To Buy or not to Buy? – An Application of Classification Models on Online Shopping Intention Data

Wen Cai

Overview

The purpose of this analysis project is to combine my understanding of online shopping in the retail industry with my knowledge in data analytics, to demonstrate the analytical techniques I have learned from the MicroMasters Program in Analytics offered by Georgia Institute of Technology.

The Online Shoppers Purchasing Intention Dataset from the UCI Machine Learning Repository, has 12,330 data points and consists of both numerical and categorical attributes. The models I have used here include logistic regression, support vector machines, k-nearest-neighbour, which help to detect online shoppers purchasing patterns and forcast their intention. The outline of the analysis is as follows:

- Conduct exploratory analysis;
- Develop research questions about the data;
- Complete data preprocessing for the modeling;
- Apply learning algorithm to compare various models' performances and answer the questions.

#1. Exploratory Analysis

There are ten numerical and eight categorical variables. The last variable *Revenue* can be used as the class label and needs to be converted to 1s or 0s for the classification models in the analysis. There is no missing value in this dataset.

```r
rm(list = ls())

library(dplyr)
```

```
##
## Attaching package: 'dplyr'

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

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

```r
library(ggplot2)
library(ggcorrplot)

data <- read.csv("online_shoppers_intention.csv", stringsAsFactors = FALSE, header = TRUE)

str(data)
```

```
## 'data.frame':    12330 obs. of  18 variables:
##  $ Administrative         : int  0 0 0 0 0 0 0 1 0 0 ...
##  $ Administrative_Duration: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Informational          : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Informational_Duration : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ ProductRelated         : int  1 2 1 2 10 19 1 0 2 3 ...
##  $ ProductRelated_Duration: num  0 64 0 2.67 627.5 ...
##  $ BounceRates            : num  0.2 0 0.2 0.05 0.02 ...
##  $ ExitRates              : num  0.2 0.1 0.2 0.14 0.05 ...
##  $ PageValues             : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ SpecialDay             : num  0 0 0 0 0 0 0.4 0 0.8 0.4 ...
##  $ Month                  : chr  "Feb" "Feb" "Feb" "Feb" ...
##  $ OperatingSystems       : int  1 2 4 3 3 2 2 1 2 2 ...
##  $ Browser                : int  1 2 1 2 3 2 4 2 2 4 ...
##  $ Region                 : int  1 1 9 2 1 1 3 1 2 1 ...
##  $ TrafficType            : int  1 2 3 4 4 3 3 5 3 2 ...
##  $ VisitorType            : chr  "Returning_Visitor" "Returning_Visitor" "Returning_Visitor" "Returni
##  $ Weekend                : logi  FALSE FALSE FALSE FALSE TRUE FALSE ...
##  $ Revenue                : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
```

```r
sum(is.na(data))
```

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

```r
data[data == "?"] # sometimes the question mark is used to indicate missing values
```

```
## character(0)
```

85.4% (10,422) of the customers did not complete the transaction while those who completed transactions, only take up 15.5% (1908) of the dataset. Around 26.15% of the online shopping happened at weekends.

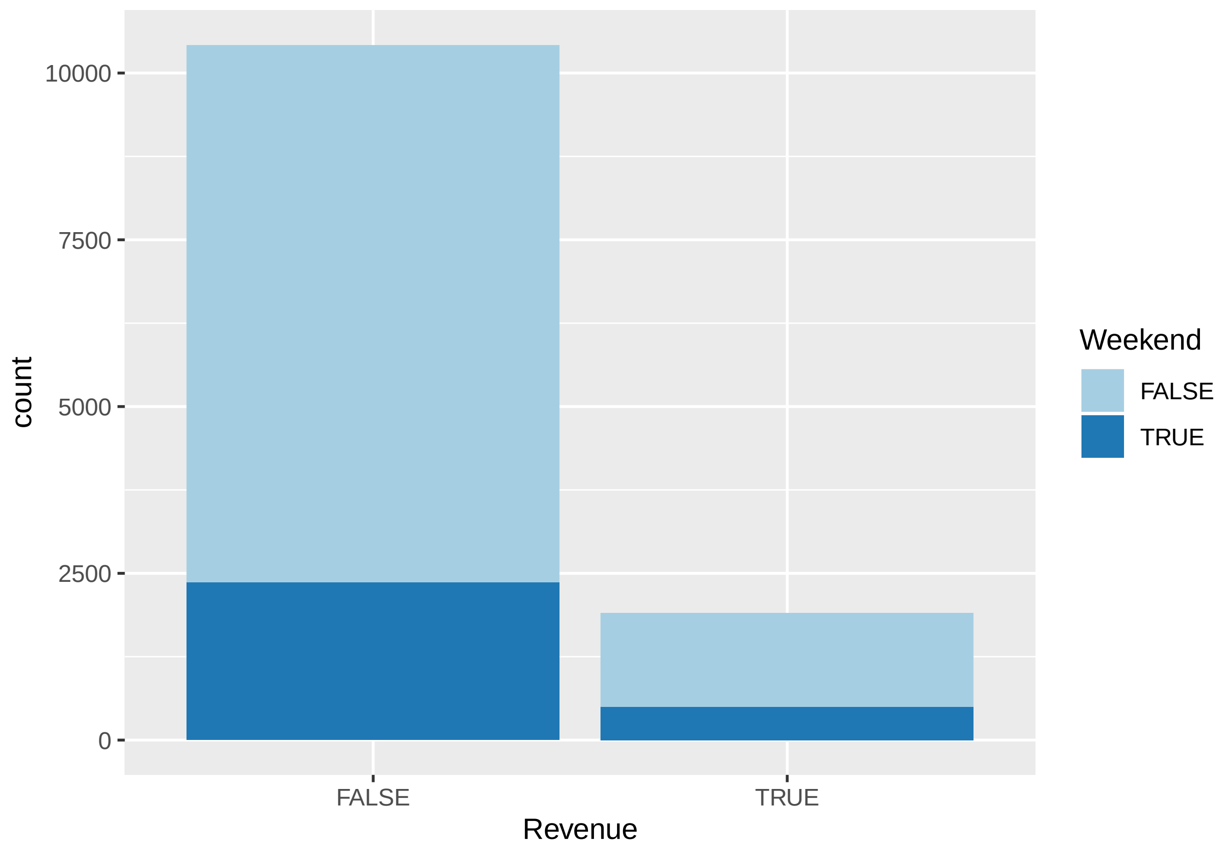
```r
data %>% filter(Revenue == 'FALSE') %>% nrow()/nrow(data)
```

