ONLINE ADVERTISING ANALYSIS

Introduction

a) Specifying the Question

The main objective of the study is to build a model that can help identify which individuals are most likely to click on a Kenyan entrepreneur's online cryptography course ads, using information from this dataset.

b) Defining the Metric for Success

- Determining and visualising the descriptive statistics of the variables in the dataset.
- Determining and visualising the relationships between the status (clicked on ad or not) and the predictor variables.
- Build a model that can predict if an individual will click on an ad.

c) Understanding the context

Advertising is a key aspect of any business, because it is how a business expands its reach. In the past, advertising channels were mainly through newspapers, magazines, billboards, radios, and television. While most of these channels are still relevant, the emergence of online advertising revolutionised the marketing field. It is an effective form of advertising that allows for a more targeted approach as opposed to an overly broad audience for a particular service/product. Collecting general data on individuals who click on a particular online ad helps businesses plan their marketing approach more effectively.

d) Recording the Experimental Design

- Determine the main objectives.
- Load and preview the dataset.
- Understand the data.
- Prepare the dataset - Identify outliers, anomalies, duplicates, missing values, and determine how deal with them, drop unnecessary columns etc.
- Analyse the dataset using univariate, bivariate, and multivariate analysis techniques.
- Challenge the solution.
- Conclusion and recommendations

e) Data Relevance

The dataset provided (here) is relevant to the research question. It has relevant information such as on age, sex, location, whether someone clicked on an ad or not etc.

Loading the dataset

```
library(readr)
library(data.table)

df <- fread("http://bit.ly/IPAdvertisingData")

df <- data.frame(df)
```

Checking the Data

Determining the no. of records in the dataset:

```
dim(df)

## [1] 1000   10

#the dataset has 1000 rows and 10 columns
```

Previewing the top of the dataset:

```
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
```

Previewing the bottom of the dataset:

```
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
```

Checking datatype of each column:

```
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 standardization" "Organic bottom-line service-desk"
"Triple-buffered reciprocal time-frame" ...
##  $ City                    : chr  "Wrightburgh" "West Jodi" "Davidton"
"West Terrifurt" ...
##  $ Male                    : int  0 1 0 1 0 1 0 1 1 1 ...
##  $ Country                 : chr  "Tunisia" "Nauru" "San Marino" "Italy"
...
##  $ Timestamp               : POSIXct, format: "2016-03-27 00:53:11" "2016-
04-04 01:39:02" ...
##  $ Clicked.on.Ad           : int  0 0 0 0 0 0 0 1 0 0 ...
```

Tidying the Dataset

```
#checking column names
colnames(df)
```

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

```
##  [7] "Male"                      "Country"
##  [9] "Timestamp"                 "Clicked.on.Ad"
```

```
#converting column names to lowercase
colnames(df) = tolower(colnames(df))
colnames(df)
```

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

```
#checking for missing values
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
```

There were no missing values in any of the columns

```
#checking for duplicates
df[duplicated(df),]
```

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

There were no duplicate rows

```
#timestamp should be converted to datetime format
str(df$timestamp)
```

```
##  POSIXct[1:1000], format: "2016-03-27 00:53:11" "2016-04-04 01:39:02"
"2016-03-13 20:35:42" ...
```

```
#in character format
```

```
#loading the lubridate library to work with dates
library(lubridate)
```

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

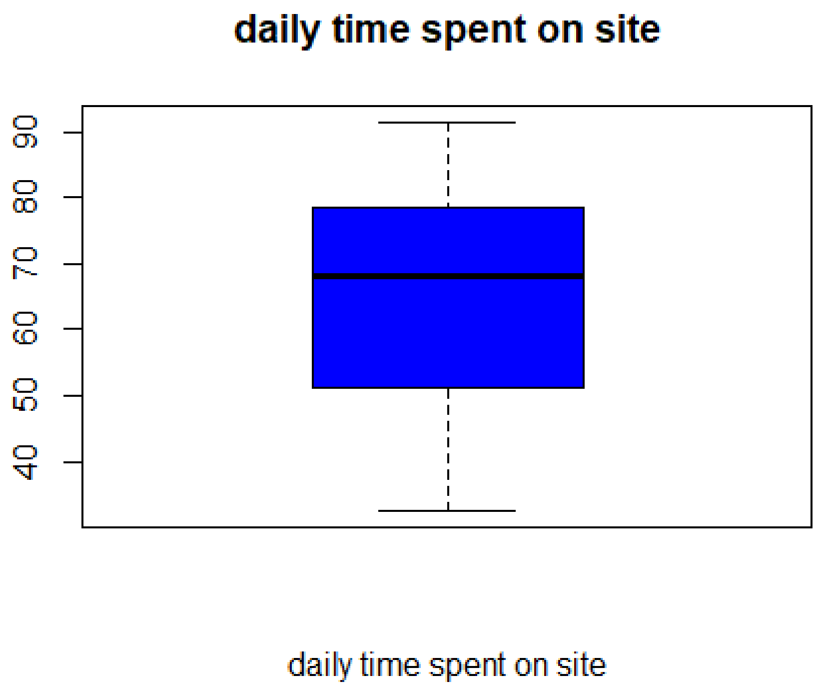
```
## The following objects are masked from 'package:data.table':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year

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

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

#creating columns with months, hour of day, and day of week extracted from
timestamp
df$month = as.factor(month(df$timestamp, label=TRUE))
df$day = as.factor(wday(df$timestamp, label=TRUE))
df$hour = hour(df$timestamp)

#previewing dataframe with new columns
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
```

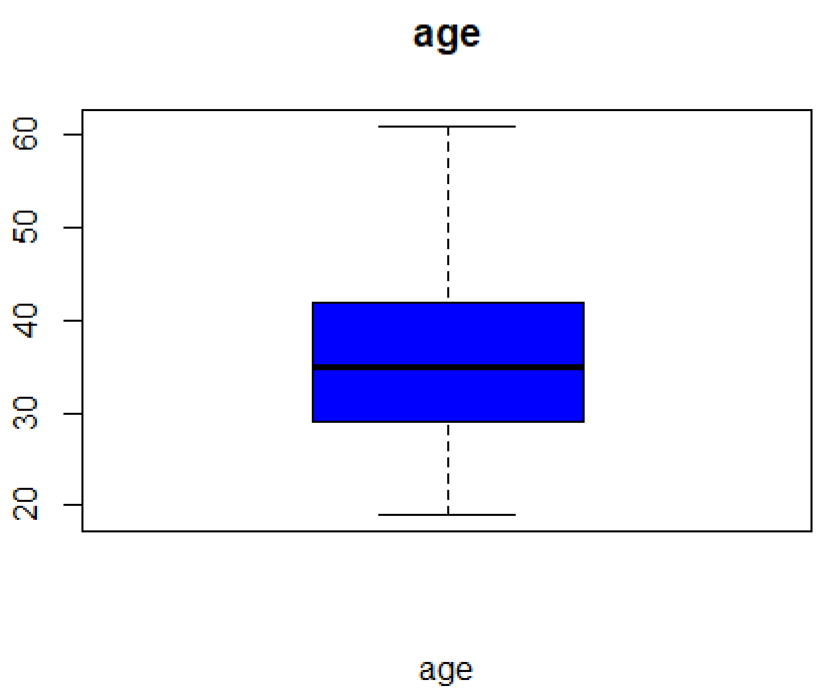
```
## 2    Monitored national standardization      West Jodi    1       Nauru
## 3      Organic bottom-line service-desk       Davidton    0 San Marino
## 4 Triple-buffered reciprocal time-frame West Terrifurt    1       Italy
## 5         Robust logistical utilization   South Manuel    0     Iceland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

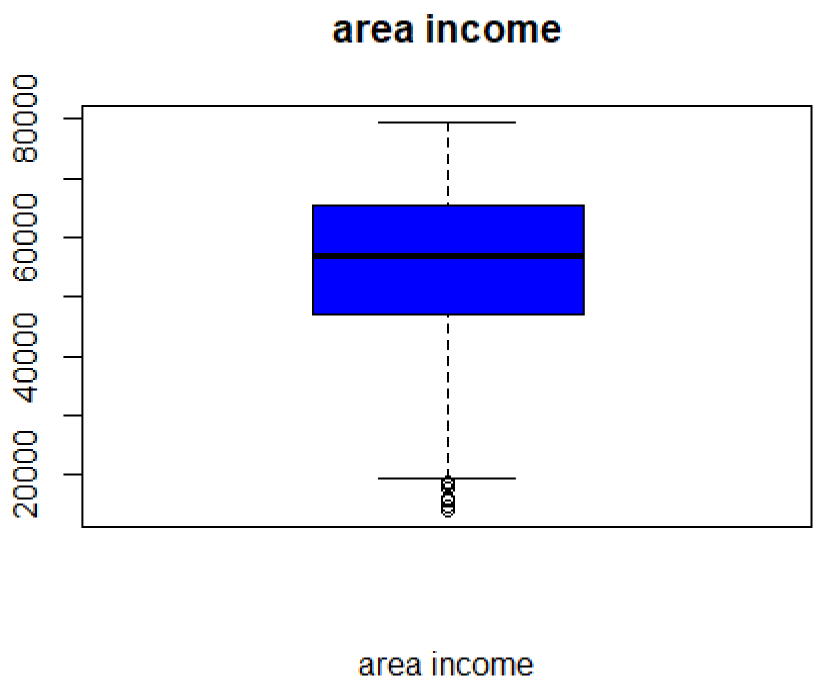
```
#separating continuous and categorical
colnames(df)
```

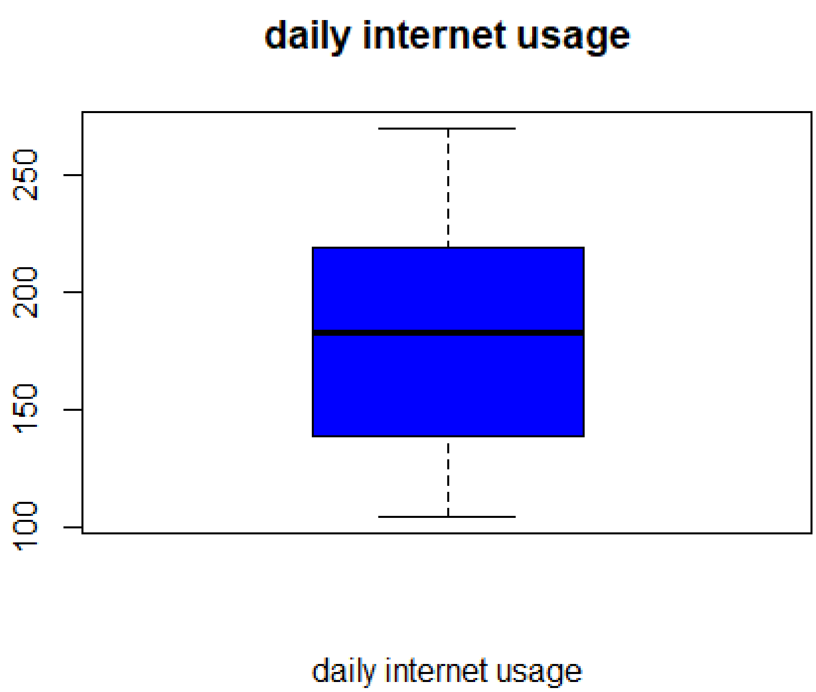
```
##  [1] "daily.time.spent.on.site" "age"
##  [3] "area.income"              "daily.internet.usage"
##  [5] "ad.topic.line"            "city"
##  [7] "male"                     "country"
##  [9] "timestamp"                "clicked.on.ad"
## [11] "month"                    "day"
## [13] "hour"
```

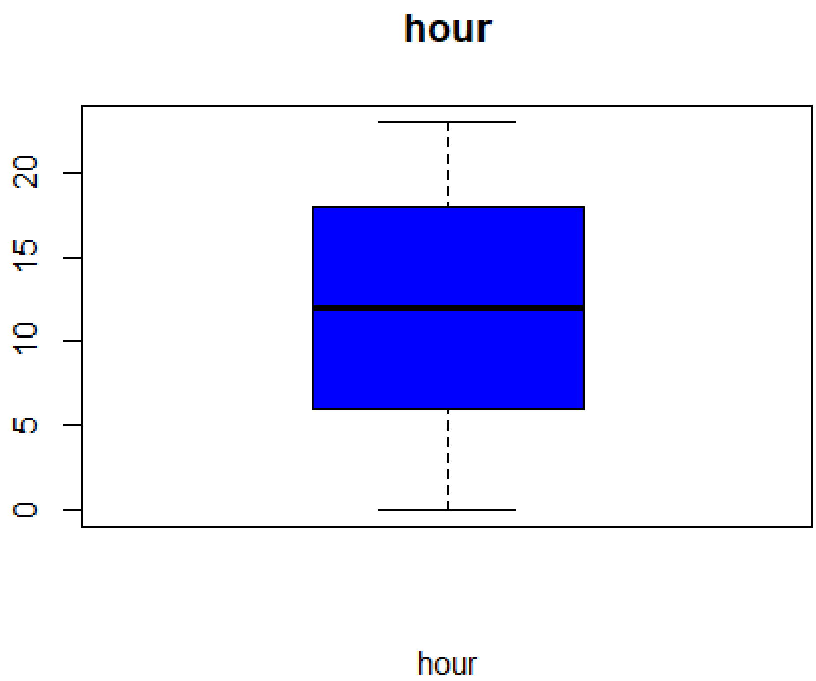
```
contin = c( "daily.time.spent.on.site", "age", "area.income",
            "daily.internet.usage", "hour")
cat = c("ad.topic.line", "city", "male", "country", "day", "month")

