Practical 10

-Shivam Tawari A-58

Aim: Write a program in R for implementing Random forest on a given dataset.

Theory:

Classification:
Classification is the method of predicting the class of a given input data point. Classification problems fall under supervised machine learning method.

There are two types of classification:

- Binary Classification:
When there are only 2 classes in the output variable, we call it Binary classification problem.

- Multiclass Classification:
When there are more than 2 classes to be predicted, this is exactly what multi-class classification means.

Random Forest:
Random Forest algorithm is a supervised classification and regression algorithm.

Random Forest randomly creates a forest with several trees.
Generally, the more trees in the forest the more robust the forest looks like.
Similarly, in the random forest classifier the higher the number of trees in the forest, greater is the accuracy of the results.
Random forest builds multiple Decision trees and glues them together to get a more accurate and stable prediction.
Random Forest is a Bagging method.
Decision Trees are built on the entire dataset using all the predictor variables, whereas Random forest are used to create multiple Decision Trees, such that each decision tree is built only on a part of the dataset.

Creating a Random Forest:

① Create a Bootstrapped Dataset:
Bootstrapping is an estimation method used to make predictions on a data set by resampling it.
To create a bootstrapped data set, we must randomly select samples from the

Original data set.
A point to note here is that we can select the same sample more than once.

② Creating Decision Trees:
Our next task is to build a Decision tree by using the bootstrapped data set created in the previous step.
Since, we're making a random forest we will not consider the entire dataset.

③ Go back to step 1 and repeat:
Random Forest is a collection of Decision Trees, therefore these steps needs to be re-iterated many times in order to create a good prediction.
Each decision tree in a Random forest will give the output and finally the outcome of Random forest will be the majority vote or mean in case of Random Forest Regressor.

④ Predicting the outcome of a New data point:
We first take one observation from testing data and run this data down through all the decision trees and keep a track of the class predicted

by each tree.
After running the data down all the trees in Random forest, we check which class got the majority votes.
To conclude, we bootstrapped the data and used the aggregate from all the trees to make a decision, this process is known as Bagging.

⑤ Evaluate the Model:
The final step is to evaluate the Random Forest model.
In bootstrapping dataset, we leave about 1/3rd of the original dataset. This sample dataset that does not include in the bootstrapped dataset is known as the Out-of-Bag (OOB) dataset. This OOB dataset is then used to check the accuracy of the model.
The proportion of OOB samples that are incorrectly classified is called the Out-of-Bag error.

Code:

```
install. packages("stats")
install. packages ("dplyr")
install. packages ("caret")
install. packages ("random forest")
library (stats)
library (dplyr)
library (caret)
library (random Forest)

mydata = iris
head (my data)

index = sample (2, nrow(data), replace = True,
                                    prob = c (0.7, 0.3))

Training     =   mydata [index == 1,]
Testing      =   mydata [index == 2,]
RFM          =   random forest (species ~, data = Training)
Species_pred =   predict (RFM, Testing)
CFM          =   table (Testing $ Species, Species_pred)

Classification_Accuracy = sum (diag (CFM))/ sum (CFM)
print (paste ("Accuracy:", Classification_Accuracy))
```

Conclusion: Hence, we have successfully implemented Random Forest Algorithm on a given dataset in R.

Code:

```r
# Shivam Tawari A-58
install.packages("stats")
install.packages("dplyr",dependencies = TRUE)
install.packages("caret")
install.packages("randomForest")

library("stats")
library("dplyr")
library("caret")
library("randomForest")

mydata = iris

head(mydata)

str(mydata)
View(mydata)

#splitting dataset
index = sample(2, nrow(mydata), replace=TRUE, prob=c(0.7, 0.3))

#Training data
Training = mydata[index==1,]

#Testing
Testing = mydata[index==2,]

# Model Creation
RFM = randomForest(Species~., data=Training)


# Evaluate Model
Species_Pred = predict(RFM, Testing)
Testing$Species_Pred = Species_Pred

View(Testing)

# Confusion Matrix
CFM = table(Testing$Species, Testing$Species_Pred)
CFM

Classification_Accuracy = sum(diag(CFM)/sum(CFM))
Classification_Accuracy
```

