Define the Research Question

Identifying individuals most likely to click her ads.

The Metric of Success

Being able to identify individuals who are most likely to click her ads from our analysis.

The Context

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.

Experimental Design Taken

For us to meet our objective, the following experimental design was taken:

- Defining the research question.
- Setting the metric of success.
- Checking the appropriateness of the available data to answer the given question.
- Reading the dataset.
- Cleaning the dataset.
- Finding and dealing with outliers, anomalies and missing data within the dataset.
- Performing univariate and bivariate analysis.
- From the insights provide recommendations and a conclusion.

Appropriateness of the available data to answer the given question.

The data provided for analysis is very appropriate since it contains different variables which will help in answering our research question. The data also contains 1000 entries with no missing data or duplicates hence it is enough to conduct our analysis.

Reading the dataset

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

A data.table: 6 × 10

Daily Time Spent on Site	Age	Area Income	Daily Internet Usage	Ad Topic Line	City	Male	Country	Timest
<dbl>	<int>	<dbl>	<dbl>	<chr>	<chr>	<int>	<chr>	<dt
68.95	35	61833.90	256.09	Cloned 5thgeneration orchestration	Wrightburgh	0	Tunisia	2016 00:5
80.23	31	68441.85	193.77	Monitored national standardization	West Jodi	1	Nauru	2016 01:3
69.47	26	59785.94	236.50	Organic bottom-line service-desk	Davidton	0	San Marino	2016 20:3
74.15	29	54806.18	245.89	Triple-buffered reciprocal time-frame	West Terrifurt	1	Italy	2016 02:3

```
#Previewing the last 10 rows of the dataset
tail(data, n=10)
```

A data.table: 10 × 10

Daily Time Spent on Site	Age	Area Income	Daily Internet Usage	Ad Topic Line	City	Male	Country	Ti
<dbl>	<int>	<dbl>	<dbl>	<chr>	<chr>	<int>	<chr>	
35.79	44	33813.08	165.62	Enterprise-wide tangible model	North Katie	1	Tonga	
38.96	38	36497.22	140.67	Versatile mission-critical application	Mauricefurt	1	Comoros	
69.17	40	66193.81	123.62	Extended leadingedge solution	New Patrick	0	Montenegro	
64.20	27	66200.96	227.63	Phased zero tolerance extranet	Edwardsmouth	1	Isle of Man	
43.70	28	63126.96	173.01	Front-line bifurcated ability	Nicholasland	0	Mayotte	
72.97	30	71384.57	208.58	Fundamental modular algorithm	Duffystad	1	Lebanon	
[illegible]	[illegible]	[illegible]	[illegible]	Grass-roots [illegible]	[illegible]	[illegible]	Bosnia and	

Checking the dataset

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

10

Our dataset has 10 columns.

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

1000

Our dataset has 1000 rows.

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

1000 · 10

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

10

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

```
Classes ‘data.table’ and '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 natio
 $ 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 01:39:0
 $ Clicked on Ad           : int  0 0 0 0 0 0 0 1 0 0 ...
 - attr(*, ".internal.selfref")=<externalptr>
```

We can see the data types of the various variables in the dataset. Male and Clicked on Ad have the wrong data type and hence need to be corrected.

```
#Splitting the timestamp dataset to year, month, day and hour so that we can get as much info
data$Year <- format(as.POSIXct(data$Timestamp, format="%Y-%m-%d %H:%M:%S"), "%Y")
data$Month <- format(as.POSIXct(data$Timestamp, format="%Y-%m-%d %H:%M:%S"), "%m")
data$Day <- format(as.POSIXct(data$Timestamp, format="%Y-%m-%d %H:%M:%S"), "%d")
data$Hour <- format(as.POSIXct(data$Timestamp, format="%Y-%m-%d %H:%M:%S"), "%H")
head(data)
```

A data.table: 6 × 14

Daily Time Spent on Site	Age	Area Income	Daily Internet Usage	Ad Topic Line	City	Male	Country	Timest
<dbl>	<int>	<dbl>	<dbl>	<chr>	<chr>	<int>	<chr>	<dt
68.95	35	61833.90	256.09	Cloned 5thgeneration orchestration	Wrightburgh	0	Tunisia	2016 00:5
80.23	31	68441.85	193.77	Monitored national standardization	West Jodi	1	Nauru	2016 01:3
69.47	26	59785.94	236.50	Organic bottom-line service-desk	Davidton	0	San Marino	2016 20:3
				Triple-buffered				2016

```
# drop the timestamp column since it is no longer useful
data$Timestamp <- NULL
colnames(data)
```

'Daily Time Spent on Site' · 'Age' · 'Area Income' · 'Daily Internet Usage' · 'Ad Topic Line' · 'City' · 'Male' · 'Country' · 'Clicked on Ad' · 'Year' · 'Month' · 'Day' · 'Hour'

```
#checking the data types of the new columns.
str(data)
```

```
Classes ‘data.table’ and 'data.frame':  1000 obs. of  13 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 natio
 $ 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" ...
 $ Clicked on Ad           : int  0 0 0 0 0 0 0 1 0 0 ...
 $ Year                    : chr  "2016" "2016" "2016" "2016" ...
 $ Month                   : chr  "03" "04" "03" "01" ...
 $ Day                     : chr  "27" "04" "13" "10" ...
 $ Hour                    : chr  "00" "01" "20" "02" ...
 - attr(*, ".internal.selfref")=<externalptr>
```

The year, day, month and hour column have the wrong data type and so we have to change the datatype to factor.

```
# Correcting the data types of the year, month, day, hour and male columns.
data$Year <- as.factor(data$Year)
data$Month <- as.factor(data$Month)
data$Day <- as.factor(data$Day)
data$Hour <- as.factor(data$Hour)
data$Male <- as.factor(data$Male)
```

```
#checking to see if the columns have been assigned the right data types.
str(data)
```

```
Classes ‘data.table’ and 'data.frame':  1000 obs. of  13 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 natio
 $ City                    : chr  "Wrightburgh" "West Jodi" "Davidton" "West Terrifurt"
 $ Male                    : Factor w/ 2 levels "0","1": 1 2 1 2 1 2 1 2 2 2 ...
 $ Country                 : chr  "Tunisia" "Nauru" "San Marino" "Italy" ...
 $ Clicked on Ad           : int  0 0 0 0 0 0 0 1 0 0 ...
 $ Year                    : Factor w/ 1 level "2016": 1 1 1 1 1 1 1 1 1 1 ...
 $ Month                   : Factor w/ 7 levels "01","02","03",..: 3 4 3 1 6 5 1 3 4 7
 $ Day                     : Factor w/ 31 levels "01","02","03",..: 27 4 13 10 3 19 28
 $ Hour                    : Factor w/ 24 levels "00","01","02",..: 1 2 21 3 4 15 21 2
 - attr(*, ".internal.selfref")=<externalptr>
```

```
#Removing white spaces from the column names
names(data)<-make.names(names(data),unique = TRUE)
```

```
#changing the data type of the column Clicked on Ad.
data$Clicked.on.Ad <- as.factor(data$Clicked.on.Ad)
```

```
#confirming if the column has been assigned the right data type.
str(data)
```

```
Classes ‘data.table’ and 'data.frame':  1000 obs. of  13 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 natio
 $ City                    : chr  "Wrightburgh" "West Jodi" "Davidton" "West Terrifurt"
 $ Male                    : Factor w/ 2 levels "0","1": 1 2 1 2 1 2 1 2 2 2 ...
 $ Country                 : chr  "Tunisia" "Nauru" "San Marino" "Italy" ...
 $ Clicked.on.Ad           : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 2 1 1 ...
 $ Year                    : Factor w/ 1 level "2016": 1 1 1 1 1 1 1 1 1 1 ...
```

```
 $ Month                   : Factor w/ 7 levels "01","02","03",..: 3 4 3 1 6 5 1 3 4 7
 $ Day                     : Factor w/ 31 levels "01","02","03",..: 27 4 13 10 3 19 28
 $ Hour                    : Factor w/ 24 levels "00","01","02",..: 1 2 21 3 4 15 21 2
 - attr(*, ".internal.selfref")=<externalptr>
```

```
#Previewing the dataset to see if the whitespaces have been removed
head(data)
```

A data.table: 6 ×

Daily.Time.Spent.on.Site	Age	Area.Income	Daily.Internet.Usage	Ad.Topic.Line
<dbl>	<int>	<dbl>	<dbl>	<chr>
68.95	35	61833.90	256.09	Cloned 5thgeneration orchestration
80.23	31	68441.85	193.77	Monitored national standardization
69.47	26	59785.94	236.50	Organic bottom-line service-desk
74.15	29	54806.18	245.89	Triple-buffered reciprocal time-frame
68.37	35	73889.99	225.58	Robust logistical utilization
59.99	23	59761.56	226.74	Sharable client-driven software

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

'Daily.Time.Spent.on.Site' · 'Age' · 'Area.Income' · 'Daily.Internet.Usage' · 'Ad.Topic.Line' · 'City' · 'Male' · 'Country' · 'Clicked.on.Ad' · 'Year' · 'Month' · 'Day' · 'Hour'

```
#Checking the class of Age column in the dataset.
class(data$Age)
```

'integer'

Cleaning the dataset

Data Completeness

```
#Checking if there is missing data in our dataset
is.na(data)
```

A matrix: 1000 × 13

Daily.Time.Spent.on.Site	Age	Area.Income	Daily.Internet.Usage	Ad.Topic.L
FALSE	FALSE	FALSE	FALSE	FAL
FALSE	FALSE	FALSE	FALSE	FAL
FALSE	FALSE	FALSE	FALSE	FAL

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

Daily.Time.Spent.on.Site: 0 Age: 0 Area.Income: 0 Daily.Internet.Usage: 0
Ad.Topic.Line: 0 City: 0 Male: 0 Country: 0 Clicked.on.Ad: 0 Year:
0 Month: 0 Day: 0 Hour: 0

There are no missing values in the dataset.

Data Consistency

```
#Checking for duplicated rows
duplicated_rows <- data[duplicated(data),]
duplicated_rows
```

A data.table: 0 × 13

Daily.Time.Spent.on.Site	Age	Area.Income	Daily.Internet.Usage	Ad.Topic.Line
<dbl>	<int>	<dbl>	<dbl>	<chr>

There are no duplicates in the dataset.

```
#Changing the Male column to Sex for easier understanding.
names(data)[names(data) == "Male"] <- "Sex"
head(data)
```

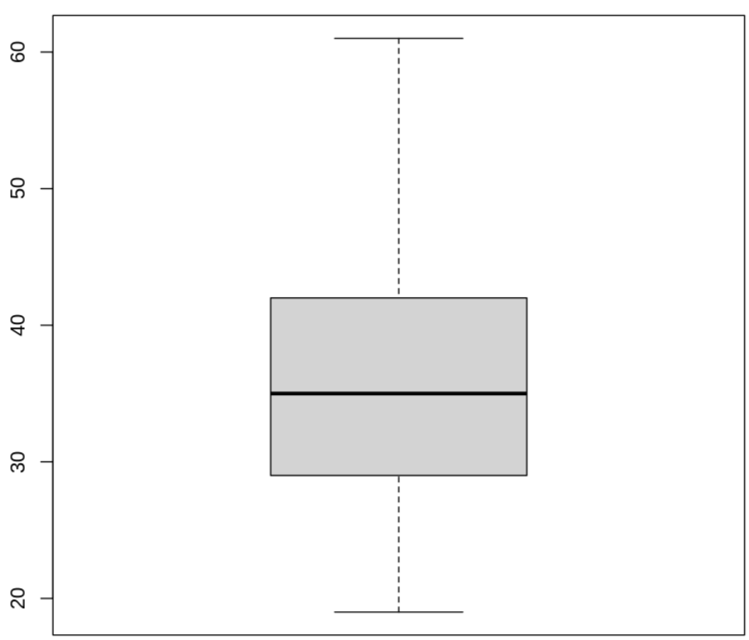
A data.table: 6 ×

Daily.Time.Spent.on.Site	Age	Area.Income	Daily.Internet.Usage	Ad.Topic.Line
<dbl>	<int>	<dbl>	<dbl>	<chr>
68.95	35	61833.90	256.09	Cloned 5thgeneration orchestration
80.23	31	68441.85	193.77	Monitored national standardization
69.47	26	59785.94	236.50	Organic bottom-

Data Validity

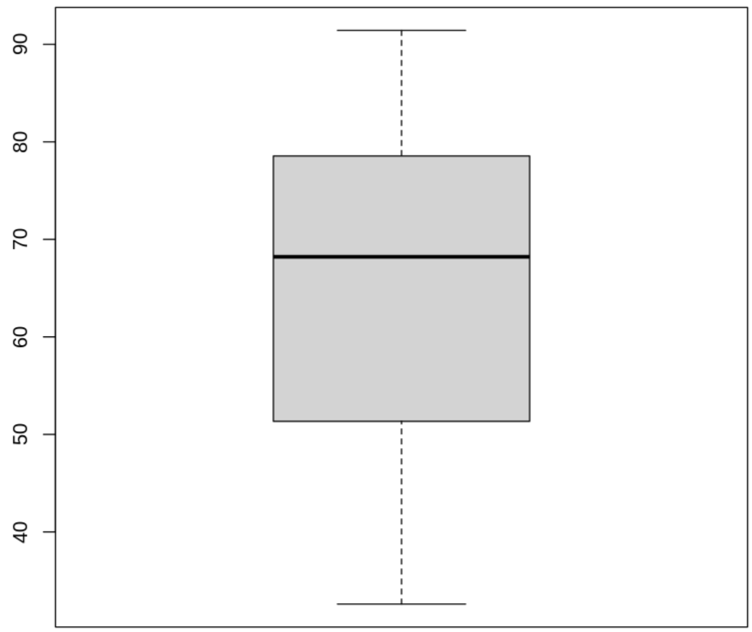
Triple-buffered

```
#Checking if there are outliers Age column.
#We are going to use boxplots
boxplot(data$Age)
```



