Blog_Churn

Snow

9/2/2021

numerical_df DF

Define the question

I am a data science for a blogger. The question is making conclusion on who is likely to click on the ads in the blog and derive insights. Build a model that can predict if a person will click an ad or not based on the features in the dataframe ## Metric for success

In order to work on the above problem, you need to do the following:

- Define the question- the metric for success, the context,experimental design taken and the appropriateness of the available data to answer the given question.
- Find and deal with outliers, anomalies, and missing data within the dataset.
- Perform univariate and bivariate analysis.
- From your insights provide a conclusion and recommendation.
- Build a model using classification using decision trees and Support Vector Machine
- Get an accuracy => 80%

Data Understanding (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.

In order to work on the above problem, you need to do the following:

- Define the question, the metric for success, the context, experimental design taken and the appropriateness of the available data to answer the given question.
- Find and deal with outliers, anomalies, and missing data within the dataset.
- Perform univariate and bivariate analysis.
- From your insights provide a conclusion and recommendation.
- Build models and get the metrics

Experimental design

1. Import the data to R
2. Perform data exploration
3. Define metrics for success
4. Perform Uni variate and Bivariate data Analysis
5. Provide conclusion

Loading the current working directory

```
ad_df <- read.csv('advertising.csv', header = TRUE, sep = ',')
```

Data Exploration

```
# Standardize column names with standard naming convention ie lowercase and replace spaces with '_'

# replace the spaces with underscores using gsub() function
names(ad_df) <- gsub(" ","_", names(ad_df))

# lowercase
names(ad_df) <- tolower(names(ad_df))

# display the column names to confirm the changes
colnames(ad_df)
```

```
## [1] "daily.time.spent.on.site" "age"
## [3] "area.income"              "daily.internet.usage"
## [5] "ad.topic.line"            "city"
## [7] "male"                     "country"
## [9] "timestamp"                "clicked.on.ad"
```

```
# Preview dataset
head(ad_df)
```

```
##   daily.time.spent.on.site age area.income daily.internet.usage
## 1                    68.95  35    61833.90               256.09
## 2                    80.23  31    68441.85               193.77
## 3                    69.47  26    59785.94               236.50
## 4                    74.15  29    54806.18               245.89
## 5                    68.37  35    73889.99               225.58
## 6                    59.99  23    59761.56               226.74
##                          ad.topic.line           city male    country
## 1    Cloned 5thgeneration orchestration    Wrightburgh    0    Tunisia
## 2    Monitored national standardization      West Jodi    1      Nauru
## 3      Organic bottom-line service-desk       Davidton    0 San Marino
## 4 Triple-buffered reciprocal time-frame West Terrifurt    1      Italy
## 5         Robust logistical utilization   South Manuel    0    Iceland
## 6       Sharable client-driven software      Jamieberg    1     Norway
```

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

```
# Finding the Shape of the dataset
dim(ad_df)
```

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

```
# Finding the datatypes of the data
str(ad_df)
```

```
## 'data.frame':    1000 obs. of  10 variables:
##  $ daily.time.spent.on.site: num  69 80.2 69.5 74.2 68.4 ...
##  $ age                     : int  35 31 26 29 35 23 33 48 30 20 ...
##  $ area.income             : num  61834 68442 59786 54806 73890 ...
##  $ daily.internet.usage    : num  256 194 236 246 226 ...
##  $ ad.topic.line           : chr  "Cloned 5thgeneration orchestration" "Monitored national standardi
##  $ city                    : chr  "Wrightburgh" "West Jodi" "Davidton" "West Terrifurt" ...
##  $ male                    : int  0 1 0 1 0 1 0 1 1 1 ...
##  $ country                 : chr  "Tunisia" "Nauru" "San Marino" "Italy" ...
##  $ timestamp               : chr  "2016-03-27 00:53:11" "2016-04-04 01:39:02" "2016-03-13 20:35:42"
##  $ clicked.on.ad           : int  0 0 0 0 0 0 0 1 0 0 ...
```

Data cleaning

```
# checking for missing Data
colSums(is.na(ad_df))
```

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

There is no missing values in the dataset.

```
# Check for duplicated data in the ad_Df
ad_df1 <- ad_df[duplicated(ad_df),]
ad_df1
```

```
##  [1] daily.time.spent.on.site age                      area.income
```

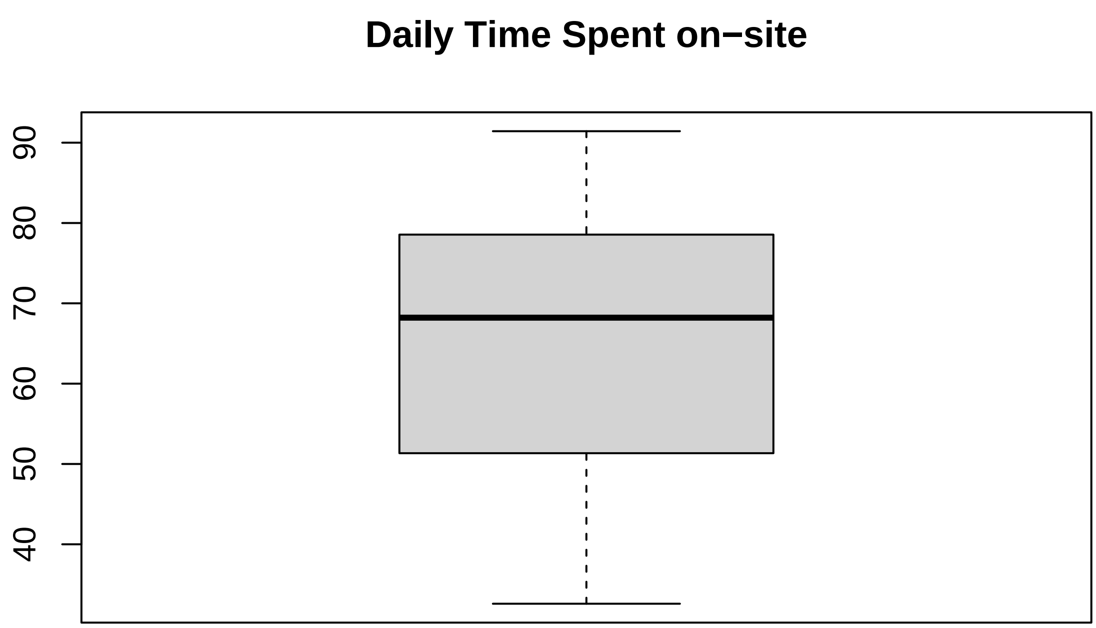
```
##  [4] daily.internet.usage     ad.topic.line            city
##  [7] male                     country                  timestamp
## [10] clicked.on.ad
## <0 rows> (or 0-length row.names)
```