```
## [1] 0.8452555
```

```r
data %>% filter(Revenue == 'TRUE' & Weekend == 'TRUE') %>% nrow()/1908
```

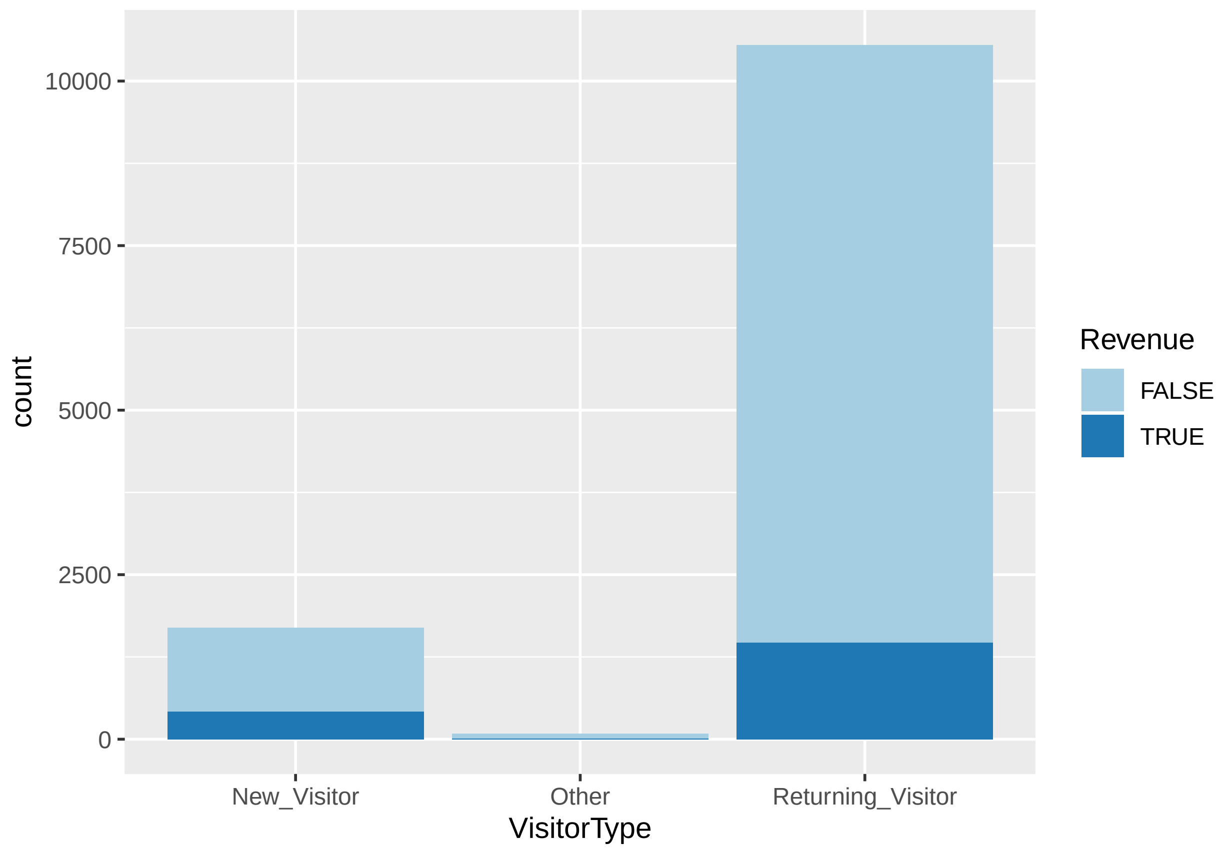
```
## [1] 0.2615304
```

```r
ggplot(data, aes(Revenue, fill = Weekend)) +
  geom_bar() +
  scale_fill_brewer(palette = 'Paired')
```



Returning visitors were much more than new visitors.

