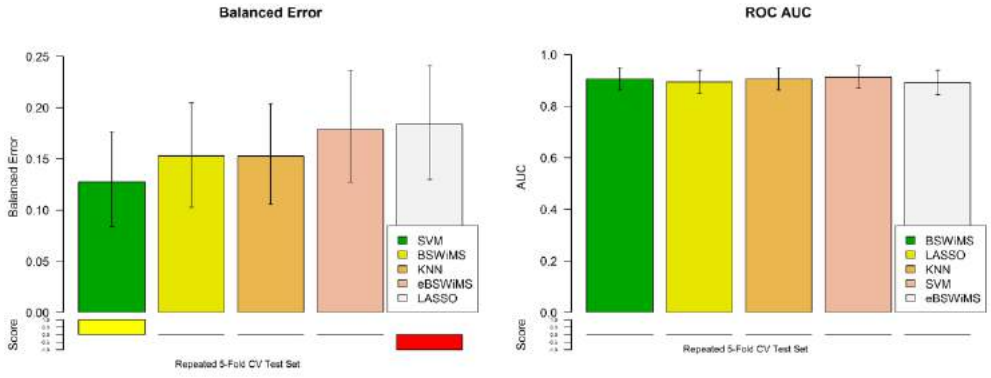


Figure 6: 5-fold test results comparisons of the Balanced Error Rate (left) and the ROC AUC (right)

`barPlotCiError(…)` get the performance data with their corresponding 95%CI to create the plots and rank them from the one with the best score performance to the method with the lowest score, where the score is just the sum of how many times a method was statistically superior to other method minus the times that the same method was statistically inferior to the other method.

Exploring the BSWiMS model

The summary of the method can be extracted by executing the `summary(…)` function. This function will return a set of objects that describe the model coefficients and performance. The following code snippet shows how to access the top 20 model coefficients then rank them according to their z-value:

```
R> sm <- summary(BSWiMSMODEL_ALL)
R> smcoeff <- sm$coefficients[order(-sm$coefficients[,"Frequency"]),]
pander::pander(smcoeff[1:20,c("Estimate","lower","OR","upper","z.IDI","Frequency")])
```

Table 1 shows the returned coefficients along with their diagnostic odds, the z value of the IDI test and the frequency of selection.

Table 1

	Estimate	lower	OR	upper	z.IDI	Frequency
vars	-0.219	0.8015	0.8033	0.8458	8.327	1
varg	-0.1125	0.8777	0.8936	0.9176	8.307	1
vari	-0.07706	0.9223	0.9258	0.9458	7.069	1
vbri	0.4204	1.32	1.523	1.849	5.782	1
vbst	-0.4389	0.5536	0.6447	0.7797	5.641	1
phci	0.133	1.107	1.142	1.202	5.098	1
mhci	0.08533	1.063	1.089	1.127	4.946	1
tmi	0.03928	1.024	1.04	1.058	4.566	1
hic	0.06972	1.035	1.072	1.149	3.907	1
phcg	0.04626	1.04	1.047	1.069	4.532	0.96
mhcn	0.09496	1.086	1.1	1.171	2.944	0.96
at	-0.2148	0.7445	0.8067	0.8981	5.246	0.92
abrs	0.2852	1.167	1.33	1.604	4.263	0.92
abri	0.1634	1.094	1.178	1.27	4.222	0.92
mdic	0.6181	1.495	1.855	3.265	5.601	0.88
hvc	-0.09726	0.8786	0.9073	0.9616	3.281	0.88
rnf	-0.2133	0.7561	0.8079	0.9207	6.308	0.84
eat	-0.1833	0.7734	0.8326	0.9101	4.871	0.84
vass	-0.09313	0.8851	0.9111	0.9592	3.543	0.76
mdt	-0.1273	0.8174	0.8804	0.9544	3.361	0.76

An alternative way to display the relative importance and their association between features is to the display the analysis of the extracted nuggets:

```
R> gplots::heatmap.2(BSWiMSMODEL_ALL$bagging$formulaNetwork[1:20,1:20],trace = "none",mar = c(5,5),main =
"B:SWiMS Formula Network", cexRow = 0.75,cexCol = 0.75)
```

The `BSWiMSMODEL_ALL$bagging$formulaNetwork` contains an analysis of how many times a feature was found by the repeated `BSWiMS` method and how many times that feature was found concordant to other nugget feature. Figure 7(a) shows the heatmap of the top 20 features of the `formulaNetwork` object. The heat map shows that (tmi, phcg) or (vari,hic) paired features were concurrent most of the time. That means that the nugget on average had only two features either (tmi, phcg) or (vari,hic). There were other associations (vbst,vbri) but not as frequent.

Furthermore, we can explore if there are more association between features using the **`reportEquivalentVariables`**(…)function. This function will explore a model nugget and it will return a set of equivalent model-nuggets were each new nugget is different to the parent nugget by a single equivalent feature. The next code snippet extracted the equivalent model-nuggets from the top BSWiMS nugget.

```
R> eq <- reportEquivalentVariables(BSWiMSMODEL_ALL$BSWiMS.model$back.model,pvalue = 0.05,
                                   data = ALLScaled$scaledData,
                                   variableList = BSWiMSMODEL_ALL$univariate,
                                   Outcome = "Class",
                                   type = "LOGIT");

R> sm <- summary(eq$equivalentModel)
pander::pander(sm$coefficients[,c("Estimate","lower","OR","upper","z.IDI","Frequency")])
```

Table 2: Summary of top equivalent model features.

