Introduction to Machine Learning in R with caret

Keaton Wilson

8/30/2018

Introduction to Machine Learning in R with caret

Part 1 - What is machine learning? What are the tenets, what is the basic workflow?

Discussion - three questions (5-minutes with the person sitting next to you - then we'll come together and discuss as a group)

1. What is machine learning?

2. How is it different than statistics?

Some important things to know and think about:

1. Prediction is usually more important than explanation

2. Two major types of problems - regression and classification

3. Splitting the data to prevent overfitting

Classifcation Problem - Wine varietal identifier

Here is the scenario: we've been contacted by a famous vignter in Italy because she suspects that one of the prized varietals (a rare version of *Aglianicone* that her family has grown for 7 generations) from her vinyard has been stolen, and is being grown and sold to make competitively delicious wine in the United States. The competing winemaker claims that the varietal being grown in the US is from a closely related varietal from the same region, that he obtained legally.

Our customer has hired us to develop an algorithm to determine the likelihood that this is the wine being sold by the competitor was made from the varietal grown on her farm. Unfortunately, we don't have fancy genomic data to work with, but she has provided us with chemical profiles of a bunch of different wines made from both her grapes and two varietals that the competitor claims to be working with. The owner of the competing US vinyard has graciously provided us with the same type of data from a bunch of his wines to make comparisons on - he's looking to clear his name (and probably doesn't also believe that an algorithm can predict whether or not a given wine comes from a certain regional varietal)

Part 2 - Examining the Data

```r
# Getting libraries we need loaded
library(caret)
library(tidyverse)
```

```r
#Reading in the data from the github repo
wine_train = read_csv(file = "https://raw.githubusercontent.com/keatonwilson/classification_workshop_1/r
wine_test = read_csv(file = "https://raw.githubusercontent.com/keatonwilson/classification_workshop_1/ma
#Overviews
glimpse(wine_train)
```

```
## Observations: 152
## Variables: 14
## $ varietal        <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1...
## $ alcohol         <dbl> 13.20, 13.16, 14.37, 13.24, 14.39, 14.06, 14.8...
## $ malic_acid      <dbl> 1.78, 2.36, 1.95, 2.59, 1.87, 2.15, 1.64, 1.35...
## $ ash             <dbl> 2.14, 2.67, 2.50, 2.87, 2.45, 2.61, 2.17, 2.27...
## $ alkalinity      <dbl> 11.2, 18.6, 16.8, 21.0, 14.6, 17.6, 14.0, 16.0...
## $ magnesium       <int> 100, 101, 113, 118, 96, 121, 97, 98, 105, 95, ...
## $ total_phenol    <dbl> 2.65, 2.80, 3.85, 2.80, 2.50, 2.60, 2.80, 2.98...
## $ flavanoids      <dbl> 2.76, 3.24, 3.49, 2.69, 2.52, 2.51, 2.98, 3.15...
## $ nonflav_phenols <dbl> 0.26, 0.30, 0.24, 0.39, 0.30, 0.31, 0.29, 0.22...
## $ proantho        <dbl> 1.28, 2.81, 2.18, 1.82, 1.98, 1.25, 1.98, 1.85...
## $ color           <dbl> 4.38, 5.68, 7.80, 4.32, 5.25, 5.05, 5.20, 7.22...
## $ hue             <dbl> 1.05, 1.03, 0.86, 1.04, 1.02, 1.06, 1.08, 1.01...
## $ OD              <dbl> 3.40, 3.17, 3.45, 2.93, 3.58, 3.58, 2.85, 3.55...
## $ proline         <int> 1050, 1185, 1480, 735, 1290, 1295, 1045, 1045,...
```

```r
summary(wine_train)
```

```
##     varietal        alcohol        malic_acid         ash       
##  Min.   :1.000   Min.   :11.03   Min.   :0.740   Min.   :1.700  
##  1st Qu.:1.000   1st Qu.:12.37   1st Qu.:1.607   1st Qu.:2.200  
##  Median :2.000   Median :13.05   Median :1.875   Median :2.360  
##  Mean   :1.928   Mean   :13.02   Mean   :2.354   Mean   :2.363  
##  3rd Qu.:3.000   3rd Qu.:13.68   3rd Qu.:3.170   3rd Qu.:2.560  
##  Max.   :3.000   Max.   :14.83   Max.   :5.800   Max.   :3.220  
##    alkalinity      magnesium       total_phenol     flavanoids   
##  Min.   :11.20   Min.   : 70.00   Min.   :0.980   Min.   :0.340  
##  1st Qu.:17.20   1st Qu.: 88.75   1st Qu.:1.748   1st Qu.:1.215  
##  Median :19.05   Median : 98.00   Median :2.355   Median :2.120  
##  Mean   :19.42   Mean   : 99.65   Mean   :2.302   Mean   :2.012  
##  3rd Qu.:21.50   3rd Qu.:106.00   3rd Qu.:2.800   3rd Qu.:2.812  
##  Max.   :30.00   Max.   :162.00   Max.   :3.880   Max.   :3.750  
##  nonflav_phenols     proantho         color             hue        
##  Min.   :0.1300   Min.   :0.410   Min.   : 1.280   Min.   :0.5400  
##  1st Qu.:0.2700   1st Qu.:1.250   1st Qu.: 3.200   1st Qu.:0.7875  
##  Median :0.3400   Median :1.555   Median : 4.800   Median :0.9750  
##  Mean   :0.3667   Mean   :1.604   Mean   : 5.069   Mean   :0.9601  
##  3rd Qu.:0.4500   3rd Qu.:1.970   3rd Qu.: 6.263   3rd Qu.:1.1200  
##  Max.   :0.6600   Max.   :3.280   Max.   :13.000   Max.   :1.7100  
##        OD           proline      
##  Min.   :1.290   Min.   : 278.0  
##  1st Qu.:2.007   1st Qu.: 510.0  
##  Median :2.775   Median : 679.0  
##  Mean   :2.620   Mean   : 755.1  
##  3rd Qu.:3.170   3rd Qu.:1016.2  
##  Max.   :4.000   Max.   :1547.0  
```

```r
#Checking for NAs
sum(is.na(wine_train))
```

```
## [1] 0
```

