CS-GY 6923 Machine Learning

Professor: Dr. Raman Kannan

Homework 3:
Ensemble techniques

Hansheng Li

Contents

1. **Start up** 3
2. **Cross Validation** 3
3. **Leave One out Cross Validation** 7
4. **Boosting** 12
5. **Original classification result** 16
6. **Comparison** 17

1. Start up

We will continue to use the code from HW 02, so no more extra import needed. In this assignment, I have chosen three Ensemble Techniques which are Cross Validation, Leave One out Cross Validation, and Bagging. And we going to compare Variance, bias, and accuracy to see how much improvement have been made by Ensemble Techniques.

2. Cross Validation

Multinational Logistic Regression:

```
> ################################
> ## cross - validation methods##
> ################################
> 
> # Multinational Logistic Regression #
> 
> # training the model by assigning Class column
> # as target variable and rest other column
> # as independent variable
> log.cv <- train(Class ~., data = data,
+                 method = "multinom",
+                 trControl = trainControl(
+                   method = "CV",
+                   number = 10,
+                   verboseIter = T
+                 ))
+ Fold01: decay=0e+00
# weights:  126 (102 variable)
initial  value 23839.345236
iter  10 value 18372.055625
iter  20 value 15025.622376
iter  30 value 10919.859504
.....
iter  80 value 2793.901318
iter  90 value 2779.793130
iter 100 value 2772.876570
final  value 2772.876570
stopped after 100 iterations
> print(log.cv)
Penalized Multinomial Regression

13611 samples
```

```
  16 predictor
   7 classes: 'BARBUNYA', 'BOMBAY', 'CALI', 'DERMASON', 'HOROZ', 'SEKER', 'SIRA'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 12251, 12251, 12249, 12249, 12250, 12248, ...
Resampling results across tuning parameters:

  decay  Accuracy   Kappa
  0e+00  0.9260157  0.9105411
  1e-04  0.9255748  0.9100060
  1e-01  0.9212403  0.9047508

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was decay = 0.
> log.cv_pred <- factor(predict(log.cv , data.valid), ordered = T)
> log.cv_cM <- table(log.cv_pred, data.valid$Class)
> log.cv_summary <-
    data.frame( Bias = mean(predict(log.cv, data.train) != data.train$Class),
+                               Variance = mean(log.cv_pred != data.valid$Class),
+                               Accuracy = sum(diag(log.cv_cM)) / sum(log.cv_cM))
> log.cv_summary
        Bias   Variance  Accuracy
1 0.07179595 0.07639569 0.9236043
```

SVM:

```
> # Support Vector Machine Classification
> svm.cv <- train(Class ~., data = data,
+                 method = "svmLinear",
+                 trControl = trainControl(
+                   method = "CV",
+                   number = 10,
+                   verboseIter = T
+                 ))
+ Fold01: C=1
- Fold01: C=1
……
- Fold10: C=1
Aggregating results
Fitting final model on full training set
> print(svm.cv)
```

```
Support Vector Machines with Linear Kernel

13611 samples
   16 predictor
    7 classes: 'BARBUNYA', 'BOMBAY', 'CALI', 'DERMASON', 'HOROZ', 'SEKER', 'SIRA'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 12250, 12251, 12250, 12249, 12250, 12250, ...
Resampling results:

  Accuracy   Kappa
  0.9262365  0.9107779

Tuning parameter 'C' was held constant at a value of 1
> svm.cv_pred <- factor(predict(svm.cv , data.valid), ordered = T)
> svm.cv_cM <- table(svm.cv_pred, data.valid$Class)
> svm.cv_summary <-
           data.frame( Bias = mean(predict(svm.cv, data.train) != data.train$Class),
+                                Variance = mean(svm.cv_pred!= data.valid$Class),
+                                Accuracy = sum(diag(svm.cv_cM)) / sum(svm.cv_cM))
> svm.cv_summary
        Bias   Variance  Accuracy
1 0.07148105 0.07492654 0.9250735
```

