PREDICTIVE ANALYTICS

Northeastern University

ALY6020, SPRING 2020

MODULE 4 PROJECT ASSIGNMENT

WEEK 4: RANDOM FORESTS

SUBMITTED BY: SHIVANI ADSAR

NUID: 001399374

SUBMITTED TO: PROF. NA YU

DATE: 05/02/2020

Introduction

The assignment provides practical experience of implementing the Random Forest and Support Vector Machine algorithm. They are supervised machine learning algorithm used for classification. We have used the "Glass" and "Abalone" datasets to perform classification analysis and predictions.

Analysis

Forensic Glass Dataset

- The Glass Dataset is an inbuilt dataset in R studio. This dataset has 214 instances and 10 attributes. The dataset is used for studying the classifications of glass.[1]
- In order to perform the predictions, we have used the Random Forest and Support Vector Machine algorithms.

Importing the Dataset

- The Glass dataset has been imported initially into a dataframe. Using the "str" function, the structure of the dataset has been analysed into 214 observations and 10 variables.

```
> str(glass)
'data.frame':   214 obs. of  10 variables:
 $ RI  : num  3.01 -0.39 -1.82 -0.34 -0.58 ...
 $ Na  : num  13.6 13.9 13.5 13.2 13.3 ...
 $ Mg  : num  4.49 3.6 3.55 3.69 3.62 3.61 3.6 3.61 3.58 3.6 ...
 $ Al  : num  1.1 1.36 1.54 1.29 1.24 1.62 1.14 1.05 1.37 1.36 ...
 $ Si  : num  71.8 72.7 73 72.6 73.1 ...
 $ K   : num  0.06 0.48 0.39 0.57 0.55 0.64 0.58 0.57 0.56 0.57 ...
 $ Ca  : num  8.75 7.83 7.78 8.22 8.07 8.07 8.17 8.24 8.3 8.4 ...
 $ Ba  : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Fe  : num  0 0 0 0 0 0.26 0 0 0 0.11 ...
 $ type: Factor w/ 6 levels "WinF","WinNF",..: 1 1 1 1 1 1 1 1 1 1 ...
```

Fig.1: Structure of the Glass dataset

The "str" function is used to view the structure of the dataset for understanding the variables for further analysis.

- We have used the set.seed() function to set the sequence for generation of random numbers in a particular sequence.[1]
- Moreover, the randomForest package is installed as we have executed random forest on a function. This package is used for classification and regression analysis.

```
> set.seed(17)
> fgl.rf <- randomForest(type ~ ., data = glass, mtry = 2, importance = TRUE, do.trace = 100)
ntree      OOB      1      2      3      4      5      6
  100:  20.09% 14.29% 18.42% 58.82% 23.08% 22.22% 13.79%
  200:  21.03% 10.00% 22.37% 64.71% 23.08% 33.33% 13.79%
  300:  18.69% 10.00% 18.42% 64.71% 23.08% 11.11% 13.79%
  400:  19.16% 11.43% 17.11% 64.71% 23.08% 22.22% 13.79%
  500:  19.63% 11.43% 19.74% 58.82% 23.08% 22.22% 13.79%
```

Fig.2: Output of Random Forest() function

Since we are predicting on the "type" variable given in the glass data. Also, we are taking 2 variables for splitting at our tree node, hence mtry=2.We have indicated TRUE for importance which tells the accuracy if we add or delete variables.

- It can be observed from the above output, that, for the 100th n tree value, the OOB error rate is 20%. Similarly, for 200th tree, the value is 21% etc, 18.69% of error rate for 300th n tree, 19.16% for 400th n tree and 19.63% for 500th n tree.

- Further, we have printed the results of the Random Forest function.

```
> print(fgl.rf)

Call:
 randomForest(formula = type ~ ., data = glass, mtry = 2, importance = TRUE,      do.trace = 100)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 2

        OOB estimate of  error rate: 19.63%
Confusion matrix:
      WinF WinNF Veh Con Tabl Head class.error
WinF    62     6   2   0    0    0   0.1142857
WinNF   11    61   1   1    1    1   0.1973684
Veh      7     3   7   0    0    0   0.5882353
Con      0     2   0  10    0    1   0.2307692
Tabl     0     2   0   0    7    0   0.2222222
Head     1     3   0   0    0   25   0.1379310
```

Fig.3: Output of print function

We can observe that the random forest algorithm has split in to 2 variables with the error rate being 19.63%. The confusion matrix has classified into categories of glass types. Also, the function is forming 500 trees with classification.

