Predicting Ad Clicks on A Website

Booorayan

26/10/2019

# importing required libraries  
library(dplyr)

##   
## Attaching package: 'dplyr'

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

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

library(aod)  
library(ggplot2)  
library(xgboost) # for xgboost

##   
## Attaching package: 'xgboost'

## The following object is masked from 'package:dplyr':  
##   
## slice

library(tidyverse) # general utility functions

## -- Attaching packages ------------------------------------------------------------- tidyverse 1.2.1 --

## v tibble 2.1.1 v purrr 0.3.2  
## v tidyr 0.8.3 v stringr 1.4.0  
## v readr 1.3.1 v forcats 0.4.0

## -- Conflicts ---------------------------------------------------------------- tidyverse\_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()  
## x xgboost::slice() masks dplyr::slice()

library(randomForest)

## randomForest 4.6-14

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

##   
## Attaching package: 'randomForest'

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

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

library(Metrics)  
library(caret)

## Loading required package: lattice

##   
## Attaching package: 'caret'

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

## The following object is masked from 'package:purrr':  
##   
## lift

# loading the csv file and storing it a variable clickad  
clickad <- read.csv("advertising.csv")

# previewing the first five obseervations in the dataframe   
head(clickad)

## 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 last five observations in the dataframe  
tail(clickad)

## 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 for dimensions of the dataframe  
cols <- dim(clickad)  
cols

## [1] 1000 10

# dataframe has 1000 rows and 10 columns

# checking the names of columns in the dataset  
colnames(clickad)

## [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"

# displaying the internal structure of the datframe  
strr <- str(clickad)

## '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 : Factor w/ 1000 levels "Adaptive 24hour Graphic Interface",..: 92 465 567 904 767 806 223 724 108 455 ...  
## $ City : Factor w/ 969 levels "Adamsbury","Adamside",..: 962 904 112 940 806 283 47 672 885 713 ...  
## $ Male : int 0 1 0 1 0 1 0 1 1 1 ...  
## $ Country : Factor w/ 237 levels "Afghanistan",..: 216 148 185 104 97 159 146 13 83 79 ...  
## $ Timestamp : Factor w/ 1000 levels "2016-01-01 02:52:10",..: 440 475 368 57 768 690 131 334 549 942 ...  
## $ Clicked.on.Ad : int 0 0 0 0 0 0 0 1 0 0 ...

strr

## NULL

# output reveals we have 6 numerical variables and 4 columns of datatype factors

# Checking for the sum of missing values in each column  
miss <- colSums(is.na(clickad))  
miss

## 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

# Output reveals no column has missing values

# Checking for duplicate values   
dup\_val <- clickad[duplicated(clickad),]  
dup\_val

## [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   
## <0 rows> (or 0-length row.names)

# Dataframe has no duplicated values

library(tidyverse)  
  
# converting timestamp column to type date  
clickad$Timestamp <- as.Date(clickad$Timestamp)  
  
# creating a column time that is split from timestamp column  
clickad$Time <- format(as.POSIXct(clickad$Timestamp, format="%Y-%m-%d %H:%M:%S"), "%H:%M:%S")  
  
# creating date column that is split/stripped from timestamp column  
clickad$Date <- format(as.POSIXct(clickad$Timestamp, format="%Y-%m-%d %H:%M:%S"), "%Y:%m:%d")  
  
head(clickad)

## 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 Time Date  
## 1 2016-03-27 0 03:00:00 2016:03:27  
## 2 2016-04-04 0 03:00:00 2016:04:04  
## 3 2016-03-13 0 03:00:00 2016:03:13  
## 4 2016-01-10 0 03:00:00 2016:01:10  
## 5 2016-06-03 0 03:00:00 2016:06:03  
## 6 2016-05-19 0 03:00:00 2016:05:19

library(lubridate)

##   
## Attaching package: 'lubridate'

## The following object is masked from 'package:base':  
##   
## date

# splitting the date column into year, month and day columns using mutate function  
clickad = clickad %>%  
 mutate(Date = ymd(Date)) %>%  
 mutate\_at(vars(Date), funs(year, month, day))

## Warning: funs() is soft deprecated as of dplyr 0.8.0  
## please use list() instead  
##   
## # Before:  
## funs(name = f(.)  
##   
## # After:   
## list(name = ~f(.))  
## This warning is displayed once per session.

head(clickad)

## 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 Time Date year month day  
## 1 2016-03-27 0 03:00:00 2016-03-27 2016 3 27  
## 2 2016-04-04 0 03:00:00 2016-04-04 2016 4 4  
## 3 2016-03-13 0 03:00:00 2016-03-13 2016 3 13  
## 4 2016-01-10 0 03:00:00 2016-01-10 2016 1 10  
## 5 2016-06-03 0 03:00:00 2016-06-03 2016 6 3  
## 6 2016-05-19 0 03:00:00 2016-05-19 2016 5 19

# splitting the time column into hour and minute columns  
clickad = clickad %>%  
 mutate(Time = hms(Time)) %>%  
 mutate\_at(vars(Time), funs(hour, minute))  
  
head(clickad)

## 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 Time Date year month day hour minute  
## 1 2016-03-27 0 3H 0M 0S 2016-03-27 2016 3 27 3 0  
## 2 2016-04-04 0 3H 0M 0S 2016-04-04 2016 4 4 3 0  
## 3 2016-03-13 0 3H 0M 0S 2016-03-13 2016 3 13 3 0  
## 4 2016-01-10 0 3H 0M 0S 2016-01-10 2016 1 10 3 0  
## 5 2016-06-03 0 3H 0M 0S 2016-06-03 2016 6 3 3 0  
## 6 2016-05-19 0 3H 0M 0S 2016-05-19 2016 5 19 3 0

# selecting a subset of the dataframe without the columns timestamp, time and date and storing it in a new variable  
# the columns are dropped to remove duplicate data  
clickad.nn <- subset(clickad, select = c(1,2,3,4,5,6,7,8,10,13,14,15,16,17))  
head(clickad.nn)

## 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  
## Clicked.on.Ad year month day hour minute  
## 1 0 2016 3 27 3 0  
## 2 0 2016 4 4 3 0  
## 3 0 2016 3 13 3 0  
## 4 0 2016 1 10 3 0  
## 5 0 2016 6 3 3 0  
## 6 0 2016 5 19 3 0

# dropping column 5 and 6 because they have a high number of unique values  
clickad.nn <- subset(clickad, select = c(1,2,3,4,7,8,10,13,14,15,16,17))  
  
head(clickad.nn)

