Capstone: User Shopping Intentions

Pratyush Singh

5/13/2021

Contents

Introduction

The tremendous growth of the internet population has increased the trend of online shopping, commonly known Business to Customer (B2C), exponentially. More than 310 million active customers have bought nearly 136 billion USD goods in 2016 from Amazon which is the leading e-commerce company in the world. The number of active users of Alibaba.com, one of the largest online shopping platform, was 82.67 thousand in 2017 that was increased by 43.3% from the previous year .

With e-commerce becoming more and more prevalent in today's economy it is important for businesses within this sector to understand what factors into a site visitor making a purchase, and being able to put their attention on potential customers. Understanding the behavior and intention of online customers has become immensely important for marketing, improving customer experience which, in return, increases sales.

For this Analysis We will use Online Shoppers Purchasing Intention data set provided on the UC Irvine's Machine Learning Repository. There are three categories of variables in the data , data related to page visited by user, google analytics metrics and user visit data . Details about data are available on the following link : https://archive.ics.uci.edu/ml/datasets/Online+Shoppers+Purchasing+Intention+Dataset#

We will divide the data into two parts initially users and validation set (final hold-out set). We will then divide user set into training and testing set for model selection and evaluation.

For Evaluation we will use overall accuracy for all models i.e, the proportion of correctly predicted estimates in the test set.

Analysis

We will load the following libraries

```
library(tidyverse)
library(caret)
library(data.table)
library(corrplot)
library(readr)
library(fastDummies)
library(matrixStats)
library(gam)
library(MASS)
library(randomForest)
library(cowplot)
```

We will load the RData (Environment) of the code.r to avoid re-running the models . Most of the codes are also either left out of the report or not evaluated to keep the report neat . To re-knit the report it is highly recommended to run the code.R file and load its Environment here.

```
load("shopper-intention.RData")
```

Data Mining and Preparation

```
##############################################################################
# Creating user and validation set (final hold-out test set)
##############################################################################

temp <- tempfile()
download.file("https://archive.ics.uci.edu/ml/
              machine-learning-databases/00468/online_shoppers_intention.csv",
              "online_shoppers_intention.csv")

# ignore warning after unzip
d <- read.csv("online_shoppers_intention.csv")
```

```
class(d)
```

```
## [1] "data.frame"
```

We will convert the categorical and Boolean variables into dummy Dummy variables and One-Hot Encoded Variables.

```
#Created Dummy variables for categorical data and cleaning data
dat <- d %>% mutate(VisitorType = as.factor(VisitorType) ,
                            Month = as.factor(Month),
                            Weekend = as.integer(Weekend),
                            Revenue = as.integer(Revenue)) %>%
  dummy_columns(select_columns = c("VisitorType","Month") ) %>%
  dplyr::select(-Month , - VisitorType )
```

Now We Divide the data into user and validation set . Where the user set contains 90% of the data and the validation set has the remaining 10%.

```
# Validation set will be 10% of users data
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)`
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
test_index <- createDataPartition(y = dat$Revenue, times = 1, p = 0.1, list = FALSE)
users <- dat[-test_index,]
validation <- dat[test_index,]
```

Exploration

```
glimpse(users)
```

```
## Rows: 11,097
## Columns: 29
## $ Administrative                <int> 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0~
## $ Administrative_Duration       <dbl> 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ~
```

```
## $ Informational                 <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ Informational_Duration        <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ ProductRelated                <int> 1, 2, 1, 2, 10, 19, 1, 0, 2, 3, 3, 16, 7~
## $ ProductRelated_Duration       <dbl> 0.000000, 64.000000, 0.000000, 2.666667,~
## $ BounceRates                   <dbl> 0.200000000, 0.000000000, 0.200000000, 0~
## $ ExitRates                     <dbl> 0.200000000, 0.100000000, 0.200000000, 0~
## $ PageValues                    <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ SpecialDay                    <dbl> 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.4, 0.0, ~
## $ OperatingSystems              <int> 1, 2, 4, 3, 3, 2, 2, 1, 2, 2, 1, 1, 1, 2~
## $ Browser                       <int> 1, 2, 1, 2, 3, 2, 4, 2, 2, 4, 1, 1, 1, 5~
## $ Region                        <int> 1, 1, 9, 2, 1, 1, 3, 1, 2, 1, 3, 4, 1, 1~
## $ TrafficType                   <int> 1, 2, 3, 4, 4, 3, 3, 5, 3, 2, 3, 3, 3, 3~
## $ Weekend                       <int> 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0~
## $ Revenue                       <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ VisitorType_New_Visitor       <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ VisitorType_Other             <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ VisitorType_Returning_Visitor <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
## $ Month_Aug                     <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ Month_Dec                     <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ Month_Feb                     <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
## $ Month_Jul                     <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ Month_June                    <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ Month_Mar                     <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ Month_May                     <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ Month_Nov                     <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ Month_Oct                     <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ Month_Sep                     <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
```

Our Target Class is Revenue Which states weather the user purchased something(1) or not (0).

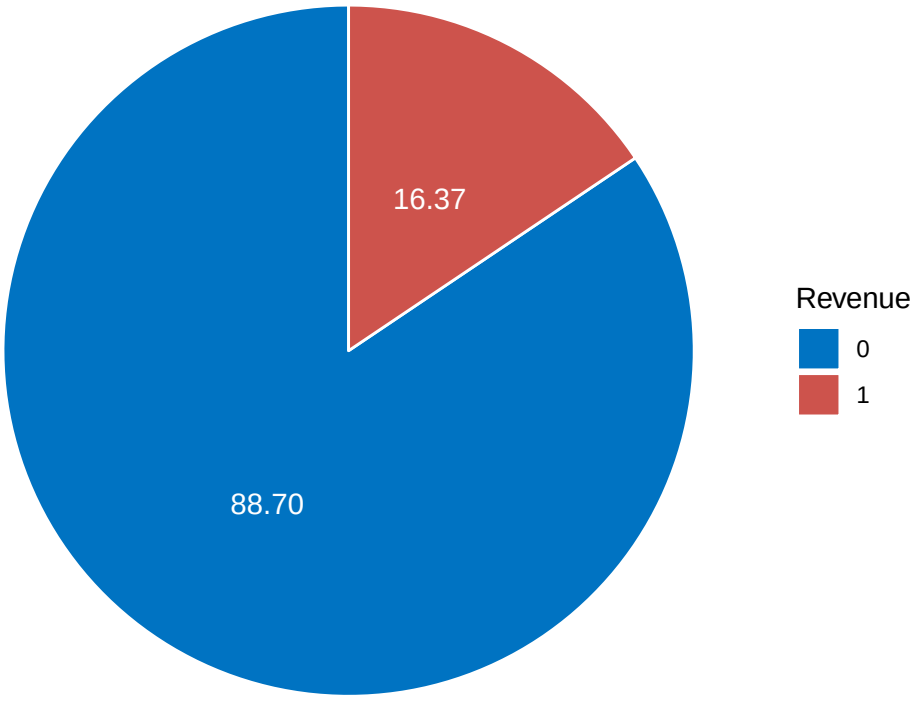

Figure 1: Proportion of Revenue

It is evident that majority of users (88.7%) do not end up making a purchase and only 16.37% users make purchase on the website for a session.

