BUAN 6356.003

BUSINESS ANALYTICS WITH R

PROJECT REPORT

GROUP- II

CONTENTS

TITLE

INSIGHTS
ON
E-SHOPPING

TEAM MEMBERS

1. Anu Challa
   Net ID: axx220008

2. Ashna Reddy
   Net ID: axm220066

3. Vishal Kanna
   Net ID: vxn220000

4. Ashwin Kumar
   Net ID: axa210288

ABSTRACT

The research question that we are attempting to answer with our analysis is a predictive question, and is stated as follows:

"Can we forecast if a user will make a purchase on an e-commerce website given their clickstream and session data?"

Due to the quick growth of online goods purchases, internet shopping has become a big industry. With little to no physical presence, such as a traditional brick and mortar store, businesses now frequently offer their items online. For these kinds of businesses, finding an answer to the forementioned question is essential to ensuring that they can continue to be profitable. This data can be utilized to prompt prospective customers to finish an online transaction in real-time, hence raising total purchase conversion rates. The use of social proof to advertise popular products and exit intent overlays on websites are examples of nudges. Online bill payment and shopping both benefit significantly from internet technologies. Data mining is the method that can be utilized to extract valuable information from a lot of data or material. To predict the right transaction or buy intention of clients or consumers, we now use data mining and machine learning algorithms. The study aims to anticipate online buyers' purchasing decisions using 18 variables gathered from their browser data and page information. We looked at the customers' past purchases of the goods to understand their predicting intentions. In this study, we examine consumer buy intentions using empirical data and create a more accurate model to predict consumer purchase intentions.

PROPOSED WORK / SYSTEM

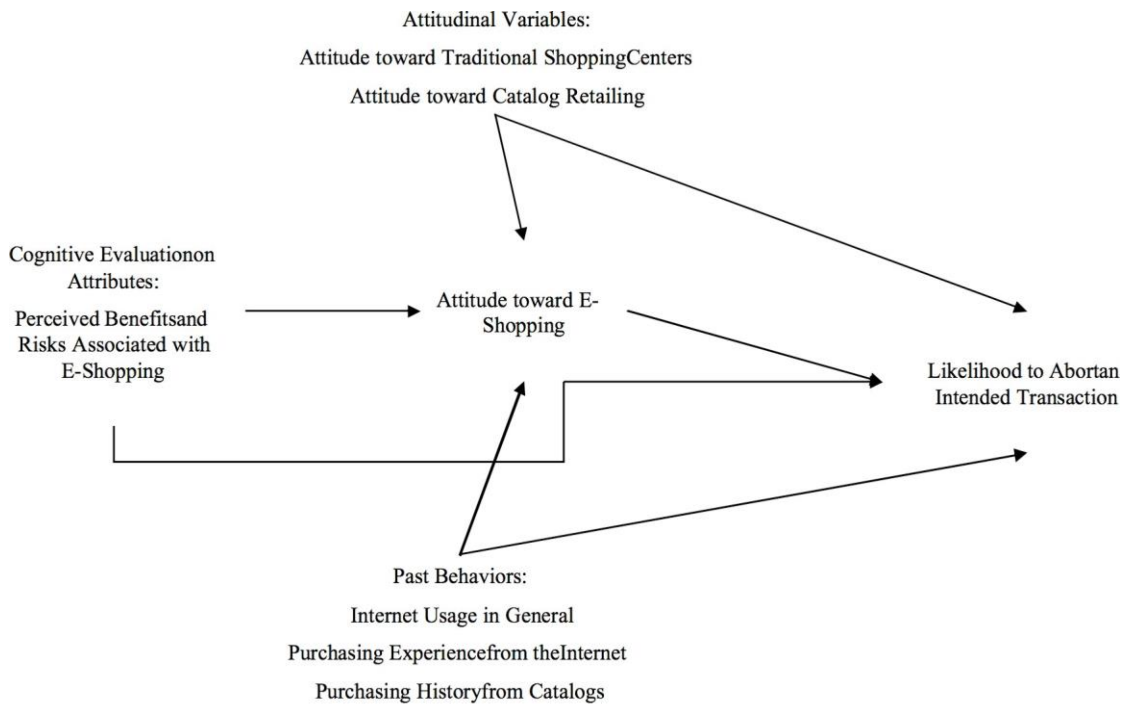


Fig. Likelihood to abort an online transaction: influences from cognitive evaluations, attitudes, and behavioural variables

In this paper, we propose a real-time online shopper behaviour prediction system that predicts the visitor’s shopping intent as soon as the website is visited. We came to the conclusion that it is possible to develop a machine learning model to predict purchase conversion for an e-commerce website; however, before an organization commits to implementing a machine learning model, we would recommend more viable alternatives be first considered. To do that, we rely on session and visitor information, and we investigate naïve Bayes classifier, C4.5 decision tree, C5.0 decision tree, KNN, SVM, and Random Forest. We performed exploratory data analysis to acquire a deep understanding of the underlying nature of data. After experimenting with traditional machine learning techniques such as the tree-based algorithm and the support vector machine, we selected the one that performed the best to predict the purchasing intention and offered a few potential strategies to encourage more customers to complete their purchases. Furthermore, we use oversampling to improve the performance and the scalability of each classifier. Simultaneously, we perform target feature selection to depict the contextual importance of certain desired attributes when compared to the relevant attributes from the same dataset.

To summarize, we make a comparison chart of the accuracies between the different prediction models. The results show that random forest produces significantly higher accuracy and F1 score than the compared techniques. Finally, optimization techniques are applied to the random forest model by selecting the desired features with an ulterior motive to improve the accuracy.

PREPARE PROBLEM STATEMENT

The research question that we are attempting to answer with our analysis is a predictive question, and is stated as follows:

“Can we forecast if a user will make a purchase on an e-commerce website given their clickstream and session data?”

LOAD LIBRARIES

```
library(ggplot)
library(ggplot2)
library(ggdensity)
library(gridExtra)
library(GGally)
library(caret)
library(data.table)
library(ggpubr)
library(ROSE)
library(class)
library(tree)
library(dtree)
library(randomForest)
library(mltools)
library(rsample)
library(e1071)
library(pheatmap)
```

```
library(keras)
library(dummies)
library(mlbench)
library(reticulate)
library(dplyr)
library(infotheo)
library(praznik)
library(ggpubr)
library(corrgram)
library(ggcorr)
library(klaR)
library(caret)
library(tidyverse)
library(data.table)
library(tidymodels)
library(partykit)
library(rpart)
library(rpart.plot)
library(e1071)
library(C50)
library(forecast)
require(randomForest)
library(RWeka)
```

LOAD DATASET

The name of the dataset is called *online_shoppers_intention.csv* and it was downloaded in its raw form from an open-source website.

```
df <- read.csv(file = "online_shoppers_intention.csv")
```

Final.R* df

Filter

	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductRelated	ProductRelated_Duration	BounceRates	ExitRates	PageValues
1	0	0.00000	0	0	1	0.000000	0.200000000	0.200000000	0.00(
2	0	0.00000	0	0	2	64.000000	0.000000000	0.100000000	0.00(
3	0	0.00000	0	0	1	0.000000	0.200000000	0.200000000	0.00(
4	0	0.00000	0	0	2	2.666667	0.050000000	0.140000000	0.00(
5	0	0.00000	0	0	10	627.500000	0.020000000	0.050000000	0.00(
6	0	0.00000	0	0	19	154.216667	0.015789474	0.024561404	0.00(
7	0	0.00000	0	0	1	0.000000	0.200000000	0.200000000	0.00(
8	1	0.00000	0	0	0	0.000000	0.200000000	0.200000000	0.00(
9	0	0.00000	0	0	2	37.000000	0.000000000	0.100000000	0.00(
10	0	0.00000	0	0	3	738.000000	0.000000000	0.022222222	0.00(
11	0	0.00000	0	0	3	395.000000	0.000000000	0.066666667	0.00(
12	0	0.00000	0	0	16	407.750000	0.018750000	0.025833333	0.00(
13	0	0.00000	0	0	7	280.500000	0.000000000	0.028571429	0.00(
14	0	0.00000	0	0	6	98.000000	0.000000000	0.066666667	0.00(
15	0	0.00000	0	0	2	68.000000	0.000000000	0.100000000	0.00(
16	2	53.00000	0	0	23	1668.285119	0.008333333	0.016312636	0.00(
17	0	0.00000	0	0	1	0.000000	0.200000000	0.200000000	0.00(
18	0	0.00000	0	0	13	334.966667	0.000000000	0.007692308	0.00(

Showing 1 to 19 of 12,330 entries, 18 total columns

Fig. Overview of the dataset

SUMMARIZE DATA

DESCRIPTIVE STATISTICS

Dataset Summary:

- This dataset explains about the shopper's intention of purchasing products online.
- The different attributes and products can help in studying the pattern of the purchase and the insights from this can be used to analyze the shopper's intention.

About the Data:

- We have taken the data from *archive.ics.uci.edu/ml/*. The nature of e-shopping varies from country to country, geographically. This dataset includes information on the estimated levels of intentions biased in shoppers to shop products online. Data is gathered from a survey to identify an underlying pattern. This data can be used to identify the behavioural patterns of the online shoppers.
- The data contains 12,330 records and 18 columns.
- The data has no missing values.

Column Descriptions:

Administrative: This is the number of pages of this type (administrative) that the user visited.

Administrative_Duration: This is the amount of time spent in this category of pages.

Informational: This is the number of pages of this type (informational) that the user visited.

Informational_Duration: This is the amount of time spent in this category of pages.

ProductRelated: This is the number of pages of this type (product related) that the user visited.

ProductRelated_Duration: This is the amount of time spent in this category of pages.

BounceRates: The percentage of visitors who enter the website through that page and exit without triggering any additional tasks.

ExitRates: The percentage of pageviews on the website that end at that specific page.

PageValues: The average value of the page averaged over the value of the target page and/or the completion of an eCommerce transaction.

SpecialDay: This value represents the closeness of the browsing date to special days or holidays (eg Mother's Day or Valentine's day) in which the transaction is more likely to be finalized. More information about how this value is calculated below.

Month: Contains the month the pageview occurred, in string form.

OperatingSystems: An integer value representing the operating system that the user was on when viewing the page.

Browser: An integer value representing the browser that the user was using to view the page.

Region: An integer value representing which region the user is located in.

TrafficType: An integer value representing what type of traffic the user is categorized into.

VisitorType: A string representing whether a visitor is New Visitor, Returning Visitor, or Other.

Weekend: A boolean representing whether the session is on a weekend.

Revenue: A boolean representing whether or not the user completed the purchase.

```
> str(df)
'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" "Returning_Visitor" ...
 $ Weekend                : logi  FALSE FALSE FALSE FALSE TRUE FALSE ...
 $ Revenue                : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
```

Fig. Structure of the dataset

```
> head(df)
  Administrative Administrative_Duration Informational Informational_Duration ProductRelated ProductRelated_Duration
1              0                       0             0                      0              1                0.000000
2              0                       0             0                      0              2               64.000000
3              0                       0             0                      0              1                0.000000
4              0                       0             0                      0              2                2.666667
5              0                       0             0                      0             10              627.500000
6              0                       0             0                      0             19              154.216667
  BounceRates ExitRates PageValues SpecialDay Month OperatingSystems Browser Region TrafficType       VisitorType Weekend
1  0.20000000 0.2000000          0          0   Feb                1       1      1           1 Returning_Visitor   FALSE
2  0.00000000 0.1000000          0          0   Feb                2       2      1           2 Returning_Visitor   FALSE
3  0.20000000 0.2000000          0          0   Feb                4       1      9           3 Returning_Visitor   FALSE
4  0.05000000 0.1400000          0          0   Feb                3       2      2           4 Returning_Visitor   FALSE
5  0.02000000 0.0500000          0          0   Feb                3       3      1           4 Returning_Visitor    TRUE
6  0.01578947 0.0245614          0          0   Feb                2       2      1           3 Returning_Visitor   FALSE
  Revenue
1   FALSE
2   FALSE
3   FALSE
4   FALSE
5   FALSE
6   FALSE
```

Fig. Head details of the dataset

```
> summary(df)
 Administrative    Administrative_Duration Informational     Informational_Duration ProductRelated    ProductRelated_Duration
 Min.   : 0.000   Min.   :   0.00         Min.   : 0.0000   Min.   :   0.00        Min.   :  0.00   Min.   :    0.0
 1st Qu.: 0.000   1st Qu.:   0.00         1st Qu.: 0.0000   1st Qu.:   0.00        1st Qu.:  7.00   1st Qu.:  184.1
 Median : 1.000   Median :   7.50         Median : 0.0000   Median :   0.00        Median : 18.00   Median :  598.9
 Mean   : 2.315   Mean   :  80.82         Mean   : 0.5036   Mean   :  34.47        Mean   : 31.73   Mean   : 1194.8
 3rd Qu.: 4.000   3rd Qu.:  93.26         3rd Qu.: 0.0000   3rd Qu.:   0.00        3rd Qu.: 38.00   3rd Qu.: 1464.2
 Max.   :27.000   Max.   :3398.75         Max.   :24.0000   Max.   :2549.38        Max.   :705.00   Max.   :63973.5
  BounceRates         ExitRates          PageValues        SpecialDay          Month           OperatingSystems    Browser
 Min.   :0.000000   Min.   :0.00000   Min.   :  0.000   Min.   :0.00000   Length:12330       Min.   :1.000    Min.   : 1.000
 1st Qu.:0.000000   1st Qu.:0.01429   1st Qu.:  0.000   1st Qu.:0.00000   Class :character   1st Qu.:2.000    1st Qu.: 2.000
 Median :0.003112   Median :0.02516   Median :  0.000   Median :0.00000   Mode  :character   Median :2.000    Median : 2.000
 Mean   :0.022191   Mean   :0.04307   Mean   :  5.889   Mean   :0.06143                      Mean   :2.124    Mean   : 2.357
 3rd Qu.:0.016813   3rd Qu.:0.05000   3rd Qu.:  0.000   3rd Qu.:0.00000                      3rd Qu.:3.000    3rd Qu.: 2.000
 Max.   :0.200000   Max.   :0.20000   Max.   :361.764   Max.   :1.00000                      Max.   :8.000    Max.   :13.000
     Region        TrafficType     VisitorType          Weekend          Revenue
 Min.   :1.000   Min.   : 1.00   Length:12330       Mode :logical   Mode :logical
 1st Qu.:1.000   1st Qu.: 2.00   Class :character   FALSE:9462      FALSE:10422
 Median :3.000   Median : 2.00   Mode  :character   TRUE :2868      TRUE :1908
 Mean   :3.147   Mean   : 4.07
 3rd Qu.:4.000   3rd Qu.: 4.00
 Max.   :9.000   Max.   :20.00
```

Fig. Summary of the dataset

DATA VISUALIZATION

Exploratory Data Analysis:

The first 6 attributes from the dataset represent the number of pages visited of different types and time spent, of which the medians of numbers are 1, 0 and 18 and the medians of time are 9, 0, 608.9 respectively. It illustrates that only a small portion of visitors choose to dig in information about one product, but the probability to explore more about related products is relatively much higher.

Fig. Number of pages visited Vs. the total time spent respectively on different types of pages

The customers who completed transaction tended to browse more and spend more time on Administrative and ProductRelated pages while it seems that they spend less time on Informational pages, which is a bit surprising since it means the majority are loyal customers who added items to the cart and click check-out.

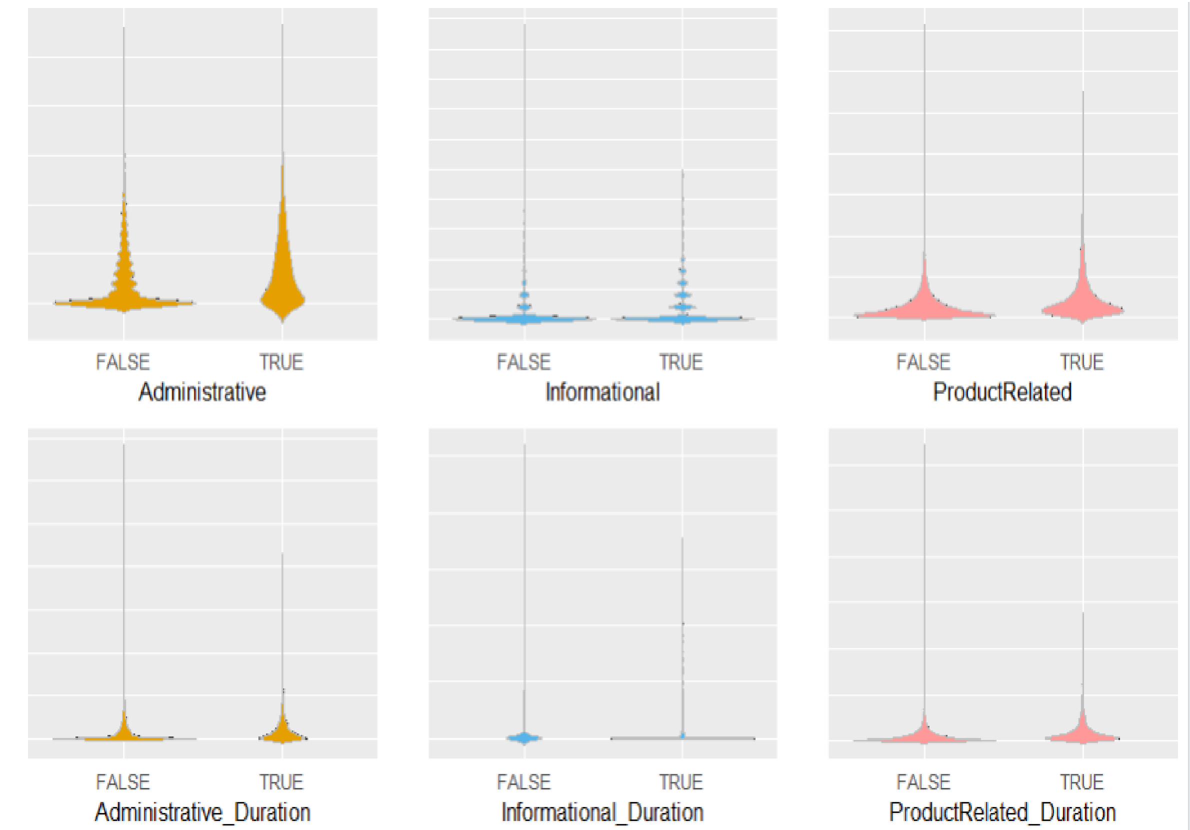


Fig. Side-by-side comparison chart of the time spent on different page types by the customers who completed a transaction

The “Bounce Rates”, “Exit Rates” and “Page Values” features represent the metrics measured by “Google Analytics” for each page in the e-commerce site. The value of “Bounce Rates” feature for a web page refers to the percentage of visitors who enter the site from that page and then leave without triggering any other requests to the analytics server during that session. The value of “Exit Rates” feature for a specific web page is calculated as for all page viewers to the page, the percentage that was the last in the session. The “Page Values” feature represents the average value for a web page that a user visited before completing an e-commerce site.

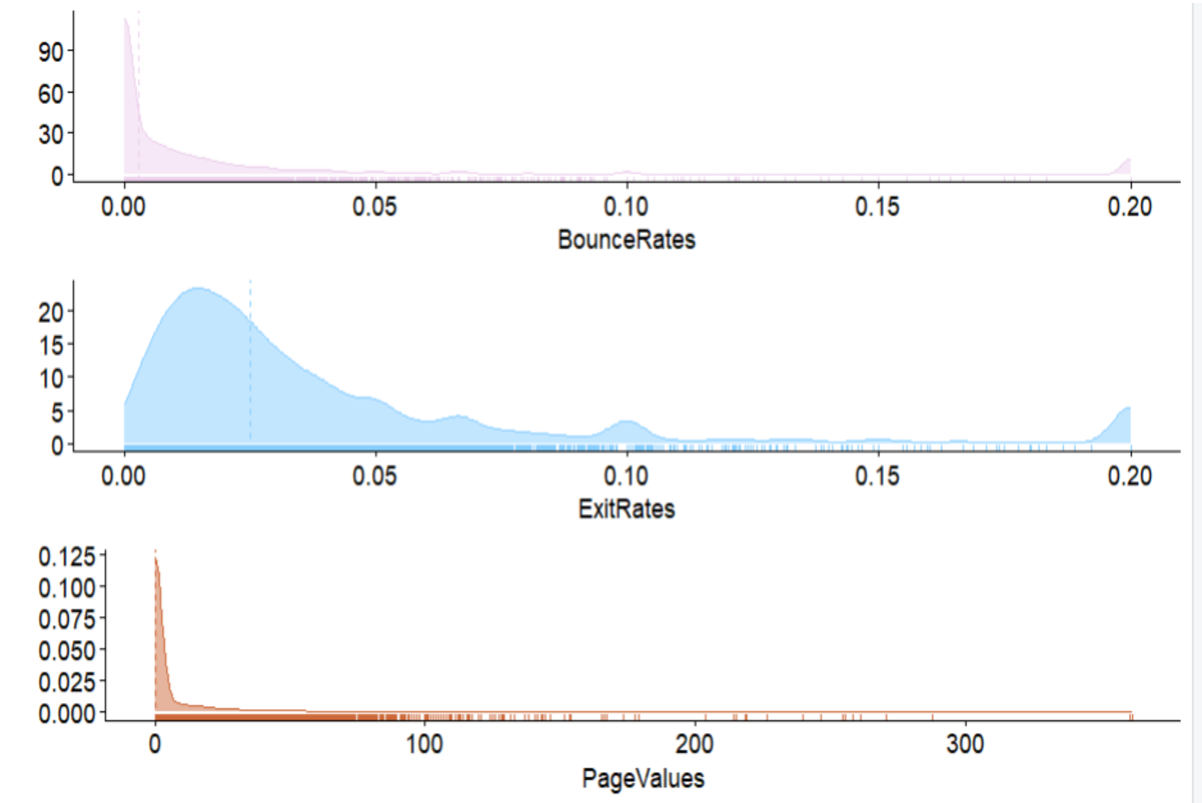


Fig. “Bounce Rates”, “Exit Rates” and “Page Values” features extracted from the corresponding dataset

The chart attached below shows no significant difference of BounceRates between the two customer categories, the ExitRates of T-customers is in general lower than that of F-customers, because they stayed on the pages with higher probability, the Pagevalues of F-customers is way less than that of T-customers because they spent less time on related pages.