## 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  
## Country Clicked.on.Ad year month day hour minute  
## 1 Tunisia 0 2016 3 27 3 0  
## 2 Nauru 0 2016 4 4 3 0  
## 3 San Marino 0 2016 3 13 3 0  
## 4 Italy 0 2016 1 10 3 0  
## 5 Iceland 0 2016 6 3 3 0  
## 6 Norway 0 2016 5 19 3 0

#install.packages("plyr")  
library("dplyr")  
library("plyr")

## -------------------------------------------------------------------------

## You have loaded plyr after dplyr - this is likely to cause problems.  
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:  
## library(plyr); library(dplyr)

## -------------------------------------------------------------------------

##   
## Attaching package: 'plyr'

## The following object is masked from 'package:lubridate':  
##   
## here

## The following object is masked from 'package:purrr':  
##   
## compact

## The following objects are masked from 'package:dplyr':  
##   
## arrange, count, desc, failwith, id, mutate, rename, summarise,  
## summarize

# displaying the value count of each country in the country column using coun tfunction from plyr library  
count(clickad.nn$Country)

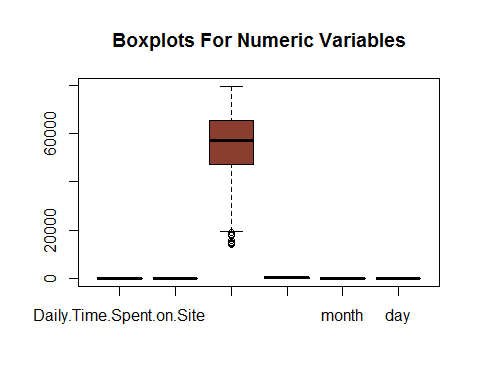
## x freq  
## 1 Afghanistan 8  
## 2 Albania 7  
## 3 Algeria 6  
## 4 American Samoa 5  
## 5 Andorra 2  
## 6 Angola 4  
## 7 Anguilla 6  
## 8 Antarctica (the territory South of 60 deg S) 3  
## 9 Antigua and Barbuda 5  
## 10 Argentina 2  
## 11 Armenia 3  
## 12 Aruba 1  
## 13 Australia 8  
## 14 Austria 5  
## 15 Azerbaijan 3  
## 16 Bahamas 7  
## 17 Bahrain 5  
## 18 Bangladesh 4  
## 19 Barbados 5  
## 20 Belarus 6  
## 21 Belgium 5  
## 22 Belize 5  
## 23 Benin 2  
## 24 Bermuda 1  
## 25 Bhutan 2  
## 26 Bolivia 6  
## 27 Bosnia and Herzegovina 7  
## 28 Bouvet Island (Bouvetoya) 5  
## 29 Brazil 5  
## 30 British Indian Ocean Territory (Chagos Archipelago) 1  
## 31 British Virgin Islands 3  
## 32 Brunei Darussalam 5  
## 33 Bulgaria 6  
## 34 Burkina Faso 4  
## 35 Burundi 7  
## 36 Cambodia 7  
## 37 Cameroon 5  
## 38 Canada 5  
## 39 Cape Verde 1  
## 40 Cayman Islands 5  
## 41 Central African Republic 2  
## 42 Chad 4  
## 43 Chile 4  
## 44 China 6  
## 45 Christmas Island 6  
## 46 Colombia 2  
## 47 Comoros 2  
## 48 Congo 4  
## 49 Cook Islands 3  
## 50 Costa Rica 6  
## 51 Cote d'Ivoire 4  
## 52 Croatia 6  
## 53 Cuba 5  
## 54 Cyprus 8  
## 55 Czech Republic 9  
## 56 Denmark 3  
## 57 Djibouti 2  
## 58 Dominica 5  
## 59 Dominican Republic 4  
## 60 Ecuador 5  
## 61 Egypt 5  
## 62 El Salvador 6  
## 63 Equatorial Guinea 4  
## 64 Eritrea 7  
## 65 Estonia 3  
## 66 Ethiopia 7  
## 67 Falkland Islands (Malvinas) 4  
## 68 Faroe Islands 3  
## 69 Fiji 7  
## 70 Finland 5  
## 71 France 9  
## 72 French Guiana 4  
## 73 French Polynesia 5  
## 74 French Southern Territories 5  
## 75 Gabon 6  
## 76 Gambia 2  
## 77 Georgia 4  
## 78 Germany 1  
## 79 Ghana 4  
## 80 Gibraltar 3  
## 81 Greece 8  
## 82 Greenland 5  
## 83 Grenada 4  
## 84 Guadeloupe 2  
## 85 Guam 4  
## 86 Guatemala 4  
## 87 Guernsey 3  
## 88 Guinea 3  
## 89 Guinea-Bissau 2  
## 90 Guyana 5  
## 91 Haiti 2  
## 92 Heard Island and McDonald Islands 3  
## 93 Holy See (Vatican City State) 3  
## 94 Honduras 5  
## 95 Hong Kong 6  
## 96 Hungary 6  
## 97 Iceland 3  
## 98 India 2  
## 99 Indonesia 6  
## 100 Iran 5  
## 101 Ireland 3  
## 102 Isle of Man 3  
## 103 Israel 4  
## 104 Italy 5  
## 105 Jamaica 5  
## 106 Japan 4  
## 107 Jersey 6  
## 108 Jordan 1  
## 109 Kazakhstan 4  
## 110 Kenya 4  
## 111 Kiribati 1  
## 112 Korea 5  
## 113 Kuwait 2  
## 114 Kyrgyz Republic 6  
## 115 Lao People's Democratic Republic 4  
## 116 Latvia 4  
## 117 Lebanon 6  
## 118 Lesotho 1  
## 119 Liberia 8  
## 120 Libyan Arab Jamahiriya 4  
## 121 Liechtenstein 6  
## 122 Lithuania 3  
## 123 Luxembourg 7  
## 124 Macao 3  
## 125 Macedonia 2  
## 126 Madagascar 6  
## 127 Malawi 4  
## 128 Malaysia 3  
## 129 Maldives 4  
## 130 Mali 4  
## 131 Malta 6  
## 132 Marshall Islands 1  
## 133 Martinique 4  
## 134 Mauritania 2  
## 135 Mauritius 4  
## 136 Mayotte 6  
## 137 Mexico 6  
## 138 Micronesia 8  
## 139 Moldova 6  
## 140 Monaco 3  
## 141 Mongolia 6  
## 142 Montenegro 2  
## 143 Montserrat 1  
## 144 Morocco 3  
## 145 Mozambique 1  
## 146 Myanmar 5  
## 147 Namibia 2  
## 148 Nauru 3  
## 149 Nepal 3  
## 150 Netherlands 4  
## 151 Netherlands Antilles 6  
## 152 New Caledonia 2  
## 153 New Zealand 4  
## 154 Nicaragua 3  
## 155 Niger 3  
## 156 Niue 3  
## 157 Norfolk Island 5  
## 158 Northern Mariana Islands 3  
## 159 Norway 2  
## 160 Pakistan 5  
## 161 Palau 4  
## 162 Palestinian Territory 3  
## 163 Panama 2  
## 164 Papua New Guinea 5  
## 165 Paraguay 3  
## 166 Peru 8  
## 167 Philippines 6  
## 168 Pitcairn Islands 2  
## 169 Poland 6  
## 170 Portugal 3  
## 171 Puerto Rico 6  
## 172 Qatar 6  
## 173 Reunion 2  
## 174 Romania 1  
## 175 Russian Federation 3  
## 176 Rwanda 5  
## 177 Saint Barthelemy 2  
## 178 Saint Helena 5  
## 179 Saint Kitts and Nevis 1  
## 180 Saint Lucia 2  
## 181 Saint Martin 4  
## 182 Saint Pierre and Miquelon 5  
## 183 Saint Vincent and the Grenadines 6  
## 184 Samoa 6  
## 185 San Marino 3  
## 186 Sao Tome and Principe 2  
## 187 Saudi Arabia 4  
## 188 Senegal 8  
## 189 Serbia 5  
## 190 Seychelles 3  
## 191 Sierra Leone 2  
## 192 Singapore 6  
## 193 Slovakia (Slovak Republic) 2  
## 194 Slovenia 1  
## 195 Somalia 5  
## 196 South Africa 8  
## 197 South Georgia and the South Sandwich Islands 2  
## 198 Spain 3  
## 199 Sri Lanka 4  
## 200 Sudan 2  
## 201 Suriname 2  
## 202 Svalbard & Jan Mayen Islands 6  
## 203 Swaziland 2  
## 204 Sweden 4  
## 205 Switzerland 4  
## 206 Syrian Arab Republic 3  
## 207 Taiwan 7  
## 208 Tajikistan 3  
## 209 Tanzania 3  
## 210 Thailand 4  
## 211 Timor-Leste 5  
## 212 Togo 3  
## 213 Tokelau 4  
## 214 Tonga 5  
## 215 Trinidad and Tobago 3  
## 216 Tunisia 4  
## 217 Turkey 8  
## 218 Turkmenistan 6  
## 219 Turks and Caicos Islands 5  
## 220 Tuvalu 4  
## 221 Uganda 4  
## 222 Ukraine 5  
## 223 United Arab Emirates 6  
## 224 United Kingdom 3  
## 225 United States Minor Outlying Islands 4  
## 226 United States of America 5  
## 227 United States Virgin Islands 4  
## 228 Uruguay 5  
## 229 Uzbekistan 2  
## 230 Vanuatu 6  
## 231 Venezuela 7  
## 232 Vietnam 3  
## 233 Wallis and Futuna 4  
## 234 Western Sahara 7  
## 235 Yemen 3  
## 236 Zambia 4  
## 237 Zimbabwe 6

