STATS 7022 - Data Science PG Assignment 2

Dang Thinh Nguyen

2024-07-10

Question 1: Data Analysis

1 Load libraries

```
pacman::p_load(tidyverse, bookdown, readr, dplyr, tidymodels, pROC)
```

2 Load data

```
# Read in the data
data <- readRDS('./diamonds.rds')

# Display the first 10 lines of the data
data %>% head(10)
```

```
## # A tibble: 10 x 8
##    carat c.grade depth table price     x     y volume
##    <dbl> <chr>   <dbl> <dbl> <int> <dbl> <dbl>  <dbl>
##  1  0.52 2Y4      61.3    56  1743  5.21  5.17   85.7
##  2  0.31 1Z3      59.7    58   788  4.4   4.45   51.7
##  3  0.94 2X4      61.2    57  7120  6.29  6.33   NA
##  4  0.7  1W4      61.7    58  2317  5.68  5.73  115.
##  5  2.13 2U6      62.6    56 12356  8.21  8.14  342.
##  6  0.5  1X2      60.9    58  2016  5.13  5.09   81.2
##  7  0.43 1V6      62.8    58   839  4.85  4.83   71.2
##  8  1.03 2W3      61.4    55  7481  6.5   6.53  170.
##  9  0.54 1Y5      59.5    61  1356  5.31  5.28   88.3
## 10  0.31 2X4      62.3    55   802  4.38  4.35   51.8
```

3 Data Cleaning and Pre-processing

3.1 Remove Missing Prices and Volumes

```
# Check missing values
sum(is.na(data))
```

```
## [1] 201
```

```r
# Remove missing values
data_cleaned <- na.omit(data)

# Check missing values after removing
sum(is.na(data_cleaned))
```

```
## [1] 0
```

3.2 Derive Cut, Colour, and Clarity

3.2.1 Cut

```r
# Create 'cut' column based on the first character of 'c.grade'
data_cleaned <- data_cleaned %>%
  mutate(
    cut = ifelse(substr(c.grade, 1, 1) == '1', 'premium', 'ideal')
  )

# Create table of 'cut'
cut_table <- data_cleaned %>% count(cut)

# Display the table
cut_table
```

```
## # A tibble: 2 x 2
##   cut         n
##   <chr>   <int>
## 1 ideal   17149
## 2 premium 10924
```

3.2.2 Colour

```r
# Create 'colour' column based on the second character of 'c.grade'
data_cleaned <- data_cleaned %>%
  mutate(
    colour = substr(c.grade, 2, 2)
  )

# Create table of 'cut'
colour_table <- data_cleaned %>% count(colour)

# Display the table
colour_table
```

```
## # A tibble: 7 x 2
##   colour     n
##   <chr>  <int>
```

```
## 1 T       1360
## 2 U       2788
## 3 V       4397
## 4 W       6155
## 5 X       4840
## 6 Y       4989
## 7 Z       3544
```

3.2.3 Clarity

```r
# Create a map for clarity
clarity_map <- c('0' = 'IF',
                 '1' = 'VVS1',
                 '2' = 'VVS2',
                 '3' = 'VS1',
                 '4' = 'VS2',
                 '5' = 'SI1',
                 '6' = 'SI2',
                 '7' = 'I1')

# Create 'clarity' column based on the third character of 'c.grade'
data_cleaned <- data_cleaned %>%
  mutate(clarity = clarity_map[substr(c.grade, 3, 3)])

# Create table of 'cut'
clarity_table <- data_cleaned %>% count(clarity)

# Display the table
clarity_table
```

```
## # A tibble: 8 x 2
##   clarity     n
##   <chr>   <int>
## 1 I1        267
## 2 IF       1122
## 3 SI1      6265
## 4 SI2      4421
## 5 VS1      4396
## 6 VS2      6698
## 7 VVS1     2118
## 8 VVS2     2786
```

3.3 Select variables

```r
# Select variables
data_2 <- data_cleaned %>%
  dplyr::select(cut, price, volume)

# Display the first 10 lines of the data
data_2 %>% head(10)
```

```
## # A tibble: 10 x 3
##    cut     price volume
##    <chr>   <int>  <dbl>
##  1 ideal    1743   85.7
##  2 premium   788   51.7
##  3 premium  2317  115.
##  4 ideal   12356  342.
##  5 premium  2016   81.2
##  6 premium   839   71.2
##  7 ideal    7481  170.
##  8 premium  1356   88.3
##  9 ideal     802   51.8
## 10 ideal     921   57.4
```

3.4 Convert Cut to categorical

```r
# Convert 'cut' to factor
data_2 <- data_2 %>%
  mutate(cut = as.factor(cut))

# Display the first 10 lines of the data
data_2 %>% head(10)
```

```
## # A tibble: 10 x 3
##    cut     price volume
##    <fct>   <int>  <dbl>
##  1 ideal    1743   85.7
##  2 premium   788   51.7
##  3 premium  2317  115.
##  4 ideal   12356  342.
##  5 premium  2016   81.2
##  6 premium   839   71.2
##  7 ideal    7481  170.
##  8 premium  1356   88.3
##  9 ideal     802   51.8
## 10 ideal     921   57.4
```

4 Model

4.1 Logistic regression model with cut as the response variable and price as the predictor

```r
# Preprocessor
recipe_cp <- recipe(cut ~ price, data = data_2) %>%
  step_normalize()

# Logistic regression model
log_reg_model <- logistic_reg() %>%
  set_mode('classification') %>%
  set_engine('glm')
```

```r
# Workflow
wf_cp <- workflow() %>%
  add_recipe(recipe_cp) %>%
  add_model(log_reg_model)
wf_cp
```

```
## == Workflow ====================================================================
## Preprocessor: Recipe
## Model: logistic_reg()
##
## -- Preprocessor ----------------------------------------------------------------
## 1 Recipe Step
##
## * step_normalize()
##
## -- Model -----------------------------------------------------------------------
## Logistic Regression Model Specification (classification)
##
## Computational engine: glm
```