#function to replace period in column names with blankspace
repl <- function(x){
  gsub(".", " ", x,fixed=TRUE)

}

#checking for outliers in continuous columns
for (x in contin){
  boxplot(df[x], main=repl(x), xlab=repl(x), col="blue")
}
```


daily time spent on site
90
80
70
60
50
40
daily time spent on site


age
60
50
40
30
20
age


area income
80000
60000
40000
20000
area income


daily internet usage
250
200
150
100
daily internet usage



There were no outliers in daily time sent on site, age, daily internet usage and hour columns. There were some outliers in the area income column

```
boxplot.stats(df$area.income)$out

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

The outliers will not be dropped because it is expected that some areas have lower income than others

```
#checking for number of unique values in categorical columns
for (x in cat){
  print(paste(x, length(unique(df[[x]]))))
}

## [1] "ad.topic.line 1000"
## [1] "city 969"
## [1] "male 2"
## [1] "country 237"
## [1] "day 7"
## [1] "month 7"
```

Ad topic line is made up entirely of unique values. Will drop it.

```
#dropping ad topic line column
df = subset(df, select=-c(ad.topic.line))
colnames(df)
```

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

```
##checking for anomalies in categorical columns
for (x in cat[3:6]){
  if (x != "country"){

    print(paste(x, unique(df[x])))

  }
}
```

```
## [1] "male 0:1"
## [1] "day c(1, 2, 6, 5, 4, 7, 3)"
## [1] "month c(3, 4, 1, 6, 5, 7, 2)"
```

```
#no anomalous values
```

Univariate Analysis

```
#loading ggplot 2 library for visualisation
library(ggplot2)
```

```
#loading psych library o use statistical functions
library("psych")
```

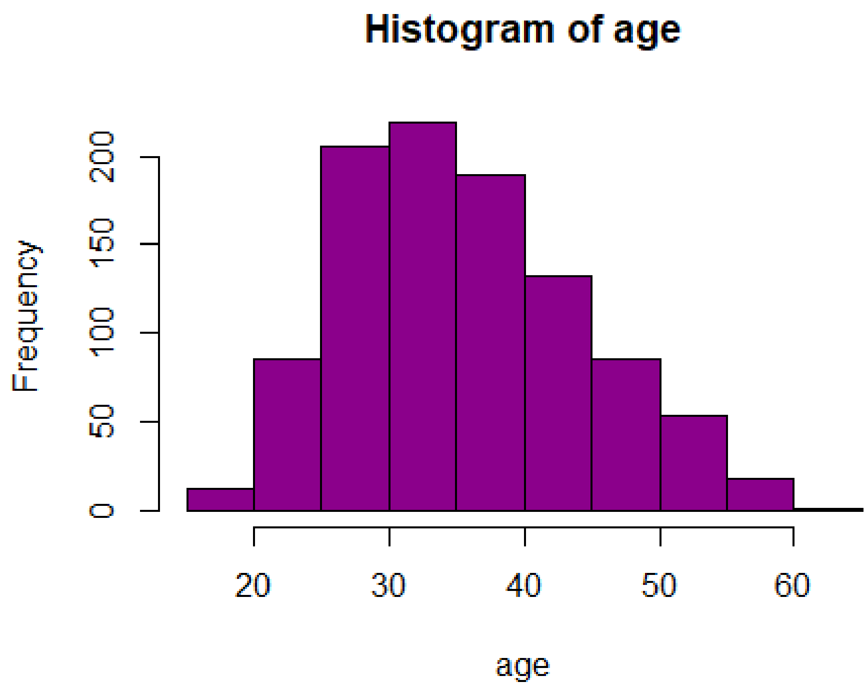
```
##
## Attaching package: 'psych'
```

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

```
#statistical sumary of age variable
describe(df$age)
```

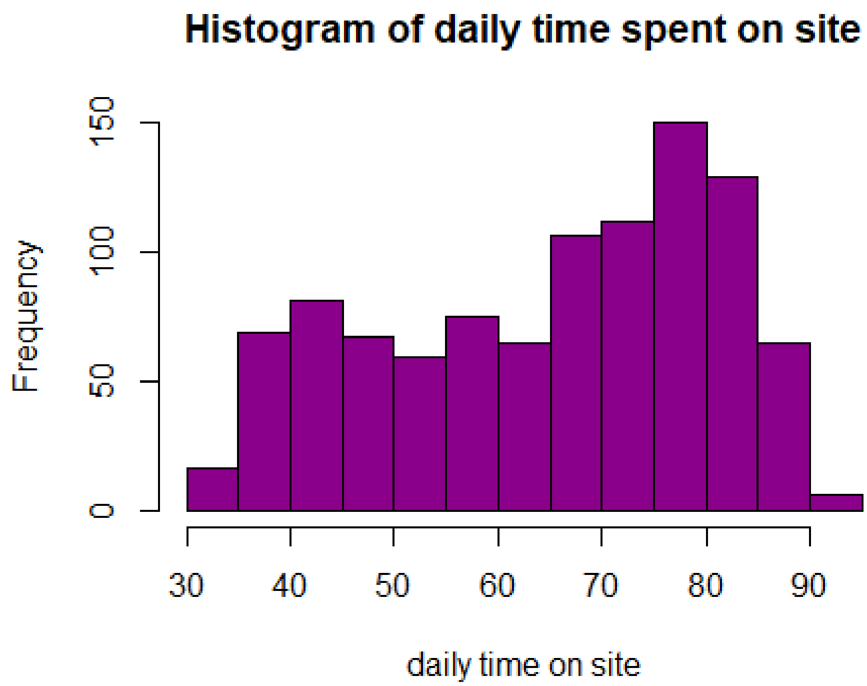
```
##    vars    n  mean   sd median trimmed mad min max range skew kurtosis
se
## X1    1 1000 36.01 8.79     35   35.51 8.9  19  61    42 0.48    -0.41
0.28
```

```
#plotting age histogram
hist(df$age, col="darkmagenta",
     main="Histogram of age",
     xlab="age")
```



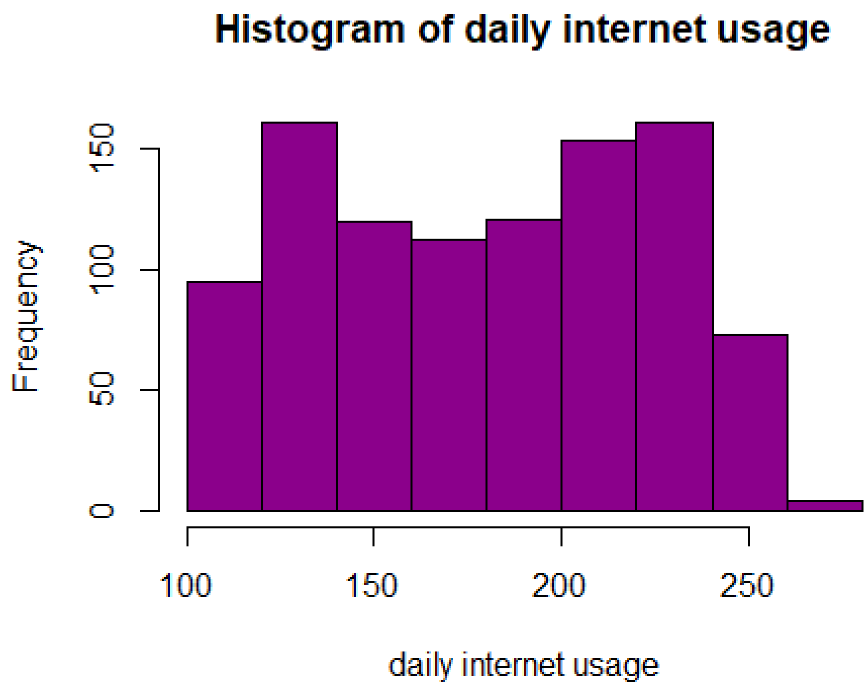
Most people in the dataset were between 30-35

```
#statistical sumary of daily time on site variable
describe(df$daily.time.spent.on.site)

##    vars    n mean    sd median trimmed   mad  min   max range  skew
kurtosis
## X1    1 1000   65 15.85  68.22   65.74 17.92 32.6 91.43 58.83 -0.37     -
1.1
##     se
## X1 0.5

#histogram of daily time on site
hist(df$daily.time.spent.on.site, col="darkmagenta",
     main="Histogram of daily time spent on site",
     xlab="daily time on site")
```



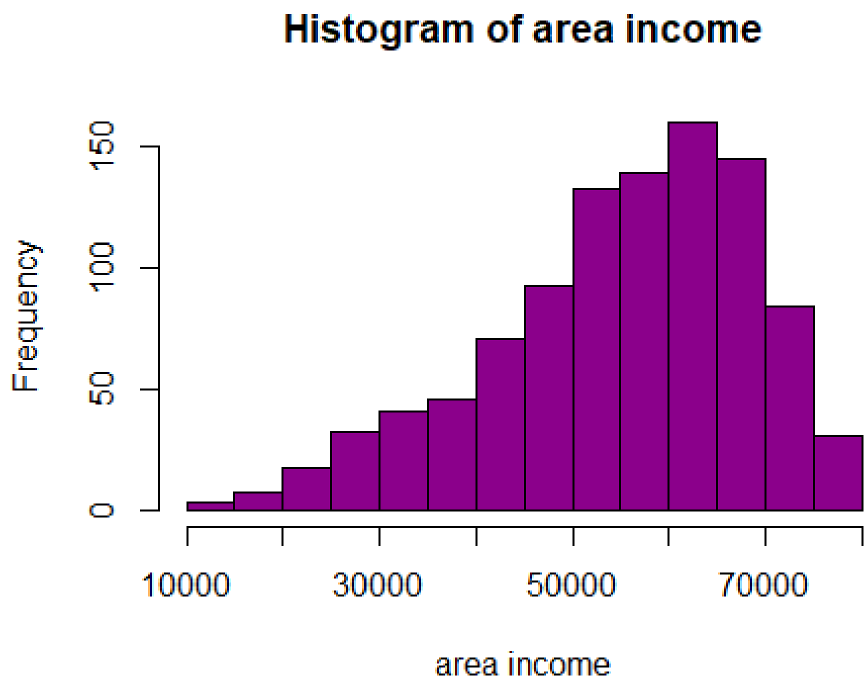
Most of the people spent 75-80 minutes on the site daily

```
#statistical sumary of daily internet usage variable
describe(df$daily.internet.usage)

##    vars    n mean   sd median trimmed   mad    min    max  range  skew
kurtosis
## X1    1 1000  180 43.9 183.13  179.99 58.61 104.78 269.96 165.18 -0.03
-1.28
##       se
## X1 1.39

#histogram of daily internet usage
hist(df$daily.internet.usage, col="darkmagenta",
     main="Histogram of daily internet usage",
     xlab="daily internet usage")
```



