CSC 6740 Data Mining

Predicting Blood Donation Patterns

Surabhi Priya- Raghvi Ravi

GeorgiaState University | COLLEGE OF ARTS & SCIENCES

Content

Motivation

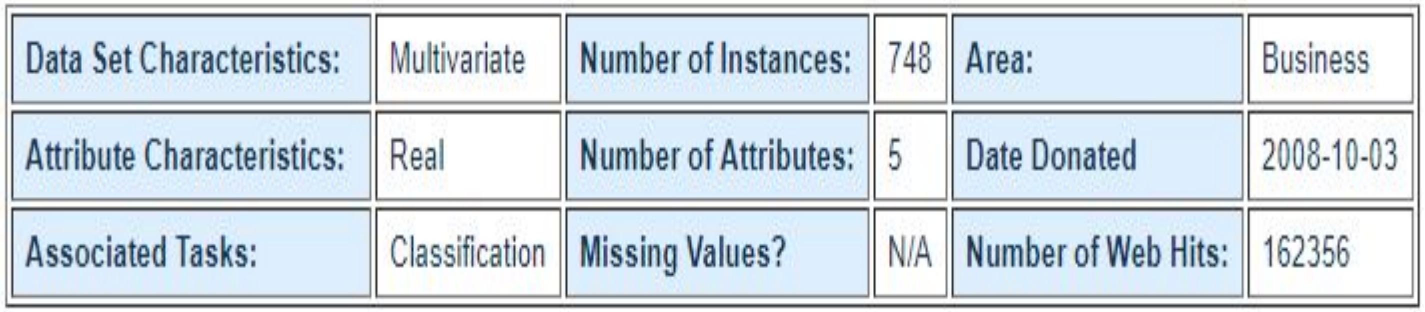
- Blood is the most precious gift that anyone can give to another person – the gift of life.
- A decision to donate your blood can save a life.
- Blood and blood products are used for patients of all ages for many reasons – from cancer patients or surgical patients, to those with battlefield injuries, military members depend on blood donors everyday.
- Approximately 32,000 pints of blood are used each day in the United States.
- One out of every 10 people entering a hospital needs blood
- 60% of the US population is eligible to donate - only 5% do on a yearly basis.
- To predict the possibility of blood donations by analyzing the current Data Set.

Data Selection

Data Set Name: Blood Transfusion Service Center Data Set

Data Source:
https://archive.ics.uci.edu/ml/datasets/Blood+Transfusion+Service+Center

Data Set Description:



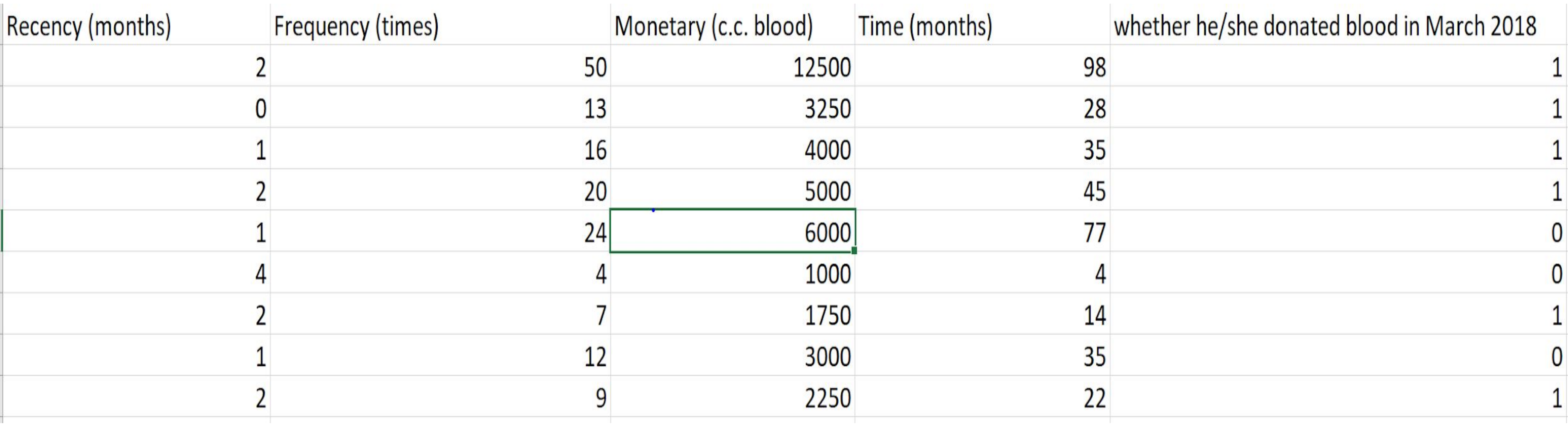
Data Set Characteristics:	Multivariate	Number of Instances:	748	Area:	Business
Attribute Characteristics:	Real	Number of Attributes:	5	Date Donated	2008-10-03
Associated Tasks:	Classification	Missing Values?	N/A	Number of Web Hits:	162356

Attribute Information:

- Recency - months since last donation
- Frequency - total number of donation
- Monetary - total blood donated in c.c.
- Time - months since first donation
- Class - a binary variable representing whether a person donated blood or

Data

Snippet of Data:



Recency (months)	Frequency (times)	Monetary (c.c. blood)	Time (months)	whether he/she donated blood in March 2018
2	50	12500	98	1
0	13	3250	28	1
1	16	4000	35	1
2	20	5000	45	1
1	24	6000	77	0
4	4	1000	4	0
2	7	1750	14	1
1	12	3000	35	0
2	9	2250	22	1

Attribute Information:

- Recency - months since last donation
- Frequency - total number of donation
- Monetary - total blood donated in c.c.
- Time - months since first donation
- Class - a binary variable representing whether a person donated blood or

Data Cleaning

For Data Pre-processing and finding accuracy ‘R’ language is used.

- **Converting .data file into .csv file**

```
input_data <- read.csv(file.choose(),sep=",")
write.csv(input_data,"transfusion.csv", row.names=F)
```

- **Checking the attributes and predicted values for null values**

```
apply (transfusion, MARGIN = 2, FUN = function(x) sum(is.na(x)))
        Recency     Frequency   Monetary   Time   Class
           0            0           0        0      0
```

- **Scaling the datasets**

  All the attributes are scaled to make their values between 0 and 1.

```
transfusion <- read.csv (“Path to .csv file", header = TRUE)
maxs = apply (transfusion, MARGIN = 2, max)
mins = apply (transfusion, MARGIN = 2, min)
scaled_set = as.data.frame (scale (transfusion, center = mins, scale = maxs-mins)
```

- **Looking up for categorical attributes and making them categorical if they are not categorical**

  We will make the class variable categorical in our case.

  ```
  scaled_set$class <- factor(scaled_set$class)   #class must be categorical va
  ```

- **Finding out correlations**

  Look for variables that will contribute to our classification result by observing their correlation with the class variable as well as themselves.

  ```
  correlation_matrix<-cor (scaled_set [,1:5])
  print(correlation_matrix)
  highly_correlated<-findCorrelation (correlation_matrix, cutoff = 0.95)
  print(highly_correlated)
  ```

	Recency	Frequency	Monetary	Time	Class
Recency	1.0000000	-0.1827455	-0.1827455	0.16061809	-0.27986887
Frequency	-0.182745 5	1.0000000	1.0000000	0.63494027	0.21863344
Monetary	-0.1827455	1.0000000	1.0000000	0.63494027	0.21863344
Time	0.1606181	0.6349403	0.6349403	1.00000000	-0.03585441
Class	-0.2798689	0.2186334	0.2186334	-0.03585441	1.00000000

Since Frequency and Monetary have the same values we have removed Monetary from the dataset for the analysis.

