Advertisment Analysis

Wangechi Mungai

18/03/2022

1.Defining the Question

a) Specifying the Question

A Kenyan entrepreneur has created an online cryptography course and would want to advertise it on her blog. She currently targets audiences originating from various countries. In the past, she ran ads to advertise a related course on the same blog and collected data in the process. She would now like to employ your services as a Data Science Consultant to help her identify which individuals are most likely to click on her ads.

b) Defining the Metric for Success The model will be appraised successful if it will be able to predict in the right way which variable influences the clicking of ads. We will set 96% as our minimum accuracy score for the models.

c) Understanding the context

The Kenyan Entrepreneur would like to identifying the factors that influence the clicking of ads which is vital for the her.

d) Recording the Experimental Design

The following are the experimental design i took in order to complete this project:

1.Importing all the necessary libraries

2.Loading the dataset

3.Reading, cleaning the dataset

4.Performing Exploratory Data Analysis

5.Performing data modelling using Decision Trees and Support Vector Machine

6.Giving conclusions and recommendations.

e)Reading the Data

```
df <- read.csv("http://bit.ly/IPAdvertisingData")
```

f)Checking the Data

```
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
```

```
#previewing tail of dataset
tail(df)
```

```
##      Daily.Time.Spent.on.Site Age Area.Income Daily.Internet.Usage
## 995                     43.70  28    63126.96               173.01
## 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
## 995         Front-line bifurcated ability  Nicholasland    0
## 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
## 995                 Mayotte 2016-04-04 03:57:48             1
## 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
```

```
#taking a glance of the dataset
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
```

```
glimpse(df)
```

```
## Rows: 1,000
## Columns: 10
## $ Daily.Time.Spent.on.Site <dbl> 68.95, 80.23, 69.47, 74.15, 68.37, 59.99, 88.~
## $ Age                      <int> 35, 31, 26, 29, 35, 23, 33, 48, 30, 20, 49, 3~
## $ Area.Income              <dbl> 61833.90, 68441.85, 59785.94, 54806.18, 73889~
## $ Daily.Internet.Usage     <dbl> 256.09, 193.77, 236.50, 245.89, 225.58, 226.7~
## $ Ad.Topic.Line            <chr> "Cloned 5thgeneration orchestration", "Monito~
## $ City                     <chr> "Wrightburgh", "West Jodi", "Davidton", "West~
## $ Male                     <int> 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, ~
## $ Country                  <chr> "Tunisia", "Nauru", "San Marino", "Italy", "I~
## $ Timestamp                <chr> "2016-03-27 00:53:11", "2016-04-04 01:39:02",~
## $ Clicked.on.Ad            <int> 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, ~
```

```
#previewing the shape of the dataset
dim(df)
```

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

The dataset contains 1000 rows and 10 columns

```
#previewing the descriptive statistics of 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
```

```
#checking the datatypes of the columns
sapply(df, class)
```

```
## Daily.Time.Spent.on.Site                    Age            Area.Income
##                "numeric"              "integer"              "numeric"
##     Daily.Internet.Usage          Ad.Topic.Line                    City
##                "numeric"            "character"            "character"
##                     Male                Country              Timestamp
##                "integer"            "character"            "character"
##            Clicked.on.Ad
##                "integer"
```

All datatypes are correct except timestamp which should be change to date datatype in data cleaning.

```
#checking for any null values
colSums(is.na(df))
```

```
## Daily.Time.Spent.on.Site                    Age            Area.Income
##                        0                      0                      0
##     Daily.Internet.Usage          Ad.Topic.Line                    City
##                        0                      0                      0
##                     Male                Country              Timestamp
##                        0                      0                      0
##            Clicked.on.Ad
##                        0
```

There are no missing values in the dataset

```
#checking for duplicate values
anyDuplicated(df)
```

```
## [1] 0
```

There are no duplicates in the dataset

g)Data Cleaning

```
#coverting timestamp column data type to date datatype
df$Timestamp <- as.Date(df$Timestamp)
class(df$Timestamp)
```

```
## [1] "Date"
```

```
#converting column names to lower case
colnames(df) = tolower(colnames(df))
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"
```

```r
#droping Ad.Topic.Line column

df = subset(df, select = -c(ad.topic.line))
```

```r
head(df)
```

```
##   daily.time.spent.on.site age area.income daily.internet.usage           city
## 1                    68.95  35    61833.90                256.09    Wrightburgh
## 2                    80.23  31    68441.85                193.77      West Jodi
## 3                    69.47  26    59785.94                236.50       Davidton
## 4                    74.15  29    54806.18                245.89 West Terrifurt
## 5                    68.37  35    73889.99                225.58   South Manuel
## 6                    59.99  23    59761.56                226.74      Jamieberg
##   male    country  timestamp clicked.on.ad
## 1    0    Tunisia 2016-03-27             0
## 2    1      Nauru 2016-04-04             0
## 3    0 San Marino 2016-03-13             0
## 4    1      Italy 2016-01-10             0
## 5    0    Iceland 2016-06-03             0
## 6    1     Norway 2016-05-19             0
```

```r
#checking for outliers for numerical cols
num_cols <- unlist(lapply(df, is.numeric))         # Identify numeric columns
num_cols
```

```
## daily.time.spent.on.site                      age             area.income
##                     TRUE                     TRUE                     TRUE
##     daily.internet.usage                     city                     male
##                     TRUE                    FALSE                     TRUE
##                  country                timestamp            clicked.on.ad
##                    FALSE                    FALSE                     TRUE
```

```r
#displayig the numerical columns
df_num <- df[ , num_cols]
head(df_num, 5)
```

```
##   daily.time.spent.on.site age area.income daily.internet.usage male
## 1                    68.95  35    61833.90                256.09    0
## 2                    80.23  31    68441.85                193.77    1
## 3                    69.47  26    59785.94                236.50    0
## 4                    74.15  29    54806.18                245.89    1
## 5                    68.37  35    73889.99                225.58    0
##   clicked.on.ad
## 1             0
## 2             0
## 3             0
## 4             0
## 5             0
```

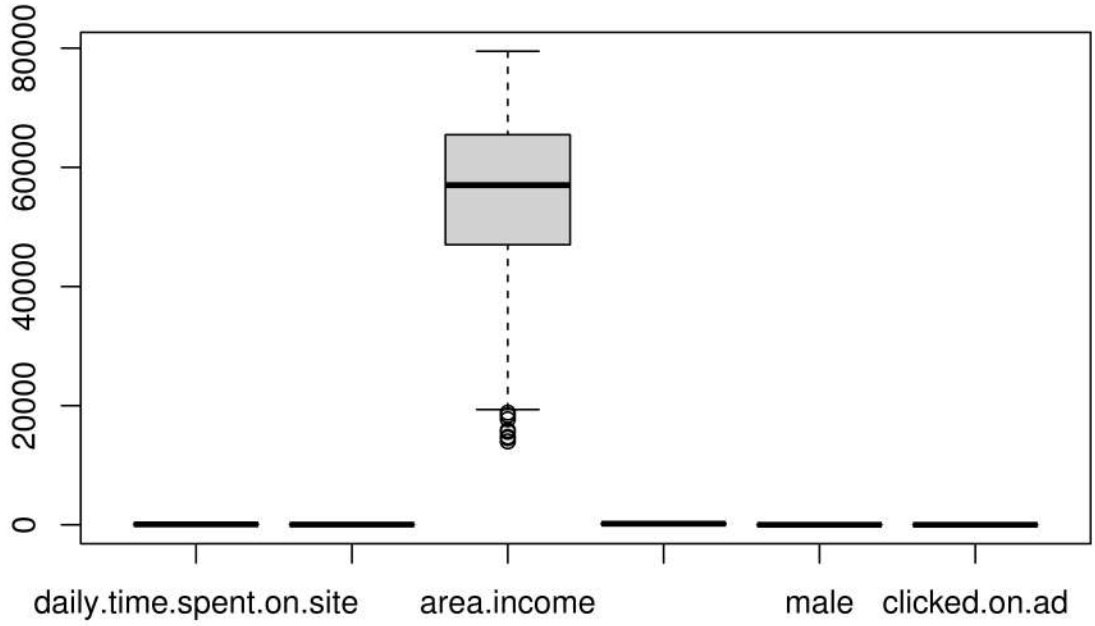
```r
outlier <- function(x){
  out <- boxplot.stats(x)$out
  return((length(out)/  1000)*100)
}
```

```
# Get outlier count per column
sapply(df[,c("daily.time.spent.on.site","age","area.income","daily.internet.usage","male","clicked.on.ad
```

```
## daily.time.spent.on.site                     age             area.income
##                      0.0                     0.0                     0.8
##     daily.internet.usage                    male           clicked.on.ad
##                      0.0                     0.0                     0.0
```

only the area income outliers has outliers.

```
 #visualizing the outiers
   boxplot(df_num)
```



