Application of Random Forest Algorithm on Feature Subset Selection and Classification and Regression

Jitendra Kumar Jaiswal
Research Scholar
School of Advanced Sciences,
VIT University, Vellore, India
jitendra.kjaiswal@vit.ac.in

Rita Samikannu
Associate Professor
School of Advanced Sciences,
VIT University, Vellore, India
ritasamikannu@vit.ac.in

Abstract: **Feature subset selection becomes quite important and predominant in the case of data sets those are contained with higher number of variables. It discards insignificant variables and produces efficient and improved prediction performance on the class variables that is more cost effective and more reliable understanding of the data. Random forest has been emerged as a quite efficient and robust algorithm that can handle feature selection problem even with the higher number of variables. It is also very much efficient while dealing with Missing data imputation, classification, and regression problems. It can also handle outliers and noisy data very well. In this paper we applied the concept of random forest algorithm on the feature subset selection and classification and regression to perform the comparative study of the random forest algorithm in different perspectives.**

Keywords: Random Forest; Feature Selection; Classification; Regression; Gini Index; Wrapper and Filter Methods.

I. Introduction

In the area of data processing and analysis, a dataset may have many numbers of variables or attributes which determine the applicability and usability of the data. It is obvious that, every attribute may not be contributing substantially for different applications rather than a subset of variables can provide us an equivalent and effective attention. Finding the subset or relevant features may be termed as feature selection or feature subset selection in the area of machine learning or statistics. This is also known as variable selection, attribute selection, variable or attribute subset selection. This approach may reduce the data training time and effort, rationalize the data interpretation, and reduce the overfitting scenario by enhanced data generalization [1].

In this paper we have considered the Chronic_Kidney_Disease data from UCI Machine Learning Repository [2]. For the determination of feature selection, we have followed the *Boruta* package in the R programming language [3]. This package is based on wrapper algorithms basically and works on the random forest method for the determination of the subset of the important features. This algorithm performs a top-down approach for relevant features with the comparison on the set of original attributes.

A. *Literature Survey for Feature Selection and Random Forest Algorithm*

The concept of feature selection came into picture after 1995 around. Blum and Langlay [5] had given a survey on two problems: issues of selecting relevant features and issues of selecting relevant examples, and produced a general framework to compare different algorithms.

Many works have been done on the ranking of variables for the features selection [6] [7]. Two well-known methods, Boosting [8] and Bagging [9] were proposed to generate many classifiers and aggregate their results for the classification tree.

We have elaborated random forest algorithm with different tools in section-II and applied it for the feature selection with the help of *Boruta* and *randomForest* packages from the R programming language. In the section-III, we have discussed about classification and regression techniques with the help of *randomForest* package from the R programming language. In the section-IV, we have summarized our paper work and proposed some future scope.

II. Feature Selection by Random Forest

A forest is the collection of trees; a random forest is the collection of classification trees. Classification tree, sometimes also called as decision tree, is the construction of a tree which consist the members of the class variable on its leaf nodes and the entities of other dependent variables reside on the intermediate nodes. Class variables are also called as decision variables or predictor variables, which may comprise, for example, yes/no decision to predict a disease or loan approval, spam/no spam in the case of emails, good/bad/moderate for the product qualities, 0-9 for handwritten digits in the case pattern recognition, etc. Random forests develop many classification trees, and to add a new classification tree to the forest, add it down to the each of the trees in the forest. Each tree provides its classification and we consider it as its vote for that class. The forest considers the classification receiving the most votes from all the trees in the forest.

978-1-5090-5573-9/17 $31.00 © 2017 IEEE
DOI 10.1109/WCCCT.2016.25

Authorized licensed use limited to: University College Dublin. Downloaded on April 27,2023 at 10:38:20 UTC from IEEE Xplore. Restrictions apply.

Construction of the Random Forest

1. If there are N cases in the training set, select all the cases randomly, as a separate set of data other than the original data set.
2. For T number of attributes in the training data set, select $t << T$ attributes at random such that at each node the best set of t variables is used to split the node. The value if t should be stable while tree growing.
3. Each tree will be growing to the extreme possible extent without pruning.