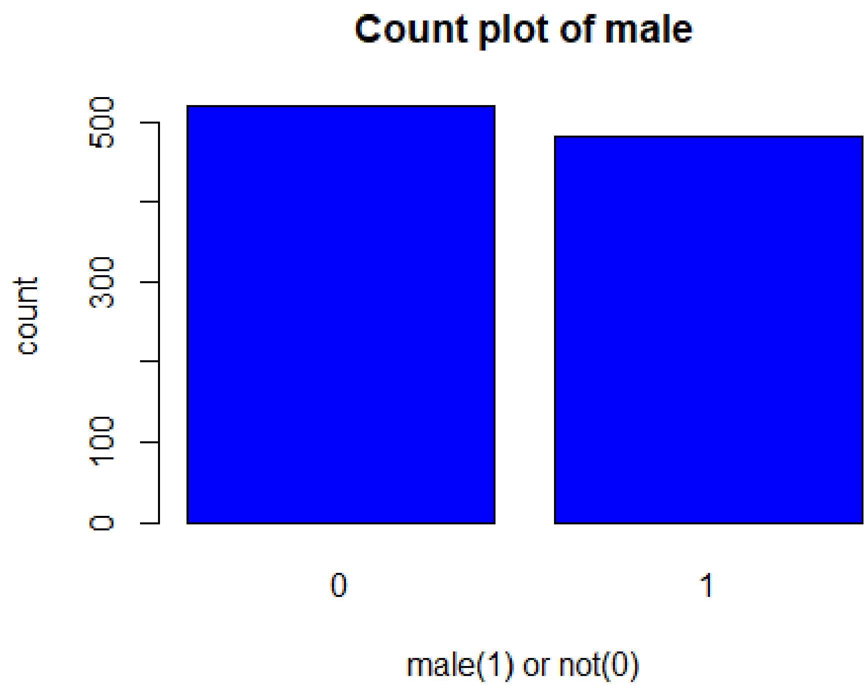
For most in the dataset, daily internet usage was between 120 and 140.

```
#statistical summary of area income variable
describe(df$area.income)

##    vars    n  mean       sd  median  trimmed      mad     min     max
range
## X1    1 1000 55000 13414.63 57012.3 56038.94 13316.62 13996.5 79484.8
65488.3
##     skew kurtosis     se
## X1 -0.65    -0.11 424.21

#histogram of area income
hist(df$area.income, col="darkmagenta",
     main="Histogram of area income",
     xlab="area income")
```



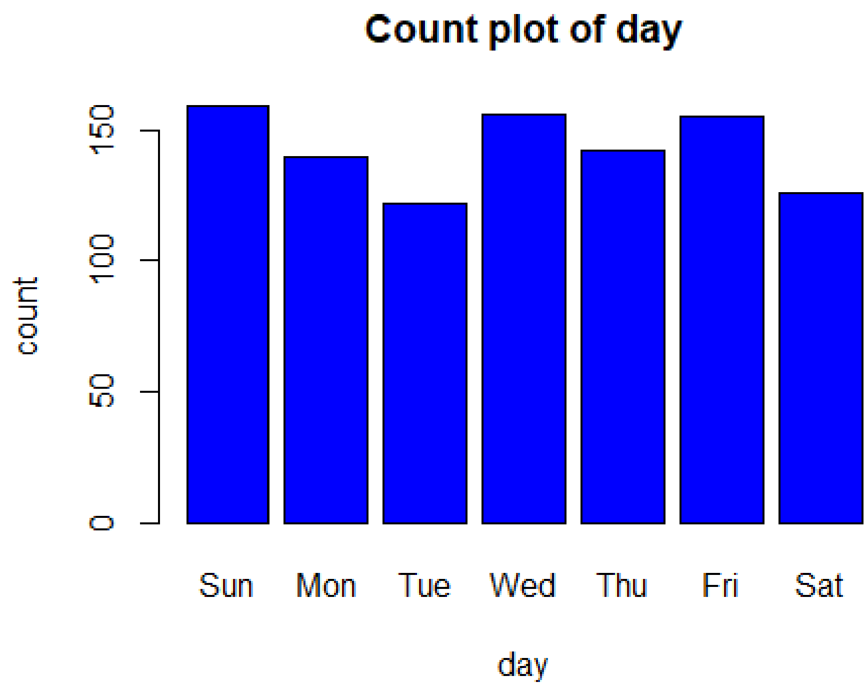
Most of the area income values lied between 60000 to 65000

```
#Count plot of male
barplot(table(df$male), col="blue", main="Count plot of male",
        xlab = "male(1) or not(0)", ylab="count")
```



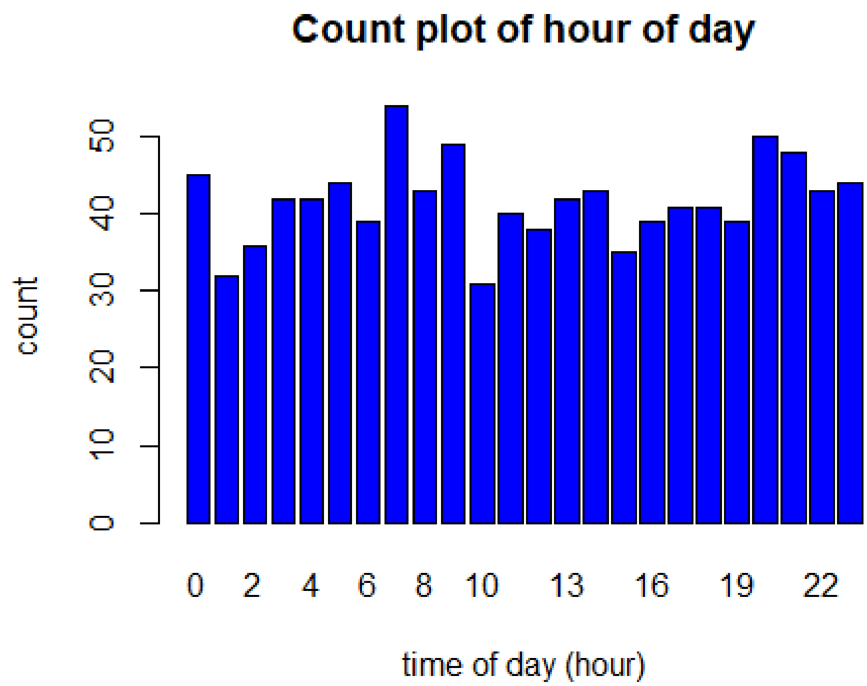
There were less males than females in the dataset

```
#count plot of day of week
barplot(table(df$day), col="blue", main="Count plot of day",
        xlab = "day", ylab="count")
```



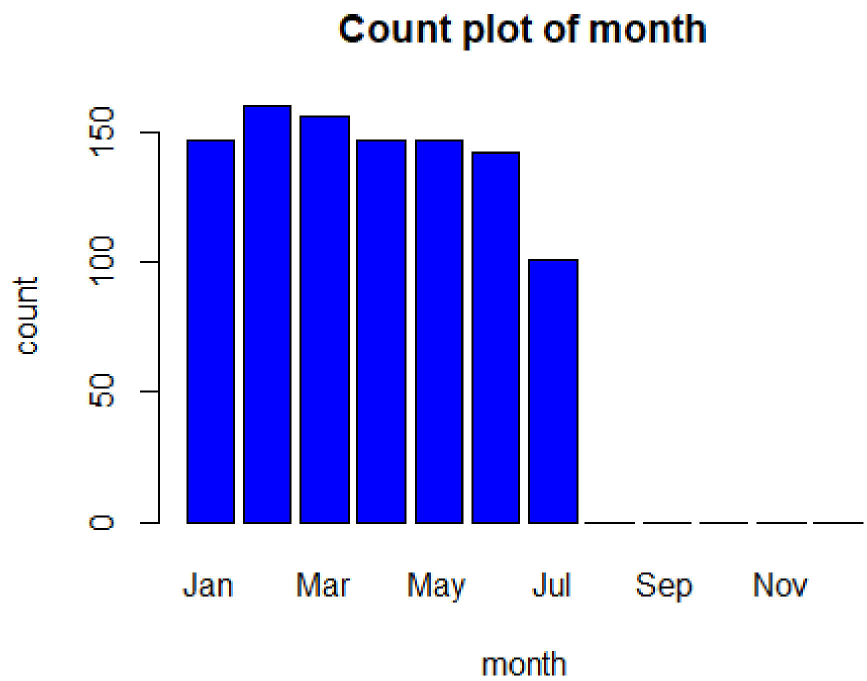
Sunday was the most represented day in the dataset

```
#frequency table of day column highlighting the observation above further
table(df$day)

##
## Sun Mon Tue Wed Thu Fri Sat
## 159 140 122 156 142 155 126

#count plot of  hour of day
barplot(table(df$hour), col="blue", main="Count plot of hour of day",
        xlab = "time of day (hour)", ylab="count")
```



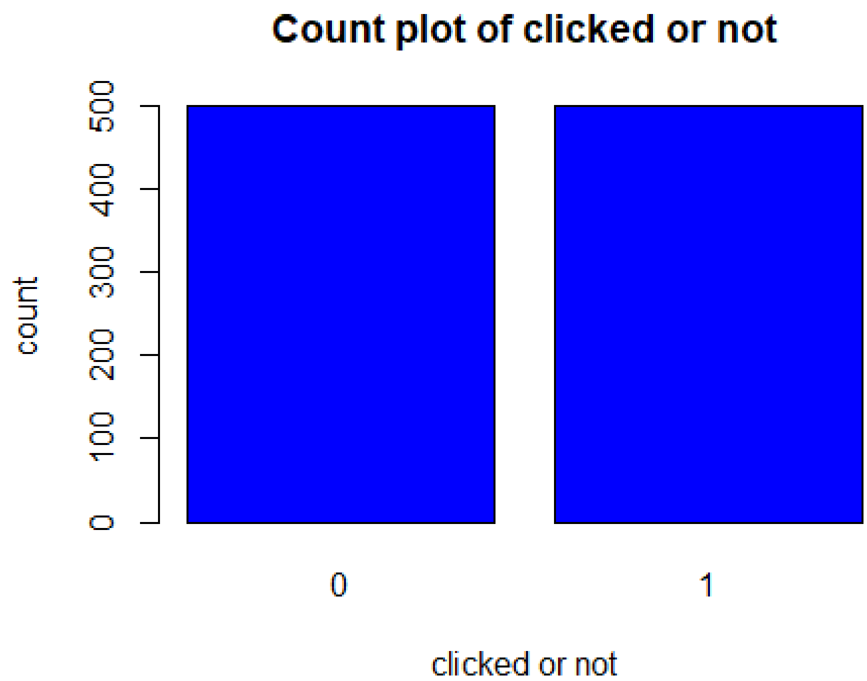
Most observations had a timestamp of 7 am

```
#count plot of month
barplot(table(df$month), col="blue", main="Count plot of month",
        xlab = "month", ylab="count")
```



