supervised learning

kelvin njunge

4/6/2021

Research Question

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

1. Defining the question

1.1 Specifying the data analytic objective

Our main aim is to do thorough exploratory data analysis for univariate and bivariate and come up with recommendations for our client.

1.2 Defining the metric for success

We aim to do a elaborate visualizations for univariate and bivariate anaysis.

1.3. Recording the experimental design

- Loading the data
- Checking the data
- Tidying the data
- univariate analysis
- Bivariate analysis
- Challenge the solution
- Recommendation
- Follow up questions

1.4. Data Relevance

The data provided was relevant for our analysis

Loading the data

```r
getwd()
```

```
## [1] "C:/Users/Ricky/Documents"
```

```r
df <- read.csv("C:\\Users\\Ricky\\Documents\\Adobe\\advertising.csv")
```

Viewing the top 6 entries

```r
head(df)
```

```
##   Daily.Time.Spent.on.Site Age Area.Income Daily.Internet.Usage
## 1                    68.95  35    61833.90               256.09
## 2                    80.23  31    68441.85               193.77
## 3                    69.47  26    59785.94               236.50
## 4                    74.15  29    54806.18               245.89
## 5                    68.37  35    73889.99               225.58
## 6                    59.99  23    59761.56               226.74
##                          Ad.Topic.Line           City Male    Country
## 1    Cloned 5thgeneration orchestration    Wrightburgh    0    Tunisia
## 2    Monitored national standardization      West Jodi    1      Nauru
## 3      Organic bottom-line service-desk       Davidton    0 San Marino
## 4 Triple-buffered reciprocal time-frame West Terrifurt    1      Italy
## 5         Robust logistical utilization   South Manuel    0    Iceland
## 6       Sharable client-driven software      Jamieberg    1     Norway
##             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
```

Viewing the bottom 6 entries

```r
tail(df)
```

```
##      Daily.Time.Spent.on.Site Age Area.Income Daily.Internet.Usage
## 995                     43.70  28    63126.96               173.01
## 996                     72.97  30    71384.57               208.58
## 997                     51.30  45    67782.17               134.42
## 998                     51.63  51    42415.72               120.37
## 999                     55.55  19    41920.79               187.95
## 1000                    45.01  26    29875.80               178.35
##                             Ad.Topic.Line          City Male
## 995         Front-line bifurcated ability  Nicholasland    0
## 996         Fundamental modular algorithm     Duffystad    1
## 997       Grass-roots cohesive monitoring   New Darlene    1
```

```
## 998          Expanded intangible solution South Jessica    1
## 999  Proactive bandwidth-monitored policy   West Steven    0
## 1000      Virtual 5thgeneration emulation   Ronniemouth    0
##                    Country           Timestamp Clicked.on.Ad
## 995                Mayotte 2016-04-04 03:57:48             1
## 996                Lebanon 2016-02-11 21:49:00             1
## 997  Bosnia and Herzegovina 2016-04-22 02:07:01             1
## 998               Mongolia 2016-02-01 17:24:57             1
## 999              Guatemala 2016-03-24 02:35:54             0
## 1000                Brazil 2016-06-03 21:43:21             1
```

viewing the info of the dataset

```
str(df)
```

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

*** 3. Tidying the data***

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

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

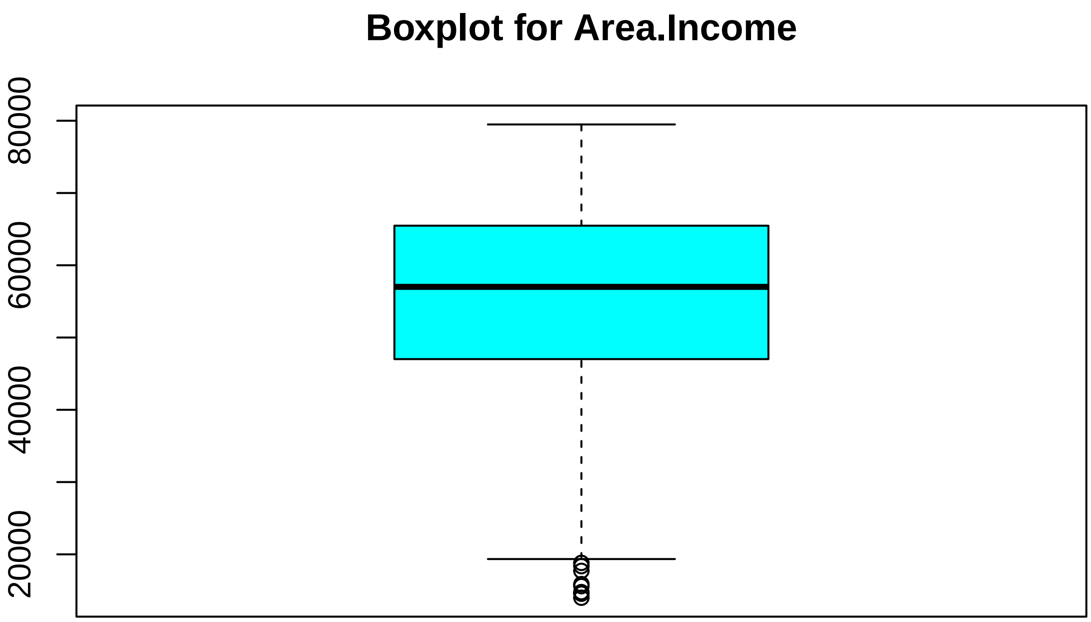
Checking for duplicates

```
sum(duplicated(df))
```

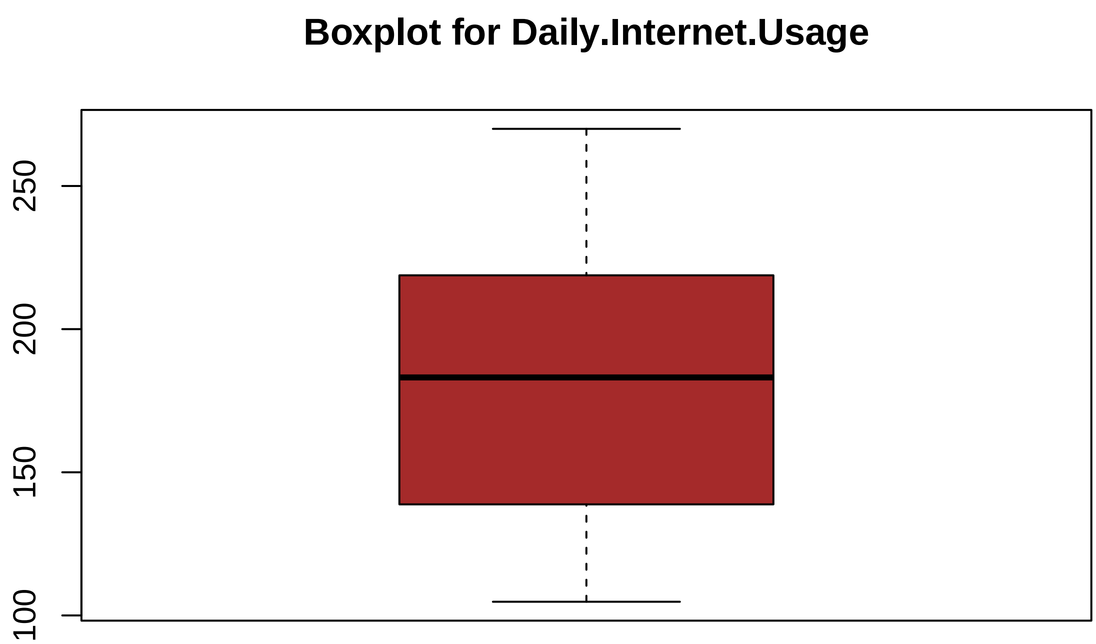
```
## [1] 0
```

There are no missing values in our data. There are no duplicates in our data ### ***Checking for outliers***

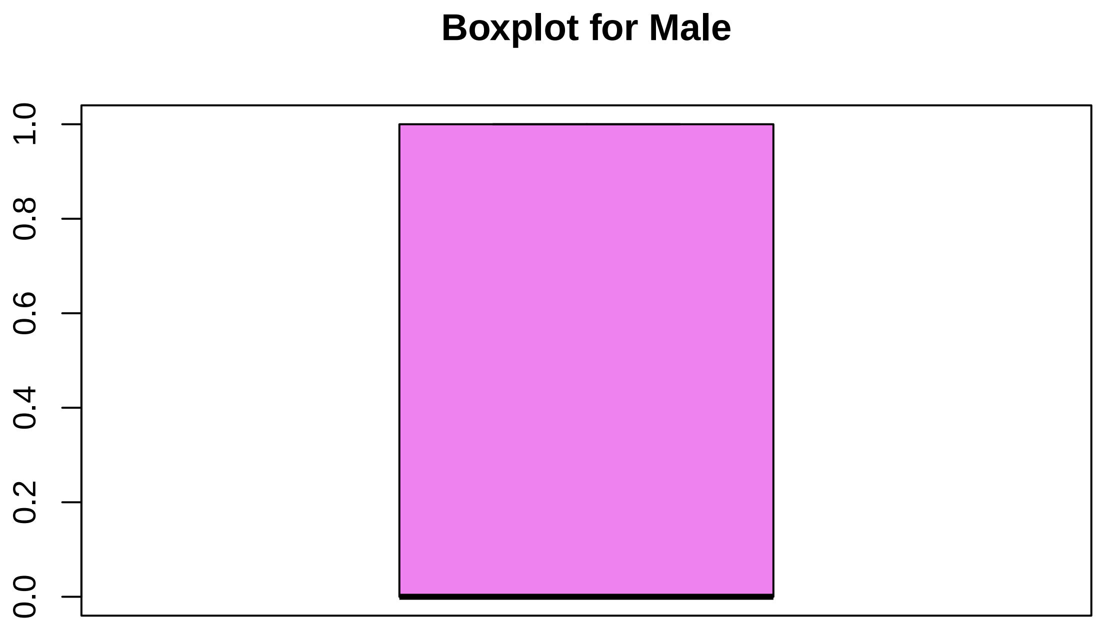
```
boxplot(df$Area.Income,main="Boxplot for Area.Income",col = "cyan")
```



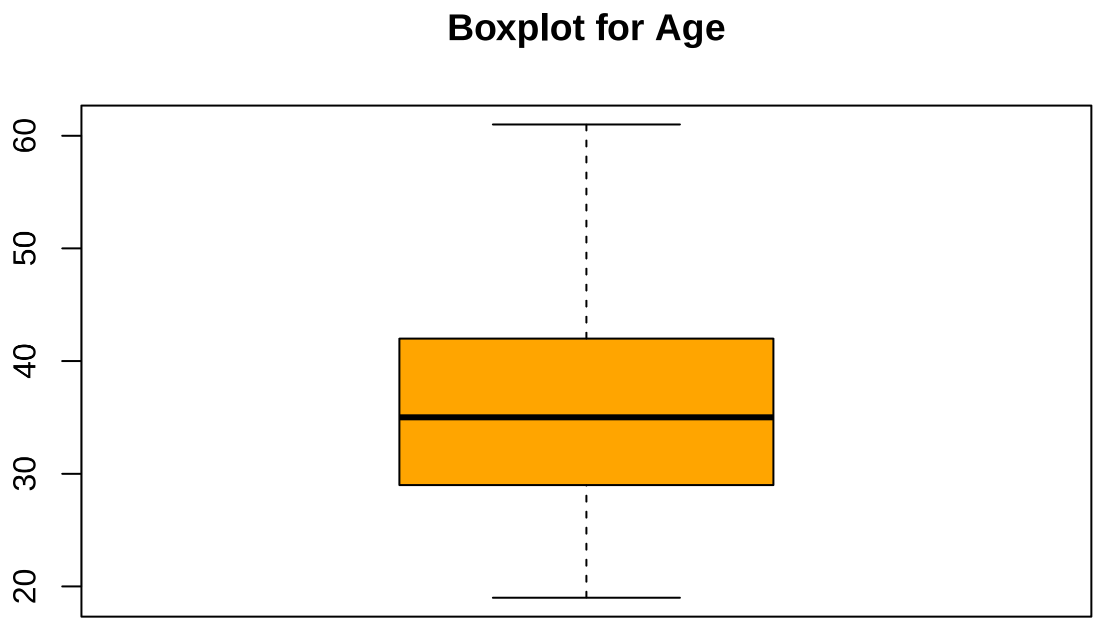

```
boxplot(df$Daily.Internet.Usage,main="Boxplot for Daily.Internet.Usage",col="brown")
```