there are a number of outliers in area income column but will not be removed since they are neccesary for our analysis

h) Exploratory Data Analysis Univariate Analysis

```
# describing our columns

psych::describe(df)
```

```
## Warning in FUN(newX[, i], ...): no non-missing arguments to min; returning Inf

## Warning in FUN(newX[, i], ...): no non-missing arguments to max; returning -Inf
```

```
##                          vars    n     mean       sd   median  trimmed      mad
## daily.time.spent.on.site    1 1000    65.00    15.85    68.22    65.74    17.92
## age                         2 1000    36.01     8.79    35.00    35.51     8.90
## area.income                 3 1000 55000.00 13414.63 57012.30 56038.94 13316.62
## daily.internet.usage        4 1000   180.00    43.90   183.13   179.99    58.61
## city*                       5 1000   487.32   279.31   485.50   487.51   356.57
## male                        6 1000     0.48     0.50     0.00     0.48     0.00
## country*                    7 1000   116.41    69.94   114.50   115.82    89.70
## timestamp                   8 1000      NaN       NA       NA      NaN       NA
## clicked.on.ad               9 1000     0.50     0.50     0.50     0.50     0.74
##                             min      max    range  skew kurtosis     se
## daily.time.spent.on.site  32.60    91.43    58.83 -0.37    -1.10   0.50
## age                       19.00    61.00    42.00  0.48    -0.41   0.28
## area.income             13996.50 79484.80 65488.30 -0.65    -0.11 424.21
## daily.internet.usage      104.78   269.96   165.18 -0.03    -1.28   1.39
## city*                       1.00   969.00   968.00  0.00    -1.19   8.83
## male                        0.00     1.00     1.00  0.08    -2.00   0.02
## country*                    1.00   237.00   236.00  0.08    -1.23   2.21
## timestamp                    Inf     -Inf     -Inf    NA       NA     NA
## clicked.on.ad               0.00     1.00     1.00  0.00    -2.00   0.02
```

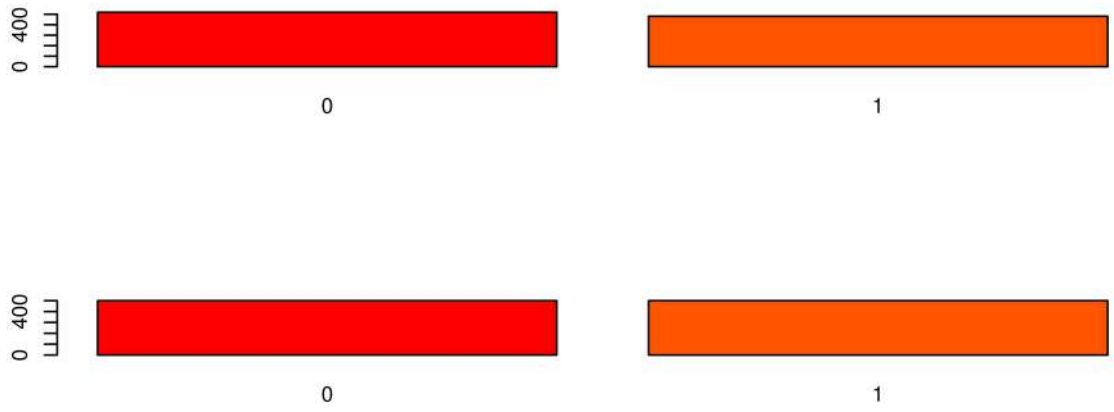
the variables are fairly skewed

```
# Frequency distribution of the categorical variables
sapply(df[, c("male","clicked.on.ad")], table)
```

```
##   male clicked.on.ad
## 0  519           500
## 1  481           500
```

