Data622_HWk2

Alexis Mekueko

3/30/2022

Github Link Web Link

Assignment:

Based on the latest topics presented, bring a dataset of your choice and create a Decision Tree where you can solve a classification or regression problem and predict the outcome of a particular feature or detail of the data used. Switch variables to generate 2 decision trees and compare the results. Create a random forest for regression and analyze the results. Based on real cases where desicion trees went wrong, and 'the bad & ugly' aspects of decision trees (https://decizone.com/blog/the-good-the-bad-the-ugly-of-using-decision-trees), how can you change this perception when using the decision tree you created to solve a real problem? Format: document with screen captures & analysis.

Import Data and Data Structure

We imported the data from local drive. Another option could be to load the date from Github.

```
## 'data.frame':    614 obs. of  13 variables:
##  $ Loan_ID          : chr  "LP001002" "LP001003" "LP001005" "LP001006" ...
##  $ Gender           : chr  "Male" "Male" "Male" "Male" ...
##  $ Married          : chr  "No" "Yes" "Yes" "Yes" ...
##  $ Dependents       : chr  "0" "1" "0" "0" ...
##  $ Education        : chr  "Graduate" "Graduate" "Graduate" "Not Graduate" ...
##  $ Self_Employed    : chr  "No" "No" "Yes" "No" ...
##  $ ApplicantIncome  : int  5849 4583 3000 2583 6000 5417 2333 3036 4006 12841 ...
##  $ CoapplicantIncome: num  0 1508 0 2358 0 ...
##  $ LoanAmount       : int  NA 128 66 120 141 267 95 158 168 349 ...
##  $ Loan_Amount_Term : int  360 360 360 360 360 360 360 360 360 360 ...
##  $ Credit_History   : int  1 1 1 1 1 1 1 0 1 1 ...
##  $ Property_Area    : chr  "Urban" "Rural" "Urban" "Urban" ...
##  $ Loan_Status      : chr  "Y" "N" "Y" "Y" ...
```

Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
LP001002	Male	No	0	Graduate	No	5849	0	NA	360	1	Urban	Y
LP001003	Male	Yes	1	Graduate	No	4583	1508	128	360	1	Rural	N
LP001005	Male	Yes	0	Graduate	Yes	3000	0	66	360	1	Urban	Y
LP001006	Male	Yes	0	Not Graduate	No	2583	2358	120	360	1	Urban	Y

Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
LP001008	Male	No	0	Graduate	No	6000	0	141	360	1	Urban	Y
LP001011	Male	Yes	2	Graduate	Yes	5417	4196	267	360	1	Urban	Y
LP001013	Male	Yes	0	Not Graduate	No	2333	1516	95	360	1	Urban	Y
LP001014	Male	Yes	3+	Graduate	No	3036	2504	158	360	0	Semiurban	N

Dataset Description Variables ========== Descriptions

Loan_ID ========== Unique Loan ID

Gender =========== Male/Female

Married =========== Appliquant marital status (Y/N)

Dependents ========= Number of dependents

Education ========== Applicant Education (Graduate/Undergraduate)

Self_Employed ====== Self_employed (Y/N)

ApplicantIncome ==== Applicant income

CoapplicantIncome == Coapplicant income

LoanAmount ========= Loan amount in thousands dollars

Loan_Amount_Term === Term of loan in months

Credit_History ===== Credit history meets guidelines

Property_Area ====== Urban, semi-urban, rural

Loan_Status ======== Loan approved (Y/N)

This dataset is a typical format which banks use to screen/select applicant for a loan. There 614 records with 13 variables. The datatypes in this dataset are mostly character and numerical. There are some variables (Loan_Status,Self_Employed, Married,Dependents etc) with characters datatype that should be factor with two levels (yes/no or 0/1). The variable "Credit_History" should be in term of number of years. We assume the bank uses '1' to say the customer meets the minimum number of years to qualify for a loan and '0' for those who don't meet the minimum years. Normally, a customer with a credit history = 0 should be denied a loan. Is it true on this bank record? Answer is no. Therefore the decision to approve a loan for a customer relies on the combination with other variables other than the dependent/target 'Loan_Status'. Based on the information about the structure of the dataset, we can conclude that we have a labeled data. Therefore, we can be confident in using supervised learning on this dataset. As we know, supervised learning model account for a classification model and we will predict the state of client loan approval.

Cleaning Data

```
#install.packages('Amelia')
#install.packages('DataExplorer')

library(Amelia)
```

```
## Loading required package: Rcpp
```

```
## ##
## ## Amelia II: Multiple Imputation
## ## (Version 1.8.0, built: 2021-05-26)
## ## Copyright (C) 2005-2022 James Honaker, Gary King and Matthew Blackwell
## ## Refer to http://gking.harvard.edu/amelia/ for more information
## ##
```

```r
#sum(is.na(loanDF))
misValues <- sum(is.na(loanDF))# Returning the column names with missing values

#sum(is.na(basket1a$X.1))
#misValues1 <- sum(is.na()
# Filling the empty spece with "NA"
#us_d <- dplyr::na_if(us_d, "")
#is.null(us_d)
#if (is.na(us_d)|| us_d== '')
#is.empty(" ")
#apply(myData, 2, function(myCol){  sum(myCol == "1") > 0

emptyValue <- sum(emptyValue <- sapply(loanDF, function(x) all(is.na(x) | x == '' )) )

cat("The dataset contains missing values for a total record of : " , misValues)
```

```
## The dataset contains missing values for a total record of :  86
```

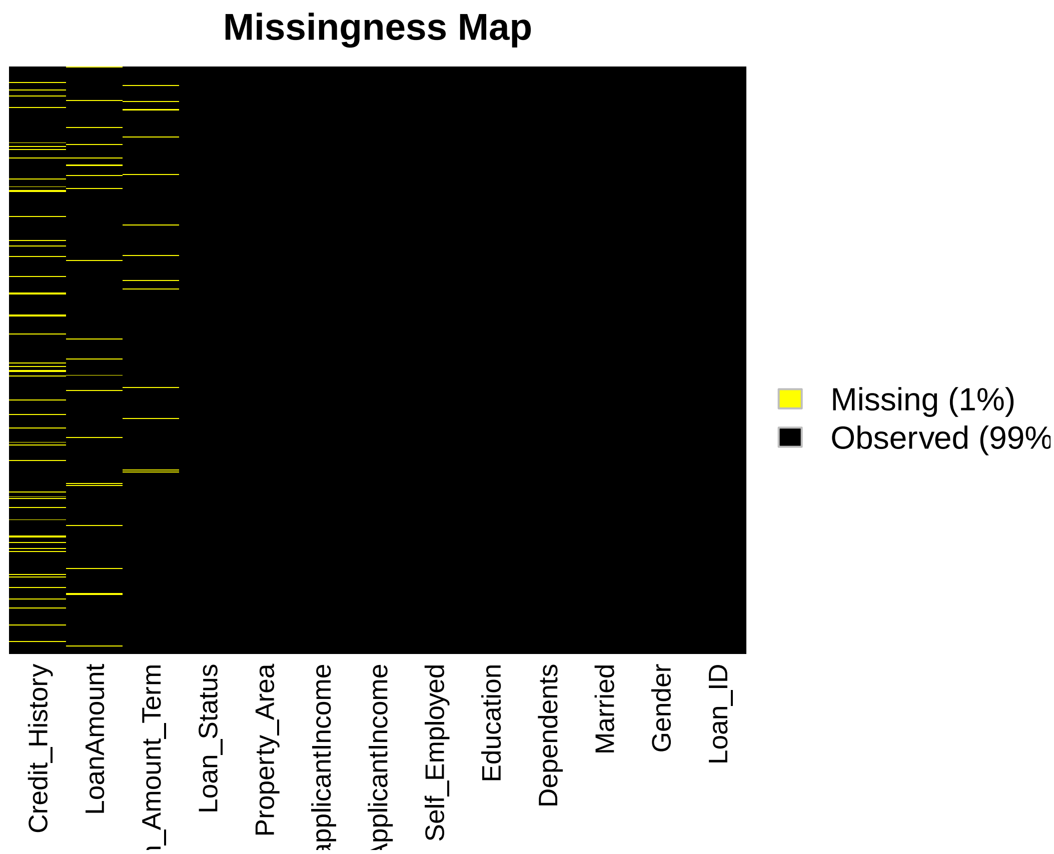
```r
print("\n")
```

```
## [1] "\n"
```

```r
cat("The dataset contains empty values for a total record of : " , emptyValue)
```

```
## The dataset contains empty values for a total record of :  0
```

```r
missmap(loanDF,col=c('yellow','black'),y.at=1,y.labels=' ',legend=TRUE)
```




```
#count(loanDF$Credit_History)
```

