Computational Social Science

Supervised Machine Learning

Dr. Thomas Davidson

Rutgers University

March 30, 2022

Plan

1. Course updates
2. Classification algorithms
3. Intro to machine learning in R

Course updates

Homework 3

- Homework 3 due Friday at 5pm
- Office hours today at 5pm
  - No appointment needed

Recap

- Supervised learning optimizes for predictive accuracy (focus on $\hat{Y}$ not $\hat{\beta}$)
- Problems of over and under-fitting
  - Out-of-sample validation and cross-validation
  - Regularization
- Evaluating model performance
  - Precision, recall, F1, ROC/AUC

Recap

- Given some outcome Y and a matrix of features X, we want to find a function $Y = f(X)$ that best predicts the outcome

Recap

Recap

Predicting penguins

- $Y = 1$ if the bird is a penguin, otherwise $Y = 0$
- X is a matrix including information on birds including their diet, wingspan, coloring, locations, etc.
  - Some of the information will be useful (e.g. ability to fly) but other information will be less meaningful (e.g. coloring)
- Goal is to find $f(X)$ to predict whether a given bird is a penguin
- The quality of the prediction will depend on both the information contained in X and the properties of the function $f()$.

Classification algorithms

- Logistic regression
- Support vector machines (SVM)
- Decision trees and random forests
- Neural networks
- And many others

Machine learning in R

Tidymodels

- `tidymodels` is a set of packages designed to use tidy principles to conduct machine-learning.
  - See https://www.tidymodels.org/packages/ for a list of packages.

Source: tidymodels tutorial.

Machine learning in R

Loading `tidymodels`

The `tidymodels` package loads all of the sub-packages, as well as the `tidyverse` packages. We're going to be using a sample of data from the General Social Survey (GSS). The goal will be to predict whether a respondent has a college degree (or higher) as a function of their survey responses.

```
library(tidyverse)
library(tidymodels)
data <- read_csv("data/gss_sample.csv")

data <- data %>%
    replace(is.na(.), 0) %>%
    mutate(sex = as.factor(sex),
           race = as.factor(race),
           degree = as.factor(degree),
           bible = as.factor(bible)) %>%
    select(!educ)
```

Machine learning in R

Splitting data

We can use the `initial_split` command to create a train-test split, where 20% of the data are held-out for testing.

```
set.seed(08901)
data_split <- initial_split(data, prop = 0.8)
print(data_split)
```

```
## <Analysis/Assess/Total>
## <1872/469/2341>
```

Viewing the data

```
data_split %>% training() %>% head()
```

```
## # A tibble: 6 x 11
##     age sex   race  realrinc  hrs1 marital bible childs paeduc maedu
##   <dbl> <fct> <fct>    <dbl> <dbl>   <dbl> <fct>  <dbl>  <dbl>  <dbl
## 1    53 0     1       10782.    60       2 2          2      0
## 2    72 1     1        1589      0       1 3          0     15     1
## 3    65 0     1       17025     23       1 1          0     12     1
## 4    33 1     2       37455     50       5 2          0     14     1
## 5    36 1     1           0     60       1 1          2     14     1
## 6    27 1     3       37455     43       5 0          0     16     1
```

Machine learning in R

Pre-processing

We will use the `recipes` package to pre-process the data.

```
data_recipe <- training(data_split) %>%
  recipe(degree ~ .) %>%
  step_center(all_numeric_predictors(), -all_outcomes()) %>%
  step_scale(all_numeric_predictors(), -all_outcomes()) %>%
  prep()

data_recipe
```

```
## Recipe
##
## Inputs:
##
##        role #variables
##     outcome          1
##   predictor         10
##
## Training data contained 1872 data points and no missing data.
```

Machine learning in R

Pre-processing the test data

The previous chunk only applied these transformations to the training data. We want to also modify the test data so that they are the same dimensions. We can apply the `recipe` to the new data using the `bake` command. We also want to load the training data into a variable using the `juice` command. This extracts the data directly from the recipe.

```
data_testing <- data_recipe %>%
  bake(testing(data_split))

data_training <- juice(data_recipe)
```

Machine learning in R

Specifying a model

The `parsnip` command allows us to specify a model. ML models in R exist across a range of different packages and `parsnip` gives them a standardized syntax. We define the model, choose the package (in this case `randomForest`), then use `fit` to train the model.

```
library(randomForest)
rf <-  rand_forest(trees = 1000, mode = "classification") %>%
  set_engine("randomForest") %>%
  fit(degree ~ ., data = data_training)
```

