Lab 6: Introduction to Machine Learning in R

2024-05-09

What is 'Machine Learning'?

It's not AI...

Machine Learning is a field of mathematics that involves training a model on data in order to predict unseen observations

Lecture 13 from class goes over more details about machine learning. We are going to be focused in how we do it in R.

We are going to focus on decision trees.

Decision Trees

The idea (as you discussed in lecture) is that the model separates the predictor variables into a certain number of groups, and then predicts outcomes using the most common "value" or "class" in that group.

```
library(pacman)
p_load(tidyverse, dplyr, janitor, tidymodels)

# Download data
data = iris %>% clean_names() # already in R
  # "clean_names()" is from the "janitor" packages


# Set seed
set.seed(213565)


# Split into 75/25 training/testing
train_test_split = data %>% initial_split(prop = 0.75) # from "tidymodels" package

data_train = train_test_split %>% training()
data_test = train_test_split %>% testing()

  # Cross Validation Data (into 2 groups)
data_cv = data_train %>% vfold_cv(v = 2)

# Create Recipe
  # prepares our data for training
data_recipe = data_train %>%
```

```
                recipe(species ~.)

  # "clean" recipe to allow for training
data_clean = data_recipe %>%
                prep() %>%
    # tells it to estimate the required parameters from a training set
                juice()
    # applies all operations to the training set

# NOTE: you need both "data_recipe" and "data_clean" as objects



# Define the Decision Tree
model_tree = decision_tree(
    mode = "classification",
      # we are going to be predictions the species, not a number
    cost_complexity = tune(),
    tree_depth = tune(),
      # these are variables we will tune with ML
    min_n = 5) %>%
      # the minimum number of observations in node required before another "split"
  set_engine("rpart")
      # "rpart" fits a model as a set of if/then statements


# Define workflow
tree_workflow = workflow() %>%
    # gives the order in which things happen
  add_model(model_tree) %>%
    # the decision tree model from abovbe
  add_recipe(data_recipe)
    # the recipe


# Tune (this one takes a while)
tree_tune_fit = tree_workflow %>%
      # take the workflow...
  tune_grid(
    data_cv, # training data set
    grid = expand_grid(
      cost_complexity = seq(0, 10, by = 0.5),
      tree_depth = seq(1, 10, by = 1)
    ),
      # all possible sizes of tree "depth" and "complexity"

    metrics = metric_set(accuracy)
      # metric for performance
  )


# Finalize Workflow for plotting and predicting
best_flow = tree_workflow %>%
```

```
      # again, take workflow...
  finalize_workflow(select_best(tree_tune_fit,
                    metric =  "accuracy")) %>%
      # select the "best" performing model according to accuracy
  fit(data = data_clean)
      # fit to the data!



# Extract fitted model
best_tree = best_flow %>% extract_fit_parsnip()
```

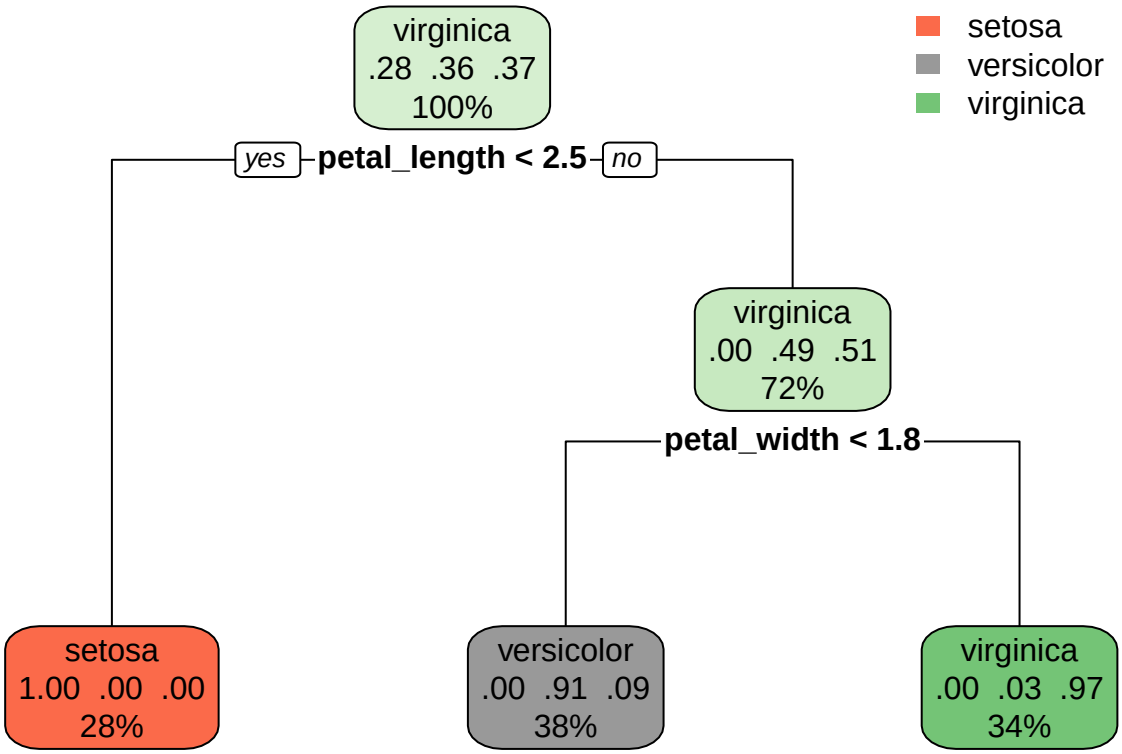
Predicting Flower Species

Plot our tree

```
p_load(rpart.plot)

# Plot the decision tree
best_tree$fit %>% rpart.plot()
```

```
## Warning: Cannot retrieve the data used to build the model (model.frame: object '..y' not found).
## To silence this warning:
##     Call rpart.plot with roundint=FALSE,
##     or rebuild the rpart model with model=TRUE.
```



Predict on Testing Data

```
# Predict onto test data
y_hat = best_flow %>%
          predict(new_data = data_test,
                  type = "class") # recall bc "species" is a class, not a number

# Display results:

tree_results = conf_mat( # called a "confusion matrix"
  data = tibble(
    y_hat = y_hat %>% unlist(), # get predicted values
    y = data_test$species # get actual values
  ),
  truth = y, estimate = y_hat
)

tree_results
```

```
##             Truth
## Prediction   setosa versicolor virginica
##   setosa         19          0         0
##   versicolor      0         10         1
##   virginica       0          0         8
```

That's pretty good!