edx Capstone : Choose Your Own Project

Taisuke Matsuki

06 June 2024

Introduction

This project is related to the "Choose Your Own Project Submission" project in HervardX: PH125.9x Data Science and is a capstone course.

I will prepare and set up a data set of my own choosing. I then conduct an exploratory data analysis to develop two different machine learning models. The results obtained by applying the machine learning algorithms will be defined. Finally, the project will conclude with a conclusion. The provided dataset is "Customer Personality Analysis," which, as the theme suggests, is an open dataset in "Kaggle".

The model I intend to develop is a "Linear regression", and the second is a "Random forest".

The rationale is that I am currently a data analyst in the data analysis team of an FMCG company in Japan.

I have not yet had the opportunity to develop machine learning models, or rather, there are few cases where machine learning is needed. However, in order to become a more skilled data scientist, I believe it is necessary to have experience in understanding and implementing models that are frequently used in Japan. I am willing to take on this challenge, despite the difficulty. When gathering information on social networking sites, such as Twitter, it seems that logistic regression, decision trees, and LightGBM are often used in the Japanese business scene, as well as linear regression and random forests, which were selected for this project. This background also influenced my approach to this modeling challenge. Furthermore, the nature of the dataset is similar to the data I usually handle in my own work, and I have domain knowledge in marketing, so I am confident that I can produce good results by leveraging my strengths.

Supplemental information on Accuracy Metrics

Model1 Linear Regression Model Accuracy metric to be used R-Squared

Model2 Random Forest Model Accuracy metric to be used Accuracy

Recommendation Algorithm Creation Process

1.Data Set loading

2.Exploratory data analysis (EDA)

3.Feature engineering as needed

4.Build several models to increase accuracy

5.Accuracy evaluation

Data Set loading & Data Wrangling & Feature Engineering

```
##########################################################
# Load libraries you need or may need at once
##########################################################
if(!require(tidyverse)) install.packages("tidyverse")
if(!require(lubridate)) install.packages("lubridate")
if(!require(zoo)) install.packages("zoo")
if(!require(scales)) install.packages("scales")
if(!require(patchwork)) install.packages("patchwork")
if(!require(editData)) install.packages("editData")
if(!require(corrplot)) install.packages("corrplot")
if(!require(car)) install.packages("car")
if(!require(ggrepel)) install.packages("ggrepel")
if(!require(caret)) install.packages("caret")
if(!require(randomForest)) install.packages("randomForest")
if(!require(rsample)) install.packages("rsample")
if(!require(imputeTS)) install.packages("imputeTS")
if(!require(Boruta)) install.packages("Boruta")
if(!require(knitr)) install.packages("knitr")



library(tidyverse)
library(lubridate)
library(zoo)
library(scales)
library(patchwork)
library(editData)
library(corrplot)
library(car)
library(ggrepel)
library(caret)
library(Boruta)
```

```r
library(imputeTS)
library(randomForest)
library(rsample)
library(knitr)


#################################
# Loading Data Sets
#################################
df <- read.csv("Purchase_dataset.csv")


################################
# Data Wrangling
################################
# Confirmation of data types in each column
dim(df)
```

```
## [1] 2240   29
```

```r
str(df)
```

```
## 'data.frame':    2240 obs. of  29 variables:
##  $ ID                 : int  5524 2174 4141 6182 5324 7446 965 6177 4855 5899 ...
##  $ Year_Birth         : int  1957 1954 1965 1984 1981 1967 1971 1985 1974 1950 ...
##  $ Education          : chr  "Graduation" "Graduation" "Graduation" "Graduation" ...
##  $ Marital_Status     : chr  "Single" "Single" "Together" "Together" ...
##  $ Income             : int  58138 46344 71613 26646 58293 62513 55635 33454 30351 5648 ...
##  $ Kidhome            : int  0 1 0 1 1 0 0 1 1 1 ...
##  $ Teenhome           : int  0 1 0 0 0 1 1 0 0 1 ...
##  $ Dt_Customer        : chr  "04-09-2012" "08-03-2014" "21-08-2013" "10-02-2014" ...
##  $ Recency            : int  58 38 26 26 94 16 34 32 19 68 ...
##  $ MntWines           : int  635 11 426 11 173 520 235 76 14 28 ...
##  $ MntFruits          : int  88 1 49 4 43 42 65 10 0 0 ...
##  $ MntMeatProducts    : int  546 6 127 20 118 98 164 56 24 6 ...
##  $ MntFishProducts    : int  172 2 111 10 46 0 50 3 3 1 ...
##  $ MntSweetProducts   : int  88 1 21 3 27 42 49 1 3 1 ...
##  $ MntGoldProds       : int  88 6 42 5 15 14 27 23 2 13 ...
##  $ NumDealsPurchases  : int  3 2 1 2 5 2 4 2 1 1 ...
```

```
##  $ NumWebPurchases    : int  8 1 8 2 5 6 7 4 3 1 ...
##  $ NumCatalogPurchases: int  10 1 2 0 3 4 3 0 0 0 ...
##  $ NumStorePurchases  : int  4 2 10 4 6 10 7 4 2 0 ...
##  $ NumWebVisitsMonth  : int  7 5 4 6 5 6 6 8 9 20 ...
##  $ AcceptedCmp3       : int  0 0 0 0 0 0 0 0 0 1 ...
##  $ AcceptedCmp4       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ AcceptedCmp5       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ AcceptedCmp1       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ AcceptedCmp2       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Complain           : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Z_CostContact      : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ Z_Revenue          : int  11 11 11 11 11 11 11 11 11 11 ...
##  $ Response           : int  1 0 0 0 0 0 0 0 1 0 ...
```

```
summary(df)
```

```
##        ID          Year_Birth    Education         Marital_Status
##  Min.   :    0   Min.   :1893   Length:2240        Length:2240
##  1st Qu.: 2828   1st Qu.:1959   Class :character   Class :character
##  Median : 5458   Median :1970   Mode  :character   Mode  :character
##  Mean   : 5592   Mean   :1969
##  3rd Qu.: 8428   3rd Qu.:1977
##  Max.   :11191   Max.   :1996
##
##      Income          Kidhome          Teenhome      Dt_Customer
##  Min.   :  1730   Min.   :0.0000   Min.   :0.0000   Length:2240
##  1st Qu.: 35303   1st Qu.:0.0000   1st Qu.:0.0000   Class :character
##  Median : 51382   Median :0.0000   Median :0.0000   Mode  :character
##  Mean   : 52247   Mean   :0.4442   Mean   :0.5062
##  3rd Qu.: 68522   3rd Qu.:1.0000   3rd Qu.:1.0000
##  Max.   :666666   Max.   :2.0000   Max.   :2.0000
##  NA's   :24
##     Recency         MntWines         MntFruits     MntMeatProducts
##  Min.   : 0.00   Min.   :   0.00   Min.   :  0.0   Min.   :   0.0
##  1st Qu.:24.00   1st Qu.:  23.75   1st Qu.:  1.0   1st Qu.:  16.0
##  Median :49.00   Median : 173.50   Median :  8.0   Median :  67.0
##  Mean   :49.11   Mean   : 303.94   Mean   : 26.3   Mean   : 166.9
##  3rd Qu.:74.00   3rd Qu.: 504.25   3rd Qu.: 33.0   3rd Qu.: 232.0
##  Max.   :99.00   Max.   :1493.00   Max.   :199.0   Max.   :1725.0
##
```

```
##  MntFishProducts  MntSweetProducts  MntGoldProds    NumDealsPurchases
##  Min.   :  0.00   Min.   :  0.00   Min.   :  0.00   Min.   : 0.000
##  1st Qu.:  3.00   1st Qu.:  1.00   1st Qu.:  9.00   1st Qu.: 1.000
##  Median : 12.00   Median :  8.00   Median : 24.00   Median : 2.000
##  Mean   : 37.53   Mean   : 27.06   Mean   : 44.02   Mean   : 2.325
##  3rd Qu.: 50.00   3rd Qu.: 33.00   3rd Qu.: 56.00   3rd Qu.: 3.000
##  Max.   :259.00   Max.   :263.00   Max.   :362.00   Max.   :15.000
##
##  NumWebPurchases  NumCatalogPurchases NumStorePurchases NumWebVisitsMonth
##  Min.   : 0.000   Min.   : 0.000      Min.   : 0.00     Min.   : 0.000
##  1st Qu.: 2.000   1st Qu.: 0.000      1st Qu.: 3.00     1st Qu.: 3.000
##  Median : 4.000   Median : 2.000      Median : 5.00     Median : 6.000
##  Mean   : 4.085   Mean   : 2.662      Mean   : 5.79     Mean   : 5.317
##  3rd Qu.: 6.000   3rd Qu.: 4.000      3rd Qu.: 8.00     3rd Qu.: 7.000
##  Max.   :27.000   Max.   :28.000      Max.   :13.00     Max.   :20.000
##
##   AcceptedCmp3      AcceptedCmp4      AcceptedCmp5      AcceptedCmp1
##  Min.   :0.00000   Min.   :0.00000   Min.   :0.00000   Min.   :0.00000
##  1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000
##  Median :0.00000   Median :0.00000   Median :0.00000   Median :0.00000
##  Mean   :0.07277   Mean   :0.07455   Mean   :0.07277   Mean   :0.06429
##  3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000
##  Max.   :1.00000   Max.   :1.00000   Max.   :1.00000   Max.   :1.00000
##
##   AcceptedCmp2        Complain        Z_CostContact   Z_Revenue
##  Min.   :0.00000   Min.   :0.000000   Min.   :3     Min.   :11
##  1st Qu.:0.00000   1st Qu.:0.000000   1st Qu.:3     1st Qu.:11
##  Median :0.00000   Median :0.000000   Median :3     Median :11
##  Mean   :0.01339   Mean   :0.009375   Mean   :3     Mean   :11
##  3rd Qu.:0.00000   3rd Qu.:0.000000   3rd Qu.:3     3rd Qu.:11
##  Max.   :1.00000   Max.   :1.000000   Max.   :3     Max.   :11
##
##     Response
##  Min.   :0.0000
##  1st Qu.:0.0000
##  Median :0.0000
##  Mean   :0.1491
##  3rd Qu.:0.0000
##  Max.   :1.0000
##
```

```r
# Identify more detailed data characteristics of the columns of interest
unique(df$Education)
```

```
## [1] "Graduation" "PhD"        "Master"     "Basic"      "2n Cycle"
```

```r
unique(df$Marital_Status)
```

```
## [1] "Single"   "Together" "Married"  "Divorced" "Widow"    "Alone"    "Absurd"
## [8] "YOLO"
```

```r
unique(df$Marital_Status)
```

```
## [1] "Single"   "Together" "Married"  "Divorced" "Widow"    "Alone"    "Absurd"
## [8] "YOLO"
```

```r
unique(df$Z_CostContact)
```

```
## [1] 3
```

```r
unique(df$Z_Revenue)
```

```
## [1] 11
```

```r
# Rename columns for easier understanding
df <- df %>% rename("Education_Level" = Education,
                    "Kids_in_Family" = Kidhome,
                    "Teenager_in_Family" = Teenhome,
                    "Registration_Date" = Dt_Customer,
                    "Elapse_Date_from_Last_Purchase" = Recency,
                    "Purchase_Wines_past2years" = MntWines,
                    "Purchase_Fruits_past2years" = MntFruits,
                    "Purchase_Meat_past2years" = MntMeatProducts,
                    "Purchase_Fish_past2years" = MntFishProducts,
                    "Purchase_Snacks_past2years" = MntSweetProducts,
                    "Purchase_Golds_past2years" = MntGoldProds,
                    "Purchase_Number_in_Sales" = NumDealsPurchases,
                    "Web_Purchase" = NumWebPurchases,
                    "Store_Purchase" = NumStorePurchases,
                    "Catalog_Purchase" = NumCatalogPurchases,
                    "Web_visits_LastMonth" = NumWebVisitsMonth,
```

```
              "Complains_past2years" = Complain)

# Remove columns that were not well understood by looking at the column descriptions on the dataset
df <- df %>% select(-Z_CostContact,-Z_Revenue)


# Convert the Complains past 2 years column to a logical type
df <- df %>% mutate(Complains_past2years = case_when(Complains_past2years == 1 ~ TRUE,
                                                     Complains_past2years == 0 ~ FALSE))


str_length(df$Registration_Date) %>% unique()
```