# checking the length/no of unique in the country column  
length(unique(clickad.nn$Country))

## [1] 237

# the column has 237 unique countries.  
# the number of countries is very high. High number of unique values means the column can be dropped as part of feature engineering

# Plotting boxplots to check for outliers  
boxplot(clickad.nn[c(1,2,3,4,9,10)], plot=TRUE, main="Boxplots For Numeric Variables", col="coral4")



# output reveals the area income column has outliers (values lower than the minimum)

boxplot.stats(clickad.nn$Age, coef = 1.5, do.conf = TRUE, do.out = TRUE)

## $stats  
## [1] 19 29 35 42 61  
##   
## $n  
## [1] 1000  
##   
## $conf  
## [1] 34.35047 35.64953  
##   
## $out  
## integer(0)

# from the output of the statistics of the boxplot, we gather that the minimum age is 19 and the maximum age is 61. The median age is 35  
# the $n part reveals that the age column has 1000 none null values  
# the $out section reveals the column has no outliers

boxplot.stats(clickad.nn$Area.Income, coef = 1.5, do.conf = TRUE, do.out = TRUE)

## $stats  
## [1] 19345.36 47012.58 57012.30 65479.35 79484.80  
##   
## $n  
## [1] 1000  
##   
## $conf  
## [1] 56089.63 57934.97  
##   
## $out  
## [1] 17709.98 18819.34 15598.29 15879.10 14548.06 13996.50 14775.50 18368.57

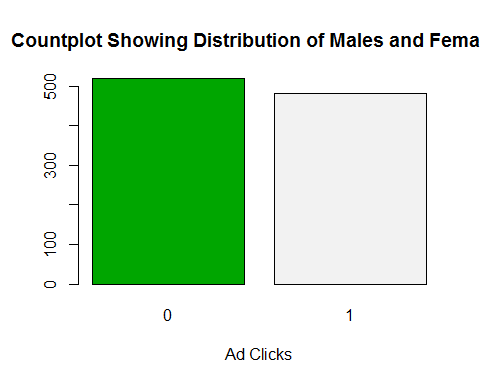
# from the output of the statistics of the boxplot, we gather that the minimum area income is 19345.36 and the maximum area income is 79484.80. The median area income is 57012.30  
# the $n part reveals that the age column has 1000 none null values  
# the $out section reveals the column has 8 outliers

# checking the variance of column 1-4 in the dataframe  
  
columns <- c(colnames(clickad.nn))  
  
for (col in columns[1:4]) {  
 print(var(clickad.nn[col]))  
   
}

## Daily.Time.Spent.on.Site  
## Daily.Time.Spent.on.Site 251.3371  
## Age  
## Age 77.18611  
## Area.Income  
## Area.Income 179952406  
## Daily.Internet.Usage  
## Daily.Internet.Usage 1927.415

# All the four columns exhibit high variance with Area income column recording the highest variance

# creating a table of total no. of males and females  
teb <- table(clickad.nn$Male)  
  
# plotting a barplot/countplot of the total no. of males and females  
barplot(teb, main = "Countplot Showing Distribution of Males and Females",xlab = "Ad Clicks",col=terrain.colors(2))



# output reveals no. of males and females is somewhat evenly distributed. The totals differ only slightly

# checking for unique values in clicked on ad column/target column  
unique(clickad.nn$Clicked.on.Ad)

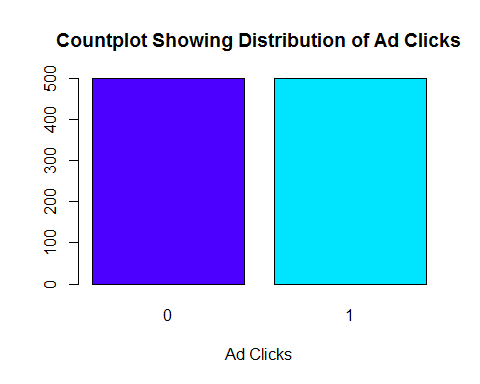
## [1] 0 1

# creating a table of the total no. of 0 and 1 in clicked on ad column  
table(clickad.nn$Clicked.on.Ad)

##   
## 0 1   
## 500 500

# target variable is balanced as both outcomes have equal no. of observations

# creating a table of total no. of 0 and 1 in the target variable  
tab <- table(clickad.nn$Clicked.on.Ad)  
  
#plotting a barplot/countplot of total no. of 0 and 1 in the target variable  
barplot(tab,main = "Countplot Showing Distribution of Ad Clicks",xlab = "Ad Clicks",col=topo.colors(2))



