Final_Project

Tommy Barron, April Chia, Taylor Kirk

2024-07-30

```
# Importing libraries

library(readr)
library(ggplot2)
library(summarytools)
library(psych)
library(stringr)
library(dplyr)
library(corrplot)
library(caret)
library(pscl)
library(pROC)
library(PRROC)
```

Data Importing and Pre-processing ——————————

```
# Importing dataset

df <- read_csv("online_shoppers_intention.csv")
```

```
# Dataset characteristics

dim(df)
```

```
[1] 12330    18
```

```
str(df)
```

```
spc_tbl_ [12,330 x 18] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ Administrative         : num [1:12330] 0 0 0 0 0 0 0 1 0 0 ...
 $ Administrative_Duration: num [1:12330] 0 0 0 0 0 0 0 0 0 0 ...
 $ Informational          : num [1:12330] 0 0 0 0 0 0 0 0 0 0 ...
 $ Informational_Duration : num [1:12330] 0 0 0 0 0 0 0 0 0 0 ...
 $ ProductRelated         : num [1:12330] 1 2 1 2 10 19 1 0 2 3 ...
 $ ProductRelated_Duration: num [1:12330] 0 64 0 2.67 627.5 ...
 $ BounceRates            : num [1:12330] 0.2 0 0.2 0.05 0.02 ...
 $ ExitRates              : num [1:12330] 0.2 0.1 0.2 0.14 0.05 ...
 $ PageValues             : num [1:12330] 0 0 0 0 0 0 0 0 0 0 ...
 $ SpecialDay             : num [1:12330] 0 0 0 0 0 0 0.4 0 0.8 0.4 ...
 $ Month                  : chr [1:12330] "Feb" "Feb" "Feb" "Feb" ...
 $ OperatingSystems       : num [1:12330] 1 2 4 3 3 2 2 1 2 2 ...
 $ Browser                : num [1:12330] 1 2 1 2 3 2 4 2 2 4 ...
 $ Region                 : num [1:12330] 1 1 9 2 1 1 3 1 2 1 ...
```

```
 $ TrafficType             : num [1:12330] 1 2 3 4 4 3 3 5 3 2 ...
 $ VisitorType             : chr [1:12330] "Returning_Visitor" "Returning_Visitor" "Returning_Visitor" "
 $ Weekend                 : logi [1:12330] FALSE FALSE FALSE FALSE TRUE FALSE ...
 $ Revenue                 : logi [1:12330] FALSE FALSE FALSE FALSE FALSE FALSE ...
 - attr(*, "spec")=
  .. cols(
  ..   Administrative = col_double(),
  ..   Administrative_Duration = col_double(),
  ..   Informational = col_double(),
  ..   Informational_Duration = col_double(),
  ..   ProductRelated = col_double(),
  ..   ProductRelated_Duration = col_double(),
  ..   BounceRates = col_double(),
  ..   ExitRates = col_double(),
  ..   PageValues = col_double(),
  ..   SpecialDay = col_double(),
  ..   Month = col_character(),
  ..   OperatingSystems = col_double(),
  ..   Browser = col_double(),
  ..   Region = col_double(),
  ..   TrafficType = col_double(),
  ..   VisitorType = col_character(),
  ..   Weekend = col_logical(),
  ..   Revenue = col_logical()
  .. )
 - attr(*, "problems")=<externalptr>
```

The dataset contains 18 columns and 12,330 rows. We used the str() function to display the structure of each column.

```r
# Determining the number of duplicate rows

df_dup <- df[duplicated(df), ]
dim(df_dup)
```

```
[1] 121  18
```

```r
# Removing duplicate rows to reduce redundant data

df <- df[!duplicated(df), ]
```

It was determined that there were 121 duplicate rows. It is unlikely to have identical rows across all columns, so it could be an input error. We decided to remove those rows to reduce the chance of overfitting.

```r
# Finding missing values

na_counts <- sapply(df, function(x) sum(is.na(x)))
na_counts
```

```
         Administrative Administrative_Duration           Informational
                      0                       0                     128
 Informational_Duration          ProductRelated ProductRelated_Duration
                      0                       0                       0
            BounceRates               ExitRates              PageValues
                      0                       0                     135
             SpecialDay                   Month        OperatingSystems
                      0                       0                     123
```

```
       Browser             Region         TrafficType
             0                  0                   0
   VisitorType            Weekend             Revenue
             0                  0                   0
```

We used the sapply() function over the dataframe (df) which returns a frequency table of how many missing values are in each column. The results showed that the Informational, Page Values, and Operating Systems columns have 128, 135, and 123 missing values, respectively.

Handling missing values (Informational)

```r
# Informational column without the 0's

info_nzero <- subset.data.frame(df[df$Informational > 0, ])

# Probability table for unique values

probability <- table(info_nzero$Informational, useNA = "no")
prob_table <- prop.table(probability)

# Function to randomly impute one of those values based on their proportion

random_impute <- function(values, prob_table, size) {
  sample(values, size, replace = T, prob = prob_table)
}

# Separating the NA Information rows from the ones where InformationDuration is greater
# than 0, and the ones where it's less than or equal to 0

na_indices <- which(is.na(df$Informational))
na_indices_great_zero <- na_indices[df[na_indices, "Informational_Duration"] > 0]
na_indices_zero_or_less <- na_indices[df[na_indices, "Informational_Duration"] <= 0]

# Converting the probability table of unique values to numeric

unique_values <- as.numeric(names(prob_table))
probs <- as.numeric(prob_table)

# Applying the function to the full data set to randomly convert the NA value in the
# Information column to one of it's unique values according to their proportion if
# the corresponding Informational Duration Column is greater than 0

# Viewing the remaining NA values
df$Informational[na_indices_great_zero] <- random_impute(unique_values,
                                                         probs,
                                                         length(na_indices_great_zero))


# Converting the remaining NA values to 0

df$Informational[is.na(df$Informational)] <- 0
```

The Informational column is an integer value that represents the number of pages a customer visited that matched the information category within that session. The next field of Informational Duration is a numeric value representing how long the customer spent on that page. Given that information, the information column would only be an integer greater than 0 if the corresponding Informational Duration field was greater than 0. Therefore, it was decided that the best way to manage the 128 missing values in the Informational column was to impute them according to their proportion among the known values (if the corresponding Information Duration field was greater than 0), and then the remaining missing values were converted to 0's.

Handling missing values (PageValues)

```r
# Subsetting the PageValues NA values

pv_na <- df[!is.na(df$PageValues), ]

# Separate PageValues if Administrative, Information, or ProductRelated columns
# are greater than 0

pv_na <- pv_na[pv_na$Administrative > 0 |
               pv_na$Informational > 0 |
               pv_na$ProductRelated > 0, ]
summary(pv_na$PageValues)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.000   0.000   0.000   5.973   0.000 361.764
```

```r
# Replacing all NA values for the Page Value in the original DF with 0

df$PageValues[is.na(df$PageValues)] <- 0
```

The PageValue column consists of continuous numerical values that reflect the average value of the page visited by the customer prior to completing a transaction. There were 135 missing values originally. First, the missing values were removed, and then the field was further filtered to include only the remaining values if the Administrative, Informational, or ProductRelated fields were greater than 0. It was assumed that if none of these webpages were visited during a session, then there would be no record of the page value. The summary statistics were then looked at for the cleaned column. The mean is 5.97, however the first and third quartiles as well as the median are 0. This indicates that the vast majority of the webpages visited had no value. Given this and the fact that there are only 135 values, it was decided it would be best to impute the median/mode (i.e., 0) into the missing value cells.

Handling missing values (OperatingSystems)

```r
df <- df[!is.na(df$OperatingSystems), ]
```

The OperatingSystems field had 123 null values. This is a categorical variable that has been coded to a numeric data type. However, in the confines of the data we are operating with, we are unsure what operating systems are represented. Therefore, we decided it would make the most sense to remove those missing value rows.

Transformation of the data

```r
# Recoding the SpecialDay column to a categorical

df$SpecialDayGroup <- cut(df$SpecialDay,
                          breaks = c(-Inf, .2, .4, .6, .8, 1, Inf),
                          labels = c("Far", "Moderately Close",
                                     "Close", "Very Close",
```

```
                                "Too Close for on-time delivery", "Ideal Time"),
                      include.lowest = TRUE,
                      right = FALSE)
```

The numerical values in the SpecialDay field represent the closeness of the site visiting time to a specific special day (e.g. Mother's Day, Valentine's Day, etc.) in which the sessions are more likely to be finalized with a transaction. They have little meaning as numerical values, so they were recoded to categorical values.

```
# Converting Month column to factor
df$Month_factor <- as.factor(df$Month)
```

The Month column was converted into a factor type for comparison to categorical variables.

```
# Converting BounceRates and ExitRates into percentage values

df_percent <- df
df_percent$BounceRates <- df_percent$BounceRates * 100
df_percent$ExitRates <- df_percent$ExitRates * 100
```

Given the other values in the data set representing whole units (e.g. single page view, $1 value, 1 sec duration, etc.), it is easier to capture the data by converting the Bounce and Exit rates from decimals to percent values.

