



TECNOLÓGICO
NACIONAL DE MÉXICO



SEP
SECRETARÍA DE
EDUCACIÓN PÚBLICA

**National Technology of Mexico
Technological Institute of Tijuana**

ACADEMIC SUBDIRECTION
Systems and Computing Department

SEMESTER
February – June 2021

CAREER
Information and Communication Technologies Engineer

SUBJECT AND KEY:
Data Mining BDD-1703TI9A

STUDENT'S NAME AND REGISTRATION:
Velázquez Farrera César Alejandro 17212937

NAME OF THE JOB:
Evaluative Practice

UNIT TO BE EVALUATED
Unit III

TEACHER'S NAME:
Mc. José Christian Romero Hernández

Evaluative Practice 3

Instructions

Implement the Naive Bayes classification model with the Social_Network_Ads.csv data set and using the e1071 library with the naiveBayes () function.

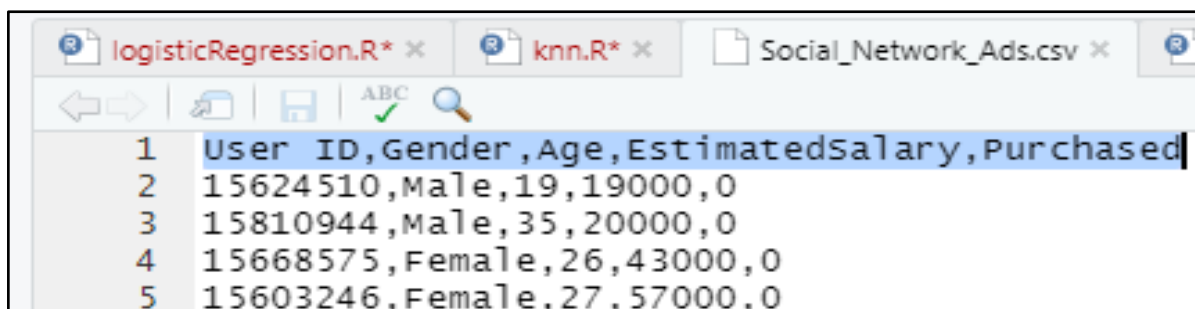
Once the classifier is obtained, do the corresponding data visualization analysis.

At the end of the development, explain in detail what the Naive Bayes classification model consists of and also the detailed explanation corresponding to data visualization.

Developing

We look for our data file on the computer with the help of the file explorer, if we are working in the folder where the file is located, we simply load it with the name of the file.

```
dataset = read.csv(file.choose())  
dataset = read.csv('Social_Network_Ads.csv')
```



	User ID,Gender,Age,EstimatedSalary,Purchased
1	15624510,Male,19,19000,0
2	15810944,Male,35,20000,0
3	15668575,Female,26,43000,0
4	15603246,Female,27,57000,0

Since the data file has columns that will not be useful, we only select the columns that we want to work with.

```
dataset = dataset[,3:5]
```

\$ Age	: int	19 35 26 27 19 27 27 32 25 35 ...
\$ EstimatedSalary:	int	19000 20000 43000 57000 76000 58000 84000 150000 330...
\$ Purchased	: int	0 0 0 0 0 0 0 1 0 0 ...

We load the library "caTools" that has several functions for statistics.

With the function "set.seed (n)" we sow a randomness seed where "n" is the starting point used in the generation of the sequence of random numbers.

```
library(caTools)
set.seed(123)
```

Next, we divide the data in the "Purchased" column, with a ratio of 0.75, that is, 75% of the data will be taken as true and the remaining 25% as false.

Later we take the values of the division of the data that were true and we use them for training, the rest of the data was labeled as false will be used to test the effectiveness of the trained model with the data labeled as true.

```
split = sample.split(dataset$Purchased, SplitRatio = 0.75)
training_set = subset(dataset, split == TRUE)
test_set = subset(dataset, split == FALSE)
```

For training and testing it is necessary to normalize the data, so we use the "scale" function. The characteristic scale is a method used to normalize the range of independent variables or characteristics of the data. In data processing, it is also known as data normalization and is usually done during the data pre-processing step.

