Unsupervised Learning with R

STEPHEN ODHIAMBO OGAJA

2022-06-04

TARGETED CRYPTOGRAPHY ADVERTISING

1. Defining the Question

a) Specifying the Question

Which individuals are more likely to click on the cryptography course adverts?

b) Defining the metric of success

The project will be considered a success when we can identify which individuals will click on the advert and factors that affect ad clicks.

c) Understanding the context

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

d) Recording the experimental design

```
1. Data sourcing/loading
2. Data Understanding
3. Data Relevance
4. External Dataset Validation
5. Data Preparation
6. Univariate Analysis
7. Bivariate Analysis
8. Multivariate Analysis
9. Implementing the solution
10. Challenging the solution
11. Conclusion
12. Follow up questions
```

e) Data Relevance

For relevant data, the data should be able to provide meaningful insights that can be used to isolate users who are most likely to click in the ads.

2. Data Understanding

Loading Libraries

a) Reading the data

```
# let's import our dataset
adverts <- read.csv("advertising.csv")
```

Let's read our dataset

b) Checking the Data

```
# let's preview the top of our dataset
head(adverts)
```

Top records

```
##   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
```

```r
# let's preview the last 6 records of our dataset
tail(adverts)
```

Bottom records

```
##      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
```

```r
# let's see the number of rows and columns in our dataset
cat("The dataset has ", nrow(adverts), "rows and ", ncol(adverts), "columns")
```

The shape of the dataset

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

c) data types of the variables

```r
str(adverts)
```

```
## 'data.frame':    1000 obs. of  10 variables:
##  $ Daily.Time.Spent.on.Site: num  69 80.2 69.5 74.2 68.4 ...
##  $ Age                     : int  35 31 26 29 35 23 33 48 30 20 ...
##  $ Area.Income             : num  61834 68442 59786 54806 73890 ...
##  $ Daily.Internet.Usage    : num  256 194 236 246 226 ...
##  $ Ad.Topic.Line           : chr  "Cloned 5thgeneration orchestration" "Monitored national standardi
##  $ City                    : chr  "Wrightburgh" "West Jodi" "Davidton" "West Terrifurt" ...
##  $ Male                    : int  0 1 0 1 0 1 0 1 1 1 ...
```

```
##  $ Country                : chr  "Tunisia" "Nauru" "San Marino" "Italy" ...
##  $ Timestamp              : chr  "2016-03-27 00:53:11" "2016-04-04 01:39:02" "2016-03-13 20:35:42"
##  $ Clicked.on.Ad          : int  0 0 0 0 0 0 0 1 0 0 ...
```

R stores the dataframe and views the variables as lists so to see the the various data types of this list we use the str function.

```
duplicates <- adverts[duplicated(adverts), ]
duplicates
```

let's check for duplicates in the dataframe

```
##  [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)
```

The dataframe does not contain duplicate values.

```
colSums(is.na(adverts))
```

let's check for missing data in each column

```
## 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
```

The dataset's columns does not have missing data.

let's check for outliers in the dataset

```
num_cols <- adverts[,unlist(lapply(adverts, is.numeric))]
head(num_cols)
```