- The confusion matrix has classified the observations in to 6 glass categories like Window float glass (Wf), Window non-float glass (Wnf), vehicle window glass (Veh), containers (Con), tableware (Tabl) and vehicle headlamps (Head). As observed, the class error for vehicle window glass is the highest while for tableware is lowest, 0.11.

Model Comparison

- We have compared the random forest with the support vector machine using 10 repetitions and the errorrest functions.
  - We have used the "ipred" library which is used for improving the predictors by bagging and classification. Also, the set.seed(131) function is used for fixing a constant series of numbers for random number generation.
  - We have executed the "errorrest" for estimating the error rate of the classifier. This has been executed for 10 iterations in the "glass" dataset on our Random Forest model.

```
> library(ipred)
> set.seed(131)
> error.RF <- numeric(10)
> for(i in 1:10) error.RF[i] <- errorest(type ~ ., data = glass, model = randomForest, mtry = 2)$error
> summary(error.RF)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.1729  0.1869  0.1963  0.1967  0.2091  0.2150
```

Fig.4: Output of errorrest() function on Random Forest

As we can observe, the errorrest() function displays the minimum, median, mean and maximum values of the error rates, with Mean being 0.19, Median being 0.19, Maximum being 0.21 and Minimum as 0.17.

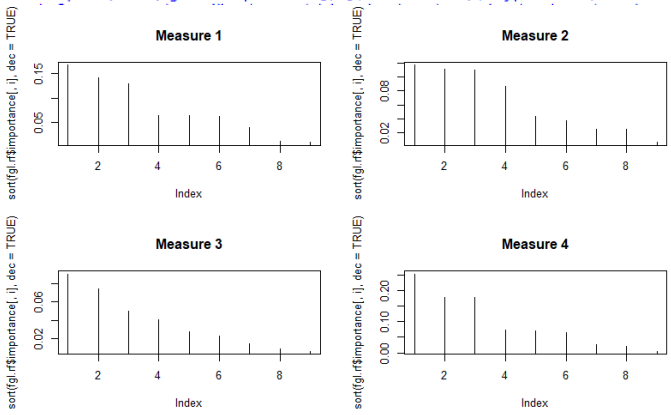
  - We have used the "e1071" package for Support Vector Machine implementation that performs classification analysis. A Support Vector Machine algorithm is a classifier that classifies the data based on a hyperplane and categorizes the data.

```
> library(e1071)
> set.seed(563)
> error.SVM <- numeric(10)
> for (i in 1:10) error.SVM[i] <- errorest(type ~ ., data = glass,model = svm, cost = 10, gamma = 1.5)$error
> summary(error.SVM)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.3505  0.3657  0.3762  0.3762  0.3820  0.4112
```

Fig.5: Output of errorest() function on SVM

- As we can observe, the errorrest() function displays the minimum, median, mean and maximum values of the error rates, with Mean being 0.37, Median being 0.37, Maximum being 0.41 and Minimum as 0.35.
- On the basis of the errorrest() analysis on Random Forest and Support Vector Machine algorithms, we can see that, the error rate for Random Forest algorithm is lesser in comparison to the Support Vector Machine algorithm. The median error rate for Random Forest being 19% and Support Vector Machine being 37%.
- Further, we have used the par(mfrow=c(2,2)) function which is used for displaying a multiple panel layout with 2 rows and 2 columns. Also a loop has been executed for 4 iterations to plot the measures.

```
> par(mfrow = c(2, 2))
> for (i in 1:4)
+ for (i in 1:4)
+     plot(sort(fgl.rf$importance[,i], dec = TRUE), type = "h", main = paste("Measure", i))
```



As we can observe, the 4 Measures as per the importance have been plotted. The Measure 1 shows that a distinct differentiation with least error rate of 20% as compared to other Measures.

Fig.6: Variable Importance for Glass Dataset

- We can conclude that, after analysing the "Glass" dataset using Random Forest and Support Vector Machine algorithms, the error rate for Random Forest was 19% while for Support Vector Machine, the errorrest() function showed an error rate of 37% which was comparatively higher. Hence, Random Forest shows better predictions for the "Glass" dataset.