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

```
df$Registration_Year <- str_sub(df$Registration_Date,start = 7)

class(df$Registration_Year)
```

```
## [1] "character"
```

```
unique(df$Registration_Year)
```

```
## [1] "2012" "2014" "2013"
```

```
df$Registration_Month <- str_sub(df$Registration_Date,start = 4, end = 5)

class(df$Registration_Month)
```

```
## [1] "character"
```

```
unique(df$Registration_Month)
```

```
##  [1] "09" "03" "08" "02" "01" "11" "05" "06" "10" "12" "04" "07"
```

```
df$Registration_Day <- str_sub(df$Registration_Date,start = 1, end = 2)

class(df$Registration_Day)
```

```
## [1] "character"
```

```r
unique(df$Registration_Day)
```

```
##  [1] "04" "08" "21" "10" "19" "09" "13" "06" "15" "24" "31" "28" "03" "23" "11"
## [16] "18" "02" "27" "20" "22" "29" "01" "12" "05" "07" "17" "14" "30" "25" "16"
## [31] "26"
```

```r
df$Registration_Date <- str_c(df$Registration_Year,df$Registration_Month,df$Registration_Month, sep
  as.Date()

# Identify the base date in this data set
summary(df$Registration_Date) # I found that the most recent registration date is 06/06/2014.
```

```
##         Min.      1st Qu.       Median         Mean      3rd Qu.         Max.
## "2012-07-07" "2013-01-01" "2013-07-07" "2013-07-01" "2013-12-12" "2014-06-06"
```

```r
df %>% filter(Registration_Date >= "2014-06-06") %>%
  arrange(desc(Elapse_Date_from_Last_Purchase)) %>%
  summary(Elapse_Date_from_Last_Purchase) # I found that the maximum number of days is 99.
```

```
##        ID          Year_Birth   Education_Level    Marital_Status
##  Min.   :    1   Min.   :1946   Length:74          Length:74
##  1st Qu.: 2794   1st Qu.:1959   Class :character   Class :character
##  Median : 4882   Median :1966   Mode  :character   Mode  :character
##  Mean   : 5328   Mean   :1967
##  3rd Qu.: 7383   3rd Qu.:1976
##  Max.   :10785   Max.   :1986
##      Income      Kids_in_Family  Teenager_in_Family Registration_Date
##  Min.   : 4023   Min.   :0.000   Min.   :0.0000     Min.   :2014-06-06
##  1st Qu.:36833   1st Qu.:0.000   1st Qu.:0.0000     1st Qu.:2014-06-06
##  Median :50991   Median :0.000   Median :1.0000     Median :2014-06-06
##  Mean   :52918   Mean   :0.473   Mean   :0.5811     Mean   :2014-06-06
##  3rd Qu.:69248   3rd Qu.:1.000   3rd Qu.:1.0000     3rd Qu.:2014-06-06
##  Max.   :85485   Max.   :2.000   Max.   :2.0000     Max.   :2014-06-06
##  Elapse_Date_from_Last_Purchase Purchase_Wines_past2years
##  Min.   : 0.00                  Min.   :   1.00
##  1st Qu.:29.00                  1st Qu.:  22.25
##  Median :49.00                  Median :  68.00
##  Mean   :48.59                  Mean   : 229.08
##  3rd Qu.:70.25                  3rd Qu.: 458.00
```

```
##  Max.   :99.00                  Max.   :1060.00
##  Purchase_Fruits_past2years Purchase_Meat_past2years Purchase_Fish_past2years
##  Min.   :  0.00             Min.   :  1.0            Min.   :  0.0
##  1st Qu.:  0.00             1st Qu.:  8.5            1st Qu.:  0.0
##  Median :  4.00             Median : 25.0            Median :  5.0
##  Mean   : 23.88             Mean   :126.7            Mean   : 34.2
##  3rd Qu.: 34.75             3rd Qu.:200.0            3rd Qu.: 34.0
##  Max.   :164.00             Max.   :835.0            Max.   :227.0
##  Purchase_Snacks_past2years Purchase_Golds_past2years Purchase_Number_in_Sales
##  Min.   :  0.0              Min.   :  0.00            Min.   : 0.000
##  1st Qu.:  0.0              1st Qu.:  3.25            1st Qu.: 1.000
##  Median :  3.5              Median : 10.50            Median : 1.000
##  Mean   : 19.0              Mean   : 34.11            Mean   : 2.108
##  3rd Qu.: 20.0              3rd Qu.: 35.00            3rd Qu.: 2.000
##  Max.   :189.0              Max.   :241.00            Max.   :15.000
##   Web_Purchase    Catalog_Purchase Store_Purchase   Web_visits_LastMonth
##  Min.   : 0.000   Min.   : 0.000   Min.   : 0.000   Min.   : 1.000
##  1st Qu.: 1.250   1st Qu.: 0.000   1st Qu.: 3.000   1st Qu.: 2.000
##  Median : 3.000   Median : 1.000   Median : 4.000   Median : 5.000
##  Mean   : 3.297   Mean   : 2.014   Mean   : 4.892   Mean   : 4.527
##  3rd Qu.: 4.000   3rd Qu.: 3.000   3rd Qu.: 6.000   3rd Qu.: 6.000
##  Max.   :10.000   Max.   :10.000   Max.   :13.000   Max.   :19.000
##   AcceptedCmp3      AcceptedCmp4      AcceptedCmp5      AcceptedCmp1
##  Min.   :0.00000   Min.   :0.00000   Min.   :0.00000   Min.   :0.0000
##  1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.0000
##  Median :0.00000   Median :0.00000   Median :0.00000   Median :0.0000
##  Mean   :0.06757   Mean   :0.08108   Mean   :0.02703   Mean   :0.1081
##  3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.0000
##  Max.   :1.00000   Max.   :1.00000   Max.   :1.00000   Max.   :1.0000
##   AcceptedCmp2     Complains_past2years    Response       Registration_Year
##  Min.   :0.00000   Mode :logical        Min.   :0.00000   Length:74
##  1st Qu.:0.00000   FALSE:74             1st Qu.:0.00000   Class :character
##  Median :0.00000                        Median :0.00000   Mode  :character
##  Mean   :0.02703                        Mean   :0.06757
##  3rd Qu.:0.00000                        3rd Qu.:0.00000
##  Max.   :1.00000                        Max.   :1.00000
##  Registration_Month Registration_Day
##  Length:74          Length:74
##  Class :character   Class :character
##  Mode  :character   Mode  :character
```

```
##
##
##
```

```
# Add the latest base date column in this data set
df_Latest <- df %>% filter(Registration_Date >= "2014-06-06")
as.integer(df_Latest$Registration_Date) + 99 %>%
  as.Date() # I found that the Reference Date is 09/13/2014.
```

```
##  [1] "2014-09-13" "2014-09-13" "2014-09-13" "2014-09-13" "2014-09-13"
##  [6] "2014-09-13" "2014-09-13" "2014-09-13" "2014-09-13" "2014-09-13"
## [11] "2014-09-13" "2014-09-13" "2014-09-13" "2014-09-13" "2014-09-13"
## [16] "2014-09-13" "2014-09-13" "2014-09-13" "2014-09-13" "2014-09-13"
## [21] "2014-09-13" "2014-09-13" "2014-09-13" "2014-09-13" "2014-09-13"
## [26] "2014-09-13" "2014-09-13" "2014-09-13" "2014-09-13" "2014-09-13"
## [31] "2014-09-13" "2014-09-13" "2014-09-13" "2014-09-13" "2014-09-13"
## [36] "2014-09-13" "2014-09-13" "2014-09-13" "2014-09-13" "2014-09-13"
## [41] "2014-09-13" "2014-09-13" "2014-09-13" "2014-09-13" "2014-09-13"
## [46] "2014-09-13" "2014-09-13" "2014-09-13" "2014-09-13" "2014-09-13"
## [51] "2014-09-13" "2014-09-13" "2014-09-13" "2014-09-13" "2014-09-13"
## [56] "2014-09-13" "2014-09-13" "2014-09-13" "2014-09-13" "2014-09-13"
## [61] "2014-09-13" "2014-09-13" "2014-09-13" "2014-09-13" "2014-09-13"
## [66] "2014-09-13" "2014-09-13" "2014-09-13" "2014-09-13" "2014-09-13"
## [71] "2014-09-13" "2014-09-13" "2014-09-13" "2014-09-13"
```

```
df$Reference_Date_on_Dataset <- as.Date("2014-09-13")

# Sort Columns
df <- df %>% select(1:8,Reference_Date_on_Dataset,everything())


# Add an age column
df$Reference_Year_on_Dataset <- year(df$Reference_Date_on_Dataset)
df$Reference_Year_on_Dataset <- as.integer(df$Reference_Year_on_Dataset)
df$Age <- df$Reference_Year_on_Dataset - df$Year_Birth
df <- df %>% select(ID,Year_Birth,Age,everything())

# Add registration month column for aggregate
df <- df %>% select(-Registration_Year,-Registration_Month,-Registration_Day)
df$Registration_Month <- floor_date(df$Registration_Date,"month")
```

```r
# Add Total Children in Family column
df$Total_Children_in_Family <- df$Kids_in_Family + df$Teenager_in_Family
df <- df %>% select(1:8,Total_Children_in_Family,everything())

# Missing value of Income: NA is complemented by the mean value.
summary(df$Income)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    1730   35303   51382   52247   68522  666666      24
```

```r
Avg_Income <- mean(df$Income, na.rm = T)
Avg_Income
```

```
## [1] 52247.25
```

```r
df <- df %>% replace_na(list(Income = 52247))

# Education Level Label encoding  ##################
# Basic:1 Graduation:2 2n Cycle:3 Master:3 PhD:4

df <- df %>% mutate(Education_Level_Label = case_when(Education_Level == "Basic" ~ 1,
                                                      Education_Level == "Graduation" ~ 2,
                                                      Education_Level == "2n Cycle" ~ 3,
                                                      Education_Level == "Master" ~ 3,
                                                      Education_Level == "PhD" ~ 4))
df <- df %>% select(1:4,Education_Level_Label,everything())

# Marital Status Label encoding  ##################
# Basic:1 Graduation:2 2n Cycle:3 Master:3 PhD:4
unique(df$Marital_Status)
```