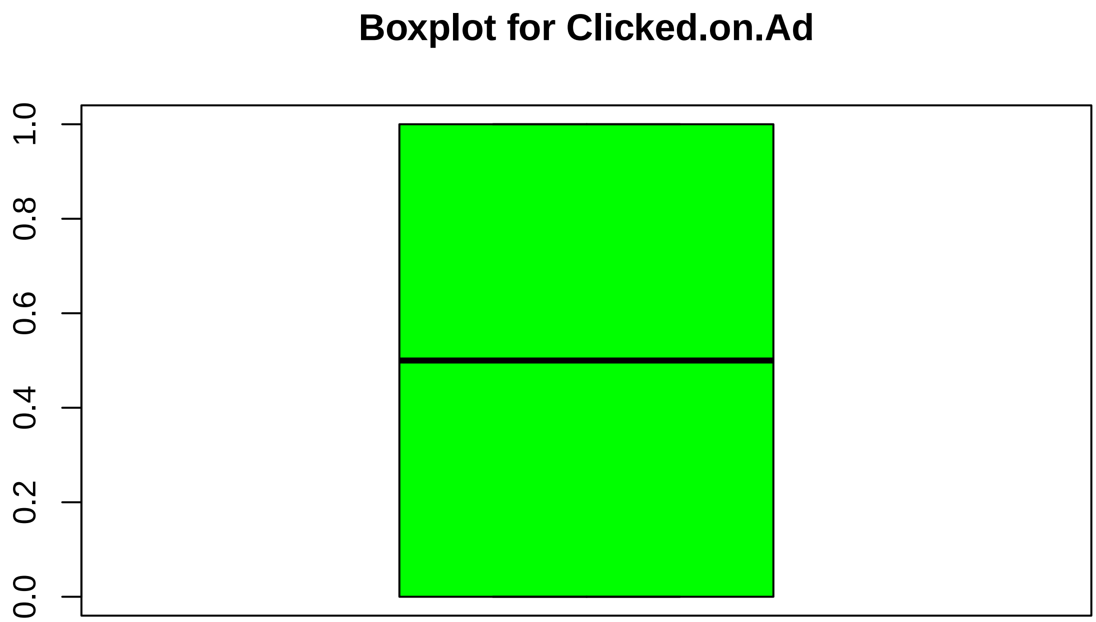

```
boxplot(df$Male,main="Boxplot for Male",col = "violet")
```




```r
boxplot(df$Age,main="Boxplot for Age",col = "orange")
```




```
boxplot(df$Clicked.on.Ad,main="Boxplot for Clicked.on.Ad",col = "green")
```



Area income was the only column that had outliers ## ***4. Univariate analysis***

4.1. measures of central tendency

4.1.1 mean

```
print("The mean for numeric variables is:")
```

```
## [1] "The mean for numeric variables is:"
```

```
colMeans(df[sapply(df,is.numeric)])
```

```
## Daily.Time.Spent.on.Site                      Age               Area.Income
##                  65.0002                  36.0090                55000.0001
##     Daily.Internet.Usage                     Male               Clicked.on.Ad
##                 180.0001                   0.4810                    0.5000
```

```
cat("The median for daily time spent on site is",median(df$Daily.Time.Spent.on.Site))
```

```
## The median for daily time spent on site is 68.215
```

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

```r
cat("The median for age is",median(df$Age))
```

```
## The median for age is 35
```

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

```r
cat("The median for Area.Income is",median(df$Area.Income))
```

```
## The median for Area.Income is 57012.3
```

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

```r
cat("The median for daily Internet Usage is",median(df$Daily.Internet.Usage))
```

```
## The median for daily Internet Usage is 183.13
```

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

```r
cat("The median for Clicked on Ad",median(df$Clicked.on.Ad))
```

```
## The median for Clicked on Ad 0.5
```

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

4.1.3 mode

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

```r
cat("The mode for daily time spent on site is",getmode(df$Daily.Time.Spent.on.Site))
```

```
## The mode for daily time spent on site is 62.26
```

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

```r
cat("The mode for age is",getmode(df$Age))
```

```
## The mode for age is 31
```

```
cat("\n")
```

```
cat("The mode for Area.Income is",getmode(df$Area.Income))
```

```
## The mode for Area.Income is 61833.9
```

```
cat("\n")
```

```
cat("The mode for daily Internet Usage is",getmode(df$Daily.Internet.Usage))
```

```
## The mode for daily Internet Usage is 167.22
```

```
cat("\n")
```

```
cat("The mode for Clicked on Ad",getmode(df$Clicked.on.Ad))
```

```
## The mode for Clicked on Ad 0
```

```
cat("\n")
```

4.1.4 standard deviation

```
cat("The standard deviation for age is",sd(df$`Age`))
```

```
## The standard deviation for age is 8.785562
```

```
cat("\n")
```

```
cat("The standard deviation for Area.Income is",sd(df$`Area Income`))
```

```
## The standard deviation for Area.Income is NA
```

```
cat("\n")
```

```
cat("The mode for daily time spent on site is",sd(df$Daily.Time.Spent.on.Site))
```

```
## The mode for daily time spent on site is 15.85361
```

```
cat("\n")
```

4.1.5. variance

```
cat("The variance for daily time spent on site is",var(df$Daily.Time.Spent.on.Site))
```

```
## The variance for daily time spent on site is 251.3371
```

```
cat("\n")
```

```
cat("The variance for age is",var(df$Age))
```

```
## The variance for age is 77.18611
```

```
cat("The variance for daily Internet Usage is",var(df$Daily.Internet.Usage))
```

```
## The variance for daily Internet Usage is 1927.415
```

```
cat("The variance for Clicked on Ad",var(df$Clicked.on.Ad))
```

```
## The variance for Clicked on Ad 0.2502503
```

```
cat("The variance for Area.Income is",var(df$Area.Income))
```

```
## The variance for Area.Income is 179952406
```

```
cat("The variance for male is",var(df$Male))
```

```
## The variance for male is 0.2498889
```

4.2.2 maximum

```
library(dplyr)
```

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

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

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

```
df %>% summarise_if(is.numeric, max)
```

```
##   Daily.Time.Spent.on.Site Age Area.Income Daily.Internet.Usage Male
## 1                    91.43  61     79484.8               269.96    1
##   Clicked.on.Ad
## 1             1
```

minimum of the columns

```
df %>% summarise_if(is.numeric,min)
```

```
##   Daily.Time.Spent.on.Site Age Area.Income Daily.Internet.Usage Male
## 1                     32.6  19     13996.5               104.78    0
##   Clicked.on.Ad
## 1             0
```

4.2.3 Range

```
cat("The range for daily time spent on site is",range(df$Daily.Time.Spent.on.Site))
```

```
## The range for daily time spent on site is 32.6 91.43
```

```
cat("The range for age is",range(df$Age))
```

```
## The range for age is 19 61
```

```
cat("The range for Area income is",range(df$Area.Income))
```

```
## The range for Area income is 13996.5 79484.8
```

```
cat("The range for male is",range(df$Male))
```

```
## The range for male is 0 1
```

```
cat("The range  for daily internet usage is,",range(df$Daily.Internet.Usage))
```

```
## The range  for daily internet usage is, 104.78 269.96
```

```
cat("The range for clicked on ad",range(df$Clicked.on.Ad))
```

```
## The range for clicked on ad 0 1
```

4.2.3 Quantile

```
cat("The Quantile for age is",quantile(df$`Age`))
```

```
## The Quantile for age is 19 29 35 42 61
```

```
cat("The Quantile for male is",quantile(df$`Male`))
```

```
## The Quantile for male is 0 0 0 1 1
```

Summary

```
summary(df)
```

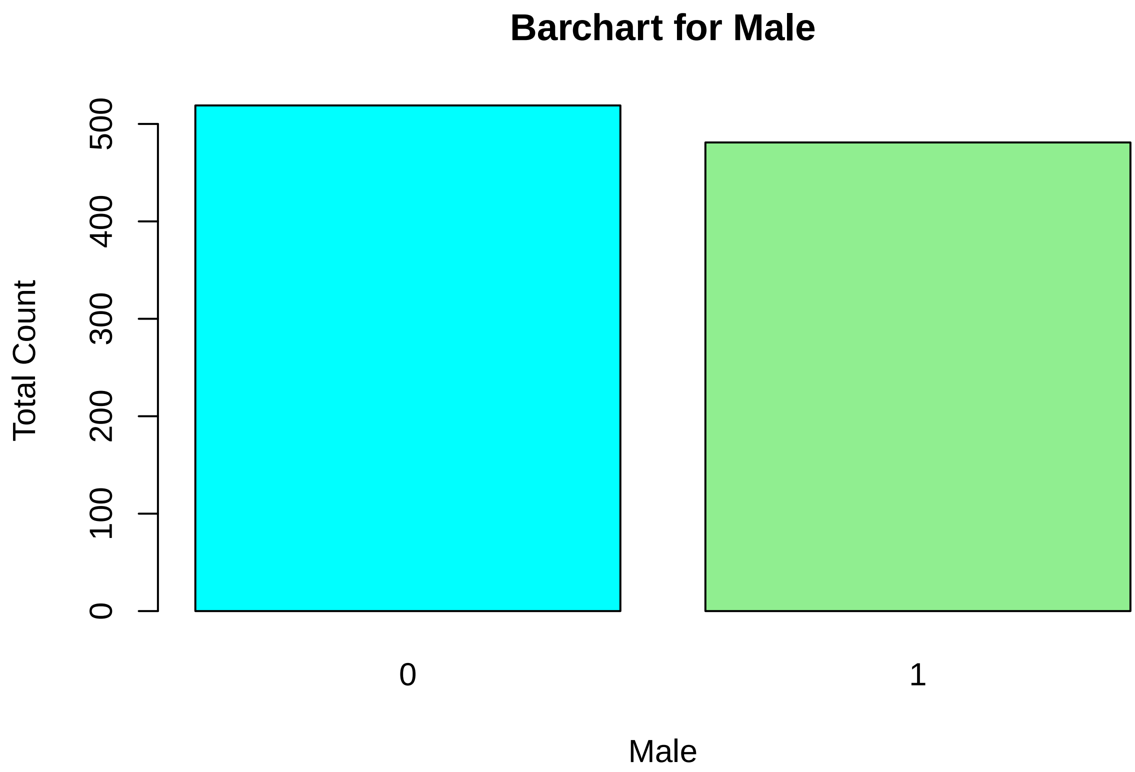
```
##  Daily.Time.Spent.on.Site      Age          Area.Income    Daily.Internet.Usage
##  Min.   :32.60            Min.   :19.00   Min.   :13996   Min.   :104.8
##  1st Qu.:51.36            1st Qu.:29.00   1st Qu.:47032   1st Qu.:138.8
##  Median :68.22            Median :35.00   Median :57012   Median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

Univariate

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

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

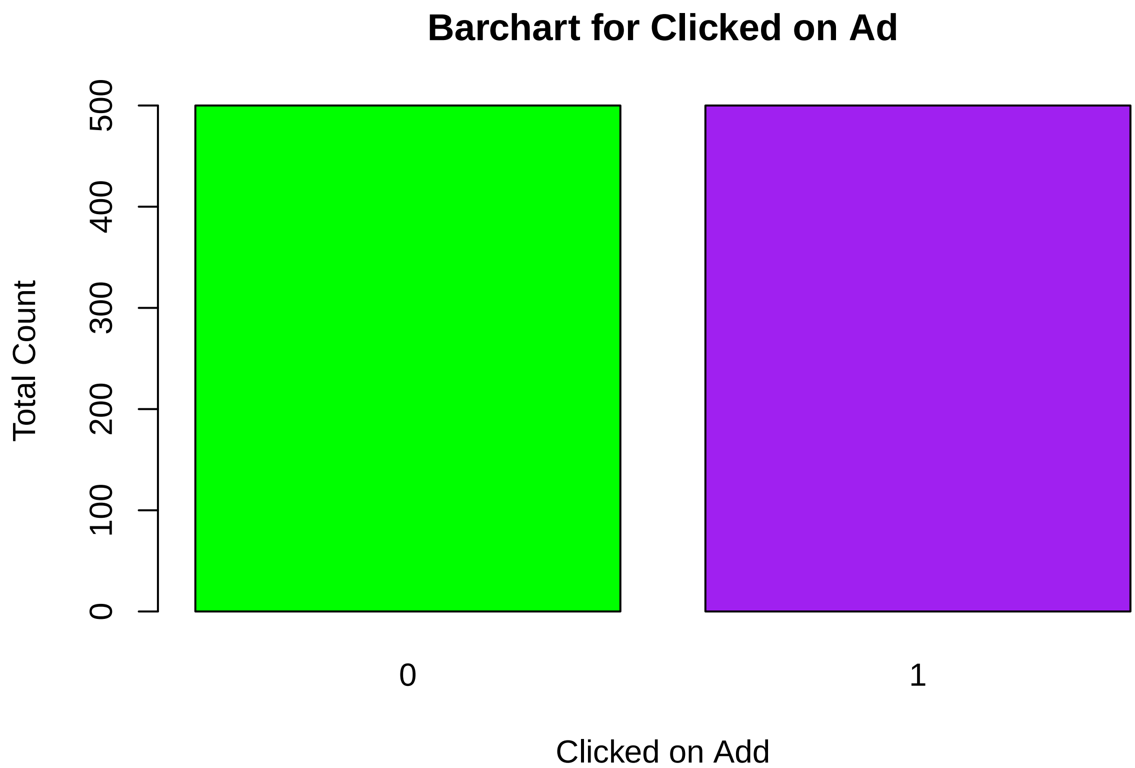
```
barplot(frequency,col=c("Cyan","lightgreen"),main="Barchart for Male",xlab = "Male",ylab = "Total Count
```




