Hannah Roach
3-11-2019

LAB 8 Extra Credit – Naïve Bayesian Classifier – Census data

Workflow Overview

1. Define the Problem *(Translating to an Analytics Question)*
2. Establish the ODBC Connection
3. Open Connections to ODBC Database
4. Build the Training Dataset and the Test Dataset from the Database
5. Extract the first 10000 records for the training data set and the remaining 10 for the test
6. Execute the NB Classifier
7. Validate the Effectiveness of the NB Classifier with a Confusion Matrix
8. Execute NB Classifier with MADlib Function Calls Within the Database

LAB Instructions

<table>
<tr><th>Step</th><th>Action</th></tr>
<tr><td>1</td><td>

<u>Define the Problem *(Translating to an Analytics Question)*:</u>

1. Use “Persons” table in the Census dataset to categorize Age, Gender, Educational qualifications and Annual Income from each record as follows:
   - 3 Age categories: >20 and ≤ 30 , >30 and ≤45 and >45
   - 2 Gender categories: M and F
   - 3 Educational Qualifications categories: >14 (Professional/Phd), >12 and ≤14 (College) and <12 (others) –
   - 3 Annual Income categories: >10000 and ≤ 50000, >50000 and ≤ 80000 and > 80000
2. Build an appropriate “training” dataset, which will be a subset of the “categorized” table with four columns age, gender, education and income.
3. Predict the annual income category a person will belong to given the Age, Gender and educational qualifications, using the Naïve Bayesian Classifier.

</td></tr>
<tr><td>2</td><td>

<u>Establish the ODBC Connection:</u>

1. Load the RODBC package, using the following command:

```
> setwd("~/LAB08")
> library("e1071")
>library('RODBC')
```

</td></tr>
<tr><td>3</td><td>

<u>Open Connections to ODBC Database:</u>

1. Ensure the username(uid) and password (pwd) are provided correctly in the following command:

```
ch <- odbcConnect("Greenplum",uid="gpadmin",
      case="postgresql",pwd="changeme")
```

</td></tr>
</table>

Step	Action
4	**Build the Training Dataset and the Test Dataset from the Database:** 1. Drop the table, NBtrain, from the database, use the following command `sqlDrop(ch,"NBtrain")` 2. Execute a SQL query using the sqlQuery command, creating the table, NBtrain, selecting 10010 random records and categorizing the variables in the categories we defined in Step1 of Part2 of this lab. Use the code below.

```
1> sqlQuery(ch, "
 CREATE TABLE NBtrain (
   age VARCHAR(8)
 , sex VARCHAR(8)
 , educ VARCHAR(8)
 , income VARCHAR(8)
 ) DISTRIBUTED BY (age)
 ;
 INSERT INTO NBtrain
 SELECT
   t1.age
 , t1.sex
 , t1.educ
 , t1.income
 FROM (
   SELECT
     CASE
      WHEN age BETWEEN 20 AND 30 THEN '20-30'
      WHEN age BETWEEN 31 AND 45 THEN '31-45'
      WHEN age > 45 THEN 'GT 45'
      ELSE 'unknown age'
    END AS age
  , CASE
      WHEN sex = 1  THEN 'M'
      WHEN sex = 2 THEN 'F'
      ELSE 'unknown sex'
    END AS sex
  , CASE
      WHEN educ >14 THEN 'Prof/Phd'
      WHEN educ BETWEEN 12 AND 14 THEN 'College'
      WHEN educ <12 THEN 'Others'
      ELSE 'unknown educ'
    END AS educ
```

Step	Action

4
Cont.
	<pre>, CASE
	      WHEN inctot BETWEEN 10000 AND 50000 THEN '10-50K'
	      WHEN inctot BETWEEN 50000+1 AND 80000 THEN '50-80K'
	      WHEN inctot > 80000 THEN 'GT 80K'
	      ELSE 'unknown i'
	  END AS income
	 FROM
	    persons
	) AS t1
	WHERE
	  not (t1.age like 'unk%' or t1.sex like 'unk%' or t1.educ like
	'unk%' or t1.income like 'unk%')
	ORDER BY RANDOM ()
	LIMIT 10010
	;
	")</pre>

