Code

# R Notebook

Stephanie Omwanda

# 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.

# Metric of Success

> The accuracy score of the model will be used to measure the model's predictive power.

# Loading and cleaning the dataset

Hide

```
# load libraries
library(readr) # provides a faster and friendly way to read rectangular data like
csvs
library(dplyr) # provides a flexible grammar of data manipulation
library(tinytex)
theme_set(theme_classic())
options(warn = -1)
```

Hide

```
# Loading the csv file
df = read_csv('advertising.csv')
```

```
Parsed with column specification:
cols(
  `Daily Time Spent on Site` = [32mcol_double()[39m,
  Age = [32mcol_double()[39m,
  `Area Income` = [32mcol_double()[39m,
  `Daily Internet Usage` = [32mcol_double()[39m,
  `Ad Topic Line` = [31mcol_character()[39m,
  City = [31mcol_character()[39m,
  Male = [32mcol_double()[39m,
  Country = [31mcol_character()[39m,
  Timestamp = [34mcol_datetime(format = "")[39m,
  `Clicked on Ad` = [32mcol_double()[39m
)
```

Hide

```
# Previewing the first five rows of the dataframe
head(df)
```

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

6 rows | 1-4 of 10 columns

Hide

```
# show information on dataset
str(df)
```

```
Classes ‘spec_tbl_df’, ‘tbl_df’, ‘tbl’ and 'data.frame':    1000 obs. of  10 varia
bles:
 $ Daily Time Spent on Site: num  69 80.2 69.5 74.2 68.4 ...
 $ Age                     : num  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 standardization" "Organic bottom-line service-desk" "Triple-buffered reci
procal time-frame" ...
 $ City                    : chr  "Wrightburgh" "West Jodi" "Davidton" "West Terri
furt" ...
 $ Male                    : num  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           : num  0 0 0 0 0 0 0 1 0 0 ...
 - attr(*, "spec")=
  .. cols(
  ..   `Daily Time Spent on Site` = [32mcol_double()[39m,
  ..   Age = [32mcol_double()[39m,
  ..   `Area Income` = [32mcol_double()[39m,
  ..   `Daily Internet Usage` = [32mcol_double()[39m,
  ..   `Ad Topic Line` = [31mcol_character()[39m,
  ..   City = [31mcol_character()[39m,
  ..   Male = [32mcol_double()[39m,
  ..   Country = [31mcol_character()[39m,
  ..   Timestamp = [34mcol_datetime(format = "")[39m,
  ..   `Clicked on Ad` = [32mcol_double()[39m
  .. )
```

Hide

```
# checking for the statistical 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
 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 10:34:06   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
```

Hide

```
# determine the dimensions of the dataset
dim(df)
```

```
[1] 1000   10
```

- We can see from the chunk above that the dataset contains 1000 observations and 10 variables

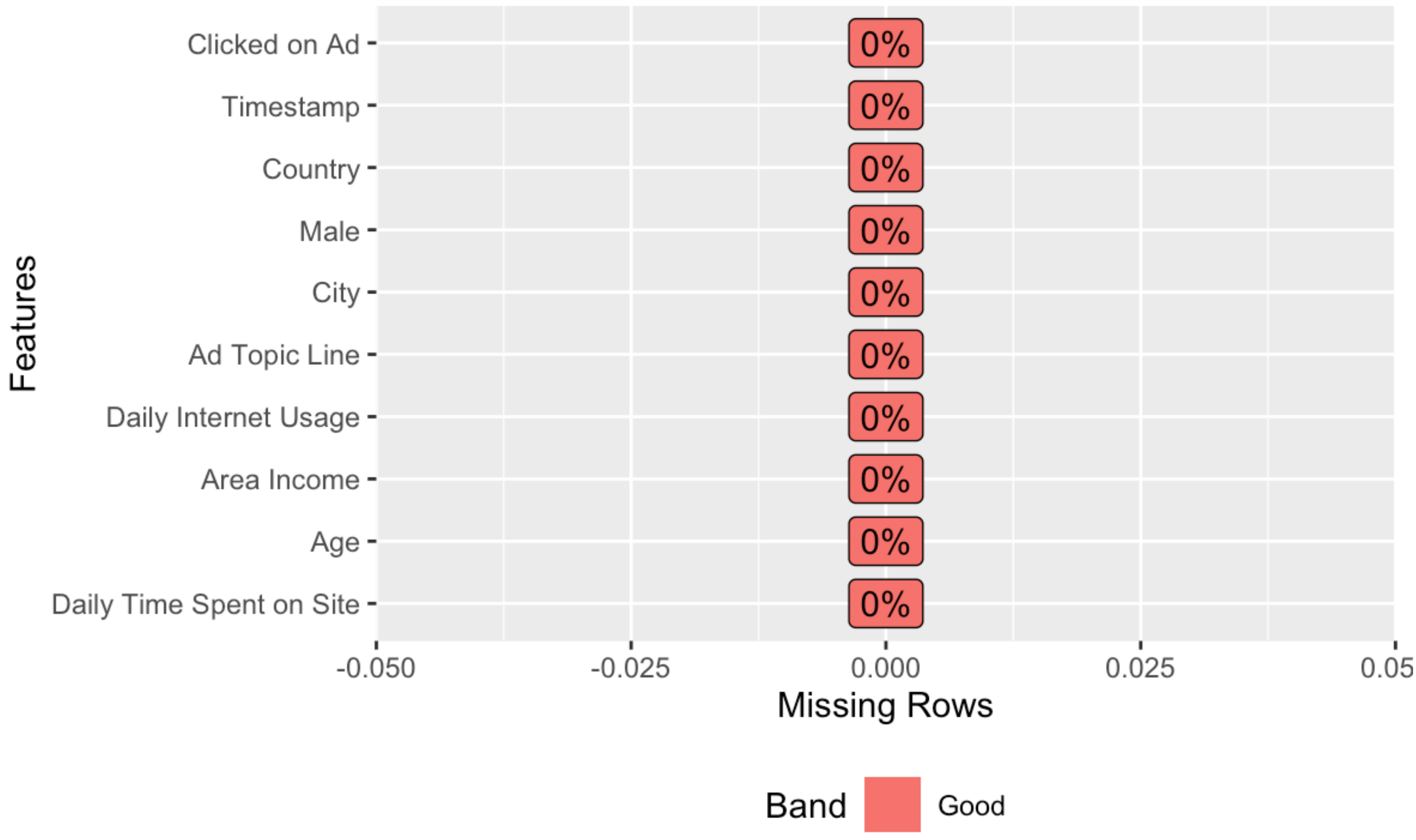
Hide

```
# checking if there exists null values by calculating the sum of the null values p
er column
colSums((is.na(df)))
```

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

Hide

```
# a plot showing missing values
library(DataExplorer) # simplifies and automates EDA processes and aids in report generation

plot_missing(df)
```




- From the two chunks above we can see that our dataset is void of null values

Hide

```
# checking for duplicates in the dataset by assigning a variable 'duplicates'
duplicates <- df[duplicated(df),]
duplicates
```

0 rows | 1-8 of 10 columns

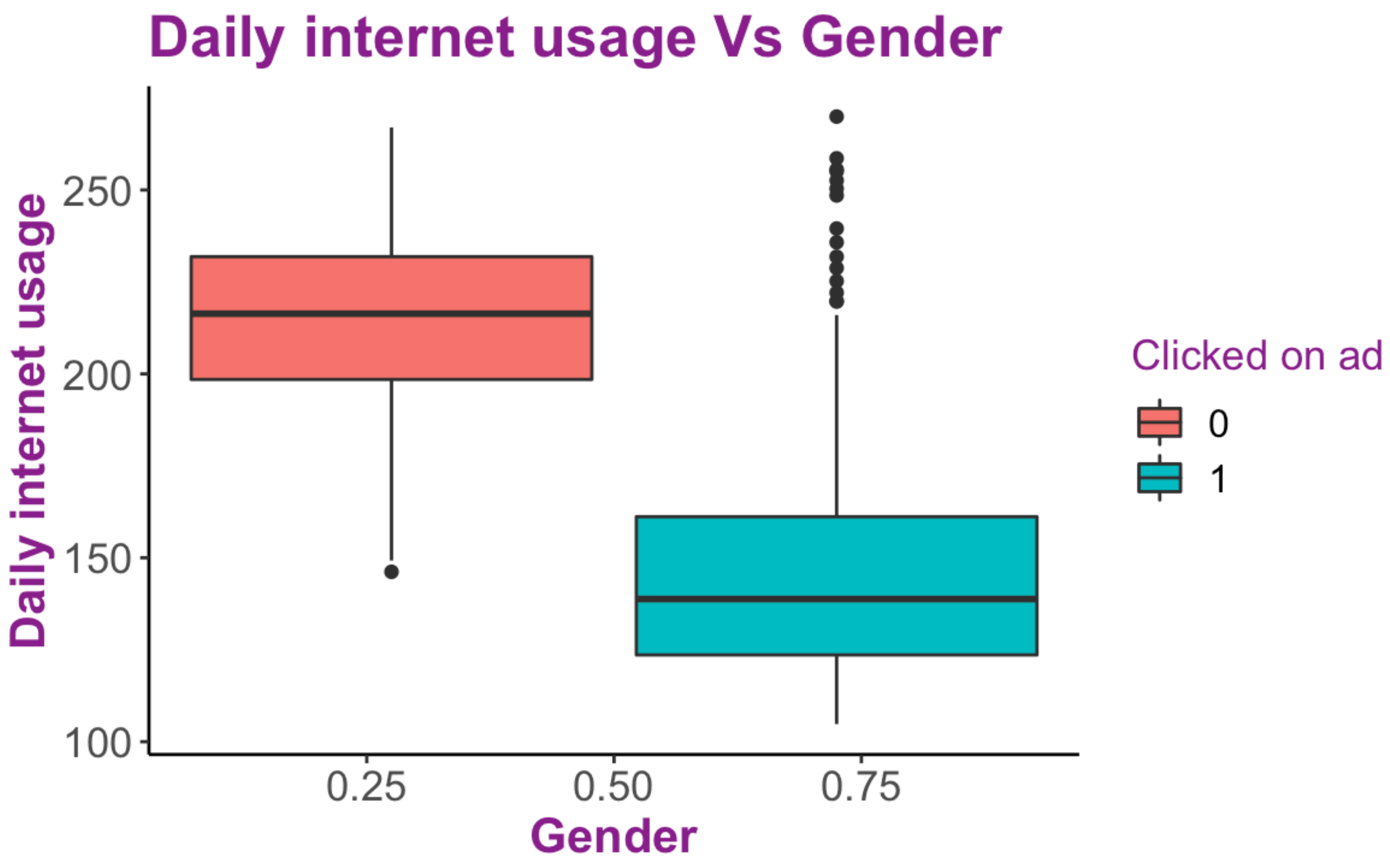
- As can be seen above the dataset is clear of duplicates.

# Checking the dataset for any outliers

Hide

```
# Plotting boxplots
options(repr.plot.width = 13, repr.plot.height = 7)
ggplot(data = df, aes(x = gender, y = daily_internet_usage)) +
    geom_boxplot(aes(fill = factor(clicked_on_ad))) +
    labs(title = 'Daily internet usage Vs Gender', y = 'Daily internet usage', x =
'Gender', fill = 'Clicked on ad') +
    scale_color_brewer(palette = 'cool') +
    theme(plot.title = element_text(size = 18, face = 'bold', color = 'darkmagenta
'),
             axis.title.x = element_text(size = 15, face = 'bold', color = 'darkma
genta'),
             axis.title.y = element_text(size = 15, face = 'bold', color = 'darkma
genta'),
             axis.text.x = element_text(size = 13),
             axis.text.y = element_text(size = 13),
             legend.title = element_text(size = 13, color = 'darkmagenta'),
             legend.text = element_text(size = 12))
```