```
frequency <- table(df$Clicked.on.Ad)
frequency
```

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

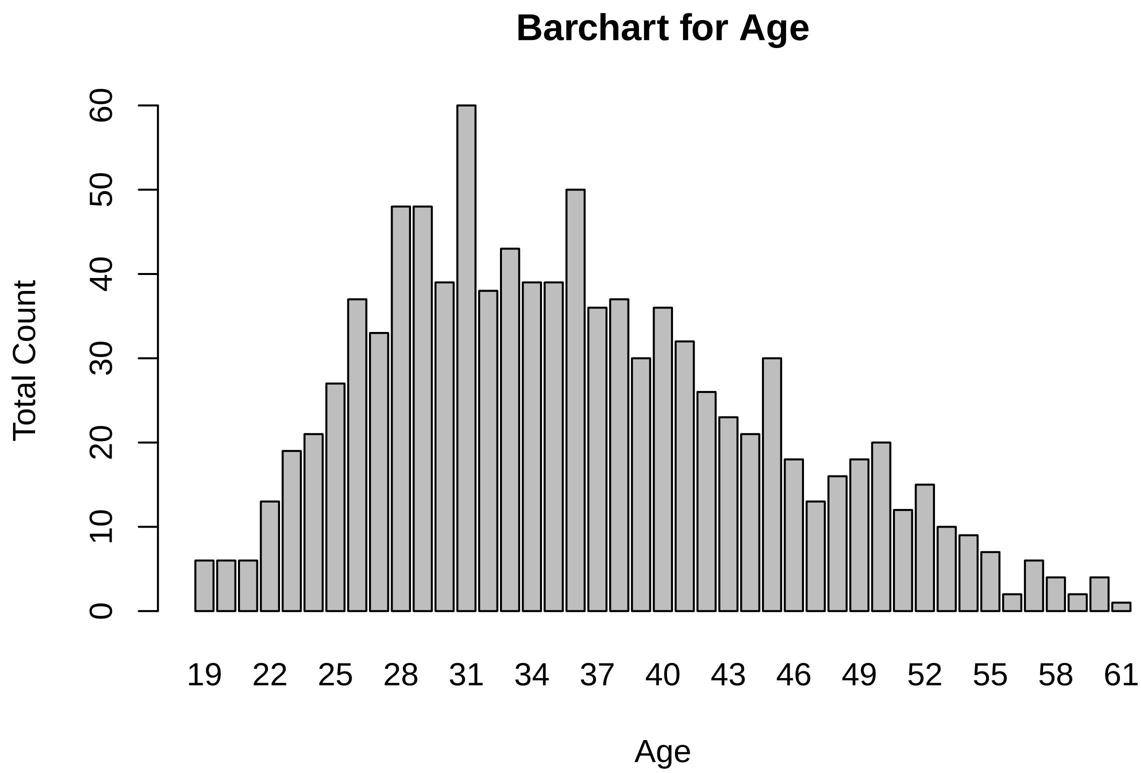
```
barplot(frequency,col=c("green","purple"),main="Barchart for Clicked on Ad",xlab = "Clicked on Add",ylal
```




```
frequency <- table(df$Age)
frequency
```

```
##
## 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44
##  6  6  6 13 19 21 27 37 33 48 48 39 60 38 43 39 39 50 36 37 30 36 32 26 23 21
## 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61
## 30 18 13 16 18 20 12 15 10  9  7  2  6  4  2  4  1
```

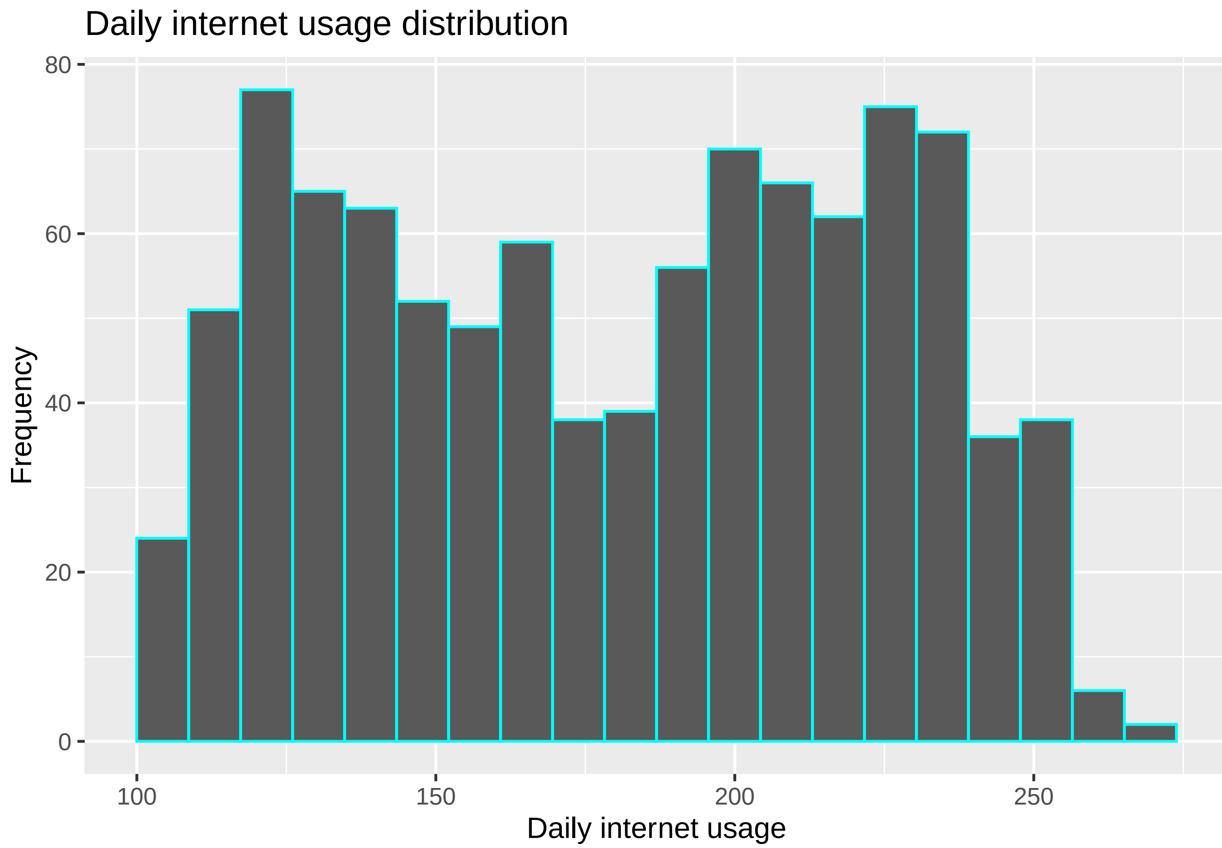
```
barplot(frequency,main="Barchart for Age",xlab = "Age",ylab = "Total Count")
```



Histograms

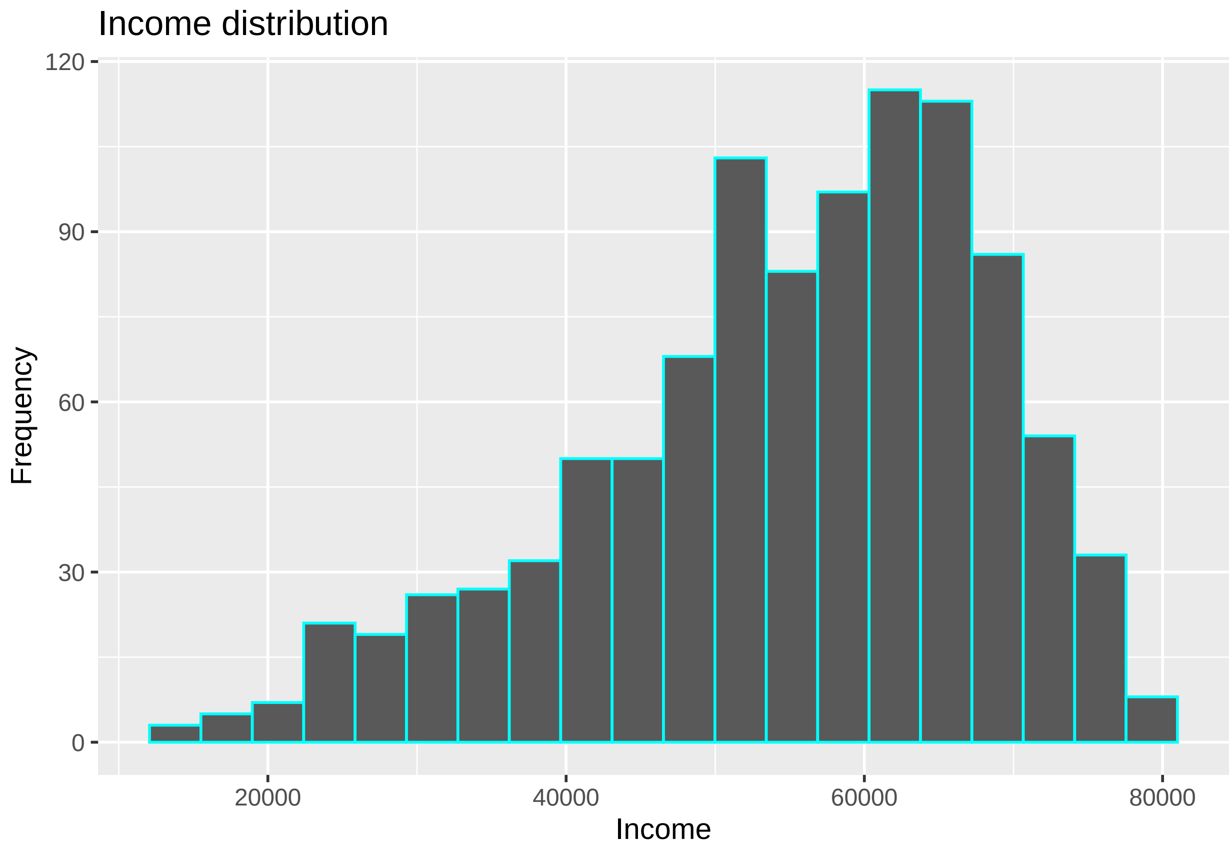
```
library("ggplot2")
```

```
ggplot(df, aes(Daily.Internet.Usage)) + geom_histogram(bins = 20, color = 'cyan') +
    labs(title = 'Daily internet usage distribution', x = 'Daily internet usage', y = 'Frequency')
```



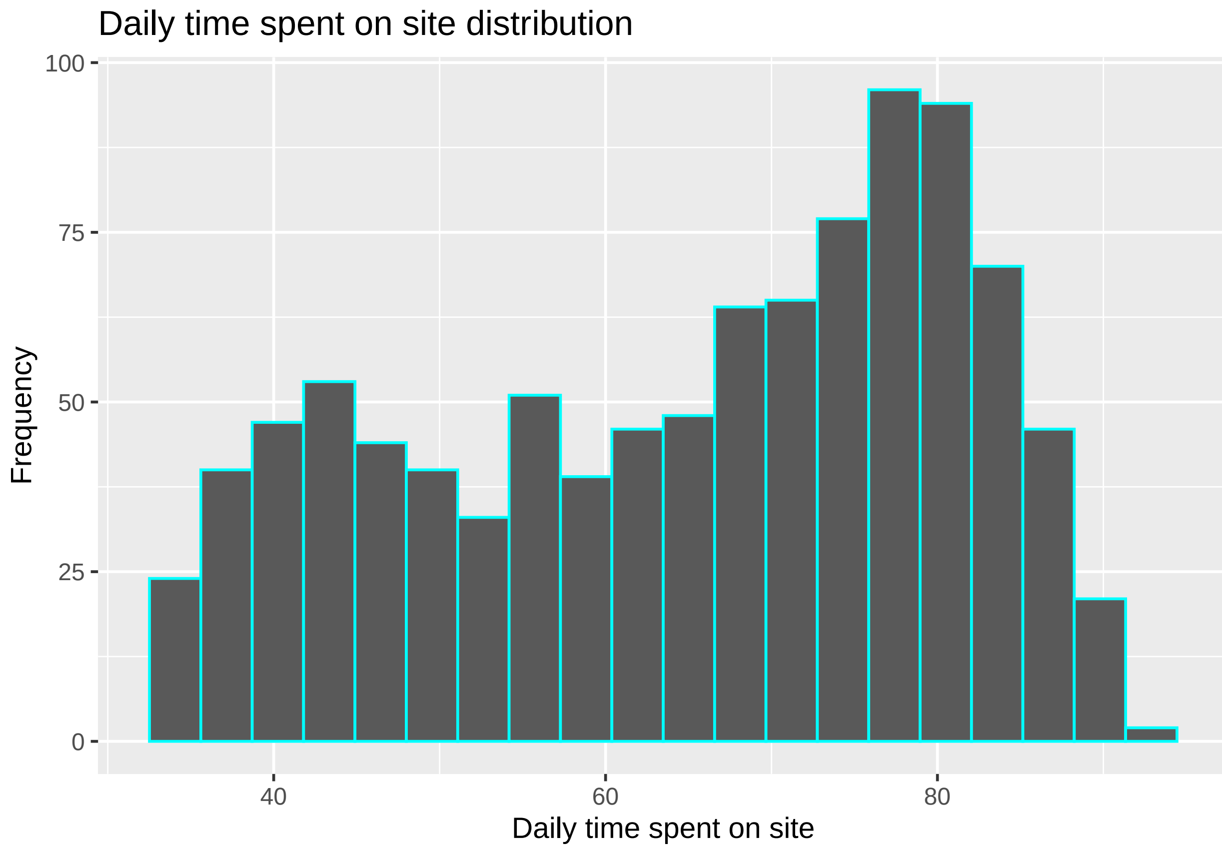
There is no particular pattern for the daily internet usage