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

We install and load the library ("e1071"), which contains statistical functions, among them "naiveBayes"

```
install.packages('e1071')
library(e1071)
```

We adopt the training data set, that is, the classification with the "naiveBayes" method which is used to fit the model. A curious fact is that it uses a kernel.

```
classifier = naiveBayes(x = training_set[-3], y = training_set$Purchased)
```

Obtaining the probability of predicting correct classifications with the test data. We print the prediction.

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

We save the resulting data in a table with the test data and the corresponding predictions, that is, we create a confusion matrix. Then we print it.

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

To continue we will need a special package that contains the "ElemStatLearn" library, since the package is obsolete for new versions of R and does not come by default in the RStudio tools, it is necessary to download and install it manually. The file will be found in the following link: <https://cran.r-project.org/src/contrib/Archive/ElemStatLearn/>, the installation will be done with the following command, substituting the appropriate file address.

```
install.packages("~/home/daniel-  
camacho/Descargas/ElemStatLearn_2015.6.26.tar.gz", repos=NULL,  
type="source")
```

1. Once the package is installed, we load the "ElemStatLearn" library.
2. With the command "set" we take the training data, then we create two variables "x1", "x2" where we use the function "seq" (sequence), we will define a minimum and maximum range, we will add a positive integer and we will multiply it by the length of "0.01"
3. Next we define a grid that we expand using the dimensions defined in "x1" and "x2".
4. We define the columns of the grid using a vector, which will be the year and estimated salary.

5. Next we test the logistic regression model but with the grid that contains the columns that we define.
6. We graph the data with a maximum of -1, we put a title to the graph that is inside (grid), we name the axes, we assign the limits for the axes keeping the data within the dimensions of the grid.
7. We place an outline and add a range of numbers conditional on the minimum and maximum numerical range of the grid.
8. We add points to the graph comparing the data of the year and estimated salaries with the predictions of the logistic regression model, where if they are different, a red color will be assigned and if they are true a green one, in the case of errors they will be placed red points within the green area that has been defined as true sets and, on the contrary, a green point in the area defined as false.

We import the "ElemStatLearn" library which contains statistical functions

```
library(ElemStatLearn)
```

We declare set as the test data set

```
set = training_set
```

This section creates the red and green background region. So every 0.01 is interpreted as 0 or 1 and is green or red. -1 and +1 give us the space around the edges so that the points are not obstructed.

```
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)
```

We assign values to each axis.

```
colnames(grid_set) = c('Age', 'EstimatedSalary')
```

We use the classifier to predict each of the bits of each pixel.

```
y_grid = predict(classifier, newdata = grid_set)
```

We graph the data, assign a title to each axis and to the graph.

```
plot(set[, -3], main = 'SVM Radial Kernel (Test set)',
      xlab = 'Age', ylab = 'Estimated Salary',
```

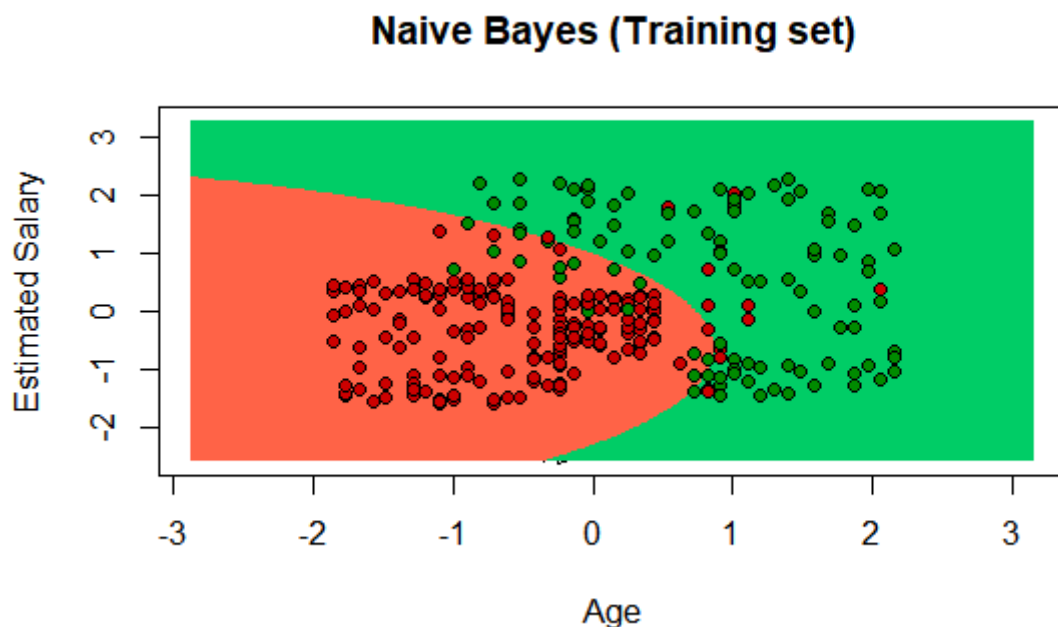
```
xlim = range(X1), ylim = range(X2))
```