Cleaned dataframe

```
data <- df_percent
```

Data Analysis and Visualization

```
# Changing the levels of SpecialDayGroup for plotting purposes

data$SpecialDayGroup <- factor(data$SpecialDayGroup,
                               levels = c("Far", "Moderately Close",
                                          "Close", "Very Close",
                                          "Too Close for on-time delivery", "Ideal Time"))

# Changing the Levels of the Month_factor so that the Months are in order

data$Month_factor <- factor(data$Month_factor, levels = c("Feb", "Mar",
                                                          "May", "June",
                                                          "Jul", "Aug",
                                                          "Sep", "Oct",
                                                          "Nov", "Dec"))
# Data sets used for plotting

# Removing 'Far' from Special day
data_far_removed <- data[data$SpecialDayGroup != "Far", ]
# Keeping only True values for Revenue
only_true <- data_far_removed[data_far_removed$Revenue != "False", ]
# Only True values for Revenue in the full data set
only_true_full <- data[data$Revenue != "False", ]
```

Descriptive Statistics and Visualizations for Measures of Distribution and Centrality

```r
# Descriptive Statistics of Numerical Data
describe(data[, 1:9])
```

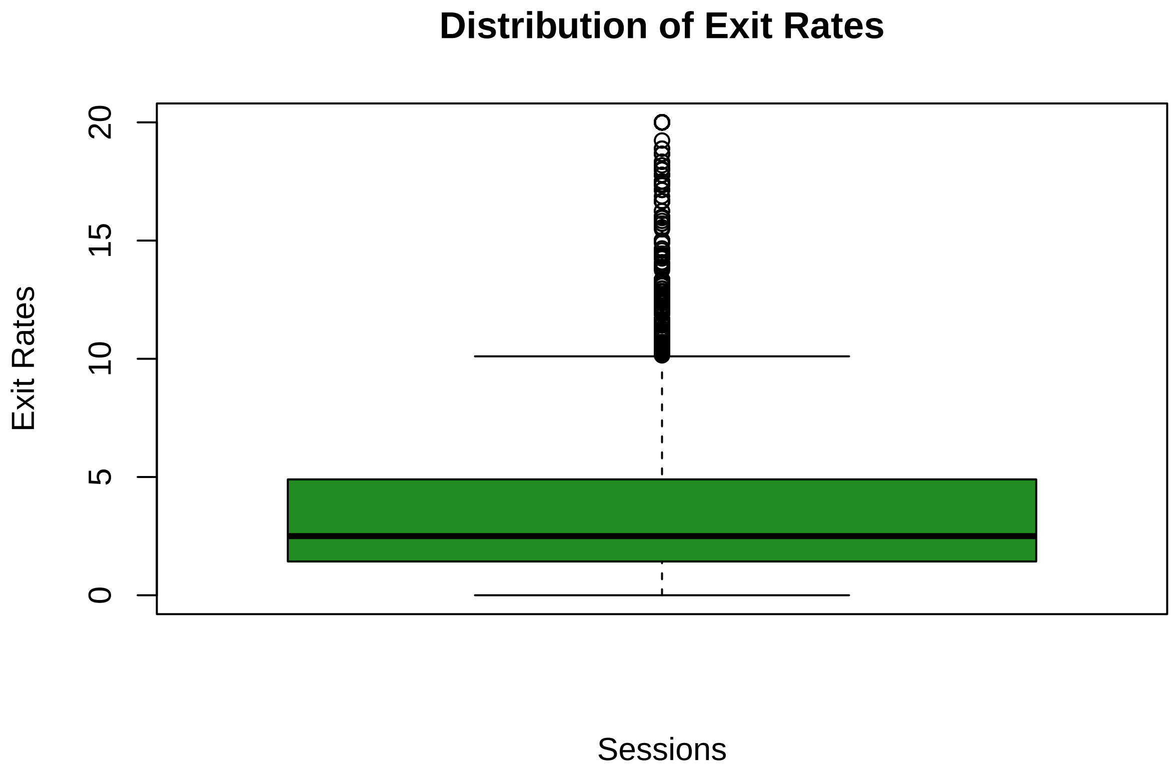
```
                        vars     n    mean      sd median trimmed    mad min
Administrative             1 12086    2.33    3.33   1.00    1.65   1.48   0
Administrative_Duration    2 12086   81.57  176.74   9.00   42.79  13.34   0
Informational              3 12086    0.51    1.27   0.00    0.18   0.00   0
Informational_Duration     4 12086   34.84  141.74   0.00    3.70   0.00   0
ProductRelated             5 12086   32.00   44.47  18.00   23.01  19.27   0
ProductRelated_Duration    6 12086 1204.50 1913.15 608.86  830.36 744.97   0
BounceRates                7 12086    2.05    4.54   0.29    0.82   0.44   0
ExitRates                  8 12086    4.16    4.63   2.50    3.14   2.06   0
PageValues                 9 12086    5.91   18.66   0.00    1.30   0.00   0
                             max    range skew kurtosis    se
Administrative             27.00    27.00 1.95     4.67  0.03
Administrative_Duration  3398.75  3398.75 5.52    49.05  1.61
Informational              24.00    24.00 4.02    26.79  0.01
Informational_Duration   2549.38  2549.38 7.55    75.50  1.29
ProductRelated            705.00   705.00 4.31    30.83  0.40
ProductRelated_Duration 63973.52 63973.52 7.27   138.50 17.40
BounceRates                20.00    20.00 3.15     9.24  0.04
ExitRates                  20.00    20.00 2.23     4.60  0.04
PageValues                361.76   361.76 6.39    65.51  0.17
```

```r
boxplot(data$Administrative,
        col = "forestgreen",
        varwidth = TRUE,
        main = "Admin Sites Visited During Session",
        xlab = "Sessions",
        ylab = "Sites Visited")

boxplot(data$Administrative_Duration,
        col = "forestgreen",
        varwidth = TRUE,
        main = "Duration of Time Spent on Admin Site",
        xlab = "Sessions",
        ylab = "Duration")

boxplot(data$Informational,
        col = "forestgreen",
        varwidth = TRUE,
        main = "Information Sites Visited During Session",
        xlab = "Sessions",
        ylab = "Sites Visited")

boxplot(data$Informational_Duration,
        col = "forestgreen",
        varwidth = TRUE,
        main = "Duration of Time Spent on Information Site",
        xlab = "Sessions",
        ylab = "Duration")
```

```r
boxplot(data$ProductRelated,
        col = "forestgreen",
        varwidth = TRUE,
        main = "Product Sites Visited During Session",
        xlab = "Sessions",
        ylab = "Sites Visited")

boxplot(data$ProductRelated_Duration,
        col = "forestgreen",
        varwidth = TRUE,
        main = "Duration of Time Spent on Product Site",
        xlab = "Sessions",
        ylab = "Duration")

boxplot(data$BounceRates,
        col = "forestgreen",
        varwidth = TRUE,
        main = "Distribution of Bounce Rates",
        xlab = "Sessions",
        ylab = "Bounce Rates")

boxplot(data$PageValues,
        col = "forestgreen",
        varwidth = TRUE,
        main = "Distribution of Page Values",
        xlab = "Sessions",
        ylab = "Page Values")
```

