Introduction to Machine Learning in R with tidymodels

Keaton Wilson

5/22/2019

Introduction to Machine Learning in R with caret

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

Discussion - two 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

Part 2 - Getting your hands dirty:

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)

Examining the Data

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

#Reading in the data from the github repo
wine_data = read_csv("https://tinyurl.com/y82aefsj")

#Overviews
glimpse(wine_data)
```

```
## Rows: 178
## Columns: 14
## $ varietal        <dbl> 1, 1, 1, 1, 2, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
## $ alcohol         <dbl> 14.23, 13.20, 13.16, 14.37, 13.24, 14.20, 14.39, 14...
## $ malic_acid      <dbl> 1.71, 1.78, 2.36, 1.95, 2.59, 1.76, 1.87, 2.15, 1.6...
## $ ash             <dbl> 2.43, 2.14, 2.67, 2.50, 2.87, 2.45, 2.45, 2.61, 2.1...
## $ alkalinity      <dbl> 15.6, 11.2, 18.6, 16.8, 21.0, 15.2, 14.6, 17.6, 14....
## $ magnesium       <dbl> 127, 100, 101, 113, 118, 112, 96, 121, 97, 98, 105,...
## $ total_phenol    <dbl> 2.80, 2.65, 2.80, 3.85, 2.80, 3.27, 2.50, 2.60, 2.8...
## $ flavanoids      <dbl> 3.06, 2.76, 3.24, 3.49, 2.69, 3.39, 2.52, 2.51, 2.9...
## $ nonflav_phenols <dbl> 0.28, 0.26, 0.30, 0.24, 0.39, 0.34, 0.30, 0.31, 0.2...
## $ proantho        <dbl> 2.29, 1.28, 2.81, 2.18, 1.82, 1.97, 1.98, 1.25, 1.9...
## $ color           <dbl> 5.64, 4.38, 5.68, 7.80, 4.32, 6.75, 5.25, 5.05, 5.2...
## $ hue             <dbl> 1.04, 1.05, 1.03, 0.86, 1.04, 1.05, 1.02, 1.06, 1.0...
## $ OD              <dbl> 3.92, 3.40, 3.17, 3.45, 2.93, 2.85, 3.58, 3.58, 2.8...
## $ proline         <dbl> 1065, 1050, 1185, 1480, 735, 1450, 1290, 1295, 1045...
```

```r
summary(wine_data)
```

```
##     varietal        alcohol        malic_acid         ash
##  Min.   :1.000   Min.   :11.03   Min.   :0.740   Min.   :1.360
##  1st Qu.:1.000   1st Qu.:12.36   1st Qu.:1.603   1st Qu.:2.210
##  Median :2.000   Median :13.05   Median :1.865   Median :2.360
##  Mean   :1.921   Mean   :13.00   Mean   :2.336   Mean   :2.367
##  3rd Qu.:2.000   3rd Qu.:13.68   3rd Qu.:3.083   3rd Qu.:2.560
##  Max.   :3.000   Max.   :14.83   Max.   :5.800   Max.   :3.230
##                                                  NA's   :1
##    alkalinity      magnesium       total_phenol     flavanoids
##  Min.   :10.60   Min.   : 70.00   Min.   :0.980   Min.   :0.340
##  1st Qu.:17.20   1st Qu.: 88.00   1st Qu.:1.742   1st Qu.:1.220
##  Median :19.50   Median : 98.00   Median :2.355   Median :2.140
##  Mean   :19.49   Mean   : 99.75   Mean   :2.295   Mean   :2.036
##  3rd Qu.:21.50   3rd Qu.:107.00   3rd Qu.:2.800   3rd Qu.:2.880
##  Max.   :30.00   Max.   :162.00   Max.   :3.880   Max.   :5.080
##                  NA's   :1                        NA's   :1
##  nonflav_phenols    proantho         color             hue
##  Min.   :0.1300   Min.   :0.410   Min.   : 1.280   Min.   :0.4800
##  1st Qu.:0.2700   1st Qu.:1.250   1st Qu.: 3.220   1st Qu.:0.7825
##  Median :0.3400   Median :1.560   Median : 4.690   Median :0.9650
##  Mean   :0.3622   Mean   :1.593   Mean   : 5.058   Mean   :0.9574
##  3rd Qu.:0.4400   3rd Qu.:1.950   3rd Qu.: 6.200   3rd Qu.:1.1200
##  Max.   :0.6600   Max.   :3.580   Max.   :13.000   Max.   :1.7100
##  NA's   :1        NA's   :1
##        OD           proline
##  Min.   :1.270   Min.   : 278.0
##  1st Qu.:1.930   1st Qu.: 500.5
##  Median :2.780   Median : 673.5
##  Mean   :2.608   Mean   : 746.9
##  3rd Qu.:3.170   3rd Qu.: 985.0
##  Max.   :4.000   Max.   :1680.0
##  NA's   :1
```