The error rate for the random forest depends on the following two factors:

I. The error rate will increase if and only if the correlation between any two trees from the forest will be increasing.

II. The strength of the tree is determined by the error rate that is lower the error rate stronger the tree and that will strengthen the forest.

Properties of the Random Forest:

- It has been considered as unexcelled algorithm in accuracy.
- It is very much efficient on huge data sets even with hundreds and thousands of input variables without overfitting and there is no requirement data pruning.
- It is applied for the feature subset selection and missing data imputation and performs very efficiently.
- An internal unbiased estimate of the generalization error is produced by the random forest algorithm in the process of forest construction.
- The generated forest can perform well for the future adding data.

Working Process of the Random Forest

There are two data objects generated by the random forest: random attribute data set and original data set. The training set is about two third of the original data set and rest are called as out-of-bag (oob) data, that is used to estimate unbiased classification error as trees are added to the forest.

The proximities are calculated for each pair of the cases after the built of the tree. If the two pairs compete for the same terminal node of the tree, their proximities are increased by one. Eventually the proximities are normalized by dividing by the total number of the trees in the forest.

The Error Estimation with oob Data

Out-of-bag estimation was proposed by Tibshirani [10], as an important ingredient for the calculation of generalization error. It is not required to cross-validate or a separate test to calculate an unbiased error estimate of the validation set in the random forest, since it performs eternally during the execution. Each tree is built using a random sampling with replacement from the original data. About one third of the cases are left out as oob data that are not used in the built of the $p-th$ tree. Put each case from oob data in the build of the $p-th$ tree down to the $p-th$ tree to get a classification. With this process a test classification is achieved for each case in about one-third of the trees. Eventually, consider q to be the class variable with maximum votes every time from m cases of oob. The oob error is estimated with the factor that q is not equal to the true class of m averaged over all cases. This estimation is unbiased for many cases.

Variable Importance

For the oob cases, let the number of vote cast for the correct class is x and for the permuted oob variables, it is y. the average of $(x\text{-}y)$ over all trees in the forest is considered as the raw importance score for variable x.

If we get these score values independent for every tree, then we compute standard error by standard computation, and the correlation for these scores are quite low. For a huge data set having many number of variables we first run this algorithm for all variables, and again execute same process for the considered variables from the first run.

Gini Importance

Gini index is the accuracy (or inaccuracy) measure of the decision trees. The Gini inaccuracy criterion for the parent node is always higher than the two descendent nodes that are split from the parent node. Improvement in the Gini decrease of each individual attribute for every tree in the forest provides surplus variable importance that is often quite consistent with the permutation importance measure.

Variables Interactions

Variables interaction is calculated at the terminal node with the spit of variables either in left side or in right side. There may be k interaction on m variables on a terminal node that makes a split on k either systematically less possible or more possible. This interaction is calculated with Gini values for each tree in the forest. This is considered as rank for each tree for every two variables and the absolute differences of ranks are averaged over all trees.

The rank for each tree is also estimated with hypothesis that assumes the independence of two variables with each other. Subtract the lower rank value from higher rank value. Higher difference implies that a split on higher ranked variable inhibits a split on lower rank variable and vice versa.

Proximities Estimation

Proximity estimation is a very important tool in the random forest. It creates a $N \times N$ matrix. After growing a tree, put complete data down to this tree. If the two variables at the same terminal node are having proximities, increase it by one for both. Eventually, normalize it by taking average over the total number of trees in the forest.

With the large data set $N \times N$ matrix could not fit into the processing memory, so $N \times T$ matrix is processed, where T is the total number of trees in the forest.

Authorized licensed use limited to: University College Dublin. Downloaded on April 27,2023 at 10:38:20 UTC from IEEE Xplore. Restrictions apply.

Data Scaling

The proximity between the two variables *p* and *q* is a matrix {*proximity (p, q)*} which is symmetric, positive definite, and bounded over by 1, with the diagonal elements 1. In the Euclidean space of dimension, the values {1-*proximity (p, q)*} are squared distances and not exceeding the total number of cases.