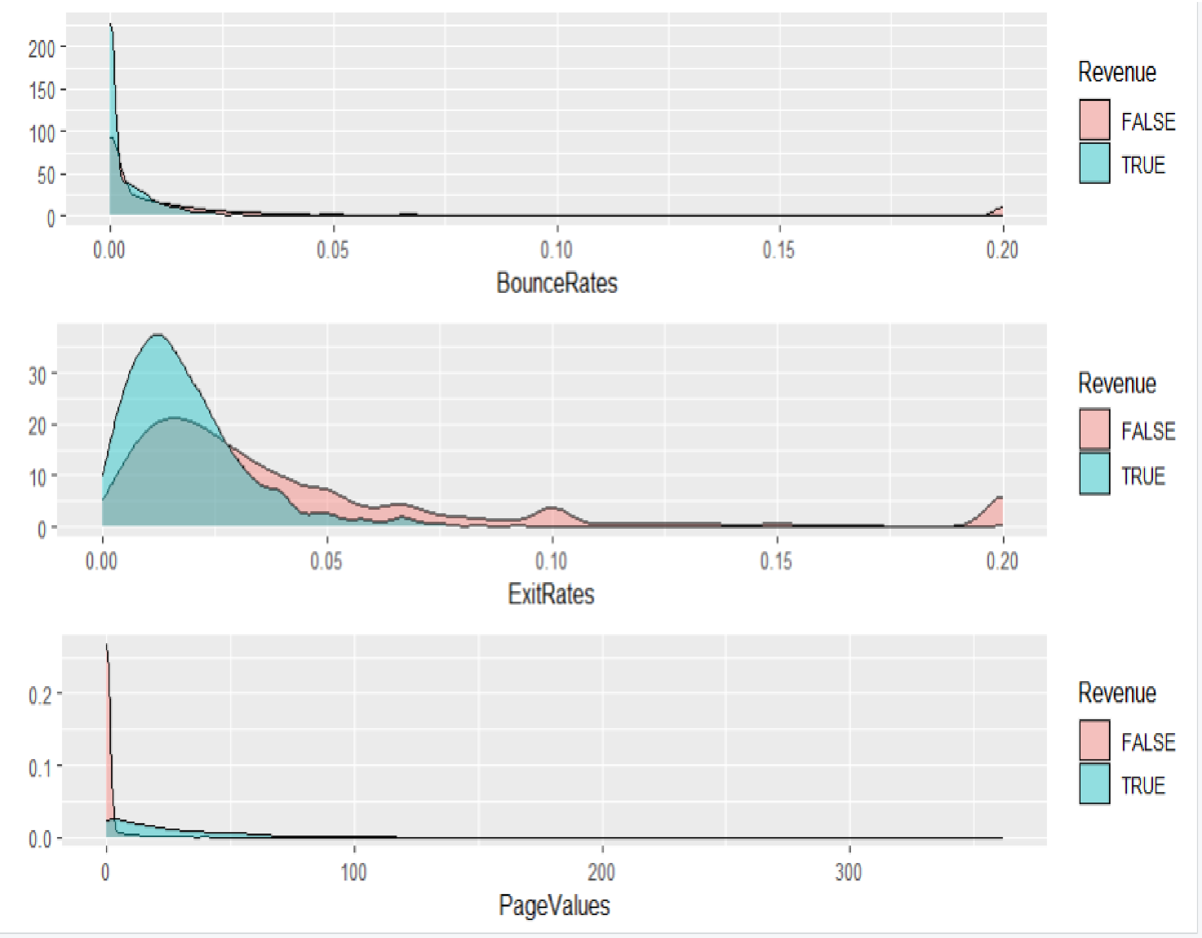


Fig. "Bounce Rates", "Exit Rates" and "Page Values" feature comparison between T-customer and F-customer

The “Special Day” feature indicates the closeness of the site visiting time to a specific special day in which the sessions are more likely to be finalized with transaction. The majority of transaction are done close to none of the special days, since there are merely 1 or 2 holidays, but shopping is around 365 days. The maximum value of this feature is 1 on exactly the date of the special day and the minimum is 0 if the date is too far from any of the special days, other nonzero values represent the influence of the closest special day. T-customers were more likely to purchase on non-special days, which is kind of consistent with our observation that the majority of customers are loyal ones, so their decisions are less affected by whether it is near special days.

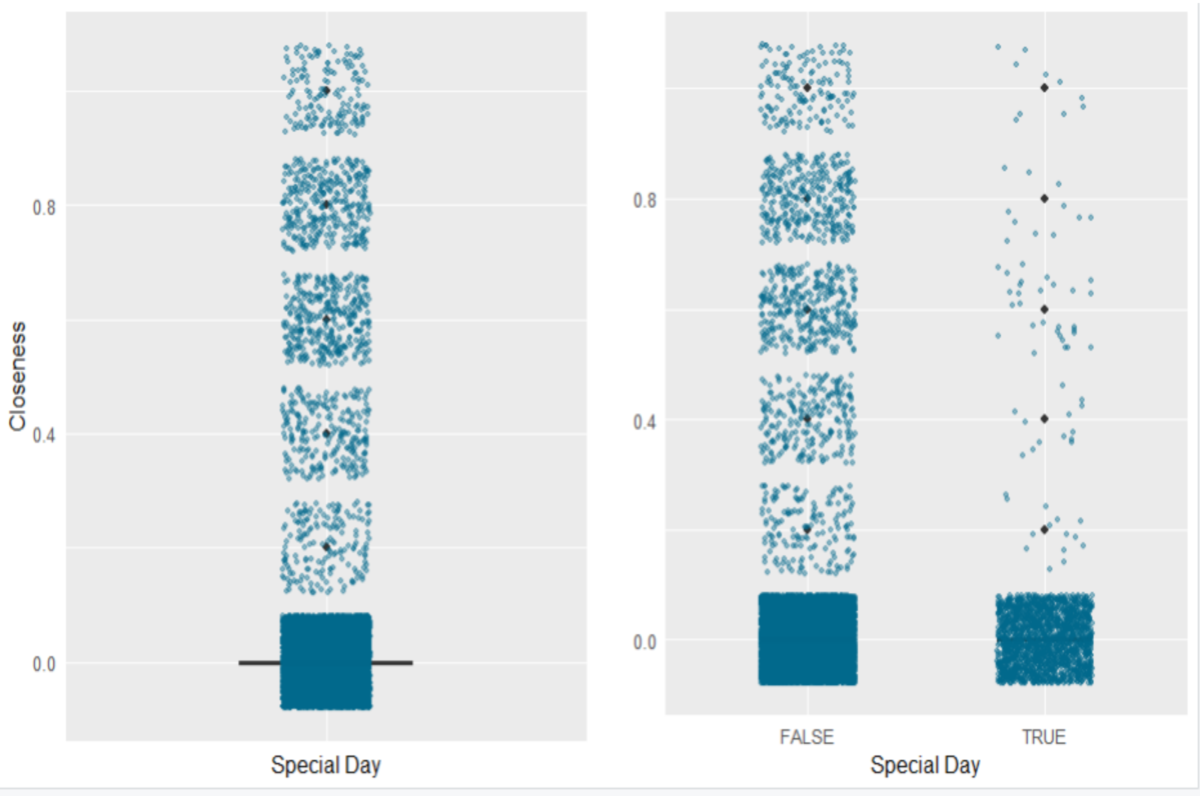


Fig. Transaction behavioural pattern between T-customers and F-customers in the event of Special and Non-special days

The “Month”, “OperatingSystems”, “Browser”, “Region”, “TrafficType” and “Weekend” are straightforward features. The “VisitorType” feature indicates whether the visitor is a new visitor or returning.

The majority of purchasing happened in March, May, November and December, in other words, head and tail of a year. It reasonable since May is the time switching from winter to spring, more families and individuals buy goods for the new season, not to say, November and December are good time to prepare for Christmas, the most important day in a year, therefore the deals witnessed such a significant rise. There shows no significant difference between the trends of two customer categories since the reasons like seasons apply to everyone.

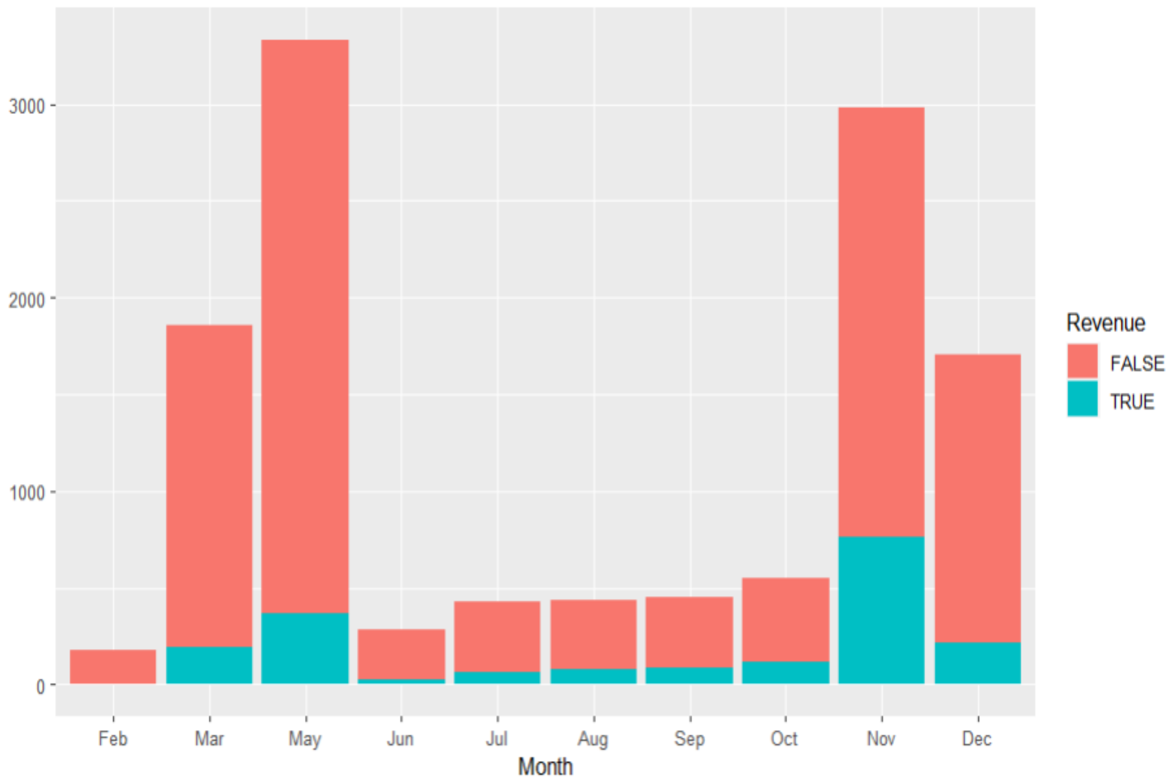


Fig. Month-wise distribution of purchasing behaviour depicted between T-customers and F-customers

There are different types of Operating Systems, Browser, Region and Traffic Type, although details are not given on what each category means. On retrospective, it might be helpful to classify customers. It shows that number of deals happening in weekdays are approximately 3.5 times that in weekends, there is no big difference between them. Return customers are over 5 times new customers, the reason could be that the majority of customers explore deeper and then complete the transaction.

Fig. T-customer and F-customer categorization based upon OS, browser, region, traffic type, weekend, and visitor type

The target "Revenue" demonstrates that the majority of customers failed to complete the transaction, which means that the dataset is extremely imbalanced.

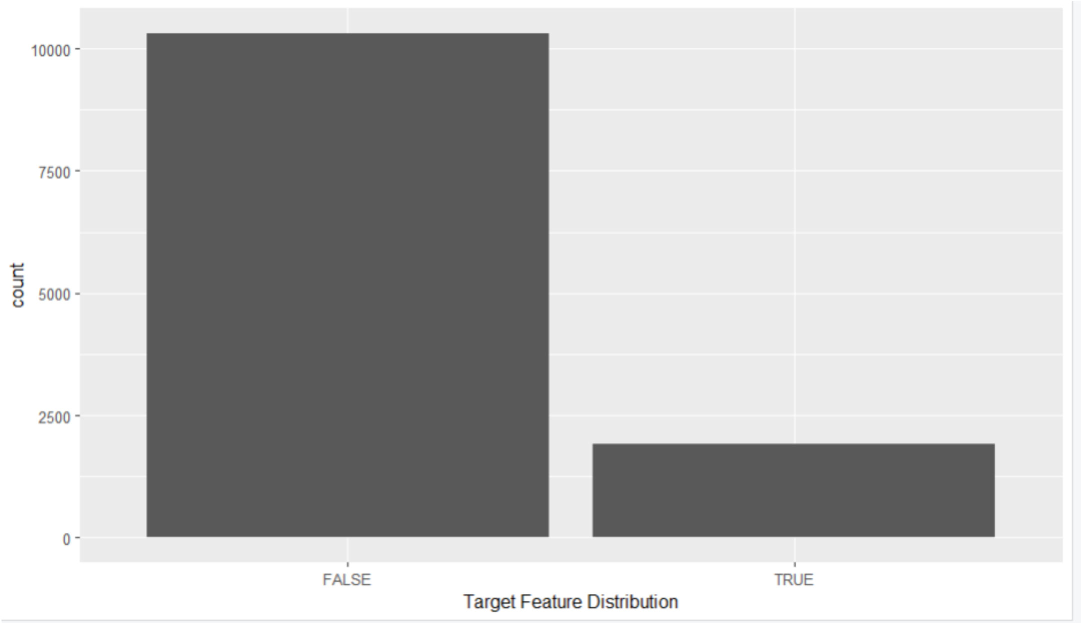


Fig. Number of customers completing/ cancelling the transaction Vs. Target feature distribution

There appears to exist very high-correlated pairs like BounceRates&ExitRates and ProductRelated&ProductRelated_Duration. But, one of each pairs might be dropped considering their importance to accuracy of our model.

Fig. Measure of correlation between different attributes of the dataset

PREPARE DATA

DATA CLEANING

```
> which(is.na(df))
integer(0)
>
```

It is evident that there is no missing data present in the dataset. However, we noticed the presence of redundant data. In an effort to remove duplicate data, the following operation is performed.

```
#removing duplicates
df_duplicate <- nrow(df[duplicated(df),])
df <- df[!duplicated(df),]
str(df)
```

```
> str(df)
'data.frame':	12205 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" "Returning_Visitor" ...
 $ Weekend                : logi  FALSE FALSE FALSE FALSE TRUE FALSE ...
 $ Revenue                : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
>
```

Fig. Structure of data frame after successful removal of redundant data

Inconsistent data entries with respect to the visited web pages may exist in collected data. An inconsistency case is when a customer visits a site repeatedly, we lose information about which pages customer browse in which session. Therefore, we also lose the information about transaction, which prevents us from classifying the intention of the visitor. In order to handle data inconsistency cases, following algorithm applied to clean gathered statistical data.

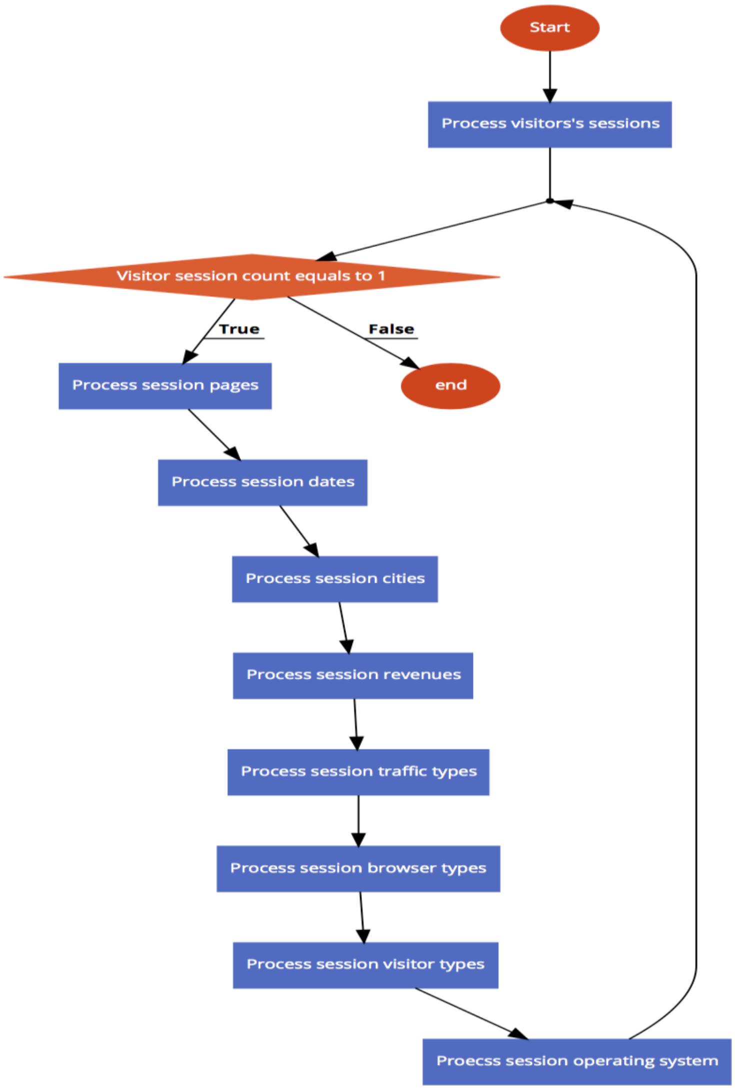


Fig. Handling data inconsistency algorithm

After processing the dataset and cleaning the inconsistencies, the numerical and categorical features used in the purchasing intention prediction model are identified. The dataset consists of feature vectors belonging to 12330 sessions. The dataset was formed so that each session would belong to a different user in a one-year period to avoid any proclivity to a specific campaign, special day, user profile, or period. Of the 12330 sessions in the dataset, 84.5 percentage (10422) were negative class samples that did not end with shopping, and the rest (1908) were positive class samples ending with shopping.

Feature Name	**Feature Description**	**Min Value**	**Max Value**	**Std. Dev.**
Administrative	Number of pages visited by the visitor about account management	0	27	3.32
Administrative Duration	Total amount of time (in seconds) spent by the visitor on account management related pages	0	3398	176.70
Informational	Number of pages visited by the visitor about website, communication and address information of the shopping site	0	24	1.26
Informational Duration	Total amount of time (in seconds) spent by the visitor on informational pages	0	2549	140.64
Product Related	Number of pages visited by visitor about product related pages	0	705	44.45
Product Related Duration	Total amount of time (in seconds) spent by the visitor on product related pages	0	63973	1912.25
Bounce Rate	Average bounce rate value of the pages visited by the visitor	0	0.2	0.04
Exit Rate	Average exit rate value of the pages visited by the visitor	0	0.2	0.05
Page Value	Average page value of the pages visited by the visitor	0	361	18.55
Special Day	Closeness of the site visiting time to a special day	0	1.0	0.19

The above table shows the numerical features along with their statistical parameters. Among these features, "Administrative", "Administrative Duration", "Informational", "Informational Duration", "Product Related" and "Product Related Duration" represent the number of different types of pages visited by the customer and total time spent in each of these page types in seconds. The values of these features are derived from the URL information of the pages visited by the user and updated in real time when a user takes an action, e.g. moving from one page to another.

Feature Name	**Feature Description**	**Number of Categorical Values**
Operating Systems	Operating system of the visitor	8
Browser	Browser of the visitor	13
Region	Geographic region from which the session has been started by the visitor	9
Traffic Type	Traffic source by which the visitor has arrived at the website (e.g. banner, SMS, direct)	20
Visitor Type	Visitor type as “New Visitor”, “Returning Visitor” and “Other”	3
Weekend	Boolean value indicating whether the date of the visit is weekend	2
Month	Month value of the visit date	12
Revenue	Class label indicating whether the visit has been finalized with a transaction	2

The above table shows the categorical features along with their categorical values. The "Operating Systems", " Browser", " Traffic Type" and “Visitor Type” features shown in the table represent the metrics measured by Google Analytics for each page in the e-commerce site. “Weekend” and “Month” features are derived by looking date of visit. They give information about whether the date of visit is at the end of the week or not and the moth of the visit respectively. “Revenue” feature indicates that whether the visit results in transaction finalization.

FEATURE SELECTION

Feature selection is the process of selecting a subset of relevant attributes to be used in making the model in machine learning. Effective feature selection eliminates redundant variables and keeps only the best subset of predictors in the model which also gives shorter training times. Besides this, it avoids the curse of dimensionality and enhance generalization by reducing overfitting.

In this research, feature selection techniques are applied to improve the classification performance and/or scalability of the system. Thus, we aim to investigate if better or similar classification performance can be achieved with a smaller number of features. An alternative of feature selection is the use a feature extraction technique such as Principal Component Analysis for dimensionality reduction. However, in this case, the features in the reduced space will be the linear combinations of 17 attributes, which brings the need of tracking all features during the visit and updating the feature vector after a new action is taken by the visitor. Therefore, it has been deemed appropriate to apply feature selection instead of feature extraction within the scope of this research. For feature ranking, instead of wrapper algorithms that require a learning algorithm to be used and consequently can result in reduced feature sets specific to that classifier, filter-based algorithms are tested in which no classification algorithm is used.