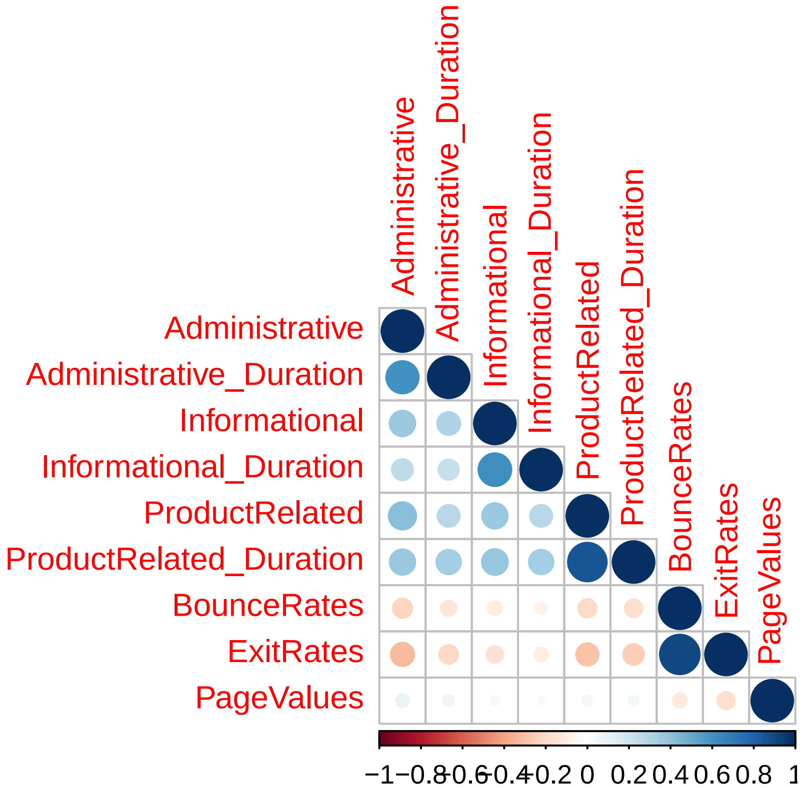
The data for the numerical values is highly skewed to the right due to the presence of several outliers and the majority of the data being concentrated around or at 0. We'll use an example of the boxplot distribution of Exit Rates to illustrate the distribution below. Exit Rates have one of the lowest skews, so the distributions of the other box plots are either similar or more extreme.

```r
boxplot(data$ExitRates,
        col = "forestgreen",
        varwidth = TRUE,
        main = "Distribution of Exit Rates",
        xlab = "Sessions",
        ylab = "Exit Rates")
```


Distribution of Exit Rates
Exit Rates
20
15
10
5
0
Sessions

Correlation Plot of Numerical Data

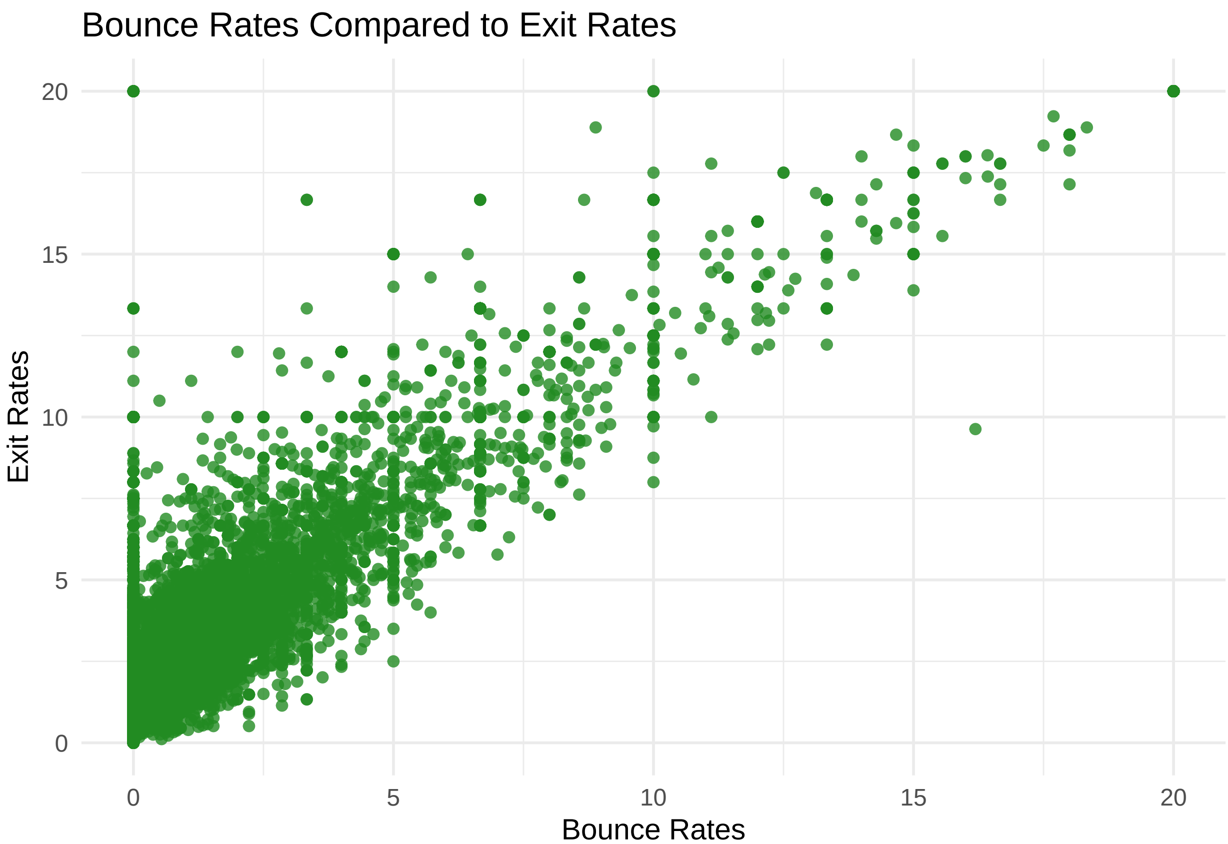
```
correlation <- cor(data[, 1:9])

corrplot(correlation, method = "circle", type = "lower")
```



The correlation plot illustrates a strong positive correlation between the Product Related and Product Related Duration fields as well as the Exit Rates and Bounce Rates, which we can visualize with scatter plots.

Scatter Plots of Highly Correlated Numerical Values

```
ggplot(data, aes(x = BounceRates, y = ExitRates)) +
  geom_point(color = "forestgreen", alpha = .8) +
  labs(title = "Bounce Rates Compared to Exit Rates",
       x = "Bounce Rates",
       y = "Exit Rates") +
  theme_minimal()
```

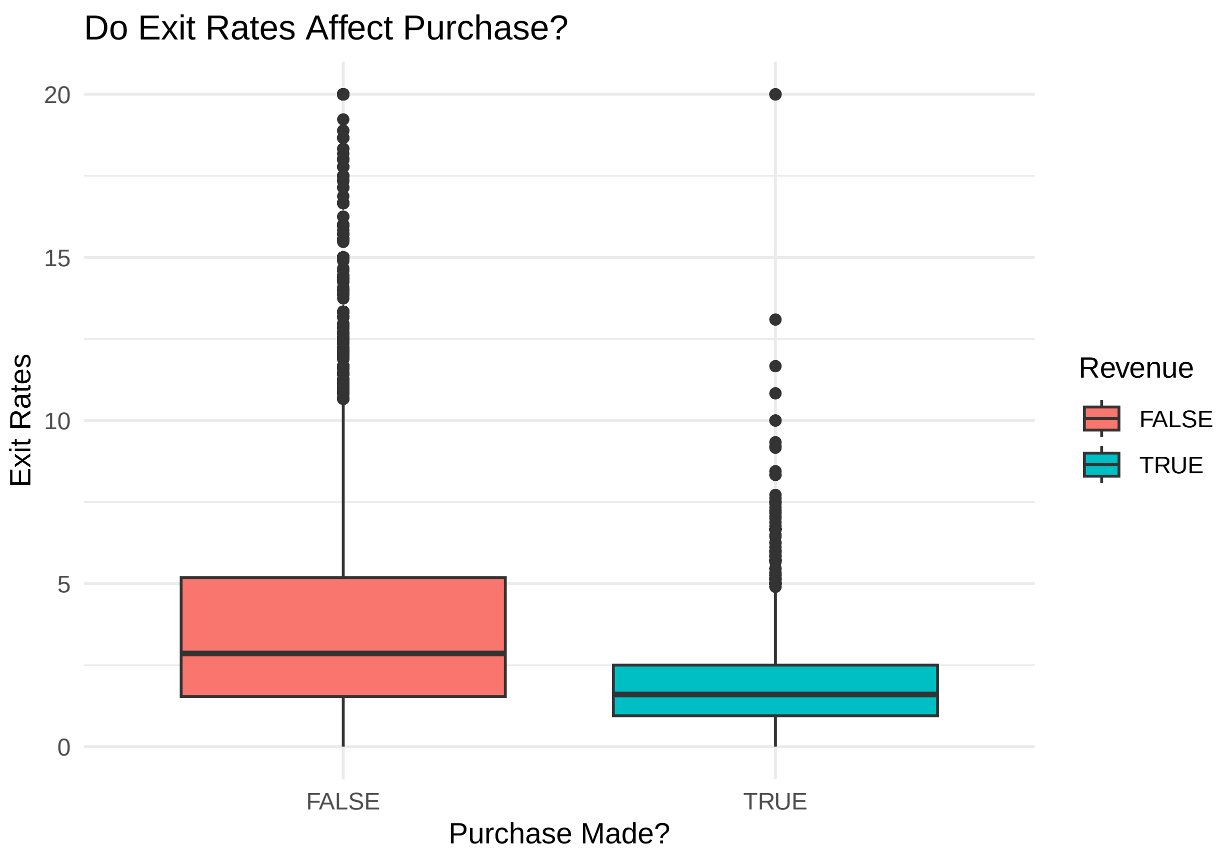



```
ggplot(data, aes(x = ProductRelated, y = ProductRelated_Duration)) +
  geom_point(color = "forestgreen", alpha = .8) +
  labs(title = "Product Sites Visited Compared to Time Spent",
       x = "Number of Product Related Sites Visited During Session",
       y = "Duration of Time Spent") +
  theme_minimal()
```

As Revenue is our dependent variable, we first explored the relationship, if any, of Revenue with the other numerical values. Two of the more significant relationships can be visualized below. Exit Rates appear to have a strong negative relationship while Page Values appear to have a strong positive relationship.