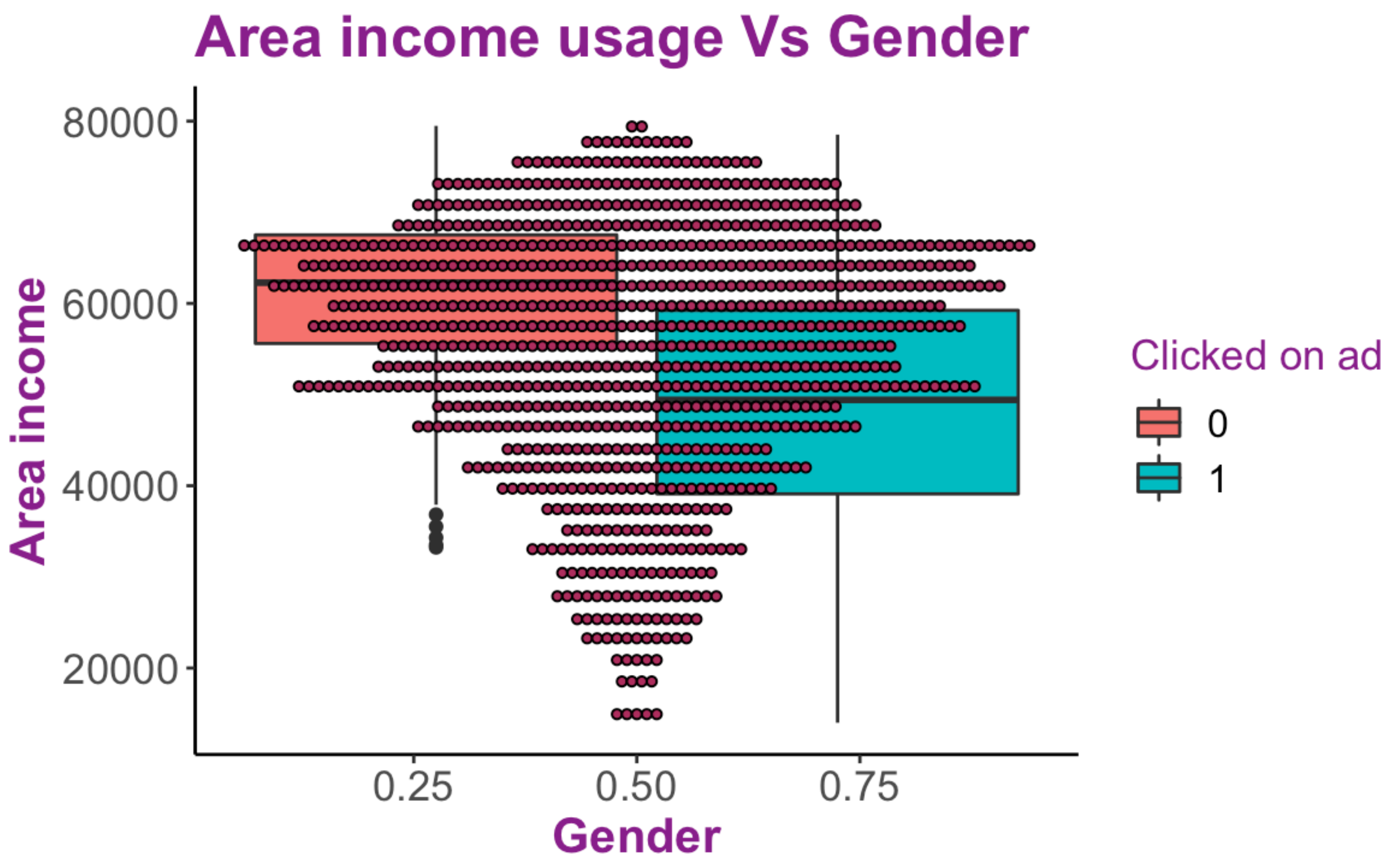
```
Unknown palette cool
```




Hide

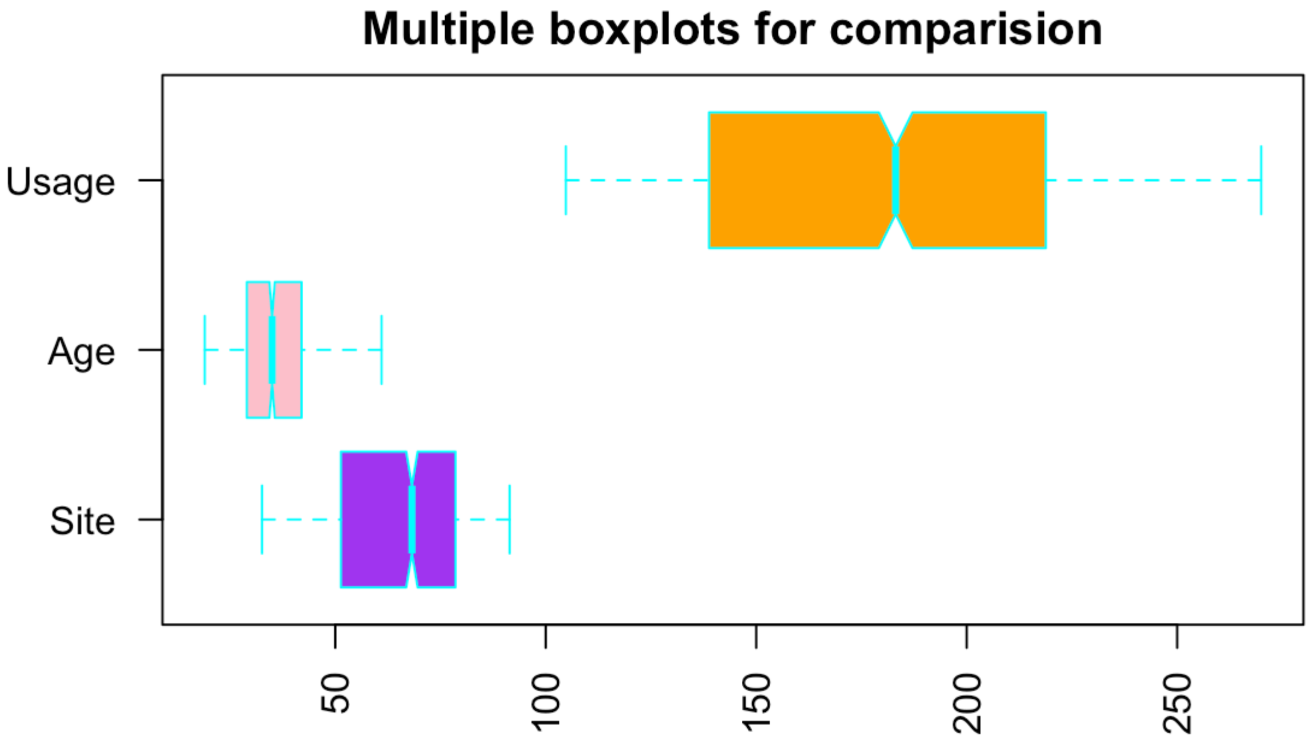
```
# a plot showing income usage in relation to gender
options(repr.plot.width = 13, repr.plot.height = 7)
ggplot(data = df, aes(x = gender, y = area_income)) +
    geom_boxplot(aes(fill = factor(clicked_on_ad))) +
    geom_dotplot(binwidth = NULL, binaxis = 'y', stackdir = 'center', dotsize = .5
, fill = 'maroon') +
    labs(title = 'Area income usage Vs Gender', y = 'Area income', x = 'Gender', f
ill = 'Clicked on ad') +
    scale_color_brewer(palette = 'cool') +
    theme(plot.title = element_text(size = 18, face = 'bold', color = 'darkmagenta
'),
              axis.title.x = element_text(size = 15, face = 'bold', color = 'darkma
genta'),
              axis.title.y = element_text(size = 15, face = 'bold', color = 'darkma
genta'),
              axis.text.x = element_text(size = 13),
              axis.text.y = element_text(size = 13),
              legend.title = element_text(size = 13, color = 'darkmagenta'),
              legend.text = element_text(size = 12))
```

```
Unknown palette cool
```




Hide

```
# plotting multiple boxplots
options(repr.plot.width = 13, repr.plot.height = 7)
boxplot(df$daily_time_spent_on_site, df$age, df$daily_internet_usage,
main = "Multiple boxplots for comparision",
at = c(1,2,3),
names = c("Site", "Age","Usage"),
las = 2,
col = c("purple","pink","orange"),
border = "cyan",
horizontal = TRUE,
notch = TRUE
)
```




Hide

```
# The male column should be renamed to gender
colnames(df)[colnames(df) == 'male'] = 'gender'
```

Hide

```
library(tidyverse) # makes it easier to install and load multiple tidyverse packag
es

# Changing column names to lower case
colnames(df) = tolower(str_replace_all(colnames(df), c(' ' = '_')))

# Checking whether the column names have been renames appriopriately
print(colnames(df))
```

```
 [1] "daily_time_spent_on_site" "age"
 [3] "area_income"              "daily_internet_usage"
 [5] "ad_topic_line"            "city"
 [7] "gender"                   "country"
 [9] "timestamp"                "clicked_on_ad"
```

Hide

```
# Checking the datatypes for each column

columns = colnames(df)
for (column in seq(length(colnames(df)))){
    print(columns[column])
    print(str(df[, column]))
    cat('\n')
}
```

```
[1] "daily_time_spent_on_site"
Classes ‘tbl_df’, ‘tbl’ and 'data.frame':   1000 obs. of  1 variable:
 $ daily_time_spent_on_site: num  69 80.2 69.5 74.2 68.4 ...
NULL

[1] "age"
Classes ‘tbl_df’, ‘tbl’ and 'data.frame':   1000 obs. of  1 variable:
 $ age: num  35 31 26 29 35 23 33 48 30 20 ...
NULL

[1] "area_income"
Classes ‘tbl_df’, ‘tbl’ and 'data.frame':   1000 obs. of  1 variable:
 $ area_income: num  61834 68442 59786 54806 73890 ...
NULL

[1] "daily_internet_usage"
Classes ‘tbl_df’, ‘tbl’ and 'data.frame':   1000 obs. of  1 variable:
 $ daily_internet_usage: num  256 194 236 246 226 ...
NULL

[1] "ad_topic_line"
Classes ‘tbl_df’, ‘tbl’ and 'data.frame':   1000 obs. of  1 variable:
 $ ad_topic_line: chr  "Cloned 5thgeneration orchestration" "Monitored national st
andardization" "Organic bottom-line service-desk" "Triple-buffered reciprocal time
-frame" ...
NULL

[1] "city"
Classes ‘tbl_df’, ‘tbl’ and 'data.frame':   1000 obs. of  1 variable:
 $ city: chr  "Wrightburgh" "West Jodi" "Davidton" "West Terrifurt" ...
NULL

[1] "gender"
Classes ‘tbl_df’, ‘tbl’ and 'data.frame':   1000 obs. of  1 variable:
 $ gender: num  0 1 0 1 0 1 0 1 1 1 ...
NULL

[1] "country"
Classes ‘tbl_df’, ‘tbl’ and 'data.frame':   1000 obs. of  1 variable:
 $ country: chr  "Tunisia" "Nauru" "San Marino" "Italy" ...
```

```
NULL

[1] "timestamp"
Classes ‘tbl_df’, ‘tbl’ and 'data.frame':   1000 obs. of  1 variable:
 $ timestamp: POSIXct, format: "2016-03-27 00:53:11" "2016-04-04 01:39:02" ...
NULL

[1] "clicked_on_ad"
Classes ‘tbl_df’, ‘tbl’ and 'data.frame':   1000 obs. of  1 variable:
 $ clicked_on_ad: num  0 0 0 0 0 0 0 1 0 0 ...
NULL
```

Hide

```
library(magrittr) # offers a set of operators that provide semantics that will imp
rove code

# Changing column names to their appriopriate data type
# Creating a lists of categorical and numerical columns

# List of categorical columns
cat_cols = c("ad_topic_line", "city", "gender", "country", "clicked_on_ad" )

# List of numerical columns
num_cols = c("daily_time_spent_on_site", "age", "area_income", "daily_internet_usa
ge")

# Changing columns to factors
df[,cat_cols] %<>% lapply(function(x) as.factor(as.character(x)))
```

Hide

```
# Checking whether the datatypes for each column have been changed apprippriately
columns = colnames(df)
for (column in seq(length(colnames(df)))){
    print(columns[column])
    print(str(df[, column]))
    print(nlevels(df[, column]))
    cat('\n')
}
```

```
[1] "daily_time_spent_on_site"
Classes ‘tbl_df’, ‘tbl’ and 'data.frame':   1000 obs. of  1 variable:
 $ daily_time_spent_on_site: num  69 80.2 69.5 74.2 68.4 ...
NULL
[1] 0

[1] "age"
Classes ‘tbl_df’, ‘tbl’ and 'data.frame':   1000 obs. of  1 variable:
```

```
 $ age: num  35 31 26 29 35 23 33 48 30 20 ...
NULL
[1] 0

[1] "area_income"
Classes ‘tbl_df’, ‘tbl’ and 'data.frame':   1000 obs. of  1 variable:
 $ area_income: num  61834 68442 59786 54806 73890 ...
NULL
[1] 0

[1] "daily_internet_usage"
Classes ‘tbl_df’, ‘tbl’ and 'data.frame':   1000 obs. of  1 variable:
 $ daily_internet_usage: num  256 194 236 246 226 ...
NULL
[1] 0

[1] "ad_topic_line"
Classes ‘tbl_df’, ‘tbl’ and 'data.frame':   1000 obs. of  1 variable:
 $ ad_topic_line: Factor w/ 1000 levels "Adaptive 24hour Graphic Interface",..: 92
465 567 904 767 806 223 724 108 455 ...
NULL
[1] 0

[1] "city"
Classes ‘tbl_df’, ‘tbl’ and 'data.frame':   1000 obs. of  1 variable:
 $ city: Factor w/ 969 levels "Adamsbury","Adamside",..: 962 904 112 940 806 283 4
7 672 885 713 ...
NULL
[1] 0

[1] "gender"
Classes ‘tbl_df’, ‘tbl’ and 'data.frame':   1000 obs. of  1 variable:
 $ gender: Factor w/ 2 levels "0","1": 1 2 1 2 1 2 1 2 2 2 ...
NULL
[1] 0

[1] "country"
Classes ‘tbl_df’, ‘tbl’ and 'data.frame':   1000 obs. of  1 variable:
 $ country: Factor w/ 237 levels "Afghanistan",..: 216 148 185 104 97 159 146 13 8
3 79 ...
NULL
[1] 0

[1] "timestamp"
Classes ‘tbl_df’, ‘tbl’ and 'data.frame':   1000 obs. of  1 variable:
 $ timestamp: POSIXct, format: "2016-03-27 00:53:11" "2016-04-04 01:39:02" ...
NULL
[1] 0

[1] "clicked_on_ad"
```

```
Classes ‘tbl_df’, ‘tbl’ and 'data.frame':   1000 obs. of  1 variable:
 $ clicked_on_ad: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 2 1 1 ...
NULL
[1] 0
```

Hide

```
# Frequency tables
# 0-female, 1-male
levels(df$gender) = c("Female", "Male")
table(df$gender)
```

```

Female   Male
   519    481
```

> - We can see that the gender column is seen to be almost evenly distributed with Females being slightly higher than the males.

