Customer Web Insights

Kennedy Muriuki

12/09/2020

Moringa School Independent Project Week 12

Business Understanding

a) Specifying the Question

A Kenyan client wishes to advertise their online course on an online platform. They wish to identify their audiences better so as to generate more leads and consequently improve the performance of their business. As a data scientist, I am tasked with analysing and identifying individuals that are most likely to click on her advertisement.

b) Defining the Metric for Success

The metric for this analysis is to identify distinguishing features that can identify/isolate her target audience from the rest of the population online.

d) Recording the Experimental Design

This study will follow the following life cycle:

1. Defining the question
2. Reading the data
3. Checking for missing data and outliers
4. Perform EDA
5. Implementing the solution
6. Challenging the solution
7. Follow up questions

Loading the dataset

```r
# loading the dataset which is in csv format

df <- read.csv(file.choose(),header = T)
```

```r
# Viewing the first five rows of the dataset

head(df,5)
```

```
##   Daily.Time.Spent.on.Site Age Area.Income Daily.Internet.Usage
## 1                    68.95  35    61833.90               256.09
## 2                    80.23  31    68441.85               193.77
## 3                    69.47  26    59785.94               236.50
## 4                    74.15  29    54806.18               245.89
## 5                    68.37  35    73889.99               225.58
##                           Ad.Topic.Line           City Male    Country
## 1    Cloned 5thgeneration orchestration    Wrightburgh    0    Tunisia
## 2    Monitored national standardization      West Jodi    1      Nauru
## 3      Organic bottom-line service-desk       Davidton    0 San Marino
## 4 Triple-buffered reciprocal time-frame West Terrifurt    1      Italy
## 5         Robust logistical utilization   South Manuel    0    Iceland
##             Timestamp Clicked.on.Ad
## 1 2016-03-27 00:53:11             0
## 2 2016-04-04 01:39:02             0
## 3 2016-03-13 20:35:42             0
## 4 2016-01-10 02:31:19             0
## 5 2016-06-03 03:36:18             0
```

```r
# viewing the last five entries of the dataset

tail(df,5)
```

```
##      Daily.Time.Spent.on.Site Age Area.Income Daily.Internet.Usage
## 996                     72.97  30    71384.57               208.58
## 997                     51.30  45    67782.17               134.42
## 998                     51.63  51    42415.72               120.37
## 999                     55.55  19    41920.79               187.95
## 1000                    45.01  26    29875.80               178.35
##                              Ad.Topic.Line          City Male
## 996         Fundamental modular algorithm     Duffystad    1
## 997       Grass-roots cohesive monitoring   New Darlene    1
## 998          Expanded intangible solution South Jessica    1
## 999  Proactive bandwidth-monitored policy   West Steven    0
## 1000      Virtual 5thgeneration emulation   Ronniemouth    0
##                     Country           Timestamp Clicked.on.Ad
## 996                 Lebanon 2016-02-11 21:49:00             1
## 997  Bosnia and Herzegovina 2016-04-22 02:07:01             1
## 998                Mongolia 2016-02-01 17:24:57             1
## 999               Guatemala 2016-03-24 02:35:54             0
## 1000                 Brazil 2016-06-03 21:43:21             1
```

```r
# checking the dimension of the dataset

dim(df)
```

```
## [1] 1000   10
```

```r
# displaying the column names in the dataset

names(df)
```

```
##  [1] "Daily.Time.Spent.on.Site" "Age"
```

```
##  [3] "Area.Income"              "Daily.Internet.Usage"
##  [5] "Ad.Topic.Line"            "City"
##  [7] "Male"                     "Country"
##  [9] "Timestamp"                "Clicked.on.Ad"
```

```r
# generating a sumary of the dataset

summary(df)
```

```
##  Daily.Time.Spent.on.Site      Age         Area.Income    Daily.Internet.Usage
##  Min.   :32.60            Min.   :19.00   Min.   :13996   Min.   :104.8
##  1st Qu.:51.36            1st Qu.:29.00   1st Qu.:47032   1st Qu.:138.8
##  Median :68.22            Median :35.00   Median :57012   Median :183.1
##  Mean   :65.00            Mean   :36.01   Mean   :55000   Mean   :180.0
##  3rd Qu.:78.55            3rd Qu.:42.00   3rd Qu.:65471   3rd Qu.:218.8
##  Max.   :91.43            Max.   :61.00   Max.   :79485   Max.   :270.0
##  Ad.Topic.Line          City                Male          Country
##  Length:1000        Length:1000        Min.   :0.000   Length:1000
##  Class :character   Class :character   1st Qu.:0.000   Class :character
##  Mode  :character   Mode  :character   Median :0.000   Mode  :character
##                                        Mean   :0.481
##                                        3rd Qu.:1.000
##                                        Max.   :1.000
##   Timestamp         Clicked.on.Ad
##  Length:1000        Min.   :0.0
##  Class :character   1st Qu.:0.0
##  Mode  :character   Median :0.5
##                     Mean   :0.5
##                     3rd Qu.:1.0
##                     Max.   :1.0
```

```r
# checking to see the structure of the dataset

str(df)
```

```
## 'data.frame':    1000 obs. of  10 variables:
##  $ Daily.Time.Spent.on.Site: num  69 80.2 69.5 74.2 68.4 ...
##  $ Age                     : int  35 31 26 29 35 23 33 48 30 20 ...
##  $ Area.Income             : num  61834 68442 59786 54806 73890 ...
##  $ Daily.Internet.Usage    : num  256 194 236 246 226 ...
##  $ Ad.Topic.Line           : chr  "Cloned 5thgeneration orchestration" "Monitored national standardi
##  $ City                    : chr  "Wrightburgh" "West Jodi" "Davidton" "West Terrifurt" ...
##  $ Male                    : int  0 1 0 1 0 1 0 1 1 1 ...
##  $ Country                 : chr  "Tunisia" "Nauru" "San Marino" "Italy" ...
##  $ Timestamp               : chr  "2016-03-27 00:53:11" "2016-04-04 01:39:02" "2016-03-13 20:35:42"
##  $ Clicked.on.Ad           : int  0 0 0 0 0 0 0 1 0 0 ...
```

The dataset contains 1000 entries and 10 columns. The dataset contains six numerical columns and 4 categorical columns. The categorical columns have been loaded as factors.

Checking for missing values

```r
# checking to see if the dataset contains any missing values

any(is.na(df))
```

```
## [1] FALSE
```

Cheking for duplicates

```r
# verifying if the dataset contains any duplicated data

any(duplicated(df))
```

```
## [1] FALSE
```

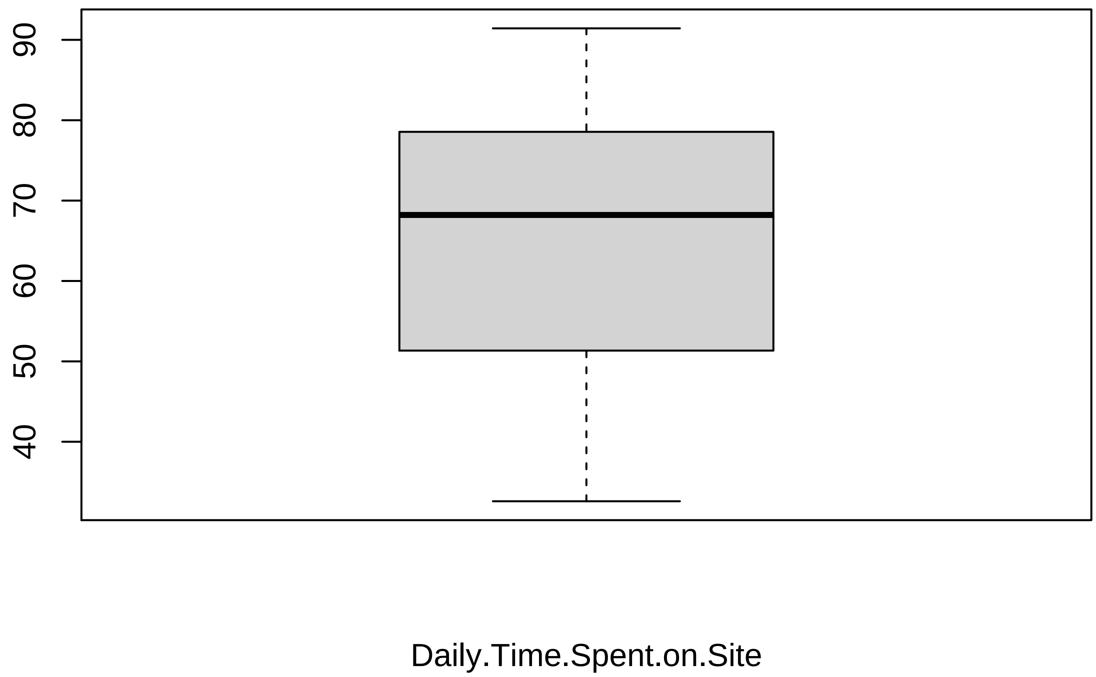
The dataset doesn't have missing values or duplicates therefore this dataset is of good quality.