```
ggplot(data, aes(VisitorType, fill =  Revenue)) +
    geom_bar() +
    scale_fill_brewer(palette = 'Paired')
```



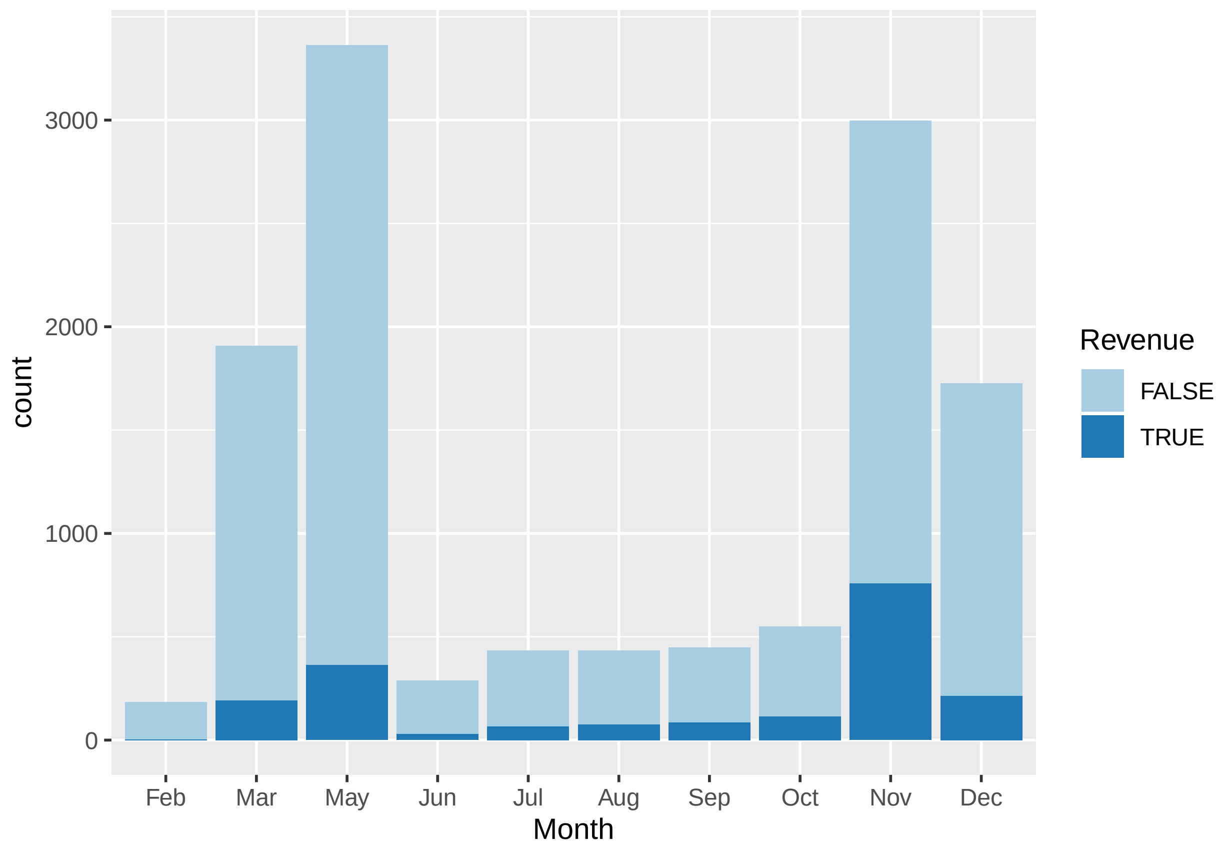
Only ten months of data were included in the data set, no January and April data. March, May, November and December were the four months with significant online shopping performance (both browsing and purchasing). Usually the holiday seasons account for shopping intention, but why March and May, particularly, the performance in May even better than November? It was not explained. In a business context, we need to investigate that:

- where the data was from?
- how it was compiled?
- whether there were unique situations?

