Problem Definition

Kira Plastinina is a Russian brand that is sold through a defunct chain of retail stores in Russia, Ukraine, Kazakhstan, Belarus, China, Philippines, and Armenia. The brand's Sales and Marketing team would like to understand their customer's behavior from data that they have collected over the past year. More specifically, they would like to learn the characteristics of customer groups.

Data Sourcing

Data to be used for analysis is found here . The dataset consists of 10 numerical and 8 categorical attributes. The 'Revenue' attribute can be used as the class label.

Experimental design taken

- Problem Definition
- Data Sourcing
- Check the Data
- Perform Data Cleaning
- Perform Exploratory Data Analysis (Univariate, Bivariate & Multivariate)
- Implement the Solution
- Challenge the Solution
- Follow up Questions

Appropriateness of the available data

The data provided for analysis is very appropriate since it contains different variables which will help in answering our research question.

Checking the Dataset.

```
#Reading the dataset.
#Previewing the first 6 columns of the dataset.
library("data.table")
data = fread('http://bit.ly/EcommerceCustomersDataset')
head(data)
```

```
#Previewing the last 6 columns of the dataset.
tail(data)
```

```
#Checking the number of columns in the dataset.
ncol(data)
```

The dataset has 18 columns.

```
#Checking the number of rows in the dataset
nrow(data)
```

The dataset has 12330 rows.

```
#Checking the dimensions of the dataset.
dim(data)
```

```
#Checking the length of the dataset.
length(data)
```

```
#checking the summary of the dataset.
summary(data)
```

```
#Checking the structure of the dataset
str(data)
```

```
Classes ‘data.table’ and 'data.frame':  12330 obs. of  18 variables:
 $ Administrative         : int  0 0 0 0 0 0 0 1 0 0 ...
 $ Administrative_Duration: num  0 0 -1 0 0 0 -1 -1 0 0 ...
 $ Informational          : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Informational_Duration : num  0 0 -1 0 0 0 -1 -1 0 0 ...
 $ ProductRelated         : int  1 2 1 2 10 19 1 1 2 3 ...
 $ ProductRelated_Duration: num  0 64 -1 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_Vis
 $ Weekend                : logi  FALSE FALSE FALSE FALSE TRUE FALSE ...
 $ Revenue                : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
 - attr(*, ".internal.selfref")=<externalptr>
```

```
#Listing variables in our dataset.
names(data)
```

```
#Checking the column of the evenue column.
class(data$Revenue)
```

Cleaning the dataset.

Data Completeness.

```
#Checking for null values.
#Finding the total missing values in each column
colSums(is.na(data))
```

```
#Omitting the rows with null values
data_clean <-  na.omit(data)
colSums(is.na(data_clean))
```

```
#Checking the dimensions of the new dataset without null values.
dim(data_clean)
```

Data Consistency

```
#Checking for duplicated rows
anyDuplicated(data_clean)
```

The dataset has 159 duplicates.

```
# showing these unique items and assigning to a variable unique_items below
data_unique <- data_clean[!duplicated(data_clean), ]
data_unique
```

```
#Checking to see if duplicates have been dropped.
dim(data_unique)
```

```
#Changing the column names to lowercase
names(data_unique)[names(data_unique) == "Administrative"] <- "administrative"
names(data_unique)[names(data_unique) == "Administrative_Duration"] <- "administrative_durati
names(data_unique)[names(data_unique) == "Informational"] <- "informational"
names(data_unique)[names(data_unique) == "Informational_Duration"] <- "informational_duratior
names(data_unique)[names(data_unique) == "ProductRelated"] <- "productrelated"
names(data_unique)[names(data_unique) == "ProductRelated_Duration"] <- "productrelated_durati
names(data_unique)[names(data_unique) == "BounceRates"] <- "bouncerates"
names(data_unique)[names(data_unique) == "ExitRates"] <- "exitrates"
names(data_unique)[names(data_unique) == "PageValues"] <- "pagevalues"
names(data_unique)[names(data_unique) == "SpecialDay"] <- "specialday"
names(data_unique)[names(data_unique) == "Month"] <- "month"
names(data_unique)[names(data_unique) == "OperatingSystems"] <- "operatingsystems"
```

```
names(data_unique)[names(data_unique) == "Browser"] <- "browser"
names(data_unique)[names(data_unique) == "Region"] <- "region"
names(data_unique)[names(data_unique) == "TrafficType"] <- "traffictype"
names(data_unique)[names(data_unique) == "VisitorType"] <- "visitortype"
names(data_unique)[names(data_unique) == "Weekend"] <- "weekend"
names(data_unique)[names(data_unique) == "Revenue"] <- "revenue"
head(data_unique)
```

```
#Checking if the column names have changed to lower case.
names(data_unique)
```