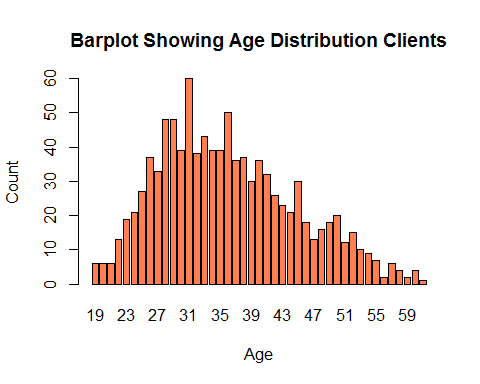
# countplot shows the values are evenly balanced at 500

# viewing a summary of the dataframe  
summary(clickad.nn[c("Daily.Time.Spent.on.Site", "Age", "Area.Income", "Daily.Internet.Usage")])

## Daily.Time.Spent.on.Site Age Area.Income   
## Min. :32.60 Min. :19.00 Min. :13996   
## 1st Qu.:51.36 1st Qu.:29.00 1st Qu.:47032   
## Median :68.22 Median :35.00 Median :57012   
## Mean :65.00 Mean :36.01 Mean :55000   
## 3rd Qu.:78.55 3rd Qu.:42.00 3rd Qu.:65471   
## Max. :91.43 Max. :61.00 Max. :79485   
## Daily.Internet.Usage  
## Min. :104.8   
## 1st Qu.:138.8   
## Median :183.1   
## Mean :180.0   
## 3rd Qu.:218.8   
## Max. :270.0

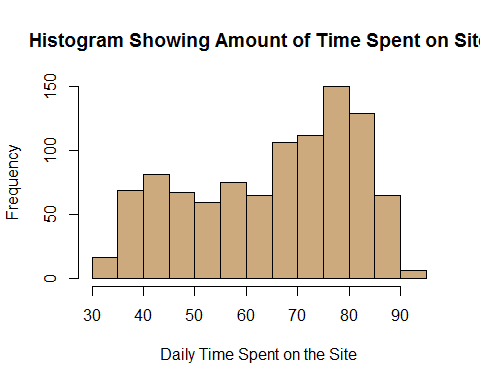
# according to output, there is need to normalize the data to reduce bias towards high values (i.e area income)

# storing age column in a new variable  
age <- clickad.nn$Age  
  
# creating a frequency table   
age.freq <- table(age)  
  
# plotting a barplot of the frequency table  
barplot(age.freq,xlab = "Age", ylab = "Count", main = "Barplot Showing Age Distribution Clients",col = "coral")



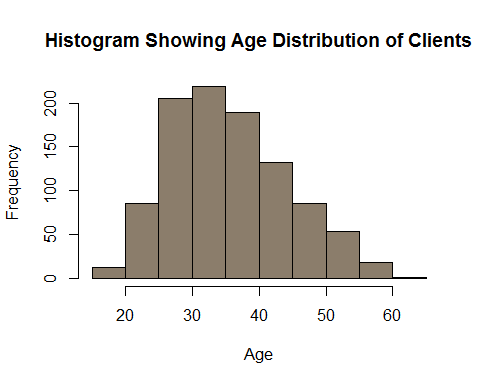
# output reveals most clients are aged between 28 and 36 years

# plotting a histogram to show distribution of daily time spent site by clients  
hist(clickad.nn$Daily.Time.Spent.on.Site, freq = T,col = "burlywood3",xlab = "Daily Time Spent on the Site", main = "Histogram Showing Amount of Time Spent on Site")



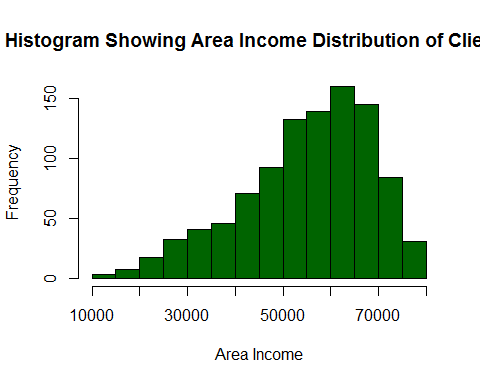
# acccording to the histogram, most users spend between 60-85hrs on the site

hist(clickad.nn$Age, freq = T,col = "bisque4",xlab = "Age", main="Histogram Showing Age Distribution of Clients")



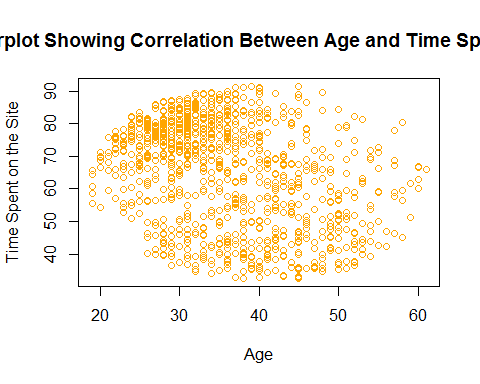
# most site's clients are between the ages of 25 and 40

hist(clickad.nn$Area.Income, freq = T,col = "darkgreen",xlab = "Area Income", main="Histogram Showing Area Income Distribution of Clients")



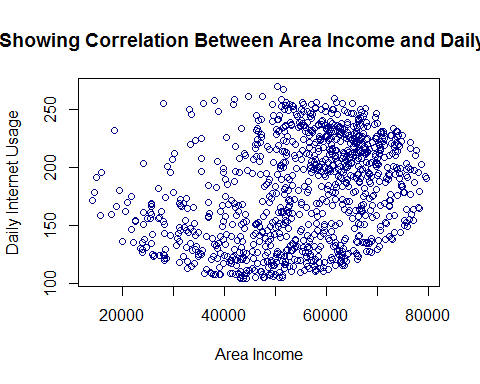
# most site's clients are between the ages of 25 and 40

tsoin <- clickad.nn$Daily.Time.Spent.on.Site  
# plotting a scatter plot between age and time spent on the site  
plot(age, tsoin, xlab="Age", ylab="Time Spent on the Site",main = "Scatterplot Showing Correlation Between Age and Time Spent on Site", col="orange")



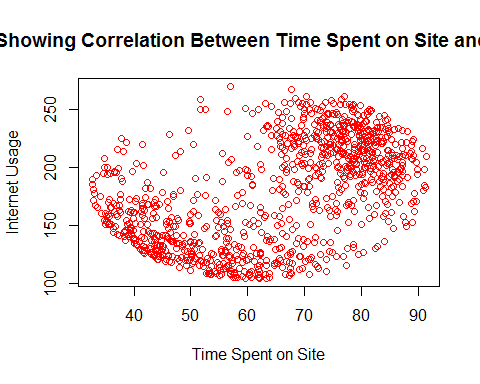
# there is a slight concentration of data points at ages 25-40 and time spent 70-90, showing a slight correlation.