	Estimate	lower	OR	upper	z.IDI	Frequency
vari	-1.233	0.2251	0.2914	0.3896	9.363	0.6429
vars	-0.04711	0.9396	0.954	0.9685	6.097	0.03571
varg	-0.03692	0.9524	0.9637	0.9753	6.083	0.03571
hic	0.3891	1.266	1.476	2.058	4.968	0.4286
phci	0.02266	1.013	1.023	1.033	4.359	0.03571
phcg	0.01617	1.008	1.016	1.024	3.972	0.03571
tmi	0.02537	1.01	1.026	1.041	3.288	0.07143
phcn	0.0106	1.004	1.011	1.017	3.186	0.03571
vbst	-0.02882	0.9542	0.9716	0.9893	3.124	0.03571
mhcg	0.03316	1.012	1.034	1.056	3.085	0.03571
mdi	0.002213	1.001	1.002	1.004	2.981	0.07143
phct	0.03768	1.012	1.038	1.065	2.909	0.03571
mhct	0.03384	1.011	1.034	1.058	2.883	0.03571
mdic	0.02655	1.007	1.027	1.047	2.719	0.03571
tmg	0.02542	1.007	1.026	1.045	2.68	0.07143
tms	0.02282	1.006	1.023	1.041	2.606	0.03571
mhcn	0.02413	1.005	1.024	1.044	2.506	0.03571
mdn	0.03226	1.006	1.033	1.06	2.437	0.03571
mdg	0.02617	1.005	1.027	1.049	2.408	0.03571

The analysis of each feature and its association with other features is in the `equivalentMatrix.` Matrix that is part of the equivalent model object. The matrix elements are seen in table 3.

```
R> pander::pander(eq$equivalentMatrix)
```

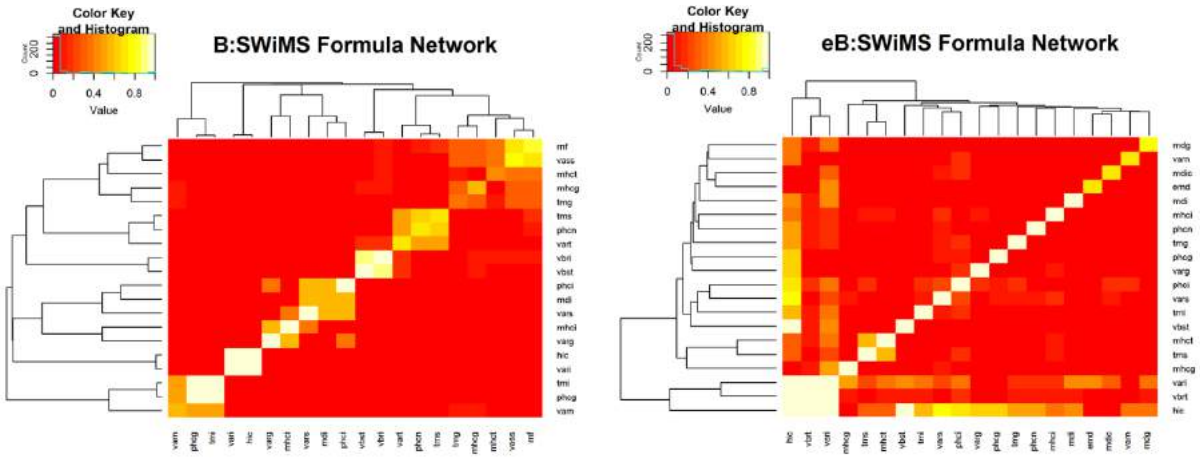
Table 3: The equivalent matrix `reportEquivalentVariables(…)$equivalentMatrix` matrix

Name	Locus	Extended Name	Unit Performance	Full Performance	Delta Performance	Improvement Fraction	p.value
vari	vari	vari:vari	0.8112	0.8418	0.06633	0.5408	1.47e-12
varg	vari	varg:vari	0.7857	0.7959	0.02041	0.5	5.898e-10
vars	vari	vars:vari	0.7755	0.8316	0.05612	0.5	5.406e-10
tmi	vari	tmi:vari	0.7551	0.7908	0.01531	0.2449	2.101e-05
tmg	vari	tmg:vari	0.7143	0.7857	0.0102	0.2653	0.0007421
phcg	vari	phcg:vari	0.7296	0.7908	0.01531	0.2551	3.559e-05
phci	vari	phci:vari	0.75	0.801	0.02551	0.3265	6.524e-06
phcn	vari	phcn:vari	0.7041	0.7806	0.005102	0.2347	0.0007219
emd	vari	emd:vari	0.6735	0.7908	0.01531	0.1224	0.02579
vbst	vari	vbst:vari	0.5969	0.7857	0.0102	0.2143	0.0008923
mdi	vari	mdi:vari	0.6122	0.7806	0.005102	0.1122	0.000542
tmi	hic	tmi:hic	0.7551	0.8316	0.02041	0.2551	0.006563
hic	hic	hic:hic	0.7755	0.8418	0.03061	0.2959	9.666e-05
tmg	hic	tmg:hic	0.7143	0.8265	0.01531	0.1939	0.01454
tms	hic	tms:hic	0.7194	0.8163	0.005102	0.2347	0.004576
mdic	hic	mdic:hic	0.7194	0.8214	0.0102	0.2041	0.003278
vbrs	hic	vbrs:hic	0.6837	0.8265	0.01531	0.2959	0.01438
mhcn	hic	mhcn:hic	0.7143	0.8316	0.02041	0.2551	0.006107
mhcg	hic	mhcg:hic	0.7092	0.8316	0.02041	0.2959	0.001019
emd	hic	emd:hic	0.6735	0.8367	0.02551	0.2551	0.004058
mhcs	hic	mhcs:hic	0.7041	0.8367	0.02551	0.2653	0.01605
vbss	hic	vbss:hic	0.6786	0.8367	0.02551	0.1939	0.009954
mdn	hic	mdn:hic	0.6071	0.8265	0.01531	0.2041	0.007396
mdi	hic	mdi:hic	0.6122	0.8214	0.0102	0.2347	0.003527
mds	hic	mds:hic	0.5969	0.8316	0.02041	0.1837	0.01041
mdg	hic	mdg:hic	0.5918	0.8214	0.0102	0.2041	0.00801
mhct	hic	mhct:hic	0.5306	0.8214	0.0102	0.1939	0.001966
phct	hic	phct:hic	0.5102	0.8163	0.005102	0.2755	0.001815