Data Validity

```
#Converting the categorical columns to their appropriate data types.
data_unique$operatingsystems <- as.factor(data_unique$operatingsystems)
data_unique$browser <- as.factor(data_unique$browser)
data_unique$region <- as.factor(data_unique$region)
data_unique$traffictype <- as.factor(data_unique$traffictype)
data_unique$weekend <- as.factor(data_unique$weekend)
data_unique$revenue <- as.factor(data_unique$revenue)
data_unique$month <- as.factor(data_unique$month)
data_unique$visitortype <- as.factor(data_unique$visitortype)
```

```
#Checking if the data types have been corrected.
str(data_unique)
```

```
Classes ‘data.table’ and 'data.frame':  12199 obs. of  18 variables:
 $ administrative         : int  0 0 0 0 0 0 0 1 0 0 ...
 $ administrative_duration: num  0 0 -1 0 0 0 -1 -1 0 0 ...
```

```
 $ informational          : int  0 0 0 0 0 0 0 0 0 0 ...
 $ informational_duration : num  0 0 -1 0 0 0 -1 -1 0 0 ...
 $ productrelated         : int  1 2 1 2 10 19 1 1 2 3 ...
 $ productrelated_duration: num  0 64 -1 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/ 10 levels "Aug","Dec","Feb",..: 3 3 3 3 3 3 3 3
 $ 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                : Factor w/ 2 levels "FALSE","TRUE": 1 1 1 1 2 1 1 2 1 1 ...
 $ revenue                : Factor w/ 2 levels "FALSE","TRUE": 1 1 1 1 1 1 1 1 1 1 ...
 - attr(*, ".internal.selfref")=<externalptr>
```

```
#Checking for outliers in the numerical variables.
options(repr.plot.width = 10, repr.plot.height = 10)
boxplot(data_unique$administrative, main = 'Administrarive')
boxplot(data_unique$administrative_duration, main = 'Administrative_duration')
boxplot(data_unique$informational, main = 'Informational')
boxplot(data_unique$informational_duration, main = 'Informational_duration')
boxplot(data_unique$productrelated, main = 'Productrelated')
boxplot(data_unique$productrelated_duration, main = 'Productrelated_duration')
boxplot(data_unique$bouncerates, main = 'Bouncerates')
boxplot(data_unique$exitrates, main = 'Exitrates')
boxplot(data_unique$pagevalues, main = 'Pagevalues')
boxplot(data_unique$specialday, main = 'Specialday')
```

We have many outliers in our dataset.

```
#Checking for anomalies in the operating system column
levels(data_unique$operatingsystems)
```

```
There are no anomalies in this column.
```

```
#Checking for anomalies in the Browser column
levels(data_unique$browser)
```

There are no anomalies.

```
#Checking for anomalies in the region column
levels(data_unique$region)
```

There are no anomalies.

```
#Checking for anomalies in the traffictype column
levels(data_unique$traffictype)
```

There are no anomalies.

```
#Checking for anomalies in the weekend column
levels(data_unique$weekend)
```

There are no anomalies.

```
#Checking for anomalies in the revenue column
levels(data_unique$revenue)
```

There are no anomalies.

```
#Checking for anomalies in the month column
levels(data_unique$month)
```

There are no anomalies.

```
#Checking for anomalies in the visitortype column
levels(data_unique$visitortype)
```

There are no anomalies.

Univariate Analysis

Measures of central tendency and Measures of dispersion.

```
#Checking the summary of different variables in the dataset.
summary(data_unique)
```

```
#Checking the description of different variables in the dataset.
install.packages("psych")
```

```
library("psych")
describe(data_unique)
```

Univariate graphs

```
#Frequency table of the revenue column.
```

```
revenue <- data_unique$revenue
revenue_frequency <- table(revenue)
revenue_frequency
```

```
#Bar graph representing revenue frequency.
#options(repr.plot.width = 10, repr.plot.height = 10)
barplot(c(revenue_frequency), main="A barplot representing the Revenue column.",
        xlab="Revenue",
        ylab="frequency",
        cex.main=2, cex.lab=1.7,cex.sub=1.2,
        width=c(30,30),
        col=c("blue","orange"))
```

From the frequency table and the bar chart we can observe that there are more false revenues than true revenues.

```
#Frequency table of the operating systems column.
operatingsystems <- data_unique$operatingsystems
os_frequency <- table(operatingsystems)
os_frequency
```

```
#Bar graph representing operating systems frequency.
#options(repr.plot.width = 10, repr.plot.height = 10)
barplot(c(os_frequency), main="A barplot representing the Operating System column.",
        xlab="Operating system",
        ylab="frequency",
        cex.main=2, cex.lab=1.7,cex.sub=1.2,
        width=c(30,30),
        col=c("blue","orange"))
```