Pattern Evaluation

- Now, decide on the number of folds for cross validation. (We have used 5-fold cross validation in our analysis)
- And then, split the data into training dataset and test dataset according to n folds.(train_data – 80% and test_data – 20%)
- For every n fold, we have trained the classifier and tested the model.
- Calculated the accuracy for each classifier by applying following Machine Learning Models:

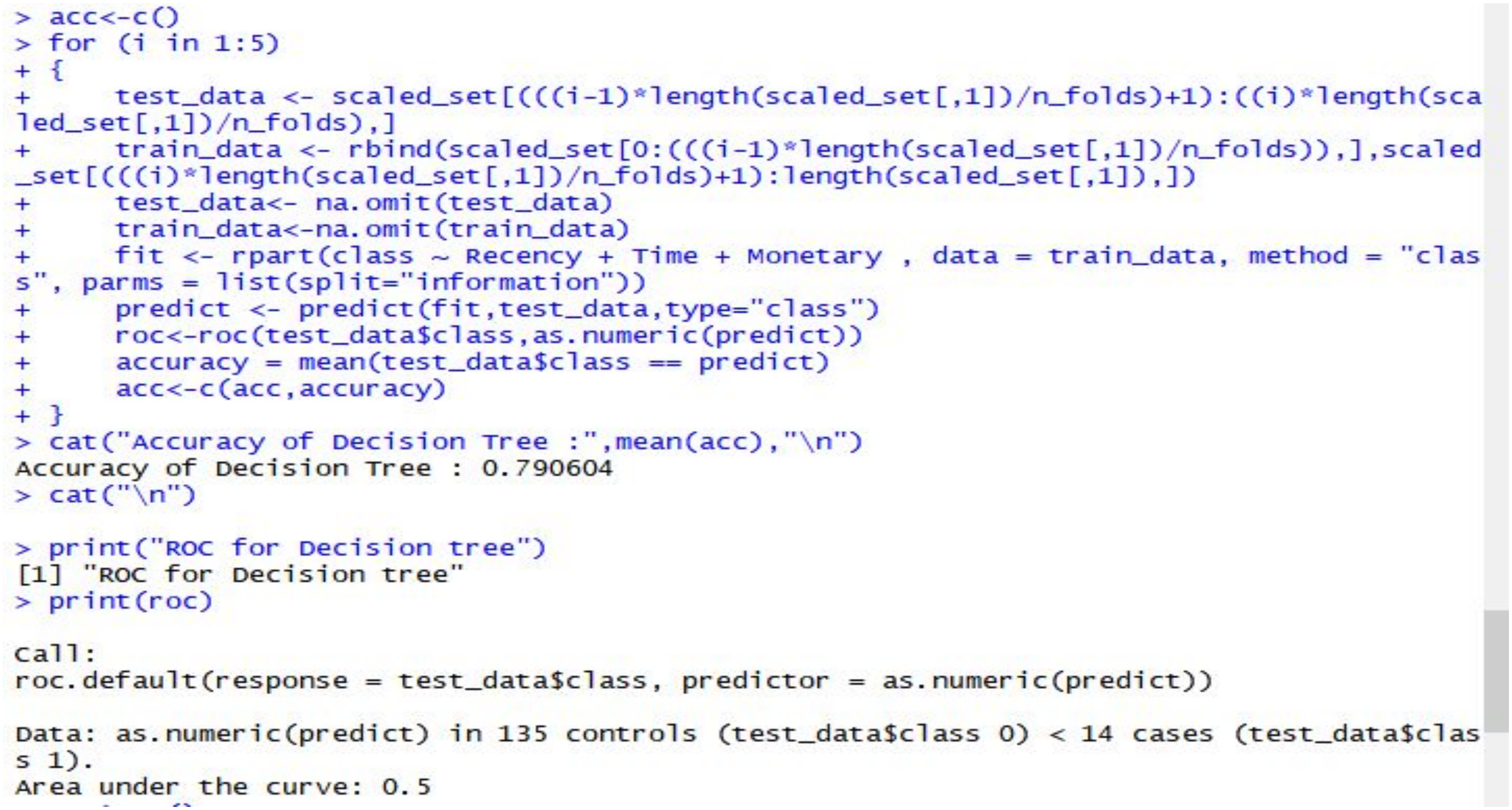
❑ Decision Tree
❑ Perceptron
❑ Neural Net
❑ Deep Learning
❑ SVM
❑ Naïve Bayes
❑ Logistic Regression
❑ K-nearest Neighbors
❑ Bagging