The limits to the plotted values are created, the line between green and red is also created.

```
contour(X1, X2, matrix(as.numeric(y_grid), length(X1),  
length(X2)), add = TRUE)
```

Here we review all the prediction data and use “ifelse” to color the points, taking into account that the points are the real data, the background is the pixel-by-pixel determination of the prediction and the points are plotted on top of the background create the image.

```
points(grid_set, pch = '.', col = ifelse(y_grid == 1, 'springgreen3',  
'tomato'))  
points(set, pch = 21, bg = ifelse(set[, 3] == 1, 'green4', 'red3'))
```



In the same way as in the previous code, the evaluation and classification of the data is done, only this time, for the graph it will not be the training data but the test data, for which in the "set" command the data is assigned of the variable “test_set” obtained in

the division of the data corresponding to 25% of the entire dataset that contains the three initial columns.

We import the "ElemStatLearn" library which contains statistical functions

```
library(ElemStatLearn)
```

We declare set as the test data set

```
set = test_set
```

This section creates the red and green background region. So every 0.01 is interpreted as 0 or 1 and is green or red. -1 and +1 give us the space around the edges so that the points are not obstructed.

```
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)
```

We assign values to each axis.

```
colnames(grid_set) = c('Age', 'EstimatedSalary')
```

We use the classifier to predict each of the bits of each pixel.

```
y_grid = predict(classifier, newdata = grid_set)
```

We graph the data, assign a title to each axis and to the graph.

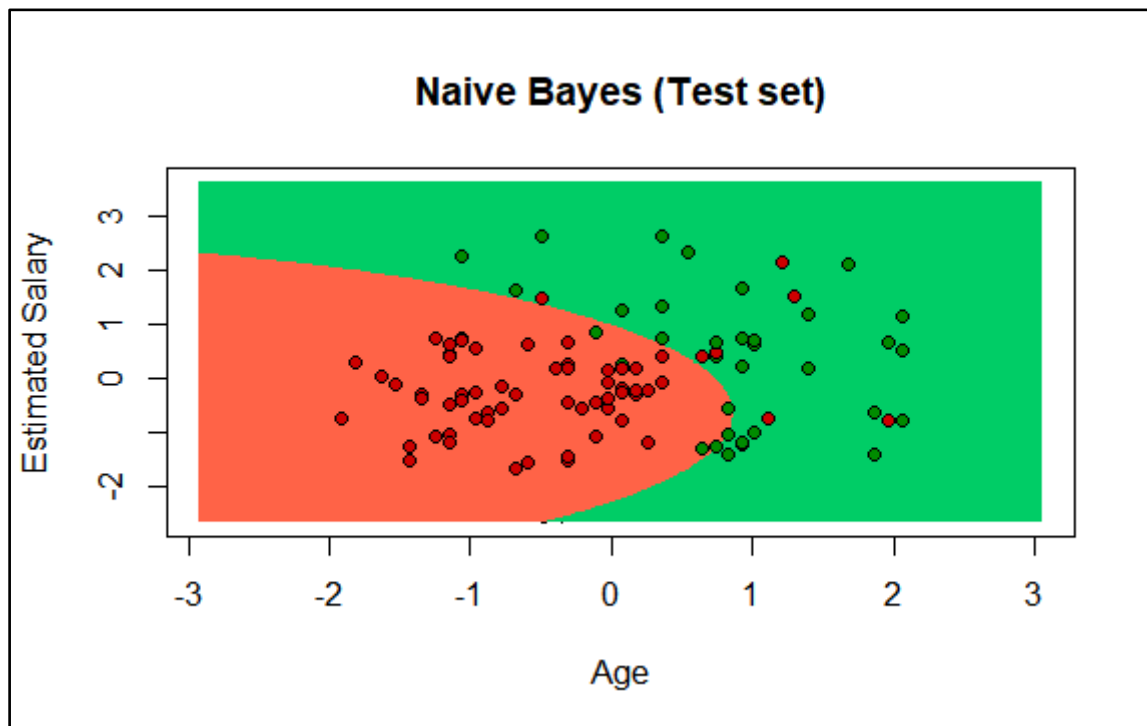
```
plot(set[, -3], main = 'SVM Radial Kernel (Test set)',
     xlab = 'Age', ylab = 'Estimated Salary',
     xlim = range(X1), ylim = range(X2))
```