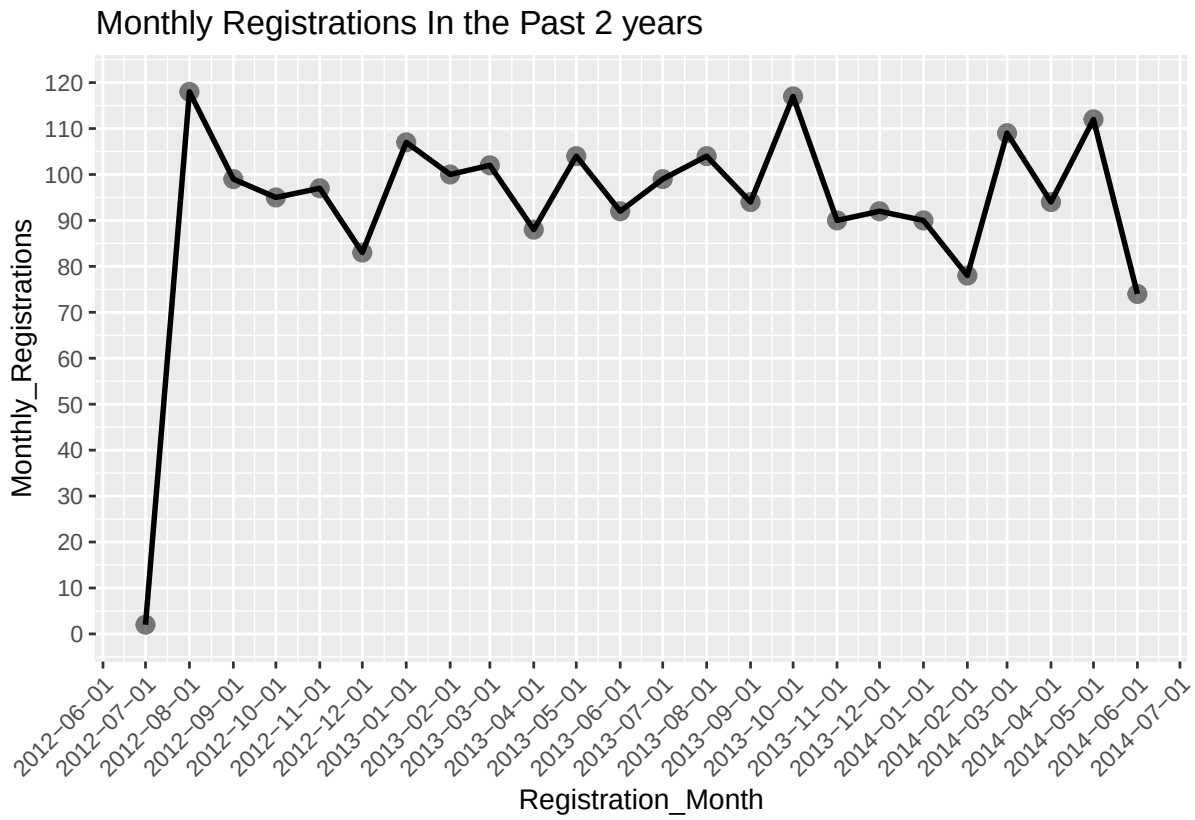
```
## [1] "Single"   "Together" "Married"  "Divorced" "Widow"    "Alone"    "Absurd"
## [8] "YOLO"
```

```r
df <- df %>% mutate(Current_Marital_Label = case_when(Marital_Status == "Single" ~ FALSE,
                                                      Marital_Status == "Together" ~ TRUE,
                                                      Marital_Status == "Married" ~ TRUE,
                                                      Marital_Status == "Divorced" ~ FALSE,
                                                      Marital_Status == "Widow" ~ FALSE,
```

```
                                                Marital_Status == "Alone" ~ FALSE,
                                                Marital_Status == "Absurd" ~ FALSE,
                                                Marital_Status == "YOLO" ~ FALSE)) %>%
  select(1:6,Current_Marital_Label,everything())
```

Exploratory data analysis (EDA)

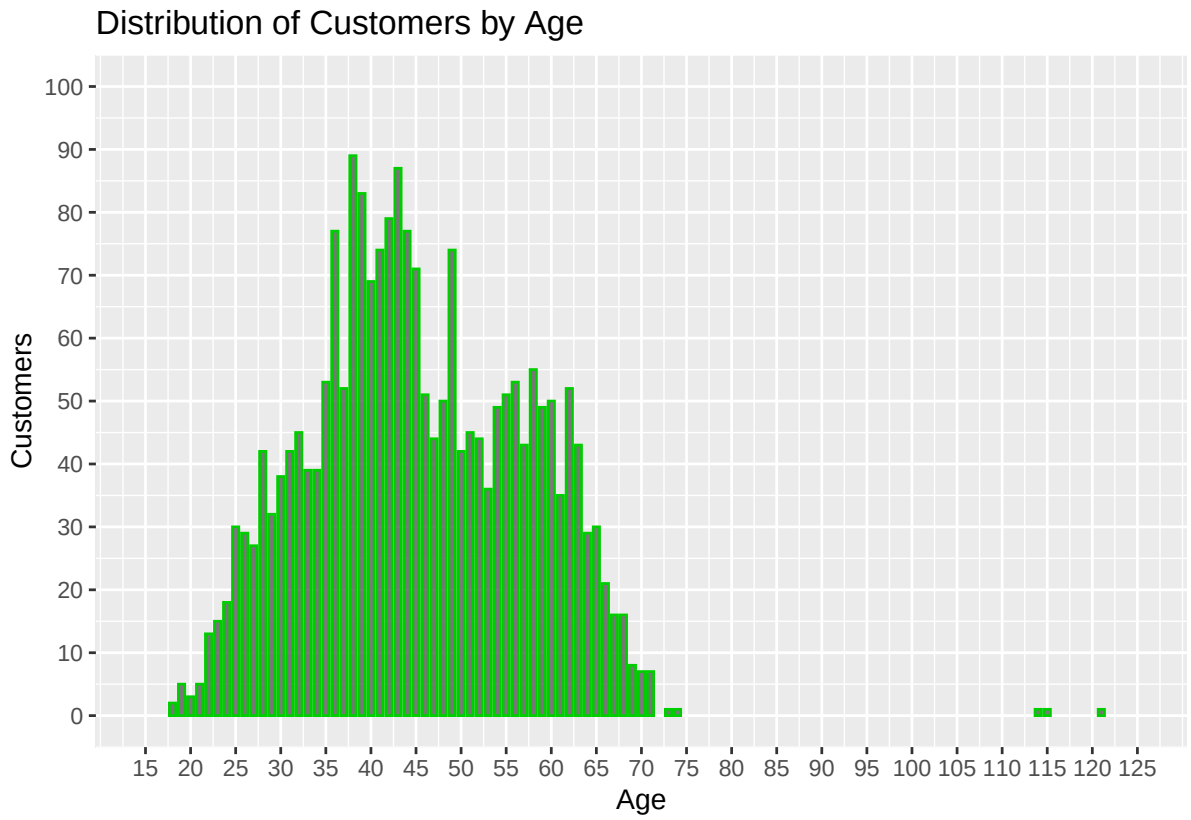
```
# Confirmation the number of registrants by month
df_Monthly_Registrations <- df %>% distinct(ID,.keep_all = T) %>%
  group_by(Registration_Month) %>%
  summarise(Monthly_Registrations = n())

df_Monthly_Registrations %>%
  ggplot(aes(Registration_Month,Monthly_Registrations))+
  geom_line(linewidth = 1)+
  geom_point(size = 3,alpha = 0.5)+
  scale_y_continuous(breaks = seq(0,120,10),limits = c(0,120))+
  scale_x_date(date_breaks = "1 month")+
  theme(axis.text.x = element_text(angle = 45, hjust = 1))+
  labs(title = "Monthly Registrations In the Past 2 years")
```




```
# Confirmation the age distribution of Customers
summary(df$Age)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   18.00   37.00   44.00   45.19   55.00  121.00
```

```
df %>% group_by(Age) %>%
  summarise(Customers_by_Age = n()) %>%
  ggplot(aes(Age,Customers_by_Age))+
  geom_bar(stat = "identity", position = "dodge", width = 0.7, alpha = 0.8,colour="green3")+
  scale_y_continuous(breaks = seq(0,100,10),limits = c(0,100))+
  scale_x_continuous(breaks = seq(15,125,5),limits = c(15,125),labels = label_comma())+
  labs(title = "Distribution of Customers by Age", y = "Customers")
```