Continue

- ❑Random Forest
- ❑AdaBoost

Decision Tree

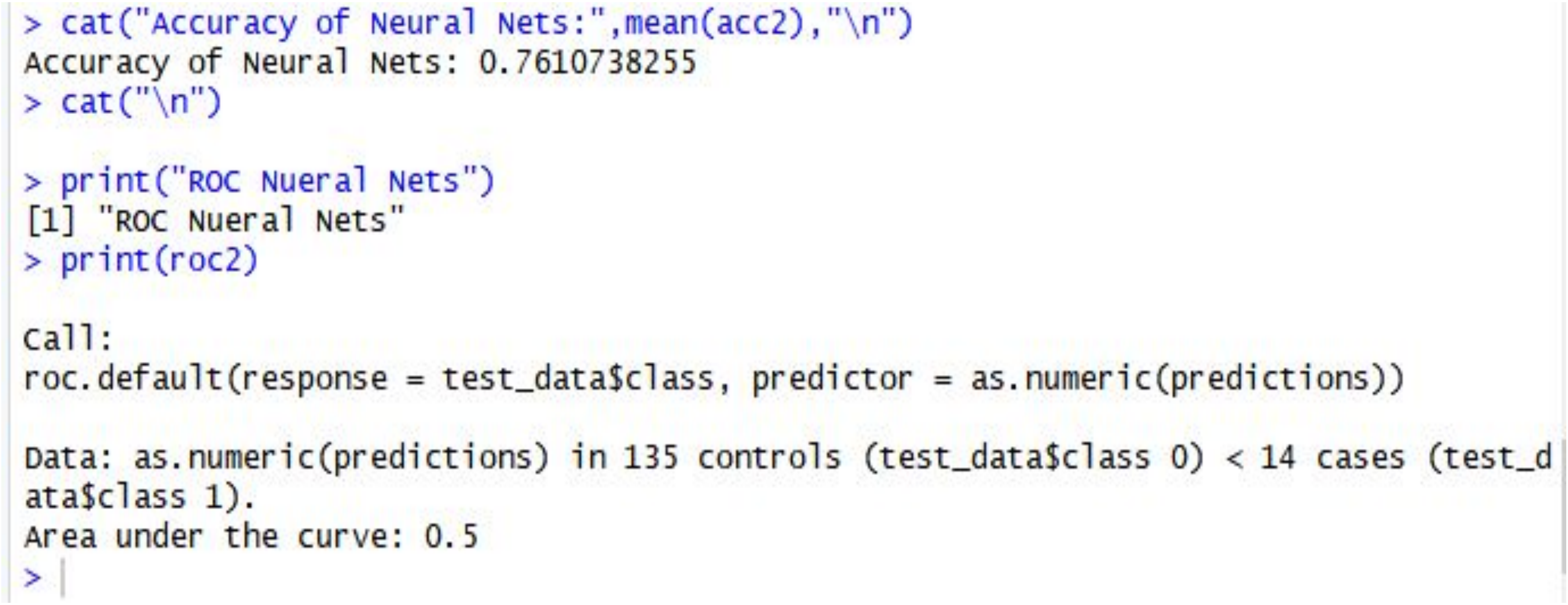
Decision trees are used extensively in machine learning because they are easy to use, easy to interpret, and easy to operationalize.
Accuracy and ROC value when Decision Tree algorithm is applied:



```
> acc<-c()
> for (i in 1:5)
+ {
+     test_data <- scaled_set[(((i-1)*length(scaled_set[,1])/n_folds)+1):((i)*length(sca
led_set[,1])/n_folds),]
+     train_data <- rbind(scaled_set[0:(((i-1)*length(scaled_set[,1])/n_folds)),],scaled
_set[(((i)*length(scaled_set[,1])/n_folds)+1):length(scaled_set[,1]),])
+     test_data<- na.omit(test_data)
+     train_data<-na.omit(train_data)
+     fit <- rpart(class ~ Recency + Time + Monetary , data = train_data, method = "clas
s", parms = list(split="information"))
+     predict <- predict(fit,test_data,type="class")
+     roc<-roc(test_data$class,as.numeric(predict))
+     accuracy = mean(test_data$class == predict)
+     acc<-c(acc,accuracy)
+ }
> cat("Accuracy of Decision Tree :",mean(acc),"\n")
Accuracy of Decision Tree : 0.790604
> cat("\n")

> print("ROC for Decision tree")
[1] "ROC for Decision tree"
> print(roc)

Call:
roc.default(response = test_data$class, predictor = as.numeric(predict))

Data: as.numeric(predict) in 135 controls (test_data$class 0) < 14 cases (test_data$clas
s 1).
Area under the curve: 0.5
```

Neural Networks

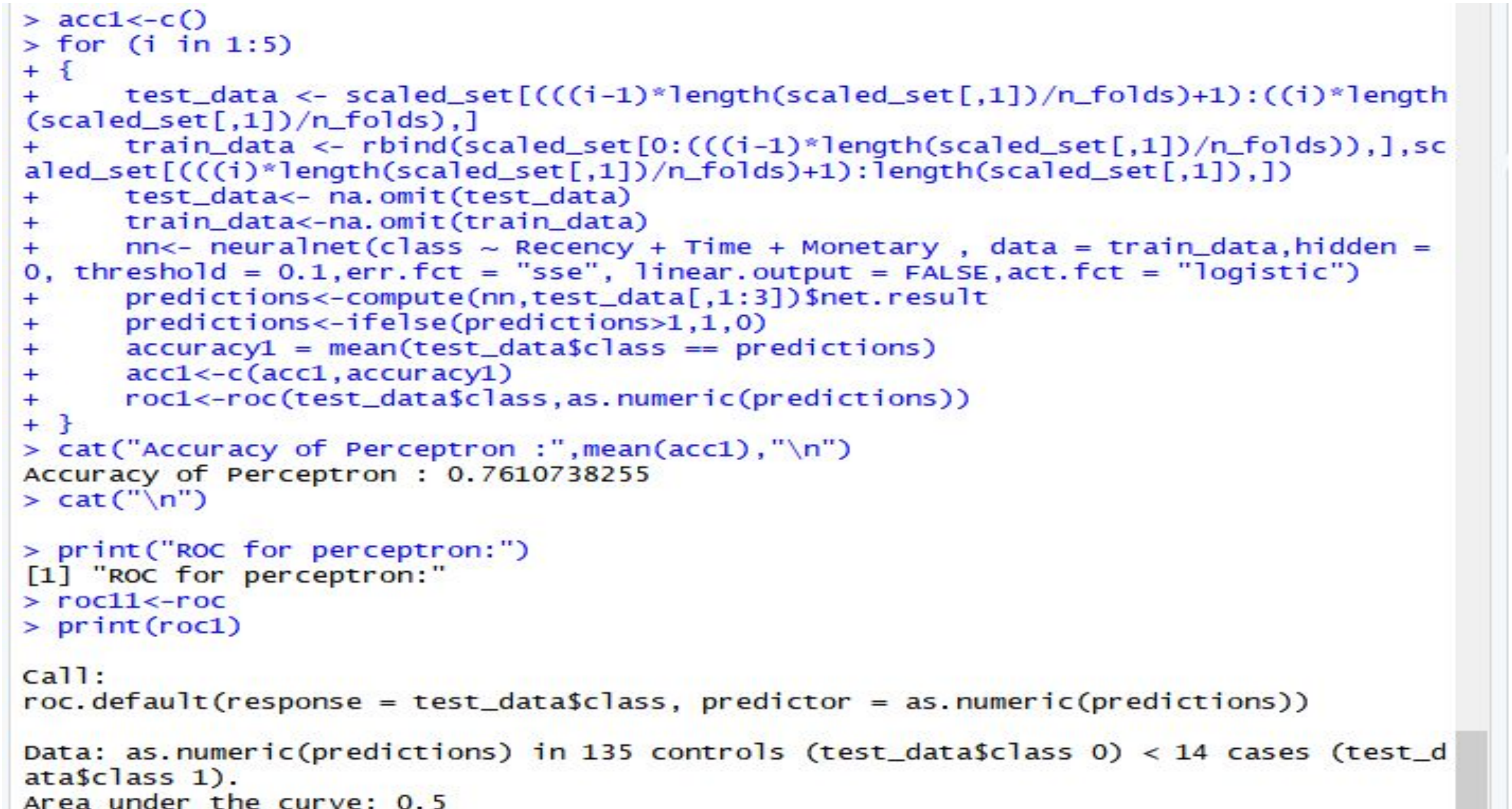
Neural networks represent a brain metaphor for information processing. These models are an exact replica of how the brain actually functions. Neural networks have been shown to be very promising systems in many forecasting applications and business classification applications due to their ability to “learn” from the data , their nonparametric nature(i.e., no rigid assumptions), and their ability to generalize.



```
> cat("Accuracy of Neural Nets:",mean(acc2),"\n")
Accuracy of Neural Nets: 0.7610738255
> cat("\n")

> print("ROC Nueral Nets")
[1] "ROC Nueral Nets"
> print(roc2)

Call:
roc.default(response = test_data$class, predictor = as.numeric(predictions))

Data: as.numeric(predictions) in 135 controls (test_data$class 0) < 14 cases (test_d
ata$class 1).
Area under the curve: 0.5
> |
```