Operation system 2 is the most used operating system followed by operation system 1 and then 3.

```
#Frequency table of the browser column.
browser <- data_unique$browser
browser_frequency <- table(browser)
browser_frequency
```

```
#Bar graph representing browser frequency.
#options(repr.plot.width = 10, repr.plot.height = 10)
barplot(c(browser_frequency), main="A barplot representing the browser column.",
        xlab="browser",
        ylab="frequency",
        cex.main=2, cex.lab=1.7,cex.sub=1.2,
        width=c(30,30),
        col=c("blue","orange"))
```

Browser 2 is the most frequently used browser followed by browser 1.

```
#Frequency table of the region column.
region <- data_unique$region
region_frequency <- table(region)
region_frequency
```

```
#Bar graph representing region frequency.
#options(repr.plot.width = 10, repr.plot.height = 10)
barplot(c(region_frequency), main="A barplot representing the region column.",
        xlab="Region",
        ylab="frequency",
        cex.main=2, cex.lab=1.7,cex.sub=1.2,
        width=c(30,30),
        col=c("blue","orange"))
```

Region 1 is the most occurring region while region 5 is the least occuring region.

```
#Frequency table of the traffic type column.
traffictype <- data_unique$traffictype
tt_frequency <- table(traffictype)
tt_frequency
```

```
#Bar graph representing Traffic type frequency.
#options(repr.plot.width = 10, repr.plot.height = 10)
barplot(c(tt_frequency), main="A barplot representing the traffic type column.",
        xlab="traffic type",
        ylab="frequency",
        cex.main=2, cex.lab=1.7,cex.sub=1.2,
        width=c(30,30),
        col=c("blue","orange"))
```

Traffic type two is the most frequent used followed by traffic type 1.

```
#Frequency table of the weekend column.
weekend <- data_unique$weekend
weekend_frequency <- table(weekend)
weekend_frequency
```

```
#Bar graph representing Weekend frequency.
#options(repr.plot.width = 10, repr.plot.height = 10)
barplot(c(weekend_frequency), main="A barplot representing the Weekend column.",
        xlab="Weekend",
        ylab="frequency",
        cex.main=2, cex.lab=1.7,cex.sub=1.2,
        width=c(30,30),
        col=c("blue","orange"))
```

Weekdays are more than weekends.

```
#Frequency table of the month column
month <- data_unique$month
month_frequency <- table(month)
month_frequency
```

```
#Bar graph representing Month frequency.
#options(repr.plot.width = 10, repr.plot.height = 10)
barplot(c(month_frequency), main="A barplot representing the Month column.",
        xlab="Month",
        ylab="frequency",
        cex.main=2, cex.lab=1.7,cex.sub=1.2,
        width=c(30,30),
        col=c("blue","orange"))
```

The most frequent month was May followed by November. The least frequent month is February.

```
#Frequency table of the visitor type column.
visitortype <- data_unique$visitortype
visitortype_frequency <- table(visitortype)
visitortype_frequency
```

```
#Bar graph representing Visitor type frequency.
#options(repr.plot.width = 10, repr.plot.height = 10)
barplot(c(visitortype_frequency), main="A barplot representing the visitortype column.",
        xlab="Visitor type",
        ylab="frequency",
        cex.main=2, cex.lab=1.7,cex.sub=1.2,
        width=c(30,30),
        col=c("blue","orange"))
```

Most visitors were returning visitors while the least were the other visitors.

```
#Histograms of all the numerical variables
options(repr.plot.width = 10, repr.plot.height = 10)
hist(data_unique$administrative)
hist(data_unique$administrative_duration)
hist(data_unique$informational)
hist(data_unique$informational_duration)
hist(data_unique$productrelated)
hist(data_unique$productrelated_duration)
hist(data_unique$bouncerates)
hist(data_unique$exitrates)
hist(data_unique$pagevalues)
hist(data_unique$specialday)
```

The histograms representing the numerical columns are all skewed to the right.

Bivariate Analysis.

```
#Specifying the numeric variables.
```

```
#Checking the correlation of numeric variables
data_numeric <- data_unique[,1:10]
cor(data_numeric)
```

```
#A heat map to visualize correlations.
options(repr.plot.width = 15, repr.plot.height = 10)
install.packages("ggcorrplot")
library(ggcorrplot)
corr_data <- cor(data_numeric)
ggcorrplot(round(corr_data, 2) ,lab = T,type = 'lower')
```