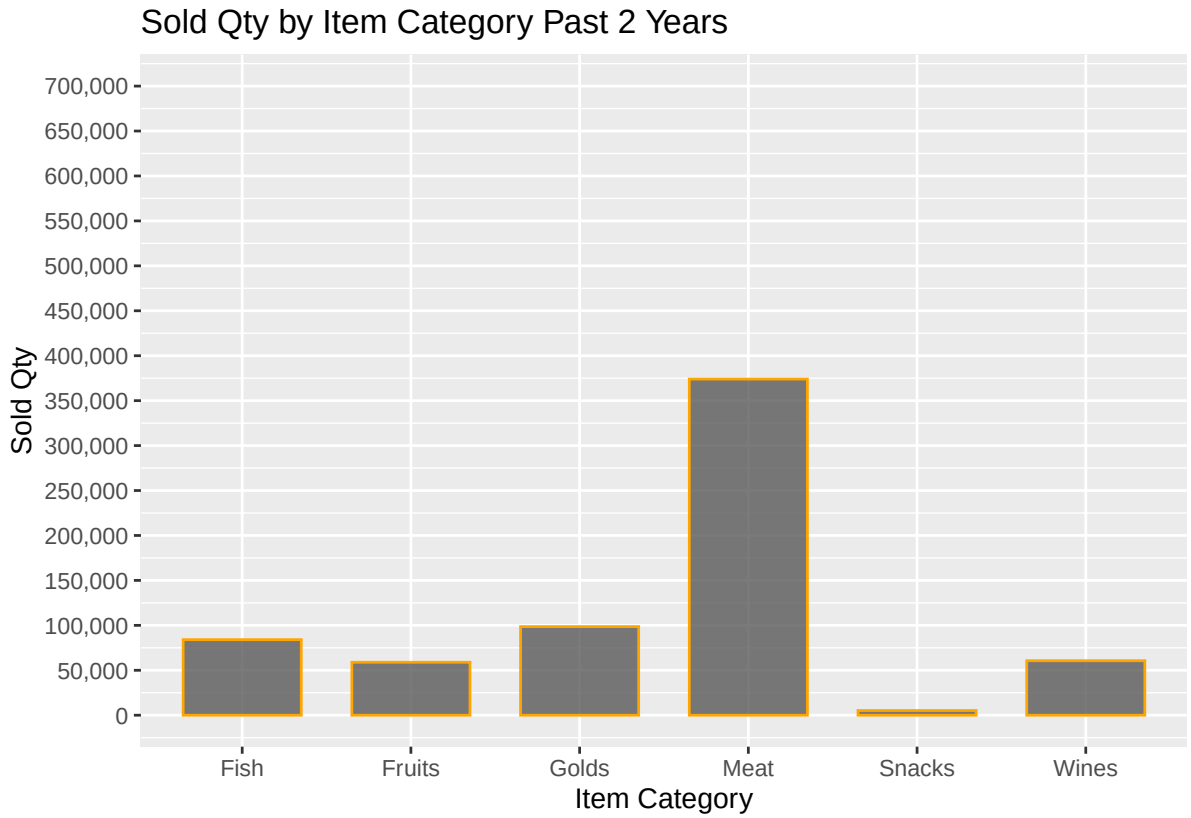



```
# Confirmation the number of products purchased in the past two years
df_Sold_Qty <- df %>% select(16:21) %>% apply(2,sum) %>% as.data.frame()
df_Sold_Qty <- rownames_to_column(df_Sold_Qty)
str(df_Sold_Qty)
```

```
## 'data.frame':    6 obs. of  2 variables:
##  $ rowname: chr  "Purchase_Fruits_past2years" "Purchase_Meat_past2years" "Purchase_Fish_past2year
##  $ .      : int  58917 373968 84057 60621 98609 5208
```

```
df_Sold_Qty <- df_Sold_Qty %>%  rename("Item_Category" = rowname)
df_Sold_Qty$Sold_Qty <- df_Sold_Qty$.
df_Sold_Qty <- df_Sold_Qty %>% select(1,3)
df_Sold_Qty <- df_Sold_Qty %>% arrange(desc(Sold_Qty))

df_Sold_Qty %>%
  ggplot(aes(Item_Category,Sold_Qty))+
  geom_bar(stat = "identity", position = "dodge",width = 0.7,alpha = 0.8, colour = "orange1")+
  scale_y_continuous(breaks = seq(0,700000,50000),limits = c(0,700000),labels = label_comma())+
  scale_x_discrete(labels = c("Fish","Fruits","Golds","Meat","Snacks","Wines"))+
```

```
  labs(x = "Item Category", y= "Sold Qty",title = "Sold Qty by Item Category Past 2 Years")
```

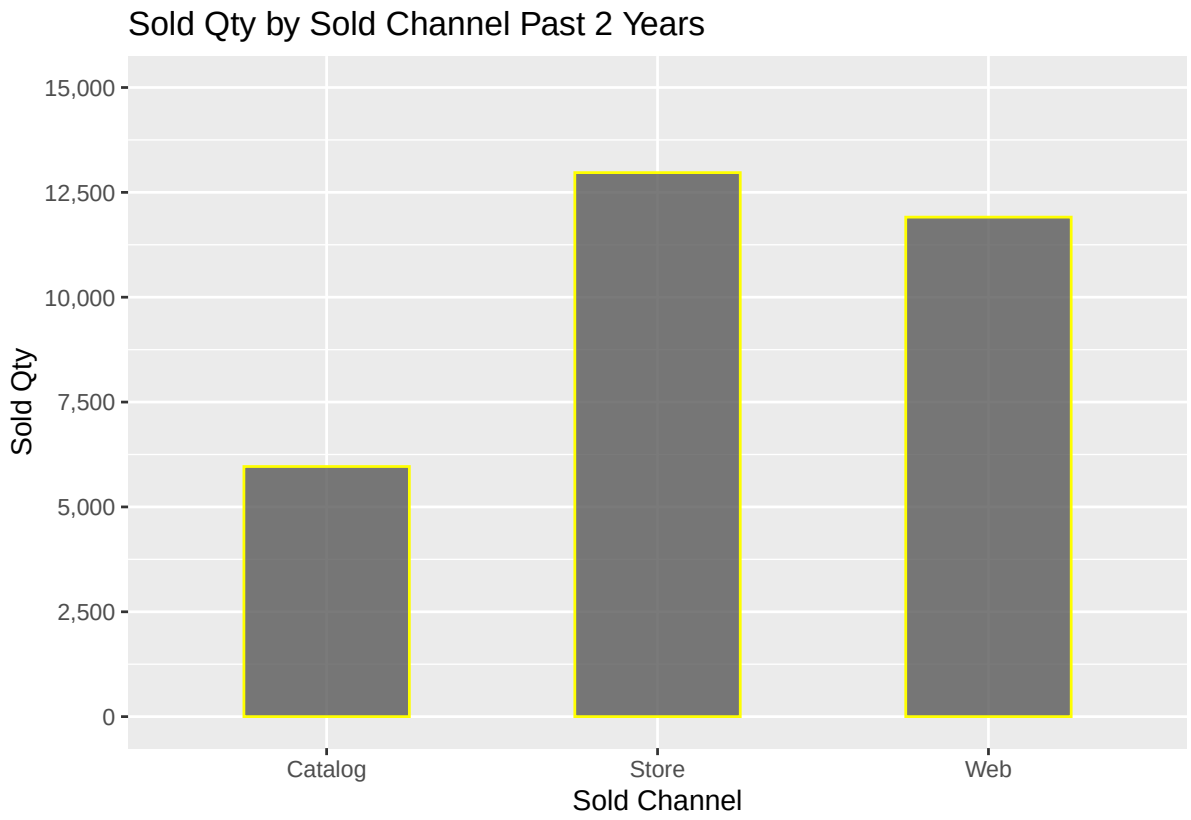



```
# Confirmation the number of purchases by sales channel over the past two years
df_Sold_Channel <- df %>% select(23:25) %>% apply(2,sum) %>% as.data.frame()
df_Sold_Channel <- rownames_to_column(df_Sold_Channel)
str(df_Sold_Channel)
```

