Sean Toon

IT-460-18500-M01 Machine Learning 2024 C-5

10/4/24

4-1 Lab Assignment and Report Brief: C5.0 Machine Learning

The data set that I am working with in this project includes examples of 1000 loans that were obtained from a credit agency in Germany. The data set includes the details about the loan along with characteristics of the loan and the loan applicant. There is also class variable that indicates whether the loan went into default. I used the following command to view the 17 features included in the set as along with some of the values stored after creating a object named “credit”.

```
> str(credit)
'data.frame':   1000 obs. of  17 variables:
 $ checking_balance    : chr  "< 0 DM" "1 - 200 DM" "unknown" "< 0 DM" ...
 $ months_loan_duration: int  6 48 12 42 24 36 24 36 12 30 ...
 $ credit_history      : chr  "critical" "good" "critical" "good" ...
 $ purpose             : chr  "furniture/appliances" "furniture/appliances" "education" "furniture/appliances" ...
 $ amount              : int  1169 5951 2096 7882 4870 9055 2835 6948 3059 5234 ...
 $ savings_balance     : chr  "unknown" "< 100 DM" "< 100 DM" "< 100 DM" ...
 $ employment_duration : chr  "> 7 years" "1 - 4 years" "4 - 7 years" "4 - 7 years" ...
 $ percent_of_income   : int  4 2 2 2 3 2 3 2 2 4 ...
 $ years_at_residence  : int  4 2 3 4 4 4 4 2 4 2 ...
 $ age                 : int  67 22 49 45 53 35 53 35 61 28 ...
 $ other_credit        : chr  "none" "none" "none" "none" ...
 $ housing             : chr  "own" "own" "own" "other" ...
 $ existing_loans_count: int  2 1 1 1 2 1 1 1 1 2 ...
 $ job                 : chr  "skilled" "skilled" "unskilled" "skilled" ...
 $ dependents          : int  1 1 2 2 2 2 1 1 1 1 ...
 $ phone               : chr  "yes" "no" "no" "no" ...
 $ default             : chr  "no" "yes" "no" "no" ...
```

My next step was to split the data into a training set and a test set. For this portion of the project, I used 90 percent of the data for training and 10 percent for testing performance using the last command in the screenshot. The 1000 represents the total number of examples in the set and 900 represents the number of examples that will be used for the training set. It’s also important to

make the selections process random for the sets so I used a seed value. To create the seed value, I used the first command with a value of “123”.

```
> RNGversion("3.5.2"); set.seed(123)
Warning message:
In RNGkind("Mersenne-Twister", "Inversion", "Rounding") :
  non-uniform 'Rounding' sampler used
>  train sample <- sample(1000, 900)
```

I then created the training and tests objects using the following commands. The first command is for the training sample, and I used the negation symbol to get the rest of the data set in the second command:

```
>  credit_train <- credit[train_sample, ]
> credit_test  <- credit[-train_sample, ]
```

The first decision tree I created was with this code:

```
> library(C50)
> credit_model <- C5.0(credit_train[-17], credit_train$default)
Error: C5.0 models require a factor outcome
>  credit_train$default <- as.factor(credit_train$default)
>
> credit_model <- C5.0(credit_train[-17], credit_train$default)
> credit_model

Call:
C5.0.default(x = credit_train[-17], y = credit_train$default)

Classification Tree
Number of samples: 900
Number of predictors: 16

Tree size: 57

Non-standard options: attempt to group attributes
```

I used the C5.0 algorithm to train the decision tree in the C50 with the top command. The second command had an error because “C5.0 models require a factor outcome”. I then used the third command to create a factor outcome. The command then worked, which created the model and

excluded the 17th column because it is the class variable “default”. The last command was to display the base data of the tree.

The next step was to evaluate the model performance by applying the decision tree to the test dataset. I did so by using the predict function which is the first command in the screenshot below. To get a better visualization of the vector of predicted class values created with the predict function and their comparison to the actual class values, I then created a cross table using the third command in the screenshot:

```
> credit_pred <- predict(credit_model, credit_test)
> library(gmodels)
> CrossTable(credit_test$default, credit_pred,
+              prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
+              dnn = c('actual default', 'predicted default'))
```

The last steps of this stage of my project were to try to improve the performance of the training model because of a very high error rate that was generated using my methods so far. The method that I used to try and improve the accuracy of the decision trees is boosting. Boosting is a process where many decision trees are built and then the trees vote on the best class for each example. The first command I used for boosting is shown below. The number of trials declared in the command sets the upper limit of how many trees to add.

```
> credit_boost10 <- C5.0(credit_train[-17], credit_train$default,
+                        trials = 10)
```

