Competition 1

Tim Reznicek

2024-3-15

```
##
## Attaching package: 'igraph'

## The following objects are masked from 'package:stats':
##
##     decompose, spectrum

## The following object is masked from 'package:base':
##
##     union

## Loading required package: ggplot2

##
## Attaching package: 'plotly'

## The following object is masked from 'package:ggplot2':
##
##     last_plot

## The following object is masked from 'package:igraph':
##
##     groups

## The following object is masked from 'package:stats':
##
##     filter

## The following object is masked from 'package:graphics':
##
##     layout

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:igraph':
##
##     as_data_frame, groups, union
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:igraph':
##
##     %--%, union

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
# orders dataframe
orders <- read.csv("contest1data/orders.csv")
orders$date <- as.Date(orders$date)
str(orders)
```

```
## 'data.frame':    816574 obs. of  4 variables:
##  $ date  : Date, format: "2020-08-01" "2020-08-01" ...
##  $ userID: int  14477 14477 33883 33883 33883 33883 19087 42266 4286 4286 ...
##  $ itemID: int  2375 14961 8927 27830 29700 27432 9798 18630 31949 10468 ...
##  $ count : int  1 2 1 2 1 4 1 3 1 1 ...
```

```
summary(orders)
```

```
##       date                userID          itemID          count
##  Min.   :2020-08-01   Min.   :    0   Min.   :    1   Min.   :  1.000
##  1st Qu.:2020-09-19   1st Qu.:11553   1st Qu.: 8350   1st Qu.:  1.000
##  Median :2020-11-09   Median :23134   Median :16962   Median :  1.000
##  Mean   :2020-11-04   Mean   :23087   Mean   :16526   Mean   :  1.409
##  3rd Qu.:2020-12-20   3rd Qu.:34660   3rd Qu.:24576   3rd Qu.:  1.000
##  Max.   :2021-01-31   Max.   :46137   Max.   :32775   Max.   :100.000
```

```
# items dataframe
items <- read.csv("contest1data/items.csv")
str(items)
```

```
## 'data.frame':    32776 obs. of  8 variables:
##  $ itemID        : int  22665 28640 13526 21399 8504 32122 31956 6237 16971 18385 ...
##  $ manufacturerID: int  861 1366 1090 1090 768 5 1388 1492 288 288 ...
##  $ f1            : int  4 10 10 10 4 4 4 4 6 6 ...
##  $ f2            : int  0 1 0 1 1 1 0 1 0 0 ...
##  $ f3            : int  490 537 511 511 484 491 491 491 314 314 ...
##  $ f4            : int  2 0 0 0 0 0 0 3 0 0 ...
##  $ f5            : int  66 101 0 0 66 66 66 66 45 45 ...
##  $ category      : chr  "[2890, 855, 3908, 3909]" "" "[3270, 163, 284, 1694, 12, 3837, 2422, 3595, 3
```

```r
# categories dataframe
categories <- read.csv("contest1data/categories.csv", stringsAsFactors = FALSE)
str(categories)
```

```
## 'data.frame':    4332 obs. of  2 variables:
##  $ category       : int  0 1 2 3 4 5 6 7 8 9 ...
##  $ parent_category: int  75 1499 1082 3498 1623 2478 1582 3027 2364 3590 ...
```

```r
# test dataframe
test <- read.csv("contest1data/test.csv")
str(test)
```

```
## 'data.frame':    10000 obs. of  3 variables:
##  $ userID: int  0 0 13 15 15 20 24 34 34 46 ...
##  $ itemID: int  20664 28231 2690 1299 20968 8272 11340 21146 31244 31083 ...
##  $ ID    : chr  "0-20664" "0-28231" "13-2690" "15-1299" ...
```