Correlation Attribute Evaluation, Information Gain Attribute Evaluation and Minimum Redundancy Maximum Relevance Filters were used in our experiments. In mRMR algorithm, the aim is to maximize the relevance between the selected set of features and class variable while avoiding the redundancy among the selected features. Thus, maximum classification accuracy is aimed to be obtained with minimal subset of features.

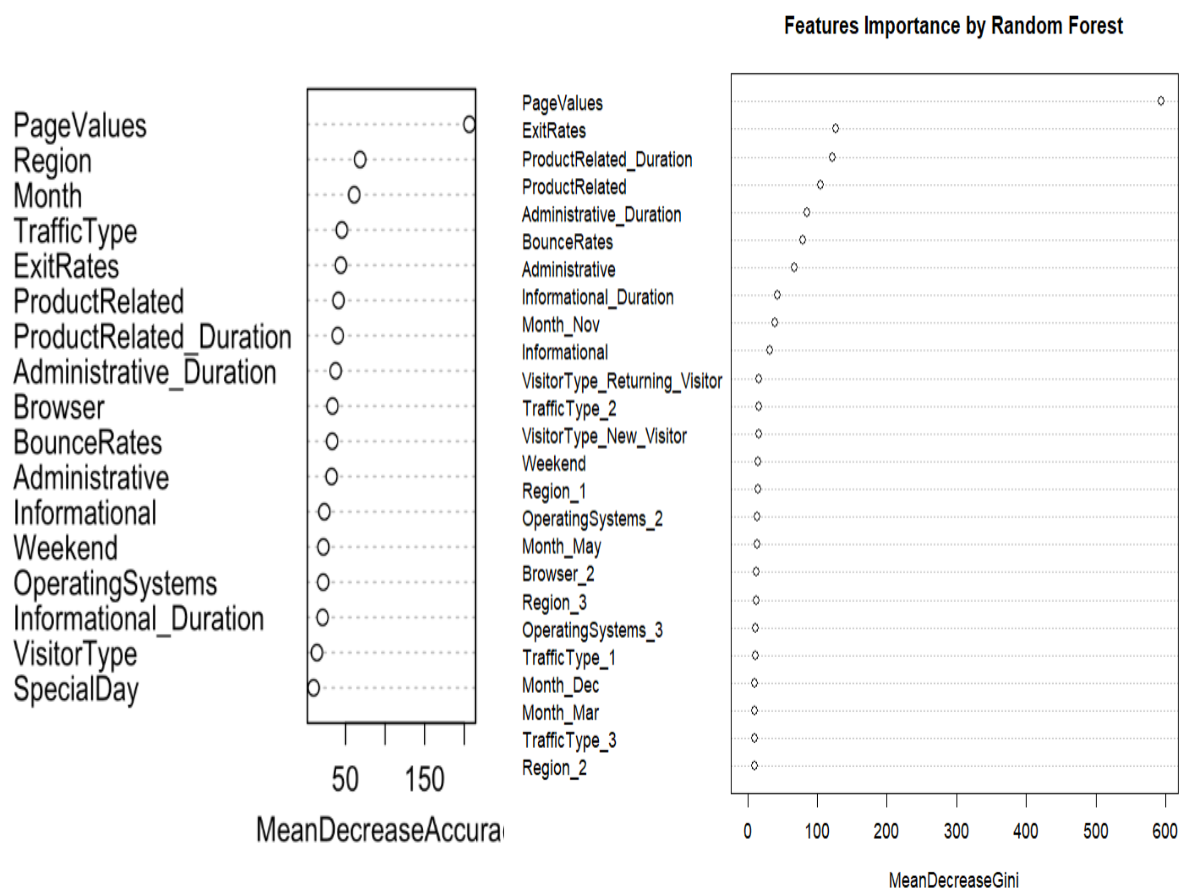


Fig. Target feature selection measuring MDA and MDG

Function “varImpPlot” is used which measures the importance of features by 2 metrics: MeanDecreaseAccuracy and MeanDecreaseGini, MDA means the decrease of accuracy after exclusion or permutation of a single variable, MDG means the decrease of node impurity.

The result of varImpPlot shows that PageValues is definitely the most important one, attributes like Month, ExitRates, ProductRelated/ProductRelated_Duration are among the most important ones as well. Both measurements show that "SpecialDay" is not an important feature, which is consistent with our result of exploratory analysis.

DATA TRANSFORMATION/ PRE-PROCESSING

The attribute "Month" includes 10 months except January and April, and we change "June" to "Jun" for convenience of plotting bar plot in order during exploratory data analysis.

```
#Renaming June to Jun for convenience of plotting
df$Month <- as.character(df$Month)
df$Month[df$Month == "June"] <- "Jun"
df$Month <- as.factor(df$Month)
df$Month = factor(df$Month, levels = month.abb)
```

Fig. Code snippet

One-Hot Encoding:

One-Hot Encoding is a method to encode categorical variables to numerical data that Machine Learning algorithms can deal with. One-Hot encoding is most used during feature engineering for a ML Model. It converts categorical values into a new categorical column and assign a binary value of 1 or 0 to those columns.

Also known as Dummy Encoding, One-Hot Encoding is a method to encode categorical variables, where no such ordinal relationship exists, to numerical data that Machine Learning algorithms can deal with. One hot encoding is the most widespread approach, and it works very well unless your categorical variable takes on a large number of unique values. One hot encoding creates new, binary columns, indicating the presence of each possible value from the original data. These columns store ones and zeros for each row, indicating the categorical value of that row.

```
> print("Original dataset")
[1] "Original dataset"
> print(str(df_new))
'data.frame':	12205 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                  : Factor w/ 12 levels "Jan","Feb","Mar",..: 2 2 2 2 2 2 2 2 2 2 ...
 $ OperatingSystems       : Factor w/ 8 levels "1","2","3","4",..: 1 2 4 3 3 2 2 1 2 2 ...
 $ Browser                : Factor w/ 13 levels "1","2","3","4",..: 1 2 1 2 3 2 4 2 2 4 ...
 $ Region                 : Factor w/ 9 levels "1","2","3","4",..: 1 1 9 2 1 1 3 1 2 1 ...
 $ TrafficType            : Factor w/ 20 levels "1","2","3","4",..: 1 2 3 4 4 3 3 5 3 2 ...
 $ VisitorType            : Factor w/ 3 levels "New_Visitor",..: 3 3 3 3 3 3 3 3 3 3 ...
 $ Weekend                : int  0 0 0 0 1 0 0 1 0 0 ...
 $ Revenue                : int  0 0 0 0 0 0 0 0 0 0 ...
NULL
> 
```

Fig. One-hot encoding to save copy of the original dataset

After one-hot encoding:

We transform categorical attributes into factor types and do one-hot encoding, save a copy of the original data since some methods may not perform better without encoding.

```
> encoded_df <- one_hot(as.data.table(df[,-18]))
> df <- cbind(encoded_df, df[,18]);
> colnames(df)[colnames(df)=="V2"] = "Revenue"
> #df$Revenue <- as.factor(df$Revenue)
> print("After one-hot encoding")
[1] "After one-hot encoding"
> print(str(df))
Classes ‘data.table’ and 'data.frame':  12205 obs. of  77 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_Jan                     : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Month_Feb                     : int  1 1 1 1 1 1 1 1 1 1 ...
 $ Month_Mar                     : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Month_Apr                     : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Month_May                     : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Month_Jun                     : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Month_Jul                     : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Month_Aug                     : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Month_Sep                     : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Month_Oct                     : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Month_Nov                     : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Month_Dec                     : int  0 0 0 0 0 0 0 0 0 0 ...
 $ OperatingSystems_1            : int  1 0 0 0 0 0 0 1 0 0 ...
 $ OperatingSystems_2            : int  0 1 0 0 0 1 1 0 1 1 ...
 $ OperatingSystems_3            : int  0 0 0 1 1 0 0 0 0 0 ...
 $ OperatingSystems_4            : int  0 0 1 0 0 0 0 0 0 0 ...
 $ OperatingSystems_5            : int  0 0 0 0 0 0 0 0 0 0 ...
 $ OperatingSystems_6            : int  0 0 0 0 0 0 0 0 0 0 ...
 $ OperatingSystems_7            : int  0 0 0 0 0 0 0 0 0 0 ...
 $ OperatingSystems_8            : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Browser_1                     : int  1 0 1 0 0 0 0 0 0 0 ...
 $ Browser_2                     : int  0 1 0 1 0 1 0 1 1 0 ...
 $ Browser_3                     : int  0 0 0 0 1 0 0 0 0 0 ...
 $ Browser_4                     : int  0 0 0 0 0 0 1 0 0 1 ...
 $ Browser_5                     : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Browser_6                     : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Browser_7                     : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Browser_8                     : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Browser_9                     : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Browser_10                    : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Browser_11                    : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Browser_12                    : int  0 0 0 0 0 0 0 0 0 0
```

Fig. Performing one hot encoding on all columns except Revenue

Split training and testing data of original dataset before one-hot encoding:

```
split_df_new                  Large initial_split (4 elements,  1.3 MB)
  $ data  :'data.frame': 12205 obs. of  18 variables:
   ..$ Administrative         : int [1:12205] 0 0 0 0 0 0 0 1 0 0 ...
   ..$ Administrative_Duration: num [1:12205] 0 0 0 0 0 0 0 0 0 0 ...
   ..$ Informational          : int [1:12205] 0 0 0 0 0 0 0 0 0 0 ...
   ..$ Informational_Duration : num [1:12205] 0 0 0 0 0 0 0 0 0 0 ...
   ..$ ProductRelated         : int [1:12205] 1 2 1 2 10 19 1 0 2 3 ...
   ..$ ProductRelated_Duration: num [1:12205] 0 64 0 2.67 627.5 ...
   ..$ BounceRates            : num [1:12205] 0.2 0 0.2 0.05 0.02 ...
   ..$ ExitRates              : num [1:12205] 0.2 0.1 0.2 0.14 0.05 ...
   ..$ PageValues             : num [1:12205] 0 0 0 0 0 0 0 0 0 0 ...
   ..$ SpecialDay             : num [1:12205] 0 0 0 0 0 0 0.4 0 0.8 0.4 ...
   ..$ Month                  : Factor w/ 12 levels "Jan","Feb","Mar",..: 2 2 2 2 2 2 ...
   ..$ OperatingSystems       : Factor w/ 8 levels "1","2","3","4",..: 1 2 4 3 3 2 2 1...
   ..$ Browser                : Factor w/ 13 levels "1","2","3","4",..: 1 2 1 2 3 2 4 ...
   ..$ Region                 : Factor w/ 9 levels "1","2","3","4",..: 1 1 9 2 1 1 3 1...
   ..$ TrafficType            : Factor w/ 20 levels "1","2","3","4",..: 1 2 3 4 4 3 3 ...
   ..$ VisitorType            : Factor w/ 3 levels "New_Visitor",..: 3 3 3 3 3 3 3 3 3...
   ..$ Weekend                : int [1:12205] 0 0 0 0 1 0 0 1 0 0 ...
   ..$ Revenue                : int [1:12205] 0 0 0 0 0 0 0 0 0 0 ...
  $ in_id : int [1:8542] 1 2 3 4 5 6 7 9 10 12 ...
  $ out_id: logi NA
  $ id    : tibble [1 x 1] (S3: tbl_df/tbl/data.frame)
   ..$ id: chr "Resample1"
  - attr(*, "class")= chr [1:3] "initial_split" "mc_split" "rsplit"
```

Fig. Splitting the dataset using initial_split() function

```
train_df_new                  8542 obs. of 18 variables
  $ Administrative         : int  0 0 0 0 0 0 0 0 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 2 3 16 ...
  $ 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 0.8 0.4 0.4 ...
  $ Month                  : Factor w/ 12 levels "Jan","Feb","Mar",..: 2 2 2 2 2 2 2 2...
  $ OperatingSystems       : Factor w/ 8 levels "1","2","3","4",..: 1 2 4 3 3 2 2 2 ...
  $ Browser                : Factor w/ 13 levels "1","2","3","4",..: 1 2 1 2 3 2 4 2 4...
  $ Region                 : Factor w/ 9 levels "1","2","3","4",..: 1 1 9 2 1 1 3 2 1 ...
  $ TrafficType            : Factor w/ 20 levels "1","2","3","4",..: 1 2 3 4 4 3 3 3 2...
  $ VisitorType            : Factor w/ 3 levels "New_Visitor",..: 3 3 3 3 3 3 3 3 3 ...
  $ Weekend                : int  0 0 0 0 1 0 0 0 0 0 ...
  $ Revenue                : int  0 0 0 0 0 0 0 0 0 0 ...
```

Fig. Training data

```
test_df_new                 3663 obs. of 18 variables
  $ Administrative         : int  1 0 0 0 0 0 0 0 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  0 3 7 6 1 8 2 3 1 5 ...
  $ ProductRelated_Duration: num  0 395 280 98 0 ...
  $ BounceRates            : num  0.2 0 0 0 0.2 0 0.2 0 0.2 0 ...
  $ ExitRates              : num  0.2 0.0667 0.0286 0.0667 0.2 ...
  $ PageValues             : num  0 0 0 0 0 0 0 0 0 0 ...
  $ SpecialDay             : num  0 0 0 0 0 1 0 0 0 0 ...
  $ Month                  : Factor w/ 12 levels "Jan","Feb","Mar",..: 2 2 2 2 2 2 2 2...
  $ OperatingSystems       : Factor w/ 8 levels "1","2","3","4",..: 1 1 1 2 1 2 3 3 2 ...
  $ Browser                : Factor w/ 13 levels "1","2","3","4",..: 2 1 1 5 1 2 3 2 2...
  $ Region                 : Factor w/ 9 levels "1","2","3","4",..: 1 3 1 1 4 5 1 1 4 ...
  $ TrafficType            : Factor w/ 20 levels "1","2","3","4",..: 5 3 3 3 3 1 3 5 1...
  $ VisitorType            : Factor w/ 3 levels "New_Visitor",..: 3 3 3 3 3 3 3 3 3 3 ...
  $ Weekend                : int  1 0 0 0 0 1 0 0 1 0 ...
  $ Revenue                : int  0 0 0 0 0 0 0 0 0 0 ...
```

Fig. Testing data

```
> print("Original dataset")
[1] "Original dataset"
> table(train_df_new$Revenue) %>% prop.table()

        0         1 
0.8437134 0.1562866 
> table(test_df_new$Revenue) %>% prop.table()

        0         1 
0.8435708 0.1564292 
> 
```

Fig. Calculating the value of each cell in a table as a proportion of all values.

Split training and testing data after one-hot encoding:

```
> print("After one-hot encoding")
[1] "After one-hot encoding"
> table(train_data$Revenue) %>% prop.table()

        0         1 
0.8437134 0.1562866 
> table(test_data$Revenue) %>% prop.table()

        0         1 
0.8435708 0.1564292 
> 
```

```
split                    Large initial_split (4 elements,   4.2 MB)
    $ data  :Classes 'data.table' and 'data.frame':      12205 obs. of  77 variables:
    ..$ Administrative              : int [1:12205] 0 0 0 0 0 0 0 1 0 0 ...
    ..$ Administrative_Duration     : num [1:12205] 0 0 0 0 0 0 0 0 0 0 ...
    ..$ Informational               : int [1:12205] 0 0 0 0 0 0 0 0 0 0 ...
    ..$ Informational_Duration      : num [1:12205] 0 0 0 0 0 0 0 0 0 0 ...
    ..$ ProductRelated              : int [1:12205] 1 2 1 2 10 19 1 0 2 3 ...
    ..$ ProductRelated_Duration     : num [1:12205] 0 64 0 2.67 627.5 ...
    ..$ BounceRates                 : num [1:12205] 0.2 0 0.2 0.05 0.02 ...
    ..$ ExitRates                   : num [1:12205] 0.2 0.1 0.2 0.14 0.05 ...
    ..$ PageValues                  : num [1:12205] 0 0 0 0 0 0 0 0 0 0 ...
    ..$ SpecialDay                  : num [1:12205] 0 0 0 0 0 0.4 0 0.8 0.4 ...
    ..$ Month_Jan                   : int [1:12205] 0 0 0 0 0 0 0 0 0 0 ...
    ..$ Month_Feb                   : int [1:12205] 1 1 1 1 1 1 1 1 1 1 ...
    ..$ Month_Mar                   : int [1:12205] 0 0 0 0 0 0 0 0 0 0 ...
    ..$ Month_Apr                   : int [1:12205] 0 0 0 0 0 0 0 0 0 0 ...
    ..$ Month_May                   : int [1:12205] 0 0 0 0 0 0 0 0 0 0 ...
    ..$ Month_Jun                   : int [1:12205] 0 0 0 0 0 0 0 0 0 0 ...
    ..$ Month_Jul                   : int [1:12205] 0 0 0 0 0 0 0 0 0 0 ...
    ..$ Month_Aug                   : int [1:12205] 0 0 0 0 0 0 0 0 0 0 ...
    ..$ Month_Sep                   : int [1:12205] 0 0 0 0 0 0 0 0 0 0 ...
    ..$ Month_Oct                   : int [1:12205] 0 0 0 0 0 0 0 0 0 0 ...
    ..$ Month_Nov                   : int [1:12205] 0 0 0 0 0 0 0 0 0 0 ...
    ..$ Month_Dec                   : int [1:12205] 0 0 0 0 0 0 0 0 0 0 ...
    ..$ OperatingSystems_1          : int [1:12205] 1 0 0 0 0 0 0 1 0 0 ...
    ..$ OperatingSystems_2          : int [1:12205] 0 1 0 0 0 1 1 0 1 1 ...
    ..$ OperatingSystems_3          : int [1:12205] 0 0 0 1 1 0 0 0 0 0 ...
    ..$ OperatingSystems_4          : int [1:12205] 0 0 1 0 0 0 0 0 0 0 ...
    ..$ OperatingSystems_5          : int [1:12205] 0 0 0 0 0 0 0 0 0 0 ...
    ..$ OperatingSystems_6          : int [1:12205] 0 0 0 0 0 0 0 0 0 0 ...
    ..$ OperatingSystems_7          : int [1:12205] 0 0 0 0 0 0 0 0 0 0 ...
    ..$ OperatingSystems_8          : int [1:12205] 0 0 0 0 0 0 0 0 0 0 ...
    ..$ Browser_1                   : int [1:12205] 1 0 1 0 0 0 0 0 0 0 ...
    ..$ Browser_2                   : int [1:12205] 0 1 0 1 0 1 0 1 1 0 ...
    ..$ Browser_3                   : int [1:12205] 0 0 0 0 1 0 0 0 0 0 ...
    ..$ Browser_4                   : int [1:12205] 0 0 0 0 0 0 1 0 0 1 ...
```

Fig. Splitting the dataset using initial_split() function after one-hot encoding

```
train_data               8542 obs. of 77 variables
   $ Administrative              : int  0 0 0 0 0 1 0 0 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 1 2 19 1 0 2 3 3 16 ...
   $ ProductRelated_Duration     : num  0 0 2.67 154.22 0 ...
   $ BounceRates                 : num  0.2 0.2 0.05 0.0158 0.2 ...
   $ ExitRates                   : num  0.2 0.2 0.14 0.0246 0.2 ...
   $ PageValues                  : num  0 0 0 0 0 0 0 0 0 0 ...
   $ SpecialDay                  : num  0 0 0 0.4 0 0.8 0.4 0 0.4 ...
   $ Month_Jan                   : int  0 0 0 0 0 0 0 0 0 0 ...
   $ Month_Feb                   : int  1 1 1 1 1 1 1 1 1 1 ...
   $ Month_Mar                   : int  0 0 0 0 0 0 0 0 0 0 ...
   $ Month_Apr                   : int  0 0 0 0 0 0 0 0 0 0 ...
   $ Month_May                   : int  0 0 0 0 0 0 0 0 0 0 ...
   $ Month_Jun                   : int  0 0 0 0 0 0 0 0 0 0 ...
   $ Month_Jul                   : int  0 0 0 0 0 0 0 0 0 0 ...
   $ Month_Aug                   : int  0 0 0 0 0 0 0 0 0 0 ...
   $ Month_Sep                   : int  0 0 0 0 0 0 0 0 0 0 ...
   $ Month_Oct                   : int  0 0 0 0 0 0 0 0 0 0 ...
   $ Month_Nov                   : int  0 0 0 0 0 0 0 0 0 0 ...
   $ Month_Dec                   : int  0 0 0 0 0 0 0 0 0 0 ...
   $ OperatingSystems_1          : int  1 0 0 0 0 1 0 0 1 1 ...
   $ OperatingSystems_2          : int  0 0 0 1 1 0 1 1 0 0 ...
   $ OperatingSystems_3          : int  0 0 1 0 0 0 0 0 0 0 ...
   $ OperatingSystems_4          : int  0 1 0 0 0 0 0 0 0 0 ...
   $ OperatingSystems_5          : int  0 0 0 0 0 0 0 0 0 0 ...
   $ OperatingSystems_6          : int  0 0 0 0 0 0 0 0 0 0 ...
   $ OperatingSystems_7          : int  0 0 0 0 0 0 0 0 0 0 ...
   $ OperatingSystems_8          : int  0 0 0 0 0 0 0 0 0 0 ...
   $ Browser_1                   : int  1 1 0 0 0 0 0 0 1 1 ...
   $ Browser_2                   : int  0 0 1 1 0 1 1 0 0 0 ...
   $ Browser_3                   : int  0 0 0 0 0 0 0 0 0 0 ...
   $ Browser_4                   : int  0 0 0 0 1 0 0 1 0 0 ...
   $ Browser_5                   : int  0 0 0 0 0 0 0 0 0 0 ...
   $ Browser_6                   : int  0 0 0 0 0 0 0 0 0 0 ...
```