Boxplots Comparing Revenue to Numerical Variables with Significance

```
ggplot(data, aes(x = Revenue, y = ExitRates, fill = Revenue)) +
  geom_boxplot() +
  labs(title = "Do Exit Rates Affect Purchase?",
       x = "Purchase Made?",
       y = "Exit Rates") +
  theme_minimal()
```




```
ggplot(data, aes(x = Revenue, y = PageValues, fill = Revenue)) +
  geom_boxplot() +
  labs(title = "Do Higher Page Values Lead to More Purchases?",
       x = "Purchase Made?",
       y = "Page Value") +
  theme_minimal()
```

The following are bar charts to visualize the distribution of the categorical variables among their different levels.

Bar Charts of Categorical Data

```r
# SpecialDayGroup Bar Chart (Log scale and True scale)
# The log scale version was included to account for the heavy class imbalance of the
# 'Far' category.


ggplot(data, aes(x = SpecialDayGroup, fill = SpecialDayGroup)) +
  geom_bar(alpha = .8) +
  labs(title = "Special Day",
       x = "On Time Delivery for Holiday?",
       y = "Number of Sessions (True Scale)",
       fill = "Proximity to Special Day") +
  theme_minimal() +
  scale_x_discrete(labels = function(x) str_wrap(x, width = 10)) +
  scale_fill_viridis_d(option = "magma")

ggplot(data, aes(x = SpecialDayGroup, fill = SpecialDayGroup)) +
  geom_bar(alpha = .8) +
  scale_y_log10() +
  labs(title = "Special Day",
       x = "On Time Delivery for Holiday?",
       y = "Number of Sessions (log scale)",
       fill = "Proximity to Special Day") +
```

```r
  theme_minimal() +
  scale_x_discrete(labels = function(x) str_wrap(x, width = 10)) +
  scale_fill_viridis_d(option = "magma")

# Month of Session Chart

ggplot(data, aes(x = Month_factor, fill = Month_factor)) +
  geom_bar(alpha = .8) +
  labs(title = "Month of Session",
       x = "Month",
       y = "Number of Sessions",
       fill = "Month") +
  theme_minimal() +
  scale_fill_viridis_d(option = "D")

# Weekend Chart

ggplot(data, aes(x = Weekend)) +
  geom_bar(fill = "forestgreen", alpha = .8) +
  labs(title = "Weekend Sessions",
       x = "Weekend?",
       y = "Number of Sessions") +
  theme_minimal()

# Visitor Chart

ggplot(data, aes(x = VisitorType)) +
  geom_bar(fill = "forestgreen", alpha = .8) +
  labs(title = "Returning Visitor or New?",
       x = "Type of Visitor",
       y = "Number of Sessions") +
  theme_minimal()

#Revenue Chart

ggplot(data, aes(x = Revenue)) +
  geom_bar(fill = "forestgreen", alpha = .8) +
  labs(title = "Was a Purchase Made This Session?",
       x = "Purchase",
       y = "Number of Sessions") +
  theme_minimal()
```

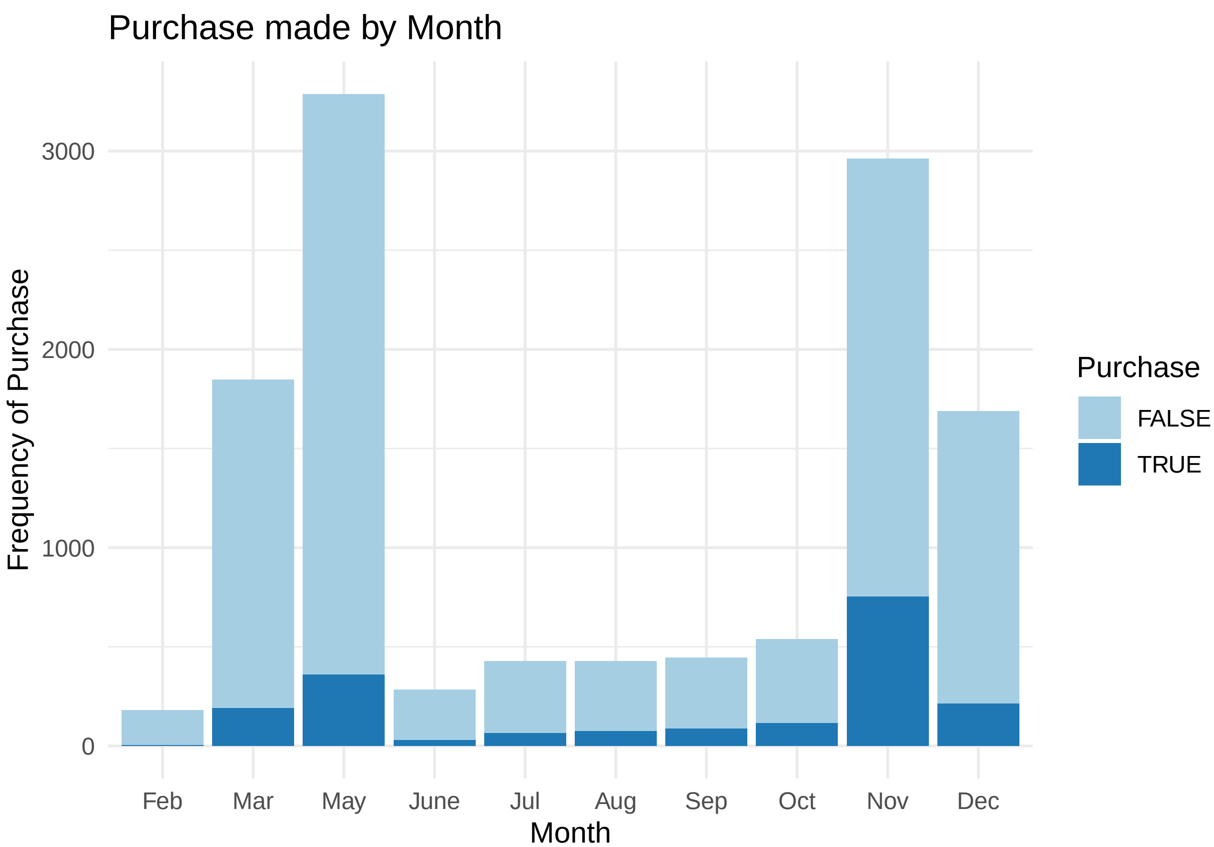
Stacked Bar Charts and Heat Maps Comparing Revenue to Other Significant Predictors

```r
#Proportions Table then Stacked Bar (Month ~ Revenue)

contingency_table <- table(data$Month, data$Revenue)
proportions <- prop.table(contingency_table, margin = 1)
proportions
```

```
              FALSE        TRUE
```

```
  Aug   0.82435597 0.17564403
  Dec   0.87314760 0.12685240
  Feb   0.98342541 0.01657459
  Jul   0.84777518 0.15222482
  June  0.90106007 0.09893993
  Mar   0.89713048 0.10286952
  May   0.89051095 0.10948905
  Nov   0.74569402 0.25430598
  Oct   0.78849722 0.21150278
  Sep   0.80717489 0.19282511
```

```r
ggplot(data, aes(x = Month_factor, fill = Revenue)) +
  geom_bar() +
  labs(title = "Purchase made by Month",
       x = "Month",
       y = "Frequency of Purchase",
       fill = "Purchase") +
  theme_minimal() +
  scale_fill_brewer(palette = "Paired")
```

