MATH 154 - HW8 - k-NN & Trees

Ra-Zakee Muhammad

due: Thursday, November 4, 2021

Amber Helped me with all problems on this hw set

assignment

1. **Pod Q** Describe one thing you learned from someone in your pod this week (it could be: content, logistical help, background material, R information, something fun, etc.) 1-3 sentences.

I learned from Brian that he has a business for water accesibility.

2. **Exam 1** Look at the solutions to both the in-class and take-home parts of the exam. Explain (provide a little bit of detail) one idea that you understand now but did not understand at the time of the test.

One idea that i understand now but didnt before was that a failiur to reject the null hypothesis doesnt necesarily indicate that the null hypothesis is true because a hypothesis test is only a measurement on the data but not the null hypothesis itself. This was a problem that came up in the take home problem 2.

3. **CV** Problem 3 in section 5.4 of *An Introduction to Statistical Learning*, https://web.stanford.edu/~hastie/ISLR2/ISLRv2_website.pdf. [Hint: see the text and the class notes for reminders.]

We now review V-fold cross-validation.

(a) Explain how V-fold cross-validation is implemented.

Randomly divide the data into even groups, then repeat the following procedure times: (1) choose one of the groups of data to be the validation set (2) train the model with the remaining − 1 folds (3) calculate the error rate (MSE or test error rate, for example) on the validation set. The estimate for the test error is the average of the error rates.

(b) What are the advantages and disadvantages of V-fold cross validation relative to:

i. The validation set approach? (i.e., a "test" set)

(The validation set approach is a train/test split.) Advantages: The test error rate can have high variance depending how the data are split ((observations going to build the model versus observations used to validate the model). The variablility of k-fold CV is less than the variability of the validation set approach. Disadvantages: Computationally more expensive.

ii. LOOCV? (leave one out cross validation - or n-fold CV)

Advantages: k-fold CV is at a sweet(er) spot on the bias-variance tradeoff. LOOCV has low bias but suffers from high variance because it averages many highly correlated quantities. This is also less computationally expensive than LOOCV. Since LOOCV is a special case of k-fold CV, I'm not sure if there are disadvantages of k-fold validation in relation to LOOCV.

4. **OJ and CART** Problem 9 in section 8.4 of *An Introduction to Statistical Learning*, http://www-bcf.usc.edu/~gareth/ISL/. Although some parts have been adjusted to coincide with the syntax we are using in class.

This problem uses the `OJ` data which is available in the **ISLR** package.

Hint 1: use the **tidymodels** syntax we've covered in class (slides, notes, etc.). Don't use the base R syntax in the text.

Hint 2: you'll probably want to set a seed early on (or often?) so that you can report on the output you need.

Hint 3: in the console, type `?OJ` to understand the dataset

(a) Create a training set containing a random sample of 800 observations (about 75%), and a test set containing the remaining observations.

```
set.seed(47074)
OJ_split <- initial_split(OJ, prop = 0.75)
OJ_train <- training(OJ_split)
OJ_test <- testing(OJ_split)
```

