Midterm III Study Guide and Review

AJ Staff

March 08 2022

Exam III Study Guide

Sample Midterm

Q1. Briefly describe what the following set of codes do. Why do we need to split the data into training and test set?

```r
data_Smarket <- Smarket
split <- initial_split(data_Smarket, strata = Direction, prop = 4/5)
Smarket_train <- training(split)
Smarket_test <- testing(split)
```

Answer: The data set is loaded in, then feeded into `initial_split` which divides the data set into a training set and a test set. Since `prop` is also specified, this means that the first 4/5 samples for training.

Q2. Evaluate the following trained model to produce a data-frame of the actual and predicted `Direction` in the test dataset. Call this data-frame `Smarket_results`.

```r
Smarket_recipe <- recipe(Direction ~ Lag1 + Lag2 + Lag3 + Year + Volume, data = Smarket_train) %>%
  step_center(all_predictors()) %>%
  step_scale(all_predictors()) %>%
  prep()
Smarket_knn_spec <- nearest_neighbor(mode = "classification",
                             engine = "kknn",
                             weight_func = "rectangular",
                             neighbors = 5)
Smarket_workflow <- workflow() %>%
  add_recipe(Smarket_recipe) %>%
  add_model(Smarket_knn_spec)
Smarket_fit <- fit(Smarket_workflow, data = Smarket_train)
```

```r
tree_fit <- Smarket_fit %>%
extract_fit_parsnip()

tree_fit
## parsnip model object
##
## Fit time:  52ms
##
## Call:
## kknn::train.kknn(formula = ..y ~ ., data = data, ks = min_rows(5,     data, 5), kernel = ~"rectangula
##
## Type of response variable: nominal
## Minimal misclassification: 0.4854855
## Best kernel: rectangular
## Best k: 5
```

Q3. Construct a confusion matrix from the prediction results from Q2. Calculate by hand the sensitivity, specificity, accuracy, and positive predictive value of the classifier.

Answer:

Sensitivity: True Positives/(True Positives+False Negatives)= 72/(72+58) = .554 Specificity: True Negatives/(True Negatives+False Positives)= 50/(50+71) = .413 Accuracy: = True Positives + True Negatives / (True Positives + True Negatives + False Positives + False Negatives) = (72+50)/(72+50+71+58) = .486

```
tree_last_fit <- Smarket_workflow %>%
last_fit(split)

tree_predictions <- tree_last_fit %>% collect_predictions()
conf_mat(tree_predictions, truth = Direction, estimate = .pred_class)
##           Truth
## Prediction Down Up
##       Down   50 58
##       Up     71 72
```

Q4. Now, run a 10-fold cross validation using appropriate functions in `tidymodels` package by adding `tune()` as a placeholder for the number of neighbors. Find an optimal number of nearest neighbor. What metrics did you base your conclusion on?

```
Smarket_folds <- vfold_cv(Smarket_train, v = 10, strata = Direction)
tree_grid <- grid_regular(cost_complexity(),
                          tree_depth(),
                          min_n(),
                          levels = 2)
```

Q5. Give your answers to the following set of problems.

a. Explain the difference between unsupervised learning and supervised learning.

Answer: It has partially to do with the labeling of the data. Unsupervised learning is when the data is unlabeled, and supervised is when it is pre-labeled.

b. Explain how you can use total within cluster sum of squares to find the "best" choice of K in a K-means clustering algorithm.

Answer: The total within-cluster variation is the sum of squared Euclidean distances between items, and by calculating distances from the centroid, we are able to iteratively find the "best" arrangment.

c. Suppose your friend wants to design an app that allows the user to set a number (x) between 1 and 1000, and displays it's logarithm (base 10). His trial produced an error. Modify the following code to rectify the error.

```
library(shiny)
ui <- fluidPage(
  sliderInput("x", label = "If x is", min = 1, max = 1000, value = 100),
  "then the logarithm of x is",
  textOutput("logarithm")
)
server <- function(input, output, session) {
  output$logarithm <- renderText({
    log10(input$x)
  })
}
shinyApp(ui, server)
```

d. Briefly explain why do we preprocess data in k nearest neighbors algorithm.

Answer: This allows the data to be more suitable for a classifier, and thus use our observed data to make predictions.

e. What does the following chunk of code do?

```
table_covid_cases <- bow(url = "https://usafacts.org/visualizations/coronavirus-covid-19-spread-map/sta
  scrape() %>%
  html_elements(css = "table") %>%
  html_table()
```

Answer: Scrapes the table at the bottom of that link which is covid spread data on the state of Minnesota.