Tecnológico Nacional de México
Instituto Tecnológico de Tijuana

Subdirección Académica
Departamento de Sistemas y Computación

Semestre:
Febrero – Junio 2021

Carrera:
Ingeniería en Tecnologías de la Información y Comunicaciones

Materia y serie:
Minería de datos BDD-1703TI9A

Unidad a evaluar: Unidad III

Nombre de la Tarea:
Práctica 4

Nombre del Alumno:
Hernández Negrete Juan Carlos 16212021
Sifuentes Martinez Manuel Javier 17212934

Nombre del docente:
José Christian Romero Hernández

Data visualization analysis in the logistic regression model

Change of directory

How It has been working with the previous practices, the getwd function is used to obtain the current directory and setwd to establish a new one.

```
getwd ()
setwd ("../Desktop/DataMining/MachineLearning/LogisticRegression")
getwd ()
```

Import and section dataset

Once in the established directory, the dataset to be used is loaded into R Studio's memory. The dataset comes with more columns than necessary to carry out the present practice, so they select only the necessary data, and it is done by selecting the dataset, indicating that it takes all the data and from column 3 to 5.

```
dataset < - read.csv ('Social_Network_Ads.csv')
dataset <- dataset [, 3:5]
```

Encoding the destination function as a factor

The next step is to encode the values found in the Purchased column as a factor, in this way the data as categorical to be able to graph.

```
dataset $ Purchased = factor (dataset $ Purchased, levels = c (0, 1))
```

Divide the dataset into training and test

sets Using set.seed and defining a number within the function, a sequence of random numbers is generated but being able to get to replicate the results. Then, thanks to the caTools library, sample.split is used to be able to divide the dataset into 2, a training set and a test set, defining through the .75 that there will be 300 data for the first set and 100 for the second. respectively.

```
library(caTools)
set.seed (123)
split <- sample.split (dataset $ Purchased, SplitRatio = 0.75)
training_set <- subset (dataset, split == TRUE)
test_set <- subset (dataset, split == FALSE)
```

Scale of characteristics

The scale function is used on column 3 to normalize the data, and thus be able to improve the predictive precision of the algorithm.

```
training_set [-3] = scale (training_set [-3])
test_set [-3] = scale (test_set [-3])
```

Adaptation of the logistic regression to the training set

The random forest library is loaded into memory to access the method of the same name. and the independent variables "x" and dependent "y" are defined, which would be what the algorithm is going to train with and what it is going to test with. The last parameter refers to the number of trees you want, it can be changed depending on the need and also on the processing power and memory that you have.

```
library(randomForest)
set.seed (123)
classifier = randomForest (x = training_set [-3],
                           y = training_set $ Purchased,
                           ntree =10)
```

Predicting the results of the test set

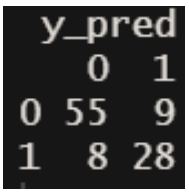
The next line executes the classifier on the test set and it is specified that the prediction is made on column 3 of the mentioned set.

```
y_pred = predict (classifier, newdata = test_set [-3])
y_pred
```

Making a confusion matrix We

proceed to make a confusion matrix. This is to evaluate how accurate the prediction made in the previous step was. Adding their false positives and false negatives is that this percentage of error is obtained. In this case, a 17% error was obtained. This can vary depending on the number of trees specified above.

```
cm = table (test_set [, 3], y_pred)
cm
```



	y_pred	
	0	1
0	55	9
1	8	28

Visualization of the results of the training set

The structure used throughout this unit for the construction of graphs for the visualization of results is maintained. Matching the dataset to use to the set variable. After defining the green and red area using the parameter by, setting it to 0.01, this means that it will be classified as green or red. The +1 and -1 indicate the separation in the space around the edges. The col (names) function is used to define the names of the “x” and “y” axes.

The previously made prediction was saved in the classifier variable and is used to graph the results obtained. All of the above is graphed thanks to the plot function, defining the axes with the variables X1 and X2 and changing the main title and the axes with main and xlab ylab. Then, through the contour function, a boundary is generated between the green and the red area. Finally, there is the points function, in which the data stored in the variable ypred is reviewed and ifelse is used to color the points. Running everything you can see a graph with a dispersion of green and red dots with two background colors of the same color.

```
library(ElemStatLearn)
set = training_set
X1 = seq (min (set [, 1]) - 1, max (set [, 1]) + 1, by = 0.01)
X2 = seq (min (set [, 2]) - 1, max (set [, 2]) + 1, by = 0.01)
grid_set = expand.grid (X1, X2)
colnames (grid_set) = c ('Age', 'EstimatedSalary')
y_grid = predict (classifier, grid_set)
plot (set [, -3],
     main = 'Random Forest Classification (Training set)',
     xlab = 'Age', ylab = 'Estimated Salary',
     xlim = range (X1), ylim = range (X2))
contour ( X1, X2, matrix (as.numeric (y_grid), length (X1), length
(X2)), add = TRUE)
points (grid_set, pch = '.', Col = ifelse (y_grid == 1, 'springgreen3',
'tomato'))
points (set, pch = 21, bg = ifelse (set [, 3] == 1, 'green4', 'red3'))
```

