ILLINOIS INSTITUTE OF TECHNOLOGY

Department of Computer Science

CSP571 Data Preparation and Analysis

Finding the pattern behind the online shoppers purchasing intention

Naveen Raju Sreerama Raju Govinda Raju
nsreeramarajugovinda@hawk.iit.edu
A20516868

Karthik Kumar Kaiplody
kkumarkaiplody@hawk.iit.edu
A20523668

Prof. Jawahar Panchal
Date:04/30/2023

Table of Contents

1) Abstract

Our objective is to analyze trends in the online shoppers purchasing intention dataset using exploratory data analysis techniques, and build machine learning models to predict the purchasing intentions of visitors to a store's website. We plan to approach this as both a clustering and classification problem, grouping similar customers based on their purchasing behavior and predicting whether a new customer is likely to make a purchase based on their browsing and purchasing behavior. By applying these techniques, we hope to gain insights into customer behavior and optimize marketing strategies to increase sales.

2) Overview

The project is focused on analysing an online shoppers purchase intention dataset, which contains information about users' browsing and purchasing behavior on an e-commerce website. The goal is to understand the factors that influence users' purchase decisions and to develop machine learning models that can predict whether a user is likely to make a purchase or not.

The first step in the project is exploratory data analysis (EDA), which involves visualizing and summarizing the dataset to gain insights into the underlying patterns and relationships. This is followed by data preprocessing, which includes cleaning the data, handling missing values, and transforming the data into a format that is suitable for machine learning algorithms.

Next, the we have implemented several machine learning algorithms to predict purchase intention. The algorithms include Naive Bayes Classifier, K Nearest Neighbor, Random Forest, Support Vector Machines (SVM), and XGBoost classification. Each algorithm was trained and evaluated using different performance metrics such as accuracy, precision, recall, and F1 score.

Additionally, the we have applied clustering algorithms to segment the users based on their purchasing behavior. The clustering algorithms used in this project include K-means clustering, DBSCAN clustering, and Hierarchical clustering. These algorithms aim to group similar users together and identify patterns in their behavior.

Overall, the project aims to provide insights into the factors that influence online shoppers' purchase decisions and develop models that can predict users' purchase intentions accurately. The results of this project can help businesses optimize their marketing strategies and improve their conversion rates.

3) Data Sources

The data set used that is being used in this project was obtained from the UC Irvine Machine Learning Repository.

Data set contributes:

1. C. Okan Sakar
Department of Computer Engineering, Faculty of
Engineering and Natural Sciences, Bahcesehir University,
34349 Besiktas, Istanbul, Turkey

2. Yomi Kastro
Inveon Information Technologies Consultancy and Trade,
34335 Istanbul, Turke

4) Data set description

The dataset consists of feature vectors belonging to 12,330 sessions.The dataset consists of both numericaland categorical attributes.The 'Revenue' attribute can be used as the class label.

Attribute	Type	Description
Administrative	Numerical	Page category
Administrative Duration	Numerical	Total time spent in this page
Informational	Numerical	Page category
Informational Duration	Numerical	Total time spent in this page
Product Related	Numerical	Page category
Product Related Duration	Numerical	Total time spent in this page
Bounce rate	Numerical	The bounce rate for a web page is the percentage of visitors who navigate away from the site after viewing only that page, without interacting with the page or visiting any other pages on the site. So, it's not just about the visitors who enter the site from that page, but rather about visitors who land on the page and then leave without taking any further action.
Exit rate	Numerical	The exit rate for a web page is the percentage of visitors who leave the site after viewing that page as the last page in their session. Unlike bounce rate, which only takes into account the visitors who leave after viewing a single page, the exit rate includes visitors who may have viewed multiple pages on the site before leaving after viewing the specific page in question.

		To calculate the exit rate for a specific web page, you would divide the number of exits from that page by the total number of page views for that page.
Page value	Numerical	The Page Value feature is a metric in Google Analytics that represents the average value of a page that a user visited before completing an e-commerce transaction or a goal conversion on a website. It is calculated by dividing the total value of all transactions or goal completions by the number of unique page views for a particular page or set of pages.
Special day	Numerical	The "Special Day" feature indicates the closeness of the site visiting time to a specific special day (e.g. Mother's Day, Valentine's Day) in which the sessions are more likely to be finalized with transaction. The value of this attribute is determined by considering the dynamics of e-commerce such as the duration between the order date and delivery date. For example, for Valentina's day, this value takes a nonzero value between February 2 and February 12, zero before and after this date unless it is close to another special day, and its maximum value of 1 on February 8.
Operating system	Categorical	Operating system of the visitor.
Browser	Categorical	Browser of the visitor.
Region	Categorical	Geographic region from which the session has been strated by the visitor.
Traffic type	Categorical	Traffic sources by which the visitor has arrived at the website.
Visitor type	Categorical	Visitor type as "New visitor", "Returning Visitor" and "Other"
Weekend	Categorical	True if it is either Saturday or Sunday. Or else it is False.
Month of the year	Categorical	Jan, Feb,…..,December
Revenue	Categorical	True is customer purchased anything or else it is False

5)Data Processing

5.1) Check number of observations with NA values

In this data we do not have any observations with NA values.

5.2) Convert Month feature data type to factor data type

Initially the Month feature column data type is character. "as.factor" is used to convert month column to factor data type, this is used when the column is categorical attribute with fixed number of categories, in our cases it is months of a year.

5.3) Fixing naming convention of month names in Month column

First check count of number of observations for each month.

```
Aug  Dec  Feb  Jul June  Mar  May  Nov  Oct  Sep
433 1727  184  432  288 1907 3364 2998  549  448
```

Use "as.factor" to convert month column to factor data type from character data type, this is used when the column is categorical attribute with fixed number of categories, in our cases it is months of a year.

```
Jan  Feb  Mar  Apr  May  Jun  Jul  Aug  Sep  Oct  Nov  Dec
  0  184 1907    0 3364    0  432  433  448  549 2998 1727
```

Even though the dataset includes data for the month of June, when we convert the data type to factor, we observe that the number of observations for June is zero. This is because in the original dataset, June is encoded as "June" instead of "Jun". Therefore, we need to convert "June" to "Jun" and then convert the data type back to factor.

We have observed that when using the table function on the Month column of the dataset, it shows the total number of observations for the month of Jun.

```
Aug  Dec  Feb  Jul  Jun  Mar  May  Nov  Oct  Sep
433 1727  184  432  288 1907 3364 2998  549  448
```

Now, when we analyze the data using the "str" function, we can observe that the Month column is ordered with 12 levels.

$ Month : Factor w/ 12 levels "Jan","Feb","Mar",..: 2 2 2 2 2 2 2 2 2 2 …

5.4) Data per-processing for model training

5.4.1)Transforming categorical attributes into “factor” data type and then perform one-hot encoding.

Categorical attributes are: OperatingSystems, Browser, Region, TrafficType, VisitorType

5.4.2)Convert Revenue attribute data type to a factor.

5.4.3)Transforming Boolean attributes into “int” data type

Boolean attributes:Weekend, Revenue

5.4.4) Have a copy of the dataset, one with one-hot encoding and the other without.

5.4.5) Create a train-test split for the dataset, one with one-hot encoding and the other without, using a 70/30 proportion for training and testing

5.4.6) One hot encoded train and test data pre-processing

- Separate numerical and categorical attributes
- Scale the numerical attributes. Scale function is used to standardize by subtracting mean for each value and dividing by standard deviation, this bring value to have mean zero and standard deviation of one.
- Combine categorical and scales numerical attributes.
- There is huge imbalance in data set as Revenue=0 is the majority. Hence the algorithm tries to over fit on majority class. Hence we use “ovun.sample” to over-sample minority class. The function tries to generate synthetic data points of minority class using “SMOTE” algorithm, this creates new observation by interpolating between the given sample in feature space. Here we are trying to increase minority class observations by two times.
  Oversampling is done only to the train set.
- There are 77 columns in data after one hot encoding, in which last column is target variable.

```
 [1] "Administrative"              "Administrative_Duration"     "Informational"
 [4] "Informational_Duration"      "ProductRelated"              "ProductRelated_Duration"
 [7] "BounceRates"                 "ExitRates"                   "PageValues"
[10] "SpecialDay"                  "Month_Jan"                   "Month_Feb"
[13] "Month_Mar"                   "Month_Apr"                   "Month_May"
[16] "Month_Jun"                   "Month_Jul"                   "Month_Aug"
[19] "Month_Sep"                   "Month_Oct"                   "Month_Nov"
[22] "Month_Dec"                   "OperatingSystems_1"          "OperatingSystems_2"
[25] "OperatingSystems_3"          "OperatingSystems_4"          "OperatingSystems_5"
[28] "OperatingSystems_6"          "OperatingSystems_7"          "OperatingSystems_8"
[31] "Browser_1"                   "Browser_2"                   "Browser_3"
[34] "Browser_4"                   "Browser_5"                   "Browser_6"
[37] "Browser_7"                   "Browser_8"                   "Browser_9"
[40] "Browser_10"                  "Browser_11"                  "Browser_12"
[43] "Browser_13"                  "Region_1"                    "Region_2"
[46] "Region_3"                    "Region_4"                    "Region_5"
[49] "Region_6"                    "Region_7"                    "Region_8"
[52] "Region_9"                    "TrafficType_1"               "TrafficType_2"
[55] "TrafficType_3"               "TrafficType_4"               "TrafficType_5"
[58] "TrafficType_6"               "TrafficType_7"               "TrafficType_8"
[61] "TrafficType_9"               "TrafficType_10"              "TrafficType_11"
[64] "TrafficType_12"              "TrafficType_13"              "TrafficType_14"
[67] "TrafficType_15"              "TrafficType_16"              "TrafficType_17"
[70] "TrafficType_18"              "TrafficType_19"              "TrafficType_20"
[73] "VisitorType_New_Visitor"     "VisitorType_Other"           "VisitorType_Returning_Visitor"
[76] "Weekend"                     "Revenue"
```

