online cryptography course project

Dennis Kiarie

8/27/2021

1. Define the Question

1.1 Research Question

Our Research seeks to identify which individuals are most likely to click on ads from the advertisement website.

1.2 Metric of Success

To identify individuals who are relatively more likely to click on course advertisement ads.

1.3 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.

1.4 Experimental Design

1. Loading Data into RStudio.
2. Checking the Data.
3. Tidying the Data
4. Conducting Exploratory Data Analysis i.e Univariate, Bivariate and Multivariate Analysis.
5. Challenging the Solution by providing insights on how we can make improvements.
6. Recommendations.

1.5 Data Relevance

The data provided was collected in the past from an advertisement website hence its appropriate for our analysis.

The dataset for this Analysis can be found from this link:[http://bit.ly/IPAdvertisingData].

Description of Variables used in this Analysis are :

1.Daily Time Spent on Site; consumer time on site in minutes.

2.Age; customer age in years.

3.Area; geographical area of a consumer.

4.Income; Avg. Income of a consumer.

5.Daily Internet Usage; Avg. minutes a day consumer is on the internet.

6.Ad Topic Line; Headline of the advertisement.

7.City; city of consumer.

8.Male; whether or not a consumer was a male.

9.Country; country of a consumer.

10.Timestamp; Time at which consumer clicked on Ad or closed window

11.Clicked on Ad; 1 indicated clicking on Ad and 0 indicates no clicking on ad.

The dataset used in this Analysis has 1,000 rows and 10 columns.

```
library(tinytex)
```

2.Data Preparation

```
## Importing libraries
#---
#
library(pacman)
pacman :: p_load(pacman, dplyr, GGally, ggplot2, ggthemes, ggvis, httr, lubridate, plotly, rio , rmarkd
```

```
## Installing package into 'C:/Users/Denoo/OneDrive/Documents/R/win-library/4.1'
## (as 'lib' is unspecified)

## Warning: package 'rtools' is not available for this version of R
##
## A version of this package for your version of R might be available elsewhere,
## see the ideas at
## https://cran.r-project.org/doc/manuals/r-patched/R-admin.html#Installing-packages

## Warning: unable to access index for repository http://www.stats.ox.ac.uk/pub/RWin/bin/windows/contri
##   cannot open URL 'http://www.stats.ox.ac.uk/pub/RWin/bin/windows/contrib/4.1/PACKAGES'

## Warning: 'BiocManager' not available.  Could not check Bioconductor.
##
## Please use `install.packages('BiocManager')` and then retry.

## Warning in p_install(package, character.only = TRUE, ...):

## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'rtools'

## Warning in pacman::p_load(pacman, dplyr, GGally, ggplot2, ggthemes, ggvis, : Failed to install/load:
## rtools
```

```
theme_set(theme_classic())
options(warn = -1)
```

```
## Loading the data from a csv file
#---
#
ad <- read.csv("C:/Users/Denoo/Downloads/advertising.csv", na.strings = "")

##preview the first 6 rows
#---
#
head(ad)
```

```
##   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
```

```
##preview the last 6 records of the dataset
#---
#
tail(ad)
```

```
##      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
```

3. Checking the data

```
## we check for the number of rows and columns
#---
```

```
#
cat("Rows:", nrow(ad), "\nCols:", ncol(ad))
```

```
## Rows: 1000
## Cols: 10
```

```
##we check the type
#---
#
class(ad)
```

```
## [1] "data.frame"
```

```
##we check if datatypes are appropriate
#---
#
glimpse(ad)
```

```
## 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, ~
```

The column for Timestamp has an inappropriate datatype which will be rectified during data cleaning process.

```
##we check the column names for easier reference
#---
#
names(ad)
```

```
##  [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"
```

```
## we Check for unique characters
sapply(ad, function(x) length(unique(x)))
```

```
## Daily.Time.Spent.on.Site                      Age              Area.Income
##                      900                       43                     1000
```

```
##     Daily.Internet.Usage            Ad.Topic.Line                     City
##                      966                     1000                      969
##                     Male                  Country                Timestamp
##                        2                      237                     1000
##            Clicked.on.Ad
##                        2
```

4.Tidying the data

```
##we change the column names to lowercase for easier manipulation
#---
#
colnames(ad) = tolower(colnames(ad))
colnames(ad)
```

```
##  [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"
```

```
##we replace spaces in column names for easier manipulation
#---

names(ad) = str_replace_all(names(ad), c(' ' = '_'))
names(ad)
```

```
##  [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"
```

```
##we rename column male to gender
#---
#

colnames(ad)[colnames(ad) == 'male'] = 'gender'
```

```
## we check for the datatypes for each column
#---
#
columns = names(ad)
for (column in seq(length(names(ad)))){
    print(columns[column])
    print(class(ad[, column]))
    cat('\n')
}
```

```
## [1] "daily.time.spent.on.site"
## [1] "numeric"
##
```

```
## [1] "age"
## [1] "integer"
##
## [1] "area.income"
## [1] "numeric"
##
## [1] "daily.internet.usage"
## [1] "numeric"
##
## [1] "ad.topic.line"
## [1] "character"
##
## [1] "city"
## [1] "character"
##
## [1] "gender"
## [1] "integer"
##
## [1] "country"
## [1] "character"
##
## [1] "timestamp"
## [1] "character"
##
## [1] "clicked.on.ad"
## [1] "integer"
```

```
## we create a list of categorical columns
#---
#
cat_cols = c("ad_topic_line", "city", "gender", "country", "clicked_on_ad" )

## we create a List of numerical columns
num_cols = c("daily_time_spent_on_site", "age", "area_income", "daily_internet_usage")
```

```
## we Change the datatypes
#---
#
ad$gender <- as.factor(ad$gender)
ad$clicked.on.ad <- as.factor(ad$clicked.on.ad)

##we check for datatypes
sapply(ad, class)
```

```
## daily.time.spent.on.site                      age             area.income
##                "numeric"                "integer"               "numeric"
##     daily.internet.usage            ad.topic.line                     city
##                "numeric"              "character"             "character"
##                   gender                  country               timestamp
##                 "factor"              "character"             "character"
##            clicked.on.ad
##                 "factor"
```

```
##we check for duplicates
#---
#
anyDuplicated(ad)
```

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

```
#There are no duplicates
```

```
##we check for missing values
#---
#
colSums(is.na(ad))
```

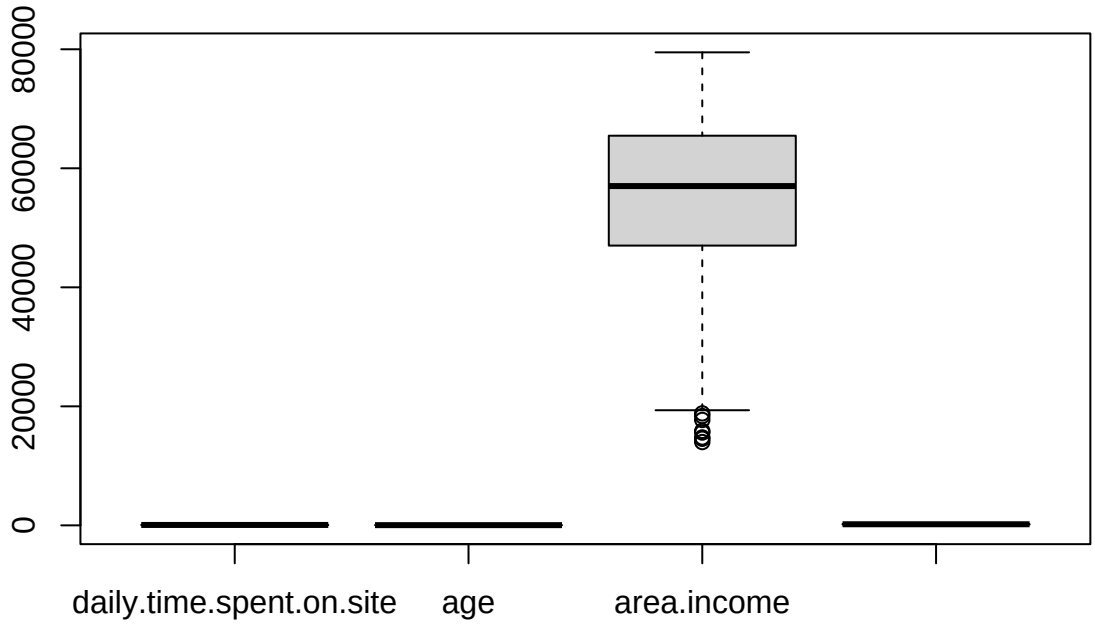
```
## daily.time.spent.on.site                      age             area.income
##                        0                        0                        0
##     daily.internet.usage           ad.topic.line                     city
##                        0                        0                        0
##                   gender                  country                timestamp
##                        0                        0                        0
##            clicked.on.ad
##                        0
```

```
#we have no missing values
```

```
##we check for outliers in numerical columns
#---
#
df <- subset(ad, select = -c(ad.topic.line,city,    gender, country,    timestamp,  clicked.on.ad))

##preview 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
```

```
## we Plot boxplots to check for outliers
#---
#
boxplot(df)
```




```
# Only area.income variable has a few outliers of individuals  earning less than 20000.
#We investigate them so as to see whether they are legit or not.
```

```
## we select all the rows with outliers
#---
#
outliers <- df %>% dplyr::filter(area.income < 20000)
outliers
```

```
##    daily.time.spent.on.site age area.income daily.internet.usage
## 1                     49.89  39    17709.98               160.03
## 2                     63.88  38    19991.72               136.85
## 3                     48.09  33    19345.36               180.42
## 4                     57.86  30    18819.34               166.86
## 5                     64.63  45    15598.29               158.80
## 6                     58.05  32    15879.10               195.54
## 7                     66.26  47    14548.06               179.04
## 8                     68.58  41    13996.50               171.54
## 9                     52.67  44    14775.50               191.26
## 10                    62.79  36    18368.57               231.87
```

Conclusion

We choose to work with outliers.This because individuals earn different amount of income. Also, from our dataset the currency of income is not defined and we are not told if they are annual,monthly or weekly

income. Therefore, removing this outliers will affect the accuarcy of the data analysis and the result will be inconclusive.

```
## we Change column timestamp to datetime
#---
#
ad$Timestamp <- as.POSIXct(ad$timestamp, "%Y-%m-%d %H:%M:%S",tz = "GMT")

##preview the data
head(ad)
```

```
##   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 gender    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           Timestamp
## 1 2016-03-27 00:53:11             0 2016-03-27 00:53:11
## 2 2016-04-04 01:39:02             0 2016-04-04 01:39:02
## 3 2016-03-13 20:35:42             0 2016-03-13 20:35:42
## 4 2016-01-10 02:31:19             0 2016-01-10 02:31:19
## 5 2016-06-03 03:36:18             0 2016-06-03 03:36:18
## 6 2016-05-19 14:30:17             0 2016-05-19 14:30:17
```

```
## we split time and date from Timestamp
#---
#
ad$date = format(ad$Timestamp, "%y/%m/%d")
ad$time = format(ad$Timestamp, "%H:%M:%S")
ad$date <- as.Date(ad$date)

#preview the data
head(ad)
```

```
##   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 gender    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           Timestamp       date     time
## 1 2016-03-27 00:53:11             0 2016-03-27 00:53:11 0016-03-27 00:53:11
## 2 2016-04-04 01:39:02             0 2016-04-04 01:39:02 0016-04-04 01:39:02
## 3 2016-03-13 20:35:42             0 2016-03-13 20:35:42 0016-03-13 20:35:42
## 4 2016-01-10 02:31:19             0 2016-01-10 02:31:19 0016-01-10 02:31:19
## 5 2016-06-03 03:36:18             0 2016-06-03 03:36:18 0016-06-03 03:36:18
## 6 2016-05-19 14:30:17             0 2016-05-19 14:30:17 0016-05-19 14:30:17
```

```
##we drop the column Timestamp
#---
#
final_df = subset(ad, select = -c(Timestamp))
#preview the data
head(final_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 gender    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       date     time
## 1 2016-03-27 00:53:11             0 0016-03-27 00:53:11
## 2 2016-04-04 01:39:02             0 0016-04-04 01:39:02
## 3 2016-03-13 20:35:42             0 0016-03-13 20:35:42
## 4 2016-01-10 02:31:19             0 0016-01-10 02:31:19
## 5 2016-06-03 03:36:18             0 0016-06-03 03:36:18
## 6 2016-05-19 14:30:17             0 0016-05-19 14:30:17
```

```
## we use frequency tables to check data distribution in gender column
#---
#
table(final_df$gender)
```