The male were 481 while those who are not male were 519. Those who clicked on ads and those who did not click had an equal number which is 500

```
# Creating histogram plots to visually view the categorical variables
par(mfrow=c(3,1))
categ <- c("male","clicked.on.ad")
for(i in categ) {
    counts <- table(df[,i])
    name <- names(df)[i]
    barplot(counts, main=name, col = heat.colors(5))}
```




```
#finding mean of age column
x <-mean(df$age)
print(paste(x, "is the mean for age column"))
```

```
## [1] "36.009 is the mean for age column"
```

```
#median
y <-median(df$age)
print(paste(y, "is the median for age column"))
```

```
## [1] "35 is the median for age column"
```

```
#mode
getmode <- function(v) {
   uniqv <- unique(v)
   uniqv[which.max(tabulate(match(v, uniqv)))]
}
z <- getmode(df$age)
print(paste(z, "is the mode for age column"))
```

```
## [1] "31 is the mode for age column"
```

```r
#minimun
a <-min(df$age)
print(paste(a, "is the minimum value for age column"))
```

```
## [1] "19 is the minimum value for age column"
```

```r
#maximum
b <-max(df$age)
print(paste(b, "is the maximum value for age column"))
```

```
## [1] "61 is the maximum value for age column"
```

```r
#range

c <-range(df$age)
print(paste(c, "is the range for age column"))
```

```
## [1] "19 is the range for age column" "61 is the range for age column"
```

```r
#quarntile
c <-range(df$age)
print(paste(c, "is the quarntile for age column"))
```

```
## [1] "19 is the quarntile for age column" "61 is the quarntile for age column"
```

```r
#standard deviation
d <-sd(df$age)
print(paste(c, "is the standard dev for age column"))
```

```
## [1] "19 is the standard dev for age column"
## [2] "61 is the standard dev for age column"
```

```r
#variance
e <-var(df$age)
print(paste(e, "is the var for age column"))
```

```
## [1] "77.1861051051051 is the var for age column"
```

```r
#finding mean of daily.time.spent.on.site column
x <-mean(df$daily.time.spent.on.site)
print(paste(x, "is the mean for daily.time.spent.on.site column"))
```

```
## [1] "65.0002 is the mean for daily.time.spent.on.site column"
```

```r
#median
y <-median(df$daily.time.spent.on.site)
print(paste(y, "is the median for daily.time.spent.on.site column"))
```

```
## [1] "68.215 is the median for daily.time.spent.on.site column"
```

```r
#mode
getmode <- function(v) {
   uniqv <- unique(v)
   uniqv[which.max(tabulate(match(v, uniqv)))]
}
z <- getmode(df$daily.time.spent.on.site)
print(paste(z, "is the mode for daily.time.spent.on.site column"))
```

```
## [1] "62.26 is the mode for daily.time.spent.on.site column"
```

```r
#minimun
a <-min(df$daily.time.spent.on.site)
print(paste(a, "is the minimum value for daily.time.spent.on.site column"))
```

```
## [1] "32.6 is the minimum value for daily.time.spent.on.site column"
```

```r
#maximum
b <-max(df$daily.time.spent.on.site)
print(paste(b, "is the maximum value for daily.time.spent.on.site column"))
```

```
## [1] "91.43 is the maximum value for daily.time.spent.on.site column"
```

```r
#range

c <-range(df$daily.time.spent.on.site)
print(paste(c, "is the range for daily.time.spent.on.site column"))
```

```
## [1] "32.6 is the range for daily.time.spent.on.site column"
## [2] "91.43 is the range for daily.time.spent.on.site column"
```

```r
#quarntile
c <-range(df$daily.time.spent.on.site)
print(paste(c, "is the quarntile for daily.time.spent.on.site column"))
```

```
## [1] "32.6 is the quarntile for daily.time.spent.on.site column"
## [2] "91.43 is the quarntile for daily.time.spent.on.site column"
```

```r
#standard deviation
d <-sd(df$daily.time.spent.on.site)
print(paste(c, "is the standard dev for daily.time.spent.on.site column"))
```

```
## [1] "32.6 is the standard dev for daily.time.spent.on.site column"
## [2] "91.43 is the standard dev for daily.time.spent.on.site column"
```

```r
#variance
e <-var(df$daily.time.spent.on.site)
print(paste(e, "is the var for daily.time.spent.on.site column"))
```

```
## [1] "251.337094854855 is the var for daily.time.spent.on.site column"
```

```r
#finding mean of daily.internet.usage
x <-mean(df$daily.internet.usage)
print(paste(x, "is the mean for daily.internet.usage column"))
```

```
## [1] "180.0001 is the mean for daily.internet.usage column"
```

```r
#median
y <-median(df$daily.internet.usage)
print(paste(y, "is the median for daily.internet.usage column"))
```

```
## [1] "183.13 is the median for daily.internet.usage column"
```

```r
#mode
getmode <- function(v) {
   uniqv <- unique(v)
   uniqv[which.max(tabulate(match(v, uniqv)))]
}
z <- getmode(df$daily.internet.usage)
print(paste(z, "is the mode for daily.internet.usage column"))
```

```
## [1] "167.22 is the mode for daily.internet.usage column"
```

```r
#minimun
a <-min(df$daily.internet.usage)
print(paste(a, "is the minimum value for daily.internet.usage column"))
```

```
## [1] "104.78 is the minimum value for daily.internet.usage column"
```

```r
#maximum
b <-max(df$daily.internet.usage)
print(paste(b, "is the maximum value for daily.internet.usage column"))
```

```
## [1] "269.96 is the maximum value for daily.internet.usage column"
```

```r
#range

c <-range(df$daily.internet.usage)
print(paste(c, "is the range for daily.internet.usage column"))
```

```
## [1] "104.78 is the range for daily.internet.usage column"
## [2] "269.96 is the range for daily.internet.usage column"
```

```r
#quarntile
c <-range(df$daily.internet.usage)
print(paste(c, "is the quarntile for daily.internet.usage column"))
```

```
## [1] "104.78 is the quarntile for daily.internet.usage column"
## [2] "269.96 is the quarntile for daily.internet.usage column"
```

```r
#standard deviation
d <-sd(df$daily.internet.usage)
print(paste(c, "is the standard dev for daily.internet.usage column"))
```

```
## [1] "104.78 is the standard dev for daily.internet.usage column"
## [2] "269.96 is the standard dev for daily.internet.usage column"
```

```r
#variance
e <-var(df$daily.internet.usage)
print(paste(e, "is the var for daily.internet.usage column"))
```

```
## [1] "1927.41539618619 is the var for daily.internet.usage column"
```

```r
#finding mean of clicked.on.ad
x <-mean(df$clicked.on.ad)
print(paste(x, "is the mean for clicked.on.ad column"))
```

```
## [1] "0.5 is the mean for clicked.on.ad column"
```

```r
#median
y <-median(df$clicked.on.ad)
print(paste(y, "is the median for clicked.on.ad column"))
```

```
## [1] "0.5 is the median for clicked.on.ad column"
```

```r
#mode
getmode <- function(v) {
   uniqv <- unique(v)
   uniqv[which.max(tabulate(match(v, uniqv)))]
}
z <- getmode(df$clicked.on.ad)
print(paste(z, "is the mode for clicked.on.ad column"))
```

```
## [1] "0 is the mode for clicked.on.ad column"
```

```r
#minimun
a <-min(df$clicked.on.ad)
print(paste(a, "is the minimum value for clicked.on.ad column"))
```

```
## [1] "0 is the minimum value for clicked.on.ad column"
```

```r
#maximum
b <-max(df$clicked.on.ad)
print(paste(b, "is the maximum value for clicked.on.ad column"))
```

```
## [1] "1 is the maximum value for clicked.on.ad column"
```

```r
#range

c <-range(df$clicked.on.ad)
print(paste(c, "is the range for clicked.on.ad column"))
```

```
## [1] "0 is the range for clicked.on.ad column"
## [2] "1 is the range for clicked.on.ad column"
```

```r
#quarntile
c <-range(df$clicked.on.ad)
print(paste(c, "is the quarntile for clicked.on.ad column"))
```

```
## [1] "0 is the quarntile for clicked.on.ad column"
## [2] "1 is the quarntile for clicked.on.ad column"
```

```r
#standard deviation
d <-sd(df$clicked.on.ad)
print(paste(c, "is the standard dev for clicked.on.ad column"))
```

```
## [1] "0 is the standard dev for clicked.on.ad column"
## [2] "1 is the standard dev for clicked.on.ad column"
```

```r
#variance
e <-var(df$clicked.on.ad)
print(paste(e, "is the var for clicked.on.ad column"))
```

```
## [1] "0.25025025025025 is the var for clicked.on.ad column"
```

```r
#finding mean of male

x <-mean(df$male)
print(paste(x, "is the mean for male column"))
```

```
## [1] "0.481 is the mean for male column"
```

```r
#median
y <-median(df$male)
print(paste(y, "is the median for male column"))
```

```
## [1] "0 is the median for male column"
```

```r
#mode
getmode <- function(v) {
   uniqv <- unique(v)
   uniqv[which.max(tabulate(match(v, uniqv)))]
}
z <- getmode(df$male)
print(paste(z, "is the mode for male column"))
```

```
## [1] "0 is the mode for male column"
```

```r
#minimun
a <-min(df$male)
print(paste(a, "is the minimum value for male column"))
```

```
## [1] "0 is the minimum value for male column"
```

```r
#maximum
b <-max(df$male)
print(paste(b, "is the maximum value for male column"))
```

```
## [1] "1 is the maximum value for male column"
```

```r
#range

c <-range(df$male)
print(paste(c, "is the range for male column"))
```

```
## [1] "0 is the range for male column" "1 is the range for male column"
```

```r
#quarntile
c <-range(df$male)
print(paste(c, "is the quarntile for male column"))
```

```
## [1] "0 is the quarntile for male column" "1 is the quarntile for male column"
```

```r
#standard deviation
d <-sd(df$male)
print(paste(c, "is the standard dev for male column"))
```

```
## [1] "0 is the standard dev for male column"
## [2] "1 is the standard dev for male column"
```

```r
#variance
e <-var(df$male)
print(paste(e, "is the var for male column"))
```

```
## [1] "0.249888888888889 is the var for male column"
```

```r
#finding mean of area.income

x <-mean(df$area.income)
print(paste(x, "is the mean for area.income column"))
```

```
## [1] "55000.00008 is the mean for area.income column"
```

```r
#median
y <-median(df$area.income)
print(paste(y, "is the median for area.income column"))
```

```
## [1] "57012.3 is the median for area.income column"
```

```r
#mode
getmode <- function(v) {
   uniqv <- unique(v)
   uniqv[which.max(tabulate(match(v, uniqv)))]
}
z <- getmode(df$area.income)
print(paste(z, "is the mode for area.income column"))
```

```
## [1] "61833.9 is the mode for area.income column"
```

```r
#minimun
a <-min(df$area.income)
print(paste(a, "is the minimum value for area.income column"))
```

```
## [1] "13996.5 is the minimum value for area.income column"
```

```r
#maximum
b <-max(df$area.income)
print(paste(b, "is the maximum value for area.income column"))
```

```
## [1] "79484.8 is the maximum value for area.income column"
```

```r
#range

c <-range(df$area.income)
print(paste(c, "is the range for area.income column"))
```

```
## [1] "13996.5 is the range for area.income column"
## [2] "79484.8 is the range for area.income column"
```

```r
#quarntile
c <-range(df$area.income)
print(paste(c, "is the quarntile for area.income column"))
```

```
## [1] "13996.5 is the quarntile for area.income column"
## [2] "79484.8 is the quarntile for area.income column"
```

```r
#standard deviation
d <-sd(df$area.income)
print(paste(c, "is the standard dev for area.income column"))
```

```
## [1] "13996.5 is the standard dev for area.income column"
## [2] "79484.8 is the standard dev for area.income column"
```

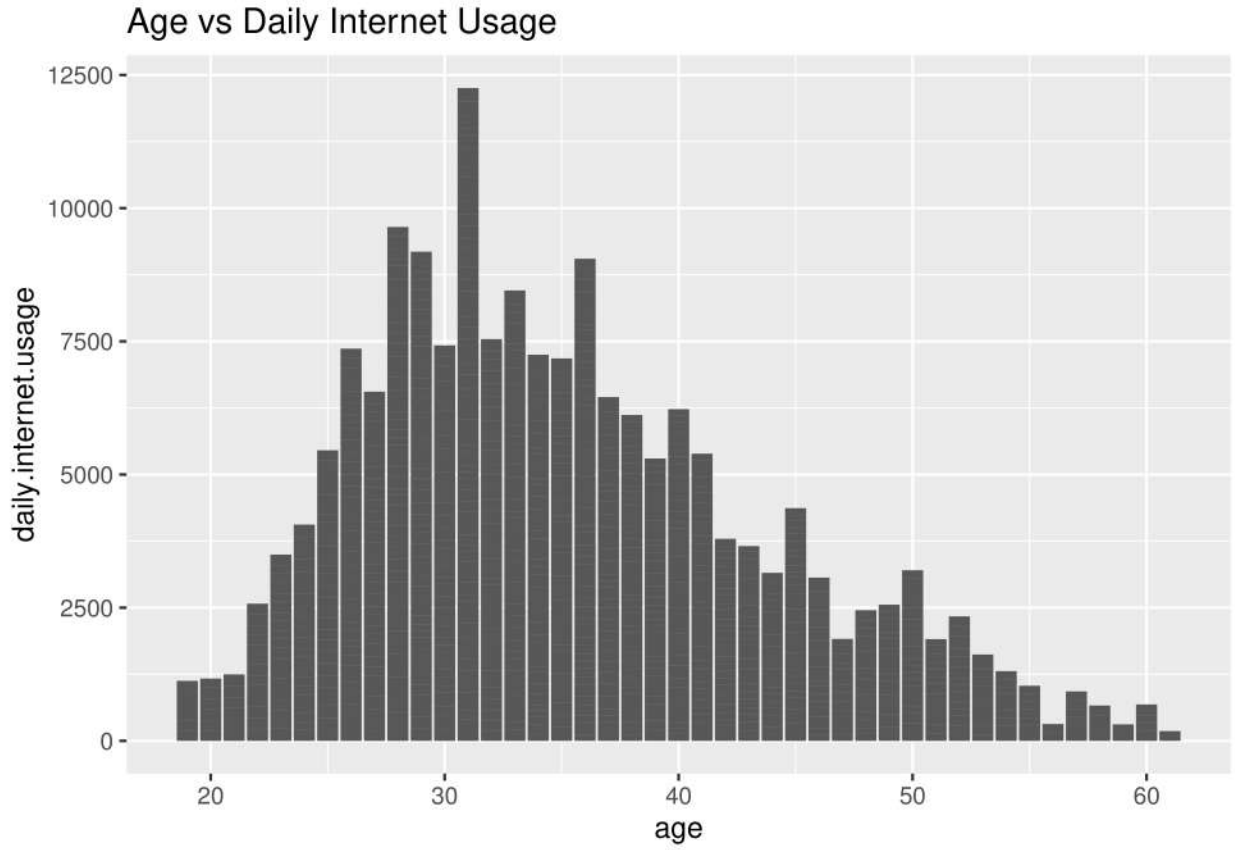
```r
#variance
e <-var(df$area.income)
print(paste(e, "is the var for area.income column"))
```

```
## [1] "179952405.951775 is the var for area.income column"
```