Hide

```
#0-yes,1-no
levels(df$clicked_on_ad) = c("Yes", "No")
table(df$clicked_on_ad)
```

```

Yes  No
500 500
```

> - For the number of people who did or did not click on ads we have an even distribution

# Exploratory Data Analysis

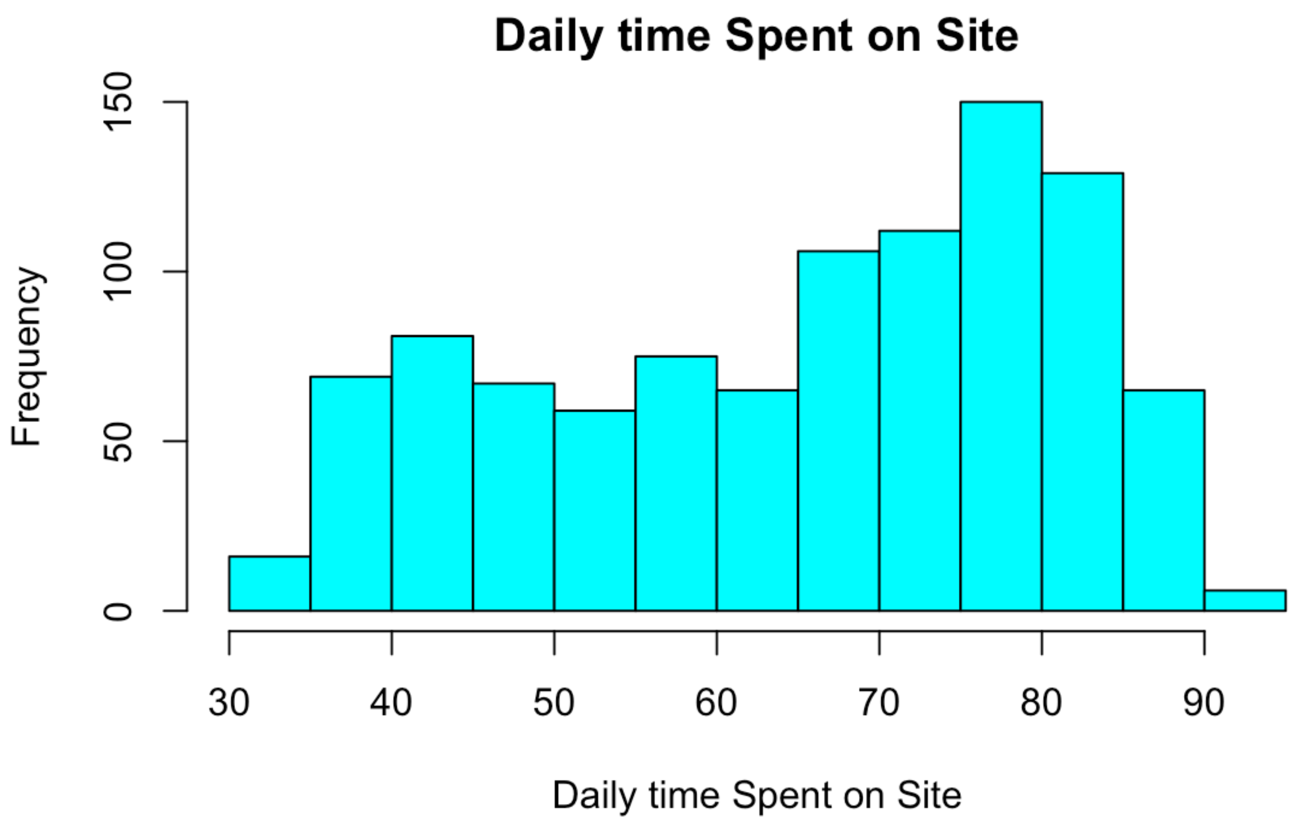
> This is where we explore the data so as to: * maximize insights on the data set * uncover underlying structure * extract important variables * detect outliers and anomalies * test underlying assumptions * develop models with great explanatory predictive power * determine optimal factor settings

> Here we will perform : * univariate analysis * bivariate analysis * multivariate analysis

# Univariate Analysis

Hide

```
# Daily time spent on site distribution
#
x = hist(df$daily_time_spent_on_site,
         main = "Daily time Spent on Site",
         xlab = "Daily time Spent on Site",
         col = "cyan",
)
```



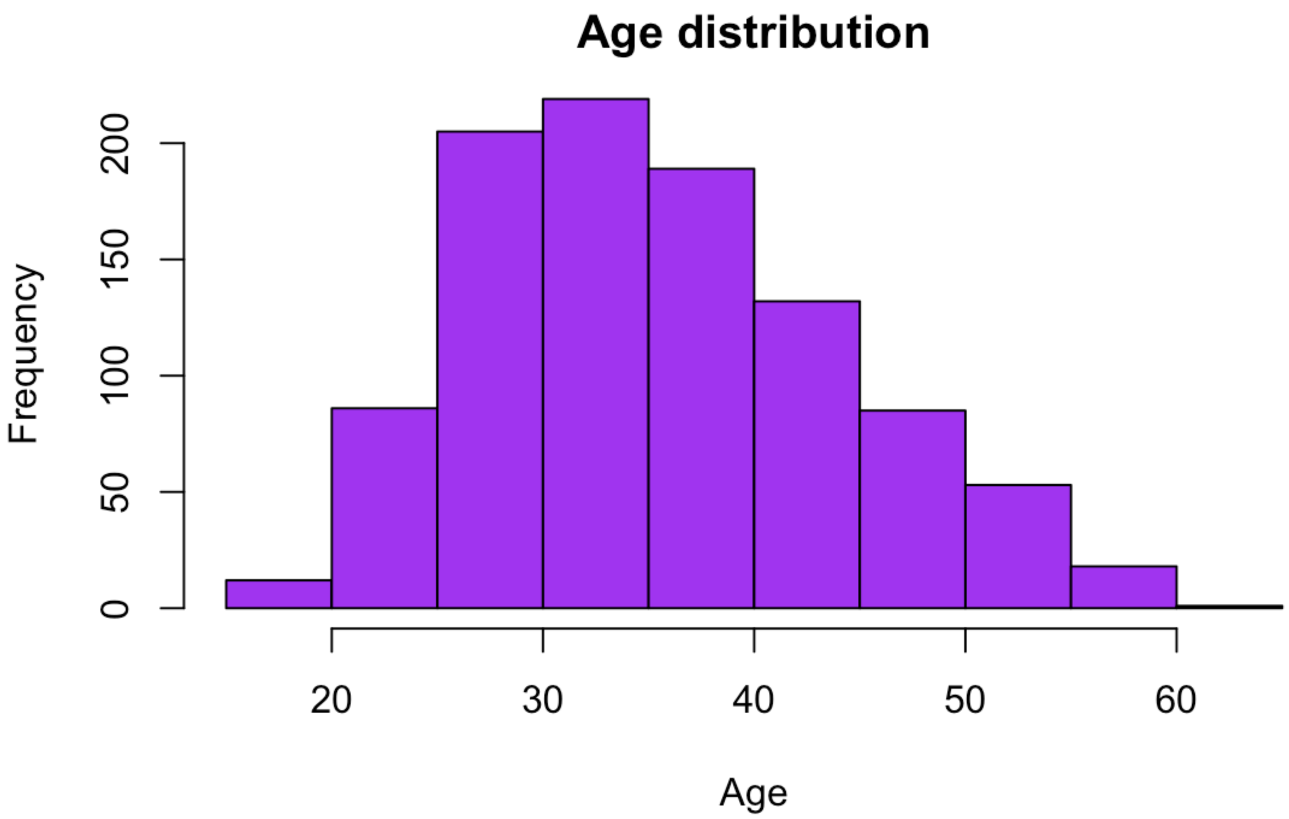

Hide

```
summary(df$daily_time_spent_on_site)
```

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

Hide

```
# Age distribution plot
#
y = hist(df$age,
         main = "Age distribution",
         xlab = "Age",
         col = "purple",
)
```



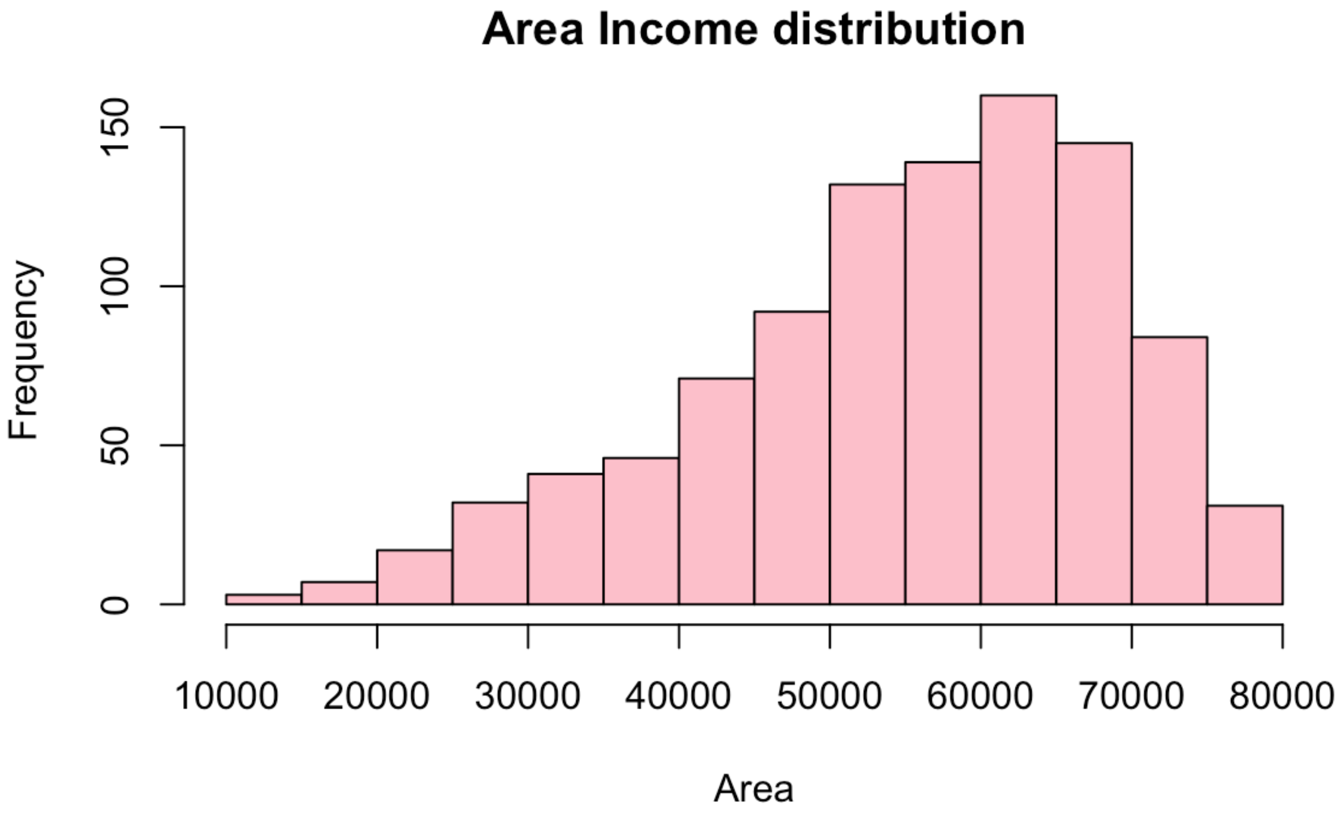

Hide

```
summary(df$age)
```

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

Hide

```
# a plot showing area income distribution
z = hist(df$area_income,
         main = "Area Income distribution",
         xlab = "Area",
         col = "pink",
)
```