Let the average value of *proximity (p, q)* is *proximity (-, q)* over the first coordinate, *proximity (p, -),* over the second coordinate, and *proximity (-, -)*, the average over both coordinate. Then the proximity matrix

$$pm\,(p,q) = 0.5 * [\,proximity\ \ (p,q) - proximity\ \ (p,-),\ proximity\ \ (-,q) + proximity\ \ (-,-)]$$

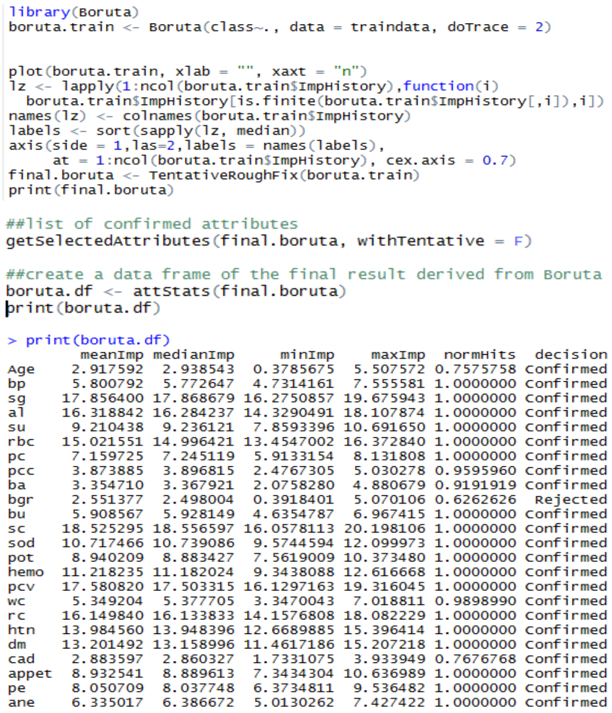
is the distance matrix of the inner product that is symmetric and positive definite. If the Eigen values and Eigen vectors of the proximity matrix *pm* are $l(t)$ and $p_t(p)$ respectively, then the vectors

$$x(p) = (\Theta\, l(1)\, p_1(p), \Theta\, l(2)\, p_2(p), \ldots)$$

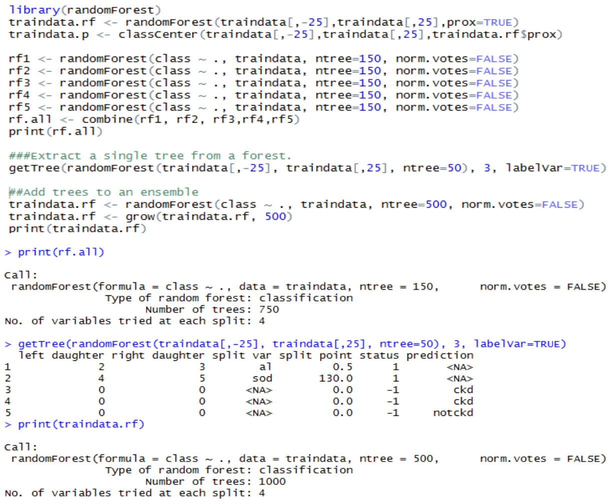
have squared distances between them with the value {1-*proximity (p, q)*}. The values $\Theta\, l(t)\, p_t(p)$ are considered as the t^{th} scaling coordinates.

Random forests perform the approximation of vectors *x(p)* by extracting some of the largest values of Eigenvalues of the *pm* matrix and their corresponding values of the Eigenvectors.