The highly correlated variables in our case are, administrative and administrative duration, informational and informational duration, product related and product related duration and lastly exitrates and bounce rates.

```
#Checking the covariance of the numerical variables.
cov(data_numeric)
```

```
#Administrative Vs Revenue
options(repr.plot.width = 10, repr.plot.height = 10)
library(ggplot2)
ggplot(data_unique,
       aes(x = administrative,
           fill = revenue, color = revenue)) +
  geom_density(alpha = 0.4) +
  labs(title = "Administrative distribution by Revenue")
```

```
#Administrative duration Vs Revenue
library(ggplot2)
ggplot(data_unique,
       aes(x = administrative_duration,
           fill = revenue, color = revenue)) +
  geom_density(alpha = 0.4) +
  labs(title = "Administrative_duration distribution by Revenue")
```

```
#Informational duration Vs Revenue
library(ggplot2)
ggplot(data_unique,
       aes(x = informational,
           fill = revenue, color = revenue)) +
  geom_density(alpha = 0.4) +
  labs(title = "Informational distribution by Revenue")
```

```
#Informational duration Vs Revenue
library(ggplot2)
ggplot(data_unique,
       aes(x = informational_duration,
           fill = revenue, color = revenue)) +
  geom_density(alpha = 0.4) +
  labs(title = "Informational duration distribution by Revenue")
```

```
#Product Related Vs Revenue
library(ggplot2)
ggplot(data_unique,
       aes(x = productrelated,
           fill = revenue, color = revenue)) +
  geom_density(alpha = 0.4) +
  labs(title = "Product Related distribution by Revenue")
```

```
#Product related duration Vs Revenue
library(ggplot2)
ggplot(data_unique,
       aes(x = productrelated_duration,
           fill = revenue, color = revenue)) +
  geom_density(alpha = 0.4) +
  labs(title = "Product related duration distribution by Revenue")
```

```
#Bounce rates Vs Revenue
library(ggplot2)
ggplot(data_unique,
       aes(x = bouncerates,
           fill = revenue, color = revenue)) +
  geom_density(alpha = 0.4) +
  labs(title = "Bounce rates distribution by Revenue")
```

```
#Exit rates Vs Revenue
library(ggplot2)
ggplot(data_unique,
       aes(x = exitrates,
           fill = revenue, color = revenue)) +
  geom_density(alpha = 0.4) +
  labs(title = "Exit rates distribution by Revenue")
```

```
#Page values Vs Revenue.
library(ggplot2)
ggplot(data_unique,
       aes(x = pagevalues,
           fill = revenue, color = revenue)) +
  geom_density(alpha = 0.4) +
  labs(title = "Page values distribution by Revenue")
```

```
#Special day Vs Revenue.
library(ggplot2)
ggplot(data_unique,
       aes(x = specialday,
           fill = revenue)) +
           geom_density(alpha = 0.4) +
  labs(title = "Special day distribution by Revenue")
```

```
#Operating system Vs Revenue
# stacked bar chart
library(ggplot2)
ggplot(data_unique,
       aes(x = operatingsystems,
           fill = revenue)) +
  geom_bar(position = "stack")
```

```
#Browser Vs Revenue
# stacked bar chart
ggplot(data_unique,
       aes(x = browser,
           fill = revenue)) +
  geom_bar(position = "stack")
```

```
#Region Vs Revenue
# stacked bar chart
ggplot(data_unique,
       aes(x = region,
           fill = revenue)) +
  geom_bar(position = "stack")
```

```
#Traffic type Vs Revenue.
# stacked bar chart
ggplot(data_unique,
       aes(x = traffictype,
           fill = revenue)) +
  geom_bar(position = "stack")
```

```
#Weekend Vs Revenue
# stacked bar chart
ggplot(data_unique,
       aes(x = weekend,
           fill = revenue)) +
  geom_bar(position = "stack")
```

```
#Month Vs Revenue
# stacked bar chart
ggplot(data_unique,
       aes(x = month,
           fill = revenue)) +
  geom_bar(position = "stack")
```

```
#Visitor type Vs Revenue
# stacked bar chart
ggplot(data_unique,
       aes(x = visitortype,
```

```
          fill = revenue)) +
  geom_bar(position = "stack")
```