# defining our x and y for the scatter  
ar.income <- clickad.nn$Area.Income  
int.con <- clickad.nn$Daily.Internet.Usage  
# plotting a scatter plot between arean income and daily internet usage  
plot(ar.income, int.con, xlab="Area Income", ylab="Daily Internet Usage", main = "Scatterplot Showing Correlation Between Area Income and Daily Internet Usage", col="blue4")



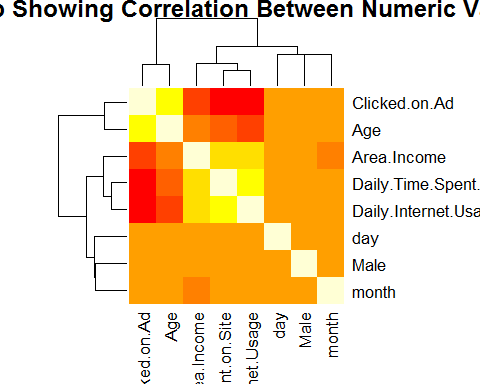
# scatter reveals slight correlation. High Income areas also record high daily internet usage

# plotting scatter plot between time spent on site and internet usage  
plot(tsoin, int.con, ylab="Internet Usage", xlab="Time Spent on Site", main = "Scatterplot Showing Correlation Between Time Spent on Site and Internet Usage",type = 'p', col="red")



# High internet usage corresponds directly to time spent on site

# getting the correlation of variables  
dat.cor <- cor(clickad.nn[c(1,2,3,4,5,7,9,10)], method = "pearson")  
palette = colorRampPalette(c("red", "orange", "white")) (20)  
# plotting a heatmap to show correlation of variables  
heatmap(dat.cor,main = "Heatmap Showing Correlation Between Numeric Variables",symm=T )

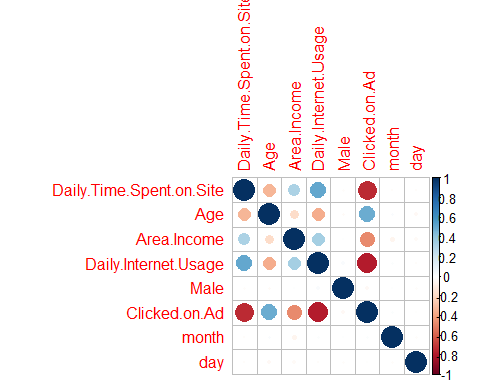


#install.packages("corrplot")

library(corrplot)

## corrplot 0.84 loaded

# plotting correlation plot to show correlation of variables  
corrplot(dat.cor)



# correlation plot reveals strong correlation between time spent on site and daily internet usage  
# plot also shows a relatively strong correlation between time spent on site and area income.

merged.frame <- clickad.nn  
head(merged.frame)

## 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  
## Country Clicked.on.Ad year month day hour minute  
## 1 Tunisia 0 2016 3 27 3 0  
## 2 Nauru 0 2016 4 4 3 0  
## 3 San Marino 0 2016 3 13 3 0  
## 4 Italy 0 2016 1 10 3 0  
## 5 Iceland 0 2016 6 3 3 0  
## 6 Norway 0 2016 5 19 3 0

# displaying internal structure of variables  
str(merged.frame)

## 'data.frame': 1000 obs. of 12 variables:  
## $ Daily.Time.Spent.on.Site: num 69 80.2 69.5 74.2 68.4 ...  
## $ Age : int 35 31 26 29 35 23 33 48 30 20 ...  
## $ Area.Income : num 61834 68442 59786 54806 73890 ...  
## $ Daily.Internet.Usage : num 256 194 236 246 226 ...  
## $ Male : int 0 1 0 1 0 1 0 1 1 1 ...  
## $ Country : Factor w/ 237 levels "Afghanistan",..: 216 148 185 104 97 159 146 13 83 79 ...  
## $ Clicked.on.Ad : int 0 0 0 0 0 0 0 1 0 0 ...  
## $ year : num 2016 2016 2016 2016 2016 ...  
## $ month : num 3 4 3 1 6 5 1 3 4 7 ...  
## $ day : int 27 4 13 10 3 19 28 7 18 11 ...  
## $ hour : num 3 3 3 3 3 3 3 3 3 3 ...  
## $ minute : num 0 0 0 0 0 0 0 0 0 0 ...

# dataframe now has 11 numerical variables and 1 factor column

#merged.frame[sapply(merged.frame, is.factor)] <- data.matrix(merged.frame[sapply(merged.frame, is.factor)])  
# converting the facor column to numeric data type  
merged.frame = merged.frame %>% mutate\_if(is.factor, as.numeric)

head(merged.frame)

## 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  
## Country Clicked.on.Ad year month day hour minute  
## 1 216 0 2016 3 27 3 0  
## 2 148 0 2016 4 4 3 0  
## 3 185 0 2016 3 13 3 0  
## 4 104 0 2016 1 10 3 0  
## 5 97 0 2016 6 3 3 0  
## 6 159 0 2016 5 19 3 0

# checking for unique values in the year, hour and minute columns  
unique(merged.frame$year)

## [1] 2016

unique(merged.frame$hour)

## [1] 3

unique(merged.frame$minute)

## [1] 0

# all the three columns have a single unique value and can thus be dropped

# dropping year, hour and minute columns because they are constant   
  
merged.fr <- subset(merged.frame, select = c(1,2,3,4,5,6,7,9,10))  
head(merged.fr)

## 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  
## Country Clicked.on.Ad month day  
## 1 216 0 3 27  
## 2 148 0 4 4  
## 3 185 0 3 13  
## 4 104 0 1 10  
## 5 97 0 6 3  
## 6 159 0 5 19

tail(merged.fr)

