Technical Demonstrations - MTTP

Ethan Ericson

2024-03-02

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.3     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(RColorBrewer) #Make colors for ggplots
library(readxl) #Reading Excel Files In
```

```
## Warning: package 'readxl' was built under R version 4.3.2
```

```
library(vcd) #Mosaic
```

```
## Warning: package 'vcd' was built under R version 4.3.2
```

```
## Loading required package: grid
```

```
library(caret)#Machine Learning
```

```
## Warning: package 'caret' was built under R version 4.3.2
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```r
library(randomForest)#Machine Learning
```

```
## Warning: package 'randomForest' was built under R version 4.3.2
```

```
## randomForest 4.7-1.1
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
##
## The following object is masked from 'package:dplyr':
##
##     combine
##
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
library(ggpmisc)#R and R^2 statistics
```

```
## Warning: package 'ggpmisc' was built under R version 4.3.2
```

```
## Loading required package: ggpp
```

```
## Warning: package 'ggpp' was built under R version 4.3.2
```

```
## Registered S3 methods overwritten by 'ggpp':
##   method                  from
##   heightDetails.titleGrob ggplot2
##   widthDetails.titleGrob  ggplot2
##
## Attaching package: 'ggpp'
##
## The following object is masked from 'package:ggplot2':
##
##     annotate
##
## Registered S3 method overwritten by 'ggpmisc':
##   method                  from
##   as.character.polynomial polynom
```

```r
library(gplots)#Balloonplots
```

```
## Warning: package 'gplots' was built under R version 4.3.2
```

```
##
## Attaching package: 'gplots'
##
## The following object is masked from 'package:stats':
##
##     lowess
```

Setup

Loading in Data

```
data_set <- read_excel("(US) Sample - Superstore.xls")

returns <- read_excel("(US) Sample - Superstore.xls", sheet = "Returns")
```

Filtering out duplicates from returns dataset

```
returns <- distinct(returns)
```

Left-joining Returns

```
data_set_returns <- data_set %>%
  left_join(returns, "Order ID") %>%
  mutate(Returned = coalesce(Returned, "No"))
```

Chi Squared Analysis and Mosaics

Base R Contingency Table

```
# Filtering Data by South Region
south_returns <- data_set_returns %>%
  filter(Region == "South")

# Creating Contingency Tables
cont_table = table(south_returns$Returned, south_returns$State)

cont_table
```

```
##
##       Alabama Arkansas Florida Georgia Kentucky Louisiana Mississippi
##   No       59       60     372     174      135        41          48
##   Yes       2        0      11      10        4         1           5
##
##       North Carolina South Carolina Tennessee Virginia
##   No              241             42       164      215
##   Yes               8              0        19        9
```

```
chi_squared_result = chisq.test(cont_table)
```

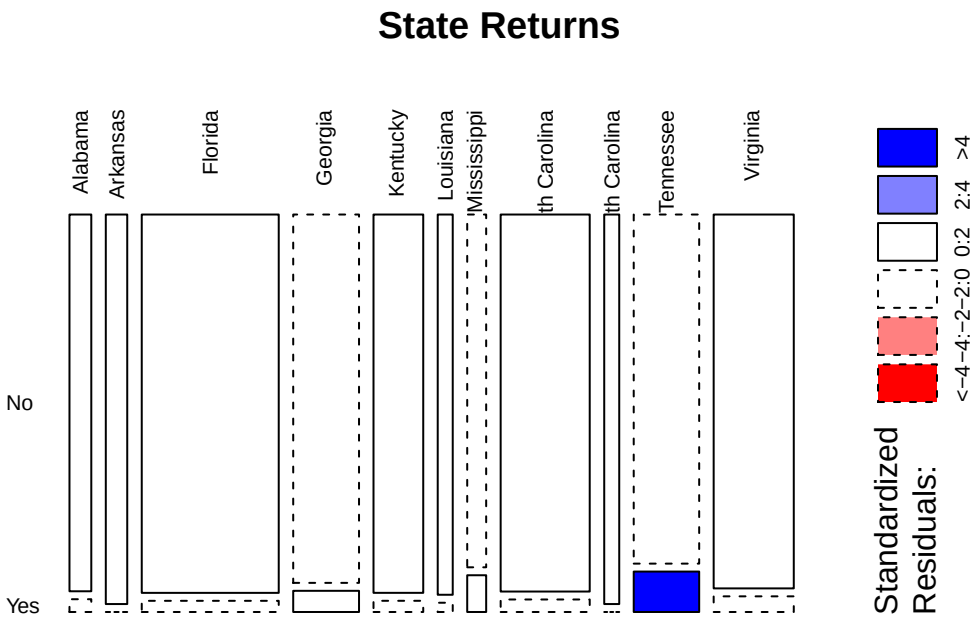
```
## Warning in chisq.test(cont_table): Chi-squared approximation may be incorrect
```