Decision tree:

```
> # Decision Tree
> tree.cv <- train(Class ~., data = data,
+                  method = "rpart",
+                  trControl = trainControl(
+                    method = "CV",
+                    number = 10,
+                    verboseIter = T
+                  ))
+ Fold01: cp=0.1647
- Fold01: cp=0.1647
- Fold10: cp=0.1647
Aggregating results
Selecting tuning parameters
Fitting cp = 0.165 on full training set
> print(tree.cv)
```

```
CART

13611 samples
   16 predictor
    7 classes: 'BARBUNYA', 'BOMBAY', 'CALI', 'DERMASON', 'HOROZ', 'SEKER', 'SIRA'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 12248, 12250, 12250, 12249, 12250, 12247, ...
Resampling results across tuning parameters:

  cp          Accuracy   Kappa
  0.1647293  0.5931715  0.4864191
  0.1705912  0.4694109  0.3208308
  0.1992052  0.3471001  0.1409502

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was cp = 0.1647293.
> data.train$Class <- factor(data.train$Class, ordered = FALSE)
> data.valid$Class <- factor(data.valid$Class, ordered = FALSE)
> tree.cv_pred <- predict(tree.cv , data.valid, type = "raw")
> tree.cv_cM <- table(tree.cv_pred, data.valid$Class)
> tree.cv_summary <-
            data.frame( Bias = mean(predict(tree.cv, data.train, type = "raw") != data.train$Class),
+                                Variance = mean(tree.cv_pred != data.valid$Class),
+                                Accuracy = sum(diag(tree.cv_cM)) / sum(tree.cv_cM))
> tree.cv_summary
       Bias  Variance  Accuracy
1 0.4682481 0.4608227 0.5391773
```

KNN:

```
> # K-Nearest Neighbor
> knn.cv <- train(Class ~., data = data,
+                 method = "knn",
+                 trControl = trainControl(
+                   method = "CV",
+                   number = 10,
+                   verboseIter = T
+                 ))
+ Fold01: k=5
- Fold01: k=5
```

```
+ Fold01: k=7
- Fold01: k=7
.....
+ Fold10: k=9
- Fold10: k=9
Aggregating results
Selecting tuning parameters
Fitting k = 5 on full training set
> print(knn.cv)
k-Nearest Neighbors

13611 samples
   16 predictor
    7 classes: 'BARBUNYA', 'BOMBAY', 'CALI', 'DERMASON', 'HOROZ', 'SEKE
          R', 'SIRA'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 12248, 12249, 12250, 12250, 12250, 12251, ...
Resampling results across tuning parameters:

  k  Accuracy   Kappa
  5  0.7321261  0.6746749
  7  0.7253680  0.6662656
  9  0.7228692  0.6630517

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 5.
> knn.cv_pred <- predict(knn.cv , data.valid)
> knn.cv_cM <- table(knn.cv_pred, data.valid$Class)
> knn.cv_summary <-
            data.frame( Bias = mean(predict(knn.cv, data.train) != data.t
          rain$Class),
+                                 Variance = mean(knn.cv_pred!= data.vali
          d$Class),
+                                 Accuracy = sum(diag(knn.cv_cM)) / sum(k
          nn.cv_cM))
> knn.cv_summary
       Bias  Variance  Accuracy
1 0.1783353 0.1767875 0.8232125
```

3. Leave One out Cross Validation

Multinational Logistic Regression:

```
>  log.loocv <- train(Class ~., data = data,
```

```
+                     method = "multinom",
+                     trControl = trainControl(
+                       method = "LGOCV",
+                       number = 10,
+                       p = 0.1,
+                       verboseIter = T
+                     ))
+ Resample01: decay=0e+00
# weights:  126 (102 variable)
initial  value 2654.221443
iter  10 value 2031.935250
.....
final  value 2780.792544
stopped after 100 iterations
> print(log.loocv)
Penalized Multinomial Regression

13611 samples
   16 predictor
    7 classes: 'BARBUNYA', 'BOMBAY', 'CALI', 'DERMASON', 'HOROZ', 'SEKE
          R', 'SIRA'

No pre-processing
Resampling: Repeated Train/Test Splits Estimated (10 reps, 10%)
Summary of sample sizes: 1364, 1364, 1364, 1364, 1364, 1364, ...
Resampling results across tuning parameters:

  decay  Accuracy   Kappa
  0e+00  0.9182739  0.9011821
  1e-04  0.9184617  0.9014083
  1e-01  0.9153997  0.8976753

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was decay = 1e-04.
> log.loocv_pred <- predict(log.loocv , data.valid)
> log.loocv_cM <- table(log.loocv_pred, data.valid$Class)
> log.loocv_summary <-
           data.frame( Bias = mean(predict(log.loocv, data.train) != dat
          a.train$Class),
+                                   Variance = mean(log.loocv_pred != da
          ta.valid$Class),
+                                   Accuracy = sum(diag(log.loocv_cM)) /
           sum(log.loocv_cM))
> log.loocv_summary
        Bias   Variance  Accuracy
1 0.07179595 0.07664055 0.9233595
```