Fig. Training data

```
test_data                    3663 obs. of 77 variables
   $ Administrative            : int  0 0 2 0 0 0 0 4 0 0 ...
   $ Administrative_Duration   : num  0 0 53 0 0 0 0 64.6 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  2 10 23 1 13 20 2 32 10 8 ...
   $ ProductRelated_Duration   : num  64 628 1668 0 335 ...
   $ BounceRates               : num  0 0.02 0.00833 0.2 0 ...
   $ ExitRates                 : num  0.1 0.05 0.01631 0.2 0.00769 ...
   $ PageValues                : num  0 0 0 0 0 0 0 0 0 0 ...
   $ SpecialDay                : num  0 0 0 0 0 0 0 0 0 0 ...
   $ Month_Jan                 : int  0 0 0 0 0 0 0 0 0 0 ...
   $ Month_Feb                 : int  1 1 1 1 1 1 1 1 1 1 ...
   $ Month_Mar                 : int  0 0 0 0 0 0 0 0 0 0 ...
   $ Month_Apr                 : int  0 0 0 0 0 0 0 0 0 0 ...
   $ Month_May                 : int  0 0 0 0 0 0 0 0 0 0 ...
   $ Month_Jun                 : int  0 0 0 0 0 0 0 0 0 0 ...
   $ Month_Jul                 : int  0 0 0 0 0 0 0 0 0 0 ...
   $ Month_Aug                 : int  0 0 0 0 0 0 0 0 0 0 ...
   $ Month_Sep                 : int  0 0 0 0 0 0 0 0 0 0 ...
   $ Month_Oct                 : int  0 0 0 0 0 0 0 0 0 0 ...
   $ Month_Nov                 : int  0 0 0 0 0 0 0 0 0 0 ...
   $ Month_Dec                 : int  0 0 0 0 0 0 0 0 0 0 ...
   $ OperatingSystems_1        : int  0 0 1 1 1 0 0 0 1 0 ...
   $ OperatingSystems_2        : int  1 0 0 0 0 1 0 1 0 0 ...
   $ OperatingSystems_3        : int  0 1 0 0 0 0 1 0 0 1 ...
   $ OperatingSystems_4        : int  0 0 0 0 0 0 0 0 0 0 ...
   $ OperatingSystems_5        : int  0 0 0 0 0 0 0 0 0 0 ...
   $ OperatingSystems_6        : int  0 0 0 0 0 0 0 0 0 0 ...
   $ OperatingSystems_7        : int  0 0 0 0 0 0 0 0 0 0 ...
   $ OperatingSystems_8        : int  0 0 0 0 0 0 0 0 0 0 ...
   $ Browser_1                 : int  0 0 1 1 1 0 0 0 1 0 ...
   $ Browser_2                 : int  1 0 0 0 0 0 0 1 0 1 ...
   $ Browser_3                 : int  0 1 0 0 0 0 1 0 0 0 ...
   $ Browser_4                 : int  0 0 0 0 0 1 0 0 0 0 ...
   $ Browser_5                 : int  0 0 0 0 0 0 0 0 0 0 ...
   $ Browser_6                 : int  0 0 0 0 0 0 0 0 0 0 ...
```

Fig. Testing data

In the next step, we pre-process the continuous attributes by splitting from categorical ones and binding later.

```
test_numerical               3663 obs. of 10 variables
   $ Administrative         : int  0 0 2 0 0 0 0 4 0 0 ...
   $ Administrative_Duration: num  0 0 53 0 0 0 0 64.6 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  2 10 23 1 13 20 2 32 10 8 ...
   $ ProductRelated_Duration: num  64 628 1668 0 335 ...
   $ BounceRates            : num  0 0.02 0.00833 0.2 0 ...
   $ ExitRates              : num  0.1 0.05 0.01631 0.2 0.00769 ...
   $ PageValues             : num  0 0 0 0 0 0 0 0 0 0 ...
   $ SpecialDay             : num  0 0 0 0 0 0 0 0 0 0 ...
   - attr(*, ".internal.selfref")=<externalptr>
```

Fig. Numerical attributes in testing data

```
test_categorical        3663 obs. of 67 variables
    $ Month_Jan                : int  0 0 0 0 0 0 0 0 0 0 ...
    $ Month_Feb                : int  1 1 1 1 1 1 1 1 1 1 ...
    $ Month_Mar                : int  0 0 0 0 0 0 0 0 0 0 ...
    $ Month_Apr                : int  0 0 0 0 0 0 0 0 0 0 ...
    $ Month_May                : int  0 0 0 0 0 0 0 0 0 0 ...
    $ Month_Jun                : int  0 0 0 0 0 0 0 0 0 0 ...
    $ Month_Jul                : int  0 0 0 0 0 0 0 0 0 0 ...
    $ Month_Aug                : int  0 0 0 0 0 0 0 0 0 0 ...
    $ Month_Sep                : int  0 0 0 0 0 0 0 0 0 0 ...
    $ Month_Oct                : int  0 0 0 0 0 0 0 0 0 0 ...
    $ Month_Nov                : int  0 0 0 0 0 0 0 0 0 0 ...
    $ Month_Dec                : int  0 0 0 0 0 0 0 0 0 0 ...
    $ OperatingSystems_1       : int  0 0 1 1 1 0 0 0 1 0 ...
    $ OperatingSystems_2       : int  1 0 0 0 0 1 0 1 0 0 ...
    $ OperatingSystems_3       : int  0 1 0 0 0 0 1 0 0 1 ...
    $ OperatingSystems_4       : int  0 0 0 0 0 0 0 0 0 0 ...
    $ OperatingSystems           t  0 0 0 0 0 0 0 0 0 0 ...
    $ OperatingSystems           t  0 0 0 0 0 0 0 0 0 0 ...
    $ OperatingSystems           t  0 0 0 0 0 0 0 0 0 0 ...
    $ OperatingSystems           t  0 0 0 0 0 0 0 0 0 0 ...
    $ Browser_1                  t  0 0 1 1 1 0 0 0 1 0 ...
    $ Browser_2                : int  1 0 0 0 0 0 0 1 0 1 ...
    $ Browser_3                : int  0 1 0 0 0 0 1 0 0 0 ...
    $ Browser_4                : int  0 0 0 0 0 1 0 0 0 0 ...
    $ Browser_5                : int  0 0 0 0 0 0 0 0 0 0 ...
    $ Browser_6                : int  0 0 0 0 0 0 0 0 0 0 ...
    $ Browser_7                : int  0 0 0 0 0 0 0 0 0 0 ...
    $ Browser_8                : int  0 0 0 0 0 0 0 0 0 0 ...
    $ Browser_9                : int  0 0 0 0 0 0 0 0 0 0 ...
    $ Browser_10               : int  0 0 0 0 0 0 0 0 0 0 ...
    $ Browser_11               : int  0 0 0 0 0 0 0 0 0 0 ...
    $ Browser_12               : int  0 0 0 0 0 0 0 0 0 0 ...
    $ Browser_13               : int  0 0 0 0 0 0 0 0 0 0 ...
    $ Region_1                 : int  1 1 0 0 1 0 1 1 0 1 ...
    $ Region_2                 : int  0 0 0 0 0 0 0 0 0 0 ...
    $ Region_3                 : int  0 0 0 0 0 0 0 0 1 0 ...

$ OperatingSystems_4
: int 0 0 0 0 0 0 0 0
0 0 ...
```

Fig. Categorical attributes in testing data

```
train_numerical         8542 obs. of 10 variables
    $ Administrative         : int  0 0 0 0 0 1 0 0 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 1 2 19 1 0 2 3 3 16 ...
    $ ProductRelated_Duration: num  0 0 2.67 154.22 0 ...
    $ BounceRates            : num  0.2 0.2 0.05 0.0158 0.2 ...
    $ ExitRates              : num  0.2 0.2 0.14 0.0246 0.2 ...
    $ PageValues             : num  0 0 0 0 0 0 0 0 0 0 ...
    $ SpecialDay             : num  0 0 0 0 0.4 0 0.8 0.4 0 0.4 ...
    - attr(*, ".internal.selfref")=<externalptr>
```

Fig. Numerical attributes in training data

```
train_categorical        8542 obs. of 67 variables
  $ Month_Jan              : int  0 0 0 0 0 0 0 0 0 0 ...
  $ Month_Feb              : int  1 1 1 1 1 1 1 1 1 1 ...
  $ Month_Mar              : int  0 0 0 0 0 0 0 0 0 0 ...
  $ Month_Apr              : int  0 0 0 0 0 0 0 0 0 0 ...
  $ Month_May              : int  0 0 0 0 0 0 0 0 0 0 ...
  $ Month_Jun              : int  0 0 0 0 0 0 0 0 0 0 ...
  $ Month_Jul              : int  0 0 0 0 0 0 0 0 0 0 ...
  $ Month_Aug              : int  0 0 0 0 0 0 0 0 0 0 ...
  $ Month_Sep              : int  0 0 0 0 0 0 0 0 0 0 ...
  $ Month_Oct              : int  0 0 0 0 0 0 0 0 0 0 ...
  $ Month_Nov              : int  0 0 0 0 0 0 0 0 0 0 ...
  $ Month_Dec              : int  0 0 0 0 0 0 0 0 0 0 ...
  $ OperatingSystems_1     : int  1 0 0 0 0 1 0 0 1 1 ...
  $ OperatingSystems_2     : int  0 0 0 1 1 0 1 1 0 0 ...
  $ OperatingSystems_3     : int  0 0 1 0 0 0 0 0 0 0 ...
  $ OperatingSystems_4     : int  0 1 0 0 0 0 0 0 0 0 ...
  $ OperatingSystems_5     : int  0 0 0 0 0 0 0 0 0 0 ...
  $ OperatingSystems_6     : int  0 0 0 0 0 0 0 0 0 0 ...
  $ OperatingSystems_7     : int  0 0 0 0 0 0 0 0 0 0 ...
  $ OperatingSystems_8     : int  0 0 0 0 0 0 0 0 0 0 ...
  $ Browser_1              : int  1 1 0 0 0 0 0 0 1 1 ...
  $ Browser_2              : int  0 0 1 1 0 1 1 0 0 0 ...
  $ Browser_3              : int  0 0 0 0 0 0 0 0 0 0 ...
  $ Browser_4              : int  0 0 0 0 1 0 0 1 0 0 ...
  $ Browser_5              : int  0 0 0 0 0 0 0 0 0 0 ...
  $ Browser_6              : int  0 0 0 0 0 0 0 0 0 0 ...
  $ Browser_7              : int  0 0 0 0 0 0 0 0 0 0 ...
  $ Browser_8              : int  0 0 0 0 0 0 0 0 0 0 ...
  $ Browser_9              : int  0 0 0 0 0 0 0 0 0 0 ...
  $ Browser_10             : int  0 0 0 0 0 0 0 0 0 0 ...
  $ Browser_11             : int  0 0 0 0 0 0 0 0 0 0 ...
  $ Browser_12             : int  0 0 0 0 0 0 0 0 0 0 ...
  $ Browser_13             : int  0 0 0 0 0 0 0 0 0 0 ...
  $ Region_1               : int  1 0 0 1 0 1 0 1 0 0 ...
  $ Region_2               : int  0 0 1 0 0 0 1 0 0 0 ...
  $ Region_3               : int  0 0 0 0 1 0 0 0 1 0 ...
```

Fig. Categorical attributes in training data

Utilization of scaling function in the testing and training data:

```
test_scaled     num [1:3663, 1:10] -0.6959 -0.6959 -0.0851 -0.6959 -0.6959 ...
```

```
train_scaled           Large matrix (85420 elements,  687.1 kB)
  - attr(*, "dimnames")=List of 2
   ..$ : NULL
   ..$ : chr [1:10] "Administrative" "Administrative_Duration" "Informational" "Inform…
  - attr(*, "scaled:center")= Named num [1:10] 2.28 79.12 0.51 35.23 31.41 ...
   ..- attr(*, "names")= chr [1:10] "Administrative" "Administrative_Duration" "Inform…
  - attr(*, "scaled:scale")= Named num [1:10] 3.27 176.11 1.29 143.78 42.56 ...
   ..- attr(*, "names")= chr [1:10] "Administrative" "Administrative_Duration" "Inform…
```

Scaling is a technique used for comparing data that isn't measured in the same way. The normalizing of a dataset using the mean value and standard deviation is known as scaling. When working with vectors or columns in a data frame, scaling is frequently employed. You will almost always receive meaningless results if you do not normalize the vectors or columns you are utilizing. Scale() is a built-in R function that centers and/or scales the columns of a numeric matrix by default. Only if the value provided is numeric, the scale() function subtracts the values of each column by the matching "center" value from the argument.

This is also known as data standardization, and it basically involves converting each original value into a z-score. If the value is numeric, the scale() method divides the values of each column by the corresponding scale value from the input. Otherwise, the standard deviation or root-mean-square values are used to split the numbers.

$$\text{x-scaled} = (x - \bar{x}) / s$$

where:

x: real x-value

$\bar{x}$: Sample mean

s: Sample SD

Oversampling:

Due to the imbalance of dataset, the classifier will always try to predict the target as False since it achieves higher accuracy, to solve this problem, we oversample the training data to be balanced while leaving test data unchanged.

Oversampling using the ovun.sample() function:

N	14414
N_df_new	14414

```
df_over                         14414 obs. of 77 variables
    $ Administrative             : num  -0.696 -0.696 -0.696 -0.696 -0.696 ...
    $ Administrative_Duration    : num  -0.449 -0.449 -0.449 -0.449 -0.449 ...
    $ Informational              : num  -0.395 -0.395 -0.395 -0.395 -0.395 ...
    $ Informational_Duration     : num  -0.245 -0.245 -0.245 -0.245 -0.245 ...
    $ ProductRelated             : num  -0.714 -0.714 -0.691 -0.292 -0.714 ...
    $ ProductRelated_Duration    : num  -0.621 -0.621 -0.619 -0.54 -0.621 ...
    $ BounceRates                : num  3.949 3.949 0.647 -0.106 3.949 ...
    $ ExitRates                  : num  3.415 3.415 2.119 -0.374 3.415 ...
    $ PageValues                 : num  -0.315 -0.315 -0.315 -0.315 -0.315 ...
    $ SpecialDay                 : num  -0.308 -0.308 -0.308 -0.308 1.722 ...
    $ Month_Jan                  : int  0 0 0 0 0 0 0 0 0 0 ...
    $ Month_Feb                  : int  1 1 1 1 1 1 1 1 1 1 ...
    $ Month_Mar                  : int  0 0 0 0 0 0 0 0 0 0 ...
    $ Month_Apr                  : int  0 0 0 0 0 0 0 0 0 0 ...
    $ Month_May                  : int  0 0 0 0 0 0 0 0 0 0 ...
    $ Month_Jun                  : int  0 0 0 0 0 0 0 0 0 0 ...
    $ Month_Jul                  : int  0 0 0 0 0 0 0 0 0 0 ...
    $ Month_Aug                  : int  0 0 0 0 0 0 0 0 0 0 ...
    $ Month_Sep                  : int  0 0 0 0 0 0 0 0 0 0 ...
    $ Month_Oct                  : int  0 0 0 0 0 0 0 0 0 0 ...
    $ Month_Nov                  : int  0 0 0 0 0 0 0 0 0 0 ...
    $ Month_Dec                  : int  0 0 0 0 0 0 0 0 0 0 ...
    $ OperatingSystems_1         : int  1 0 0 0 0 1 0 0 1 1 ...
    $ OperatingSystems_2         : int  0 0 0 1 1 0 1 1 0 0 ...
    $ OperatingSystems_3         : int  0 0 1 0 0 0 0 0 0 0 ...
    $ OperatingSystems_4         : int  0 1 0 0 0 0 0 0 0 0 ...
    $ OperatingSystems_5         : int  0 0 0 0 0 0 0 0 0 0 ...
    $ OperatingSystems_6         : int  0 0 0 0 0 0 0 0 0 0 ...
    $ OperatingSystems_7         : int  0 0 0 0 0 0 0 0 0 0 ...
    $ OperatingSystems_8         : int  0 0 0 0 0 0 0 0 0 0 ...
    $ Browser_1                  : int  1 1 0 0 0 0 0 0 1 1 ...
    $ Browser_2                  : int  0 0 1 1 0 1 1 0 0 0 ...
    $ Browser_3                  : int  0 0 0 0 0 0 0 0 0 0 ...
    $ Browser_4                  : int  0 0 0 0 1 0 0 1 0 0 ...
    $ Browser_5                  : int  0 0 0 0 0 0 0 0 0 0 ...
    $ Browser_6                  : int  0 0 0 0 0 0 0 0 0 0
```

```
df_new_over                14414 obs. of 18 variables
  $ Administrative          : int  0 0 0 0 0 0 0 0 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 2 3 16 ...
  $ 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 0.8 0.4 0.4 ...
  $ Month                   : Factor w/ 12 levels "Jan","Feb","Mar",..: 2 2 2 2 2 2 2…
  $ OperatingSystems        : Factor w/ 8 levels "1","2","3","4",..: 1 2 4 3 3 2 2 2 2 …
  $ Browser                 : Factor w/ 13 levels "1","2","3","4",..: 1 2 1 2 3 2 4 2 4…
  $ Region                  : Factor w/ 9 levels "1","2","3","4",..: 1 1 9 2 1 1 3 2 1 …
  $ TrafficType             : Factor w/ 20 levels "1","2","3","4",..: 1 2 3 4 4 3 3 3 2…
  $ VisitorType             : Factor w/ 3 levels "New_Visitor",..: 3 3 3 3 3 3 3 3 3 3 …
  $ Weekend                 : int  0 0 0 0 1 0 0 0 0 0 ...
  $ Revenue                 : int  0 0 0 0 0 0 0 0 0 0 ...
```

Most of the machine learning classification algorithms assume and work better when the number of instances of each class are equal roughly. However, the assumption is not same as in many real-world problems such a fraud detection, network intrusion detection, oil-spill detection, etc. It is often the situation that the ratios of the intermediary classes in these datasets are extremely different which causes imbalanced dataset problem. The problem of dataset in this example is usually related to a case when the minority class has important information which tend to be misclassified when compared to the majority class and it will cause errors in making decision on prediction accuracy of minority class.

There are two methods that should be performed to solve this problem which focuses on either data level aspect by using over sampling and under sampling techniques or focuses on algorithm level aspect by using cost-sensitive learning techniques. There are different cases to choose one of these methods. For example, applying algorithm level solution might not be possible since there are not many cost-sensitive implementations of all learning algorithms. In a different case, datasets might be already large, and size of the training set needs to be reduced to make learning possible. Therefore, applying under sampling might be feasible. Despite that, using under sampling in some datasets might not be feasible since it discards potentially useful data. Although, oversampling has disadvantages as overfitting training data, it is most frequently used to solve imbalanced dataset problem. One of the well-known over sampling method is Synthetic Minority Over Sampling Technique.

In this study, oversampling method is applied to get better results since the ratio of 12330 sessions in the dataset is 84.5 percentage (10422) for negative class samples which do not end with shopping and the rest (1908) are positive class samples ending with successful transactions.

Split features and target for further use:

```
#Splitting features and target
features_df_new <- setdiff(names(train_df_new), "Revenue")
features <- setdiff(names(train_data), "Revenue")
```

features	chr [1:76] "Administrative" "Administrative_Duration" "Informat…
features_df_new	chr [1:17] "Administrative" "Administrative_Duration" "Informat…

In the pursuit of identifying mutually existent data in two diferent datasets, setdiff() function in R Programming Language is used to find the elements which are in the first Object but not in the second Object.

EVALUATE ALGORITHMS

In this part, the issues which this research focused on are explained in detail. The visitor behaviour analysis model is designed as a binary classification (1s and 0s) problem measuring the user's intention to finalize the transaction. In order to predict the purchasing intention of the visitor using aggregated page view data kept track during the visit along with some session and user information. The extracted features are fed to decision tree (DT), support vector machines (SVM), K- Nearest Neighbour (KNN), Naïve Bayes classifiers as input. Therewithal oversampling and feature selection pre-processing steps are used to improve the performance and scalability of the classifiers.

Prediction:

In the scope of this research, the supervised learning algorithms such as k-Nearest Neighbour (KNN), Support Vector Machine (SVM) classifiers are included in the modelling of the visitor behaviour analysis, considering the real time use of the system. Since the system needs to be updated with new examples, Decision Tree (DT) algorithms, which have online learning implementations, are selected for comparison. Besides, SVM's classification ability has been impeccable given its applications.

If SVM achieves significantly higher accuracies than the other classifiers, an online passive-aggressive implementation can be used to dynamically update the SVM model (tuned SVM) with new examples. Using confusion matrix, the performance of the algorithms is compared using accuracy, F1-score and true positive/negative rates.

Naïve Bayes Classifier:

Naive Bayes is a simple technique for constructing classifiers: models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from some finite set. There is not a single algorithm for training such classifiers, but a family of algorithms based on a common principle: all naive Bayes classifiers assume that the value of a particular feature is independent of the value of any other feature, given the class variable. For example, a fruit may be considered to be an apple if it is red, round, and about 10 cm in diameter. A naive Bayes classifier considers each of these features to contribute independently to the probability that this fruit is an apple, regardless of any possible correlations between the color, roundness, and diameter features.