There are no duplicated records in the dataset

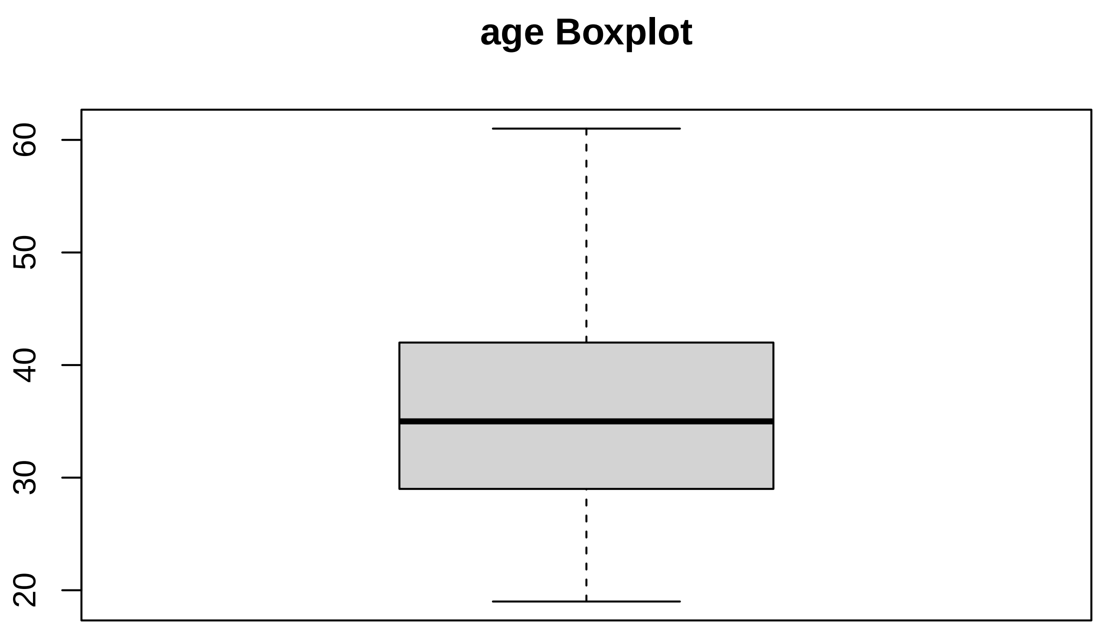
```r
str(ad_df)
```

```
## 'data.frame':    1000 obs. of  10 variables:
##  $ daily.time.spent.on.site: num  69 80.2 69.5 74.2 68.4 ...
##  $ age                     : int  35 31 26 29 35 23 33 48 30 20 ...
##  $ area.income             : num  61834 68442 59786 54806 73890 ...
##  $ daily.internet.usage    : num  256 194 236 246 226 ...
##  $ ad.topic.line           : chr  "Cloned 5thgeneration orchestration" "Monitored national standardi
##  $ city                    : chr  "Wrightburgh" "West Jodi" "Davidton" "West Terrifurt" ...
##  $ male                    : int  0 1 0 1 0 1 0 1 1 1 ...
##  $ country                 : chr  "Tunisia" "Nauru" "San Marino" "Italy" ...
##  $ timestamp               : chr  "2016-03-27 00:53:11" "2016-04-04 01:39:02" "2016-03-13 20:35:42"
##  $ clicked.on.ad           : int  0 0 0 0 0 0 0 1 0 0 ...
```

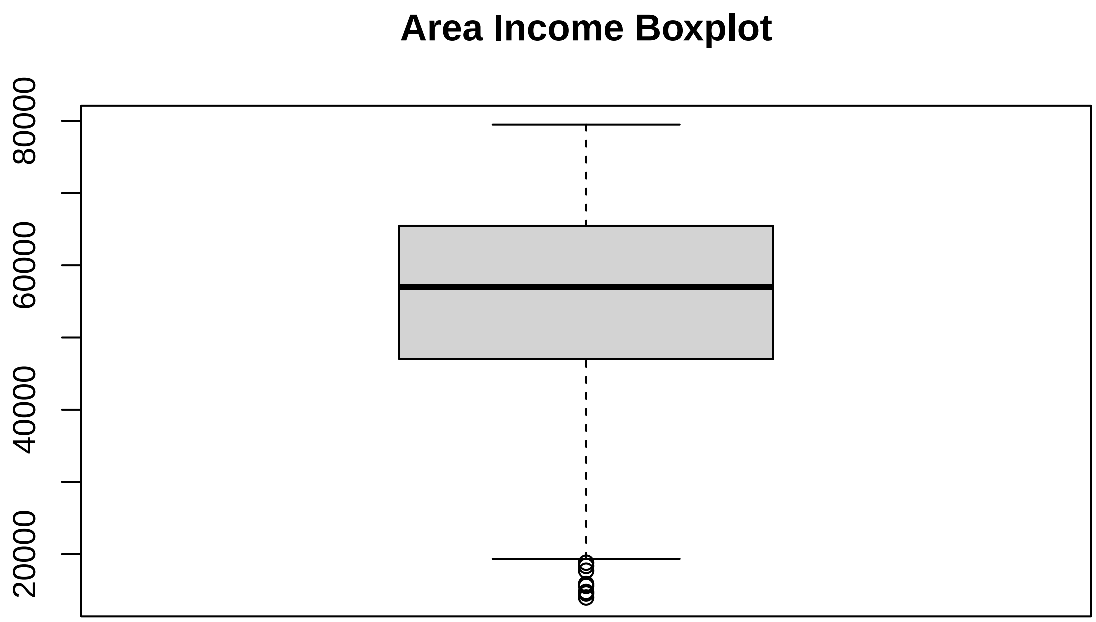
```r
boxplot(ad_df$daily.time.spent.on.site, main = 'Daily Time Spent on-site')
```



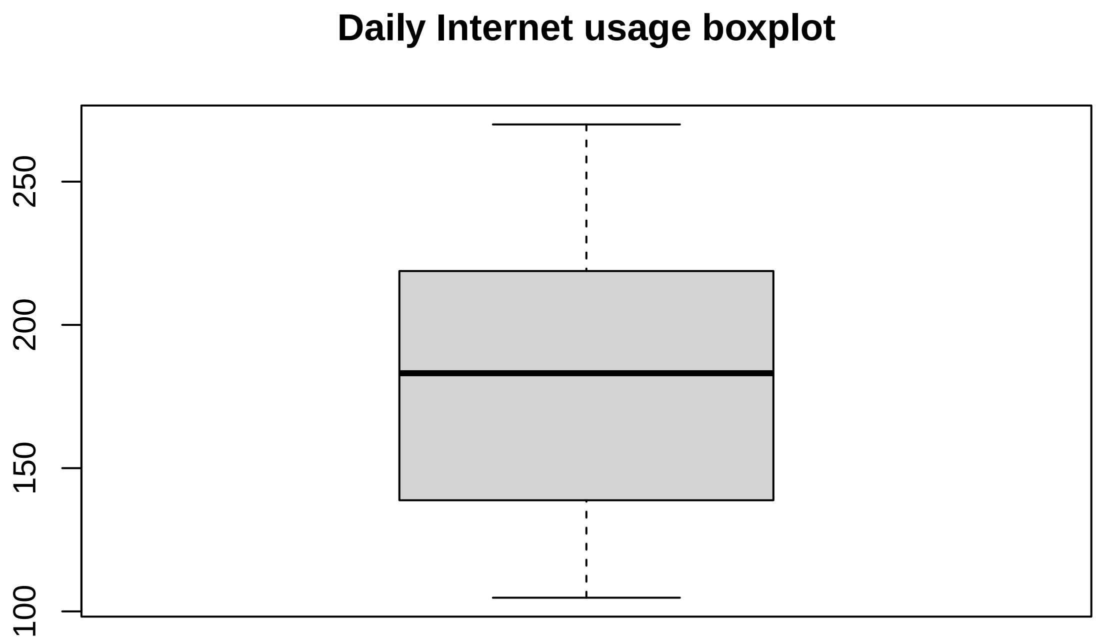
```r
boxplot(ad_df$age, main = 'age Boxplot')
```




```r
boxplot(ad_df$area.income, main = 'Area Income Boxplot')
```