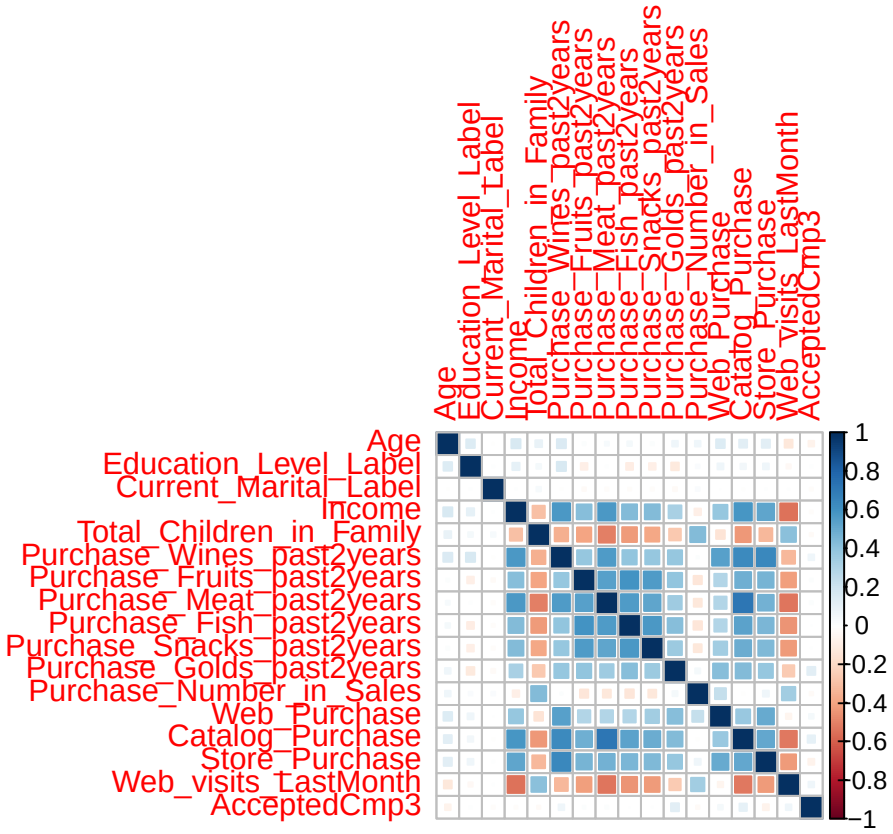
```
## 'data.frame':    3 obs. of  2 variables:
##  $ rowname: chr  "Catalog_Purchase" "Store_Purchase" "Web_visits_LastMonth"
##  $ .      : int  5963 12970 11909
```

```
df_Sold_Channel <- df_Sold_Channel %>%  rename("Sold_Channel" = rowname)
df_Sold_Channel$Sold_Qty <- df_Sold_Channel$.
df_Sold_Channel <- df_Sold_Channel %>% select(1,3)
df_Sold_Channel <- df_Sold_Channel %>% arrange(desc(Sold_Qty))

df_Sold_Channel %>%
  ggplot(aes(Sold_Channel,Sold_Qty))+
  geom_bar(stat = "identity", position = "dodge",width = 0.5,alpha = 0.8, colour = "yellow1")+
```

```
  scale_y_continuous(breaks = seq(0,15000,2500),limits = c(0,15000),labels = label_comma())+
  scale_x_discrete(labels = c("Catalog","Store","Web"))+
  labs(x = "Sold Channel", y= "Sold Qty",title = "Sold Qty by Sold Channel Past 2 Years")
```




```
# Confirmation of correlation coefficients
corr_df <- df %>% select(-Kids_in_Family,-ID,-Year_Birth,-Education_Level,-Marital_Status,-Teenager_
                         -Elapse_Date_from_Last_Purchase,-Response,-Reference_Year_on_Dataset,-Regis
  select(-18,-19,-20,-21,-22)


corr <- cor(corr_df, method = "pearson")
corrplot(corr, method = "square")
```




```
# Add Age Generation Column
df$Age_Group <- round(df$Age,digits = -1)
```

Build Linear regression Models & Accuracy evaluation

Supplemental Explanation

Because of the relatively large number of features, feature engineering and data wrangling are quite key. The best model is built by combining several variables and comparing R-Squared while also taking into account multicollinearity.

```
######################
# Data Splitting
######################
# Split Train data to 80% and Test data to 20%.
# Let's review the purpose of each data.
# Train data: data used to train the model
# Test data: data used to check the generalization performance of the final selected model
```

```r
nrow(df) # Number of data sets = 2,240
```

```
## [1] 2240
```

```r
set.seed(2000)
df_split_dataset <- initial_split(df, prop = 0.8)
df_train <- training(df_split_dataset)
df_test <- testing(df_split_dataset)


############################################################
# Train a Linear Regression model using Train data.
############################################################
# Model 1
Model_lr_1 <- lm(df_train$Income ~ df_train$Age + df_train$Purchase_Meat_past2years
                 + df_train$Purchase_Wines_past2years + df_train$Purchase_Snacks_past2years
                 + df_train$Catalog_Purchase + df_train$Store_Purchase + df_train$Web_Purchase
                 + df_train$Web_visits_LastMonth
                 + df_train$Total_Children_in_Family + df_train$Education_Level_Label,
                 data = df_train)

summary(Model_lr_1)
```

```
##
## Call:
## lm(formula = df_train$Income ~ df_train$Age + df_train$Purchase_Meat_past2years +
##     df_train$Purchase_Wines_past2years + df_train$Purchase_Snacks_past2years +
##     df_train$Catalog_Purchase + df_train$Store_Purchase + df_train$Web_Purchase +
##     df_train$Web_visits_LastMonth + df_train$Total_Children_in_Family +
##     df_train$Education_Level_Label, data = df_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -101243   -6304    -543    5152  629162
##
## Coefficients:
##                                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     45755.504   2931.404  15.609  < 2e-16 ***
## df_train$Age                       45.480     38.530   1.180  0.23801
```

```
## df_train$Purchase_Meat_past2years      20.781      3.210   6.474 1.23e-10 ***
## df_train$Purchase_Wines_past2years     15.878      2.052   7.737 1.69e-14 ***
## df_train$Purchase_Snacks_past2years    29.142     13.791   2.113  0.03474 *
## df_train$Catalog_Purchase             660.988    248.402   2.661  0.00786 **
## df_train$Store_Purchase               281.470    196.753   1.431  0.15273
## df_train$Web_Purchase                1343.066    214.281   6.268 4.58e-10 ***
## df_train$Web_visits_LastMonth       -3591.270    249.984 -14.366  < 2e-16 ***
## df_train$Total_Children_in_Family    3704.276    706.126   5.246 1.74e-07 ***
## df_train$Education_Level_Label        914.242    539.313   1.695  0.09021 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 18510 on 1781 degrees of freedom
## Multiple R-squared:  0.4943, Adjusted R-squared:  0.4914
## F-statistic: 174.1 on 10 and 1781 DF,  p-value: < 2.2e-16
```

```
Model_1_R2 <- summary(Model_lr_1)$adj.r.squared

# Check the multicollinearity
# Generally, if the Multicollinearity is less than or equal to 10, there is no Multicollinearity pro
vif(Model_lr_1)
```

```
##                        df_train$Age   df_train$Purchase_Meat_past2years
##                            1.111219                            2.727906
##  df_train$Purchase_Wines_past2years df_train$Purchase_Snacks_past2years
##                            2.455487                            1.655489
##           df_train$Catalog_Purchase             df_train$Store_Purchase
##                            2.789059                            2.105675
##               df_train$Web_Purchase       df_train$Web_visits_LastMonth
##                            1.789893                            1.876865
##   df_train$Total_Children_in_Family      df_train$Education_Level_Label
##                            1.460931                            1.074757
```

```
# Model 2
Model_lr_2 <- lm(df_train$Income ~ df_train$Purchase_Meat_past2years
                 + df_train$Purchase_Wines_past2years + df_train$Purchase_Snacks_past2years
                 + df_train$Catalog_Purchase + df_train$Store_Purchase + df_train$Web_Purchase
                 + df_train$Web_visits_LastMonth
                 + df_train$Total_Children_in_Family,
                 data = df_train)
```

```r
summary(Model_lr_2)
```

```
##
## Call:
## lm(formula = df_train$Income ~ df_train$Purchase_Meat_past2years +
##     df_train$Purchase_Wines_past2years + df_train$Purchase_Snacks_past2years +
##     df_train$Catalog_Purchase + df_train$Store_Purchase + df_train$Web_Purchase +
##     df_train$Web_visits_LastMonth + df_train$Total_Children_in_Family,
##     data = df_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -102107   -6169    -549    5254  628298
##
## Coefficients:
##                                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)                      50266.291   1996.653  25.175  < 2e-16 ***
## df_train$Purchase_Meat_past2years     20.161      3.197   6.306 3.61e-10 ***
## df_train$Purchase_Wines_past2years    16.637      2.023   8.225 3.73e-16 ***
## df_train$Purchase_Snacks_past2years   24.979     13.664   1.828  0.06771 .
## df_train$Catalog_Purchase            685.830    248.219   2.763  0.00579 **
## df_train$Store_Purchase              273.553    196.840   1.390  0.16479
## df_train$Web_Purchase               1378.313    213.159   6.466 1.29e-10 ***
## df_train$Web_visits_LastMonth      -3668.675    246.388 -14.890  < 2e-16 ***
## df_train$Total_Children_in_Family   3934.673    695.295   5.659 1.77e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 18520 on 1783 degrees of freedom
## Multiple R-squared:  0.4929, Adjusted R-squared:  0.4906
## F-statistic: 216.6 on 8 and 1783 DF,  p-value: < 2.2e-16
```

```r
Model_2_R2 <- summary(Model_lr_2)$adj.r.squared

# Check the multicollinearity
vif(Model_lr_2)
```

```
##   df_train$Purchase_Meat_past2years  df_train$Purchase_Wines_past2years
##                            2.701802                            2.382159
```

```
## df_train$Purchase_Snacks_past2years           df_train$Catalog_Purchase
##                            1.622605                            2.780701
##             df_train$Store_Purchase               df_train$Web_Purchase
##                            2.104325                            1.768493
##       df_train$Web_visits_LastMonth   df_train$Total_Children_in_Family
##                            1.820473                            1.414300
```

```
# Model 3
Model_lr_3 <- lm(df_train$Income ~ df_train$Purchase_Meat_past2years
                 + df_train$Purchase_Wines_past2years + df_train$Catalog_Purchase
                 + df_train$Store_Purchase + df_train$Web_Purchase
                 + df_train$Web_visits_LastMonth
                 + df_train$Total_Children_in_Family,
                 data = df_train)

summary(Model_lr_3)
```

```
##
## Call:
## lm(formula = df_train$Income ~ df_train$Purchase_Meat_past2years +
##     df_train$Purchase_Wines_past2years + df_train$Catalog_Purchase +
##     df_train$Store_Purchase + df_train$Web_Purchase + df_train$Web_visits_LastMonth +
##     df_train$Total_Children_in_Family, data = df_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -105014   -6265    -563    5230  628043
##
## Coefficients:
##                                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)                       50705.425   1983.451  25.564  < 2e-16 ***
## df_train$Purchase_Meat_past2years    21.206      3.148   6.737 2.18e-11 ***
## df_train$Purchase_Wines_past2years   16.364      2.019   8.106 9.58e-16 ***
## df_train$Catalog_Purchase           719.034    247.716   2.903  0.00375 **
## df_train$Store_Purchase             321.384    195.222   1.646  0.09989 .
## df_train$Web_Purchase              1450.116    209.646   6.917 6.41e-12 ***
## df_train$Web_visits_LastMonth     -3734.170    243.929 -15.308  < 2e-16 ***
## df_train$Total_Children_in_Family  3760.473    689.186   5.456 5.54e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 18530 on 1784 degrees of freedom
## Multiple R-squared:  0.492,  Adjusted R-squared:  0.49
## F-statistic: 246.8 on 7 and 1784 DF,  p-value: < 2.2e-16
```

```
Model_3_R2 <- summary(Model_lr_3)$adj.r.squared
# Check the multicollinearity
vif(Model_lr_3)
```

```
##  df_train$Purchase_Meat_past2years df_train$Purchase_Wines_past2years
##                           2.615410                           2.369173
##          df_train$Catalog_Purchase            df_train$Store_Purchase
##                           2.765811                           2.067144
##              df_train$Web_Purchase      df_train$Web_visits_LastMonth
##                           1.708444                           1.781980
##  df_train$Total_Children_in_Family
##                           1.387734
```

```
######################
# Create RMSE function
######################
RMSE <- function(m, o) {
  tmp <- sqrt(mean((m-o)^2))
  return(tmp)
}


#############################################################################################
# Calculate predictions for each model, calculate RMSE, and select the optimal model.
#############################################################################################
pred_Model_1 <- predict(Model_lr_1,newdata = df_test)
pred_Model_2 <- predict(Model_lr_2,newdata = df_test)
pred_Model_3 <- predict(Model_lr_3,newdata = df_test)

RMSE_Model1 <- RMSE(pred_Model_1,df_test$Income)
RMSE_Model2 <- RMSE(pred_Model_2,df_test$Income)
RMSE_Model3 <- RMSE(pred_Model_3,df_test$Income)


##################################
```

```r
# Summarize the results in a table
#################################
Model <- c("Model 1", "Model 2", "Model 3")
R2 <- c(Model_1_R2,Model_2_R2,Model_3_R2)
RMSE <- c(RMSE_Model1,RMSE_Model2,RMSE_Model3)

report_table <- data.frame(Model = Model ,R2 = R2, RMSE = RMSE)

kable(report_table, caption = "Summary of Results for each Models")
```

