Assignment 001

Tom Ben-Shahar

February 13, 2025

Setup

```r
# Load Packages
library(pacman)
p_load(tidyverse, readr, janitor, broom, tidymodels, magrittr, kknn, recipes)
library(tidymodels)

# Load Data
train <- read_csv("data/train.csv")
```

```
## Rows: 1460 Columns: 81
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (43): MSZoning, Street, Alley, LotShape, LandContour, Utilities, LotConf...
## dbl (38): Id, MSSubClass, LotFrontage, LotArea, OverallQual, OverallCond, Ye...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
test <- read_csv("data/test.csv")
```

```
## Rows: 1459 Columns: 80
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (43): MSZoning, Street, Alley, LotShape, LandContour, Utilities, LotConf...
## dbl (37): Id, MSSubClass, LotFrontage, LotArea, OverallQual, OverallCond, Ye...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
# Clean Data
train <- clean_names(train, case = 'big_camel')
test <- clean_names(test, case = 'big_camel')

train <- train %>% mutate(across(where(is.character), as.factor))
test <- test %>% mutate(across(where(is.character), as.factor))

test %<>% mutate(SalePrice = NA)
```

Model Specification

```r
# Create first model
# Create Recipe
#setdiff(names(test), names(train))
rec_1 <- recipe(SalePrice ~ ., data = train)

# Define Processing Steps
rec_1 <- rec_1 %>%
  # Mean imputation for numeric predictors
  step_impute_mean(all_predictors() & all_numeric()) %>%
  # KNN imputation for categorical predictors
  step_impute_knn(all_predictors() & all_nominal(), neighbors = 5) %>%
  # Create dummies for categorical variables
  step_dummy(all_predictors() & all_nominal()) %>%
  step_impute_mean(all_numeric_predictors()) %>%
  step_impute_mode(all_nominal_predictors())

# Apply processing to data
#train_1 <- rec_1 %>% prep() %>% juice()
# test_1 <- bake(proc_1, new_data = test)  # Go to office hours and ask Andrew why this isnt working ;

# Create regression model
reg_1 <- linear_reg() %>%
  set_engine("lm")
  # fit(SalePrice ~. , data = train_1)

# Predict
# test_1 <- test %>%
#   mutate(SalePrice = predict(reg_1, new_data = test))

# Workflow
workflow_1 =
  workflow() %>%
  add_model(reg_1) %>%
  add_recipe(rec_1) %>%
  fit(data = train)

test_hat_1 = workflow_1 %>% predict(new_data = test)
mod_out_1 <- test %>%
  mutate(SalePrice = test_hat_1$.pred) %>%
  select("Id", "SalePrice")

# Create Second Model
mod_2 <- nearest_neighbor(neighbors = 5) %>%
  set_engine("kknn") %>%
  set_mode("regression")

workflow_2 <- workflow() %>%
  add_model(mod_2) %>%
  add_recipe(rec_1) %>%
  fit(data = train)
```

```
test_hat_2 = workflow_2 %>%  predict(new_data = test)
mod_out_2 <- test %>%
  mutate(SalePrice = test_hat_2$.pred) %>%
  select("Id", "SalePrice")

# Write model outputs
write.csv(file = "data/model_1.csv", mod_out_1, row.names = FALSE) # Model 1 wins, placing me at 2848 o
write.csv(file = "data/model_2.csv", mod_out_2, row.names = FALSE)
```