The plot of missing values shows that there are definitely missing values(86 records) withing the dataset. Let's take a look at this missing values.

```
library(VIM)
```

```
## Loading required package: colorspace
```

```
##
## Attaching package: 'colorspace'
```

```
## The following object is masked from 'package:pROC':
##
##     coords
```

```
## Loading required package: grid
```

```
## VIM is ready to use.
```

```
## Suggestions and bug-reports can be submitted at: https://github.com/statistikat/VIM/issues
```

```
##
## Attaching package: 'VIM'
```

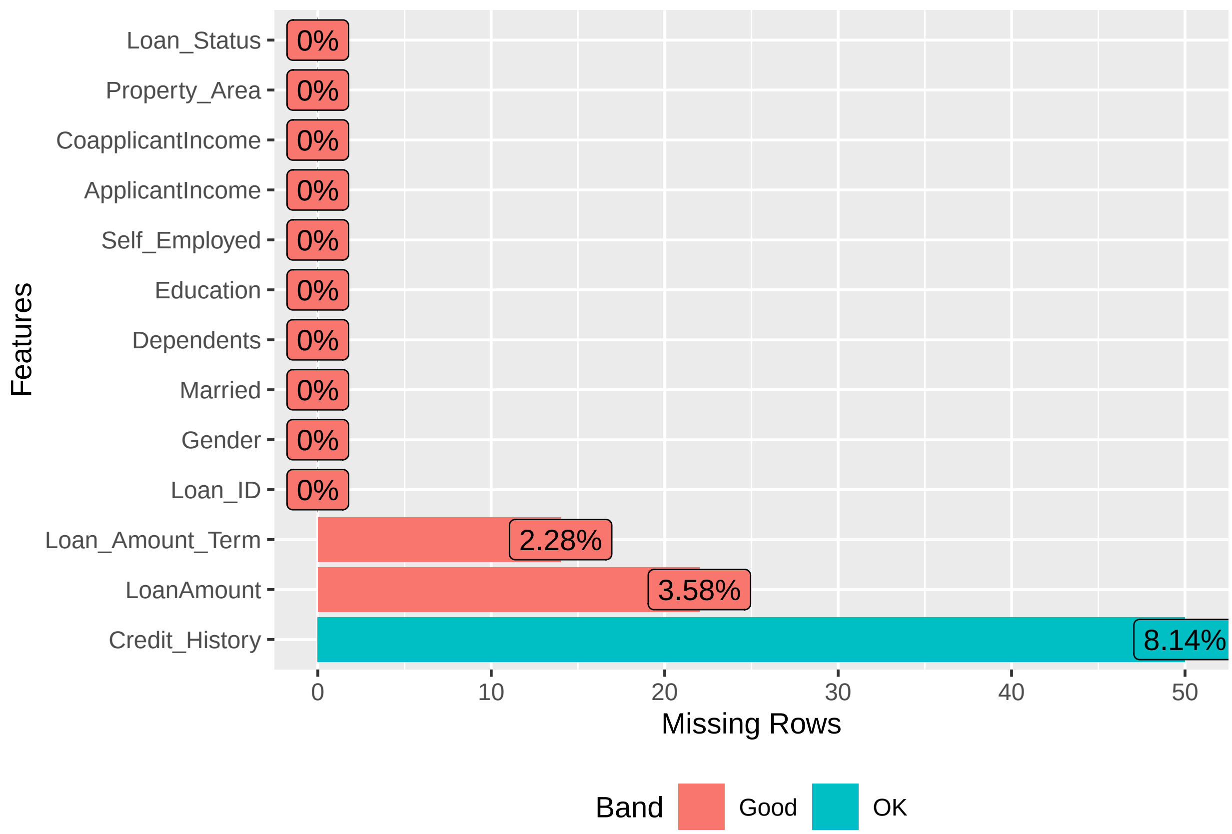
```
## The following object is masked from 'package:datasets':
##
##     sleep
```

```r
#aggr(loanDF)
#vis_miss(loanDF)



missing.values <- function(df){
    df %>%
    gather(key = "variables", value = "val") %>%
    mutate(is.missing = is.na(val)) %>%
    group_by(variables, is.missing) %>%
    dplyr::summarise(number.missing = n()) %>%
    filter(is.missing==T) %>%
    dplyr::select(-is.missing) %>%
    arrange(desc(number.missing))
}

missing.values(loanDF)%>%
  kable()
```

```
## `summarise()` has grouped output by 'variables'. You can override using the
## `.groups` argument.
```

variables	number.missing
Credit_History	50
LoanAmount	22
Loan_Amount_Term	14

```r
library(DataExplorer)
plot_missing(loanDF)
```




```
#gg_miss_upset(loanDF)

# dev.off()
# print(plot(1))


#count((data1000R$Order.Priority))

#sum(is.na(data1000R$Order.Priority))
# Not sure why the code below does not work
# data1000R %>%
#   group_by(data1000R$Order.Priority) %>%
#   summarize(Count=n()) %>%
#   mutate(Percent = (Count/sum(Count))*100) %>%
#   arrange(desc(Count))
```