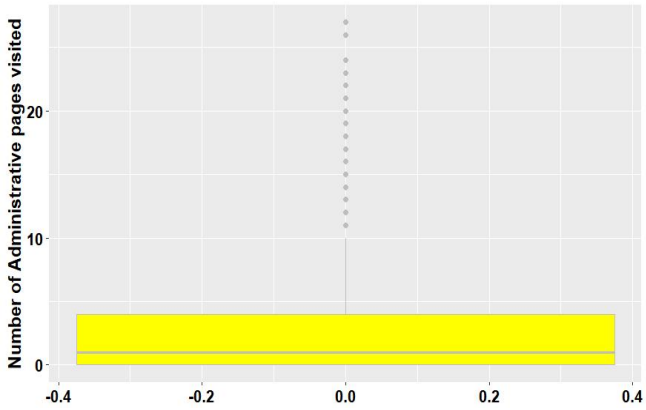
5.4.7) Preprocess the training and testing data for the dataset without one-hot encoding

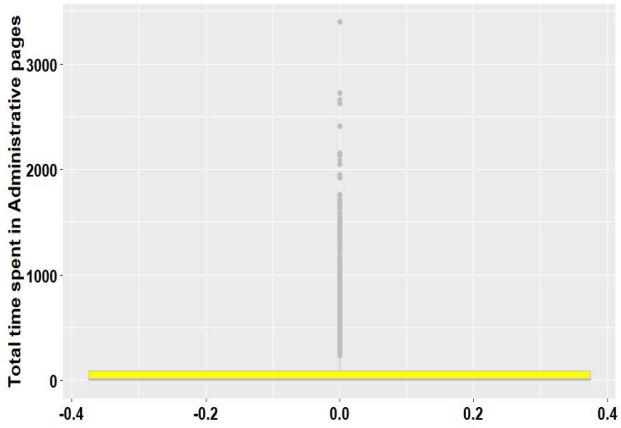
- There is huge imbalance in data set as Revenue=0 is the majority. Hence the algorithm tries to over fit on majority class. Hence we use “ovun.sample” to over-sample minority class. The function tries to generate synthetic data points of minority class using “SMOTE” algorithm, this creates new observation by interpolating between the given sample in feature space. Here we are trying to increase minority class observations by two times.
  Oversampling is done only to the train set.

6)Data Analysis

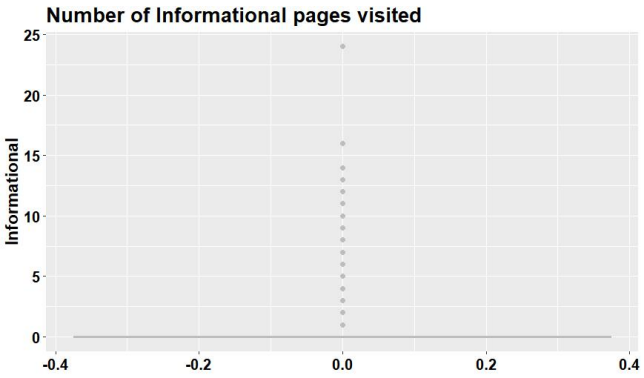
6.1) Exploring data distribution of different page category and time spent in it.

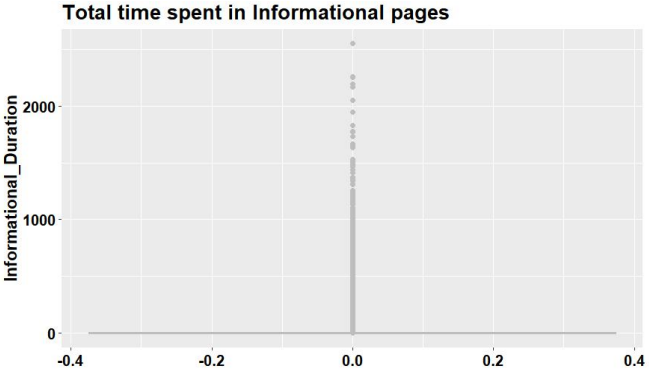
Exploring data pattern of “Administrative” and “Administrative_Duration”



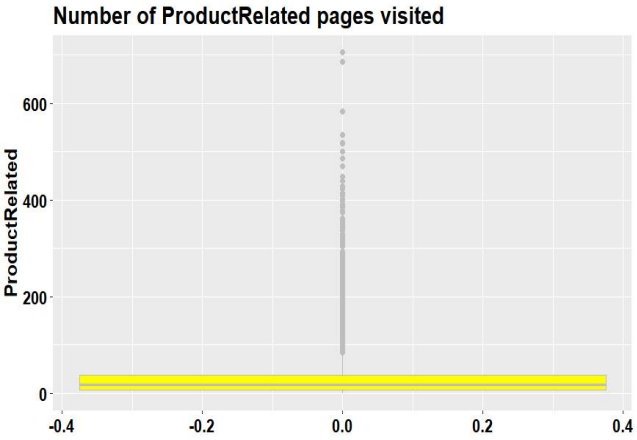


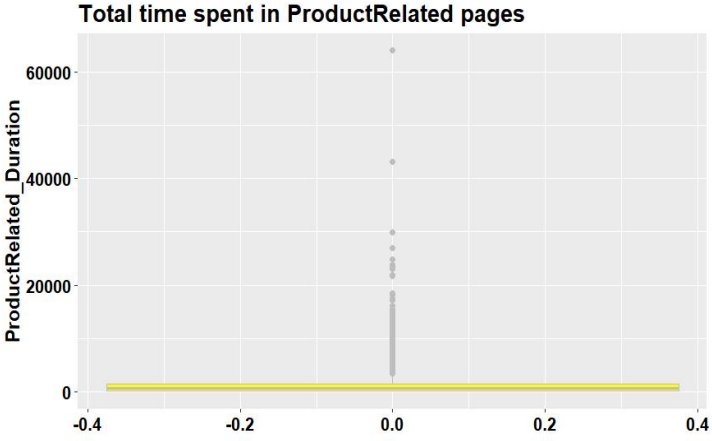
Exploring data pattern of “Informational” and “Informational_Duration”





Exploring data pattern of “Product Related” and “Product Related Duration”





Analysing summary of all above plots, we can observe that:

- Median of Number of Administrative pages visited= 1
- Median of Number of Informational pages visited= 0
- Median of Number of Product Related pages visited= 18

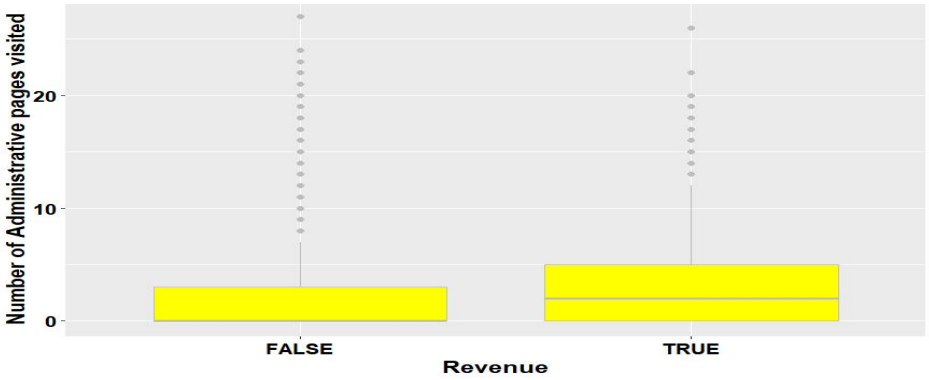
- Median time spent in Administrative pages= 7.5
- Median time spent in Informational Related pages= 0
- Median time spent in Product Related pages= 598.9369

- Analysing number of pages visit of 3 different page categories it clearly says that customers are interested more in Product related pages rather than knowing information of the product in detail.

- Analysing total time spent in 3 different page categories, it clearly says that customers spend most of the time in product related pages whereas they are not interested in spending time in information related pages.

6.2) Exploring the data distribution of different page categories versus the target variable Revenue, as well as the time spent on each page category versus the target variable Revenue.