Cross Validation

```
set.seed(65198153)

# Assign folds
cv_sample <- sample(rep(1:5, length.out = as.integer(nrow(train))))
train$fold <- cv_sample

train %<>% mutate(id = Id)

# Initialize dataframe for predictions
#cv_test_hat_1 <- train %>% select(id)  # Keeps IDs for merging later

for (i in 1:5) {
  train_fold <- train %>% filter(fold != i)
  test_fold <- train %>% filter(fold == i)

  #print(class(test_fold)) # Should be "data.frame" or "tbl_df"
  #print(str(test_fold))    # Should show structured columns


  # Print debugging info
  #print(paste("Fold:", i, "Train size:", nrow(train_fold), "Test size:", nrow(test_fold)))

  # Skip if no test data
  if (nrow(test_fold) == 0) {
    print(paste("Skipping fold", i, "as test set is empty"))
    next
  }

  # Fit models
  workflow_i_model_1 <- workflow() %>%
    add_model(reg_1) %>%
    add_recipe(rec_1) %>%
    fit(data = train_fold)

  workflow_i_model_2 <- workflow() %>%
    add_model(mod_2) %>%
    add_recipe(rec_1) %>%
    fit(data = train_fold)

  # Generate column name
  col_name_1 <- paste0("mod_1_fold_", i)
  col_name_2 <- paste0("mod_2_fold_", i)

  # Predict and store results
  preds_1 <- workflow_i_model_1 %>% predict(new_data = test_fold) %>% mutate(id = test_fold$id)
  preds_2 <- workflow_i_model_2 %>% predict(new_data = test_fold) %>% mutate(id = test_fold$id)

  preds_1 %<>% rename(!!col_name_1 := .pred)
  preds_2 %<>% rename(!!col_name_2 := .pred)

  #print(head(preds))  # Debug: Check predictions structure

  #cv_test_hat_1[[col_name]] <- preds %>% pull(.pred)  # Extract predictions

  train <- train %>%
    left_join(preds_1 %>% select(id, !!col_name_1), by = "id") %>%
    #rename(!!col_name_1 := .pred) %>%
    left_join(preds_2 %>% select(id, !!col_name_2), by = "id")# %>%
    #rename(!!col_name_2 := .pred)

}

# Combine columns
train <- train %>%
  mutate(
    mod_1_pred = case_when(
      fold == 1 ~ mod_1_fold_1,
      fold == 2 ~ mod_1_fold_2,
      fold == 3 ~ mod_1_fold_3,
      fold == 4 ~ mod_1_fold_4,
      fold == 5 ~ mod_1_fold_5
    ),
    mod_2_pred = case_when(
      fold == 1 ~ mod_2_fold_1,
      fold == 2 ~ mod_2_fold_2,
      fold == 3 ~ mod_2_fold_3,
      fold == 4 ~ mod_2_fold_4,
      fold == 5 ~ mod_2_fold_5
    )
  )

#not sure where the .x and .y stuff came from...
```

RMSE

```
rmse_df <- data.frame(mod1 = 1:5, mod2 = 1:5)

for (i in 1:5) {
  test_fold <- train %>% filter(fold == i)
  rmse_df$mod1[i] <- rmse(test_fold, truth = SalePrice, estimate = mod_1_pred) %>% pull(.estimate)
  rmse_df$mod2[i] <- rmse(test_fold, truth = SalePrice, estimate = mod_2_pred) %>% pull(.estimate)
}

avg_rmse_df_1 <- mean(rmse_df$mod1)
avg_rmse_df_2 <- mean(rmse_df$mod2)
sd_rmse_df_1 <-sd(rmse_df$mod1)
sd_rmse_df_2 <-sd(rmse_df$mod2)

avg_rmse_df_1
```

```
## [1] 43121.15
```

```
avg_rmse_df_2
```

```
## [1] 45796.65
```

```
sd_rmse_df_1
```

```
## [1] 15524.71
```

```
sd_rmse_df_2
```

```
## [1] 5046.304
```

7 Model 1 RMSE < Model 2 RMSE

Model 1 RMSE SD > Model 2 RMSE SD

8 It thus appears that model 1 (LM) is superior, as it has lower RMSE.

9 Cross validation gives a more robust account of model performance.

10 Cross validation is more reliable as it tests several validation sets, not just one.

11 While model 1 has lower RMSE, it has higher variability. When I ran this with a different seed, Model 1 underperformed. This validation method is thus not necessarily pointing at the ideal model.