Most observations were from February

```
#frequency table highlighting that observations were from January to July,
with most in February
table(df$month)

##
## Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
## 147 160 156 147 147 142 101   0   0   0   0   0

#count plot of clicked or not
barplot(table(df$clicked.on.ad), col="blue",
        main="Count plot of clicked or not",
        xlab = "clicked or not", ylab="count")
```



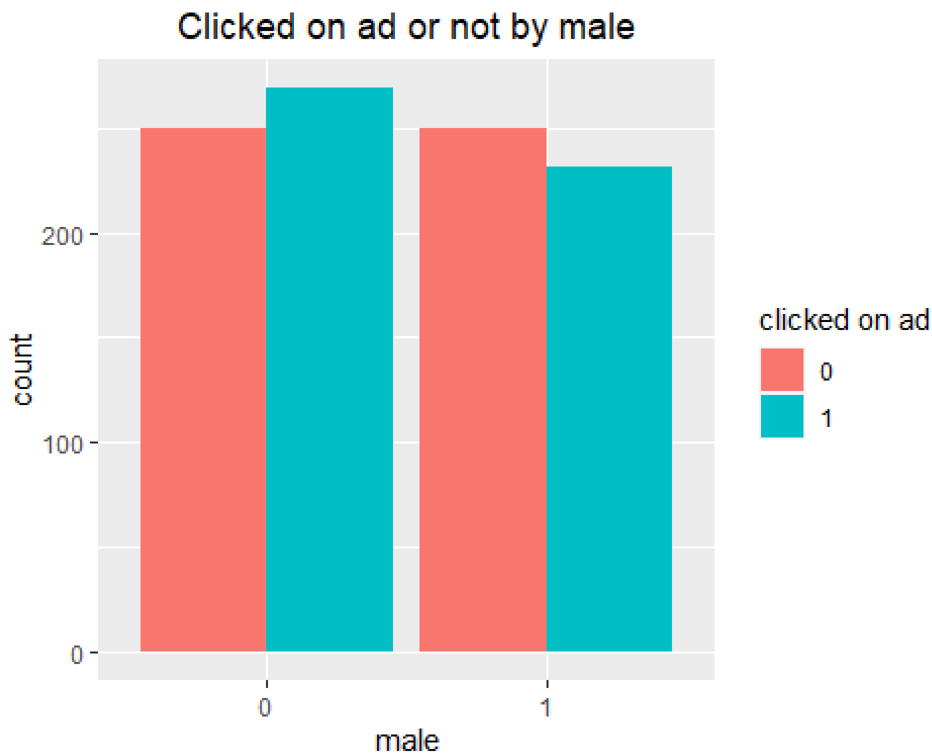
The dataset was balanced. There was an equal number of observations who clicked on the ad and those who did not.

```
table(df$clicked.on.ad)

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

Bivariate Analysis

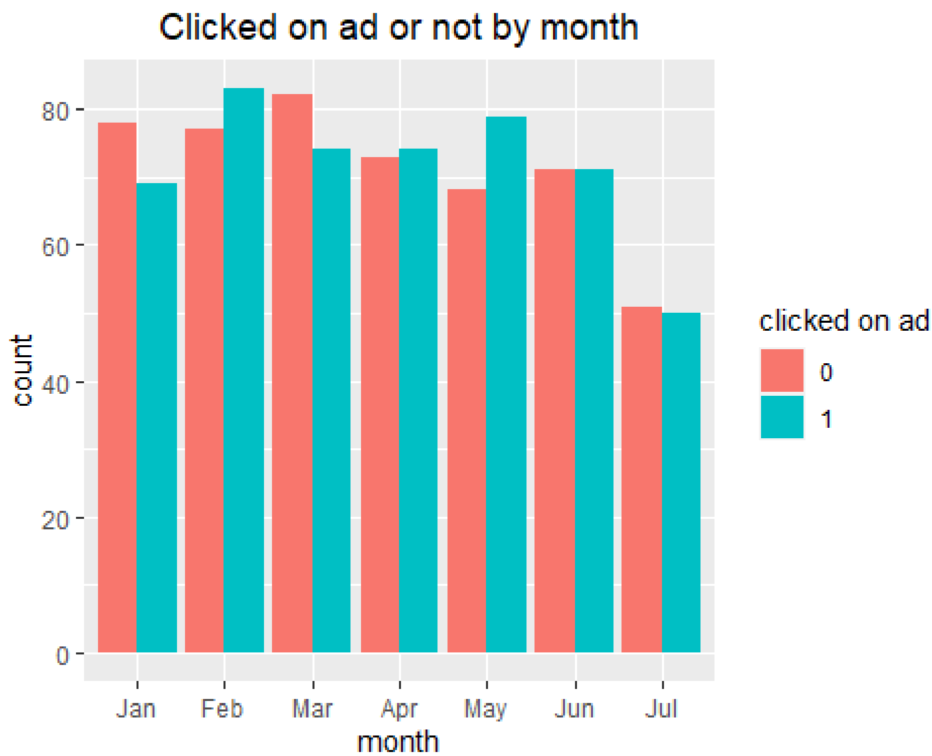
```
#loading library to use functions
library("dplyr")

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:data.table':
##
##     between, first, last

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

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

```
#plotting clicked on ad by male
ggplot() + geom_bar(
    data=df,
    aes(x=factor(male), fill = factor(clicked.on.ad)
    ), position="dodge") + labs(title = "Clicked on ad or not by male",
          y="count", x="male", fill="clicked on ad") + theme(plot.title =
element_text(hjust=0.5))
```



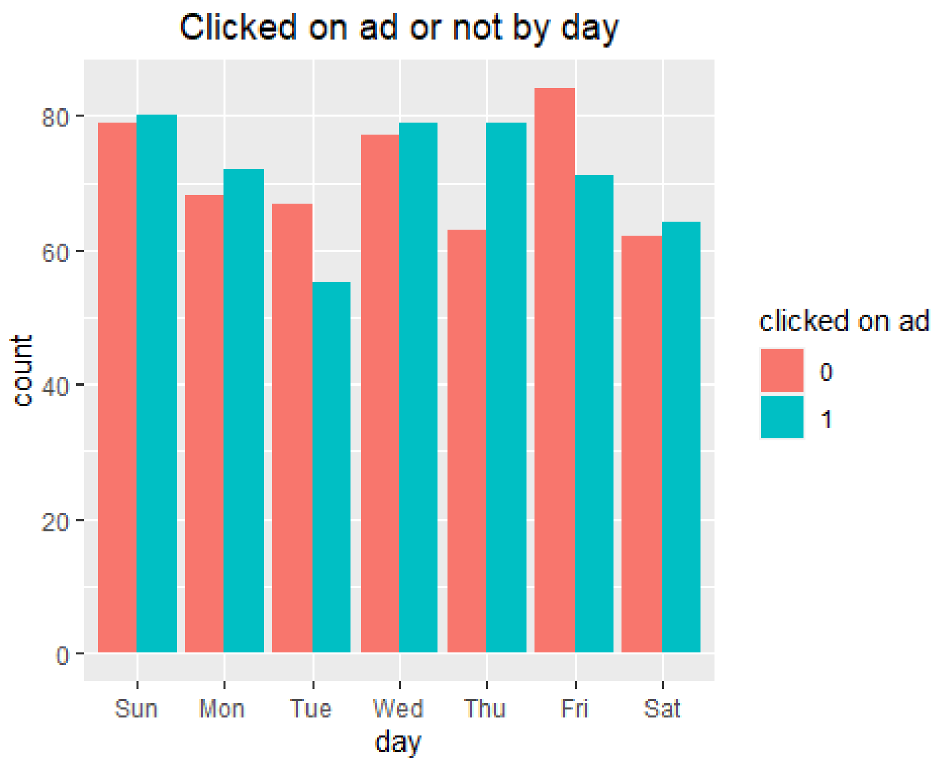
Among the females, the majority clicked on the ad while among males the majority did not click on the ad.

```
#clicked on ad by month
ggplot() + geom_bar(
    data=df,
    aes(x=factor(month), fill = factor(clicked.on.ad)
    ), position="dodge") + labs(title = "Clicked on ad or not by month",
          y="count", x="month", fill="clicked on ad") + theme(plot.title =
element_text(hjust=0.5))
```



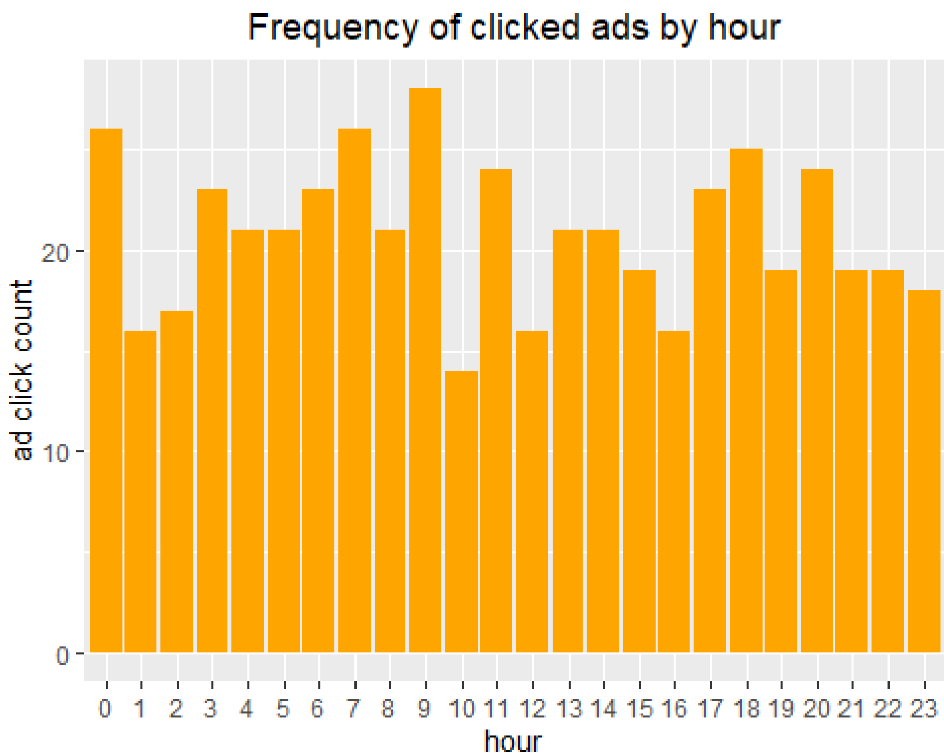
February followed by May had the highest frequencies of ad clicks. Additionally, the proportions of those who clicked on the ad were higher than those that did not in those months.

```
#clicked on ad by day
ggplot() + geom_bar(
    data=df,
    aes(x=factor(day), fill = factor(clicked.on.ad)
    ), position="dodge") + labs(title = "Clicked on ad or not by day",
           y="count", x="day", fill="clicked on ad") + theme(plot.title =
element_text(hjust=0.5))
```