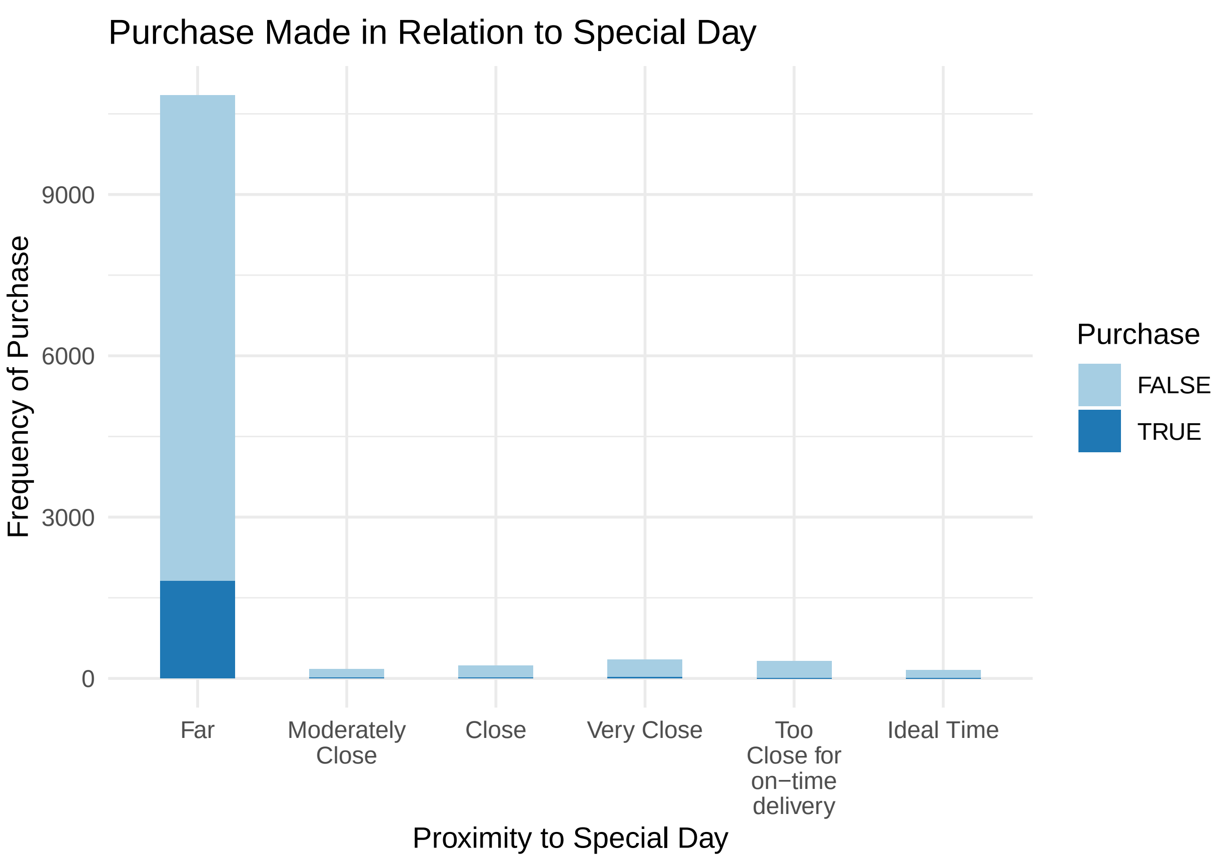


November has the most significant relationship to Revenue compared to the other months. A proportional table was included to put the relationships into a better context. From the stacked bar chart, May appears to have the second most significant relationship, however, the proportional chart shows that October has the second highest likelihood of a purchase being made, despite May having a higher absolute number of purchases. This highlights a potential area of opportunity to convert more sessions to purchases during the month of May, or increase traffic during the month of October.

```r
# Proportions Table then Stacked Bar with and without "Far" (SpecialDayGroup ~ Revenue)

group_rev <- table(data$SpecialDayGroup, data$Revenue)
group_rev_prop <- prop.table(group_rev, margin = 1)
group_rev_prop
```

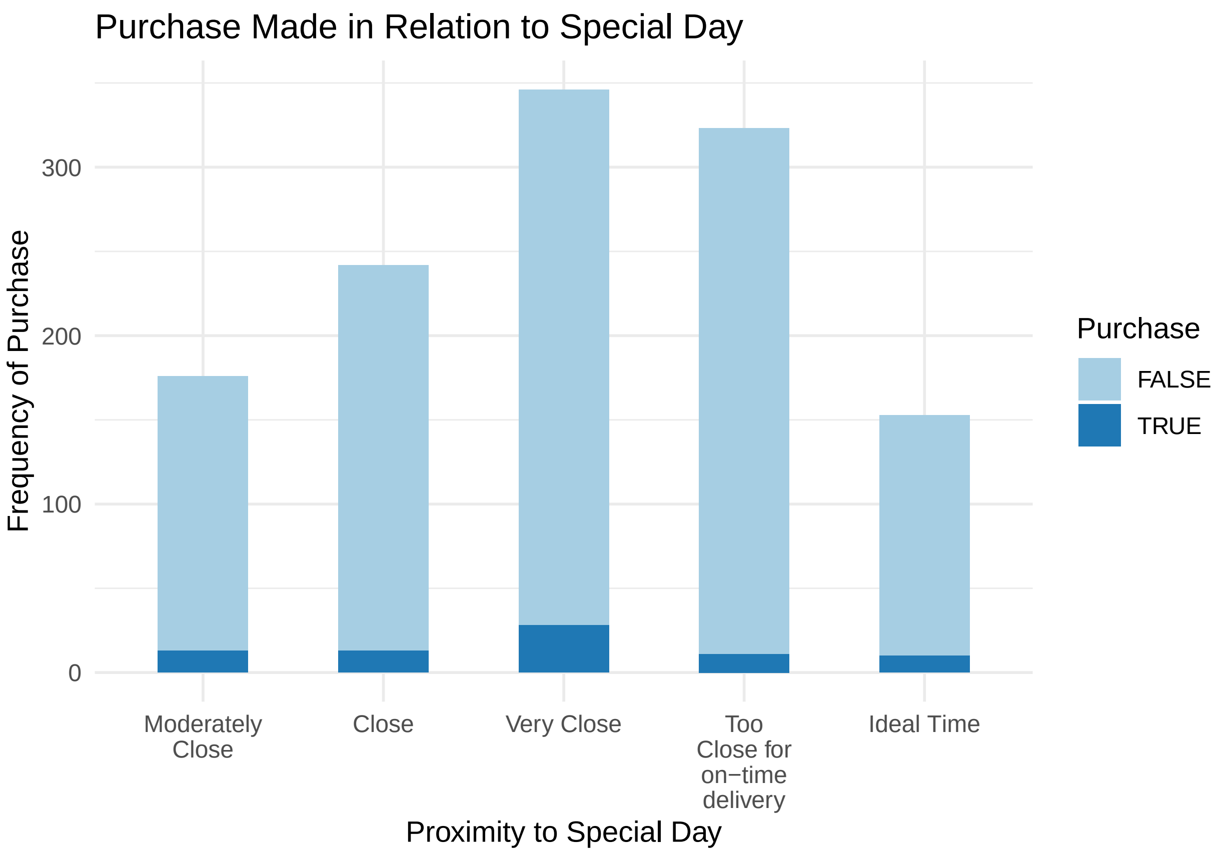
```
                                  FALSE       TRUE
  Far                        0.83284160 0.16715840
  Moderately Close           0.92613636 0.07386364
  Close                      0.94628099 0.05371901
  Very Close                 0.91907514 0.08092486
  Too Close for on-time delivery 0.96594427 0.03405573
  Ideal Time                 0.93464052 0.06535948
```

```r
ggplot(data, aes(x = SpecialDayGroup, fill = Revenue)) +
  geom_bar(width = .5) +
  labs(title = "Purchase Made in Relation to Special Day",
       x = "Proximity to Special Day",
       y = "Frequency of Purchase",
       fill = "Purchase") +
  theme_minimal() +
  scale_fill_brewer(palette = "Paired") +
  scale_x_discrete(labels = function(x) str_wrap(x, width = 10))
```




```r
ggplot(data_far_removed, aes(x = SpecialDayGroup, fill = Revenue)) +
  geom_bar(width = .5) +
```