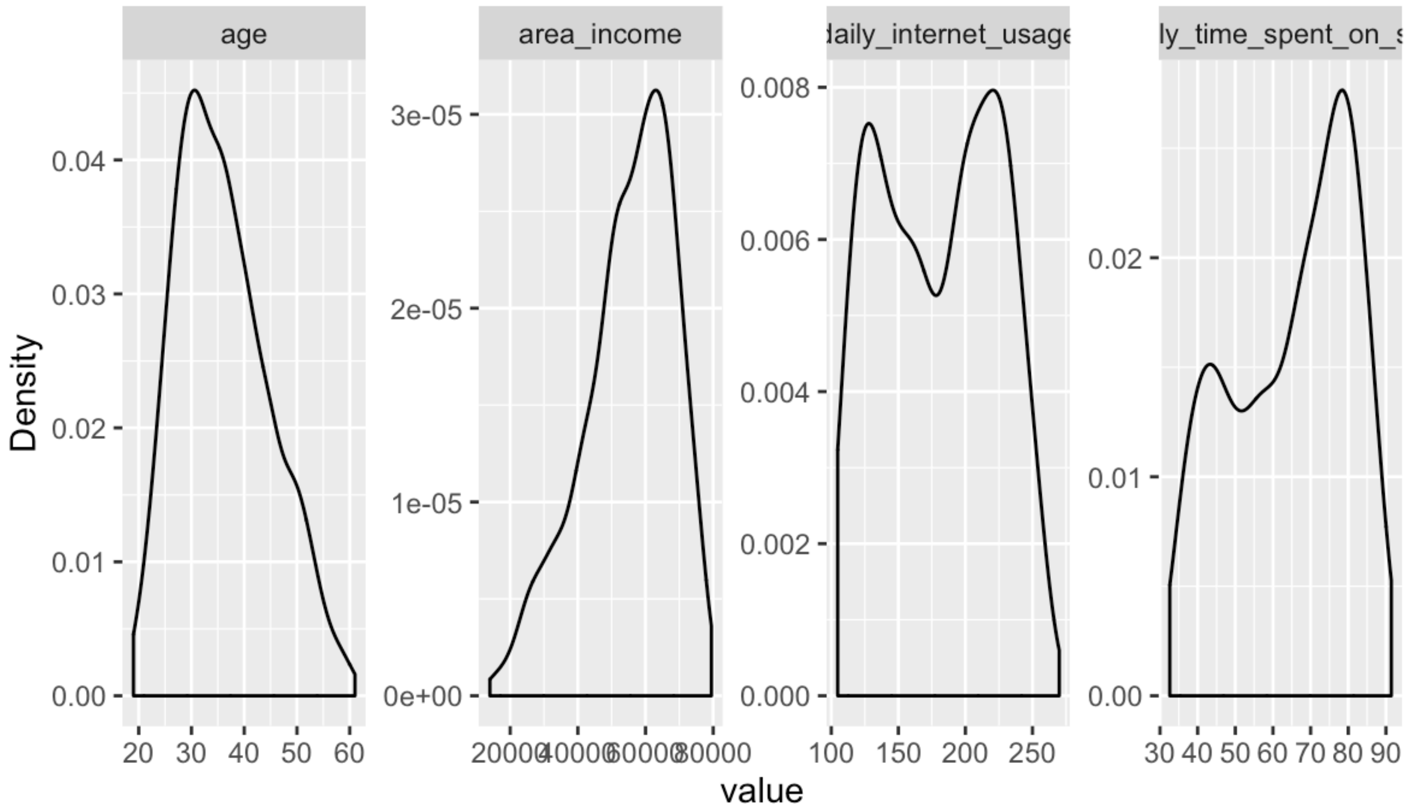

Hide

```
summary(df$area_income)
```

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

Hide

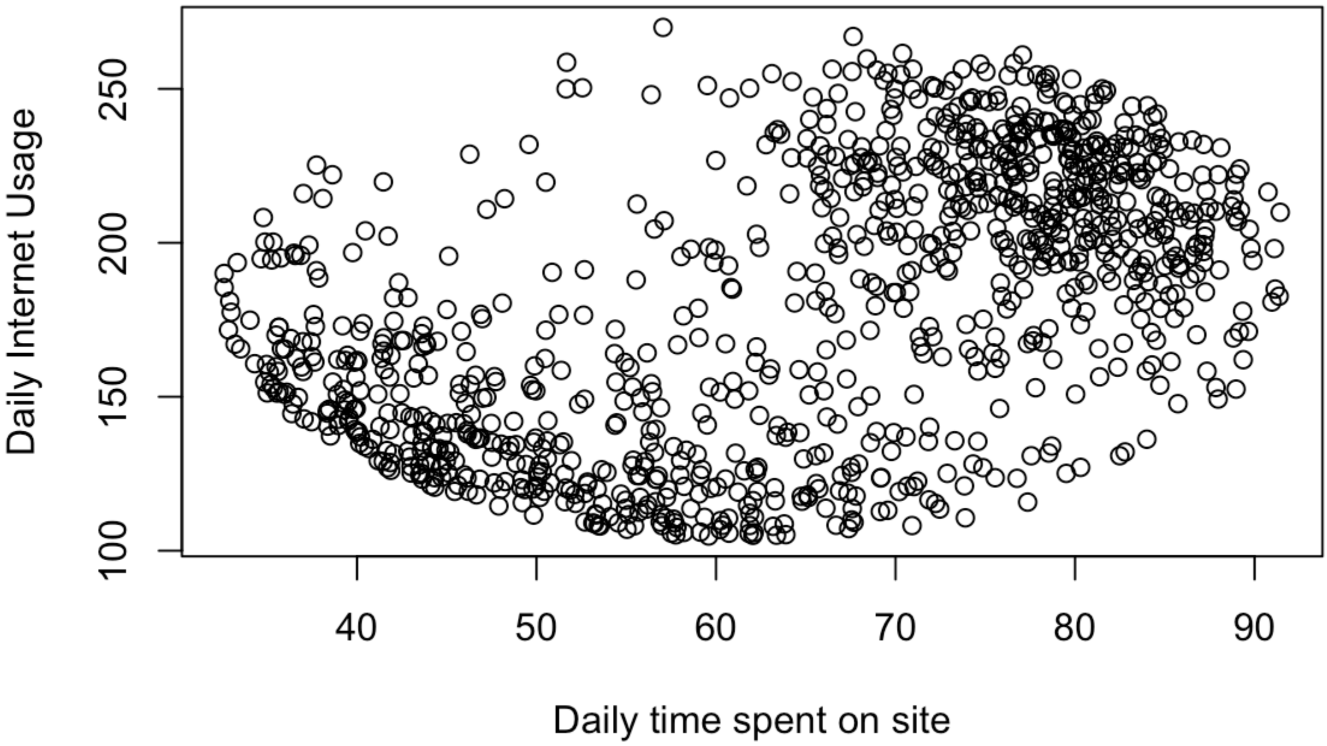
```
#density plots for univariate analysis
library(DataExplorer)
plot_density(df)
```




# Bivariate Analysis

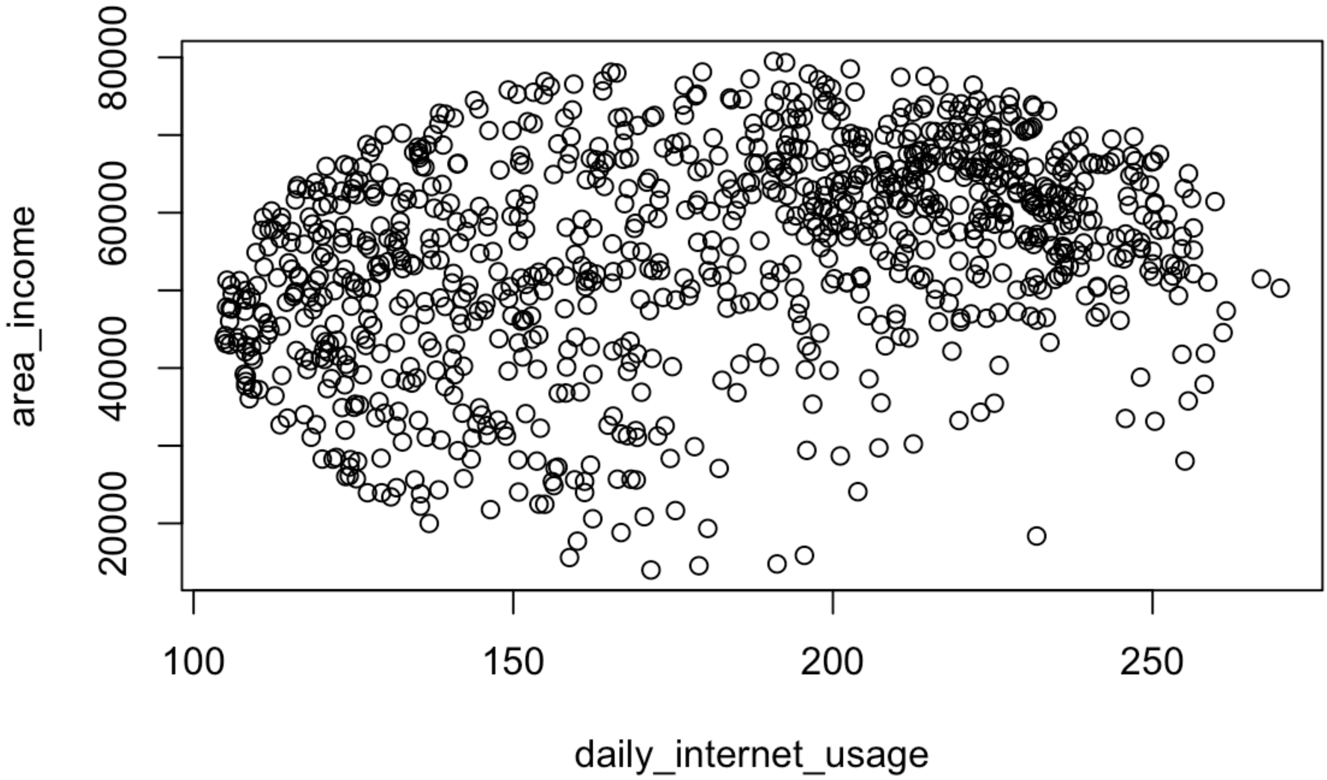
Hide

```
# scatter plots
library(DataExplorer)
# a scatter plot showing relations between daily internet and time spent on the si
te
timespent <- df$daily_time_spent_on_site
internetusage<- df$daily_internet_usage
plot(timespent, internetusage, xlab="Daily time spent on site", ylab="Daily Intern
et Usage")
```




Hide

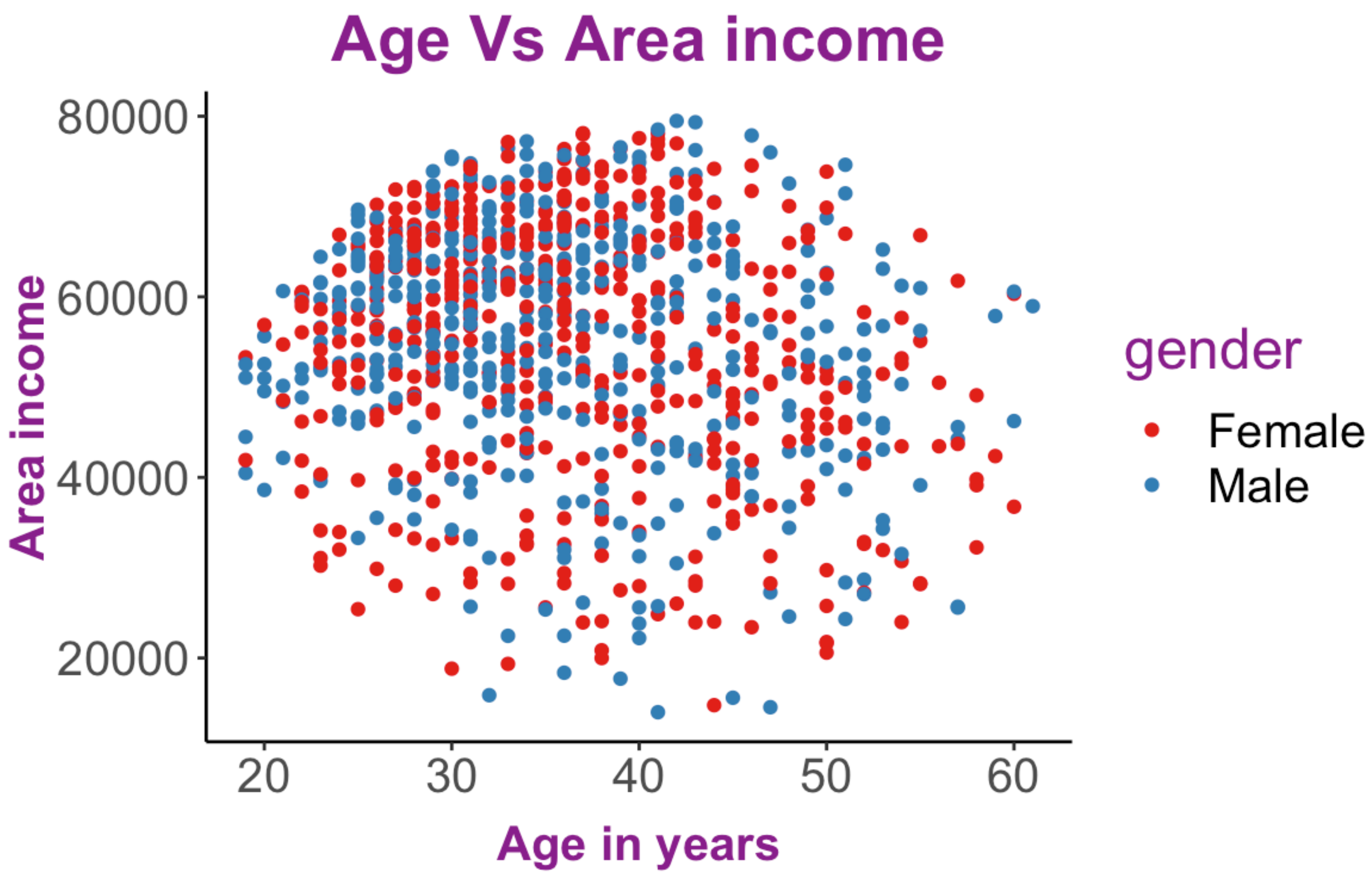
```
# a scatter plot showing relations between internet usage and area income
plot(area_income ~ daily_internet_usage, data = df)
```




# Scatter plots showing relationships between variables

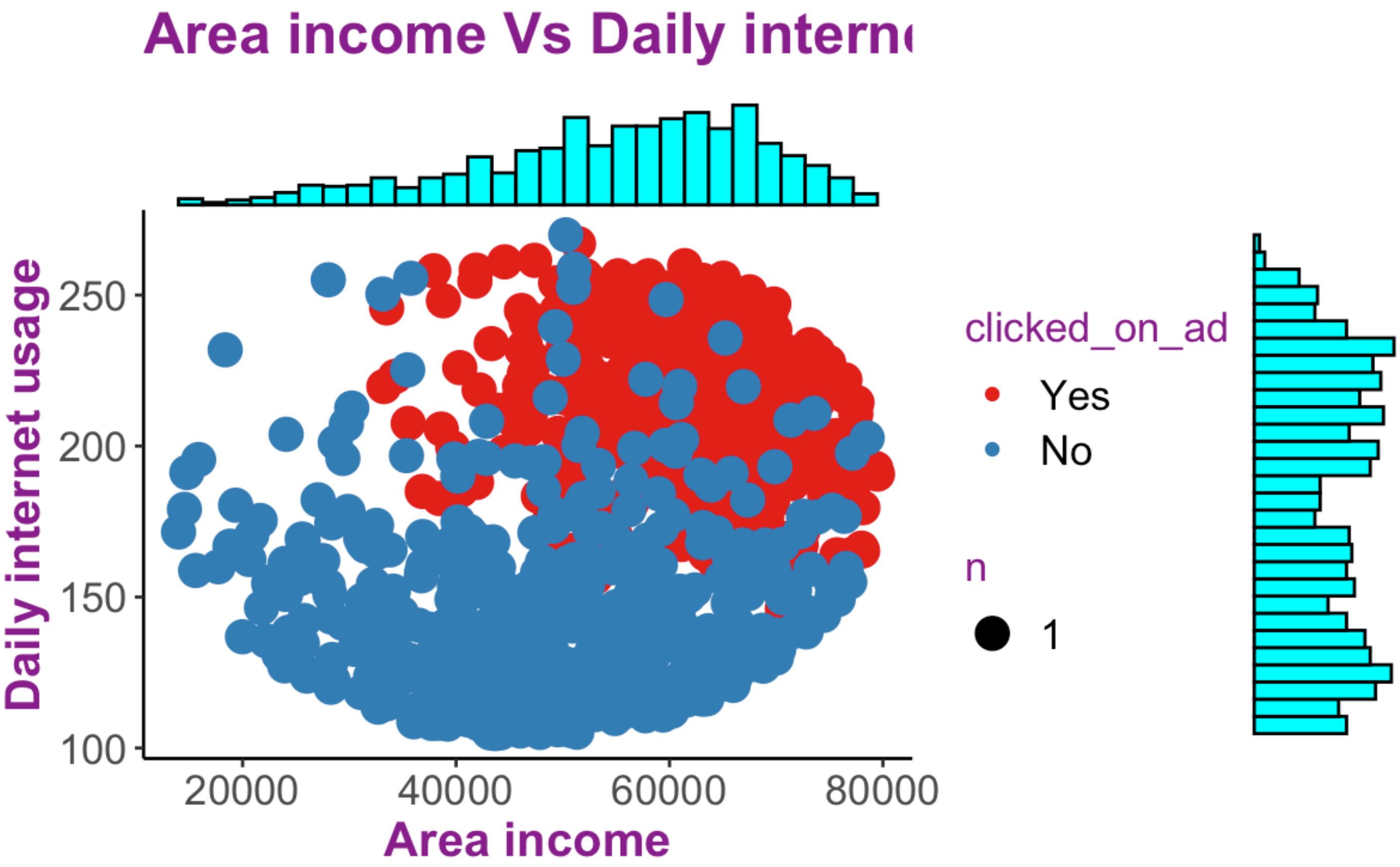
Hide

```
# Plotting a scatter plot of age vs income
options(repr.plot.width = 13, repr.plot.height = 7)
gg = ggplot(data = df, aes(x = age, y = area_income, col = gender)) +
    geom_point() +
    labs(title = 'Age Vs Area income', x = 'Age in years', y = 'Area income') +
    scale_color_brewer(palette = 'Set1') +
    theme(plot.title=element_text(size=20, face="bold", color="darkmagenta",hjust=
0.5, lineheight=1.2),
          plot.subtitle=element_text(size=15, face="bold", hjust=0.5),
         axis.title.x = element_text(color = 'darkmagenta', size = 15, face = 'bol
d', vjust = -0.5),
         axis.title.y = element_text(color = 'darkmagenta', size = 15, face = 'bol
d', vjust = 0.5),
         axis.text.y = element_text(size = 15),axis.text.x = element_text(size = 1
5),
         legend.title = element_text(size = 18, color = 'darkmagenta'),
        legend.text = element_text(size = 15))
plot(gg)
```