```
ggplot(df, aes(Area.Income)) + geom_histogram(bins = 20, color = 'cyan') +
    labs(title = 'Income distribution', x = 'Income', y = 'Frequency')
```



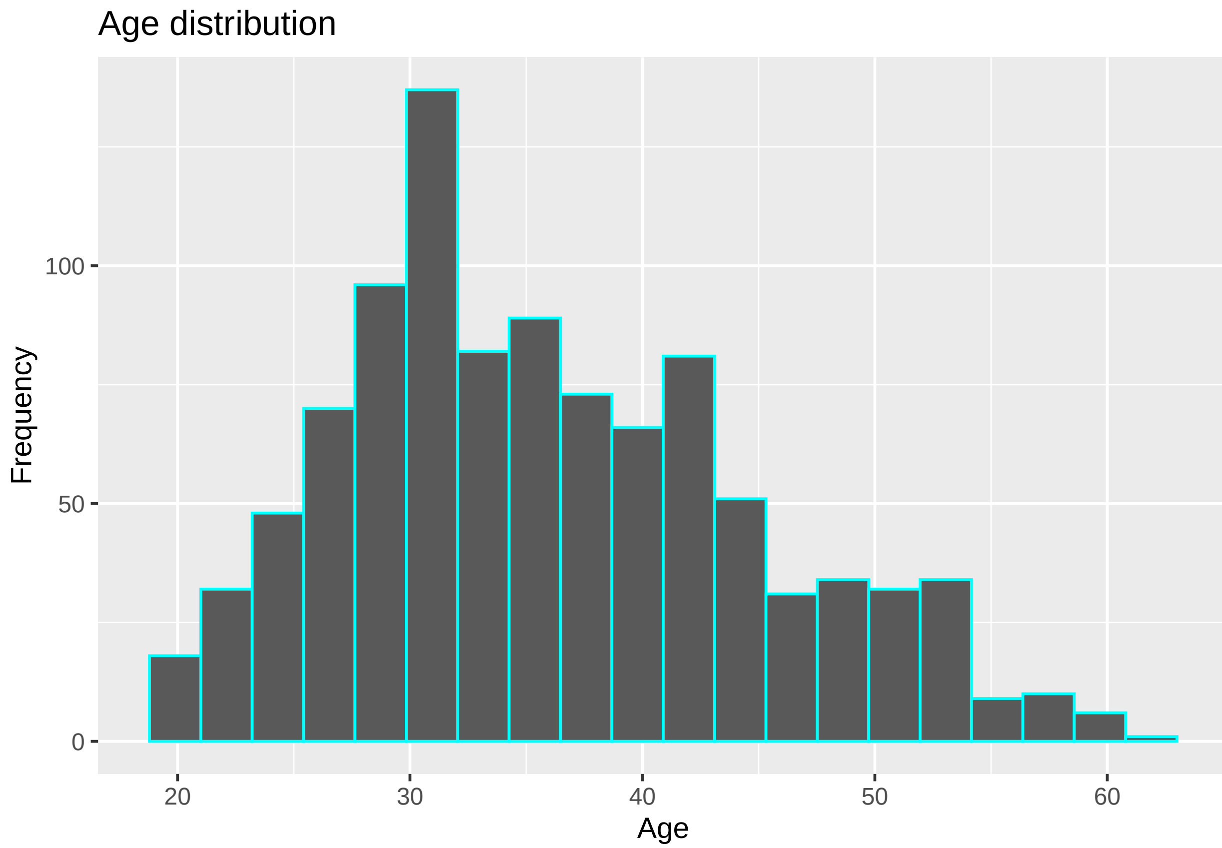
Most People were earning between 50,000 and 70,000.

```
ggplot(df, aes(Daily.Time.Spent.on.Site)) + geom_histogram(bins = 20, color = 'cyan') +
    labs(title = 'Daily time spent on site distribution', x = 'Daily time spent on site', y = 'Frequency
```



Majority of the people spent about 60-80 minutes on the blog site.

```
ggplot(df, aes(Age)) + geom_histogram(bins = 20, color = 'cyan') +
    labs(title = 'Age distribution', x = 'Age', y = 'Frequency')
```



Most of the respondents are between the ages of 25 and 50.

Bivariate analysis

```r
covarince <- cov(df[,sapply(df,is.numeric)])
covarince
```

covariance

```
##                          Daily.Time.Spent.on.Site           Age   Area.Income
## Daily.Time.Spent.on.Site              251.3370949 -4.617415e+01  6.613081e+04
## Age                                   -46.1741459  7.718611e+01 -2.152093e+04
## Area.Income                         66130.8109082 -2.152093e+04  1.799524e+08
## Daily.Internet.Usage                  360.9918827 -1.416348e+02  1.987625e+05
## Male                                   -0.1501864 -9.242142e-02  8.867509e+00
## Clicked.on.Ad                          -5.9331431  2.164665e+00 -3.195989e+03
##                          Daily.Internet.Usage        Male Clicked.on.Ad
## Daily.Time.Spent.on.Site         3.609919e+02 -0.15018639 -5.933143e+00
## Age                             -1.416348e+02 -0.09242142  2.164665e+00
## Area.Income                      1.987625e+05  8.86750903 -3.195989e+03
## Daily.Internet.Usage             1.927415e+03  0.61476667 -1.727409e+01
## Male                             6.147667e-01  0.24988889 -9.509510e-03
## Clicked.on.Ad                   -1.727409e+01 -0.00950951  2.502503e-01
```

correlation of all numeric variable

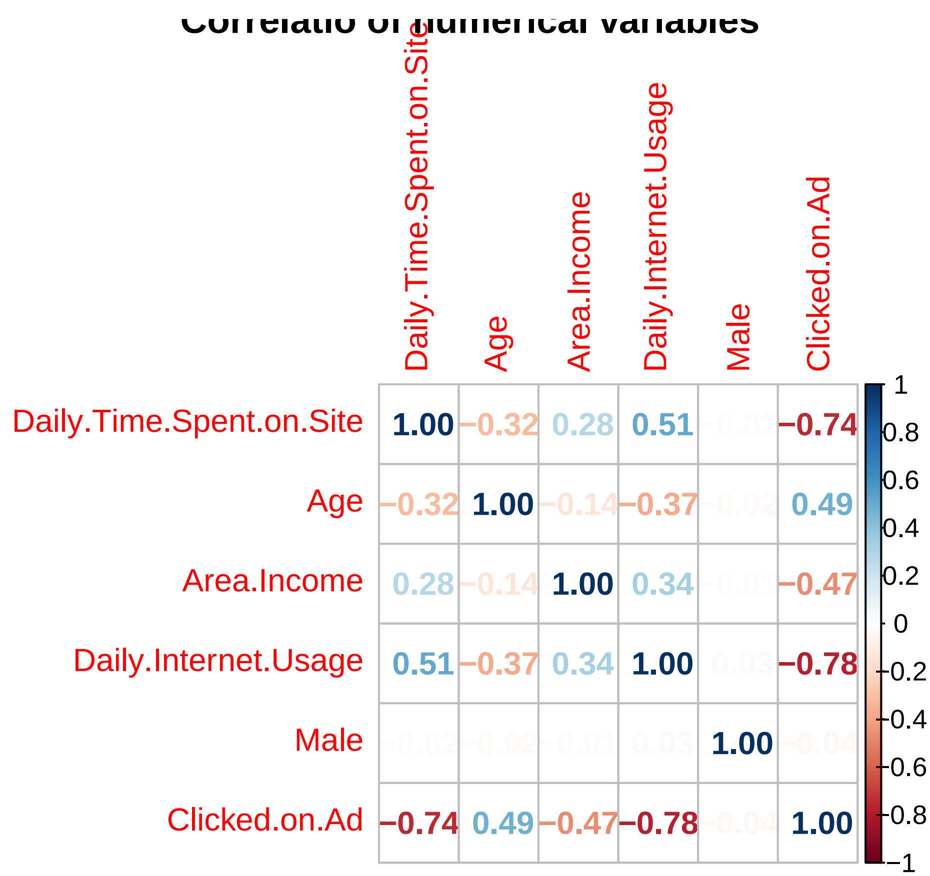
```
df.cor = cor(df[,sapply(df,is.numeric)],method = c('spearman'))
df.cor
```

```
##                          Daily.Time.Spent.on.Site         Age Area.Income
## Daily.Time.Spent.on.Site               1.00000000 -0.31686155  0.28313439
## Age                                   -0.31686155  1.00000000 -0.13595396
## Area.Income                            0.28313439 -0.13595396  1.00000000
## Daily.Internet.Usage                   0.51410805 -0.37086395  0.33916021
## Male                                  -0.01592213 -0.02315468 -0.01436909
## Clicked.on.Ad                         -0.74487253  0.48633733 -0.46722440
##                          Daily.Internet.Usage        Male Clicked.on.Ad
## Daily.Time.Spent.on.Site           0.51410805 -0.01592213   -0.74487253
## Age                               -0.37086395 -0.02315468    0.48633733
## Area.Income                        0.33916021 -0.01436909   -0.46722440
## Daily.Internet.Usage               1.00000000  0.02820432   -0.77660702
## Male                               0.02820432  1.00000000   -0.03802747
## Clicked.on.Ad                     -0.77660702 -0.03802747    1.00000000
```

```
library(corrplot)
```

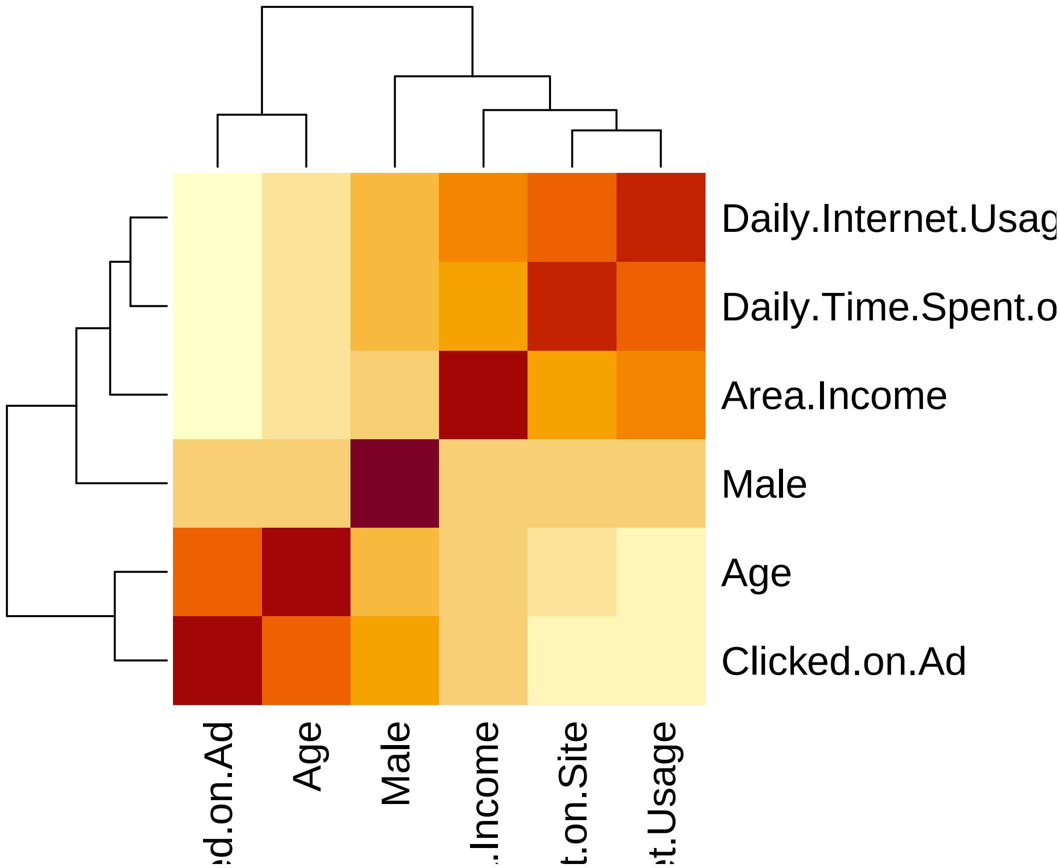
```
## corrplot 0.90 loaded
```