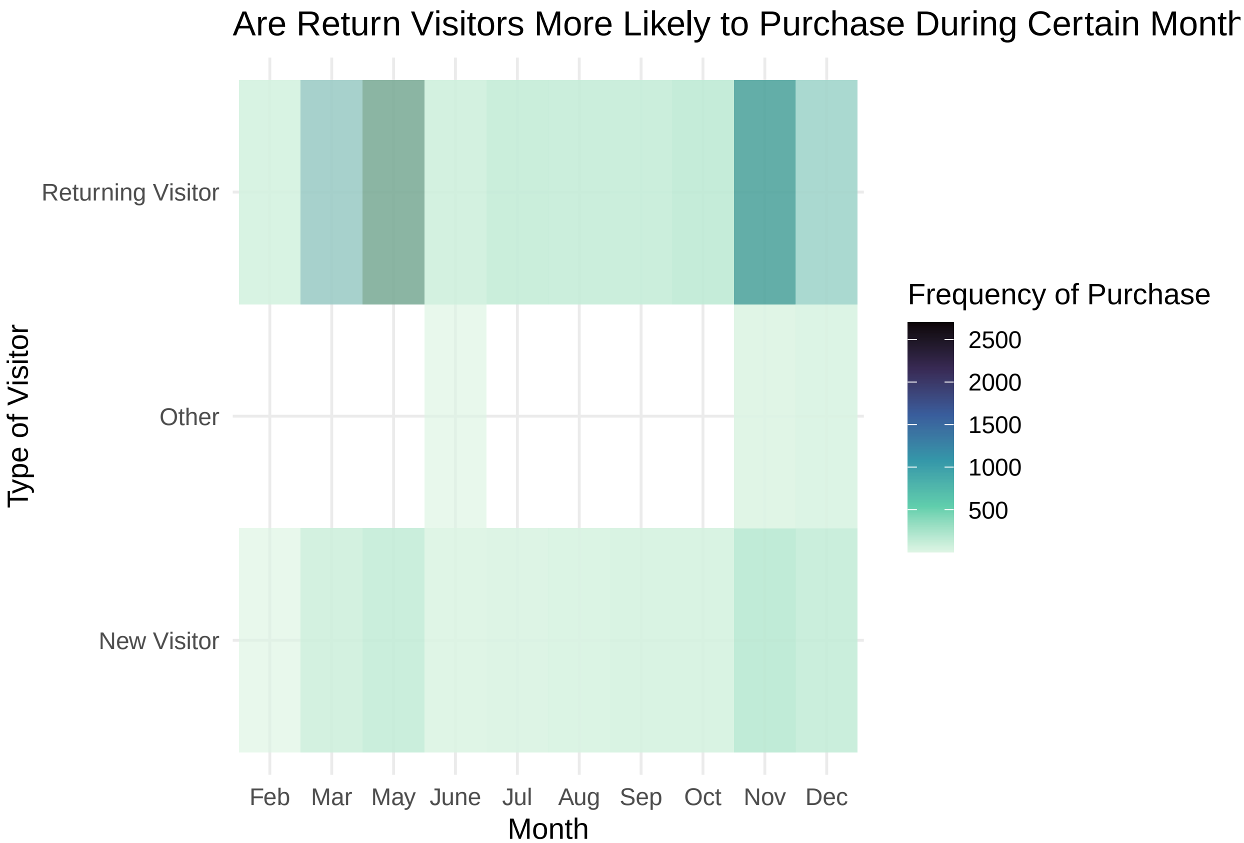
```r
  labs(title = "Purchase Made in Relation to Special Day",
       x = "Proximity to Special Day",
       y = "Frequency of Purchase",
       fill = "Purchase") +
  theme_minimal() +
  scale_fill_brewer(palette = "Paired") +
  scale_x_discrete(labels = function(x) str_wrap(x, width = 10))
```



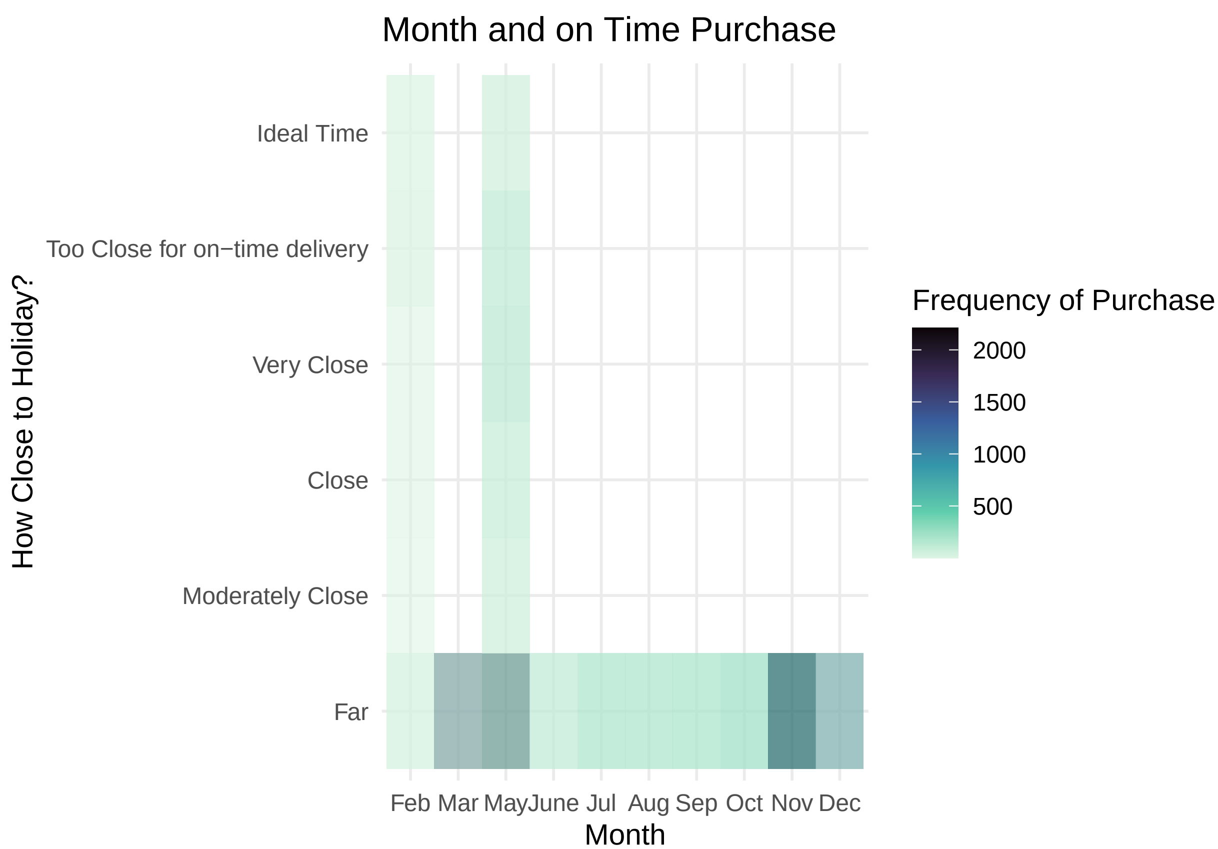
Special day group shows a higher likelihood, both in an absolute and in a proportional sense, of purchases being made far away from any sort of special day. We included a graph with the 'Far' category removed to get a clearer sense of the relationship with the other categories.

```r
# Heat map to compare whether returning visitor were more likely to make purchases in
# certain months than others

heatmap_data <- only_true_full %>%
  group_by(Month_factor, VisitorType, Revenue) %>%
  summarise(Frequency = n(), groups = "drop")

ggplot(heatmap_data, aes(x = Month_factor, y = VisitorType, fill = Frequency)) +
  geom_tile(alpha = .7) +
  scale_fill_viridis_c(option = "mako", direction = -1) +
  labs(title = "Are Return Visitors More Likely to Purchase During Certain Months?",
       x = "Month",
       y = "Type of Visitor",
       fill = "Frequency of Purchase") +
  scale_y_discrete(labels = c("Returning_Visitor" = "Returning Visitor",
```