Abalone Dataset

- The Abalone Dataset has been taken from the UCI Machine Learning Repository. This dataset has 4177 instances and 8 attributes. The dataset is used for studying the ages of male and female abalone shells.[2]
- In order to perform the predictions, we have used the Random Forest and Support Vector Machine Algorithm.

Importing the Dataset

- The Abalone dataset has been imported initially into a dataframe. Using the "str" function, the structure of the dataset has been analysed into 214 observations and 10 variables.

```
> str(abalone)
'data.frame':   4177 obs. of  9 variables:
 $ Sex           : Factor w/ 3 levels "F","I","M": 3 3 1 3 2 2 1 1 3 1 ...
 $ Length        : num  0.455 0.35 0.53 0.44 0.33 0.425 0.53 0.545 0.475 0.55 ...
 $ Diameter      : num  0.365 0.265 0.42 0.365 0.255 0.3 0.415 0.425 0.37 0.44 ...
 $ Height        : num  0.095 0.09 0.135 0.125 0.08 0.095 0.15 0.125 0.125 0.15 ...
 $ Whole.weight  : num  0.514 0.226 0.677 0.516 0.205 ...
 $ Shucked.weight: num  0.2245 0.0995 0.2565 0.2155 0.0895 ...
 $ Viscera.weight: num  0.101 0.0485 0.1415 0.114 0.0395 ...
 $ Shell.weight  : num  0.15 0.07 0.21 0.155 0.055 0.12 0.33 0.26 0.165 0.32 ...
 $ Rings         : int  15 7 9 10 7 8 20 16 9 19 ...
```

The "str" function is used to view the structure of the dataset for understanding the variables for further analysis.

Fig.7: Structure of the Abalone dataset

- As we can observe, the "Sex" parameter is a factor and needs to be converted into a factor variable. Also, we have checked our dataset for null values to clean the data.

```
> abalone$Sex<-as.factor(abalone$Sex)
> sum(is.na(abalone))
[1] 0
```

As we can observe, our data contains 0 null values and is ready for further processing.

Fig.8: Conversion into Factor

- We have used the set.seed() function to set the sequence for generation of random numbers in a particular sequence.[1]
- Moreover, the randomForest package is installed as we have executed random forest on a function. This package is used for classification and regression analysis.

```
> set.seed(17)
> abalone.rf <- randomForest(Sex ~ ., data = abalone, mtry = 2, importance = TRUE, do.trace = 100)
ntree      OOB      1      2      3
  100:  46.25% 60.37% 24.29% 53.47%
  200:  45.15% 58.91% 23.85% 52.09%
  300:  45.03% 59.53% 23.92% 51.18%
  400:  45.20% 59.91% 23.77% 51.44%
  500:  45.03% 60.29% 23.55% 50.85%
```

Since we are predicting on the "Sex" variable given in the Abalone data. Also, we are taking 2 variables for splitting at our tree node, hence mtry=2.We have indicated TRUE for importance which tells the accuracy if we add or delete variables.

Fig.9: Output of Random Forest() function

- It can be observed from the above output, that, for the 100th n tree value, the OOB error rate is 46%. Similarly, for 200th tree, the value is 45.15% etc, 45.03% of error rate for 300th n tree, 45.20% for 400th n tree and 45.03% for 500th n tree.

- Further, we have printed the results of the Random Forest function.

```
> print(abalone.rf)

Call:
 randomForest(formula = Sex ~ ., data = abalone, mtry = 2, importance = TRUE,      do.trace = 100)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 2

        OOB estimate of  error rate: 45.27%
Confusion matrix:
    F    I   M class.error
F 512  158 637   0.6082632
I 105 1031 206   0.2317437
M 530  255 743   0.5137435
```

We can observe that the random forest algorithm has split in to 2 variables with the error rate being 45.27%. The confusion matrix has classified into categories of glass types. Also, the function is forming 500 trees with classification.

Fig.10: Output of print function

- The confusion matrix has classified the observations in to 3 Sex categories like Females, Males and Infants. As observed, the class error for Females is the highest while for Infants is lowest, 0.23.