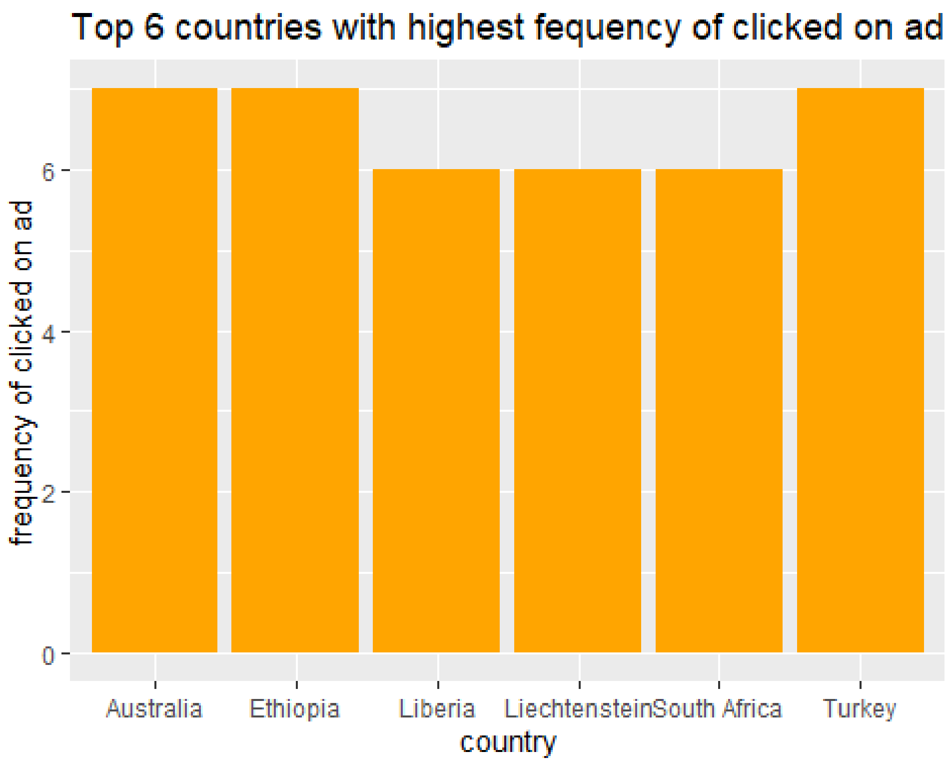
Sunday followed by Wednesday and Thursday had the highest frequencies of ad clicks.

```
#ad clicks by hour
ggplot() + geom_bar(
    data=filter(df, clicked.on.ad == 1),
    aes(x=factor(hour)), fill="orange") + labs(title = "Frequency of clicked
ads by hour",
           y="ad click count", x="hour") + theme(plot.title =
element_text(hjust=0.5))
```



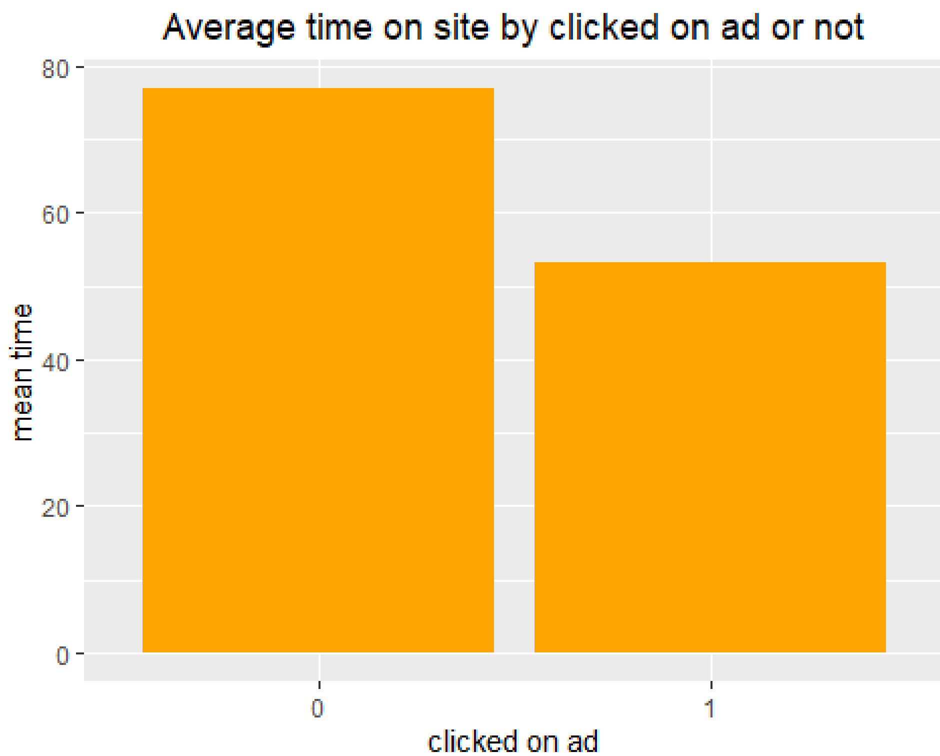
9 a.m. was the hour with the most ad clicks

```
#ad clicks by country
clicked <- data.frame(table(filter(df, clicked.on.ad == 1)$country))
clicked <- head(clicked[order(clicked$Freq, decreasing = TRUE),])

ggplot() + geom_col(
    data=clicked,
    aes(x=Var1, y=Freq),
    fill="orange") + labs(title = "Top 6 countries with highest fequency of
clicked on ad",
           x="country", y="frequency of clicked on ad") + theme(plot.title =
element_text(hjust=0.5))
```



The top 3 countries with the highest frequencies of ad clicks were Ethiopia, Australia and Turkey.

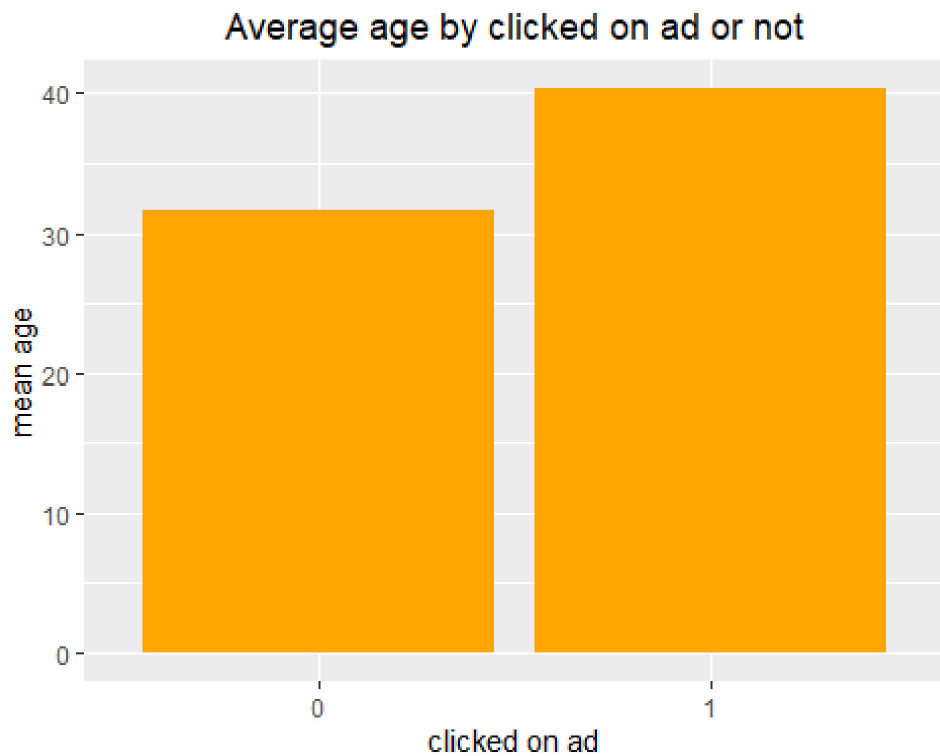
```
#creating data frame with average time on site by clicked ad
time = df %>% group_by(clicked.on.ad) %>%
  summarise(mean_time=mean(daily.time.spent.on.site))
time

## # A tibble: 2 × 2
##   clicked.on.ad mean_time
##           <int>     <dbl>
## 1             0      76.9
## 2             1      53.1

#plotting above
ggplot() + geom_col(
    data=time,
    aes(x=as.factor(clicked.on.ad), y=mean_time),
    fill="orange") + labs(title = "Average time on site by clicked on ad or 
not",
           y="mean time", x="clicked on ad") + theme(plot.title = 
element_text(hjust=0.5))
```




```
#    + ggtitle("Average time on site by clicked on ad or not")
```

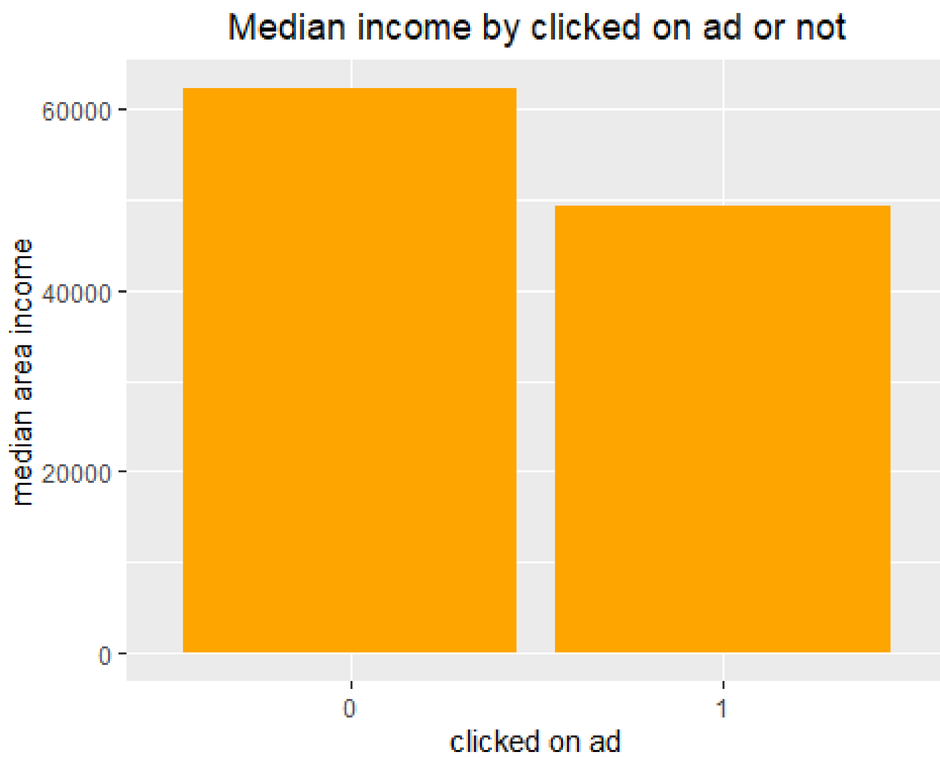
The average time on the site for those who clicked on the ad was lower than for those who did not click on the ad

```
#average age by clicked on ad
age = df %>% group_by(clicked.on.ad) %>%
  summarise(mean_age=mean(age))

ggplot() + geom_col(
    data=age,
    aes(x=as.factor(clicked.on.ad), y=mean_age),
    fill="orange") + labs(title = "Average age by clicked on ad or not",
           y="mean age", x="clicked on ad") + theme(plot.title =
element_text(hjust=0.5))
```



