HW 10

April 29, 2021

1 IST 387 HW 10

[3]:
```
# Enter your name here: Connor Hanan
```

1.0.1 Attribution statement: (choose only one and delete the rest)

[4]:
```
# 1. I did this homework by myself, with help from the book and the professor.
```

Support vector machines (SVM) are a highly flexible and powerful method of doing **supervised machine learning**. Supervised learning means that there is a **criterion one is trying to predict**. The typical strategy is to **divide data** into a **training set** and a **test set** (for example, **two-thirds training** and **one-third test**), train the model on the training set, and then see how well the model does on the test set.

In this homework, we will use another banking dataset to train an SVM model to **classify potential borrowers into 2 groups of credit risk – reliable borrowers** and **borrowers posing a risk**. You can learn more about the variables in the dataset here: https://archive.ics.uci.edu/ml/datasets/Statlog+%28German+Credit+Data%29

This kind of classification algorithms is used in many aspects of our lives – from credit card approvals to stock market predictions, and even some medical diagnoses.

1.1 Part 1: Load and condition the data

A. Copy the contents of the following .csv file into a dataframe called **credit**:

https://ist387.s3.us-east-2.amazonaws.com/data/GermanCredit.csv

You will also need to install( ) and library( ) **kernlab** and **caret**.

[26]:
```
#install.packages('kernlab')
#install.packages('caret')
#install.packages('e1071')
```

```
also installing the dependency ‘proxy’


Updating HTML index of packages in '.Library'
```

```
Making 'packages.html' …
 done
```

[27]:
```
library(tidyverse)
library(kernlab)
library(caret)
library(e1071)
```

[8]:
```
credit <- read_csv("https://ist387.s3.us-east-2.amazonaws.com/data/GermanCredit.
↪csv")
```

```
  Column specification

cols(
  .default = col_character(),
  duration = col_double(),
  amount = col_double(),
  installment_rate = col_double(),
  present_residence = col_double(),
  age = col_double(),
  number_credits = col_double(),
  people_liable = col_double(),
  credit_risk = col_double()
)
  Use `spec()` for the full
column specifications.
```

B. Which variable contains the outcome we are trying to predict, **credit risk**? For the purposes of this analysis, we will focus only on the numeric variables and save them in a new dataframe called **cred**:

[9]:
```
cred <- data.frame(duration=credit$duration,
                   amount=credit$amount,
                   installment_rate=credit$installment_rate,
                   present_residence=credit$present_residence,
                   age=credit$age,
                   credit_history=credit$number_credits,
                   people_liable=credit$people_liable,
                   credit_risk=as.factor(credit$credit_risk))
```

C. Although all variables in **cred** except **credit__risk** are coded as numeric, the values of one of them are also **ordered factors** rather than actual numbers. In consultation with the **data description link** from the intro, write a comment identifying the **factor variable** and briefly **describe** each variable in the dataframe.

[11]:
```
str(cred)

#credit_risk is the factor varibale in this dataframe
#duration is duration in months
#amount is credit amount
#installment_rate is in percentage of disposable income
#present_residence is how long it has been the primary residence
#age is in years
#credit_history is number of existing creditsat this bank
#people_liable is number people liable to provide maintenance for
```

```
'data.frame':   1000 obs. of  8 variables:
 $ duration         : num  6 48 12 42 24 36 24 36 12 30 …
 $ amount           : num  1169 5951 2096 7882 4870 …
 $ installment_rate : num  4 2 2 2 3 2 3 2 2 4 …
 $ present_residence: num  4 2 3 4 4 4 4 2 4 2 …
 $ age              : num  67 22 49 45 53 35 53 35 61 28 …
 $ credit_history   : num  2 1 1 1 2 1 1 1 1 2 …
 $ people_liable    : num  1 1 2 2 2 2 1 1 1 1 …
 $ credit_risk      : Factor w/ 2 levels "0","1": 2 1 2 2 1 2 2 2 2 1 …
```

1.2 Part 2: Create training and test data sets

D. Using techniques discussed in class, create **two datasets** – one for **training** and one for **testing**.

[12]:
```
trainlist <- createDataPartition(y=cred$credit_risk, p=.70, list=FALSE)
trainset <- cred[trainlist,]
testset <- cred[-trainlist,]
```

E. Use the dim( ) function to demonstrate that the resulting training data set and test data set contain the appropriate number of cases.