```
chi_squared_result
```

```
##
##  Pearson's Chi-squared test
##
## data:  cont_table
## X-squared = 29.133, df = 10, p-value = 0.001186
```

Base R Mosaic Plot

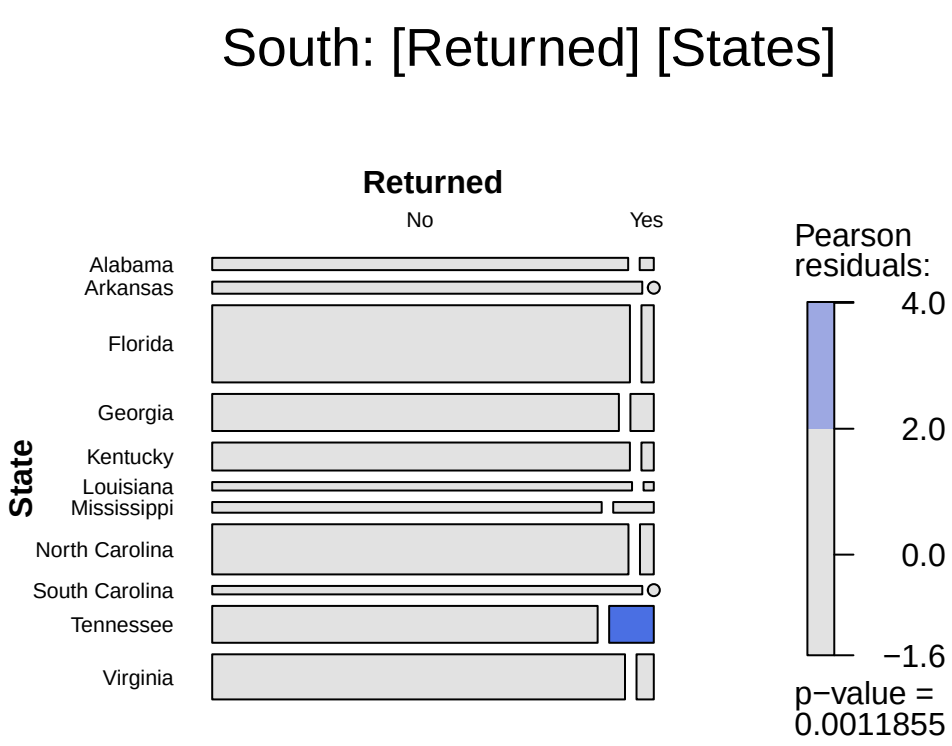
```
mosaicplot(t(cont_table), shade = TRUE, las = 2, main = "State Returns")
```



VCD Mosaic Plot