## Daily.Time.Spent.on.Site Age Area.Income Daily.Internet.Usage Male  
## 995 43.70 28 63126.96 173.01 0  
## 996 72.97 30 71384.57 208.58 1  
## 997 51.30 45 67782.17 134.42 1  
## 998 51.63 51 42415.72 120.37 1  
## 999 55.55 19 41920.79 187.95 0  
## 1000 45.01 26 29875.80 178.35 0  
## Country Clicked.on.Ad month day  
## 995 136 1 4 4  
## 996 117 1 2 11  
## 997 27 1 4 22  
## 998 141 1 2 1  
## 999 86 0 3 24  
## 1000 29 1 6 3

# creating a function to normalize data to reduce bias   
normalize <- function(x) {  
 return((x - min(x)) / (max(x) - min(x)))  
}

# applying the normalize function to numeric variables to normalize them  
feat.norm <- as.data.frame(lapply(merged.fr[c(1, 2, 3, 4, 5, 6, 8, 9)], normalize))  
  
# displayinga a summary of the variables to confirm the change has been effected  
summary(feat.norm)

## Daily.Time.Spent.on.Site Age Area.Income   
## Min. :0.0000 Min. :0.0000 Min. :0.0000   
## 1st Qu.:0.3189 1st Qu.:0.2381 1st Qu.:0.5044   
## Median :0.6054 Median :0.3810 Median :0.6568   
## Mean :0.5507 Mean :0.4050 Mean :0.6261   
## 3rd Qu.:0.7810 3rd Qu.:0.5476 3rd Qu.:0.7860   
## Max. :1.0000 Max. :1.0000 Max. :1.0000   
## Daily.Internet.Usage Male Country month   
## Min. :0.0000 Min. :0.000 Min. :0.0000 Min. :0.0000   
## 1st Qu.:0.2061 1st Qu.:0.000 1st Qu.:0.2288 1st Qu.:0.1667   
## Median :0.4743 Median :0.000 Median :0.4809 Median :0.5000   
## Mean :0.4554 Mean :0.481 Mean :0.4890 Mean :0.4695   
## 3rd Qu.:0.6902 3rd Qu.:1.000 3rd Qu.:0.7500 3rd Qu.:0.6667   
## Max. :1.0000 Max. :1.000 Max. :1.0000 Max. :1.0000   
## day   
## Min. :0.0000   
## 1st Qu.:0.2333   
## Median :0.4667   
## Mean :0.4828   
## 3rd Qu.:0.7333   
## Max. :1.0000

head(feat.norm)

## Daily.Time.Spent.on.Site Age Area.Income Daily.Internet.Usage Male  
## 1 0.6178820 0.3809524 0.7304725 0.9160310 0  
## 2 0.8096209 0.2857143 0.8313752 0.5387456 1  
## 3 0.6267211 0.1666667 0.6992003 0.7974331 0  
## 4 0.7062723 0.2380952 0.6231599 0.8542802 1  
## 5 0.6080231 0.3809524 0.9145678 0.7313234 0  
## 6 0.4655788 0.0952381 0.6988280 0.7383460 1  
## Country month day  
## 1 0.9110169 0.3333333 0.86666667  
## 2 0.6228814 0.5000000 0.10000000  
## 3 0.7796610 0.3333333 0.40000000  
## 4 0.4364407 0.0000000 0.30000000  
## 5 0.4067797 0.8333333 0.06666667  
## 6 0.6694915 0.6666667 0.60000000

# merging the frame of the normalized frame with clicked on ad column  
merged.frm <- cbind(feat.norm, merged.fr[c(7)])  
  
head(merged.frm)

## Daily.Time.Spent.on.Site Age Area.Income Daily.Internet.Usage Male  
## 1 0.6178820 0.3809524 0.7304725 0.9160310 0  
## 2 0.8096209 0.2857143 0.8313752 0.5387456 1  
## 3 0.6267211 0.1666667 0.6992003 0.7974331 0  
## 4 0.7062723 0.2380952 0.6231599 0.8542802 1  
## 5 0.6080231 0.3809524 0.9145678 0.7313234 0  
## 6 0.4655788 0.0952381 0.6988280 0.7383460 1  
## Country month day Clicked.on.Ad  
## 1 0.9110169 0.3333333 0.86666667 0  
## 2 0.6228814 0.5000000 0.10000000 0  
## 3 0.7796610 0.3333333 0.40000000 0  
## 4 0.4364407 0.0000000 0.30000000 0  
## 5 0.4067797 0.8333333 0.06666667 0  
## 6 0.6694915 0.6666667 0.60000000 0

tail(merged.frm)

## Daily.Time.Spent.on.Site Age Area.Income Daily.Internet.Usage  
## 995 0.1886792 0.2142857 0.7502174 0.41306454  
## 996 0.6862145 0.2619048 0.8763103 0.62840538  
## 997 0.3178650 0.6190476 0.8213020 0.17944061  
## 998 0.3234744 0.7619048 0.4339587 0.09438189  
## 999 0.3901071 0.0000000 0.4264012 0.50351132  
## 1000 0.2109468 0.1666667 0.2424754 0.44539290  
## Male Country month day Clicked.on.Ad  
## 995 0 0.5720339 0.5000000 0.10000000 1  
## 996 1 0.4915254 0.1666667 0.33333333 1  
## 997 1 0.1101695 0.5000000 0.70000000 1  
## 998 1 0.5932203 0.1666667 0.00000000 1  
## 999 0 0.3601695 0.3333333 0.76666667 0  
## 1000 0 0.1186441 0.8333333 0.06666667 1

# split data into testing & training  
# set seed to work with same values/samples  
set.seed(1234)  
  
# 80-20 train/test split   
train.index <- createDataPartition(merged.frm$Clicked.on.Ad, p = .2, list = F)  
trainn <- merged.frm[train.index, ]  
test <- merged.frm[-train.index, ]  
head(test)

## Daily.Time.Spent.on.Site Age Area.Income Daily.Internet.Usage Male  
## 1 0.6178820 0.3809524 0.7304725 0.9160310 0  
## 2 0.8096209 0.2857143 0.8313752 0.5387456 1  
## 3 0.6267211 0.1666667 0.6992003 0.7974331 0  
## 4 0.7062723 0.2380952 0.6231599 0.8542802 1  
## 6 0.4655788 0.0952381 0.6988280 0.7383460 1  
## 8 0.5677375 0.6904762 0.1618126 0.1633370 1  
## Country month day Clicked.on.Ad  
## 1 0.91101695 0.3333333 0.8666667 0  
## 2 0.62288136 0.5000000 0.1000000 0  
## 3 0.77966102 0.3333333 0.4000000 0  
## 4 0.43644068 0.0000000 0.3000000 0  
## 6 0.66949153 0.6666667 0.6000000 0  
## 8 0.05084746 0.3333333 0.2000000 1