Application of *Boruta* Package from R on the Data set Chronik_Kidney_Disease from the UCI machine learning repository:



```
library(Boruta)
boruta.train <- Boruta(class~., data = traindata, doTrace = 2)


plot(boruta.train, xlab = "", xaxt = "n")
lz <- lapply(1:ncol(boruta.train$ImpHistory),function(i)
  boruta.train$ImpHistory[is.finite(boruta.train$ImpHistory[,i]),i])
names(lz) <- colnames(boruta.train$ImpHistory)
labels <- sort(sapply(lz, median))
axis(side = 1,las=2,labels = names(labels),
     at = 1:ncol(boruta.train$ImpHistory), cex.axis = 0.7)
final.boruta <- TentativeRoughFix(boruta.train)
print(final.boruta)

##list of confirmed attributes
getSelectedAttributes(final.boruta, withTentative = F)

##create a data frame of the final result derived from Boruta
boruta.df <- attStats(final.boruta)
print(boruta.df)

> print(boruta.df)
        meanImp medianImp     minImp    maxImp  normHits  decision
Age    2.917592  2.938543  0.3785675  5.507572 0.7575758 Confirmed
bp     5.800792  5.772647  4.7314161  7.555581 1.0000000 Confirmed
sg    17.856400 17.868679 16.2750857 19.675943 1.0000000 Confirmed
al    16.318842 16.284237 14.3290491 18.107874 1.0000000 Confirmed
su     9.210438  9.236121  7.8593396 10.691650 1.0000000 Confirmed
rbc   15.021551 14.996421 13.4547002 16.372840 1.0000000 Confirmed
pc     7.159725  7.245119  5.9133154  8.131808 1.0000000 Confirmed
pcc    3.873885  3.896815  2.4767305  5.030278 0.9595960 Confirmed
ba     3.354710  3.367921  2.0758280  4.880679 0.9191919 Confirmed
bgr    2.551377  2.498004  0.3918401  5.070106 0.6262626  Rejected
bu     5.908567  5.928149  4.6354787  6.967415 1.0000000 Confirmed
sc    18.525295 18.556597 16.0578113 20.198106 1.0000000 Confirmed
sod   10.717466 10.739086  9.5744594 12.099973 1.0000000 Confirmed
pot    8.940209  8.883427  7.5619009 10.373480 1.0000000 Confirmed
hemo  11.218235 11.182024  9.3438088 12.616668 1.0000000 Confirmed
pcv   17.580820 17.503315 16.1297163 19.316045 1.0000000 Confirmed
wc     5.349204  5.377705  3.3470043  7.018811 0.9898990 Confirmed
rc    16.149840 16.133833 14.1576808 18.082229 1.0000000 Confirmed
htn   13.984560 13.948396 12.6689885 15.396414 1.0000000 Confirmed
dm    13.201492 13.158996 11.4617186 15.207218 1.0000000 Confirmed
cad    2.883597  2.860327  1.7331075  3.933949 0.7676768 Confirmed
appet  8.932541  8.889613  7.3434304 10.636989 1.0000000 Confirmed
pe     8.050709  8.037748  6.3734811  9.536482 1.0000000 Confirmed
ane    6.335017  6.386672  5.0130262  7.427422 1.0000000 Confirmed
```



```
library(randomForest)
traindata.rf <- randomForest(traindata[,-25],traindata[,25],prox=TRUE)
traindata.p <- classCenter(traindata[,-25],traindata[,25],traindata.rf$prox)

rf1 <- randomForest(class ~ ., traindata, ntree=150, norm.votes=FALSE)
rf2 <- randomForest(class ~ ., traindata, ntree=150, norm.votes=FALSE)
rf3 <- randomForest(class ~ ., traindata, ntree=150, norm.votes=FALSE)
rf4 <- randomForest(class ~ ., traindata, ntree=150, norm.votes=FALSE)
rf5 <- randomForest(class ~ ., traindata, ntree=150, norm.votes=FALSE)
rf.all <- combine(rf1, rf2, rf3,rf4,rf5)
print(rf.all)

###Extract a single tree from a forest.
getTree(randomForest(traindata[,-25], traindata[,25], ntree=50), 3, labelVar=TRUE)

##Add trees to an ensemble
traindata.rf <- randomForest(class ~ ., traindata, ntree=500, norm.votes=FALSE)
traindata.rf <- grow(traindata.rf, 500)
print(traindata.rf)

> print(rf.all)

Call:
 randomForest(formula = class ~ ., data = traindata, ntree = 150,      norm.votes = FALSE)
               Type of random forest: classification
                     Number of trees: 750
No. of variables tried at each split: 4

> getTree(randomForest(traindata[,-25], traindata[,25], ntree=50), 3, labelVar=TRUE)
  left daughter right daughter split var split point status prediction
1             2              3        al         0.5      1       <NA>
2             4              5       sod       130.0      1       <NA>
3             0              0      <NA>         0.0     -1        ckd
4             0              0      <NA>         0.0     -1        ckd
5             0              0      <NA>         0.0     -1     notckd
> print(traindata.rf)

Call:
 randomForest(formula = class ~ ., data = traindata, ntree = 500,      norm.votes = FALSE)
               Type of random forest: classification
                     Number of trees: 1000
No. of variables tried at each split: 4
```