Next, the vignette will show you how to visualize the model matrix network of the equivalent models. The following code will use the CV object to visualize all the association found by the equivalent matrix of the top model-nugget:

```
#50 Models from 5-fold CV repeated 10 times
R> nev <- length(FRESAMODEL_ALL$cvObject$equiFormulas.list)/50;
R> fn <- nev*bmEqi$formulaNetwork
R> ford <- colMeans(fn)
fn <- fn[order(-ford),order(-ford)]
fn[fn > 1] <- 1

gplots::heatmap.2(fn[1:20,1:20],trace = "none",mar = c(5,5),main = "eB:SWiMS Formula Network", cexRow =
0.75,cexCol = 0.75)
```

Figure 7(b) shows the model-network of the CV models. The heat-map shows that features vari, vbrt, hic and vbst have significant contribution in separating cases and controls.



(a) (b)

Figure 7: Heat map of feature network created by analysis of extracted model-nuggets. (a) Network of BSWiMS model. (b)

Finally, we can use **`baggedModel(…)`** function to refine the 95%CI of the BSWiMS model. To achieve this goal, we will bootstrap each one of the model-nuggets 100 times in the bagging procedure. By bootstrapping the bagging process we will get better estimates of each model coefficient and importance inside each nugget.

```
#bagging 100 times the model-nuggets inside bagging.

R> bmAll <- baggedModel(BSWiMSMODEL_ALL$formula.list,ALLScaled$scaledData,type = "LOGIT",n_bootstrap =
100)
R> smALL <- summary(bmAll$bagged.model)
R> pander::pander(smALL$coefficients)
```

Table 4 shows the first 18 coefficients of the bagged model.

Table 4: 18 features of the BSWiMS model

Feature	Estimate	lower	OR	upper	z.IDI	z.NRI	Frequency
varg	-0.1159	0.8665	0.8906	0.9272	8.304	9.276	1
vars	-0.2239	0.7581	0.7994	0.8622	8.271	9.479	1
vbsg	-0.6235	0.4544	0.536	0.6891	7.394	9.34	0.32
vari	-0.08346	0.899	0.9199	0.9548	7.111	9.367	1
vbrg	0.3957	1.322	1.485	1.948	6.667	6.837	0.4
rnf	-0.2228	0.7467	0.8003	0.943	6.297	7.286	0.84
abrg	0.4611	1.363	1.586	2.212	5.982	6.31	0.72
vbrs	0.2774	1.2	1.32	1.567	5.737	6.126	0.6
vbri	0.4636	1.355	1.59	2.162	5.682	6.33	1
mdic	0.6986	1.577	2.011	3.92	5.639	6.678	0.88
vbst	-0.4944	0.5127	0.6099	0.8398	5.581	6.244	1
at	-0.2586	0.7011	0.7722	0.9259	5.247	5.87	0.92
phci	0.1422	1.092	1.153	1.231	5.158	7.178	1
mhci	0.08836	1.055	1.092	1.145	4.927	6.859	1
eat	-0.223	0.7316	0.8001	0.951	4.886	5.803	0.84
mhct	0.1055	1.064	1.111	1.201	4.743	5.324	0.44
eag	-0.1093	0.8566	0.8965	0.9705	4.707	5.412	0.2
phcg	0.05096	1.031	1.052	1.075	4.679	5.855	0.96

Univariate Feature Selection *vs.* BSWiMS

The BSWiMS method is based on selecting informative features in each model nugget. This section will explore the univariate analysis and univariate filtering and compare the feature ranks of BSWiMS, Statistical tests: Wilcoxon, t-student, Kendall correlation, Spearman correlation and the classic minimum Redundancy Maximum Relevance(mRMR) feature selection algorithm[8].