Bivariate analysis

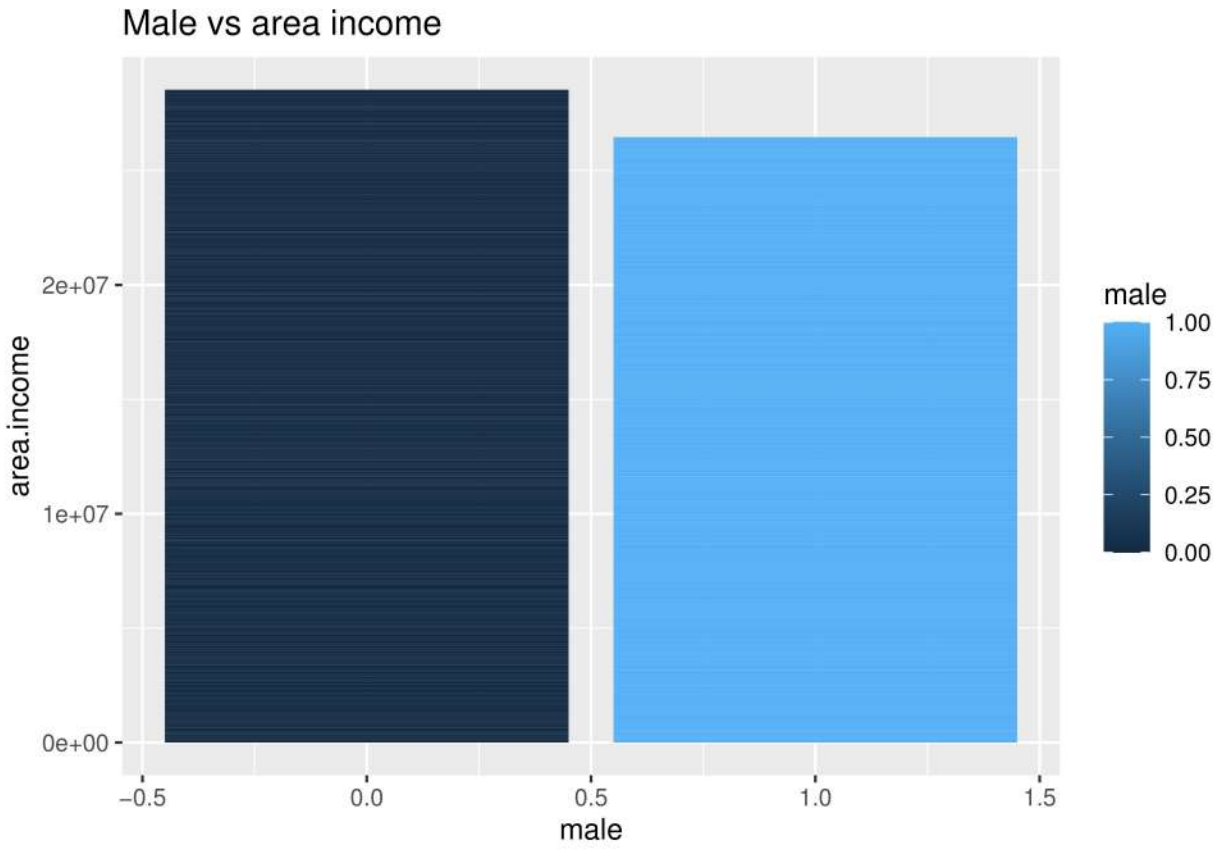
```
library(ggplot2)
```

```
p<-ggplot(data=df, aes(x=age, y=daily.internet.usage)) +
  geom_bar(stat="identity" ) + ggtitle("Age vs Daily Internet Usage")
p
```

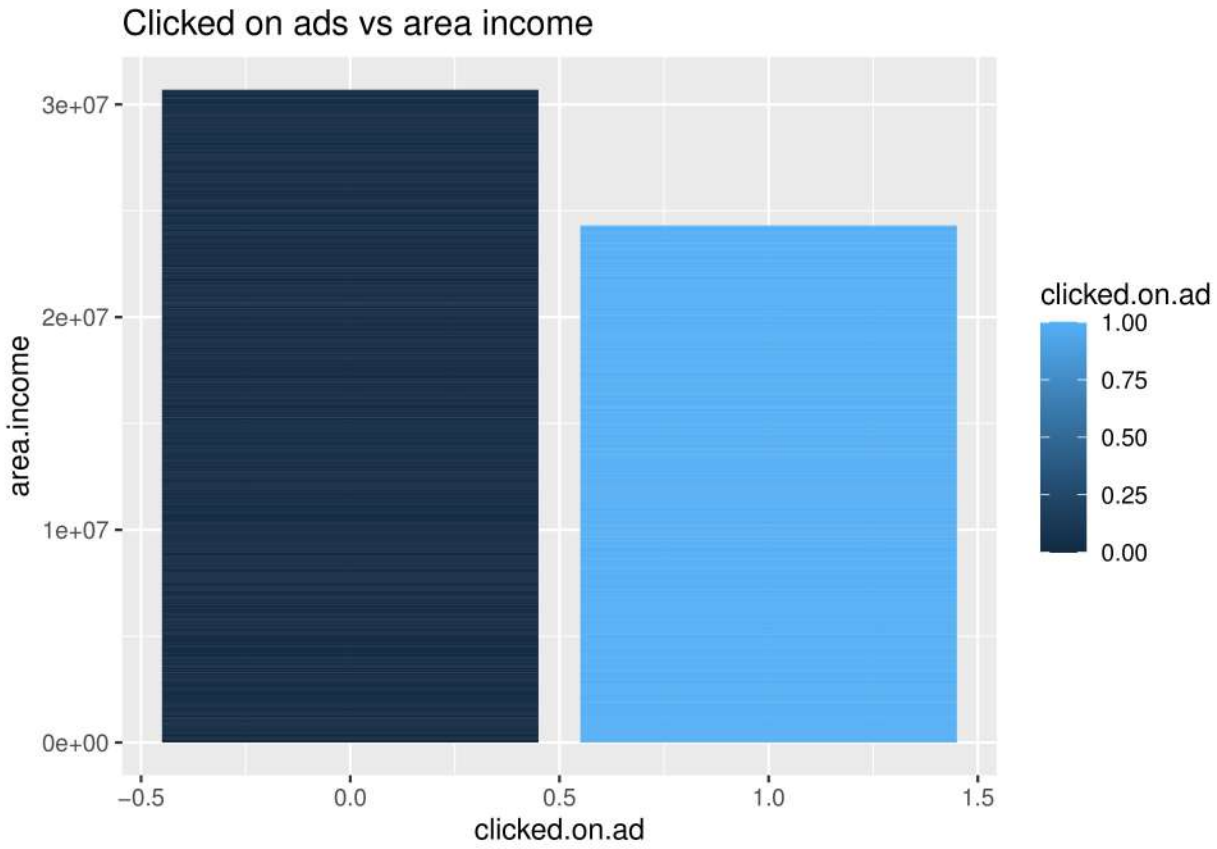


The age between 20 and 45 have the highest number of records of daily internet usage

```
#gender vs area income
p<-ggplot(data=df, aes(x=male, y=area.income,fill=male)) +
  geom_bar(stat="identity") + ggtitle("Male vs area income")
p
```




```
#clicked on ad vs area income
p<-ggplot(data=df, aes(x=clicked.on.ad
, y=area.income, fill=clicked.on.ad)) +
  geom_bar(stat="identity") + ggtitle("Clicked on ads vs area income")
p
```




```
#checking the covariance between age an click on ad
x <- df$age
y <- df$clicked.on.ad
cov(x,y)
```

```
## [1] 2.164665
```

```
cor(x,y)
```

```
## [1] 0.4925313
```