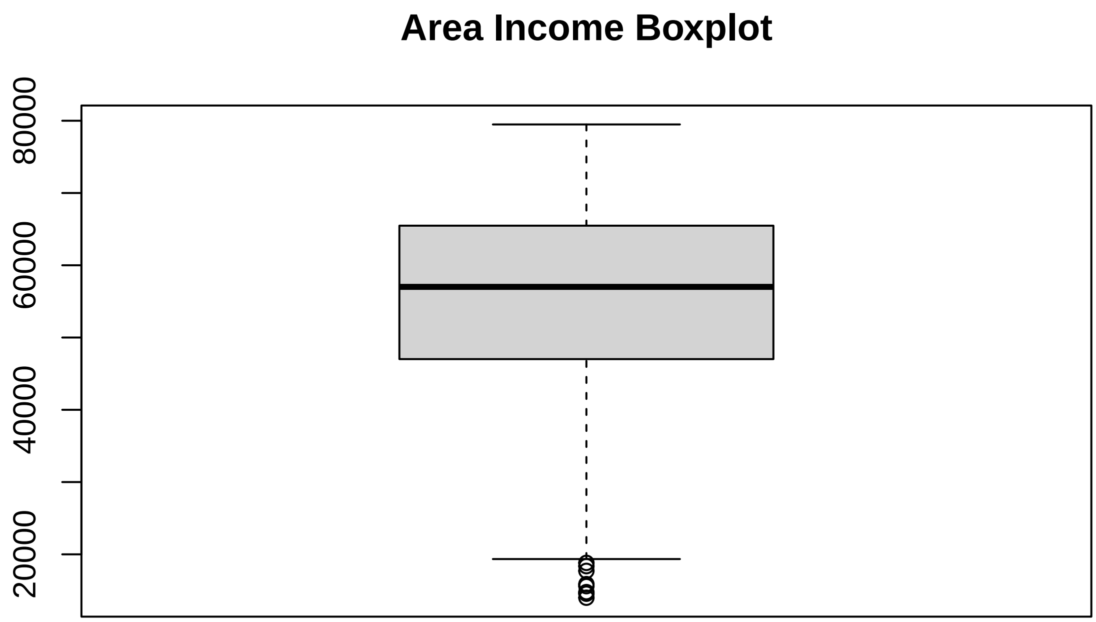



```
boxplot(ad_df$daily.internet.usage, main = 'Daily Internet usage boxplot')
```



From the boxplots, only the Area_income column has outliers.

```
#Print out the outliers
boxplot(ad_df$area.income, main = 'Area Income Boxplot')$out
```




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

There are outliers that do not look like they are in the extreme. There are areas where poverty is prevelant in such areas the total income could be that small.

```r
str (ad_df)
```

```
## 'data.frame':    1000 obs. of  10 variables:
##  $ daily.time.spent.on.site: num  69 80.2 69.5 74.2 68.4 ...
##  $ age                     : int  35 31 26 29 35 23 33 48 30 20 ...
##  $ area.income             : num  61834 68442 59786 54806 73890 ...
##  $ daily.internet.usage    : num  256 194 236 246 226 ...
##  $ ad.topic.line           : chr  "Cloned 5thgeneration orchestration" "Monitored national standardi
##  $ city                    : chr  "Wrightburgh" "West Jodi" "Davidton" "West Terrifurt" ...
##  $ male                    : int  0 1 0 1 0 1 0 1 1 1 ...
##  $ country                 : chr  "Tunisia" "Nauru" "San Marino" "Italy" ...
##  $ timestamp               : chr  "2016-03-27 00:53:11" "2016-04-04 01:39:02" "2016-03-13 20:35:42"
##  $ clicked.on.ad           : int  0 0 0 0 0 0 0 1 0 0 ...
```

```r
ad_df[['timestamp']] <- as.POSIXct(ad_df[['timestamp']],
                                   format = "%Y-%m-%d %H:%M:%S")
str(ad_df)
```

```
## 'data.frame':    1000 obs. of  10 variables:
```

```
##  $ daily.time.spent.on.site: num  69 80.2 69.5 74.2 68.4 ...
##  $ age                     : int  35 31 26 29 35 23 33 48 30 20 ...
##  $ area.income             : num  61834 68442 59786 54806 73890 ...
##  $ daily.internet.usage    : num  256 194 236 246 226 ...
##  $ ad.topic.line           : chr  "Cloned 5thgeneration orchestration" "Monitored national standardi
##  $ city                    : chr  "Wrightburgh" "West Jodi" "Davidton" "West Terrifurt" ...
##  $ male                    : int  0 1 0 1 0 1 0 1 1 1 ...
##  $ country                 : chr  "Tunisia" "Nauru" "San Marino" "Italy" ...
##  $ timestamp               : POSIXct, format: "2016-03-27 00:53:11" "2016-04-04 01:39:02" ...
##  $ clicked.on.ad           : int  0 0 0 0 0 0 0 1 0 0 ...
```

The timestamp column is now in the correct dtype

Univariate Data Analysis

###Numerical Columns

```r
summary(ad_df)
```

```
##  daily.time.spent.on.site      age         area.income    daily.internet.usage
##  Min.   :32.60            Min.   :19.00   Min.   :13996   Min.   :104.8
##  1st Qu.:51.36            1st Qu.:29.00   1st Qu.:47032   1st Qu.:138.8
##  Median :68.22            Median :35.00   Median :57012   Median :183.1
##  Mean   :65.00            Mean   :36.01   Mean   :55000   Mean   :180.0
##  3rd Qu.:78.55            3rd Qu.:42.00   3rd Qu.:65471   3rd Qu.:218.8
##  Max.   :91.43            Max.   :61.00   Max.   :79485   Max.   :270.0
##  ad.topic.line          city                 male          country
##  Length:1000        Length:1000        Min.   :0.000   Length:1000
##  Class :character   Class :character   1st Qu.:0.000   Class :character
##  Mode  :character   Mode  :character   Median :0.000   Mode  :character
##                                        Mean   :0.481
##                                        3rd Qu.:1.000
##                                        Max.   :1.000
##    timestamp                      clicked.on.ad
##  Min.   :2016-01-01 02:52:10   Min.   :0.0
##  1st Qu.:2016-02-18 02:55:42   1st Qu.:0.0
##  Median :2016-04-07 17:27:29   Median :0.5
##  Mean   :2016-04-10 10:56:04   Mean   :0.5
##  3rd Qu.:2016-05-31 03:18:14   3rd Qu.:1.0
##  Max.   :2016-07-24 00:22:16   Max.   :1.0
```

```r
# Mean
mean.age <- mean(ad_df$age)
mean.age
```

age

```
## [1] 36.009
```

```r
#median
median.age <- median (ad_df$age)
median.age
```

```
## [1] 35
```

```r
# Function to get the mode.
getmode <- function(v) {
  uniqv <- unique(v)
  uniqv[which.max(tabulate(match(v, uniqv)))]
}
```

```r
mode.age <- getmode(ad_df$age)
mode.age
```

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

####Area income

```r
mean.areaincome <- mean(ad_df$area.income)
mean.areaincome
```

```
## [1] 55000
```