The FRESA.CAD `univariateRankVariables(…)` function, can be used to compute basic descriptive statistics of the data that can be useful to describe differences between cases and control for each feature in the data frame. The following code is an example of the use of this function to display the subjects with abnormal values.

```
# The features to be analyzed by univariate ranking

R> varlist <- colnames(ALLScaled$scaledData)
R> varlist <- varlist[varlist != "Class"]
R> varlist <- cbind(varlist,varlist)

R> univ <- univariateRankVariables(varlist,"Class~1","Class",ALLScaled$scaledData,
                           categorizationType = "Tail",
                           type = "LOGIT",
                           cateGroups = c(0.05, 0.95),
                           raw.dataFrame = GlaucomaM_mat)

R> rownames(univ) <- univ$parent

# The data table 5
R> selFeatures <- names(ALLcv$featureFrequency[ALLcv$featureFrequency > 25])
R> pander::pander(univ[selFeatures,c("controlMean","caseMean","ROCAUC","caseN_Z_Low_Tail",
"caseN_Z_Hi_Tail")])
```

The first three lines of the code create a list of features to be analyzed. The user has the freedom to select a specific set of features to be analyzed and described in the varlist matrix. In this case, we select all the data features. The parameter: `type = "LOGIT"` is indicating that generalized linear models with a logit link will be used to rank features. The `categorizationType = "Tail"` with `cateGroups = c(0.05, 0.95)` parameters are telling the function to define a new set of model terms for the logistic function that will divide the data into two new categories: The lower tail subjects (p<0.5) and upper tail (p>0.95). The `raw.dataFrame = GlaucomaM_mat` parameters are indicating that the raw data is the `GlaucomaM_mat` matrix. Once the descriptive statistics are computed for all features, a set of columns are selected to be displayed in a table, and only features that had a relatively high selection frequency in the holdout CV test. Table 5 shows the result of the last command that creates the formatted table.

The visualization of the subjects with abnormal features values is done using the **`heatMaps`**`(…)` function as shown in the next line of code:

```
# The heat map showing the location of abnormal values  figure 8
R>  hm <- heatMaps(variableList = univ,Outcome = "Class",
                   data = ALLScaled$scaledData,
                   title = "Heat Map: univ",Scale = c(-2,2),
                   hCluster = "col",cexRow = 0.20,cexCol = 0.3,srtCol = 45)
```

Figure 8 displays the heat map, were we can see how the glaucoma subjects have features with abnormally low values (p<0.05, the blue colors) and features with abnormal high values (p>0.95, the red colors). Control subjects as expected did no show many blue nor red subjects.

Table 5: Descriptive statistics of the more frequently selected features of the Glaucoma data sets. Mean values, ROC area under the curve (ROCAUC), and a number of subjects with abnormal values (caseN_Z_Low_Tail and caseN_Z_Hi_Tail). These two columns represent the number of subjects with abnormal signal values ($p<0.05$), and the number of subjects with abnormally high signal values ($p>0.95$)

	controlMean	caseMean	ROCAUC	caseN_Z_Low_Tail	caseN_Z_Hi_Tail
mhci	0.02176	0.1076	0.594	0	23
varg	0.4139	0.1785	0.7119	45	0
vari	0.115	0.0448	0.7574	54	0
vars	0.1068	0.04506	0.6879	41	0
hic	0.2114	0.3986	0.6369	0	31
phci	-0.09373	0.00898	0.674	0	38
tmi	-0.1097	0.03664	0.6709	2	37
rnf	0.2252	0.1396	0.6106	26	0
vbri	0.06443	0.1466	0.616	0	27
abri	0.2331	0.417	0.5983	0	23
phcg	-0.1216	-0.03546	0.6866	0	40
vbst	0.1122	0.1558	0.551	0	14
mhcg	0.06633	0.122	0.6218	0	28
eat	0.389	0.4238	0.5056	0	5
varn	0.1774	0.08224	0.6893	42	0
abrs	0.2502	0.4088	0.5201	0	8
phcn	-0.07168	0.006918	0.6968	0	43
mdic	0.1722	0.2904	0.6091	0	25
tms	-0.1192	0.03959	0.6177	0	27
vbrt	0.07163	0.1228	0.5762	0	19
abrg	0.976	1.608	0.5455	0	13
mhcn	0.04119	0.1064	0.59	0	22
at	0.454	0.464	0.5052	0	3
hvc	0.4077	0.3132	0.5685	17	1
tmg	-0.1524	-0.03356	0.6891	0	41