This trend follows through other features as well where majority of users do not make purchase and hence the Revenue is 0.

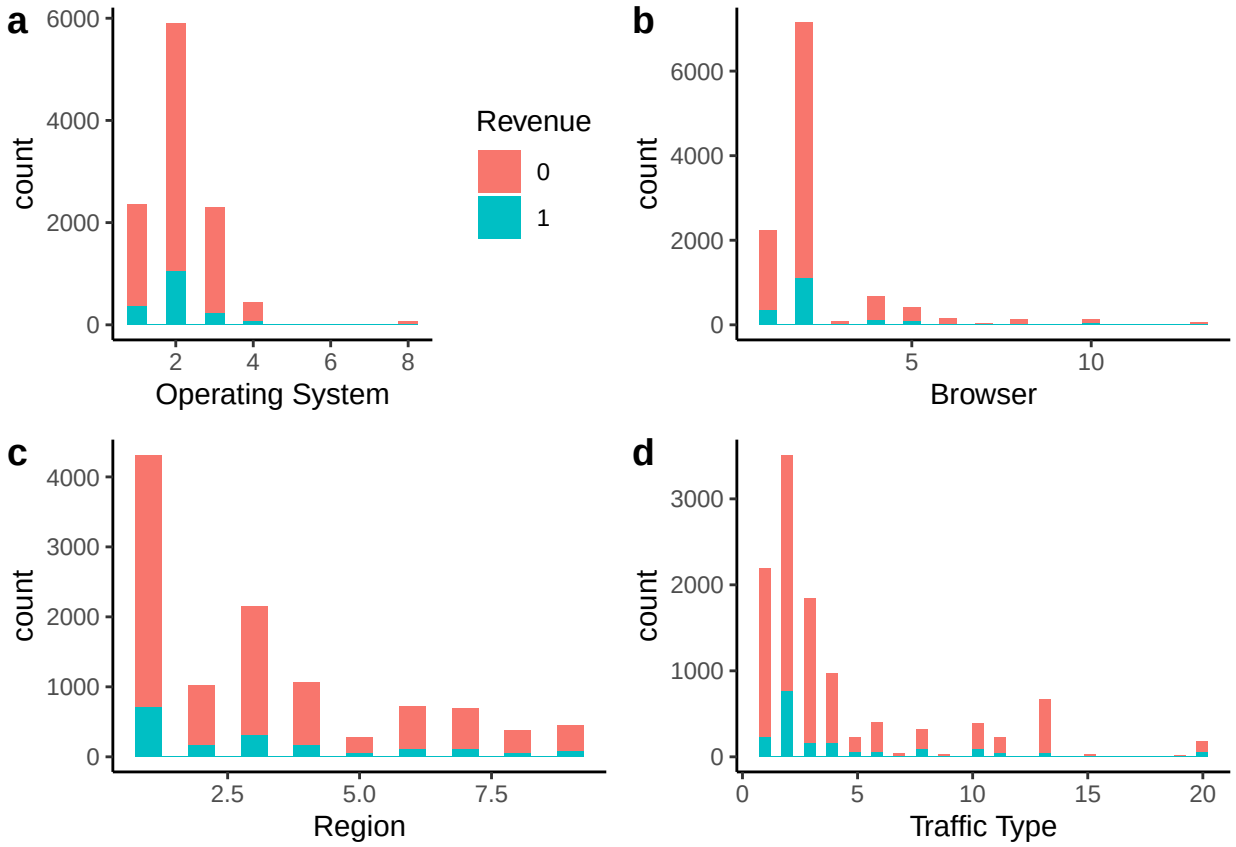

Figure 2: Distribution of Features by Revenue

We can also compare distributions of other features exclusively in the case where purchase is made and when purchase is not made to compare the difference.