SVM:

```
> # Support Vector Machine Classification
> svm.loocv <- train(Class ~., data = data,
+                    method = "svmLinear",
+                    trControl = trainControl(
+                      method = "LGOCV",
+                      number = 10,
+                      p = 0.1,
+                      verboseIter = T
+                    ))
+ Resample01: C=1
……
- Resample10: C=1
Aggregating results
Fitting final model on full training set
> print(svm.loocv)
Support Vector Machines with Linear Kernel

13611 samples
   16 predictor
    7 classes: 'BARBUNYA', 'BOMBAY', 'CALI', 'DERMASON', 'HOROZ', 'SEKE
        R', 'SIRA'

No pre-processing
Resampling: Repeated Train/Test Splits Estimated (10 reps, 10%)
Summary of sample sizes: 1364, 1364, 1364, 1364, 1364, 1364, ...
Resampling results:

  Accuracy   Kappa
  0.9200131  0.9032572

Tuning parameter 'C' was held constant at a value of 1
> svm.loocv_pred <- predict(svm.loocv , data.valid)
> svm.loocv_cM <- table(svm.loocv_pred, data.valid$Class)
> svm.loocv_summary <-
           data.frame( Bias = mean(predict(svm.loocv, data.train) != dat
        a.train$Class),
+                                  Variance = mean(svm.loocv_pred != da
        ta.valid$Class),
+                                  Accuracy = sum(diag(svm.loocv_cM)) /
           sum(svm.loocv_cM))
> svm.loocv_summary
        Bias   Variance  Accuracy
1 0.07148105 0.07492654 0.9250735
```

Decision tree:

```
> # Decision Tree
> tree.loocv  <- train(Class ~., data = data,
+                      method = "rpart",
+                      trControl = trainControl(
+                        method = "LGOCV",
+                        number = 10,
+                        p = 0.1,
+                        verboseIter = T
+                      ))
+ Resample01: cp=0.1647
……
- Resample10: cp=0.1647
Aggregating results
Selecting tuning parameters
Fitting cp = 0.165 on full training set
> print(tree.loocv)
CART

13611 samples
   16 predictor
    7 classes: 'BARBUNYA', 'BOMBAY', 'CALI', 'DERMASON', 'HOROZ', 'SEKE
          R', 'SIRA'

No pre-processing
Resampling: Repeated Train/Test Splits Estimated (10 reps, 10%)
Summary of sample sizes: 1364, 1364, 1364, 1364, 1364, 1364, ...
Resampling results across tuning parameters:

  cp         Accuracy   Kappa
  0.1647293  0.6461746  0.5591477
  0.1705912  0.5035111  0.3689403
  0.1992052  0.3773577  0.1900813

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was cp = 0.1647293.
> tree.loocv_pred <- predict(tree.loocv , data.valid, type = "raw")
> tree.loocv_cM <- table(tree.loocv_pred, data.valid$Class)
> tree.loocv_summary <-
           data.frame( Bias = mean(predict(tree.loocv, data.train, type
          = "raw") != data.train$Class),
+                                  Variance = mean(tree.loocv_pred !=
          factor(data.valid$Class, ordered = FALSE )),
+                                  Accuracy = sum(diag(tree.loocv_cM))
           / sum(tree.loocv_cM))
> tree.loocv_summary
```

```
       Bias  Variance  Accuracy
1 0.4682481 0.4608227 0.5391773
```