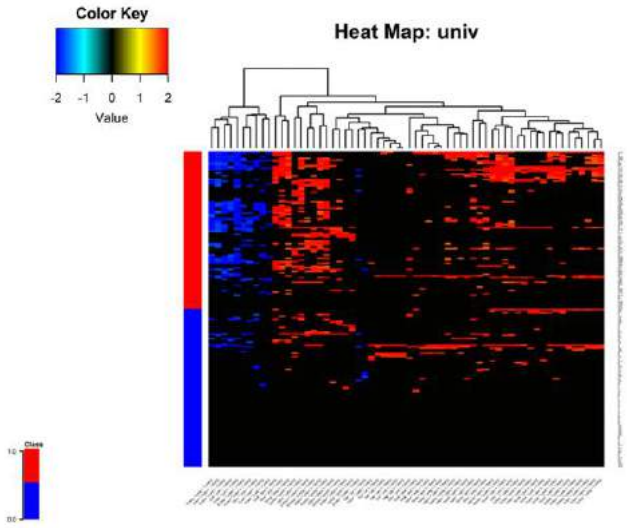
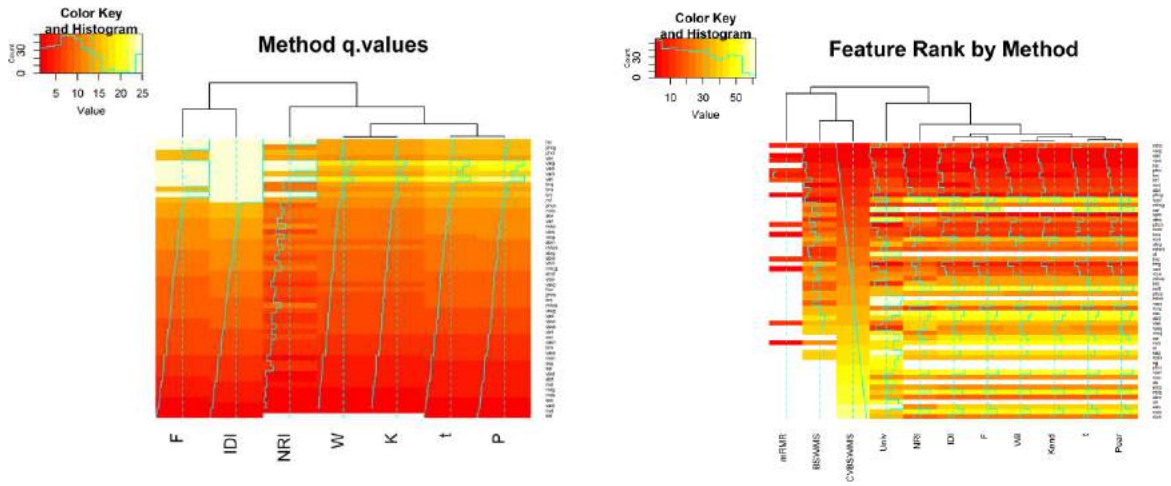


Figure 8: Heat map showing features with the abnormal signal. The red colors indicate a subject with a feature signal that is abnormally high ($p>0.95$). The blue colors indicate subjects whose feature signal is abnormally low ($p<0.05$)

The comparison of the BSWiMS features to other common feature selection starts by finding the adjusted p-values of the association of the features to the outcome. This is done by executing the following lines of code:

```
R> q_values <- univariate_Logit(data = ALLScaled$scaledData, Outcome = "Class", pvalue = 0.05)

R> thenames <- names(ALLcv$featureFrequency)
R> mv <- match(thenames,names(ALLcv$featureFrequency))

R> qValueMatrix <- q_values
R> idiqValueMatrix <- q_values
R> mv <- cbind(mv,match(thenames,names(q_values)))

R> q_values <- univariate_Logit(data = ALLScaled$scaledData, Outcome = "Class", uniTest = "zNRI",pvalue =
0.05)
R> qValueMatrix <- cbind(idiqValueMatrix,q_values[names(idiqValueMatrix)])
R> mv <- cbind(mv,match(thenames,names(q_values)))

R> q_values <- univariate_residual(data = ALLScaled$scaledData, Outcome = "Class",pvalue = 0.05,type =
"LOGIT")
R> qValueMatrix <- cbind(qValueMatrix,q_values[names(idiqValueMatrix)])
R> mv <- cbind(mv,match(thenames,names(q_values)))

R> q_values <- univariate_tstudent(data = ALLScaled$scaledData, Outcome = "Class",pvalue = 0.05)
R> qValueMatrix <- cbind(qValueMatrix,q_values[names(idiqValueMatrix)])
R> mv <- cbind(mv,match(thenames,names(q_values)))

R> q_values <- univariate_Wilcoxon(data = ALLScaled$scaledData, Outcome = "Class", pvalue = 0.05)
R> qValueMatrix <- cbind(qValueMatrix,q_values[names(idiqValueMatrix)])
R> mv <- cbind(mv,match(thenames,names(q_values)))

R> q_values <- univariate_correlation(data = ALLScaled$scaledData, Outcome = "Class", pvalue = 0.05)
R> qValueMatrix <- cbind(qValueMatrix,q_values[names(idiqValueMatrix)])
R> mv <- cbind(mv,match(thenames,names(q_values)))

R> q_values <- univariate_correlation(data = ALLScaled$scaledData, Outcome = "Class", pvalue = 0.05,
method = "pearson")

#The qValueMatrix has the qValues of all filter methods.
R> qValueMatrix <- cbind(qValueMatrix,q_values[names(idiqValueMatrix)])

R> colnames(qValueMatrix) <- c("IDI","NRI","F","t","W","K","P")
#Do the log transform to display the heatmap
R> qValueMatrix <- -log10(qValueMatrix)
R> qValueMatrix[is.infinite(qValueMatrix)] <- 25
#the Heatmap of the q-values
R> gplots::heatmap.2(qValueMatrix,Rowv = FALSE,dendrogram = "col",
                  main = "Method q.values",cexRow = 0.4,mar = c(5,5))
```

The code extracts features that improve the net reclassification (NRI), the integrated discriminant (IDI) in logistic models. It also extracts features that separate cases and controls based on Wilcoxon-test and t-tests. Two correlation filters were used: The Pearson correlation and the Kendall correlation. All these approaches compute the p-values for every single feature, then those values are FDR adjusted by the BH procedure finally they return the feature named vector of the q-values that were lower than the specified p-value. The last lines of the code name the matrix that stored all the discovered features and displays the resulting features using the heatmap.2 function. Figure 9(a) shows the corresponding plot.