Perceptron

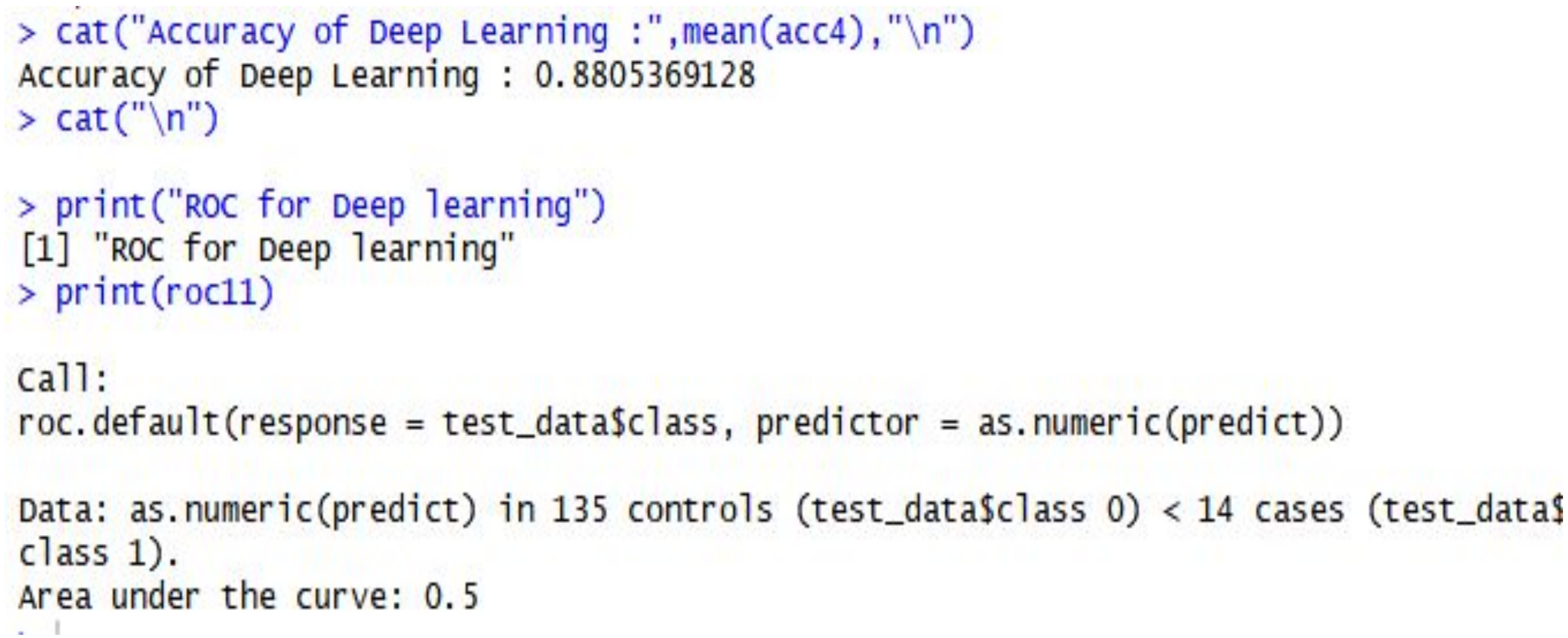
Perceptron is a single layer neural network. *Perceptron is usually used to classify the data into two parts. Therefore, it is also known as a Linear Binary Classifier.*



```
> acc1<-c()
> for (i in 1:5)
+ {
+     test_data <- scaled_set[(((i-1)*length(scaled_set[,1])/n_folds)+1):((i)*length
(scaled_set[,1])/n_folds),]
+     train_data <- rbind(scaled_set[0:(((i-1)*length(scaled_set[,1])/n_folds)),],sc
aled_set[(((i)*length(scaled_set[,1])/n_folds)+1):length(scaled_set[,1]),])
+     test_data<- na.omit(test_data)
+     train_data<-na.omit(train_data)
+     nn<- neuralnet(class ~ Recency + Time + Monetary , data = train_data,hidden =
0, threshold = 0.1,err.fct = "sse", linear.output = FALSE,act.fct = "logistic")
+     predictions<-compute(nn,test_data[,1:3])$net.result
+     predictions<-ifelse(predictions>1,1,0)
+     accuracy1 = mean(test_data$class == predictions)
+     acc1<-c(acc1,accuracy1)
+     roc1<-roc(test_data$class,as.numeric(predictions))
+ }
> cat("Accuracy of Perceptron :",mean(acc1),"\n")
Accuracy of Perceptron : 0.7610738255
> cat("\n")

> print("ROC for perceptron:")
[1] "ROC for perceptron:"
> roc11<-roc
> print(roc1)

Call:
roc.default(response = test_data$class, predictor = as.numeric(predictions))

Data: as.numeric(predictions) in 135 controls (test_data$class 0) < 14 cases (test_d
ata$class 1).
Area under the curve: 0.5
```

Deep Learning

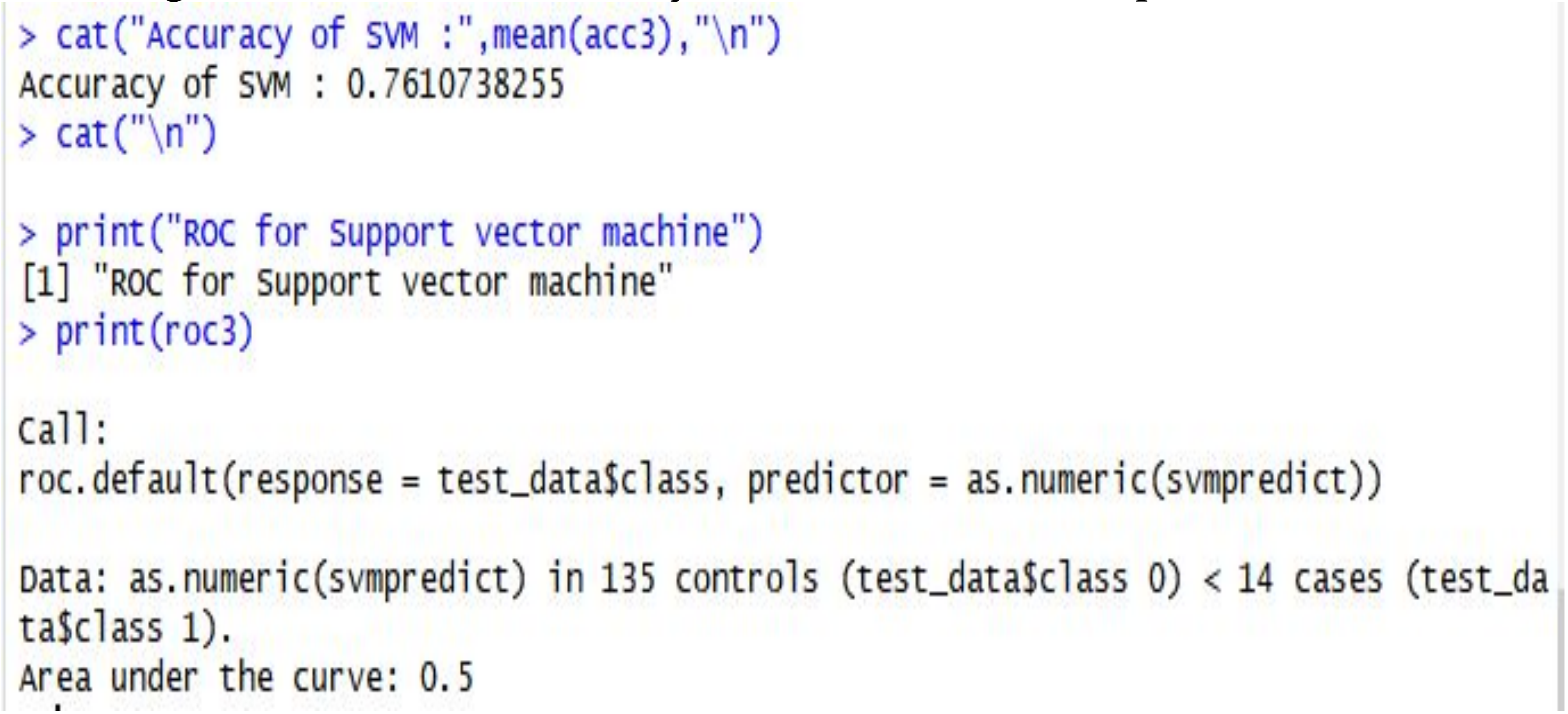
Deep Learning is basically a neural network implementation with large number of layers.



```
> cat("Accuracy of Deep Learning :",mean(acc4),"\n")
Accuracy of Deep Learning : 0.8805369128
> cat("\n")

> print("ROC for Deep learning")
[1] "ROC for Deep learning"
> print(roc11)

Call:
roc.default(response = test_data$class, predictor = as.numeric(predict))

Data: as.numeric(predict) in 135 controls (test_data$class 0) < 14 cases (test_data$
class 1).
Area under the curve: 0.5
```