Ok, so this looks good. We have our item we want to classify in column 1, and all of our features in the rest. For our varietal numbers, 1 and 2 are the local varietals not owned by our customer, but varietal 3 is her special grape. So we're looking for the presence of any wines made from varietal 3 in the test set.

What do we need to do before we jump into to trying to build some algorithms?

Preprocess! In particular, we need to center and scale the data. *caret* can do this for us.

Part 3 - Preprocessing

```r
#Setting up the preprocessing algorithm
set.seed(42)
pp = preProcess(wine_train[,-1], method = c("center", "scale"), outcome = wine_train$varietal)

wine_train_pp = predict(pp, wine_train)
wine_train_pp
```

```
## # A tibble: 152 x 14
##    varietal alcohol malic_acid    ash alkalinity magnesium total_phenol
##       <int>   <dbl>      <dbl>  <dbl>      <dbl>     <dbl>        <dbl>
##  1        1   0.227   -0.517   -0.877     -2.55     0.0242        0.568
##  2        1   0.178    0.00503  1.20      -0.255    0.0935        0.813
##  3        1   1.67    -0.364    0.537     -0.813    0.925         2.53
##  4        1   0.276    0.212    1.99       0.488    1.27          0.813
##  5        1   1.69    -0.436    0.340     -1.49    -0.253         0.323
##  6        1   1.29    -0.184    0.969     -0.565    1.48          0.487
##  7        1   2.24    -0.643   -0.759     -1.68    -0.184         0.813
##  8        1   1.04    -0.904   -0.366     -1.06    -0.114         1.11
##  9        1   1.34    -0.175   -0.249     -0.441    0.371         1.06
## 10        1   1.36    -0.787   -0.170     -0.813   -0.322        -0.167
## # ... with 142 more rows, and 7 more variables: flavanoids <dbl>,
## #   nonflav_phenols <dbl>, proantho <dbl>, color <dbl>, hue <dbl>,
## #   OD <dbl>, proline <dbl>
```

```r
#We also need to add this same processing algorithm to the test data.
wine_test_pp = predict(pp, wine_test)

#We also need to make the varietal category a factor in both datasets
wine_train_pp = wine_train_pp %>%
  mutate(varietal = factor(varietal))
wine_test_pp = wine_test_pp %>%
  mutate(varietal = as.factor(varietal))
```