(a) (b)

Figure 9: (a) the heat map showing the log10 of the p-values of the association between the features with the binary outcome. (b) The heat map plot shows the rank of the statistically associated features to the outcome.

The last component of the comparison is to get the rank of the features and compare the ranks to the ranks observed by BSWiMS selection frequency, the CV BSWiMS, and the rank obtained by the classic mRMR method. The next lines of code achieve that:

```
# Add the mRMR and CV Sel Frequency
# mv already has MV,IDI,NRI,F,t,Wil,Kend and Pear
R> mv <- cbind(mv,match(thenames,names(q_values)))
R> mrmr <- mRMR.classic_FRESA(data = ALLScaled$scaledData, Outcome = "Class")
R> mv <- cbind(mv,match(thenames,names(mrmr)))

#mv contains the feature rank by the method, adding the univariate
R> mv <- cbind(mv,match(thenames,as.character(univ$parent)))

#Lets add the repeated BSWiMS
R> sm <- summary(BSWiMSMODEL_ALL)
R> smcoeff <- sm$coefficients
R> BSWiMFf <- smcoeff$Frequency
R> names(BSWiMFf) <- rownames(smcoeff)
R> BSWiMFf <- BSWiMFf[order(-BSWiMFf)]

R> mv <- cbind(mv,match(thenames,names(BSWiMFf)))

#Set the names
R> colnames(mv) <- c("CVBSWiMS","IDI","NRI","F","t","Wil","Kend","Pear","mRMR","Univ","BSWiMS")
R> rownames(mv) <- thenames
#Heat map
R> gplots::heatmap.2(mv,Rowv = FALSE,dendrogram = "col",mar = c(5,5),
                  main = "Feature Rank by Method",cexRow = 0.4,cexCol = 0.75)
```

Figure 9(b) shows the heat map generated by the last line of code. The features are ordered by the repeated holdout CV BSWiMS method and clustered in how similar are the feature ranks. As expected the ranks of the CV BSWiMS and the repeated BSWiMS are very similar and with close proximity to the mRMR. The univariate methods share similar rankings but different from BSWiMS. It is clear that there are several features that were poorly ranked by univariate methods, but that is top-ranked by the BSWiMS method. This result can be explained by the fact that BSWiMS is a wrapper multivariate method that selects features based on the importance to add discriminant power; hence, correlated features can be detected by multivariate methods, something that univariate analyzes can´t.

References

[1] Y. Benjamini and Y. Hochberg, "CONTROLLING THE FALSE DISCOVERY RATE - A PRACTICAL AND POWERFUL APPROACH TO MULTIPLE TESTING," *Journal of the Royal Statistical Society Series B-Methodological,* vol. 57, no. 1, pp. 289-300, 1995.

[2] L. Breiman, "Random forests," *Machine Learning,* vol. 45, no. 1, pp. 5-32, Oct 2001.

[3] T. Hothorn and B. Lausen, "Double-bagging: combining classifiers by bootstrap aggregation," *Pattern Recognition,* vol. 36, no. 6, pp. 1303-1309, Jun 2003, Art. no. Pii s0031-3203(02)00169-3.

[4] J. H. Kim, "Estimating classification error rate: Repeated cross-validation, repeated hold-out and bootstrap," *Computational Statistics & Data Analysis,* vol. 53, no. 11, pp. 3735-3745, Sep 2009.

[5] M. J. Pencina, R. B. D'Agostino, Sr., R. B. D'Agostino, Jr., and R. S. Vasan, "Evaluating the added predictive ability of a new marker: From area under the ROC curve to reclassification and beyond," *Statistics in Medicine,* vol. 27, no. 2, pp. 157-172, Jan 30 2008.

[6] J. Neumann, C. Schnorr, and G. Steidl, "Combined SVM-based feature selection and classification," *Machine Learning,* vol. 61, no. 1-3, pp. 129-150, Nov 2005.

[7] R. Tibshirani, "Regression shrinkage and selection via the Lasso," *Journal of the Royal Statistical Society Series B-Methodological,* vol. 58, no. 1, pp. 267-288, 1996 1996.

[8] C. Ding and H. Peng, "Minimum redundancy feature selection from microarray gene expression data," *Journal of bioinformatics and computational biology,* vol. 3, no. 2, pp. 185-205, 2005-Apr 2005.

Appendix A: Holdout Cross-Validation and the Ensemble Prediction

The repeated holdout CV and the repeated k-fold cross validation provide an assessment of the test performance by evaluating the mean prediction of the repeated test results. In this section, we will validate that the repeated assessment approach gives accurate estimations of the models in an independent test set. Furthermore, we will introduce the ensemble prediction, as an alternative to the linear prediction of BSWiMS. The validation will start by creating an independent test set that will contain 50% of the original set. The other 50% of the data will be used to train and get estimations of the test performance.

```
# Code snipped that creates a train/test holdout sets

R> trainControlSample <- sample(nrow(ALLcontrolSubjects),nrow(ALLcontrolSubjects)/2)
R> trainCaseSample <- sample(nrow(ALLCaseSubjects),nrow(ALLCaseSubjects)/2)

R> trainSet <- rbind(ALLcontrolSubjects[trainControlSample,],ALLCaseSubjects[trainCaseSample,])
R> testSet <- rbind(ALLcontrolSubjects[-trainControlSample,],ALLCaseSubjects[-trainCaseSample,])

R> controlSubjects <- subset(trainSet,Class == 0)

R> trainScaled <- FRESAScale(trainSet,refFrame = controlSubjects,method = "RankInv")
R> testScaled <- FRESAScale(testSet,refFrame = controlSubjects,method = "RankInv")
```