selecting only numeric columns

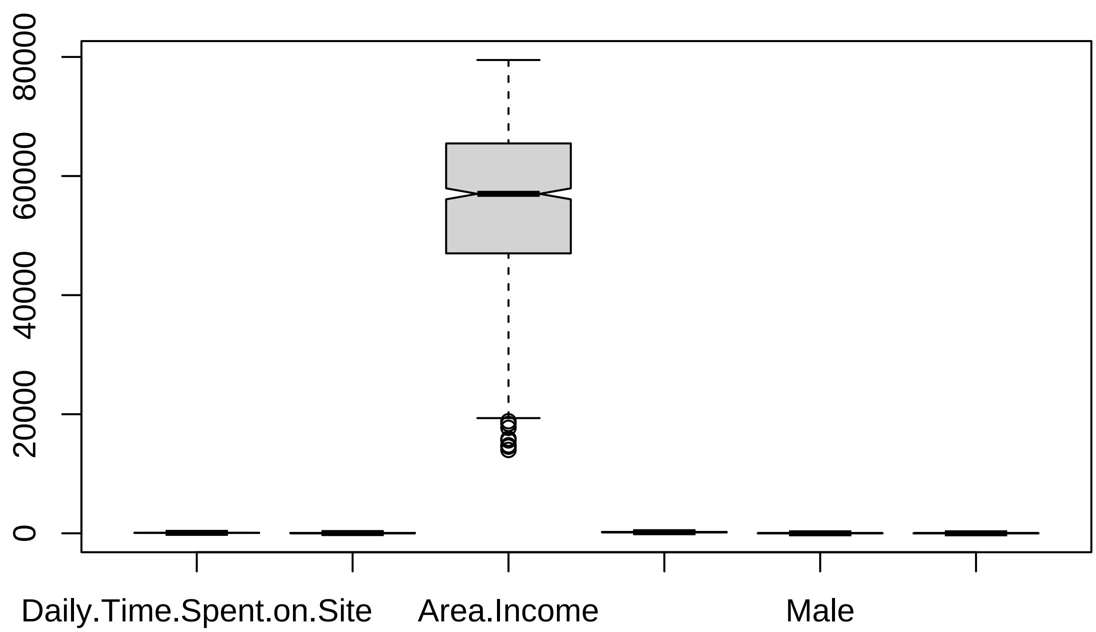
```
##   Daily.Time.Spent.on.Site Age Area.Income Daily.Internet.Usage Male
## 1                    68.95  35    61833.90               256.09    0
## 2                    80.23  31    68441.85               193.77    1
## 3                    69.47  26    59785.94               236.50    0
## 4                    74.15  29    54806.18               245.89    1
## 5                    68.37  35    73889.99               225.58    0
## 6                    59.99  23    59761.56               226.74    1
##   Clicked.on.Ad
## 1             0
## 2             0
## 3             0
## 4             0
## 5             0
## 6             0
```

6 columns are numerical in nature

```
boxplot(num_cols, notch = TRUE)
```

let's check for outliers in the numerical columns using BOXPLOT

```
## Warning in (function (z, notch = FALSE, width = NULL, varwidth = FALSE, : some
## notches went outside hinges ('box'): maybe set notch=FALSE
```



The Area.Income variable has outliers which will be imputed.

```r
boxplot.stats(adverts$Area.Income)$out
```

let's see the values which are outliers in the Area.Income variable

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

let's check for outliers using Z-SCORES

```r
z_scores <- as.data.frame(sapply(num_cols, function(num_cols) (abs(num_cols-mean(num_cols))/sd(num_cols
head(z_scores)
```

The z-score indicates the number of standard deviations a given value deviates from the mean.

```
##   Daily.Time.Spent.on.Site       Age Area.Income Daily.Internet.Usage      Male
## 1                0.2491419 0.1148475  0.50943618            1.7331628 0.9622138
## 2                0.9606516 0.5701399  1.00202882            0.3136484 1.0382307
## 3                0.2819420 1.1392555  0.35677007            1.2869451 0.9622138
## 4                0.5771428 0.7977862  0.01444841            1.5008289 1.0382307
## 5                0.2125572 0.1148475  1.40816290            1.0382112 0.9622138
## 6                0.3160289 1.4807248  0.35495265            1.0646335 1.0382307
##   Clicked.on.Ad
## 1     0.9994999
## 2     0.9994999
## 3     0.9994999
## 4     0.9994999
## 5     0.9994999
## 6     0.9994999
```

We will drop values with a Z-Score of more than 3 or -3. They are the outliers

```r
no_outliers <- z_scores[!rowSums(z_scores>3), ]
head(no_outliers)
```

Removing the outliers

```
##   Daily.Time.Spent.on.Site       Age Area.Income Daily.Internet.Usage      Male
## 1                0.2491419 0.1148475  0.50943618            1.7331628 0.9622138
## 2                0.9606516 0.5701399  1.00202882            0.3136484 1.0382307
## 3                0.2819420 1.1392555  0.35677007            1.2869451 0.9622138
## 4                0.5771428 0.7977862  0.01444841            1.5008289 1.0382307
## 5                0.2125572 0.1148475  1.40816290            1.0382112 0.9622138
## 6                0.3160289 1.4807248  0.35495265            1.0646335 1.0382307
##   Clicked.on.Ad
## 1     0.9994999
```

```
## 2     0.9994999
## 3     0.9994999
## 4     0.9994999
## 5     0.9994999
## 6     0.9994999
```

```
dim(num_cols)
```

let's check the number of observations after removing outliers

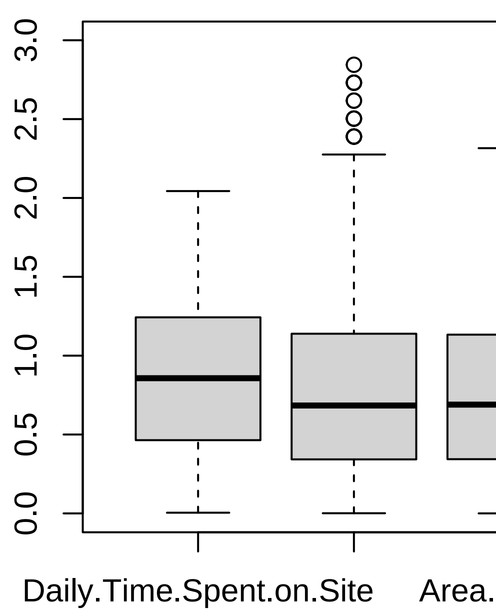
```
## [1] 1000    6
```

```
dim(no_outliers)
```

```
## [1] 998   6
```