```r
#making varietal a factor
wine_data = wine_data %>%
  mutate(varietal = as.factor(varietal))

#Checking for NAs
sum(is.na(wine_data))
```

```
## [1] 6
```

```r
#Uh oh - conversation about missing data.

# wine_data = wine_data %>%
#   drop_na()

#or we can deal with it in pre-processing.
```

Ok, so this looks good. We have our item we want to classify in column 1, and all of our features in the rest.

The tidymodels workflow

Here is the basic workflow for building ML models with tidymodels:
1. Split data into training and testing data
2. Make a recipe for preprocessing the data
3. Specify our model (and what hyperparameters we want to tune)
4. Set up our resampling scheme to tune
5. Fit models
6. Choose best model and evluate on test data

We'll talk about the steps above that need a bit of explanation as we go through things.

Training and testing split and preprocessing

```r
#Setting up the preprocessing algorithm
set.seed(42)

# changing varietal to a factor
wine_data = wine_data %>%
  mutate(varietal = factor(varietal))
```

```r
#Train and test split
data_split = initial_split(wine_data, prop = 0.80, strata = 'varietal')
wine_train = training(data_split)
wine_test = testing(data_split)

#Let's start to build a recipe
wine_rec = recipe(varietal ~ ., data = wine_train) %>%
  step_knnimpute(all_predictors()) %>%
  step_scale(all_predictors()) %>%
  step_nzv(all_predictors())

wine_prepped = prep(wine_rec, training = wine_train)
training_example = bake(wine_prepped, new_data = wine_train)
training_example
```

```
## # A tibble: 144 x 14
##    alcohol malic_acid   ash alkalinity magnesium total_phenol flavanoids
##      <dbl>      <dbl> <dbl>      <dbl>     <dbl>        <dbl>      <dbl>
##  1    16.0       1.60  7.74       3.32      6.96         4.32       2.77
##  2    15.9       2.13  9.66       5.52      7.03         4.57       3.25
##  3    17.4       1.76  9.04       4.99      7.86         6.28       3.50
##  4    16.0       2.33 10.4        6.23      8.21         4.57       2.70
##  5    17.2       1.59  8.86       4.51      7.79         5.33       3.40
##  6    17.4       1.69  8.86       4.33      6.68         4.08       2.53
##  7    17.0       1.94  9.44       5.22      8.42         4.24       2.52
##  8    16.8       1.22  8.21       4.75      6.82         4.86       3.16
##  9    17.1       1.95  8.32       5.34      7.31         4.81       3.33
## 10    16.6       1.56  8.72       4.75      6.19         4.24       2.77
## # ... with 134 more rows, and 7 more variables: nonflav_phenols <dbl>,
## #   proantho <dbl>, color <dbl>, hue <dbl>, OD <dbl>, proline <dbl>,
## #   varietal <fct>
```

So this is how you would view the data, and you could build things stepwise like this, but we're going to include it into a larger workflow.

Model Specification, 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 one type of model: Support Vector Machines

I'm not going to go into the math of how classification algorithms 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

A simple explanation (really awesome) of how SVM works: https://www.youtube.com/watch?v=efR1C6CvhmE

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.

Let's setup our model first.