The missing values are present in these variables (Loan_Amount_Term, LoanAmount and Credit_History). Since the dataset is a small in size, deleting these missing values will reduce the dataset. Instead of deleting, we can apply imputation on these missing values.

```
#if (is.na(loanDF$Self_Employed) || loanDF$Self_Employed == '')
count(loanDF$Gender)
```

```
##        x freq
## 1          13
```

```
## 2 Female  112
## 3   Male  489
```

```
count(loanDF$Married)
```

```
##     x freq
## 1        3
## 2  No  213
## 3 Yes  398
```

```
count(loanDF$Self_Employed)
```

```
##     x freq
## 1       32
## 2  No  500
## 3 Yes   82
```

```
count(loanDF$Credit_History)
```

```
##    x freq
## 1  0   89
## 2  1  475
## 3 NA   50
```

```
print("The above frequency distribution shows that there are 04 variable with some blank/empty values")
```

```
## [1] "The above frequency distribution shows that there are 04 variable with some blank/empty values"
```

```
#loanDF$Gender[loanDF$Gender==""]<-NA
#loanDF[loanDF==""]<- c('NA')

# Works but does not fix the issue with blanks value
#loanDF <- loanDF %>%
#                  mutate_all(na_if,"")

# Works but does not fix the issue with blank value
## define a empty function
# empty_as_na <- function(x){
#     if("factor" %in% class(x)) x <- as.character(x) ## since ifelse won't work with factors
#     ifelse(as.character(x)!="", x, NA) <NA>
# }

# Works but the issue with blank value is still present
## transform all columns
#loanDF %>%
#  mutate_each(funs(empty_as_na))

#loanDF[loanDF=="NA"]<- c('<NA>')
#loanDF <- loanDF %>%
#   mutate(across(everything(), ~ifelse(.=="", NA, as.character(.))))


print("\n")
```

```
## [1] "\n"
```

```r
print("Let's see if sum of missing values will cath these blank values since we applied a function earli
```

```
## [1] "Let’s see if sum of missing values will cath these blank values since we applied a function earl
```

```r
print("\n")
```

```
## [1] "\n"
```

```r
cat("Sum of missing values within variable = Credit_History is:  ", sum(is.na(loanDF$Credit_History)))
```

```
## Sum of missing values within variable = Credit_History is:   50
```

```r
print("\n")
```

```
## [1] "\n"
```

```r
cat("Sum of missing values within variable = Gender is:  ", sum(is.na(loanDF$Gender)))
```

```
## Sum of missing values within variable = Gender is:   0
```

```r
print("\n")
```

```
## [1] "\n"
```

```r
cat("Sum of missing values within variable = Self_Employed is:  ", sum(is.na(loanDF$Self_Employed)))
```

```
## Sum of missing values within variable = Self_Employed is:   0
```

```r
print("\n")
```

```
## [1] "\n"
```

```r
cat("Sum of missing values within variable = Married is:  ", sum(is.na(loanDF$Married)))
```

```
## Sum of missing values within variable = Married is:   0
```

```r
print("\n")
```

```
## [1] "\n"
```

```r
#View(loanDF)
```

Somehow there are some empty values. These aren’t easy to check because the mapping of missing values above missed them. We will fill in the empty/blank values with ‘NA’. Then, check again before performing imputation.

```
#loanDF <- read.csv("Loan.csv", header=T, na.strings=c("",'NA'))

#loanDF$Gender[loanDF$Gender == " "]<- NA
# loanDF$Gender[loanDF$Gender == ""  | loanDF$Gender== " "] <- NA
# loanDF$Dependents[loanDF$Dependents == ""  | loanDF$Dependents== " "] <- NA
# loanDF$Self_Employed[loanDF$Self_Employed == ""  | loanDF$Self_Employed== " "] <- NA
# loanDF$Married[loanDF$Married == ""  | loanDF$Married== " "] <- NA

#loanDF$Self_Employed[is.na(loanDF$Self_Employed)] <- mean(loanDF$Self_Employed, na.rm = TRUE)

#if (!require("tidyverse")) install.packages("tidyverse")

# loanDF %>%
#   mutate(Gender = if_else(is.na(Gender),
#                           calc_mode(Gender),
#                           Gender))
#
# calc_mode <- function(x){
#
#   # List the distinct / unique values
#   distinct_values <- unique(x)
#
#   # Count the occurrence of each distinct value
#   distinct_tabulate <- tabulate(match(x, distinct_values))
#
#   # Return the value with the highest occurrence
#   distinct_values[which.max(distinct_tabulate)]
# }
#
#
# loanDF %>%
#   mutate(across(everything(), ~replace_na(.x, calc_mode(.x))))

#
# getmode <- function(v){
#   v=v[nchar(as.character(v))>0]
#   uniqv <- unique(v)
#   uniqv[which.max(tabulate(match(v, uniqv)))]
# }
#
# for (cols in colnames(df)) {
#   if (cols %in% names(df[,sapply(df, is.numeric)])) {
#     df<-df%>%mutate(!!cols := replace(!!rlang::sym(cols), is.na(!!rlang::sym(cols)), mean(!!rlang::sym
#
#   }
#   else {
#
#     df<-df%>%mutate(!!cols := replace(!!rlang::sym(cols), !!rlang::sym(cols)=="", getmode(!!rlang::sym
#
#   }
# }
#
# df
```

```r
# The above attempts work but somehow the issue is still persisting. This time , we are going to try pr

loanDF$Married <- loanDF$Married %>% replace_na("NA")

loanDF$Gender <- loanDF$Gender %>% replace_na("NA")

loanDF$Dependents <- loanDF$Dependents %>% replace_na("NA")

loanDF$Self_Employed <- loanDF$Self_Employed %>% replace_na("NA")


count(loanDF$Gender)
```

```
##        x freq
## 1          13
## 2 Female  112
## 3   Male  489
```

```r
count(loanDF$Self_Employed)
```

```
##     x freq
## 1        32
## 2  No  500
## 3 Yes   82
```

```r
count(loanDF$Credit_History)
```

```
##    x freq
## 1  0   89
## 2  1  475
## 3 NA   50
```

```r
count(loanDF$Married)
```

```
##     x freq
## 1         3
## 2  No  213
## 3 Yes  398
```

```r
print("\n")
```

```
## [1] "\n"
```

```r
cat("Sum of missing values within variable = Credit_History is:  ", sum(is.na(loanDF$Credit_History)))
```

```
## Sum of missing values within variable = Credit_History is:   50
```

```r
print("\n")
```

```
## [1] "\n"
```

```r
cat("Sum of missing values within variable = Gender is:  ", sum(is.na(loanDF$Gender)))
```

```
## Sum of missing values within variable = Gender is:   0
```

```r
print("\n")
```

```
## [1] "\n"
```

```r
cat("Sum of missing values within variable = Self_Employed is:  ", sum(is.na(loanDF$Self_Employed)))
```

```
## Sum of missing values within variable = Self_Employed is:   0
```

```r
print("\n")
```

```
## [1] "\n"
```

```r
cat("Sum of missing values within variable = Married is:  ", sum(is.na(loanDF$Married)))
```

```
## Sum of missing values within variable = Married is:   0
```

```r
print("\n")
```

```
## [1] "\n"
```

```r
#View(loanDF)
```

let's perform imputation.

```r
#df[!(is.na(df$start_pc) | df$start_pc==""), ]
#df <- with(df, df[!(start_pc == "" | is.na(start_pc)), ])
#test for non-zero string length using nzchar.
#df <- with(df, df[!(nzchar(start_pc) | is.na(start_pc)), ])

#loanDF1 <- loanDF1[-which(loanDF1$Gender == ""), ]


library(mice)
```