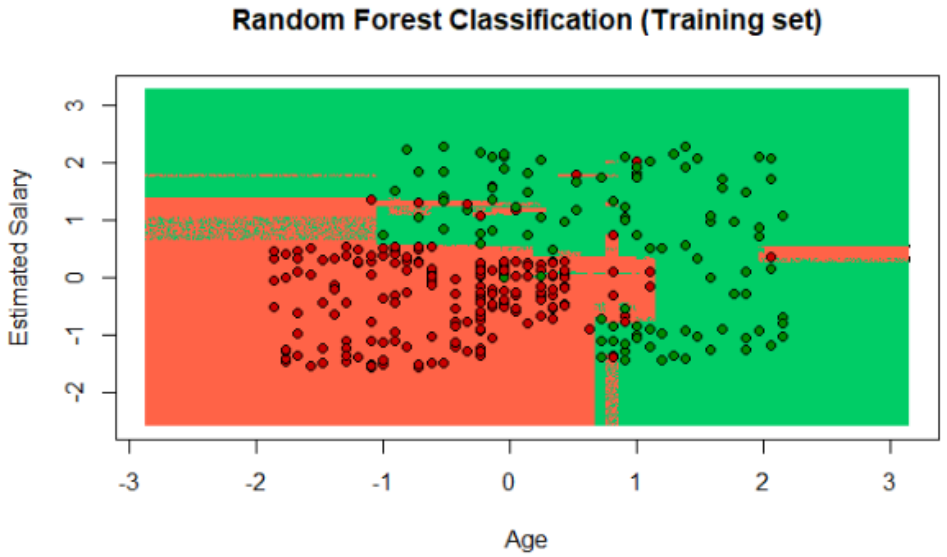
Visualization of the test set results

Constructing the graph with Test data follows the same steps as above, just changing the data source.

```
library(ElemStatLearn)
set = test_set
X1 = seq (min (set [, 1]) - 1, max (set [, 1]) + 1, by = 0.01)
X2 = seq (min (set [, 2]) - 1, max (set [, 2]) + 1, by = 0.01)
grid_set = expand.grid (X1, X2)
colnames (grid_set) = c ('Age', 'EstimatedSalary')
y_grid = predict (classifier, grid_set)
plot (set [, -3], main = 'Random Forest Classification (Test set)',
     xlab = 'Age', ylab = 'Estimated Salary',
     xlim = range (X1), ylim = range (X2))
contour ( X1, X2, matrix (as.numeric (y_grid), length (X1), length
(X2)), add = TRUE)
points (grid_set, pch = '.', Col = ifelse (y_grid == 1, 'springgreen3',
'tomato'))
points (set, pch = 21, bg = ifelse (set [, 3] == 1, 'green4', 'red3'))
```

Conclusion of the graph of the training set

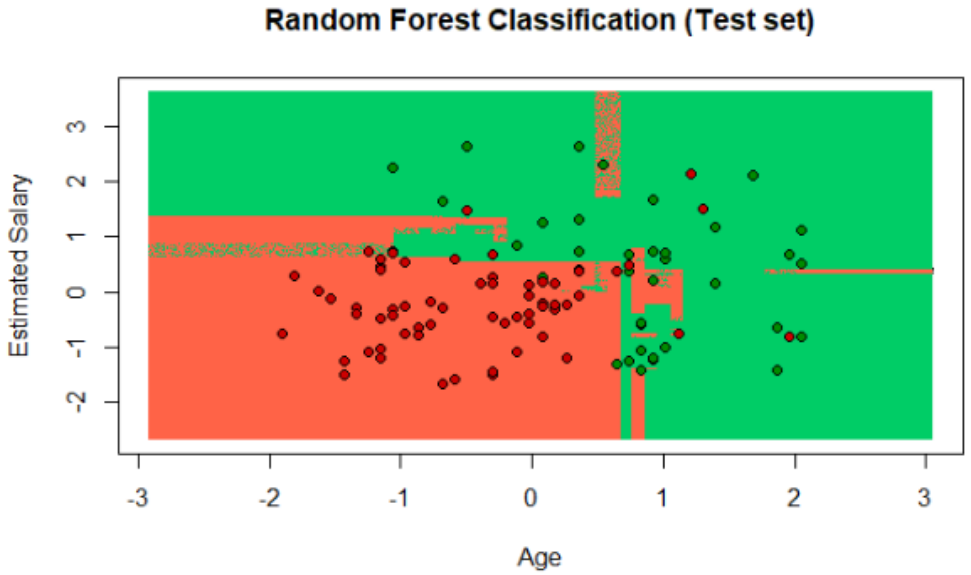
It can be seen that there are marked areas of red and others of green, the percentage in which these appear in the graph varies depending on the accuracy of the algorithm, and is directly related to the number of trees to be made. Entering the interpretation of the results, the green background represents customers who bought SUV, while the red area is those who did not, this as long as the points correspond with their background color, otherwise they are erroneous predictions made by the algorithm.