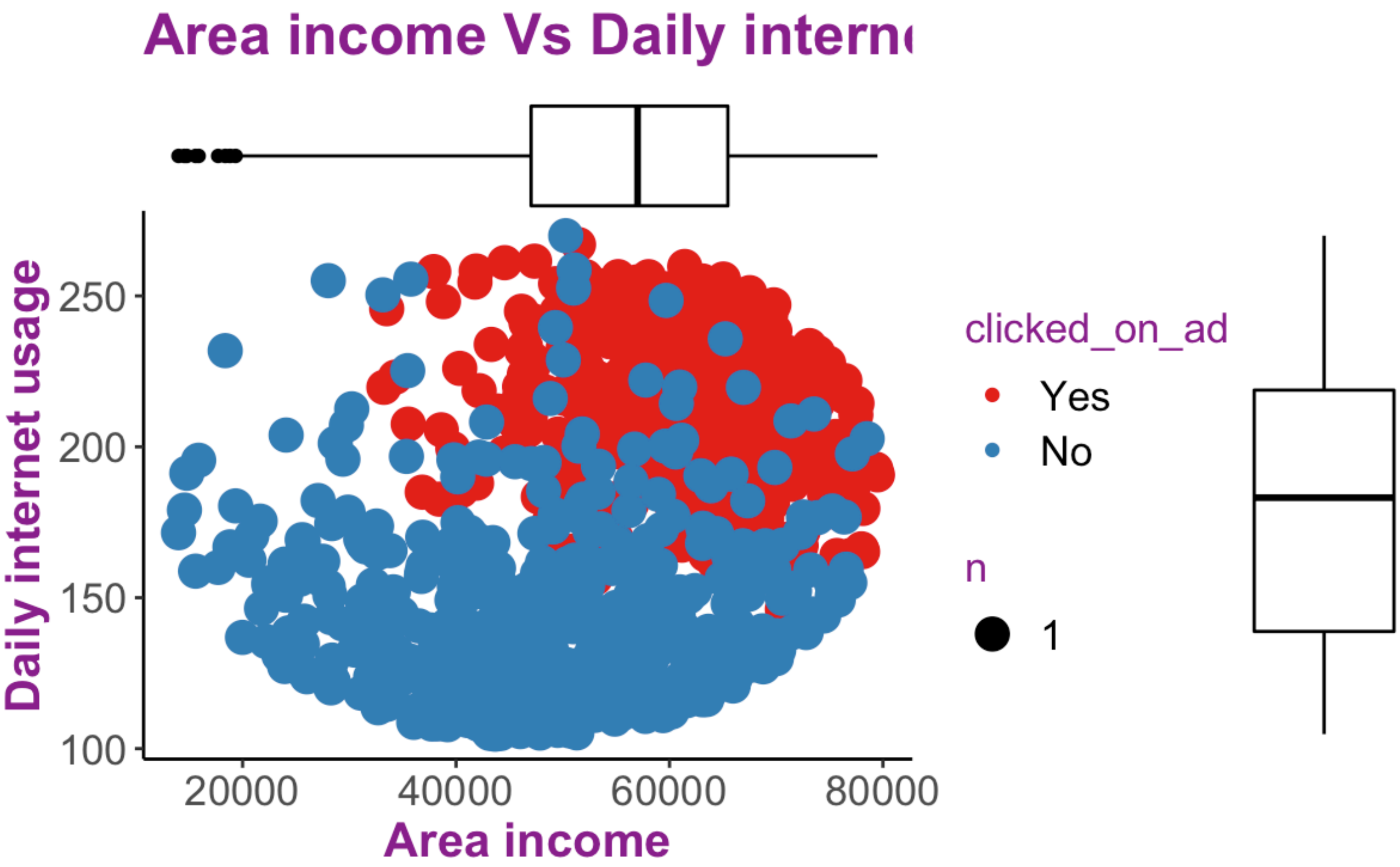

Hide

```
library (ggExtra) # used to add marginal histograms, densityplots and boxplots to
scatter plots

options(repr.plot.width = 13, repr.plot.height = 7)
g = ggplot(data =df, aes(x =area_income, y = daily_internet_usage, col= clicked_on
_ad)) +
    geom_count() +
    labs(title = 'Area income Vs Daily internet usage', y = 'Daily internet usage'
, x = 'Area income', fill = 'Clicked on ad') +
    scale_color_brewer(palette = 'Set1') +
    theme(plot.title = element_text(size = 18, face = 'bold', color = 'darkmagenta
'),
             axis.title.x = element_text(size = 15, face = 'bold', color = 'darkma
genta'),
             axis.title.y = element_text(size = 15, face = 'bold', color = 'darkma
genta'),
             axis.text.x = element_text(size = 13),
             axis.text.y = element_text(size = 13),
             legend.title = element_text(size = 13, color = 'darkmagenta'),
             legend.text = element_text(size = 13))

ggMarginal(g, type = "histogram", fill="cyan")
```



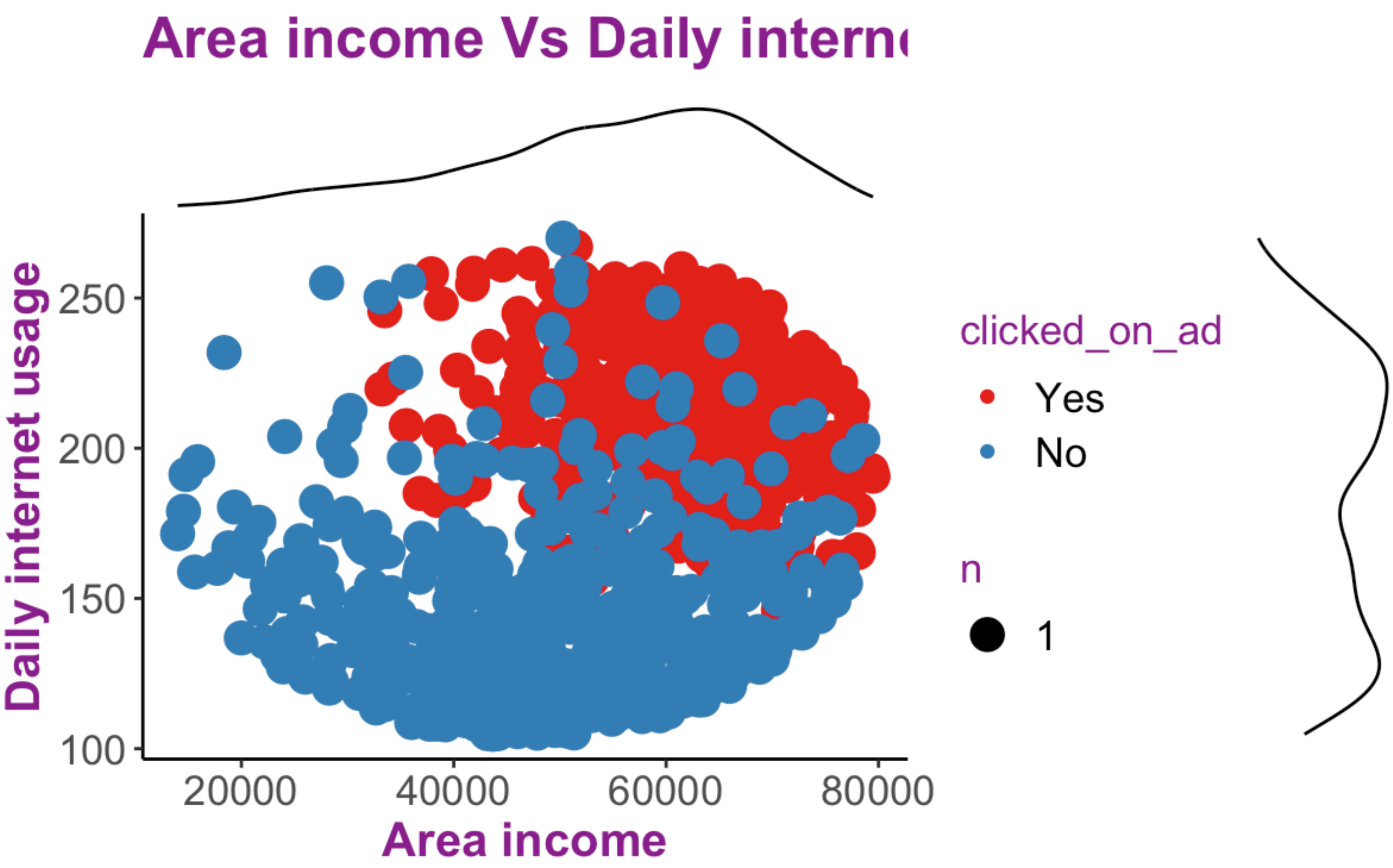

Hide

```
ggMarginal(g, type = "boxplot", fill="transparent")
```




Hide

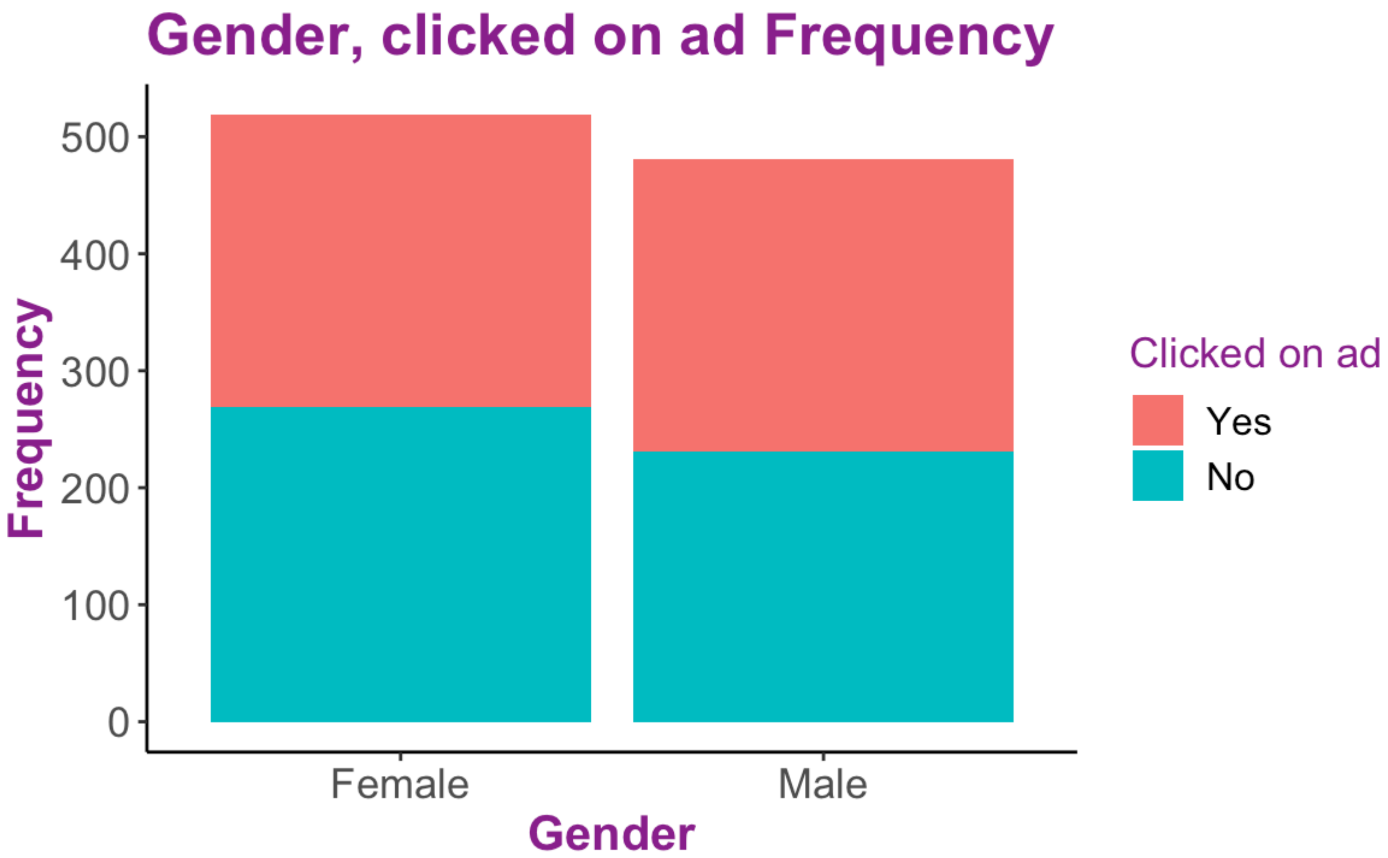
```
ggMarginal(g, type = "density", fill="transparent")
```