In many practical applications, parameter estimation for naive Bayes models uses the method of maximum likelihood; in other words, one can work with the naive Bayes model without accepting Bayesian probability. Using Bayes' theorem, the conditional probability can be decomposed as:

$$p(C_k \mid \mathbf{x}) = \frac{p(C_k)\, p(\mathbf{x} \mid C_k)}{p(\mathbf{x})}$$

In plain English, using Bayesian probability terminology, the above equation can be written as

$$\text{posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{evidence}}$$

```
x_df_new                    8542 obs. of 17 variables
   $ Administrative         : int  0 0 0 0 0 0 0 0 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 2 3 16 ...
   $ 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 0.8 0.4 0.4 ...
   $ Month                  : Factor w/ 12 levels "Jan","Feb","Mar",..: 2 2 2 2 2 2 2 2…
   $ OperatingSystems       : Factor w/ 8 levels "1","2","3","4",..: 1 2 4 3 3 2 2 2 2 …
   $ Browser                : Factor w/ 13 levels "1","2","3","4",..: 1 2 1 2 3 2 4 2 4…
   $ Region                 : Factor w/ 9 levels "1","2","3","4",..: 1 1 9 2 1 1 3 2 1 …
   $ TrafficType            : Factor w/ 20 levels "1","2","3","4",..: 1 2 3 4 4 3 3 3 2…
   $ VisitorType            : Factor w/ 3 levels "New_Visitor",..: 3 3 3 3 3 3 3 3 3 3 …
   $ Weekend                : int  0 0 0 0 1 0 0 0 0 0 ...
```

```
y_df_new              int [1:8542] 0 0 0 0 0 0 0 0 0 0 ...
```

Fig. 'X' and 'Y' axis attributes for Naïve Bayes Classifier

Splitting training and testing data:

```
x                         8542 obs. of 76 variables
   $ Administrative             : num  -0.696 -0.696 -0.696 -0.696 -0.696 ...
   $ Administrative_Duration    : num  -0.449 -0.449 -0.449 -0.449 -0.449 ...
   $ Informational              : num  -0.395 -0.395 -0.395 -0.395 -0.395 ...
   $ Informational_Duration     : num  -0.245 -0.245 -0.245 -0.245 -0.245 ...
   $ ProductRelated             : num  -0.714 -0.714 -0.691 -0.292 -0.714 ...
   $ ProductRelated_Duration    : num  -0.621 -0.621 -0.619 -0.54 -0.621 ...
   $ BounceRates                : num  3.949 3.949 0.647 -0.106 3.949 ...
   $ ExitRates                  : num  3.415 3.415 2.119 -0.374 3.415 ...
   $ PageValues                 : num  -0.315 -0.315 -0.315 -0.315 -0.315 ...
   $ SpecialDay                 : num  -0.308 -0.308 -0.308 -0.308 1.722 ...
   $ Month_Jan                  : int  0 0 0 0 0 0 0 0 0 0 ...
   $ Month_Feb                  : int  1 1 1 1 1 1 1 1 1 1 ...
   $ Month_Mar                  : int  0 0 0 0 0 0 0 0 0 0 ...
   $ Month_Apr                  : int  0 0 0 0 0 0 0 0 0 0 ...
   $ Month_May                  : int  0 0 0 0 0 0 0 0 0 0 ...
   $ Month_Jun                  : int  0 0 0 0 0 0 0 0 0 0 ...
   $ Month_Jul                  : int  0 0 0 0 0 0 0 0 0 0 ...
   $ Month_Aug                  : int  0 0 0 0 0 0 0 0 0 0 ...
   $ Month_Sep                  : int  0 0 0 0 0 0 0 0 0 0 ...
   $ Month_Oct                  : int  0 0 0 0 0 0 0 0 0 0 ...
   $ Month_Nov                  : int  0 0 0 0 0 0 0 0 0 0 ...
   $ Month_Dec                  : int  0 0 0 0 0 0 0 0 0 0 ...
   $ OperatingSystems_1         : int  1 0 0 0 0 1 0 0 1 1 ...
   $ OperatingSystems_2         : int  0 0 0 1 1 0 1 1 0 0 ...
   $ OperatingSystems_3         : int  0 0 1 0 0 0 0 0 0 0 ...
   $ OperatingSystems_4         : int  0 1 0 0 0 0 0 0 0 0 ...
   $ OperatingSystems_5         : int  0 0 0 0 0 0 0 0 0 0 ...
   $ OperatingSystems_6         : int  0 0 0 0 0 0 0 0 0 0 ...
   $ OperatingSystems_7         : int  0 0 0 0 0 0 0 0 0 0 ...
   $ OperatingSystems_8         : int  0 0 0 0 0 0 0 0 0 0 ...
   $ Browser_1                  : int  1 1 0 0 0 0 0 0 1 1 ...
   $ Browser_2                  : int  0 0 1 1 0 1 1 0 0 0 ...
   $ Browser_3                  : int  0 0 0 0 0 0 0 0 0 0 ...
   $ Browser_4                  : int  0 0 0 0 1 0 0 1 0 0 ...
   $ Browser_5                  : int  0 0 0 0 0 0 0 0 0 0 ...
   $ Browser_6                  : int  0 0 0 0 0 0 0 0 0 0 ...
   $ Browser_7                  : int  0 0 0 0 0 0 0 0 0 0 ...
   $ Browser_8                  : int  0 0 0 0 0 0 0 0 0 0 ...
   $ Browser_9                  : int  0 0 0 0 0 0 0 0 0 0 ...
```

y	Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...

train_control() function: a function to compute performance metrics across resamples. The function trainControl generates parameters that further control how models are created, with possible values: method type.

```
train_control          List of  27
  $ method           : chr "cv"
  $ number           : num 10
  $ repeats          : logi NA
  $ search           : chr "grid"
  $ p                : num 0.75
  $ initialWindow    : NULL
  $ horizon          : num 1
  $ fixedWindow      : logi TRUE
  $ skip             : num 0
  $ verboseIter      : logi FALSE
  $ returnData       : logi TRUE
  $ returnResamp     : chr "final"
  $ savePredictions  : logi FALSE
  $ classProbs       : logi FALSE
  $ summaryFunction  :function (data, lev = NULL, model = NULL)
  $ selectionFunction: chr "best"
  $ preProcOptions   :List of 6
   ..$ thresh   : num
   ..$ ICAcomp  : num
   ..$ k        : num 5
   ..$ freqCut  : num 19
   ..$ uniqueCut: num 10
   ..$ cutoff   : num 0.9
  $ sampling         : NULL
  $ index            : NULL
  $ indexOut         : NULL
  $ indexFinal       : NULL
  $ timingSamps      : num 0
  $ predictionBounds : logi [1:2] FALSE FALSE
  $ seeds            : logi NA
  $ adaptive         :List of 4
   ..$ min     : num 5
   ..$ alpha   : num 0.05
   ..$ method  : chr "gls"
   ..$ complete: logi TRUE
  $ trim             : logi FALSE
  $ allowParallel    : logi TRUE
```

```
$ preProcOptions
:List of 6
```

With one-hot encoding:

```
nb.ml                          Large train (21 elements,  8.5 MB)
  $ method       : chr "nb"
  $ modelInfo    :List of 13
   ..$ label      : chr "Naive Bayes"
   ..$ library    : chr "klaR"
   ..$ loop       : NULL
   ..$ type       : chr "Classification"
   ..$ parameters:'data.frame':  3 obs. of  3 variables:
   .. ..$ parameter: chr [1:3] "fL" "usekernel" "adjust"
   .. ..$ class    : chr [1:3] "numeric" "logical" "numeric"
   .. ..$ label    : chr [1:3] "Laplace Correction" "Distribution Type" "Bandwidth Adj…
   ..$ grid       :function (x, y, len = NULL, search = "grid")
   ..$ fit        :function (x, y, wts, param, lev, last, classProbs, ...)
   ..$ predict    :function (modelFit, newdata, submodels = NULL)
   ..$ prob       :function (modelFit, newdata, submodels = NULL)
   ..$ predictors:function (x, ...)
   ..$ tags       : chr "Bayesian Model"
   ..$ levels     :function (x)
   ..$ sort       :function (x)
  $ modelType    : chr "Classification"
  $ results      :'data.frame':   2 obs. of  7 variables:
   ..$ usekernel : logi [1:2] FALSE TRUE
   ..$ fL        : num [1:2] 0 0
   ..$ adjust    : num [1:2] 1 1
   ..$ Accuracy  : num [1:2] NaN 0.844
   ..$ Kappa     : num [1:2] NaN -0.000233
   ..$ AccuracySD: num [1:2] NA 0.000542
   ..$ KappaSD   : num [1:2] NA 0.000737
  $ pred         : NULL
  $ bestTune     :'data.frame':   1 obs. of  3 variables:
   ..$ fL       : num 0
   ..$ usekernel: logi TRUE
   ..$ adjust   : num 1
  $ call         : language train.default(x = x, y = y, method = "nb", trControl = trai…
  $ dots         : list()
  $ metric       : chr "Accuracy"
  $ control      :List of 27
   ..$ method            : chr "cv"
   ..$ number            : num 10
   ..$ repeats           : logi NA
```

Evaluation metric:

```
> print("With one-hot encoding")
[1] "With one-hot encoding"
> print(confusionMatrix(nb.ml))
Cross-Validated (10 fold) Confusion Matrix

(entries are percentual average cell counts across resamples)

          Reference
Prediction    0    1
         0 84.4 15.6
         1  0.0  0.0

 Accuracy (average) : 0.8436
```

Without one-hot encoding:

```
nb.ml_df_new              List of  21
  $ method       : chr "nb"
  $ modelInfo    :List of 13
   ..$ label       : chr "Naive Bayes"
   ..$ library     : chr "klaR"
   ..$ loop        : NULL
   ..$ type        : chr "Classification"
   ..$ parameters:'data.frame':   3 obs. of  3 variables:
   .. ..$ parameter: chr [1:3] "fL" "usekernel" "adjust"
   .. ..$ class    : chr [1:3] "numeric" "logical" "numeric"
   .. ..$ label    : chr [1:3] "Laplace Correction" "Distribution Type" "Bandwidth Adj…
   ..$ grid       :function (x, y, len = NULL, search = "grid")
   ..$ fit        :function (x, y, wts, param, lev, last, classProbs, ...)
   ..$ predict    :function (modelFit, newdata, submodels = NULL)
   ..$ prob       :function (modelFit, newdata, submodels = NULL)
   ..$ predictors:function (x, ...)
   ..$ tags       : chr "Bayesian Model"
   ..$ levels     :function (x)
   ..$ sort       :function (x)
  $ modelType    : chr "Classification"
  $ results      :'data.frame':   2 obs. of  7 variables:
   ..$ usekernel : logi [1:2] FALSE TRUE
   ..$ fL        : num [1:2] 0 0
   ..$ adjust    : num [1:2] 1 1
   ..$ Accuracy  : num [1:2] 0.808 0.84
   ..$ Kappa     : num [1:2] 0.418 0.485
   ..$ AccuracySD: num [1:2] 0.0146 0.0102
   ..$ KappaSD   : num [1:2] 0.0322 0.0271
  $ pred         : NULL
  $ bestTune     :'data.frame':   1 obs. of  3 variables:
   ..$ fL        : num 0
   ..$ usekernel: logi TRUE
   ..$ adjust    : num 1
  $ call         : language train.default(x = x_df_new, y = as.factor(y_df_new), method…
  $ dots         : list()
  $ metric       : chr "Accuracy"
  $ control      :List of 27
   ..$ method            : chr "cv"
   ..$ number            : num 10
   ..$ repeats           : logi NA
```

Evaluation metrics:

```
> print("Without one-hot encoding")
[1] "Without one-hot encoding"
> print(confusionMatrix(nb.ml_df_new))
Cross-Validated (10 fold) Confusion Matrix

(entries are percentual average cell counts across resamples)

          Reference
Prediction    0    1
         0 73.0  4.6
         1 11.3 11.0

 Accuracy (average) : 0.8404
```

First, we try to predict using NBC, which is fast and scalable. NBC classifies a new customer by conditional probabilities of all the features, picks the class with highest probability:

$P(H|MultipleEvidences)=P(E1|H)*P(E2|H)*...*P(En|H)*P(H)/P(MultipleEvidences)P(H|MultipleEvidences)=P(E1|H)*P(E2|H)*...*P(En|H)*P(H)/P(MultipleEvidences)$

The accuracy is 84.36%, even worse than guessing all are negative, it fails to predict any positive case right if using one-hot encoding, but a good start to explore other methods since it is relatively a simple model.

k- Nearest Neighbour:

The k-nearest neighbours algorithm, also known as KNN or k-NN, is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point. While it can be used for either regression or classification problems, it is typically used as a classification algorithm, working off the assumption that similar points can be found near one another.

Regression problems use a similar concept as classification problem, but in this case, the average the k nearest neighbours is taken to make a prediction about a classification. The main distinction here is that classification is used for discrete values, whereas regression is used with continuous ones. However, before a classification can be made, the distance must be defined. Euclidean distance is most commonly used calculation criterion.

Euclidean distance: This is the most commonly used distance measure, and it is limited to real-valued vectors. Using the below formula, it measures a straight line between the query point and the other point being measured.

$$d(x,y) = \sqrt{\sum_{i=1}^{n} (y_i - x_i)^2}$$

Train the model and predict:

```
knn_model     Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
```

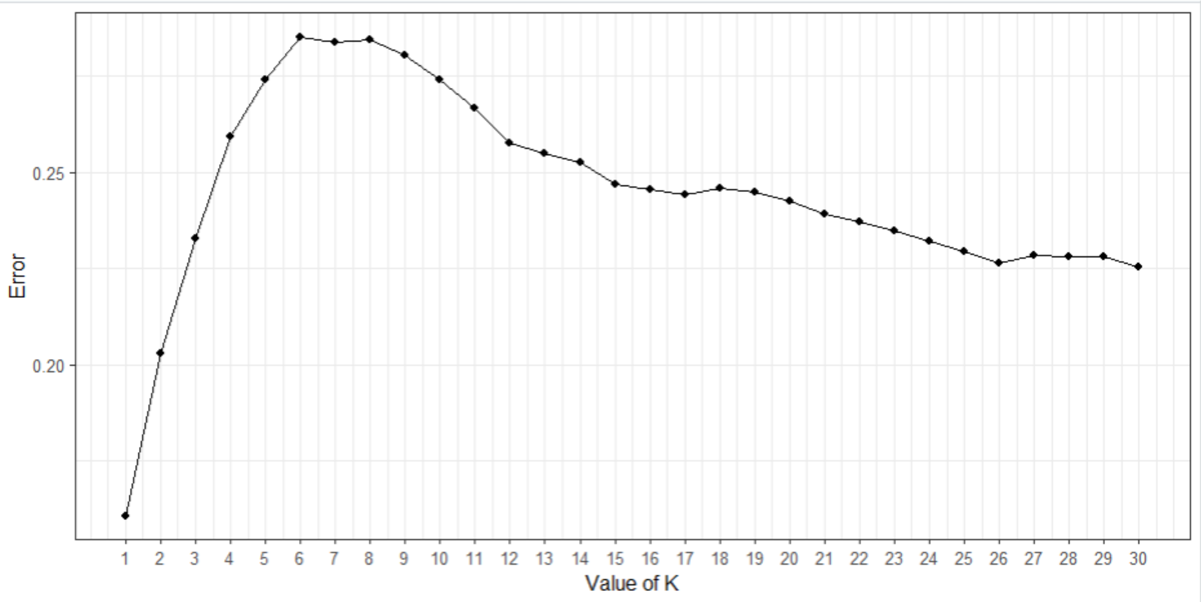
Confusion Matrix and Evaluation Metrics for default k-NN:

```
> #Confusion Matrix and Metrics
> print("Default k-NN")
[1] "Default k-NN"
> CM_knn_default <- confusionMatrix(knn_model, factor(test_data$Revenue))
> print(CM_knn_default)
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 2793  306
         1  297  267

               Accuracy : 0.8354
                 95% CI : (0.823, 0.8473)
    No Information Rate : 0.8436
    P-Value [Acc > NIR] : 0.9166

                  Kappa : 0.3722

 Mcnemar's Test P-Value : 0.7446

            Sensitivity : 0.9039
            Specificity : 0.4660
         Pos Pred Value : 0.9013
         Neg Pred Value : 0.4734
             Prevalence : 0.8436
         Detection Rate : 0.7625
   Detection Prevalence : 0.8460
      Balanced Accuracy : 0.6849

       'Positive' Class : 0

> 
```

Visualizing accuracies of different k:

```
errors                 num [1:30] 0.161 0.203 0.233 0.259 0.274 ...
```

```
knn.error              30 obs. of 2 variables
   $ k     : num  1 2 3 4 5 6 7 8 9 10 ...
   $ errors: num  0.161 0.203 0.233 0.259 0.274 ...
```



From the chart, it can be understood that the error is computed to be at its lowest when k-value equal to 1.

Result of the best-performance model:

When we try k-NN, an unsupervised clustering algorithm, by default it achieves accuracy of 83.54%, we further try visualizing model errors of different k, which shows that k=1 is the one that achieves the highest accuracy. Its result is approximately identical to NBC, but it is still reasonable since there are continuous and categorical attributes and k-nn is an expert on cluster, not on classification.

Evaluation metrics:

```
> print("Best-performance k-NN")
[1] "Best-performance k-NN"
> CM_knn_best <- confusionMatrix(k_nn, factor(test_data$Revenue))
> print(CM_knn_best)
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 2822  321
         1  268  252

               Accuracy : 0.8392
                 95% CI : (0.8269, 0.851)
    No Information Rate : 0.8436
    P-Value [Acc > NIR] : 0.77421

                  Kappa : 0.3669

 Mcnemar's Test P-Value : 0.03214

            Sensitivity : 0.9133
            Specificity : 0.4398
         Pos Pred Value : 0.8979
         Neg Pred Value : 0.4846
             Prevalence : 0.8436
         Detection Rate : 0.7704
   Detection Prevalence : 0.8580
      Balanced Accuracy : 0.6765

       'Positive' Class : 0

>
```

Tree-based methods/ Decision Trees:

The other classifiers used to predict the commercial intent of the visitors are the variants of decision tree algorithms. Decision tree is an efficient non-parametric method that can be used for both classification and regression. A decision tree has two main components: internal decision nodes and terminal leaves. Each internal node in the tree implements a test function using one or more features and each branch descending from that node is labelled with the corresponding discrete outcome.

During testing, when a new instance is given, the test pointed out by the root node is applied to the instance and according to the output of the decision node the next internal node that will be visited is determined. This process is then repeated for the subtree rooted at the new node until a leaf node is encountered which is the output of the constructed tree for the given test instance.

C4.5 Decision Trees:

In this study, we use C4.5 algorithm to generate an individual decision tree for classification. C4.5 builds decision trees from a set of training data in the same way as ID3, using the concept of information entropy. At each node of the tree, C4.5 chooses the attribute of the data that most effectively splits its set of samples into subsets enriched in one class or the other. The splitting criterion is the normalized information gain (difference in entropy). The attribute with the highest normalized information gain is chosen to make the decision. The C4.5 algorithm then recurses on the partitioned sublists.

This algorithm has a few base cases:

- All the samples in the list belong to the same class. When this happens, it simply creates a leaf node for the decision tree saying to choose that class.
- None of the features provide any information gain. In this case, C4.5 creates a decision node higher up the tree using the expected value of the class.
- Instance of previously unseen class encountered. Again, C4.5 creates a decision node higher up the tree using the expected value.

Training and testing the model:

```
train                               9154 obs. of 77 variables
    $ Administrative              : int  0 0 0 0 0 0 0 0 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 2 10 19 1 2 3 7 6 ...
    $ ProductRelated_Duration     : num  0 64 2.67 627.5 154.22 ...
    $ BounceRates                 : num  0.2 0 0.05 0.02 0.0158 ...
    $ ExitRates                   : num  0.2 0.1 0.14 0.05 0.0246 ...
    $ PageValues                  : num  0 0 0 0 0 0 0 0 0 0 ...
    $ SpecialDay                  : num  0 0 0 0 0 0.4 0.8 0.4 0 0 ...
    $ Month_Jan                   : int  0 0 0 0 0 0 0 0 0 0 ...
    $ Month_Feb                   : int  1 1 1 1 1 1 1 1 1 1 ...
    $ Month_Mar                   : int  0 0 0 0 0 0 0 0 0 0 ...
    $ Month_Apr                   : int  0 0 0 0 0 0 0 0 0 0 ...
    $ Month_May                   : int  0 0 0 0 0 0 0 0 0 0 ...
    $ Month_Jun                   : int  0 0 0 0 0 0 0 0 0 0 ...
    $ Month_Jul                   : int  0 0 0 0 0 0 0 0 0 0 ...
    $ Month_Aug                   : int  0 0 0 0 0 0 0 0 0 0 ...
    $ Month_Sep                   : int  0 0 0 0 0 0 0 0 0 0 ...
    $ Month_Oct                   : int  0 0 0 0 0 0 0 0 0 0 ...
    $ Month_Nov                   : int  0 0 0 0 0 0 0 0 0 0 ...
    $ Month_Dec                   : int  0 0 0 0 0 0 0 0 0 0 ...
    $ OperatingSystems_1          : int  1 0 0 0 0 0 0 0 1 0 ...
    $ OperatingSystems_2          : int  0 1 0 0 1 1 1 1 0 1 ...
    $ OperatingSystems_3          : int  0 0 1 1 0 0 0 0 0 0 ...
    $ OperatingSystems_4          : int  0 0 0 0 0 0 0 0 0 0 ...
    $ OperatingSystems_5          : int  0 0 0 0 0 0 0 0 0 0 ...
    $ OperatingSystems_6          : int  0 0 0 0 0 0 0 0 0 0 ...
    $ OperatingSystems_7          : int  0 0 0 0 0 0 0 0 0 0 ...
    $ OperatingSystems_8          : int  0 0 0 0 0 0 0 0 0 0 ...
    $ Browser_1                   : int  1 0 0 0 0 0 0 0 1 0 ...
    $ Browser_2                   : int  0 1 1 0 1 0 1 0 0 0 ...
    $ Browser_3                   : int  0 0 0 1 0 0 0 0 0 0 ...
    $ Browser_4                   : int  0 0 0 0 0 1 0 1 0 0
```

```
test                                3051 obs. of 77 variables
    $ Administrative              : int  0 1 0 0 0 0 0 1 0 0 ...
    $ Administrative_Duration     : num  0 0 0 0 0 0 0 9 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 0 3 16 2 1 2 46 15 14 ...
    $ ProductRelated_Duration     : num  0 0 395 408 68 ...
    $ BounceRates                 : num  0.2 0.2 0 0.0187 0 ...
    $ ExitRates                   : num  0.2 0.2 0.0667 0.0258 0.1 ...
    $ PageValues                  : num  0 0 0 0 0 0 0 0 0 0 ...
    $ SpecialDay                  : num  0 0 0 0.4 0 0 0.8 0 0 0 ...
    $ Month_Jan                   : int  0 0 0 0 0 0 0 0 0 0 ...
    $ Month_Feb                   : int  1 1 1 1 1 1 1 1 1 1 ...
    $ Month_Mar                   : int  0 0 0 0 0 0 0 0 0 0 ...
    $ Month_Apr                   : int  0 0 0 0 0 0 0 0 0 0 ...
    $ Month_May                   : int  0 0 0 0 0 0 0 0 0 0 ...
    $ Month_Jun                   : int  0 0 0 0 0 0 0 0 0 0 ...
    $ Month_Jul                   : int  0 0 0 0 0 0 0 0 0 0 ...
    $ Month_Aug                   : int  0 0 0 0 0 0 0 0 0 0 ...
    $ Month_Sep                   : int  0 0 0 0 0 0 0 0 0 0 ...
    $ Month_Oct                   : int  0 0 0 0 0 0 0 0 0 0 ...
    $ Month_Nov                   : int  0 0 0 0 0 0 0 0 0 0 ...
    $ Month_Dec                   : int  0 0 0 0 0 0 0 0 0 0 ...
    $ OperatingSystems_1          : int  0 1 1 1 0 1 0 0 1 1 ...
    $ OperatingSystems_2          : int  0 0 0 0 0 0 1 1 0 0 ...
    $ OperatingSystems_3          : int  0 0 0 0 1 0 0 0 0 0 ...
    $ OperatingSystems_4          : int  1 0 0 0 0 0 0 0 0 0 ...
    $ OperatingSystems_5          : int  0 0 0 0 0 0 0 0 0 0 ...
    $ OperatingSystems_6          : int  0 0 0 0 0 0 0 0 0 0 ...
    $ OperatingSystems_7          : int  0 0 0 0 0 0 0 0 0 0 ...
    $ OperatingSystems_8          : int  0 0 0 0 0 0 0 0 0 0 ...
    $ Browser_1                   : int  1 0 1 1 0 1 0 0 1 1 ...
    $ Browser_2                   : int  0 1 0 0 1 0 0 1 0 0 ...
    $ Browser_3                   : int  0 0 0 0 0 0 0 0 0 0 ...
```