```r
summary(test)
```

```
##      userID          itemID           ID
##  Min.   :    0   Min.   :    6   Length:10000
##  1st Qu.:11589   1st Qu.: 8398   Class :character
##  Median :22841   Median :16807   Mode  :character
##  Mean   :23064   Mean   :16505
##  3rd Qu.:34919   3rd Qu.:24527
##  Max.   :46130   Max.   :32766
```

```r
# how many manufacturerIDs exist, what is the most common?
summary(items$manufacturerID)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     0.0   322.0   648.0   702.8  1073.0  1513.0
```

```r
mode(items$manufacturerID)
```

```
## [1] "numeric"
```

```r
# Excluding empty values, summary of f columns
sapply(items[, c("f1", "f2", "f3", "f4", "f5")], function(x) summary(x[x != -1]))
```

```
##                f1        f2       f3        f4       f5
## Min.     0.000000 0.0000000   0.0000 0.0000000   0.0000
## 1st Qu.  4.000000 0.0000000 453.0000 0.0000000  44.0000
## Median   6.000000 1.0000000 491.0000 0.0000000  82.0000
## Mean     6.451971 0.8179461 446.6449 0.8393607  86.0087
## 3rd Qu. 10.000000 1.0000000 510.0000 3.0000000 137.0000
## Max.    10.000000 3.0000000 538.0000 4.0000000 190.0000
```

```r
# how many missing values exist in each column?
missing_counts <- sapply(items, function(x) sum(is.na(x) | x == -1 | x == ""))
print(missing_counts)
```

```
##         itemID manufacturerID             f1             f2             f3
##              0              0              4              0            466
##             f4             f5       category
##            617           4264           6788
```

```r
# degree of missingness
with(items, table(category == "", f5 == -1))
```

```
##
##         FALSE  TRUE
##   FALSE 22961  3027
##   TRUE   5551  1237
```

```r
with(items, table(category == "", f4 == -1))
```

```
##
##         FALSE  TRUE
##   FALSE 25394   594
##   TRUE   6765    23
```

```r
with(items, table(category == "", f3 == -1))
```

```
##
##         FALSE  TRUE
##   FALSE 25549   439
##   TRUE   6761    27
```

```r
with(items, table(f3 == -1, f5 == -1))
```

```
##
##         FALSE  TRUE
##   FALSE 28440  3870
##   TRUE     72   394
```

```r
with(items, table(f4 == -1, f5 == -1))
```

```
##
##         FALSE  TRUE
##   FALSE 28271  3888
##   TRUE    241   376
```

```r
with(items, table(f4 == -1, f3 == -1))
```

```
##
##         FALSE  TRUE
##   FALSE 32121    38
##   TRUE    189   428
```

```r
# category column
cleaned_categories <- gsub("]", "", as.character(items$category[!is.na(items$category) & items$category
# Split the cleaned categories into individual categories
categories <- strsplit(cleaned_categories, ",")
# Now, you can proceed to find unique categories and count them as before
unique_categories <- unique(unlist(categories))
category_counts <- table(unlist(categories))

# Print the counts of each category
#print(category_counts)

print("could reorder this to show the most represented at the top or pie chart or something.")
```