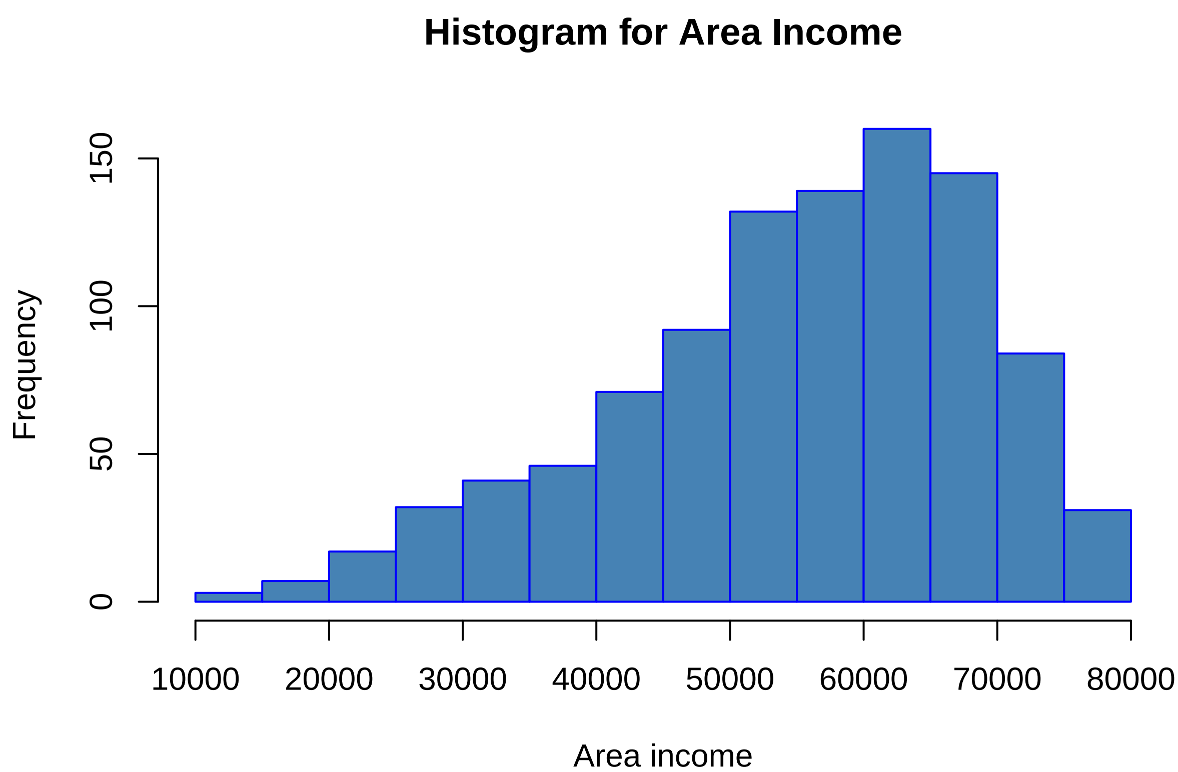
```r
median.areaincome <- median(ad_df$area.income)
median.areaincome
```

```
## [1] 57012.3
```

```r
mode.areaincome <- getmode(ad_df$area.income)
mode.areaincome
```

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

```r
hist(ad_df$area.income,
      main="Histogram for Area Income",
     xlab="Area income",
     border="blue",
     col="steelblue",)
```




```
mean.daily.internet <- mean(ad_df$daily.internet.usage)
mean.daily.internet
```

daily.internet.usage

```
## [1] 180.0001
```

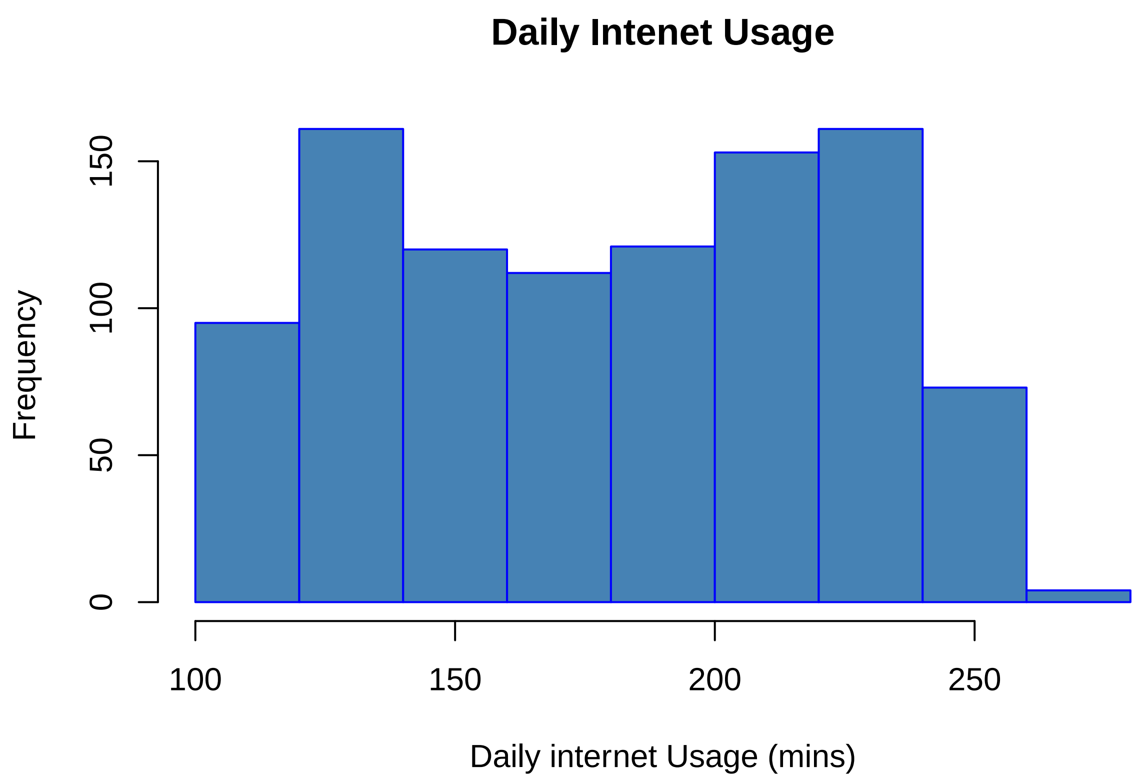
```
median.daily.internet <- median(ad_df$daily.internet.usage)
median.daily.internet
```

```
## [1] 183.13
```

```
mode.daily.internet <- getmode(ad_df$daily.internet.usage)
mode.daily.internet
```

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

```
hist(ad_df$daily.internet.usage,
     main = 'Daily Intenet Usage',
      xlab="Daily internet Usage (mins)",
     border="blue",
     col="steelblue")
```




```
mean.dtsos <- mean(ad_df$daily.time.spent.on.site)
mean.dtsos
```

Daily time spent on site

```
## [1] 65.0002
```

```
median.dtsos <- median(ad_df$daily.time.spent.on.site)
median.dtsos
```

```
## [1] 68.215
```

```
mode.dtsos <- getmode(ad_df$daily.time.spent.on.site)
mode.dtsos
```

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

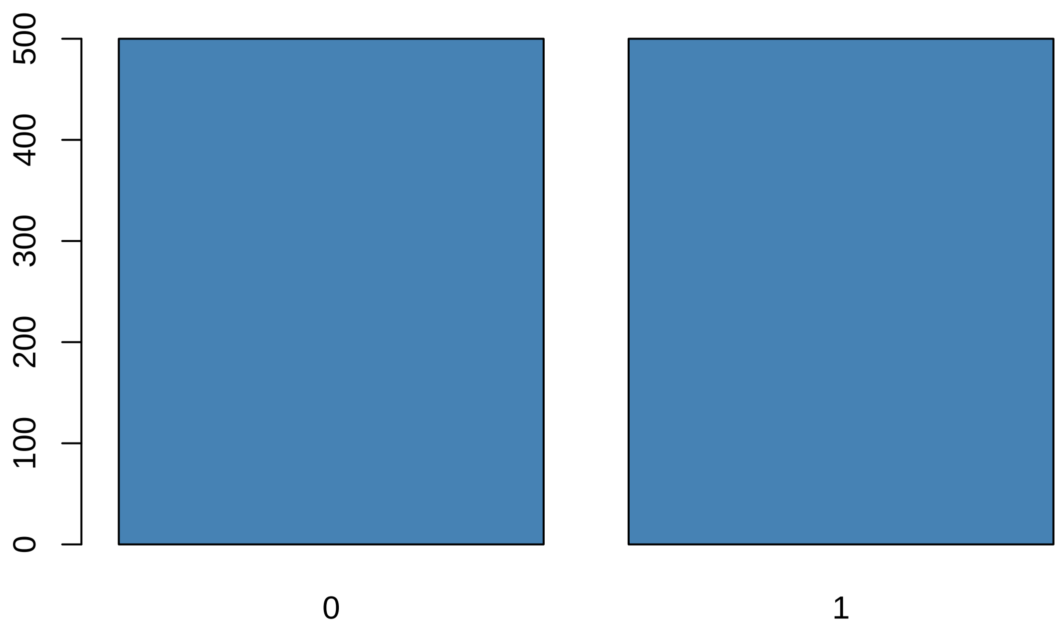
```
uniq_clickers <- unique(ad_df$clicked.on.ad, )
length(uniq_clickers)
```

clicked.on.ad

```
## [1] 2
```

There are two categories of the people who clicked on ads Let us plot the frequency of each

```
clickers <- ad_df$clicked.on.ad
clickers_frequency <- table (clickers)
barplot(clickers_frequency, col = "steelblue")
```



There are 500 people who clicked on ads and another 500 did not click on the ads.