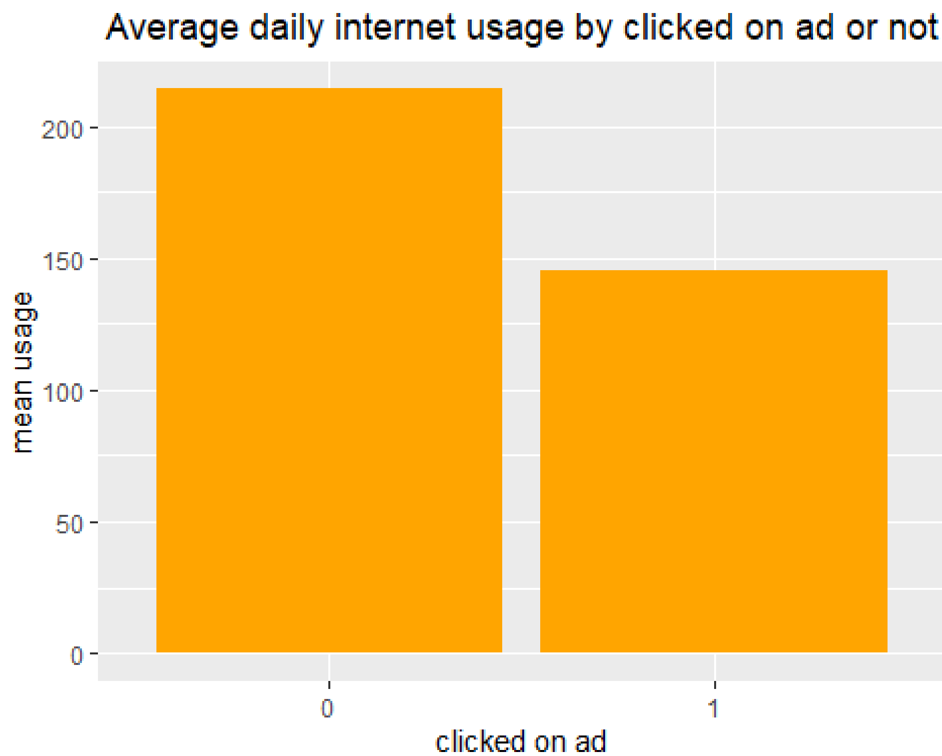
The average age of those who clicked on the ad was higher than the average age of those who did not.

```
#meadian area income by clicked on ad
areainc = df %>% group_by(clicked.on.ad) %>%
  summarise(median_areainc=median(area.income))

ggplot() + geom_col(
    data=areainc,
    aes(x=as.factor(clicked.on.ad), y=median_areainc),
    fill="orange") + labs(title = "Median income by clicked on ad or not",
           y="median area income", x="clicked on ad") + theme(plot.title =
element_text(hjust=0.5))
```



The median area income of those who did not click on the ad was higher than that of those who clicked on the ad.

```
#internet usage by clicked or not
usage = df %>% group_by(clicked.on.ad) %>%
  summarise(mean_usage=mean(daily.internet.usage))

ggplot() + geom_col(
    data=usage,
    aes(x=as.factor(clicked.on.ad), y=mean_usage),
    fill="orange") + labs(title = "Average daily internet usage by clicked on 
ad or not",
          y="mean usage", x="clicked on ad") + theme(plot.title = 
element_text(hjust=0.5))
```



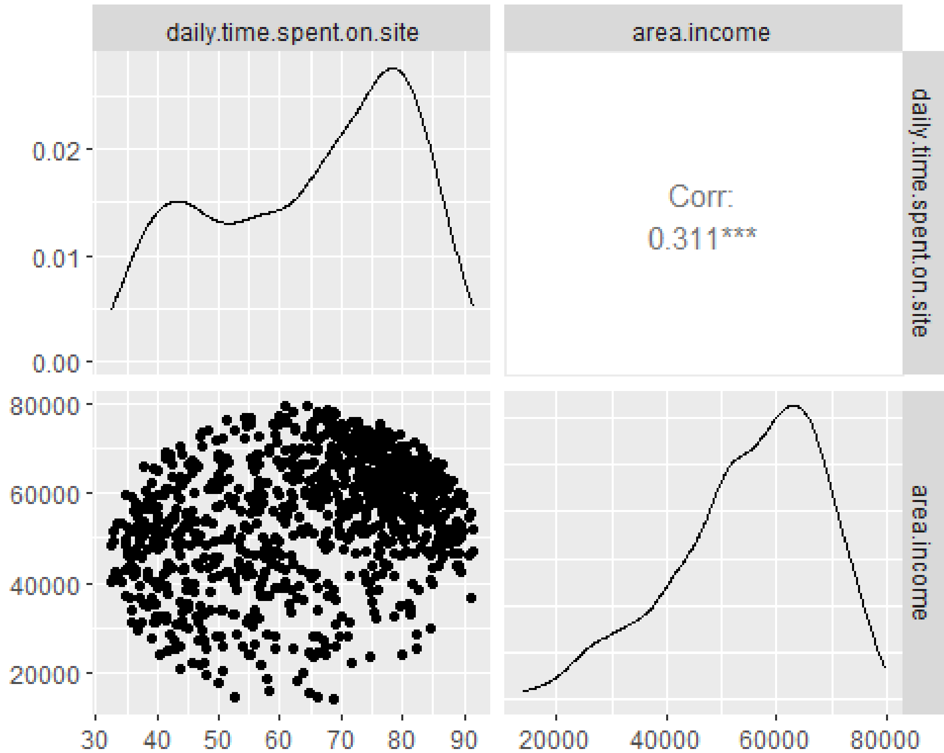
The average daily internet usage of those who clicked on the ad was lower than that of those who did not click on the ad.

Scatterplots of continuous columns

```
#continuous columns
contin[1:4]

## [1] "daily.time.spent.on.site" "age"
## [3] "area.income"              "daily.internet.usage"

#creating dataframe that containing the continuous variables
scatterp = subset(df, select = c(daily.time.spent.on.site, area.income, age,
daily.internet.usage))
head(scatterp)

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

#loading library for pair plot
library(GGally)
```

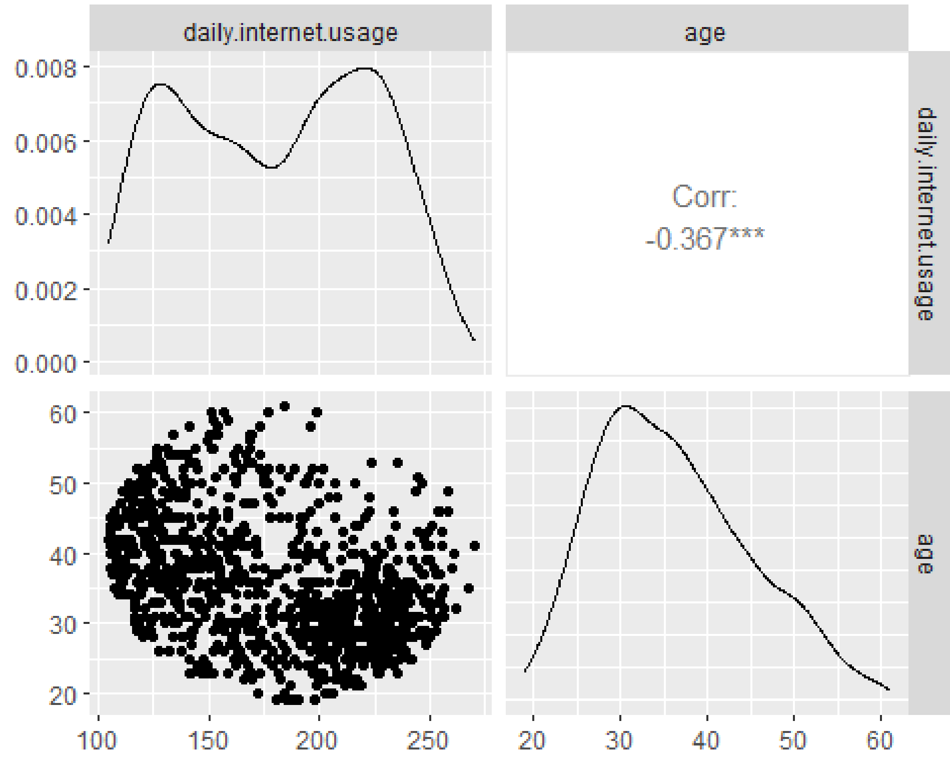
```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
#plotting scatterplots of continuous variables
ggpairs(subset(df, select = c(daily.time.spent.on.site, area.income)))
```



There is a moderate positive correlation between daily time spent on site and area income.

```
ggpairs(subset(df, select = c(daily.internet.usage, age)))
```



There is a moderate negative correlation between age and daily internet usage.

Correlation matrix

```
#converting categorical to numerical
#removing timestamp column
#dataframe for correlation matrix
cor_matrix_df <- subset(df, select = -timestamp)

cor_matrix_df$city <- as.numeric(factor(cor_matrix_df$city))
cor_matrix_df$country <- as.numeric(factor(cor_matrix_df$country))
cor_matrix_df$month <- as.numeric(factor(cor_matrix_df$month))
cor_matrix_df$day <- as.numeric(factor(cor_matrix_df$day))

#checking that datatype conversion worked
str(cor_matrix_df)

## 'data.frame':    1000 obs. of  11 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 ...
##  $ city                    : num  962 904 112 940 806 283 47 672 885 713
...
##  $ male                    : int  0 1 0 1 0 1 0 1 1 1 ...
##  $ country                 : num  216 148 185 104 97 159 146 13 83 79 ...
##  $ clicked.on.ad           : int  0 0 0 0 0 0 0 1 0 0 ...
```

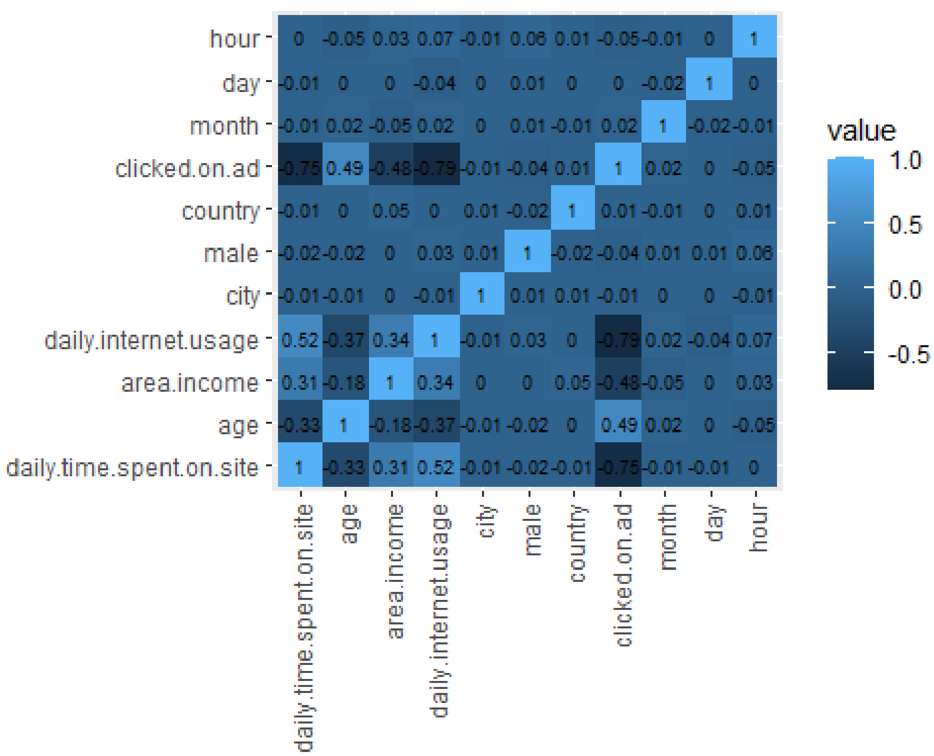
```
##  $ month                       : num  3 4 3 1 6 5 1 3 4 7 ...
##  $ day                         : num  1 2 1 1 6 5 5 2 2 2 ...
##  $ hour                        : int  0 1 20 2 3 14 20 1 9 1 ...
```

```r
library(reshape2)
```

```
##
## Attaching package: 'reshape2'
```

```
## The following objects are masked from 'package:data.table':
##
##     dcast, melt
```

```r
#plotting the correlation heatmap
datam = melt(round(cor(cor_matrix_df),2))
ggplot(data=datam, aes(x=Var1, y=Var2, fill=value)) + geom_tile() +
geom_text(aes(Var2, Var1, label=value), color="black",size=2.5) +
theme(axis.text.x=element_text(angle=90,vjust=0.5,hjust=1), axis.title.x =
element_blank(), axis.title.y = element_blank())
```



The main column of interest is clicked on ad. According to the correlation heatmap above, clicked on ad seems to be most strongly correlated to daily internet usage, daily time spent on site, age, and area income in that order. Modelling will reveal more on these relationships