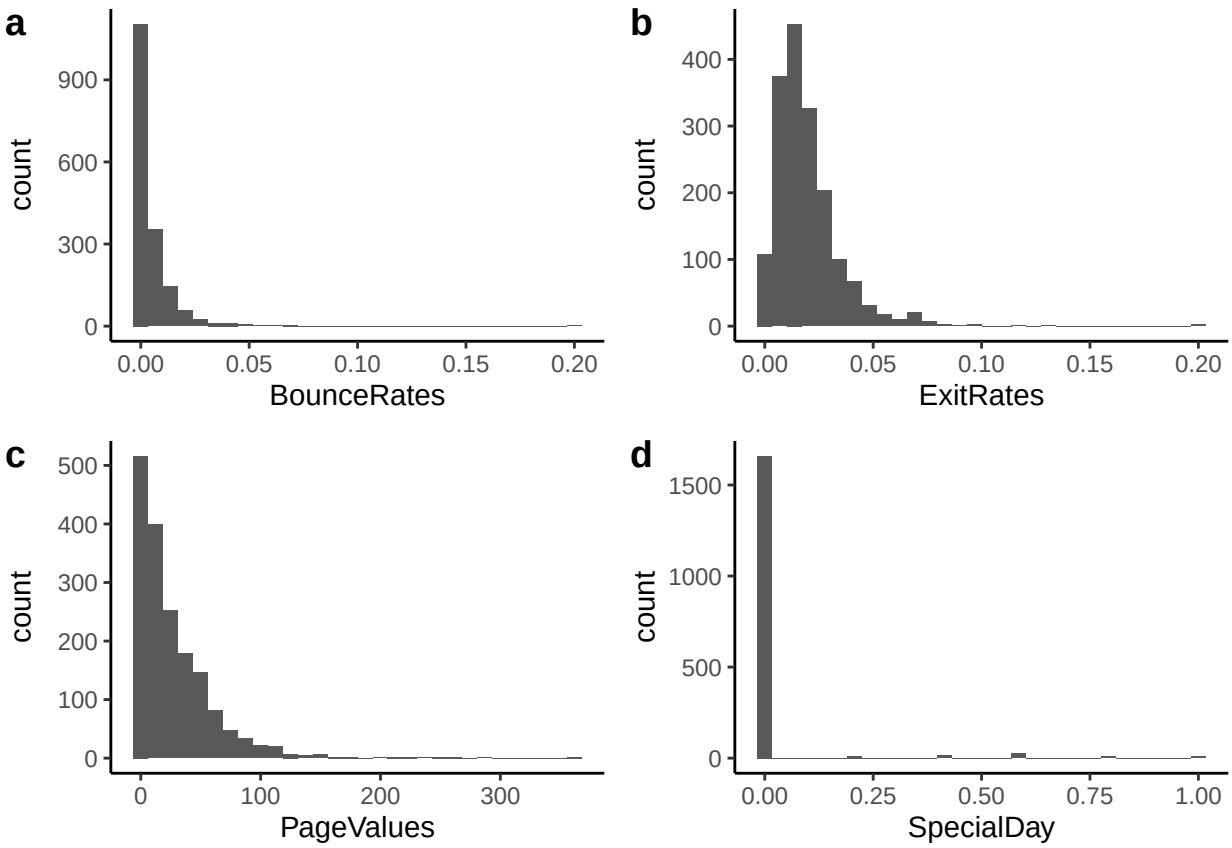

Figure 3: Distribution of features when Revenue is True

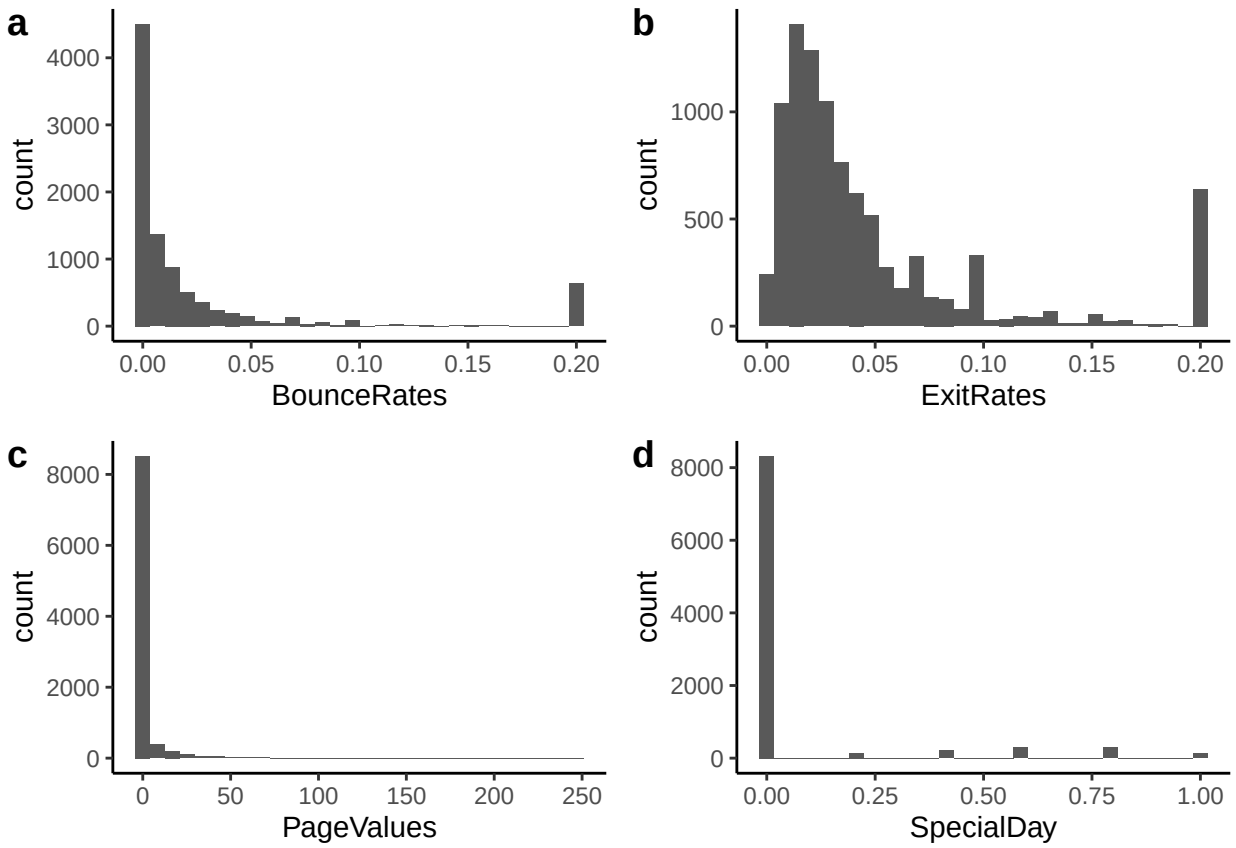

Figure 4: Distribution of features where Revenue is False

We see that when purchase is made more Pages are viewed (PageValue has higher distribution spread) and Exit Rates are lower . These are expected differences in the features.

There are Page Related features(Administrative , Informational , Product Related) and there respective Duration in the data. We will Explore their relation over Revenue class.