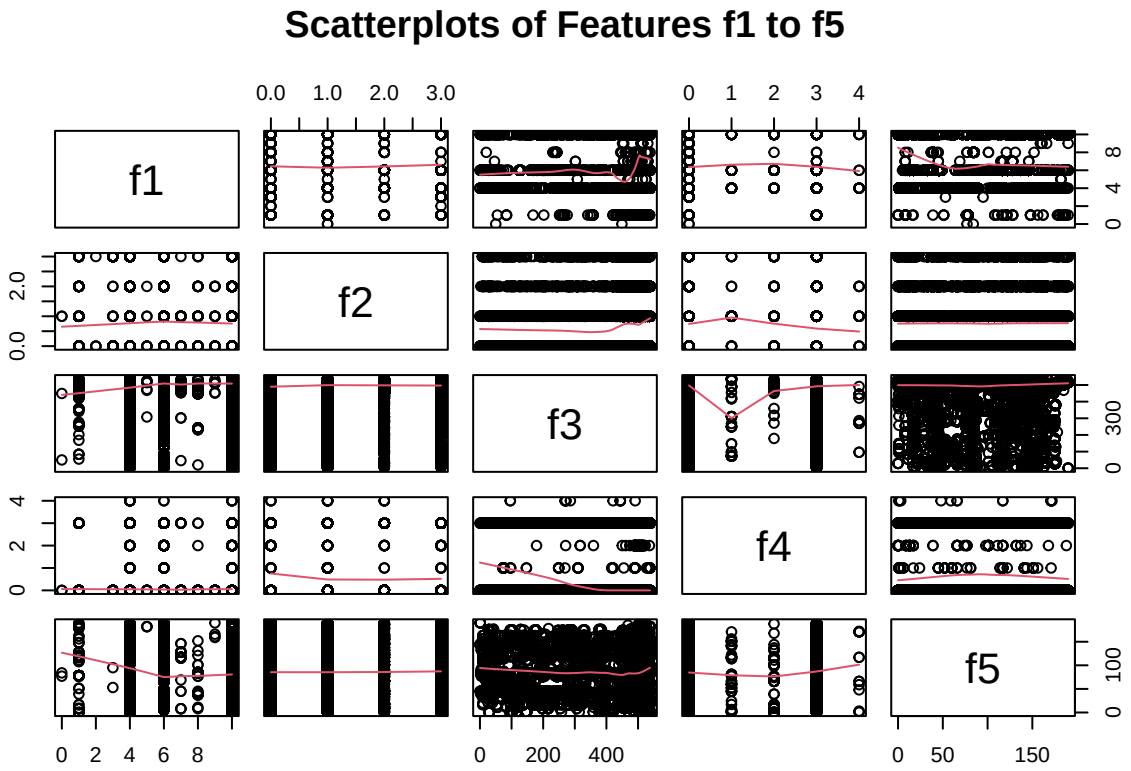
```
## [1] "could reorder this to show the most represented at the top or pie chart or something."
```

```r
# g <- graph_from_data_frame(d = categories, directed = TRUE)
# pdf("category_hierarchy.pdf", width = 50, height = 10)
# plot(g, layout = layout_as_tree(g), edge.arrow.size = 0.5, vertex.label = V(g)$name, vertex.size = 10
# dev.off()
```

```r
# # Assuming 'g' is your igraph object
# coords <- layout_as_tree(g)  # Get coordinates for tree layout
# edge_x <- c(); edge_y <- c()
# for(e in E(g)){
#   # Extract the vertex ids for the ends of each edge
#   ends_ids <- ends(g, e, names = FALSE)
#
#   # Use these ids to index into coords directly
#   edge_x <- c(edge_x, coords[ends_ids[1], 1], coords[ends_ids[2], 1], NA)
#   edge_y <- c(edge_y, coords[ends_ids[1], 2], coords[ends_ids[2], 2], NA)
# }
#
# node_x <- coords[,1]
# node_y <- coords[,2]
# text_labels <- V(g)$name
#
# # Create edges
# p <- plot_ly(type='scatter', mode='lines', x=~edge_x, y=~edge_y, line=list(color='black')) %>%
#   add_trace(x=~node_x, y=~node_y, mode='markers+text', text=~text_labels, textposition='bottom center
#
# # Customize layout
# p <- layout(p, title='Category Hierarchy')
#
# # Display the plot
# p
```

```r
items_filtered <- items
items_filtered[items_filtered == -1] <- NA  # Replace -1 with NA for filtering

# Select only the columns f1 to f5
features <- items_filtered[,c("f1", "f2", "f3", "f4", "f5")]
```

```r
#scatterplots of all combinations of f1 to f5
pairs(features, panel = panel.smooth, main = "Scatterplots of Features f1 to f5")
```