Categorical Columns

####ad.topic.line

```
uniq_topic <- unique(ad_df$ad.topic.line, )
length(uniq_topic)
```

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

There are 1000 unique topic lines meaning it would be impossible to get a good visualization.

```
uniq_city <- unique(ad_df$city, )
length(uniq_city)
```

city

```
## [1] 969
```

There are 969 unique cities hence it would also be impossible to get a good visualization

```
uniq_country <- unique(ad_df$country)
length(uniq_country)
```

country

```
## [1] 237
```

There are 237 unique countries.

```
library(sf)
```

```
## Linking to GEOS 3.9.0, GDAL 3.2.1, PROJ 7.2.1
```

```
library(raster)
```

```
## Loading required package: sp
```

```
library(dplyr)
```

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

```
## The following objects are masked from 'package:raster':
##
##     intersect, select, union
```

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

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

```
library(spData)
```

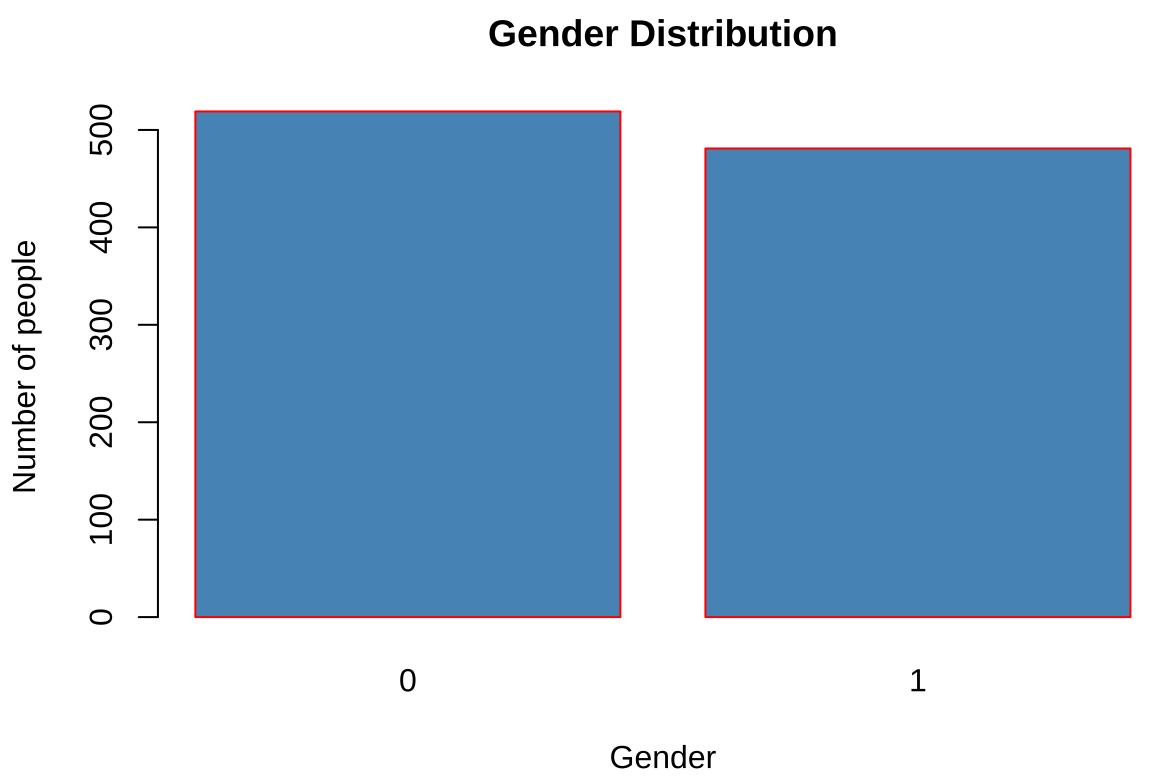
```
## To access larger datasets in this package, install the spDataLarge
## package with: `install.packages('spDataLarge',
## repos='https://nowosad.github.io/drat/', type='source')`
```

```
#library(spDataLarge)
library(tmap)    # for static and interactive maps
library(leaflet) # for interactive maps
library(ggplot2)
```

```
country <- ad_df$country
countyfreq <- table(country)
```

Gender

```
male <- ad_df$male
male_freq <- table(male)
barplot(male_freq, main= 'Gender Distribution', xlab="Gender",
ylab="Number of people",
border="red",
col="steelblue")
```



###Overall Summary

```
summary(ad_df)
```

```
##  daily.time.spent.on.site      age         area.income    daily.internet.usage
##  Min.   :32.60            Min.   :19.00   Min.   :13996   Min.   :104.8
##  1st Qu.:51.36            1st Qu.:29.00   1st Qu.:47032   1st Qu.:138.8
##  Median :68.22            Median :35.00   Median :57012   Median :183.1
##  Mean   :65.00            Mean   :36.01   Mean   :55000   Mean   :180.0
##  3rd Qu.:78.55            3rd Qu.:42.00   3rd Qu.:65471   3rd Qu.:218.8
##  Max.   :91.43            Max.   :61.00   Max.   :79485   Max.   :270.0
##  ad.topic.line          city                 male          country
##  Length:1000        Length:1000        Min.   :0.000   Length:1000
##  Class :character   Class :character   1st Qu.:0.000   Class :character
##  Mode  :character   Mode  :character   Median :0.000   Mode  :character
##                                        Mean   :0.481
##                                        3rd Qu.:1.000
##                                        Max.   :1.000
##    timestamp                    clicked.on.ad
##  Min.   :2016-01-01 02:52:10   Min.   :0.0
##  1st Qu.:2016-02-18 02:55:42   1st Qu.:0.0
##  Median :2016-04-07 17:27:29   Median :0.5
##  Mean   :2016-04-10 10:56:04   Mean   :0.5
##  3rd Qu.:2016-05-31 03:18:14   3rd Qu.:1.0
##  Max.   :2016-07-24 00:22:16   Max.   :1.0
```

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:raster':
##
##     intersect, union