```
state_v_returns_S <-  xtabs(~Returned + State, data = south_returns)
# S = South

mosaic(
  t(state_v_returns_S), # Contingency Table
  gp = shading_hcl, # Colors
```

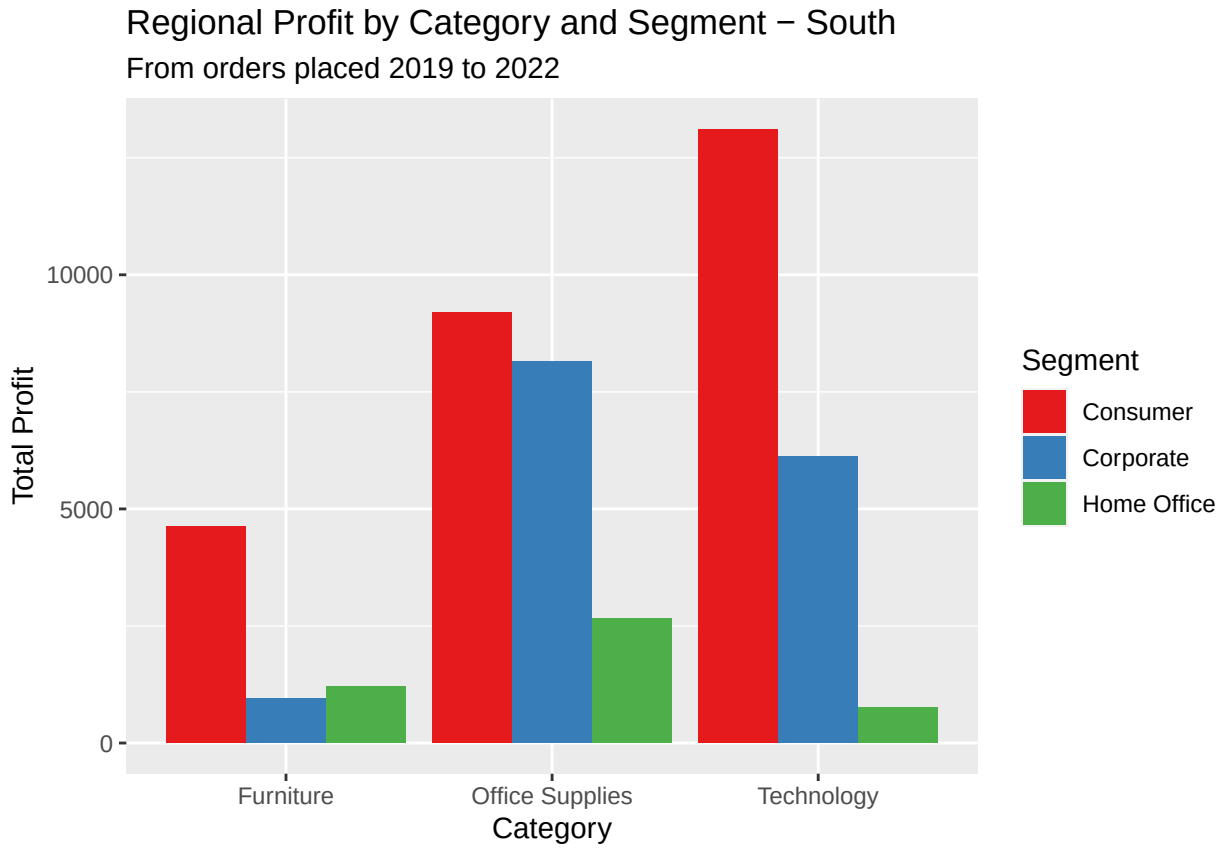
```
  main = "South: [Returned] [States]", # Title
  labeling = labeling_border # Aestetic Customization
    (
      varnames = c(TRUE, TRUE),
      offset_varnames = c(0, 0, 0, 3),
      rot_labels = c(0,0, 0, 0),
      offset_label = c(0.5,0,0, 0.5),
      just_labels = c("center","right"),
      gp_labels = gpar(fontsize = 8)
    )
)
```



Bar Chart

```
# Colors
brewer_palette <- brewer.pal(n = 4, name = "Set1")