SVM

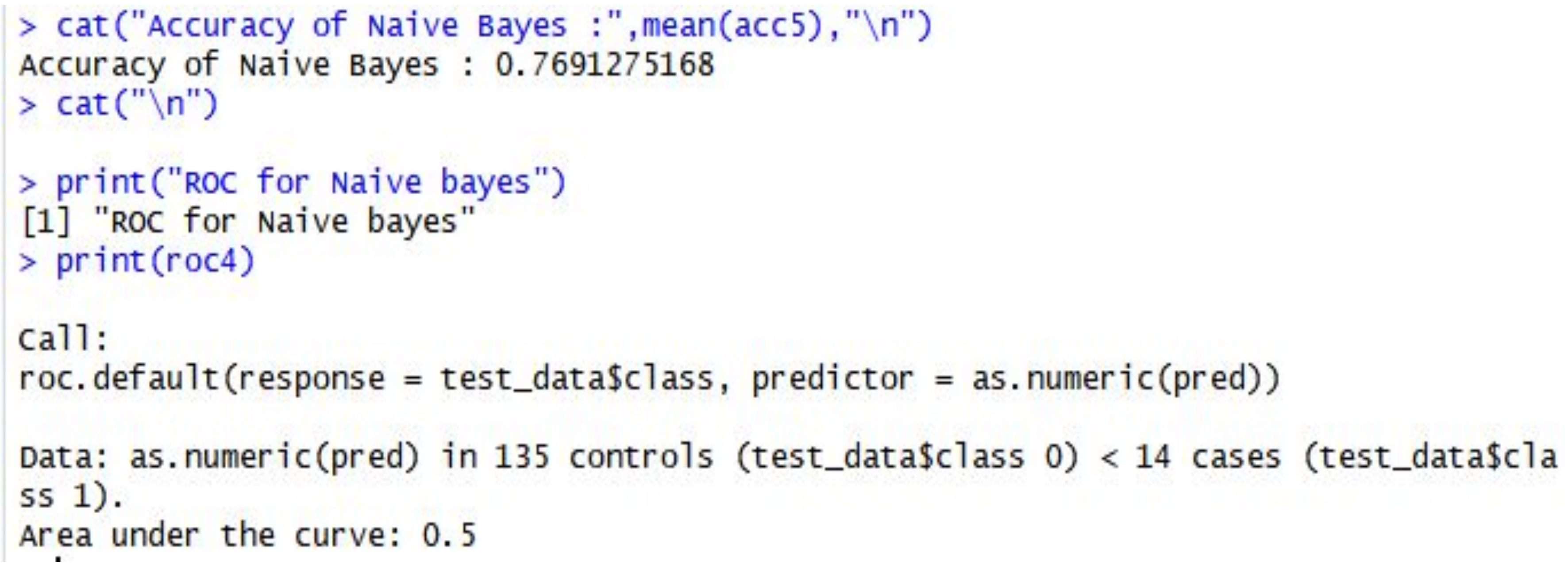
“Support Vector Machine” (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems.



```
> cat("Accuracy of SVM :",mean(acc3),"\n")
Accuracy of SVM : 0.7610738255
> cat("\n")

> print("ROC for Support vector machine")
[1] "ROC for Support vector machine"
> print(roc3)

Call:
roc.default(response = test_data$class, predictor = as.numeric(svmpredict))

Data: as.numeric(svmpredict) in 135 controls (test_data$class 0) < 14 cases (test_da
ta$class 1).
Area under the curve: 0.5
```

Naïve Bayes

- Statistical method for classification
- Supervised Learning Model
- Assumes an underlying probabilistic model, the Bayes Theorem
- Can solve problems involving both categorical and continuous values attributes



```
> cat("Accuracy of Naive Bayes :",mean(acc5),"\n")
Accuracy of Naive Bayes : 0.7691275168
> cat("\n")

> print("ROC for Naive bayes")
[1] "ROC for Naive bayes"
> print(roc4)

Call:
roc.default(response = test_data$class, predictor = as.numeric(pred))

Data: as.numeric(pred) in 135 controls (test_data$class 0) < 14 cases (test_data$cla
ss 1).
Area under the curve: 0.5
```

Logistic Regression

Logistic regression is a statistical method for analyzing a dataset in which there are one or more independent variables that determine an outcome. The outcome is measured with a dichotomous variable (in which there are only two possible outcomes).