Exploring data pattern of "Administrative" versus "Revenue"

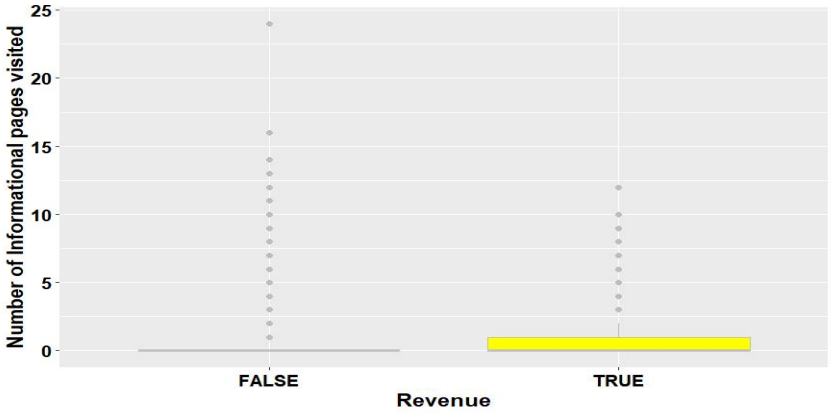


- Median number of administrative pages visited by a customer who end up buying is = 2.
- Median number of administrative pages visited by a customer who did not end up buying is = 0

Exploring data pattern of "Administrative_Duration" versus "Revenue"

- Median duration of time spent in administrative pages by a customer who end up buying is= 52.36667
- Median duration of time spent in administrative pages by a customer who did not end up buying is= 0

Exploring data pattern of “Informational” versus “Revenue”



- Median number of Informational pages visited by a customer who end up buying is= 0.
- Median number of Informational pages visited by a customer who did not end up buying is= 0

Exploring data pattern of “Informational_Duration” versus “Revenue”

- Median number of duration spent on Informational pages by a customer who end up buying is= 0
- Median number of duration spent on Informational pages by a customer who did not end up buying is= 0

Exploring data pattern of “Product Related” versus “Revenue”

- Median number of Product Related pages visited by a customer who end up buying is= 29.
- Median number of Product Related pages visited by a customer who did not end up buying is= 16

Exploring data pattern of “Product Related Duration” versus “Revenue”

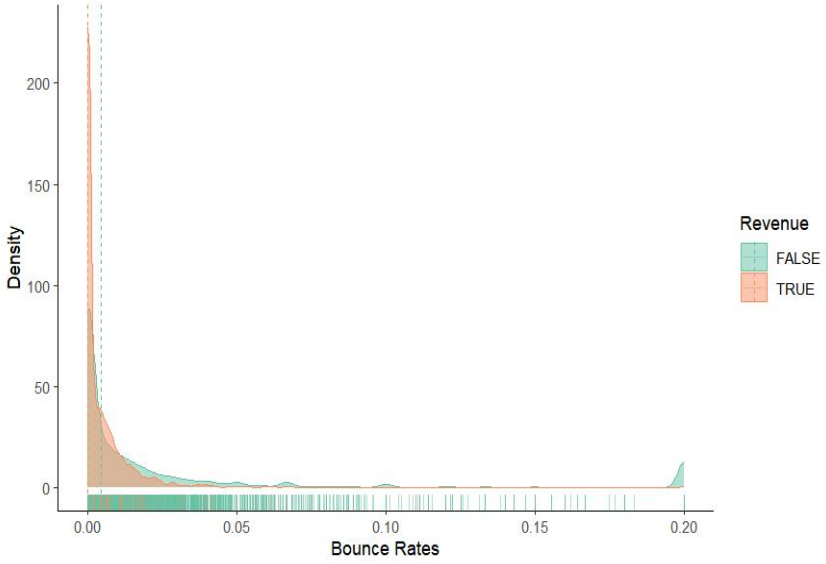
- Median duration of Product Related pages visited by a customer who end up buying is= 1109.906
- Median duration of Product Related pages visited by a customer who did not end up buying is= 510.19

Based on above analysis,these are the observations made:

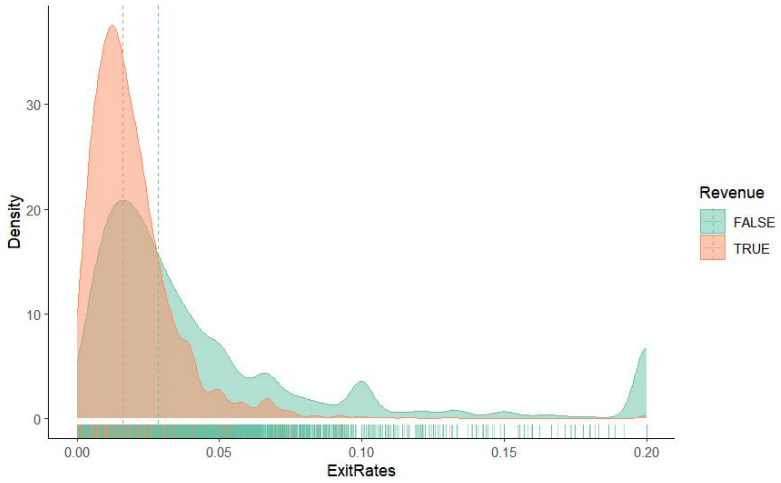
- People who end up buying will mostly visit administrative page and spend almost 52seconds.
- People who end up not buying will mostly not visit administrative page.
- People are least interested in visiting informational page.
- People who end up buying will mostly visit product related page and spend almost 1109 seconds.
- People who will end up buying will mostly visit product related page and spend almost 510 seconds.
- But people who end up buying will visit more product related than the ones who don't.

6.3) Exploring the data distribution of “Bounce Rates”, “Exit Rates” and “Page Values” features versus the target variable Revenue respectively.

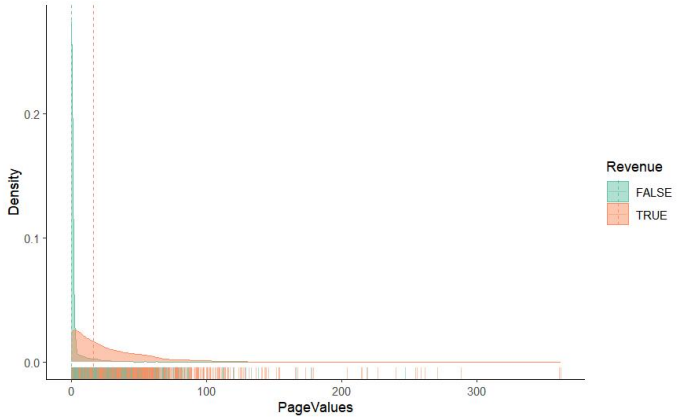
Exploring data pattern of “Bounce Rates” versus “Revenue”



Exploring data pattern of “Exit Rates” versus “Revenue”

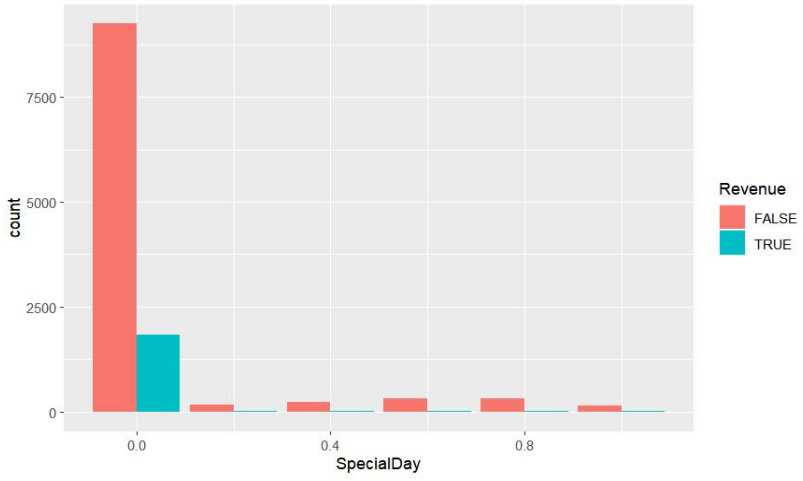


Exploring data pattern of “Page Values” versus “Revenue”



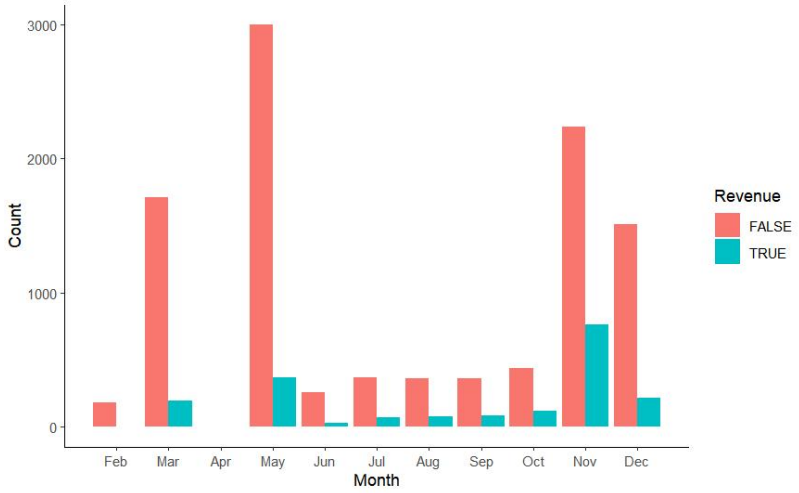
There is no noticeable disparity in Bounce Rates between customers who made a purchase and those who did not. However, customers who ended up making a purchase had lower Exit Rates on average, indicating that they were more likely to remain on the website's pages. Additionally, customers who did not make a purchase had significantly lower Page Values, suggesting that they spent less time on related pages.