The C4.5 algorithm extended the ID3 tree construction algorithm by allowing numerical attributes, dealing with missing values and performing tree pruning after construction.

```
=== Summary ===

Correctly Classified Instances        8709               94.2839 %
Incorrectly Classified Instances       528                5.7161 %
Kappa statistic                          0.7611
Mean absolute error                      0.0987
Root mean squared error                  0.2222
Relative absolute error                 37.6929 %
Root relative squared error             61.4006 %
Total Number of Instances             9237
```

Evaluation metrics:

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction FALSE TRUE
##      FALSE  2476  205
##      TRUE    126  272
##
##                Accuracy : 0.8925
##                  95% CI : (0.881, 0.9032)
##     No Information Rate : 0.8451
##     P-Value [Acc > NIR] : 1.577e-14
##
##                   Kappa : 0.5597
##
##  Mcnemar's Test P-Value : 1.809e-05
##
##             Sensitivity : 0.9516
##             Specificity : 0.5702
##          Pos Pred Value : 0.9235
##          Neg Pred Value : 0.6834
##              Prevalence : 0.8451
##          Detection Rate : 0.8042
##    Detection Prevalence : 0.8707
##       Balanced Accuracy : 0.7609
##
##        'Positive' Class : FALSE
```

We tried some tree-based methods, first we apply C4.5 to build a decision tree and prune it, which achieves an accuracy, not as good as expected.

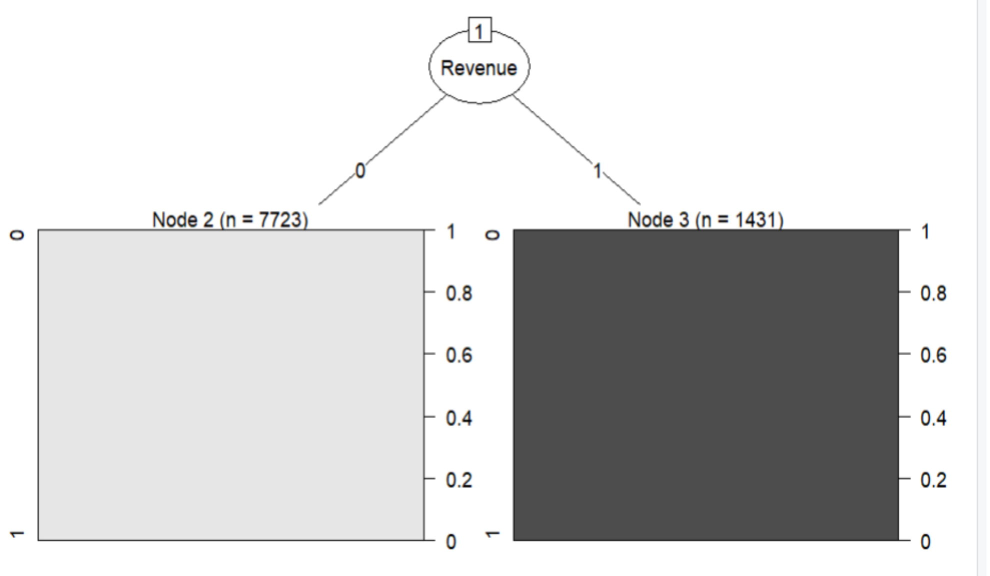
C5.0 Boosted Trees:

The C5.0 algorithm has become the industry standard for producing decision trees, because it does well for most types of problems directly out of the box. Compared to more advanced and sophisticated machine learning models (e.g. Neural Networks and Support Vector Machines), the decision trees under the C5.0 algorithm generally perform nearly as well but are much easier to understand and deploy.

C5.0 uses the concept of entropy for measuring purity. The entropy of a sample of data indicates how mixed the class values are; the minimum value of 00 indicates that the sample is completely homogenous, while 11 indicates the maximum amount of disorder. The definition of entropy can be specified as:

$$\text{Entropy}(S)=\sum_{i}= -p_i \log_2(pi)$$

One of the benefits of the C5.0 algorithm is that it is opinionated about pruning; it takes care of many of the decisions automatically using fairly reasonable defaults. Its overall strategy is to *post prune* the tree. It does this by first growing



a large tree that overfits the training data. Afterwards, nodes and branches that have little effect on the classification errors are removed.

Evaluation metrics:

```
> p_dtree<-predict(dtree,test)
> confusionMatrix(table(p_dtree,test$Revenue))
Confusion Matrix and Statistics


p_dtree    0    1
      0 2574    0
      1    0  477

               Accuracy : 1
                 95% CI : (0.9988, 1)
    No Information Rate : 0.8437
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 1

 Mcnemar's Test P-Value : NA

            Sensitivity : 1.0000
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 1.0000
             Prevalence : 0.8437
         Detection Rate : 0.8437
   Detection Prevalence : 0.8437
      Balanced Accuracy : 1.0000

       'Positive' Class : 0
```

However, C5.0 decision tree algorithm is inadequate for applying regression and predicting continuous values. We will continue to explore other prediction options.

Decision Tree Model:

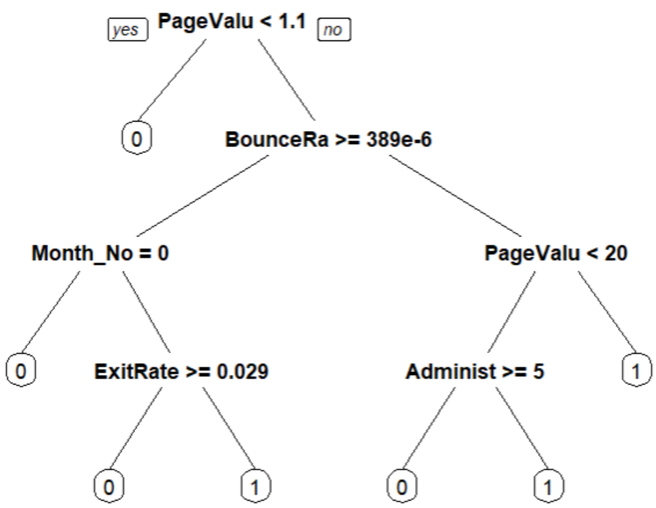


Fig. Plotting the decision tree model and determining its accuracy

```
decision_tree                List of  14
 $ frame                   :'data.frame':    13 obs. of  9 variables:
  ..$ var        : chr [1:13] "PageValues" "<leaf>" "BounceRates" "Month_Nov" ...
  ..$ n          : int [1:13] 9154 7146 2008 1100 732 368 89 279 908 346 ...
  ..$ wt         : num [1:13] 9154 7146 2008 1100 732 ...
  ..$ dev        : num [1:13] 1431 281 858 477 260 ...
  ..$ yval       : num [1:13] 1 1 2 1 1 2 1 2 2 2 ...
  ..$ complexity: num [1:13] 0.20405 0 0.10203 0.04612 0.00699 ...
  ..$ ncompete  : int [1:13] 4 0 4 4 0 4 0 0 4 4 ...
  ..$ nsurrogate: int [1:13] 5 0 5 5 0 5 0 0 5 5 ...
  ..$ yval2      : num [1:13, 1:6] 1 1 2 1 1 2 1 2 2 2 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:6] "" "" "" "" ...
 $ where                   : Named int [1:9154] 2 2 2 2 2 2 2 2 2 2 ...
  ..- attr(*, "names")= chr [1:9154] "1" "2" "3" "4" ...
 $ call                    : language rpart(formula = Revenue ~ ., data = train, method = "cl…
 $ terms                   :Classes 'terms', 'formula'  language Revenue ~ Administrative + A…
  .. ..- attr(*, "variables")= language list(Revenue, Administrative, Administrative_Dur…
  .. ..- attr(*, "factors")= int [1:77, 1:76] 0 1 0 0 0 0 0 0 0 0 ...
  .. .. ..- attr(*, "dimnames")=List of 2
  .. .. .. ..$ : chr [1:77] "Revenue" "Administrative" "Administrative_Duration" "Inform…
  .. .. .. ..$ : chr [1:76] "Administrative" "Administrative_Duration" "Informational" "…
  .. ..- attr(*, "term.labels")= chr [1:76] "Administrative" "Administrative_Duration" "…
  .. ..- attr(*, "order")= int [1:76] 1 1 1 1 1 1 1 1 1 1 ...
  .. ..- attr(*, "intercept")= int 1
  .. ..- attr(*, "response")= int 1
  .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
  .. ..- attr(*, "predvars")= language list(Revenue, Administrative, Administrative_Dura…
  .. ..- attr(*, "dataClasses")= Named chr [1:77] "factor" "numeric" "numeric" "numeric"…
  .. .. ..- attr(*, "names")= chr [1:77] "Revenue" "Administrative" "Administrative_Dura…
 $ cptable                 : num [1:6, 1:5] 0.2041 0.102 0.0461 0.0161 0.0157 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:6] "1" "2" "3" "4" ...
  .. ..$ : chr [1:5] "CP" "nsplit" "rel error" "xerror" ...
 $ method                  : chr "class"
 $ parms                   :List of 3
  ..$ prior: num [1:2(1d)] 0.844 0.156
```

Evaluation metrics:

Computing the predictive accuracy in the test set:

```
> pred_new <- predict(decision_tree, test, type = "class")
> mean(pred_new == test$Revenue)
[1] 0.8970829
>
```

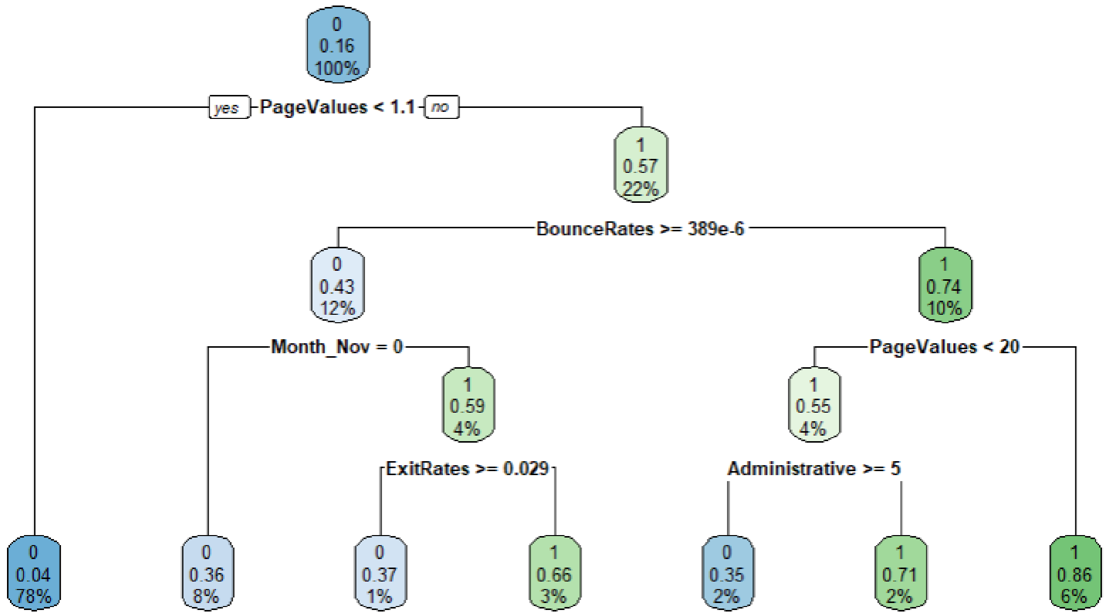


Fig. Result of rpart.plot() function

Random Forest:

On the other hand, random forest is based on constructing a forest, e.g. a set of diverse and accurate classification trees, using bagging resampling technique and combining the predictions of the individual trees using a voting strategy. The steps of the random forest construction algorithm are shown below:

Step 1: Given N instances in the original training set, create a subsample with bagging, (i.e.) choose N instances at random with replacement from the original data which constitutes the training set.

Step 2: Suppose that each instance is represented with M input variables in the original input space. A number m is specified, which is much less than M, such that at each node, m variables are selected at random out of the M and the best split on these m is used to split the node.

Step 3: According to the pre-determined stopping criteria, each tree is grown to the largest extent possible without pruning.

Step 4: Repeat this process until desired number of trees is obtained for the forest.

The random forest algorithm proved itself to be effective for many classification problems such as gene classification, remote sensing classification, land-cover classification, or image classification. In addition to its high accuracy, it has been shown that random forest has fewer number of hyper-parameters to be fine-tuned by the user when compared to state-of-art methods such as SVM. For these reasons, random forest is determined to be used as another classification algorithm in our purchasing intention prediction module. The hyper-parameters of the algorithm are the size of each bag, number of input variables used to determine the best split in each step which is referred to as m in the above-given algorithm, and number of trees in the forest. In our experiments, the size of each bag is set to N, m to $\log_2|M|$, and the number of trees in the forest to 100.

Training the model:

```
rf.fit                     Large randomForest.formula (19 elements,  29.9 MB)
  $ call           : language randomForest(formula = factor(Revenue) ~ ., data = train, mtry…
  $ type           : chr "classification"
  $ predicted      : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
   ..- attr(*, "names")= chr [1:9154] "1" "2" "3" "4" ...
  $ err.rate       : num [1:500, 1:3] 0.162 0.16 0.15 0.145 0.142 ...
   ..- attr(*, "dimnames")=List of 2
   .. ..$ : NULL
   .. ..$ : chr [1:3] "OOB" "0" "1"
  $ confusion      : num [1:2, 1:3] 7.60e+03 8.28e+02 1.24e+02 6.03e+02 1.61e-02 ...
   ..- attr(*, "dimnames")=List of 2
   .. ..$ : chr [1:2] "0" "1"
   .. ..$ : chr [1:3] "0" "1" "class.error"
  $ votes          : 'matrix' num [1:9154, 1:2] 1 1 0.988 0.994 0.989 ...
   ..- attr(*, "dimnames")=List of 2
   .. ..$ : chr [1:9154] "1" "2" "3" "4" ...
   .. ..$ : chr [1:2] "0" "1"
  $ oob.times      : num [1:9154] 162 191 172 177 179 193 174 174 207 186 ...
  $ classes        : chr [1:2] "0" "1"
  $ importance     : num [1:76, 1] 67.8 85.3 30.9 42.7 100.6 ...
   ..- attr(*, "dimnames")=List of 2
   .. ..$ : chr [1:76] "Administrative" "Administrative_Duration" "Informational" "Informati…
   .. ..$ : chr "MeanDecreaseGini"
  $ importanceSD   : NULL
  $ localImportance: NULL
  $ proximity      : NULL
  $ ntree          : num 500
  $ mtry           : num 5
  $ forest         :List of 14
    ..$ ndbigtree : int [1:500] 1451 1659 1477 1659 1523 1669 1565 1281 1603 1455
```

Evaluation metrics:

```
> confusionMatrix(table(rf.pred,test$Revenue))
Confusion Matrix and Statistics


rf.pred    0    1
      0 2531  287
      1   43  190

               Accuracy : 0.8918
                 95% CI : (0.8803, 0.9026)
    No Information Rate : 0.8437
    P-Value [Acc > NIR] : 9.851e-15

                  Kappa : 0.4821

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9833
            Specificity : 0.3983
         Pos Pred Value : 0.8982
         Neg Pred Value : 0.8155
             Prevalence : 0.8437
         Detection Rate : 0.8296
   Detection Prevalence : 0.9236
      Balanced Accuracy : 0.6908

       'Positive' Class : 0
```

Computing the prediction accuracy in the testing and training set:

```
 rf.pred      Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
 rf.pred2     Large factor (9154 elements,  623.2 kB)
   - attr(*, "names")= chr [1:9154] "1" "2" "3" "4" ...
```

```
> mean(rf.pred == test$Revenue)
[1] 0.8918387
> rf.pred2 <- predict(rf.fit, train)
> mean(rf.pred2 == train$Revenue)
[1] 0.9672274
```

However, tree-based model is effective for classification, we then try Random Forest to build a bagging of 100 different trees, number of variables for each tree is set to $\log_2|M|$, where M is the number of variables. The accuracy achieves 89.18%, it gives us momentum to try more other methods.

Support Vector Machine Algorithm:

Support Vector Machines (SVM) classifier, whose classification ability has been mostly accurate, is also included in our analysis. Although SVM does not have a straightforward implementation for online learning, an online passive-aggressive implementation can be used to dynamically update the SVM model with new examples if it achieves significantly higher accuracies than the other classifiers used in this study. SVM is a discriminant-based algorithm which aims to find the optimal separation boundary called hyperplane to discriminate the classes from each other. The closest samples to these hyperplanes are called support vectors, and the discriminant is represented as the weighted sum of this subset of samples which limits the complexity of the problem. The optimization problem to find an optimal separating hyperplane is defined as:

$$\min \frac{\$}{2} \|w\|^2 + C \sum_{,\$}^{k}) \text{ subject to } r^t (w^T x^t + w_{:}) \geq 1 -)$$

where w is a weight vector defining the discriminant, C the regularization parameter, ξ= (ξ1, ξ2, …., ξk) vector of slack variables, and rt the actual value of sample t. The slack variables are defined to tolerate the error on training set in order to avoid overfitting and so improve the generalization ability of the model. The regularization (cost) parameter, C, is a hyper-parameter of the algorithm which is used to control the complexity of the model that is fitted to the data. Higher values of C decrease the tolerance of the model on training set instances and hence may cause overfitting on the training set.

Although SVM is a linear classifier, it is capable of modeling non-linear interactions by mapping the original input space into a higher dimensional feature space using a kernel function. Thus, the linear model in the new space corresponds to a nonlinear model in the original space. In this study, linear and Radial Basis Function (RBF) kernels are used. The RBF is defined as:

$$K(x^t, x) = \exp\left[-\frac{\|x^t - x\|}{*s^2}\right]$$

where xt is the center and s defines the radius. As noted, we repeat train/validation split procedure for 100 times and report the average performance of each classifier on the validation sets. To avoid overfitting and report unbiased results, the values of hyper parameters, C and s, are optimized using grid search on a randomly selected single train/validation partition and the specified values are used for the rest of the partitions.

Linear SVM:

Training the model:

```
svm_fit              Large svm.formula (30 elements,  6.5 MB)
  $ call             : language svm(formula = as.factor(Revenue) ~ ., data = df_over, kernel =…
  $ type             : num 0
  $ kernel           : num 0
  $ cost             : num 1
  $ degree           : num 3
  $ gamma            : num 0.0132
  $ coef0            : num 0
  $ nu               : num 0.5
  $ epsilon          : num 0.1
  $ sparse           : logi FALSE
  $ scaled           : logi [1:76] FALSE FALSE FALSE FALSE FALSE FALSE ...
  $ x.scale          : NULL
  $ y.scale          : NULL
  $ nclasses         : int 2
  $ levels           : chr [1:2] "0" "1"
  $ tot.nSV          : int 6463
  $ nSV              : int [1:2] 3226 3237
  $ labels           : int [1:2] 1 2
  $ SV               : num [1:6463, 1:76] -0.403 0.506 2.928 0.506 0.203 ...
   ..- attr(*, "dimnames")=List of 2
   .. ..$ : chr [1:6463] "20" "37" "39" "42" ...
   .. ..$ : chr [1:76] "Administrative" "Administrative_Duration" "Informational" "Informati…
  $ index            : int [1:6463] 20 37 39 42 68 73 118 120 121 123 ...
  $ rho              : num -0.403
  $ compprob         : logi FALSE
  $ probA            : NULL
  $ probB            : NULL
  $ sigma            : NULL
```

```
pred                 Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
```

Evaluation metrics:

```
> linear_SVM <- confusionMatrix(pred, factor(test_data$Revenue))
> print(linear_SVM)
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 2764  128
         1  326  445