Taking into account both the missingness relationships, how many values are missing in each feature, and the relationships between features, I have devised the following method to clean the features.
f1: make them 0
f2: no missing
f3 and f4 have high overlap in missingness f3: median (400 something) f4: median (0) f5: median as well, not sure if I can do any good predictions on this.

What about categories? To start out I will consider missing categories to be empty. Eventually I would like to find features related to categories and potentially fill them in that way.

Join orders and items on itemID.

```r
# Join the datasets on 'itemID'
joined_data <- inner_join(orders, items, by = "itemID")
```

Cleaning f1-f5

```r
joined_data$f1[joined_data$f1 == -1] <- 0

# Calculate median for f3, excluding -1
median_f3 <- median(joined_data$f3[joined_data$f3 != -1], na.rm = TRUE)
```

```r
# Replace -1 with median in f3
joined_data$f3[joined_data$f3 == -1] <- median_f3

# f4
median_f4 <- median(joined_data$f4[joined_data$f4 != -1], na.rm = TRUE)
joined_data$f4[joined_data$f4 == -1] <- median_f4

# f5
median_f5 <- median(joined_data$f5[joined_data$f5 != -1], na.rm = TRUE)
joined_data$f5[joined_data$f5 == -1] <- median_f5
```

Feature Extraction from Date:

```r
# Create a new column 'day_of_week' to store the numeric day of the week
joined_data$day_of_week_numeric <- sapply(weekdays(joined_data$date), function(x) {
  match(x, c("Saturday", "Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday"))
})
```

Design Matrix

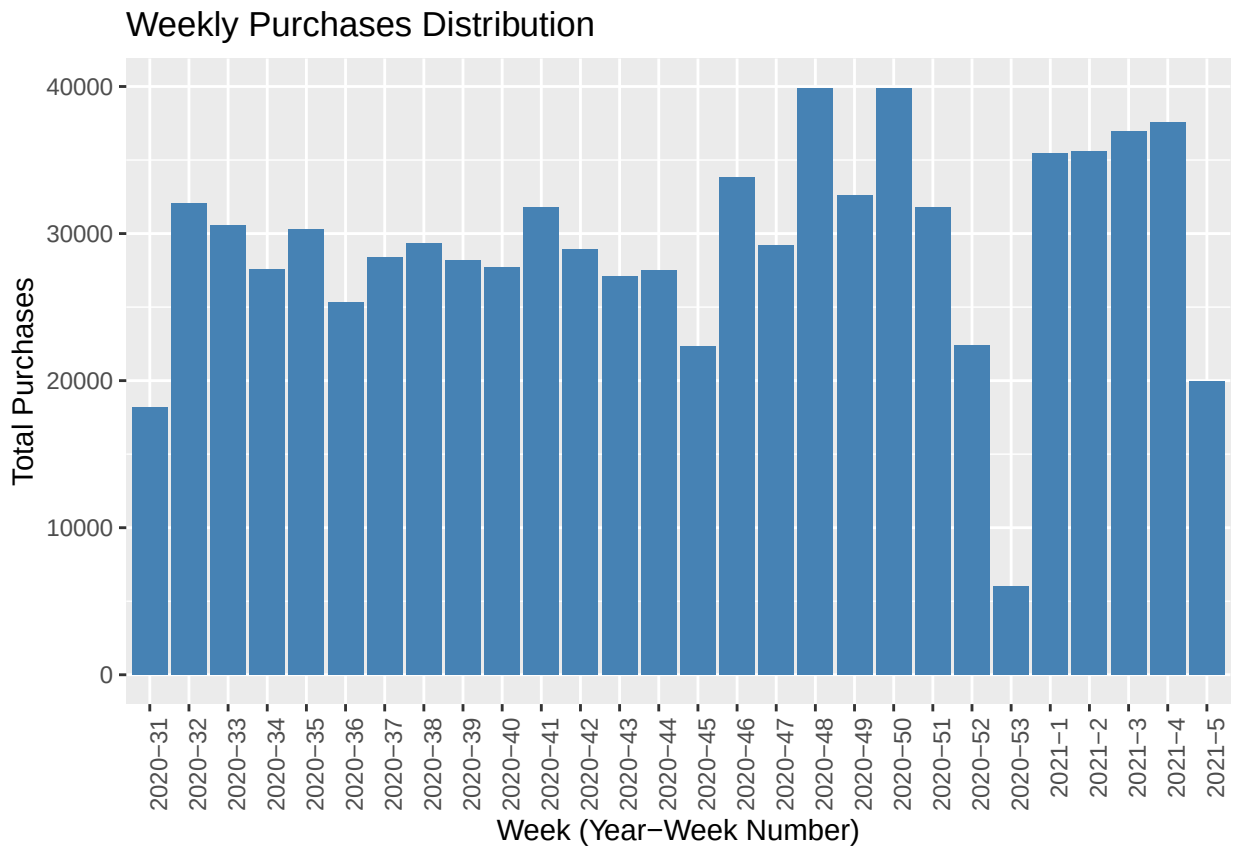
How many purchases are made by week?

```r
# Convert "date" column to Date type
joined_data$date <- as.Date(joined_data$date)

# Extract year and week number
joined_data$year <- year(joined_data$date)
joined_data$week <- week(joined_data$date)

# Combine year and week for unique identification across years
joined_data$year_week <- paste(joined_data$year, joined_data$week, sep="-")
```

```r
weekly_purchases <- joined_data %>%
  group_by(year_week) %>%
  summarise(total_purchases = n())

# View the first few rows to verify
head(weekly_purchases)
```

```
## # A tibble: 6 x 2
##   year_week total_purchases
##   <chr>               <int>
## 1 2020-31             18201
## 2 2020-32             32073
## 3 2020-33             30538
## 4 2020-34             27593
## 5 2020-35             30324
## 6 2020-36             25308
```

```r
ggplot(weekly_purchases, aes(x = year_week, y = total_purchases)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  labs(title = "Weekly Purchases Distribution",
       x = "Week (Year-Week Number)",
       y = "Total Purchases")
```




```r
print(weekly_purchases)
```

```
## # A tibble: 28 x 2
##    year_week total_purchases
##    <chr>               <int>
##  1 2020-31             18201
##  2 2020-32             32073
##  3 2020-33             30538
##  4 2020-34             27593
##  5 2020-35             30324
##  6 2020-36             25308
##  7 2020-37             28409
##  8 2020-38             29314
##  9 2020-39             28210
## 10 2020-40             27686
## # i 18 more rows
```