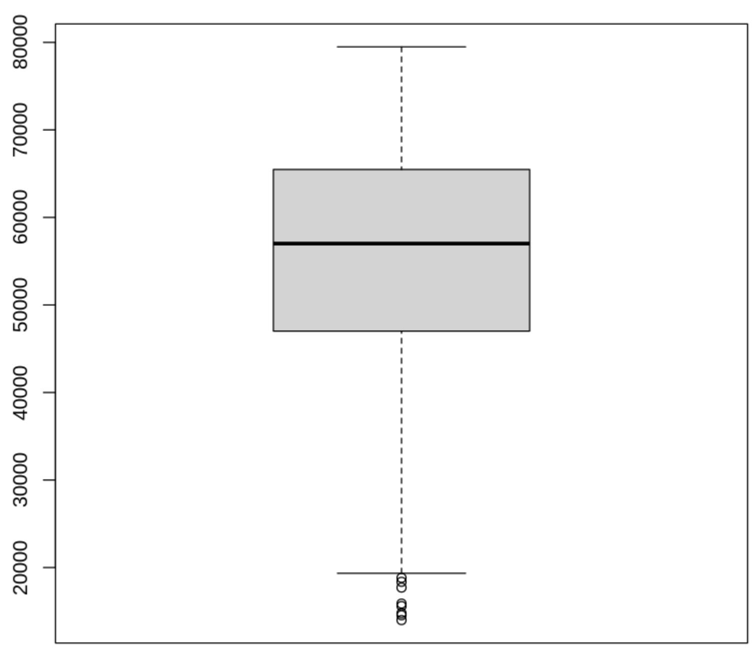
There are no outliers in the Age column.

```
#Checking for outliers in the daily time spent on site column.
boxplot(data$Daily.Time.Spent.on.Site)
```



There are no outliers in the Daily.Time.Spent.on.Site column.

```
#Checking for any outliers in the Area Income column.
boxplot(data$Area.Income)
```



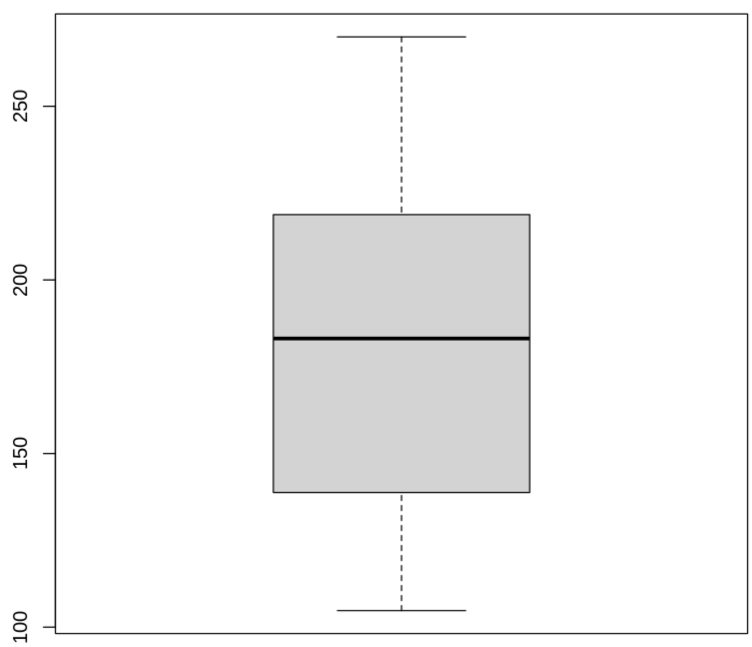
There are few outliers in the Area.Income column.

```
#Listing outliers in the Area.Income column
boxplot.stats(data$Area.Income)$out
```

17709.98 · 18819.34 · 15598.29 · 15879.1 · 14548.06 · 13996.5 · 14775.5 · 18368.57

Outliers are those with an income of less than 20,000. I choose not to drop the outliers since they will not affect our analysis.

```
#Checking for outliers in the daily internet column
boxplot(data$Daily.Internet.Usage)
```



There are no outliers in the daily internet usage column.

```
#Checking for anomalies in the Sex column
levels(data$Sex)
```

'0' · '1'

There are no anomalies in the sex column since we have only levels 0(representing female) and 1(representing male).

```
#Checking for anomalies in the clicked on ad column
levels(data$Clicked.on.Ad)
```

'0' · '1'

There are no anomalies in the clicked on ad column since we only have the two levels that is 0(representing those who did not click on the ad) and 1(representing those who clicked on the ad).

Univariate Analysis

Measures of central tendency

```
#Checking the mean, median and mode of the Daily time spent on site column
dt.mean <- mean(data$Daily.Time.Spent.on.Site)
dt.mean
dt.median <- median(data$Daily.Time.Spent.on.Site)
dt.median
getmode <- function(v) {
   uniqv <- unique(v)
   uniqv[which.max(tabulate(match(v, uniqv)))]
}
dt.mode <- getmode(data$Daily.Time.Spent.on.Site)
dt.mode
```

65.0002
68.215
62.26

Daily Time Spent on Site Measures of central tendency

- Mean - 65.002
- Median - 68.215
- Mode - 62.26

```
#Checking the mean, median and mode of the age column
age.mean <- mean(data$Age)
age.mean
age.median <- median(data$Age)
age.median
getmode <- function(v) {
   uniqv <- unique(v)
```

```
    uniqv[which.max(tabulate(match(v, uniqv)))]
}
age.mode <- getmode(data$Age)
age.mode
```

36.009
35
31

Age Measures of central tendency

- Mean - 36.009
- Median - 35
- Mode - 31

```
#Checking the mean, median and mode of the area income column
ai.mean <- mean(data$Area.Income)
ai.mean
ai.median <- median(data$Area.Income)
ai.median
getmode <- function(v) {
    uniqv <- unique(v)
    uniqv[which.max(tabulate(match(v, uniqv)))]
}
ai.mode <- getmode(data$Area.Income)
ai.mode
```

55000.00008
57012.3
61833.9

Area income Measures of central tendency

- Mean - 55000.00008
- Median - 57012.3
- Mode -61833.9

```
#Checking the mean, median and mode of the daily internet usage column
diu.mean <- mean(data$Daily.Internet.Usage)
diu.mean
diu.median <- median(data$Daily.Internet.Usage)
diu.median
getmode <- function(v) {
    uniqv <- unique(v)
    uniqv[which.max(tabulate(match(v, uniqv)))]
}
```

```
diu.mode <- getmode(data$Daily.Internet.Usage)
diu.mode
```

180.0001
183.13

Daily Internet Usage Measures of central tendency

- Mean - 180.0001
- Median - 183.13
- Mode -167.22

```
#Checking the mode of the country column
getmode <- function(v) {
   uniqv <- unique(v)
   uniqv[which.max(tabulate(match(v, uniqv)))]
}
country.mode <- getmode(data$Country)
country.mode
```

'Czech Republic'

Czech Republic is the most frequent country.

```
#Checking the mode of the city column
getmode <- function(v) {
   uniqv <- unique(v)
   uniqv[which.max(tabulate(match(v, uniqv)))]
}
city.mode <- getmode(data$City)
city.mode
```

'Lisamouth'

Lisamouth is the most frequent city.

```
#Checking the mode of the sex column
getmode <- function(v) {
   uniqv <- unique(v)
   uniqv[which.max(tabulate(match(v, uniqv)))]
}
sex.mode <- getmode(data$Sex)
sex.mode
```

0

▶ **Levels**:

Most people from the data collected are female.

```
#Checking the mode of the Daily.Internet.Usage column
getmode <- function(v) {
   uniqv <- unique(v)
   uniqv[which.max(tabulate(match(v, uniqv)))]
}
diu.mode <- getmode(data$Daily.Internet.Usage)
diu.mode
```

167.22

Most people had a daily internet usage of 167.22

```
#Checking the mode of the Year column
getmode <- function(v) {
   uniqv <- unique(v)
   uniqv[which.max(tabulate(match(v, uniqv)))]
}
yr.mode <- getmode(data$Year)
yr.mode
```

2016
▶ **Levels**:

2016 is the only year represented in the dataset.

```
#Checking the mode of the Year column
getmode <- function(v) {
   uniqv <- unique(v)
   uniqv[which.max(tabulate(match(v, uniqv)))]
}
mth.mode <- getmode(data$Month)
mth.mode
```

02
▶ **Levels**:

February is the month that appears multiple times.

```
#Checking the mode of the Year column
getmode <- function(v) {
   uniqv <- unique(v)
   uniqv[which.max(tabulate(match(v, uniqv)))]
}
```

```
dy.mode <- getmode(data$Day)
dy.mode
```

03
▶ **Levels**:

Day 3 of the month appears the most.

```
#Checking the mode of the Year column
getmode <- function(v) {
   uniqv <- unique(v)
   uniqv[which.max(tabulate(match(v, uniqv)))]
}
hr.mode <- getmode(data$Hour)
hr.mode
```

07
▶ **Levels**:

7:00am appears the most in our dataset.

Measures of dispersion

```
#Checking the min, max, range, quantile, variance, standard deviation  of Daily time spent or
dt.min <- min(data$Daily.Time.Spent.on.Site)
dt.min
dt.max <- max(data$Daily.Time.Spent.on.Site)
dt.max
dt.range <- range(data$Daily.Time.Spent.on.Site)
dt.range
dt.quantile <- quantile(data$Daily.Time.Spent.on.Site)
dt.quantile
dt.var <- var(data$Daily.Time.Spent.on.Site)
dt.var
dt.sd <- sd(data$Daily.Time.Spent.on.Site)
dt.sd
```

32.6
91.43
32.6 · 91.43
0%: 32.6 25%: 51.36 50%: 68.215 75%: 78.5475 100%: 91.43
251.337094854855
15.8536145675002

```
#Checking for the skewness of daily time spent on site.
install.packages('moments')
library(moments)
skewness(data$Daily.Time.Spent.on.Site)
```

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

-0.371202614867441

Since the skewness is negative, it means we have a left skewed distribution

```
#Checking for Kurtosis
kurtosis(data$Daily.Time.Spent.on.Site)
```

1.90394215401081

Has a leptokurtic distribution since the kurtosis is > 0.

Daily Time Spent on Site Measures of dispersion

- Min - 32.6
- Max - 91.43
- Range - 32.691.43
- Quantile - 0% - 32.6 25% - 51.36 50% - 68.215 75% - 78.5475 100% - 91.43
- Variance - 251.337094854855
- Standard deviation - 15.8536145675002

```
#Checking the min, max, range, quantile, variance, standard deviation of the age column
age.min <- min(data$Age)
age.min
age.max <- max(data$Age)
age.max
age.range <- range(data$Age)
age.range
age.quantile <- quantile(data$Age)
age.quantile
age.var <- var(data$Age)
age.var
age.sd <- sd(data$Age)
age.sd
```

19
61
19 · 61
0%: 19 25%: 29 50%: 35 75%: 42 100%: 61
77.1861051051051

```
#checking for skewness of the age column
skewness(data$Age)
```

0.478422676206608

Since the skewness is positive, it means we have a right skewed distribution

```
#Checking for Kurtosis
kurtosis(data$Age)
```

2.59548176807726

Has a leptokurtic distribution since the kurtosis is > 0.

Age Measures of dispersion

- Min - 19
- Max - 61
- Range - 19-61
- Quantile - 0%: 19 25%: 29 50%: 35 75%: 42 100%: 61
- Variance - 77.1861051051051
- Standard deviation - 18.78556231012592

```
#Checking the min, max, range, quantile, variance, standard deviation of the area income colu
ai.min <- min(data$Area.Income)
ai.min
ai.max <- max(data$Area.Income)
ai.max
ai.range <- range(data$Area.Income)
ai.range
ai.quantile <- quantile(data$Area.Income)
ai.quantile
ai.var <- var(data$Area.Income)
ai.var
ai.sd <- sd(data$Area.Income)
ai.sd
```

13996.5
79484.8

```
#Checking for the skewness of the Area income column
skewness(data$Area.Income)
```

-0.649396701694076

Since the skewness is negative, it means we have a left skewed distribution.

```
#Checking for Kurtosis
kurtosis(data$Area.Income)
```

2.89469406161926

Has a leptokurtic distribution since the kurtosis is > 0.

Area Income Measures of dispersion

- Min - 13996.5
- Max - 79484.8
- Range - 13996.579484.8
- Quantile - 0% - 13996.5 25% - 47031.8025 50% - 57012.3 75% - 65470.635 100% - 79484.8
- Variance - 179952405.951775
- Standard deviation - 13414.6340222824

```
#Checking the min, max, range, quantile, variance, standard deviation of the daily internet u
diu.min <- min(data$Daily.Internet.Usage)
diu.min
diu.max <- max(data$Daily.Internet.Usage)
diu.max
diu.range <- range(data$Daily.Internet.Usage)
diu.range
diu.quantile <- quantile(data$Daily.Internet.Usage)
diu.quantile
diu.var <- var(data$Daily.Internet.Usage)
diu.var
diu.sd <- sd(data$Daily.Internet.Usage)
diu.sd
```

104.78
269.96

```
#Checking for the skewness of the daily internet usage column.
skewness(data$Daily.Internet.Usage)
```

-0.0334870316434409

Since the skewness is negative, it means we have a left skewed distribution.

```
#Checking for Kurtosis
kurtosis(data$Daily.Internet.Usage)
```

1.72770118094819

Has a leptokurtic distribution since the kurtosis is > 0.