Hide

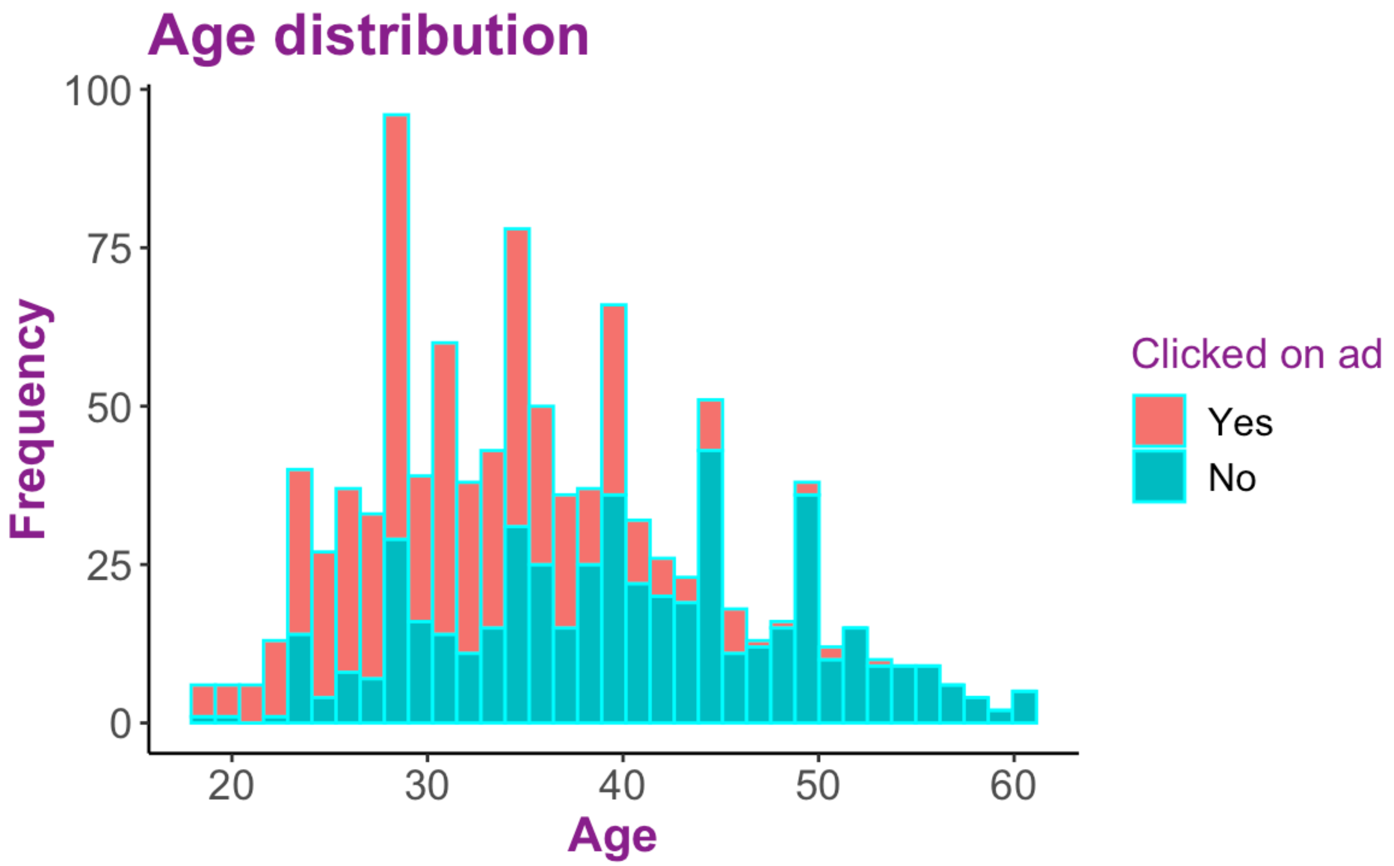
```
# A frequency plot in relation to gender
options(repr.plot.width = 13, repr.plot.height = 7)
ggplot(data = df, aes(x = gender))+
    geom_bar(aes(fill = clicked_on_ad))+
    labs(title = 'Gender, clicked on ad Frequency', y = 'Frequency', x = 'Gender',
fill = 'Clicked on ad') +
    scale_color_brewer(palette = 'cool') +
    theme(plot.title = element_text(size = 18, face = 'bold', color = 'darkmagenta
'),
             axis.title.x = element_text(size = 15, face = 'bold', color = 'darkma
genta'),
             axis.title.y = element_text(size = 15, face = 'bold', color = 'darkma
genta'),
             axis.text.x = element_text(size = 13),
             axis.text.y = element_text(size = 13),
             legend.title = element_text(size = 13, color = 'darkmagenta'),
             legend.text = element_text(size = 12))
```

```
Unknown palette cool
```



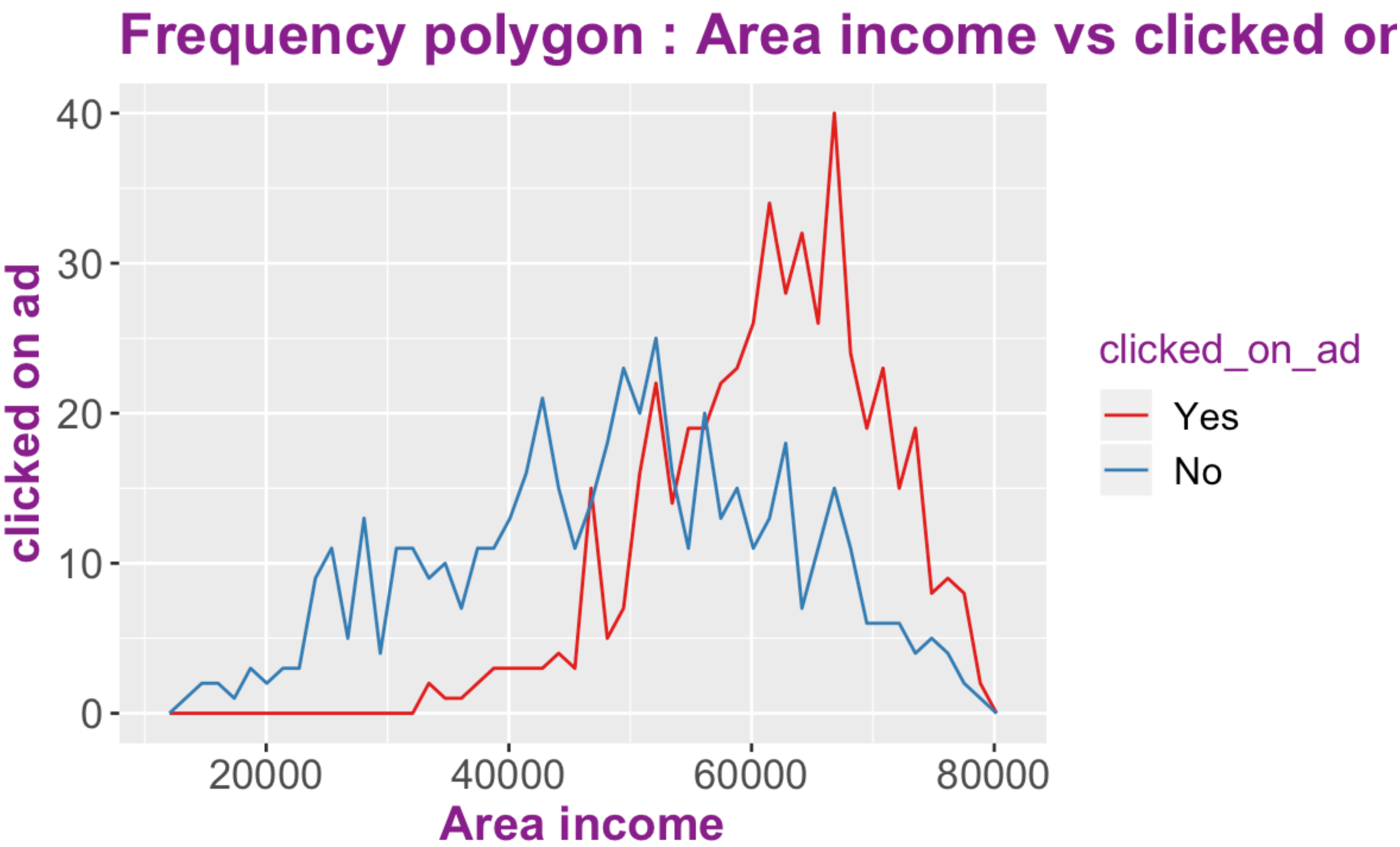

Hide

```
# Plotting a a pair of histograms
options(repr.plot.width = 13, repr.plot.height = 7)
ggplot(data = df, aes(x = age, fill = clicked_on_ad))+
    geom_histogram(bins = 35, color = 'cyan') +
    labs(title = 'Age distribution', x = 'Age', y = 'Frequency', fill = 'Clicked o
n ad') +
        scale_color_brewer(palette = 'Set1') +
        theme(plot.title = element_text(size = 18, face = 'bold', color = 'darkmag
enta'),
             axis.title.x = element_text(size = 15, face = 'bold', color = 'darkma
genta'),
             axis.title.y = element_text(size = 15, face = 'bold', color = 'darkma
genta'),
             axis.text.x = element_text(size = 13, angle = 0),
             axis.text.y = element_text(size = 13),
             legend.title = element_text(size = 13, color = 'darkmagenta'),
             legend.text = element_text(size = 12))
```




Hide

```
# Frequency polygon
options(repr.plot.width = 13, repr.plot.height = 7)
ggplot(data = df, aes(x = area_income, col = clicked_on_ad))+
    geom_freqpoly(bins = 50)+
    labs(title = 'Frequency polygon : Area income vs clicked on ad', x = 'Area inc
ome', y = 'clicked on ad', fill = 'Clicked on ad') +
        scale_color_brewer(palette = 'Set1') +
        theme(plot.title = element_text(size = 18, face = 'bold', color = 'darkmag
enta'),
             axis.title.x = element_text(size = 15, face = 'bold', color = 'darkma
genta'),
             axis.title.y = element_text(size = 15, face = 'bold', color = 'darkma
genta'),
             axis.text.x = element_text(size = 13),
             axis.text.y = element_text(size = 13),
             legend.title = element_text(size = 13, color = 'darkmagenta'),
             legend.text = element_text(size = 12))
```



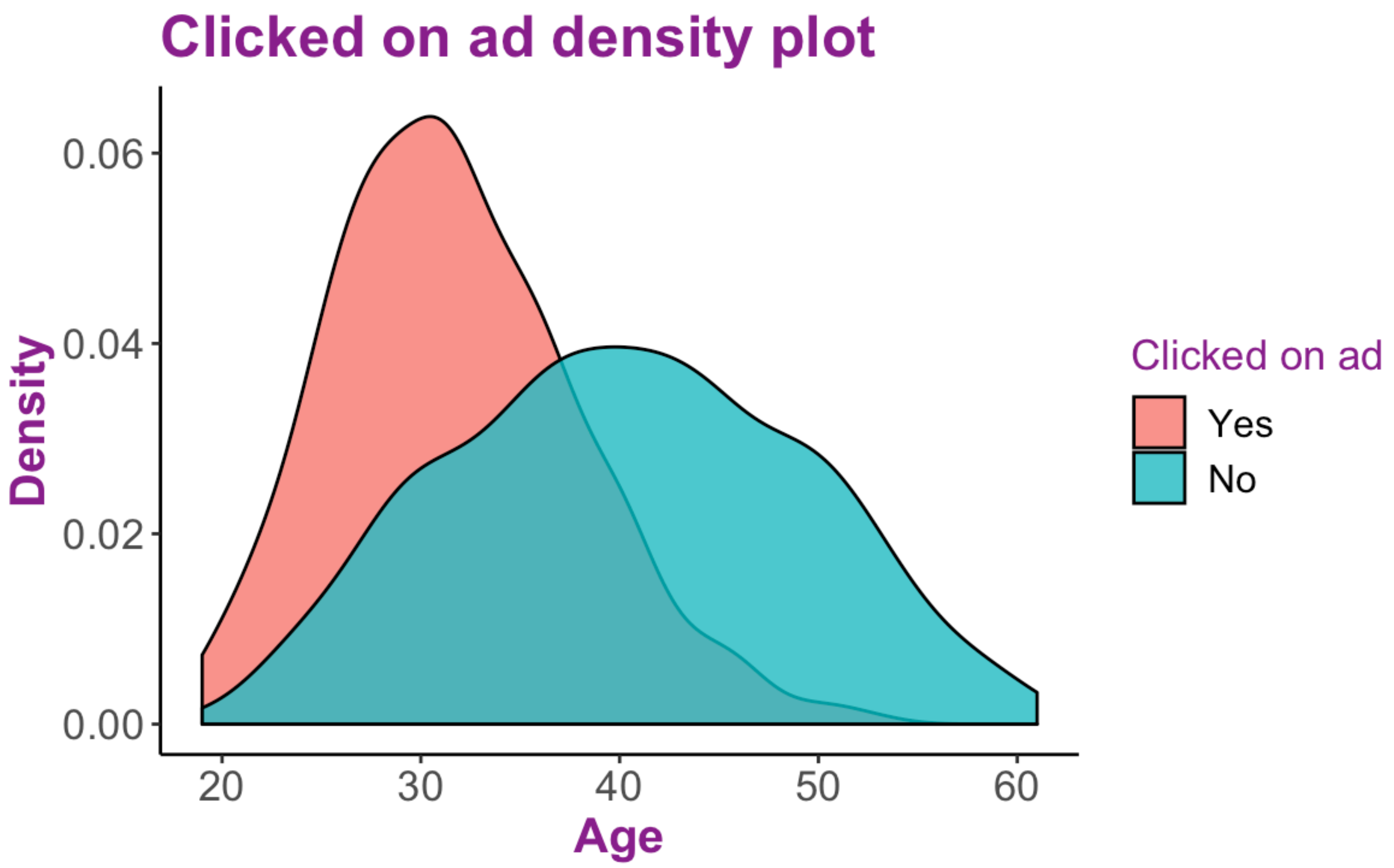

Hide

```
# Plotting density plot
options(repr.plot.width = 13, repr.plot.height = 7)
p1 = ggplot(data = df, aes(age)) +
        geom_density(aes(fill=factor(clicked_on_ad)), alpha = 0.8) +
        labs(title = 'Clicked on ad density plot', x = 'Age', y = 'Density', fill
= 'Clicked on ad') +
        scale_color_brewer(palette = 'cool') +
        theme(plot.title = element_text(size = 18, face = 'bold', color = 'darkmag
enta'),
             axis.title.x = element_text(size = 15, face = 'bold', color = 'darkma
genta'),
             axis.title.y = element_text(size = 15, face = 'bold', color = 'darkma
genta'),
             axis.text.x = element_text(size = 13, angle = 0),
             axis.text.y = element_text(size = 13),
             legend.title = element_text(size = 13, color = 'darkmagenta'),
             legend.text = element_text(size = 12))
```

```
Unknown palette cool
```