The package *Boruta* establishes an iterative comparison between randomly selected variables and the variables from the original data set. Those variables which are having significantly worst importance than the random data set are consecutively rejected.

III. Classification and Regression by Random Forest

For the *categorical type* of class or target variables, decision trees are considered as classification decision trees, for example, the consumer behavior for buying the particular goods (binary target variables, yes/no). For the *continuous variables*, it is called as regression decision trees, for example deciding house prices, that can depend on different factors or variables like size (square ft.) and types (flat, independent) of hoses.

A. Classification

Classification technique is applied to predict the decision values from the class variable for qualitative or categorical types of variables by the calculation with one or more independent or predictor variables. It is widely applied in the fields like computer science (Data Structure), medicine (diagnosis), botany (classification), and psychology (decision theory), etc. It can be very easily visualized as graphically that helps to interpret data more easily (mostly with the few number of variables) than numerical interpretation.

There are numerous applicable classification techniques, or classifiers that can be employed to predict a qualitative response. Some of the widely used classifiers are logistic regression, k-nearest neighbors, Bayes' theorem, etc.

Classification trees are getting attention due to two of its important features: hierarchical property and flexibility.

Construction of the Classification Tree

The spit patterns of the members of the class variables and the class inaccuracy measure are the two important factors. A node is subjected as pure node if it is having the entities form the single class. The objective of the tree formation is to get the pure nodes eventually. We start with

Authorized licensed use limited to: University College Dublin. Downloaded on April 27,2023 at 10:38:20 UTC from IEEE Xplore. Restrictions apply.

the impure nodes with a function of relative frequencies of the classes at that node.

$$\eta(t) = \phi\left(\nu_1, \nu_2, \ldots \nu_j\right)$$

where ν_j represents the relative frequencies of j different classes at the considered node. For the evenly distributed observations, the maximum inaccuracy measure will be $(1/j, 1/j, \ldots 1/j)$ and minimum inaccuracy measure will be $(1, 0, \ldots.. \ 0)$ for all the classes at the considered nodes. ϕ is a symmetric function of $\left(\nu_1, \nu_2, \ldots \nu_j\right)$.

There are different alternatives for inaccuracy measure and different algorithms for the tree implementation. Different tree implementation algorithms follow different inaccuracy measures. The inaccuracy measures are selected to reduce the inaccuracy that the split achieves on the considered node. Hence we can calculate the split performance s at node t as

$$\Delta\eta(s,t) = \eta(t) - \kappa(l)\eta(l) - \kappa(r)\eta(r)$$

where $\kappa(l)$ is left side proportion of the cases by the split, and for the right side, it is $\kappa(r)$.