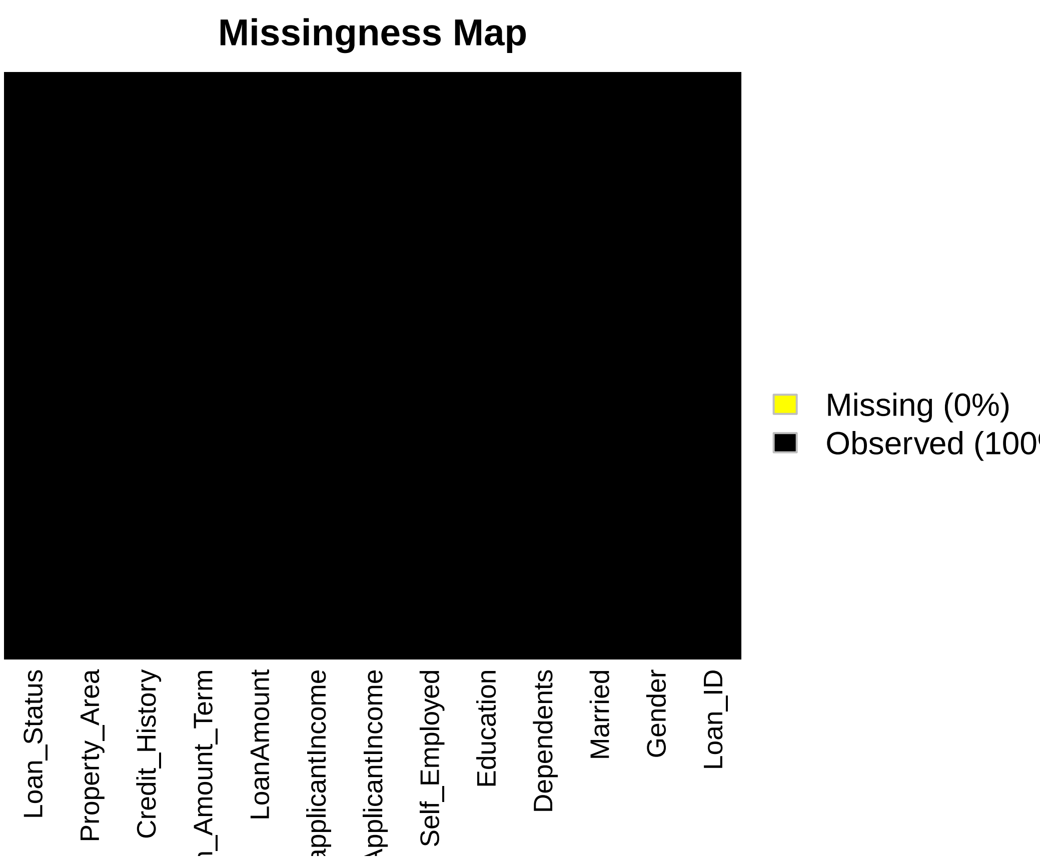
```
##
## Attaching package: 'mice'

## The following object is masked from 'package:stats':
##
##     filter
```

```
## The following objects are masked from 'package:base':
##
##     cbind, rbind
```

```r
imputed <- mice(loanDF, m=2, maxit = 2, method = 'cart', seed = 23321)
```

```
##
##  iter imp variable
##   1   1  LoanAmount  Loan_Amount_Term  Credit_History
##   1   2  LoanAmount  Loan_Amount_Term  Credit_History
##   2   1  LoanAmount  Loan_Amount_Term  Credit_History
##   2   2  LoanAmount  Loan_Amount_Term  Credit_History

## Warning: Number of logged events: 8
```

```r
#mice = multiple imputation by chained equations. The 'm' argument = number of rounds of imputation
#CART = classification and regression trees
loanDF1<- complete(imputed,2) #here I chose the second round of data imputation

missmap(loanDF1,col=c('yellow','black'),y.at=1,y.labels=' ',legend=TRUE)
```




```r
str(loanDF1)
```

```
## 'data.frame':    614 obs. of  13 variables:
```

```
##  $ Loan_ID          : chr  "LP001002" "LP001003" "LP001005" "LP001006" ...
##  $ Gender           : chr  "Male" "Male" "Male" "Male" ...
##  $ Married          : chr  "No" "Yes" "Yes" "Yes" ...
##  $ Dependents       : chr  "0" "1" "0" "0" ...
##  $ Education        : chr  "Graduate" "Graduate" "Graduate" "Not Graduate" ...
##  $ Self_Employed    : chr  "No" "No" "Yes" "No" ...
##  $ ApplicantIncome  : int  5849 4583 3000 2583 6000 5417 2333 3036 4006 12841 ...
##  $ CoapplicantIncome: num  0 1508 0 2358 0 ...
##  $ LoanAmount       : num  128 128 66 120 141 267 95 158 168 349 ...
##  $ Loan_Amount_Term : num  360 360 360 360 360 360 360 360 360 360 ...
##  $ Credit_History   : num  1 1 1 1 1 1 1 0 1 1 ...
##  $ Property_Area    : chr  "Urban" "Rural" "Urban" "Urban" ...
##  $ Loan_Status      : chr  "Y" "N" "Y" "Y" ...
```

```
#library(stringi)
#stri_isempty(loanDF1$Self_Employed)


# loanDF1$Married <- loanDF1$Married %>% replace_na("NA")
#
# loanDF$Gender <- loanDF$Gender %>% replace_na("NA")
#
# loanDF$Dependents <- loanDF$Dependents %>% replace_na("NA")
#
# loanDF$Self_Employed <- loanDF$Self_Employed %>% replace_na("NA")

#is.null(loanDF1$Gender)
# Checking for empty value again
count(loanDF1$Gender)
```

```
##        x freq
## 1          13
## 2 Female  112
## 3   Male  489
```

```
count(loanDF1$Married)
```

```
##     x freq
## 1        3
## 2  No  213
## 3 Yes  398
```

We clearly see that there is no more missing data. But there are persisting issue with blank values.

Processing Data

Let's remove the variables that we don't need for the decision trees model. Then, we will reformat the dataset into a new data frame in which some variables (Married,Dependents,Self_Employed,Credit_History and Loan_Status).

```r
loanDF1$Loan_ID <- NULL
str(loanDF1)
```

```
## 'data.frame':    614 obs. of  12 variables:
##  $ Gender           : chr  "Male" "Male" "Male" "Male" ...
##  $ Married          : chr  "No" "Yes" "Yes" "Yes" ...
##  $ Dependents       : chr  "0" "1" "0" "0" ...
##  $ Education        : chr  "Graduate" "Graduate" "Graduate" "Not Graduate" ...
##  $ Self_Employed    : chr  "No" "No" "Yes" "No" ...
##  $ ApplicantIncome  : int  5849 4583 3000 2583 6000 5417 2333 3036 4006 12841 ...
##  $ CoapplicantIncome: num  0 1508 0 2358 0 ...
##  $ LoanAmount       : num  128 128 66 120 141 267 95 158 168 349 ...
##  $ Loan_Amount_Term : num  360 360 360 360 360 360 360 360 360 360 ...
##  $ Credit_History   : num  1 1 1 1 1 1 1 0 1 1 ...
##  $ Property_Area    : chr  "Urban" "Rural" "Urban" "Urban" ...
##  $ Loan_Status      : chr  "Y" "N" "Y" "Y" ...
```