Multivariate Analysis

```
#A multivariate plot showing the relationship between revenue, administrative and weekend.
library(ggplot2)
ggplot(data_unique, aes(fill=revenue, y=administrative, x=weekend)) +
    geom_bar(position="dodge", stat="identity")
```

```
#A multivariate plot showing the relationship between revenue, administrative and visitor typ
library(ggplot2)
ggplot(data_unique, aes(fill=revenue, y=administrative, x=visitortype)) +
    geom_bar(position="dodge", stat="identity")
```

```
#A multivariate plot showing the relationship between revenue, informational and weekend.
library(ggplot2)
ggplot(data_unique, aes(fill=revenue, y=informational, x=weekend)) +
    geom_bar(position="dodge", stat="identity")
```

```
#A multivariate plot showing the relationship between revenue, informational and visitopr typ
```

```
library(ggplot2)
ggplot(data_unique, aes(fill=revenue, y=informational, x=visitortype)) +
    geom_bar(position="dodge", stat="identity")
```

```
#A multivariate plot showing the relationship between revenue, productrelated and weekend.
library(ggplot2)
ggplot(data_unique, aes(fill=revenue, y=productrelated, x=weekend)) +
    geom_bar(position="dodge", stat="identity")
```

```
#A multivariate plot showing the relationship between revenue, productrelated and visitor typ
library(ggplot2)
ggplot(data_unique, aes(fill=revenue, y=productrelated, x=visitortype)) +
    geom_bar(position="dodge", stat="identity")
```

```
#A multivariate plot showing the relationship between revenue, bouncerates and weekend.
library(ggplot2)
ggplot(data_unique, aes(fill=revenue, y=bouncerates, x=weekend)) +
    geom_bar(position="dodge", stat="identity")
```

```
#A multivariate plot showing the relationship between revenue, bounce rates and visitor type.
library(ggplot2)
```

```
ggplot(data_unique, aes(fill=revenue, y=bouncerates, x=visitortype)) +
    geom_bar(position="dodge", stat="identity")
```

```
#A multivariate plot showing the relationship between revenue, exitrates and weekend.
library(ggplot2)
ggplot(data_unique, aes(fill=revenue, y=exitrates, x=weekend)) +
    geom_bar(position="dodge", stat="identity")
```

```
#A multivariate plot showing the relationship between revenue, exitrates and visitor type.
library(ggplot2)
ggplot(data_unique, aes(fill=revenue, y=exitrates, x=visitortype)) +
    geom_bar(position="dodge", stat="identity")
```

```
#A multivariate plot showing the relationship between revenue, page values and weekend.
library(ggplot2)
ggplot(data_unique, aes(fill=revenue, y=pagevalues, x=weekend)) +
    geom_bar(position="dodge", stat="identity")
```

```
#A multivariate plot showing the relationship between revenue, administrative and weekend.
library(ggplot2)
```

```
ggplot(data_unique, aes(fill=revenue, y=pagevalues, x=visitortype)) +
    geom_bar(position="dodge", stat="identity")
```

```
#A multivariate plot showing the relationship between revenue, specialday and weekend.
library(ggplot2)
ggplot(data_unique, aes(fill=revenue, y=specialday, x=weekend)) +
    geom_bar(position="dodge", stat="identity")
```

```
#A multivariate plot showing the relationship between revenue, visitor type and administrati
library(ggplot2)
ggplot(data_unique, aes(fill=revenue, y=administrative, x=visitortype)) +
    geom_bar(position="dodge", stat="identity")
```

Reduction

```
#Checking the redundant variables.
install.packages('caret', dependencies = TRUE)
library(caret)
findCorrelation(corr_data,cutoff = .6, verbose = TRUE, names = TRUE)
```

The highly correlated variables that need to be dropped to avoid redundancy are productrelated duration, administrative, exitrates and informational.

```
#Dropping the redundant variables.
data_unique = subset(data_unique,select = -c(productrelated_duration, administrative, exitrat
head(data_unique)
```

Modelling

Supervised Models

```
#Encoding the revenue, weekend, month and visitortype columns.
data_unique$weekend <- ifelse(data_unique$weekend == "TRUE",1,0)
data_unique$month <- as.numeric(data_unique$month)
data_unique$visitortype <- as.numeric(data_unique$visitortype)
data_unique$operatingsystems <- as.numeric(data_unique$operatingsystems)
data_unique$browser <- as.numeric(data_unique$browser)
data_unique$region <- as.numeric(data_unique$region)
data_unique$traffictype <- as.numeric(data_unique$traffictype)

head(data_unique)
```

```
# Normalizing the dataset so that no particular attribute
# has more impact on clustering algorithm than others.
normalize <- function(x){
  return ((x-min(x)) / (max(x)-min(x)))
}
data_unique$administrative_duration<- normalize(data_unique$administrative_duration)
data_unique$informational_duration<- normalize(data_unique$informational_duration)
data_unique$productrelated<- normalize(data_unique$productrelated)
data_unique$bouncerates<- normalize(data_unique$bouncerates)
data_unique$pagevalues<- normalize(data_unique$pagevalues)
data_unique$specialday<- normalize(data_unique$specialday)
data_unique$month<- normalize(data_unique$month)
data_unique$operatingsystems<- normalize(data_unique$operatingsystems)
data_unique$browser<- normalize(data_unique$browser)
data_unique$region<- normalize(data_unique$region)
data_unique$traffictype<- normalize(data_unique$traffictype)
data_unique$visitortype<- normalize(data_unique$visitortype)
data_unique$weekend<- normalize(data_unique$weekend)
```

```
#Randomizing the data.
shuffle_index <- sample(1:nrow(data_unique))
head(shuffle_index)
```

```
data_unique <- data_unique[shuffle_index, ]
head(data_unique)
```

```
#Splitting the dataset in to train and test using 70-30 splitts.
intrain <- createDataPartition(y = data_unique$revenue, p = 0.7, list = FALSE)
data_train <- data_unique[intrain,]
data_test <- data_unique[-intrain,]
```

```
#Checking the dimensions of the splitts.
dim(data_train)
dim(data_test)
```

```
# checking the dimensions of our split
prop.table(table(data_unique$revenue)) * 100
prop.table(table(data_train$revenue)) * 100
prop.table(table(data_test$revenue)) * 100
```