Hide

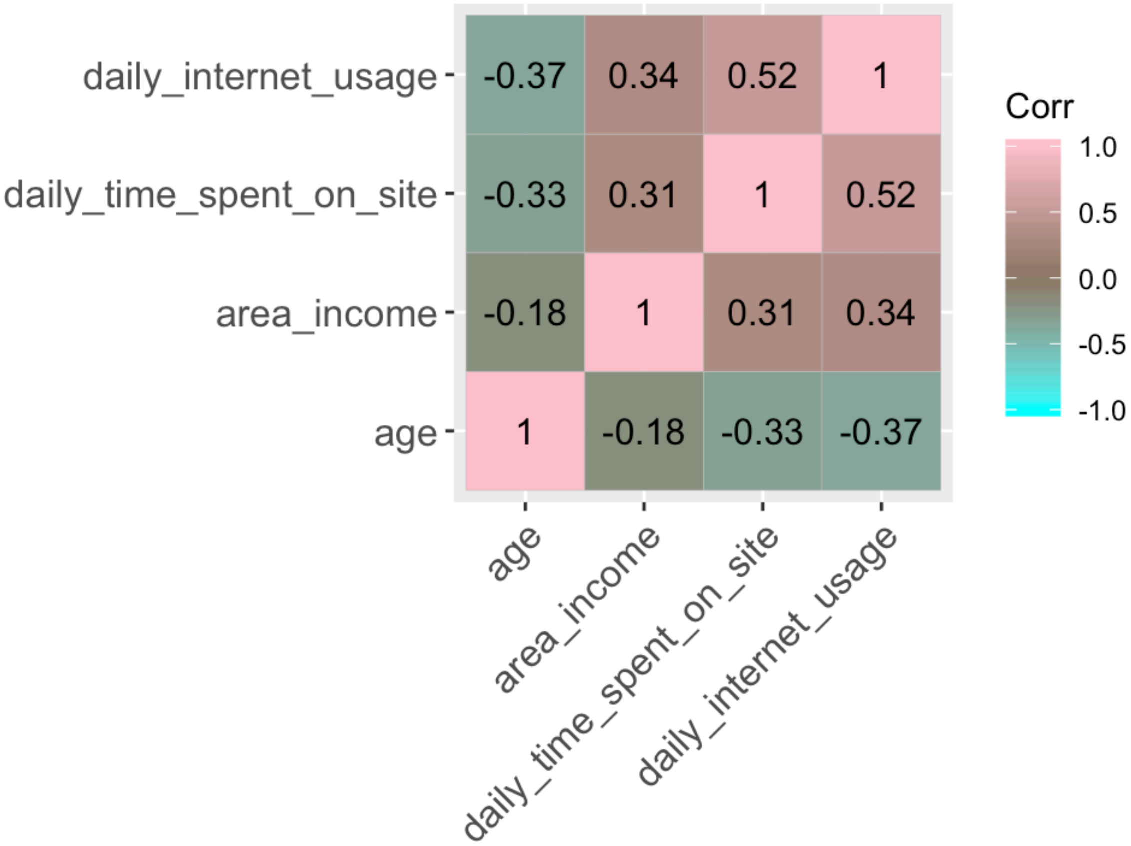
```
plot(p1)
```



# Correlation Matrix

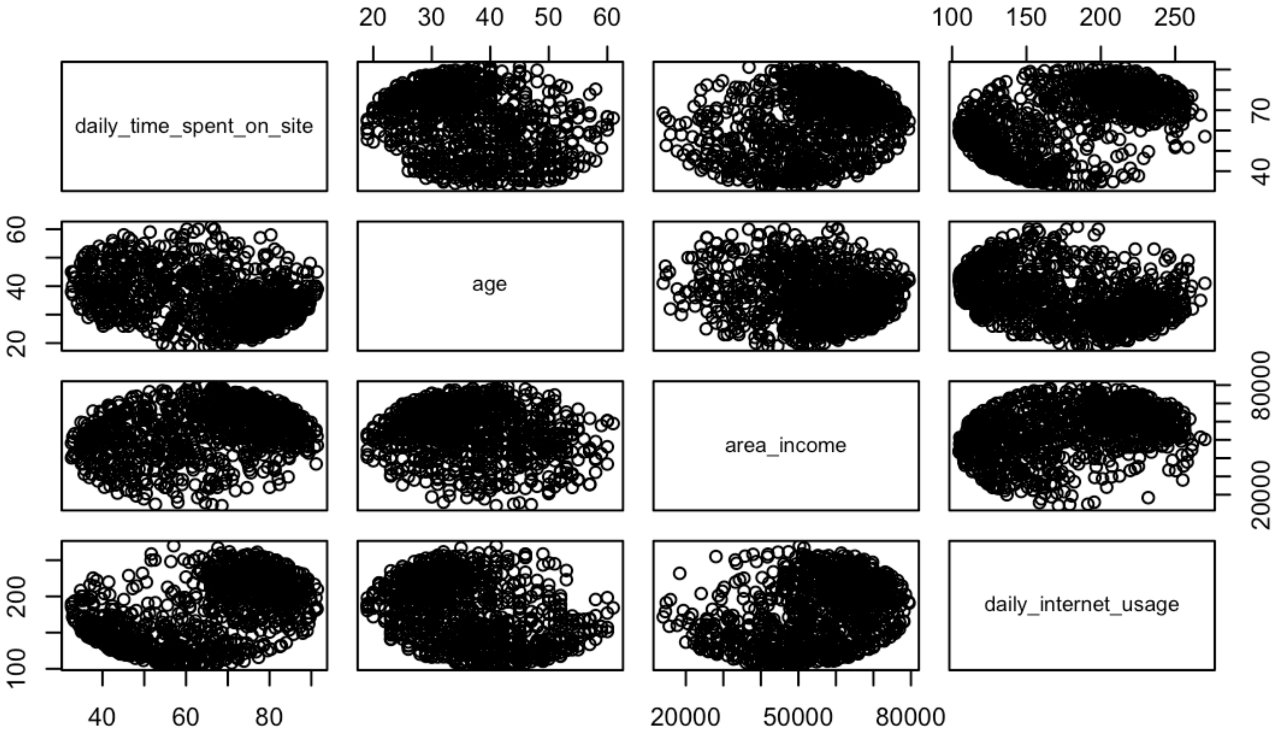
Hide

```
# multivariate analysis
# a correlation matrix showing the correlation between each variable
library(ggcorrplot)
corr = round(cor(select_if(df, is.numeric)), 2)
ggcorrplot(corr, hc.order = T, ggtheme = ggplot2::theme_grey,
   colors = c("cyan", "peachpuff4", "pink"), lab = T)
```




Hide

```
# Pairplot
pairs(df[,c(1,2,3,4)])
```




# Splitting the dataset into train and test

Hide

```
library (caret) # automating the tuning process

# Splitting the data into training and testing sets
# Setting the seed to 100, for reproducibility
set.seed(100)

# Selecting only columns that are relevant to modeling
mod_cols = c('daily_time_spent_on_site', 'age', 'area_income', 'daily_internet_usa
ge', 'gender', 'clicked_on_ad')
df = select(df, mod_cols)

# Splitting the data into 80% training and 20% testing
train_rows = createDataPartition(df$clicked_on_ad, p=0.8, list=FALSE)

# Creating the training  dataset
train = df[train_rows,]

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

# Creating the  X and Y variables
x = train
y = train$clicked_on_ad
```

# Training the model

Hide

```
# Training the model
model = train(clicked_on_ad ~ ., data = train, method = 'earth')
```

```
glm.fit: fitted probabilities numerically 0 or 1 occurredglm.fit: fitted probabili
ties numerically 0 or 1 occurredglm.fit: fitted probabilities numerically 0 or 1 o
ccurredglm.fit: fitted probabilities numerically 0 or 1 occurredglm.fit: fitted pr
obabilities numerically 0 or 1 occurredglm.fit: fitted probabilities numerically 0
or 1 occurredglm.fit: fitted probabilities numerically 0 or 1 occurredglm.fit: fit
ted probabilities numerically 0 or 1 occurredglm.fit: fitted probabilities numeric
ally 0 or 1 occurredglm.fit: fitted probabilities numerically 0 or 1 occurredglm.f
it: fitted probabilities numerically 0 or 1 occurredglm.fit: fitted probabilities
numerically 0 or 1 occurredglm.fit: fitted probabilities numerically 0 or 1 occurr
edglm.fit: fitted probabilities numerically 0 or 1 occurredglm.fit: fitted probabi
lities numerically 0 or 1 occurredglm.fit: fitted probabilities numerically 0 or 1
occurredglm.fit: fitted probabilities numerically 0 or 1 occurredglm.fit: fitted p
robabilities numerically 0 or 1 occurredglm.fit: fitted probabilities numerically
0 or 1 occurredglm.fit: fitted probabilities numerically 0 or 1 occurredglm.fit: f
itted probabilities numerically 0 or 1 occurredglm.fit: fitted probabilities numer
ically 0 or 1 occurredglm.fit: fitted probabilities numerically 0 or 1 occurredglm
.fit: fitted probabilities numerically 0 or 1 occurredglm.fit: fitted probabilitie
s numerically 0 or 1 occurredglm.fit: fitted probabilities numerically 0 or 1 occu
rredglm.fit: fitted probabilities numerically 0 or 1 occurredglm.fit: fitted proba
bilities numerically 0 or 1 occurredglm.fit: fitted probabilities numerically 0 or
1 occurredglm.fit: fitted probabilities numerically 0 or 1 occurredglm.fit: fitted
probabilities numerically 0 or 1 occurredglm.fit: fitted probabilities numerically
0 or 1 occurredglm.fit: fitted probabilities numerically 0 or 1 occurred
```

Hide

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

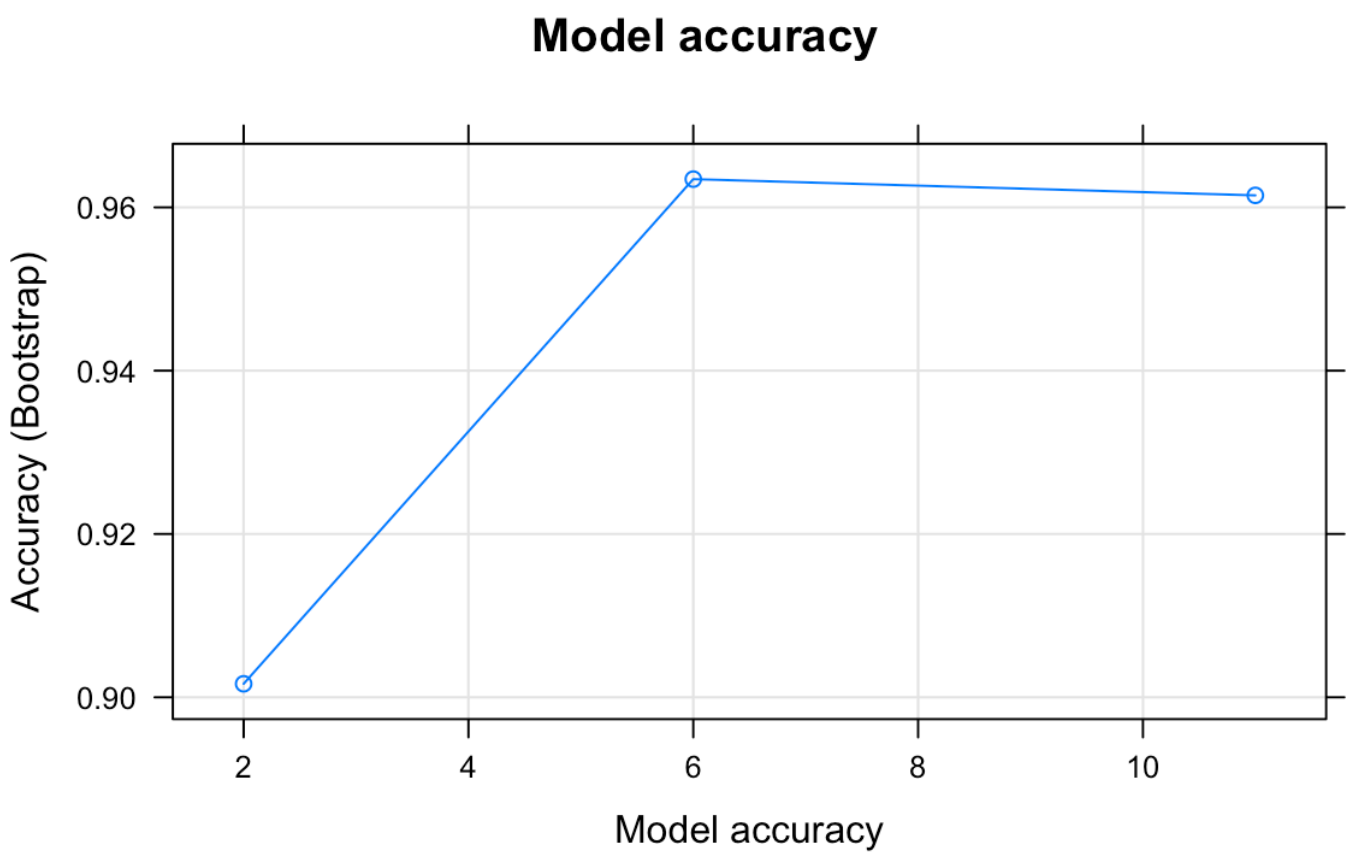
Hide

```
# 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.
```

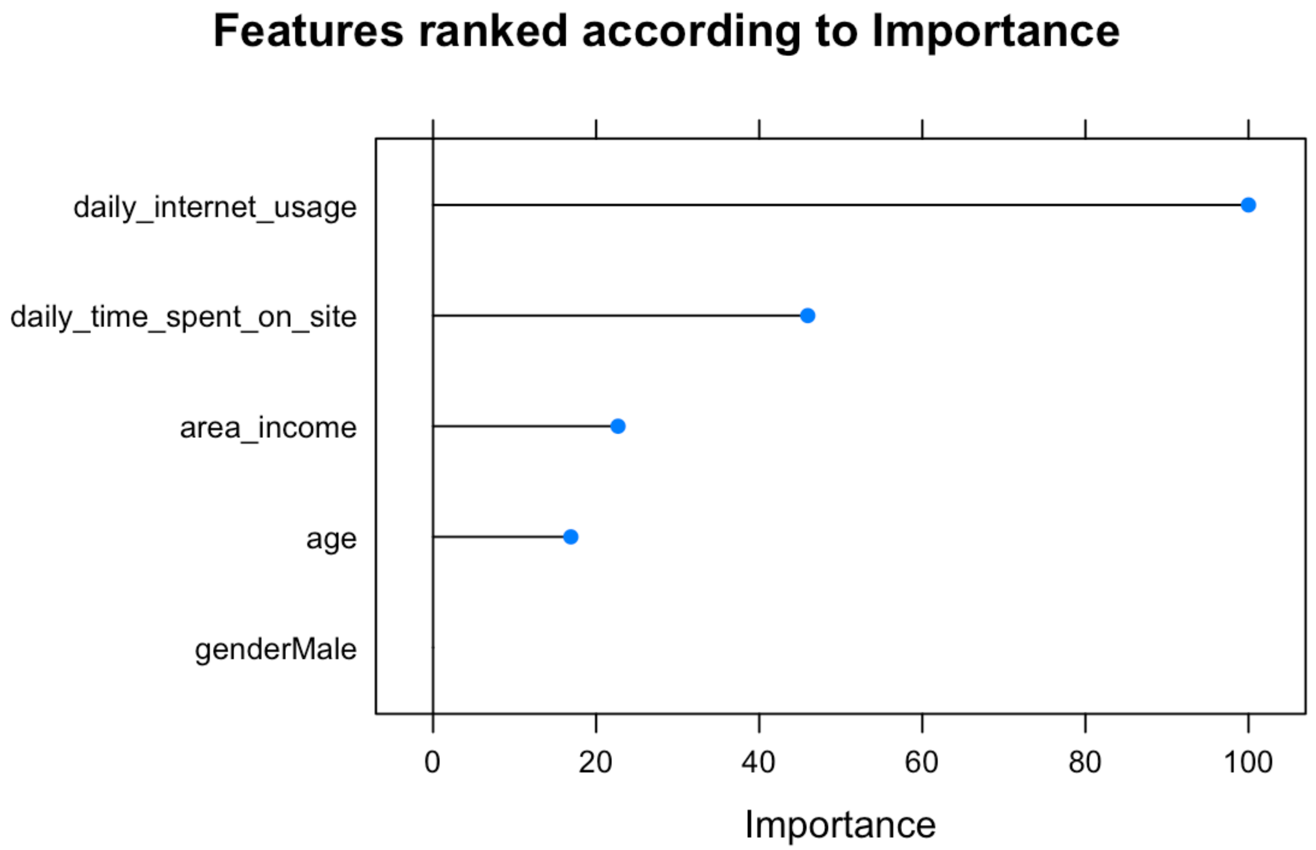
Hide

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




Hide

```
# Checking which features were important in predicting the target variable
important_features = varImp(model)

# Plotting feature importance
plot(important_features, main = 'Features ranked according to Importance')
```