KNN:

```
> # K-Nearest Neighbor
> knn.loocv <- train(Class ~., data = data,
+                    method = "knn",
+                    trControl = trainControl(
+                      method = "LGOCV",
+                      number = 10,
+                      p = 0.1,
+                      verboseIter = T
+                    ))
+ Resample01: k=5
- Resample01: k=5
……
+ Resample10: k=9
- Resample10: k=9
Aggregating results
Selecting tuning parameters
Fitting k = 9 on full training set
> print(knn.loocv)
k-Nearest Neighbors

13611 samples
   16 predictor
    7 classes: 'BARBUNYA', 'BOMBAY', 'CALI', 'DERMASON', 'HOROZ', 'SEKE
        R', 'SIRA'

No pre-processing
Resampling: Repeated Train/Test Splits Estimated (10 reps, 10%)
Summary of sample sizes: 1364, 1364, 1364, 1364, 1364, 1364, ...
Resampling results across tuning parameters:

  k  Accuracy   Kappa
  5  0.6160284  0.5334295
  7  0.6153099  0.5324363
  9  0.6165183  0.5337183

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 9.
> knn.loocv_pred <- predict(knn.loocv , data.valid)
> knn.loocv_cM <- table(knn.loocv_pred, data.valid$Class)
```

```
> knn.loocv_summary <-
            data.frame( Bias = mean(predict(knn.loocv, data.train) != dat
           a.train$Class),
+                                   Variance = mean(knn.loocv_pred != da
           ta.valid$Class),
+                                   Accuracy = sum(diag(knn.loocv_cM)) /
            sum(knn.loocv_cM))
> knn.loocv_summary
       Bias  Variance  Accuracy
1 0.2196914 0.2140059 0.7859941
```

4. Boosting

Multinational Logistic Regression:

```
> # Multinational Logistic Regression #
> log.boot <- train(Class ~., data = data,
+                   method = "multinom",
+                   trControl = trainControl(
+                     method = "boot",
+                     number = 10,
+                     verboseIter = T
+                   ))
+ Resample01: decay=0e+00
# weights:  126 (102 variable)
initial  value 26485.783039
iter  10 value 19351.454575
......
final  value 2780.792544
stopped after 100 iterations
> print(log.boot)
Penalized Multinomial Regression

13611 samples
   16 predictor
    7 classes: 'BARBUNYA', 'BOMBAY', 'CALI', 'DERMASON', 'HOROZ', 'SEKE
          R', 'SIRA'

No pre-processing
Resampling: Bootstrapped (10 reps)
Summary of sample sizes: 13611, 13611, 13611, 13611, 13611, 13611, ...
Resampling results across tuning parameters:

  decay  Accuracy   Kappa
  0e+00  0.9268254  0.9114270
```

```
  1e-04  0.9268865  0.9115006
  1e-01  0.9224944  0.9061742

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was decay = 1e-04.
> log.boot_pred <- predict(log.boot , data.valid)
> log.boot_cM <- table(log.boot_pred, data.valid$Class)
> log.boot_summary <-
            data.frame( Bias = mean(predict(log.boot, data.train) != data
           .train$Class),
+                                Variance = mean(log.boot_pred != data
           .valid$Class),
+                                Accuracy = sum(diag(log.boot_cM)) / s
           um(log.boot_cM))
> log.boot_summary
        Bias   Variance  Accuracy
1 0.07179595 0.07664055 0.9233595
```

SVM:

```
> # Support Vector Machine Classification
> svm.boot <- train(Class ~., data = data,
+                   method = "svmLinear",
+                   trControl = trainControl(
+                     method = "boot",
+                     number = 10,
+                     verboseIter = T
+                   ))
+ Resample01: C=1
- Resample01: C=1
……
- Resample10: C=1
Aggregating results
Fitting final model on full training set
> print(svm.boot)
Support Vector Machines with Linear Kernel

13611 samples
   16 predictor
    7 classes: 'BARBUNYA', 'BOMBAY', 'CALI', 'DERMASON', 'HOROZ', 'SEKE
          R', 'SIRA'

No pre-processing
Resampling: Bootstrapped (10 reps)
Summary of sample sizes: 13611, 13611, 13611, 13611, 13611, 13611, ...
```

```
Resampling results:

  Accuracy   Kappa
  0.9254708  0.9098593

Tuning parameter 'C' was held constant at a value of 1
> svm.boot_pred <- predict(svm.boot , data.valid)
> svm.boot_cM <- table(svm.boot_pred, data.valid$Class)
> svm.boot_summary <-
            data.frame( Bias = mean(predict(svm.boot, data.train) != data
           .train$Class),
+                                  Variance = mean(svm.boot_pred != data
           .valid$Class),
+                                  Accuracy = sum(diag(svm.boot_cM)) / s
           um(svm.boot_cM))
> svm.boot_summary
        Bias   Variance  Accuracy
1 0.07148105 0.07492654 0.9250735
```