The above code shows that we got a random sample of cases and controls. The random sample was merged into a train and a test set with an equal number of cases and controls. From the train data, I will compute two BSWiMS models (One simple model, and the other obtained by 25 FS-BE repetitions), and holdout CV that used 90% of the train data for training and 10% for the test, and it was repeated 50 times:

```
R> BSWiMSMODEL_1 <- BSWiMS.model(formula = "Class ~ 1",data = trainScaled$scaledData)
R> BSWiMSMODEL <- BSWiMS.model(formula = "Class ~ 1",data = trainScaled$scaledData,NumberofRepeats = 25)
R> cv <- randomCV(trainScaled$scaledData,"Class",BSWiMS.model,trainFraction = 0.9,repetitions = 50);
```

The prediction of the two BSWiMS models on the test data and the cross-validation results will be compared by the **barPlotCiError(…)**. To do that I will run the following code:

```
#Get the CV stats of the model that used all the data
R> ALLpsCV <- predictionStats_binary(ALLcv$medianTest,plotname = "Cross-Validation (All Subjects)",cex =
0.8)

#Get the CV stats of the model that used the train data

R> psCV <- predictionStats_binary(cv$medianTest,plotname = "Cross-Validation Train Set",cex = 0.8)

#Predict the test set using the simple BSWiMS and the test performance

R> BaggedPre1 <- predict(BSWiMSMODEL_1,testScaled$scaledData)
R> psBaggedPre1 <- predictionStats_binary(cbind(testSet$Class,BaggedPre1),plotname = "BSWiMS(n=1) Test
Set",cex = 0.8)

#Predict the test set using the 25 repeat BSWiMS and the test performance

R> BaggedPre25 <- predict(BSWiMSMODEL,testScaled$scaledData)
R> psBaggedPre25 <- predictionStats_binary(cbind(testSet$Class,BaggedPre25),plotname = "BSWiMS(n=25) Test
Set",cex = 0.8)

#Ensemble prediction using the 25 repeat BSWiMS and the test performance

R> ensemblepred10 <-
ensemblePredict(BSWiMSMODEL$formula.list,trainScaled$scaledData,testScaled$scaledData)
R> pm <- plotModels.ROC(ensemblepred10$predictions,main = "Test Set",cex = 0.8)

R> psensemblepred10 <- predictionStats_binary(cbind(testSet$Class,ensemblepred10$ensemblePredict),plotname
= "Ensemble Test Set",cex = 0.8)

#Create the matrix for the balanced error rate with thei 95% CI

R> balancedError <- rbind(ALL_CV = ALLpsCV$berror, Train_CV = psCV$berror,Bagg_1 =
psBaggedPre1$berror,Bagg_25 = psBaggedPre25$berror,Median_25 = psensemblepred10$berror)
#Plot
R> bpCI <- barPlotCiError(as.matrix(balancedError),metricname = "Balanced Error",
                         thesets =  c("Test Set"),
                         themethod = rownames(balancedError),
                         main = "Balanced Error",
                         offsets = c(0.5,1),
                         scoreDirection = "<",ho = 0.5,
                         args.legend = list(bg = "white",x = "bottomright"),
                         col = terrain.colors(nrow(balancedError)))

#Create the matrix of the AUC with 95%CI

R> AUC <- rbind(ALL_CV = ALLpsCV$auc, Train_CV = psCV$aucs,Bagg_1 = psBaggedPre1$aucs,Bagg_25 =
psBaggedPre25$aucs,Median_25 = psensemblepred10$aucs)
#Plot
R> AUCCI <- barPlotCiError(as.matrix(AUC),metricname = "ROC AUC",
                         thesets =  c("Test Set"),
                         themethod = rownames(balancedError),
                         main = "ROC AUC",
                         offsets = c(0.5,1),
                         args.legend = list(bg = "white",x = "bottomright"),
                         col = terrain.colors(nrow(balancedError)))
```

The code will produce the plots showcased in figures 10 and 11. Figure 10 plots were generated by **predictionStats_binary(…)** function. Specifically the creation of figure 10(e) and 10(f) required ensemble predictions form model-nuggets. The call: **ensemblePredict(formula.list,trainData,predictData)** (Highlighted in yellow in the above code) will use the elements of the formula list to train the model using the trainData sets and each model will produce a test prediction of the predictData. All the test predictions can be seen in figure 10(e), while figure 10(f) shows the final test result.