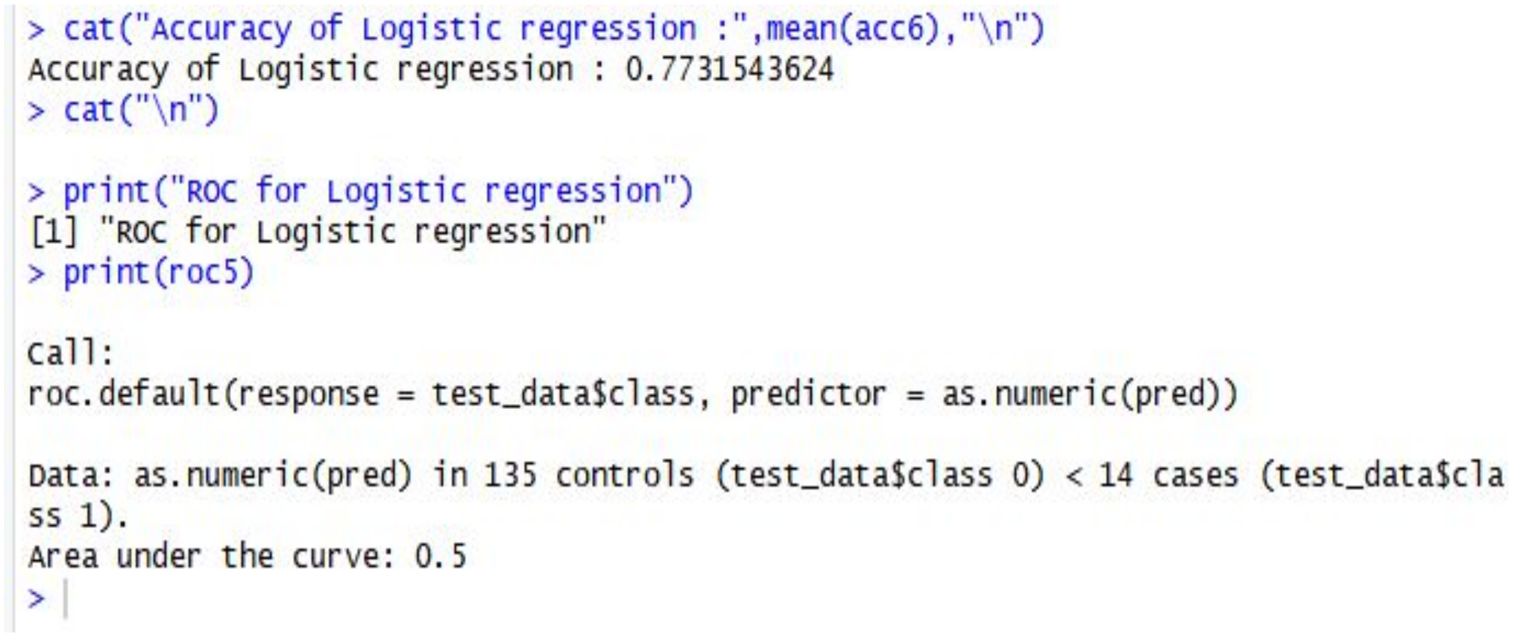
$$logit(p) = b_0 + b_1X_1 + b_2X_2 + b_3X_3 + \ldots + b_kX_k$$

where p is the probability of presence of the characteristic of interest. The logit transformation is defined as the logged odds

$$odds = \frac{p}{1-p} = \frac{probability\ of\ presence\ of\ characteristic}{probability\ of\ absence\ of\ characteristic}$$

$$logit(p) = \ln\left(\frac{p}{1-p}\right)$$

Logistic Regression

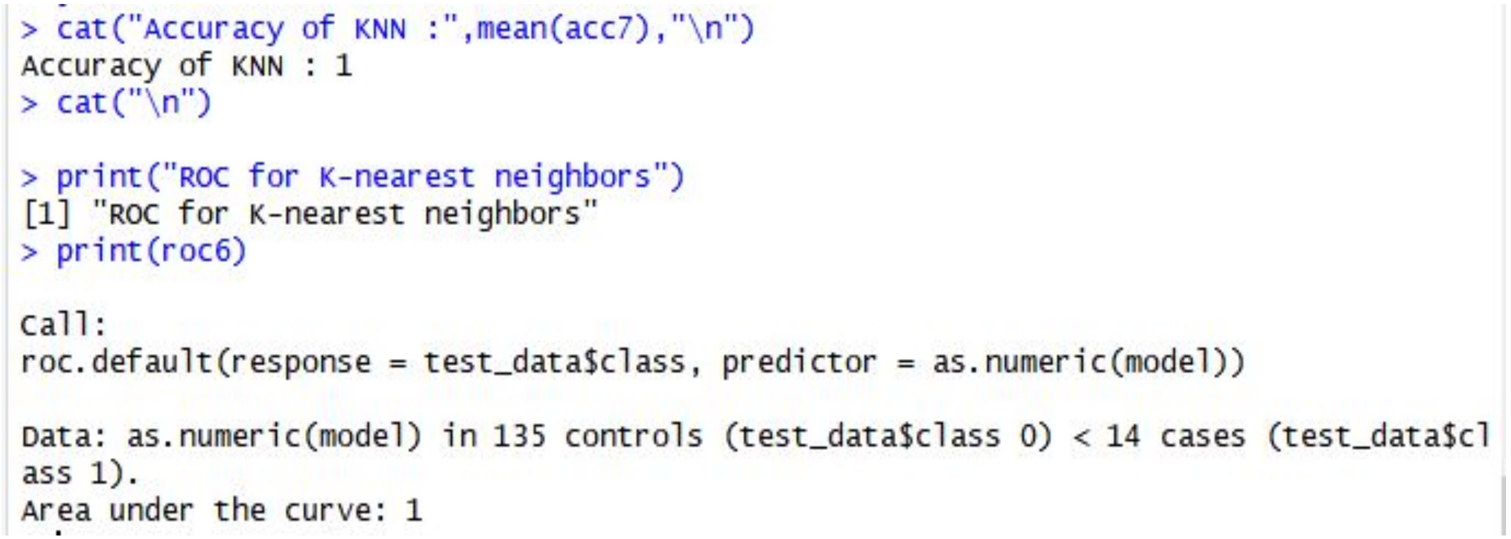


```
> cat("Accuracy of Logistic regression :",mean(acc6),"\n")
Accuracy of Logistic regression : 0.7731543624
> cat("\n")

> print("ROC for Logistic regression")
[1] "ROC for Logistic regression"
> print(roc5)

Call:
roc.default(response = test_data$class, predictor = as.numeric(pred))

Data: as.numeric(pred) in 135 controls (test_data$class 0) < 14 cases (test_data$cla
ss 1).
Area under the curve: 0.5
> 
```

ACCURACY – 77.3%

KNN

KNN is a non-parametric, lazy learning algorithm. Its purpose is to use a database in which the data points are separated into several classes to predict the classification of a new sample point.