1: Summary of Results for each Models

Model	R2	RMSE
Model 1	0.4914164	27663.58
Model 2	0.4906396	27636.39
Model 3	0.4899709	27609.03

Build Random Forest Models & Accuracy evaluation

Supplemental Explanation

What is Random Forest?

Combining multiple models is called "ensemble learning". There are three commonly used ensemble learning methods, as far as I know I will briefly describe them in the context of decision trees, 1. bagging → using multiple decision trees 2. boosting → focusing on errors in a single decision tree and learning many times to reduce those errors 3. Stacking → not only decision trees, but also various models are combined to make predictions. Random forests are the bagging of decision trees and are used in machine learning applications such as "classification" and "regression. It is unique in that it can achieve higher accuracy than using"decision trees" alone.

Random Forest Algorithm Details

The general algorithm of Random Forest is as follows

1. construct n bootstrap datasets from the original data
2. generate n decision trees from the data set
3. randomly select m features from p features
4. make a majority decision of n decision trees in the case of classification, or an average of n decision trees in the case of regression, as the final prediction In ensemble learning, the lower the correlation

between models, the more accurate the predictions, so only some features are used in the third step. The idea is that if we collect decision trees that are over-trained in different directions and take the average of the results, we can reduce the degree of over-training.

Advantages and Disadvantages of Random Forest

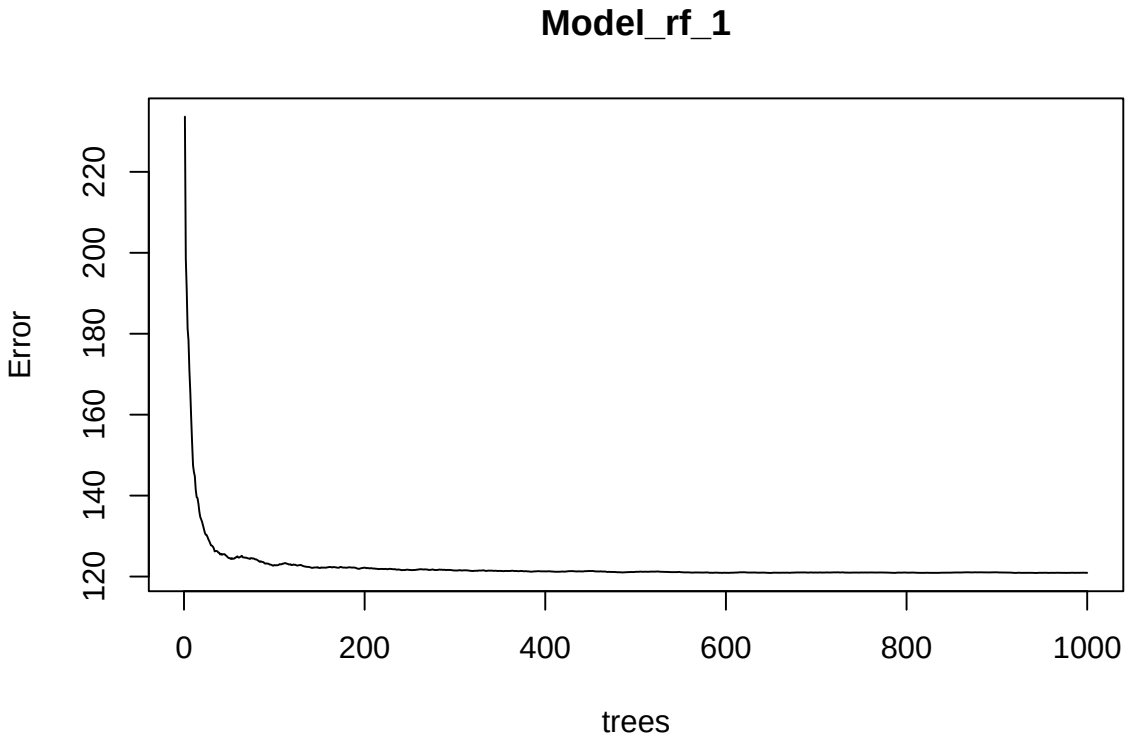
Advantages

Speedy learning and identification are possible even with large scale data. Efficient learning is possible even when the number of dimensions increases. No need for normalization or standardization of features