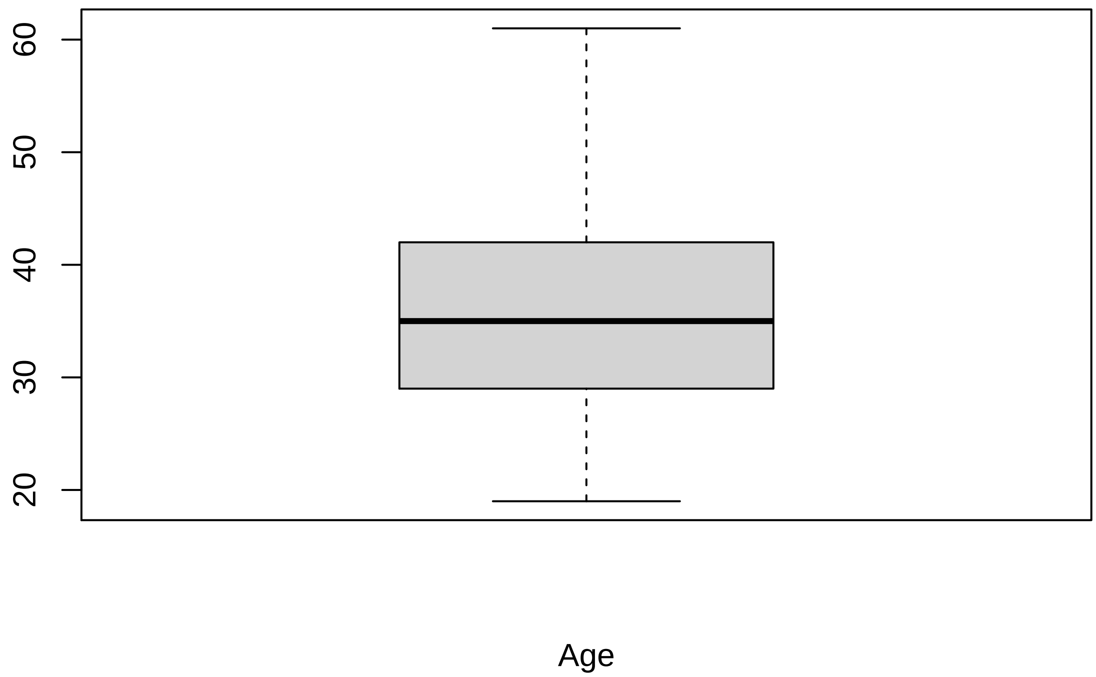
Checking for outliers

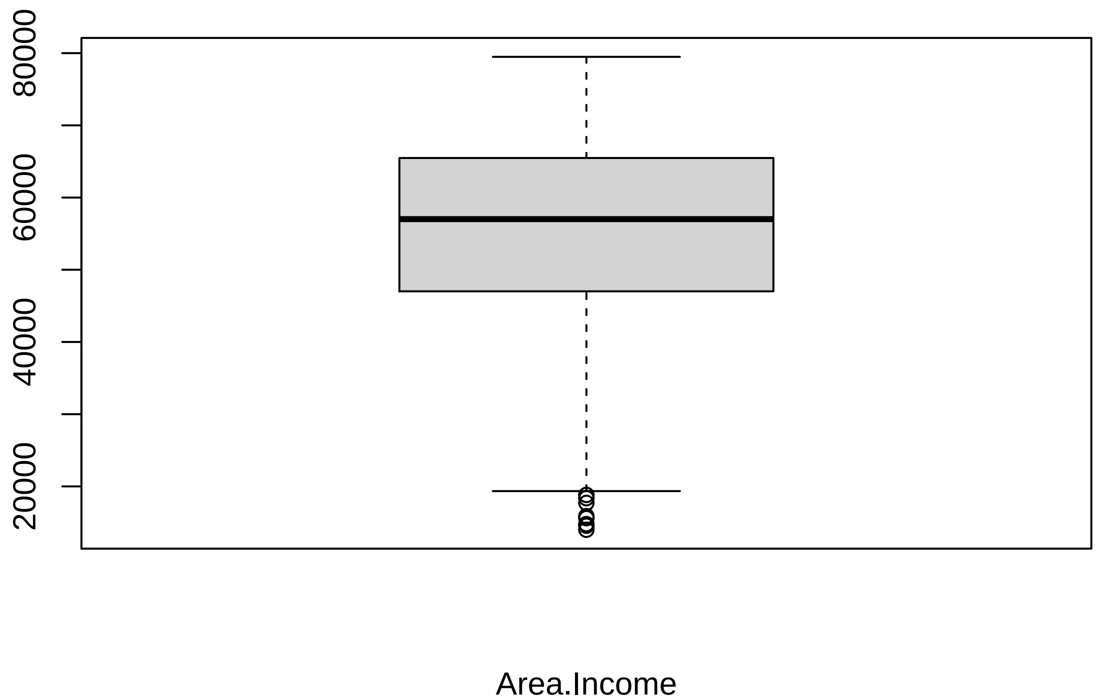
```r
# to check for outliers we draw a box and whisker plot for the numerical columns. I first select the nu

num_col <- df[,c(1,2,3,4,10)]

head(num_col)
```

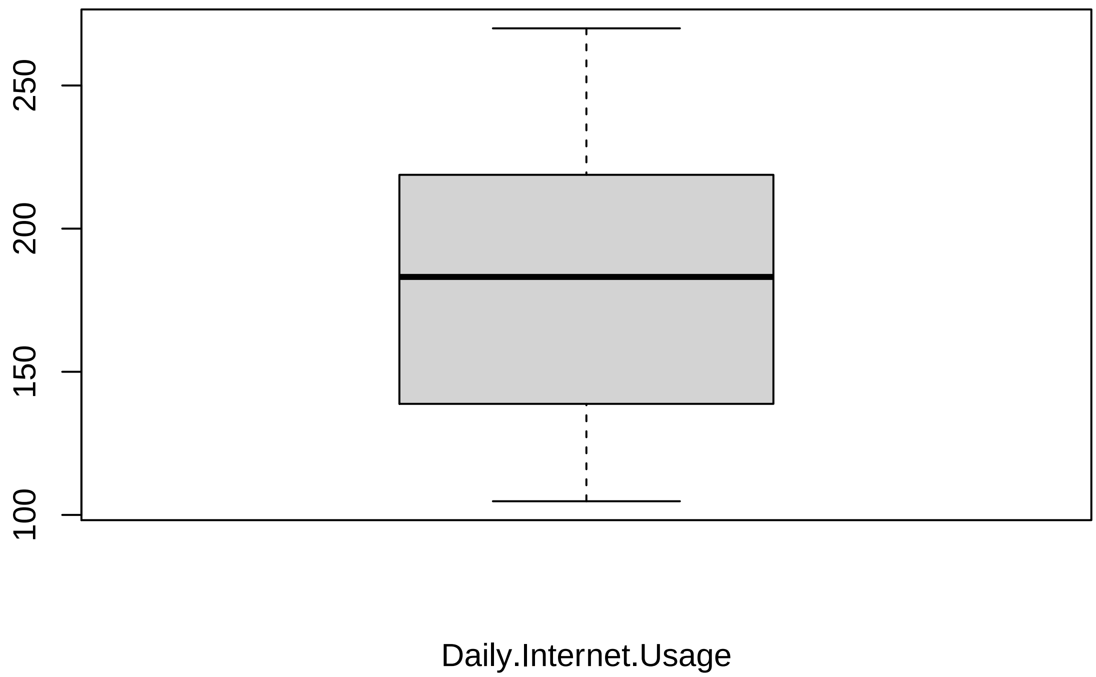
```
##   Daily.Time.Spent.on.Site Age Area.Income Daily.Internet.Usage Clicked.on.Ad
## 1                    68.95  35    61833.90               256.09             0
## 2                    80.23  31    68441.85               193.77             0
## 3                    69.47  26    59785.94               236.50             0
## 4                    74.15  29    54806.18               245.89             0
## 5                    68.37  35    73889.99               225.58             0
## 6                    59.99  23    59761.56               226.74             0
```