# Filtering by South Region; comparing Segment, Category, and Total Profit
regional_profit <- south_returns %>%
  filter(Region == "South") %>%
  select(Segment, Category, Profit) %>%
  group_by(Segment, Category) %>%
  mutate(total_profit = sum(Profit)) %>%
```

```r
  # Graphing data frame with ggplot
  ggplot(aes(x = Category, y = total_profit, fill = Segment)) +
  # Using dodge to display columns side by side
  # Using identity to override frequency with Total Profit
  geom_bar(position = "dodge", stat = "identity") +
  # Labeling axis and title
  labs(
    x = "Category",
    y = "Total Profit",
    title = "Regional Profit by Category and Segment - South",
    subtitle = "From orders placed 2019 to 2022"
  ) +
  scale_fill_manual(values = brewer_palette) # Coloring bars with palette

# Saving ggplot as png for export
ggsave("Regional Profit by Category and Segment - South.png",
       plot = regional_profit,
       width = 6,
       height = 4)

regional_profit
```



Machine Learning

Target Guided Ordinal Encoding for Segement, Category, Returned

Segment

```
# Segment Encoding

# Finding the mean of the target variable (Profit) for each segment
segment_mean_profit <- data_set_returns %>%
  group_by(Segment) %>%
  summarize(mean_profit = mean(Profit))

# Ranking the segments by mean profit
segment_mean_profit <- segment_mean_profit %>%
  mutate(segment_rank = rank(mean_profit))

# Encoding the variables
data_set_returns <- data_set_returns %>%
  left_join(segment_mean_profit, by = "Segment") %>%
  select(-mean_profit)
```

Category

```
# Category Encoding

# Finding the mean of the target variable (Profit) for each segment
category_mean_profit <- data_set_returns %>%
  group_by(Category) %>%
  summarize(mean_profit = mean(Profit))

# Ranking the segments by mean profit
category_mean_profit <- category_mean_profit %>%
  mutate(category_rank = rank(mean_profit))

# Encoding the variables
data_set_returns <- data_set_returns %>%
  left_join(category_mean_profit, by = "Category") %>%
  select(-mean_profit)
```

Returned

```
# Returned Encoding

# Finding the mean of the target variable (Profit) for each segment
returned_mean_profit <- data_set_returns %>%
  group_by(Returned) %>%
  summarize(mean_profit = mean(Profit))
```

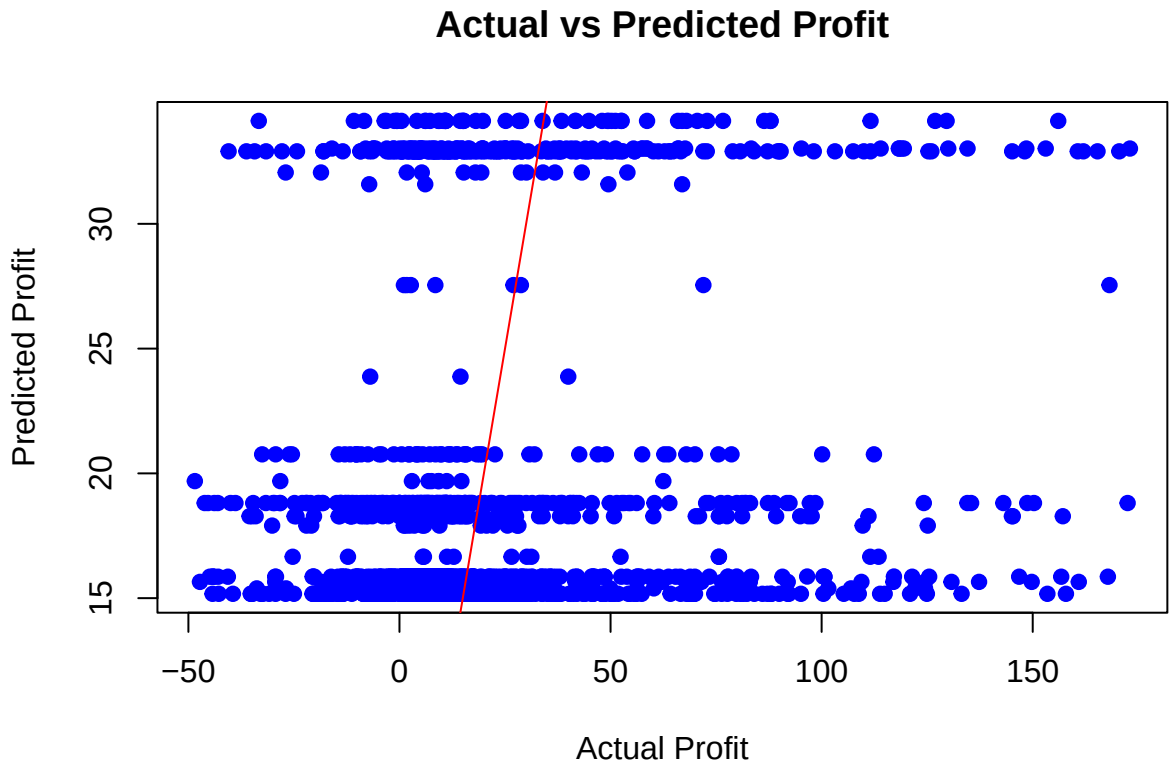
```r
# Ranking the segments by mean profit
returned_mean_profit <- returned_mean_profit %>%
  mutate(returned_rank = rank(mean_profit))