Daily Internet Used Measures of dispersion

- Min - 104.78
- Max - 269.96
- Range - 104.78269.96
- Quantile - 0% - 104.78 25% - 138.83 50% - 183.13 75% - 218.7925 100% - 269.96
- Variance - 1927.41539618619
- Standard deviation - 43.9023393019801

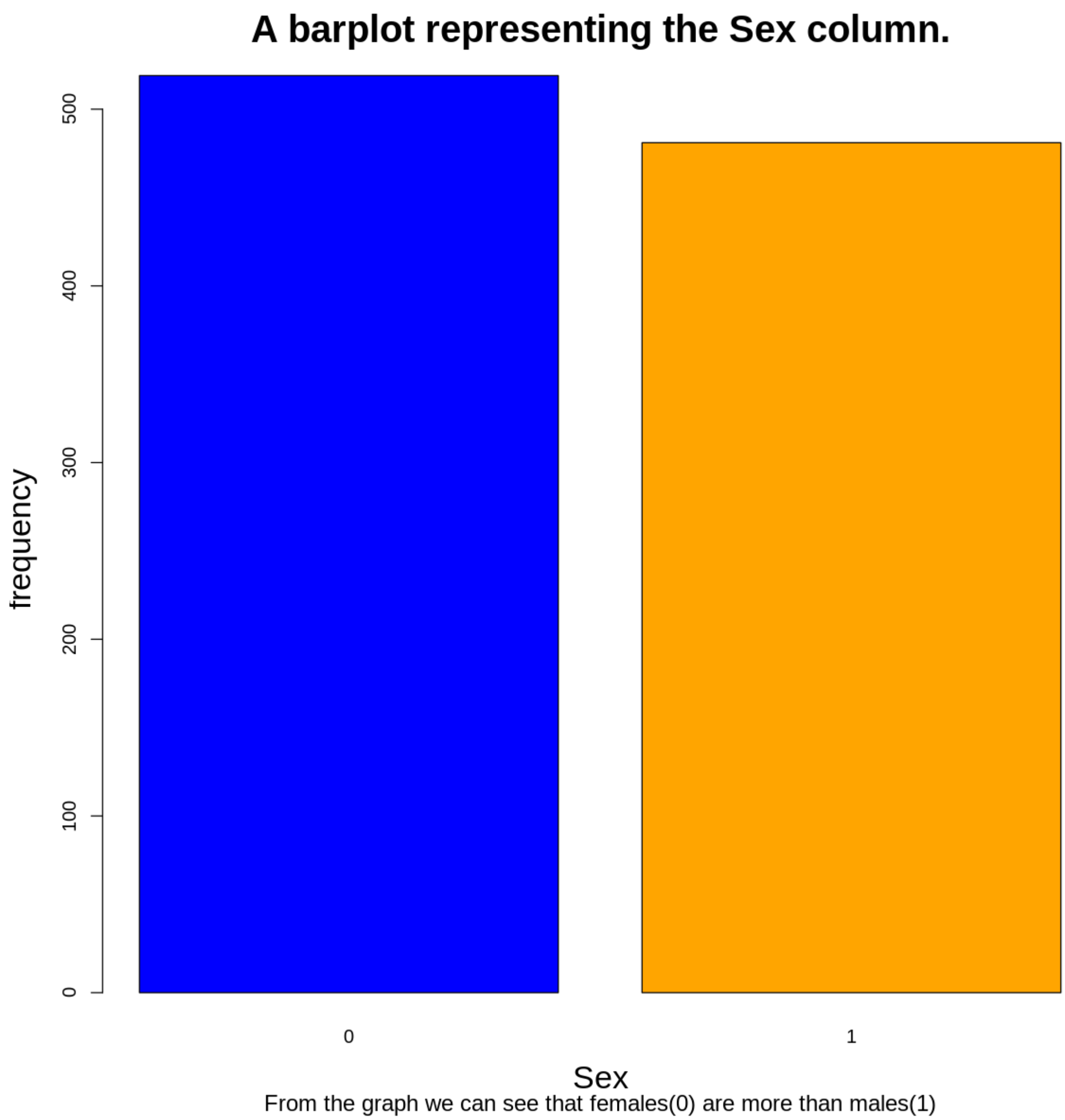
Univariate graphs

```
# Fetching the Sex column
# Computing the frequency of both male and female respondents.

sex <- data$Sex
sex_frequency <- table(sex)
sex_frequency
```

```
sex
  0   1
519 481
```

```
#Bar graph representing Sex frequency.
options(repr.plot.width = 10, repr.plot.height = 10)
barplot(c(sex_frequency), main="A barplot representing the Sex column.",
        xlab="Sex",
```

```
        ylab="frequency",
        sub="From the graph we can see that females(0) are more than males(1)",
        cex.main=2, cex.lab=1.7,cex.sub=1.2,
        width=c(30,30),
        col=c("blue","orange"))
```



We can observe from the frequency table and from the graph that most respondents are female.

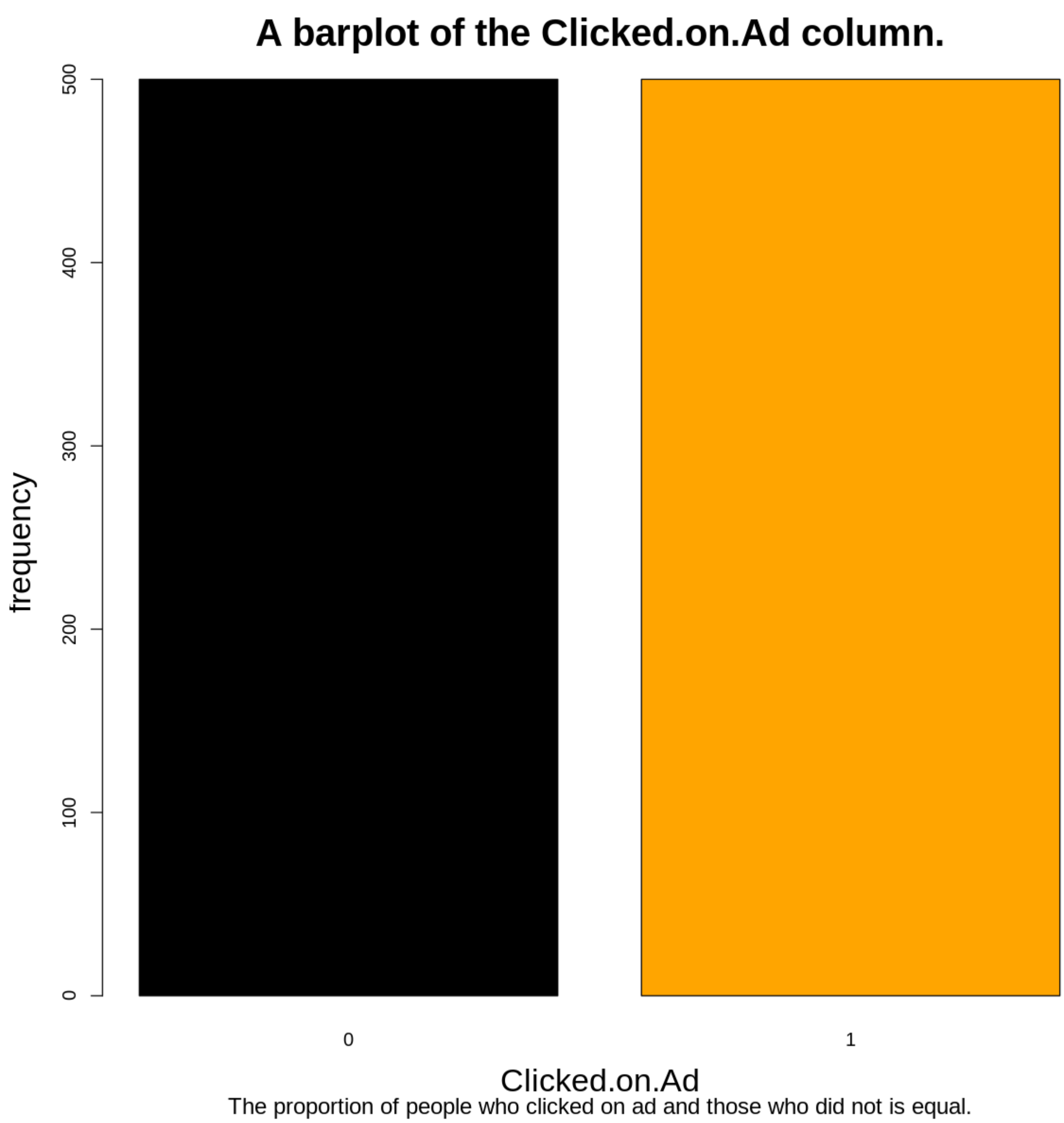
- 519 - Female
- 481 - Male

```
# Fetching the Clicked on ad column
# Computing the frequency of respondents who clicked on the ad and those who did not..
Clicked.on.Ad <- data$Clicked.on.Ad
```

```
Clicked.on.Ad_frequency <- table(Clicked.on.Ad)
Clicked.on.Ad_frequency
```

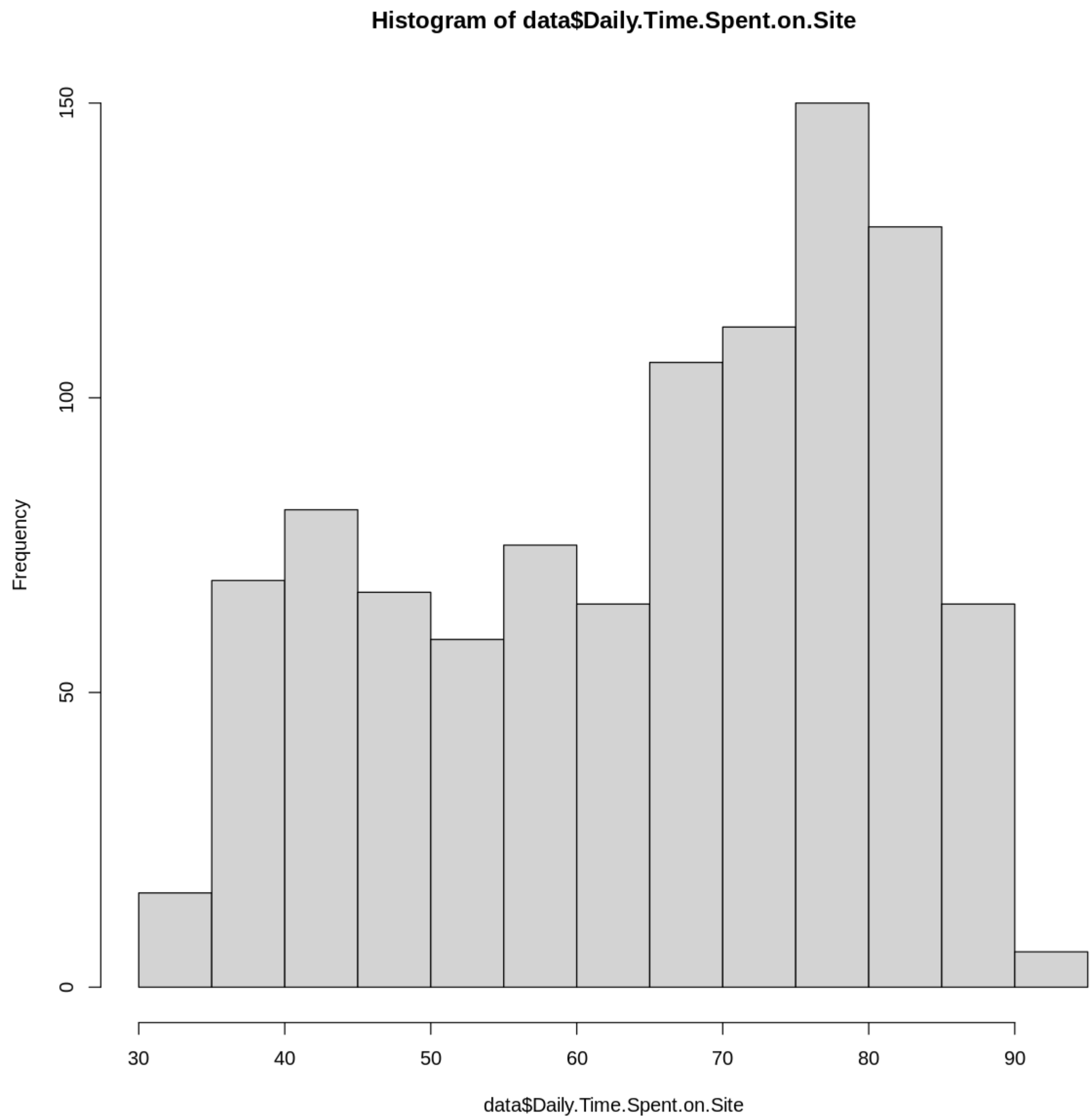
```
     Clicked.on.Ad
       0   1
```

```
#Bar graph representing clicked on ad frequency
options(repr.plot.width = 10, repr.plot.height = 10)
barplot(c(Clicked.on.Ad_frequency), main="A barplot of the Clicked.on.Ad column.",
        xlab="Clicked.on.Ad",
        ylab="frequency",
        sub="The proportion of people who clicked on ad and those who did not is equal.",
        cex.main=2, cex.lab=1.7,cex.sub=1.2,
        col=c("black","orange"))
```