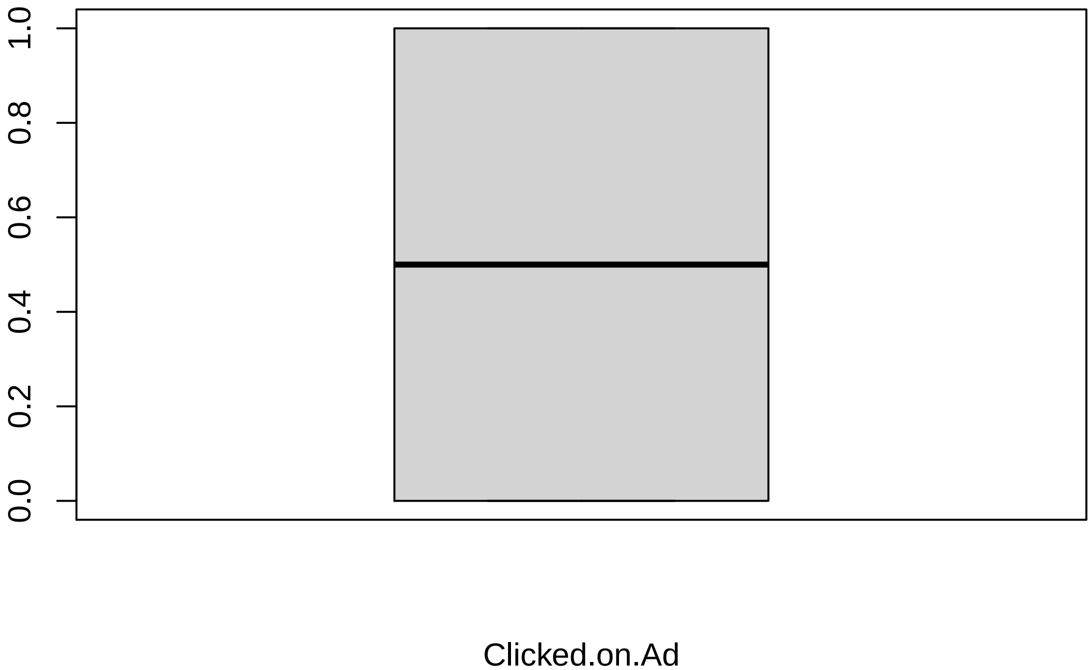
```r
# looping through the columns to generate boxplots

for (i in names(num_col)){
  x <- num_col[,i]
  boxplot(x, xlab= i)
  boxplot.stats(x)$out
}
```


90
80
70
60
50
40
Daily.Time.Spent.on.Site


60
50
40
30
20
Age


80000
60000
40000
20000
Area.Income


250
200
150
100
Daily.Internet.Usage




```r
# listing the outliers in the Area.Income column

boxplot.stats(df$Area.Income)$out
```

```
## [1] 17709.98 18819.34 15598.29 15879.10 14548.06 13996.50 14775.50 18368.57
```

The box and whisker plots generated shows the columns as having no outliers except the column Are.Income. The minimum entry for the column was 20,000. There were 8 outliers that ranged from 17,709 to 18,368. The outliers are expected in the Income column and therefore will not be removed as they contain useful information.

Univariate Exploratory Data Analysis

Measures of central tendencies

```r
# calculating the mean of the numerical columns

for (i in names(num_col)){
  x <- num_col[,i]
  mean <- mean(x)
  print(paste("The mean ", i , "is" ,  mean))
}
```

```
## [1] "The mean  Daily.Time.Spent.on.Site is 65.0002"
## [1] "The mean  Age is 36.009"
## [1] "The mean  Area.Income is 55000.00008"
## [1] "The mean  Daily.Internet.Usage is 180.0001"
## [1] "The mean  Clicked.on.Ad is 0.5"
```

```
# calculating the median for the numerical columns

for (i in names(num_col)){
  x <- num_col[,i]
  median <- median(x)
  print(paste("The median of the column", i , "is" ,  median))
}
```

```
## [1] "The median of the column Daily.Time.Spent.on.Site is 68.215"
## [1] "The median of the column Age is 35"
## [1] "The median of the column Area.Income is 57012.3"
## [1] "The median of the column Daily.Internet.Usage is 183.13"
## [1] "The median of the column Clicked.on.Ad is 0.5"
```

```
# calculating the mode of the numerical columns

# writing the function to calculate the mode of the numerical columns

getmode <- function(a){
  uniqv <- unique(a)
  uniqv[which.max(tabulate(match(a,uniqv)))]
}

# looping through the columns to get the mode

for (i in names(num_col)){
  x <- num_col[,i]
  mode <- getmode(x)
  print(paste("The mode of the column", i , "is" ,  mode))
}
```

```
## [1] "The mode of the column Daily.Time.Spent.on.Site is 62.26"
## [1] "The mode of the column Age is 31"
## [1] "The mode of the column Area.Income is 61833.9"
## [1] "The mode of the column Daily.Internet.Usage is 167.22"
## [1] "The mode of the column Clicked.on.Ad is 0"
```

```
# displaying the minimum and maximum of the numeric columns

for (i in names(num_col)){
  x <- num_col[,i]
  minimum <- min(x)
  maximum <- max(x)
  range <- maximum - minimum
  print(i)
  cat("\n")
  print(paste( "minimum of :" ,  minimum, "maximum :", maximum, "range :", range))
```

```r
  cat("\n")
}
```

```
## [1] "Daily.Time.Spent.on.Site"
##
## [1] "minimum of : 32.6 maximum : 91.43 range : 58.83"
##
## [1] "Age"
##
## [1] "minimum of : 19 maximum : 61 range : 42"
##
## [1] "Area.Income"
##
## [1] "minimum of : 13996.5 maximum : 79484.8 range : 65488.3"
##
## [1] "Daily.Internet.Usage"
##
## [1] "minimum of : 104.78 maximum : 269.96 range : 165.18"
##
## [1] "Clicked.on.Ad"
##
## [1] "minimum of : 0 maximum : 1 range : 1"
```

measures of dispersion

```r
# displaying the five number summary of the numeric columns

for (i in names(num_col)){
  x <- num_col[,i]
  quantile <- quantile(x)
  print(i)
  cat("\n")
  print(quantile)
  cat("\n")
}
```

```
## [1] "Daily.Time.Spent.on.Site"
##
##      0%     25%     50%     75%    100%
## 32.6000 51.3600 68.2150 78.5475 91.4300
##
## [1] "Age"
##
##   0%  25%  50%  75% 100%
##   19   29   35   42   61
##
## [1] "Area.Income"
##
##       0%      25%      50%      75%     100%
## 13996.50 47031.80 57012.30 65470.64 79484.80
##
```

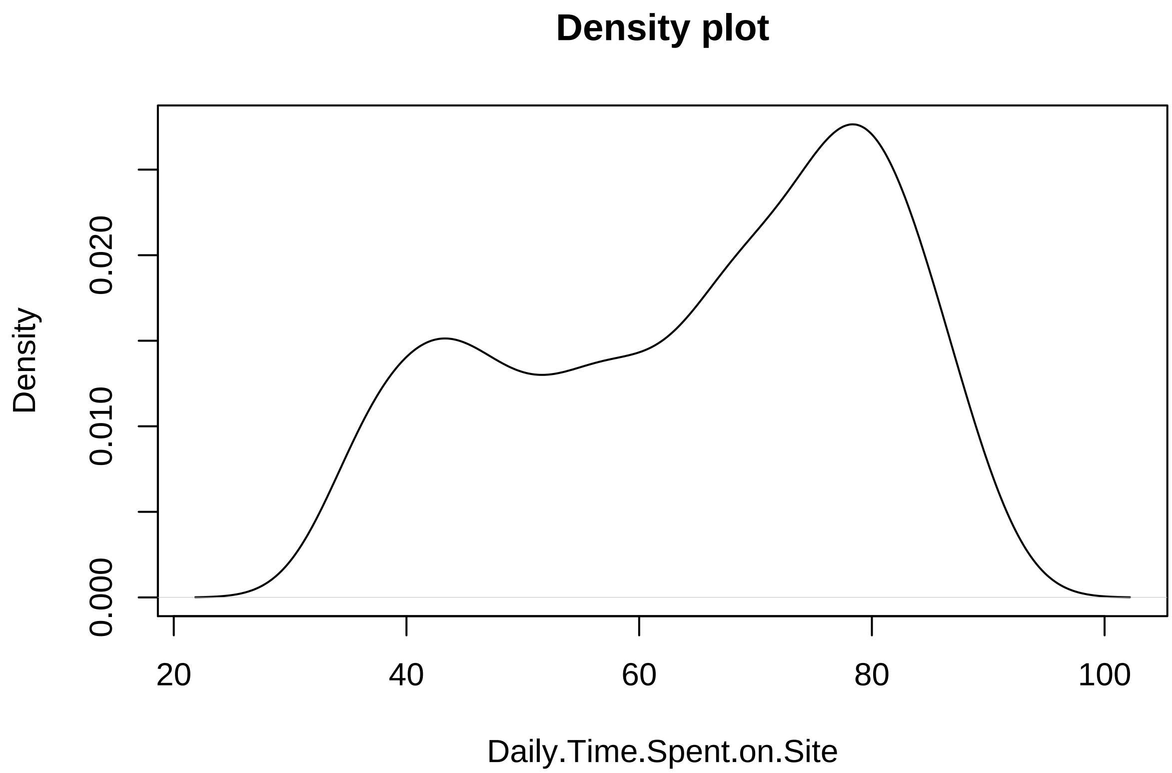
```
## [1] "Daily.Internet.Usage"
##
##       0%      25%      50%      75%     100%
## 104.7800 138.8300 183.1300 218.7925 269.9600
##
## [1] "Clicked.on.Ad"
##
##   0%  25%  50%  75% 100%
##  0.0  0.0  0.5  1.0  1.0
```