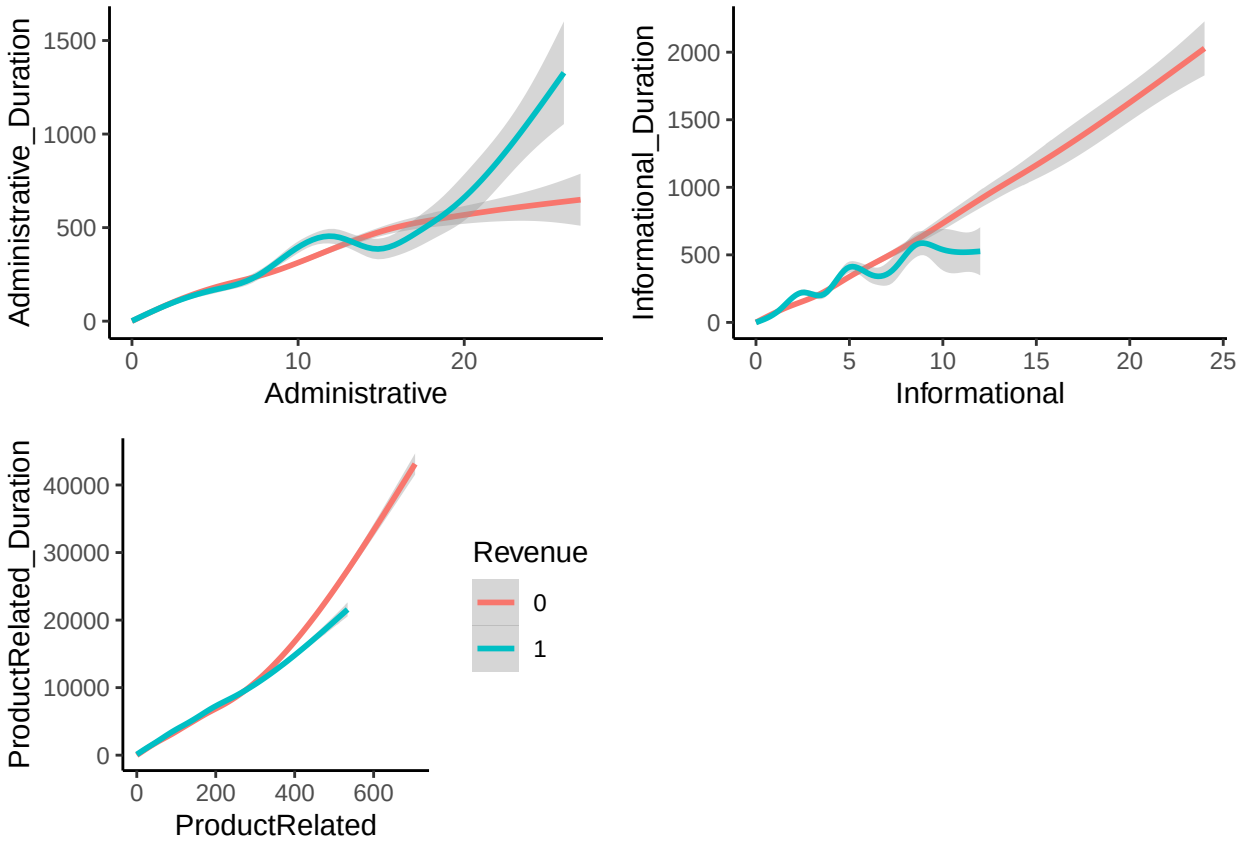

Figure 5: Effect of Duration on feature by Revenue

There seems to be strong correlation between these features and there duration over both classes which we will further explore and use to clean the data for better results.

Correlation

Let us now explore the correlation between all our variables .

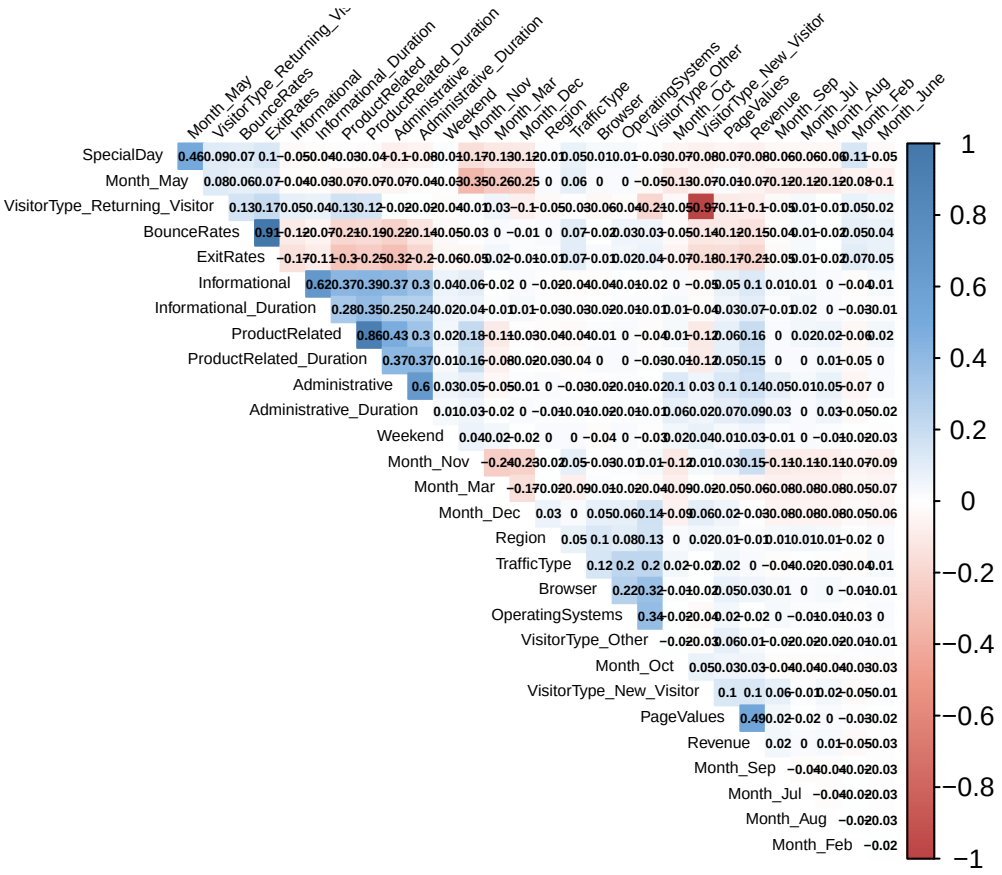

Figure 6: Correlation between Variables

We can see that all the page related features are highly correlated to each other. For a stable model we would should not have highly correlated variables so we will combine these variables together to eliminate problem.

```
AvgMinutes <- function(Count, Duration){
  if (Duration == 0) { # function to find Average Minute for page related activity
    output = 0
  }

  else if ( Duration != 0) {
    output = as.numeric(Duration)/as.numeric(Count)
  }

  return(output)
}


users <- users %>% # we apply the function for each page related feature
  rowwise() %>%
  mutate(Avg_Admin = AvgMinutes(Administrative , Administrative_Duration),
         Avg_Info = AvgMinutes(Informational,Informational_Duration),
         Avg_ProductRel = AvgMinutes(ProductRelated , ProductRelated_Duration))%>%
  ungroup() %>%
```

```
  dplyr::select(-Administrative , -Administrative_Duration , -Informational ,
         -Informational_Duration , -ProductRelated , -ProductRelated_Duration)
```