```
unique(data$Month)
```

```
##  [1] "Feb"  "Mar"  "May"  "Oct"  "June" "Jul"  "Aug"  "Nov"  "Sep"  "Dec"
```

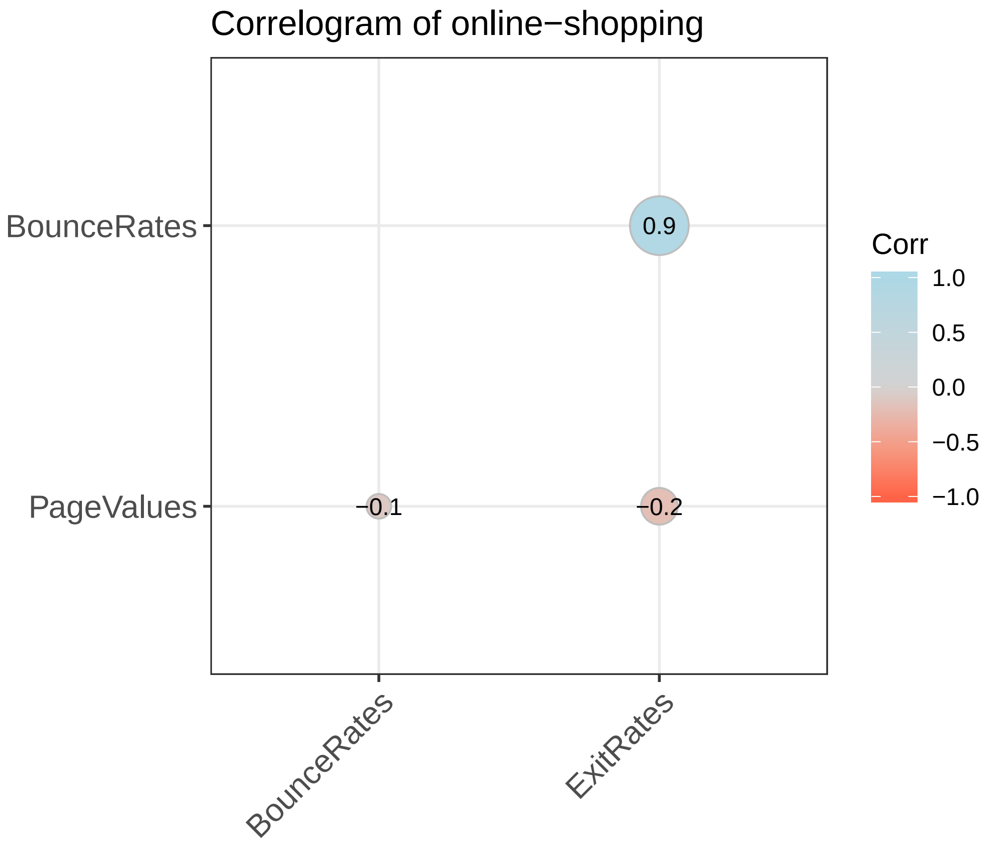
```
data$Month[data$Month == 'June'] <- 'Jun'
data$Month = factor(data$Month, levels = month.abb)

ggplot(data, aes(Month, fill =  Revenue)) +
    geom_bar() +
    scale_fill_brewer(palette = 'Paired')
```



$Administrative$, $AdministrativeDuration$, $Informational$, $InformationalDuration$, $ProductRelated$ and $ProductRelatedDuration$ represent the number of different types of pages visited by the visitor in that session and total time spent in each of these page categories.

$BounceRate$, $ExitRate$ and $PageValue$ represent the metrics measured by "Google Analytics" for each page in the e-commerce site. From the correlogram, we examine the correlation of the three variables and find the high correlation of $BounceRate$ and $ExitRate$.

```
data_GA <- data %>% select('BounceRates', 'ExitRates', 'PageValues')

# Correlation matrix
corr <- round(cor(data_GA), 1)

ggcorrplot(corr, hc.order = TRUE,
           type = "lower",
           lab = TRUE,
           lab_size = 3,
           method="circle",
           colors = c("tomato", "lightgrey", "lightblue"),
           title = "Correlogram of online-shopping",
           ggtheme=theme_bw)
```



#2. Research Questions

1) Which features have close relationship with a shopper's online purchase intention?
2) How accurate is the prediction of the online purchase intention?
3) What is a threshold probability to separate between "buy" and "not buy" response?

#3. Data Preprocessing & Logistic Regression

Logistic regression models are able to treat categorical variables as dummy variables while other models such as suport vector machines and k-means clustering* could exclusively handle numeric variables. Therefore, I need to modify variables by steps and combine data preprocessing with the model training.

First, the response *Revenue* has to be mutated as the *glm*() function only deals with 0/1 as responses. Ramdomly split the data into training, validation and test sets:

```
data <- data %>% mutate(Revenue = if_else(Revenue == 'FALSE', 0, 1))

# 70% for training
set.seed(123)
smp_size <- floor(0.7 * nrow(data))

train_indx <- sample(seq_len(nrow(data)), size = smp_size)
training_set <- data[train_indx,]

# 15% for validation
left <- data[-train_indx,]
validation_indx <- sample(seq_len(nrow(left)), size = 0.5*nrow(left))
```

```r
validation_set <- left[validation_indx,]

# 15% for test
test_set <- left[-validation_indx,]

nrow(training_set)
```

```
## [1] 8631
```

```r
nrow(validation_set)
```

```
## [1] 1849
```

```r
nrow(test_set)
```

```
## [1] 1850
```

Use all features to build the first logistic model and then select six significant ones based on P-value to build a simpler model (in case of overfitting) and compare their performance.

The simpler logistic model has lower AIC on training set and higher accuracy rate on validation set.