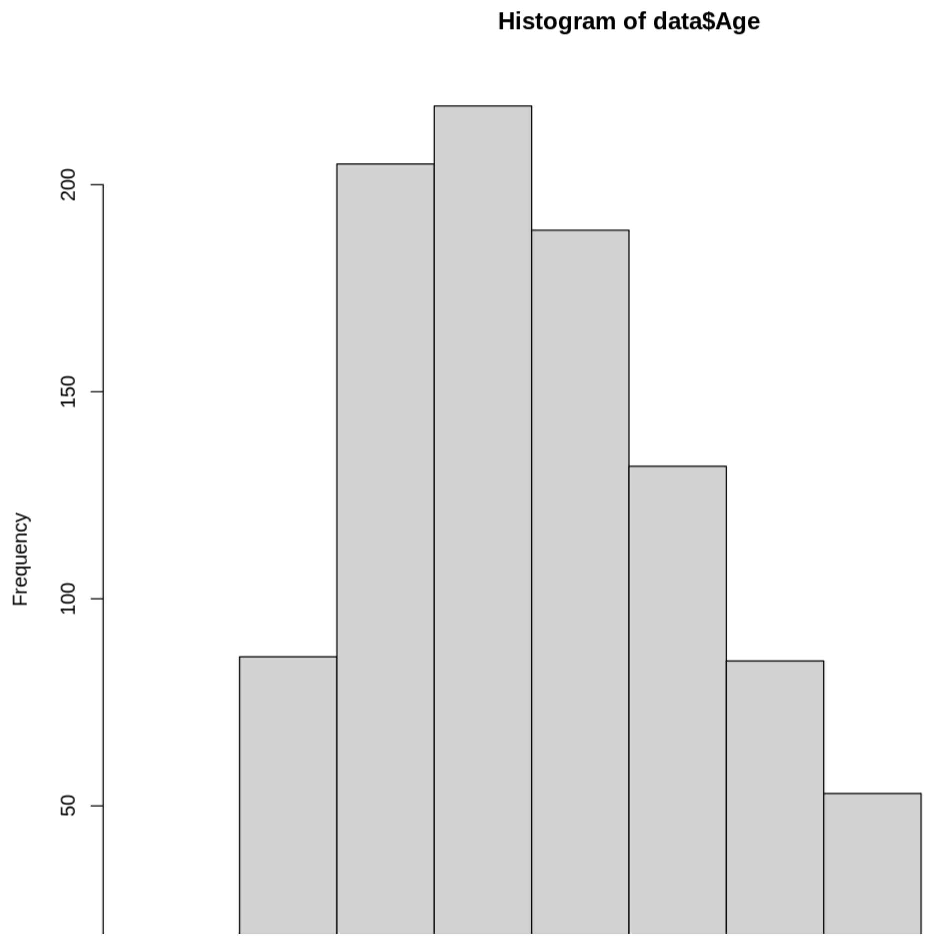
From the frequency table and the bar graph, we observe that there is a balance of those who clicked on the ad and those who did not.

```
#A histogram of the daily time spent on site.
options(repr.plot.width = 10, repr.plot.height = 10)
hist(data$Daily.Time.Spent.on.Site)
```



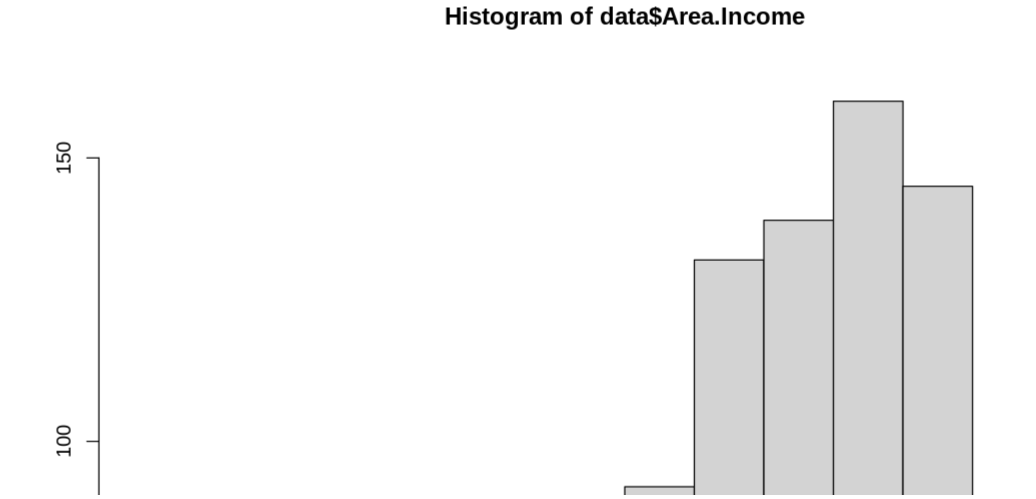
The histogram appears to be relatively uniform.

```
#A histogram of age
options(repr.plot.width = 10, repr.plot.height = 10)
hist(data$Age)
```



The histogram is skewed to the right.

```
#A histogram of Area Income
options(repr.plot.width = 10, repr.plot.height = 10)
hist(data$Area.Income)
```



The above histogram is skewed to the left.

```
#Ahistogram of daily internet usage
options(repr.plot.width = 10, repr.plot.height = 10)
hist(data$Daily.Internet.Usage)
```

Histogram of data$Daily.Internet.Usage

The histogram appears to be relatively normal.

```
#Checking the most frequent countries.
sort(table(data$Country), decreasing=TRUE)[1:10]
```

```
Czech Republic         France    Afghanistan      Australia         Cyprus 
             9              9              8              8              8 
        Greece        Liberia     Micronesia           Peru        Senegal 
             8              8              8              8              8 
```

Czeech Republic, France and Afghanistan are among the first 10 countries with the highest frequency.

```
#Checking the most frequent cities.
sort(table(data$City), decreasing=TRUE)[1:10]
```

```
      Lisamouth     Williamsport Benjaminchester       East John    East Timothy 
              3               3               2               2               2 
       Johnstad        Joneston      Lake David      Lake James       Lake Jose 
              2               2               2               2               2 
```

Lisamouth, Williamsport Benjaminchester, East John and East Timothy are among the first 10 cities with the highest frequency.

```
#Checking the most frequent ad topic line
sort(table(data$Ad.Topic.Line), decreasing=TRUE)[1:10]
```

```
      Adaptive 24hour Graphic Interface         Adaptive asynchronous attitude 
                                      1                                      1 
 Adaptive context-sensitive application Adaptive contextually-based methodology 
                                      1                                      1 
    Adaptive demand-driven knowledgebase            Adaptive uniform capability 
                                      1                                      1 
             Advanced 24/7 productivity      Advanced 5thgeneration capability 
                                      1                                      1 
        Advanced didactic conglomeration Advanced disintermediate data-warehouse 
                                      1                                      1 
```

All entries in this column occurred once.

```
#Checking the most frequent hour.
```

```
sort(table(data$Hour), decreasing=TRUE)[1:10]
```

```
07 20 09 21 00 05 23 08 14 22
54 50 49 48 45 44 44 43 43 43
```

7:00am and 9:00pm are the most frequent hours.

```
#Checking the most frequent month.
sort(table(data$Month), decreasing=TRUE)[1:7]
```

```
 02  03  01  04  05  06  07
160 156 147 147 147 142 101
```

February, March and January are the most frequent months.

```
#Checking the most frequent day.
sort(table(data$Day), decreasing=TRUE)[1:10]
```

```
03 17 15 10 04 26 05 08 16 18
46 42 41 37 36 36 35 35 35 35
```

Day 3, 17 and 15 are the most frequent days.

Bivariate Analysis.

Covariance

```
#Covariance of daily time spent on site and age.
Daily.Time.Spent.on.Site <- data$Daily.Time.Spent.on.Site
Age <- data$Age
cov(Daily.Time.Spent.on.Site, Age)
```

-46.1741459459459

The covariance is negative showing that greater values of one variable corresponds to smaller values of the other.

```
#Covariance of daily time spent on site and Area.Income.
Daily.Time.Spent.on.Site <- data$Daily.Time.Spent.on.Site
```

```
Area.Income <- data$Area.Income
cov(Daily.Time.Spent.on.Site, Area.Income)
```

66130.8109081922

The covariance is positive showing that greater values of one variable correspond to greater values of the other.

```
#Covariance of daily time spent on site and Daily.Internet.Usage.
Daily.Internet.Usage <- data$Daily.Internet.Usage
Daily.Time.Spent.on.Site <- data$Daily.Time.Spent.on.Site
cov(Daily.Time.Spent.on.Site, Daily.Internet.Usage)
```

360.991882662663

The covariance is positive showing that greater values of one variable correspond to greater values of the other.

```
#Covariance of Age and Area.Income.
Age<- data$Age
Area.Income <- data$Area.Income
cov(Age, Area.Income)
```

-21520.9257965165

The covariance is negative showing that greater values of one variable corresponds to smaller values of the other.

```
#covariance of age and daily internet usage.
Age<- data$Age
Daily.Internet.Usage <- data$Daily.Internet.Usage
cov(Age, Daily.Internet.Usage <- data$Daily.Internet.Usage)
```

-141.634815715716

The covariance is negative showing that greater values of one variable corresponds to smaller values of the other.

```
#covariance of area income and daily internet usage.
Area.Income <- data$Area.Income
Daily.Internet.Usage <- data$Daily.Internet.Usage
cov(Area.Income, Daily.Internet.Usage <- data$Daily.Internet.Usage)
```

198762.531532925

The covariance is positive showing that greater values of one variable correspond to greater values of the other.

Correlation

```
#Correlation of daily time spent on site and age.
Daily.Time.Spent.on.Site <- data$Daily.Time.Spent.on.Site
Age <- data$Age
cor(Daily.Time.Spent.on.Site, Age)
```

-0.331513342786584

The two variables have a weak negative correlation.

```
#Correlation of daily time spent on site and Area.Income.
Daily.Time.Spent.on.Site <- data$Daily.Time.Spent.on.Site
Area.Income <- data$Area.Income
cor(Daily.Time.Spent.on.Site, Area.Income)
```

0.310954412522883

The two variables have a weak positive correlation.

```
#Correlation of daily time spent on site and Daily.Internet.Usage.
Daily.Internet.Usage <- data$Daily.Internet.Usage
Daily.Time.Spent.on.Site <- data$Daily.Time.Spent.on.Site
cor(Daily.Time.Spent.on.Site, Daily.Internet.Usage)
```

0.518658475337186

The two variables have a positive correlation.

```
#Correlation of Age and Area.Income.
Age<- data$Age
Area.Income <- data$Area.Income
cor(Age, Area.Income)
```

-0.182604955032622

The two variables have a weak negative correlation.

```
#Correlation of age and daily internet usage.
```

```
Age<- data$Age
Daily.Internet.Usage <- data$Daily.Internet.Usage
cor(Age, Daily.Internet.Usage <- data$Daily.Internet.Usage)
```

-0.367208560147359

The two variables have a weak negative correlation.

```
#Correlation of area income and daily internet usage.
Area.Income <- data$Area.Income
Daily.Internet.Usage <- data$Daily.Internet.Usage
cor(Area.Income, Daily.Internet.Usage <- data$Daily.Internet.Usage)
```

0.337495532865276

The two variables have a weak positive correlation.

Selecting data that consists of people who clicked on ad.

Univariate Analysis of the people who clicked on the ad

```
#Selecting the data with click on ad as 1
clicked <- data[data$Clicked.on.Ad ==1,]
head(clicked)
```

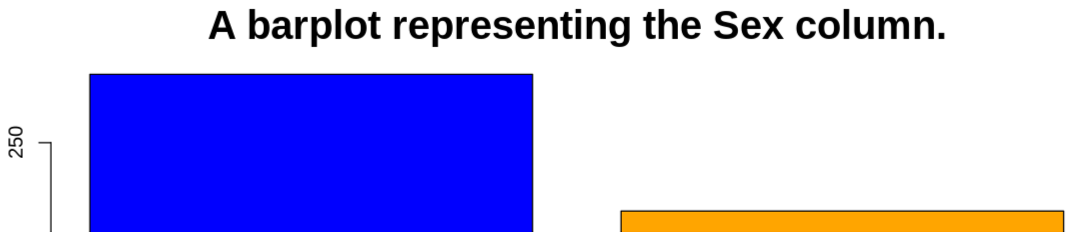
A data.table: 6 ›

Daily.Time.Spent.on.Site	Age	Area.Income	Daily.Internet.Usage	Ad.Topic.Line
<dbl>	<int>	<dbl>	<dbl>	<chr>
66.00	48	24593.33	131.76	Reactive local challenge

```
# Fetching the Sex column
sex <- clicked$Sex
sex_frequency <- table(sex)
sex_frequency
```

```
sex
  0   1
269 231
```

```
options(repr.plot.width = 10, repr.plot.height = 10)
barplot(c(sex_frequency), main="A barplot representing the Sex column.",
        xlab="Sex",
        ylab="frequency",
        sub="From the graph we can see that females(0) are more than males(1)",
        cex.main=2, cex.lab=1.7,cex.sub=1.2,
        width=c(30,30),
        col=c("blue","orange"))
```



From those who clicked on the ad, 269 were female and 231 were male.

```
#Checking the most frequent Daily.Time.Spent.on.Site.
sort(table(clicked$Daily.Time.Spent.on.Site), decreasing=TRUE)[1:10]
```

```
75.55  32.6 35.49 35.66 35.98 38.35 39.86 39.96 41.49 41.73
    3     2     2     2     2     2     2     2     2     2
```

```
#Checking the most frequent Age.
sort(table(clicked$Age), decreasing=TRUE)[1:10]
```

```
45 36 38 41 42 40 43 50 39 49
27 25 25 22 20 19 19 19 17 17
```

We can see the most frequent age of respondents who clicked on the ad as 40's, 30's and 50's.

```
#Checking the most frequent Daily.Internet.Usage.
sort(table(clicked$Daily.Internet.Usage), decreasing=TRUE)[1:10]
```

```
113.53 115.91  117.3  119.3 120.06 125.45 132.38 135.24 136.18 138.35
     2      2      2      2      2      2      2      2      2      2
```

```
#Checking the most frequent cities.
sort(table(clicked$City), decreasing=TRUE)[1:10]
```

```
  Lake David    Lake James    Lisamouth Michelleside    Millerbury    Robertfurt
           2             2            2            2             2             2
  South Lisa  West Amanda West Shannon Williamsport
           2             2            2            2
```

the cities that are more frequent are Lake David, Lake James and so on.

```
#Checking the most frequent Country.
sort(table(clicked$Country), decreasing=TRUE)[1:10]
```

```
     Australia      Ethiopia        Turkey       Liberia Liechtenstein
             7             7             7             6             6
  South Africa   Afghanistan        France       Hungary      Mayotte
             6             5             5             5             5
```

The most frequent country is Australia.

```
#Checking the most frequent Month.
sort(table(clicked$Month), decreasing=TRUE)[1:10]
```

```
  02   05   03   04   06   01   07 <NA> <NA> <NA>
  83   79   74   74   71   69   50
```

The most frequent month is February.

```
#Checking the most frequent Day.
sort(table(clicked$Day), decreasing=TRUE)[1:10]
```

```
03 23 14 09 12 15 01 10 05 17
26 22 21 20 20 20 19 19 18 18
```

The most frequent day of the month is day 3.