there is a positive linear relationship between the 2 variables as well as some correlation between the variables

```
#checking the covariance and correlation between area income and click on ad
x <- df$area.income
y <- df$clicked.on.ad
cov(x,y)
```

```
## [1] -3195.989
```

```
cor(x,y)
```

```
## [1] -0.4762546
```

there is a negative linear relationship between the 2 variables as well as a very low correlation between the variables

```
#checking the covariance and correlation between male and click on ad
x <- df$male
y <- df$clicked.on.ad
cov(x,y)
```

```
## [1] -0.00950951
```

```
cor(x,y)
```

```
## [1] -0.03802747
```

there is a negative linear relationship between the 2 variables as well as a very low correlation between the variables

```
#checking the covariance and correlation between daily.time.spent.on.site and click on ad
x <- df$daily.time.spent.on.site
y <- df$clicked.on.ad
cov(x,y)
```

```
## [1] -5.933143
```

```
cor(x,y)
```

```
## [1] -0.7481166
```

there is a negative linear relationship between the 2 variables as well as a very low correlation between the variables

```
#checking the covariance and correlation between daily.internet.usage and click on ad
x <- df$daily.internet.usage
y <- df$clicked.on.ad
cov(x,y)
```

```
## [1] -17.27409
```

```
cor(x,y)
```

```
## [1] -0.7865392
```

there is a negative linear relationship between the 2 variables as well as a very low correlation between the variables