Examining the Correlation among variables again

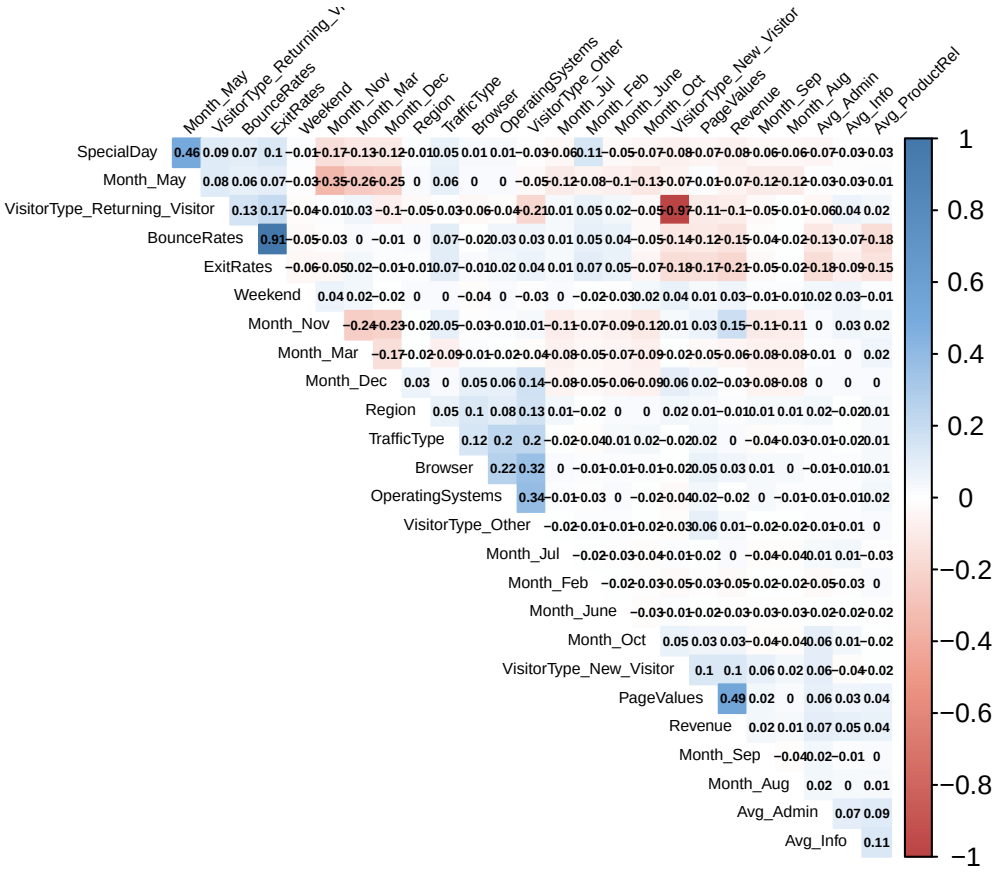

Figure 7: Correlation between variables after cleaning

Now the collinearity problem seems to be reduced significantly and we can move forward to our model selection and evaluation.

We will do the same manipulation to the validation set

```
#Doing the same for Validation set
validation <- validation %>%
  rowwise() %>%
  mutate(Avg_Admin = AvgMinutes(Administrative , Administrative_Duration),
         Avg_Info = AvgMinutes(Informational,Informational_Duration),
         Avg_ProductRel = AvgMinutes(ProductRelated , ProductRelated_Duration))%>%
  ungroup() %>%
  dplyr::select(-Administrative , -Administrative_Duration , -Informational ,
         -Informational_Duration , -ProductRelated , -ProductRelated_Duration)
```

Modeling

We will use multiple models to select the best one for our final hold-out test. We have excluded Quadratic Discriminant Analysis (qda) model because of large dimensionality of our data which is not suitable for that model.

K-Means

K-means clustering is a very famous and powerful unsupervised machine learning algorithm. It is used to solve many complex unsupervised machine learning problems.

k-means clustering tries to group similar kinds of items in form of clusters. It finds the similarity between the items and groups them into the clusters. K-means clustering algorithm works in three steps. Lets see what are these three steps.

-Select the k values. -Initialize the centroids. -Select the group and find the average.

Logistic Regression

Logistic regression (LR) is a statistical method similar to linear regression since LR finds an equation that predicts an outcome for a binary variable, Y, from one or more response variables, X. However, unlike linear regression the response variables can be categorical or continuous, as the model does not strictly require continuous data.

To predict group membership, LR uses the log odds ratio rather than probabilities and an iterative maximum likelihood method rather than a least squares to fit the final model.

Linear Discriminant Analysis

LDA is a generative model similar to naive bayes model . It is a dimensionality reduction technique which assumes similar core structure for all parameters so just one pair of parameters are required to find by the algorithm.Forcing this assumption makes the boundary a line, similar to Logistic Regression

Locally Weighted Regression (Loess)

Loess model uses linear assumption of function (taylor's theorem) and locally fits the data point of given span (subset of closest data) into linear relation. When fitting it also takes a weighted approach (more weights to points near the target point). We obtain then a fitted model that retains only the point of the model that are close to the target point. The target point then moves away on the x axis and the procedure repeats for each points.

K- Nearest Neighbour

The k-nearest neighbors (KNN) algorithm is a simple, easy-to-implement supervised machine learning algorithm that can be used to solve both classification and regression problems.

The KNN algorithm assumes that similar things exist in close proximity.

The KNN algorithm first initializes a value of K(chosen number of neighbors).Then it calculates the distance between a target and every other data point . Then orders the data on the distance and selects the k nearest points from the target in the same class as the target .

Random Forest

The main concept of random forest is to create a large number of correlated decision tree where all the decision trees act as an ensemble model,. Each of the decision trees put their prediction of a class and the final decision is based on maximum vote. The result of the Random forest algorithm is more reliable than the decision tree algorithm because each of the trees neutralizes the error of other trees. All the trees act as a committee to make the decision.

Evaluation

For Evaluation in this report we will use overall accuracy of the model . Which is the proportion of correct predictions made from the test set.

$$A_m = \frac{1}{N}\sum_{i=1}(\widehat{y_i} - y_i)$$

Where

$$y_i$$

is the actual class of index i and

$$\widehat{y_i}$$

is the estimate at index i.