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

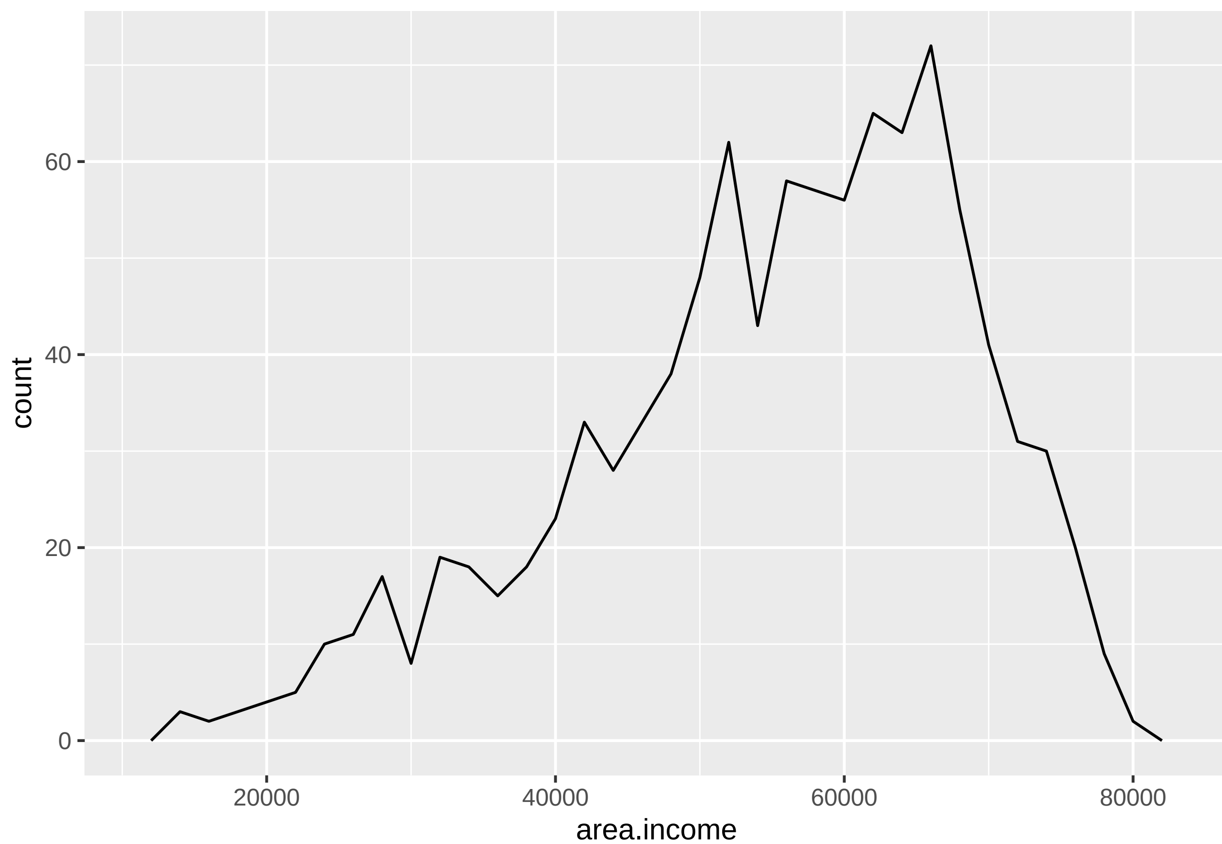
```
ad_df$Month_Yr <- format(as.Date(ad_df$timestamp), "%Y-%m")
head(ad_df)
```

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

```
## 2    Monitored national standardization      West Jodi     1      Nauru
## 3      Organic bottom-line service-desk       Davidton     0 San Marino
## 4 Triple-buffered reciprocal time-frame West Terrifurt     1      Italy
## 5         Robust logistical utilization   South Manuel     0    Iceland
## 6       Sharable client-driven software      Jamieberg     1     Norway
##             timestamp clicked.on.ad Month_Yr
## 1 2016-03-27 00:53:11             0  2016-03
## 2 2016-04-04 01:39:02             0  2016-04
## 3 2016-03-13 20:35:42             0  2016-03
## 4 2016-01-10 02:31:19             0  2016-01
## 5 2016-06-03 03:36:18             0  2016-06
## 6 2016-05-19 14:30:17             0  2016-05
```

Bivariate Analysis

```
ggplot(data = ad_df, mapping = aes(x = area.income)) +
  geom_freqpoly(mapping = aes(colour = clicked.on.ad), binwidth = 2000)
```



In areas where the income lies between 60,000 and & 70,000 there is a higher number of people clicking the ads #### Correlation

```
#creating with only interger columns
numerical_df = ad_df[c("daily.time.spent.on.site", "age", "area.income","daily.internet.usage" ,"male",
head(numerical_df)
```

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

```
correlation = cor(numerical_df)
correlation
```

```
##                          daily.time.spent.on.site         age  area.income
## daily.time.spent.on.site               1.00000000 -0.33151334  0.310954413
## age                                   -0.33151334  1.00000000 -0.182604955
## area.income                            0.31095441 -0.18260496  1.000000000
## daily.internet.usage                   0.51865848 -0.36720856  0.337495533
## male                                  -0.01895085 -0.02104406  0.001322359
## clicked.on.ad                         -0.74811656  0.49253127 -0.476254628
##                          daily.internet.usage         male clicked.on.ad
## daily.time.spent.on.site           0.51865848 -0.018950855   -0.74811656
## age                               -0.36720856 -0.021044064    0.49253127
## area.income                        0.33749553  0.001322359   -0.47625463
## daily.internet.usage               1.00000000  0.028012326   -0.78653918
## male                               0.02801233  1.000000000   -0.03802747
## clicked.on.ad                     -0.78653918 -0.038027466    1.00000000
```

```
library("PerformanceAnalytics")
```

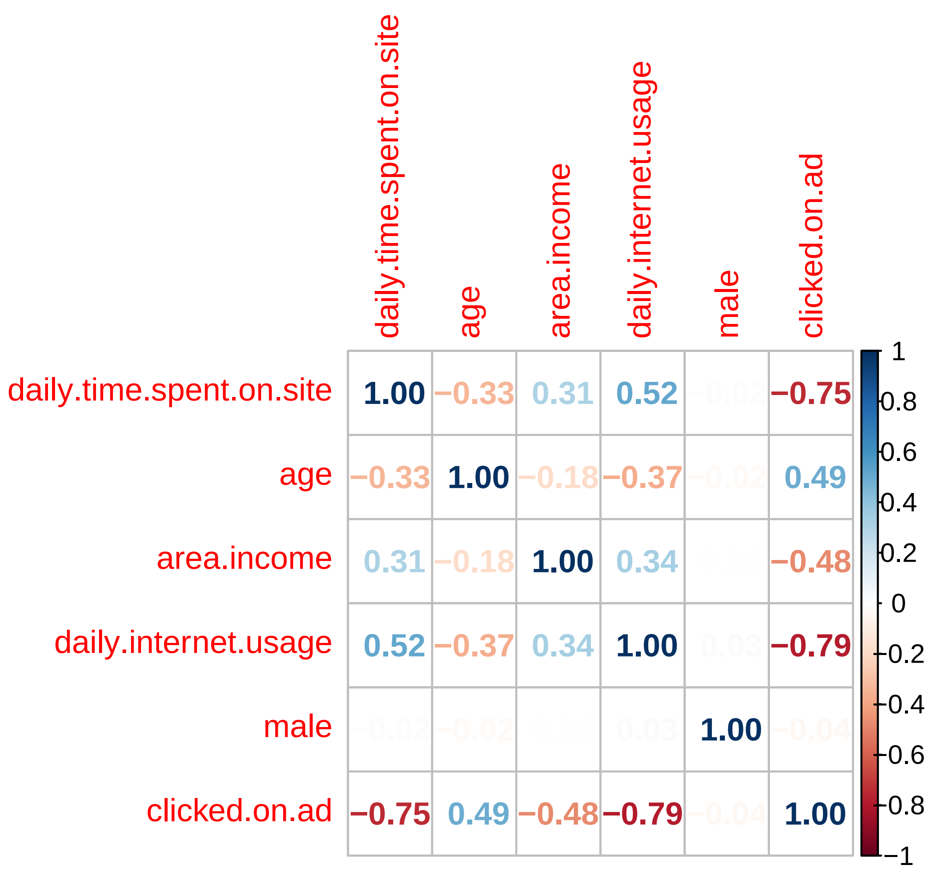
```
## Loading required package: xts

## Loading required package: zoo

## 
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
## 
##     as.Date, as.Date.numeric

## 
## Attaching package: 'xts'

## The following object is masked from 'package:leaflet':
## 
##     addLegend
```

```
## The following objects are masked from 'package:dplyr':
##
##     first, last

##
## Attaching package: 'PerformanceAnalytics'

## The following object is masked from 'package:graphics':
##
##     legend
```