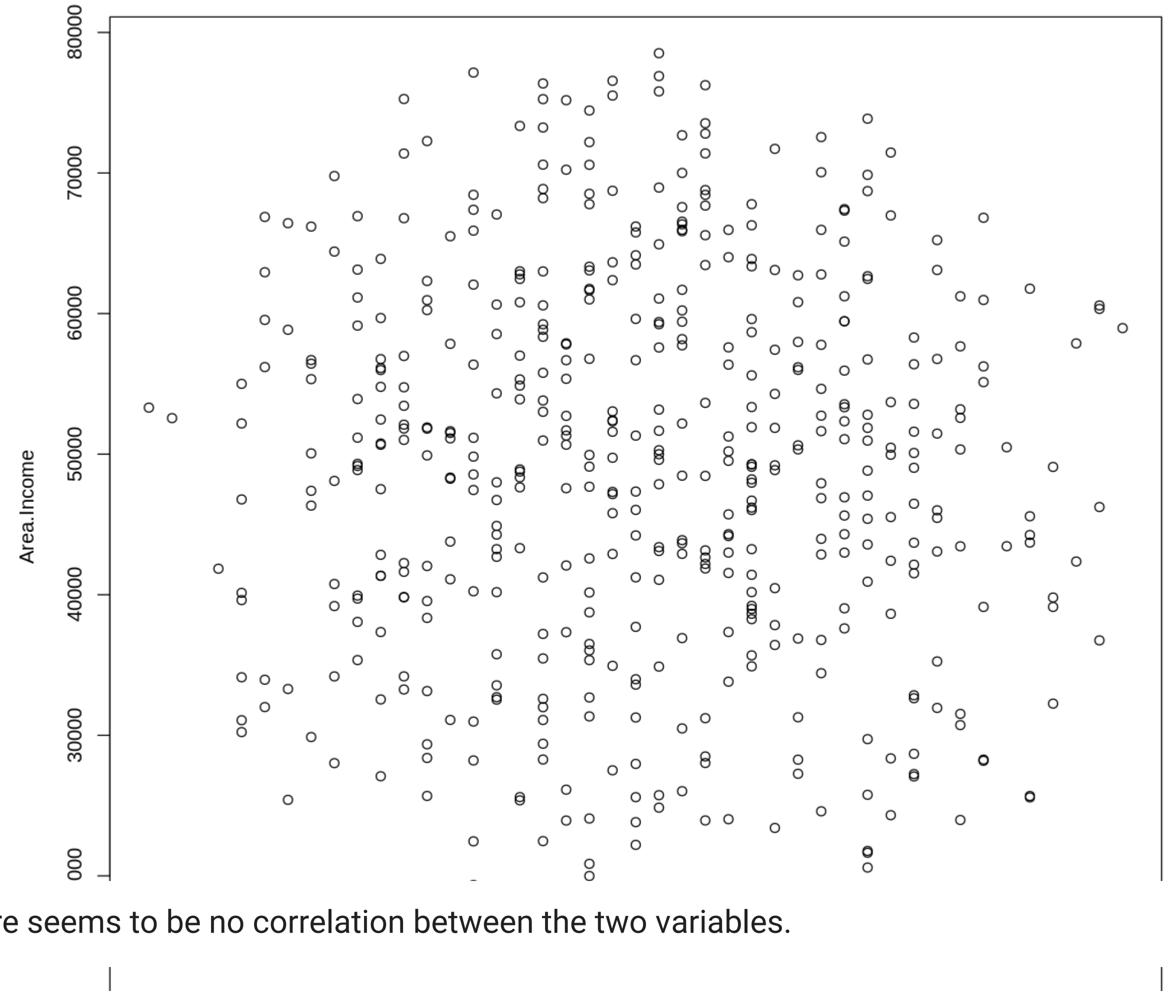
```
#Checking the most frequent Hour.
sort(table(clicked$Hour), decreasing=TRUE)[1:10]
```

```
09 00 07 18 11 20 03 06 17 04
28 26 26 25 24 24 23 23 23 21
```

The most frequent hour is 9 am.

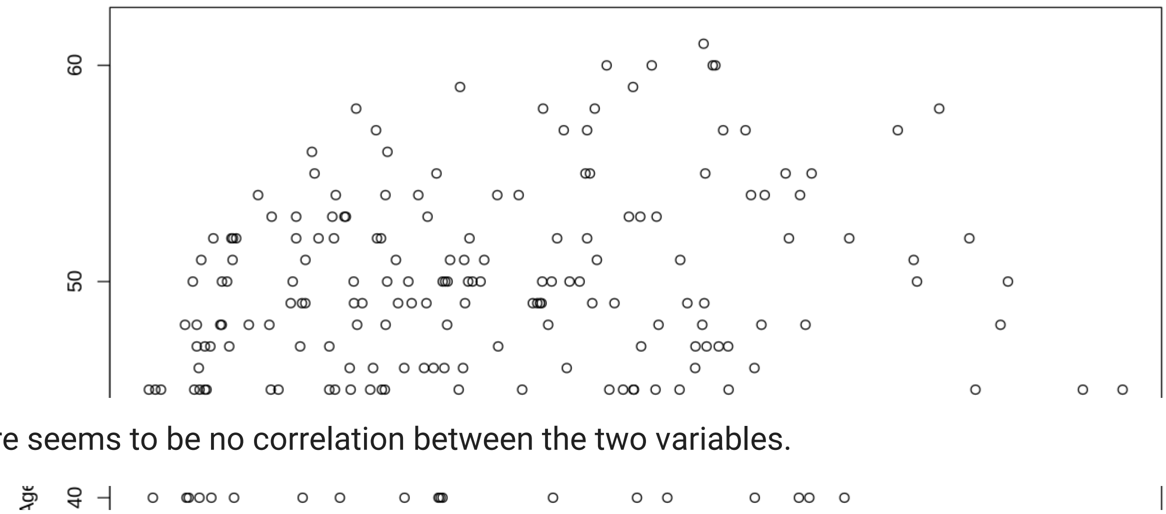
Scatter Plots

```
#A scatter plot of Age vs Area Income.
plot(clicked$Age, clicked$Area.Income, xlab="Age", ylab="Area.Income")
```



There seems to be no correlation between the two variables.

```
#A scatter plot of Age vs Daily Time Spent on Site.
plot(clicked$Daily.Time.Spent.on.Site, clicked$Age, xlab="Daily.Time.Spent.on.Site", ylab="Ag
```

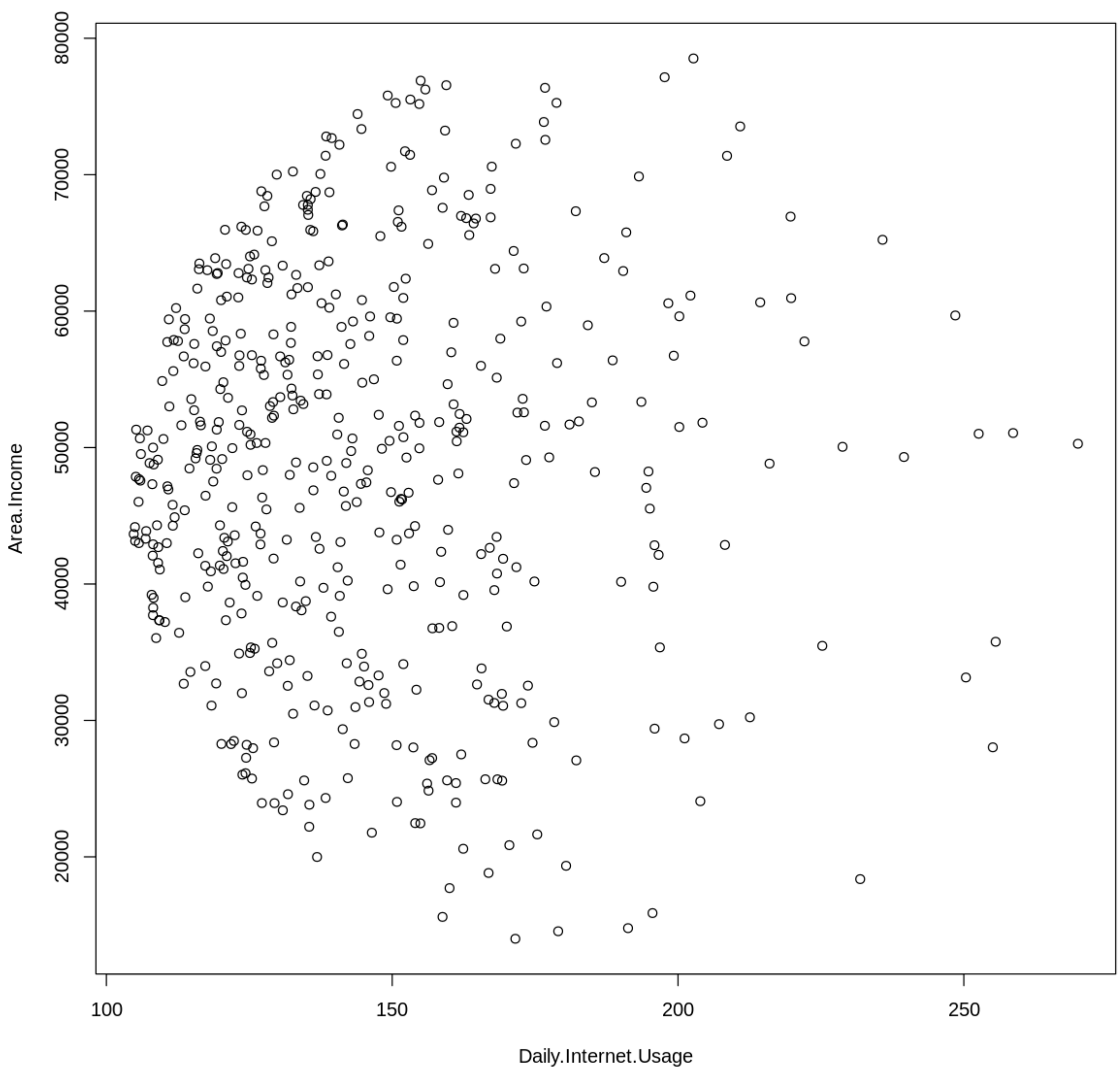


There seems to be no correlation between the two variables.

```
#A scatter plot of Age vs Daily.Internet.Usage.
plot(clicked$Daily.Internet.Usage, clicked$Age, xlab="EDaily.Internet.Usage", ylab="Age")
```

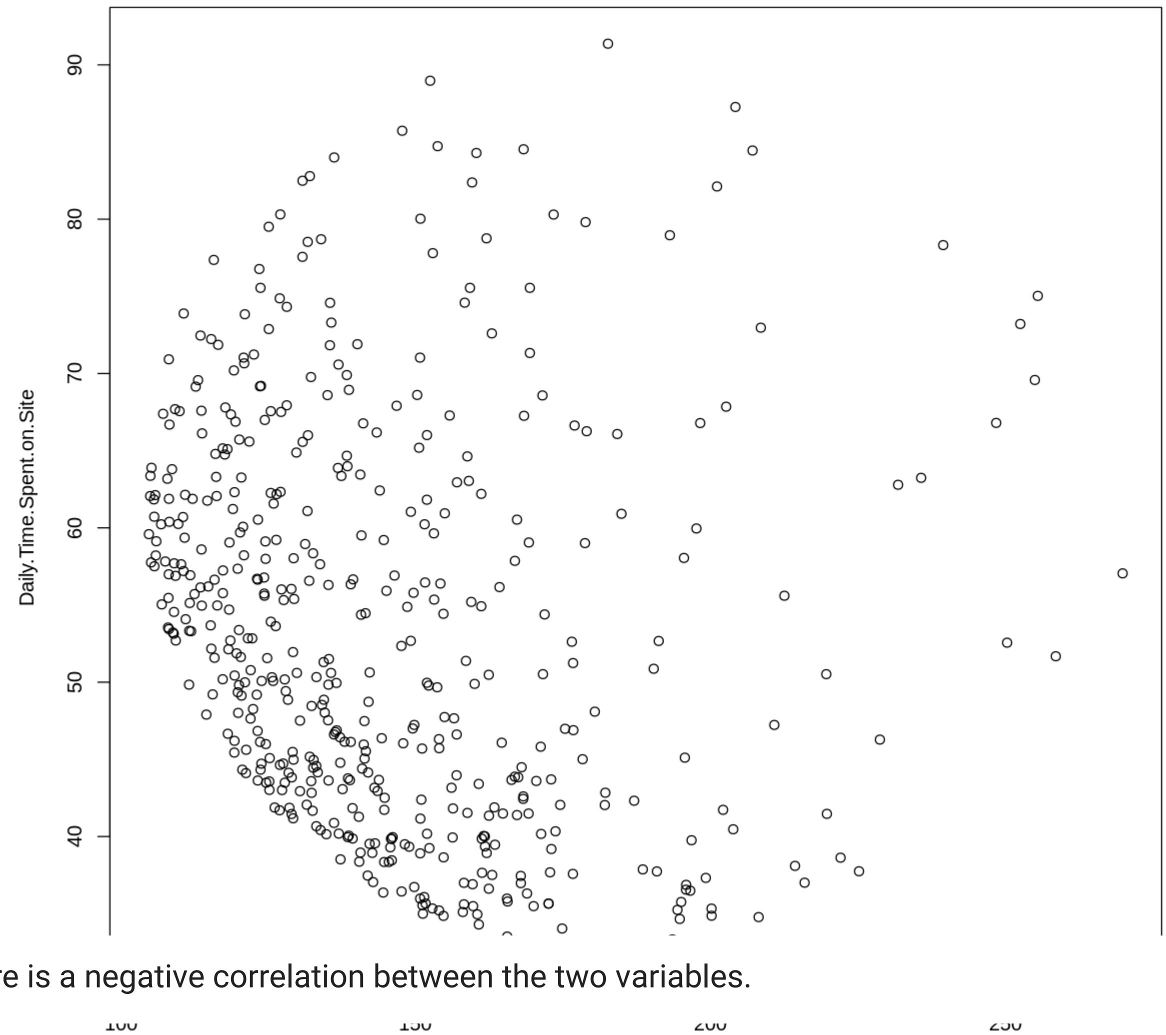
There seems to be no correlation between the two variables.

```
#A scatter plot of Area.Income vs Daily.Internet.Usage.
plot(clicked$Daily.Internet.Usage, clicked$Area.Income, xlab="Daily.Internet.Usage", ylab="Ar
```