We removed 2 observations.

```
boxplot(no_outliers)
```



let's check for outliers in the new dataframe after removing them
There are still outliers so we will use interquantile range method to remove outliers

checking and removing outliers using IQR

```
income.IQR <- 65471-47032
income.IQR <-IQR(adverts$`Area.Income`)
income.IQR
```

The Area.Income column had outliers so we focus on it

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

```
adverts_2 <- subset(adverts, adverts$`Area.Income`> (47032 - 1.5*income.IQR) & adverts$`Area.Income`<(65
```

let's save the dataframe without outliers into a new dataframe by assigning it to a variable

```
dim(adverts_2)
```

let's see the shape of the new dataframe

```
## [1] 991  10
```

We have lost 9 observations that included the outliers. We proceed with analysis.

3. Exploratory Data Analysis

{UNIVARIATE ANALYSIS}

```
summary(num_cols)
```

let's get the mean of the numerical columns

```
##  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
##       Male        Clicked.on.Ad
##  Min.   :0.000   Min.   :0.0
##  1st Qu.:0.000   1st Qu.:0.0
##  Median :0.000   Median :0.5
##  Mean   :0.481   Mean   :0.5
##  3rd Qu.:1.000   3rd Qu.:1.0
##  Max.   :1.000   Max.   :1.0
```

The summary shows: -The minimum value for each numerical variable. -The first quantile for each numerical variable -The median value for all numeric variables across the dataframe. -The mean value for all numeric variables. -The third quantile. -The maximum value for all numerical columns.

```
variance <- var(num_cols)
variance
```

let's get the variance for the numeric variables

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

variance is a measure of how far the set of data points per column is spread out from their mean eg. those of the area income seem to be far spread out from their mean when compared to that of the age column.

let's get the standard deviation of the numeric variables

```
sd.function <- function(column) {
  standard.deviations <- sd(column)
  print(standard.deviations)
}
```

let's create a function to get the standard deviations

```
sd.function(adverts_2$Daily.Time.Spent.on.Site)
```

standard deviation for daily time spent on site

```
## [1] 15.9005
```

```
sd.function(adverts_2$Age)
```

standard deviation for Age

```
## [1] 8.804716
```

```
sd.function(adverts_2$Area.Income)
```

standard deviation for Area.Income

```
## [1] 12961.5
```

```
sd.function(adverts_2$Daily.Internet.Usage)
```

standard deviation for Daily.Internet.Usage

```
## [1] 44.05386
```

Where a low standard deviation indicates that values are closer to the mean a high one indicates the standard deviation is far from the mean e.g the age column standard deviation of 8.8 displays that its values are closer to their mean than that of the Area income column whose value is 12961

```
library(moments)
skewness(num_cols)
```

let's get the skewness of the numerical column

```
## Daily.Time.Spent.on.Site                      Age              Area.Income
##              -0.37120261               0.47842268              -0.64939670
##     Daily.Internet.Usage                     Male             Clicked.on.Ad
##              -0.03348703               0.07605493               0.00000000
```

The skewness of the Age variable being positive indicates that its distribution has a longer right tail than left tail while the rest of the columns' left tails.

{BIVARIATE ANALYSIS}

```
cov(num_cols)
```

let's get the covariance of the numeric variables

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

The age variable is the only column with a positive covariance with the ad click variable, the rest have negative covariances.

```
cor(num_cols)
```

let's get the correlation coefficient

```
##                          Daily.Time.Spent.on.Site         Age  Area.Income
## Daily.Time.Spent.on.Site               1.00000000 -0.33151334  0.310954413
## Age                                   -0.33151334  1.00000000 -0.182604955
## Area.Income                            0.31095441 -0.18260496  1.000000000
## Daily.Internet.Usage                   0.51865848 -0.36720856  0.337495533
## Male                                  -0.01895085 -0.02104406  0.001322359
## Clicked.on.Ad                         -0.74811656  0.49253127 -0.476254628
##                          Daily.Internet.Usage         Male Clicked.on.Ad
## Daily.Time.Spent.on.Site           0.51865848 -0.018950855   -0.74811656
## Age                               -0.36720856 -0.021044064    0.49253127
## Area.Income                        0.33749553  0.001322359   -0.47625463
## Daily.Internet.Usage               1.00000000  0.028012326   -0.78653918
## Male                               0.02801233  1.000000000   -0.03802747
## Clicked.on.Ad                     -0.78653918 -0.038027466    1.00000000
```