IMPLEMENTING THE SOLUTION

Multiple Linear Regression

```r
# creating a subset of data for modelling
df1 = df[, c(1,2,3,4,6,9)]
head(df1)
```

```
##   daily.time.spent.on.site age area.income daily.internet.usage male
## 1                    68.95  35    61833.90                256.09    0
## 2                    80.23  31    68441.85                193.77    1
## 3                    69.47  26    59785.94                236.50    0
## 4                    74.15  29    54806.18                245.89    1
## 5                    68.37  35    73889.99                225.58    0
## 6                    59.99  23    59761.56                226.74    1
##   clicked.on.ad
## 1             0
## 2             0
## 3             0
## 4             0
## 5             0
## 6             0
```

```r
# performing multilinear regression using the lm() function.
multi_lm <- lm(clicked.on.ad ~ ., df1)
```

```r
# Generating the anova table
anova(multi_lm)
```

```
## Analysis of Variance Table
##
## Response: clicked.on.ad
##                            Df  Sum Sq Mean Sq   F value  Pr(>F)
## daily.time.spent.on.site    1 139.920 139.920 3162.2238 < 2e-16 ***
## age                         1  16.793  16.793  379.5306 < 2e-16 ***
## area.income                 1  13.721  13.721  310.0920 < 2e-16 ***
## daily.internet.usage        1  35.372  35.372  799.4083 < 2e-16 ***
## male                        1   0.213   0.213    4.8183 0.02839 *
## Residuals                 994  43.982   0.044
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# Then performing our prediction
pred<- predict(multi_lm, df1)
# Printing out our result
pred
```

```
##            1            2            3            4            5
##  0.0120592776  0.0895564625  0.0402466412 -0.0405988125  0.1055426814
##            6            7            8            9           10
##  0.1568158327  0.0390399144  1.0211693137  0.0049968479  0.2545159589
##           11           12           13           14           15
##  1.2156842338 -0.0512961722  0.9066043929  0.0605044956  1.1091386404
##           16           17           18           19           20
##  0.6122179695  1.1037856216  0.1994929475  1.0881936474  0.8245685140
```

```
##           21           22           23           24           25
## -0.0197726949 -0.0506531149  1.1763651360  0.0011952071  0.8420263187
##           26           27           28           29           30
##  0.0460146206  0.9693459870  1.0713952632  0.8869066511  0.1448351966
##           31           32           33           34           35
##  0.0060721389  0.0524592611  0.9877329322  0.4990332022  1.0686408181
##           36           37           38           39           40
##  0.0593194754  0.9484999144  0.1775342841  1.0651803529  1.1384586005
##           41           42           43           44           45
##  0.0599385457  0.0596802640  0.0579511220  0.0283183089  0.0709396914
##           46           47           48           49           50
##  1.0245123439  0.1896572706  0.1429649914  1.0682664047  1.1415786055
##           51           52           53           54           55
##  0.1351204708 -0.0090050595  0.9095358548  0.9970879356  0.9188366090
##           56           57           58           59           60
##  0.2053421465  0.5411286044  1.1699890691 -0.0288457591  1.1659269280
##           61           62           63           64           65
##  0.1964256103  0.0725232326  0.3155883872  0.1442848605  0.9727944231
##           66           67           68           69           70
##  0.2408225625  0.9802052606  0.7728655666  0.0260237942  0.7155345209
##           71           72           73           74           75
##  0.8937233176  0.2076451100  0.6496740869  1.0985959111  1.0374947670
##           76           77           78           79           80
##  0.1770243775  0.9729792846 -0.0765176580  1.1034127749  1.0765889714
##           81           82           83           84           85
##  0.0310842511  0.0140048006  0.9349737258  0.9922982344  0.2528463020
##           86           87           88           89           90
##  1.0893161721  0.1310293311  0.9252297681  1.0384697990  1.0784593033
##           91           92           93           94           95
##  0.6320743945  0.6554918711  0.0223578354  0.9399208813  0.9657906021
##           96           97           98           99          100
##  0.0280097070  1.0688503234  0.9429255925  1.1137468544  0.0228437894
##          101          102          103          104          105
##  1.1667140701  0.1144525312  0.1280118294  0.1494212627  0.2230148169
##          106          107          108          109          110
## -0.0111177190  0.0245142898  0.9385302280  1.0535833943  0.0953977127
##          111          112          113          114          115
##  0.6916910108  0.3034144226  0.1372477474  1.2183812765  0.0717056133
##          116          117          118          119          120
## -0.0626908748  0.6041787024  0.9029864966  0.4039356831  0.4682298445
##          121          122          123          124          125
## -0.0069401334  0.3371980546  0.0477479930  0.7452611944  0.8041342258
##          126          127          128          129          130
## -0.0236654433  0.6599491022  0.1275125918  0.0658803670  0.0993046922
##          131          132          133          134          135
##  1.1005613429  1.0693832287  0.5753483260  0.0498192492  1.1912251787
##          136          137          138          139          140
##  1.0402343608  1.1477010056  1.1428650779  0.2767730571  0.4462145661
##          141          142          143          144          145
##  0.1175537350  0.8898978522  0.9018106321  0.0002616096  0.2071121626
##          146          147          148          149          150
##  1.1665937683  0.9221977565  0.8626737072  0.8413750413  0.7709288740
##          151          152          153          154          155
##  0.4056350875  0.2651534440  1.1777907751  0.0556651602  0.1424773230
```

```
##          156          157          158          159          160
## -0.0229250698  1.0483599529  0.5560198490  0.0181172395  0.3756525940
##          161          162          163          164          165
##  0.1320780295 -0.0407335415 -0.0110829052  0.3413661832  1.1765280550
##          166          167          168          169          170
##  0.8299347084  1.0269699388 -0.0309483276  1.0734929080  0.0762109893
##          171          172          173          174          175
##  1.1549644005  0.2622844750  0.1080217348  0.3349321707  0.7737704012
##          176          177          178          179          180
##  0.0610597615  1.1055897502  0.1631895014  1.0503176286  0.0305542407
##          181          182          183          184          185
##  1.1788234532  0.3324889248  0.9892208983  0.0369270183  0.0360344481
##          186          187          188          189          190
##  1.2069244592  1.1157334501 -0.0548125872  0.4392637814  1.1535914431
##          191          192          193          194          195
##  1.0285601890  1.0228979381  1.0958418600  1.1563413673 -0.0527815679
##          196          197          198          199          200
##  0.7449299060  0.9975786847 -0.0567486459  0.1089769541  0.1486264353
##          201          202          203          204          205
##  0.2116603188 -0.0068872540  1.0817212336  0.2329832034  0.0908020792
##          206          207          208          209          210
##  0.9577106571  0.0494705055  0.1027191508  1.0858394430  1.1583124713
##          211          212          213          214          215
## -0.0001122570  0.8316098902 -0.0377311447  0.5145808402  0.0092010590
##          216          217          218          219          220
##  0.6606863845  1.0875697555  1.0661076748  0.8004935507  1.0665495332
##          221          222          223          224          225
## -0.0362325171  0.1090259522  0.8770395307  0.6073683932  0.2757138201
##          226          227          228          229          230
##  0.5364477713  0.9496579553  1.0816232178  0.1761124366  0.1226149018
##          231          232          233          234          235
##  0.0744750735  1.0589576823  0.6346171736 -0.0062335349  0.8526799399
##          236          237          238          239          240
##  1.2140124072  0.6398740350  0.4610079819  0.5968493409 -0.0197655466
##          241          242          243          244          245
##  0.7381102932  1.0616211742  0.0701003036  0.0416875743  0.2504575666
##          246          247          248          249          250
##  0.3309803786  1.0356487646  0.1873182925  1.1152915885  0.7327751013
##          251          252          253          254          255
##  0.0424595124  0.8798064223  0.0203103937  0.6810119296  0.9790664416
##          256          257          258          259          260
##  0.0719175808 -0.0367544190  1.0281378485  0.0037732187  0.7460384180
##          261          262          263          264          265
##  0.0691997061  0.9359641679  0.6226745091  1.1848566426 -0.0247805245
##          266          267          268          269          270
##  0.7336332951  0.5688149491  0.0411352646  0.8052682995  0.0758012723
##          271          272          273          274          275
##  0.9129762696  0.0556800444 -0.0340198240  0.3328347128 -0.0020028086
##          276          277          278          279          280
##  0.9686805304  0.0051611080  0.1785927788  0.3212166192  0.0624929601
##          281          282          283          284          285
##  1.2207657078  0.8427293992  1.1001523662  0.0284701066  1.1121161629
##          286          287          288          289          290
##  0.0746667683  0.8700666300  0.0390001899  0.9563363075  0.4573241970
```

```
##           291           292           293           294           295
##  0.5918180196  0.0260743970  0.6120204105 -0.0292750204  0.3574435104
##           296           297           298           299           300
## -0.0153694912  0.0038873516  0.0434949564  0.0198796977  0.0687825487
##           301           302           303           304           305
##  0.0100907474  0.7000176955  0.9702481093  0.5491730709  1.1346666194
##           306           307           308           309           310
##  0.1861303870  0.0104165035 -0.0294820740  0.1345355775  1.0952627673
##           311           312           313           314           315
##  0.0067193372  0.1656607166  0.3901771334 -0.0429821022  0.0009399734
##           316           317           318           319           320
##  0.8402167334  0.2437431821 -0.0048703766  0.1504052986  1.0848001311
##           321           322           323           324           325
##  0.8294443862  0.1896182250  0.2043833969  0.0659691064 -0.0089415976
##           326           327           328           329           330
##  0.9482537584  0.9057228084 -0.0119645782  0.1093908980  0.5240919184
##           331           332           333           334           335
##  0.0934575534  0.0249220757  0.8715260222 -0.0769156477  0.1334995103
##           336           337           338           339           340
##  1.0434604955  0.1492546794 -0.0319569634  0.0725549854  0.1773502720
##           341           342           343           344           345
##  1.0197481632  0.8833175988  0.0198869088  0.0237672154  0.6970220534
##           346           347           348           349           350
##  0.2162947947  0.1955124087  0.9219384416  0.2637892048  0.3980712819
##           351           352           353           354           355
## -0.0055594891  0.0815524101 -0.0115466367  0.0767346256  0.8801382960
##           356           357           358           359           360
## -0.0356091500  1.1755659389  1.1515340189  1.0856447975  0.0418491330
##           361           362           363           364           365
##  0.9367327735  0.9024012477  0.0838937925  0.5299016653  0.1003914139
##           366           367           368           369           370
##  0.7293019932  0.1387935911  0.0337813345  0.0797499457  0.0851216049
##           371           372           373           374           375
##  1.1296341521  1.1695083106  0.0624188023  1.0602804924  0.0990566520
##           376           377           378           379           380
##  0.1051195240 -0.0176090452  0.9886317927  0.6780879042 -0.0606256849
##           381           382           383           384           385
## -0.0111359636  1.1221755907  0.2768975723 -0.0041406016  0.9342317620
##           386           387           388           389           390
##  0.0121944092  0.0932267578  1.2017498950  0.1070832478  0.9893609446
##           391           392           393           394           395
## -0.0478330672  0.1096015500  0.0521491717  0.2456078325  1.0215468504
##           396           397           398           399           400
##  0.1698836577  1.1062373421  0.3449961280  0.1537965779  0.0653845474
##           401           402           403           404           405
##  1.0248729750 -0.0326261684  1.2154271439  0.0749430516  1.1099170758
##           406           407           408           409           410
## -0.0147039465  0.8521960751  0.9694381615  0.4119297054  1.1683570419
##           411           412           413           414           415
##  0.9212849746  0.0592524717  0.2584471651  1.1370511835  0.0249307498
##           416           417           418           419           420
##  1.2003945167  0.6944724757  0.0363115988  0.1633308480  0.3861557044
##           421           422           423           424           425
##  1.0350586245  0.0542317870  0.9038855918  0.9890601424  0.9857510546
```

```