               Accuracy : 0.8761
                 95% CI : (0.8649, 0.8866)
    No Information Rate : 0.8436
    P-Value [Acc > NIR] : 1.414e-08

                  Kappa : 0.5883

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.8945
            Specificity : 0.7766
         Pos Pred Value : 0.9557
         Neg Pred Value : 0.5772
             Prevalence : 0.8436
         Detection Rate : 0.7546
   Detection Prevalence : 0.7895
      Balanced Accuracy : 0.8356

       'Positive' Class : 0
```

Hence, we tried Linear Support Vector Machine by default, which achieves accuracy of 87.61%, higher than NBC, and no significant improvement by tuning.

Radial SVM:

```
svm_fit2                 Large svm.formula (30 elements,  6.5 MB)
    $ call           : language svm(formula = as.factor(Revenue) ~ ., data = df_over, kernel =…
    $ type           : num 0
    $ kernel         : num 2
    $ cost           : num 1
    $ degree         : num 3
    $ gamma          : num 0.0132
    $ coef0          : num 0
    $ nu             : num 0.5
    $ epsilon        : num 0.1
    $ sparse         : logi FALSE
    $ scaled         : logi [1:76] FALSE FALSE FALSE FALSE FALSE FALSE ...
    $ x.scale        : NULL
    $ y.scale        : NULL
    $ nclasses       : int 2
    $ levels         : chr [1:2] "0" "1"
    $ tot.nSV        : int 6422
    $ nSV            : int [1:2] 3223 3199
    $ labels         : int [1:2] 1 2
    $ SV             : num [1:6422, 1:76] -0.403 0.506 2.928 0.506 -0.705 ...
     ..- attr(*, "dimnames")=List of 2
     .. ..$ : chr [1:6422] "20" "37" "39" "42" ...
     .. ..$ : chr [1:76] "Administrative" "Administrative_Duration" "Informational" "Informati…
    $ index          : int [1:6422] 20 37 39 42 64 68 70 73 103 109 ...
    $ rho            : num -1.54
    $ compprob       : logi FALSE
    $ probA          : NULL
    $ probB          : NULL
    $ sigma          : NULL
    $ coefs          : num [1:6422, 1] 1 1 1 1 1 ...
    $ na.action      : NULL
```

Prediction result:

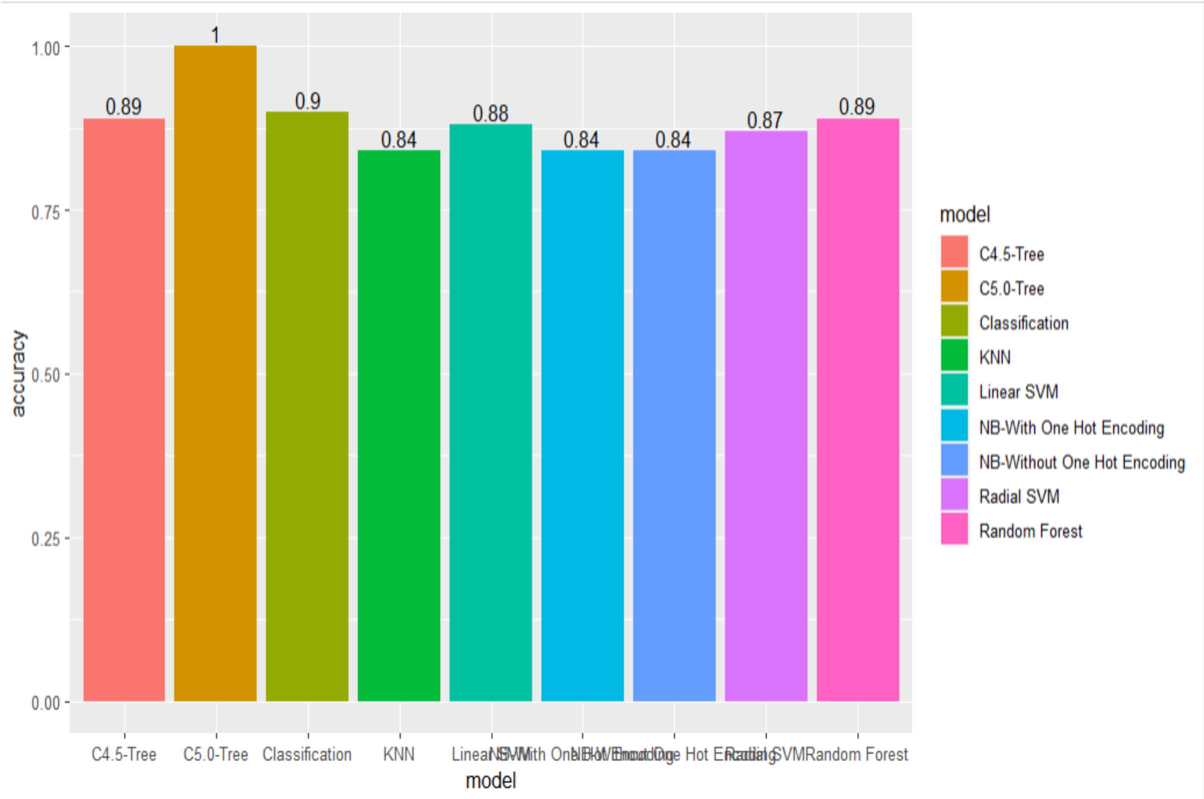
```
pred_radial        Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
```

Evaluation Metrics:

```
> #Confusion Matrix and Metrics of RBF SVM
> print("Radial SVM")
[1] "Radial SVM"
> radial_SVM <- confusionMatrix(pred_radial, factor(test_data$Revenue))
> print(radial_SVM)
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 2757  130
         1  333  443

               Accuracy : 0.8736
                 95% CI : (0.8624, 0.8842)
    No Information Rate : 0.8436
    P-Value [Acc > NIR] : 1.554e-07

                  Kappa : 0.5815

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.8922
            Specificity : 0.7731
         Pos Pred Value : 0.9550
         Neg Pred Value : 0.5709
             Prevalence : 0.8436
         Detection Rate : 0.7527
   Detection Prevalence : 0.7882
      Balanced Accuracy : 0.8327

       'Positive' Class : 0
```

Further, Radial-basis-function SVM is applied, by default it achieves accuracy of 87.36% which is approximately the same as that of Linear SVM, while after tuning it achieves surprisingly little lower accuracy, its specificity more or less remains the same, since to detect positive ones is more important, the tuned model is said to have better performance.

Comparison of accuracies between different models:



Therefore, with evidence from the bar chart, we can affirm that the Random Forest model performs the best among all the algorithms we have implemented, we further explore whether we could even improve the performance more.

IMPROVE ACCURACY

Evaluation metrics of Random Forest before optimization:

```
> confusionMatrix(table(rf.pred,test$Revenue))
Confusion Matrix and Statistics


rf.pred    0    1
      0 2531  287
      1   43  190

               Accuracy : 0.8918
                 95% CI : (0.8803, 0.9026)
    No Information Rate : 0.8437
    P-Value [Acc > NIR] : 9.851e-15

                  Kappa : 0.4821

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9833
            Specificity : 0.3983
         Pos Pred Value : 0.8982
         Neg Pred Value : 0.8155
             Prevalence : 0.8437
         Detection Rate : 0.8296
   Detection Prevalence : 0.9236
      Balanced Accuracy : 0.6908

       'Positive' Class : 0
```

Optimization of Random Forest model using target feature selection:

We tried Recursive Feature Elimination to build Random Forest model with all possible subsets of features. The result shows that PageValues is the most important attribute, consistent with the result of varImPlot. Besides, BounceRates is important as well, but since it is highly correlated with ExitRates, it is reasonable to leave it out.

Function “varImpPlot” measures the importance of features by 2 metrics: MeanDecreaseAccuracy and MeanDecreaseGini, MDA means the decrease of accuracy after exclusion or permutation of a single variable, MDG means the decrease of node impurity.

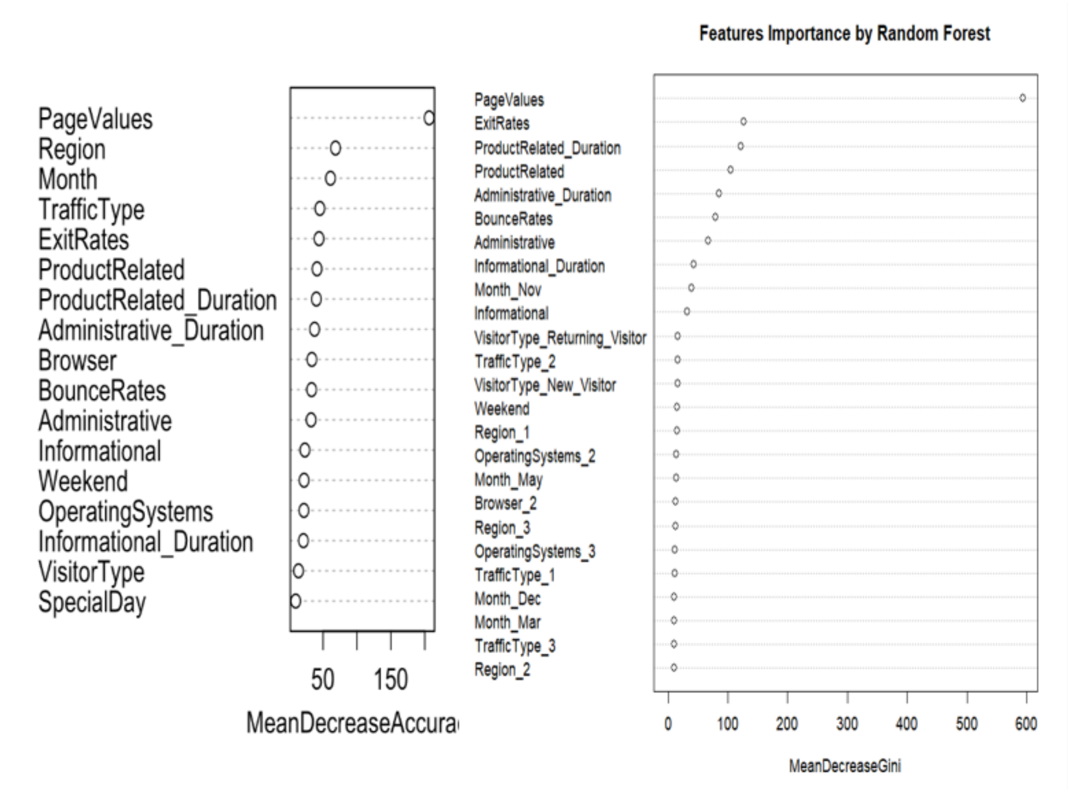


Fig. Target feature selection measuring MDA and MDG

The result of varImpPlot shows that PageValues is definitely the most important one, attributes like Month, ExitRates, ProductRelated/ProductRelated_Duration are among the most important ones as well. Both measurements show that “SpecialDay” is not an important feature, which is consistent with our result of exploratory analysis.

Next, we bin the continuous variables for use of Mutual Information and minimum Redundancy Maximum Relevance Feature Selection

```
continous_cols              14414 obs. of 7 variables
  $ Administrative_Duration: num  0 0 0 0 0 0 0 0 0 0 ...
  $ Informational_Duration : num  0 0 0 0 0 0 0 0 0 0 ...
  $ 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 0.8 0.4 0.4 ...
```

In the process, we tried scaling the continuous columns:

```
standardized_cols          14414 obs. of 7 variables
  $ Administrative_Duration: num  -0.51 -0.51 -0.51 -0.51 -0.51 ...
  $ Informational_Duration : num  -0.289 -0.289 -0.289 -0.289 -0.289 ...
  $ ProductRelated_Duration: num  -0.702 -0.672 -0.702 -0.701 -0.406 ...
  $ BounceRates            : num  5.202 -0.381 5.202 1.015 0.177 ...
  $ ExitRates              : num  4.423 1.791 4.423 2.844 0.474 ...
  $ PageValues             : num  -0.504 -0.504 -0.504 -0.504 -0.504 ...
  $ SpecialDay             : num  -0.26 -0.26 -0.26 -0.26 -0.26 ...
```

```
binned_cols                14414 obs. of 7 variables
  $ Administrative_Duration: Factor w/ 9 levels "-4","-3","-2",..: 4 4 4 4 4 4 4 4 4 4 ...
  $ Informational_Duration : Factor w/ 9 levels "-4","-3","-2",..: 5 5 5 5 5 5 5 5 5 5 ...
  $ ProductRelated_Duration: Factor w/ 9 levels "-4","-3","-2",..: 4 4 4 4 5 4 4 4 5 4 ...
  $ BounceRates            : Factor w/ 9 levels "-4","-3","-2",..: 9 5 9 6 5 5 9 5 5 5 ...
  $ ExitRates              : Factor w/ 9 levels "-4","-3","-2",..: 9 7 9 8 5 5 9 7 5 5 ...
  $ PageValues             : Factor w/ 9 levels "-4","-3","-2",..: 4 4 4 4 4 4 4 4 4 4 ...
  $ SpecialDay             : Factor w/ 9 levels "-4","-3","-2",..: 5 5 5 5 5 5 7 9 7 7 ...
```

```
binned                     14414 obs. of 18 variables
  $ Administrative         : int  0 0 0 0 0 0 0 0 0 0 ...
  $ Administrative_Duration: Factor w/ 9 levels "-4","-3","-2",..: 4 4 4 4 4 4 4 4 4 4 ...
  $ Informational          : int  0 0 0 0 0 0 0 0 0 0 ...
  $ Informational_Duration : Factor w/ 9 levels "-4","-3","-2",..: 5 5 5 5 5 5 5 5 5 5 ...
  $ ProductRelated         : int  1 2 1 2 10 19 1 2 3 16 ...
  $ ProductRelated_Duration: Factor w/ 9 levels "-4","-3","-2",..: 4 4 4 4 5 4 4 4 5 4 ...
  $ BounceRates            : Factor w/ 9 levels "-4","-3","-2",..: 9 5 9 6 5 5 9 5 5 5 ...
  $ ExitRates              : Factor w/ 9 levels "-4","-3","-2",..: 9 7 9 8 5 5 9 7 5 5 ...
  $ PageValues             : Factor w/ 9 levels "-4","-3","-2",..: 4 4 4 4 4 4 4 4 4 4 ...
  $ SpecialDay             : Factor w/ 9 levels "-4","-3","-2",..: 5 5 5 5 5 5 7 9 7 7 ...
  $ Month                  : Factor w/ 12 levels "Jan","Feb","Mar",..: 2 2 2 2 2 2 2 2 2 2 ...
  $ OperatingSystems       : Factor w/ 8 levels "1","2","3","4",..: 1 2 4 3 3 2 2 2 2 1 ...
  $ Browser                : Factor w/ 13 levels "1","2","3","4",..: 1 2 1 2 3 2 4 2 4 1 ...
  $ Region                 : Factor w/ 9 levels "1","2","3","4",..: 1 1 9 2 1 1 3 2 1 4 ...
  $ TrafficType            : Factor w/ 20 levels "1","2","3","4",..: 1 2 3 4 4 3 3 3 2 3 ...
  $ VisitorType            : Factor w/ 3 levels "New_Visitor",..: 3 3 3 3 3 3 3 3 3 3 ...
  $ Weekend                : int  0 0 0 0 1 0 0 0 0 0 ...
  $ Revenue                : int  0 0 0 0 0 0 0 0 0 0 ...
```

Mutual Information measures:

Mutual Information measures how much information the presence/absence a term contributes to making the correct classification on, similar to MDA of varImpPlot() function.

Using Mutual Information (MI) filter:

```
> MI
       names.binned..1.17.          MI
9               PageValues 0.276361565
5           ProductRelated 0.076401939
8                ExitRates 0.063842132
6  ProductRelated_Duration 0.041356751
7              BounceRates 0.037556789
1           Administrative 0.032782313
15             TrafficType 0.032461322
11                   Month 0.030685936
2  Administrative_Duration 0.028119556
3            Informational 0.012665242
10              SpecialDay 0.011158448
16             VisitorType 0.009516286
12        OperatingSystems 0.007636185
4   Informational_Duration 0.007529111
13                 Browser 0.003882431
17                 Weekend 0.001204713
14                  Region 0.001105638
>
```

The results show that PageValues, ProductRelated/ProductRelated_Duration and ExitRates are still among the most important. The is one potential problem with the method that it ignores the relevance between variables, therefore features like ProductRelated, ProductRelated_Duration are ranked top at the same time.

We further use Minimum redundancy feature selection to gain more insights, which takes into account the relevance between features and rank the other relevant ones on the tail.

Minimum redundancy feature selection:

mMRM filter:

```
> #mMRM filter
> score = MRMR(binned[1:17],binned$Revenue,17)$score
> score = as.data.frame(score)
> score
                              score
PageValues                0.2763615654
Month                     0.0141966784
BounceRates               0.0113597387
Weekend                  -0.0014656640
Informational_Duration   -0.0011793092
VisitorType              -0.0014456664
Administrative_Duration   0.0002425508
ProductRelated_Duration  -0.0042024825
OperatingSystems         -0.0053002038
ExitRates                -0.0041852420
SpecialDay               -0.0065799737
Region                   -0.0078916886
TrafficType              -0.0198952288
Informational            -0.0253803182
Browser                  -0.0461397454
Administrative           -0.0597086865
ProductRelated           -0.1315273233
```

The result shows that PageValues, Month, ExitRates and ProductRelated_Duration are still among the most important ones, it is different than the former results that features like ProductRelated have been listed on the tail, after selecting BounceRates, the ExitRates is ranked way behind than its former position.

Hence, after consideration of the above results, we decide to select the following 12 features to re-train our model, namely:

"Administrative_Duration", "Informational_Duration", "ProductRelated_Duration", "ExitRates", "PageValues", "Month", "OperatingSystems", "Browser", "Region", "TrafficType", "VisitorType", "Weekend"

FINALIZE THE MODEL

Training the Random Forest model after feature selection:

```
rf_train              List of  19
  $ call              : language randomForest(formula = as.factor(df_new_over$Revenue) ~ ., dat…
  $ type              : chr "classification"
  $ predicted         : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
   ..- attr(*, "names")= chr [1:14414] "1" "2" "3" "4" ...
  $ err.rate          : num [1:100, 1:3] 0.063 0.0623 0.0588 0.0573 0.0553 ...
   ..- attr(*, "dimnames")=List of 2
   .. ..$ : NULL
   .. ..$ : chr [1:3] "OOB" "0" "1"
  $ confusion         : num [1:2, 1:3] 6.71e+03 1.00e+01 4.95e+02 7.20e+03 6.87e-02 ...
   ..- attr(*, "dimnames")=List of 2
   .. ..$ : chr [1:2] "0" "1"
   .. ..$ : chr [1:3] "0" "1" "class.error"
  $ votes             : 'matrix' num [1:14414, 1:2] 1 1 1 1 1 1 1 1 1 1 ...
   ..- attr(*, "dimnames")=List of 2
   .. ..$ : chr [1:14414] "1" "2" "3" "4" ...
   .. ..$ : chr [1:2] "0" "1"
  $ oob.times         : num [1:14414] 38 34 31 35 33 33 36 36 33 36 ...
  $ classes           : chr [1:2] "0" "1"
  $ importance        : num [1:12, 1:4] 0.0109 0.003 0.00755 0.00348 0.21571 ...
```

Evaluation metrics after Random Forest optimization:

```
> print(Optimized_RF)
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 2866  179
         1  224  394

               Accuracy : 0.89
                 95% CI : (0.8794, 0.8999)
    No Information Rate : 0.8436
    P-Value [Acc > NIR] : 3.709e-16

                  Kappa : 0.5961

 Mcnemar's Test P-Value : 0.02839

            Sensitivity : 0.9275
            Specificity : 0.6876
         Pos Pred Value : 0.9412
         Neg Pred Value : 0.6375
             Prevalence : 0.8436
         Detection Rate : 0.7824
   Detection Prevalence : 0.8313
      Balanced Accuracy : 0.8076

       'Positive' Class : 0
```

Although, the result has not changed much, it is still worth selecting subset of features as it simplifies the model and requires less time to train. The noticeable feature is the increase in the specificity metric before and after optimization.

COMPLETE PROGRAM/ CODE

```
setwd("C:/Users/Vish/OneDrive/Desktop/BA w R/Project presentation")
rm(list=ls())

#import libraries
install.packages("ggplot2")
install.packages("gridExtra")
install.packages("ggdensity")
install.packages("ggcorr")
install.packages("dummies")
install.packages("C50")
install.packages("forecast")

library(ggplot)
library(ggplot2)
library(ggdensity)
library(gridExtra)
library(GGally)
library(caret)
library(data.table)
library(ggpubr)
library(ROSE)
library(class)
library(tree)
library(dtree)
library(randomForest)
```

```
library(mltools)
library(rsample)
library(e1071)
library(pheatmap)
library(keras)
library(dummies)
library(mlbench)
library(reticulate)
library(dplyr)
library(infotheo)
library(praznik)
library(ggpubr)
library(corrgram)
library(ggcorr)
library(klaR)
library(caret)
library(tidyverse)
library(data.table)
library(tidymodels)
library(partykit)
library(rpart)
library(rpart.plot)
library(e1071)
library(C50)
library(forecast)
require(randomForest)
library(RWeka)
```

```r
#import csv file
df <- read.csv(file = "online_shoppers_intention.csv")
View(df)

#descriptive statistics
str(df)
head(df)
summary(df)

#removing duplicates
df_duplicate <- nrow(df[duplicated(df),])
df <- df[!duplicated(df),]
str(df)

#identification of missing values
which(is.na(df))

#Renaming June to Jun for convenience of plotting
df$Month <- as.character(df$Month)
df$Month[df$Month == "June"] <- "Jun"
df$Month <- as.factor(df$Month)
df$Month = factor(df$Month, levels = month.abb)
```

```
#*******************************************************************************