The limits to the plotted values are created, the line between green and red is also created.

```
contour(X1, X2, matrix(as.numeric(y_grid), length(X1),
length(X2)), add = TRUE)
```

Here we review all the prediction data and use "ifelse" to color the points, taking into account that the points are the real data, the background is the pixel-by-pixel determination of the prediction and the points are plotted on top of the background create the image.

```
points(grid_set, pch = '.', col = ifelse(y_grid == 1, 'springgreen3',
'tomato'))
points(set, pch = 21, bg = ifelse(set[, 3] == 1, 'green4', 'red3'))
```



Conclusions and observations

Based on the observations, the first method is the easiest to implement in terms of syntax and understanding of the R language. However, when representing the results of the predictions (using the confusion matrix and), it is not easy to understand at first glance for users who are unfamiliar with the subject.

With the results obtained with the predictions, it can be concluded that the columns of age ("age") have a higher correlation than the columns of estimated salary ("EstimatedSalary") and Purchased ("Purchased"). This is evident in the point graphs with the "naive bayes" method by the line, which represents the fit, although in two dimensions it is not appreciated in the way it would be seen in third dimension.

The second method, with the use of the "ElemStatLearn" library, can generate a graph much easier to understand, at the cost of complex syntax. It can be seen in the second graph of the second method, the red points represent the people who did not buy the product and the green ones those who did.

We can conclude that the older the potential customer is, the more likely they are to sell the product. Unlike the customer being younger, there is less chance that he will purchase the product.

What are Naive Bayes models?

In a broad sense, Naive Bayes models are a special class of classification algorithms for Machine Learning, or Machine Learning, as we will refer to from now on. They are based on a statistical classification technique called "Bayes theorem."

These models are called "Naive" algorithms, or "Innocents" in Spanish. They assume that the predictor variables are independent of each other. In other words, that the presence of a certain feature in a data set is not at all related to the presence of any other feature.

They provide an easy way to build very well behaved models due to their simplicity. They do this by providing a way to calculate the 'later' probability of a certain event A occurring, given some probabilities of 'earlier' events.

The main strengths are:

- A fast and easy way to predict classes, for binary and multiclass classification problems.
- In cases where a presumption of independence is appropriate, the algorithm performs better than other classification models, even with less training data.
- The decoupling of class conditional characteristic distributions means that each distribution can be estimated independently as having only one dimension. This helps with dimensionality issues and improves performance.

The main weak points are

- Although they are quite good classifiers, Naive Bayes algorithms are notorious for being poor estimators. Therefore, the probabilities obtained should not be taken very seriously.
- The Naive independence presumption most likely does not reflect what data looks like in the real world.
- When the test data set has a characteristic that has not been observed in the training set, the model will assign a probability of zero and it will be useless to make predictions. One of the main methods to avoid this is the smoothing technique, the Laplace estimation being one of the most popular.

Bibliography

- VICTOR ROMAN. (2019-04-25). *Algoritmos Naive Bayes: Fundamentos e Implementación*. 2019-04-25, de CIENCIA&DATOS Sitio web: <https://medium.com/datos-y-ciencia/algoritmos-naive-bayes-fundamentos-e-implementaci%C3%B3n-4bcb24b307f>
- LIGDI GONZALEZ. (2019-09-20). *Naive Bayes – Teoría*. 2019-09-20, de APRENDE IA Sitio web: <https://aprendeia.com/naive-bayes-teoria-machine-learning/>
- Vídeo explicación: <https://www.youtube.com/watch?v=949tYJqRvRq>
- Ejemplo código: <https://programmerclick.com/article/84791158027/>