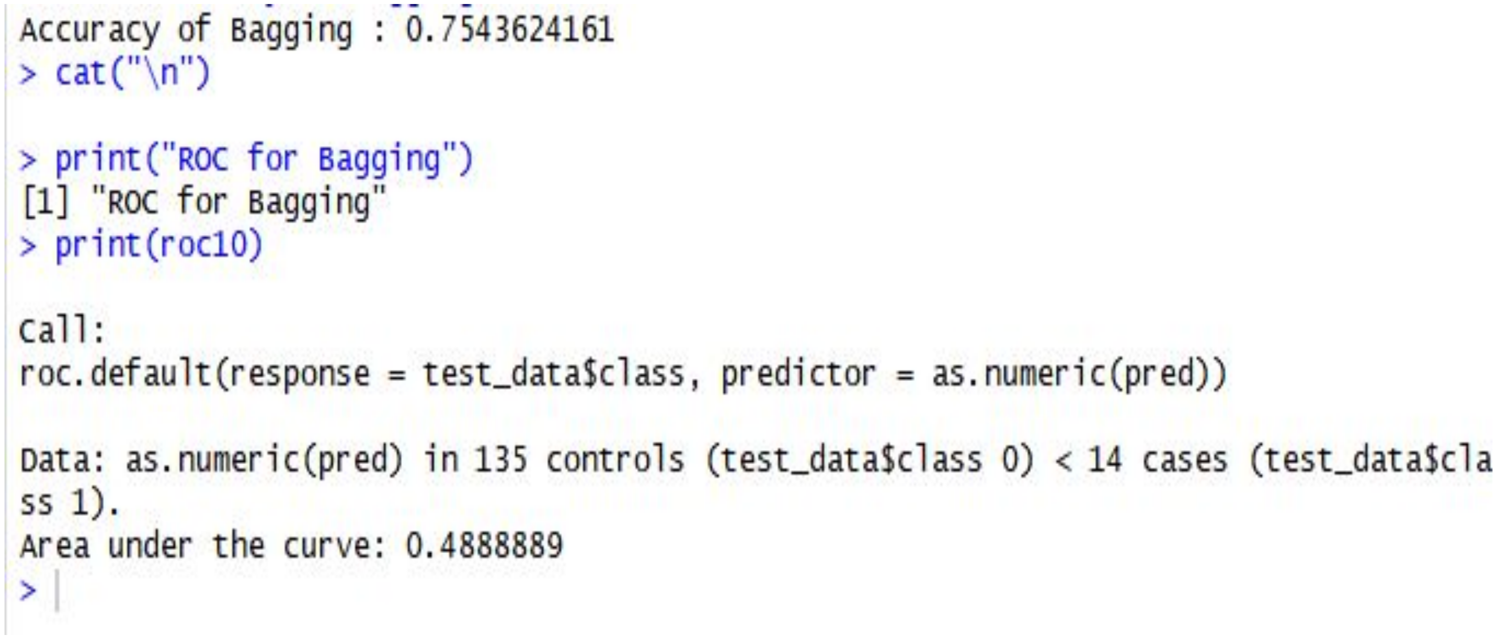


```
> cat("Accuracy of KNN :",mean(acc7),"\n")
Accuracy of KNN : 1
> cat("\n")

> print("ROC for K-nearest neighbors")
[1] "ROC for K-nearest neighbors"
> print(roc6)

Call:
roc.default(response = test_data$class, predictor = as.numeric(model))

Data: as.numeric(model) in 135 controls (test_data$class 0) < 14 cases (test_data$cl
ass 1).
Area under the curve: 1
```

ACCURACY – 100%

Bagging

Given a standard training set D of size n, bagging generates m new training sets D_{i}, each of size n′, by sampling from D uniformly and with replacement. By sampling with replacement, some observations may be repeated in each D_{i}. If n′=n, then for large n the set D_{i} is expected to have the fraction (1 - 1/e) (≈63.2%) of the unique examples of D, the rest being duplicates.



```
Accuracy of Bagging : 0.7543624161
> cat("\n")

> print("ROC for Bagging")
[1] "ROC for Bagging"
> print(roc10)

Call:
roc.default(response = test_data$class, predictor = as.numeric(pred))

Data: as.numeric(pred) in 135 controls (test_data$class 0) < 14 cases (test_data$cla
ss 1).
Area under the curve: 0.4888889
> |
```

ACCURACY – 75.4%

Random Forest

Random forest algorithm is a supervised classification algorithm. This algorithm creates the forest with a number of trees.

Random Forest pseudocode:
Randomly select “k” features from total “m” features.

Where k << m
Among the “k” features, calculate the node “d” using the best split point.

Split the node into daughter nodes using the best split.

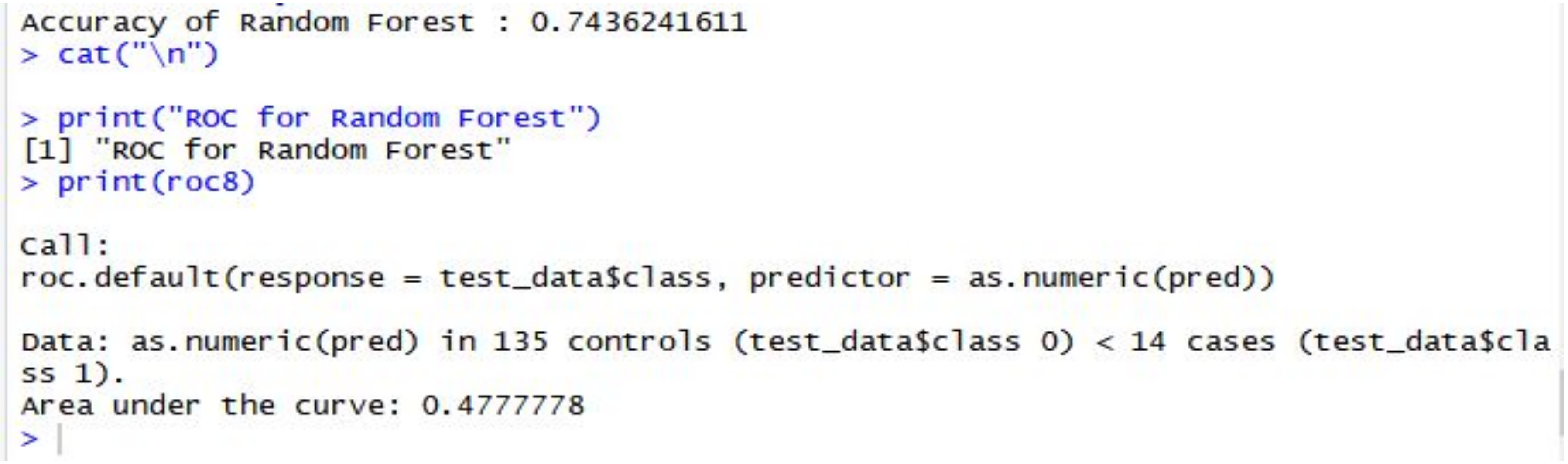
Repeat 1 to 3 steps until “l” number of nodes has been reached.

Build forest by repeating steps 1 to 4 for “n” number times to create “n” number of trees.

Random Forest

Random forest prediction pseudocode:

- Takes the test features and use the rules of each randomly created decision tree to predict the outcome and stores the predicted outcome (target)
- Calculate the votes for each predicted target.
- Consider the high voted predicted target as the final prediction from the random forest algorithm.

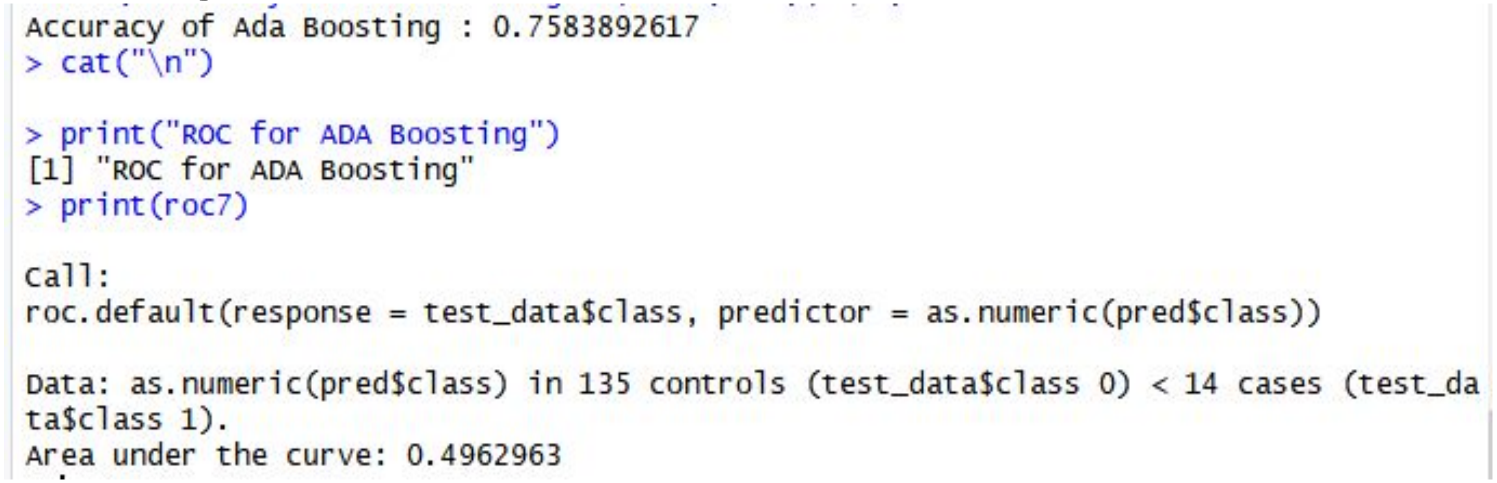


```
Accuracy of Random Forest : 0.7436241611
> cat("\n")

> print("ROC for Random Forest")
[1] "ROC for Random Forest"
> print(roc8)

Call:
roc.default(response = test_data$class, predictor = as.numeric(pred))

Data: as.numeric(pred) in 135 controls (test_data$class 0) < 14 cases (test_data$cla
ss 1).
Area under the curve: 0.4777778
> 
```