Summary and Correlation

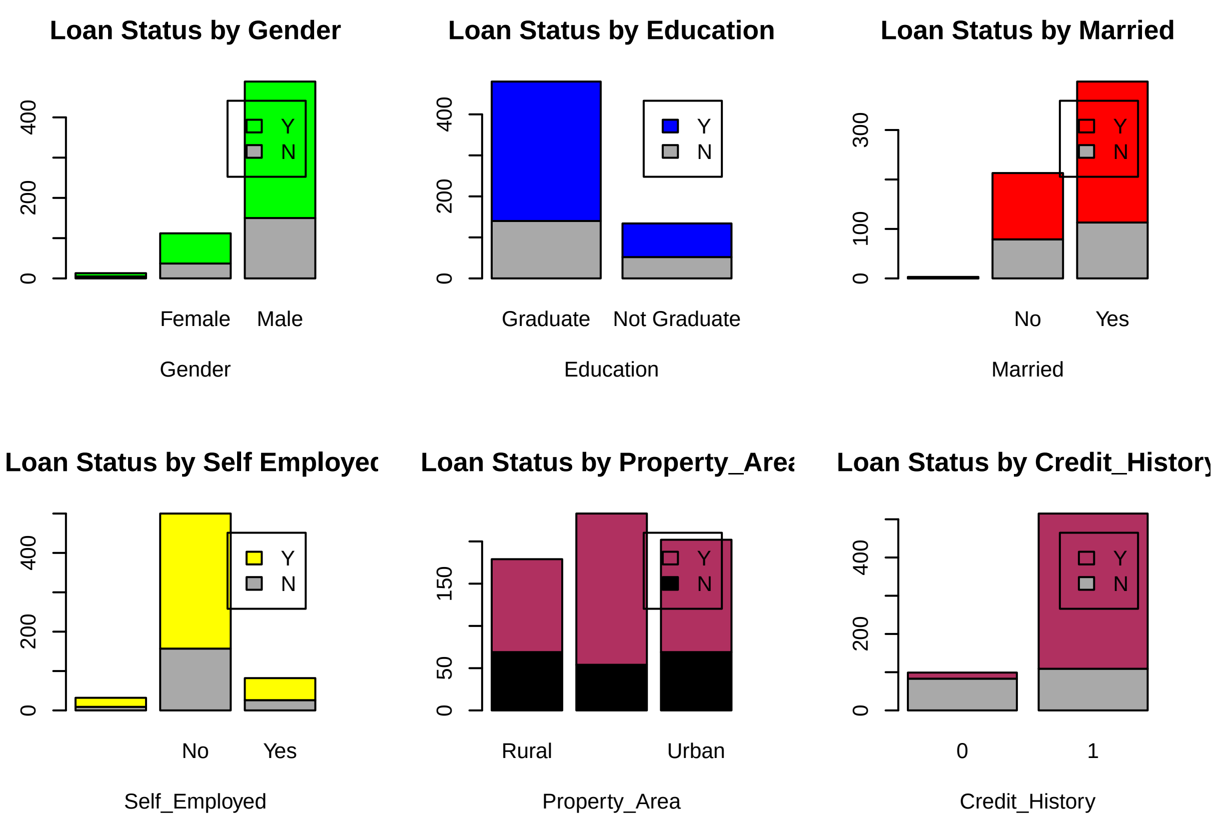
This is a summary and correlation of the popular item known as "Beverage"

```r
summary(loanDF1)
```

```
##     Gender             Married           Dependents         Education
##  Length:614         Length:614         Length:614         Length:614
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##  Self_Employed      ApplicantIncome CoapplicantIncome   LoanAmount
##  Length:614         Min.   :  150   Min.   :    0     Min.   :  9.0
##  Class :character   1st Qu.: 2878   1st Qu.:    0     1st Qu.:100.0
##  Mode  :character   Median : 3812   Median : 1188     Median :128.0
##                     Mean   : 5403   Mean   : 1621     Mean   :146.8
##                     3rd Qu.: 5795   3rd Qu.: 2297     3rd Qu.:168.0
##                     Max.   :81000   Max.   :41667     Max.   :700.0
##  Loan_Amount_Term Credit_History   Property_Area      Loan_Status
##  Min.   : 12.0    Min.   :0.0000   Length:614         Length:614
##  1st Qu.:360.0    1st Qu.:1.0000   Class :character   Class :character
##  Median :360.0    Median :1.0000   Mode  :character   Mode  :character
##  Mean   :342.3    Mean   :0.8388
##  3rd Qu.:360.0    3rd Qu.:1.0000
##  Max.   :480.0    Max.   :1.0000
```

```r
#library(psych)

#describe(loanDF1$Self_Employed)
par(mfrow=c(2,3))
corr1 <- table(loanDF1$Loan_Status, loanDF1$Gender)
barplot(corr1, main="Loan Status by Gender",
```

```
        xlab="Gender", col=c("darkgrey","green"),
        legend = rownames(corr1))

corr2 <- table(loanDF1$Loan_Status, loanDF1$Education)
barplot(corr2, main="Loan Status by Education",
        xlab="Education", col=c("darkgrey","blue"),
        legend = rownames(corr2))
corr3 <- table(loanDF1$Loan_Status, loanDF1$Married)
barplot(corr3, main="Loan Status by Married",
        xlab="Married", col=c("darkgrey","red"),
        legend = rownames(corr3))
corr4 <- table(loanDF1$Loan_Status, loanDF1$Self_Employed)
barplot(corr4, main="Loan Status by Self Employed",
        xlab="Self_Employed", col=c("darkgrey","yellow"),
        legend = rownames(corr4))
corr5 <- table(loanDF1$Loan_Status, loanDF1$Property_Area)
barplot(corr5, main="Loan Status by Property_Area",
        xlab="Property_Area", col=c("black","maroon"),
        legend = rownames(corr5))
corr6 <- table(loanDF1$Loan_Status, loanDF1$Credit_History)
barplot(corr6, main="Loan Status by Credit_History",
        xlab="Credit_History", col=c("darkgrey","maroon"),
        legend = rownames(corr6))
```