<u>Model Comparison</u>

- We have compared the random forest with the support vector machine using 10 repetitions and the errorrest functions.

- We have used the “ipred” library which is used for improving the predictors by bagging and classification. Also, the set.seed(131) function is used for fixing a constant series of numbers for random number generation.
- We have executed the “errorrest” for estimating the error rate of the classifier. This has been executed for 10 iterations in the “glass” dataset on our Random Forest model.

```
> library(ipred)
> set.seed(131)
> error.RF <- numeric(10)
> for(i in 1:10) error.RF[i] <- errorest(Sex ~ ., data = abalone, model = randomForest, mtry = 2)$error
> summary(error.RF)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.4398  0.4481  0.4500  0.4491  0.4516  0.4551
```

Fig.11: Output of errorrest() function on Random Forest

As we can observe, the errorrest() function displays the minimum, median, mean and maximum values of the error rates, with Mean being 0.44, Median being 0.45, Maximum being 0.45 and Minimum as 0.43.

- We have used the “e1071” package for Support Vector Machine implementation that performs classification analysis. A Support Vector Machine algorithm is a classifier that classifies the data based on a hyperplane and categorizes the data.

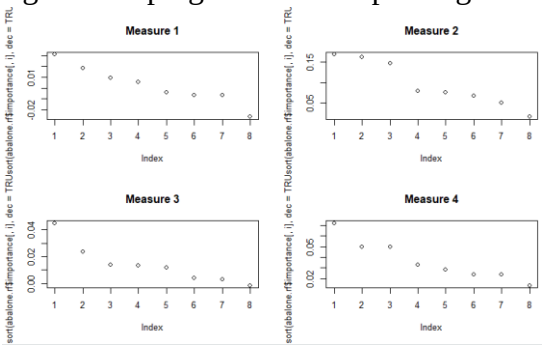
```
> library(e1071)
> set.seed(563)
> error.SVM <- numeric(10)
> for (i in 1:10) error.SVM[i] <- errorest(Sex ~ ., data = abalone,model = svm, cost = 10, gamma = 1.5)$error
> summary(error.SVM)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.4496  0.4511  0.4531  0.4529  0.4545  0.4563
```

Fig.12: Output of errorest() function on SVM

- As we can observe, the errorrest() function for SVM displays the minimum, median, mean and maximum values of the error rates, with Mean being 0.45, Median being 0.45, Maximum being 0.456 and Minimum as 0.44.
- On the basis of the errorrest() analysis on Random Forest and Support Vector Machine algorithms, we can see that, the error rate for Random Forest algorithm is lesser in comparison to the Support Vector Machine algorithm. The mean error rate for Random Forest being 44% and Support Vector Machine being 45%.
- Further, we have used the par(mfrow=c(2,2)) function which is used for displaying a multiple panel layout with 2 rows and 2 columns. Also a loop has been executed for 4 iterations to plot the measures.

```
> par(mfrow = c(2, 2))
> for (i in 1:4)
+   plot(sort(abalone.rf$importance[,i], dec = TRUE), Sex = "h", main = paste("Measure", i))
```

Fig.13: Looping function for plotting the measures



As we can observe, the 4 Measures as per the importance have been plotted. The Measure 2 shows that a distinct differentiation with least error rate of 20% as compared to other Measures.

Fig.14: Variable Importance for Abalone Dataset

Conclusion

- On the basis of our analysis, we can conclude that, after analysing the “Glass” dataset using Random Forest and Support Vector Machine algorithms, the error rate for Random Forest was 19% while for Support Vector Machine, the errorrest() function showed an error rate of 37% which was comparatively higher. Hence, Random Forest shows better predictions for the “Glass” dataset and will give more accuracy.
- Also, the analysis on the “Abalone” dataset using Random Forest and Support Vector Machine Algorithm states that, the error rate for both the algorithms is almost same being 45% and hence both the algorithms can be implemented, preferably Random Forest as it has a lesser error rate of 44 %. Hence, Random Forest algorithm will yield more accuracy.

References

[1] STRICKLAND, PRESIDENT JEFFREY. PREDICTIVE ANALYTICS USING R. LULU COM, 2015.

[2] UCI Machine Learning Repository: Abalone Data Set, archive.ics.uci.edu/ml/datasets/Abalone.