Results

We will Divide user dataset into train and test set for model selection

```
# Validation set will be 10% of users data
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)`
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
test_index <- createDataPartition(y = users$Revenue, times = 1, p = 0.1, list = FALSE)
train_set <- users[-test_index,]
test_set <- users[test_index,]

#convert class to factors for prediction and model fitting
train_set$Revenue <- factor(train_set$Revenue)
test_set$Revenue <- factor(test_set$Revenue)
```

K-Means

Firstly, We will define a function that can predict classes from the centers formed by K-Means training (as it is an unsupervised model)

```
#predict function for kmeans
predict_kmeans <- function(x, k) {
  centers <- k$centers    # extract cluster centers
  # calculate distance to cluster centers
  distances <- sapply(1:nrow(x), function(i){
```

```
    apply(centers, 1, function(y) dist(rbind(x[i,], y)))
  })
  max.col(-t(distances))  # select cluster with min distance to center
}
```

Now we can train the object and use the defined function to form results

```
# k means train object k
k <- kmeans(train_set[,-10] , 2)

y_hat_kmeans <- ifelse(predict_kmeans(test_set[,-10] , k) == 1 , 1 , 0) %>%
  factor() #ifelse function to create estimate class from kmeans train object
acc_kmeans <- confusionMatrix(y_hat_kmeans , test_set$Revenue)$overall['Accuracy']
```

```
## Accuracy
## 0.836036
```

Logistic Regression

We will use Caret package with glm (generalized linear model) to train object

```
#logical Regression model
lr <- train(Revenue~. , method= 'glm' , #generalized linear model for log regression
            data = train_set)

y_hat_lr <- predict(lr , test_set[,-10])
acc_lr <- confusionMatrix(y_hat_lr , test_set$Revenue)$overall['Accuracy']
```

```
##  Accuracy
## 0.8774775
```

Linear Discriminant Analysis (LDA)

```
lda <- train(Revenue~. ,
             method="lda" , #lda method
             data = train_set)
y_hat_lda <- predict(lda , test_set[,-10])
acc_lda <- confusionMatrix(y_hat_lda , test_set$Revenue)$overall['Accuracy']
```

```
##  Accuracy
## 0.8774775
```

The Accuracy of logistic Regression and LDA model are identical due to the similar approach of these two models which was expected.

Locally Weighted Regression (Loess)

```
loess <- train(Revenue~. ,
               method = "gamLoess" , #Loess method
               data = train_set)

y_hat_loess <- predict(loess , test_set[,-10])

acc_loess <- confusionMatrix(y_hat_loess , test_set$Revenue)$overall['Accuracy']
```

```
##  Accuracy
## 0.8810811
```

K-Nearest Neighbour (KNN)

For KNN model we can tune the parameter K using tuneControl in caret package to select the best value of K for highest accuracy

```
tuning <- data.frame(k = seq(3, 29, 2)) #tuning parameter for value of k in knn
```

Using odd values of K from 3 to 29 we will train KNN model. This algorithm takes some time to run.

```
train_knn <- train(Revenue~.,
                   method = "knn",
                   tuneGrid = tuning ,
                   data = train_set)
```

We can see the best values of K that maximizes the accuracy score of the model.

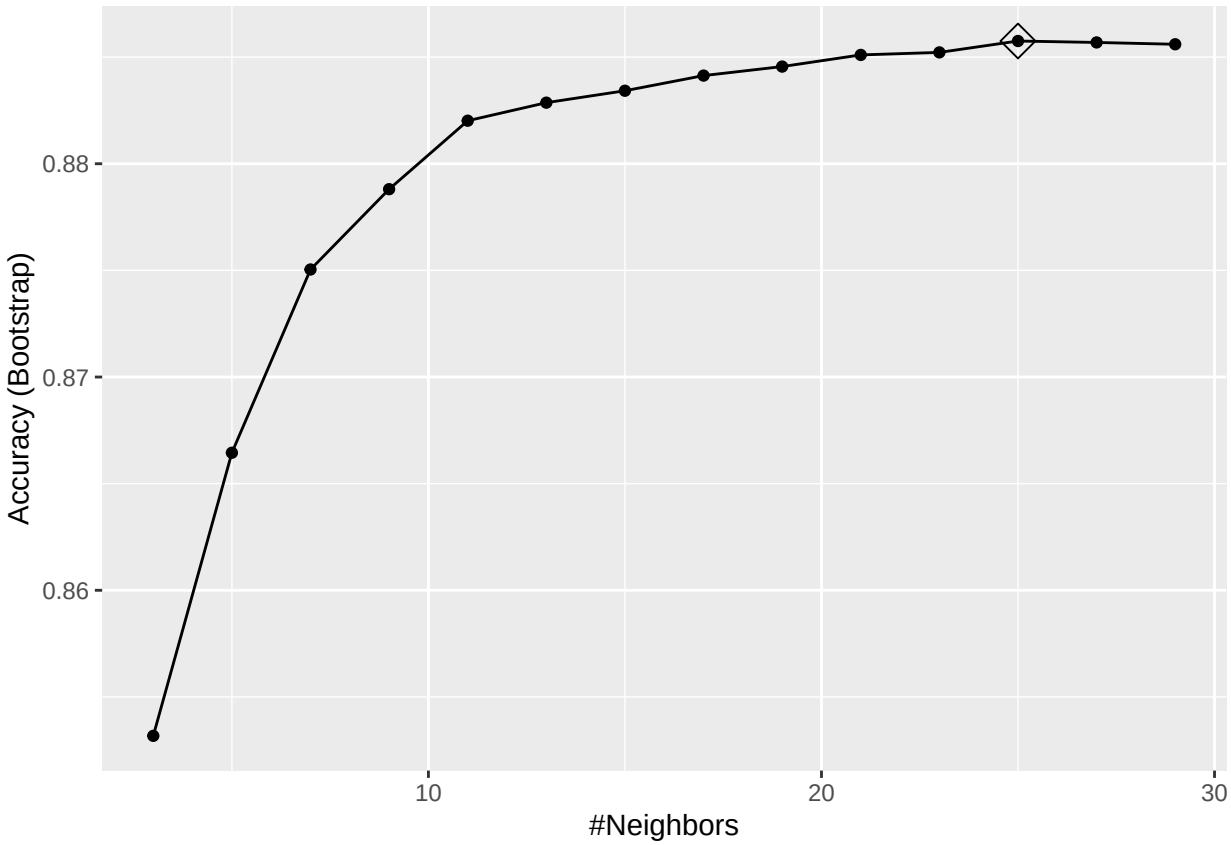

Figure 8: Optimal value of K in KNN

```
train_knn$bestTune
```

```
##     k
## 12 25
```