#EXPLORATORY DATA ANALYSIS


#Administrative pages: number of pages visited

plot1 <- ggplot(df, aes(x=1, y=Administrative)) + geom_violin() + geom_violin(trim=FALSE, fill='#E69F00', color='gray') + coord_flip() + labs(x = " ") + labs(y = "Number of Administrative pages visited") + theme(axis.text.y = element_blank(), axis.ticks = element_blank())

#Administrative pages: total time spent

plot2 <- ggplot(df, aes(x=1, y=Administrative_Duration)) + geom_violin() + geom_violin(trim=FALSE, fill='#E69F00', color='gray') + coord_flip() + labs(x = " ") + labs(y = "Total time spent in Administrative pages") + theme(axis.text.y = element_blank(), axis.ticks = element_blank())


#Informational pages: number of pages visited

plot3 <- ggplot(df, aes(x=1, y=Informational)) + geom_violin() + geom_violin(trim=FALSE, fill='#56B4E9', color='gray') + coord_flip() + labs(x = " ") + labs(y = "Number of Informational pages visited") + theme(axis.text.y = element_blank(), axis.ticks = element_blank())

#Informational pages: total time spent

plot4 <- ggplot(df, aes(x=1, y=Informational_Duration)) + geom_violin() + geom_violin(trim=FALSE, fill='#56B4E9', color='gray') + coord_flip() + labs(x = " ") + labs(y = "Total time spent in Informational pages") + theme(axis.text.y = element_blank(), axis.ticks = element_blank())


#Product related pages: number of pages visited

plot5 <- ggplot(df, aes(x=1, y=ProductRelated)) + geom_violin() + geom_violin(trim=FALSE, fill='#FF9999', color='gray') + coord_flip() + labs(x = " ") + labs(y = "Number of ProductRelated pages visited") + theme(axis.text.y = element_blank(), axis.ticks = element_blank())
```

```
#Product related: total time spent

plot6 <- ggplot(df, aes(x=1, y=ProductRelated_Duration)) + geom_violin() +
geom_violin(trim=FALSE, fill='#FF9999', color='gray') + coord_flip() + labs(x
= " ") + labs(y = "Total time spent in ProductRelated pages") +
theme(axis.text.y = element_blank(), axis.ticks = element_blank())


grid.arrange(plot1, plot2, plot3, plot4, plot5, plot6, nrow = 3, ncol = 2)


#Side-by-side comparison charts

plot1 <- ggplot(df, aes(x=Revenue, y=Administrative)) + geom_violin() +
geom_violin(trim=FALSE, fill='#E69F00', color='gray') + labs(x =
"Administrative") + labs(y = " ") + theme(axis.text.y = element_blank(),
axis.ticks = element_blank())

plot4 <- ggplot(df, aes(x=Revenue, y=Administrative_Duration)) +
geom_violin() + geom_violin(trim=FALSE, fill='#E69F00', color='gray') +
labs(x = "Administrative_Duration") + labs(y = " ") + theme(axis.text.y =
element_blank(), axis.ticks = element_blank())

plot2 <- ggplot(df, aes(x=Revenue, y=Informational)) + geom_violin() +
geom_violin(trim=FALSE, fill='#56B4E9', color='gray') + labs(x =
"Informational") + labs(y = " ") + theme(axis.text.y = element_blank(),
axis.ticks = element_blank())

plot5 <- ggplot(df, aes(x=Revenue, y=Informational_Duration)) +
geom_violin() +  geom_violin(trim=FALSE, fill='#56B4E9', color='gray') +
labs(x = "Informational_Duration") + labs(y = " ") + theme(axis.text.y =
element_blank(), axis.ticks = element_blank())

plot3 <- ggplot(df, aes(x=Revenue, y=ProductRelated)) + geom_violin() +
geom_violin(trim=FALSE, fill='#FF9999', color='gray') + labs(x =
"ProductRelated") + labs(y = " ") + theme(axis.text.y = element_blank(),
axis.ticks = element_blank())

plot6 <- ggplot(df, aes(x=Revenue, y=ProductRelated_Duration)) +
geom_violin() + geom_violin(trim=FALSE, fill='#FF9999', color='gray') +
labs(x = "ProductRelated_Duration") + labs(y = " ") + theme(axis.text.y =
element_blank(), axis.ticks = element_blank())

grid.arrange(plot1, plot2, plot3, plot4, plot5, plot6, nrow = 2, ncol = 3)
```

```
#BounceRates, ExitRates and PageValues

plot1 <- ggdensity(df, x = "BounceRates", fill = "thistle2", color = "thistle2", add = "median", rug = TRUE) + labs(y = " ")

plot2 <- ggdensity(df, x = "ExitRates", fill = "skyblue1", color = "skyblue1", add = "median", rug = TRUE) + labs(y = " ")

plot3 <- ggdensity(df, x = "PageValues", fill = "sienna3", color = "sienna3", add = "median", rug = TRUE) + labs(y = " ")

grid.arrange(plot1, plot2, plot3, nrow = 3)



plot1 <- ggplot(df, aes(x=BounceRates, fill=Revenue)) + geom_density(alpha=0.4) + labs(y = " ")

plot2 <- ggplot(df, aes(x=ExitRates, fill=Revenue)) + geom_density(alpha=0.4) + labs(y = " ")

plot3 <- ggplot(df, aes(x=PageValues, fill=Revenue)) + geom_density(alpha=0.4) + labs(y = " ")

grid.arrange(plot1, plot2, plot3, nrow = 3)


#Special and non-special days

plot1 <- ggplot(df, aes(x = factor(1), y = SpecialDay)) + geom_boxplot(width = 0.4, fill = "white") + geom_jitter(color = "deepskyblue4", width = 0.1, size = 1, alpha=0.4) + labs(x = "Special Day") + labs(y = "Closeness") + theme(axis.text.x = element_blank(), axis.ticks = element_blank())

plot2 <- ggplot(df, aes(x = Revenue, y = SpecialDay)) + geom_boxplot(width = 0.4, fill = "white") + geom_jitter(color = "deepskyblue4", width = 0.2, size = 1, alpha=0.4) + labs(x = "Special Day") + labs(y = " ") + theme(axis.ticks = element_blank())

grid.arrange(plot1, plot2, ncol = 2)
```

```
#Month-wise distribution

plot <- ggplot(data.frame(df), aes(Month, fill=Revenue)) + geom_bar() + labs(x = "Month") + labs(y = " ")

plot


#Categorization based upon OS, browser, region, traffic type, weekend and visitor type

plot1 <- ggplot(data.frame(df), aes(OperatingSystems, fill=Revenue)) + geom_bar() + labs(x = "Operating Systems") + labs(y = " ") + scale_x_continuous(breaks = 1:8)

plot2 <- ggplot(data.frame(df), aes(Browser, fill=Revenue)) + geom_bar() + labs(x = "Browser") + labs(y = " ") + scale_x_continuous(breaks = 1:13)

plot3 <- ggplot(data.frame(df), aes(Region, fill=Revenue)) + geom_bar() + labs(x = "Region") + labs(y = " ") + scale_x_continuous(breaks = 1:9)

plot4 <- ggplot(data.frame(df), aes(TrafficType, fill=Revenue)) + geom_bar() + labs(x = "Traffic Type") + labs(y = " ")

plot5 <- ggplot(data.frame(df), aes(Weekend, fill=Revenue)) + geom_bar() + labs(x = "Weekend") + labs(y = " ")

plot6 <- ggplot(data.frame(df), aes(VisitorType, fill=Revenue)) + geom_bar() + labs(x = "Visitor Type") + labs(y = " ") + scale_x_discrete(labels = c("New_Visitor" = "New", "Other" = "Other", "Returning_Visitor" = "Return"))

grid.arrange(plot1, plot2, plot3, plot4, plot5, plot6, nrow = 3, ncol = 2)


#Target feature distribution

plot <- ggplot(data.frame(df$Revenue), aes(x=df$Revenue)) + geom_bar() + labs(x = "Target Feature Distribution")

plot
```

```
#Correlation between different sets of variables

corr_map <- ggcorr(df[, 1:10], method=c("everything", "pearson"),
label=TRUE, hjust = .90, size = 3, layout.exp = 2)

corr_map


#***************************************************************
****************


#DATA PRE-PROCESSING


#Transforming categorical attributes into factor types
df <- df %>%
  mutate(OperatingSystems = as.factor(OperatingSystems),
         Browser = as.factor(Browser),
         Region = as.factor(Region),
         TrafficType = as.factor(TrafficType),
         VisitorType = as.factor(VisitorType),
         Weekend = as.integer(Weekend),
         Revenue = as.integer(Revenue)
  )

#One-hot encoding to save original copy
df_new <- df
df$Revenue <- as.factor(df$Revenue)
print("Original dataset")

print(str(df_new))
```

```
#Peforming one hot encoding on all columns except Revenue
revenueData <- df[,18]; revenueData

encoded_df <- one_hot(as.data.table(df[,-18]))
df <- cbind(encoded_df, df[,18]);
colnames(df)[colnames(df)=="V2"] = "Revenue"
#df$Revenue <- as.factor(df$Revenue)

print("After one-hot encoding")

print(str(df))

#Split training and testing data.

split_df_new <- initial_split(df_new, prop = .7, strata = "Revenue")
train_df_new <- training(split_df_new)
test_df_new <- testing(split_df_new)

print("Original dataset")

table(train_df_new$Revenue) %>% prop.table()
table(test_df_new$Revenue) %>% prop.table()

split <- initial_split(df, prop = .7, strata = "Revenue")
train_data <- training(split)
test_data <- testing(split)
```

```
print("After one-hot encoding")

table(train_data$Revenue) %>% prop.table()
table(test_data$Revenue) %>% prop.table()


#Preprocess the continuous attributes by splitting from categorical ones and binding
train_numerical <- train_data[,1:10]
train_categorical <- train_data[,11:77]
test_numerical <- test_data[,1:10]
test_categorical = test_data[,11:77]

#Utilization of scaling function
train_scaled = scale(train_numerical)
test_scaled = scale(test_numerical, center=attr(train_scaled, "scaled:center"), scale=attr(train_scaled, "scaled:scale"))

#Column binding
train_data <- cbind(train_scaled, train_categorical)
test_data <- cbind(test_scaled, test_categorical)

#Oversampling to overcome imbalance in dataset
N_df_new = 2*length(which(train_df_new$Revenue == 0))
df_new_over <- ovun.sample(Revenue~.,data = train_df_new, method= 'over', N = N_df_new, seed = 2020)$data

N = 2*length(which(train_data$Revenue == 0))
```

```
df_over <- ovun.sample(Revenue~.,data = train_data, method= 'over', N = N, seed = 2020)$data


#Splitting features and target

features_df_new <- setdiff(names(train_df_new), "Revenue")

features <- setdiff(names(train_data), "Revenue")


#*****************************************************************************


#DATA MODELING


#PREDICTION USING DIFFERENT ALGORITHMS

#(A) NAIVE BAYES CLASSFIER


x_df_new <- train_df_new[, features_df_new]

y_df_new <- train_df_new$Revenue

x <- train_data[, ..features]

y <- train_data$Revenue


train_control <- trainControl(

  method = "cv",

  number = 10

)


nb.ml_df_new <- caret::train(

  x = x_df_new,

  y = as.factor(y_df_new),
```

```
  method = "nb",
  trControl = train_control
)

nb.ml <- caret::train(
  x = x,
  y = y,
  method = "nb",
  trControl = train_control
)

print("Without one-hot encoding")
print(confusionMatrix(nb.ml_df_new))
#Accuracy
a1 <- 0.8404

print("With one-hot encoding")
print(confusionMatrix(nb.ml))
#Accuracy
a2 <- 0.8436

#*****************************************************************
****************

#(B) K-NEAREST NEIGHBOUR

#Train the model and predict
knn_model <- knn(df_over[, 1:76], test_data[, 1:76], df_over$Revenue)
```

```
#Confusion Matrix and Metrics
print("Default k-NN")

CM_knn_default <- confusionMatrix(knn_model, factor(test_data$Revenue))
print(CM_knn_default)

#Visualize accuracies of different k
knn_model <- NULL
errors <- NULL

for (i in 1:30) {
  knn_model <- knn(df_over[, 1:76], test_data[, 1:76], df_over$Revenue, k = i)
  errors[i] <- mean(knn_model != test_data$Revenue)
}

knn.error <- as.data.frame(cbind(k=1:30, errors))

ggplot(knn.error, aes(k, errors)) +
  geom_point() +
  geom_line() +
  scale_x_continuous(breaks = 1:30) +
  theme_bw() +
  xlab("Value of K") +
  ylab("Error")
```

```
#Result of the best-performance model
k_nn <- knn(df_over[, 1:76], test_data[, 1:76], df_over$Revenue, k=1)
print("Best-performance k-NN")

CM_knn_best <- confusionMatrix(k_nn, factor(test_data$Revenue))
print(CM_knn_best)
#Accuracy
a3 <- 0.8392

#****************************************************************************

#(C)TREE BASED METHODS

#Train the model

set.seed(100)

df1=df

df1$Revenue<- as.factor(df1$Revenue)
df1$Weekend<- as.factor(df1$Weekend)

index <- createDataPartition(df1$Revenue, p=0.75, list=FALSE)
train <-df1[ index,]
test <- df1[-index,]

#Decision Tree Model
```

```
decision_tree <- rpart(Revenue ~ . , method='class', data= train)
prp(decision_tree)
rpart.plot(decision_tree)

### compute the predictive accuracy in the test set

pred_new <- predict(decision_tree, test, type = "class")
mean(pred_new == test$Revenue)

#Accuracy
a4 <- 0.8970

# C4.5 Decision trees

fit<-J48(Revenue ~.,data=train)
summary(fit)

p_tree<-predict(fit,test[,1:17])
confusionMatrix(p_tree,test$Revenue)

#Accuracy
a5 <- 0.8925

# C5.0 Boosted trees

dtree<-C5.0(train,train$Revenue)
plot(dtree)
```

```
p_dtree<-predict(dtree,test)
confusionMatrix(table(p_dtree,test$Revenue))

#Accuracy
a6 <- 1

#Random Forest
#Train the model
n<-length(names(df_new_over))
m = ceiling(log2(n))
rf.fit <- randomForest(factor(Revenue)~., data = train, mtry = m)

rf.pred <- predict(rf.fit, test)
confusionMatrix(table(rf.pred,test$Revenue))

head(rf.pred)

#Compute the prediction accuracy in the testing set
mean(rf.pred == test$Revenue)

#Accuracy
a7 <- 0.8918

#Compute the prediction accuracy in the training set

rf.pred2 <- predict(rf.fit, train)
```

```
mean(rf.pred2 == train$Revenue)

#***************************************************************
****************

#(D) SVM Algorithm

#Linear SVM

#Train the model
svm_fit = svm(as.factor(Revenue)~., data=df_over, kernel = "linear", scale =
FALSE)

#Predict
pred <- predict(svm_fit, newdata = test_data)

#Confusion Matrix and Metrics of Linear SVM

print("Linear SVM")

linear_SVM <- confusionMatrix(pred, factor(test_data$Revenue))
print(linear_SVM)

#Accuracy
a8 <- 0.8761
```

```
#Radial SVM

#Train the model
svm_fit2 = svm(as.factor(Revenue)~., data=df_over, kernel = "radial", scale =
FALSE)

#Predict
pred_radial <- predict(svm_fit2, newdata = test_data)

#Confusion Matrix and Metrics of RBF SVM
print("Radial SVM")

radial_SVM <- confusionMatrix(pred_radial, factor(test_data$Revenue))
print(radial_SVM)

#Accuracy
a9 <- 0.8736

#*****************************************************************************
#Target Feature Selection

varImpPlot(rf.fit, sort = TRUE, n.var = 25, main = 'Features Importance by
Random Forest')

#*****************************************************************************
```

```
#Comparison of accuracies between different models


X<-c("NB-Without One Hot Encoding","NB-With One Hot Encoding","KNN","Classification","C4.5-Tree","C5.0-Tree","Random Forest","Linear SVM","Radial SVM")

Y<-round(c(a1,a2,a3,a4,a5,a6,a7,a8,a9),2)


X_name <- "model"

Y_name <- "accuracy"


df <- data.frame(X,Y)

names(df) <- c(X_name,Y_name)


ggplot(df,aes(x=model,y=accuracy,fill=model))+geom_bar(stat = "identity") + geom_text(aes(label=accuracy),position=position_dodge(width=0.9), vjust=-0.25)


#*****************************************************************************

#OPTIMIZATION


#Binning the continuous variables for use of Mutual Information

continous_cols <- df_new_over %>%

  dplyr::select(Administrative_Duration, Informational_Duration, ProductRelated_Duration, BounceRates, ExitRates, PageValues, SpecialDay)


#Scaling the continuous columns

standardized_cols = as.data.frame(scale(continous_cols))
```

```
col_names = names(standardized_cols)

binned_cols <- standardized_cols %>% mutate(
  Administrative_Duration=cut(Administrative_Duration, breaks = c(-Inf,-3.5,-2.5,-1.5,-0.5,0.5,1.5,2.5,3.5,Inf), labels = c(-4,-3,-2,-1,0,1,2,3,4)),
  Informational_Duration=cut(Informational_Duration, breaks = c(-Inf,-3.5,-2.5,-1.5,-0.5,0.5,1.5,2.5,3.5,Inf), labels = c(-4,-3,-2,-1,0,1,2,3,4)),
  ProductRelated_Duration=cut(ProductRelated_Duration,breaks = c(-Inf,-3.5,-2.5,-1.5,-0.5,0.5,1.5,2.5,3.5,Inf), labels = c(-4,-3,-2,-1,0,1,2,3,4)),
  BounceRates=cut(BounceRates,breaks = c(-Inf,-3.5,-2.5,-1.5,-0.5,0.5,1.5,2.5,3.5,Inf), labels = c(-4,-3,-2,-1,0,1,2,3,4)),
  ExitRates=cut(ExitRates,breaks = c(-Inf,-3.5,-2.5,-1.5,-0.5,0.5,1.5,2.5,3.5,Inf), labels = c(-4,-3,-2,-1,0,1,2,3,4)),
  PageValues=cut(PageValues,breaks = c(-Inf,-3.5,-2.5,-1.5,-0.5,0.5,1.5,2.5,3.5,Inf), labels = c(-4,-3,-2,-1,0,1,2,3,4)),
  SpecialDay=cut(SpecialDay,breaks = c(-Inf,-3.5,-2.5,-1.5,-0.5,0.5,1.5,2.5,3.5,Inf), labels = c(-4,-3,-2,-1,0,1,2,3,4))
)


binned <- df_new_over %>% mutate(
  Administrative_Duration = binned_cols$Administrative_Duration,
  Informational_Duration = binned_cols$Informational_Duration,
  ProductRelated_Duration = binned_cols$ProductRelated_Duration,
  BounceRates = binned_cols$BounceRates,
  ExitRates = binned_cols$ExitRates,
  PageValues = binned_cols$PageValues,
  SpecialDay = binned_cols$SpecialDay
)
```

```
#Mutual Information measures
#MI filter
MI = vector()
for (i in 1:17){
  MI <- c(MI, mutinformation(binned$Revenue, binned[,i]))
}
MI = data.frame(names(binned)[1:17],MI)
MI <- MI[with(MI, order(-MI)), ]

MI

#Minimum redundancy feature selection
#mMRM filter
score = MRMR(binned[1:17],binned$Revenue,17)$score
score = as.data.frame(score)
score

#RF- after feature selection
features_selected = c("Administrative_Duration", "Informational_Duration",
"ProductRelated_Duration", "ExitRates", "PageValues", "Month",
"OperatingSystems", "Browser", "Region", "TrafficType", "VisitorType",
"Weekend")
df_new_over <- df_new_over[, c(features_selected, "Revenue")]
test_df_new <- test_df_new[, c(features_selected, "Revenue")]

n<-length(names(df_new_over))
m = ceiling(log2(n))
```

```
rf_train<-
randomForest(as.factor(df_new_over$Revenue)~.,data=df_new_over,mtry=m
,ntree=100,importance=TRUE,proximity=TRUE)


#Predict
pred_2<-predict(rf_train,newdata=test_df_new)


#Confusion Matrix and Metrics
print("After feature selection")


#After feature selection
Optimized_RF <- confusionMatrix(pred_2, factor(test_df_new$Revenue))
print(Optimized_RF)
```

CONCLUSION

In this project, we explore methods to predict purchasing decision of consumers based on the 17 attributes. It is a well-structured, real-world dataset that inspires us a lot to analyze the behaviour of customers from both commercial insights and technical perspectives. The lesson worth taken away is that sometimes inference from our daily life experience could be inaccurate. For instance, we would generally believe “SpecialDay” has strong influence on the decision, since, it is in weekend we have more free time to browse the online shopping mart. Nonetheless, the result of our Random Forest model and exploratory data analysis demonstrates that there is no difference on this feature, probably due to the fact people are more used to spending their time during recess. We may also provide some suggestions based on our results, for instance, the website owners could re-distribute budget on advertisements by allocating more funds on months where customers are more likely to finalize their transactions so that it is more likely to lift the return rates. Also, they could re-design the informational pages to best cater to the customers’ demands for their longer stay, which further increases the probability of successful transactions.

Our findings support the argument that the features extracted from clickstream data during the visit convey important information regarding online purchasing intention prediction. The features that represent aggregated statistics of the clickstream data obtained during the visit are ranked near the top by the filter feature ranking algorithms. However, these metrics are also highly correlated with each other. On the other hand, although the session information-based features are less correlated with purchasing intention of the visitor, they contain unique information different from clickstream-based features.

Therefore, we apply a feature ranking method called minimum Redundancy-Maximum Relevance (mRMR) which takes such redundancies between the features into account. The findings show that choosing a minimal subset of combination of clickstream data aggregated statistics and session information such as the date and geographic region results in a more accurate and scalable system. Considering the real- time usage of the proposed system, achieving better or similar classification performance with minimal subset of features is an important factor for the e-commerce companies since a smaller number of features will be kept track during the session.