The variables have a negative correlation with the target variable apart from the age variable which has a positive correlation. Let's see that in the correlogram below

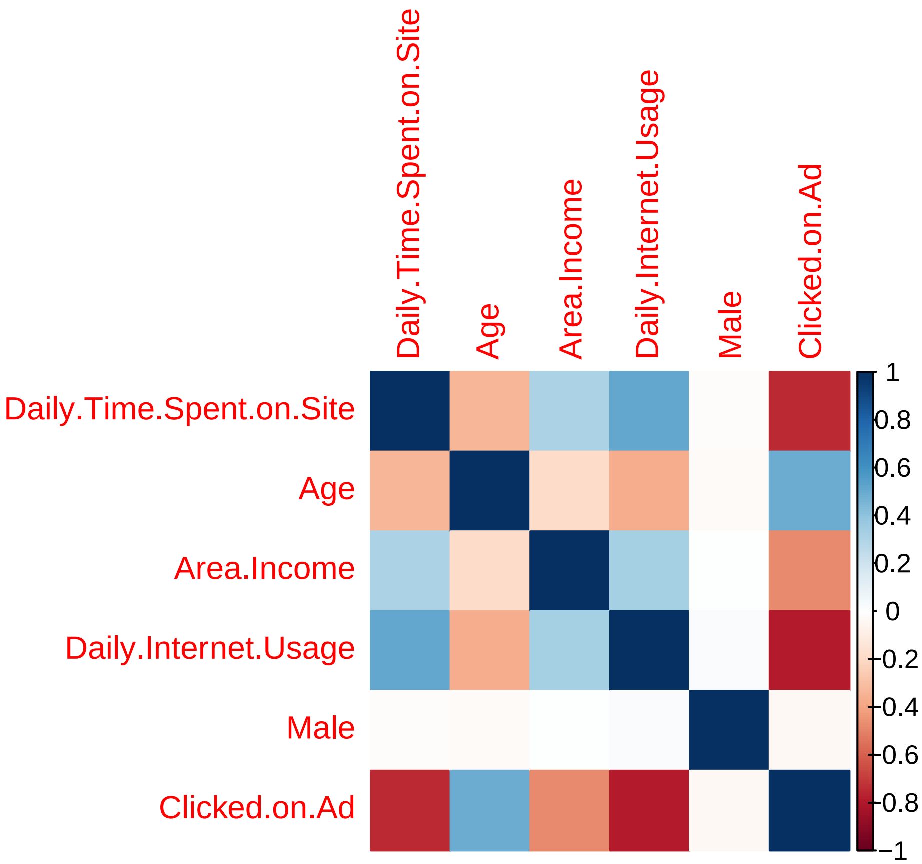
```
library(corrplot)
```

let's see the corrplot of the numeric variables

```
## corrplot 0.92 loaded
```

```r
corr_ <- cor(num_cols)

corrplot(corr_, method = 'color')
```



4. Modelling

We will do logistic regression as the study requires a classification solution

```r
# let's split the data into training and test splits
set.seed(100)
# let's select only columns that are relevant to modeling
cols = c('Daily.Time.Spent.on.Site', 'Age', 'Area.Income', 'Daily.Internet.Usage', 'Male', 'Clicked.on.A
advertising = select(adverts_2, all_of(cols))

train_rows = createDataPartition(advertising$Clicked.on.Ad, p=0.8, list=FALSE)

# training  dataset
train = advertising[train_rows,]

# test dataset
test = advertising[-train_rows,]

# lets create the  X and y variables
```

```r
X = train
y = train$Clicked.on.Ad
```