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

Conclusion

0 represent female and 1 represents male. The data we used for analysis had a slight higher number of females than males hence gender variable is fairly distributed.

```
##we rename the target variable
#---
#
levels(final_df$clicked.on.ad) = c("Yes", "No")

#preview the data
head(final_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 gender    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       date     time
## 1 2016-03-27 00:53:11           Yes 0016-03-27 00:53:11
## 2 2016-04-04 01:39:02           Yes 0016-04-04 01:39:02
## 3 2016-03-13 20:35:42           Yes 0016-03-13 20:35:42
## 4 2016-01-10 02:31:19           Yes 0016-01-10 02:31:19
## 5 2016-06-03 03:36:18           Yes 0016-06-03 03:36:18
## 6 2016-05-19 14:30:17           Yes 0016-05-19 14:30:17
```

5. Exploratory Data Analysis

5.1 Univariate Analysis

We decided to analyze all columns individually so as to pay close attention to the findings

```
time <- final_df$`daily.time.spent.on.site`
age <- final_df$age
income <- final_df$`area.income`
usage <- final_df$`daily.internet.usage`
```

5.1.1 Age

Measure of Central Tendency

```
summary(time)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   32.60   51.36   68.22   65.00   78.55   91.43
```

```
## we calculate the mode
#---
#
```

```r
getmode <- function(v) #getmode is the function name
  {
   uniqv <- unique(v)
   uniqv[which.max(tabulate(match(v, uniqv)))]
}

#we Calculate the mode using the user function.
mode <- getmode(time)
print(mode)
```

```
## [1] 62.26
```

Measure of Dispersion

```r
##install package moments
#---
#
library(moments)
```

```r
##The Variance
#---
#
var(time)
```

```
## [1] 251.3371
```

```r
##The standard deviation
#---
#
sd(time)
```

```
## [1] 15.85361
```

```r
##The range of the Variable
#---
#
range(time)
```

```
## [1] 32.60 91.43
```

```r
##The Interquartile Range
#---
#
IQR(age)
```

```
## [1] 13
```

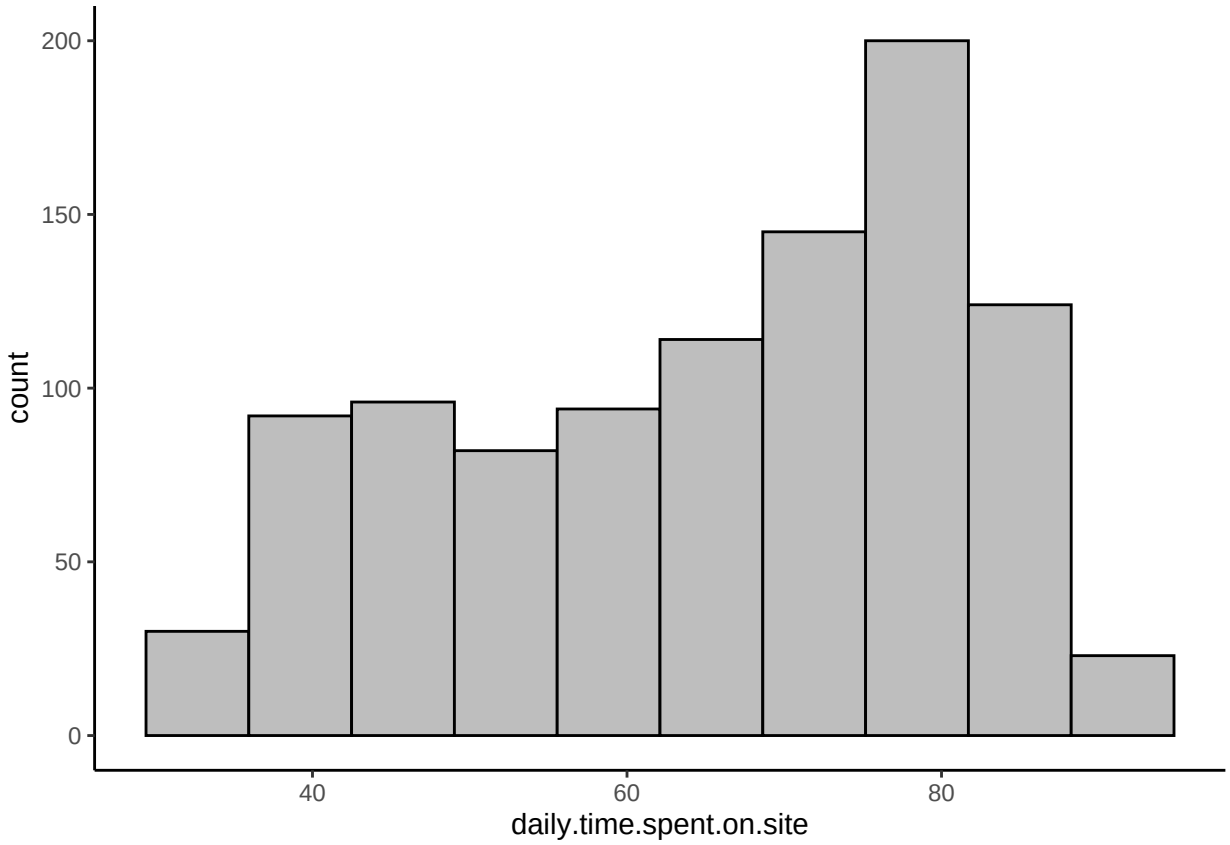
```r
##The Skew of the column
skewness(time)
```

```
## [1] -0.3712026
```

```r
##The Kurtosis
#---
#
kurtosis(time)
```

```
## [1] 1.903942
```

```r
##Histogram with density plot
#---
#
ggplot(final_df, aes(x=`daily.time.spent.on.site`)) +
 geom_histogram(colour="black", fill="grey",bins=10)
```



Conclusion

From the variable time we observed he following:

1.The mean amount of time that users spent on the site was 65 minutes while the maximum amount of time a user spent on the site was 91.43 minutes.

2.The Variance of the column was 251.3371 with a standard deviation of 15.85361.

3.The data was negatively but fairly symmetrical with a value of -0.3712026 and the distribution can be categorized as platykurtic with a kurtosis value of 1.903942.

4.From the Histogram we deduced that a pproxiamtely 125 users spent over 80 minutes daily on the site , and many users spent over 60 minutes on the site daily.

5.1.2 Age

Measure of central Tendency

```
summary(age)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   19.00   29.00   35.00   36.01   42.00   61.00
```

```
##we calculate the mode
#---
#
getmode <- function(v) #getmode is the function name
  {
   uniqv <- unique(v)
   uniqv[which.max(tabulate(match(v, uniqv)))]
}
## we Calculate the mode using the user function.
mode <- getmode(age)
print(mode)
```

```
## [1] 31
```

Measure of Disperson

```
##The Variance
#---
#
var(age)
```

```
## [1] 77.18611
```

```
##The standard deviation
#---
#
sd(age)
```

```
## [1] 8.785562
```

```
##The range of the Variable
#---
#
range(age)
```

```
## [1] 19 61
```

```
##The Interquartile Range
#---
#
IQR(age)
```

```
## [1] 13
```

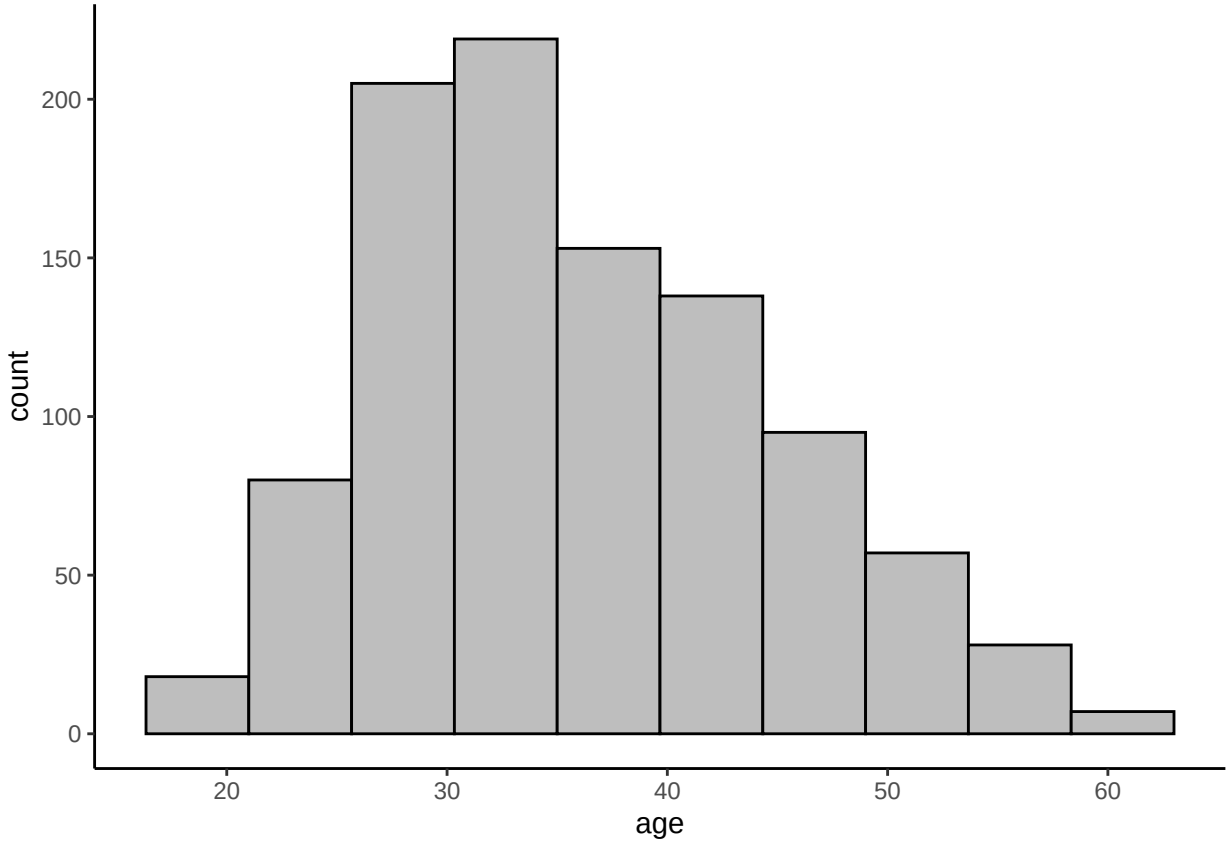
```
##The Skew of the column
skewness(age)
```

```
## [1] 0.4784227
```

```
##The Kurtosis
#---
#
kurtosis(age)
```

```
## [1] 2.595482
```

```
##Histogram with density plot
#---
#
ggplot(final_df, aes(x=`age`)) +
 geom_histogram(colour="black", fill="grey",bins=10)
```



Conclusion

1.The mean age of the consumers was 36.01 while the median age was 35 and the modal age was 31.

2.The variable was positively skewed and fairly symmetrical with a skew value of 0.4777052 , the distribution was platykurtic with a kurtosis value of 2.595482

3.The age of the consumers ranged between 19 and 61 , with majority of the users ranging between 30 and 45.

4.The interquartile age for the upper and lower quartile was 13.