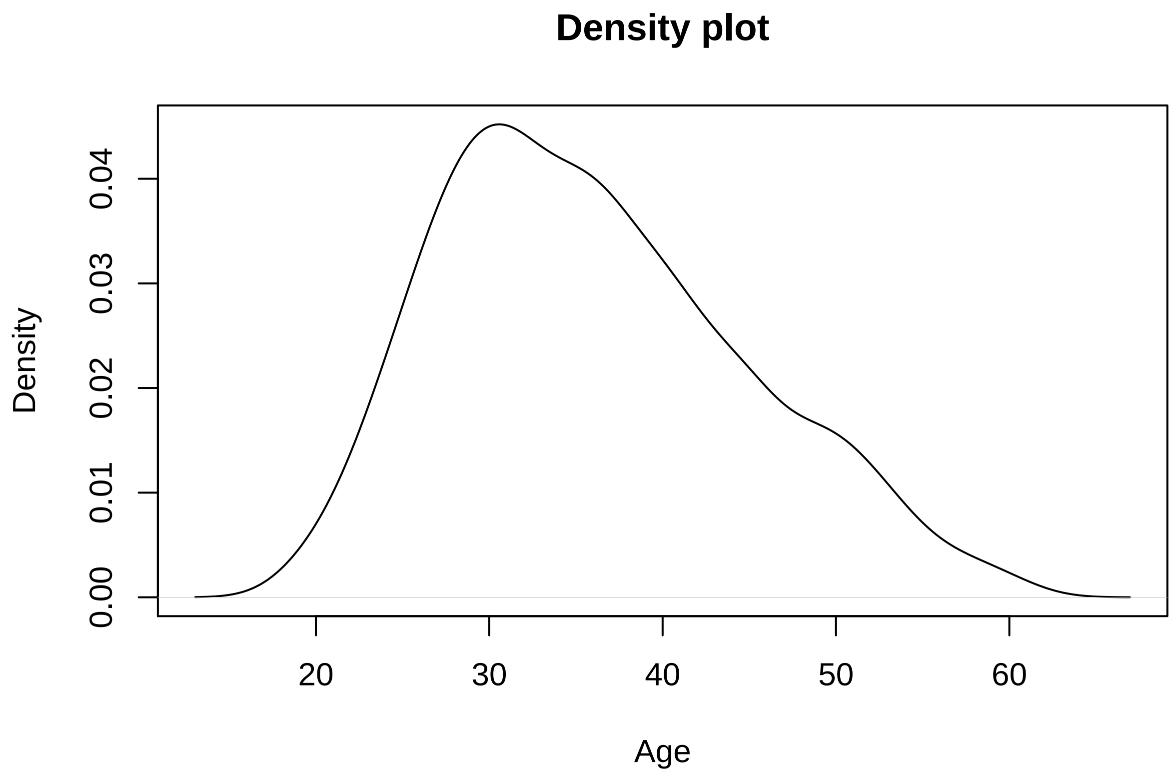
```r
# checking the variance and standard deviation of the numerical columns

for (i in names(num_col)){
  x <- num_col[,i]
  Sdev <- sd(x)
  var <- var(x)
  print(i)
  print(paste("var :",  round(var,2), "std dev :", round(var, 2)))
  cat("\n")
  cat("\n")
}
```

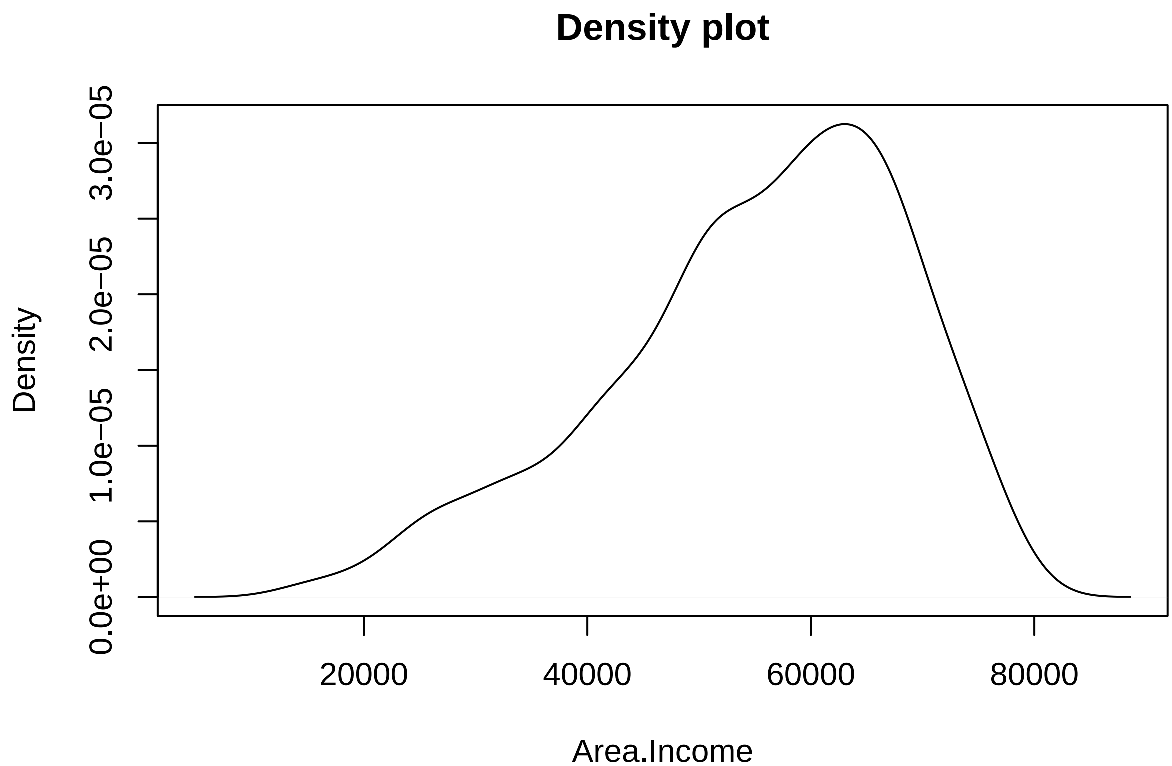
```
## [1] "Daily.Time.Spent.on.Site"
## [1] "var : 251.34 std dev : 251.34"
##
##
## [1] "Age"
## [1] "var : 77.19 std dev : 77.19"
##
##
## [1] "Area.Income"
## [1] "var : 179952405.95 std dev : 179952405.95"
##
##
## [1] "Daily.Internet.Usage"
## [1] "var : 1927.42 std dev : 1927.42"
##
##
## [1] "Clicked.on.Ad"
## [1] "var : 0.25 std dev : 0.25"
```

Density plots

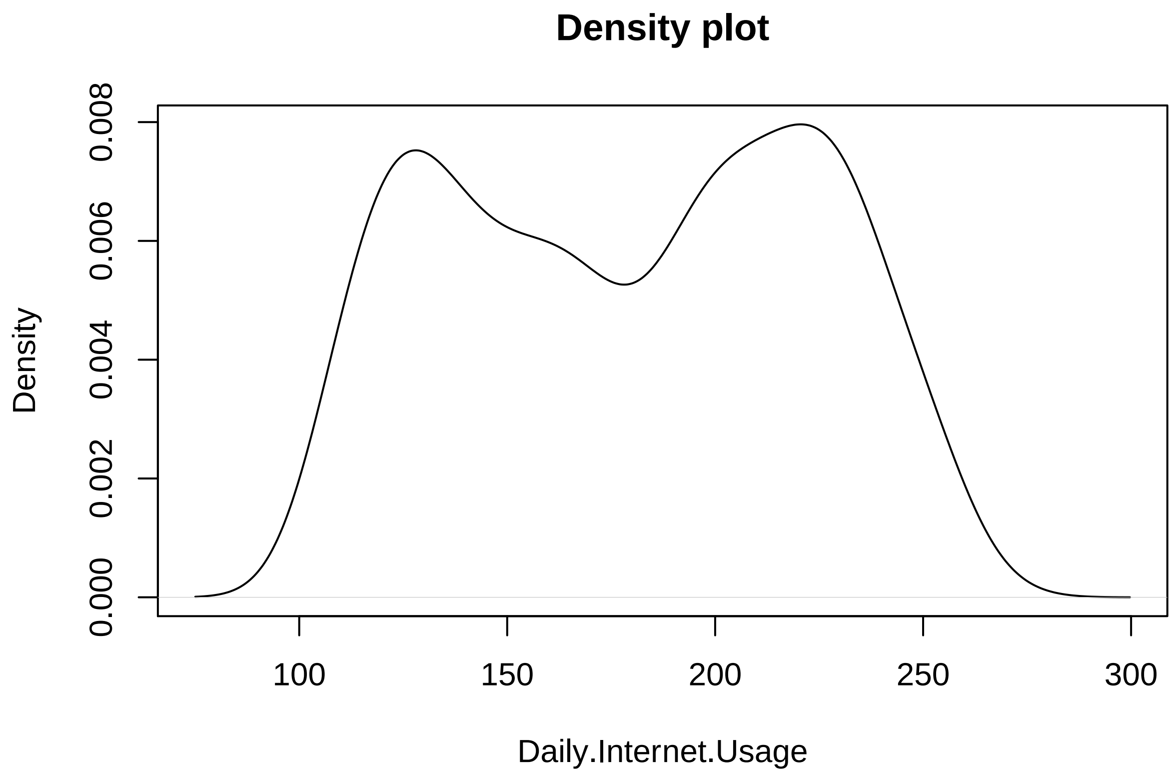
```r
# plotting density plots for the numerical columns

for (i in names(num_col)){
  x <- num_col[,i]
  plt <- density(x)
  plot(plt, xlab= i, main = "Density plot")
}
```