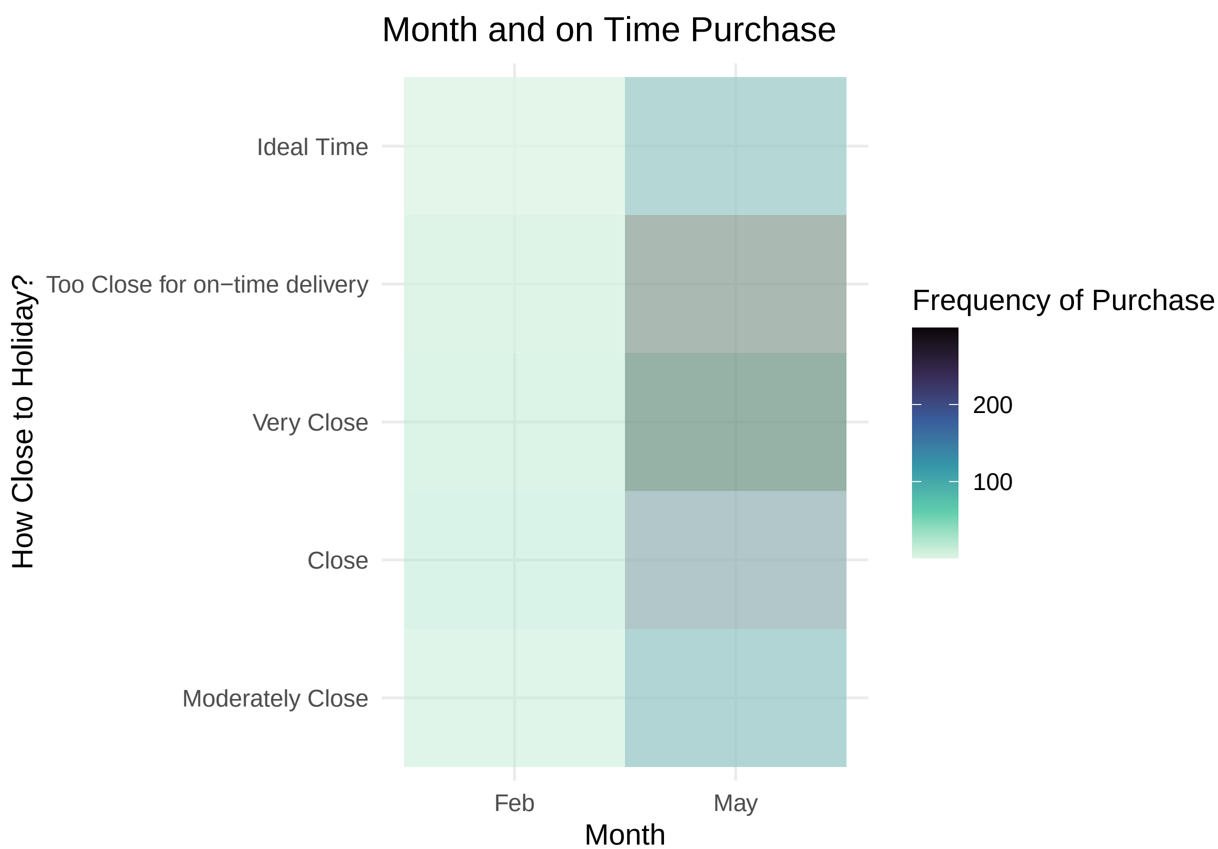
```
                                              "New_Visitor" = "New Visitor")) +
  theme_minimal()
```




```
# Heatmap to show the correlation between Month and Special Day

heatmap_data2 <- only_true_full %>%
  group_by(Month_factor, SpecialDayGroup, Revenue) %>%
  summarise(Frequency = n(), groups = "drop")

ggplot(heatmap_data2, aes(x = Month_factor, y = SpecialDayGroup, fill = Frequency)) +
  geom_tile(alpha = .5) +
  labs(title = "Month and on Time Purchase",
       x = "Month",
       y = "How Close to Holiday?",
       fill = "Frequency of Purchase") +
  scale_fill_viridis_c(option = "mako", direction = -1) +
  scale_x_discrete(labels = function(x) str_wrap(x, width = 10)) +
  theme_minimal()
```




```r
# Drop 'Far' from data to zoom in on the two months where purchases were made close to
# the holiday

heatmap_data3 <- only_true %>%
  group_by(Month_factor, SpecialDayGroup, Revenue) %>%
  summarise(Frequency = n(), groups = "drop")

ggplot(heatmap_data3, aes(x = Month_factor, y = SpecialDayGroup, fill = Frequency)) +
  geom_tile(alpha = .5) +
  labs(title = "Month and on Time Purchase",
       x = "Month",
       y = "How Close to Holiday?",
       fill = "Frequency of Purchase") +
  scale_fill_viridis_c(option = "mako", direction = -1) +
  scale_x_discrete(labels = function(x) str_wrap(x, width = 10)) +
  theme_minimal()
```



Data Analytics

```
# Recode Month and VisitorType to individual columns
data <- data %>%
  mutate(MonthFeb = ifelse(Month == "Feb", TRUE, FALSE),
         MonthMar = ifelse(Month == "Mar", TRUE, FALSE),
         MonthMay = ifelse(Month == "May", TRUE, FALSE),
         MonthJune = ifelse(Month == "June", TRUE, FALSE),
         MonthJul = ifelse(Month == "Jul", TRUE, FALSE),
         MonthSep = ifelse(Month == "Sep", TRUE, FALSE),
         MonthOct = ifelse(Month == "Oct", TRUE, FALSE),
         MonthNov = ifelse(Month == "Nov", TRUE, FALSE),
         MonthDec = ifelse(Month == "Dec", TRUE, FALSE))
data <- data %>%
  mutate(ReturningVisitor = ifelse(VisitorType == "Returning_Visitor", TRUE, FALSE))
data$MonthFeb <- as.integer(data$MonthFeb)
data$MonthMar <- as.integer(data$MonthMar)
data$MonthMay <- as.integer(data$MonthMay)
data$MonthJune <- as.integer(data$MonthJune)
data$MonthJul <- as.integer(data$MonthJul)
data$MonthSep <- as.integer(data$MonthSep)
data$MonthOct <- as.integer(data$MonthOct)
data$MonthNov <- as.integer(data$MonthNov)
data$MonthDec <- as.integer(data$MonthDec)
data$ReturningVisitor <- as.integer(data$ReturningVisitor)
data$ProductRelated_Duration <- scale(data$ProductRelated_Duration)
```

```r
data$ExitRates <- scale(data$ExitRates)
data$PageValues <- scale(data$PageValues)
```

Final Logistic Regression Model

- Most significant columns: ProductRelated_Duration, ExitRates, PageValues, Month, VisitorType
- Most significant predictor variables: ProductRelated_Duration, ExitRates, PageValues, MonthDec, MonthFeb, MonthMar, MonthMay, MonthNov, Returning_Visitor
- Accuracy: 0.8917
- Pseudo R-Squared Value: 0.3160
- VIF for Multicollinearity: All < 5, therefore no multicollinearity is present
- Confusion Matrix: Correct Classified as False = 1904; Correct Classified as True = 212; Incorrect Classified as False = 146; Incorrect Classified as True = 111
- Proportion of False in Train data: 8183/9713 = 0.8425
- Proportion of False in Test data: 2015/2373 = 0.8491
- PR-Curve AUC: 0.6429
- ROC AUC: 0.9014
- Precision Score: 0.6563
- Recall Score: 0.5922
- F1 Score: 0.6226
- Generalized Regression Formula:
- Revenue = -1.83 + 0.20(ProductRelated_Duration) - 0.84(ExitRates) + 1.54(PageValues) - 1.68(MonthFeb) - 0.46(MonthMar) - 0.59(MonthMay) + 0.60(MonthNov) - 0.54(MonthDec) - 0.24(ReturningVisitor)

```r
# Set Seed
set.seed(123)

# Create Train and Test data sets
sample <- sample(c(TRUE, FALSE), nrow(data), replace = TRUE, prob = c(0.8, 0.2))
train <- data[sample, ]
test <- data[!sample, ]

# Logistic Regression Model
model <- glm(Revenue ~ ProductRelated_Duration + ExitRates + PageValues +
                       MonthFeb + MonthMar + MonthMay +
                       MonthNov + MonthDec + ReturningVisitor,
            family = "binomial",
            data = train)
summary(model)
```

```
Call:
glm(formula = Revenue ~ ProductRelated_Duration + ExitRates +
    PageValues + MonthFeb + MonthMar + MonthMay + MonthNov +
    MonthDec + ReturningVisitor, family = "binomial", data = train)

Coefficients:
                        Estimate Std. Error z value Pr(>|z|)
(Intercept)             -1.82762    0.11580 -15.782  < 2e-16 ***
ProductRelated_Duration  0.20011    0.02794   7.161 8.02e-13 ***
ExitRates               -0.83533    0.08461  -9.873  < 2e-16 ***
PageValues               1.54206    0.05086  30.320  < 2e-16 ***
MonthFeb                -1.68138    0.63341  -2.654 0.007943 **
```

```
MonthMar               -0.46110    0.12693  -3.633 0.000281 ***
MonthMay               -0.59342    0.11035  -5.378 7.54e-08 ***
MonthNov                0.60066    0.09573   6.274 3.51e-10 ***
MonthDec               -0.54458    0.13139  -4.145 3.40e-05 ***
ReturningVisitor       -0.23960    0.09469  -2.530 0.011395 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 8460.7  on 9712  degrees of freedom
Residual deviance: 5787.3  on 9703  degrees of freedom
AIC: 5807.3

Number of Fisher Scoring iterations: 6
```

```r
# Predicted results and Accuracy
predicted <- predict(model, test, type = "response")
predicted_results <- ifelse(predicted > 0.28, 1, 0)

misClasificError <- mean(predicted_results != test$Revenue)
print(paste("Accuracy", 1-misClasificError))
```

```
[1] "Accuracy 0.891698272229246"
```

```r
# Calculation of pseudo R-Squared Value
# pscl comes from library(pscl)
pscl::pR2(model)["McFadden"]
```

```
fitting null model for pseudo-r2

 McFadden
0.3159766
```

```r
# Check the predictor variables for multicollinearity
car::vif(model)
```

```
ProductRelated_Duration               ExitRates              PageValues
               1.105437                1.095618                1.053857
               MonthFeb                MonthMar                MonthMay
               1.013645                1.407052                1.607560
               MonthNov                MonthDec        ReturningVisitor
               1.835518                1.367762                1.106296
```

```r
# Confusion Matrix to check false positive and false negatives
conf_matrix <- table(Predicted = predicted_results, Actual = test$Revenue)
cat(sprintf("Confusion Matrix:\n"))
```

```
Confusion Matrix:
```

```r
conf_matrix
```

```
         Actual
Predicted FALSE TRUE
        0  1904  146
        1   111  212
```

```r
# Tables to count the observed True/False in the Train and Test sets
cat(sprintf("Table Count for True/False in Train Dataset:"))
```

```
Table Count for True/False in Train Dataset:
```

```r
table(train$Revenue)
```

```
FALSE  TRUE
 8183  1530
```