How many test.csv purchases were never made in the training data?

```r
# Create a unique identifier in both datasets by pasting userID and itemID
joined_data$uid_item_comb <- paste(joined_data$userID, joined_data$itemID, sep = "_")
test$uid_item_comb <- paste(test$userID, test$itemID, sep = "_")

# Identify combinations in test_data not present in joined_data
unique_combinations_not_in_joined <- test %>%
  filter(!(uid_item_comb %in% joined_data$uid_item_comb))

# Count the total number of unique rows
total_unique_not_in_joined <- nrow(unique_combinations_not_in_joined)

# Print the result
print(total_unique_not_in_joined)
```

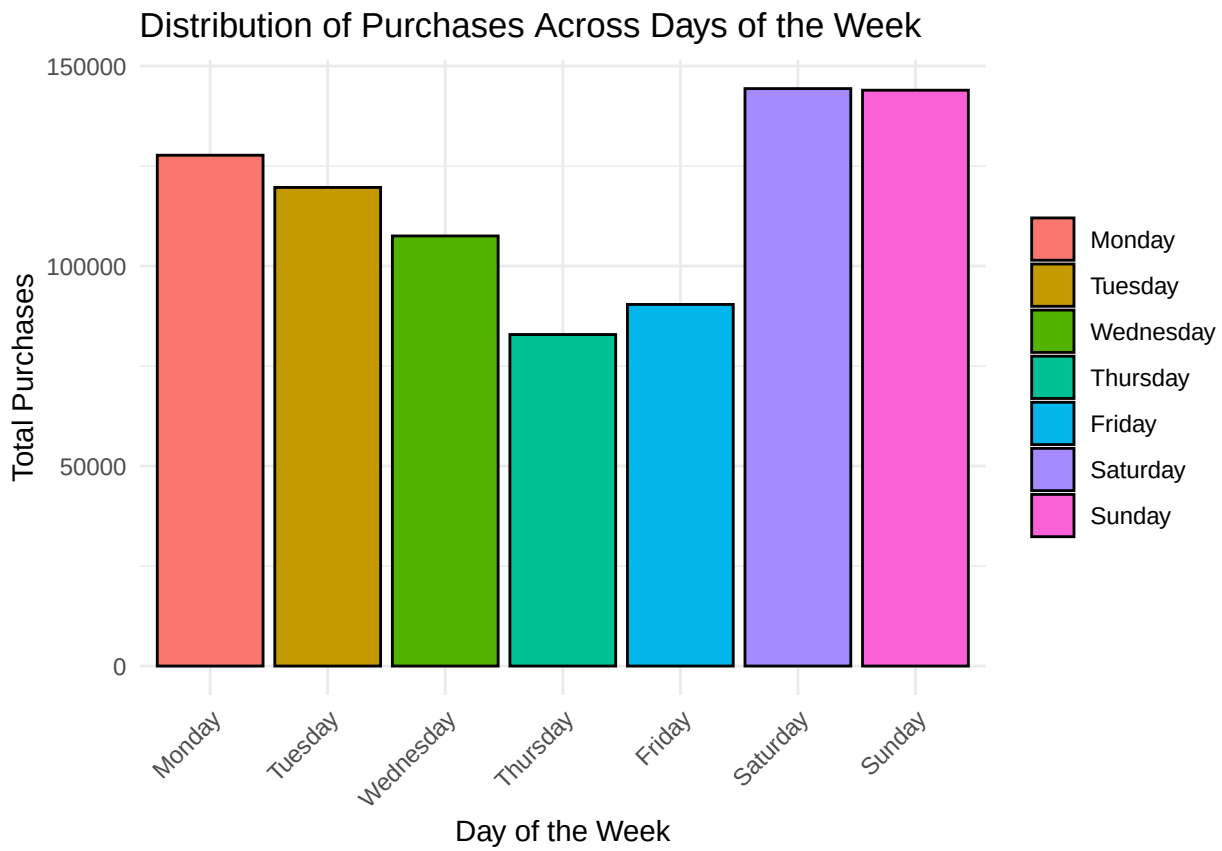
```
## [1] 53
```

What is the breakdown by day?

```r
# Extract the day of the week
joined_data$day_of_week <- weekdays(joined_data$date)
# Aggregate data
purchases_by_day <- joined_data %>%
  group_by(day_of_week) %>%
  summarise(total_purchases = n()) %>%
  mutate(day_of_week = factor(day_of_week, levels = c("Monday", "Tuesday", "Wednesday", "Thursday", "Fr

# View the result
print(purchases_by_day)
```

```
## # A tibble: 7 x 2
##   day_of_week total_purchases
##   <fct>                 <int>
## 1 Friday                90411
## 2 Monday               127724
## 3 Saturday             144373
## 4 Sunday               143978
## 5 Thursday              82892
## 6 Tuesday              119664
## 7 Wednesday            107532
```

```r
# Plot
ggplot(purchases_by_day, aes(x = day_of_week, y = total_purchases, fill = day_of_week)) +
  geom_bar(stat = "identity", color = "black") +
  theme_minimal() +
  labs(title = "Distribution of Purchases Across Days of the Week",