##           426           427           428           429           430
##  0.8456880940  0.6372548233 -0.0367676433  0.7957962340 -0.0302701906
##           431           432           433           434           435
##  0.0463816947 -0.0058460107  0.2898981850 -0.0042863864 -0.0090296264
##           436           437           438           439           440
##  0.7788884224  0.2556324846  0.0528839050  1.0555856574  0.3076943056
##           441           442           443           444           445
##  0.8245778518 -0.0337119928  1.0633521252  1.0642517441  1.0899351854
##           446           447           448           449           450
## -0.0469740093  0.5952824141  0.1521725660  1.1445102634  0.2877921134
##           451           452           453           454           455
##  0.7941498695  0.8359859363 -0.0429270672  0.1640471587  0.4606354787
##           456           457           458           459           460
##  0.0728458035  1.1042640102  0.0390860172  0.7115921017 -0.0008332432
##           461           462           463           464           465
##  1.1580807523  1.0530462713  0.2273295054  0.9852667808  0.1651884202
##           466           467           468           469           470
##  0.7801839237  0.5714059755  0.7825584070  1.0763627085  0.0865167963
##           471           472           473           474           475
##  0.7546494500  0.1640565920  0.0916090674 -0.0098148917  1.0548713450
##           476           477           478           479           480
##  0.0196088297  0.0435404636  1.1292082206  1.0228832867  0.7685736060
##           481           482           483           484           485
##  0.2881374924  0.1522352747  0.1164456302  1.0071597756  1.1055437550
##           486           487           488           489           490
##  0.7219937696  0.3217627639  0.0542390560  1.1738249560 -0.0628101680
##           491           492           493           494           495
##  1.1989246841  1.0193698933  0.2978934553  0.8602901127  0.8162152518
##           496           497           498           499           500
## -0.0010977389 -0.0141723709  1.1928031552  0.2707106639  0.7632000161
##           501           502           503           504           505
##  1.0158588557 -0.0319192069  0.1000752232  1.0452408279  1.0632227068
##           506           507           508           509           510
##  0.0599433543 -0.0272211442  1.0234705341  0.5868972853 -0.0590210277
##           511           512           513           514           515
##  0.8439067717  0.0134796013 -0.0198657182  1.0545234227 -0.0108834850
##           516           517           518           519           520
##  1.0264801808 -0.0161032230  0.5636999317  1.1092999931  1.1110735455
##           521           522           523           524           525
##  0.9666619195  0.7586108868  0.0572882750  0.7289321505 -0.0538867020
##           526           527           528           529           530
##  0.4792570008  0.9770639119  0.2871098272  1.1853952110  0.3250423985
##           531           532           533           534           535
##  0.8983088884  1.1390035046  0.1761468639  0.0658009087  0.0852130322
##           536           537           538           539           540
##  0.0326948724  0.2353807859  0.1878832391 -0.0395522565  0.0113959359
##           541           542           543           544           545
##  0.0037367848 -0.0374207234  0.1380131612  1.0101440804  0.0590776191
##           546           547           548           549           550
##  1.0321104432  0.0460754923 -0.0201187666 -0.0371234704  0.0034100283
##           551           552           553           554           555
##  0.0395053030  0.0118090105  1.0977449489  1.0089748204  1.0269536977
##           556           557           558           559           560
##  0.0815245969  1.0657415519 -0.0137496607  0.2927443305 -0.0580825324
```

```
##          561          562          563          564          565
##  1.0327232336 1.1805128004 0.1755969146 0.4887731996 0.7957156080
##          566          567          568          569          570
##  0.1330525625 1.1713750755 -0.0428822002 0.3374260359 -0.0135503270
##          571          572          573          574          575
##  1.1905171907 -0.0309790676 -0.0133501717 0.4508971464 1.1911811644
##          576          577          578          579          580
##  1.1329113343 1.0973608488 -0.0242088521 0.1370014548 0.1062376443
##          581          582          583          584          585
##  1.0192801063 0.9847192701 0.8919933295 0.5942505349 0.5113194091
##          586          587          588          589          590
##  0.0898909878 -0.0347254217 1.2531383174 -0.0331406891 0.8861023254
##          591          592          593          594          595
##  0.5732503020 1.0155118722 0.0585068484 0.1430983364 0.8532062269
##          596          597          598          599          600
##  0.7893621169 -0.0286636255 -0.0419945004 0.0694203462 0.7205412207
##          601          602          603          604          605
##  0.6299603118 1.1867281831 0.8698864091 0.0935970439 0.8904132828
##          606          607          608          609          610
##  1.0321789598 0.1587962114 0.0424240981 0.8410717193 0.2752486374
##          611          612          613          614          615
##  0.9486142408 0.9648141370 0.0962470657 -0.0279117587 0.0355549336
##          616          617          618          619          620
##  1.1041610981 1.1447434347 0.0291167165 1.0903658209 0.0595128688
##          621          622          623          624          625
##  0.1240812798 0.0056596767 0.9651283500 0.3811460377 0.1077120980
##          626          627          628          629          630
##  0.5053465879 0.0930879532 1.0325064809 1.1135743834 -0.0173615592
##          631          632          633          634          635
##  0.0724256343 -0.0426394471 0.0687697627 1.0644298816 0.9813566789
##          636          637          638          639          640
##  1.0020855787 1.1165235176 0.0882801075 0.6431069609 0.1298561780
##          641          642          643          644          645
##  0.9250440371 0.1591111462 -0.0003365608 0.3368821749 0.0577149341
##          646          647          648          649          650
##  0.9695514922 1.2073506576 0.8822469895 0.2117012088 0.0113033882
##          651          652          653          654          655
##  0.2331205777 -0.0357650403 -0.0448400360 0.1001533300 -0.0075346618
##          656          657          658          659          660
##  1.1995161347 0.1236978913 0.1452355756 0.1079059979 -0.0110369011
##          661          662          663          664          665
##  0.8373694595 0.4083612928 1.1979185154 0.9267813469 0.0228485494
##          666          667          668          669          670
##  0.6974712192 -0.0263230238 -0.0328793097 0.3637851032 0.7375738777
##          671          672          673          674          675
##  0.0892299334 0.6487520703 0.0841240990 0.9585884919 0.0575640407
##          676          677          678          679          680
## -0.0249317912 0.8613584984 0.7927229432 0.3652969335 1.0500022603
##          681          682          683          684          685
##  0.0718042237 0.9175829881 1.0720350876 0.1432062266 1.1169704330
##          686          687          688          689          690
## -0.0172674262 -0.0404155674 0.0405060422 0.0287700519 -0.0225702229
##          691          692          693          694          695
##  0.0774749015 -0.0365951407 0.8221749465 0.8633473273 0.1844081474
```

```
##          696          697          698          699          700
##  0.0105968025  1.0574014616  0.0387259731  0.1506153132  0.0054551079
##          701          702          703          704          705
##  0.3844775841  0.8492471680  0.0378489197  0.0638255647  0.0887143164
##          706          707          708          709          710
##  0.0099016982  0.5196890468  0.0308341538  0.7328165693  0.7508979320
##          711          712          713          714          715
##  1.1869871924  0.0835397514  0.2847647546  1.1691556156  0.1102069258
##          716          717          718          719          720
##  0.9965789487  1.0211087390  0.0997352461  0.4170355587  0.8877928510
##          721          722          723          724          725
## -0.0323598225  0.9537220330  0.8958695099  0.4674847655 -0.0036157817
##          726          727          728          729          730
## -0.0084645774  0.0743398450 -0.0626506493  0.2109505337 -0.0327847251
##          731          732          733          734          735
##  0.2961392608  0.0747600861  0.1541912233  1.1793893841  0.7757517465
##          736          737          738          739          740
## -0.0053564263 -0.0523107015  0.9631889116  0.4981646855  0.0400429638
##          741          742          743          744          745
##  0.9697938204  0.2057956831  0.1574715469  0.9472601173  0.7897734983
##          746          747          748          749          750
##  0.9748675074  0.0388180966  1.0799577951  0.9773996329  0.6525063047
##          751          752          753          754          755
##  0.8619117961 -0.0672620994  0.2112350578  0.1772332197  0.0799092040
##          756          757          758          759          760
##  0.0989415551  0.9671390953  0.9077574465  0.8064857849  0.4224738124
##          761          762          763          764          765
##  0.0531462280  0.1922082064  1.1032158266  0.8521352344  1.1833781541
##          766          767          768          769          770
##  1.0278232095  0.6198583174  1.1188855207  0.7814499740  0.2036400549
##          771          772          773          774          775
##  0.0546898912  0.3527352861  0.0453896081  0.5467403235  1.0225506426
##          776          777          778          779          780
##  1.1307314611  0.6136032117  0.0423085395  0.9326863553  0.0157828388
##          781          782          783          784          785
##  0.2285916510  0.7878198933 -0.0255730576  0.2237113899  1.0460110211
##          786          787          788          789          790
##  1.1208353926  0.3445668646  0.3219543848  0.0922733236  1.0366708974
##          791          792          793          794          795
##  0.9574822488  1.1545888710  0.2890989628  0.9282292471  1.2133669943
##          796          797          798          799          800
##  0.0643164170  0.0710851361 -0.0195192475  0.3249110379  0.0246893084
##          801          802          803          804          805
##  0.8212958684  1.0419099126  1.2027316741  1.2484461826  1.1389137144
##          806          807          808          809          810
##  0.1345286901  0.8978030440  1.1768322892  1.0637367340  1.0142012356
##          811          812          813          814          815
##  0.9224476755 -0.0538644435  0.1952118534  0.1073107199 -0.0168352413
##          816          817          818          819          820
## -0.0086427230  1.0902093354  0.7850933339  0.0290891489  0.1657640018
##          821          822          823          824          825
##  1.0637300123 -0.0376005707 -0.0314715337  0.0228083198  0.0781333826
##          826          827          828          829          830
##  0.1240773830 -0.0481248077  1.2044932209  0.8992165712  1.0251558725
```

```
##           831           832           833           834           835
##  0.9054051622  1.0050734507  1.1256026957  1.0160460336  0.0705359166
##           836           837           838           839           840
##  0.0467187180  1.1803569226  1.0434324557  1.1907691134  1.0853162642
##           841           842           843           844           845
##  0.7937223496  1.1158610794  0.1318746739  0.0109065344  0.0059197729
##           846           847           848           849           850
##  1.0089779387  1.0831692502  0.2441045322 -0.0017770283  1.0586704549
##           851           852           853           854           855
## -0.0648191117  1.0191851236  1.1008091143  0.0282178472  0.0889422981
##           856           857           858           859           860
##  0.4832264037 -0.0474485992  0.0524900609  1.0958635006 -0.0189879336
##           861           862           863           864           865
##  0.2332575581  0.1041189263  0.2978113682 -0.0299773402  0.1092306642
##           866           867           868           869           870
##  1.0735256268  0.0571053153  0.0843687710  0.0585156548  0.0027268448
##           871           872           873           874           875
##  0.6372760986 -0.0161181102  0.1722260598 -0.0505255422  0.2980539138
##           876           877           878           879           880
##  0.9380866074  0.9613638542  0.2455454670  0.0125655356  0.1511963091
##           881           882           883           884           885
##  1.0113758122 -0.0332107295  0.0654036535  0.9020172618 -0.0023888229
##           886           887           888           889           890
##  1.2499040201  1.1800293228  1.0446872593  0.0253634744  0.9995512634
##           891           892           893           894           895
## -0.0250757965  0.5344091112  0.7746080049  0.0579910029  0.0780028664
##           896           897           898           899           900
##  0.1657499839 -0.0649648256  0.7718662122  1.1029933257  1.1733219437
##           901           902           903           904           905
##  1.1666754010  0.9109688363  1.1420484484  0.1768936447  0.0548428493
##           906           907           908           909           910
## -0.0236079203  0.8465723265  0.0934701119  1.1248912202  0.0476091964
##           911           912           913           914           915
##  1.0727992933  0.9753439231  1.1372457180  0.0032538504  1.0900947543
##           916           917           918           919           920
##  0.9812827346  0.9882804138  0.0345360271  0.0044927270  0.2817759969
##           921           922           923           924           925
## -0.0479570345  0.9502020977  1.1289443617  1.1120560081  0.7278655725
##           926           927           928           929           930
##  1.1923645003  0.1497315429  0.2833261404  0.1422021602  0.7306007394
##           931           932           933           934           935
## -0.0410520284  1.0133993287  0.8395312925  1.0383790659 -0.0749532030
##           936           937           938           939           940
##  0.1121215565  1.0947279171  0.9715735090  0.9852524241  0.4462986166
##           941           942           943           944           945
##  1.2190310751  0.4931506635  0.7385677941  1.0282868849  0.8777531231
##           946           947           948           949           950
##  0.0863656499  0.1763349863  0.9293413918  0.3889550130  0.3001948666
##           951           952           953           954           955
##  0.6984085236  0.8963121308  0.4662933740  0.8953426912  0.2470741794
##           956           957           958           959           960
##  1.1758503767  0.9022969406  0.0156798253 -0.0578553221  0.2740704507
##           961           962           963           964           965
##  0.9331497181  0.1160661035  0.1518709324  0.1491174426  0.1063455747
```

```
##           966           967           968           969           970
##  0.9935062526  1.0176269646  0.2211942024  1.1960098792  0.4771634919
##           971           972           973           974           975
##  0.9423405581  1.2254025755  1.1303380935  0.0638822541  1.1664935243
##           976           977           978           979           980
##  1.1715681533  1.0526010187  0.9219267628  0.3441220372 -0.0327087047
##           981           982           983           984           985
##  0.8863896677  0.1249754716  0.7681693138 -0.0635875055  0.0004514132
##           986           987           988           989           990
##  0.7180822399 -0.0412543770  1.0358737366  0.1696268299 -0.0101944384
##           991           992           993           994           995
##  1.1367462503  1.1569594909  0.7240366524  0.0944511586  0.7012608222
##           996           997           998           999          1000
##  0.0773967252  0.9017130876  1.1818721696  0.5211183366  0.8436990137
```

```
# checking the accuracy of the model

mean(df$clicked.on.ad == pred)
```

```
## [1] 0
```

```
library(caret)
```