> Features that are seen to be importance are seen below from the most important to the least:

> - Daily internet usage
> - Daily time spent on the side
> - Area Income
> - Age
> - Gender (Male)

## Making prediction

Hide

```
# Previewing the predictions
y_pred = predict(model, test)
head(y_pred)
```

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

# Confusion Matrix

Hide

```
# Displaying 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
```

- We have an accuracy of 97.5% and a P-value of 1

# Cross Validation

Hide

```
library(caret) # automating the tuning process
library(caretEnsemble) # a function for creating ensembles of caret models i.e. ca
retList and caretStack
library(devtools) # a collection of developing tools package
library(usethis) # automate package and project setup tasks that are otherwise per
formed manually
library(earth) # fit MARS models

# Setting parameters
# Defining the training control
fitControl <- trainControl(
    method = 'cv',                   # k-fold cross validation
    number = 5,                      # number of folds
    savePredictions = 'final',       # saves predictions for optimal tuning parame
ter
    classProbs = T,                  # should class probabilities be returned
    summaryFunction=twoClassSummary  # results summary function
)

# Tuning the hyper parameters by setting tuneLength
set.seed(100)
model_2 = train(clicked_on_ad ~ ., data=train, method='earth', tuneLength = 5, met
ric='accuracy', trControl = fitControl)
```

```
The metric "accuracy" was not in the result set. ROC will be used instead.glm.fit:
fitted probabilities numerically 0 or 1 occurredglm.fit: fitted probabilities nume
rically 0 or 1 occurredglm.fit: fitted probabilities numerically 0 or 1 occurredgl
m.fit: fitted probabilities numerically 0 or 1 occurredglm.fit: fitted probabiliti
es numerically 0 or 1 occurredglm.fit: fitted probabilities numerically 0 or 1 occ
urredglm.fit: fitted probabilities numerically 0 or 1 occurredglm.fit: fitted prob
abilities numerically 0 or 1 occurredglm.fit: fitted probabilities numerically 0 o
r 1 occurredglm.fit: fitted probabilities numerically 0 or 1 occurredglm.fit: fitt
ed probabilities numerically 0 or 1 occurredglm.fit: fitted probabilities numerica
lly 0 or 1 occurredglm.fit: fitted probabilities numerically 0 or 1 occurredglm.fi
t: fitted probabilities numerically 0 or 1 occurredglm.fit: fitted probabilities n
umerically 0 or 1 occurredglm.fit: fitted probabilities numerically 0 or 1 occurre
d
```

Hide

```
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.
```

Hide

```
# Predict the test data and computing 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             
```

# Hyperparameter Tuning

> - We are using tuneGrid.The tuneGrid parameter lets us decide which values the main parameter will take,while tuneLength only limit the number of default parameters to use.

Hide

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

# Tuning hyper parameters by setting tuneGrid
set.seed(100)
model_3 = train(clicked_on_ad ~ ., data=train, method='earth', metric='accuracy',
tuneGrid = params, trControl = fitControl)
```

```
The metric "accuracy" was not in the result set. ROC will be used instead.glm.fit:
fitted probabilities numerically 0 or 1 occurredglm.fit: fitted probabilities nume
rically 0 or 1 occurredglm.fit: fitted probabilities numerically 0 or 1 occurredgl
m.fit: fitted probabilities numerically 0 or 1 occurredglm.fit: fitted probabiliti
es numerically 0 or 1 occurredglm.fit: fitted probabilities numerically 0 or 1 occ
urredglm.fit: fitted probabilities numerically 0 or 1 occurredglm.fit: fitted prob
abilities numerically 0 or 1 occurredglm.fit: fitted probabilities numerically 0 o
r 1 occurredglm.fit: fitted probabilities numerically 0 or 1 occurredglm.fit: fitt
ed probabilities numerically 0 or 1 occurredglm.fit: fitted probabilities numerica
lly 0 or 1 occurredglm.fit: fitted probabilities numerically 0 or 1 occurredglm.fi
t: fitted probabilities numerically 0 or 1 occurredglm.fit: fitted probabilities n
umerically 0 or 1 occurredglm.fit: fitted probabilities numerically 0 or 1 occurre
dglm.fit: fitted probabilities numerically 0 or 1 occurredglm.fit: fitted probabil
ities numerically 0 or 1 occurredglm.fit: fitted probabilities numerically 0 or 1
occurred
```

Hide

```
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.
```

Hide

```
# 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
```

## The model’s accuracy is now at 96.5

## Multivariate Adaptive Regression Splines(MARS)

> - It provides a convenient approach to capture the nonlinearity aspect of polynomial refgression by assessing cutpoints (knots), similar to step functions
> - The procedure asseses each data point for each predictor as a kmot and creates a linear regression model with the candidate features.

# Challenging the solution

> The following models will be used to challenge our current solution : * Support Vector Machine * Random Forest * Adaboost * XGBoost

# Support Vector Machine

Hide

```
set.seed(100)

# Train the model using support vector machine
model_svmRadial = train(clicked_on_ad ~ ., data=train, method='svmRadial', tuneLen
gth=1, trControl = fitControl)
```

```
The metric "Accuracy" was not in the result set. ROC will be used instead.
```

Hide

```
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
```

# Random Forest

Hide

```
set.seed(100)

# Train the model using rf
model_rf = train(clicked_on_ad ~ ., data=train, method='rf', tuneLength=5, trContr
ol = fitControl)
```

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

```
The metric "Accuracy" was not in the result set. ROC will be used instead.
```

Hide

```
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.
```

# Adaboost

Hide

```
set.seed(100)

# Training the model using adaboost
model_adaboost = train(clicked_on_ad ~ ., data=train, method='adaboost', tuneLengt
h=2, trControl = fitControl)
```

```
The metric "Accuracy" was not in the result set. ROC will be used instead.
```

Hide

```
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  0.8672969  0.9750  0.9475
  100    Adaboost.M1    0.9846563  0.9700  0.9475
  100    Real adaboost  0.8477031  0.9775  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.
```

# XGBoosts

Hide

```
set.seed(100)

# Train the model using xgboost
model_xgbDART = train(clicked_on_ad ~ ., data=train, method='xgbDART', tuneLength=
1, trControl = fitControl, verbose=F)
```

```
The metric "Accuracy" was not in the result set. ROC will be used instead.
```

Hide

```
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               0.9883437  0.9650  0.9500
  0.3  0.01       0.05       0.8               0.9873906  0.9625  0.9475
  0.3  0.01       0.95       0.6               0.9893125  0.9600  0.9500
  0.3  0.01       0.95       0.8               0.9887187  0.9525  0.9575
  0.3  0.50       0.05       0.6               0.9807813  0.9375  0.9075
  0.3  0.50       0.05       0.8               0.9756406  0.9450  0.9125
  0.3  0.50       0.95       0.6               0.9884688  0.9625  0.9425
  0.3  0.50       0.95       0.8               0.9886563  0.9700  0.9500
  0.4  0.01       0.05       0.6               0.9876094  0.9625  0.9550
  0.4  0.01       0.05       0.8               0.9886875  0.9625  0.9500
  0.4  0.01       0.95       0.6               0.9874688  0.9600  0.9550
  0.4  0.01       0.95       0.8               0.9885156  0.9700  0.9450
  0.4  0.50       0.05       0.6               0.9807969  0.9250  0.9225
  0.4  0.50       0.05       0.8               0.9786094  0.9400  0.9275
  0.4  0.50       0.95       0.6               0.9904219  0.9675  0.9550
  0.4  0.50       0.95       0.8               0.9872812  0.9625  0.9525

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
 = 0.4, gamma = 0, subsample = 0.5, colsample_bytree = 0.6, rate_drop =
 0.5, skip_drop = 0.95 and min_child_weight = 1.
```

# Model Comparison

> - We will go ahead to compare the accuracy’s of the models above and see which is the optimum solution

Hide

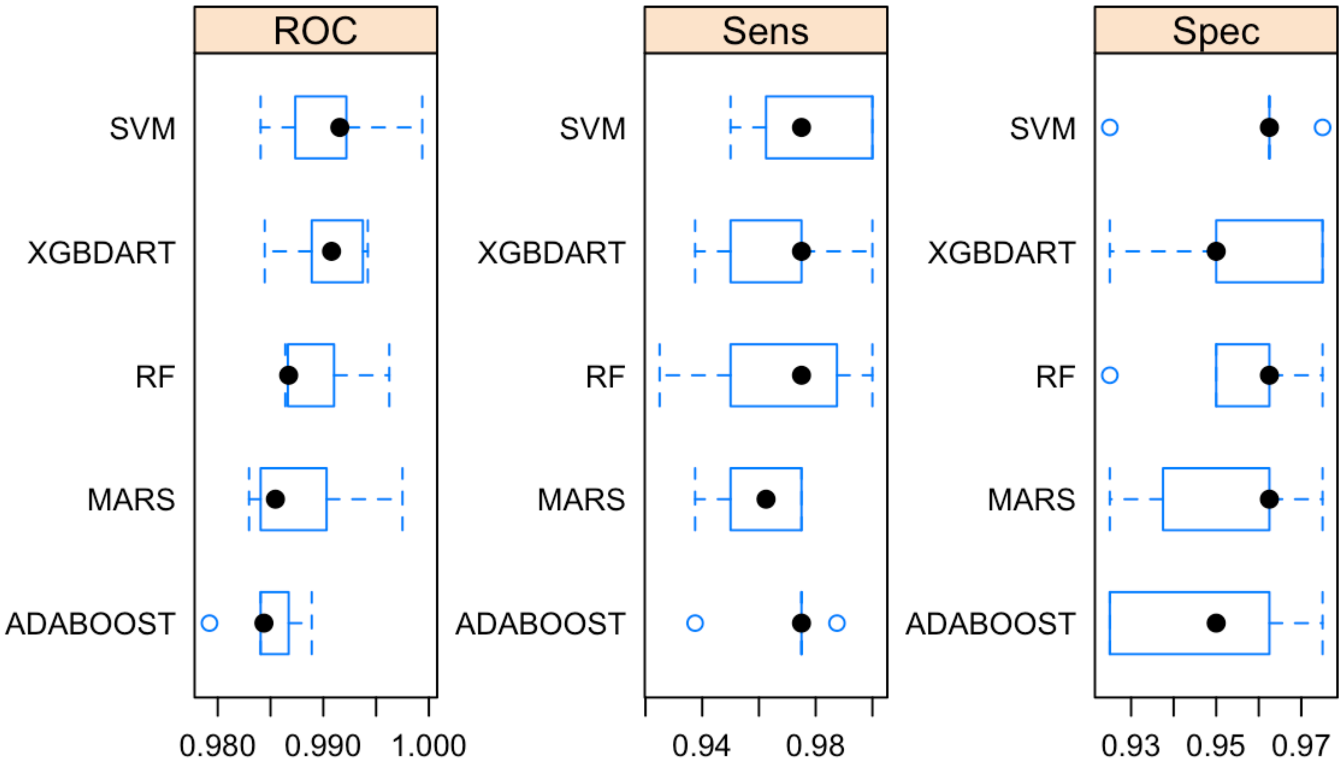
```
# Compare model performances using resample()
model_comparison = resamples(list(MARS=model_3,SVM=model_svmRadial,RF=model_rf,ADA
BOOST=model_adaboost,XGBDART=model_xgbDART))

# Summary of the models performances
summary(model_comparison)
```

```
Call:
summary.resamples(object = model_comparison)

Models: MARS, SVM, RF, ADABOOST, XGBDART
Number of resamples: 5

ROC
              Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
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
RF       0.9864063 0.9866406 0.9867187 0.9894063 0.9910156 0.9962500    0
ADABOOST 0.9792188 0.9840625 0.9843750 0.9846563 0.9867187 0.9889062    0
XGBDART  0.9844531 0.9889062 0.9907813 0.9904219 0.9937500 0.9942188    0

Sens
           Min. 1st Qu. Median   Mean 3rd Qu.   Max. NA's
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
RF       0.9250  0.9500 0.9750 0.9675  0.9875 1.0000    0
ADABOOST 0.9375  0.9750 0.9750 0.9700  0.9750 0.9875    0
XGBDART  0.9375  0.9500 0.9750 0.9675  0.9750 1.0000    0

Spec
          Min. 1st Qu. Median   Mean 3rd Qu.  Max. NA's
MARS     0.925  0.9375 0.9625 0.9525  0.9625 0.975    0
SVM      0.925  0.9625 0.9625 0.9575  0.9625 0.975    0
RF       0.925  0.9500 0.9625 0.9550  0.9625 0.975    0
ADABOOST 0.925  0.9250 0.9500 0.9475  0.9625 0.975    0
XGBDART  0.925  0.9500 0.9500 0.9550  0.9750 0.975    0
```

# A visualization of the Model comparison

Hide

```
# Draw box plots to visualize the comparison
scales = list(x=list(relation="free"), y=list(relation="free"))
bwplot(model_comparison, scales=scales)
```




# Conclusion

> - As can be seen in the chunks above, the Support Vector Machine model has the best accuracy as compared to the rest.
> - If we were provided with more data we would have obtained a greater predictive power.