I then used the predict function to evaluate the model again:

```
> credit_boost_pred10 <- predict(credit_boost10, credit_test)
```

Although the results were greater and the accuracy improved in ways, there was still an issue with how many times the training model was incorrectly predicting defaults. I then used a cost

matrix to try and solve the issue. In the following commands I created a cost matrix, assigned predicted penalty values that represent a false negative no or yes to the value of 4 versus a false positive cost of 1. I then applied it to the decision tree using the costs parameter:

```
> error_cost <- matrix(c(0, 1, 4, 0), nrow = 2,
+                      dimnames = matrix_dimensions)
> error_cost
         actual
predicted no yes
      no   0   4
      yes  1   0
> credit_cost <- C5.0(credit_train[-17], credit_train$default,
+                             costs = error_cost)
> credit_cost_pred <- predict(credit_cost, credit_test)
> CrossTable(credit_test$default, credit_cost_pred,
+              prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
+              dnn = c('actual default', 'predicted default'))
```

For my next stage of the project, I changed the percentages of training and testing data to 60% and 40%, respectively. In the command I changed the training data value to 600 examples because that is 60 percent of 1000 examples. Below is the command that I used to change the percentages and the tables with my initial percentages following with the new percentages:

```
   Cell Contents
|-------------------------|
|                       N |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  100


                | predicted default
actual default |        no |       yes | Row Total |
---------------|-----------|-----------|-----------|
            no |        59 |         8 |        67 |
               |     0.590 |     0.080 |           |
---------------|-----------|-----------|-----------|
           yes |        19 |        14 |        33 |
               |     0.190 |     0.140 |           |
---------------|-----------|-----------|-----------|
  Column Total |        78 |        22 |       100 |
---------------|-----------|-----------|-----------|
```

```
> train_sample <- sample(1000, 600)
> str(train_sample)
 int [1:600] 288 788 409 881 937 46 525 887 548 453 ...
> credit_train <- credit[train_sample, ]
> credit_test  <- credit[-train_sample, ]
> prop.table(table(credit_train$default))
```

```
   Cell Contents
|-------------------------|
|                       N |
|         N / Table Total |
|-------------------------|

Total Observations in Table:  400

```

actual default	predicted default: no	predicted default: yes	Row Total
no	250	35	285
	0.625	0.087	
yes	78	37	115
	0.195	0.092	
Column Total	328	72	400

The accuracy of the model with 100 observations is 73 percent and the accuracy of the model with 400 observations is 71.75 percent. Although the change is slight, the change decreased the accuracy of the model. This was expected because generally the larger the test set is the more accurate it is going to be for testing model performance because it covers more cases and there is more room for variation.

For the last part of my project, I used the caret package to automatically tune the trials parameter. In the code below I first started by downloading and loading the caret package. I then used the seed function to set a starting position. I defined the tree as default and then told caret to use the C5.0 decision tree algorithm. I set metric to Kappa, and I also used an object name crtl that uses a 10-fold CV and the oneSE select function. I saved the experiment in an object named m:

```
> RNGversion("3.5.2")
Warning message:
In RNGkind("Mersenne-Twister", "Inversion", "Rounding") :
  non-uniform 'Rounding' sampler used
> set.seed(300)
> m <- train(default ~ ., data = credit, method = "C5.0",
+              metric = "Kappa",
+              trControl = ctrl,
+              tuneGrid = grid)
> m
```

In the output I was able to see that the oneSE rule selected the optimal model used. I concluded that with a trials parameter with 1 increased the accuracy of the model just slightly:

```
> m
C5.0

1000 samples
  16 predictor
   2 classes: 'no', 'yes'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 900, 900, 900, 900, 900, 900, ...
Resampling results across tuning parameters:

  trials  Accuracy  Kappa
   1      0.735     0.3243679
   5      0.722     0.2941429
  10      0.725     0.2954364
  15      0.731     0.3141866
  20      0.737     0.3245897
  25      0.726     0.2972530
  30      0.735     0.3233492
  35      0.736     0.3193931

Tuning parameter 'model' was held constant at a value of tree
Tuning parameter 'winnow' was held constant at
 a value of FALSE
Kappa was used to select the optimal model using  the one SE rule.
The final values used for the model were trials = 1, model = tree and winnow = FALSE.
```

References

Lantz, B. (2019). *Machine Learning with R* (3rd ed.). Packt Publishing.

https://mbsdirect.vitalsource.com/books/9781788291552