KNN

```
# splitting into train and test sets without the target variable
train <- data_train[, -14]
test <- data_test[, -14]

# storing the training and test sets' target variable
train_rev <- data_train[, data_train$revenue]
test_rev <- data_test[, data_test$revenue]


#Checking the dimensions of the train and test splitts.
dim(train)
dim(test)
#Checking the length of the train and test target variables.
length(train_rev)
length(test_rev)
```

```
#Installing the necessary libraries.
library(class)
```

```
require(class)
model <- knn(train= train,test=test, ,cl= train_rev,k=13)
table(factor(model))
knn_table <- table(test_rev,model)
knn_table
```

```
# Check prediction against actual value in tabular form for k=13
table(model ,test_rev)
```

```
# calculating accuracy
accuracy <- sum(diag(knn_table)/(sum(rowSums(knn_table)))) * 100
print(paste("KNN accuracy score:", accuracy))
```

```
[1] "KNN accuracy score: 85.7884667942061"
```

```
set.seed(400)
ctrl <- trainControl(method="repeatedcv",repeats = 3)
knnFit <- train(revenue ~ ., data = data_train, method = "knn", trControl = ctrl, preProcess
knnFit
```

```
plot(knnFit)
```

```
library(class)
require(class)
model <- knn(train= train,test=test, ,cl= train_rev,k=15)
table(factor(model))
knn_table <- table(test_rev,model)
knn_table
```

```
# calculating accuracy
accuracy <- sum(diag(knn_table)/(sum(rowSums(knn_table)))) * 100
print(paste("KNN accuracy score:", accuracy))
```

```
[1] "KNN accuracy score: 85.6791473080077"
```

The accuracy of the KNN model has an accuracy of 85.78% but after tuning the parameters the accuracy reduces to 85.67%

Decision Trees

```
#Installing libraries
install.packages('rpart')
install.packages('caret')
install.packages('rpart.plot')
install.packages('rattle')

#Loading libraries
library(rpart,quietly = TRUE)
library(caret,quietly = TRUE)
library(rpart.plot,quietly = TRUE)
library(rattle)
#Fitting the model
#data splicing
set.seed(12345)
train <- sample(1:nrow(data_unique),size = ceiling(0.80*nrow(data_unique)),replace = FALSE)
# training set
dt_train <- data_unique[train,]
# test set
dt_test <- data_unique[-train,]
```

```
Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)
```

```
Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)

Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)

Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)

also installing the dependency ‘XML’


Loading required package: tibble

Loading required package: bitops

Rattle: A free graphical interface for data science with R.
Version 5.4.0 Copyright (c) 2006-2020 Togaware Pty Ltd.
Type 'rattle()' to shake, rattle, and roll your data.
```

```
#Checking the dimensions of the splitts.
dim(dt_test)
dim(dt_train)
```

```
# penalty matrix
penalty.matrix <- matrix(c(0,1,10,0), byrow=TRUE, nrow=2)


# building the classification tree with rpart
tree <- rpart(revenue~.,
data=dt_train,
parms = list(loss = penalty.matrix),
method = "class")


# Visualize the decision tree with rpart.plot
rpart.plot(tree, nn=TRUE)
```

```
#Testing the model
pred <- predict(object=tree,dt_test[,-14],type="class")
pred
```

```
#Calculating accuracy
t <- table(dt_test$revenue,pred)
confusionMatrix(t)
```

From our decision trees we get an accuracy of 83.89%.