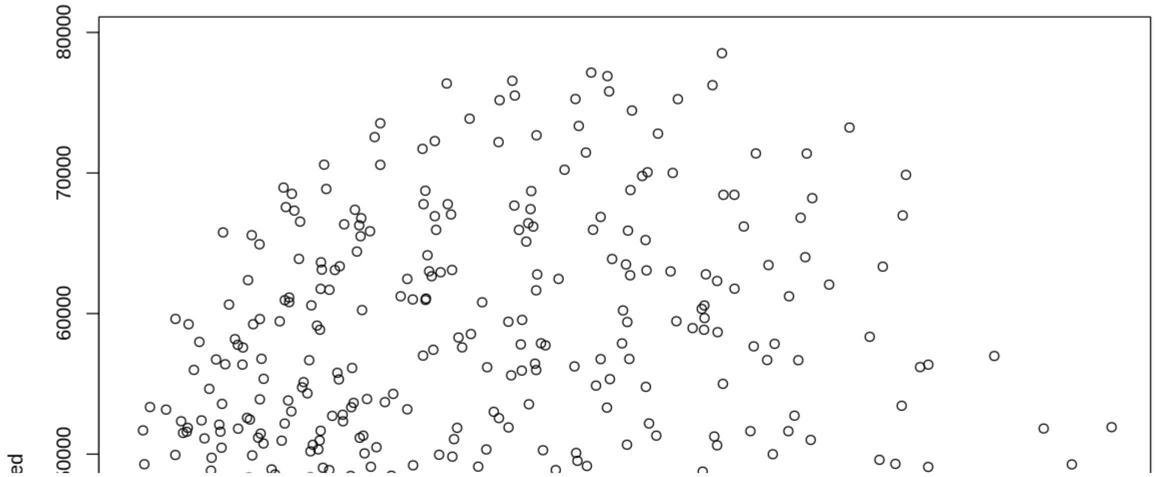
There seems to be no correlation between the two variables.

```
#A scatter plot of Daily.Internet.Usage vs Daily Time Spent on Site.
plot(clicked$Daily.Internet.Usage, clicked$Daily.Time.Spent.on.Site, xlab="Daily.Internet.Usa
```



There is a negative correlation between the two variables.

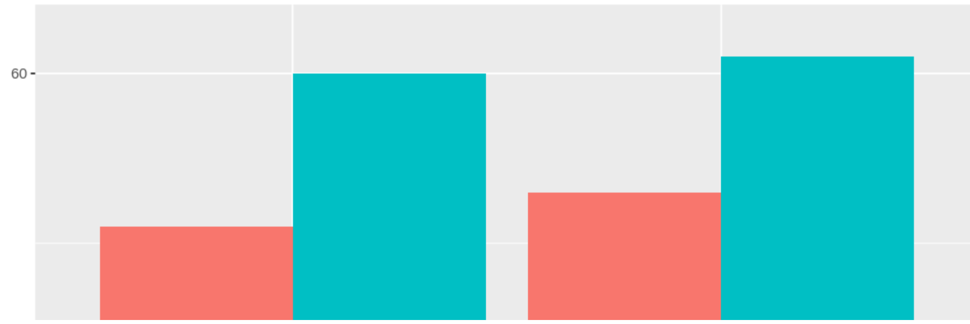
```
#A scatter plot of Area Income vs Daily Time Spent on Site.
plot(clicked$Daily.Time.Spent.on.Site, clicked$Area.Income, xlab="Area.Income", ylab="Time wa
```



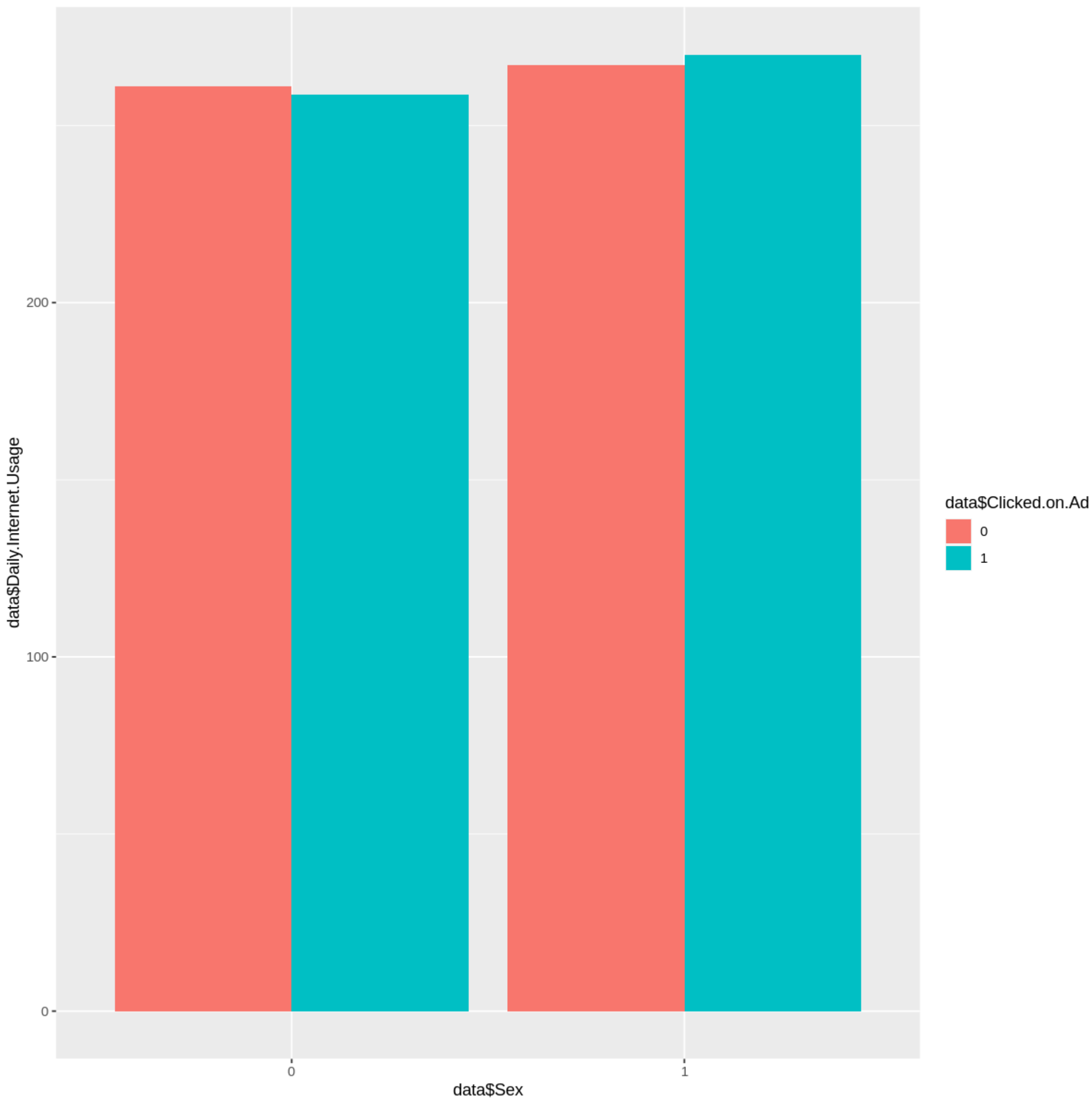
There seems to be no correlation between the two variables.

Multivariate Analysis.

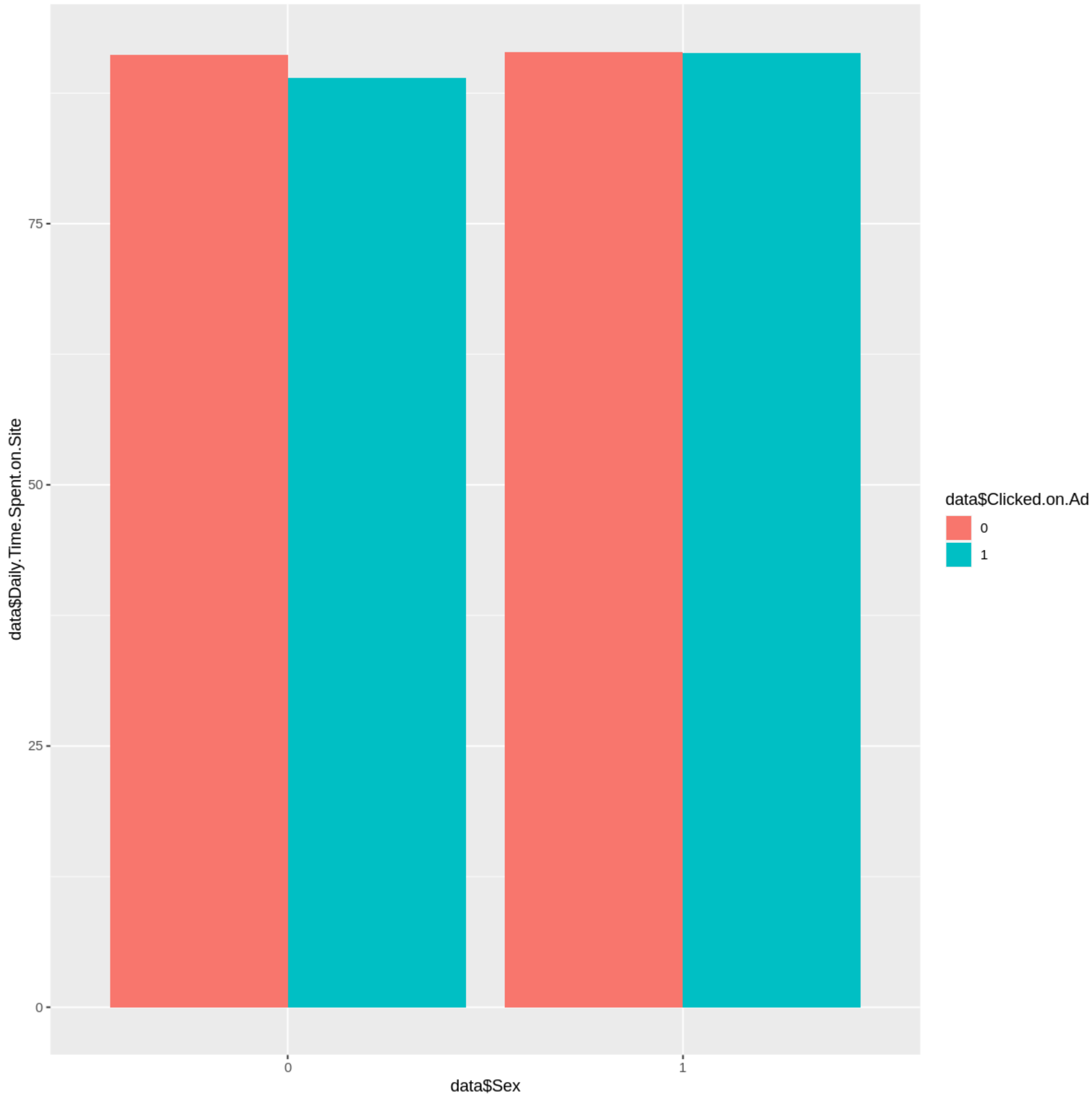
```
#A multivariate plot showing the relationship between Age, Sex and Clicked.on.Ad.
library(ggplot2)
ggplot(data, aes(fill=data$Clicked.on.Ad, y=data$Age, x=data$Sex)) +
    geom_bar(position="dodge", stat="identity")
```