# get predictors/x.train  
predictor <- trainn %>%  
 select(-c(Clicked.on.Ad, Country)) %>%  
 as.matrix()  
# define x.test  
output <- trainn$Clicked.on.Ad %>%  
 as.factor()  
  
str(output)

## Factor w/ 2 levels "0","1": 1 1 1 2 1 1 2 2 1 1 ...

class(output)

## [1] "factor"

# train a random forest model for Classification  
model <- randomForest(x = predictor, y = output,  
 ntree = 50) # number of trees  
  
# check out the details  
model

##   
## Call:  
## randomForest(x = predictor, y = output, ntree = 50)   
## Type of random forest: classification  
## Number of trees: 50  
## No. of variables tried at each split: 2  
##   
## OOB estimate of error rate: 3%  
## Confusion matrix:  
## 0 1 class.error  
## 0 97 3 0.03  
## 1 3 97 0.03

rfpred = predict(model, type="response", newdata= as.data.frame(predictor))  
  
# summary(rfpred)  
  
table(output, rfpred)

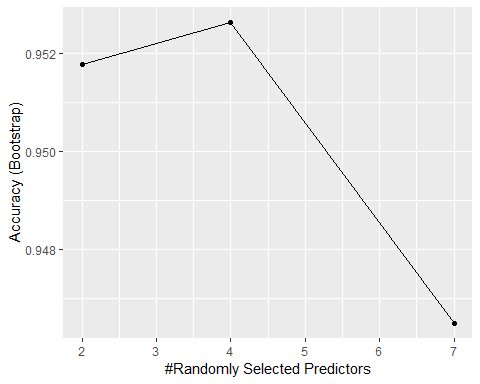
## rfpred  
## output 0 1  
## 0 100 0  
## 1 0 100

#accracy <- (385 + 388) / (385 + 15 + 12 + 388) \* 100  
#accracy

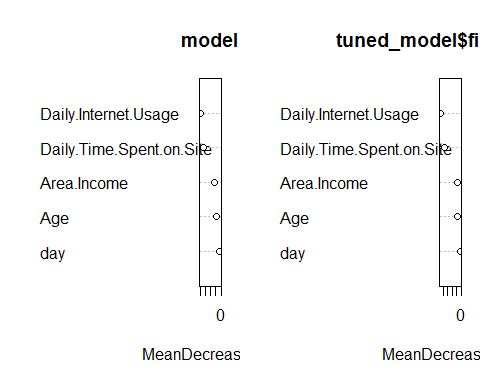
#use caret to pick a value for mtry  
  
#install.packages("caret")   
#install.packages('e1071', dependencies=TRUE)  
  
  
tuned\_model <- train(x = predictor, y = output,  
 ntree = 10, # number of trees (passed on random forest)  
 method = "rf") # random forests  
  
tuned\_model

## Random Forest   
##   
## 200 samples  
## 7 predictor  
## 2 classes: '0', '1'   
##   
## No pre-processing  
## Resampling: Bootstrapped (25 reps)   
## Summary of sample sizes: 200, 200, 200, 200, 200, 200, ...   
## Resampling results across tuning parameters:  
##   
## mtry Accuracy Kappa   
## 2 0.9517663 0.9029545  
## 4 0.9526285 0.9048145  
## 7 0.9465011 0.8923866  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was mtry = 4.

# plot the rmse for various possible training values  
ggplot(tuned\_model)



# plot both plots at once  
par(mfrow = c(1,2))  
  
varImpPlot(model, n.var = 5)  
varImpPlot(tuned\_model$finalModel, n.var = 5)



tuned\_model$finalModel

##   
## Call:  
## randomForest(x = x, y = y, ntree = 10, mtry = param$mtry)   
## Type of random forest: classification  
## Number of trees: 10  
## No. of variables tried at each split: 4  
##   
## OOB estimate of error rate: 6.06%  
## Confusion matrix:  
## 0 1 class.error  
## 0 92 6 0.06122449  
## 1 6 94 0.06000000

# training a logisitic regression model  
logit <- glm(Clicked.on.Ad ~ Daily.Time.Spent.on.Site + Age + Area.Income + Daily.Internet.Usage + Male + month + day, family="binomial", data=merged.frm)  
summary(logit)

##   
## Call:  
## glm(formula = Clicked.on.Ad ~ Daily.Time.Spent.on.Site + Age +   
## Area.Income + Daily.Internet.Usage + Male + month + day,   
## family = "binomial", data = merged.frm)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.4797 -0.1330 -0.0286 0.0171 3.2371   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 16.33150 1.76494 9.253 < 2e-16 \*\*\*  
## Daily.Time.Spent.on.Site -11.37025 1.22143 -9.309 < 2e-16 \*\*\*  
## Age 7.21594 1.10187 6.549 5.80e-11 \*\*\*  
## Area.Income -9.01030 1.24807 -7.219 5.22e-13 \*\*\*  
## Daily.Internet.Usage -10.63548 1.14590 -9.281 < 2e-16 \*\*\*  
## Male -0.42762 0.40576 -1.054 0.292   
## month -0.04069 0.62821 -0.065 0.948   
## day -0.84912 0.72887 -1.165 0.244   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 1386.29 on 999 degrees of freedom  
## Residual deviance: 180.43 on 992 degrees of freedom  
## AIC: 196.43  
##   
## Number of Fisher Scoring iterations: 8

# the summary reveals that the important features are Daily.Time.Spent.on.Site, Age, Area.Income and Daily.Internet.Usage

# making predictions on training data  
predictTrain = predict(logit, type="response")  
  
summary(predictTrain)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.001706 0.008717 0.373276 0.500000 0.999850 0.999999

tapply(predictTrain, merged.fr$Clicked.on.Ad, mean)

## 0 1   
## 0.04770271 0.95229729

# Confusion matrix for threshold of 0.5  
table(merged.fr$Clicked.on.Ad, predictTrain > 0.5)

##   
## FALSE TRUE  
## 0 490 10  
## 1 18 482

acc <- (490 + 482) / (490 + 10 + 18 + 482) \* 100  
acc

## [1] 97.2

# test set  
predictorr <- test %>%  
 select(-c(Clicked.on.Ad, Country)) %>%  
 as.matrix()  
  
outputt <- test$Clicked.on.Ad %>%  
 as.factor()

# making predicitions on test set  
predicted = predict(logit, type="response", newdata= as.data.frame(predictorr))  
   
summary(predicted)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.001706 0.008923 0.356660 0.498592 0.999824 0.999999

table(outputt,predicted >= 0.3)

##   
## outputt FALSE TRUE  
## 0 385 15  
## 1 12 388

accuracy <- (385 + 388) / (385 + 15 + 12 + 388) \* 100  
accuracy

## [1] 96.625

#tapply(predicted, trainn$Clicked.on.Ad, mean)