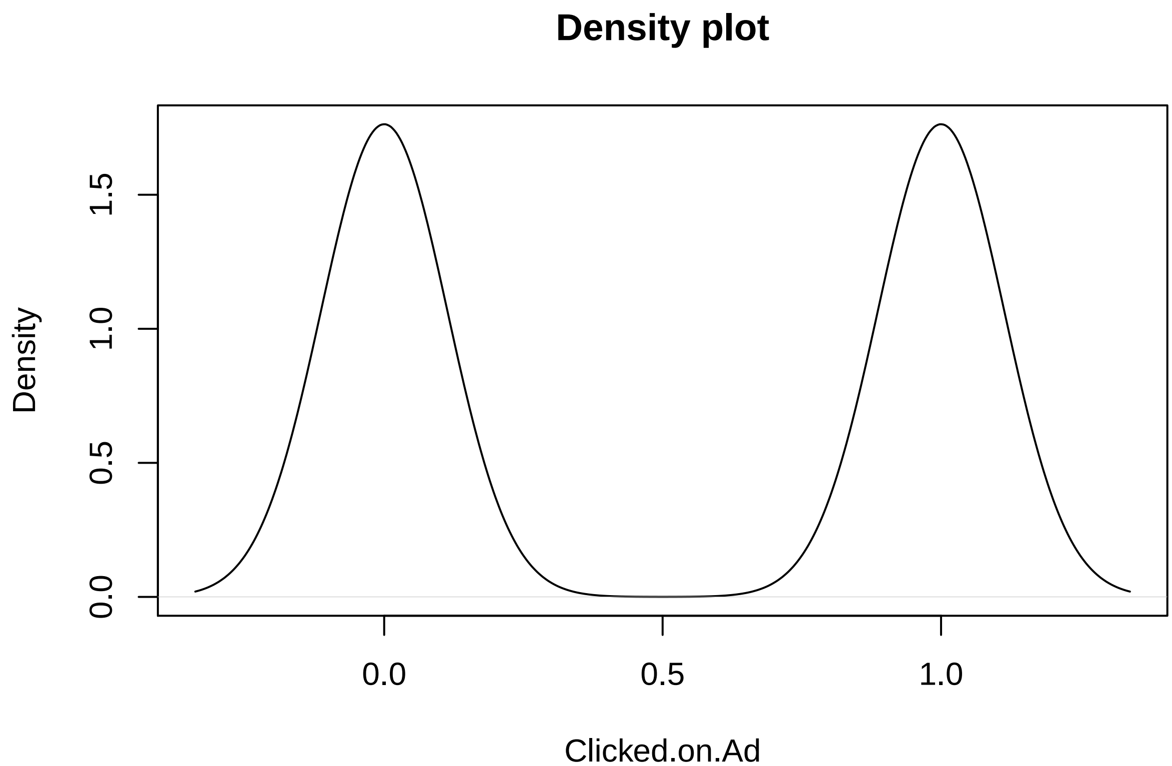

Density plot
Density
0.000
0.010
0.020
20
40
60
80
100
Daily.Time.Spent.on.Site


Density plot
Density
0.00
0.01
0.02
0.03
0.04
20
30
40
50
60
Age



Density plot
Density
3.0e−05
2.0e−05
1.0e−05
0.0e+00
20000
40000
60000
80000
Area.Income


Density plot
Density
0.000
0.002
0.004
0.006
0.008
100
150
200
250
300
Daily.Internet.Usage

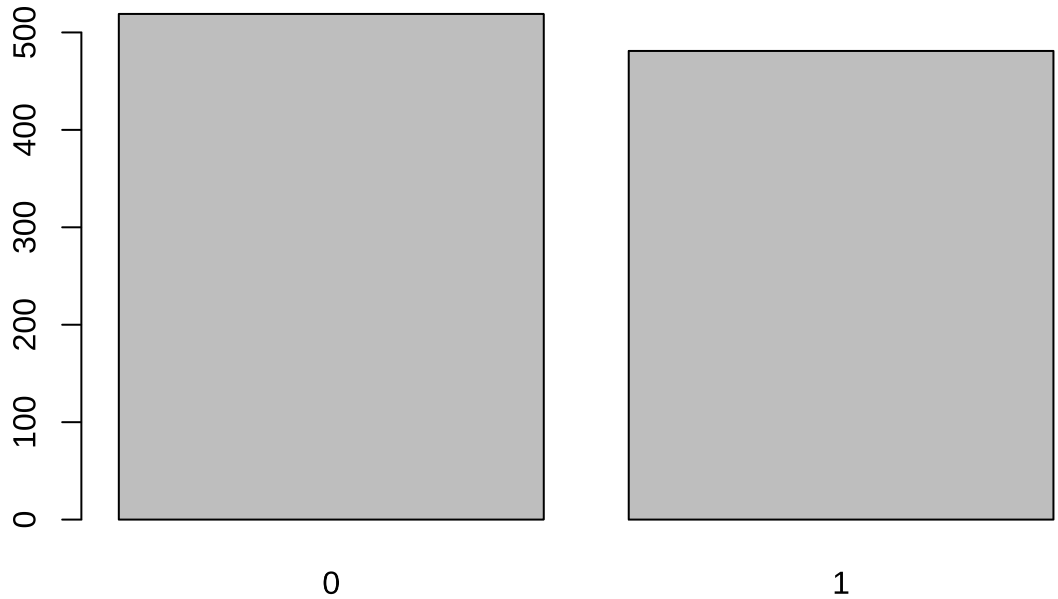


Bar Plots

```
# plotting the barplot for male column

male <- table(df$Male)
male
```

```
##
##   0   1
## 519 481
```

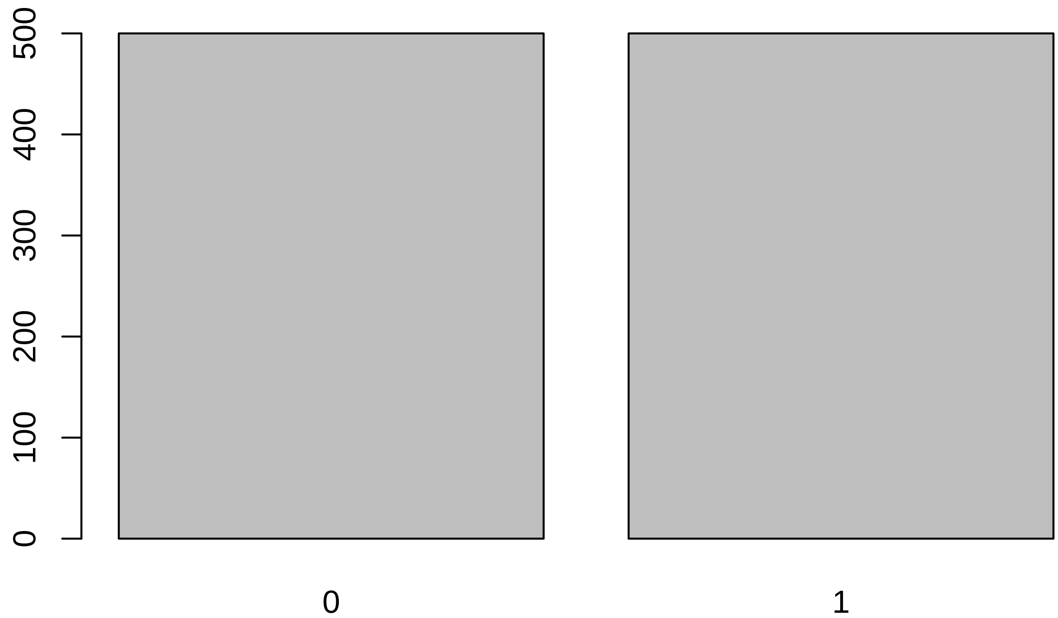
```
barplot(male)
```




```r
# plotting a barplot for the clicked on ad column
ad <- table(df$Clicked.on.Ad)
ad
```

```
##
##   0   1
## 500 500
```

```r
barplot(ad)
```