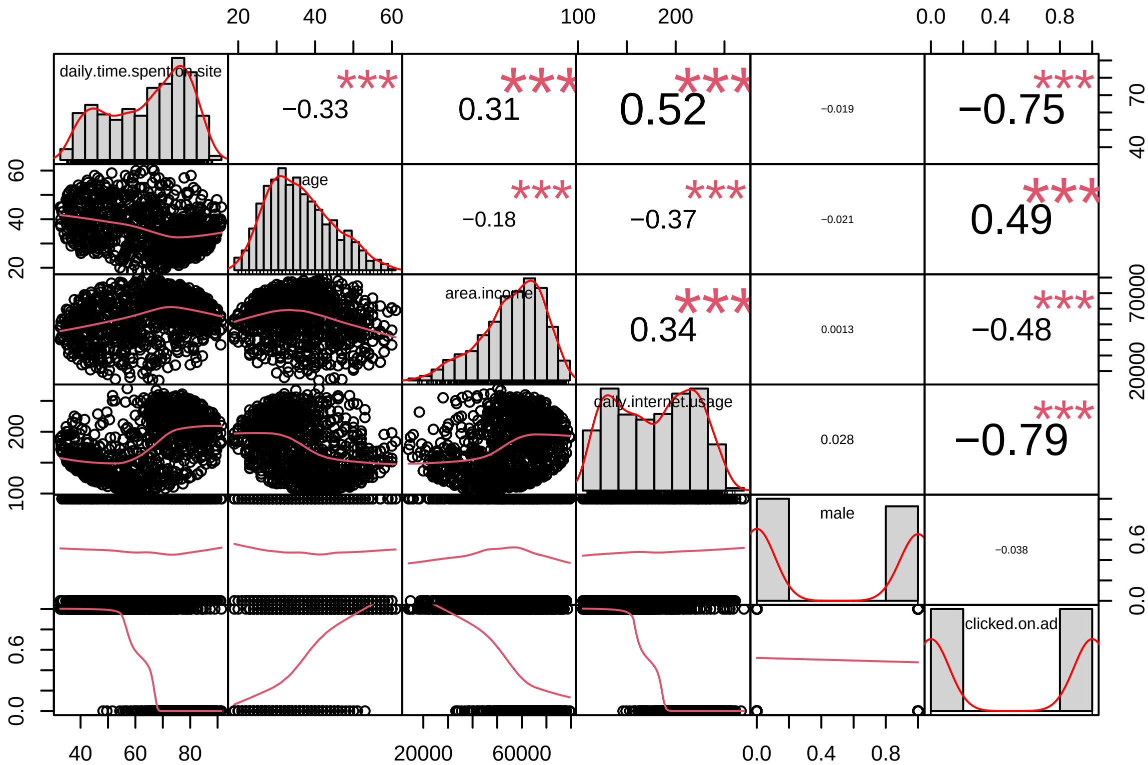
```
library(corrplot)
```

```
## corrplot 0.90 loaded
```

```
# Correlation Matrix
corrplot(correlation, method = 'number')
```




```
chart.Correlation(numerical_df, histogram = TRUE, pch = 19, )
```



The chart correlations gives a clear summary on the Bi-variate analysis of the dataframe.

Bivariate Conclusion

From the analysis we can get several deductions: - daily.time.spent.on.site and the clicked.on.ad have an inverse. - The mean age of the population is 35, and as the age increased more people clicked on ads. - There is an inverse relationship between the daily time spent on site and the number of people who click the ads - There were slightly more females in the dataset. - The gender of the users had the least effect on the number of ads clicked and barely affected any other variables. ’’’

Unsupervised learning.

```r
# preview data structure
str(numerical_df)
```

```
## 'data.frame':    1000 obs. of  6 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 ...
##  $ male                    : int  0 1 0 1 0 1 0 1 1 1 ...
##  $ clicked.on.ad           : int  0 0 0 0 0 0 0 1 0 0 ...
```

```
head(numerical_df)
```

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

```
#We have to first make sure all the columns are in numerical format
numerical_df[,1:6] <- sapply(numerical_df[,1:6], as.numeric)
head(numerical_df)
```

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

```
# Normalizing the numerical variables of the data set. Normalizing the numerical values is really effec
# as it provides a measure from 0 to 1 which corresponds to min value to the max value of the data colu
# We define a normal function which will normalize the set of values according to its minimum value and
normalize <- function(x) (
  return( ((x - min(x)) /(max(x)-min(x))) )
)
```

```
# Appliying the normalization function
numerical_df$area.income<- normalize(numerical_df$area.income)
numerical_df$daily.internet.usage<- normalize(numerical_df$daily.internet.usage)
numerical_df$daily.time.spent.on.site<- normalize(numerical_df$daily.time.spent.on.site)
numerical_df$male<- normalize(numerical_df$male)
numerical_df$age<- normalize(numerical_df$age)
head(numerical_df)
```

```
##   daily.time.spent.on.site       age area.income daily.internet.usage male
```

```
## 1                0.6178820 0.3809524   0.7304725            0.9160310    0
## 2                0.8096209 0.2857143   0.8313752            0.5387456    1
## 3                0.6267211 0.1666667   0.6992003            0.7974331    0
## 4                0.7062723 0.2380952   0.6231599            0.8542802    1
## 5                0.6080231 0.3809524   0.9145678            0.7313234    0
## 6                0.4655788 0.0952381   0.6988280            0.7383460    1
##   clicked.on.ad
## 1             0
## 2             0
## 3             0
## 4             0
## 5             0
## 6             0
```

This is a classification problem and for the models we will build two models the Decion trees model and the SVM model Lets begin

Decision Tree

Importing the important libraries in modelling.

```r
library(rpart,quietly = TRUE)
library(caret,quietly = TRUE)
library(rpart.plot,quietly = TRUE)
library(rpart.plot)
library(rattle)
```

```
## Loading required package: tibble

## Loading required package: bitops

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

```r
library(e1071)
```

```
##
## Attaching package: 'e1071'

## The following objects are masked from 'package:PerformanceAnalytics':
##
##     kurtosis, skewness

## The following object is masked from 'package:raster':
##
##     interpolate
```

Start with data preparation such as Splitting the data into training and testing set

```r
set.seed(42)
dttrain <- sample(1:nrow(numerical_df),size = ceiling(0.80*nrow(numerical_df)),replace = FALSE)
# training set
dt_train <- numerical_df[dttrain,]
# test set
dt_test <- numerical_df[-dttrain,]
```

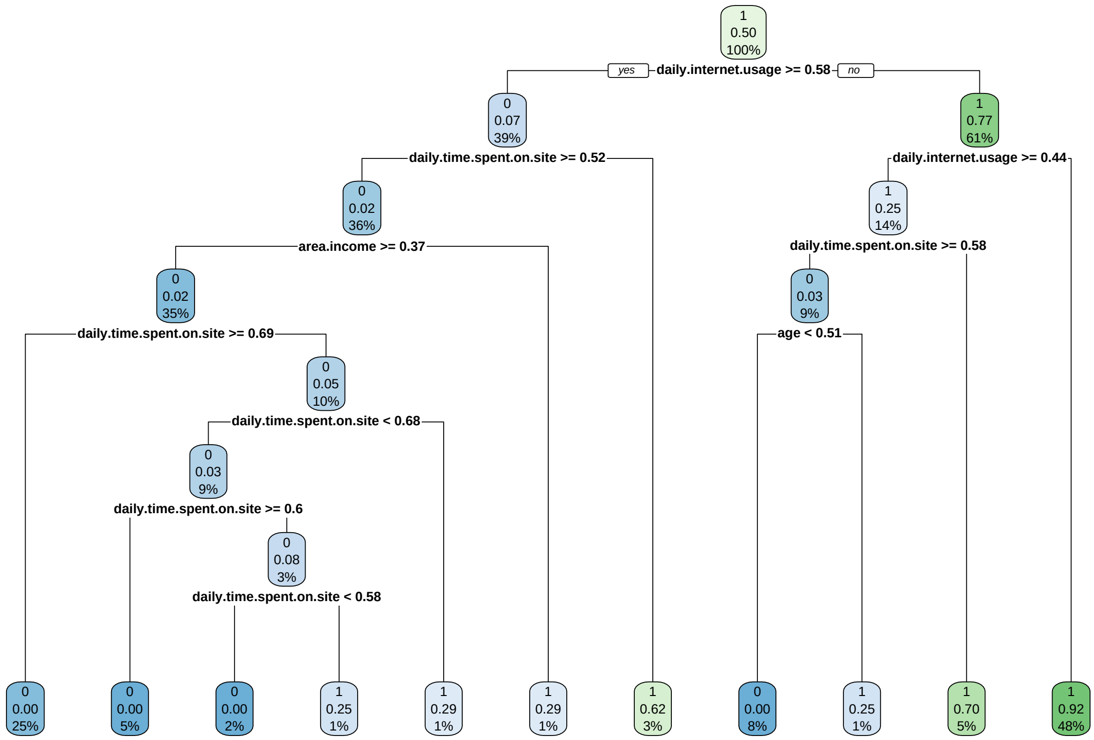
we performed an 80-20 split on the data