5.1.3 Area Income

Measure of central Tendency

```
summary(income)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   13996   47032   57012   55000   65471   79485
```

```
##we calculate the mode
#---
#
getmode <- function(v) #getmode is the function name
  {
   uniqv <- unique(v)
   uniqv[which.max(tabulate(match(v, uniqv)))]
}
## we Calculate the mode using the user function.
mode <- getmode(income)
print(mode)
```

```
## [1] 61833.9
```

Measure of Dispersion

```
##The Variance
#---
#
var(income)
```

```
## [1] 179952406
```

```
##we check the standard deviation
#---
#
sd(income)
```

```
## [1] 13414.63
```

```
##we check the range
#---
#
range(income)
```

```
## [1] 13996.5 79484.8
```

```r
##we check Interquartile Range
#---
#
IQR(income)
```

```
## [1] 18438.83
```

```r
##we check the skewness
#---
3
```

```
## [1] 3
```

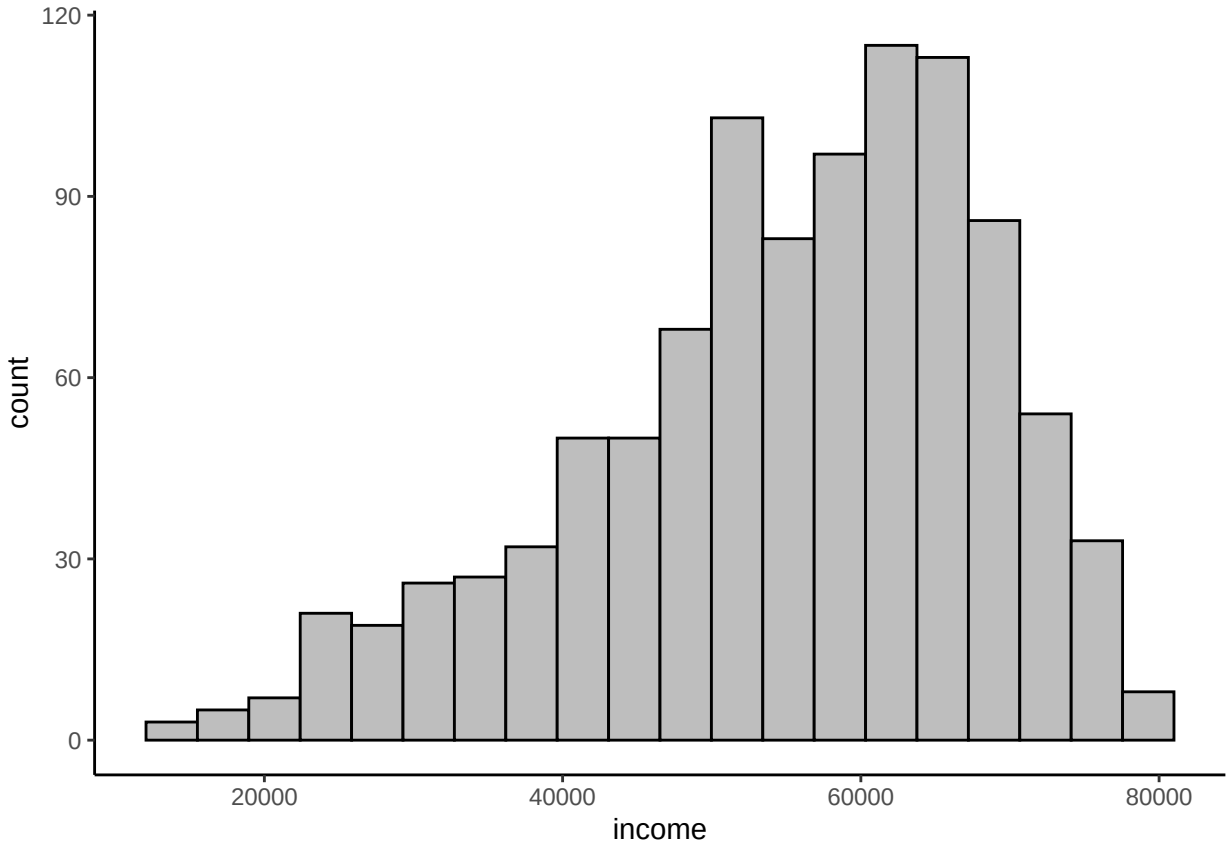
```r
skewness(income)
```

```
## [1] -0.6493967
```

```r
##we check for kurtosis
#---
#
kurtosis(income)
```

```
## [1] 2.894694
```

```r
##Histogram with density plot
#---
#
ggplot(final_df, aes(x=`income`)) +
 geom_histogram(colour="black", fill="grey",bins=20)
```



Conclusion

1.The mean income of the users was 55,000 while the median income was 57,012 and the modal income was 61,833.9.

2.The income of the site users ranged from 13,996 to 79,485 , with the interquarile range being 18483.83.

3.The data had a negative skew and was moderately skewed with a value of -0.6484229, the data had a platykurtic distribution with a value of 2.894694.

5.1.4 Daily Internet Usage

Measure of central Tendency

```
summary(usage)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   104.8   138.8   183.1   180.0   218.8   270.0
```

```
##we calculate the mode
#---
#
getmode <- function(v) #getmode is the function name
  {
   uniqv <- unique(v)
   uniqv[which.max(tabulate(match(v, uniqv)))]
}
## we Calculate the mode using the user function.
```

```
mode <- getmode(usage)
print(mode)
```

```
## [1] 167.22
```

Measure of Disperson

```
##we check for standard deviation
#---
#
sd(usage)
```

```
## [1] 43.90234
```

```
##we check for variance
#---
#
var(usage)
```

```
## [1] 1927.415
```

```
##we check for range
#---
#
range(usage)
```

```
## [1] 104.78 269.96
```

```
##we check for Interquatile range
#---
#
IQR(usage)
```

```
## [1] 79.9625
```

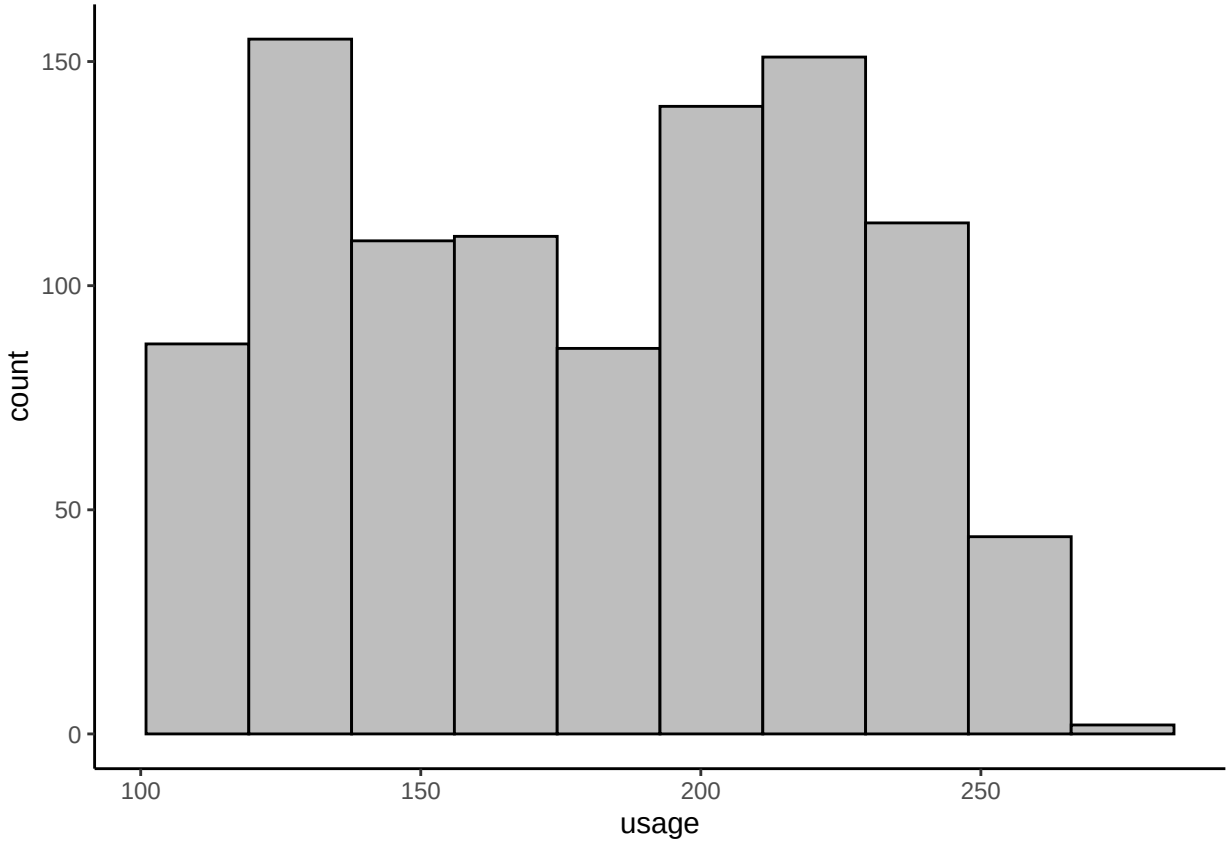
```
##we check the skewness
#---
#
skewness(usage)
```

```
## [1] -0.03348703
```

```
##we check for the kurtosis
#---
#
kurtosis(usage)
```

```
## [1] 1.727701
```

```
##Histogram with density plot
#---
#
ggplot(final_df, aes(x=`usage`)) +
 geom_histogram(colour="black", fill="grey",bins=10)
```



Conclusion

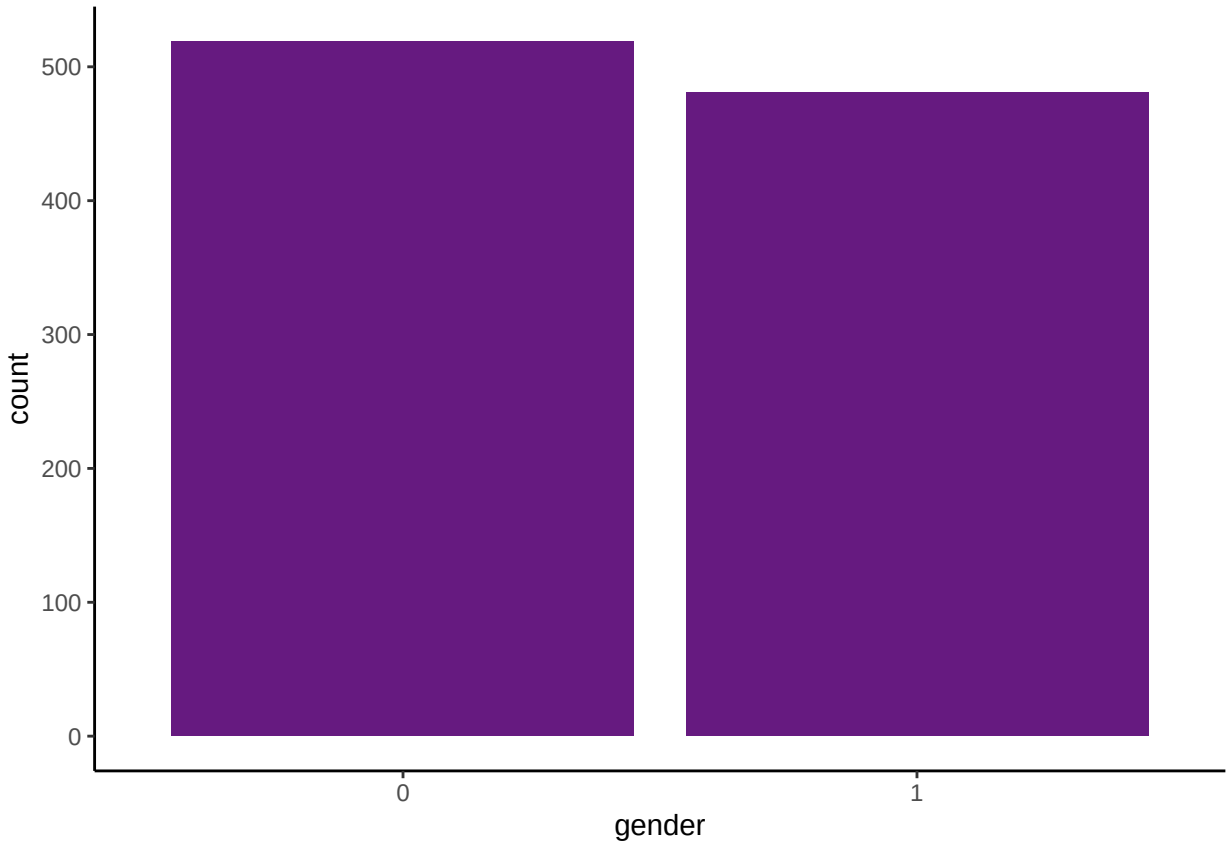
1.The Average Hours spent by users on the Internet is 180 minutes while the median is 183.1 and the mode is 167.22.

2.The Interquartile Range is 79.9625.

3.The data is negatively but fairly skewed with a skew value of -0.03343681 and the data is platykurtic with a value of 1.727701.