SVM

```
#Checking the structure of the data.
str(data_unique)
```

```
Classes ‘data.table’ and 'data.frame':  12199 obs. of  14 variables:
 $ administrative_duration: num  0.009118 0.048004 0.038091 0.048974 0.000294 ...
 $ informational_duration : num  0.006862 0.000392 0.000392 0.000392 0.000392 ...
 $ productrelated         : num  0.14043 0.00851 0.01986 0.06667 0.01986 ...
 $ bouncerates            : num  0.0098 0.0714 0 0.0435 0.2857 ...
 $ pagevalues             : num  0.0202 0 0 0.0163 0 ...
 $ specialday             : num  0 0 0 0 0 0 0 0 0 0 ...
 $ month                  : num  0.667 0.333 0.667 0.111 0.667 ...
 $ operatingsystems       : num  0.143 0.286 0 0.143 0.286 ...
 $ browser                : num  0.0833 0.0833 0.5833 0.0833 0.0833 ...
 $ region                 : num  0 0.25 0.375 0.75 0 0.25 1 0.25 0.25 0.375 ...
 $ traffictype            : num  0 0 0.1579 0.0526 0.6316 ...
 $ visitortype            : num  1 1 1 1 1 0 1 1 1 0 ...
 $ weekend                : num  1 0 1 0 0 0 0 0 0 0 ...
 $ revenue                : Factor w/ 2 levels "FALSE","TRUE": 1 1 1 1 1 2 1 1 1 1 ...
 - attr(*, ".internal.selfref")=<externalptr>
```

```
#Fitting SVM to the Training set
install.packages('e1071')
library(e1071)
install.packages('caret')
library('caret')
```

```
Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)

Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)
```

```
#Splitting the dataset to train and test using 70-30 splitts.
intrain <- createDataPartition(y = data_unique$revenue, p= 0.7, list = FALSE)
training <- data_unique[intrain,]
testing <- data_unique[-intrain,]
```

```
#Checking the dimensions of the splitts.
dim(training);
dim(testing);
```

```
#Checking if there is any missing data.
anyNA(data)
```

```
#Changing the target variable to factor.
training[["revenue"]] = factor(training[["revenue"]])
```

```
trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)
```

```
#Training the model.
svm_Linear <- train(revenue ~., data = training, method = "svmLinear",
trControl=trctrl,
preProcess = c("center", "scale"),
tuneLength = 10)
```

```
svm_Linear
```

```
#Making predictions
test_pred <- predict(svm_Linear, newdata = testing)
test_pred
```

```
#Computing the confusion matrix
confusionMatrix(table(test_pred, testing$revenue))
```

```
#Hyperparameter tuning.
grid <- expand.grid(C = c(0,0.01, 0.05, 0.1, 0.25, 0.5, 0.75, 1, 1.25, 1.5, 1.75, 2,5))
svm_Linear_Grid <- train(revenue ~., data = training, method = "svmLinear",
trControl=trctrl,
preProcess = c("center", "scale"),
tuneGrid = grid,
tuneLength = 10)
```

```
svm_Linear_Grid
plot(svm_Linear_Grid)
```

```
#Using the best parameters to predict
test_pred_grid <- predict(svm_Linear_Grid, newdata = testing)
test_pred_grid
```

```
#Confusion matrix
confusionMatrix(table(test_pred_grid, testing$revenue))
```

The SVM accuracy is 88.9% before and after tuning the parameters.

Naive bayes

```
#Loading required packages
install.packages('tidyverse')
library(tidyverse)
install.packages('ggplot2')
library(ggplot2)
install.packages('caret')
library(caret)
install.packages('caretEnsemble')
library(caretEnsemble)
install.packages('psych')
library(psych)
install.packages('Amelia')
library(Amelia)
install.packages('mice')
library(mice)
install.packages('GGally')
library(GGally)
install.packages('rpart')
library(rpart)
install.packages('randomForest')
library(randomForest)
```

```
Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)

── Attaching packages ─────────────────────────────────── tidyverse 1.3.1 ──

✓ tidyr   1.1.3     ✓ dplyr   1.0.5
✓ readr   1.4.0     ✓ stringr 1.4.0
✓ purrr   0.3.4     ✓ forcats 0.5.1

── Conflicts ────────────────────────────────────── tidyverse_conflicts() ──
x ggplot2::%+%()      masks psych::%+%()
x ggplot2::alpha()    masks psych::alpha()
x dplyr::between()    masks data.table::between()
x dplyr::filter()     masks stats::filter()
```

```
✖ dplyr::first()     masks data.table::first()
✖ dplyr::lag()       masks stats::lag()
✖ dplyr::last()      masks data.table::last()
✖ purrr::lift()      masks caret::lift()
✖ purrr::transpose() masks data.table::transpose()

Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)

Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)

Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)

also installing the dependencies ‘pbapply’, ‘gridExtra’



Attaching package: ‘caretEnsemble’


The following object is masked from ‘package:ggplot2’:

    autoplot


Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)

Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)

also installing the dependency ‘RcppArmadillo’


Loading required package: Rcpp

##
## Amelia II: Multiple Imputation
## (Version 1.8.0, built: 2021-05-26)
## Copyright (C) 2005-2021 James Honaker, Gary King and Matthew Blackwell
## Refer to http://gking.harvard.edu/amelia/ for more information
```

```
#Building a model
#split data into training and test data sets
indxTrain <- createDataPartition(y = data_unique$revenue,p = 0.75,list = FALSE)
training <- data_unique[indxTrain,]
testing <- data_unique[-indxTrain,]


#Check dimensions of the split
prop.table(table(data_unique$revenue)) * 100
prop.table(table(training$revenue)) * 100
prop.table(table(testing$revenue)) * 100
```

```
#create objects x which holds the predictor variables and y which holds the response variabl
x = training[,-14]
y = training$revenue
```

```
library(e1071)
```

```
install.packages("klaR")
library(klaR)
model = train(x,y,'nb',trControl=trainControl(method='cv',number=10))

model
```

```
#Model Evaluation
#Predict testing set
Predict <- predict(model,newdata = testing )
#Get the confusion matrix to see accuracy value and other parameter values
confusionMatrix(Predict, testing$revenue)
```

```
#Plot Variable performance
X <- varImp(model)
plot(X)
```

The accuracy of the Naive bayes model is 87.04% The most important variables that help in predicting whether revenue is true or false are pagevalues being the first followed by product related and administrative duration follow respectively.