Gini Index

One of the most important techniques for the inaccuracy measure is Gini index. For the two-class case it can be calculated as follows:

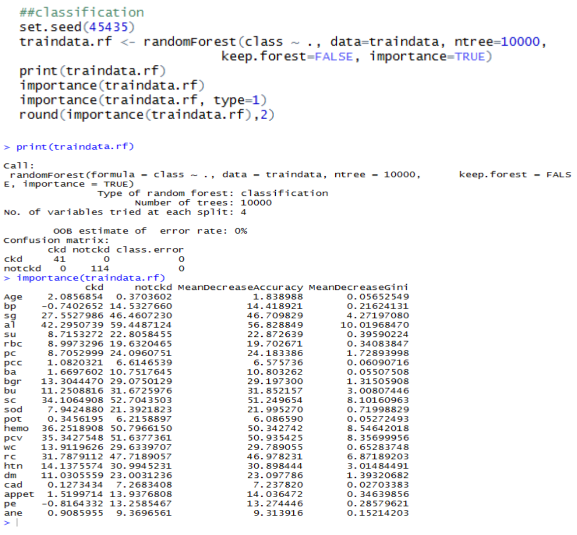
$$\eta(t) = \nu(0 \mid t)\nu(1 \mid t) = \nu(0 \mid t)(1 - \nu(0 \mid t)) = \nu(1 \mid t)(1 - \nu(1 \mid t))$$

where $\nu(0 \mid t)$ or $\nu(1 \mid t)$ is the frequency of the class label at the node.

The generalization of the Gini index with more than two classes can be produced as follows:

$$\eta(t) = \sum_j \nu(j \mid t)(1 - \nu(j \mid t))$$

where $\nu(j \mid t)$ is the relative frequency of the class j at the node t.



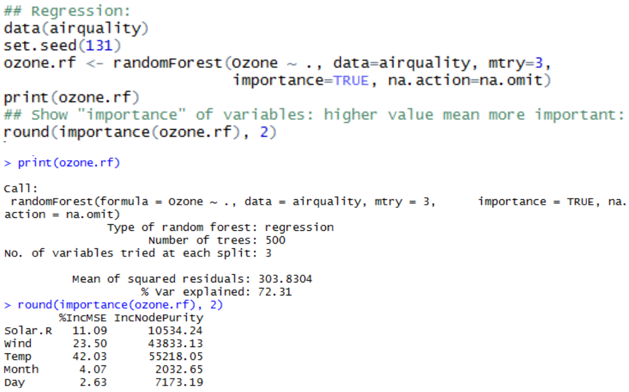
```
##classification
set.seed(45435)
traindata.rf <- randomForest(class ~ ., data=traindata, ntree=10000,
                             keep.forest=FALSE, importance=TRUE)
print(traindata.rf)
importance(traindata.rf)
importance(traindata.rf, type=1)
round(importance(traindata.rf),2)
```

```
> print(traindata.rf)

Call:
 randomForest(formula = class ~ ., data = traindata, ntree = 10000,      keep.forest = FALS
E, importance = TRUE)
               Type of random forest: classification
                     Number of trees: 10000
No. of variables tried at each split: 4

        OOB estimate of  error rate: 0%
Confusion matrix:
       ckd notckd class.error
ckd     41      0           0
notckd   0    114           0
> importance(traindata.rf)
             ckd    notckd MeanDecreaseAccuracy MeanDecreaseGini
Age    2.0856854  0.3703602             1.838988       0.05652549
bp    -0.7402652 14.5327660            14.418921       0.21624131
sg    27.5527986 46.4607230            46.709829       4.27197080
al    42.2950739 59.4487124            56.828849      10.01968470
su     8.7153272 22.8058455            22.872639       0.39590224
rbc    8.9973296 19.6320465            19.702671       0.34083847
pc     8.7052999 24.0960751            24.183386       1.72893998
pcc    1.0820321  6.6146539             6.575736       0.06090716
ba     1.6697602 10.7517645            10.803262       0.05507508
bgr   13.3044470 29.0750129            29.197300       1.31505908
bu    11.2508816 31.6725976            31.852157       3.00807446
sc    34.1064908 52.7043503            51.249654       8.10160963
sod    7.9424880 21.3921823            21.995270       0.71998829
pot    0.3456195  6.2158897             6.086590       0.05272493
hemo  36.2518908 50.7966150            50.342742       8.54642018
pcv   35.3427548 51.6377361            50.935425       8.35699956
wc    13.9119626 29.6339707            29.789055       0.65283748
rc    31.7879112 47.7189057            46.978231       6.87189203
htn   14.1375574 30.9945231            30.898444       3.01484491
dm    11.0305559 23.0031236            23.097786       1.39320682
cad    0.1273434  7.2683408             7.237820       0.02703383
appet  1.5199714 13.9376808            14.036472       0.34639856
pe    -0.8164332 13.2585467            13.274446       0.28579621
ane    0.9085955  9.3696561             9.313916       0.15214203
> |
```