(b) Fit a tree to the training data, with `Purchase` as the response and the other variables as predictors. Use the `metrics()` function to produce summary statistics about the tree, and describe the results obtained. What is the training error rate? How many terminal nodes does the tree have?

```
# recipe
OJ_cart_recipe <-  recipe(Purchase ~ ., data = OJ_train)

# model
OJ_cart_model <- decision_tree() %>%
set_engine("rpart") %>%
set_mode("classification")

# workflow
OJ_cart_wflow <-  workflow() %>%
add_model(OJ_cart_model) %>%
add_recipe(OJ_cart_recipe)

# fit
OJ_cart_fit <- fit(OJ_cart_wflow, OJ_train)

OJ_cart_fit
```

```
## == Workflow [trained] ==========================================================
## Preprocessor: Recipe
## Model: decision_tree()
##
```

```
## -- Preprocessor ----------------------------------------------------------------
## 0 Recipe Steps
##
## -- Model ----------------------------------------------------------------------
## n= 802
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 802 325 CH (0.594763 0.405237)
##    2) LoyalCH>=0.48285 494  84 CH (0.829960 0.170040)
##      4) LoyalCH>=0.7057 291  16 CH (0.945017 0.054983) *
##      5) LoyalCH< 0.7057 203  68 CH (0.665025 0.334975)
##       10) PriceDiff>=0.085 132  26 CH (0.803030 0.196970) *
##       11) PriceDiff< 0.085 71  29 MM (0.408451 0.591549)
##         22) ListPriceDiff>=0.235 23   8 CH (0.652174 0.347826) *
##         23) ListPriceDiff< 0.235 48  14 MM (0.291667 0.708333) *
##    3) LoyalCH< 0.48285 308  67 MM (0.217532 0.782468) *
```

```
# metrics
OJ_cart_fit %>%
  predict(new_data = OJ_train) %>%
  cbind(OJ_train) %>%
  metrics(truth = Purchase, estimate = .pred_class)
```

```
## # A tibble: 2 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary     0.8366584
## 2 kap      binary     0.6662177
```

```
OJ_cart_fit %>%
  predict(new_data = OJ_train) %>%
  cbind(OJ_train) %>%
  select(Purchase, .pred_class) %>%
  table()
```

```
##         .pred_class
## Purchase  CH  MM
##       CH 396  81
##       MM  50 275
```

some think kap is more robust than symbol accuracy

confusion matrix

```
OJ_cart_fit %>%
predict(new_data = OJ_train) %>% cbind(OJ_train) %>% select(Purchase, .pred_class) %>% table()
```

```
##         .pred_class
## Purchase  CH  MM
##       CH 396  81
##       MM  50 275
```

(c) Type in the name of the fit object in order to get a detailed text output. Pick one of the terminal nodes, and interpret the information displayed.

```
OJ_cart_fit[[2]]$fit
```

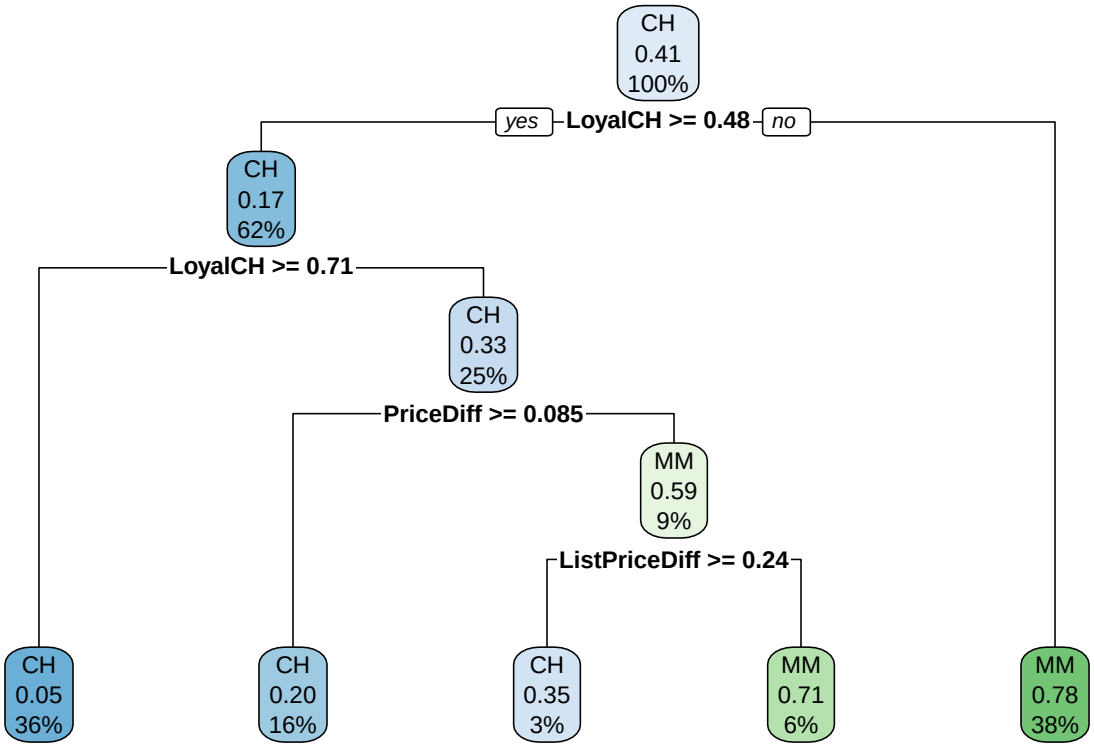
```
## parsnip model object
##
## Fit time:  27ms
## n= 802
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 802 325 CH (0.594763 0.405237)
##    2) LoyalCH>=0.48285 494  84 CH (0.829960 0.170040)
##      4) LoyalCH>=0.7057 291  16 CH (0.945017 0.054983) *
##      5) LoyalCH< 0.7057 203  68 CH (0.665025 0.334975)
##       10) PriceDiff>=0.085 132  26 CH (0.803030 0.196970) *
##       11) PriceDiff< 0.085 71  29 MM (0.408451 0.591549)
##         22) ListPriceDiff>=0.235 23   8 CH (0.652174 0.347826) *
##         23) ListPriceDiff< 0.235 48  14 MM (0.291667 0.708333) *
##    3) LoyalCH< 0.48285 308  67 MM (0.217532 0.782468) *
```