Modelling

```r
df3 <- copy(cor_matrix_df)
```

```
library(caret)

## Loading required package: lattice

#stratified train test split
set.seed(123)
train <- createDataPartition(df3$clicked.on.ad, p=.7, list=FALSE)
# training set
st_train <- df3[train,]
# test set
st_test <- df3[-train,]

#X and y train and test
X_train <- subset(st_train, select=-clicked.on.ad)
y_train <- subset(st_train, select=clicked.on.ad)
X_test <- subset(st_test, select=-clicked.on.ad)
y_test <- subset(st_test, select=clicked.on.ad)
```

1. KNN

```
#scaling the features

#scaling
X_train_sc <- scale(X_train)

#transforming test based on values obtained by test (scaler should only be
fitted on train set then used to transform both train and test to prevent
data leakage caused by fitting on entire dataset)

X_test_sc <- scale(X_test, center=attr(X_train_sc, "scaled:center"),
scale=attr(X_train_sc, "scaled:scale"))

table(df$clicked.on.ad)

##
##   0   1
## 500 500

#the dataset is balanced therefore accuracy metric can give accurate measure
of performance

# the "class" package contains the K-NN algorithm.

# cl is the class of the training data set and k is the no of neighbours to
look for


library(class)
require(class)
model <- knn(train= X_train_sc[,],test=X_test_sc[,], cl= y_train[,],k=5)

library("Metrics")
```

```
##
## Attaching package: 'Metrics'

## The following objects are masked from 'package:caret':
##
##     precision, recall

#confusion matrix
table(model, y_test[,])

##
## model   0   1
##     0 150  14
##     1   0 136

#the dataset s balanced so accuracy is an appropriate metric of evaluation
table(factor(model))

##
##   0   1
## 164 136

accuracy(y_test[,], model)

## [1] 0.9533333

#95.3% accuracy is good. 14 out of 150 for positive class are false negatives

#more detailed evaluation
caret::confusionMatrix(data=as.factor(model), reference=as.factor(y_test[,]),
positive="1")

## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 150  14
##          1   0 136
##
##                Accuracy : 0.9533
##                  95% CI : (0.9229, 0.9743)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9067
##
##  Mcnemar's Test P-Value : 0.000512
##
##             Sensitivity : 0.9067
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.9146
```

```
##               Prevalence : 0.5000
##           Detection Rate : 0.4533
##     Detection Prevalence : 0.4533
##        Balanced Accuracy : 0.9533
##
##         'Positive' Class : 1
##
```

Challenging the solution

```
#grid search for value of k

#parameter ranges
k_range = seq(3, 21, length.out=7)

#prameter grid <
paramgr <- expand.grid(k=k_range)

#train control
train_control = trainControl(method="cv", number=5, search="grid")

#setting seed for reproducibility
set.seed(0)

X_train_sc2 <- copy(X_train)

for (col in colnames(X_train_sc2)){
  X_train_sc2[col] <- scale(X_train_sc2[col])
}
#
X_train_sc2$clicked <- factor(y_train$clicked.on.ad)

search <- train(clicked ~ ., data = X_train_sc2,
                method= "knn", trControl= train_control,
                tuneGrid= paramgr, metric="Accuracy")
search

## k-Nearest Neighbors
##
## 700 samples
##  10 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 560, 560, 560, 560, 560
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    3  0.9428571  0.8857143
```

```
##    6  0.9500000  0.9000000
##    9  0.9500000  0.9000000
##   12  0.9528571  0.9057143
##   15  0.9528571  0.9057143
##   18  0.9500000  0.9000000
##   21  0.9542857  0.9085714
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 21.
```

```
mod <- knn(train= X_train_sc[,],test=X_test_sc[,], cl= y_train[,],k=21)


table(mod, y_test[,])
```

```
##
## mod   0   1
##   0 150  16
##   1   0 134
```

```
print(accuracy(y_test[,], mod))
```

```
## [1] 0.9466667
```

```
caret::confusionMatrix(data=as.factor(mod), reference=as.factor(y_test[,]),
positive="1")
```

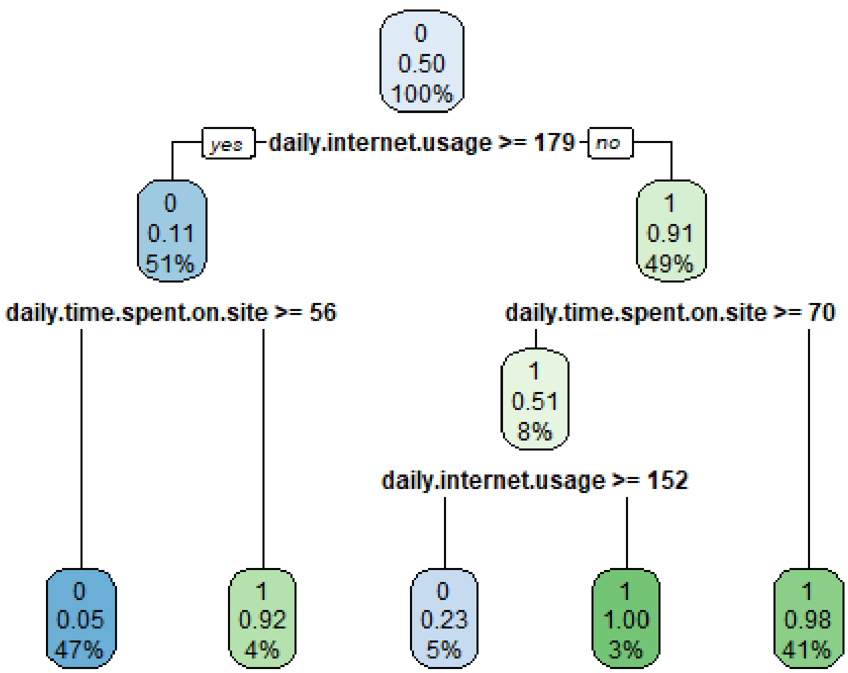
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 150  16
##          1   0 134
##
##                Accuracy : 0.9467
##                  95% CI : (0.9148, 0.9692)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8933
##
##  Mcnemar's Test P-Value : 0.0001768
##
##             Sensitivity : 0.8933
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.9036
##              Prevalence : 0.5000
##          Detection Rate : 0.4467
##    Detection Prevalence : 0.4467
##       Balanced Accuracy : 0.9467
```

```
##
##        'Positive' Class : 1
##
```

Accuracy on test set slightly lower (0.947 compared to 0.953) after using higher k value.

2. Rpart

```
library(rpart.plot)

## Loading required package: rpart

library(rpart)

#fitting the model
m <- rpart(clicked.on.ad ~ ., data = st_train,
           method = "class")


rpart.plot(m)
```




```
#plotting decision tree

#predicting and confusion matrix
p <- predict(m, X_test, type = "class")
table(p, y_test[,])

##
## p     0   1
```

```
##   0 146  12
##   1   4 138
```

```r
#more detailed evaluation
caret::confusionMatrix(data=as.factor(p), reference=as.factor(y_test[,]),
positive="1")
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction   0   1
##          0 146  12
##          1   4 138
## 
##                Accuracy : 0.9467
##                  95% CI : (0.9148, 0.9692)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : < 2e-16
## 
##                   Kappa : 0.8933
## 
##  Mcnemar's Test P-Value : 0.08012
## 
##             Sensitivity : 0.9200
##             Specificity : 0.9733
##          Pos Pred Value : 0.9718
##          Neg Pred Value : 0.9241
##              Prevalence : 0.5000
##          Detection Rate : 0.4600
##    Detection Prevalence : 0.4733
##       Balanced Accuracy : 0.9467
## 
##        'Positive' Class : 1
## 
```

```r
#accuracy of 94.7%
```

Challenging the solution

```r
st_train$clicked.on.ad <- factor(st_train$clicked.on.ad)
set.seed(42)
model <- train(clicked.on.ad ~ .,
               data = st_train,
               method = "rpart",
               tuneLength = 5,
               trControl = trainControl(method = "cv",
                                        number = 5,
                                        verboseIter = FALSE))
model
```

```
## CART
## 
```

```
## 700 samples
##  10 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 560, 560, 560, 560, 560
## Resampling results across tuning parameters:
##
##   cp           Accuracy   Kappa
##   0.000000000  0.9371429  0.8742857
##   0.005714286  0.9400000  0.8800000
##   0.027142857  0.9242857  0.8485714
##   0.062857143  0.9014286  0.8028571
##   0.797142857  0.6514286  0.3028571
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.005714286.
```

```
#predicting and evaluating
p <- predict(model, X_test)
table(p, y_test[,])
```

```
##
## p     0   1
##   0 146  12
##   1   4 138
```

```
print(accuracy(y_test[,], p))
```

```
## [1] 0.9466667
```

```
caret::confusionMatrix(data=as.factor(p), reference=as.factor(y_test[,]),
positive="1")
```

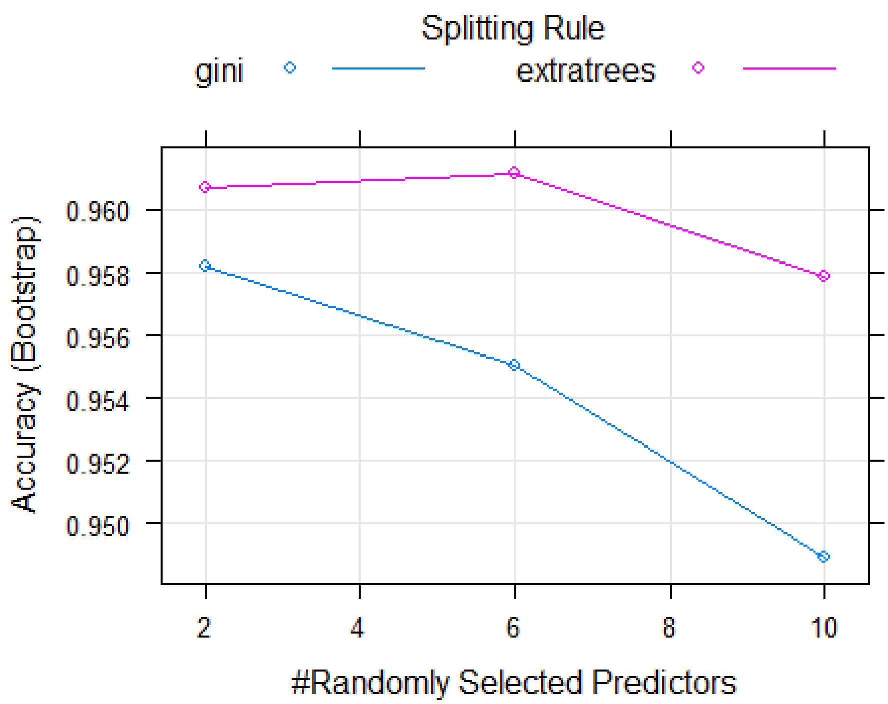
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 146  12
##          1   4 138
##
##                Accuracy : 0.9467
##                  95% CI : (0.9148, 0.9692)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.8933
##
##  Mcnemar's Test P-Value : 0.08012
##
```

```
##             Sensitivity : 0.9200
##             Specificity : 0.9733
##          Pos Pred Value : 0.9718
##          Neg Pred Value : 0.9241
##              Prevalence : 0.5000
##          Detection Rate : 0.4600
##    Detection Prevalence : 0.4733
##       Balanced Accuracy : 0.9467
##
##        'Positive' Class : 1
##
```