4.2 Logistic regression model with cut as the response variable and volume as the predictor

```r
# Preprocessor
recipe_cv <- recipe(cut ~ volume, data = data_2) %>%
  step_normalize()

# Workflow
wf_cv <- workflow() %>%
  add_recipe(recipe_cv) %>%
  add_model(log_reg_model)
wf_cv
```

```
## == Workflow ====================================================================
## Preprocessor: Recipe
## Model: logistic_reg()
##
## -- Preprocessor ----------------------------------------------------------------
## 1 Recipe Step
##
## * step_normalize()
##
## -- Model -----------------------------------------------------------------------
## Logistic Regression Model Specification (classification)
##
## Computational engine: glm
```

5 ROC Curves

```r
# Fit the models
fit_cp <- fit(wf_cp, data = data_2)
fit_cv <- fit(wf_cv, data = data_2)

# Predict probabilities and add true class to predictions
pred_price <- predict(fit_cp, data_2, type = "prob") %>%
  bind_cols(data_2 %>% dplyr::select(cut))
pred_volume <- predict(fit_cv, data_2, type = "prob") %>%
  bind_cols(data_2 %>% dplyr::select(cut))

# Calculate ROC curve metrics
roc_price <- pred_price %>%
  roc_curve(truth = cut, .pred_ideal)
roc_volume <- pred_volume %>%
  roc_curve(truth = cut, .pred_ideal)

# Add model names to each table
roc_price <- roc_price %>%
  mutate(model = "Price")
roc_volume <- roc_volume %>%
  mutate(model = "Volume")

# Combine the two tables
roc_data <- bind_rows(roc_price, roc_volume)

# Plot ROC curves
ggplot(roc_data, aes(x = 1 - specificity, y = sensitivity, color = model)) +
  geom_line() +
  geom_abline(linetype = "dashed") +
  labs(x = "1 - specificity",
       y = "sensitivity") +
  theme_minimal() +
  theme(panel.border = element_rect(color = "black", fill = NA, size = 0.5))
```

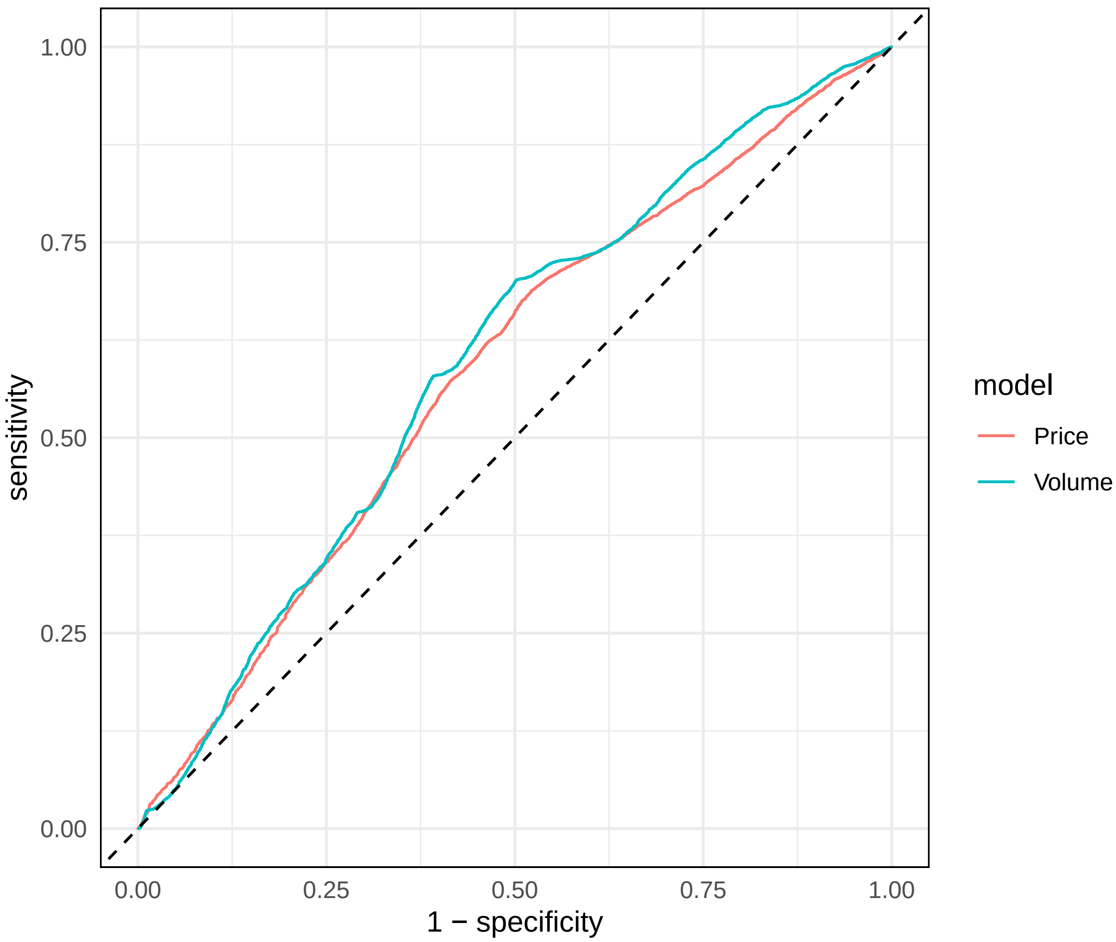

Figure 1: ROC Curves for logistic regression models.