Decision tree:

```
> # Decision Tree
> tree.boot  <- train(Class ~., data = data,
+                     method = "rpart",
+                     trControl = trainControl(
+                       method = "boot",
+                       number = 10,
+                       verboseIter = T
+                     ))
+ Resample01: cp=0.1647
- Resample01: cp=0.1647
……
- Resample10: cp=0.1647
Aggregating results
Selecting tuning parameters
Fitting cp = 0.165 on full training set
> print(tree.boot)
CART

13611 samples
   16 predictor
    7 classes: 'BARBUNYA', 'BOMBAY', 'CALI', 'DERMASON', 'HOROZ', 'SEKE
          R', 'SIRA'

No pre-processing
```

```
Resampling: Bootstrapped (10 reps)
Summary of sample sizes: 13611, 13611, 13611, 13611, 13611, 13611, ...
Resampling results across tuning parameters:

  cp          Accuracy   Kappa
  0.1647293  0.5558049  0.4369662
  0.1705912  0.4681566  0.3200987
  0.1992052  0.3311703  0.1178575

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was cp = 0.1647293.
> tree.boot_pred <- predict(tree.boot , data.valid, type = "raw")
> tree.boot_cM <- table(tree.boot_pred, data.valid$Class)
> tree.boot_summary <-
            data.frame( Bias = mean(predict(tree.boot, data.train, type =
            "raw") != data.train$Class),
+                                   Variance = mean(tree.boot_pred != fa
          ctor(data.valid$Class, ordered = FALSE )),
+                                   Accuracy = sum(diag(tree.boot_cM)) /
            sum(tree.boot_cM))
> tree.boot_summary
       Bias  Variance  Accuracy
1 0.4682481 0.4608227 0.5391773
```

KNN:

```
> # K-Nearest Neighbor
> knn.boot <- train(Class ~., data = data,
+                   method = "knn",
+                   trControl = trainControl(
+                     method = "boot",
+                     number = 10,
+                     verboseIter = T
+                   ))
+ Resample01: k=5
- Resample01: k=5
+ Resample01: k=7
……
+ Resample10: k=9
- Resample10: k=9
Aggregating results
Selecting tuning parameters
Fitting k = 5 on full training set
> print(knn.boot)
k-Nearest Neighbors
```

```
20.
21. 13611 samples
22.    16 predictor
23.     7 classes: 'BARBUNYA', 'BOMBAY', 'CALI', 'DERMASON', 'HOROZ', 'SEKE
            R', 'SIRA'
24.
25. No pre-processing
26. Resampling: Bootstrapped (10 reps)
27. Summary of sample sizes: 13611, 13611, 13611, 13611, 13611, 13611, ...
28. Resampling results across tuning parameters:
29.
30.   k  Accuracy   Kappa
31.   5  0.6998387  0.6358444
32.   7  0.6993312  0.6350086
33.   9  0.6990990  0.6345673
34.
35. Accuracy was used to select the optimal model using the largest value.
36. The final value used for the model was k = 5.
37. > knn.boot_pred <- predict(knn.boot , data.valid)
38. > knn.boot_cM <- table(knn.boot_pred, data.valid$Class)
39. > knn.boot_summary <-
             data.frame( Bias = mean(predict(knn.boot, data.train) != data
            .train$Class),
40. +                                    Variance = mean(knn.boot_pred != data
            .valid$Class),
41. +                                    Accuracy = sum(diag(knn.boot_cM)) / s
            um(knn.boot_cM))
42. > knn.boot_summary
43.        Bias  Variance  Accuracy
44. 1 0.1792799 0.1755632 0.8244368
```