B. *Regression*

Random forest grows trees for the quantitative or numerical values of the class variable as opposite to the classification which grows trees for the qualitative class variables. As a result, output values are also numerical. It is also assumed that the training data is independently selected from the original data set. The mean-square error for any predictor variable X is estimated by $E_{X,y}(y - X)^2$, for the class variable y.

For the regression we have used "Air Quality" data from UCI Machine Learning Repository [4].



```
## Regression:
data(airquality)
set.seed(131)
ozone.rf <- randomForest(Ozone ~ ., data=airquality, mtry=3,
                         importance=TRUE, na.action=na.omit)
print(ozone.rf)
## Show "importance" of variables: higher value mean more important:
round(importance(ozone.rf), 2)
```

```
> print(ozone.rf)

Call:
 randomForest(formula = Ozone ~ ., data = airquality, mtry = 3,      importance = TRUE, na.
action = na.omit)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 3

          Mean of squared residuals: 303.8304
                    % Var explained: 72.31
> round(importance(ozone.rf), 2)
        %IncMSE IncNodePurity
Solar.R   11.09      10534.24
Wind      23.50      43833.13
Temp      42.03      55218.05
Month      4.07       2032.65
Day        2.63       7173.19
```

IV. Conclusion and Further Application

In this paper we considered 'Chronic Kidney Disease' data from UCI machine learning repository and applied random forest algorithm with help of the 'Boruta' package from R programming language for the feature subset selection and achieved the reduced set that is subset of the variables. We also applied the 'randomForest' package from R programming language for the classification and regression, and observed the functioning of random forest method in different perspectives. Further we can apply this random forest algorithm for the data sets with tens, hundreds, or thousands of variables and get the subset of important features.

References

[1]. Gareth James; Daniela Witten; Trevor Hastie; Robert Tibshirani, "An introduction to statistical learning". Springer. p. 204 (2013).

[2]. https://archive.ics.uci.edu/ml/datasets/Chronic_Kidney_Disease, Date: 5th-spt'2016, 11:00am.

[3]. https://cran.r-project.org/web/packages/Boruta/Boruta.pdf, Date: 5th-spt'2016, 11:30am.

[4]. https://archive.ics.uci.edu/ml/datasets/Air+Quality, Date: 6th-spt'2016, 11:30am.

[5]. Avrim L. Blum, Pat Langley, "Selection of relevant features and examples in machine learning", 0004-3702/97/1997 Elsevier Science B.V., Artificial Intelligence 97 (1997) 245-271.

[6]. R. Bekkerman, R. El-Yaniv, N. Tishby, and Y. Winter, "Distributional word clusters vs. words for text categorization". JMLR, 3:1183–1208(2003).

[7]. R. Caruana and V. de Sa, "Benefitting from the variables that variable selection discards". JMLR, 3:1245–1264(2003).

[8]. R. Shapire, Y. Freund, P. Bartlett, and W. Lee, "Boosting the margin: A new explanation for the effectiveness of voting methods". Annals of Statistics, 26 (5):1651–1686 (1998).

[9]. L. Breiman, "Bagging predictors", Machine Learning, 24 (2):123–140 (1996).

[10]. Tibshirani, R. (1996), "Bias, Variance, and Prediction Error for Classification Rules", Technical Report, Statistics Department, University of Toronto.

Authorized licensed use limited to: University College Dublin. Downloaded on April 27,2023 at 10:38:20 UTC from IEEE Xplore. Restrictions apply.