```r
logit1 <- glm(Revenue ~., family = binomial, training_set)
summary(logit1)
```

```
##
## Call:
## glm(formula = Revenue ~ ., family = binomial, data = training_set)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -6.1989  -0.4523  -0.3314  -0.1669   3.2384
##
## Coefficients:
##                                Estimate Std. Error z value Pr(>|z|)
## (Intercept)                  -3.170e+00  6.602e-01  -4.802 1.57e-06 ***
## Administrative               -6.071e-03  1.328e-02  -0.457  0.64748
## Administrative_Duration       1.066e-04  2.294e-04   0.465  0.64226
## Informational                 1.863e-02  3.234e-02   0.576  0.56446
## Informational_Duration        5.695e-05  2.638e-04   0.216  0.82911
## ProductRelated                1.214e-03  1.490e-03   0.815  0.41488
## ProductRelated_Duration       7.467e-05  3.548e-05   2.105  0.03533 *
## BounceRates                  -1.255e+00  3.835e+00  -0.327  0.74344
## ExitRates                    -1.697e+01  2.941e+00  -5.769 7.98e-09 ***
## PageValues                    8.476e-02  2.943e-03  28.801  < 2e-16 ***
## SpecialDay                   -1.222e-01  2.752e-01  -0.444  0.65688
## MonthMar                      1.010e+00  6.519e-01   1.549  0.12135
## MonthMay                      9.652e-01  6.448e-01   1.497  0.13442
## MonthJun                      1.364e+00  6.977e-01   1.956  0.05051 .
## MonthJul                      1.562e+00  6.700e-01   2.331  0.01977 *
## MonthAug                      1.377e+00  6.716e-01   2.050  0.04038 *
```

```
## MonthSep                       1.597e+00  6.652e-01   2.401  0.01637 *
## MonthOct                       1.336e+00  6.642e-01   2.012  0.04425 *
## MonthNov                       2.081e+00  6.460e-01   3.221  0.00128 **
## MonthDec                       9.381e-01  6.530e-01   1.437  0.15082
## OperatingSystems              -3.923e-02  4.586e-02  -0.855  0.39229
## Browser                        3.114e-02  2.216e-02   1.405  0.15988
## Region                        -1.150e-02  1.571e-02  -0.732  0.46432
## TrafficType                   -1.403e-02  1.046e-02  -1.341  0.17999
## VisitorTypeOther              -1.655e-03  6.102e-01  -0.003  0.99784
## VisitorTypeReturning_Visitor -2.866e-01  1.047e-01  -2.738  0.00619 **
## WeekendTRUE                    1.784e-01  8.410e-02   2.122  0.03386 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 7431.9  on 8630  degrees of freedom
## Residual deviance: 4952.4  on 8604  degrees of freedom
## AIC: 5006.4
##
## Number of Fisher Scoring iterations: 7
```

```
validation_probs<- predict(logit1, validation_set[, -18], type = "response")
validation_pred = rep(0, nrow(validation_set))
validation_pred[validation_probs > 0.5] = 1

mean(validation_pred == validation_set$Revenue)
```

```
## [1] 0.8826393
```

```
logit2 <- glm(Revenue ~ BounceRates + ExitRates + PageValues +
              Month + VisitorType + Weekend,
              family = binomial, training_set)
summary(logit2)
```

```
##
## Call:
## glm(formula = Revenue ~ BounceRates + ExitRates + PageValues +
##     Month + VisitorType + Weekend, family = binomial, data = training_set)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -6.1638  -0.4578  -0.3418  -0.1588   3.2275
##
## Coefficients:
##                                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)                    -3.234553   0.645870  -5.008 5.50e-07 ***
## BounceRates                     0.240682   3.804764   0.063 0.949561
## ExitRates                     -20.632108   2.814502  -7.331 2.29e-13 ***
## PageValues                      0.084523   0.002921  28.939  < 2e-16 ***
## MonthMar                        1.056199   0.645415   1.636 0.101742
## MonthMay                        1.008840   0.640980   1.574 0.115510
## MonthJun                        1.471596   0.690922   2.130 0.033180 *
```