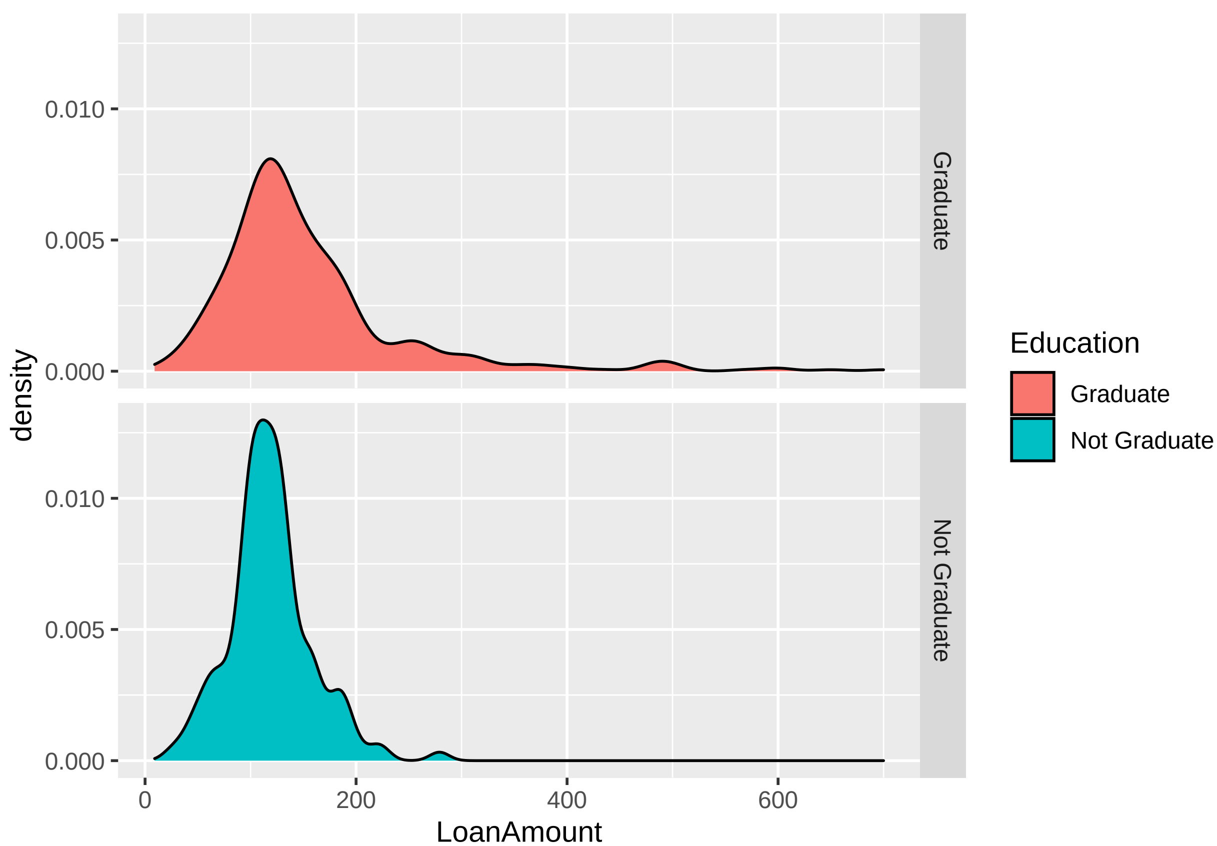
```
#as.numeric(data1000R1$Units.Sold)
#library(Hmisc)
#data1 <- data.frame(data1000R1)
#cor(loanDF1)
#cor(data1000R1[,unlist(lapply(data1000R1, is.numeric))])
#rcorr(as.matrix(data1000R1), type = "Pearson")
```

The assumption that we made early came out to be false. We see that there is a few percentage of customers getting loan approved despite the fact that they did not meet the minimum years of credit history. Therefore, the loan_status decision is based on other variables than credit_history. By curiousity, we also checked loan approval by gender and found out men dominate in applying for a loan. We wonder how would bank interprets this result. Perhaps, the workforce in the area is predominantly men power. Let's see how Married families do versus the non-married. The result is somewhat we would anticipate it right. Married families get more loan approved than non-married. More results shows that the bank trusts more graduate customers than those with no graduate degree. In addition, self-employed customers seem to not getting loan approval. One explanation could be that there are more employed customers than self-employed ones in the area.

These results still show the blanks values.

Let's see Loan approval, applicant income and loan amount distributions

```
## Warning in data(loanDF1, package = "lattice"): data set 'loanDF1' not found
```



We observed right skewed distribution with some outliers. One way to deal with outliers is to delete if there aren't many. This method might have bad effect on the rest of the data since this is a small dataset. Since the imputation by classification and regression trees (cart) does not fix the blank values, we want to try one more method, random forest (rf), then we will tranform character variables into factors.

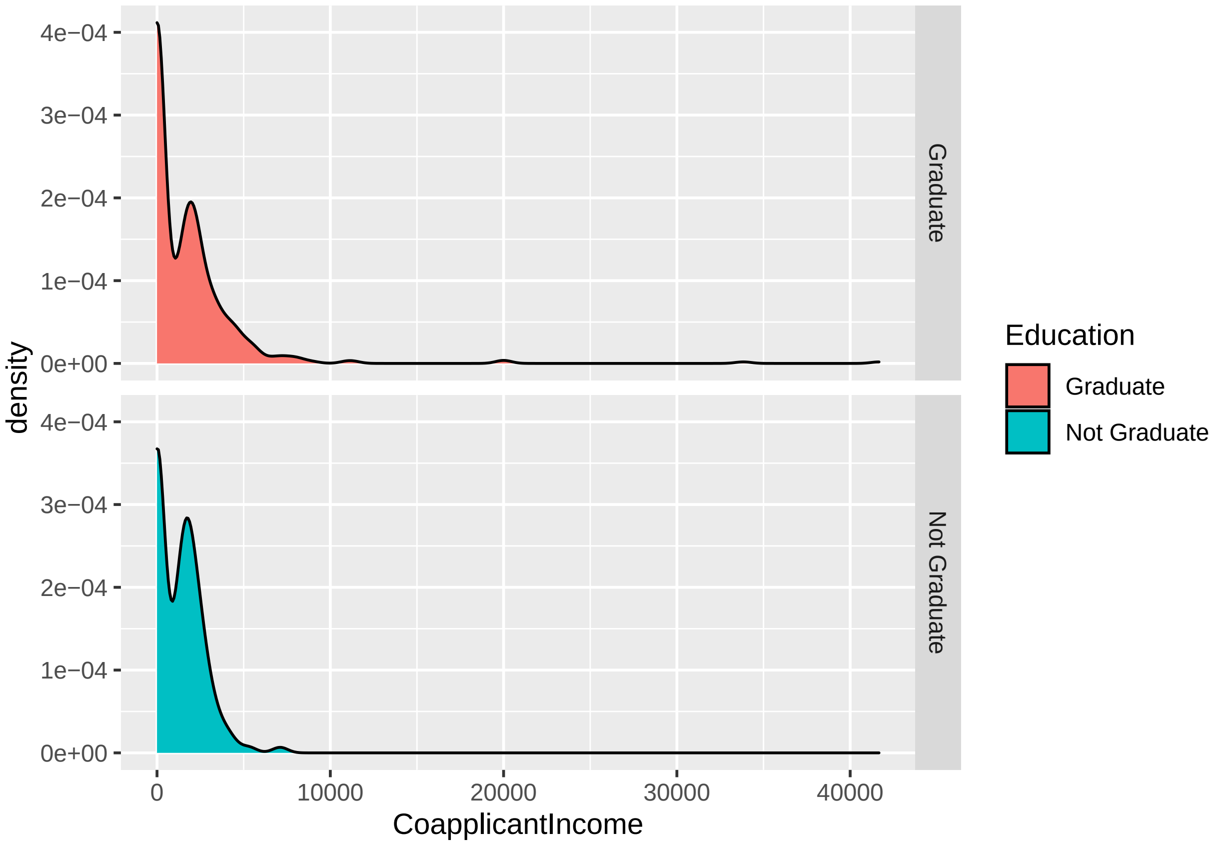
```r
imputed <- mice(loanDF1, maxit = 0)
predicts <- imputed$predictorMatrix

imputed <- mice(loanDF1, method = 'rf', predictorMatrix = predicts, m=2)
```

```
##
##  iter imp variable
##   1   1
##   1   2
##   2   1
##   2   2
##   3   1
##   3   2
##   4   1
##   4   2
##   5   1
##   5   2
```