       x = "Day of the Week",
       y = "Total Purchases") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1), # Improve readability of x labels
        legend.title = element_blank()) # Remove the legend title
```



I will extract the target variable for the training data based on if the purchase was repeated in January. I have to change what this means though.
Target variable: Was the purchase made again during a 4 week window in January?
*January has higher purchases, this could be higher first time purchases or could be higher repeats from earlier in the year. I should answer this at some point.
0 - no. 1 - first monday to sunday (1/4-1/10) 2 - and so on (1/11-1/17) 3 - (1/18 - 1/24) 4 - (1/25 - 1/31)

```
training_data <- joined_data[joined_data$date < as.Date("2021-01-04"),]
#summary(training_data)
```

```
# Filter orders for January 2021
jan_orders <- joined_data[joined_data$date >= as.Date("2021-01-04") & joined_data$date <= as.Date("2021
#summary(jan_orders)
# Add week categories to January orders
jan_orders$week_category <- cut(jan_orders$date,
                                breaks = as.Date(c("2021-01-03", "2021-01-10", "2021-01-17", "2021-01-24
                                labels = c("1", "2", "3", "4"),
                                include.lowest = TRUE,
                                right = TRUE)

# Convert 'week_category' from factors to numeric
jan_orders$week_category_numeric <- as.numeric(as.character(jan_orders$week_category))

# Now aggregate using the numeric week category
repeat_jan_orders_summary <- aggregate(week_category_numeric ~ userID + itemID, data = jan_orders, FUN =
```