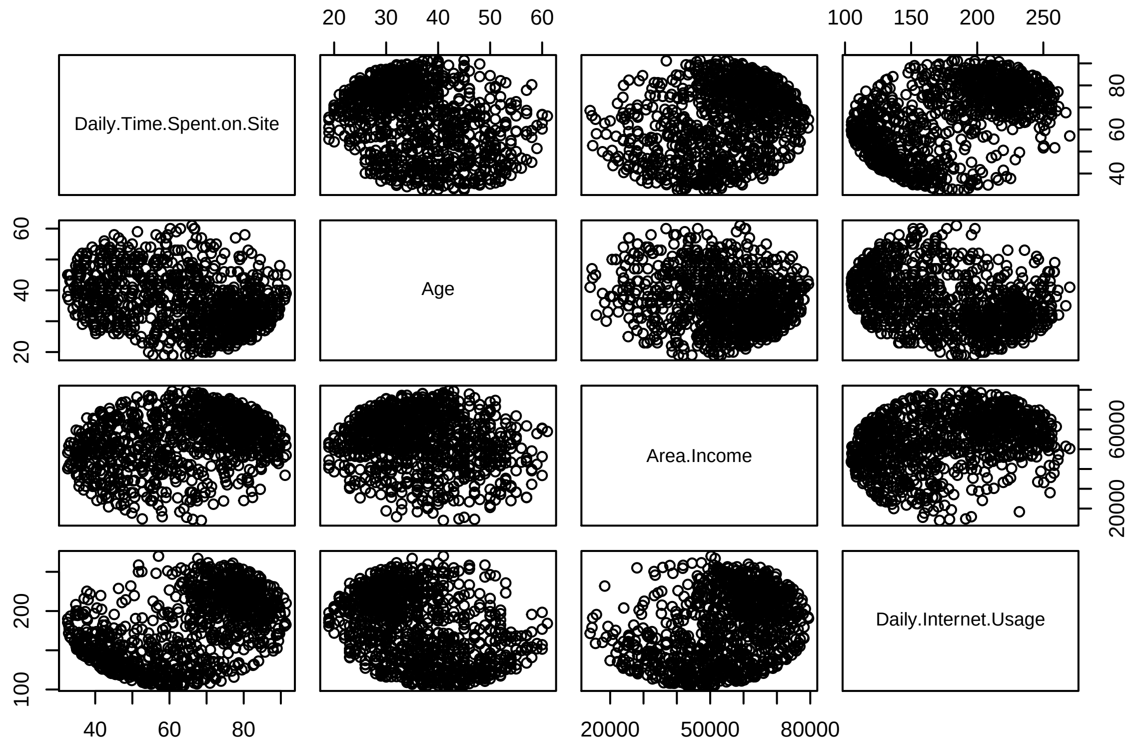
Bivariate Analysis

```r
# finding the correlation between the numeric variables and round them to 2 decimal places

res <- cor(num_col)
corr <- round(res,2)
corr
```

```
##                          Daily.Time.Spent.on.Site   Age Area.Income
## Daily.Time.Spent.on.Site                     1.00 -0.33        0.31
## Age                                         -0.33  1.00       -0.18
## Area.Income                                  0.31 -0.18        1.00
## Daily.Internet.Usage                         0.52 -0.37        0.34
## Clicked.on.Ad                               -0.75  0.49       -0.48
##                          Daily.Internet.Usage Clicked.on.Ad
## Daily.Time.Spent.on.Site                 0.52         -0.75
## Age                                     -0.37          0.49
## Area.Income                              0.34         -0.48
## Daily.Internet.Usage                     1.00         -0.79
## Clicked.on.Ad                           -0.79          1.00
```

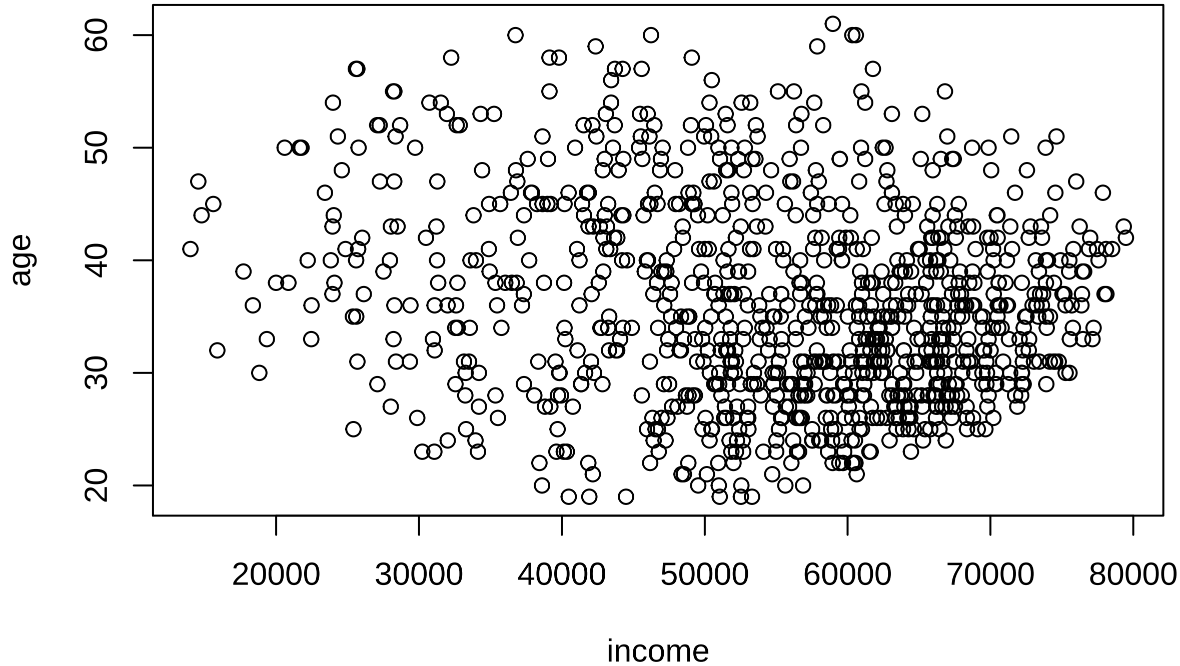
```r
# finding the scatter plots for the numeric columns
```

```r
col <- num_col[,c(1,2,3,4)]
pairs(col)
```



The variables do not show any kind of relationship since the scatter plots do not follow any discernable pattern.

```r
# the scatter plot of age and income

age <- df$Age
income <- df$Area.Income

plot(income, age)
```



Summary

1. The daily time spent online for the people who visited the site ranged from a minimum of 32.6 minutes to a maximum of 91.43 minutes. The average time spent online was 65 minutes with 50% of the visitors spending 68.2 minutes and below. However, most people spent 62.26 minutes online. The standard deviation is 15.85 minutes.

2. People visiting the website were aged between 19 and 61 years. They had an average of 36 years with most visitors having age 31 years. 50% of all visitors had an age of 35 years and below.

3. The minimum income of the visitors was 13,996.50 while the maximum income was 79,484.80. The mean income was 55,000 with 50% of the visitors having an income of 57,012.30 and below. Most people however had an income of 61,833.90 with a standard deviation of 13,414.

4. Internet usage ranged from a minimum of 104.78 to 269.96. The average internet usage was 180 and 50% of individuals had a usage of 183.13 and below. most people had a usage of 167.22 with a standard deviation of 43.

5. The number of people who clicked the ad was equal to the number of people who did not click the ad. This shows that the dataset was balanced.

6. Males were more likely not to click the ad. This is because the number of males that didn't click the ad was more that that who did.

7. The daily time spent on site and clicked on ad had a high negative correlation. This therefore means that the more time a visitor spent online, the less likely the visitor will click on an ad.

8. Daily time spent online and the daily internet usage had a high positive correlation. The more time the visitor spent online, the more internet they used up.

9. Income and internet usage had a weak positive correlation.

Data preprocessing and feature engineering

```
df$Clicked.on.Ad <- as.factor(df$Clicked.on.Ad)
```

```
# changing the column into datetime format from character


#install.packages("dplyr")
library(dplyr)
```