a) Decsion Trees

```r
# let's install tree package
#install.packages("tree")
```

```r
# let's train the model
require(tree)
```

```
## Loading required package: tree
```

```r
model_ <- tree(Clicked.on.Ad ~.,
               data = train,
               method = "ranger")
model_
```

```
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 793 198.1000 0.48550
##    2) Daily.Internet.Usage < 177.265 366  26.7000 0.92080
##      4) Daily.Time.Spent.on.Site < 71.365 318   4.9210 0.98430 *
##      5) Daily.Time.Spent.on.Site > 71.365 48  12.0000 0.50000
##       10) Daily.Internet.Usage < 152.75 19   0.9474 0.94740 *
##       11) Daily.Internet.Usage > 152.75 29   4.7590 0.20690 *
##    3) Daily.Internet.Usage > 177.265 427  42.6000 0.11240
##      6) Daily.Time.Spent.on.Site < 57.08 37   3.5680 0.89190 *
##      7) Daily.Time.Spent.on.Site > 57.08 390  14.4200 0.03846
##       14) Age < 49 383  10.6800 0.02872 *
##       15) Age > 49 7   1.7140 0.57140 *
```

```r
# let's make predictions
y_pred_ <- predict(model_)
head(y_pred_)
```

```
##          1          2          4          6          7          9
## 0.02872063 0.02872063 0.02872063 0.02872063 0.02872063 0.02872063
```

```r
# let's find the confusion matrix and the accuracy scores of the model
confusion_Matrix <- table(Actual = train$Clicked.on.Ad, predicted = y_pred_ > .5)
confusion_Matrix
```

```
##       predicted
## Actual FALSE TRUE
##      0   395   13
##      1    17  368
```

```r
# let's find the accuracy
(confusion_Matrix[[1,1]] + confusion_Matrix[[2,2]])/sum(confusion_Matrix)
```

```
## [1] 0.962169
```

The Decision Trees model gives an accuracy of 96%

b) Logistic Regression

```r
# let's now train our model
model <- glm(Clicked.on.Ad ~ .,
             data = train,
             family = "binomial")

# let's see a summary of our model
summary(model)
```

We will use the x and y set above

```
##
## Call:
## glm(formula = Clicked.on.Ad ~ ., family = "binomial", data = train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.49017  -0.14869  -0.07681   0.01797   3.11157
##
## Coefficients:
##                             Estimate Std. Error z value Pr(>|z|)
## (Intercept)               26.8407667  2.9929460   8.968  < 2e-16 ***
## Daily.Time.Spent.on.Site -0.1948214  0.0227940  -8.547  < 2e-16 ***
## Age                        0.1671951  0.0280061   5.970 2.37e-09 ***
## Area.Income               -0.0001360  0.0000212  -6.416 1.40e-10 ***
## Daily.Internet.Usage      -0.0594364  0.0072811  -8.163 3.27e-16 ***
## Male                      -0.3666537  0.4441073  -0.826    0.409
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1098.66  on 792  degrees of freedom
## Residual deviance:  151.81  on 787  degrees of freedom
## AIC: 163.81
##
## Number of Fisher Scoring iterations: 8
```

```r
# let's make predictions
y_pred <- predict(model)
```

```r
# let's run our test through the model
b <- predict(model, test, type = "response")
head(b)
```

```
##           3           5           8          22          25          35
## 0.010612058 0.016538321 0.999971477 0.003073253 0.998190722 0.999975620
```

```r
b <- predict(model, train, type = "response")
head(b)
```

```
##           1           2           4           6           7           9
## 0.012469133 0.008165030 0.005526285 0.048660104 0.009262190 0.003825157
```

```r
# let's validate the model
matrix <- table(Actual_Value = train$Clicked.on.Ad, Predicted_Value = b > .5)
matrix
```

```
##             Predicted_Value
## Actual_Value FALSE TRUE
##            0   400    8
##            1    15  370
```

```r
# let's get our accuracy score
(matrix[[1,1]] + matrix[[2,2]])/sum(matrix)
```

```
## [1] 0.9709962
```

The model attains an accuracy of 97% on logistic regression

5. Conclusion

The logistic regression model having given an accuracy score of 97% is better than the SVM model performed first.

6. RECOMMENDATIONS

a. The entrepreneur should focus on the older population as the correlation between age and advert clicks is slightly positive indicating that as age increases the more likely the clicks are made.

b. The entreoreneur should focus on regions with bigger area coverage as those with a smaller area since the correlation between area income and advert clicks is negatively weak one indicating that as area income decreases the more likely the clicks are made and vice versa.

c. She should focus on the regions with low daily internet usage because the correlation between the daily internet usage and clicks on ads is negative indicating that as internet use decreases the more likely the clicks will be made.

7. Follow up Questions

a) Did we have the right data?

Yes, the data we were provided with was correct and it fit the scope of the analysis

b) Did we need any other data to answer questions?

Yes, availability of more data on customer behavior would have given us more insight and perhaps a more accurate model

c) Did we have the right Question?

Yes we did, the main question married well with the data we were provided with.