Part 4 - Model Testing and Tuning

There are a ton of classification models to choose from - when starting ML stuff, this can be a really daunting part of the thing. Today, we're going to explore a couple of bread-and-butter models:
1. k-nn - nearest neighbord classifier

2. Naive Bayes
3. Decision Trees
4. Support Vector Machines

I'm not going to go into the math of how all of these operate at all. It's the beyond the scope of this workshop, but here is a good overview: https://medium.com/@sifium/machine-learning-types-of-classification-9497bd4f2e14

One thing that we need to talk about briefly is resampling - this is the method we're going to use to assess how 'good' a model is, without applying it to the test data. There are a couple of main ways to do this:
1. bootstrapping - random sampling within the dataset with replacement. Pulling a bunch of subsets of the data and looking at how the model performs across these subsets.
2. Repeated n-fold cross-validation - does a bunch of splitting into training and test data **within** the training set, and then averages accuracy or RMSE across all these little mini-sets.

We're going to use the second type.

```
# setting up the control object to feed to all of the subsequent models
fit_control = trainControl(method = "repeatedcv", number = 5, repeats = 5)

#models
#knn
knn_model = train(varietal ~ ., data = wine_train_pp,
                  method = "knn", trControl = fit_control)
#naive bays
bayes_model = train(varietal ~ ., data = wine_train_pp,
                  method = "nb", trControl = fit_control)

#CART
cart_model = train(varietal ~ ., data = wine_train_pp,
                  method = "rpart", trControl = fit_control)

#svm
svm_model = train(varietal ~ ., data = wine_train_pp,
                  method = "svmRadial", trControl = fit_control)
```

The models default to using accuracy as the score to determine how good they are. Accuracy is **the percentage of the predictions made by the model that are correct**.