[13]:
```
dim(trainset)
dim(testset)
```

1. 700 2. 8

1. 300 2. 8

1.3 Part 3: Build a Model using ksvm( )

F. Build a support vector model using the **ksvm( )** function using all of the variables to predict **credit_risk**. Once you have specified the model statement and the name of the training data set, you can use the same parameters as shown on page 237 of the textbook:

[14]:
```
svmout <- ksvm(credit_risk ~ ., data=trainset, kernel= "rbfdot", kpar =␣
↪"automatic",
    C = 5, cross = 3, prob.model = TRUE)
```

G. Write a block comment that summarizes what you learned from the book about those parameters. The two parameters of greatest interest are **C=5** and **cross=3**.

[15]:
```
#the C value is the penalty applied to the algorithm for missing a value - in␣
↪other words, a higher C value will make a more accurate model while a lower␣
↪C value will make a simpler model
#the cross value is used to asses the validation of the training data - this␣
↪model uses 3-fold cross validation
```

H. Store the output of the kvsm( ) run in a variable called **svmOut** and then **echo** that variable to the console. You will know that you are on the right track if your cross-validation error is reported in the neighborhood of **0.3**. The other output information is mainly diagnostic and is not of great concern at this time.

[16]:
```
svmout
```

```
Support Vector Machine object of class "ksvm"

SV type: C-svc  (classification)
 parameter : cost C = 5

Gaussian Radial Basis kernel function.
 Hyperparameter : sigma =  0.126921422719456

Number of Support Vectors : 463

Objective Function Value : -1797.982
Training error : 0.23
Cross validation error : 0.318532
Probability model included.
```

1.4 Part 4: Predict Values in the Test Data and Create a Confusion Matrix

I. Use the predict( ) function to validate the model against test data. Store the predictions in a variable named **svmPred**.

[18]:
```
svmpred <- predict(svmout,testset,type='response')
```

J. The **svmPred** object contains a list of classifications for reliable (=0) or risky (=1) borrowers. Review the contents of **svmPred** using View(), str( ), or head( ).

[19]:
```
str(svmpred)
```

```
 Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 …
```

K. Create a **confusion matrix** (a 2 x 2 table) that compares **svmPred** to the contents of **testSet$credit_risk**.

[21]:
```
table(testset$credit_risk,svmpred)
```

```
   svmpred
      0   1
0    16  74
1    14 196
```

L. Calculate an **error rate** based on what you see in the confusion matrix. See pages 243-244 of the textbook for more information.

[22]:
```
sum(diag(table(testset$credit_risk,svmpred)))/
↪sum(table(testset$credit_risk,svmpred))
```

0.706666666666667

M. Compare your calculations with the **confusionMatrix()** function from the **caret** package.

[28]:
```
confusionMatrix(testset$credit_risk,svmpred)
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0  16  74
         1  14 196

               Accuracy : 0.7067
                 95% CI : (0.6516, 0.7576)
    No Information Rate : 0.9
    P-Value [Acc > NIR] : 1

                  Kappa : 0.1373

 Mcnemar's Test P-Value : 3.187e-10

            Sensitivity : 0.53333
            Specificity : 0.72593
         Pos Pred Value : 0.17778
         Neg Pred Value : 0.93333
             Prevalence : 0.10000
         Detection Rate : 0.05333
   Detection Prevalence : 0.30000
      Balanced Accuracy : 0.62963

       'Positive' Class : 0
```

N. Explain, in a block comment: 1) why it is valuable to have a "test" dataset that is separate from a "training" dataset, and 2) what potential ethical challenges this type of automated classification may pose.

[29]:
```
#it is valuable because if you train the algorithm on the whole dataset, it␣
↪will always get a subset of it correct, as it already knows the answer - it␣
↪is better to keep part of it separate so the algorithm can test itself on␣
↪"real" data
#this type of automated classification poses ethical dilemmas around the␣
↪marginal edge cases, where we as humans can differentiate between the␣
↪options but a machine can't (think like the cat/dog picture in class, but␣
↪for, say, determining something about humans, such as gender)
```