```r
# we are defining the penalty matrix for the decision tree to ensure that the model has more accurate p
# The penalty will multiply an error by 10
# Penalty matrix
penalty.matrix <- matrix(c(0, 1, 10,0), byrow = TRUE, nrow = 2)
```

```r
dtree <- rpart(clicked.on.ad ~., data = dt_train, parms=list(loss=penalty.matrix), method = 'class')
dtree
```

```
## n= 800
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##   1) root 800 399 1 (0.498750000 0.501250000)
##     2) daily.internet.usage>=0.5780058 309 220 0 (0.928802589 0.071197411)
##       4) daily.time.spent.on.site>=0.5224375 285  70 0 (0.975438596 0.024561404)
##         8) area.income>=0.3704821 278  50 0 (0.982014388 0.017985612)
##          16) daily.time.spent.on.site>=0.690804 202  10 0 (0.995049505 0.004950495) *
##          17) daily.time.spent.on.site< 0.690804 76  40 0 (0.947368421 0.052631579)
##            34) daily.time.spent.on.site< 0.6808601 69  20 0 (0.971014493 0.028985507)
##              68) daily.time.spent.on.site>=0.6013089 43   0 0 (1.000000000 0.000000000) *
##              69) daily.time.spent.on.site< 0.6013089 26  20 0 (0.923076923 0.076923077)
##               138) daily.time.spent.on.site< 0.5802312 18   0 0 (1.000000000 0.000000000) *
##               139) daily.time.spent.on.site>=0.5802312 8   6 1 (0.750000000 0.250000000) *
##            35) daily.time.spent.on.site>=0.6808601 7   5 1 (0.714285714 0.285714286) *
##         9) area.income< 0.3704821 7   5 1 (0.714285714 0.285714286) *
##       5) daily.time.spent.on.site< 0.5224375 24   9 1 (0.375000000 0.625000000) *
##     3) daily.internet.usage< 0.5780058 491 112 1 (0.228105906 0.771894094)
##       6) daily.internet.usage>=0.4388243 111  83 1 (0.747747748 0.252252252)
##        12) daily.time.spent.on.site>=0.581506 74  20 0 (0.972972973 0.027027027)
##          24) age< 0.5119048 66   0 0 (1.000000000 0.000000000) *
##          25) age>=0.5119048 8   6 1 (0.750000000 0.250000000) *
##        13) daily.time.spent.on.site< 0.581506 37  11 1 (0.297297297 0.702702703) *
##       7) daily.internet.usage< 0.4388243 380  29 1 (0.076315789 0.923684211) *
```

```r
rpart.plot(dtree)
```




```
# Calculating the metrics of the decison Tree model
# Predictions Dtree model
predt <- predict(object = dtree, dt_test[,-6], type = 'class')
#calculating accuracy
t <- table(dt_test$clicked.on.ad, predt)
confusionMatrix(t)
```

```
## Confusion Matrix and Statistics
##
##    predt
##      0  1
##   0 81 20
##   1  2 97
##
##                Accuracy : 0.89
##                  95% CI : (0.8382, 0.9298)
##     No Information Rate : 0.585
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7804
##
##  Mcnemar's Test P-Value : 0.0002896
##
##             Sensitivity : 0.9759
##             Specificity : 0.8291
##          Pos Pred Value : 0.8020
```

```
##          Neg Pred Value : 0.9798
##              Prevalence : 0.4150
##          Detection Rate : 0.4050
##    Detection Prevalence : 0.5050
##       Balanced Accuracy : 0.9025
##
##        'Positive' Class : 0
##
```

The decision tree model has an accuracy of 89% That is quite acceptable as the metrics for success needed an accuracy of 80%

LinearSVM

```
library('caret')
# Performing an 80 - 20 split

svmtrain <- createDataPartition(y = numerical_df$clicked.on.ad, p= 0.8, list = FALSE)
training <- numerical_df[svmtrain,]

testing <- numerical_df[-svmtrain,]
```

```
# Preview the dimensions of the training and testing data
dim(training)
```

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

```
dim(testing)
```

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

```
# Building a LinearSVM model
# SVM model

classifierL = svm(formula = clicked.on.ad ~ .,
                 data = training,
                 type = 'C-classification',
                 kernel = 'linear')
```

```
# Running the metrics for the linear classification svm
y_predL = predict(classifierL, newdata = testing)

cmL = table(testing$clicked.on.ad, y_predL)
confusionMatrix(cmL)
```

```
## Confusion Matrix and Statistics
##
##    y_predL
##      0  1
##   0 98  2
```

```
##   1  2 98
##
##                Accuracy : 0.98
##                  95% CI : (0.9496, 0.9945)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.96
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.98
##             Specificity : 0.98
##          Pos Pred Value : 0.98
##          Neg Pred Value : 0.98
##              Prevalence : 0.50
##          Detection Rate : 0.49
##    Detection Prevalence : 0.50
##       Balanced Accuracy : 0.98
##
##        'Positive' Class : 0
##
```

The Linear SVM model has an accuracy of 98% which is quite an improvement from the Decision tree model. However, we can still challenge the model, to find a better modle that might account for overfitting.

Challenging the solution

```r
# Running the clasifier with a
classifierRB = svm(formula = clicked.on.ad ~ .,
                 data = training,
                 type = 'C-classification',
                 kernel = 'sigmoid')
```

```r
# Running the metrics for the linear classification svm
y_predRB = predict(classifierRB, newdata = testing)

cmRB = table(testing$clicked.on.ad, y_predRB)
confusionMatrix(cmRB)
```

```
## Confusion Matrix and Statistics
##
##    y_predRB
##      0  1
##   0 93  7
##   1  4 96
##
##                Accuracy : 0.945
##                  95% CI : (0.9037, 0.9722)
##     No Information Rate : 0.515
##     P-Value [Acc > NIR] : <2e-16
```

```
##
##                  Kappa : 0.89
##
##  Mcnemar's Test P-Value : 0.5465
##
##             Sensitivity : 0.9588
##             Specificity : 0.9320
##          Pos Pred Value : 0.9300
##          Neg Pred Value : 0.9600
##              Prevalence : 0.4850
##          Detection Rate : 0.4650
##    Detection Prevalence : 0.5000
##       Balanced Accuracy : 0.9454
##
##        'Positive' Class : 0
##
```

With a sigmoid kernel on SVM technique the accuracy is (94.5%). This might be a better model because it would account for the overfitting.

Conclusion

Was the data enough to answer the given questions? The data provided was sufficient in the analysis.

Do you have any recommendation on what data should be added? I think that the data was good, however, having more data would not be bad and training a bigger dataframe could lead to more accurate models.

The data was cleaned and used for analysis in the data frame