Output:

Console Terminal Jobs

/cloud/project/

```
> library("stats")
> library("dplyr")
> library("caret")
> library("randomForest")
>
> mydata = iris
>
> head(mydata)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
>
> str(mydata)
'data.frame':   150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1
1 1 1 ...
> View(mydata)
>
> #splitting dataset
> index = sample(2, nrow(mydata), replace=TRUE, prob=c(0.7, 0.3))

> #Training data
> Training = mydata[index==1,]
>
> #Testing
> Testing = mydata[index==2,]
>
> # Model Creation
> RFM = randomForest(Species~., data=Training)
>
> # Evaluate Model
> Species_Pred = predict(RFM, Testing)
> Testing$Species_Pred = Species_Pred
>
> View(Testing)
>
> # Confusion Matrix
> CFM = table(Testing$Species, Testing$Species_Pred)
> CFM

             setosa versicolor virginica
  setosa         16          0         0
  versicolor      0         19         1
  virginica       0          0        15
>
> Classification_Accuracy = sum(diag(CFM)/sum(CFM))
> Classification_Accuracy
[1] 0.9803922
```

Model Comparsion:

```
mydata = iris
head(mydata)
str(mydata)

#splitting dataset
index = sample(2, nrow(mydata), replace=TRUE, prob=c(0.7, 0.3))

#Training data
Training = mydata[index==1,]

#Testing
Testing = mydata[index==2,]


## Random Forest
RFM = randomForest(Species~., data=Training)

# Evaluate Model
Species_Pred = predict(RFM, Testing)
Testing$Species_Pred_RF = Species_Pred


# Confusion Matrix
CFM = table(Testing$Species, Testing$Species_Pred_RF)
CFM

Classification_Accuracy_RF = sum(diag(CFM)/sum(CFM))

```

```
## Decision Tree
DT = ctree(Species~., data=Training)

# Evaluate Model
Species_Pred = predict(DT, Testing)
Testing$Species_Pred_DT = Species_Pred

# Confusion Matrix
CFM = table(Testing$Species, Testing$Species_Pred_DT)
CFM

Classification_Accuracy_DT = sum(diag(CFM)/sum(CFM))

## Naive Bayes
NB = naiveBayes(Species~., data=Training)

# Evaluate Model
Species_Pred = predict(NB, Testing)
Testing$Species_Pred_NB = Species_Pred

# Confusion Matrix
CFM = table(Testing$Species, Testing$Species_Pred_NB)
CFM

Classification_Accuracy_NB = sum(diag(CFM)/sum(CFM))

```

```r
## Multinomial Regression
MLR = multinom(Species~., data=Training)

# Evaluate Model
Species_Pred = predict(MLR, Testing)
Testing$Species_Pred_MLR = Species_Pred

# Confusion Matrix
CFM = table(Testing$Species, Testing$Species_Pred_MLR)
CFM

Classification_Accuracy_MLR = sum(diag(CFM)/sum(CFM))

print(paste("Random Forest: ", Classification_Accuracy_RF))
print(paste("Decision Tree: ", Classification_Accuracy_DT))
print(paste("Naive Bayes: ", Classification_Accuracy_NB))
print(paste("Multinomial Logistic Regression: ", Classification_Accuracy_
```

Accuracy Comparison:

```
>
> print(paste("Random Forest: ", Classification_Accuracy_RF))
[1] "Random Forest:  0.931818181818182"
> print(paste("Decision Tree: ", Classification_Accuracy_DT))
[1] "Decision Tree:  0.954545454545455"
> print(paste("Naive Bayes: ", Classification_Accuracy_NB))
[1] "Naive Bayes:  0.977272727272727"
> print(paste("Multinomial Logistic Regression: ", Classification_Accuracy_MLR))
[1] "Multinomial Logistic Regression:  0.931818181818182"
```