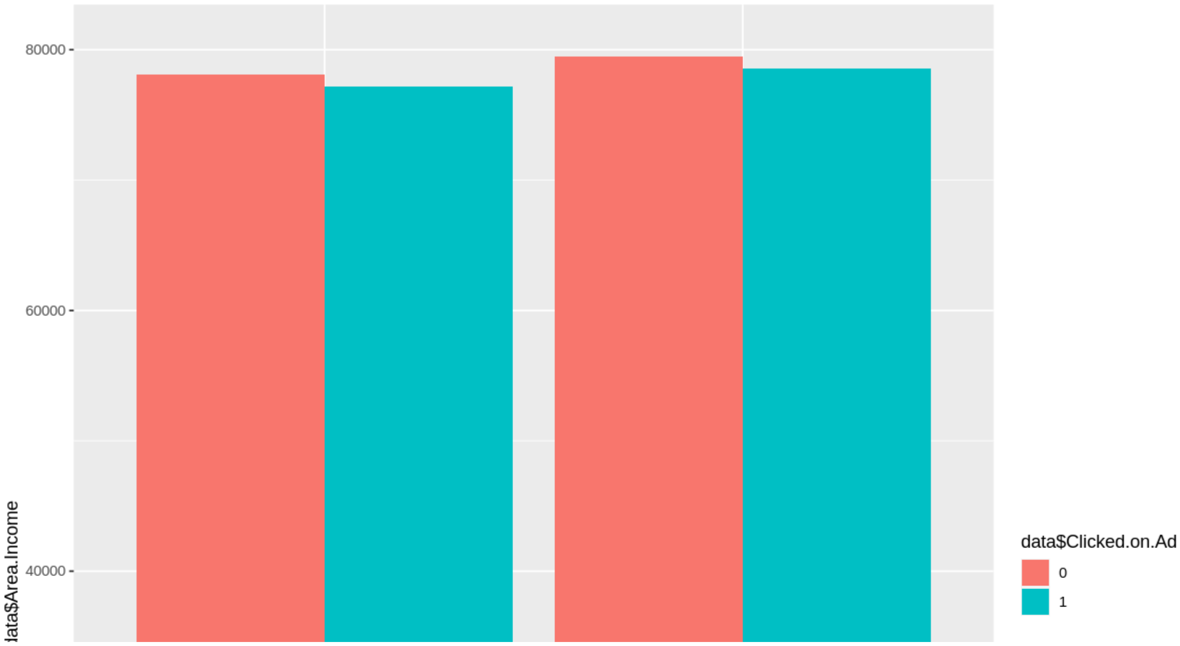

```
#A multivariate plot showing the relationship between Daily.Internet.Usage, Sex and Clicked.c
ggplot(data, aes(fill=data$Clicked.on.Ad, y=data$Daily.Internet.Usage, x=data$Sex)) +
    geom_bar(position="dodge", stat="identity")
```




```
#A multivariate plot showing the relationship between Daily.Time.Spent.on.Site, Sex and Click
ggplot(data, aes(fill=data$Clicked.on.Ad, y=data$Daily.Time.Spent.on.Site, x=data$Sex)) +
    geom_bar(position="dodge", stat="identity")
```




```
#A multivariate plot showing the relationship between Area.Income, Sex and Clicked.on.Ad.
ggplot(data, aes(fill=data$Clicked.on.Ad, y=data$Area.Income, x=data$Sex)) +
    geom_bar(position="dodge", stat="identity")
```



Conclusions and Recommendations

Conclusions

- Most people who clicked on the ad are female.
- Most people who clicked on the ad are in their 30's, 40's and 50's.
- People from Lake David and Lake James are the most frequent in terms of clicking the ad.
- People from Australia are found to click the ad the most.
- The most frequent month in which the ad was clicked is February.
- The most frequent day of the month is day 3.
- The most frequent hour is 9:00 am.

Recommendations

I would recommend the Kenyan entrepreneur to Target the following market:

- Mostly females since they seem to click her ad the most.
- People in the age of 30's, 40's and 50's since they are the ones who are the most interested.
- People from cities like King David and King James.
- People from Australia and other leading countries in terms of clicking the ad.
- She should also consider the time, day and month that people are most likely to click her ad such as, February, day 3 of the month and also 9:00 am.

Modelling

Feature Engineering

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

A data.table: 6 ×

Daily.Time.Spent.on.Site	Age	Area.Income	Daily.Internet.Usage	Ad.Topic.Line
<dbl>	<int>	<dbl>	<dbl>	<chr>
68.95	35	61833.90	256.09	Cloned 5thgeneration orchestration
80.23	31	68441.85	193.77	Monitored national standardization
69.47	26	59785.94	236.50	Organic bottom-line service-desk
74.15	29	54806.18	245.89	Triple-buffered reciprocal time-frame
68.37	35	73889.99	225.58	Robust logistical utilization
59.99	23	59761.56	226.74	Sharable client-driven software

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

```
Classes ‘data.table’ and 'data.frame':  1000 obs. of  13 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 natio
 $ City                    : chr  "Wrightburgh" "West Jodi" "Davidton" "West Terrifurt"
 $ Sex                     : Factor w/ 2 levels "0","1": 1 2 1 2 1 2 1 2 2 2 ...
 $ Country                 : chr  "Tunisia" "Nauru" "San Marino" "Italy" ...
 $ Clicked.on.Ad           : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 2 1 1 ...
 $ Year                    : Factor w/ 1 level "2016": 1 1 1 1 1 1 1 1 1 1 ...
 $ Month                   : Factor w/ 7 levels "01","02","03",..: 3 4 3 1 6 5 1 3 4 7
 $ Day                     : Factor w/ 31 levels "01","02","03",..: 27 4 13 10 3 19 28
```

```
 $ Hour                    : Factor w/ 24 levels "00","01","02",..: 1 2 21 3 4 15 21 2
 - attr(*, ".internal.selfref")=<externalptr>
```

```
#Changing the Sex, Month, Day and Hour column to numeric to make it easier for modelling.
data$Sex <- as.numeric(data$Sex)
data$Month <- as.numeric(data$Month)
data$Day <- as.numeric(data$Day)
data$Hour <- as.numeric(data$Hour)



#Dropping non numerical columns.
data$Ad.Topic.Line <- NULL
data$City <- NULL
data$Country <- NULL
data$Year <- NULL


#Previewing to confirm if changes are made.
head(data)
```

A data.table: 6 × 9

Daily.Time.Spent.on.Site	Age	Area.Income	Daily.Internet.Usage	Sex	Clicke
<dbl>	<int>	<dbl>	<dbl>	<dbl>	
68.95	35	61833.90	256.09	1	
80.23	31	68441.85	193.77	2	
69.47	26	59785.94	236.50	1	
74.15	29	54806.18	245.89	2	
68.37	35	73889.99	225.58	1	
59.99	23	59761.56	226.74	2	

```
# Normalizing the dataset so that no particular attribute
# has more impact on clustering algorithm than others.
normalize <- function(x){
  return ((x-min(x)) / (max(x)-min(x)))
}
data$Age<- normalize(data$Age)
data$Area.Income<- normalize(data$Area.Income)
data$Daily.Internet.Usage<- normalize(data$Daily.Internet.Usage)
data$Daily.Time.Spent.on.Site<- normalize(data$Daily.Time.Spent.on.Site)
data$Day<- normalize(data$Day)
data$Sex<- normalize(data$Sex)
data$Month<- normalize(data$Month)
data$Hour<- normalize(data$Hour)
```

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

748 · 553 · 763 · 725 · 447 · 554

```
#Checking if randomization has occurred.
data <- data[shuffle_index, ]
head(data)
```

A data.table: 6 × 9

Daily.Time.Spent.on.Site	Age	Area.Income	Daily.Internet.Usage	Sex	Cli
<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	
0.28097909	0.3095238	0.4138246	0.09510837	0	
0.40438552	0.9285714	0.2787655	0.29937038	0	
0.35747068	0.8333333	0.5548658	0.13022158	1	
0.75063743	0.1428571	0.5569218	0.76274367	1	
0.09348972	0.3571429	0.7122584	0.66351859	1	
0.20618732	0.3809524	0.6309596	0.13791016	1	

```
#Splitting the dataset into 70-30 splits.
dt = sort(sample(nrow(data), nrow(data)*.7))
train<-data[dt,]
test<-data[-dt,]
```

```
#Checking the dimensions of the train and test dataset.
dim(train)
dim(test)
```

700 · 9
300 · 9

```
# checking the dimensions of our splits
prop.table(table(data$Clicked.on.Ad)) * 100
prop.table(table(train$Clicked.on.Ad)) * 100
prop.table(table(test$Clicked.on.Ad)) * 100
```

```
 0  1
50 50

       0        1
48.14286 51.85714
```

KNN

```
# splitting into train and test sets without the target variable
data_train <- train[, -6]
data_test <- test[, -6]

# storing the training and test sets' target variable
train_label <- data_train[, train$Clicked.on.Ad]
test_label <- data_test[, test$Clicked.on.Ad]


#Checking the dimensions of the train and test dataset.
dim(data_train)
dim(data_test)
#Checking the length of the train and test target variable.
length(train_label)
length(test_label)
```

700 · 8
300 · 8
700
300

```
#Importing the necessary libraries
#Fitting the KNN model.
library(class)
require(class)
model <- knn(train= data_train,test=data_test, ,cl= train_label,k=13)
table(factor(model))
knn_table <- table(test_label,model)
knn_table
```

```
  0   1
168 132
          model
test_label   0   1
         0 162   1
         1   6 131
```

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

```
          test_label
     model   0   1
         0 162   6
         1   1 131
```

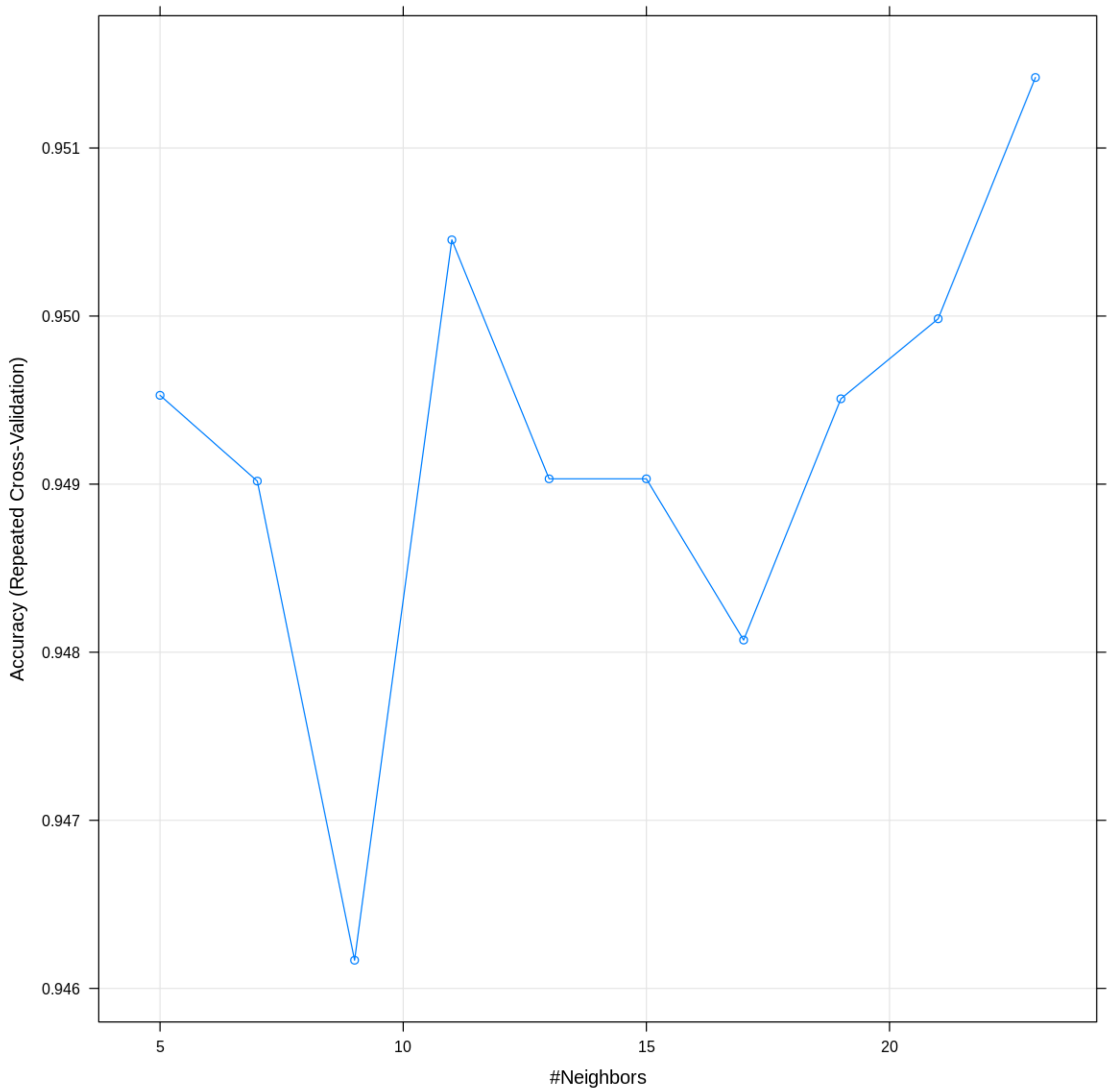
```
# calculating accuracy
accuracy <- sum(diag(knn_table)/(sum(rowSums(knn_table)))) * 100
print(paste("KNN accuracy score:", accuracy))
```

```
     [1] "KNN accuracy score: 97.6666666666667"
```

```
install.packages('e1071', dependencies=TRUE)
install.packages("caret")
library(caret)
trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)
set.seed(3333)
knn_fit <- train(Clicked.on.Ad ~., data = train, method = "knn",
 trControl=trctrl,
 preProcess = c("center", "scale"),
 tuneLength = 10)
 knn_fit
```

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

```
plot(knn_fit)
```




```
library(class)
require(class)
model1 <- knn(train= data_train,test=data_test, ,cl= train_label,k=23)
table(factor(model1))
knn_table1 <- table(test_label,model1)
knn_table1
```

```
# calculating accuracy
accuracy <- sum(diag(knn_table1)/(sum(rowSums(knn_table1)))) * 100
print(paste("KNN accuracy score:", accuracy))
```

```
[1] "KNN accuracy score: 97.6666666666667"
```

Our KNN model has an accuracy of 97.66% which remains the same even after hyperparameter tuning.