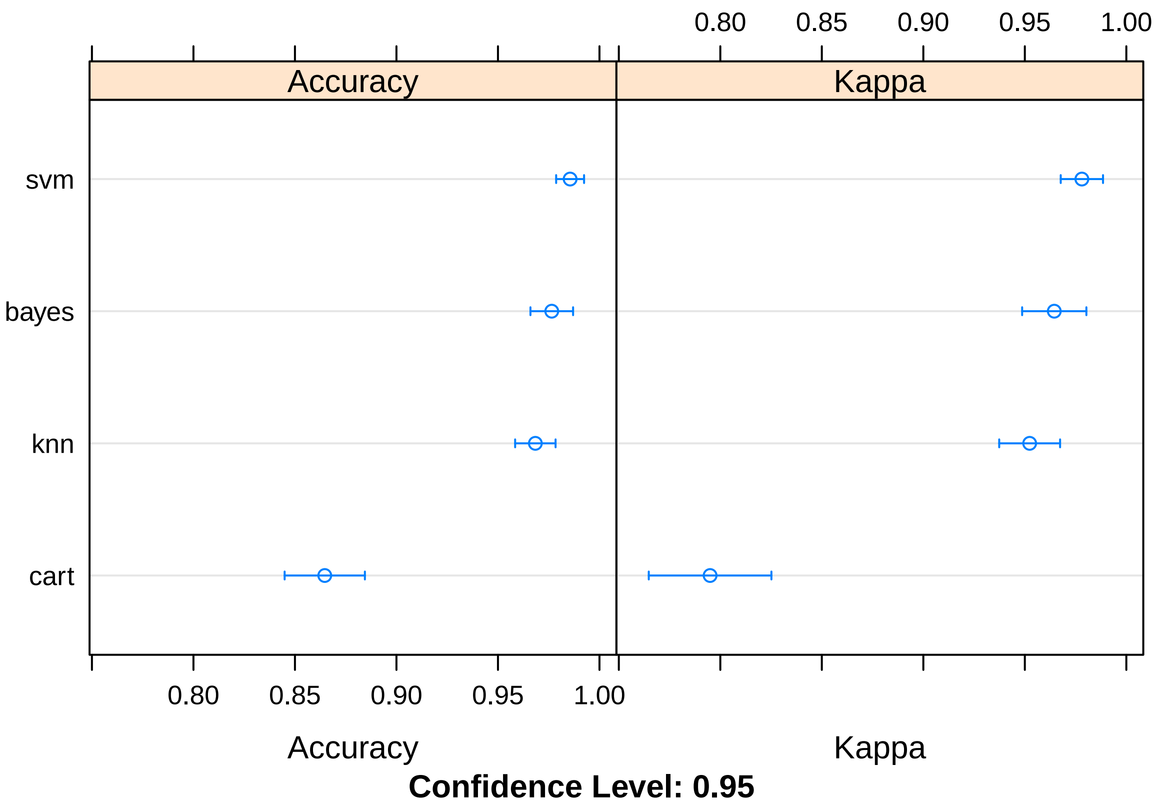
Let's comapre models

```
results = resamples(list(knn = knn_model, bayes = bayes_model, cart = cart_model, svm = svm_model))
summary(results)
```

```
##
## Call:
## summary.resamples(object = results)
##
## Models: knn, bayes, cart, svm
## Number of resamples: 25
##
## Accuracy
##            Min.   1st Qu.    Median      Mean 3rd Qu.      Max. NA's
## knn   0.9333333 0.9655172 0.9666667 0.9684245     1.0 1.0000000    0
## bayes 0.9062500 0.9666667 0.9677419 0.9765051     1.0 1.0000000    0
## cart  0.7666667 0.8333333 0.8666667 0.8646830     0.9 0.9677419    0
## svm   0.9655172 0.9677419 1.0000000 0.9855454     1.0 1.0000000    0
##
```

```
## Kappa
##            Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## knn   0.8993289 0.9479354 0.9496644 0.9523755 1.0000000 1.0000000    0
## bayes 0.8598540 0.9494949 0.9514107 0.9644991 1.0000000 1.0000000    0
## cart  0.6391753 0.7457627 0.7993311 0.7949667 0.8489933 0.9512579    0
## svm   0.9475588 0.9508716 1.0000000 0.9781022 1.0000000 1.0000000    0
```

```
dotplot(results)
```




```
svm_model
```

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 152 samples
##  13 predictor
##   3 classes: '1', '2', '3'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 5 times)
## Summary of sample sizes: 121, 121, 123, 122, 121, 121, ...
## Resampling results across tuning parameters:
##
##   C     Accuracy   Kappa
##   0.25  0.9816314  0.9721924
##   0.50  0.9829218  0.9741329
##   1.00  0.9855454  0.9781022
##
## Tuning parameter 'sigma' was held constant at a value of 0.06748482
## Accuracy was used to select the optimal model using the largest value.
```

```
## The final values used for the model were sigma = 0.06748482 and C = 1.
```

We can also look at the relative amount of false negatives and false positives with a confusion matrix.

```r
predictions = predict(svm_model, wine_train_pp)
confusionMatrix(predictions, wine_train_pp$varietal)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2  3
##          1 52  0  0
##          2  0 59  0
##          3  0  0 41
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.976, 1)
##     No Information Rate : 0.3882
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3
## Sensitivity            1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000
## Prevalence             0.3421   0.3882   0.2697
## Detection Rate         0.3421   0.3882   0.2697
## Detection Prevalence   0.3421   0.3882   0.2697
## Balanced Accuracy      1.0000   1.0000   1.0000
```

Not particularly informative, given the model did a 100% accurate job of predicting on the test-data, but you get the gist.

Part 5 - Using the model on the test data.

```r
wine_test_pp$pred = predict(svm_model, wine_test_pp)

wine_test_pp %>%
  select(varietal, pred)
```

```
## # A tibble: 10 x 2
##    varietal pred
##    <fct>    <fct>
##  1 3        3
##  2 2        2
##  3 3        3
##  4 2        2
##  5 2        2
```

```
##   6 2           2
##   7 2           2
##   8 1           1
##   9 2           2
##  10 3           3
```