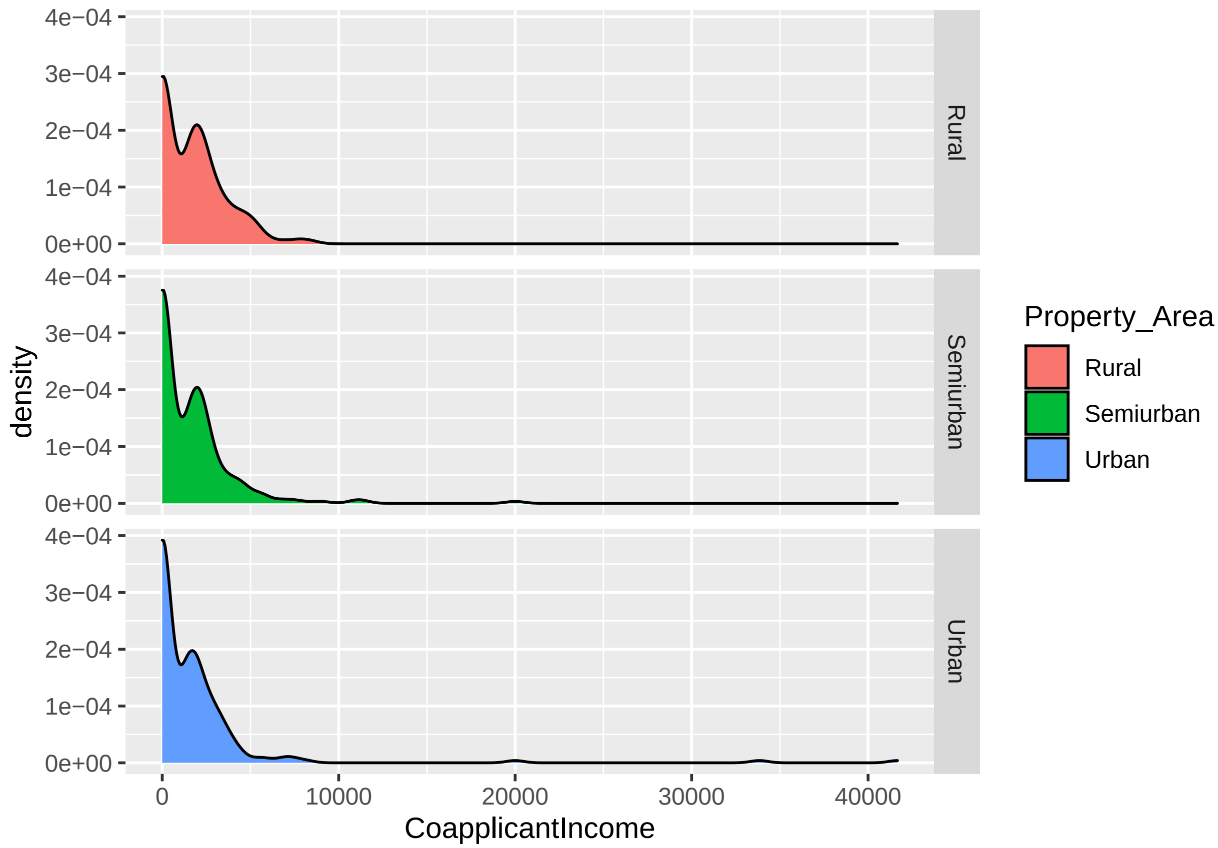
```r
loanDF1 <- complete(imputed)
count(loanDF1$Gender)
```

```
##        x freq
## 1          13
## 2 Female  112
## 3   Male  489
```

```r
data(loanDF1, package="lattice")
ggplot(data=loanDF1, aes(x=CoapplicantIncome, fill=Education)) +
  geom_density() +
  facet_grid(Education~.)
```




```
data(loanDF1, package="lattice")
ggplot(data=loanDF1, aes(x=CoapplicantIncome, fill=Property_Area)) +
  geom_density() +
  facet_grid(Property_Area~.)
```




```r
loanDF1$Gender <- as.factor(loanDF1$Gender)
loanDF1$Married <- as.factor(loanDF1$Married)
loanDF1$Dependents <- as.factor(loanDF1$Dependents)
loanDF1$Education <- as.factor(loanDF1$Education)
loanDF1$Self_Employed <- as.factor(loanDF1$Self_Employed)
loanDF1$Property_Area <- as.factor(loanDF1$Property_Area)
loanDF1$Credit_History <- as.factor(loanDF1$Credit_History)
loanDF1$Loan_Status <- as.factor(loanDF1$Loan_Status)




str(loanDF1)
```

```
## 'data.frame':    614 obs. of  12 variables:
##  $ Gender           : Factor w/ 3 levels "","Female","Male": 3 3 3 3 3 3 3 3 3 3 ...
##  $ Married          : Factor w/ 3 levels "","No","Yes": 2 3 3 3 2 3 3 3 3 3 ...
##  $ Dependents       : Factor w/ 5 levels "","0","1","2",..: 2 3 2 2 2 4 2 5 4 3 ...
##  $ Education        : Factor w/ 2 levels "Graduate","Not Graduate": 1 1 1 2 1 1 2 1 1 1 ...
##  $ Self_Employed    : Factor w/ 3 levels "","No","Yes": 2 2 3 2 2 3 2 2 2 2 ...
##  $ ApplicantIncome  : int  5849 4583 3000 2583 6000 5417 2333 3036 4006 12841 ...
##  $ CoapplicantIncome: num  0 1508 0 2358 0 ...
##  $ LoanAmount       : num  128 128 66 120 141 267 95 158 168 349 ...
##  $ Loan_Amount_Term : num  360 360 360 360 360 360 360 360 360 360 ...
##  $ Credit_History   : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 1 2 2 ...
##  $ Property_Area    : Factor w/ 3 levels "Rural","Semiurban",..: 3 1 3 3 3 3 3 2 3 2 ...
```

