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 Evaluatoria - Unidad 3

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 Naive Bayes model

¿ What is the Naive Bayes classification model?

In both probability and data mining, a Naive Bayes classifier is a probabilistic method that has its base in Bayes' theorem and is called Naive given some additional simplifications that determine the hypothesis of independence of the predictor variables.

In simple terms, the Naive Bayes classifier assumes that the presence or absence of a particular feature is not related to the presence or absence of any other feature. For example, a fruit can be considered an apple if it is red, round, and about 7 cm in diameter.

A Naive Bayes classifier considers that each of these characteristics contributes independently to the probability that this fruit is an apple, regardless of the presence or absence of the other characteristics.

In many practical applications, parameter estimation for Bayes models uses the maximum likelihood method, that is, one can work with the Naive Bayes model without accepting Bayesian probability or any of the Bayesian methods.

An advantage of the Naive Bayes classifier is that only a small amount of training data is required to estimate the parameters necessary for the classification (the measures and variations of the variables).

It is only necessary to determine the variances of the variables of each class and not the entire covariance matrix. For other probability models, naive Bayes classifiers can be trained in supervised learning settings.

Explained step by step, with more details and visualization of formulas, The classifier works as follows:

1. Let D be a training set with its associated classifier class.
2. Suppose there are m classes C1, C2, ... Cm. Given a tuple X, the classifier will predict that X belongs to the class that has the highest posterior probability, conditional on X. That is: P (Ci | X)> P (Cj | X) $1 \leq j \leq m, j \neq i$.
3. P (X) is constant for all classes, so only P (X | Ci) P (Ci) need to be maximized. To estimate the probabilities of the classes, we use: P (Ci) = | $Ci_{,D}$| / | D |. Where | $Ci_{,D}$| is the number of training tuples of the class One Hundred D.
4. To reduce the computational cost, NaïveBayes assumes that the evidence is divided into parts, that is, the attributes are independent. So the calculations reduce to: P (X | Ci) = $\prod_{k=1}^{n}$ P ($x_{k \mid Ci}$) = P (x1 | Ci) * P (x2 | Ci) ..... P (x_n| Ci).

   a) If A_k is categorical, then P (x_k| Ci) is the number of tuples of the class Ci in D that have the value x_k in Ak. Divided by | Ci, D |.

b) If Ak is continuous, then the values are assumed to follow a normal distribution with mean μ and standard deviation σ. Defined by:

$$g\mu(x,)\sigma = \frac{1e^{\frac{-\mu(x-)2}{2\sigma^2}}}{\sqrt{2\pi\sigma}}$$

5. Finally to predict the class sorter X, P (X | Ci) P (Ci) is calculated for each class Ci. The classifier predicts that the classifier class for tuple X is class Ci. Yes. P (X | Ci) P (Cj)> P (X | Cj) P (Cj). 1 ≤ j ≤ m, j ≠ i.

NaïveBayes predicts binary or multiclass outcomes. In binary problems, each record will or will not follow the modeled behavior {yes, no}. You can make predictions for multiclass problems, in which there are several possible outcomes {good, bad, fair}.

Directory change

By default, R Studio comes with a path that is probably not the one used to store the elements to be used, so you must use getwd to obtain the current directory and setwd to assign a new one.

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

Import and select dataset

Once in the desired folder, the dataset to be used with the read.csv function must be loaded into memory. Not always all columns or data from a table are needed, in this case all data is needed, but only the last 3 columns,

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

Coding of the destination function as a factor

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

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

Divide the dataset into a training and test set

Loaded in the caTools library, the sample.split function is used to divide the dataset into 2, a training set and a test set, the .75 means that 300 data will be taken for training_set and 100 for test_set ..

```
library(caTools)
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 e1071 library is loaded into memory to be able to use the naiveBaytes method . The formula parameter is not needed here, as was the case in other previously used functions. As can be seen, there are only the variables "x" and "y", the first being the independent variable, being the data with which the algorithm will be trained, and the second is the dependent variable, which is the data with the that the prediction will be tested.