6.4) Exploring the data distribution of “Special Day” features versus the target variable Revenue.



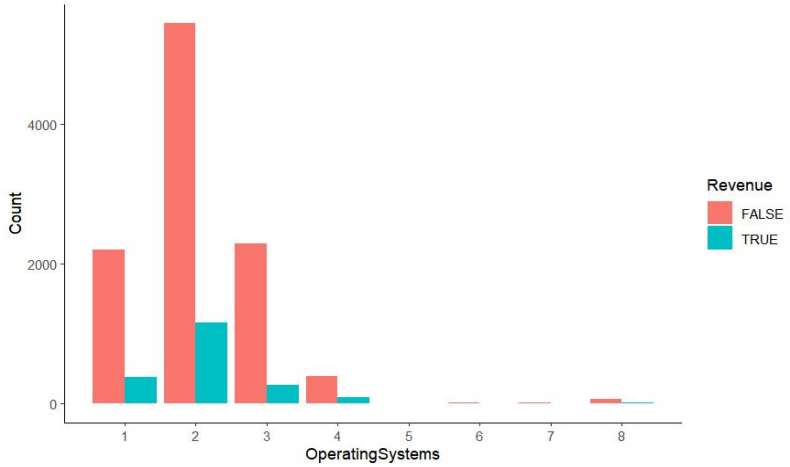
Here we can analyse that majority of revenue is on non special day, and impact of revenue due to special fay is very less.

6.5) Exploring the data distribution of "Month" features versus the target variable Revenue.



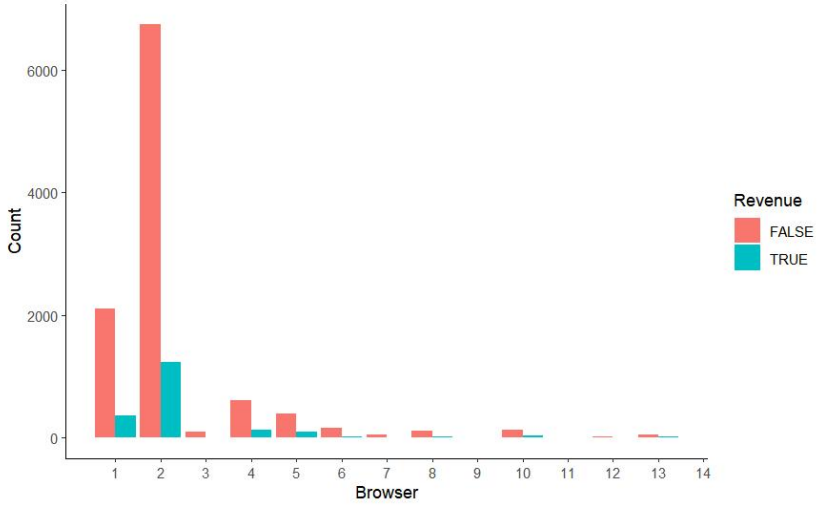
- Most revenue is from the months of March, May, November and December
- Least revenue is from the month June, July, August, September and October
- Data is not available for January and April

6.6) Exploring the data distribution of "Operating Systems" features versus the target variable Revenue.



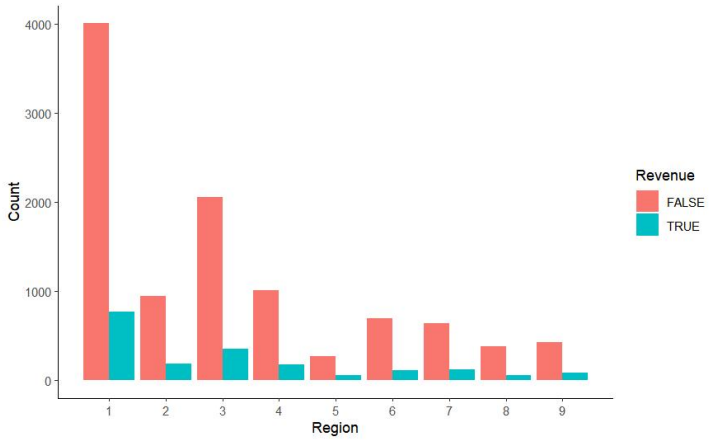
The highest revenue comes from operating system 2, and the highest non-revenue also comes from operating system 2.

6.7) Exploring the data distribution of "Browser" features versus the target variable Revenue.



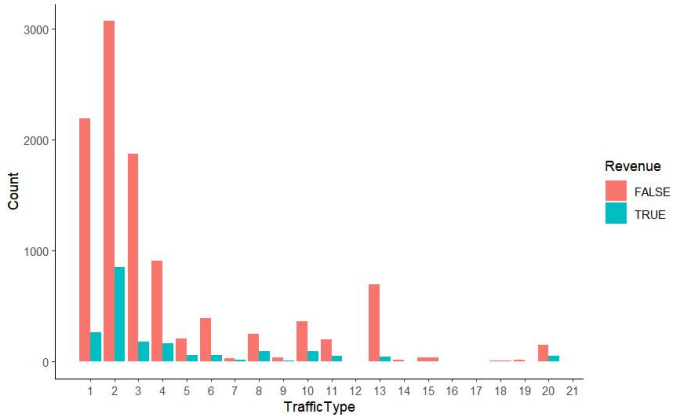
The highest revenue comes from Browser 2, and the highest non-revenue also comes from Browser 2.

6.8) Exploring the data distribution of "Region" features versus the target variable Revenue.



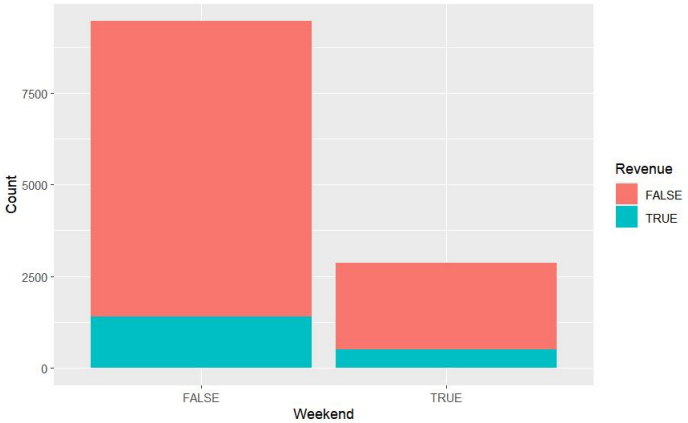
The highest revenue comes from Region 1, and the highest non-revenue also comes from Region 1.

6.9) Exploring the data distribution of "Traffic Type" features versus the target variable Revenue.



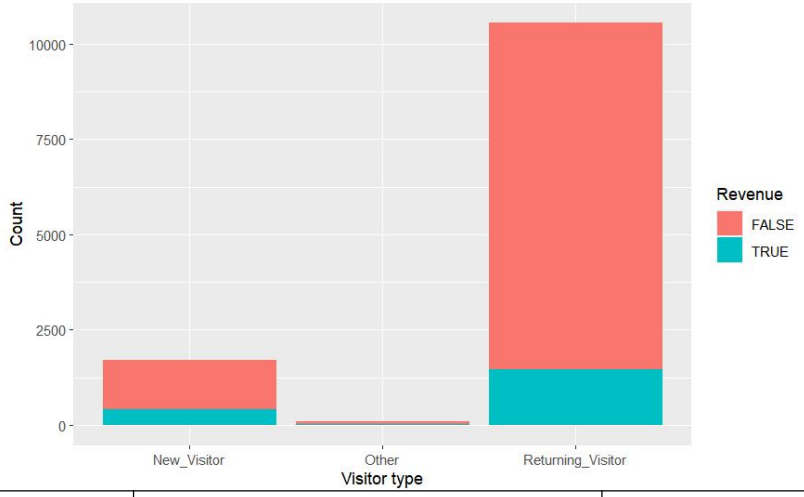
The highest revenue comes from Traffic Type2, and the highest non-revenue also comes from Traffic Type 2.

6.10) Exploring the data distribution of "Weekend" features versus the target variable Revenue.



Data shows that revenue on non weekend is almost thrice that of revenue on weekend

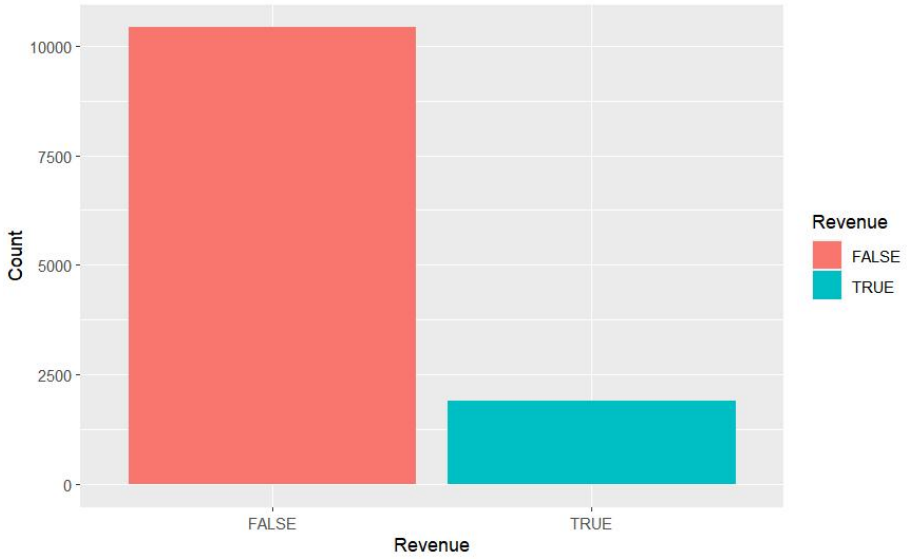
6.11) Exploring the data distribution of “Visitor Type” features versus the target variable Revenue.