```
# setting up the model
svm_mod = parsnip::svm_poly(cost = tune(),
                            degree = tune(),
                            scale_factor = tune()) %>%
  set_mode("classification") %>%
  set_engine("kernlab")

#inspect
svm_mod
```

```
## Polynomial Support Vector Machine Specification (classification)
##
## Main Arguments:
##   cost = tune()
##   degree = tune()
##   scale_factor = tune()
##
## Computational engine: kernlab
```

And now we'll setup our cross-validation scheme and tuning grid.

```
# Setting up folds
folds = vfold_cv(wine_train, strata = 'varietal', v = 10)
folds
```

```
## #  10-fold cross-validation using stratification
## # A tibble: 10 x 2
##    splits           id
##    <named list>     <chr>
##  1 <split [128/16]> Fold01
##  2 <split [129/15]> Fold02
##  3 <split [129/15]> Fold03
##  4 <split [129/15]> Fold04
##  5 <split [129/15]> Fold05
##  6 <split [129/15]> Fold06
##  7 <split [130/14]> Fold07
##  8 <split [131/13]> Fold08
##  9 <split [131/13]> Fold09
## 10 <split [131/13]> Fold10
```

```
#setting up tuning
svm_tune_grid = grid_regular(dials::cost(),
                       dials::degree(),
                       dials::scale_factor())
svm_tune_grid
```

```
## # A tibble: 27 x 3
##        cost degree scale_factor
##       <dbl>  <dbl>        <dbl>
##  1 0.000977      1 0.0000000001
##  2 0.0221        1 0.0000000001
##  3 0.5           1 0.0000000001
##  4 0.000977      2 0.0000000001
##  5 0.0221        2 0.0000000001
##  6 0.5           2 0.0000000001
##  7 0.000977      3 0.0000000001
##  8 0.0221        3 0.0000000001
##  9 0.5           3 0.0000000001
## 10 0.000977      1 0.00000316
## # ... with 17 more rows
```

Now we can put everything together into a workflow.

```
# let's build a workflow that pairs the model with the preprocessing
wine_wflow = workflow() %>%
  add_recipe(wine_rec) %>%
  add_model(svm_mod)

# Can inspect
wine_wflow
```

```
## == Workflow ==================================================================================
## Preprocessor: Recipe
## Model: svm_poly()
##
## -- Preprocessor ------------------------------------------------------------------------------
## 3 Recipe Steps
##
## * step_knnimpute()
## * step_scale()
## * step_nzv()
##
## -- Model -------------------------------------------------------------------------------------
## Polynomial Support Vector Machine Specification (classification)
##
## Main Arguments:
##   cost = tune()
##   degree = tune()
##   scale_factor = tune()
##
## Computational engine: kernlab
```

```
# Let's fit the model and simultanesouly run preprocessing - this can take a
# while
wine_svm_fit_results = wine_wflow %>%
  tune_grid(resamples = folds,
            grid = svm_tune_grid,
            metrics = metric_set(roc_auc, accuracy))

#inspect
wine_svm_fit_results
```

```
## #  10-fold cross-validation using stratification
## # A tibble: 10 x 4
##    splits           id     .metrics          .notes
##    <list>           <chr>  <list>            <list>
##  1 <split [128/16]> Fold01 <tibble [54 x 6]> <tibble [0 x 1]>
##  2 <split [129/15]> Fold02 <tibble [54 x 6]> <tibble [0 x 1]>
##  3 <split [129/15]> Fold03 <tibble [54 x 6]> <tibble [0 x 1]>
##  4 <split [129/15]> Fold04 <tibble [54 x 6]> <tibble [0 x 1]>
##  5 <split [129/15]> Fold05 <tibble [54 x 6]> <tibble [0 x 1]>
##  6 <split [129/15]> Fold06 <tibble [54 x 6]> <tibble [0 x 1]>
##  7 <split [130/14]> Fold07 <tibble [54 x 6]> <tibble [0 x 1]>
##  8 <split [131/13]> Fold08 <tibble [54 x 6]> <tibble [0 x 1]>
##  9 <split [131/13]> Fold09 <tibble [54 x 6]> <tibble [0 x 1]>
## 10 <split [131/13]> Fold10 <tibble [54 x 6]> <tibble [0 x 1]>
```