```
corrplot(df.cor,method="number",main="Correlatio of numerical variables")
```



Correlation plot
Heatmap

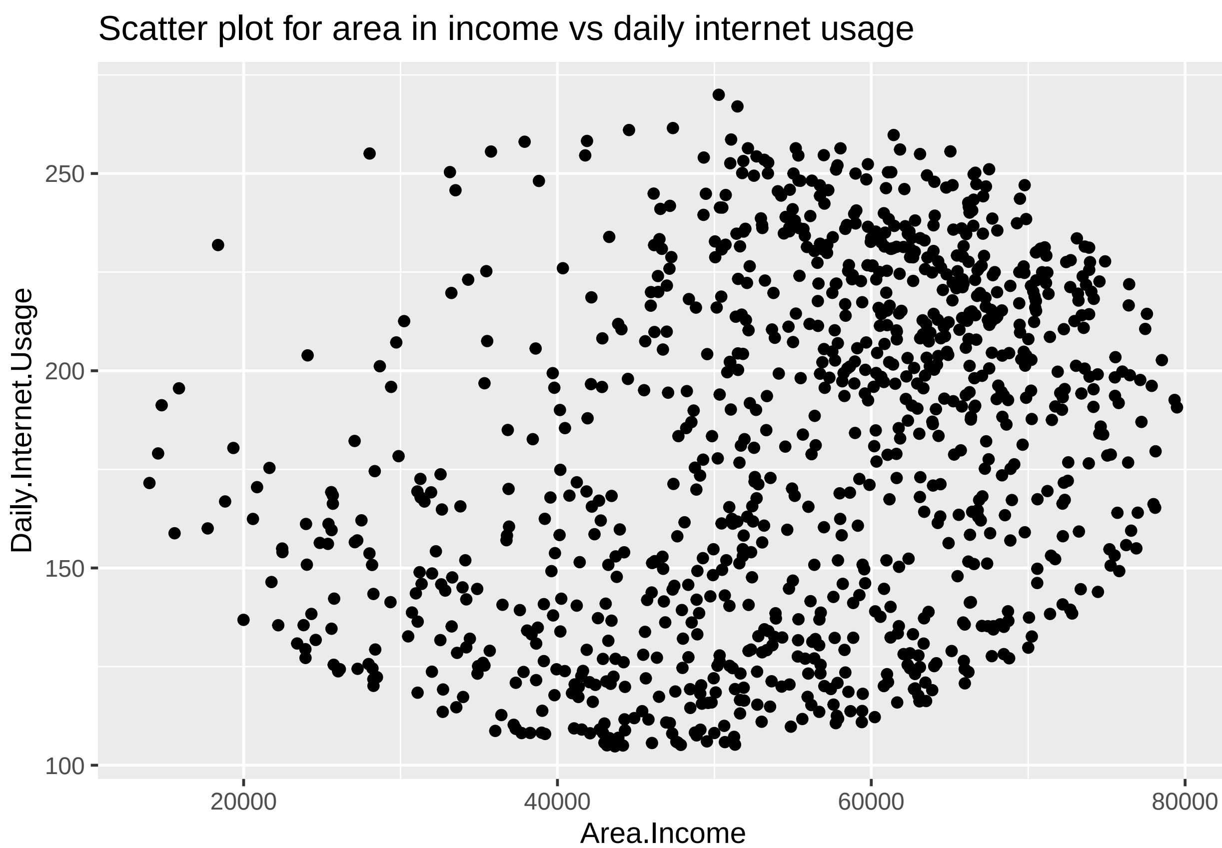
```
heatmap(x=df.cor)
```



###scatterplots

#Scatter plot for area in income vs daily internet usage

```
ggplot(df, aes(x = Area.Income, y = Daily.Internet.Usage )) +
  geom_point() + labs(title = 'Scatter plot for area in income vs daily internet usage')
```

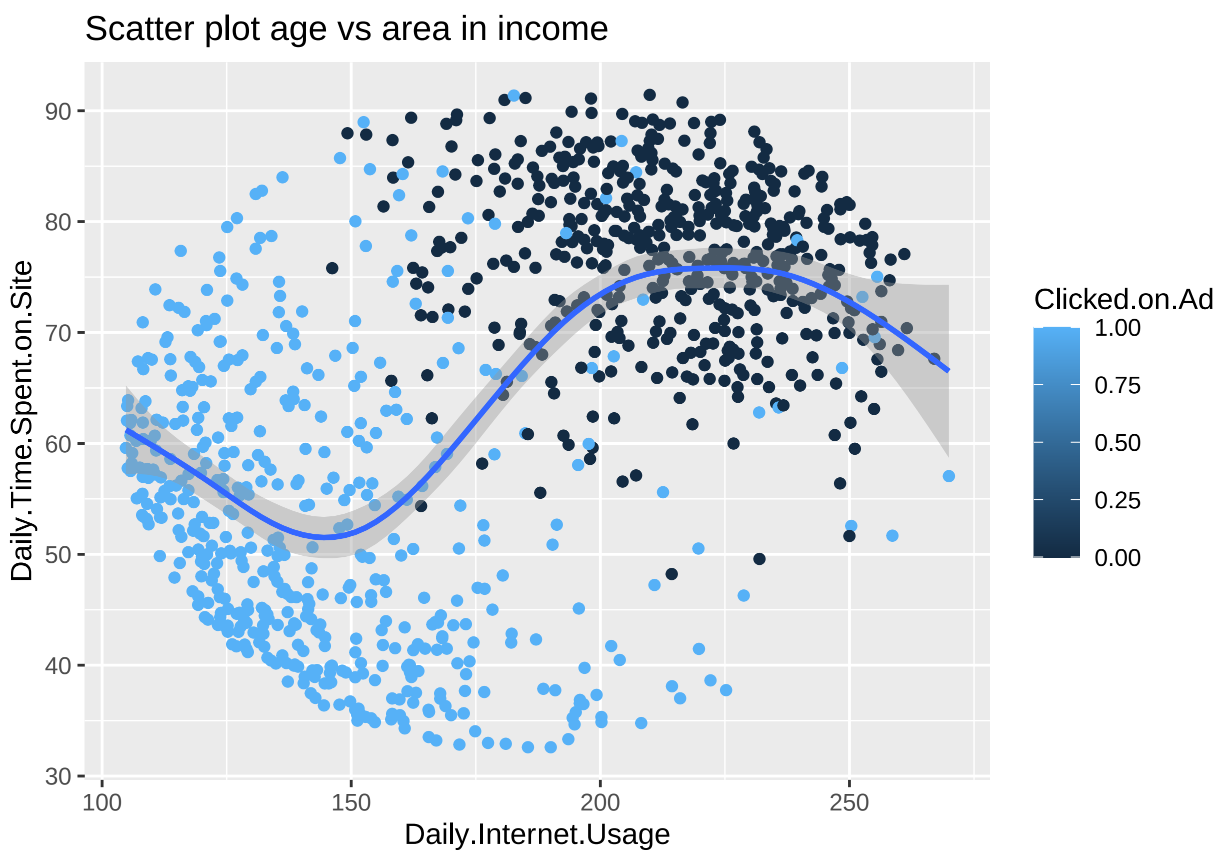



```r
#Scatter plot for age vs daily time spent on site
ggplot(df, aes(x = Age, y = Daily.Time.Spent.on.Site)) +
  geom_point() + labs(title = 'Scatter plot for age vs daily time spent on site')
```

Most people who spend time on site are between ages of 30-50

```
# Scatter plot of internet usage
ggplot(df, aes(x =Daily.Internet.Usage, y = Daily.Time.Spent.on.Site,color = Clicked.on.Ad)) +geom_poin
```

```
## 'geom_smooth()' using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



Modelling

```
library(superml)
```

```
## Warning: package 'superml' was built under R version 4.1.1

## Loading required package: R6

## Warning: package 'R6' was built under R version 4.1.1
```

```
print("Data before label encoding..\n")
```

```
## [1] "Data before label encoding..\n"
```

```
label <- LabelEncoder$new()
#Label encoding
df$City <- label$fit_transform(df$City)
#print(df$City)
df$Country <- label$fit_transform(df$Country)
#print(df$Country)
df$Ad.Topic.Line <- label$fit_transform(df$Ad.Topic.Line)
#print(df$Ad.Topic.Line)
```

KNN

```r
df2 <- subset(df, select = c(Daily.Time.Spent.on.Site,Age,Area.Income,Daily.Internet.Usage,Ad.Topic.Lin
names(df2)
```

```
## [1] "Daily.Time.Spent.on.Site" "Age"
## [3] "Area.Income"              "Daily.Internet.Usage"
## [5] "Ad.Topic.Line"            "City"
## [7] "Male"                     "Country"
## [9] "Clicked.on.Ad"
```

```r
set.seed(1234)
# Randomizing the rows, creates a uniform distribution of 150
random <- runif(150)
df_random <- df2[order(random),]
# Selecting the first 6 rows from iris_random
head(df_random)
```

```
##     Daily.Time.Spent.on.Site Age Area.Income Daily.Internet.Usage Ad.Topic.Line
## 7                      88.91  33    53852.85               208.36             6
## 64                     86.06  32    61601.05               178.92            63
## 73                     55.35  39    75509.61               153.17            72
## 98                     39.94  41    64927.19               156.30            97
## 101                    41.49  53    31947.65               169.18           100
## 110                    74.02  32    72272.90               210.54           109
##     City Male Country Clicked.on.Ad
## 7      6    0       6             0
## 64    63    1      52             0
## 73    72    1      58             1
## 98    97    0      76             1
## 101  100    0      78             1
## 110   35    0      44             0
```

```r
normal <- function(x) (
  return( ((x - min(x)) /(max(x)-min(x))) )
)
normal(1:9)
```

```
## [1] 0.000 0.125 0.250 0.375 0.500 0.625 0.750 0.875 1.000
```

```r
df_new <- as.data.frame(lapply(df_random[,-9], normal))
summary(df_new)
```

```
##  Daily.Time.Spent.on.Site      Age          Area.Income
##  Min.   :0.0000           Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.2986           1st Qu.:0.2500   1st Qu.:0.5170
##  Median :0.5959           Median :0.4000   Median :0.6706
##  Mean   :0.5425           Mean   :0.4138   Mean   :0.6244
##  3rd Qu.:0.7752           3rd Qu.:0.5750   3rd Qu.:0.7898
##  Max.   :1.0000           Max.   :1.0000   Max.   :1.0000
```

```
##  Daily.Internet.Usage Ad.Topic.Line       City              Male
##  Min.   :0.0000       Min.   :0.00   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.1903       1st Qu.:0.25   1st Qu.:0.2466   1st Qu.:0.0000
##  Median :0.4430       Median :0.50   Median :0.5000   Median :0.0000
##  Mean   :0.4427       Mean   :0.50   Mean   :0.4994   Mean   :0.4533
##  3rd Qu.:0.6633       3rd Qu.:0.75   3rd Qu.:0.7466   3rd Qu.:1.0000
##  Max.   :1.0000       Max.   :1.00   Max.   :1.0000   Max.   :1.0000
##     Country
##  Min.   :0.0000
##  1st Qu.:0.2173
##  Median :0.4252
##  Mean   :0.4584
##  3rd Qu.:0.6986
##  Max.   :1.0000
```

```
# Lets now create test and train data sets
train <- df_new[1:130,]
test <- df_new[131:150,]
train_sp <- df_random[1:130,9]
test_sp <- df_random[131:150,9]
```

```
# Lets build a model on it; cl is the class of the training data set and k is the no of neighbours to l
# in order to classify it accordingly
library(class)
require(class)
model <- knn(train= train,test=test, cl= train_sp,k=13)
table(factor(model))
```

```
##
##  0  1
##  7 13
```

```
fd<-table(test_sp,model)
fd
```

```
##        model
## test_sp  0  1
##       0  7  1
##       1  0 12
```

```
accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}
accuracy(fd)
```

```
## [1] 95
```