	FALSE	TRUE
New_Visitor	1272	422
Other	69	16
Returning_Visitor	9081	1470

Data shows that revenue from returning visitor is approximately 3.5 times as that of new visitor

6.12) Exploring the data distribution of “Revenue” feature.



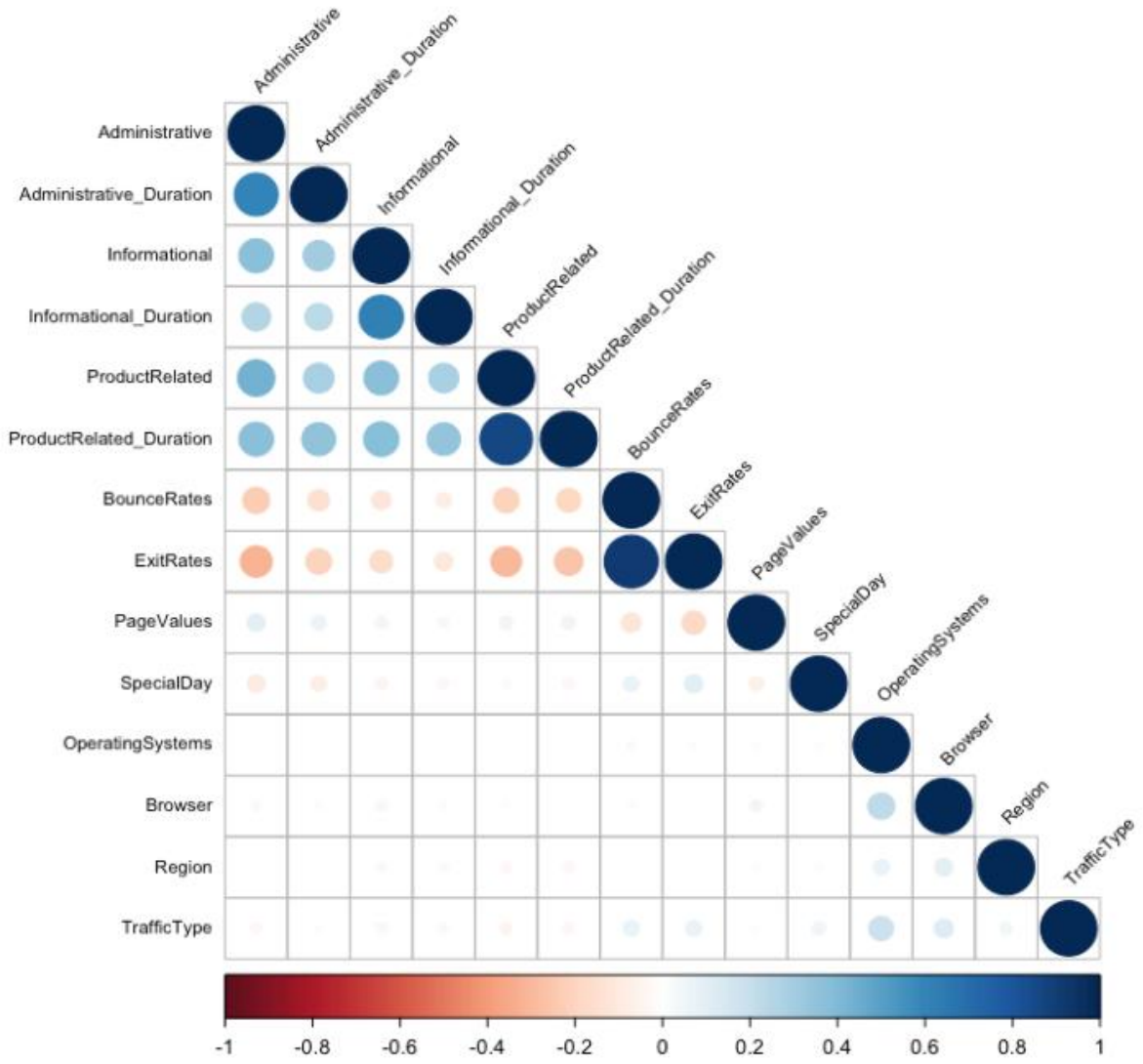
FALSE	TRUE
10422	1908

Data shows that there are more data to explain non revenue than revenue. That is there are approximately 5.5 times more data explaining non revenue that revenue.

6.13) Analyze the ratio of 'Revenue True' or 'Revenue False' in the training and testing sets.

- The proportion of 'Revenue True' and 'Revenue False' in the training set is 0.8453071 and 0.1546929, respectively.
- The proportion of 'Revenue True' and 'Revenue False' in the test set is 0.8451351 and 0.1548649, respectively.

6.14) Correlation plot



Correlation plot depicts the correlation between the features " informational" and "informational duration" , "administrative" and "administrative_duration", "product related" and "product related duration". Where we can consider only one with each pair for better modelling, which will reduce the overfitting of the data.

7) Model training

7.1) Naive Bayes Classifier

Naive Bayes is a probabilistic classifier that applies Bayes' theorem with the assumption of independence between each pair of features. In Naive Bayes, the algorithm calculates the probability of a data point belonging to each class, based on the values of its features. Then, the algorithm selects the class with the highest probability as the predicted class for the data point. The "naive" assumption that the features are independent can be a limitation, as this may not always be the case in real-world scenarios. Nevertheless, Naive Bayes often performs well in practice, especially when the dataset has a large number of features and a relatively small number of training examples.Naive Bayes classifier tries to classify new observation based on conditional probability.

- Define a train control method with 5 fold cross validation

- Train the model using “caret library” on data set without one hot encoding

Results:

Prediction	Reference	
	0	1
0	31.6	6.0
0	18.4	44.0

Average accuracy: 75.6%

- Train the model using “caret library” on data set with one hot encoding

Results:

Prediction	Reference	
	0	1
0	84.5	15.5
0	0	0

Average accuracy: 84.5%
Thus Naive Bayes classifier works better when categorical variables are one hot encoded.

7.2) k-Nearest Neighbor

k-Nearest Neighbor (k-NN) is a non-parametric machine learning algorithm used for classification and regression tasks. The algorithm makes predictions based on the proximity of a new data point to its k-nearest neighbors in the training set.The basic idea behind k-NN is that similar data points tend to belong to the same class or have similar values. The algorithm uses a distance metric, such as Euclidean or Manhattan distance, to measure the similarity between data points. The value of k, which represents the number of nearest neighbors to consider, is a hyper parameter that needs to be set before running the algorithm.For classification tasks, k-NN predicts the class label of a new data point based on the majority class of its k-nearest neighbors. For regression tasks, k-NN predicts the value of a new data point based on the average value of its k-nearest neighbors. One of the advantages of k-NN is that it is a simple and easy-to-understand algorithm that does not require any assumptions about the underlying distribution of the data. However, its performance can be sensitive to the choice of distance metric, the value of k, and the scale of the data. It can also be computationally expensive for large datasets.Overall, k-NN is a popular and useful algorithm for classification and regression tasks, especially for small to medium-sized datasets where it can achieve high accuracy with relatively low computational cost.

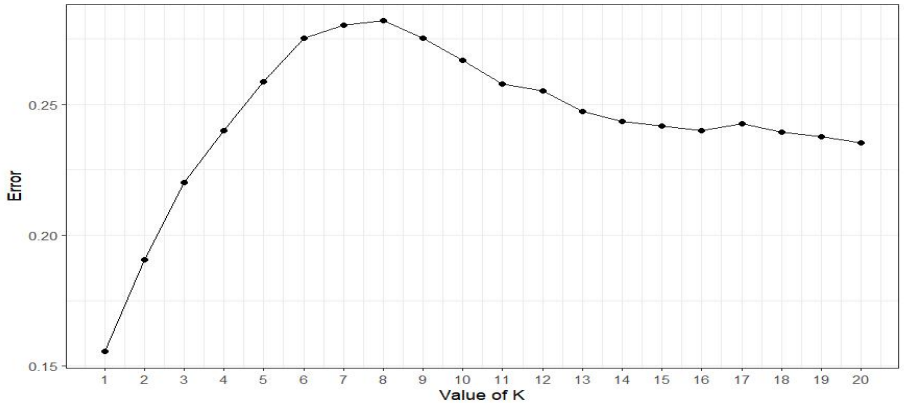
- Train the model on one hot encoded data set using “knn” function of “class” library.