The optimal value of K is 25 which will be used while making estimate of the class

```
y_hat_knn <- predict(train_knn, test_set[,-10])
acc_knn <- mean(y_hat_knn == test_set$Revenue)
```

```
## [1] 0.8756757
```

Random Forest

In Random Forest we will use manual control for training rf object . Instead of using Bootstrap , we will use repeated cross validation method using 10 fold cross validation repeated 3 times and using grid search.

```
# control for random forest train
control <- trainControl(method='repeatedcv', #repeated cross validation method for random forest
                        number=10,  # ten folds cross-validations
                        repeats=3, #repeated 3 times
                        search='grid')
```

```
# grid search method , Each axis of grid is an
# algorithm parameter and point in grid are specific combinations of parameter
```

We will also use tune mtry with odd values between 2 and 10 and also set manual ntree number to 501 . This algorithm takes high memory and time so controls , tuning and ntree should be set accordingly.

```
rf <- train(Revenue~. ,
            method = 'rf' ,
            tuneGrid = data.frame(mtry = c(3,5,7,9)), #tuning parameter for mtry
            trControl = control ,
            ntree = 501 ,  #no of trees in algorithm
            data = train_set)
```

Examining the tuning parameters for best fit.

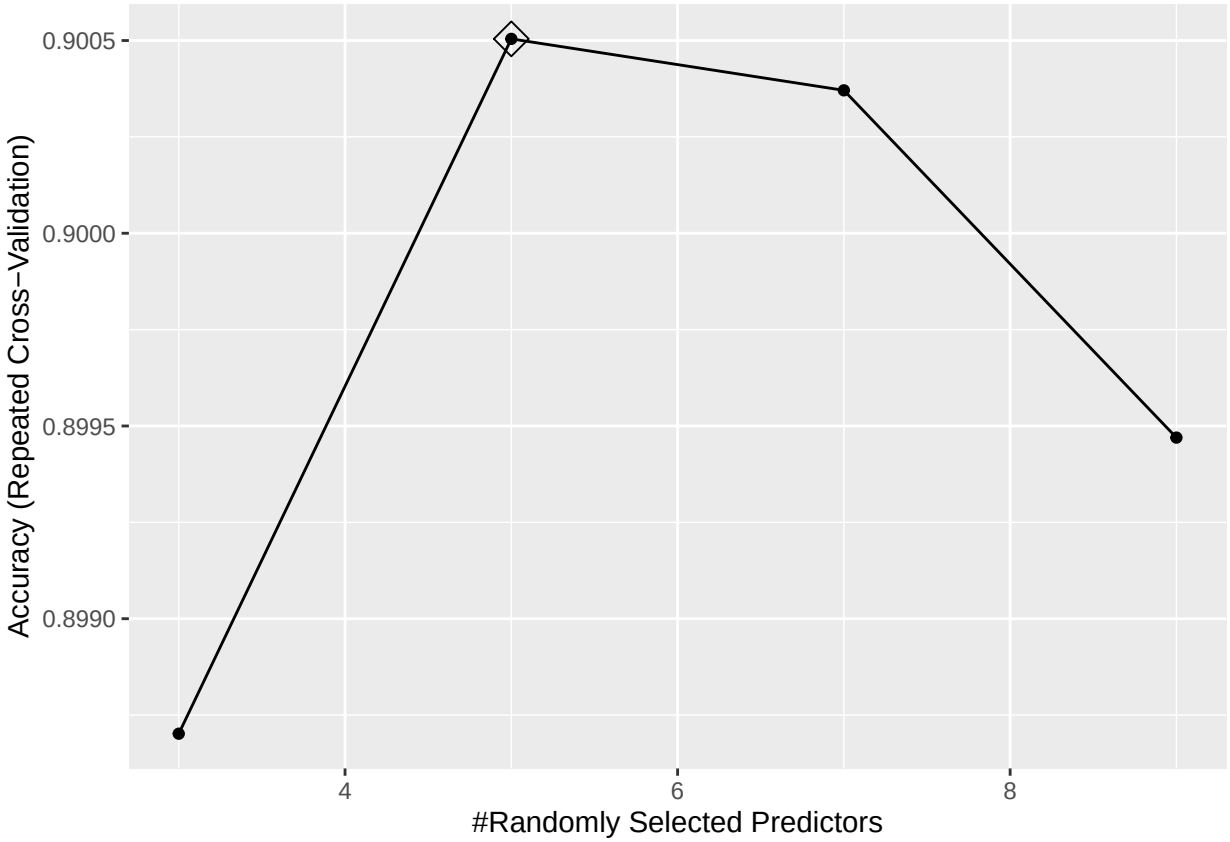

Figure 9: Optimal value of mtry in Random Forest

```
rf$bestTune
```

```
##   mtry
## 2    5
```

This value of mtry = 5 will be used in the final test as it is found to be optimum .

```
y_hat_rf <- predict(rf , test_set[,-10])
acc_rf <- mean(y_hat_rf == test_set$Revenue)
```

```
## [1] 0.8918919
```

Let us review the accuracies of all our models .

Methods	Accuracy
K-Means	0.8360360
Logical Regression	0.8774775
LDA	0.8774775
Loess	0.8810811
KNN	0.8756757
Random Forest	0.8918919

ENSEMBLE

We will also try a simple ensemble model by combining the estimates of all of the models we have tried . We will simply predict on the basis of average of all model predictions where if all models give majority to one class it is predicted otherwise the second class is predicted.

Firstly combining the estimates from all the models to a single matrix column wise.

```
#combine all model results (class estimates) into single matrix
models <- cbind(y_hat_kmeans , y_hat_knn , y_hat_lda , y_hat_loess , y_hat_lr , y_hat_rf)
```

We can use simple ifelse statement to predict class on majority of estimates by all models

```
# if majority of models predict 1(Revenue true) predict True otherwise False
y_hat_ens <- ifelse(rowMeans(models == 1) > .5 , 0 , 1 ) %>% factor()
#Accuracy of ensemble model
acc_ens <- mean(y_hat_ens == test_set$Revenue)
```

```
## [1] 0.8801802
```

Methods	Accuracy
K-Means	0.8360360
Logical Regression	0.8774775
LDA	0.8774775
Loess	0.8810811
KNN	0.8756757
Random Forest	0.8918919
Ensemble Model	0.8801802

We have observed that among all the models we have evaluated Random Forest performed the best with an over all accuracy of over 89%. Hence , we will use Random Forest model for our final hold-out test on the validation set .