```
## MonthJul                          1.682811   0.663268   2.537 0.011176 *
## MonthAug                          1.480668   0.665041   2.226 0.025985 *
## MonthSep                          1.684844   0.658653   2.558 0.010527 *
## MonthOct                          1.395436   0.657576   2.122 0.033830 *
## MonthNov                          2.235769   0.638795   3.500 0.000465 ***
## MonthDec                          1.032146   0.646291   1.597 0.110259
## VisitorTypeOther                 -0.077439   0.560605  -0.138 0.890133
## VisitorTypeReturning_Visitor     -0.125683   0.100802  -1.247 0.212459
## WeekendTRUE                       0.171364   0.083455   2.053 0.040036 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 7431.9  on 8630  degrees of freedom
## Residual deviance: 5002.3  on 8615  degrees of freedom
## AIC: 5034.3
##
## Number of Fisher Scoring iterations: 7
```

```r
validation_probs<- predict(logit2, validation_set[, -18], type = "response")
validation_pred = rep(0, nrow(validation_set))
validation_pred[validation_probs > 0.5] = 1

mean(validation_pred == validation_set$Revenue)
```

```
## [1] 0.8842618
```

#4. Data Preprocessing & Other Models

Next, the further data preparation is necessary as *ksvm*() and *kmeans*()* only work for numerical data.

```r
data_k <- data %>% mutate(Weekend = if_else(Weekend == 'FALSE', 0, 1))

data_k$Month <- match(data_k$Month, month.abb)

data_k$VisitorType[data_k$VisitorType == 'Other'] <- 0
data_k$VisitorType[data_k$VisitorType == 'New_Visitor'] <- 1
data_k$VisitorType[data_k$VisitorType == 'Returning_Visitor'] <- 2
data_k$VisitorType <- as.numeric(data_k$VisitorType) # convert character to number

summary(data_k)
```

```
##  Administrative   Administrative_Duration Informational
##  Min.   : 0.000   Min.   :   0.00         Min.   : 0.0000
##  1st Qu.: 0.000   1st Qu.:   0.00         1st Qu.: 0.0000
##  Median : 1.000   Median :   7.50         Median : 0.0000
##  Mean   : 2.315   Mean   :  80.82         Mean   : 0.5036
##  3rd Qu.: 4.000   3rd Qu.:  93.26         3rd Qu.: 0.0000
##  Max.   :27.000   Max.   :3398.75         Max.   :24.0000
##  Informational_Duration ProductRelated   ProductRelated_Duration
##  Min.   :    0.00       Min.   :  0.00   Min.   :     0.0
##  1st Qu.:    0.00       1st Qu.:  7.00   1st Qu.:   184.1
```

```
##  Median :   0.00        Median : 18.00   Median :  598.9  
##  Mean   :  34.47        Mean   : 31.73   Mean   : 1194.8  
##  3rd Qu.:   0.00        3rd Qu.: 38.00   3rd Qu.: 1464.2  
##  Max.   :2549.38        Max.   :705.00   Max.   :63973.5  
##   BounceRates          ExitRates          PageValues         SpecialDay     
##  Min.   :0.000000   Min.   :0.00000   Min.   :  0.000   Min.   :0.00000  
##  1st Qu.:0.000000   1st Qu.:0.01429   1st Qu.:  0.000   1st Qu.:0.00000  
##  Median :0.003112   Median :0.02516   Median :  0.000   Median :0.00000  
##  Mean   :0.022191   Mean   :0.04307   Mean   :  5.889   Mean   :0.06143  
##  3rd Qu.:0.016813   3rd Qu.:0.05000   3rd Qu.:  0.000   3rd Qu.:0.00000  
##  Max.   :0.200000   Max.   :0.20000   Max.   :361.764   Max.   :1.00000  
##      Month        OperatingSystems    Browser           Region     
##  Min.   : 2.000   Min.   :1.000    Min.   : 1.000   Min.   :1.000  
##  1st Qu.: 5.000   1st Qu.:2.000    1st Qu.: 2.000   1st Qu.:1.000  
##  Median : 7.000   Median :2.000    Median : 2.000   Median :3.000  
##  Mean   : 7.652   Mean   :2.124    Mean   : 2.357   Mean   :3.147  
##  3rd Qu.:11.000   3rd Qu.:3.000    3rd Qu.: 2.000   3rd Qu.:4.000  
##  Max.   :12.000   Max.   :8.000    Max.   :13.000   Max.   :9.000  
##   TrafficType     VisitorType       Weekend          Revenue      
##  Min.   : 1.00   Min.   :0.000   Min.   :0.0000   Min.   :0.0000  
##  1st Qu.: 2.00   1st Qu.:2.000   1st Qu.:0.0000   1st Qu.:0.0000  
##  Median : 2.00   Median :2.000   Median :0.0000   Median :0.0000  
##  Mean   : 4.07   Mean   :1.849   Mean   :0.2326   Mean   :0.1547  
##  3rd Qu.: 4.00   3rd Qu.:2.000   3rd Qu.:0.0000   3rd Qu.:0.0000  
##  Max.   :20.00   Max.   :2.000   Max.   :1.0000   Max.   :1.0000
```

It seems redundant to split data again. However, the above categorical data has been mutated, so it is a necessary step. Meanwhile, the data set for the above logistic model should be kept as we may need it for testing later. The set seed ensures that the training, validation and test sets are corresponding to the split data for logistic regression models. Thus we are able to compare the models' accuracy and do the selection using the validation set.