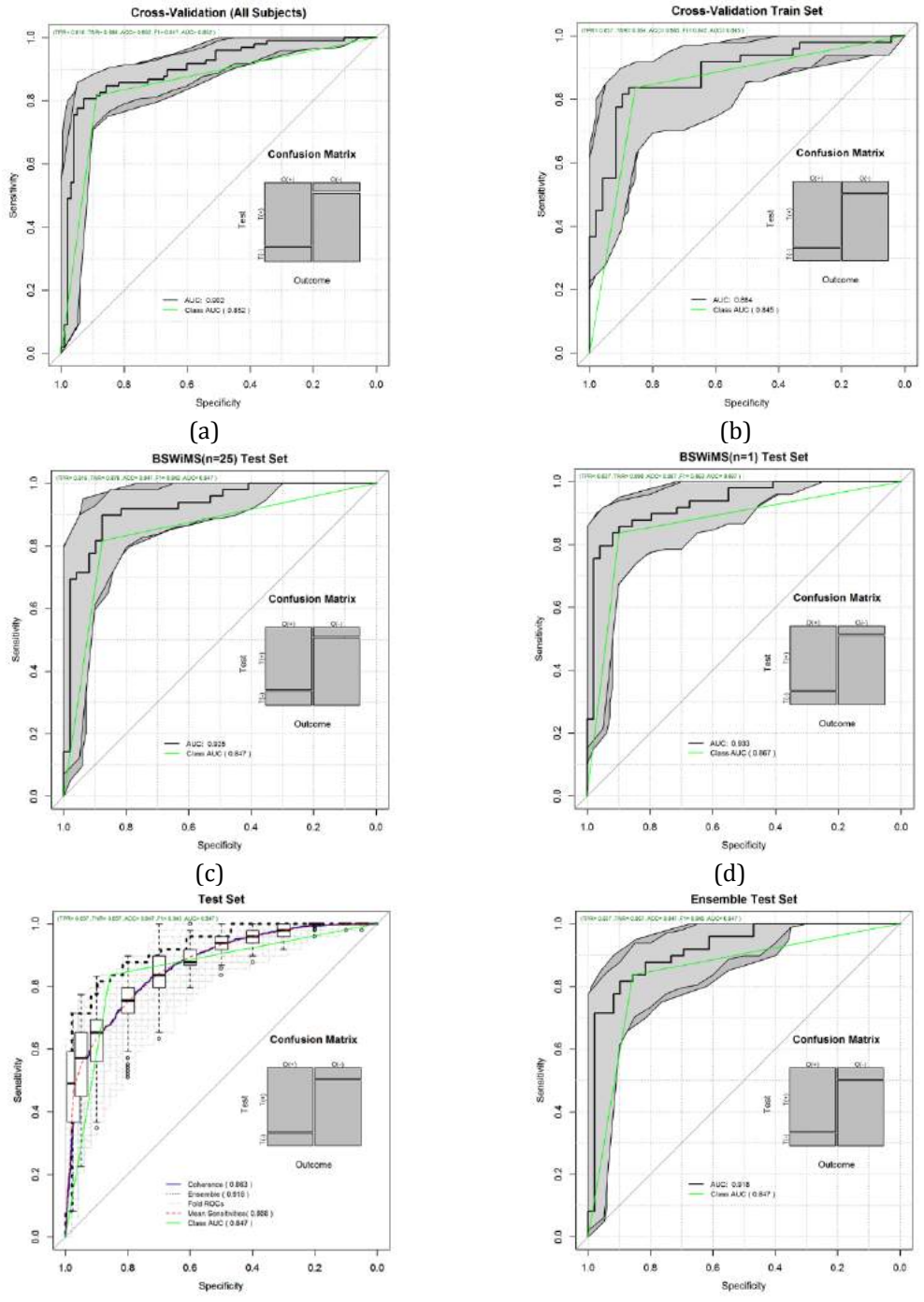


Figure 10: ROC plots of CV and 50% Holdout CV.(a) CV test result by using all the data. (b) CV results of the train data set. (c) Test result of the model repeated 25 times. (d) Test result of the model repeated only once.(e) Each one of the model nuggets predictions on the test set. (f) Test ROC of ensemble of the nuggets.

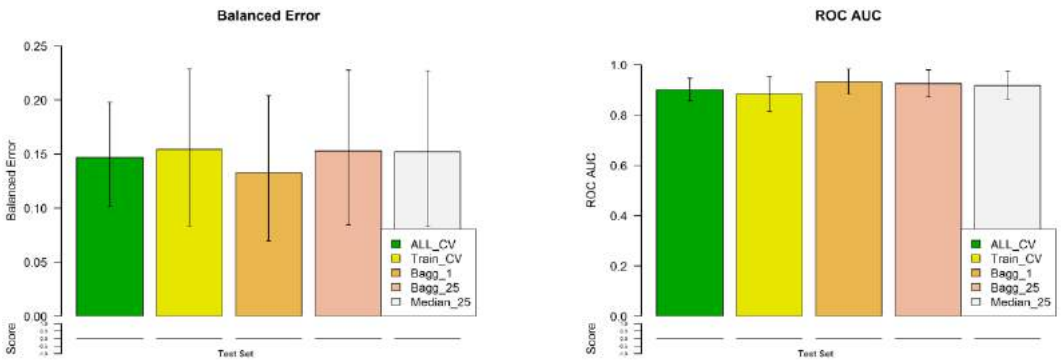


Figure 11: Error comparison and ROC AUC comparison. The left plots indicate that there was no statistical difference between test results. CV test results and 50% Holdout test results are equivalent. Right, the Comparison of the AUC yields similar conclusions. The Holdout test results were similar to the predicted CV performance.

Conclusion

The results obtained in this experiment indicate that the repeated CV performance is accurate. The CV error rate estimated using the train data was statistically equivalent to the observed performance of the test results obtained by predicting the holdout test data. None of the methods used to predict the test data was superior to any other.