Graph 1. Training set

Conclusion of the graph of the test set

It can be seen that despite being less data, the red zone did not decrease, since it does not depend on the number of data, but on the accuracy of the algorithm. Regarding its interpretation, it is the same as the previous graph, the points that coincide with their color are correct predictions, unlike those that do not coincide, with the red area being the customers who did not buy an SUV and the green area those who did.



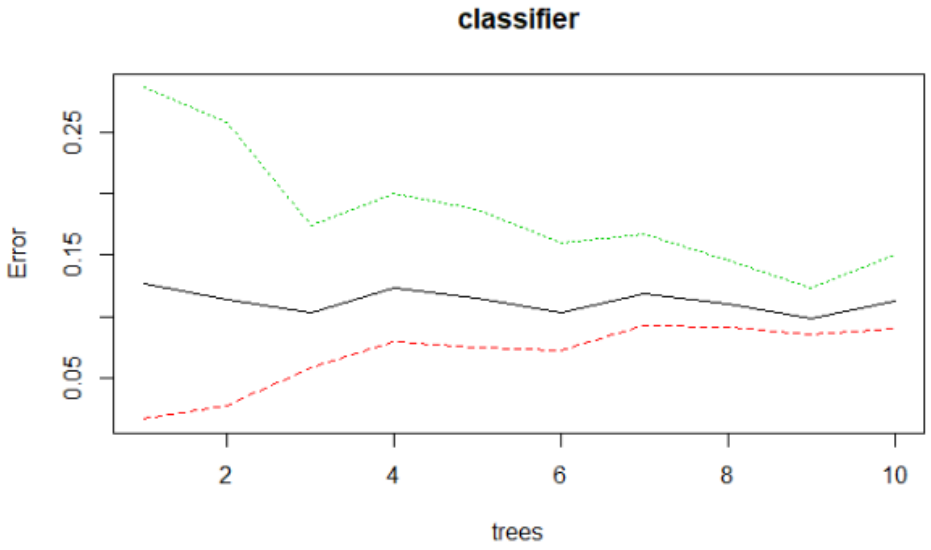
Graph 2. Test set

Choosing the number of trees

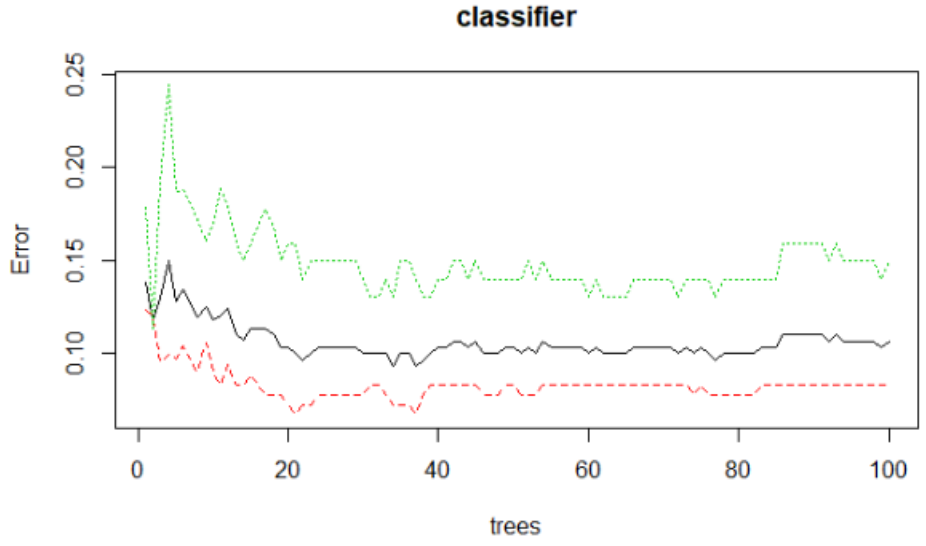
Here only the obtained values obtained from the randomForest function and stored in the classifier variable are graphed, being able to observe the error percentage.

```
plot (classifier)
```