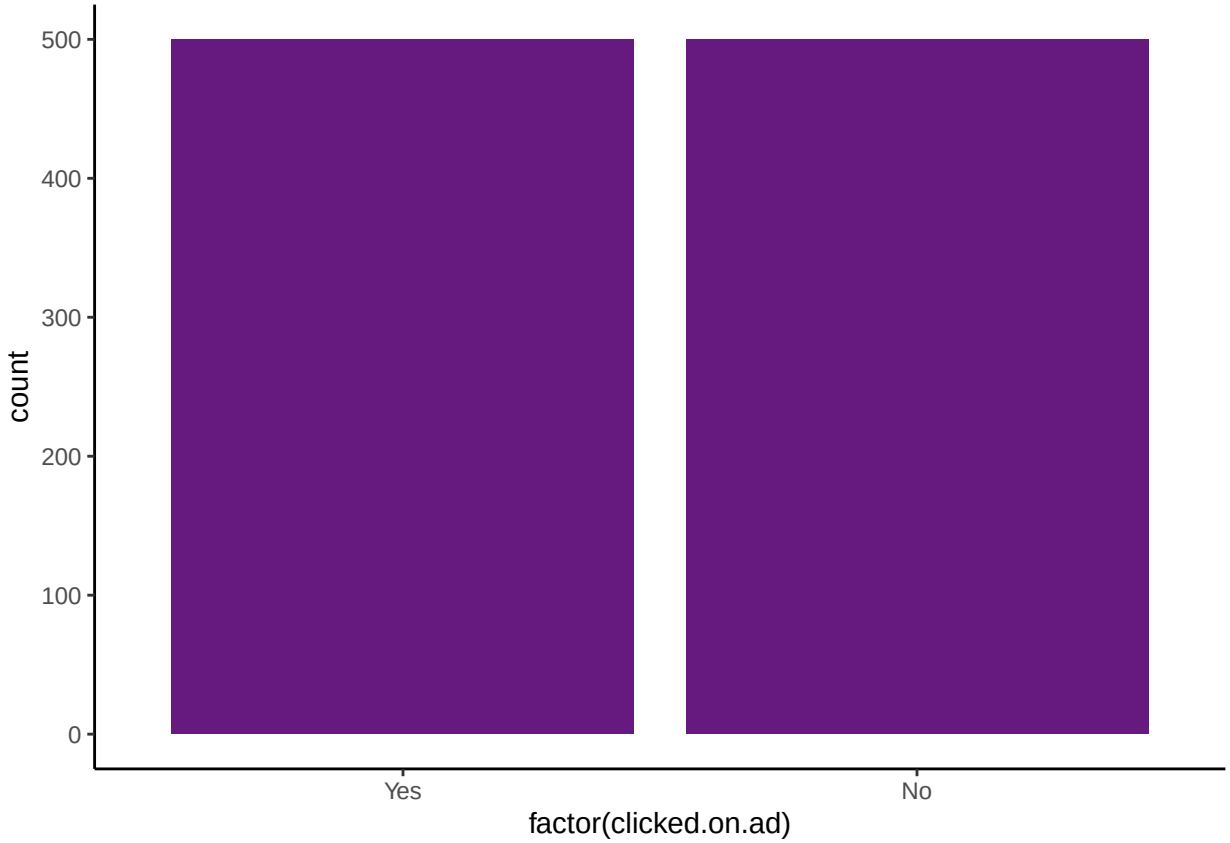
Countplots for the Categorical variables_

```
##we plot the countplot for the variable gender
#---
#
ggplot(final_df, aes(x=gender)) + geom_bar(fill=rgb(0.4,0.1,0.5))
```




```
# 0 represents female and 1 male
#From the plotted countplots the number of female was slightly higher than that of male.
```

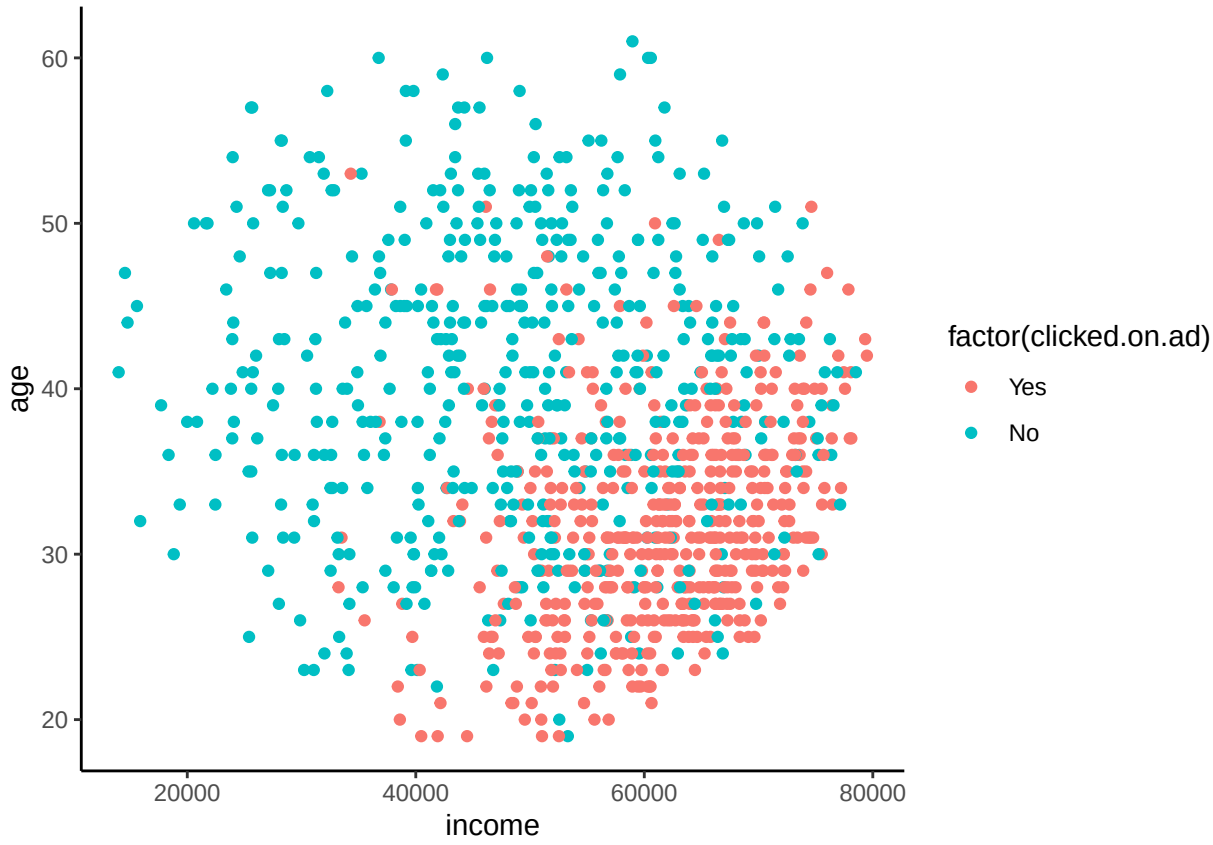
```
##we plot the countplot for the target variable i.e clicked.on.ad
#---
#
ggplot(final_df, aes(x=factor(`clicked.on.ad`))) + geom_bar( fill=rgb(0.4,0.1,0.5))
```



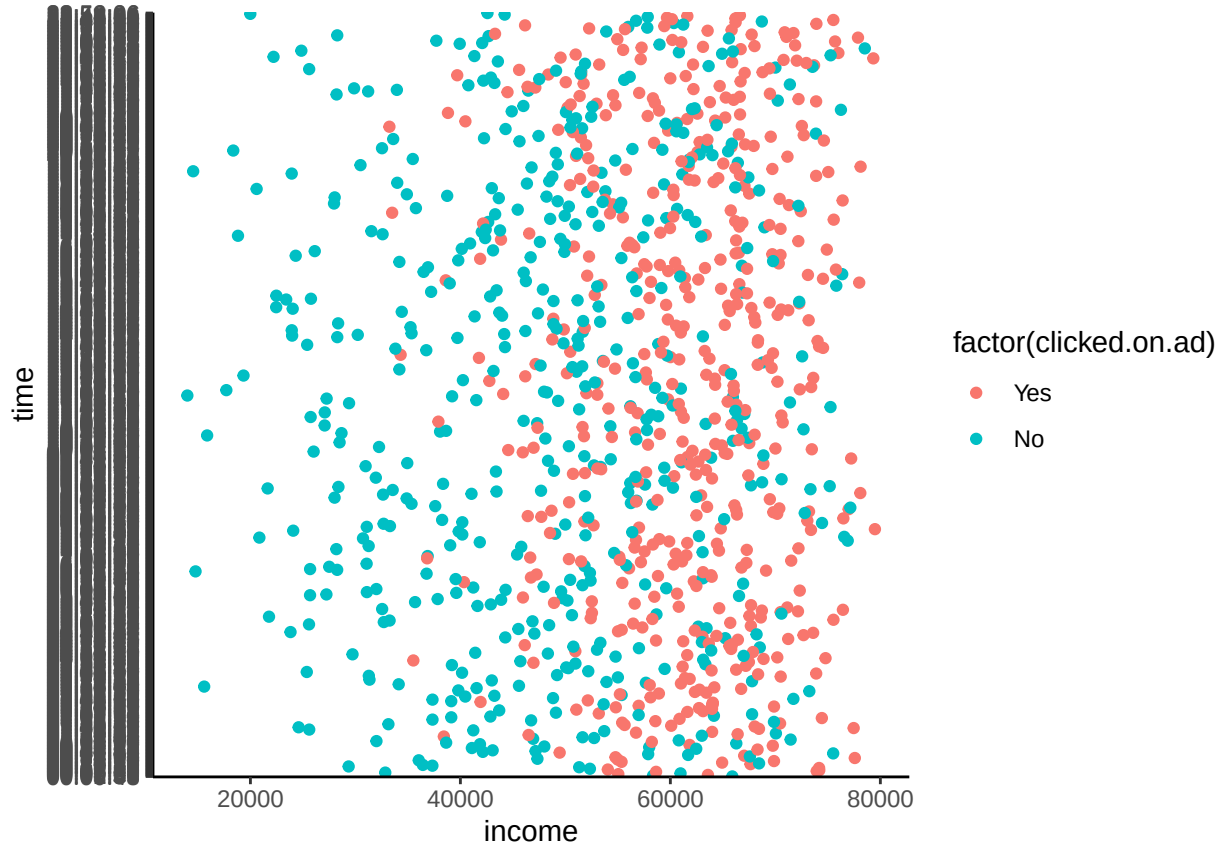

```
#We observed that the number of users on the site who clicked on the ad is equal to those that did #not
```

5.2 Bivariate Analysis

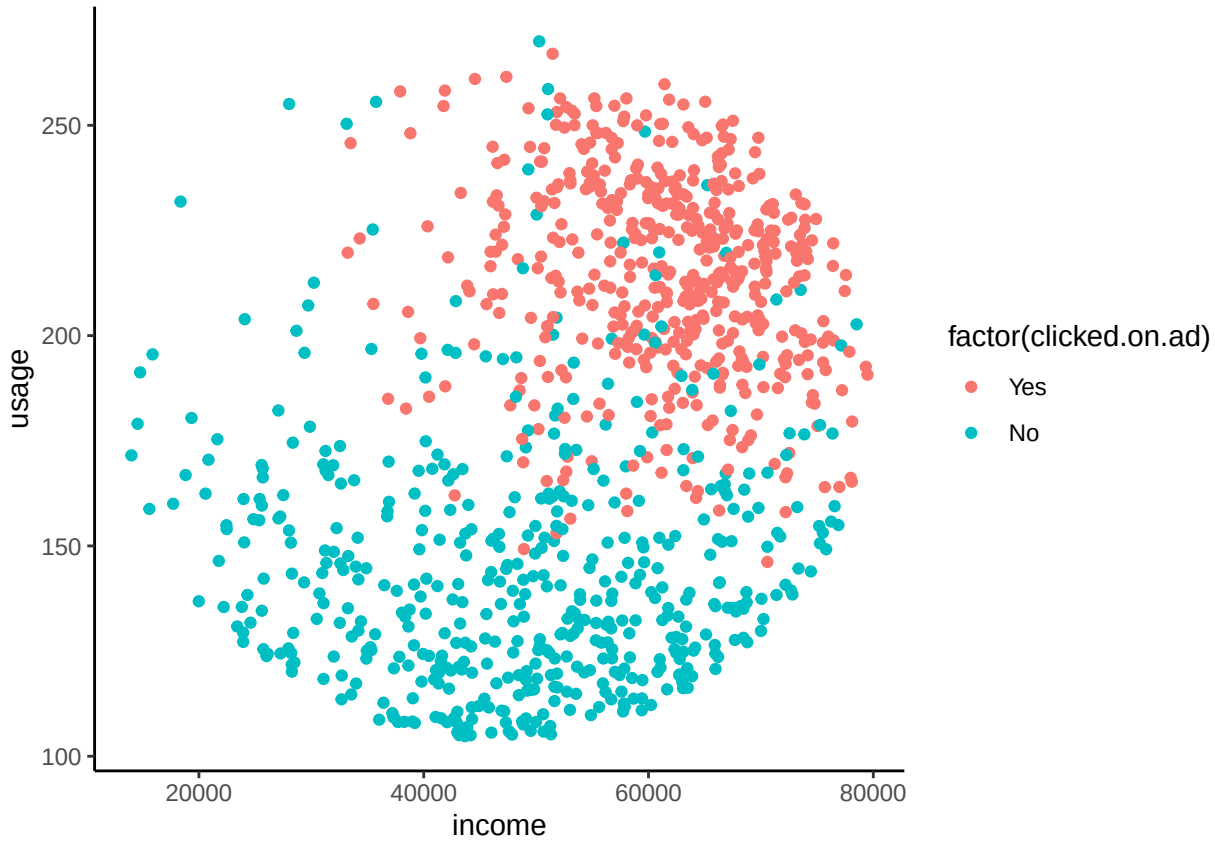
```
##Scatter plot for income and age
#---
#

ggplot(final_df, aes(x=income , y = age )) + geom_point(aes(colour= factor(`clicked.on.ad`)))
```