Knn has an accuracy score of 95%

```
library(caret)
```

***SVM**

```
## Warning: package 'caret' was built under R version 4.1.1

## Loading required package: lattice

## Warning: package 'lattice' was built under R version 4.1.1
```

```r
# So, 70% of the data is used for training and the remaining 30% is #for testing the model.
# - The "list" parameter is for whether to return a list or matrix.
# We are passing FALSE for not returning a list
#
intrain <- createDataPartition(y = df2$Clicked.on.Ad, p= 0.7, list = FALSE)
training <- df2[intrain,]
testing <- df2[-intrain,]
```

```r
dim(training);
```

```
## [1] 700   9
```

```r
dim(testing)
```

```
## [1] 300   9
```

```r
#Changing our target variable to factor
training[["Clicked.on.Ad"]] = factor(training[["Clicked.on.Ad"]])
```

```r
# The trainControl method will take three parameters:
# a) The "method" parameter defines the resampling method,
# in this demo we'll be using the repeatedcv or the repeated cross-validation method.
# b) The next parameter is the "number", this basically holds the number of resampling iterations.
# c) The "repeats " parameter contains the sets to compute for our repeated cross-validation.
# We are using setting number =10 and repeats =3
trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)
svm_Linear <- train(Clicked.on.Ad ~., data = training, method = "svmLinear",
trControl=trctrl,
preProcess = c("center", "scale"),
tuneLength = 10)
```

```r
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
##   0.9638095  0.927619
##
## Tuning parameter 'C' was held constant at a value of 1
```

```r
# We can use the predict() method for predicting results as shown below.
# We pass 2 arguements, our trained model and our testing data frame.
# ---
#
test_pred <- predict(svm_Linear, newdata = testing)
test_pred
```

```
##   [1] 0 0 0 1 0 0 1 1 1 1 0 1 0 1 0 0 0 1 1 0 1 1 1 1 0 0 1 0 0 0 0 0 1 1 1 0 1
##  [38] 1 1 1 0 1 0 0 0 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 0 1 0 1 1 1 1 0 1 1 0
##  [75] 0 0 1 0 0 1 1 0 0 0 1 0 0 0 1 1 0 0 0 0 1 1 0 0 1 0 1 0 0 0 0 1 0 0 1 0 1
## [112] 0 0 0 0 1 0 1 0 1 1 1 0 1 1 1 1 1 1 0 0 1 1 1 0 0 1 1 1 0 0 0 0 0 0 0 1 1
## [149] 1 1 0 0 1 1 0 1 0 1 0 0 0 0 0 1 0 1 0 0 1 0 1 1 1 1 0 1 1 0 1 1 0 0 1 0 0
## [186] 1 0 0 0 1 1 1 0 0 0 0 0 1 1 0 1 1 1 0 0 1 1 0 1 0 1 0 0 0 1 0 0 0 1 0 0 0
## [223] 0 0 1 1 0 1 1 1 1 0 0 1 0 0 0 1 1 0 0 1 0 1 1 0 0 0 0 1 1 1 1 0 1 0 0 0 0
## [260] 1 0 0 0 0 0 1 1 1 1 1 1 0 1 0 1 0 0 0 0 1 0 1 1 0 0 1 1 0 1 0 1 0 1 0 0 1
## [297] 1 0 1 1
## Levels: 0 1
```

```r
# Now checking for our accuracy of our model by using a confusion matrix
# ---
#
confusionMatrix(table(test_pred, testing$Clicked.on.Ad))
```

```
## Confusion Matrix and Statistics
##
##
## test_pred   0   1
##         0 148   5
##         1   2 145
##
##                Accuracy : 0.9767
##                  95% CI : (0.9525, 0.9906)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9533
##
##  Mcnemar's Test P-Value : 0.4497
##
##             Sensitivity : 0.9867
##             Specificity : 0.9667
##          Pos Pred Value : 0.9673
##          Neg Pred Value : 0.9864
##              Prevalence : 0.5000
##          Detection Rate : 0.4933
##    Detection Prevalence : 0.5100
##       Balanced Accuracy : 0.9767
##
```

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

SVM linear has an accuracy score of 97%

```r
library(tidyverse)
```

Naives bayes

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v tibble  3.1.4     v purrr   0.3.4
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1

## Warning: package 'tibble' was built under R version 4.1.1

## Warning: package 'readr' was built under R version 4.1.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x purrr::lift()   masks caret::lift()
```

```r
library(ggplot2)
library(caret)#confusionMatrix
library(caretEnsemble)
```

```
## Warning: package 'caretEnsemble' was built under R version 4.1.1

##
## Attaching package: 'caretEnsemble'

## The following object is masked from 'package:ggplot2':
##
##     autoplot
```

```r
library(psych)
```

```
##
## Attaching package: 'psych'

## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

```r
library(Amelia)#missmap
```

```
## Warning: package 'Amelia' was built under R version 4.1.1
```

```
## 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
## ## Refer to http://gking.harvard.edu/amelia/ for more information
## ##
```

```r
library(mice) #mice
```

```
## Warning: package 'mice' was built under R version 4.1.1
```

```
##
## Attaching package: 'mice'
```

```
## The following object is masked from 'package:stats':
##
##     filter
```

```
## The following objects are masked from 'package:base':
##
##     cbind, rbind
```

```r
library(GGally) #ggpairs
```

```
## Warning: package 'GGally' was built under R version 4.1.1
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```r
library(rpart)
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.1.1
```

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

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

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:psych':
##
##     outlier

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

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

```
library(tidyverse)
```

```
# describing the data
summary(df2)
```

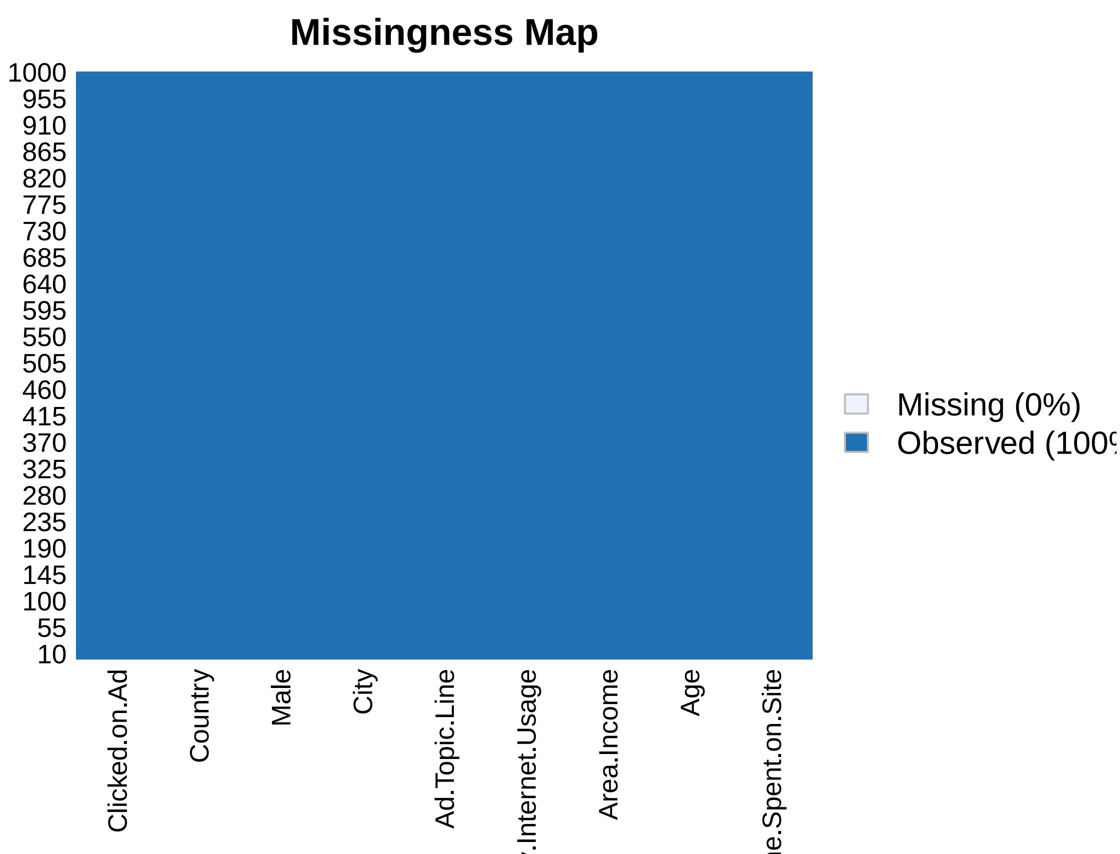
```
##  Daily.Time.Spent.on.Site      Age         Area.Income    Daily.Internet.Usage
##  Min.   :32.60            Min.   :19.00   Min.   :13996   Min.   :104.8
##  1st Qu.:51.36            1st Qu.:29.00   1st Qu.:47032   1st Qu.:138.8
##  Median :68.22            Median :35.00   Median :57012   Median :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       Clicked.on.Ad
##  Min.   :  0.0   Min.   :  0.0   Min.   :0.000   Min.   :  0.0   Min.   :0.0
##  1st Qu.:249.8   1st Qu.:234.8   1st Qu.:0.000   1st Qu.: 52.0   1st Qu.:0.0
##  Median :499.5   Median :473.5   Median :0.000   Median :107.0   Median :0.5
##  Mean   :499.5   Mean   :477.9   Mean   :0.481   Mean   :108.9   Mean   :0.5
##  3rd Qu.:749.2   3rd Qu.:721.2   3rd Qu.:1.000   3rd Qu.:162.0   3rd Qu.:1.0
##  Max.   :999.0   Max.   :968.0   Max.   :1.000   Max.   :236.0   Max.   :1.0
```

```
# We convert the output variable into a categorical variable
df2$Clicked.on.Ad <- factor(df2$Clicked.on.Ad)
df2$Clicked.on.Ad
```

```
##    [1] 0 0 0 0 0 0 0 1 0 0 1 0 1 0 1 1 1 0 1 1 0 0 1 0 1 0 1 1 1 0 0 0 1 1 1 0 1
##   [38] 0 1 1 0 0 0 0 0 1 0 0 1 1 0 0 1 1 1 0 1 1 0 1 0 0 0 0 1 0 1 1 0 1 1 0 1 1
##   [75] 1 0 1 0 1 1 0 0 1 1 0 1 0 1 1 1 1 1 0 1 1 0 1 1 1 0 1 0 0 0 0 0 0 1 1 0 1
##  [112] 1 0 1 0 0 1 1 1 1 0 0 0 1 1 0 1 0 0 0 1 1 1 0 1 1 1 1 0 0 0 1 1 0 0 1 1 1
##  [149] 1 1 0 0 1 0 0 0 1 1 0 1 0 0 0 0 1 1 1 0 1 0 1 0 0 0 1 0 1 0 1 0 1 1 1 0 0
##  [186] 1 1 0 1 1 1 1 1 1 0 1 1 0 0 0 0 0 1 0 0 1 0 0 1 1 0 1 0 1 0 1 1 1 1 1 0 0
##  [223] 1 1 0 1 1 1 0 0 0 1 1 1 1 1 1 0 1 0 1 1 0 0 0 0 1 1 1 1 0 1 0 1 1 0 0 1 0
##  [260] 1 0 1 1 1 0 1 1 0 1 0 1 0 0 0 0 1 0 0 0 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 0 0
##  [297] 0 0 0 0 0 1 1 1 1 1 0 0 0 1 0 0 1 0 0 1 0 0 0 1 1 0 0 0 0 1 1 0 0 1 0 0 1
##  [334] 0 0 1 0 0 0 0 1 1 0 0 1 0 0 1 0 1 0 0 0 0 1 0 1 1 1 0 1 1 0 1 0 1 0 0 0 0
##  [371] 1 1 0 1 0 0 0 1 1 0 0 1 0 0 1 0 0 1 0 1 0 0 0 0 1 0 1 1 0 0 1 0 1 0 1 0 1
##  [408] 1 1 1 1 0 0 1 0 1 1 0 0 0 1 0 1 1 1 1 1 0 1 0 0 0 1 0 0 1 0 0 1 0 1 0 1 1
##  [445] 1 0 1 0 1 0 1 1 0 0 1 0 1 0 1 0 1 1 0 1 0 1 1 1 1 0 1 0 0 0 1 0 0 1 1 1 0
##  [482] 0 0 1 1 1 0 0 1 0 1 1 0 1 1 0 0 1 0 1 1 0 0 1 1 0 0 1 1 0 1 0 0 1 0 1 0 1
##  [519] 1 1 1 1 0 1 0 0 1 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 1 1 1
##  [556] 0 1 0 0 0 1 1 0 0 1 0 1 0 0 0 1 0 0 1 1 1 1 0 0 0 1 1 1 1 1 0 0 1 0 1 1 1
```

```
##  [593] 0 0 1 1 0 0 0 1 1 1 1 0 1 1 0 0 1 1 1 1 0 0 0 1 1 0 1 0 0 0 1 0 0 1 0 1 1
##  [630] 0 0 0 0 1 1 1 1 0 1 0 1 0 0 0 0 1 1 1 0 0 0 0 0 0 0 1 0 0 0 0 1 1 1 1 0 1
##  [667] 0 0 1 1 0 1 0 1 0 0 1 1 0 1 0 1 1 0 1 0 0 0 0 0 0 0 1 1 0 0 1 0 0 0 0 1 1
##  [704] 0 0 0 1 0 1 1 1 0 0 1 0 1 1 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1 0
##  [741] 1 0 0 1 1 1 1 1 1 0 1 0 0 0 0 0 1 1 1 1 0 0 1 1 1 1 1 1 1 0 0 0 0 1 1 1 1
##  [778] 0 1 0 1 1 0 0 1 1 0 1 0 1 1 1 0 1 1 0 0 0 0 0 1 1 1 1 1 0 1 1 1 1 1 0 0 0
##  [815] 0 0 1 1 0 0 1 0 1 0 0 0 0 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0 0 0 1 1 0 0 1 0
##  [852] 1 1 0 1 1 0 0 1 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 1 0 0 0 1 0 0 1 0 1 1 1
##  [889] 0 1 0 1 1 0 0 0 0 1 1 1 1 1 1 0 0 0 1 0 1 0 1 1 1 0 1 1 1 0 0 0 0 1 1 1 1
##  [926] 1 0 0 0 1 0 1 1 1 0 0 1 1 1 0 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 0 1 1 0 0 0 1 0
##  [963] 0 0 0 1 1 0 1 1 1 1 1 1 0 1 1 1 1 0 0 1 0 1 0 0 1 0 1 0 0 1 1 1 0 1 1 1 1 0
## [1000] 1
## Levels: 0 1
```

```r
# We visualize our dataset by checking how many missing values
missmap(df2)
```




```r
# Splitting data into training and test data sets
# ---
#
indxTrain <- caret::createDataPartition(y = df2$Clicked.on.Ad,p = 0.75,list = FALSE)
training <- df2[indxTrain,]
testing1 <- df2[-indxTrain,]
```

```r
# Checking dimensions of the split
prop.table(table(df2$Clicked.on.Ad)) * 100
```

```
##
##  0  1
## 50 50
```

```r
prop.table(table(df2$Clicked.on.Ad)) * 100
```

```
##
##  0  1
## 50 50
```

```r
prop.table(table(df2$Clicked.on.Ad)) * 100
```

```
##
##  0  1
## 50 50
```

```r
# Comparing the clicked on add of the training and testing phase
# Creating objects x which holds the predictor variables and y which holds the response variables
# ---
#
x = training[,-9]
colSums(is.na(x))
```