```r
# 70% for training
set.seed(123)
smp_size <- floor(0.7 * nrow(data_k))

train_indx_k <- sample(seq_len(nrow(data_k)), size = smp_size)
training_set_k <- data_k[train_indx_k,]

# 15% for validation
left_k <- data_k[-train_indx_k,]
validation_indx_k <- sample(seq_len(nrow(left_k)), size = 0.5*nrow(left_k))
validation_set_k <- left_k[validation_indx_k,]

# 15% for test
test_set_k <- left_k[-validation_indx_k,]
```

#4.1 Support Vector Machines (SVM)

I have built a SVM model with a simple linear kernel, which has 84.69% of accuracy on the validation set.

```r
library(kernlab)
```

```
## 
## Attaching package: 'kernlab'
```

```
## The following object is masked from 'package:ggplot2':
##
##     alpha
```

```r
svm <- ksvm(as.matrix(training_set_k[, 1:17]),as.factor(training_set_k[, 18]),
            type = "C-svc", # Use C-classification method
            kernel = "vanilladot",
            C = 100,
            scaled = TRUE)
```

```
##  Setting default kernel parameters
```

```r
validation <- predict(svm, validation_set_k[, 1:17])
svm_acc = sum(validation == validation_set_k[, 18]) / nrow(validation_set_k)
svm_acc
```

```
## [1] 0.8469443
```

#4.2 K-nearest Neighbor Model (KNN)

As KNN is not model based and there is no training or validation step, I would like to use all the data in KNN to test its performane. However, the model ran really slow when there were a lot of observations in this case. I was not able to generate the output here due to time constraints, so the coding was attached here.

In reality, both the algorithm efficiency and its accuracy matter.

```r
library(kknn)

#check_accuracy = function(X){
  #predicted <- rep(0, (nrow(data_k))) # predictions: start with a vector of all zeros
  #for (i in 1:nrow(data_k)){
    # remove row i of the data when finding nearest neighbors
    #knn <- kknn(Revenue~., data_k[-i,], data_k[i,], k = X, scale = TRUE)
    #predicted[i] <- as.integer(fitted(knn) + 0.5)
  #}
  #accuracy = sum(predicted == data_k[, 18]) / nrow(data_k)
  #return(accuracy)
#}

#acc <- rep(0, 30) # set up a vector of 20 zeros to start
#for (X in 1:30){
  #acc[X] = check_accuracy(X)
#}
#acc
```

#4.3 Clustering*

As an unsupervised learning method, clustering is not for classification and we are not able to predict a shopper's purchase intention via clustering.

Instead, given the data including web browsing metrics, k-means clustering could be used to subset online shoppers into similar groups. If more customer profile data was given, we would be able to find out the common shopping behavior or characteristics of customers in a cluster, which may inform sales and marketing decisions.

```r
K <- 1:20
tot_withinss <- numeric()

for (k in K) {
  km <- kmeans(data_k, centers = k, nstart = 20)
  tot_w <- km$tot.withinss # generate total within-cluster sum of squares for the elbow plot later
  tot_withinss <- c(tot_withinss, tot_w)
}
```

```
## Warning: Quick-TRANSfer stage steps exceeded maximum (= 616500)

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: Quick-TRANSfer stage steps exceeded maximum (= 616500)

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: Quick-TRANSfer stage steps exceeded maximum (= 616500)

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations
```

```
## Warning: Quick-TRANSfer stage steps exceeded maximum (= 616500)

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: Quick-TRANSfer stage steps exceeded maximum (= 616500)

## Warning: Quick-TRANSfer stage steps exceeded maximum (= 616500)

## Warning: did not converge in 10 iterations

## Warning: Quick-TRANSfer stage steps exceeded maximum (= 616500)

## Warning: did not converge in 10 iterations

## Warning: Quick-TRANSfer stage steps exceeded maximum (= 616500)

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: Quick-TRANSfer stage steps exceeded maximum (= 616500)