```
library(e1071)
classifier = naiveBayes (x = training_set [-3],
                         y = training_set $ Purchased)
```

Predicting the results of the test set

The following line executes the classifier on the test set and specifies that the prediction is made on the column 3 of the mentioned set.

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

Making a confusion matrix

The last step before seeing the plotted result is to make a confusion matrix. This is to evaluate how accurate the prediction made in the previous step was. In this, the false positives and false negatives are added, thus obtaining the error percentage, that is, erroneous predictions that the algorithm made saying that a data was true or false when it was not. In this case, a 14% error was obtained.

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

```
  y_pred
    0  1
0  57  7
1   7 29
```

Display of training set results

```
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, newdata = grid_set)
plot (set [, -3],
     main = 'Naive Bayes (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
'))
```

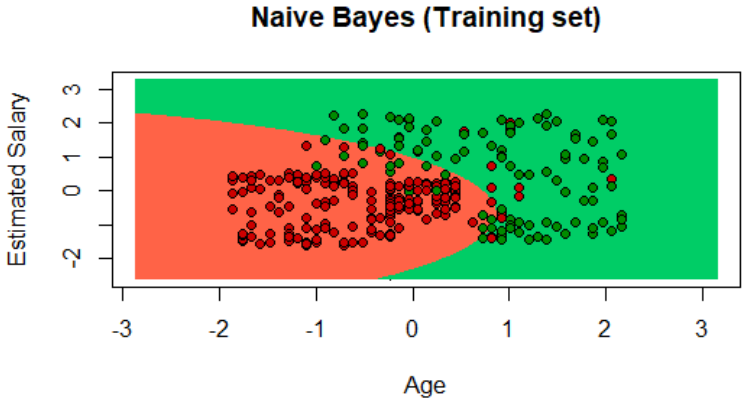
Viewing the results of the test set

The construction of the graph with test data follows the same steps as the previous one, only changing the data source.

```
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, newdata = grid_set)
plot ( set [, -3], main = 'Naive Bayes (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
'))
```

Training set graph conclusion

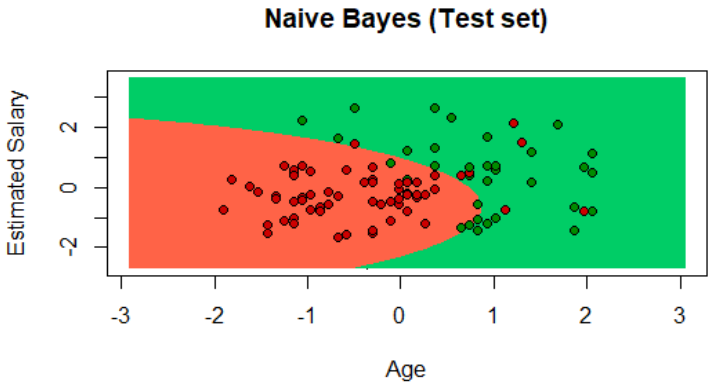
For the above code describe what the set are data training, for this the ElemStatLearn library was imported to be able to perform the statistics functions, then the variable for the test data set is declared, for the variables of X1 and X2, where there will be a region of red and green, so each 0.01 can be represented as 0 to 1 (green-red). Then the values are assigned to each axis of the graph, for this it would be the "Age" and "EstimatedSalary" fields, then the classifier was implemented in order to predict the values, they are graphed and in the PLOT section it will be assigned the title to each axis to the graph, the limits of the plotted values are created for the green and red colors in the line, and finally we use IFELSE to be able to color the plotted points, for that reason it is said that they are the real data and the bottom line is prediction.



Graph 1. Training set

Conclusion of the graph of the test set

For this graph it will be in the same way as the previous one, the evaluation and classification of the data is made, but it will not be training data but the test data, for this SET is assign the data of the variable TEST_SET that is obtained from the data corresponding to 25% of the entire dataset. And finally, IFELSE is used again to mark the points of the real data, in order to make the prediction and graph the points.



Graph 2. Test set

Conclusion

With the data obtained using the Naive Bayes method we can deduce that the Age fields have more correlation than those of EstimatedSalary, with the purchases made, therefore, this method is effective since it works mainly with algorithms of classification, since the handling of numbers is quite extensive, and each one is used in a different way, we can conclude that the Naive Bayes method offers us that great advantage.

YouTube video link:
https://youtu.be/si97rDDmjbA

GitHub:
https://github.com/JuanCarlos-Negrete/Data-Mining/tree/Unit_3/Unit_3/Evaluation