We interpret terminal node (4). Observations with loyalty to Citrus Hill greater than 0.7057 will purchase from Citrus Hill.

(d) Create a plot of the tree, and interpret the results. (Code is completely written for this part.)

```
library(rpart.plot)

OJ_cart_plot <- OJ_cart_fit %>% extract_fit_parsnip()
rpart.plot( OJ_cart_plot$fit, roundint = FALSE)
```



Loyalty to Citrus Hill is the first variable that the tree splits on, so it is one of the more important variables. If the loyalty to Citrus Hill is less than 0.48, then the customer will buy Minute Maid. If the loyalty is greater than 0.71, then the price difference and list price difference determine the purchasing between Citrus Hill and Minute Maid. If the price difference is greater than 0.085 and the list price difference is greater than 0.24, then Minute Maid will be purchased. Otherwise, the customer will purchase Citrus Hill.

(e) Predict the response on the test data, and produce a confusion matrix comparing the test labels to the predicted test labels. What is the test error rate?

```
OJ_cart_fit %>%
predict(new_data = OJ_test) %>% # predict with test data
  cbind(OJ_test) %>% # check accuracy
metrics(truth = Purchase, estimate = .pred_class)
```

```
## # A tibble: 2 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary     0.8134328
## 2 kap      binary     0.6046731
```

```
# test error rate is 1 - accuracy
# confusion matrix
OJ_cart_fit %>%
predict(new_data = OJ_test) %>% cbind(OJ_test) %>% select(Purchase, .pred_class) %>% table()
```

```
##          .pred_class
## Purchase  CH  MM
##       CH 142  34
##       MM  16  76
```

The test error rate is 18.65%.

(f) Tune the model using 10-fold cross validation to find the optimal tree depth. Try trees of depth ranging from 1 to 5.

```
set.seed(47074) # create folds
tune_vfold <- vfold_cv(OJ_train, v = 10)
# vector of tree depths
# grid is for parameters
tune_grid <- expand.grid(tree_depth = c(1, 2, 3, 4, 5))
# model
tune_cart_model <- decision_tree(tree_depth = tune()) %>% set_engine("rpart") %>%
set_mode("classification")
# workflow
tune_wflow <- workflow() %>% add_model(tune_cart_model) %>% add_recipe(OJ_cart_recipe) # initial recipe
# tuning the workflow with the vfold and the parameter grid
tuned_cart <- tune_wflow %>%
  tune_grid(resamples = tune_vfold, grid = tune_grid) # when do we plug in the dataset?
# best value of tree depth
tuned_cart %>% collect_metrics() %>% filter(.metric == "accuracy")
```