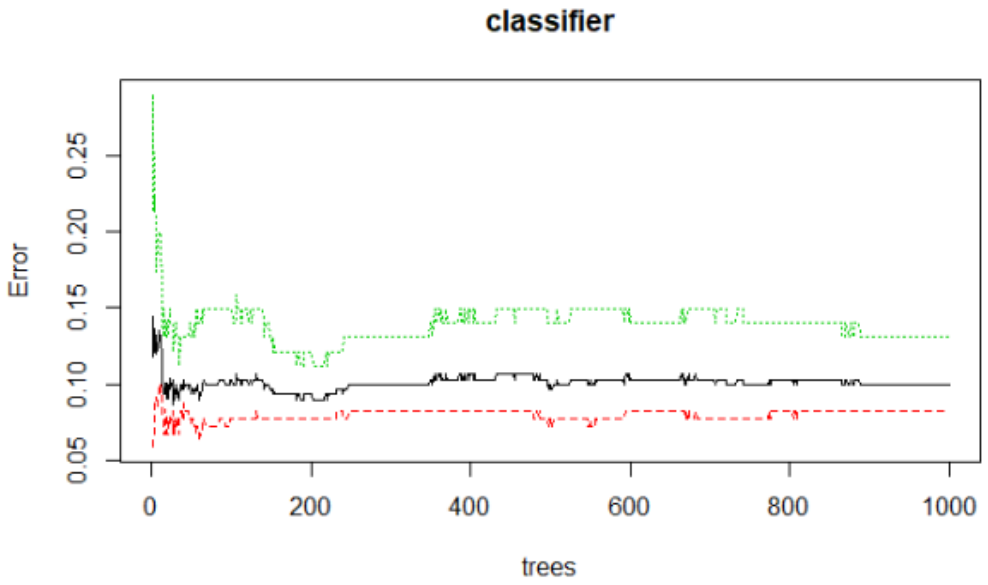
The plot will vary depending on the percentage of error and the level of accuracy of the algorithm. As more trees are generated, the graph lines have more points on which to generate their references with respect to the x-axis, yielding more data that results in more exact interpretations. In the same way, the percentage of error varies since while there are more cases, the error adjusts to this number of trees, but there comes a point where the number of trees exceeds the accuracy that can be obtained, and they only give a straight line , so it is good to define the percentage of error that does not influence the study.



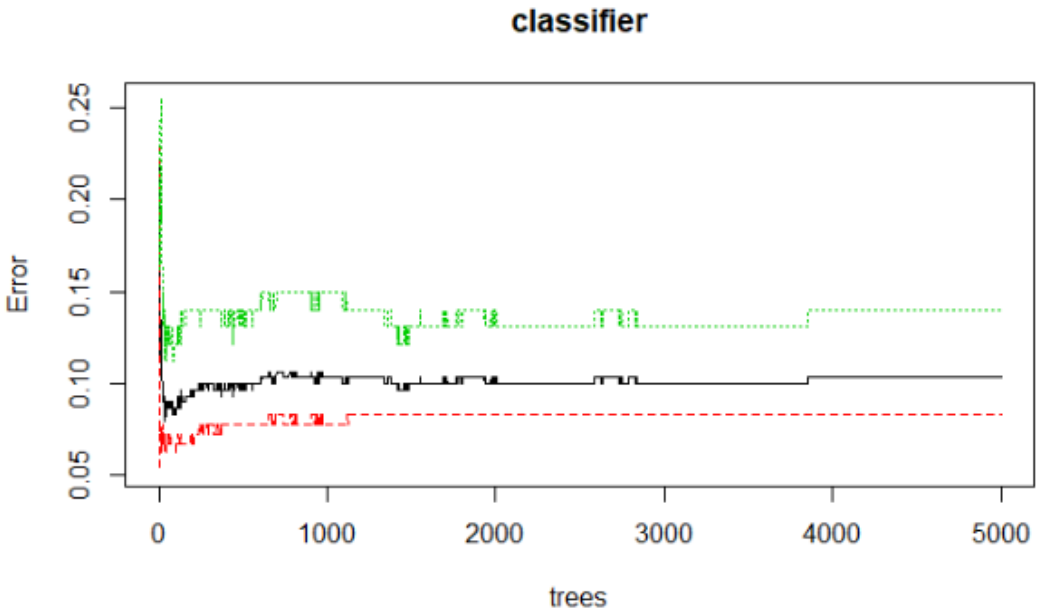
Graph 3. 10 trees



Graph 4. 100 trees



Graph 5. 1000 trees



Graph 6. 5000 trees