```r
wine_svm_fit_results$.metrics[[1]]
```

```
## # A tibble: 54 x 6
##        cost degree scale_factor .metric  .estimator .estimate
##       <dbl>  <dbl>        <dbl> <chr>    <chr>          <dbl>
##  1 0.000977      1 0.0000000001 accuracy multiclass     0.438
##  2 0.000977      1 0.0000000001 roc_auc  hand_till      0.906
##  3 0.0221        1 0.0000000001 accuracy multiclass     0.438
##  4 0.0221        1 0.0000000001 roc_auc  hand_till      0.906
##  5 0.5           1 0.0000000001 accuracy multiclass     0.438
##  6 0.5           1 0.0000000001 roc_auc  hand_till      0.906
##  7 0.000977      2 0.0000000001 accuracy multiclass     0.438
##  8 0.000977      2 0.0000000001 roc_auc  hand_till      0.906
##  9 0.0221        2 0.0000000001 accuracy multiclass     0.438
## 10 0.0221        2 0.0000000001 roc_auc  hand_till      0.906
## # ... with 44 more rows
```

So we can see how many models are actually being built here - for each fold, where there are 10 folds here, we are building a model for each hyperparameter set in our grid, which is 27 combinations. So in total, we built and assessed 270 models, and we're taking the average (of 10) accuracy of each 27 parameter combinations as our metric of how good a model is.

Now, let's collect our metrics and figure out what set of hyperparameters generated the best model.

```r
# all metrics
collect_metrics(wine_svm_fit_results)
```

```
## # A tibble: 54 x 8
##        cost degree scale_factor .metric  .estimator  mean     n std_err
##       <dbl>  <dbl>        <dbl> <chr>    <chr>      <dbl> <int>   <dbl>
##  1 0.000977      1 0.0000000001 accuracy multiclass 0.425    10 0.00901
##  2 0.000977      1 0.0000000001 roc_auc  hand_till  0.887    10 0.0243
##  3 0.000977      1 0.00000316   accuracy multiclass 0.425    10 0.00901
##  4 0.000977      1 0.00000316   roc_auc  hand_till  0.895    10 0.0177
##  5 0.000977      1 0.1          accuracy multiclass 0.425    10 0.00901
##  6 0.000977      1 0.1          roc_auc  hand_till  0.895    10 0.0246
##  7 0.000977      2 0.0000000001 accuracy multiclass 0.425    10 0.00901
##  8 0.000977      2 0.0000000001 roc_auc  hand_till  0.884    10 0.0245
##  9 0.000977      2 0.00000316   accuracy multiclass 0.425    10 0.00901
## 10 0.000977      2 0.00000316   roc_auc  hand_till  0.889    10 0.0213
## # ... with 44 more rows
```

```
# the top 5 based on roc scores
show_best(wine_svm_fit_results, metric = "roc_auc")
```

```
## # A tibble: 5 x 8
##       cost degree scale_factor .metric .estimator  mean     n std_err
##      <dbl>  <dbl>        <dbl> <chr>   <chr>      <dbl> <int>   <dbl>
## 1 0.5           2          0.1 roc_auc hand_till  0.912    10  0.0242
## 2 0.5           1          0.1 roc_auc hand_till  0.904    10  0.0207
## 3 0.000977      3          0.1 roc_auc hand_till  0.901    10  0.0216
## 4 0.0221        3          0.1 roc_auc hand_till  0.901    10  0.0225
## 5 0.000977      2          0.1 roc_auc hand_till  0.901    10  0.0223
```

```
# pull out parameters of the 'best' model
wine_svm_best = wine_svm_fit_results %>%
  select_best(metric = "roc_auc")

# add this step to the workflow
wine_wflow = wine_wflow %>%
  finalize_workflow(wine_svm_best)
```

Great! So we have our model finalized, and incorporated into our workflow, now we want to see how good the model does on out-of-sample data - or the data we saved as testing data very early in the process! This will give us a final, conservative estimate as to how our model does on new data!