ACCURACY – 74.3%

Ada Boosting

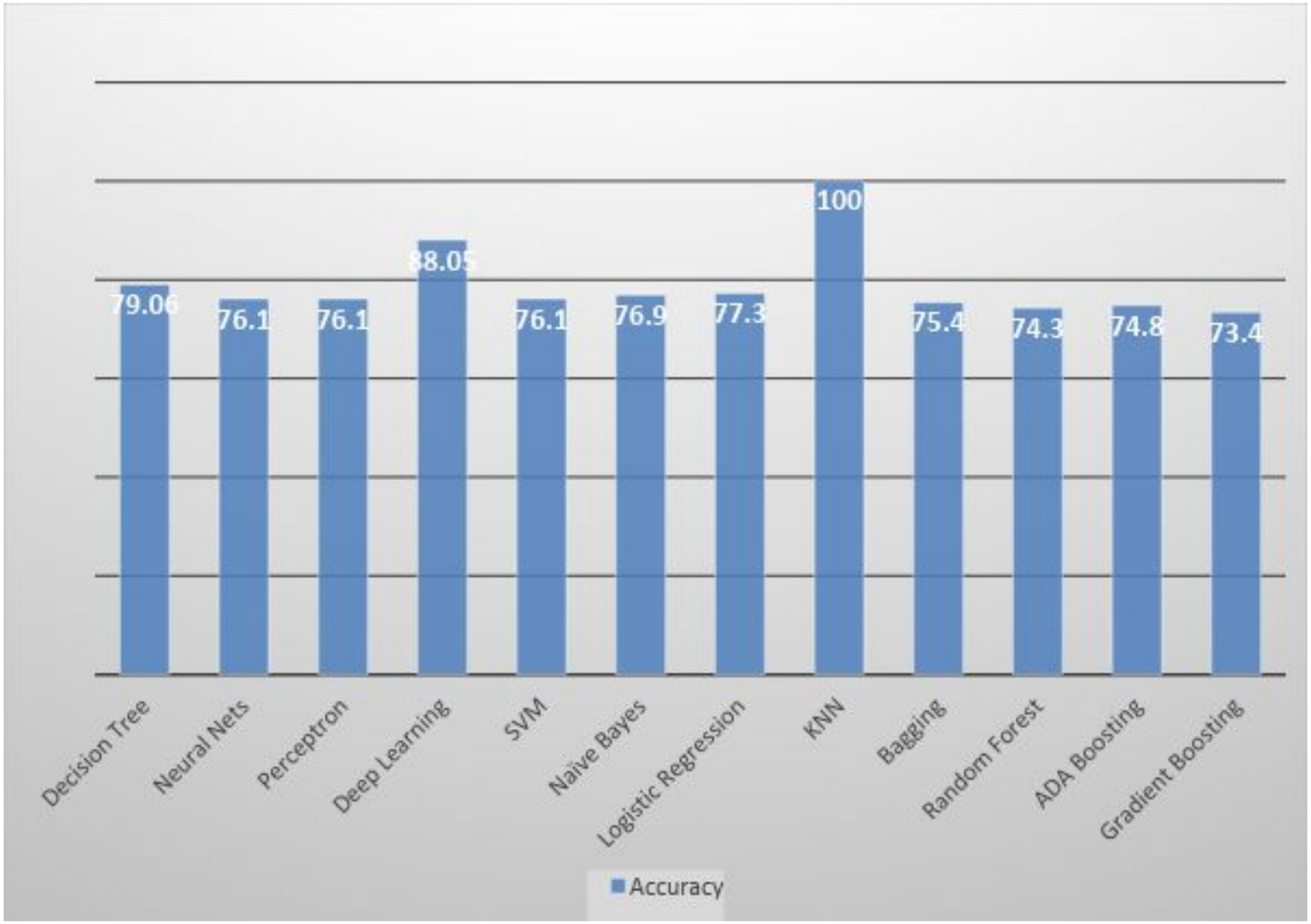
Ada boosting is nothing but Adaptive Boosting. AdaBoost is adaptive in the sense that subsequent weak learners are tweaked in favor of those instances misclassified by previous classifiers. AdaBoost is sensitive to noisy data and outliers. It retrains the algorithm iteratively by choosing the training set based on accuracy of previous training and the weight-age of each trained classifier at any iteration depends on the accuracy achieved.



```
Accuracy of Ada Boosting : 0.7583892617
> cat("\n")

> print("ROC for ADA Boosting")
[1] "ROC for ADA Boosting"
> print(roc7)

Call:
roc.default(response = test_data$class, predictor = as.numeric(pred$class))

Data: as.numeric(pred$class) in 135 controls (test_data$class 0) < 14 cases (test_da
ta$class 1).
Area under the curve: 0.4962963
```

ACCURACY –
75.8%

Comparison



79.06
76.1
76.1
88.05
76.1
76.9
77.3
100
75.4
74.3
74.8
73.4
Decision Tree
Neural Nets
Perceptron
Deep Learning
SVM
Naive Bayes
Logistic Regression
KNN
Bagging
Random Forest
ADA Boosting
Gradient Boosting
■ Accuracy

Future Scope

There is a huge scope of expansion to this project, as this data set has only 5 dependable attributes, but having more attributes like “Blood Group”, “Location” etc. could add a significant value. By this we can predict the future trends based out of location.

We also could map this donation trend with the need trend. By this we would be able to make sure that there is a balance in need and supply by transferring blood from one place to another is advance. Increasing the probability to Save Lives.

References

- https://medium.com/mlreview/gradient-boosting-from-scratch-1e317ae4587d
- https://medium.com/machine-learning-101/https-medium-com-savanpatel-chapter-6-adaboost-classifier-b945f330af06
- http://dataaspirant.com/2017/05/22/random-forest-algorithm-machine-learing/
- https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.html
- https://machinelearningmastery.com/bagging-and-random-forest-ensemble-algorithms-for-machine-learning/
- https://machinelearningmastery.com/k-nearest-neighbors-for-machine-learning/
- https://www.medcalc.org/manual/logistic_regression.php