Final Evakuation

Random Forest with all Features

Convert our target class into factor for model fitting and prediction

```
users$Revenue <- factor(users$Revenue)
validation$Revenue <- factor(validation$Revenue)
```

```
# Random Forest

rf_final <- train(Revenue~. ,
            method = 'rf' ,
            tuneGrid = rf$bestTune,  #using best tune of model selection
            trControl = control ,
            ntree = 501 ,
            data = users)

y_hat_final <- predict(rf_final , validation[,-10])
acc_final <- mean(y_hat_final == validation$Revenue)
```

```
## [1] 0.9067315
```

We can also look into variable importance in the random forest model to see how each feature contributes in the prediction

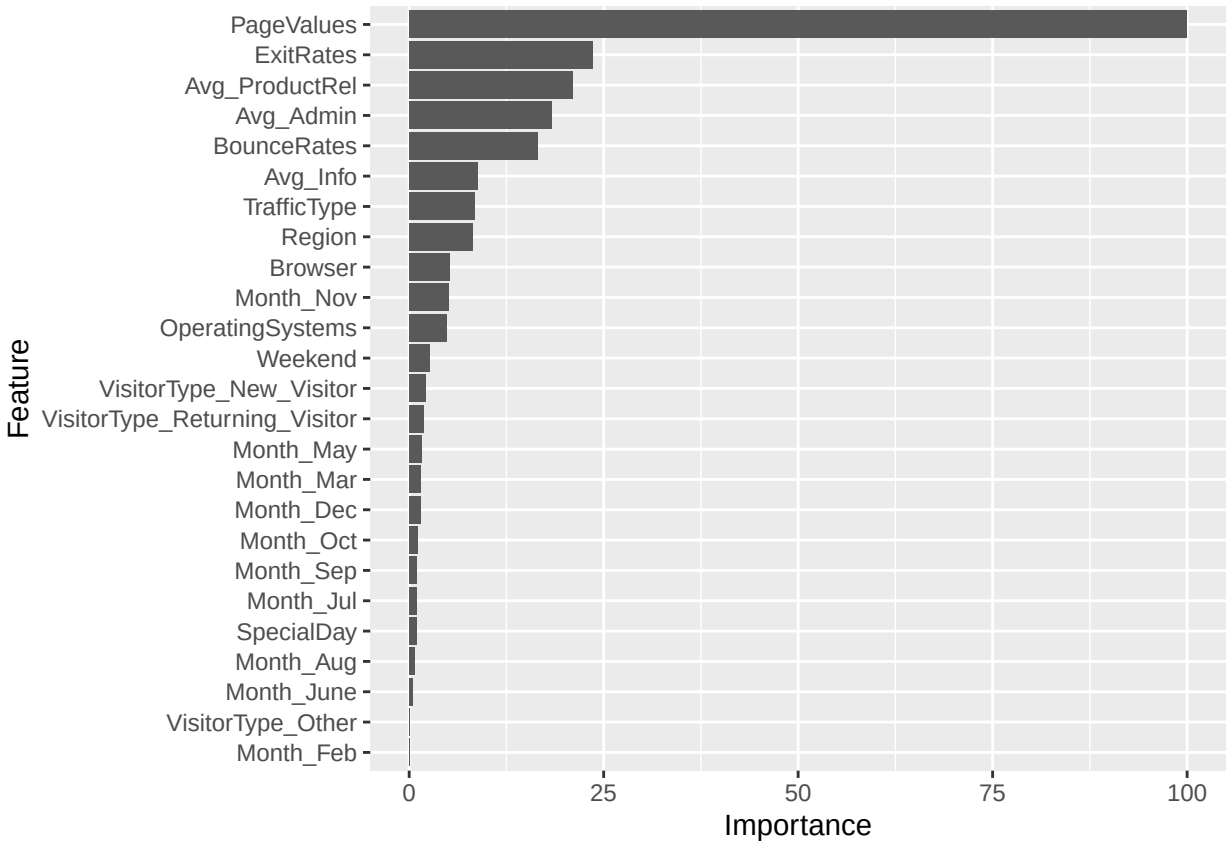

Figure 10: Variable Importance

We will now remove the objects that have significantly low variable importance (<2) to see if it improves our accuracy of the model.

Random Forest with Feature Selection

```
# Selecting features with significant importance and removing low importance
# for users and validation set
users_selected <- users %>% #Feature Selection in user set
  dplyr::select(- VisitorType_Other , - Month_Feb , - Month_June , - Month_Aug , - SpecialDay ,
         -Month_Jul , - Month_Sep , - Month_Oct , - Month_Dec , -Month_Mar , -Month_May,
         - VisitorType_Returning_Visitor)

validation_selected <- validation %>%  #Feature selection in validation set
  dplyr::select(- VisitorType_Other , - Month_Feb , - Month_June , - Month_Aug , - SpecialDay ,
         -Month_Jul , - Month_Sep , - Month_Oct , - Month_Dec , -Month_Mar , -Month_May,
         - VisitorType_Returning_Visitor)
```

```
# Training Random forest on selected features
rf_selected <- train(Revenue~. ,
                     method = 'rf' ,
                     tuneGrid = rf$bestTune,
                     trControl = control ,
```

```
                  ntree = 501 ,
                  data = users_selected)

y_hat_selected <- predict(rf_selected , validation_selected[,-9])
acc_final_selected <- mean(y_hat_selected == validation_selected$Revenue)
```

```
## [1] 0.9140308
```

Our Final Model Accuracy is :

Methods	Accuracy
Random Forest (users v/s validation)	0.9067315
Random Forest Feature Selected	0.9140308

Conclusion

In this report we have studied different supervised learning algorithms on the online shopper intention data. The goal of the work was to identify a suitable model which can predict the purchase intention of a shopper visiting the web-pages of certain online shop more accurately. For this purpose we see that all the algorithms perform significantly well.

We see that complex and memory consuming algorithms such as Random Forest and Loess . perform the best . Random Forest algorithm with manual tuning and training controls performs the best with an accuracy of more than 90%., which shows that with the given data it is possible to predict user intention with significant accuracy .

We also learned that although high dimensionality helps the algorithm . Selecting right features and algorithm can increase the accuracy and efficiency of the model significantly.Our insight also shows that Page Value is very important for revenue generation so focusing on that feature can bring great results.

Although we did try a simpler ensemble model. There are different , more complex ways of performing ensemble which we did not show here. Other methods such as XGBoost classifier can also be user in R using caret which we did not include here . The data could have had other features providing better context to the dataset and users such as buying history and whishlist/cart information could be helpful.