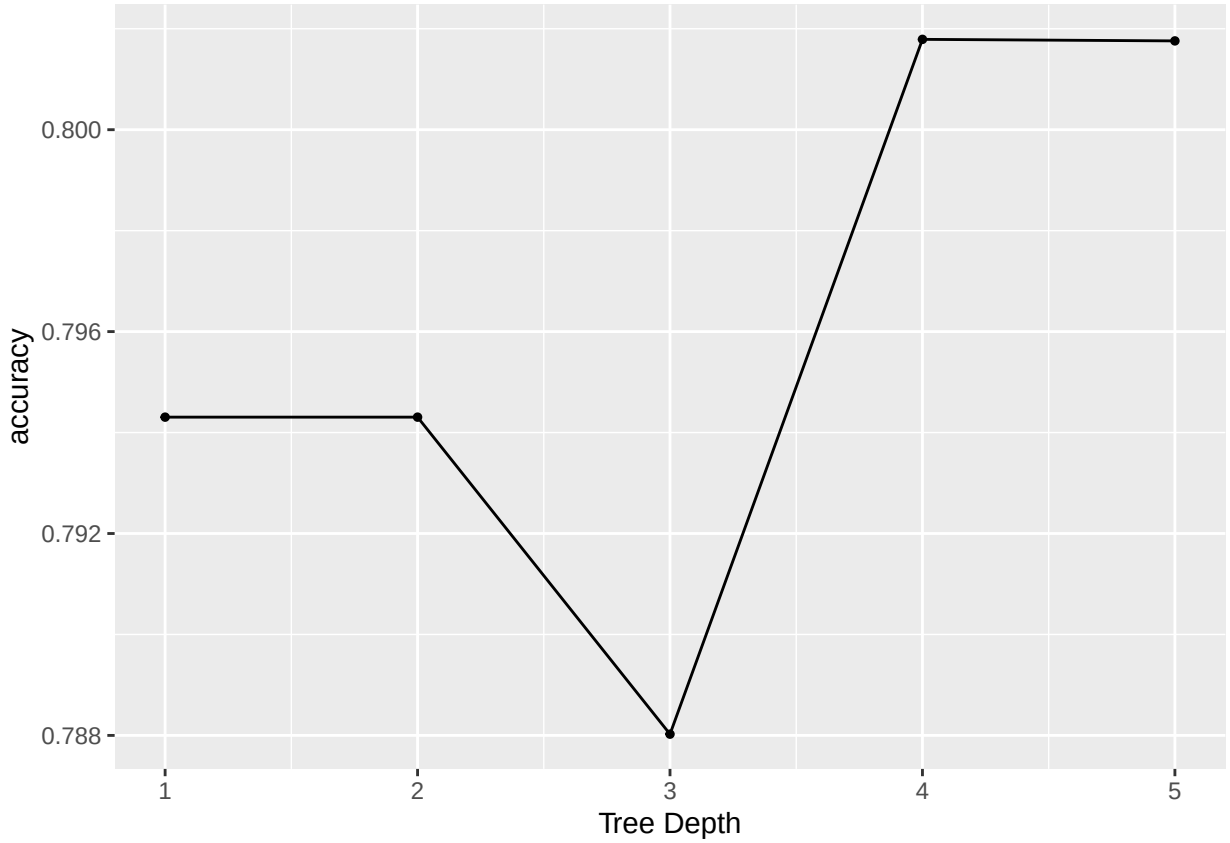
```
## # A tibble: 5 x 7
##   tree_depth .metric  .estimator      mean     n    std_err .config
##        <dbl> <chr>    <chr>          <dbl> <int>      <dbl> <chr>
## 1          1 accuracy binary     0.7943056    10 0.01662477 Preprocessor1_Model1
## 2          2 accuracy binary     0.7943056    10 0.01662477 Preprocessor1_Model2
## 3          3 accuracy binary     0.7880247    10 0.01329906 Preprocessor1_Model3
## 4          4 accuracy binary     0.8017901    10 0.01129478 Preprocessor1_Model4
## 5          5 accuracy binary     0.8017593    10 0.01221027 Preprocessor1_Model5
```

```
 tuned_cart %>% select_best("accuracy")
```

```
## # A tibble: 1 x 2
##   tree_depth .config
##        <dbl> <chr>
## 1          4 Preprocessor1_Model4
```

(g) Produce a plot with tree size on the x-axis and cross-validated classification error rate on the y-axis. Code is written completely.

```
 tuned_cart %>%
autoplot(metric = "accuracy")
```



(h) Which tree size corresponds to the lowest cross-validated classification error rate?