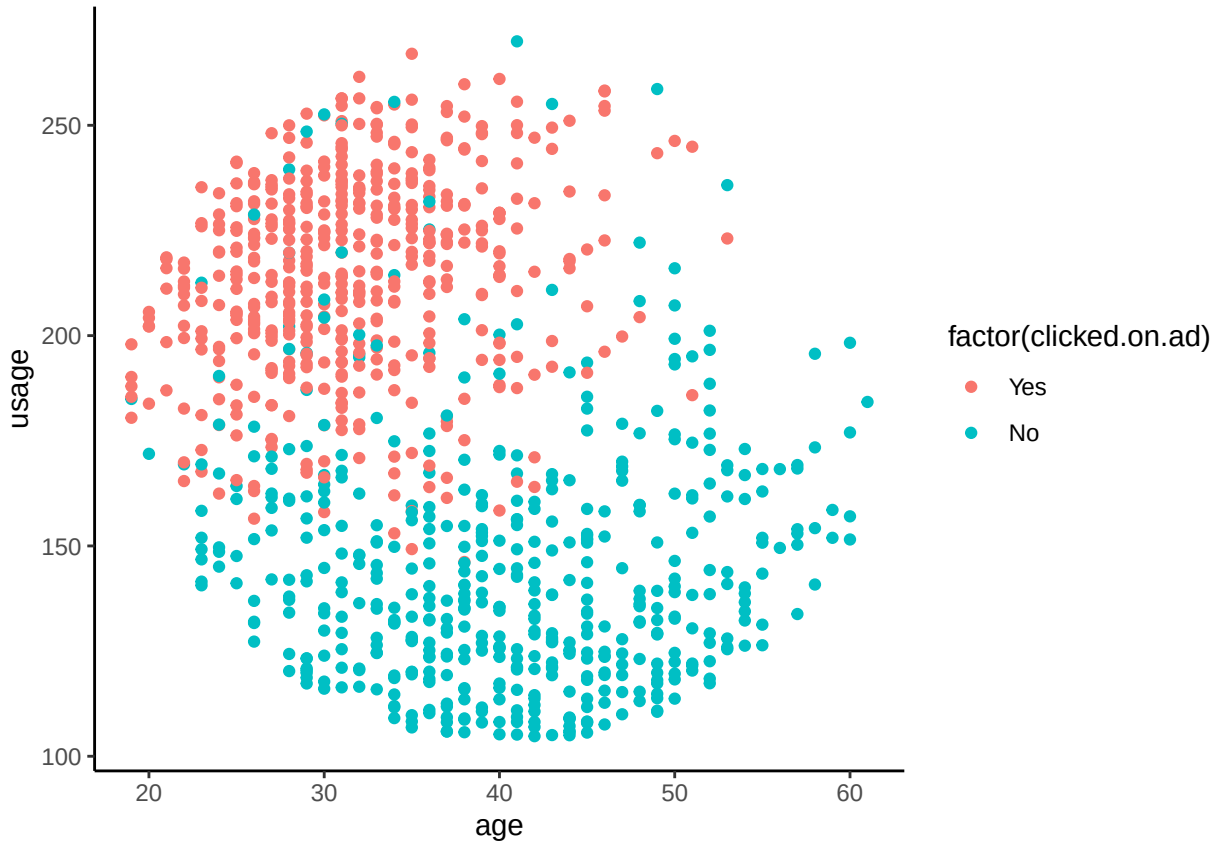

```
##Scatter plot for income and time
#---
#
ggplot(final_df, aes(x=income , y = time )) + geom_point(aes(colour= factor(`clicked.on.ad`)))
```



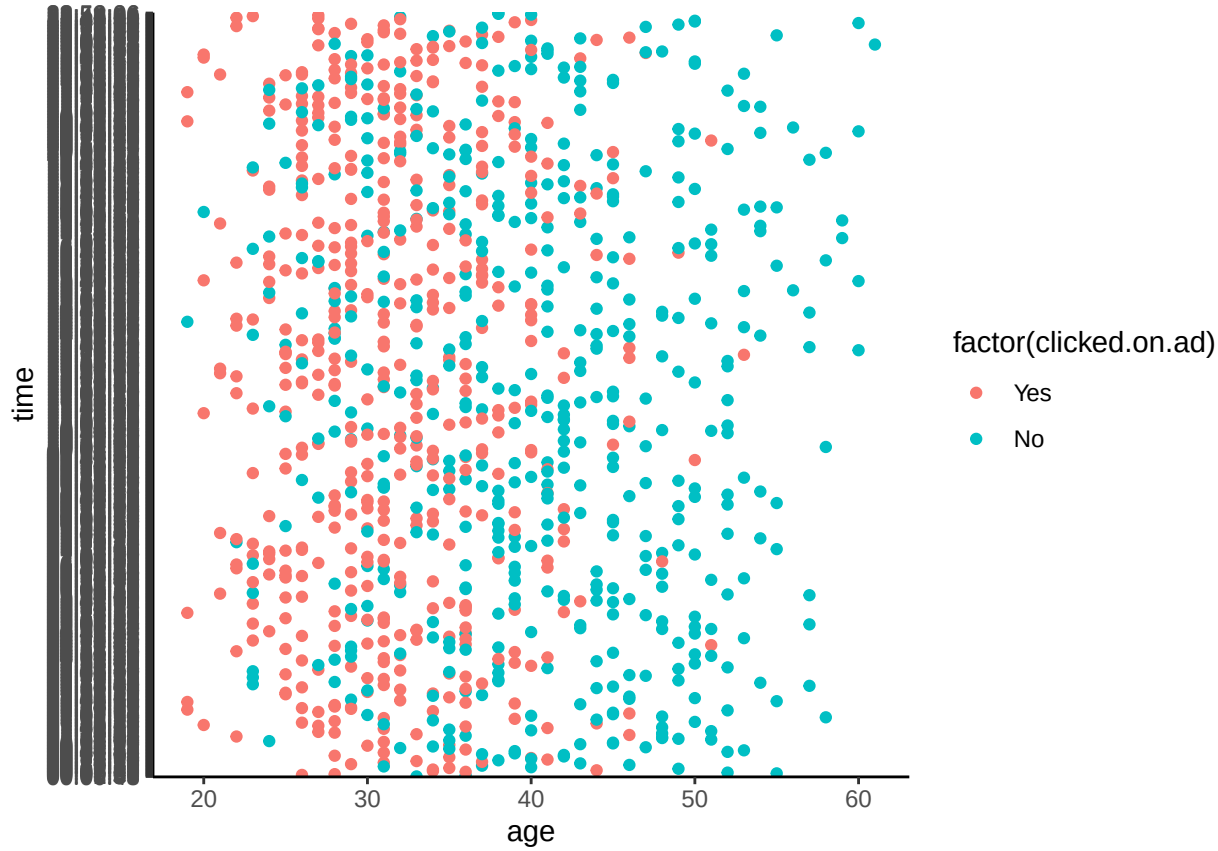

```
##Scatter plot for income and usage
#---
#
ggplot(final_df, aes(x=income , y = usage )) + geom_point(aes(colour= factor(`clicked.on.ad`)))
```



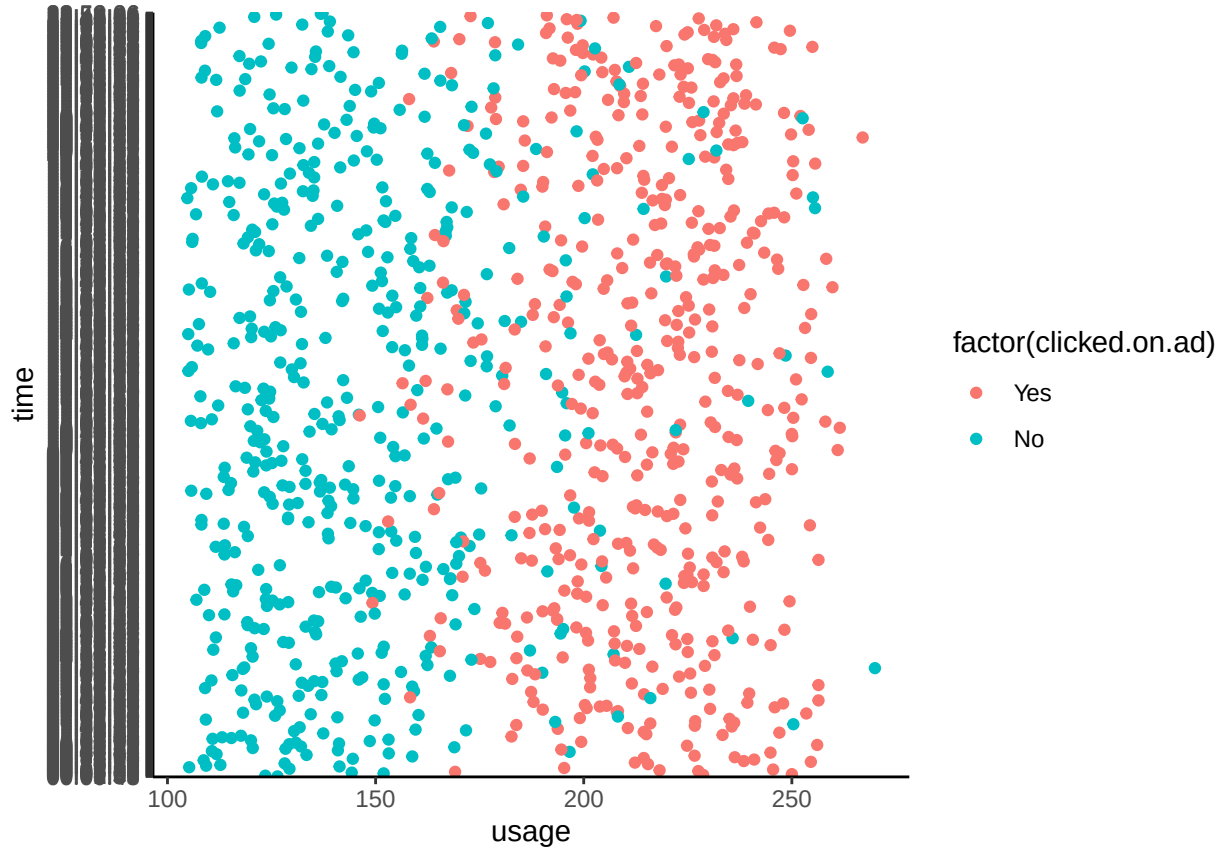

```
##scatter plot for age and usage
#---
#
ggplot(final_df, aes(x=age , y = usage )) + geom_point(aes(colour= factor(`clicked.on.ad`)))
```




```
##scatter plot for age and time
#---
#
ggplot(final_df, aes(x=age , y = time )) + geom_point(aes(colour= factor(`clicked.on.ad`)))
```




```
##scatter plot for usage against time
#---
#
ggplot(final_df, aes(x=usage , y = time )) + geom_point(aes(colour= factor(`clicked.on.ad`)))
```



Conclusion

From the above scatter plot we deduce the following:

1. The scatter plot for income against age shows that majority of the people who failed to click the ads were high income earners with age between 20 and 40 years.

2.The scatter plot for internet usage against income shows that individuals who spend over 200 minutes online are less likely to click an ad.