Using XGBoost

# training set features and labels  
predor <- trainn %>%  
 select(-c(Clicked.on.Ad, Country)) %>%  
 as.matrix()  
  
outut <- trainn$Clicked.on.Ad %>%  
 as.numeric()

# test set features and labels  
predorr <- test %>%  
 select(-c(Clicked.on.Ad, Country)) %>%  
 as.matrix()  
  
oututt <- test$Clicked.on.Ad %>%  
 as.numeric()

# put our testing & training data into two seperates Dmatrixs objects  
dtrain <- xgb.DMatrix(data = predor, label= outut)  
dtest <- xgb.DMatrix(data = predorr, label= oututt)  
head(output)

## [1] 0 0 0 1 0 0  
## Levels: 0 1

# train a model using our training data  
xg.model <- xgboost(data = dtrain, # the data   
 nround = 2, # max number of boosting iterations  
 objective = "binary:logistic") # the objective function

## [1] train-error:0.030000   
## [2] train-error:0.030000

# generate predictions for our held-out testing data  
predd <- predict(xg.model, dtest)  
  
# get & print the classification error  
err <- mean(as.numeric(predd > 0.5) != oututt)  
print(paste("test-error=", err))

## [1] "test-error= 0.0775"

# train a tuned xgboost model  
xg.model.tuned <- xgboost(data = dtrain, # the data   
 max.depth = 3, # the maximum depth of each decision tree  
 nround = 2, # max number of boosting iterations  
 objective = "binary:logistic") # the objective function

## [1] train-error:0.030000   
## [2] train-error:0.030000

# generate predictions for our held-out testing data  
preed <- predict(xg.model.tuned, dtest)  
  
# get & print the classification error  
errr <- mean(as.numeric(preed > 0.5) != oututt)  
print(paste("test-error=", errr))

## [1] "test-error= 0.0775"

# get the number of negative & positive cases in our data  
negative\_cases <- sum(outut == FALSE)  
postive\_cases <- sum(outut == TRUE)  
  
# train a model using our training data  
model.tuned <- xgboost(data = dtrain, # the data   
 max.depth = 3, # the maximum depth of each decision tree  
 nround = 10, # number of boosting rounds  
 early\_stopping\_rounds = 3, # if we dont see an improvement in this many rounds, stop  
 objective = "binary:logistic", # the objective function  
 scale\_pos\_weight = negative\_cases/postive\_cases) # control for imbalanced classes

## [1] train-error:0.030000   
## Will train until train\_error hasn't improved in 3 rounds.  
##   
## [2] train-error:0.030000   
## [3] train-error:0.030000   
## [4] train-error:0.020000   
## [5] train-error:0.015000   
## [6] train-error:0.015000   
## [7] train-error:0.015000   
## [8] train-error:0.015000   
## Stopping. Best iteration:  
## [5] train-error:0.015000

# generate predictions for our held-out testing data  
prred <- predict(model.tuned, dtest)  
  
# get & print the classification error  
erro <- mean(as.numeric(prred > 0.5) != oututt)  
print(paste("test-error=", erro))

## [1] "test-error= 0.05875"

table(oututt,prred >= 0.3)

##   
## oututt FALSE TRUE  
## 0 362 38  
## 1 20 380

# get accuracy  
accuy <- (362 + 380) / (362 + 38 + 20 + 380) \* 100  
accuy

## [1] 92.75

# train a model using our training data  
mdel.tuned <- xgboost(data = dtrain, # the data   
 max.depth = 3, # the maximum depth of each decision tree  
 nround = 10, # number of boosting rounds  
 early\_stopping\_rounds = 3, # if we dont see an improvement in this many rounds, stop  
 objective = "binary:logistic", # the objective function  
 scale\_pos\_weight = negative\_cases/postive\_cases, # control for imbalanced classes  
 gamma = 1) # add a regularization term

## [1] train-error:0.030000   
## Will train until train\_error hasn't improved in 3 rounds.  
##   
## [2] train-error:0.030000   
## [3] train-error:0.030000   
## [4] train-error:0.020000   
## [5] train-error:0.015000   
## [6] train-error:0.015000   
## [7] train-error:0.015000   
## [8] train-error:0.015000   
## Stopping. Best iteration:  
## [5] train-error:0.015000

# generate predictions for our held-out testing data  
ppred <- predict(mdel.tuned, dtest)  
  
# get & print the classification error  
errr <- mean(as.numeric(ppred > 0.5) != oututt)  
print(paste("test-error=", errr))

## [1] "test-error= 0.06125"

# generate confusion matrix using table  
table(oututt,ppred >= 0.3)

##   
## oututt FALSE TRUE  
## 0 363 37  
## 1 20 380

# get accuracy  
accy <- (363 + 380) / (363 + 37 + 20 + 380) \* 100  
accy

## [1] 92.875

Using K Nearest Neighbors

# Defining train and test sets  
knnf.train <- merged.frm[1:500, 1:8]  
knnf.test <- merged.frm[501:1000, 1:8]  
knntes.train <- merged.frm[1:500, 9]  
knntes.test <- merged.frm[501:1000, 9]  
length(knntes.test)

## [1] 500

#fitting the model with data  
library(class)  
knn.mod <- knn(train=knnf.train, test=knnf.test, cl= knntes.train, k=10)

#creating a confusion matrix using table function  
tabo <- table(knn.mod, knntes.test)  
tabo

## knntes.test  
## knn.mod 0 1  
## 0 244 23  
## 1 5 228

# get accuracy  
akuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) \* 100}  
  
akuracy(tabo)

## [1] 94.4

#install.packages("gmodels")  
library(gmodels)  
CrossTable(x = knntes.test, y=knn.mod, prop.chisq=F)

##   
##   
## Cell Contents  
## |-------------------------|  
## | N |  
## | N / Row Total |  
## | N / Col Total |  
## | N / Table Total |  
## |-------------------------|  
##   
##   
## Total Observations in Table: 500   
##   
##   
## | knn.mod   
## knntes.test | 0 | 1 | Row Total |   
## -------------|-----------|-----------|-----------|  
## 0 | 244 | 5 | 249 |   
## | 0.980 | 0.020 | 0.498 |   
## | 0.914 | 0.021 | |   
## | 0.488 | 0.010 | |   
## -------------|-----------|-----------|-----------|  
## 1 | 23 | 228 | 251 |   
## | 0.092 | 0.908 | 0.502 |   
## | 0.086 | 0.979 | |   
## | 0.046 | 0.456 | |   
## -------------|-----------|-----------|-----------|  
## Column Total | 267 | 233 | 500 |   
## | 0.534 | 0.466 | |   
## -------------|-----------|-----------|-----------|  
##   
##