A tree with depth 4.

(i) Produce a pruned tree corresponding to the optimal tree size obtained using cross-validation.

```
# pruned model with tree_depth explicitly set, not tuned
pruned_cart_model <- decision_tree(tree_depth = 3) %>% set_engine("rpart") %>%
set_mode("classification")
# wflow is (pruned) model and same old recipe
pruned_wflow <- workflow() %>% add_model(pruned_cart_model) %>% add_recipe(OJ_cart_recipe)
# fit the workflow with the train data
pruned_fit <- fit(pruned_wflow, OJ_train)
# metrics for training data
pruned_fit %>%
predict(new_data = OJ_train) %>%
cbind(OJ_train) %>%
metrics(truth = Purchase, estimate = .pred_class)
```

```
## # A tibble: 2 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary     0.8279302
## 2 kap      binary     0.6522444
```

```
pruned_fit %>%
predict(new_data = OJ_train) %>% cbind(OJ_train) %>% select(Purchase, .pred_class) %>% table()
```

```
##          .pred_class
## Purchase  CH  MM
##       CH 381  96
##       MM  42 283
```

(j) Compare the training error rates between the pruned and unpruned trees. Which is higher?

Pruned tree training error rate is 1-0.82793=17.2%. Unpruned tree training error rate is 16.33%. Pruned training error rate is higher because the best tree depth, for my random seed to split the data, is actually 4.

(k) Compare the test error rates between the pruned and unpruned trees. Which is higher?

```
# unpruned
OJ_cart_fit %>%
predict(OJ_test) %>%
cbind(OJ_test) %>%
metrics(truth = Purchase, estimate = .pred_class)
```

```
## # A tibble: 2 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary     0.8134328
## 2 kap      binary     0.6046731
```

```
# pruned
pruned_fit %>%
predict(OJ_test) %>%
cbind(OJ_test) %>%
metrics(truth = Purchase, estimate = .pred_class)
```

```
## # A tibble: 2 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary     0.7985075
## 2 kap      binary     0.5793023
```

5. **OJ and k-NN** Using the same test and training data from the previous problem, use k-NN to classify the OJ data

(a) Using cross validation, fit models for each of k=1, 5, 11. (Which predictor variables will you use?)

```
# recipe, normalize because knn
knn_recipe <- recipe(Purchase ~ ., data = OJ_train) %>% step_normalize(all_numeric_predictors())
# tune_vfold
# update parameter grid
tune_grid <- data.frame(neighbors = c(1, 5, 11)) # parameter neighrs is k
# model
```

```
knn_tune <- nearest_neighbor(neighbors = tune()) %>% set_engine("kknn") %>%
set_mode("classification")
knn_tune_wflow <- workflow() %>% add_model(knn_tune) %>% add_recipe(knn_recipe)
knn_tune_wflow %>% tune_grid(resamples = tune_vfold,
grid = tune_grid) %>% collect_metrics() %>%
filter(.metric == "accuracy")
```

```
## # A tibble: 3 x 7
##   neighbors .metric  .estimator      mean     n     std_err .config
##       <dbl> <chr>    <chr>          <dbl> <int>       <dbl> <chr>
## 1         1 accuracy binary     0.7432407    10 0.01409977  Preprocessor1_Model1
## 2         5 accuracy binary     0.7656173    10 0.01090229  Preprocessor1_Model2
## 3        11 accuracy binary     0.7730710    10 0.009251404 Preprocessor1_Model3
```

(b) Which value of k gave the most accurate predictions for the Purchase variable? For that k, find the accuracy of the training model fit to the test data.

The most accurate predictions came from = 11. The accuracy is 1-0.773 = 22.7%.

(c) In part (a) you were required to adjust the explanatory variables in a way that was not required for the CART classification. What was the adjustment and why was it necessary?

We needed to first normalize the numeric predictor variables. k-NN uses the Euclidean distance, which is not robust to scale. For example, the same variable in units miles, feet, and inches would result in different regions drawn from kNN.