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

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

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

```
#install.packages("hablar")
#library(hablar)
```

```
head(df)
```

```
##   Daily.Time.Spent.on.Site Age Area.Income Daily.Internet.Usage
## 1                    68.95  35    61833.90               256.09
## 2                    80.23  31    68441.85               193.77
## 3                    69.47  26    59785.94               236.50
## 4                    74.15  29    54806.18               245.89
## 5                    68.37  35    73889.99               225.58
## 6                    59.99  23    59761.56               226.74
##                          Ad.Topic.Line           City Male    Country
## 1    Cloned 5thgeneration orchestration    Wrightburgh    0    Tunisia
## 2    Monitored national standardization      West Jodi    1      Nauru
## 3      Organic bottom-line service-desk       Davidton    0 San Marino
## 4 Triple-buffered reciprocal time-frame West Terrifurt    1      Italy
## 5         Robust logistical utilization   South Manuel    0    Iceland
## 6       Sharable client-driven software      Jamieberg    1     Norway
##             Timestamp Clicked.on.Ad
## 1 2016-03-27 00:53:11             0
## 2 2016-04-04 01:39:02             0
## 3 2016-03-13 20:35:42             0
## 4 2016-01-10 02:31:19             0
## 5 2016-06-03 03:36:18             0
## 6 2016-05-19 14:30:17             0
```

```r
# splitting the date and time column
df[["Timestamp"]] <- as.POSIXct(df$Timestamp, tz=Sys.timezone())
str(df)
```

```
## 'data.frame':    1000 obs. of  10 variables:
##  $ Daily.Time.Spent.on.Site: num  69 80.2 69.5 74.2 68.4 ...
##  $ Age                     : int  35 31 26 29 35 23 33 48 30 20 ...
##  $ Area.Income             : num  61834 68442 59786 54806 73890 ...
##  $ Daily.Internet.Usage    : num  256 194 236 246 226 ...
##  $ Ad.Topic.Line           : chr  "Cloned 5thgeneration orchestration" "Monitored national standardi
##  $ City                    : chr  "Wrightburgh" "West Jodi" "Davidton" "West Terrifurt" ...
##  $ Male                    : int  0 1 0 1 0 1 0 1 1 1 ...
##  $ Country                 : chr  "Tunisia" "Nauru" "San Marino" "Italy" ...
##  $ Timestamp               : POSIXct, format: "2016-03-27 00:53:11" "2016-04-04 01:39:02" ...
##  $ Clicked.on.Ad           : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 2 1 1 ...
```

```r
# splitting the timestamp column into time and date

Time <- format(as.POSIXct(strptime(df$Timestamp,"%Y-%m-%d %H:%M:%S",tz="")) ,format = "%H:%M:%S")
head(Time)
```

```
## [1] "00:53:11" "01:39:02" "20:35:42" "02:31:19" "03:36:18" "14:30:17"
```

```r
# getting the date column
Dates <- format(as.POSIXct(strptime(df$Timestamp,"%Y-%m-%d %H:%M:%S",tz="")) ,format = "%Y-%m-%d")
head(Dates)
```

```
## [1] "2016-03-27" "2016-04-04" "2016-03-13" "2016-01-10" "2016-06-03"
## [6] "2016-05-19"
```

```r
df$Dates <- Dates
df$Time <- Time
```

```r
# separating the columns further into subgroups
library(tidyr)

df <- separate(df, "Dates", c("Year", "Month", "Day"), sep = "-")

df <- separate(df, "Time", c("Hour", "Minutes", "Seconds"), sep = ":")

colnames(df)
```

```
##  [1] "Daily.Time.Spent.on.Site" "Age"
##  [3] "Area.Income"              "Daily.Internet.Usage"
##  [5] "Ad.Topic.Line"            "City"
##  [7] "Male"                     "Country"
##  [9] "Timestamp"                "Clicked.on.Ad"
## [11] "Year"                     "Month"
## [13] "Day"                      "Hour"
## [15] "Minutes"                  "Seconds"
```

```r
# converting the relevant columns into factors

df$Male = factor(df$Male)
df$Year = factor(df$Year)
df$Month = factor(df$Month)
df$Day = factor(df$Day)
df$Hour = factor(df$Hour)
df$Minutes = factor(df$Minutes)
df$Seconds = factor(df$Seconds)
```

```r
# removing the timestamp column
df$Timestamp<-NULL
```

```r
head(df)
```

```
##   Daily.Time.Spent.on.Site Age Area.Income Daily.Internet.Usage
## 1                    68.95  35    61833.90               256.09
## 2                    80.23  31    68441.85               193.77
## 3                    69.47  26    59785.94               236.50
## 4                    74.15  29    54806.18               245.89
## 5                    68.37  35    73889.99               225.58
## 6                    59.99  23    59761.56               226.74
##                           Ad.Topic.Line           City Male    Country
## 1    Cloned 5thgeneration orchestration    Wrightburgh    0    Tunisia
## 2    Monitored national standardization      West Jodi    1      Nauru
## 3      Organic bottom-line service-desk       Davidton    0 San Marino
## 4 Triple-buffered reciprocal time-frame West Terrifurt    1      Italy
## 5         Robust logistical utilization   South Manuel    0    Iceland
## 6       Sharable client-driven software      Jamieberg    1     Norway
##   Clicked.on.Ad Year Month Day Hour Minutes Seconds
## 1             0 2016    03  27   00      53      11
## 2             0 2016    04  04   01      39      02
## 3             0 2016    03  13   20      35      42
## 4             0 2016    01  10   02      31      19
## 5             0 2016    06  03   03      36      18
## 6             0 2016    05  19   14      30      17
```

```r
# removing the columns that are not needed

df$Ad.Topic.Line<-NULL
df$Country<-NULL
df$City<-NULL
df$Year<-NULL
```

Implimenting the solution

Decision trees

```r
#Installing libraries
#install.packages('rpart')
```

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

```r
# loading the libraries
library('rpart')
library('caret')
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

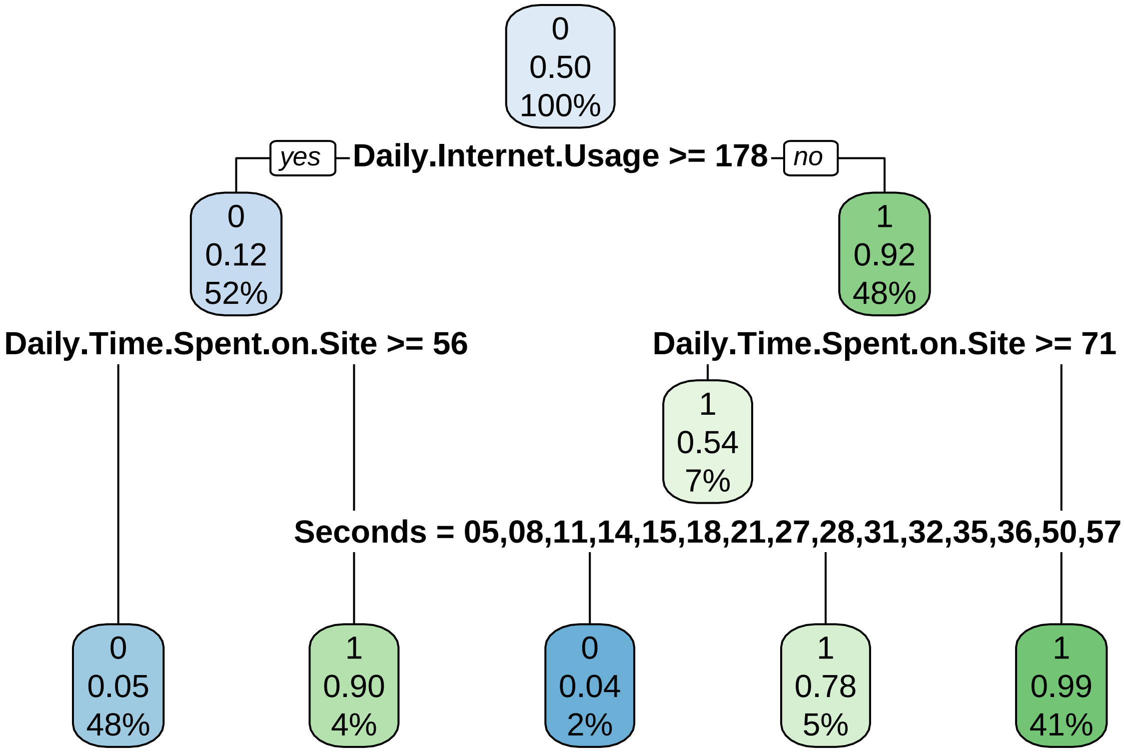
```r
library('rpart.plot')
library('rattle')
```

```
## 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.
```

```r
# fitting the decision tree

x <- rpart(Clicked.on.Ad ~ ., data = df,
           method = "class")
rpart.plot(x)
```




```
# generating a confusion matrix to evaluate the performance

pred <- predict(x, df, type = "class")
table(pred, df$Clicked.on.Ad)
```

```
##
## pred   0   1
##    0 481  24
##    1  19 476
```

```
# visualizing the decision trees

#output.tree <- ctree(Clicked.on.Ad ~ ., data = df)

#plot(output.tree)
```

Random Forest

```
# loading the random forest library
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:rattle':
##
##     importance

## The following object is masked from 'package:ggplot2':
##
##     margin

## The following object is masked from 'package:dplyr':
##
##     combine
```

```
# Set a random seed
set.seed(40)
# Training using 'random forest' algorithm
model <- train(Clicked.on.Ad ~ .,
               data = df,
               method = 'rf',
               trControl = trainControl(method = 'cv',number = 10))
model
```

```
## Random Forest
##
## 1000 samples
##   10 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 900, 900, 900, 900, 900, 900, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa
##     2   0.932     0.864
##    92   0.962     0.924
##   182   0.957     0.914
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 92.
```

The random forest classifier has an accuracy of 96.2. This is a high accuracy level after cross validating the model 10 times.

Naive Bayes algorithm

```r
# Splitting data into training and test data sets
# ---
#
train_set <- createDataPartition(y = df$Clicked.on.Ad,p = 0.75,list = FALSE)

training <- df[train_set,]

testing <- df[-train_set,]
```

```r
# Creating objects x which holds the predictor variables and y which holds the response variables
# ---

y = training$Clicked.on.Ad

x =  subset(training, select = -c(Clicked.on.Ad) )
```

```r
# training the model
model = train(x,y,'nb',trControl=trainControl(method='cv',number=10))
```

```r
# evaluating the model

Predict <- predict(model,newdata = testing )