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

```r
y = training$Clicked.on.Ad
```

```r
# Loading our inbuilt e1071 package that holds the Naive Bayes function.
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 4.1.1
```

```r
# Now building our model
#model = train(x,y,'nb',trControl=trainControl(method='cv',number=10))
```

```r
# Model Evalution
# Predicting our testing set
#
#Predict <- predict(model,newdata = testing1 )
```

```r
# Getting the confusion matrix to see accuracy value and other parameter values
#confusionMatrix(Predict, testing1$Clicked.on.Ad)
```

Naive bayes has an accuracy score of 95%

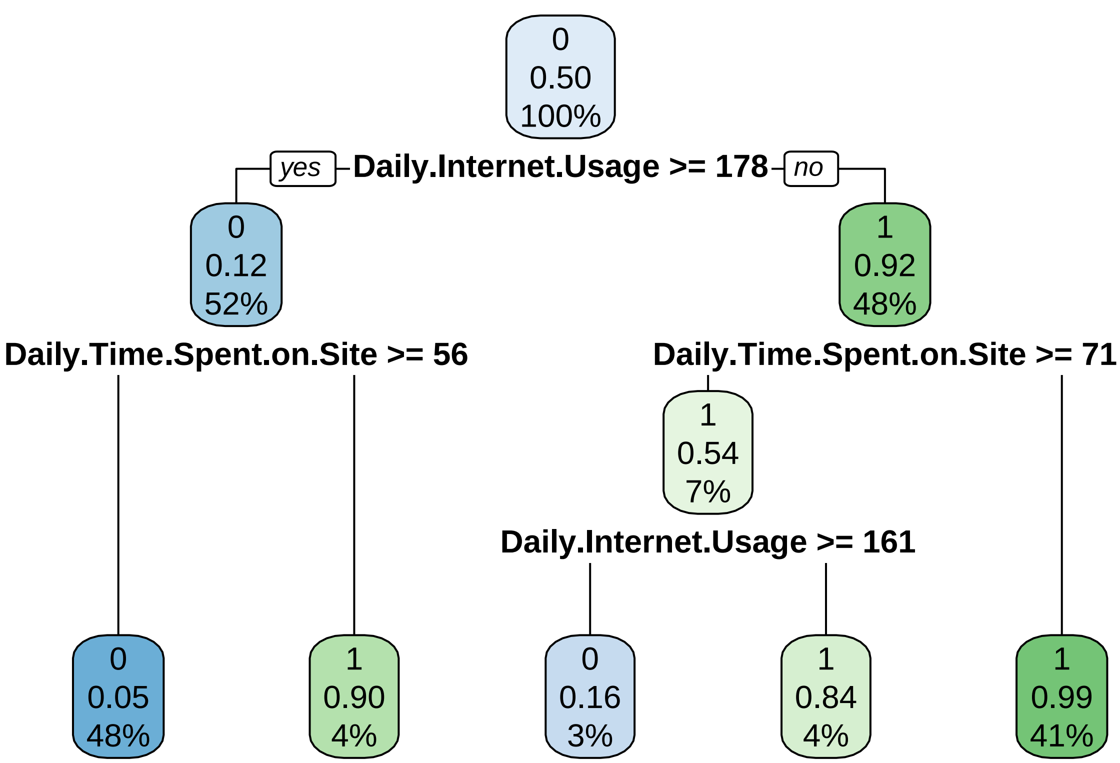
```r
set.seed(100)
id<- sample(2,nrow(df2),prob =c(0.7,0.3),replace = T)
traindf<- df2[id==1,]
testdf<- df2[id==2,]
```

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

Decision trees

```
## Warning: package 'rpart.plot' was built under R version 4.1.1
```

```r
m <- rpart(Clicked.on.Ad ~., data = df2,
           method = "class")
rpart.plot(m)
```



```r
p <- predict(m,df2, type = "class")
cm <- table(p,df$Clicked.on.Ad)
cm
```

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

```r
accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}
accuracy(cm)
```

```
## [1] 95.7
```

Decision tree has an accuracy score of 96%

Hyperparameter turning for decision tree Training the decision tree model

```r
df$Timestamp<- NULL
names(df)
```

```
## [1] "Daily.Time.Spent.on.Site" "Age"
## [3] "Area.Income"              "Daily.Internet.Usage"
## [5] "Ad.Topic.Line"            "City"
## [7] "Male"                     "Country"
## [9] "Clicked.on.Ad"
```

```r
names(df)
```

```
## [1] "Daily.Time.Spent.on.Site" "Age"
## [3] "Area.Income"              "Daily.Internet.Usage"
## [5] "Ad.Topic.Line"            "City"
## [7] "Male"                     "Country"
## [9] "Clicked.on.Ad"
```

```r
make.names(names(df))
```

```
## [1] "Daily.Time.Spent.on.Site" "Age"
## [3] "Area.Income"              "Daily.Internet.Usage"
## [5] "Ad.Topic.Line"            "City"
## [7] "Male"                     "Country"
## [9] "Clicked.on.Ad"
```

```r
colnames(df) <- make.names(colnames(df),unique = T)
```

```r
library(mlr)
```

```
## Warning: package 'mlr' was built under R version 4.1.1
```

```
## Loading required package: ParamHelpers

## Warning: package 'ParamHelpers' was built under R version 4.1.1

## Warning message: 'mlr' is in 'maintenance-only' mode since July 2019.
## Future development will only happen in 'mlr3'
## (<https://mlr3.mlr-org.com>). Due to the focus on 'mlr3' there might be
## uncaught bugs meanwhile in {mlr} - please consider switching.

##
## Attaching package: 'mlr'

## The following object is masked from 'package:e1071':
##
##     impute

## The following object is masked from 'package:caret':
##
##     train
```

```r
dfTask <- makeClassifTask(data = df, target = "Clicked.on.Ad")
tree <- makeLearner("classif.rpart")
```

```r
# Printing available rpart hyperparameters
ls()
```

```
##  [1] "accuracy"   "cm"         "covarince"  "df"         "df.cor"
##  [6] "df_new"     "df_random"  "df2"        "dfTask"     "fd"
## [11] "frequency"  "getmode"    "id"         "indxTrain"  "intrain"
## [16] "label"      "m"          "model"      "normal"     "p"
## [21] "random"     "svm_Linear" "test"       "test_pred"  "test_sp"
## [26] "testdf"     "testing"    "testing1"   "train"      "train_sp"
## [31] "traindf"    "training"   "trctrl"     "tree"       "x"
## [36] "y"
```

```r
getParamSet(tree)
```

```
##                    Type len  Def   Constr Req Tunable Trafo
## minsplit        integer   -   20 1 to Inf   -    TRUE     -
## minbucket       integer   -    - 1 to Inf   -    TRUE     -
## cp              numeric   - 0.01   0 to 1   -    TRUE     -
## maxcompete      integer   -    4 0 to Inf   -    TRUE     -
## maxsurrogate    integer   -    5 0 to Inf   -    TRUE     -
## usesurrogate   discrete   -    2    0,1,2   -    TRUE     -
## surrogatestyle discrete   -    0      0,1   -    TRUE     -
## maxdepth        integer   -   30  1 to 30   -    TRUE     -
## xval            integer   -   10 0 to Inf   -   FALSE     -
## parms           untyped   -    -        -   -    TRUE     -
```

Defining the hyperparameter space for tuning

```r
treeParamSpace <- makeParamSet(
  makeIntegerParam("minsplit", lower = 5, upper = 20),
  makeIntegerParam("minbucket", lower = 3, upper = 10),
  makeNumericParam("cp", lower = 0.01, upper = 0.1),
  makeIntegerParam("maxdepth", lower = 3, upper = 10))
```

```r
# Defining the random search
randSearch <- makeTuneControlRandom(maxit = 200)
cvForTuning <- makeResampleDesc("CV", iters = 5)
```

Performing hyperparameter

```r
library(parallelMap)
```

```
## Warning: package 'parallelMap' was built under R version 4.1.1
```

```r
library(detector)
```

```
## Warning: package 'detector' was built under R version 4.1.1
```

```
##
## Attaching package: 'detector'
```

```
## The following object is masked from 'package:purrr':
##
##     detect
```

```r
library(parallel)
parallelStartSocket(cpus = detectCores())
```

```
## Starting parallelization in mode=socket with cpus=4.
```

```r
tunedTreePars <- tuneParams(tree, task = dfTask,
                            resampling = cvForTuning,
                            par.set = treeParamSpace,
                            control = randSearch)
```

```
## [Tune] Started tuning learner classif.rpart for parameter set:
```

```
##              Type len Def      Constr Req Tunable Trafo
## minsplit  integer   -   -     5 to 20   -    TRUE     -
## minbucket integer   -   -     3 to 10   -    TRUE     -
## cp        numeric   -   - 0.01 to 0.1   -    TRUE     -
## maxdepth  integer   -   -     3 to 10   -    TRUE     -
```

```
## With control class: TuneControlRandom
```

```
## Imputation value: 1

## Exporting objects to slaves for mode socket: .mlr.slave.options

## Mapping in parallel: mode = socket; level = mlr.tuneParams; cpus = 4; elements = 200.

## [Tune] Result: minsplit=10; minbucket=5; cp=0.0121; maxdepth=8 : mmce.test.mean=0.0570000
```

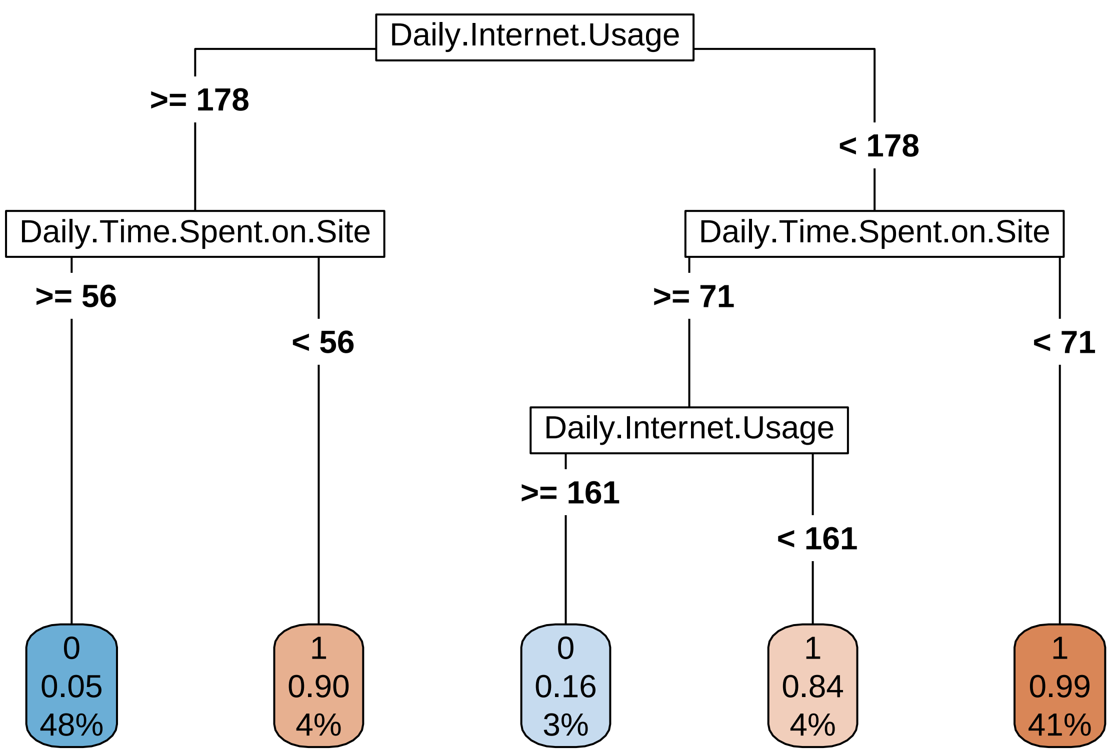
```
parallelStop()
```

```
## Stopped parallelization. All cleaned up.
```