```
# Evaluating on test data
# Last fit adds a step at the end of the workflow that builds on the training
# data and then evaluates on the test data we generated from our split

svm_fit = wine_wflow %>%
  last_fit(data_split)

test_performance = svm_fit %>% collect_metrics()
test_performance
```

```
## # A tibble: 2 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy multiclass     0.794
## 2 roc_auc  hand_till      0.866
```

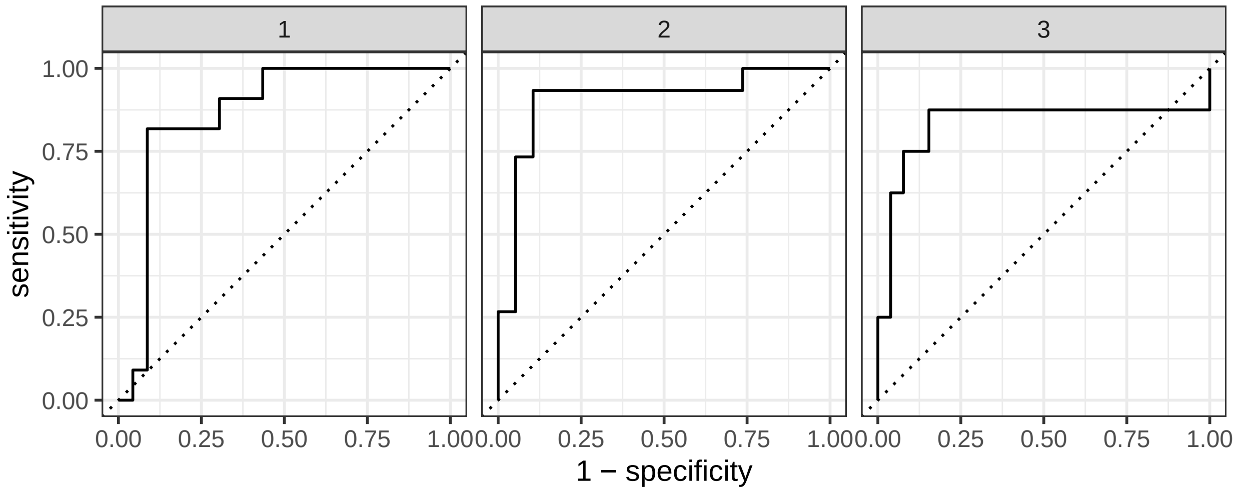
```
# overall, not too bad. We got ~ 80% accruacy overall, and our roc_auc score
# was 0.87.

# confusion matrix
test_predictions = collect_predictions(svm_fit)
test_predictions %>%
  conf_mat(truth = varietal, estimate = .pred_class)
```

```
##            Truth
## Prediction  1  2  3
##           1  9  1  1
##           2  1 12  1
##           3  1  2  6
```

```
# let's plot the roc curve
svm_roc = svm_fit$.predictions[[1]] %>%
  roc_curve(truth = varietal, .pred_1:.pred_3)

autoplot(svm_roc)
```



ROC curves are a pretty standard way of showing how good your model is when classifying things. A nice explanation is here https://www.youtube.com/watch?v=4jRBRDbJemM.

Without a lot of explanation: the y-axis shows the true positive rate, (sensitivity) and the x-axis shows the false positive rate (1-specificity).

Each point is the result of a confusion matrix generated at different thresholds (cutoffs that determine what probability you'll need to identify a sample as one varietal or another).

Summary

So that was a lot of work! But we're left with a pretty decent model - one that predicts the correct varietal of wine about 80% of the time. Improving this score is a big chunk of doing ML in the real-world. Are there different features we could create that would help? Should we expand our tuning grid more widely to see if there are a set of parameters that are better suited? We've only tested one type of model in a galaxy of options… perhaps comparing a few models to this one would be good!

Lots of options!

Continuing Practice

Some resources if you want to get better at this:
1. Tidymodels learning 2. Kaggle - an online community of data scientists - lots of cool datasets to play with, and competitions!
3. https://kbroman.org/pkg_primer/pages/resources.html - great list of resources!
4. Machine Learning with R - Brett Lantz. Great book!
5. Great article on different algorithm types