Results:

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 2847  296
         1  280  277

               Accuracy : 0.8443
                 95% CI : (0.8322, 0.8559)
    No Information Rate : 0.8451
    P-Value [Acc > NIR] : 0.5652

                  Kappa : 0.3984

 Mcnemar's Test P-Value : 0.5320

            Sensitivity : 0.9105
            Specificity : 0.4834
         Pos Pred Value : 0.9058
         Neg Pred Value : 0.4973
             Prevalence : 0.8451
         Detection Rate : 0.7695
   Detection Prevalence : 0.8495
      Balanced Accuracy : 0.6969

       'Positive' Class : 0
```

- Visualizing KNN with different k values(number of nearest neighbor)

Let's choose k range from 1 to 20.



Here we observe that for k=1 model has lowest error.
Thus, Let us train knn with k=1.

Results:

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 2847  296
         1  280  277

               Accuracy : 0.8443
                 95% CI : (0.8322, 0.8559)
    No Information Rate : 0.8451
    P-Value [Acc > NIR] : 0.5652

                  Kappa : 0.3984

 Mcnemar's Test P-Value : 0.5320

            Sensitivity : 0.9105
            Specificity : 0.4834
         Pos Pred Value : 0.9058
         Neg Pred Value : 0.4973
             Prevalence : 0.8451
         Detection Rate : 0.7695
   Detection Prevalence : 0.8495
      Balanced Accuracy : 0.6969

       'Positive' Class : 0
```

7.3) Random Forest

Random Forest is a popular machine learning algorithm used for both classification and regression tasks. It is an ensemble learning method that combines multiple decision trees to improve the accuracy and reduce the over fitting of the model. The basic idea behind Random Forest is to create a large number of decision trees using a random subset of the features and a random subset of the training data. Each tree is trained independently, and its predictions are combined to make the final prediction. This approach reduces the variance of the model and improves its generalization performance. The algorithm also uses a technique called bagging (bootstrap aggregating) to further reduce the variance of the model. Bagging involves randomly sampling the training data with replacement to create multiple subsets of the data, which are then used to train each decision tree. For classification tasks, Random Forest predicts the class label of a new data point based on the majority vote of the individual trees. For regression tasks, it predicts the value of a new data point based on the average value of the predictions of the individual trees. Random Forest is a powerful and flexible algorithm that can handle both categorical and continuous data. It is also less prone to over fitting than a single decision tree, and can be used for feature selection and variable importance analysis. However, it can be computationally expensive for large datasets and may not perform well on datasets with imbalanced class distributions. Overall, Random Forest is a popular and effective algorithm that can achieve high accuracy and generalization performance for both classification and regression tasks.

- Random forests consider a random subset of predictor variables at each split to decrease correlation between trees and improve model performance. The parameter "mtry" determines the number of variables to consider at each split. Typically, a popular approach is to set "mtry" equal to the square root of the number of predictors. However, the random Forest package offers a more advanced method for selecting "mtry" based on the number of predictors. This method uses the formula "m = ceiling(log2(n))" where "m" is the rounded-up base 2 logarithm of the number of predictors in the dataset "n". This formula has been found to perform well for a diverse range of datasets. ntree=100, collection 100 trees to be constructed.
- Model is trained on data set without one-hot encoding

Results:

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 2955  189
         1  172  384

               Accuracy : 0.9024
                 95% CI : (0.8924, 0.9118)
    No Information Rate : 0.8451
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.6227

 Mcnemar's Test P-Value : 0.3997

            Sensitivity : 0.9450
            Specificity : 0.6702
         Pos Pred Value : 0.9399
         Neg Pred Value : 0.6906
             Prevalence : 0.8451
         Detection Rate : 0.7986
   Detection Prevalence : 0.8497
      Balanced Accuracy : 0.8076

       'Positive' Class : 0

Thus accuracy of random forest is 89.81%
```

- Recursive Feature Elimination:

Output of RFE provides valuable information about the importance of different features and how they affect the performance of the model. This information can be used to improve the accuracy and interpret ability of the final model.

a) Recursive Feature Elimination is used to build Random Forest model with all possible subsets of features.
b) "rfeControl" function is used with 10 fold cross validation.
c) "rfe" function is used for training.

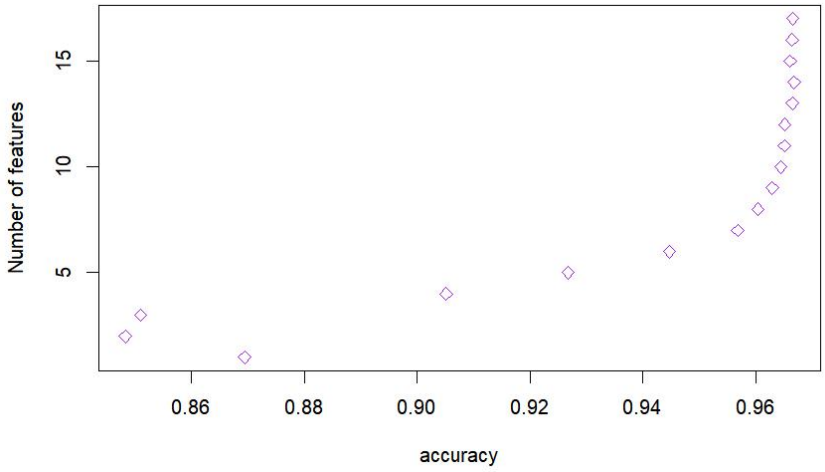
Output:

Variables	Accuracy	Kappa	AccuracySD	KappaSD
1	0.869431951	0.738863401	0.007049609	0.014092158
2	0.848321599	0.6966427	0.006334549	0.012659938
3	0.850994757	0.701988995	0.006064325	0.012121113
4	0.905140123	0.810278194	0.013344707	0.026692774
5	0.926730159	0.853460616	0.019189933	0.038381349
6	0.944686211	0.889372426	0.01243196	0.024866753
7	0.956819031	0.913638362	0.006278806	0.012557185
8	0.960383398	0.920766867	0.005161325	0.010322675
9	0.962919476	0.925838934	0.005074801	0.010149926

10	0.964427311	0.928854527	0.004939367	0.009879258
11	0.965112524	0.93022482	0.005336765	0.010674711
12	0.965112618	0.930224951	0.005394665	0.010790778
13	0.966483514	0.932966754	0.00542278	0.010846924
14	0.966757628	0.933515008	0.005766589	0.011534296
15	0.96607218	0.932144105	0.00573966	0.011480498
16	0.966415115	0.932830057	0.005339922	0.010680789
17	0.966552148	0.933103963	0.005851419	0.011704343

a) Variables: The number of variables (i.e., features) in the subset being evaluated.
b) Accuracy: The accuracy of the model when trained and tested on the current subset of variables.
c) Kappa: The kappa statistic, a measure of agreement between predicted and observed values, for the current subset of variables.
d) AccuracySD: The standard deviation of the accuracy across the different cross-validation folds.

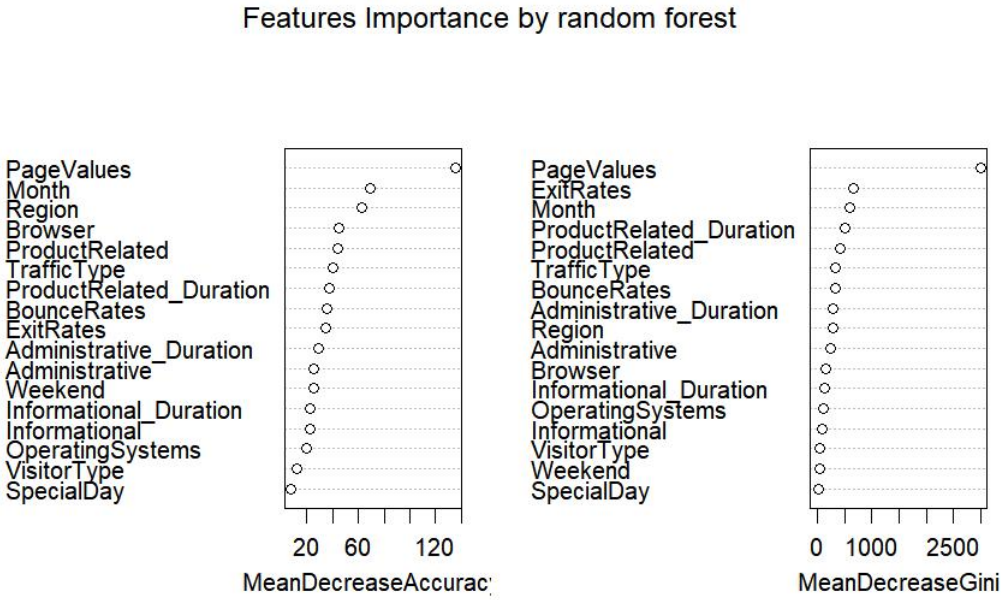
- Number of features vs Accuracy plot



We can see accuracy increased drastically initially as more feature variables are added later on it became

- Variable importance plot

The function "varImpPlot" uses two metrics, MeanDecreaseAccuracy (MDA) and MeanDecreaseGini (MDG), to measure the importance of features. MDA measures the decrease in accuracy when a single variable is excluded or changed, while MDG measures the decrease in node impurity.



This shows that feature variables like Page values is most important one. Other than this few of the other important feature variables are Month, ExitRates, ProductRelated_Duration.

In both plots we see Visitor Type and Special Day is least significant.

- Feature variable importance table

This table is computed based on "information" function using trained random forest trained model which was shown initially shown above.

Feature variable importance table	MeanDecreaseAccuracy
Administrative	25.740664
Administrative_Duration	29.428259
Informational	22.772219
Informational_Duration	22.896964
ProductRelated	44.383527
ProductRelated_Duration	37.415946
BounceRates	35.813305
ExitRates	34.568105
PageValues	135.122725
SpecialDay	8.007717
Month	68.915250
OperatingSystems	19.990634
Browser	45.293001
Region	62.972472
TrafficType	39.930089
VisitorType	12.686405
Weekend	25.540213

Top 10 features are:

PageValues, Month, Region, Browser, ProductRelated, TrafficType, ProductRelated_Duration, BounceRates, ExitRates and Administrative_Duration.

Hence, train Random forest based on top 10 features.