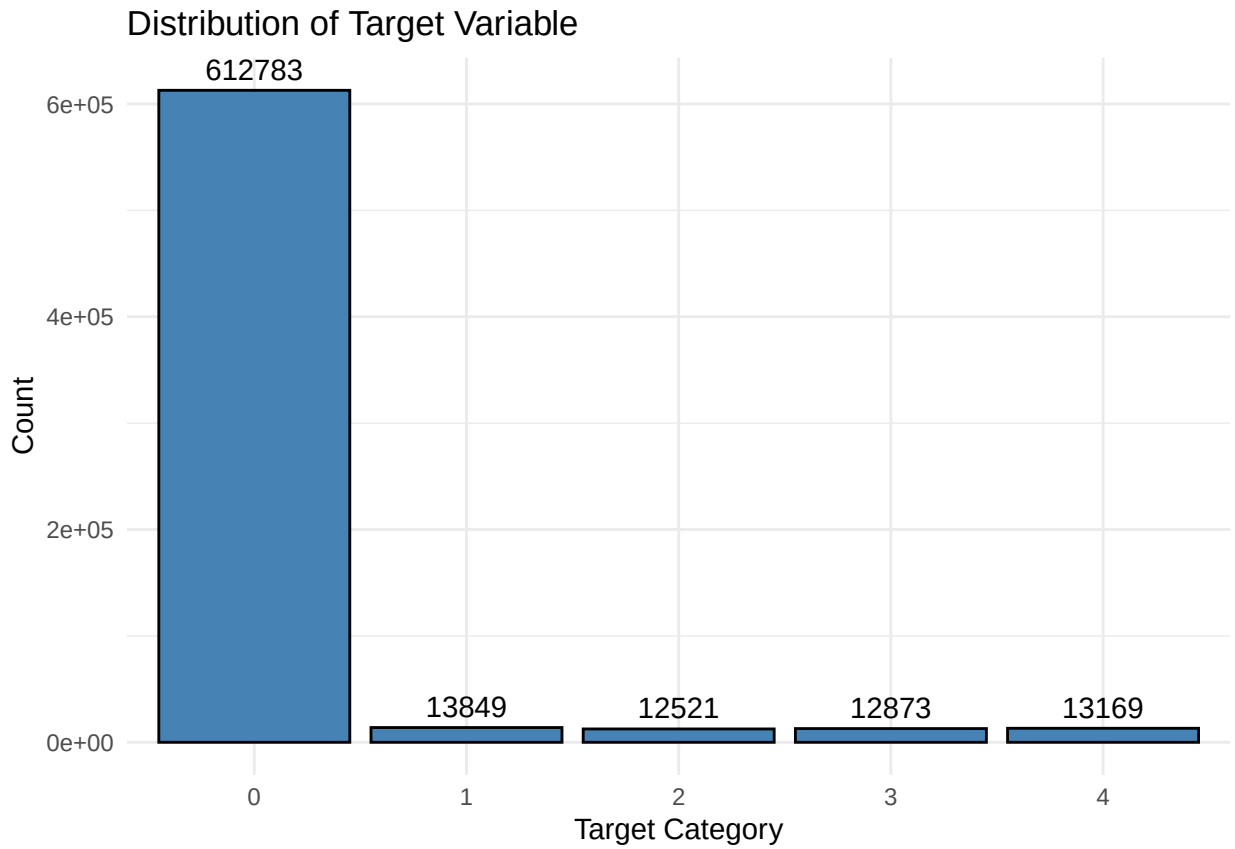
```r
# Optionally, if you want to keep the factor labels in the summary
repeat_jan_orders_summary$week_category <- as.factor(repeat_jan_orders_summary$week_category_numeric)

# Merge this summary back into your training data to create the target variable
training_data$target <- 0  # Default to no repeat orders in January
training_data <- merge(training_data, repeat_jan_orders_summary, by = c("userID", "itemID"), all.x = TRU
training_data$target[!is.na(training_data$week_category)] <- as.numeric(as.character(training_data$week
training_data$target[is.na(training_data$target)] <- 0
```

```r
summary(training_data$target)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0000  0.0000  0.1957  0.0000  4.0000
```

```r
training_data$target <- as.factor(training_data$target)
# Create the bar chart
ggplot(training_data, aes(x = target)) +
  geom_bar(fill = "steelblue", color = "black", aes(y = ..count..)) +  # This plots the bars
  geom_text(stat='count', aes(label=..count.., y=..count..), vjust=-0.5, color="black") + # This adds t
  theme_minimal() +
  labs(title = "Distribution of Target Variable",
       x = "Target Category",
       y = "Count")
```

```
## Warning: The dot-dot notation (`..count..`) was deprecated in ggplot2 3.4.0.
## i Please use `after_stat(count)` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```




```
# I wanted to see what the breakdown would be for the sample_submission which has a score of 0.18 our o
sample_submission <- read.csv("contest1data/sample_submission.csv")
summary(sample_submission)
```

```
##       ID                 target
##  Length:10000       Min.   :0.000
##  Class :character   1st Qu.:1.000
##  Mode  :character   Median :2.000
##                     Mean   :1.705
##                     3rd Qu.:3.000
##                     Max.   :4.000
```

I submitted all 0s and got 0.24294
Next feature extract the day of the week from the training data, fill in empty values and so on.