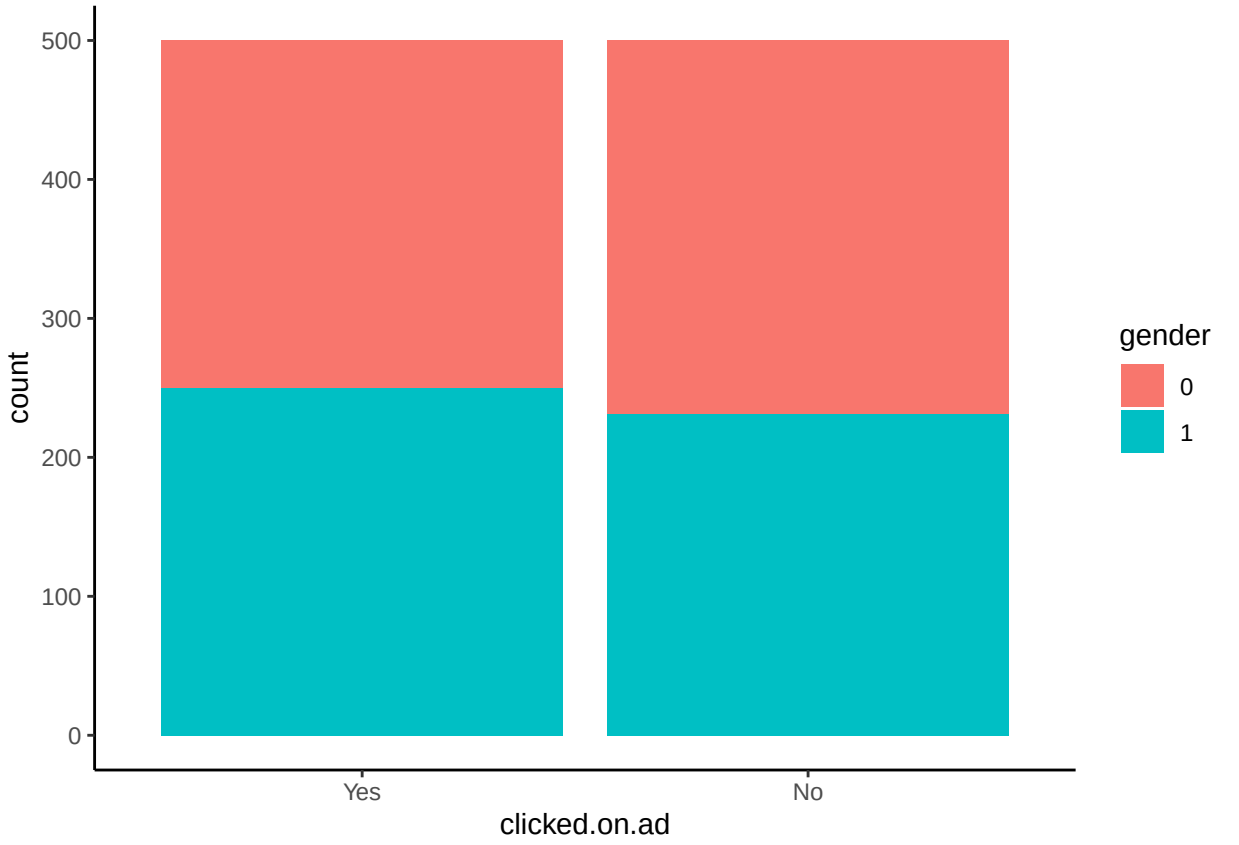
3. The scatter plot for Age against time shows that younger individuals spend alot of tme online and are less tolerant to ads as compared to individuals with age above 35 years.

4. Finally, we observed from the scatter plot for time against usage that the more time one spends online the more the usage.

We continue to Explore which individuals are more likely to click an ad using other visualizations other than scatter plot

```
##Who is likely to click on an ad, female or male?
#---
#
library(ggplot2)

## we use stacked bar chart
#---
```

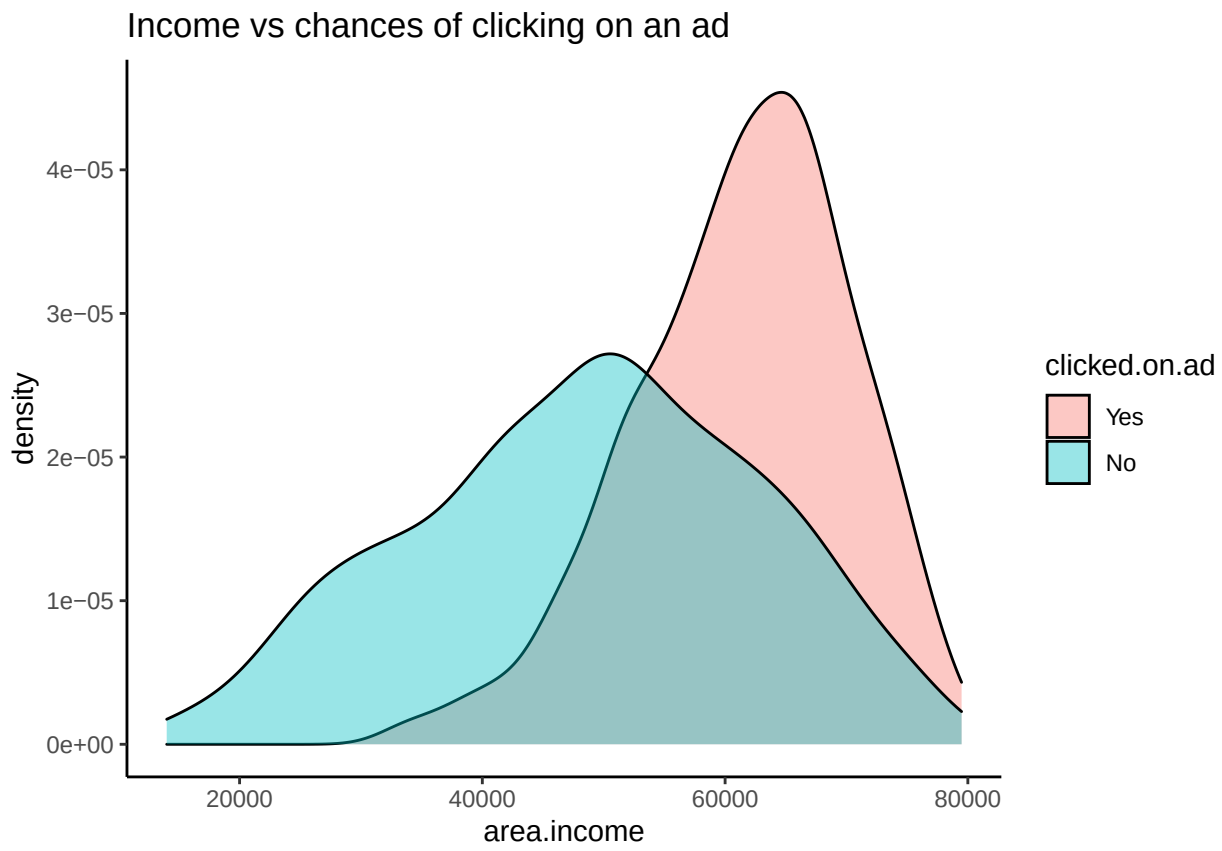
```
#
ggplot(final_df, aes(x = clicked.on.ad,  fill = gender)) + geom_bar(position = "stack")
```



Conclusion

From the above stacked bar chart we observed that Female (where male=0) are more likely to click an ad as compared to male individuals.

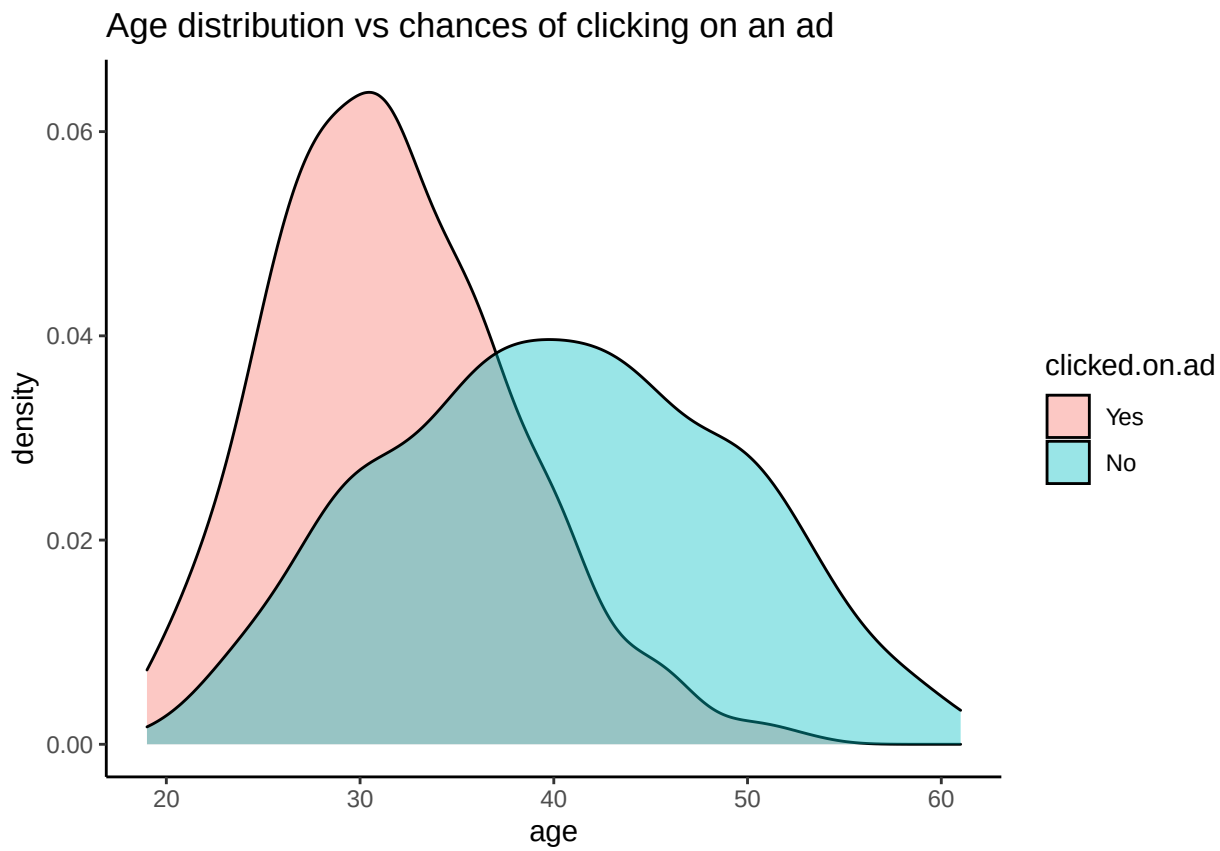
```
##Income class and it's relationship to clicking an ad
#---
#
ggplot(final_df, aes(x = area.income,  fill = clicked.on.ad)) +geom_density(alpha = 0.4) +
  labs(title = "Income vs chances of clicking on an ad")
```



Conclusion

We observed that the income range for people who click on an ad is large as compared to those individuals who failed to click the ad . People from all ranges of income are likely to click on an ad but most of the group with an income of above 50000 are less likely to click on an ad.

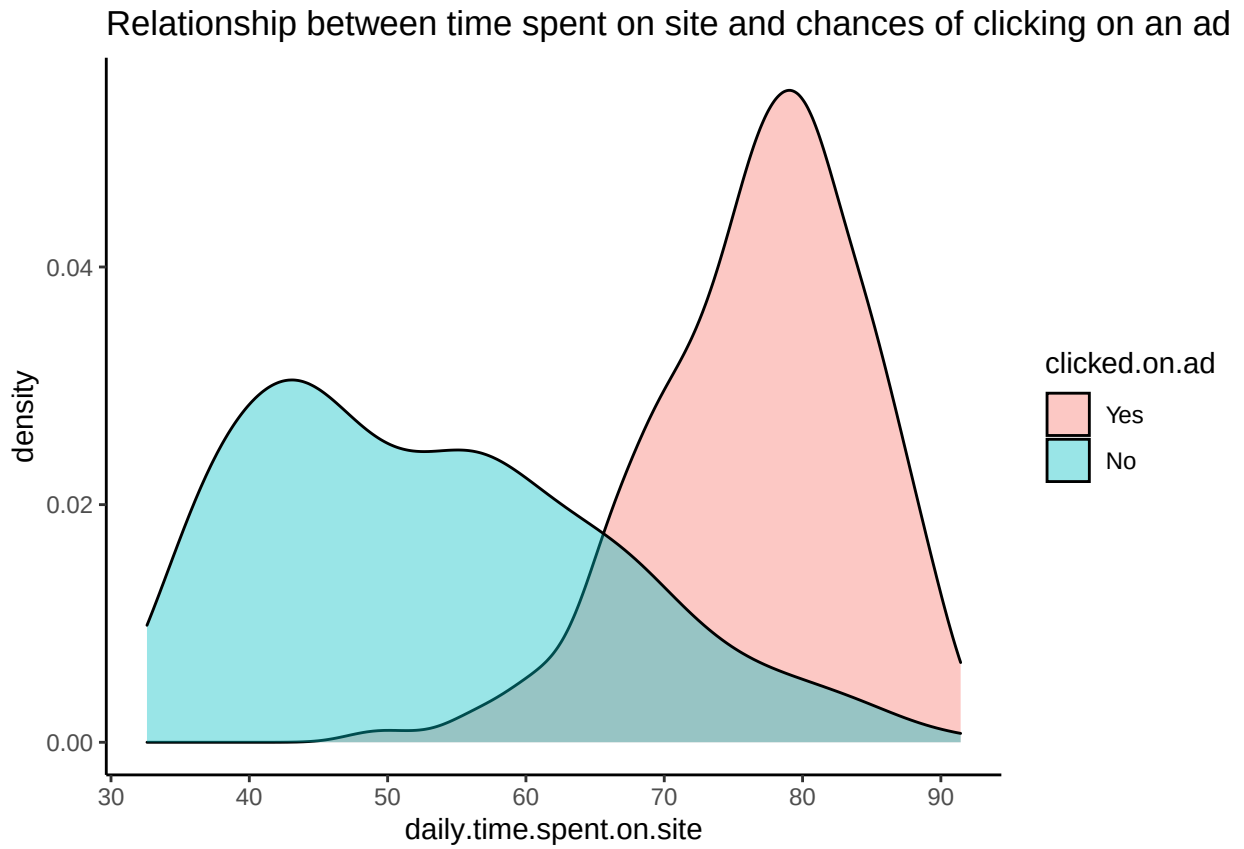
```
##Age and it's relationship to clicking an ad
#---
#
ggplot(final_df, aes(x = age,  fill = clicked.on.ad)) +geom_density(alpha = 0.4) +
  labs(title = "Age distribution vs chances of clicking on an ad")
```



Conclusion

We observed that People above age 35 are more likely to click the ad.

```
## Time spent on site and it's relationship to clicking an ad
ggplot(final_df, aes(x = daily.time.spent.on.site,  fill = clicked.on.ad)) +
  geom_density(alpha = 0.4) +
  labs(title = "Relationship between time spent on site and chances of clicking on an ad")
```



Conclusion

We observed that individuals who spend more time online are less likely to click an add as compared to those who spend less time.