Disadvantages

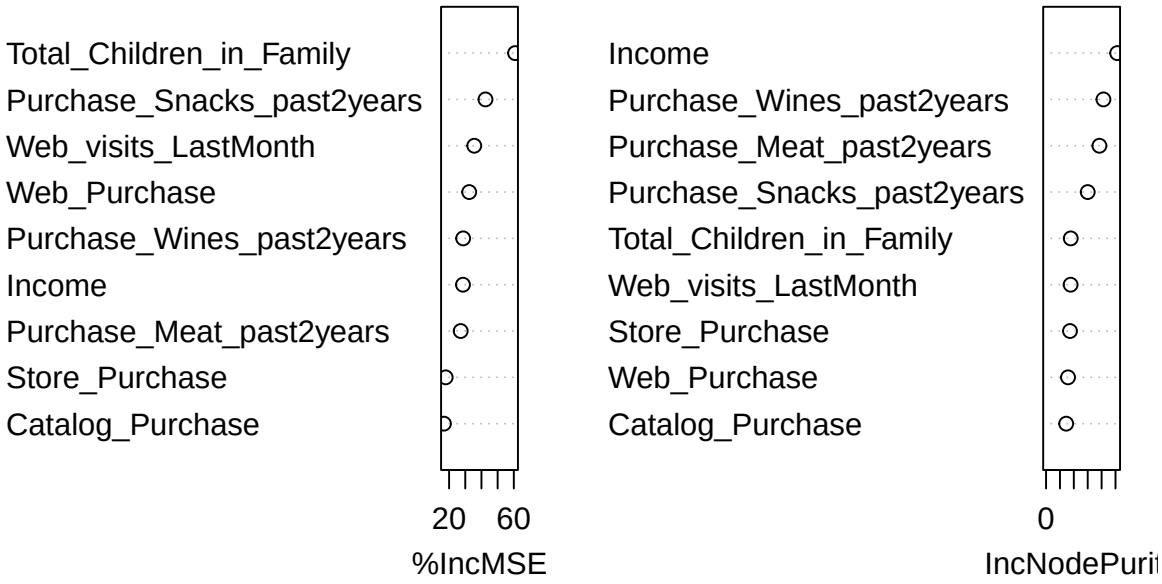
Over-learning due to decision trees is likely to occur. Accuracy may not improve if training data is small.

```
############################################################
# Build a Random Forest model using Train data.
############################################################

df_test$Age <- as.numeric(df_test$Age)
num_trees <- 1000

# Model 1
set.seed(3000)
Model_rf_1 <- randomForest(Age_Group ~ Income + Purchase_Meat_past2years + Purchase_Wines_past2years
                    + Catalog_Purchase + Store_Purchase + Web_Purchase + Web_visits_LastMonth + Tota
                    data = df_train, ntree = num_trees,
                    importance = TRUE, proximity = TRUE)

pred_rf_1 <- predict(Model_rf_1, df_test) %>% round(.,digits = 0)
pred_rf_1 <- round(pred_rf_1,digits = -1)
plot(Model_rf_1)
```




```
varImpPlot(Model_rf_1)
```

Model_rf_1

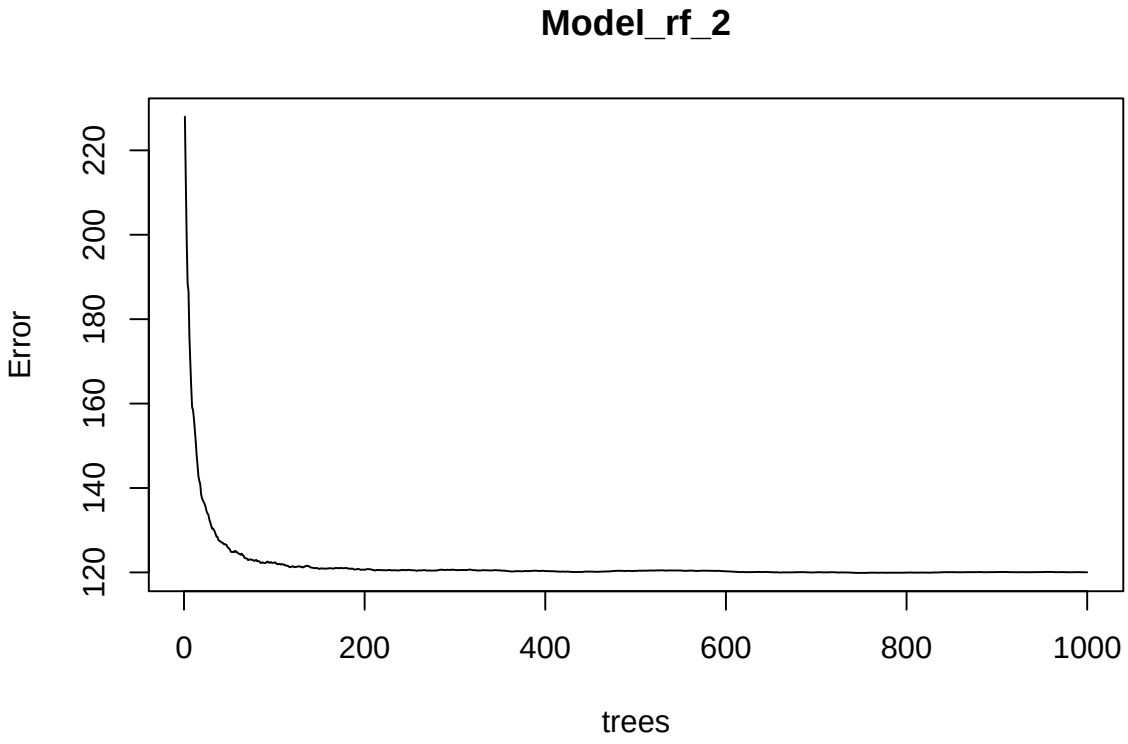



```
confusionMatrix(as.factor(pred_rf_1),as.factor(df_test$Age_Group))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction 20 30 40 50 60 70 120
##        20   1  0  0  0  0  0   0
##        30   3  6  8  0  0  0   0
##        40   7 49 75 15 24  4   0
##        50   4 16 74 77 63  7   1
##        60   0  0  1  1  8  4   0
##        70   0  0  0  0  0  0   0
##        120  0  0  0  0  0  0   0
##
## Overall Statistics
##
##                Accuracy : 0.3728
##                  95% CI : (0.3278, 0.4194)
##     No Information Rate : 0.3527
```

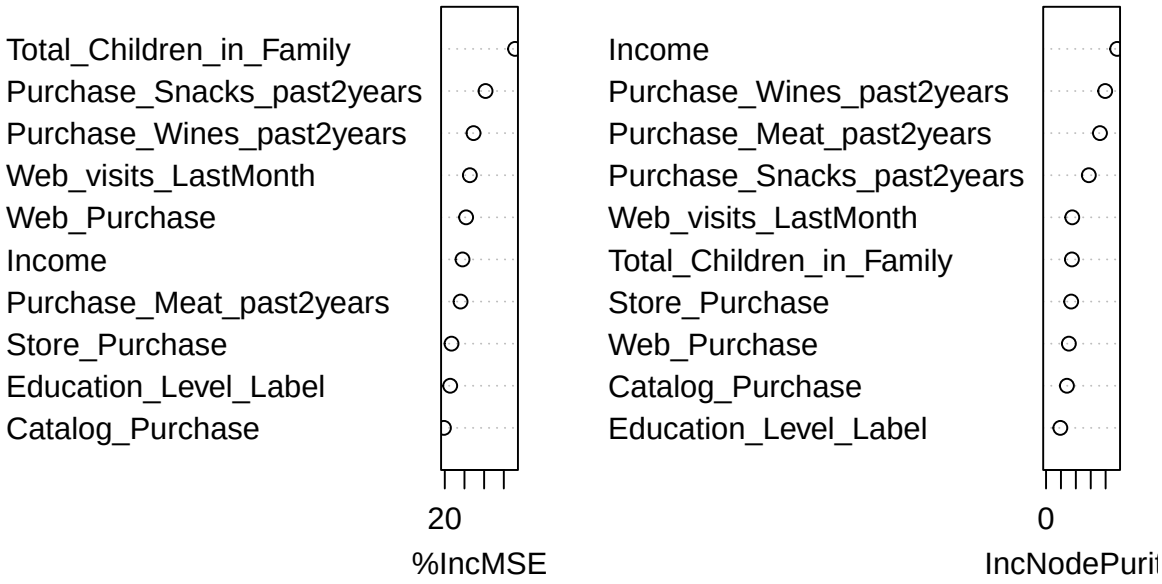
```
##     P-Value [Acc > NIR] : 0.1999
##
##                   Kappa : 0.1503
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 20 Class: 30 Class: 40 Class: 50 Class: 60
## Sensitivity           0.066667   0.08451    0.4747    0.8280   0.08421
## Specificity           1.000000   0.97082    0.6586    0.5352   0.98300
## Pos Pred Value        1.000000   0.35294    0.4310    0.3182   0.57143
## Neg Pred Value        0.968680   0.84919    0.6971    0.9223   0.79954
## Prevalence            0.033482   0.15848    0.3527    0.2076   0.21205
## Detection Rate        0.002232   0.01339    0.1674    0.1719   0.01786
## Detection Prevalence  0.002232   0.03795    0.3884    0.5402   0.03125
## Balanced Accuracy     0.533333   0.52766    0.5667    0.6816   0.53361
##                      Class: 70 Class: 120
## Sensitivity            0.00000   0.000000
## Specificity            1.00000   1.000000
## Pos Pred Value             NaN        NaN
## Neg Pred Value         0.96652   0.997768
## Prevalence             0.03348   0.002232
## Detection Rate         0.00000   0.000000
## Detection Prevalence   0.00000   0.000000
## Balanced Accuracy      0.50000   0.500000
```

```r
# Model 2
set.seed(4000)
Model_rf_2 <- randomForest(Age_Group ~ Income + Purchase_Meat_past2years + Purchase_Wines_past2years
                    + Catalog_Purchase + Store_Purchase + Web_Purchase + Web_visits_LastMonth + Tota
                    + Education_Level_Label,
                    data = df_train, ntree = num_trees,
                    importance = TRUE, proximity = TRUE)

pred_rf_2 <- predict(Model_rf_2, df_test) %>% round(.,digits = 0)
pred_rf_2 <- round(pred_rf_2,digits = -1)
plot(Model_rf_2)
```




```
varImpPlot(Model_rf_2)
```