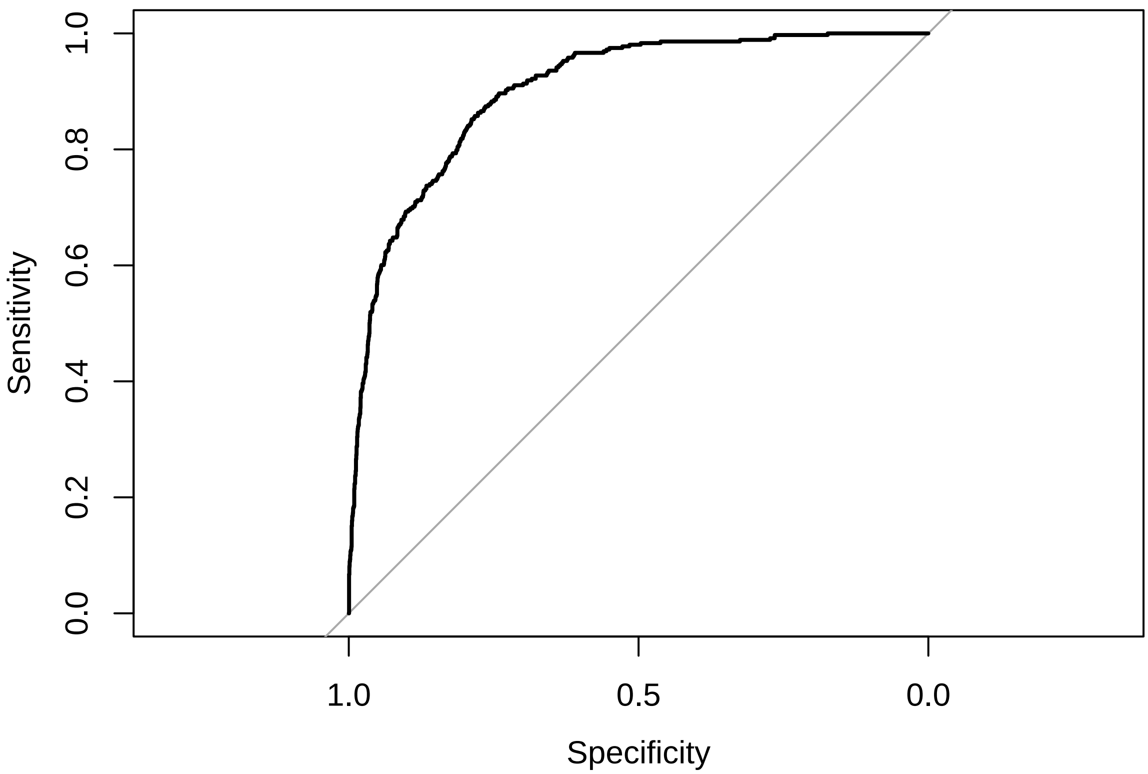
```r
cat(sprintf("\nTable Count for True/False in Test Dataset:"))
```

```
Table Count for True/False in Test Dataset:
```

```r
table(test$Revenue)
```

```
FALSE  TRUE
 2015   358
```

```r
# Receiver Operating Characteristic Curve
# roc() comes from library(pROC)
roc_curve <- roc(test$Revenue, predicted)
plot(roc_curve)
```

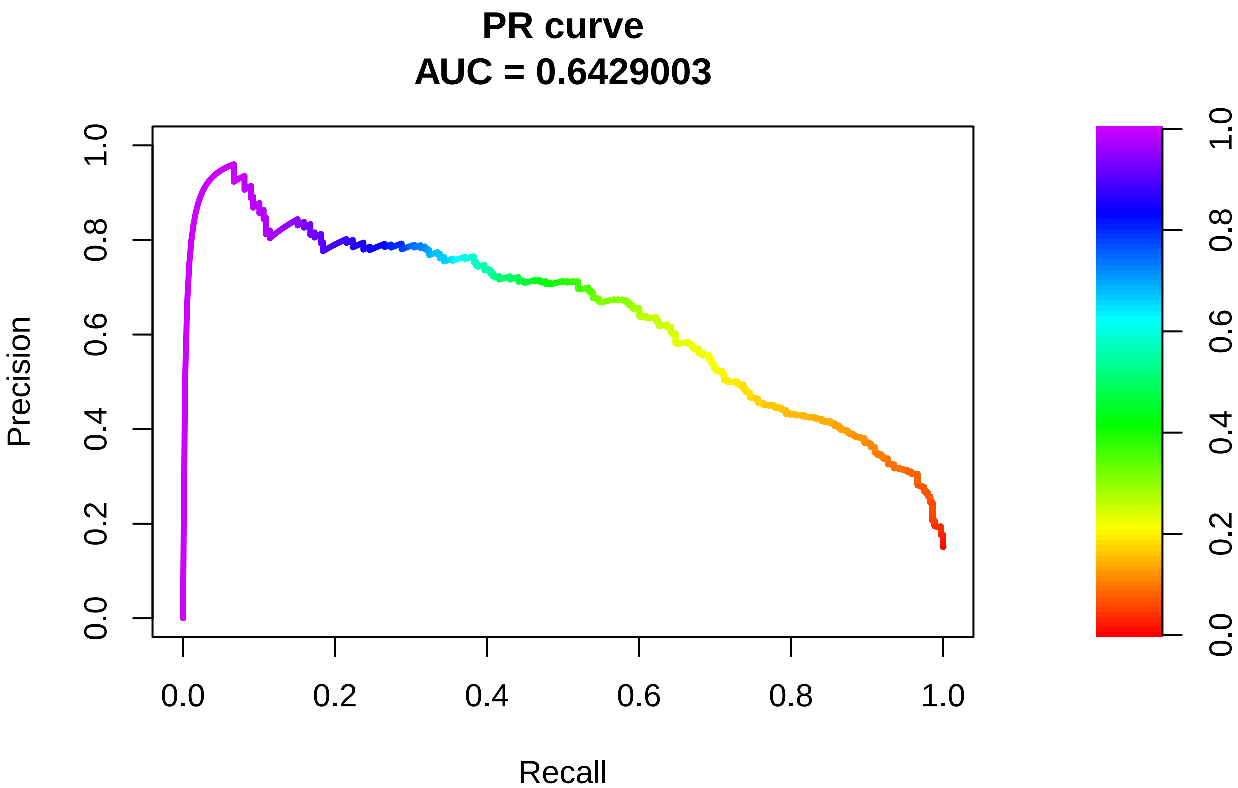



```r
auc_value <- auc(roc_curve)
print(auc_value)
```

```
Area under the curve: 0.9014
```

```r
# Precision-Recall Curve
# pr.curve() comes from library(PRROC)
pr_curve <- pr.curve(scores.class0 = predicted[test$Revenue == 1],
                     scores.class1 = predicted[test$Revenue == 0],
```

```
                    curve = TRUE)

plot(pr_curve)
```




```
precision <- conf_matrix[2, 2] / sum(conf_matrix[2, ])
recall <- conf_matrix[2, 2] / sum(conf_matrix[, 2])
print(paste("Precision Score:", precision))
```

```
[1] "Precision Score: 0.656346749226006"
```

```
print(paste("Recall Score:", recall))
```

```
[1] "Recall Score: 0.592178770949721"
```

```
f1_score <- 2 * ((precision * recall) / (precision + recall))
print(paste("F1 Score:", f1_score))
```

```
[1] "F1 Score: 0.622613803230543"
```

References

Encord Blog. (2023, July 18) F1 Score in Machine Learning. Encord. https://encord.com/blog/f1-score-in-machine-learning/#:~:text=The%20F1%20score%20ranges%20between%200%20and%201%2C,can%20concurrently%20attain%20high%20precision%20and%20high%20recall.

Evidently AI. (n.d.) How to explain the ROC curve and ROC AUC score. Evidently AI. https://www.evidentlyai.com/classification-metrics/explain-roc-curve

One-Off Coder. (2024, Apr. 03). Pseudo r-squared for logistic regression. Data Science Topics. https://datascience.oneoffcoder.com/psuedo-r-squared-logistic-regression.html

Shah, C. (2020). Hands-on Introduction to Data Science. Cambridge University Press. https://doi.org/10.1017/9781108560412