5	<u>**Extract the first 10000 records for the training data set and the remaining 10 for the test**</u>
	
	1. Use the sqlFetch command for reading data into an R data frame.
	
	`NBtrain <- (sqlFetch(ch,"NBtrain"))`
	
	2. Extract the training dataset
	`> NBtrain1 <- NBtrain[1:10000,]`
	
	3. Extract the test dataset
	`> NBtest <- NBtrain[10001:10010,]`
	
	4. Close the ODBC channel
	`> odbcClose(ch)`

6

Execute the NB Classifier:

1. Run the model as you did in Part 1. Use the following command:

```
# model
model <- naiveBayes(income ~.,NBtrain1,laplace=.01)
model
```

3. Screenshot the results

```
> model

Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
10-50K 50-80K GT 80K
0.8009 0.1225 0.0766

Conditional probabilities:
        age
Y          20-30  31-45  GT 45
  10-50K 0.2065 0.3402 0.4532
  50-80K 0.0784 0.4008 0.5208
  GT 80K 0.0313 0.3499 0.6188

        sex
Y            F     M
  10-50K 0.488 0.512
  50-80K 0.304 0.696
  GT 80K 0.221 0.779

        educ
Y         College Others Prof/Phd
  10-50K  0.2518 0.7347   0.0135
  50-80K  0.4988 0.4449   0.0563
  GT 80K  0.5222 0.3238   0.1541
```

Step	Action

3. Run the predict function using the following command:

```
# predict with testdata
results <- predict (model,NBtest[1:10,-1])
results
```

Screenshot

```
> results
 [1] 10-50K 10-50K 10-50K 10-50K 10-50K 10-50K 10-50K 10-50K 10-50K 10-50K
Levels: 10-50K 50-80K GT 80K
```

5. Use the parameter “type” with a value “raw” and you can see how the scores are very close to 0 or 1 rather than looking like realistic probabilities

```
> results1 <- predict (model,NBtest[1:10,-1],type="raw")
> results1
```

Record the results of the predict function:

```
> results1
      10-50K 50-80K GT 80K
 [1,]  0.929 0.0536 0.0177
 [2,]  0.782 0.1475 0.0702
 [3,]  0.782 0.1475 0.0702
 [4,]  0.929 0.0536 0.0177
 [5,]  0.584 0.2403 0.1761
 [6,]  0.929 0.0536 0.0177
 [7,]  0.584 0.2403 0.1761
 [8,]  0.584 0.2403 0.1761
 [9,]  0.840 0.1058 0.0539
[10,]  0.929 0.0536 0.0177
```

Note down your observations below: 10-50K has the highest probabilities

Step	Action
7	<u>**Validate the Effectiveness of the NB Classifier with a Confusion Matrix:**</u>

1. Compare the results of the previous step with the actual data that you have in NBtest. Build a confusion matrix for the predictions vs. actual values:

```
> conf <- table(actual=NBtest[1:10,4],predicted=results)
> conf
```

The diagonals of “conf” give the count of correctly classified instances by class. The off-diagonals tell how many instances of each class are mis-classified. What class does the model predict best?
10-50k
Screenshot Matrix

```
> conf
          predicted
actual    10-50K 50-80K GT 80K
  10-50K       5      0      0
  50-80K       5      0      0
  GT 80K       0      0      0
>
```

4. What % of data did the NB Classifier predicted correctly? You can calculate this as the sum of the diagonal elements of the confusion matrix normalized by the total number of test cases

```
> accuracy  <- sum(diag(conf))/sum(conf)
> accuracy
50%
```

END OF LAB