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations
```

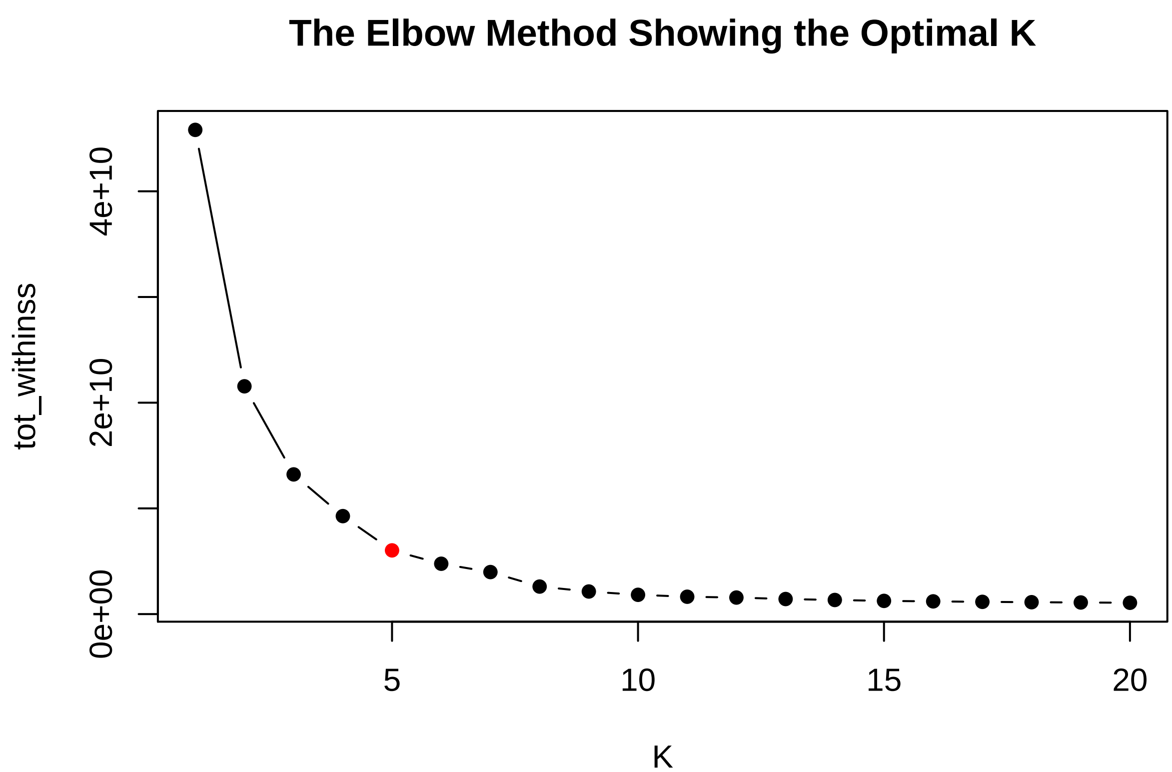
```
## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: Quick-TRANSfer stage steps exceeded maximum (= 616500)

## Warning: did not converge in 10 iterations

## Warning: Quick-TRANSfer stage steps exceeded maximum (= 616500)

## Warning: did not converge in 10 iterations

## Warning: Quick-TRANSfer stage steps exceeded maximum (= 616500)

## Warning: Quick-TRANSfer stage steps exceeded maximum (= 616500)

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: Quick-TRANSfer stage steps exceeded maximum (= 616500)

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: Quick-TRANSfer stage steps exceeded maximum (= 616500)

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations
```

```
## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: Quick-TRANSfer stage steps exceeded maximum (= 616500)

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations
```

```r
data.frame(K, tot_withinss)
```

```
##     K tot_withinss
## 1   1  45809196214
## 2   2  21549402377
## 3   3  13215040825
## 4   4   9269090288
## 5   5   6028025933
## 6   6   4762230737
## 7   7   3975861671
## 8   8   2604257774
## 9   9   2136397274
## 10 10   1823542212
## 11 11   1641955884
## 12 12   1565256142
## 13 13   1425573840
## 14 14   1334998994
## 15 15   1250303648
## 16 16   1204453629
## 17 17   1157218741
## 18 18   1125261377
## 19 19   1098645527
## 20 20   1070787420
```

```r
plot(data.frame(K, tot_withinss), type = "b", pch = 16,
     col = ifelse((tot_withinss < 6028025935 & tot_withinss > 6000000000), "red", "black"),
     main = "The Elbow Method Showing the Optimal K")
```



According to the prediction accuracy on the validation set and the algorithm efficiency, the logistic regression model with six features stands out (88.43% accuracy on the validation set), in which $ExitRates$, $PageValues$ and $Month$ are significant to the shopping intention. In other words, the percentage that were the last in the session for all pageviews to the page, the average value for a web page that a user visited before completing an e-commerce transaction, and the shopping timing highly correlated with the shopping intention.

To answer the second question, we may use the test set to estimate the model's general quality: the model has the accuracy of 87.68% for the prediction of an online purchase intention.

```r
test_probs<- predict(logit2, test_set[, -18], type = "response")
test_pred = rep(0, nrow(test_set))
test_pred[test_probs > 0.5] = 1

mean(test_pred == test_set$Revenue)
```

```
## [1] 0.8767568
```

Finally, regarding a threshold probability to separate between "buy" and "not buy" response, 0.5 as probability was simply used in the above logistic model: if any precition higher than 0.5, the response would be rounded up to one.

In a business context, it would be more complicated. Do we prefer to detect a less strong purchase intention and thus take actions such as emailing offers to motivate customers to buy? Or would we like to better target leads to sales due to marketing cost constraints?

In the former scenario, a lower threshold probability may be adopted while a higher one would be considered in the latter.