Computational Social Science

Supervised Machine Learning

Dr. Thomas Davidson

Rutgers University

March 28, 2024

Plan

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

Course updates

Homework

- Homework 3 was due yesterday
- Homework 4 on supervised machine learning released next week

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.

Pre-Process → Train → Validate

rsample recipes parsnip yardstick

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/2018_gss_sample.csv")

table(data$degree)
```

```
##
##    0    1
## 1568  710
```

Machine learning in R

Data cleaning

```r
colnames(data)
```

```
##  [1] "age"       "sex"       "race"      "sibs"      "paeduc"    "maeduc"
##  [7] "family16"  "fund16"    "incom16"   "relig16"   "mawrkgrw"  "othlang
## [13] "born"      "parborn"   "granborn"  "zodiac"    "degree"
```

```r
data <- data %>%
  mutate(across(-c(age, sibs), as.factor))
```

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.

```r
set.seed(987123)
data_split <- initial_split(data, prop = 0.8)
print(data_split)
```

```
## <Training/Testing/Total>
## <1822/456/2278>
```

Viewing the traing data

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

```
## # A tibble: 6 x 17
##     age sex   race   sibs paeduc maeduc family16 fund16 incom16 reli
##   <dbl> <fct> <fct> <dbl> <fct>  <fct>  <fct>    <fct>  <fct>   <fct
## 1    86 0     1         4 12     12     1        1      3       1
## 2    34 1     1         1 14     14     1        2      3       2
## 3    22 0     2         8 12     18     5        2      1       1
## 4    58 0     1         4 8      12     1        2      5       2
## 5    68 1     1         4 5      12     1        3      3       1
## 6    43 0     1        12 0      0      1        2      1       2
## # i 6 more variables: othlang <fct>, born <fct>, parborn <fct>, gran
## #   zodiac <fct>, degree <fct>
```

Machine learning in R

Pre-processing using `recipe`

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

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

data_recipe
```

Machine learning in R

Extracting data from `recipe`

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 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

Fitting 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)

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

```
## # A tibble: 6 x 2
##   degree .pred_class
##   <fct>  <fct>
## 1 0      0
## 2 0      0
## 3 0      0
## 4 1      1
## 5 0      0
## 6 0      0
```

Machine learning in R

Calculating metrics (in-sample)

```
precision <- train_preds %>% precision(truth=degree, estimate = .pred_c
recall <- train_preds %>% recall(truth=degree, estimate = .pred_class)
print(bind_rows(precision, recall))
```

```
## # A tibble: 2 x 3
##   .metric   .estimator .estimate
##   <chr>     <chr>          <dbl>
## 1 precision binary         0.952
## 2 recall    binary         0.998
```

Machine learning in R

Making predictions (out-of-sample)

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

Machine learning in R

Calculating metrics (out-of-sample)

```
precision <- test_preds %>% precision(truth=degree, estimate = .pred_cl
recall <- test_preds %>% recall(truth=degree, estimate = .pred_class)
print(bind_rows(precision, recall))
```

```
## # A tibble: 2 x 3
##   .metric   .estimator .estimate
##   <chr>     <chr>          <dbl>
## 1 precision binary         0.765
## 2 recall    binary         0.925
```

Machine learning in R

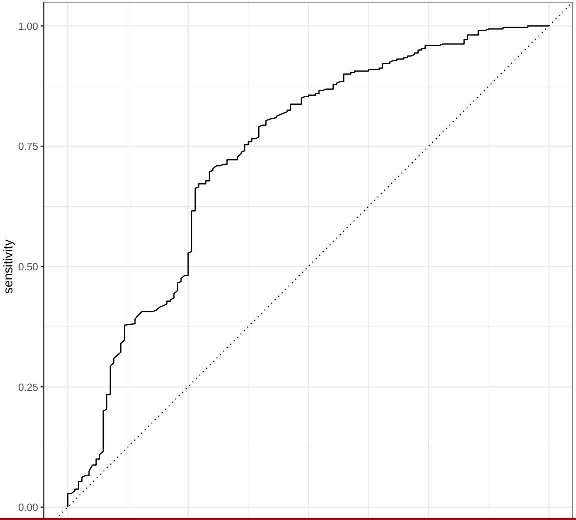
Calculating metrics: Predicted probabilities

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(degree, .pred_0, .pred_1) %>% bind_cols(preds))
```

```
## # A tibble: 6 x 4
##   degree .pred_0 .pred_1 .pred_class
##   <fct>    <dbl>   <dbl> <fct>
## 1 1        0.595   0.405 0
## 2 0        0.899   0.101 0
## 3 0        0.527   0.473 0
## 4 1        0.346   0.654 1
## 5 1        0.41    0.59  1
## 6 0        0.653   0.347 0
```

Calculating metrics: ROC



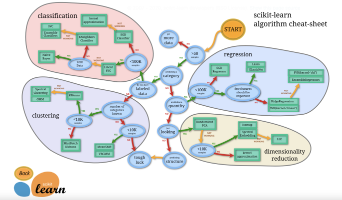
Calculating metrics: AUC

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

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.734
```

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.

Machine learning in R

Next week

- Supervised machine learning to perform text classification
- Cross-validation, parameter searches, and model comparison
- Data quality and predictive performance