Unsupervised Models.

K-Means - Clustering

```
#Splitting the data
data_new <- data_unique[, c(1:13)]
data_class <- data_unique$revenue
head(data_new)
```

```
#Previewing the data
head(data_class)
```

```
#installing packages that will enable us to compute the number of clusters
#loading the packages
pkgs <- c("factoextra",  "NbClust")
install.packages(pkgs)
library(factoextra)
library(NbClust)
```

```
Installing packages into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)
```

```
fviz_nbclust(data_new, FUNcluster = kmeans, method = c("silhouette", "wss", "gap_stat"))
# Elbow method
fviz_nbclust(data_new, kmeans, method = "wss") +
    geom_vline(xintercept = 4, linetype = 2)+
  labs(subtitle = "Elbow method")
```

```
# Silhouette method
fviz_nbclust(data_new, kmeans, method = "silhouette")+
  labs(subtitle = "Silhouette method")

# Gap statistic
# nboot = 50 to keep the function speedy.
# recommended value: nboot= 500 for your analysis.
# Use verbose = FALSE to hide computing progression.
set.seed(123)
fviz_nbclust(data_new, kmeans, nstart = 25,  method = "gap_stat", nboot = 50)+
  labs(subtitle = "Gap statistic method")
```

```
# Applying the K-means clustering algorithm with no. of centroids(k)=3
# ---
#
result<- kmeans(data_new,3)

# Previewing the no. of records in each cluster
#
result$size
```

```
# Getting the value of cluster center datapoint value(6 centers for k=6)
# ---
#
result$centers
```

```
# Getting the cluster vector that shows the cluster where each record falls
# ---
#
result$cluster
```

```
# Verifying the results of clustering
# ---
#
par(mfrow = c(1,1), mar = c(5,4,2,2))

# Plotting to see how different variable data points have been distributed in clusters
plot(data_new[,1:2], col = result$cluster)
```

```
# Plotting to see how different variable data points have been distributed
# originally as per "class" attribute in dataset
# ---
#
# visualizing the clusters
#set_plot_dimensions(6, 6)
```

```
par(mfrow = c(1,1), mar = c(5,4,2,2))
# plotting Administrative vs Informational
plot(data_new[,1:2], col = data_class)
```

```
# showing how the clusters respond to the classes
table(result$cluster, data_class)
```

```
# Plotting to see how different variable data points have been distributed in clusters
# ---
#
```

```
plot(data_new[c(3,4)], col = result$cluster)
plot(data_new[c(3,4)], col = data_class)
```

```
# Result of table shows that Cluster 1 corresponds to False revenue,
# Cluster 2 corresponds True revenue
# ---
#
table(result$cluster, data_class)
```

We can conclude that from the three clusters, False revenue was more than True revenue.

Hierachical_clustering

```
#Assigning a variable to numerical data
num_data <- subset(data_unique, select = -revenue)
```

```
# first we compute the euclidean distance
d <- dist(num_data, method = "euclidean")
```

```
# then we compute hierarchical clustering using the Ward method
hier <- hclust(d, method = "ward.D2" )
```

```
# Convert hclust into a dendrogram and plot
hcd <- as.dendrogram(hier)
# Define nodePar
nodePar <- list(lab.cex = 0.6, pch = c(20, 19),
                cex = 0.7, col = c("green","yellow"))
plot(hcd,  xlab = "Height", nodePar = nodePar, main = "Cluster dendrogram",
     edgePar = list(col = c("red","blue"), lwd = 2:1), horiz = TRUE)
```

Challenging the solution

The data contained outliers and missing data. The null values were dropped causing us to lose some data and that might have affected our analysis and modelling and maybe we could have had better accuracies. Prescence of outliers may have also affected our modelling and analysis.

Follow up questions

Did we have the right data?

- Yes

Did we have the right research question?

- Yes

Did we have enough data?

- Yes, but maybe if we did not have any missing values modells could have performed better.

Conclusions

- Most customers have a false revenue.
- Most customers use operating system 2.
- Browser 2 is the most frequently used.
- Region 1 is the most frequent.
- Traffic type 2 is the most frequent.
- Weekdays are more frequent than weekends.
- The most frequent month is May followed by November.
- Most visitors are returning visitors.
- From clustering, all the clusters had more False revenues than True revenues.

The accuracy of the supervised models are as follows;

- KNN - 87.78% before tuning and 87.67% after tuning.
- Decision Trees - 83.89%
- SVM - 88.9%
- Naive Bayes - 87.04%

Recommendations

- The company should use the SVM model to predict if the revenue is true or false.
- The company should also concentrate on the most frequent variables for example, operation system 2, browser 2, etc.