cross validation

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

```
#cross validation
set.seed(40)
multiple_lm <- train(clicked.on.ad ~ ., df1,
               method = "lm",
               trControl = trainControl(method = "cv",
                                        number = 10,
                                        verboseIter = FALSE))
```

```
## Warning in train.default(x, y, weights = w, ...): You are trying to do
## regression and your outcome only has two possible values Are you trying to do
## classification? If so, use a 2 level factor as your outcome column.
```

```
summary(multiple_lm)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.65251 -0.11577 -0.03069  0.05081  1.03147
##
```

```
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)                2.309e+00  5.755e-02  40.113   <2e-16 ***
## daily.time.spent.on.site -1.279e-02  5.058e-04 -25.294   <2e-16 ***
## age                         8.983e-03  8.283e-04  10.845   <2e-16 ***
## area.income               -6.173e-06  5.351e-07 -11.536   <2e-16 ***
## daily.internet.usage      -5.260e-03  1.867e-04 -28.169   <2e-16 ***
## male                      -2.926e-02  1.333e-02  -2.195   0.0284 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2104 on 994 degrees of freedom
## Multiple R-squared:  0.8241, Adjusted R-squared:  0.8232
## F-statistic: 931.2 on 5 and 994 DF,  p-value: < 2.2e-16
```

```
multiple_lm
```

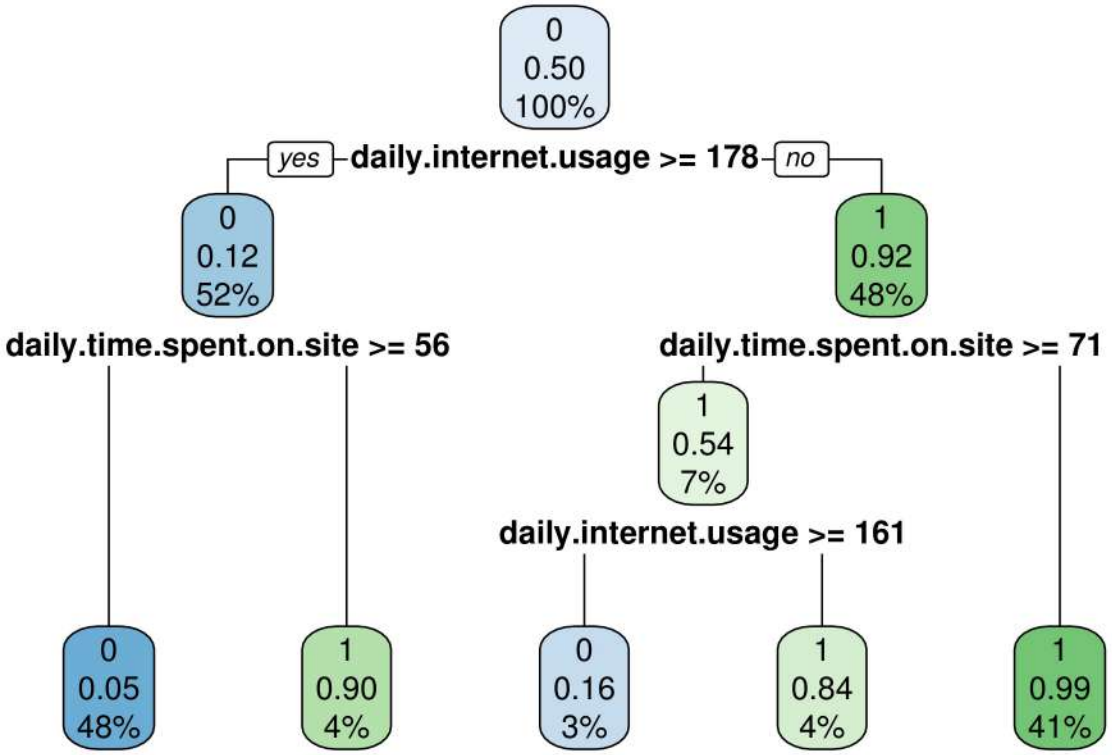
```
## Linear Regression
##
## 1000 samples
##    5 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 900, 900, 900, 900, 900, 900, ...
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.2110003  0.8237839  0.1442428
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

DECISION TREES

```
#getting the neccesary libraries
library(rpart)
library(rpart.plot)
```

```
model <- rpart(clicked.on.ad ~ ., data =  df1,
           method = "class")
```

```
# Plotting our model
rpart.plot(model)
```




```
# predicting

predd <- predict(model, df1, type = "class")
table(predd, df1$clicked.on.ad)
```

```
##
## predd   0   1
##     0 485  28
##     1  15 472
```

```
# checking the accuracy of the model

mean(df$clicked.on.ad == predd)
```

```
## [1] 0.957
```

SVM

```
#splitting the dataset into training and testing
intrain <- createDataPartition(y = df1$clicked.on.ad, p= 0.8, list = FALSE)
training <- df1[intrain,]
testing <- df1[-intrain,]
```

```r
# checking the shape of train &test
# ---
#
dim(training);
```

```
## [1] 800   6
```

```r
dim(testing)
```

```
## [1] 200   6
```

```r
#convert the click ad column to a factor

training[["clicked.on.ad"]] = factor(training[["clicked.on.ad"]])
```

```r
library(caret)
```

```r
trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)
svm <- train(clicked.on.ad ~., data = training, method = "svmLinear",
trControl=trctrl,
preProcess = c("center", "scale"),
tuneLength = 10)
```

```r
# check the results of the model
svm
```

```
## Support Vector Machines with Linear Kernel
##
## 800 samples
##   5 predictor
##   2 classes: '0', '1'
##
## Pre-processing: centered (5), scaled (5)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 720, 720, 720, 720, 720, 720, ...
## Resampling results:
##
##   Accuracy  Kappa
##   0.9675    0.935
##
## Tuning parameter 'C' was held constant at a value of 1
```

```r
# We will use the predict() method for predicting results as shown below.

#
prediction <- predict(svm, newdata = testing)
prediction
```

```
##   [1] 0 0 1 1 1 1 0 0 0 1 1 0 0 0 1 0 0 1 1 0 0 0 0 0 1 0 1 1 1 1 1 0 1 0 1 1 0
##  [38] 0 0 0 1 1 0 1 0 1 0 1 0 0 0 1 0 1 1 1 1 0 1 0 1 0 0 1 0 1 0 0 1 0 1 0 1 0
```

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

```
# checking the accuracy of the model

confusionMatrix(table(prediction, testing$clicked.on.ad))
```

```
## Confusion Matrix and Statistics
##
##
## prediction  0  1
##          0 99  4
##          1  1 96
##
##                Accuracy : 0.975
##                  95% CI : (0.9426, 0.9918)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.95
##
##  Mcnemar's Test P-Value : 0.3711
##
##             Sensitivity : 0.9900
##             Specificity : 0.9600
##          Pos Pred Value : 0.9612
##          Neg Pred Value : 0.9897
##              Prevalence : 0.5000
##          Detection Rate : 0.4950
##    Detection Prevalence : 0.5150
##       Balanced Accuracy : 0.9750
##
##        'Positive' Class : 0
##
```

Conclusion and Recommendations From the analysis done, we have realized age has a high impact on click on ads. It has a high correlation and covariance of: 2.164665& 0.4925313 respectively. Area income,Daily Internet Usage,Daily Time.Spent on.Site had a negative correlation and covariance which means they do not influence the clicking of ads.

SVM had an accuracy of 96.7 while decision tree had 95.7. As a Data Science consultant, I would recommend the support vector machine model to be used for this research since it had the highest accuracy score.