```
          Reference
Prediction    0    1
         0 2933  190
         1  194  383

               Accuracy : 0.8962
                 95% CI : (0.8859, 0.9059)
    No Information Rate : 0.8451
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.6046

 Mcnemar's Test P-Value : 0.8783

            Sensitivity : 0.9380
            Specificity : 0.6684
         Pos Pred Value : 0.9392
         Neg Pred Value : 0.6638
             Prevalence : 0.8451
         Detection Rate : 0.7927
   Detection Prevalence : 0.8441
      Balanced Accuracy : 0.8032

       'Positive' Class : 0
```

a) Thus accuracy of random forest on top 10 feature variable is 89.6%
b) Whereas random forest on all feature variable is 90.24%
c) But such a small difference in accuracy is acceptable as model complexity has decreased.

7.4) Support Vector Machine

Support Vector Machine (SVM) is a popular machine learning algorithm used for both classification and regression tasks. The main idea behind SVM is to find the best possible boundary, called hyperplane, that separates the data into different classes. SVM works by mapping the input data to a higher-dimensional feature space where it becomes easier to find a hyperplane that separates the classes. The goal of SVM is to maximize the margin, which is the distance between the hyperplane and the nearest data points from each class. SVM is particularly useful when dealing with high-dimensional data and when there is no clear linear separation between the classes. In such cases, SVM can use a kernel function to transform the data into a higher-dimensional space where it becomes easier to separate the classes. The most commonly used kernel functions are linear, polynomial, and radial basis function (RBF). In SVM, there are two types of classifiers: linear and nonlinear. In the linear case, the boundary between the classes is a straight line. In the nonlinear case, the boundary can be a complex curve or surface. SVM is also known for its ability to handle imbalanced datasets, where one class has significantly more samples than the other.
SVM has several hyperparameters that can be tuned to optimize its performance, such as the kernel type, kernel parameters, regularization parameter, and others.

- Train the model using data set with one hot encoding.
- Use “svm” function from “e1071” library.

Linear kernel and radial basis function (RBF) kernel are two commonly used kernel functions in SVM:

The linear kernel computes the dot product between two input vectors in the original feature space. It is a simple and efficient kernel that works well when the data is linearly separable. However, it may not be effective when the data is non linearly separable.

The RBF kernel, on the other hand, maps the input data to an infinite-dimensional feature space using a Gaussian function. It can effectively capture complex nonlinear relationships between the features, making it a popular choice in many applications. The RBF kernel is more flexible and can often lead to better classification accuracy, but it may be computationally expensive and prone to over fitting if the kernel parameter is not properly tuned.

a) Parameters used during training: Kernel : linear
Result:

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 2794  121
         1  333  452

               Accuracy : 0.8773
                 95% CI : (0.8663, 0.8877)
    No Information Rate : 0.8451
    P-Value [Acc > NIR] : 1.453e-08

                  Kappa : 0.5928

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

            Sensitivity : 0.8935
            Specificity : 0.7888
         Pos Pred Value : 0.9585
         Neg Pred Value : 0.5758
             Prevalence : 0.8451
         Detection Rate : 0.7551
   Detection Prevalence : 0.7878
      Balanced Accuracy : 0.8412

       'Positive' Class : 0
```

b) Parameters used during training: Kernel : radial

Result:

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 2824  120
         1  303  453

               Accuracy : 0.8857
                 95% CI : (0.875, 0.8958)
    No Information Rate : 0.8451
    P-Value [Acc > NIR] : 8.165e-13

                  Kappa : 0.6136

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

            Sensitivity : 0.9031
            Specificity : 0.7906
         Pos Pred Value : 0.9592
         Neg Pred Value : 0.5992
             Prevalence : 0.8451
         Detection Rate : 0.7632
   Detection Prevalence : 0.7957
      Balanced Accuracy : 0.8468