# Encoding the variables
data_set_returns <- data_set_returns %>%
  left_join(returned_mean_profit, by = "Returned") %>%
  select(-mean_profit)

view(data_set_returns)
```

Random Forest Regression Analysis

```r
# Creating the training and testing data
set.seed(123) # Setting a random seed for reproducability
trainIndex <- createDataPartition(data_set$Profit, p = 0.8, list = FALSE)
train_data <- data_set_returns[trainIndex, ]
test_data <- data_set_returns[-trainIndex, ]

# Trimming outliers of train data
lower_percentile <- 0.05
upper_percentile <- 0.95

# Calculating the lower and upper quantiles
lower_threshold <- quantile(train_data$Profit, lower_percentile)
upper_threshold <- quantile(train_data$Profit, upper_percentile)

# Trimming outliers from the training data
train_data_trimmed <- train_data[train_data$Profit >= lower_threshold &
                                   train_data$Profit <= upper_threshold, ]

#Trimming outliers of test data
lower_threshold <- quantile(test_data$Profit, lower_percentile)
upper_threshold <- quantile(test_data$Profit, upper_percentile)

# Trimming outliers from the training data
test_data_trimmed <- test_data[test_data$Profit >= lower_threshold &
                                 test_data$Profit <= upper_threshold, ]

# Training the model
model <- train(Profit ~ Segment + Category + Returned,
               data = train_data_trimmed, method = "rf",
               na.action = na.omit,
               preProcess=c("scale","center"))

# Evaluating the model with Mean Squared Error (MSE)
predictions <- predict(model, newdata = test_data_trimmed)
mse <- mean((predictions - test_data_trimmed$Profit)^2)
print(paste("MSE:", mse))
```

```
## [1] "MSE: 1080.94646326448"
```

```r
# Plot
results <- data.frame(Actual = test_data_trimmed$Profit,
                      Predicted = predictions)

# Plot actual vs predicted values
plot(results$Actual, results$Predicted,
     xlab = "Actual Profit",
     ylab = "Predicted Profit",
     main = "Actual vs Predicted Profit",
     col = "blue",
     pch = 19)

# Adding a diagonal line for comparison
abline(0, 1, col = "red")
```




```r
# Predicting profit given "new data"
new_data <- data.frame(Segment = "Consumer", Category = "Technology",
                       Returned = "No")
predicted_profit <- predict(model, newdata = new_data)

predicted_profit
```

```
##        1
## 32.90644
```