Accuracy of 0.947 same as 0.947 without tuning

3. Random forests (ranger)

```
set.seed(12)
model <- train(clicked.on.ad ~ .,
               data = st_train,
               method = "ranger")
model

## Random Forest
##
## 700 samples
##  10 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 700, 700, 700, 700, 700, 700, ...
## Resampling results across tuning parameters:
##
##   mtry  splitrule   Accuracy   Kappa
##    2    gini        0.9582173  0.9161632
##    2    extratrees  0.9606866  0.9211554
##    6    gini        0.9550464  0.9098127
##    6    extratrees  0.9611383  0.9220517
##   10    gini        0.9489542  0.8976326
##   10    extratrees  0.9578843  0.9155416
##
## Tuning parameter 'min.node.size' 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 mtry = 6, splitrule = extratrees
##  and min.node.size = 1.

plot(model)
```




```
#predicting and evaluating
p <- predict(model, X_test)
table(p, y_test[,])

##
## p     0   1
##    0 148   7
##    1   2 143

print(accuracy(y_test[,], p))

## [1] 0.97

caret::confusionMatrix(data=as.factor(p), reference=as.factor(y_test[,]),
positive="1")

## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 148   7
##          1   2 143
##
##                Accuracy : 0.97
##                  95% CI : (0.9438, 0.9862)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : <2e-16
##
```

```
##                  Kappa : 0.94
##
##  Mcnemar's Test P-Value : 0.1824
##
##             Sensitivity : 0.9533
##             Specificity : 0.9867
##          Pos Pred Value : 0.9862
##          Neg Pred Value : 0.9548
##              Prevalence : 0.5000
##          Detection Rate : 0.4767
##    Detection Prevalence : 0.4833
##       Balanced Accuracy : 0.9700
##
##        'Positive' Class : 1
##
```

Accuracy of 0.97 which is the highest so far

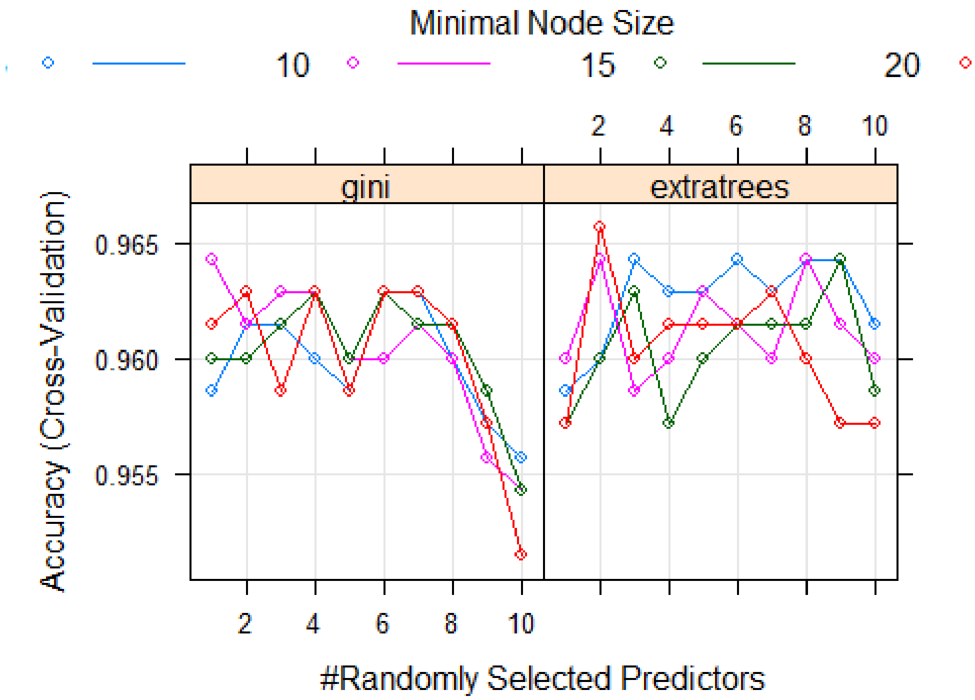
Challenging the solution

```
set.seed(42)
Grid <- expand.grid(.mtry = c(1,2,3,4,5,6,7,8,9,10),
                     .splitrule = c("gini", "extratrees"),
                     .min.node.size = c(5, 10, 15,20)

                     )

model <- train(clicked.on.ad ~ .,
               data = st_train,
               method = "ranger",
               tuneGrid = Grid,
               trControl = trainControl(method = "cv",
                                        number = 5,
                                        verboseIter = FALSE))

model

## Random Forest
##
## 700 samples
##  10 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 560, 560, 560, 560, 560
## Resampling results across tuning parameters:
##
##   mtry  splitrule   min.node.size  Accuracy   Kappa
##    1    gini         5             0.9585714  0.9171429
##    1    gini        10             0.9642857  0.9285714
```

```
##    1    gini        15              0.9600000  0.9200000
##    1    gini        20              0.9614286  0.9228571
##    1    extratrees   5              0.9585714  0.9171429
##    1    extratrees  10              0.9600000  0.9200000
##    1    extratrees  15              0.9571429  0.9142857
##    1    extratrees  20              0.9571429  0.9142857
##    2    gini         5              0.9614286  0.9228571
##    2    gini        10              0.9614286  0.9228571
##    2    gini        15              0.9600000  0.9200000
##    2    gini        20              0.9628571  0.9257143
##    2    extratrees   5              0.9600000  0.9200000
##    2    extratrees  10              0.9642857  0.9285714
##    2    extratrees  15              0.9600000  0.9200000
##    2    extratrees  20              0.9657143  0.9314286
##    3    gini         5              0.9614286  0.9228571
##    3    gini        10              0.9628571  0.9257143
##    3    gini        15              0.9614286  0.9228571
##    3    gini        20              0.9585714  0.9171429
##    3    extratrees   5              0.9642857  0.9285714
##    3    extratrees  10              0.9585714  0.9171429
##    3    extratrees  15              0.9628571  0.9257143
##    3    extratrees  20              0.9600000  0.9200000
##    4    gini         5              0.9600000  0.9200000
##    4    gini        10              0.9628571  0.9257143
##    4    gini        15              0.9628571  0.9257143
##    4    gini        20              0.9628571  0.9257143
##    4    extratrees   5              0.9628571  0.9257143
##    4    extratrees  10              0.9600000  0.9200000
##    4    extratrees  15              0.9571429  0.9142857
##    4    extratrees  20              0.9614286  0.9228571
##    5    gini         5              0.9585714  0.9171429
##    5    gini        10              0.9600000  0.9200000
##    5    gini        15              0.9600000  0.9200000
##    5    gini        20              0.9585714  0.9171429
##    5    extratrees   5              0.9628571  0.9257143
##    5    extratrees  10              0.9628571  0.9257143
##    5    extratrees  15              0.9600000  0.9200000
##    5    extratrees  20              0.9614286  0.9228571
##    6    gini         5              0.9628571  0.9257143
##    6    gini        10              0.9600000  0.9200000
##    6    gini        15              0.9628571  0.9257143
##    6    gini        20              0.9628571  0.9257143
##    6    extratrees   5              0.9642857  0.9285714
##    6    extratrees  10              0.9614286  0.9228571
##    6    extratrees  15              0.9614286  0.9228571
##    6    extratrees  20              0.9614286  0.9228571
##    7    gini         5              0.9628571  0.9257143
##    7    gini        10              0.9614286  0.9228571
##    7    gini        15              0.9614286  0.9228571
##    7    gini        20              0.9628571  0.9257143
```

```
##    7    extratrees   5            0.9628571  0.9257143
##    7    extratrees  10            0.9600000  0.9200000
##    7    extratrees  15            0.9614286  0.9228571
##    7    extratrees  20            0.9628571  0.9257143
##    8    gini         5            0.9600000  0.9200000
##    8    gini        10            0.9600000  0.9200000
##    8    gini        15            0.9614286  0.9228571
##    8    gini        20            0.9614286  0.9228571
##    8    extratrees   5            0.9642857  0.9285714
##    8    extratrees  10            0.9642857  0.9285714
##    8    extratrees  15            0.9614286  0.9228571
##    8    extratrees  20            0.9600000  0.9200000
##    9    gini         5            0.9571429  0.9142857
##    9    gini        10            0.9557143  0.9114286
##    9    gini        15            0.9585714  0.9171429
##    9    gini        20            0.9571429  0.9142857
##    9    extratrees   5            0.9642857  0.9285714
##    9    extratrees  10            0.9614286  0.9228571
##    9    extratrees  15            0.9642857  0.9285714
##    9    extratrees  20            0.9571429  0.9142857
##   10    gini         5            0.9557143  0.9114286
##   10    gini        10            0.9542857  0.9085714
##   10    gini        15            0.9542857  0.9085714
##   10    gini        20            0.9514286  0.9028571
##   10    extratrees   5            0.9614286  0.9228571
##   10    extratrees  10            0.9600000  0.9200000
##   10    extratrees  15            0.9585714  0.9171429
##   10    extratrees  20            0.9571429  0.9142857
## 
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were mtry = 2, splitrule = extratrees
##  and min.node.size = 20.
```

```
plot(model)
```




```
#predicting and evaluating
p <- predict(model, X_test)
table(p, y_test[,])

##
## p     0   1
##   0 148   8
##   1   2 142

print(accuracy(y_test[,], p))

## [1] 0.9666667

caret::confusionMatrix(data=as.factor(p), reference=as.factor(y_test[,]),
positive="1")

## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 148   8
##          1   2 142
##
##                Accuracy : 0.9667
##                  95% CI : (0.9396, 0.9839)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : <2e-16
##
```

```
##                  Kappa : 0.9333
##
##  Mcnemar's Test P-Value : 0.1138
##
##            Sensitivity : 0.9467
##            Specificity : 0.9867
##         Pos Pred Value : 0.9861
##         Neg Pred Value : 0.9487
##             Prevalence : 0.5000
##         Detection Rate : 0.4733
##   Detection Prevalence : 0.4800
##      Balanced Accuracy : 0.9667
##
##       'Positive' Class : 1
##
```

Accuracy of 0.967 lower than ranger before further tuning (0.970)

Conclusion and Recommendations

Conclusion

The objectives of the study were achieved. Following data preparation (where missing values, duplicates, outliers, column creation etc were dealt with accordingly), univariate , bivariate analysis and modelling were carried out providing valuable insights.

Some univariate analysis highlights:

- Most people in the dataset were between 30-35
- Most of the people spent 75-80 minutes on the site daily
- Most of the area income values lied between 60000 to 65000
- Most observations had a timestamp of 7 am
- The dataset was balanced. There was an equal number of observations who clicked on the ad and those who did not. etc

Some bivariate analysis highlights:

- Among the females, the majority clicked on the ad while among males the majority did not click on the ad.
- February followed by May had the highest frequencies of ad clicks. Additionally, the proportions of those who clicked on the ad were higher than those that did not in those months.
- The average daily internet usage of those who clicked on the ad was lower than that of those who did not click on the ad.

- The average age of those who clicked on the ad was higher than the average age of those who did not.
- The median area income of those who did not click on the ad was higher than that of those who clicked on the ad. etc

Modelling:

The dataset was balanced so accuracy metric was suitable as the main gauge of performance

KNN

- First model using 5 nearest neighbours - 0.953 accuracy
- After tuning, accuracy 0.947 - lower

Decision tree(rpart)

- First model - accuracy 0.947
- after tuning, accuracy remained the same

Random forests(ranger)

- First model - accuracy 0.97. Best model
- After tuning, accuracy of 0.967, lower than above

Recommendations

The predictive model to be used should be the random forest built with ranger function without further tuning of hyperparameters. It had the highest accuracy (0.97)

As the average age of those who clicked on the ad was higher, it is more likely that older professionals click on the ad. Adverts highlighting flexible times and/or part-time options will likely improve the number clicking on the ads.

Females are more likely to click on the ads compared to males. Including tag lines further encouraging women to apply may increase ad traffic.

Individuals from Australia, Ethiopia and Turkey are more likely to click on the ads compared to other countries. We recommend running more ads in these areas as they show higher interest compared to other countries.

February followed by May had the highest frequencies of ad clicks. We therefore recommend running the ads in these months.

Since Sundays, followed by Thursdays and Wednesdays had the highest frequencies of clicks, we recommend running ads on these days of the week, most importantly on Sundays.

We recommend running the ads at around 9 am since that is the time with the highest frequency of ad clicks.