       'Positive' Class : 0
```

Accuracy of SVM using linear kernel=87.73%
Accuracy of SVM using Radial basis kernel=88.56%
Using Radial basis kernel in SVM improves accuracy by marginal percentage.

7.5) XG Boost

XGBoost (extreme Gradient Boosting) is a powerful machine learning algorithm used for regression, classification, and ranking problems. It is based on the gradient boosting framework and uses an ensemble of decision trees to make predictions.The main idea behind XGBoost is to iteratively add decision trees to the model and optimize a loss function that measures the difference between the predicted and actual values. In each iteration, XGBoost calculates the gradients of the loss function with respect to the predicted values and uses them to update the model parameters, including the weights of the decision trees. The algorithm also applies regularization techniques, such as L1 and L2 regularization, to prevent overfitting. One of the key strengths of XGBoost is its speed and scalability, which is achieved through parallel processing and optimization techniques. XGBoost can handle large datasets with millions of examples and thousands of features, and can be run on a single machine or in distributed environments. XGBoost has several hyperparameters that can be tuned to optimize its performance, such as the learning rate, maximum depth of the decision trees, and regularization parameters. The choice of hyperparameters depends on the specific problem and the characteristics of the data. XGBoost has been used successfully in various applications, including predicting stock prices, image classification, and natural language processing.

- Training is done on top ten features selected during random forest training
- Training parameters:

a) objective = "binary:logistic"
b) eta = 0.3
c) max_depth = 6
d) eval_metric = "auc"
e) Nrounds = 100
f) Early stopping rounds = 10

objective: the loss function to be optimized during training. In this case, binary:logistic is used as the objective since it is a binary classification problem.

eta: the learning rate, which controls the step size at each iteration while moving towards a minimum of a loss function. A larger learning rate can result in faster convergence but may also cause overshooting. In this case, eta is set to 0.3, which is a common starting value.

Max depth: the maximum depth of a tree. A larger value can lead to a more complex model, but may also result in overfitting. In this case, max_depth is set to 6.

Eval metric: the evaluation metric used to monitor the performance of the model during training. In this case, auc (Area Under the Receiver Operating Characteristic Curve) is used since it is a common metric for binary classification problems.

nrounds: the maximum number of boosting rounds (or trees) to perform. In this case, the value is set to 100, which means that the model will train up to 100 trees.

Early stopping rounds: a parameter that specifies the number of rounds to continue training without seeing an improvement in the evaluation metric specified.

- Evaluation metric:

a) “auc “is used as the eval_metric parameter. auc stands for "Area Under the Receiver Operating Characteristic Curve", which is a common metric used to evaluate binary classification models like XGBoost. The ROC curve is a plot of the true positive rate (TPR) against the false positive rate (FPR) for different classification thresholds. The AUC is the area under this curve, and it ranges from 0 to 1, with higher values indicating better performance.

b) In XGBoost, the auc metric is calculated by first predicting the probabilities of the positive class (i.e., the probability of the target variable being 1) for the test set. These predicted probabilities are then used to calculate the ROC curve, and the AUC is the area under the curve.

c) During training, the auc metric is calculated after each boosting round to monitor the model's performance. The model with the highest AUC on the validation set is usually selected as the final model, as it has the best balance between true positives and false positives.

d) In summary, auc is a widely used evaluation metric for binary classification problems like XGBoost, and it measures the model's ability to distinguish between positive and negative examples. The higher the AUC, the better the model's performance.

Will train until test_auc hasn't improved in 10 rounds.

Output:

[1]	train	auc:0.940757	test	auc:0.909651
[2]	train	auc:0.949137	test	auc:0.918012
[3]	train	auc:0.952685	test	auc:0.922717
[4]	train	auc:0.954565	test	auc:0.926051
[5]	train	auc:0.957042	test	auc:0.926611
[6]	train	auc:0.958906	test	auc:0.927626
[7]	train	auc:0.961152	test	auc:0.927922
[8]	train	auc:0.963121	test	auc:0.927406
[9]	train	auc:0.963852	test	auc:0.926375
[10]	train	auc:0.965559	test	auc:0.925545
[11]	train	auc:0.967358	test	auc:0.925664
[12]	train	auc:0.969143	test	auc:0.924629
[13]	train	auc:0.970182	test	auc:0.924950
[14]	train	auc:0.970923	test	auc:0.924509
[15]	train	auc:0.973319	test	auc:0.923537
[16]	train	auc:0.974383	test	auc:0.923199
[17]	train	auc:0.975294	test	auc:0.923111

Stopping. Best iteration:
[7] train-auc:0.961152test-auc:0.927922

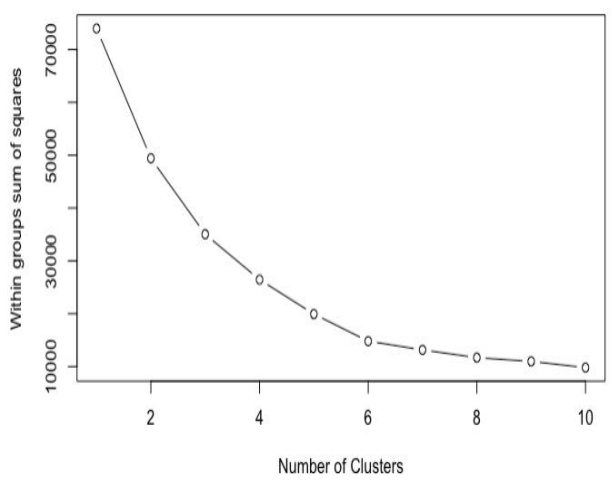
7.6) Clustering techniques implemented:

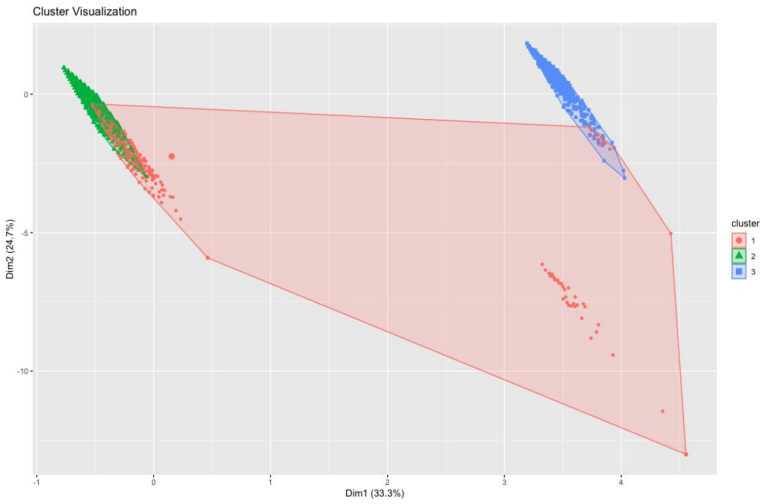
To understand the data aggregation we implemented clustering methods as below. Here we have taken into account the only the "Browser", "Region", "TrafficType", "VisitorType" features.

7.6.1) **K-means clustering:**

This is widely used unsupervised learning algorithm that partitions a dataset into K-clusters based on the similarity between data points. K-means clustering is a simple and fast algorithm which is suitable for both small and large datasets.

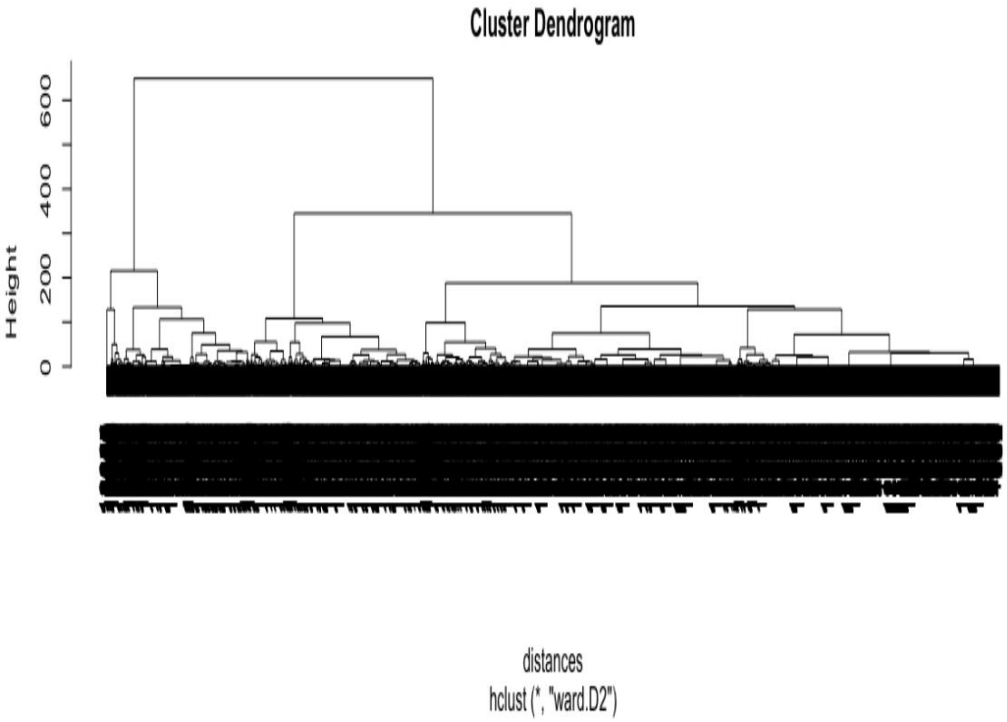
From the elbow plot we have selected 3 clusters and looks like the cluster one intersects both the clusters.





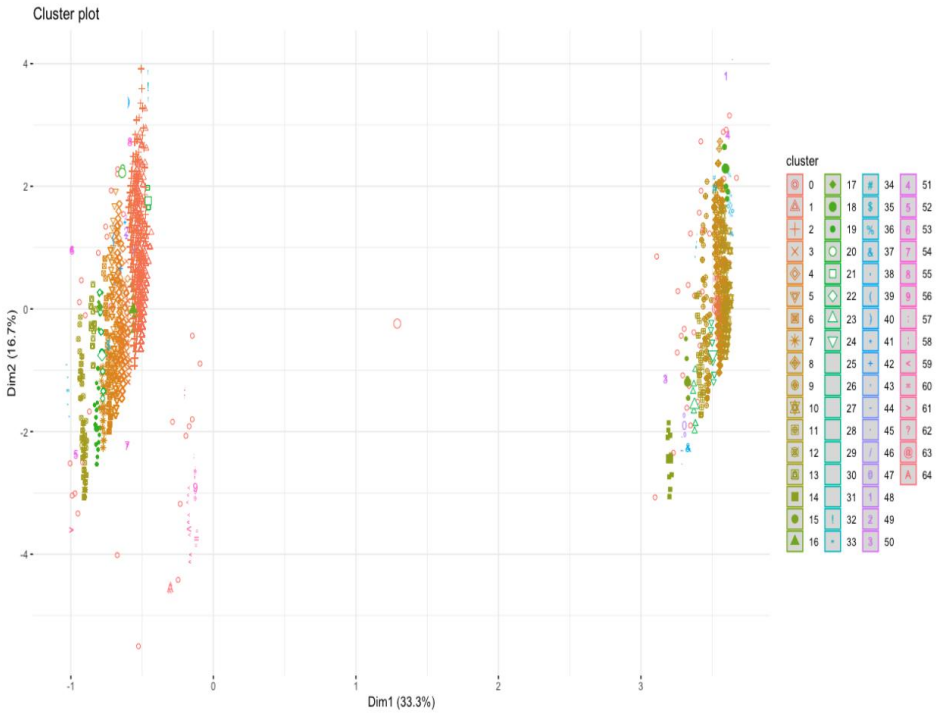
7.6.2) **Hierarchical clustering:**

This is another unsupervised learning algorithm that clusters data points into a tree-like structure based on the similarity between them. Hierarchical clustering can be either agglomerative (bottom-up) or divisive (top-down).



7.6.3) **DBSCAN (Density-Based Spatial Clustering of Applications with Noise):**

As name says this is a density-based clustering algorithm that groups data points into clusters based on their density. DBSCAN is particularly useful for datasets with irregular shapes and noises. But our dataset was mostly without the noises and hence the results were as below,



8) Conclusion

- Analysing number of page visit of 3 different page categories it clearly says that customers are interested more in Product related pages rather than knowing information of the product in detail.
- Revenue is generated by the customers who visit the product page and spend more time on it, which intuitively mean whom ever spends more time on administrative and informational page will only hop around rather than end up buying.
- Discounts can be given to the ones who spend more time on Product related page.
- There is no noticeable disparity in Bounce Rates between customers who made a purchase and those who did not.
- However, customers who ended up making a purchase had lower Exit Rates on average, indicating that they were more likely to remain on the website's pages.
- Additionally, customers who did not make a purchase had significantly lower Page Values, suggesting that they spent less time on related pages.

9) Bibliography

[1]. https://jurnal-ppi.kominfo.go.id/index.php/jppi/article/view/341

[2]. Real-time prediction of online shoppers' purchasing intention using multilayer perceptron and LSTM recurrent neural networks [https://link.springer.com/article/10.1007/s00521-018-3523-0]

[3]. Data Clustering: A Review [https://dl.acm.org/doi/pdf/10.1145/331499.331504]

[4]. A Comparison Study of Credit Card Fraud Detection: Supervised versus Unsupervised [https://arxiv.org/pdf/1904.10604.pdf]

[5]. Real-Time Prediction of Online Shoppers Purchasing Intention Using Random Forest [https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7256375/]