5. Original classification result

Here is the result that have not use ensemble techniques.

```
1.  > data.valid$Class <- factor(data.valid$Class, ordered = T)
2.  > log.sum <-
             data.frame( Bias = mean(predict(log.fit, data.train) != data.
            train$Class),
3.  +             Variance = mean(log.pred != data.valid$Class),
4.  +             Accuracy = sum(diag(log.cM)) / sum(log.cM))
```

```
> svm.sum <-
             data.frame( Bias = mean(predict(svm.fit, data.train) != data.
            train$Class),
+             Variance = mean(svm.pred != data.valid$Class),
+             Accuracy = sum(diag(svm.cM)) / sum(svm.cM))
> tree.sum <-
             data.frame( Bias = mean(predict(log.fit, data.train) != data.
            train$Class),
+             Variance = mean(tree.pred != data.valid$Class),
+             Accuracy = sum(diag(tree.cM)) / sum(tree.cM))
> knn.sum <- data.frame( Variance = mean(knn.pred != data.valid$Class),
+             Accuracy = sum(diag(knn.cM)) / sum(knn.cM))
> log.sum
        Bias   Variance  Accuracy
1 0.07232077 0.07713026 0.9228697
> svm.sum
        Bias   Variance Accuracy
1 0.06623281 0.07590597 0.924094
> tree.sum
        Bias Variance Accuracy
1 0.07232077 0.130999 0.869001
> knn.sum
   Variance  Accuracy
1 0.3574927 0.6425073
```

6. Comparison

Here we put all the ensemble techniques result together to compare with the classification without ensemble techniques.

In Logistic Regression we can see all three ensemble techniques have improved performance compared to the original one slightly, Bias have been decreased, Variance have been decreased, and accuracy have been increased. It also shows that Logistic Regression is easy to get overfitting, by using ensemble techniques can be very helpful for prevent Logistic Regression get over fitting. All three ensemble techniques performed very close, cross validation did slightly better.

In Support vector machines we get similar result as Logistic Regression, the interesting thing is the bias have been increased, but the variance has been decreased, and accuracy have been increased which is pretty good. Ensemble techniques have made a better tread off between Bias and Variance. Another interesting thing is all three ensemble techniques gives the same result. This may be because of fold have been set to 10 which is not enough to bring a big difference between them. Or maybe because of this SVM model completion is close to saturation.

For the decision tree, the performance have significantly worse, this may be due to ensemble techniques error or need more optimized calculations and need adjustment to the parameters.

In KNN we can see the ensemble techniques have been very effectively in improving performance in all the aspects. The boosting gives the best result, this is due to in subsequent bags, boosting has chosen those data instances that had been modeled poorly in the overall system before, so that the performance can keep being improved very effectively.

```
> # Comparison
> log.cv_summary
        Bias   Variance  Accuracy
1 0.07179595 0.07639569 0.9236043
> log.loocv_summary
        Bias   Variance  Accuracy
1 0.07179595 0.07664055 0.9233595
> log.boot_summary
        Bias   Variance  Accuracy
1 0.07179595 0.07664055 0.9233595
> log.sum
        Bias   Variance  Accuracy
1 0.07232077 0.07713026 0.9228697


> svm.cv_summary
        Bias   Variance  Accuracy
1 0.07148105 0.07492654 0.9250735
```

```
> svm.loocv_summary
        Bias   Variance  Accuracy
1 0.07148105 0.07492654 0.9250735
> svm.boot_summary
        Bias   Variance  Accuracy
1 0.07148105 0.07492654 0.9250735
> svm.sum
        Bias   Variance Accuracy
1 0.06623281 0.07590597 0.924094


> tree.cv_summary
       Bias  Variance  Accuracy
1 0.4682481 0.4608227 0.5391773
> tree.loocv_summary
       Bias  Variance  Accuracy
1 0.4682481 0.4608227 0.5391773
> tree.boot_summary
       Bias  Variance  Accuracy
1 0.4682481 0.4608227 0.5391773
> tree.sum
        Bias Variance Accuracy
1 0.07232077 0.130999 0.869001


> knn.cv_summary
       Bias  Variance  Accuracy
1 0.1783353 0.1767875 0.8232125
> knn.loocv_summary
       Bias  Variance  Accuracy
1 0.2196914 0.2140059 0.7859941
> knn.boot_summary
       Bias  Variance  Accuracy
1 0.1792799 0.1755632 0.8244368
> knn.sum
   Variance  Accuracy
1 0.3574927 0.6425073
```