We plot correlation matrix for numerical variables

```
##Selecting the Numerical Variables of the dataset
#---
#
corr <- select(ad , age , 'area.income', `clicked.on.ad`,
               `daily.internet.usage`, `daily.time.spent.on.site`, gender )
```

1.6 Implementing the solution

```
##Splitting the data into training and testing sets
#---
#
set.seed(100)

## Selecting only columns that are relevant to modeling
#---
#
mod_cols = c('daily.time.spent.on.site', 'age', 'area.income', 'daily.internet.usage', 'gender', 'click
final_df = select(final_df, all_of(mod_cols))

## Splitting the data into 80% training and 20% testing
```

```
#---
#
train_rows = createDataPartition(final_df$clicked.on.ad, p=0.8, list=FALSE)

## Creating the training data set
#---
#
train = final_df[train_rows,]

## Creating the test dataset
#---
#
test = final_df[-train_rows,]

##Creating the  X and Y variables
#---
#
x = train
y = train$clicked.on.ad
```

```
install.packages("e1071")
```

```
## Installing package into 'C:/Users/Denoo/OneDrive/Documents/R/win-library/4.1'
## (as 'lib' is unspecified)

## package 'e1071' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\Denoo\AppData\Local\Temp\RtmpmyJSfi\downloaded_packages
```

```
## we train the model
#---
#
model = train(clicked.on.ad ~ ., data = train, method = 'earth')
```

```
## Loading required package: earth

## Loading required package: Formula

## Loading required package: plotmo

## Loading required package: plotrix

## Loading required package: TeachingDemos

##
## Attaching package: 'TeachingDemos'

## The following object is masked _by_ '.GlobalEnv':
##
##     outliers
```

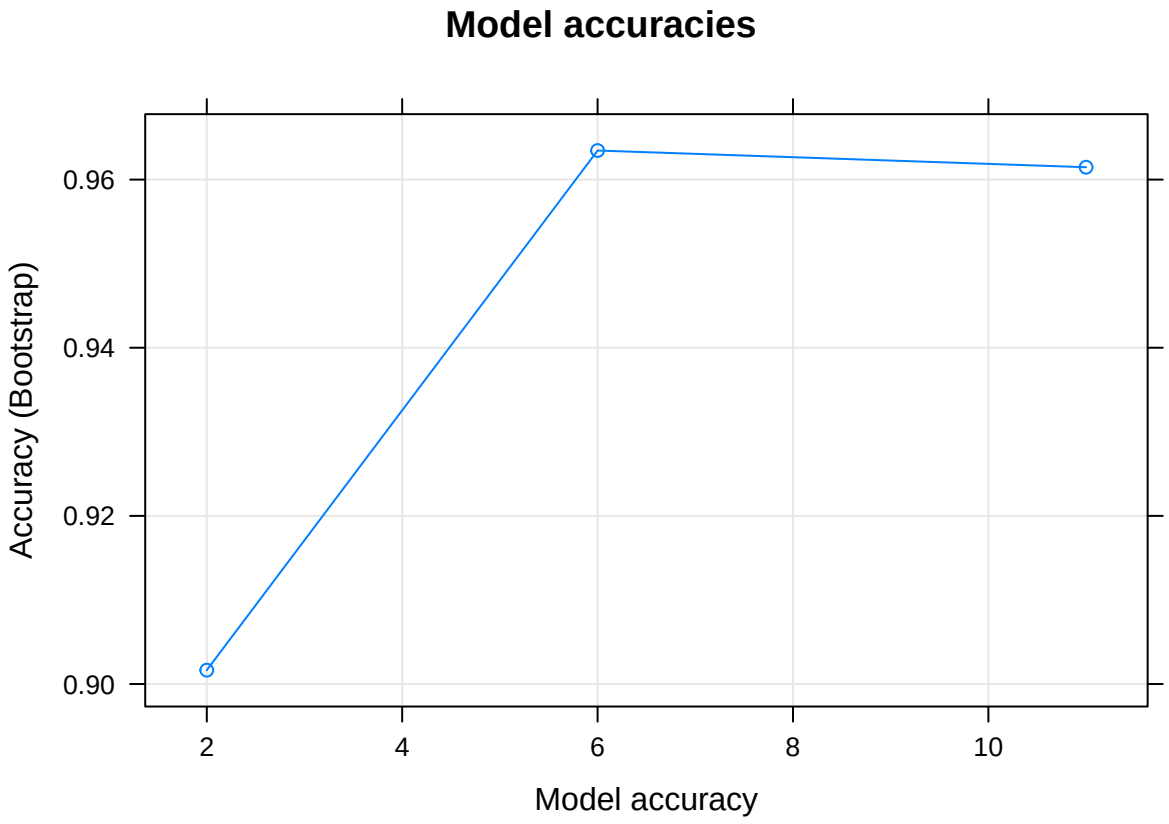
```
## The following object is masked from 'package:plotly':
##
##     subplot
```

```
## Making predictions using the training set
#---
#
pred = predict(model)
```

```
## Displaying the parameters and their values in the model
#---
#
model
```

```
## Multivariate Adaptive Regression Spline
##
## 800 samples
##   5 predictor
##   2 classes: 'Yes', 'No'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 800, 800, 800, 800, 800, 800, ...
## Resampling results across tuning parameters:
##
##   nprune  Accuracy   Kappa
##    2      0.9016503  0.8031409
##    6      0.9634582  0.9268097
##   11      0.9614716  0.9228648
##
## Tuning parameter 'degree' 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 nprune = 6 and degree = 1.
```

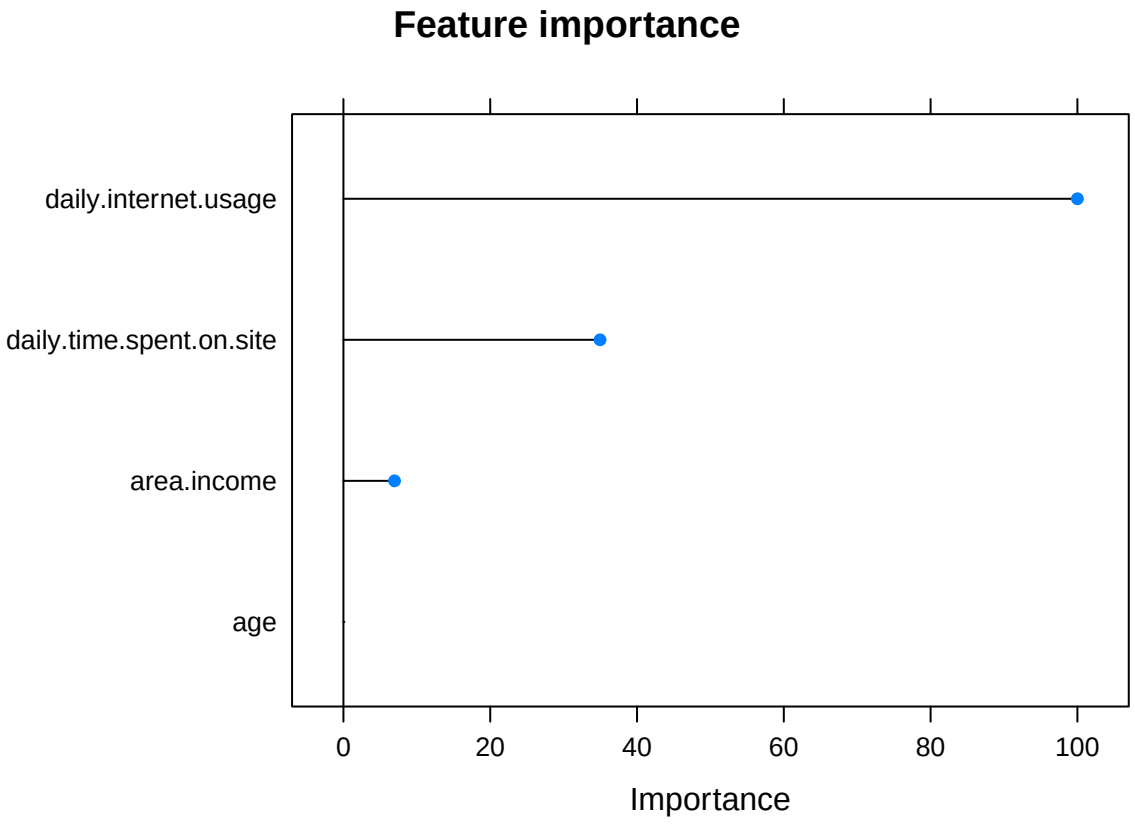
```
## we Plot the model to show various iterations of the hyperparameters
#---
#
plot(model, main = 'Model accuracies', xlab = 'Model accuracy')
```




```
## we Check features which are important in predicting the target variable
#---
#
install.packages("varImp")
```

```
## Installing package into 'C:/Users/Denoo/OneDrive/Documents/R/win-library/4.1'
## (as 'lib' is unspecified)

## package 'varImp' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\Denoo\AppData\Local\Temp\RtmpmyJSfi\downloaded_packages
```

```
important_features = varImp(model)

##we Plot feature importance
#---
#
plot(important_features, main = 'Feature importance')
```



Conclusion

Features that are important in predicting the target in rank of importance are the following:

1.Daily internet use 2.Daily time spent on site 3.Area income 4.Age

```
##Making predictions
#---
# Previewing the first five predictions
y_pred = predict(model, test)
head(y_pred)
```

```
## [1] Yes Yes No  Yes No  No
## Levels: Yes No
```

```
##we plot the confusion matrix
#---
#
confusionMatrix(reference = test$clicked.on.ad, data = y_pred, mode='everything', positive = 'Yes')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Yes No
##        Yes  98  3
##        No    2 97
```

```
##
##                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 : 1
##
##             Sensitivity : 0.9800
##             Specificity : 0.9700
##          Pos Pred Value : 0.9703
##          Neg Pred Value : 0.9798
##               Precision : 0.9703
##                  Recall : 0.9800
##                      F1 : 0.9751
##              Prevalence : 0.5000
##          Detection Rate : 0.4900
##    Detection Prevalence : 0.5050
##       Balanced Accuracy : 0.9750
##
##        'Positive' Class : Yes
##
```

```
##setting the cross validation parameters
#---
# Defining the training control

fitControl <- trainControl(
    method = 'cv',
    number = 5,
    savePredictions = 'final',
    classProbs = T,
    summaryFunction=twoClassSummary
)

##we tune the hyperparameters by setting up tuneLength
#---
#
set.seed(100)
model_2 = train(clicked.on.ad ~ ., data=train, method='earth', tuneLength = 5, metric='accuracy', trCon
model_2
```

```
## Multivariate Adaptive Regression Spline
##
## 800 samples
##   5 predictor
##   2 classes: 'Yes', 'No'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 640, 640, 640, 640, 640
## Resampling results across tuning parameters:
```

```
##
##   nprune  ROC        Sens    Spec
##    2      0.9539062  0.9100  0.8900
##    4      0.9787969  0.9675  0.9300
##    6      0.9869375  0.9750  0.9500
##    8      0.9873125  0.9650  0.9525
##   11      0.9872188  0.9700  0.9575
##
## Tuning parameter 'degree' was held constant at a value of 1
## ROC was used to select the optimal model using the largest value.
## The final values used for the model were nprune = 8 and degree = 1.
```

```
##we Predict the test data and compute the confusion matrix
#---
#
y_pred_2 <- predict(model_2, test)
confusionMatrix(reference = test$clicked.on.ad, data = y_pred_2, mode='everything', positive='Yes')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Yes No
##        Yes  98  3
##        No    2 97
##
##                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 : 1
##
##             Sensitivity : 0.9800
##             Specificity : 0.9700
##          Pos Pred Value : 0.9703
##          Neg Pred Value : 0.9798
##               Precision : 0.9703
##                  Recall : 0.9800
##                      F1 : 0.9751
##              Prevalence : 0.5000
##          Detection Rate : 0.4900
##    Detection Prevalence : 0.5050
##       Balanced Accuracy : 0.9750
##
##        'Positive' Class : Yes
##
```

```
##Hyper parameter tuning using tuneGrid¶
#---
#
```

```r
##we Define the parameters to tune
#---
#
params = expand.grid(nprune = c(2, 4, 6, 8, 10),
                        degree = c(1, 2, 3))

##Tuning hyper parameters by setting tune Grid
#---
#
set.seed(100)
model_3 = train(clicked.on.ad ~ ., data=train, method='earth', metric='accuracy', tuneGrid = params, tr
model_3
```

```
## Multivariate Adaptive Regression Spline
##
## 800 samples
##   5 predictor
##   2 classes: 'Yes', 'No'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 640, 640, 640, 640, 640
## Resampling results across tuning parameters:
##
##   degree  nprune  ROC        Sens    Spec
##   1        2      0.9539062  0.9100  0.8900
##   1        4      0.9787969  0.9675  0.9300
##   1        6      0.9869375  0.9750  0.9500
##   1        8      0.9873125  0.9650  0.9525
##   1       10      0.9870937  0.9675  0.9500
##   2        2      0.9535938  0.9125  0.8900
##   2        4      0.9843594  0.9600  0.9400
##   2        6      0.9856875  0.9625  0.9425
##   2        8      0.9860156  0.9725  0.9500
##   2       10      0.9862344  0.9700  0.9525
##   3        2      0.9535938  0.9125  0.8900
##   3        4      0.9856250  0.9700  0.9500
##   3        6      0.9876406  0.9700  0.9450
##   3        8      0.9880625  0.9600  0.9525
##   3       10      0.9861250  0.9725  0.9525
##
## ROC was used to select the optimal model using the largest value.
## The final values used for the model were nprune = 8 and degree = 3.
```

```r
##Predicting the test set and computing the confusion matrix
#---
#
y_pred_3 = predict(model_3, test)
confusionMatrix(reference = test$clicked.on.ad, data = y_pred_3, mode='everything', positive='Yes')
```