Machine learning in R

Making predictions (in-sample)

```
preds <- predict(rf, data_training)
bind_cols(data_training, preds) %>% select(degree, .pred_class)
```

```
## # A tibble: 1,872 x 2
##    degree .pred_class
##    <fct>  <fct>
##  1 0      0
##  2 1      1
##  3 1      1
##  4 1      1
##  5 1      1
##  6 1      1
##  7 1      1
##  8 0      0
##  9 1      1
## 10 0      0
## # ... with 1,862 more rows
```

Calculating metrics (in-sample)

```
precision <- bind_cols(data_training, preds) %>% precision(truth=degree
recall <- bind_cols(data_training, preds) %>% recall(truth=degree, esti
print(bind_rows(precision, recall))
```

```
## # A tibble: 2 x 3
##   .metric   .estimator .estimate
##   <chr>     <chr>          <dbl>
## 1 precision binary         0.998
## 2 recall    binary         1
```

Machine learning in R

Making predictions (out-of-sample)

```
preds <- predict(rf, data_testing)
bind_cols(data_testing, preds) %>% select(degree, .pred_class)
```

```
## # A tibble: 469 x 2
##    degree .pred_class
##    <fct>  <fct>
##  1 1      1
##  2 1      1
##  3 0      0
##  4 0      0
##  5 0      0
##  6 0      1
##  7 1      0
##  8 1      0
##  9 0      0
## 10 1      1
## # ... with 459 more rows
```

Machine learning in R

Calculating metrics (out-of-sample)

```
precision <- bind_cols(data_testing, preds) %>% precision(truth=degree,
recall <- bind_cols(data_testing, preds) %>% recall(truth=degree, estim
print(bind_rows(precision, recall))
```

```
## # A tibble: 2 x 3
##   .metric   .estimator .estimate
##   <chr>     <chr>          <dbl>
## 1 precision binary         0.766
## 2 recall    binary         0.912
```

Machine learning in R

Calculating metrics: Predicted probabilities

We can also extract the predicted probabilities by adding an argument to the `predict` function.

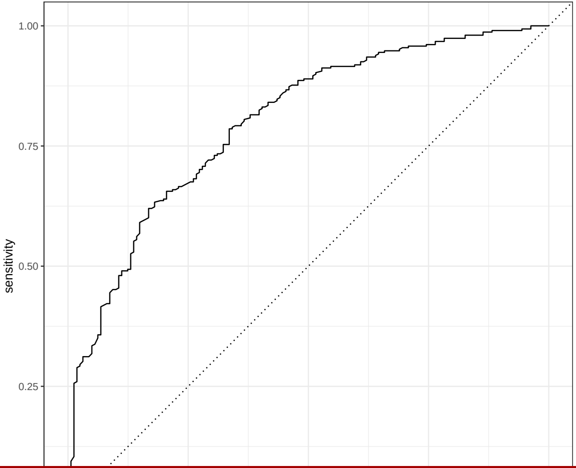
```
probs <- rf %>%
  predict(data_testing, type = "prob") %>%
  bind_cols(data_testing)
head(probs %>% select(.pred_0, .pred_1, degree))
```

```
## # A tibble: 6 x 3
##   .pred_0 .pred_1 degree
##     <dbl>   <dbl> <fct>
## 1   0.428   0.572 1
## 2   0.266   0.734 1
## 3   0.721   0.279 0
## 4   0.96    0.04  0
## 5   0.736   0.264 0
## 6   0.227   0.773 0
```

Machine learning in R

Calculating metrics: ROC

```
probs %>% roc_curve(degree, .pred_0) %>% autoplot()
```



Machine learning in R

Calculating metrics: AUC

```
probs %>% roc_auc(degree, .pred_0)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.800
```

Machine learning in R

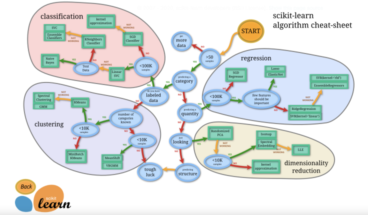
Next week

- We will go into more depth using `tidymodels` to implement cross-validation and a hyperparameter search and will evaluate multiple different models
- Supervised machine learning to perform text classification
- Considering how the inputs and label quality affect classifier performance

Machine learning in R

Alternatives

- Python has a more developed ML ecosystem than R.
  - `scikit-learn` provides a suite of tools for most machine-learning tasks except deep-learning, which requires specialized libraries.



Source: scikit-learn documentation. See this tutorial for how to run `scikit-learn` using R.