Training the model with the tuned hyperparameters

```
# Training the final tuned model
tunedTree <- setHyperPars(tree, par.vals = tunedTreePars$x)
tunedTreeModel <- train(tunedTree, dfTask)
```

```
treeModelData <- getLearnerModel(tunedTreeModel)
rpart.plot(treeModelData, roundint = FALSE,
box.palette = "BuBn",
type = 5)
```



Exploring the model

```r
printcp(treeModelData, digits = 3)
```

```
##
## Classification tree:
## rpart::rpart(formula = f, data = d, xval = 0, minsplit = 10,
##     minbucket = 5, cp = 0.0120877194521017, maxdepth = 8)
##
## Variables actually used in tree construction:
## [1] Daily.Internet.Usage     Daily.Time.Spent.on.Site
##
## Root node error: 500/1000 = 0.5
##
## n= 1000
##
##       CP nsplit rel error
## 1 0.8040      0     1.000
## 2 0.0680      1     0.196
## 3 0.0210      2     0.128
## 4 0.0121      4     0.086
```

```r
# Cross-validating the model-building process
outer <- makeResampleDesc("CV", iters = 5)
treeWrapper <- makeTuneWrapper("classif.rpart", resampling = cvForTuning,
                               par.set = treeParamSpace,
                               control = randSearch)
parallelStartSocket(cpus = detectCores())
```

```
## Starting parallelization in mode=socket with cpus=4.
```

```r
cvWithTuning <- resample(treeWrapper,dfTask, resampling = outer)
```

```
## Exporting objects to slaves for mode socket: .mlr.slave.options

## Resampling: cross-validation

## Measures:             mmce

## Mapping in parallel: mode = socket; level = mlr.resample; cpus = 4; elements = 5.

##

## Aggregated Result: mmce.test.mean=0.0570000

##
```

```r
parallelStop()
```

```
## Stopped parallelization. All cleaned up.
```

```r
# Extracting the cross-validation result
cvWithTuning
```

```
## Resample Result
## Task: df
## Learner: classif.rpart.tuned
## Aggr perf: mmce.test.mean=0.0570000
## Runtime: 94.3075
```

```r
library(randomForest)
dfforest <- randomForest(Clicked.on.Ad ~ Daily.Time.Spent.on.Site+Age+
             Area.Income+Daily.Internet.Usage+Ad.Topic.Line+
             City+Country+Male,data=traindf)
dfforest
```

Random Forest

```
##
## Call:
##  randomForest(formula = Clicked.on.Ad ~ Daily.Time.Spent.on.Site +      Age + Area.Income + Daily.In
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 3.83%
## Confusion matrix:
##     0   1 class.error
## 0 331  14  0.04057971
## 1  12 322  0.03592814
```

```r
predforest <- predict(dfforest,testdf,type="class")
predforest
```

```
##    7   12   15   22   24   27   28   32   34   36   39   42   43   44   47   48
##    0    0    1    0    0    1    1    0    1    0    1    0    0    0    0    0
##   63   74   77   78   79   83   84   87   90   91   95  100  107  108  111  112
##    0    1    1    0    1    1    1    0    1    1    1    0    0    1    1    0
##  115  117  118  120  127  128  132  133  144  146  147  152  156  157  158  161
##    0    1    1    1    1    0    1    1    0    1    1    0    0    1    1    0
##  168  169  170  174  175  177  182  183  190  191  194  200  202  203  204  206
##    0    1    0    0    1    1    1    1    1    1    1    0    0    1    0    1
##  207  212  217  218  224  233  238  239  240  242  244  246  252  255  256  260
##    0    1    1    1    1    1    1    1    0    1    0    1    1    1    0    1
##  262  264  265  266  269  270  272  276  287  288  298  301  303  306  310  312
##    1    1    0    1    1    0    0    1    1    0    0    0    1    0    1    0
##  318  320  324  326  327  330  339  341  348  350  357  359  363  369  370  371
##    0    1    0    1    1    1    0    1    1    0    1    1    0    0    0    1
##  380  385  389  393  399  405  408  409  414  416  418  420  421  423  424  428
##    0    1    0    0    0    1    1    1    1    1    0    0    1    1    1    0
```

```
##  431  436  437  438  440  446  453  454  457  462  463  464  466  475  476  477
##    0    1    1    0    0    0    0    0    1    1    0    1    1    1    0    0
##  483  488  491  499  502  507  509  515  520  522  524  528  529  532  533  534
##    0    0    1    0    0    0    1    0    1    1    1    0    1    1    0    0
##  539  540  544  545  546  549  550  552  554  556  557  561  564  565  567  569
##    0    0    1    0    1    0    0    0    1    0    1    1    1    1    1    0
##  570  573  577  581  586  588  590  594  595  596  597  606  611  614  621  622
##    0    0    1    1    0    1    1    0    1    1    0    1    1    0    0    0
##  626  628  638  639  642  643  648  651  654  655  657  658  663  664  670  681
##    1    1    0    1    0    0    1    0    0    0    0    0    1    1    1    0
##  684  686  687  688  691  692  696  697  700  703  709  712  713  717  723  725
##    0    0    0    0    0    0    0    1    0    0    1    0    0    1    1    0
##  727  728  738  739  740  742  743  745  747  750  760  762  768  771  774  776
##    0    0    1    0    0    0    0    1    0    1    1    0    1    0    1    1
##  777  783  785  790  791  800  805  808  810  811  814  817  819  821  824  825
##    1    0    1    1    1    0    1    1    1    1    0    1    0    1    0    0
##  826  830  834  843  846  847  851  856  857  861  864  865  868  871  872  875
##    0    1    1    0    1    1    0    1    0    0    0    0    0    1    0    0
##  876  877  880  882  885  888  889  890  891  892  905  906  915  921  924  928
##    1    1    0    0    0    1    0    1    0    1    0    0    1    0    1    0
##  930  932  935  936  938  940  941  942  945  946  948  951  952  954  955  956
##    1    1    0    0    1    0    1    1    1    0    1    1    1    1    1    1
##  957  960  963  966  968  970  974  976  978  981  982  985  986  990  994  996
##    1    0    0    1    0    1    0    1    1    1    0    0    1    0    0    0
## 1000
##    1
## Levels: 0 1
```

```
confusionMatrix(table(predforest,testdf$Clicked.on.Ad))
```

```
## Confusion Matrix and Statistics
##
##
## predforest   0   1
##          0 149   8
##          1   6 158
##
##                Accuracy : 0.9564
##                  95% CI : (0.9279, 0.976)
##     No Information Rate : 0.5171
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9127
##
##  Mcnemar's Test P-Value : 0.7893
##
##             Sensitivity : 0.9613
##             Specificity : 0.9518
##          Pos Pred Value : 0.9490
##          Neg Pred Value : 0.9634
##              Prevalence : 0.4829
##          Detection Rate : 0.4642
##    Detection Prevalence : 0.4891
##       Balanced Accuracy : 0.9565
```

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

```r
ranTree=table(predforest,testdf$Clicked.on.Ad)
ranTree
```

```
##
## predforest   0   1
##          0 149   8
##          1   6 158
```

```r
accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}
accuracy(ranTree)
```

```
## [1] 95.63863
```

```r
importance(dfforest)
```

```
##                          MeanDecreaseGini
## Daily.Time.Spent.on.Site       102.519899
## Age                             36.806715
## Area.Income                     42.403856
## Daily.Internet.Usage           134.176101
## Ad.Topic.Line                    7.270005
## City                             7.212031
## Country                          7.176939
## Male                             1.344746
```

```r
important <- importance(dfforest)
important
```

```
##                          MeanDecreaseGini
## Daily.Time.Spent.on.Site       102.519899
## Age                             36.806715
## Area.Income                     42.403856
## Daily.Internet.Usage           134.176101
## Ad.Topic.Line                    7.270005
## City                             7.212031
## Country                          7.176939
## Male                             1.344746
```

```r
Important_Features <- data.frame(Feature = row.names(important), Importance = important[, 1])
Important_Features
```

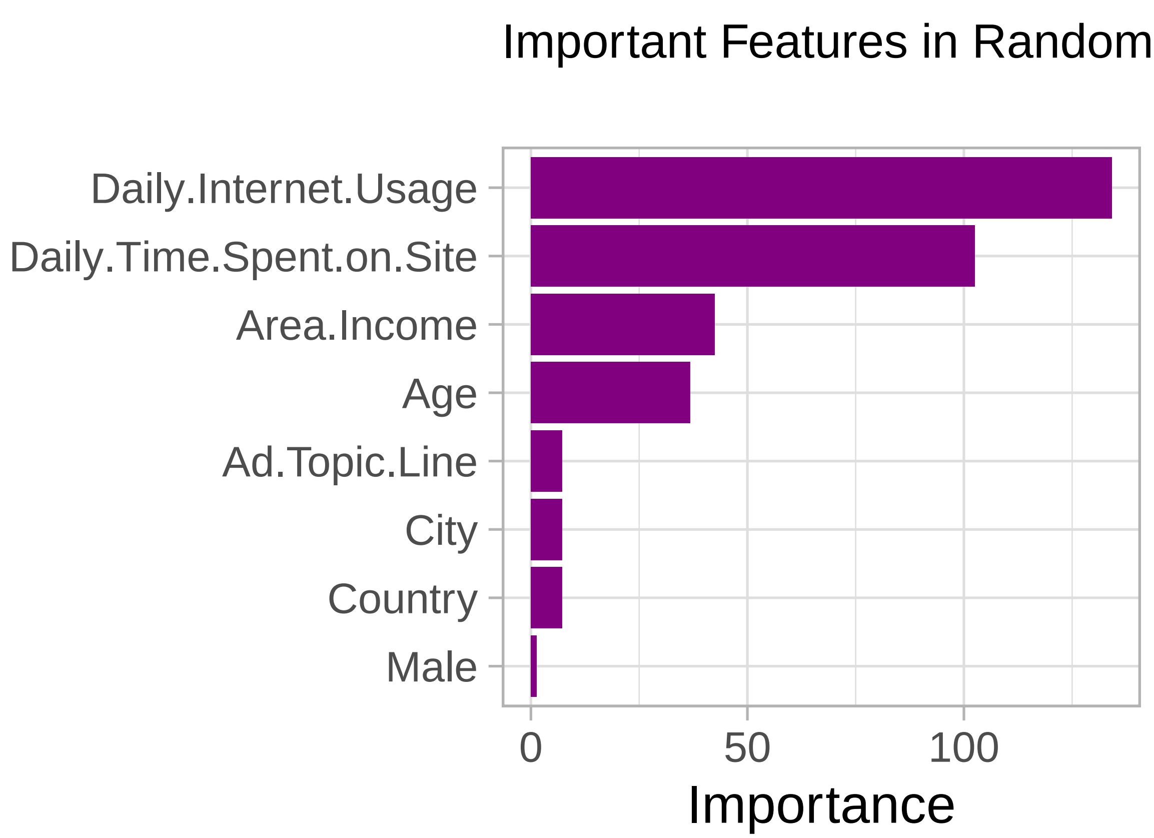
```
##                                           Feature Importance
## Daily.Time.Spent.on.Site Daily.Time.Spent.on.Site 102.519899
## Age                                           Age  36.806715
## Area.Income                           Area.Income  42.403856
## Daily.Internet.Usage         Daily.Internet.Usage 134.176101
```

```
## Ad.Topic.Line                         Ad.Topic.Line    7.270005
## City                                           City    7.212031
## Country                                     Country    7.176939
## Male                                           Male    1.344746
```

```
plot_ <- ggplot(Important_Features,
    aes(x= reorder(Feature,
Importance) , y = Importance) ) +
geom_bar(stat = "identity",
        fill = "#800080") +
coord_flip() +
theme_light(base_size = 20) +
xlab("") +
ylab("Importance")+
ggtitle("Important Features in Random Forest\n") +
theme(plot.title = element_text(size=18))
ggsave("important_features.png",
      plot_)
```

```
## Saving 6.5 x 4.5 in image
```

```
plot_
```



Conclusions

1.) Knn model has an accuracy score of 95%,svm has an accuracy score of 97% ,naive bayes has an accuracy score of 95% , decision tree has an accuracy score of 96%.

2.) The age between of 28 and 48 record the highest ad click on the site

3.) Tunisia ,Italy and san marino are the 3 top countries with the highest internet usage.

4.) Myanmar,Nauru and Grenad spend the most time on the site.

Recommendations

1.) Svm should be the best method to be used for comparison.

2.) The ads posted on the client site should be more relevant to this demographic between late twenties and early forties.