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

```
## Prediction Yes No
##        Yes  96  3
##        No    4 97
##
##                Accuracy : 0.965
##                  95% CI : (0.9292, 0.9858)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.93
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9600
##             Specificity : 0.9700
##          Pos Pred Value : 0.9697
##          Neg Pred Value : 0.9604
##               Precision : 0.9697
##                  Recall : 0.9600
##                      F1 : 0.9648
##              Prevalence : 0.5000
##          Detection Rate : 0.4800
##    Detection Prevalence : 0.4950
##       Balanced Accuracy : 0.9650
##
##        'Positive' Class : Yes
##
```

Conclusion

The model score is 96.5%

1.7 Challenging the solution

In our Analysis we used the following models to challenge the solution:

1.7.1 Ada boost

```
set.seed(100)

##we train the model using adaboost
model_adaboost = train(clicked.on.ad ~ ., data=train, method='adaboost', tuneLength=2, trControl = fitC
model_adaboost
```

```
## AdaBoost Classification Trees
##
## 800 samples
##   5 predictor
##   2 classes: 'Yes', 'No'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 640, 640, 640, 640, 640
## Resampling results across tuning parameters:
##
```

```
##   nIter  method         ROC        Sens    Spec
##    50    Adaboost.M1    0.9836875  0.9675  0.9450
##    50    Real adaboost  0.8600156  0.9725  0.9450
##   100    Adaboost.M1    0.9846563  0.9700  0.9475
##   100    Real adaboost  0.8523594  0.9800  0.9475
##
## ROC was used to select the optimal model using the largest value.
## The final values used for the model were nIter = 100 and method = Adaboost.M1.
```

1.7.2 Random Forest

```
set.seed(100)

# Train the model using rf
model_rf = train(clicked.on.ad ~ ., data=train, method='rf', tuneLength=5, trControl = fitControl)
```

```
## note: only 4 unique complexity parameters in default grid. Truncating the grid to 4 .
```

```
model_rf
```

```
## Random Forest
##
## 800 samples
##   5 predictor
##   2 classes: 'Yes', 'No'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 640, 640, 640, 640, 640
## Resampling results across tuning parameters:
##
##   mtry  ROC        Sens    Spec
##   2     0.9894063  0.9675  0.9550
##   3     0.9876563  0.9675  0.9550
##   4     0.9865625  0.9625  0.9500
##   5     0.9845625  0.9500  0.9525
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

1.7.3 Xgboost

```
set.seed(100)

##we train the model using xgboost
model_xgbDART = train(clicked.on.ad ~ ., data=train, method='xgbDART', tuneLength=1, trControl = fitCon
model_xgbDART
```

```
## eXtreme Gradient Boosting
##
## 800 samples
```

```
##   5 predictor
##   2 classes: 'Yes', 'No'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 640, 640, 640, 640, 640
## Resampling results across tuning parameters:
##
##   eta  rate_drop  skip_drop  colsample_bytree  ROC        Sens    Spec
##   0.3  0.01       0.05       0.6               0.9869062  0.9575  0.9500
##   0.3  0.01       0.05       0.8               0.9896406  0.9700  0.9525
##   0.3  0.01       0.95       0.6               0.9889688  0.9650  0.9500
##   0.3  0.01       0.95       0.8               0.9886563  0.9675  0.9450
##   0.3  0.50       0.05       0.6               0.9801563  0.9250  0.9300
##   0.3  0.50       0.05       0.8               0.9772344  0.9250  0.9175
##   0.3  0.50       0.95       0.6               0.9887500  0.9650  0.9500
##   0.3  0.50       0.95       0.8               0.9887656  0.9600  0.9550
##   0.4  0.01       0.05       0.6               0.9893750  0.9550  0.9575
##   0.4  0.01       0.05       0.8               0.9884844  0.9675  0.9500
##   0.4  0.01       0.95       0.6               0.9879375  0.9625  0.9525
##   0.4  0.01       0.95       0.8               0.9875312  0.9600  0.9550
##   0.4  0.50       0.05       0.6               0.9791094  0.9325  0.9125
##   0.4  0.50       0.05       0.8               0.9756719  0.9325  0.9225
##   0.4  0.50       0.95       0.6               0.9878750  0.9650  0.9550
##   0.4  0.50       0.95       0.8               0.9877031  0.9575  0.9475
##
## Tuning parameter 'nrounds' was held constant at a value of 50
## Tuning
##  parameter 'subsample' was held constant at a value of 0.5
## Tuning
##  parameter 'min_child_weight' was held constant at a value of 1
## ROC was used to select the optimal model using the largest value.
## The final values used for the model were nrounds = 50, max_depth = 1, eta
##  = 0.3, gamma = 0, subsample = 0.5, colsample_bytree = 0.8, rate_drop =
##  0.01, skip_drop = 0.05 and min_child_weight = 1.
```

1.7.4 Support Vector Machine (SVM)

```
set.seed(100)

# Train the model using support vector machine
model_svmRadial = train(clicked.on.ad ~ ., data=train, method='svmRadial', tuneLength=1, trControl = fi
model_svmRadial
```

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 800 samples
##   5 predictor
##   2 classes: 'Yes', 'No'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 640, 640, 640, 640, 640
```

```
## Resampling results:
##
##   ROC        Sens    Spec
##   0.9909062  0.9775  0.9575
##
## Tuning parameter 'sigma' was held constant at a value of 0.2158025
##
## Tuning parameter 'C' was held constant at a value of 0.25
```

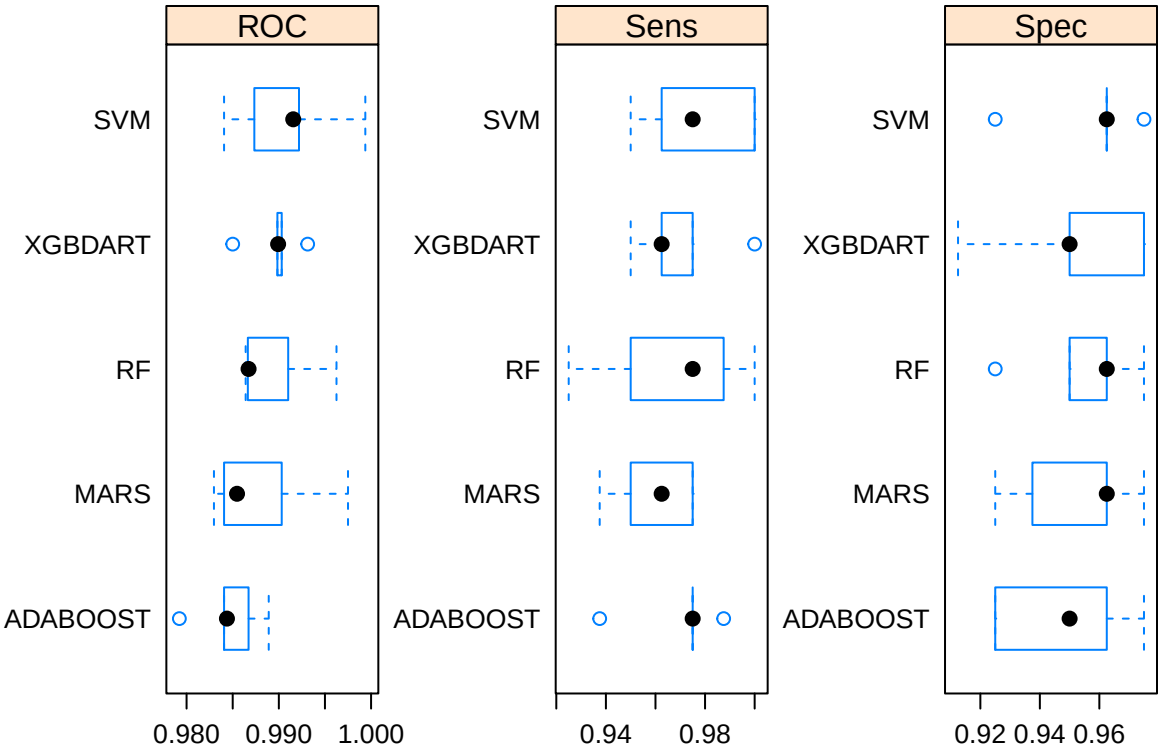
1.7.5 Model Comparison

```
##we compare the model performances
#---
#
models_compare = resamples(list(ADABOOST=model_adaboost, RF=model_rf, XGBDART=model_xgbDART, MARS=model_

## Summary of the models performances
#---
#
summary(models_compare)
```

```
##
## Call:
## summary.resamples(object = models_compare)
##
## Models: ADABOOST, RF, XGBDART, MARS, SVM
## Number of resamples: 5
##
## ROC
##                Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## ADABOOST 0.9792188 0.9840625 0.9843750 0.9846563 0.9867187 0.9889062    0
## RF       0.9864063 0.9866406 0.9867187 0.9894063 0.9910156 0.9962500    0
## XGBDART  0.9850000 0.9898438 0.9899219 0.9896406 0.9903125 0.9931250    0
## MARS     0.9829688 0.9840625 0.9854688 0.9880625 0.9903125 0.9975000    0
## SVM      0.9840625 0.9873438 0.9915625 0.9909062 0.9921875 0.9993750    0
##
## Sens
##            Min. 1st Qu. Median   Mean 3rd Qu.   Max. NA's
## ADABOOST 0.9375  0.9750 0.9750 0.9700  0.9750 0.9875    0
## RF       0.9250  0.9500 0.9750 0.9675  0.9875 1.0000    0
## XGBDART  0.9500  0.9625 0.9625 0.9700  0.9750 1.0000    0
## MARS     0.9375  0.9500 0.9625 0.9600  0.9750 0.9750    0
## SVM      0.9500  0.9625 0.9750 0.9775  1.0000 1.0000    0
##
## Spec
##            Min. 1st Qu. Median   Mean 3rd Qu.  Max. NA's
## ADABOOST 0.9250  0.9250 0.9500 0.9475  0.9625 0.975    0
## RF       0.9250  0.9500 0.9625 0.9550  0.9625 0.975    0
## XGBDART  0.9125  0.9500 0.9500 0.9525  0.9750 0.975    0
## MARS     0.9250  0.9375 0.9625 0.9525  0.9625 0.975    0
## SVM      0.9250  0.9625 0.9625 0.9575  0.9625 0.975    0
```

1.7.6 Visualize Model comparison

```
# Draw box plots to compare models
scales = list(x=list(relation="free"), y=list(relation="free"))
bwplot(models_compare, scales=scales)
```



Conclusion

Support Vector Machine (SVM) performs better than the other models.

1.8 Recommendations

Other than ads the Kenyan entrepreneur should Come up with a better way of advertising the online cryptography course to high income earners and also to those individuals who spend alot of time online.