Model_rf_2

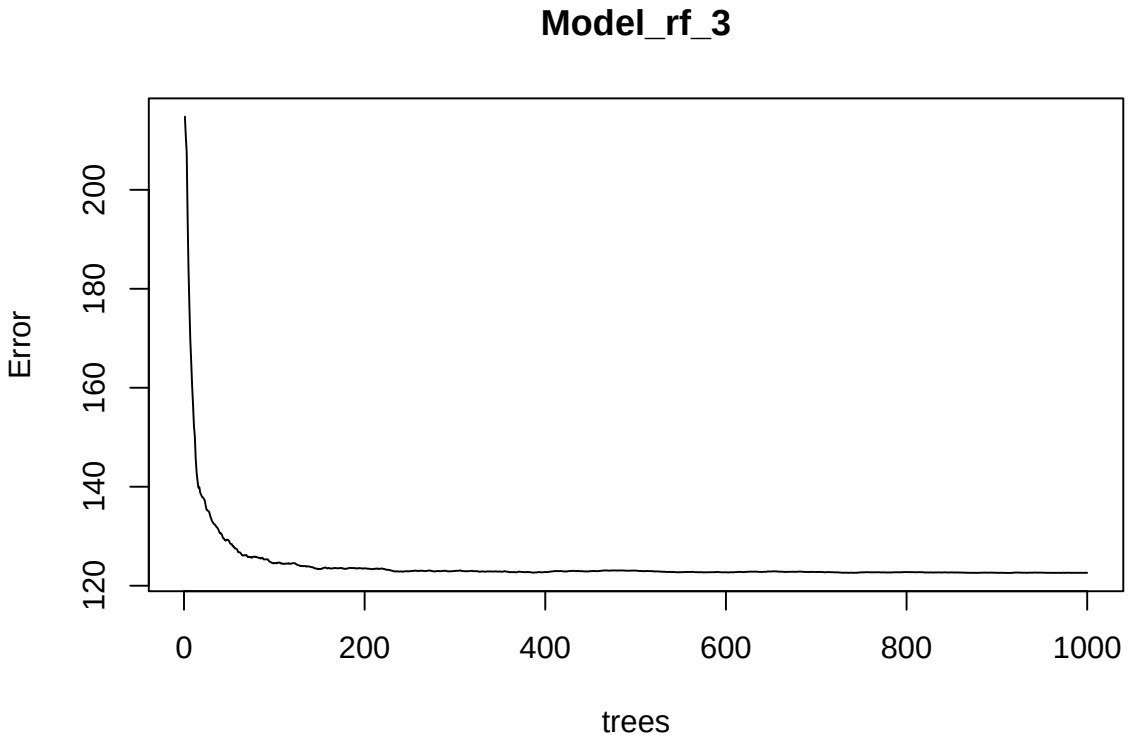



```
confusionMatrix(as.factor(pred_rf_2),as.factor(df_test$Age_Group))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction 20 30 40 50 60 70 120
##        20   1  0  0  0  0  0   0
##        30   3  8  5  0  0  0   0
##        40   8 48 78 17 23  5   0
##        50   3 15 74 75 66  6   1
##        60   0  0  1  1  6  4   0
##        70   0  0  0  0  0  0   0
##        120  0  0  0  0  0  0   0
##
## Overall Statistics
##
##                Accuracy : 0.375
##                  95% CI : (0.33, 0.4217)
##     No Information Rate : 0.3527
```

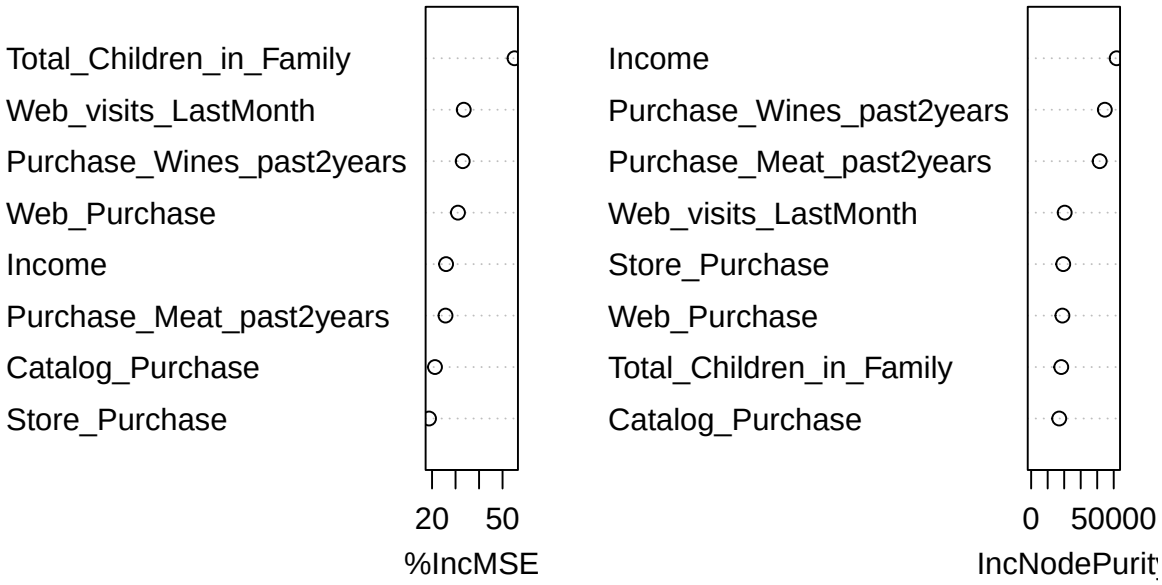
```
##     P-Value [Acc > NIR] : 0.1736
##
##                   Kappa : 0.1513
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 20 Class: 30 Class: 40 Class: 50 Class: 60
## Sensitivity           0.066667   0.11268    0.4937    0.8065   0.06316
## Specificity           1.000000   0.97878    0.6517    0.5352   0.98300
## Pos Pred Value        1.000000   0.50000    0.4358    0.3125   0.50000
## Neg Pred Value        0.968680   0.85417    0.7026    0.9135   0.79587
## Prevalence            0.033482   0.15848    0.3527    0.2076   0.21205
## Detection Rate        0.002232   0.01786    0.1741    0.1674   0.01339
## Detection Prevalence  0.002232   0.03571    0.3996    0.5357   0.02679
## Balanced Accuracy     0.533333   0.54573    0.5727    0.6708   0.52308
##                      Class: 70 Class: 120
## Sensitivity            0.00000   0.000000
## Specificity            1.00000   1.000000
## Pos Pred Value             NaN        NaN
## Neg Pred Value         0.96652   0.997768
## Prevalence             0.03348   0.002232
## Detection Rate         0.00000   0.000000
## Detection Prevalence   0.00000   0.000000
## Balanced Accuracy      0.50000   0.500000
```

```r
# Model 3
set.seed(5000)
Model_rf_3 <- randomForest(Age_Group ~ Income + Purchase_Meat_past2years + Purchase_Wines_past2years
                           + Catalog_Purchase + Store_Purchase + Web_Purchase + Web_visits_LastMonth
                           data = df_train, ntree = num_trees,
                           importance = TRUE, proximity = TRUE)

pred_rf_3 <- predict(Model_rf_3, df_test) %>% round(.,digits = 0)
pred_rf_3 <- round(pred_rf_3,digits = -1)
plot(Model_rf_3)
```




```
varImpPlot(Model_rf_3)
```

Model_rf_3




```
confusionMatrix(as.factor(pred_rf_3),as.factor(df_test$Age_Group))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction 20 30 40 50 60 70 120
##        20   0  0  0  0  0  0   0
##        30   4  7  6  0  0  0   0
##        40   7 47 75 14 17  6   0
##        50   4 16 76 78 77  5   1
##        60   0  1  1  1  1  4   0
##        70   0  0  0  0  0  0   0
##        120  0  0  0  0  0  0   0
##
## Overall Statistics
##
##                Accuracy : 0.3594
##                  95% CI : (0.3149, 0.4057)
##     No Information Rate : 0.3527
```

```
##     P-Value [Acc > NIR] : 0.4006
##
##                   Kappa : 0.1348
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 20 Class: 30 Class: 40 Class: 50 Class: 60
## Sensitivity            0.00000   0.09859    0.4747    0.8387  0.010526
## Specificity            1.00000   0.97347    0.6862    0.4958  0.980170
## Pos Pred Value             NaN   0.41176    0.4518    0.3035  0.125000
## Neg Pred Value         0.96652   0.85151    0.7057    0.9215  0.786364
## Prevalence             0.03348   0.15848    0.3527    0.2076  0.212054
## Detection Rate         0.00000   0.01562    0.1674    0.1741  0.002232
## Detection Prevalence   0.00000   0.03795    0.3705    0.5737  0.017857
## Balanced Accuracy      0.50000   0.53603    0.5804    0.6672  0.495348
##                      Class: 70 Class: 120
## Sensitivity            0.00000   0.000000
## Specificity            1.00000   1.000000
## Pos Pred Value             NaN        NaN
## Neg Pred Value         0.96652   0.997768
## Prevalence             0.03348   0.002232
## Detection Rate         0.00000   0.000000
## Detection Prevalence   0.00000   0.000000
## Balanced Accuracy      0.50000   0.500000
```

Conclusion

The accuracy did not improve as much as expected, leaving quite a lot of room for improvement. In the future, we intend to apply and develop these models. However, since the model construction itself is not so difficult, we reaffirmed the importance of having options to improve the accuracy, feature engineering, and data wrangling.

References

1.https://aismiley.co.jp/ai_news/random-forests

2.https://datachemeng.com/randomforest

3.https://www.nri.com/jp/knowledge/glossary/lst/alphabet/light_gbm

4.https://rpubs.com/hide/444023