# Getting the confusion matrix to see accuracy value and other parameter values
# ---
#
confusionMatrix(Predict, testing$Clicked.on.Ad )
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 122   5
##          1   3 120
##
##                Accuracy : 0.968
##                  95% CI : (0.9379, 0.9861)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.936
##
##  Mcnemar's Test P-Value : 0.7237
##
##             Sensitivity : 0.9760
##             Specificity : 0.9600
##          Pos Pred Value : 0.9606
##          Neg Pred Value : 0.9756
##              Prevalence : 0.5000
##          Detection Rate : 0.4880
##    Detection Prevalence : 0.5080
##       Balanced Accuracy : 0.9680
```

```
##
##        'Positive' Class : 0
##
```

Support Vector Machines

```r
# inputting the train control and using the repeated cross validation method

trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 5)
```

```r
model_svm <- train(Clicked.on.Ad ~., data = training, method = "svmLinear",
                   trControl=trctrl,
                   preProcess = c("center", "scale"),
                   tuneLength = 10)
model_svm
```

```
## Support Vector Machines with Linear Kernel
##
## 750 samples
##  10 predictor
##   2 classes: '0', '1'
##
## Pre-processing: centered (182), scaled (182)
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 674, 675, 675, 674, 676, 676, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.9119847  0.8239783
##
## Tuning parameter 'C' was held constant at a value of 1
```

```r
# evaluating the model and checking how well it has performed
test_pred <- predict(model_svm, newdata = testing)

confusionMatrix(table(test_pred, testing$Clicked.on.Ad))
```

```
## Confusion Matrix and Statistics
##
##
## test_pred   0   1
##         0 116  15
##         1   9 110
##
##                Accuracy : 0.904
##                  95% CI : (0.8605, 0.9375)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.808
##
```

```
##  Mcnemar's Test P-Value : 0.3074
##
##             Sensitivity : 0.9280
##             Specificity : 0.8800
##          Pos Pred Value : 0.8855
##          Neg Pred Value : 0.9244
##              Prevalence : 0.5000
##          Detection Rate : 0.4640
##    Detection Prevalence : 0.5240
##       Balanced Accuracy : 0.9040
##
##        'Positive' Class : 0
##
```

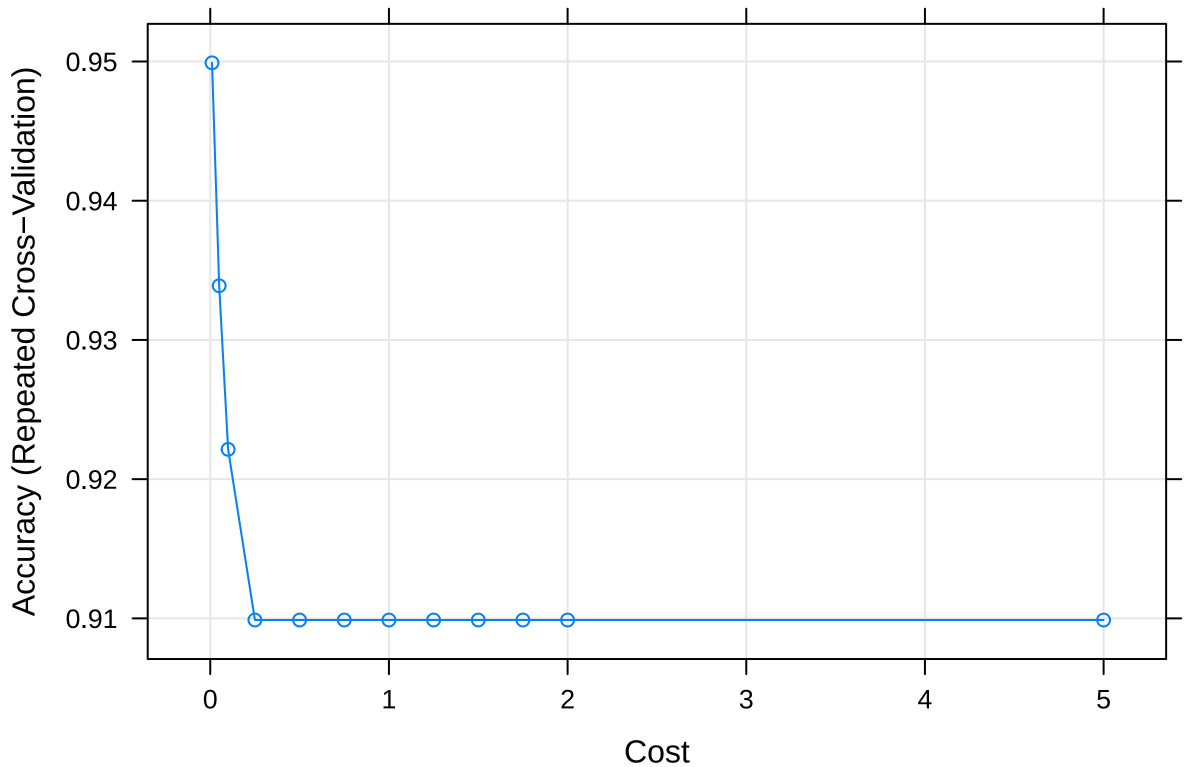
The SVM linear model performs poorly as compared to the previous models. The model achieves an accuracy of 91% with the training data and an accuracy of 94% with the test data. To improve the accuracy of the model, I will perform a gridsearch to obtain the best parameters and fit the model again. The parameter to optimize will be C.

```
# setting up the grid
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))

# fitting the grid to obtain the best parameter
svm_Linear_Grid <- train(Clicked.on.Ad ~., data = training, method = "svmLinear",
                         trControl=trctrl,
                         preProcess = c("center", "scale"),
                         tuneGrid = grid,
                         tuneLength = 10)
svm_Linear_Grid
```

```
## Support Vector Machines with Linear Kernel
##
## 750 samples
##  10 predictor
##   2 classes: '0', '1'
##
## Pre-processing: centered (182), scaled (182)
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 674, 674, 676, 675, 675, 676, ...
## Resampling results across tuning parameters:
##
##   C     Accuracy   Kappa
##   0.00        NaN        NaN
##   0.01  0.9499039  0.8998133
##   0.05  0.9338920  0.8678080
##   0.10  0.9221399  0.8443007
##   0.25  0.9098827  0.8197773
##   0.50  0.9098827  0.8197773
##   0.75  0.9098827  0.8197773
##   1.00  0.9098827  0.8197773
##   1.25  0.9098827  0.8197773
##   1.50  0.9098827  0.8197773
##   1.75  0.9098827  0.8197773
```

```
##   2.00  0.9098827  0.8197773
##   5.00  0.9098827  0.8197773
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was C = 0.01.
```

```r
# plotting different values of C
plot(svm_Linear_Grid)
```




```r
test_pred_grid <- predict(svm_Linear_Grid, newdata = testing)
test_pred_grid
```

```
##   [1] 0 0 1 0 1 0 1 0 0 0 0 1 1 1 1 1 0 1 0 1 0 1 0 1 0 1 1 0 0 1 1 1 0 0 0 0 0
##  [38] 0 1 1 1 0 1 0 1 1 1 0 0 1 0 1 1 1 1 1 1 0 0 1 0 1 1 0 0 1 0 0 0 1 0 0 0 0
##  [75] 0 0 0 1 0 0 0 1 0 1 0 1 0 0 0 1 0 0 0 0 0 0 1 1 1 1 1 0 0 1 1 1 1 0 1 0 1
## [112] 1 1 1 1 1 1 0 0 0 1 0 0 1 0 1 0 1 1 0 0 0 0 0 0 0 1 1 1 1 0 1 1 1 1 0 1 0
## [149] 0 0 1 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 1 1 0 0 1 1 0 1 0 0 1 0 1 0 1 1 1 1
## [186] 1 0 1 0 0 1 1 0 0 0 0 0 1 0 0 0 0 0 1 1 0 1 1 1 0 0 1 0 0 1 0 0 0 1 1 0 0
## [223] 1 1 1 0 1 0 1 1 1 0 1 1 1 1 1 0 0 0 0 0 1 0 1 0 1 1 1 0
## Levels: 0 1
```

```r
confusionMatrix(table(test_pred_grid, testing$Clicked.on.Ad))
```

```
## Confusion Matrix and Statistics
```

```
##
##
## test_pred_grid   0   1
##              0 122  11
##              1   3 114
##
##                Accuracy : 0.944
##                  95% CI : (0.9078, 0.969)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.888
##
##  Mcnemar's Test P-Value : 0.06137
##
##             Sensitivity : 0.9760
##             Specificity : 0.9120
##          Pos Pred Value : 0.9173
##          Neg Pred Value : 0.9744
##              Prevalence : 0.5000
##          Detection Rate : 0.4880
##    Detection Prevalence : 0.5320
##       Balanced Accuracy : 0.9440
##
##        'Positive' Class : 0
##
```

There is a significance increase in the accuracy of the model from 94% to 96.8%. The algorithm also misclassifies only 8 data points.

challenging the model

Logistic

```r
#Fitting a logistic regression model

# logmodel <- glm(Clicked.on.Ad ~ ., family = "quasibinomial", data = train)
```

follow up questions

1. Did we have the right data? Yes. The data was the right one for a classification problem
2. Would we have obtained better results if we had more data Probably not. The data was enough to work with. More operations needs to be performed to the data to make it usable for analysis.