Decision Trees

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

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

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

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

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

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

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.
```

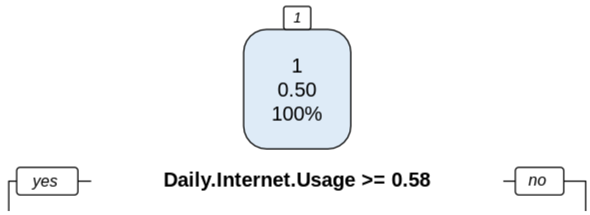
```
dim(dt_test)
dim(dt_train)
```

200 · 9
800 · 9

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

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

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




```
#Testing the model
pred <- predict(object=tree,dt_test[,-6],type="class")
pred
```

```
1: 1 2: 0 3: 1 4: 1 5: 1 6: 1 7: 1 8: 0 9: 1 10:
  1 11: 1 12: 1 13: 1 14: 0 15: 0 16: 0 17: 1 18:
  1 19: 1 20: 0 21: 1 22: 0 23: 1 24: 1 25: 0 26:
  0 27: 1 28: 0 29: 0 30: 0 31: 0 32: 1 33: 1 34:
  0 35: 1 36: 1 37: 1 38: 0 39: 1 40: 1 41: 0 42:
  1 43: 1 44: 1 45: 0 46: 0 47: 1 48: 1 49: 1 50:
  1 51: 0 52: 0 53: 1 54: 1 55: 1 56: 0 57: 0 58:
  1 59: 0 60: 0 61: 1 62: 1 63: 0 64: 0 65: 0 66:
  0 67: 1 68: 0 69: 0 70: 0 71: 0 72: 1 73: 1 74:
  1 75: 0 76: 1 77: 1 78: 1 79: 1 80: 1 81: 0 82:
  0 83: 0 84: 1 85: 1 86: 1 87: 1 88: 1 89: 1 90:
  0 91: 1 92: 0 93: 0 94: 0 95: 1 96: 1 97: 1 98:
  1 99: 0 100: 1 101: 0 102: 1 103: 1 104: 0 105: 1
106: 0 107: 0 108: 1 109: 1 110: 0 111: 1 112: 1 113:
  1 114: 0 115: 1 116: 1 117: 1 118: 1 119: 1 120: 0
121: 1 122: 1 123: 0 124: 0 125: 1 126: 0 127: 1 128:
  1 129: 1 130: 0 131: 0 132: 0 133: 0 134: 0 135: 1
136: 0 137: 0 138: 1 139: 0 140: 0 141: 0 142: 1 143:
  0 144: 0 145: 0 146: 0 147: 1 148: 1 149: 1 150: 1
151: 0 152: 1 153: 1 154: 1 155: 1 156: 1 157: 1 158:
```

```
length(pred)
```

200

```
length(dt_test$Clicked.on.Ad)
```

200

```
#Calculating accuracy
t <- table(dt_test$Clicked.on.Ad,pred)
confusionMatrix(t)
```

```
Confusion Matrix and Statistics

   pred
     0  1
  0 82 15
  1  6 97

               Accuracy : 0.895
                 95% CI : (0.844, 0.9338)
    No Information Rate : 0.56
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.7892

 Mcnemar's Test P-Value : 0.08086

            Sensitivity : 0.9318
            Specificity : 0.8661
         Pos Pred Value : 0.8454
         Neg Pred Value : 0.9417
             Prevalence : 0.4400
```

Using decision trees our accuracy is 89.5%

```
      Balanced Accuracy : 0.8989
```

svm

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

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


Attaching package: ‘e1071’


The following objects are masked from ‘package:moments’:

    kurtosis, moment, skewness


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

```
#Splitting into test and train
intrain <- createDataPartition(y = data$Clicked.on.Ad, p= 0.7, list = FALSE)
```

```
training <- data[intrain,]
testing <- data[-intrain,]
```

```
#Checking the dimensions of the train and test dataset.
dim(training);
dim(testing);
```

700 · 9
300 · 9

```
#Checking if there are any null values in the data.
anyNA(data)
```

FALSE

```
#Changing the target variable into a factor.
training[["Clicked.on.Ad"]] = factor(training[["Clicked.on.Ad"]])
```

```
#Finding the best parameters of the model.
trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)
```

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

```
svm_Linear
```

```
Support Vector Machines with Linear Kernel

700 samples
  8 predictor
  2 classes: '0', '1'

Pre-processing: centered (8), scaled (8)
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 630, 630, 630, 630, 630, 630, ...
Resampling results:

  Accuracy  Kappa
  0.97      0.94

Tuning parameter 'C' was held constant at a value of 1
```

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

```
1·1·1·1·0·1·1·1·1·1·0·0·1·0·1·1·0·0·1·0·0·1·1·1·0·1·1·0·1·1·1·
1·1·1·1·1·1·0·0·0·0·0·0·1·1·0·1·1·0·1·0·0·0·1·1·1·0·0·0·1·0·1·
0·0·0·1·1·0·0·0·0·1·0·1·1·1·1·1·1·0·1·0·0·0·0·0·0·0·1·0·0·0·1·
1·0·0·0·0·0·0·1·1·0·0·1·1·0·1·0·0·1·0·0·0·0·1·0·0·1·1·0·1·1·0·
0·1·0·0·0·1·1·1·1·1·1·0·1·1·1·0·0·1·1·0·1·1·0·1·0·0·1·0·1·1·0·
0·0·0·0·1·0·1·0·0·0·1·1·1·1·1·0·0·1·0·0·0·1·1·0·1·1·1·0·0·1·0·
0·0·0·0·1·0·1·0·1·1·0·1·1·0·0·0·1·0·1·1·0·0·1·1·1·1·1·0·1·1·1·
1·1·0·0·1·1·0·1·1·1·1·1·0·1·1·0·0·0·0·1·0·1·1·1·0·1·0·1·1·1·0·
```

```
#Computing the confusion matrix
confusionMatrix(table(test_pred, testing$Clicked.on.Ad))
```

```
Confusion Matrix and Statistics


test_pred   0   1
        0 145   6
        1   5 144

               Accuracy : 0.9633
                 95% CI : (0.9353, 0.9816)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9267

 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9667
            Specificity : 0.9600
         Pos Pred Value : 0.9603
         Neg Pred Value : 0.9664
             Prevalence : 0.5000
         Detection Rate : 0.4833
   Detection Prevalence : 0.5033
      Balanced Accuracy : 0.9633

       'Positive' Class : 0
```

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

```
Warning message:
“model fit failed for Fold01.Rep1: C=0.00 Error in .local(x, ...) :
  No Support Vectors found. You may want to change your parameters
”
Warning message:
“model fit failed for Fold02.Rep1: C=0.00 Error in .local(x, ...) :
  No Support Vectors found. You may want to change your parameters
”
Warning message:
“model fit failed for Fold03.Rep1: C=0.00 Error in .local(x, ...) :
  No Support Vectors found. You may want to change your parameters
”
Warning message:
“model fit failed for Fold04.Rep1: C=0.00 Error in .local(x, ...) :
  No Support Vectors found. You may want to change your parameters
”
Warning message:
“model fit failed for Fold05.Rep1: C=0.00 Error in .local(x, ...) :
  No Support Vectors found. You may want to change your parameters
”
Warning message:
“model fit failed for Fold06.Rep1: C=0.00 Error in .local(x, ...) :
  No Support Vectors found. You may want to change your parameters
”
Warning message:
“model fit failed for Fold07.Rep1: C=0.00 Error in .local(x, ...) :
  No Support Vectors found. You may want to change your parameters
”
Warning message:
“model fit failed for Fold08.Rep1: C=0.00 Error in .local(x, ...) :
  No Support Vectors found. You may want to change your parameters
”
Warning message:
“model fit failed for Fold09.Rep1: C=0.00 Error in .local(x, ...) :
  No Support Vectors found. You may want to change your parameters
”
Warning message:
“model fit failed for Fold10.Rep1: C=0.00 Error in .local(x, ...) :
  No Support Vectors found. You may want to change your parameters
”
Warning message:
“model fit failed for Fold01.Rep2: C=0.00 Error in .local(x, ...) :
  No Support Vectors found. You may want to change your parameters
”
Warning message:
“model fit failed for Fold02.Rep2: C=0.00 Error in .local(x, ...) :
  No Support Vectors found. You may want to change your parameters
”
Warning message:
“model fit failed for Fold03.Rep2: C=0.00 Error in .local(x, ...) :
  No Support Vectors found. You may want to change your parameters
”
```

```
#Making predictions with the model after tuning.
test_pred_grid <- predict(svm_Linear_Grid, newdata = testing)
test_pred_grid
```

```
1·1·1·1·0·1·1·1·1·1·0·0·1·0·1·1·0·0·1·0·0·1·1·1·0·1·1·0·1·1·1·
1·1·1·1·1·1·0·0·0·0·0·0·1·1·0·1·1·0·1·0·0·0·1·1·1·0·0·0·1·0·1·
0·0·0·1·1·0·0·0·0·1·0·1·1·1·1·1·1·0·1·0·0·0·0·0·0·0·1·0·0·0·1·
1·0·0·0·0·0·0·1·1·0·0·1·1·0·1·0·0·1·0·0·0·0·1·0·0·1·1·0·1·1·0·
0·1·0·0·0·1·1·1·1·1·1·0·1·1·1·0·0·1·1·0·1·1·0·1·0·0·1·0·1·1·0·
0·0·0·0·1·0·1·0·0·0·1·1·1·1·1·0·0·1·0·0·0·1·1·0·1·1·1·0·0·1·0·
0·0·0·0·1·0·1·0·1·1·0·1·1·0·0·0·1·0·1·1·0·0·1·1·1·1·1·0·1·1·1·
1·1·0·0·1·1·0·1·1·1·1·1·0·1·1·0·0·0·0·1·0·1·1·1·0·1·0·1·1·1·0·
1·0·0·0·0·0·0·1·1·0·0·1·0·0·0·0·0·0·1·0·0·1·1·0·1·1·0·0·1·1·1·
```

```
#Computing the confusion matrix after tuning.
confusionMatrix(table(test_pred_grid, testing$Clicked.on.Ad))
```

```
Confusion Matrix and Statistics


test_pred_grid   0   1
             0 145   6
             1   5 144

               Accuracy : 0.9633
                 95% CI : (0.9353, 0.9816)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9267

 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9667
            Specificity : 0.9600
         Pos Pred Value : 0.9603
         Neg Pred Value : 0.9664
             Prevalence : 0.5000
         Detection Rate : 0.4833
   Detection Prevalence : 0.5033
      Balanced Accuracy : 0.9633

       'Positive' Class : 0
```

The SVM model has an accuracy of 96.33% and it retains the same accuracy even after tuning.

Naive Bayes

```
#Loading required packages
install.packages('tidyverse')
library(tidyverse)
install.packages('ggplot2')
library(ggplot2)
```

```
install.packages('caret')
library(caret)
install.packages('caretEnsemble')
library(caretEnsemble)
install.packages('psych')
library(psych)
install.packages('Amelia')
library(Amelia)
install.packages('mice')
library(mice)
install.packages('GGally')
library(GGally)
install.packages('rpart')
library(rpart)
install.packages('randomForest')
library(randomForest)
```

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

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

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

── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
✖ dplyr::between()   masks data.table::between()
✖ dplyr::filter()    masks stats::filter()
✖ dplyr::first()     masks data.table::first()
✖ dplyr::lag()       masks stats::lag()
✖ dplyr::last()      masks data.table::last()
✖ purrr::lift()      masks caret::lift()
✖ purrr::transpose() masks data.table::transpose()

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

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

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


Attaching package: ‘caretEnsemble’


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

    autoplot
```

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


Attaching package: 'psych'


The following objects are masked from 'package:ggplot2':

    %+%, alpha


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

Loading required package: Rcpp

##
## Amelia II: Multiple Imputation
## (Version 1.8.0, built: 2021-05-26)
## Copyright (C) 2005-2021 James Honaker, Gary King and Matthew Blackwell
```

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


#Check dimensions of the split
prop.table(table(data$Clicked.on.Ad)) * 100
prop.table(table(training$Clicked.on.Ad)) * 100
prop.table(table(testing$Clicked.on.Ad)) * 100
```

```
 0  1
50 50

 0  1
50 50

 0  1
50 50
```

```
#create objects x which holds the predictor variables and y which holds the response variable
x = training[,-6]
y = training$Clicked.on.Ad


library(e1071)
```

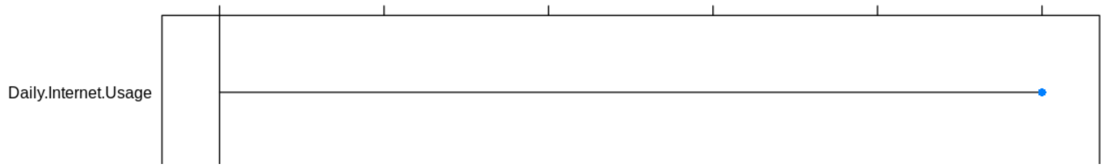
```
install.packages("klaR")
library(klaR)
model = train(x,y,'nb',trControl=trainControl(method='cv',number=10))

model
```

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

Loading required package: MASS


Attaching package: ‘MASS’


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

    select


Naive Bayes

750 samples
  8 predictor
  2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 675, 674, 674, 676, 675, 674, ...
Resampling results across tuning parameters:

  usekernel  Accuracy   Kappa
  FALSE      0.9679953  0.9359924
   TRUE      0.9640128  0.9280190

Tuning parameter 'fL' was held constant at a value of 0
Tuning
 parameter 'adjust' was held constant at a value of 1
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were fL = 0, usekernel = FALSE and adjust
 = 1.
```

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

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 120   3
         1   5 122

               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 : 0.9600
            Specificity : 0.9760
         Pos Pred Value : 0.9756
         Neg Pred Value : 0.9606
             Prevalence : 0.5000
         Detection Rate : 0.4800
```

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



Naive Bayes model has an accuracy of 96.8%. From the graph above we can observe that Daily.Internet.Usage plays a major role in determining whether a person clicks on an ad or not.Maybe it is because those people with limited data can avoid clicking on ads to save on data.

Conclusions

Our model performance is as follows:

- Naive Bayes model - 96.8%
- SVM model - 96.33%
- Decision Trees - 89.5%
- KNN model - 97.66%

We can conclude that our best model in this case is KNN model.

The most important features that help in determining if a person clicks an ad or not are;

- Daily internet usage
- Daily time spent on site
- Age
- Area income
- Hour

Recommendations

- The Kenyan Entrepreneur should use the KNN model to predict if a person is most likely to click her ad or not since it has the highest accuracy of 97.66%
- Since we have the most important features that can help us predict if a person will click on an ad or not, the Kenyan entrepreneur should try concentrationg on those features since those are her target market. For example, she should try and concentrate on people with high daily internet usage since they are most likely to click her ad. She should also concentrate on those who spend more time on the site and those between the age of 30 and 50.