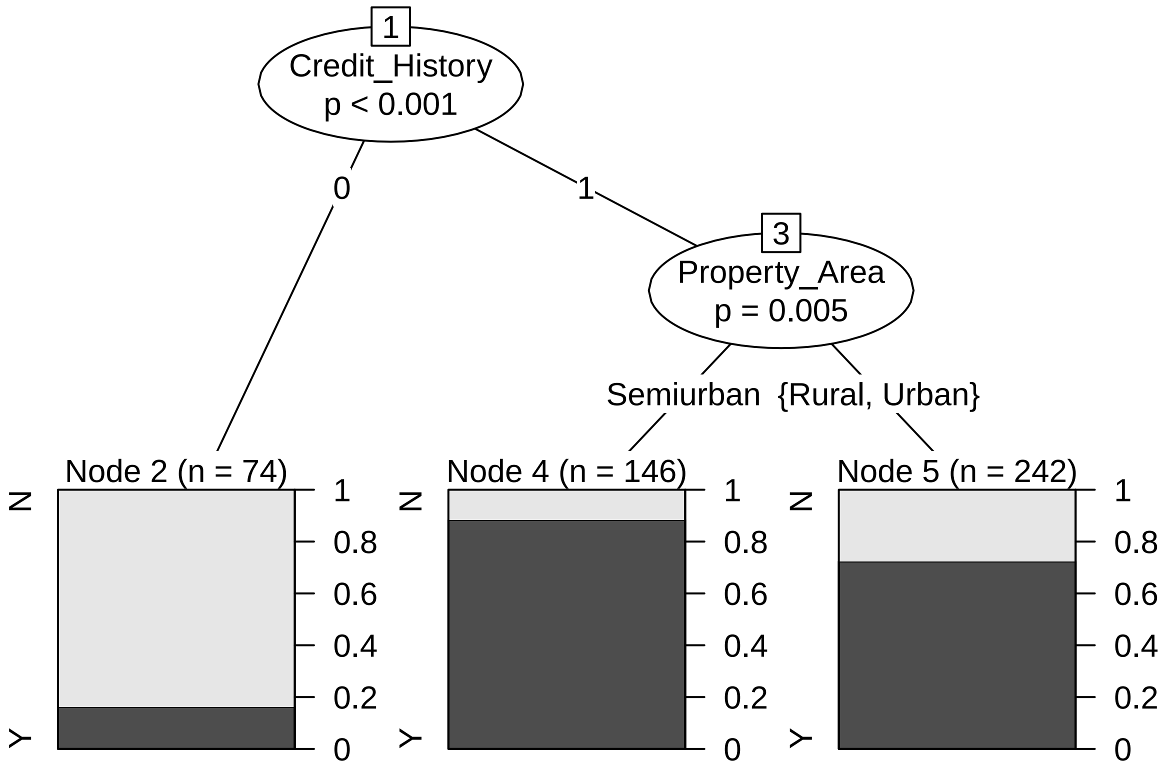
```
##  $ Loan_Status      : Factor w/ 2 levels "N","Y": 2 1 2 2 2 2 2 1 2 1 ...
```

Building Model1 Decision Trees

```r
library(caTools)
library(party)
```

```
## Loading required package: mvtnorm

## Loading required package: modeltools

## Loading required package: stats4

##
## Attaching package: 'modeltools'

## The following object is masked from 'package:arules':
##
##     info

## The following object is masked from 'package:plyr':
##
##     empty

## The following object is masked from 'package:BayesFactor':
##
##     posterior

## Loading required package: strucchange

## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following object is masked from 'package:tsibble':
##
##     index

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

## Loading required package: sandwich
```

```
loanDF2 <- loanDF1 %>%
                   dplyr::select(Gender, Married, Dependents, Education, Self_Employed, Property_Area, C

data1 = sample.split(loanDF2, SplitRatio = 0.80)
train1 <- subset(loanDF2, data1 == TRUE)
test1 <- subset(loanDF2, data1 == FALSE)
model1 <- ctree(Loan_Status ~ ., train1)
plot(model1)
```



Prediction of model1

```
pred1 <- predict(model1, test1)
classifier1 <- table(test1$Loan_Status, pred1)
classifier1
```

```
##    pred1
##       N   Y
##   N  21  25
##   Y   4 102
```

Accuracy of Model 1

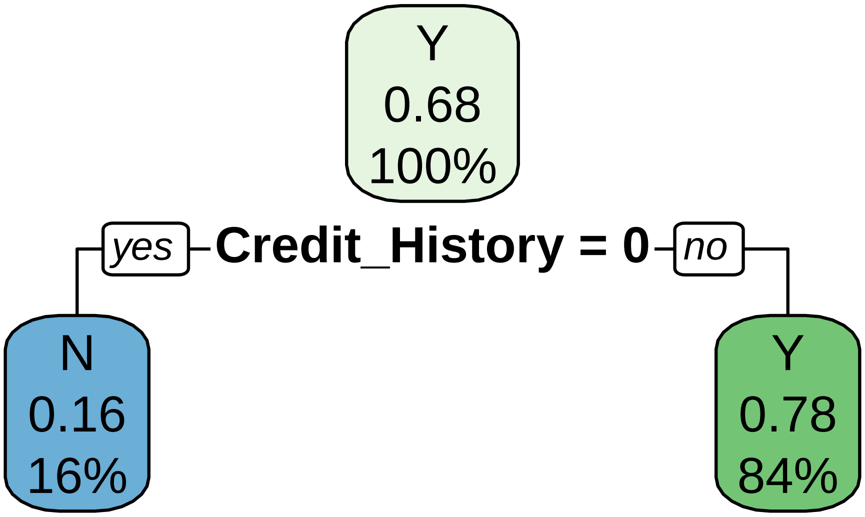
```
accuracy1 <- sum(diag(classifier1))/sum(classifier1)
accuracy1
```

```
## [1] 0.8092105
```

```
#str(loanDF2)

# # load package
# #install.packages("ggstatsplot")
# library(ggstatsplot)
#
# # correlogram
# ggstatsplot::ggcorrmat(
#   data = data1000R1,
#   type = "parametric", # parametric for Pearson, nonparametric for Spearman's correlation
#   colors = c("darkred", "white", "steelblue") # change default colors
# )
```

Let's try rpart function

```
library(rpart)
library(rpart.plot)
library(caret)

model2 <- rpart(Loan_Status ~.,method="class", data=train1)

rpart.plot(model2, tweak =1.6)
```




```
model2.pred <- predict(model2, test1, type="class")
model2.accuracy <- table(test1$Loan_Status, model2.pred, dnn=c("Actual", "Predicted"))
model2.accuracy
```

```
##       Predicted
## Actual   N   Y
##      N  21  25
##      Y   4 102
```

```
confusionMatrix(predict(model2, type = "class"), train1$Loan_Status)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   N   Y
##          N  62  12
##          Y  84 304
##
##                Accuracy : 0.7922
##                  95% CI : (0.7523, 0.8283)
##     No Information Rate : 0.684
##     P-Value [Acc > NIR] : 1.396e-07
##
##                   Kappa : 0.4458
```

```
##
##  Mcnemar's Test P-Value : 4.280e-13
##
##             Sensitivity : 0.4247
##             Specificity : 0.9620
##          Pos Pred Value : 0.8378
##          Neg Pred Value : 0.7835
##              Prevalence : 0.3160
##          Detection Rate : 0.1342
##    Detection Prevalence : 0.1602
##       Balanced Accuracy : 0.6933
##
##        'Positive' Class : N
##
```

```
# set.seed(232)
#
# library(caTools)
# data1000R1s <- sample.split(data1000R1, SplitRatio = 0.70)
# train1 <- subset(data1000R1, data1000R1s == TRUE)
# test1 <- subset(data1000R1, data1000R1s == FALSE)
#
# model1 <- lm(Total.Profit~., train1)
# summary(model1)
# plot (model1, which = 2)
#
# plot (model1, which = 1)
```

Model3 Random Forest

```
library(randomForest)
```

```
## randomForest 4.7-1
```

```
## 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
```

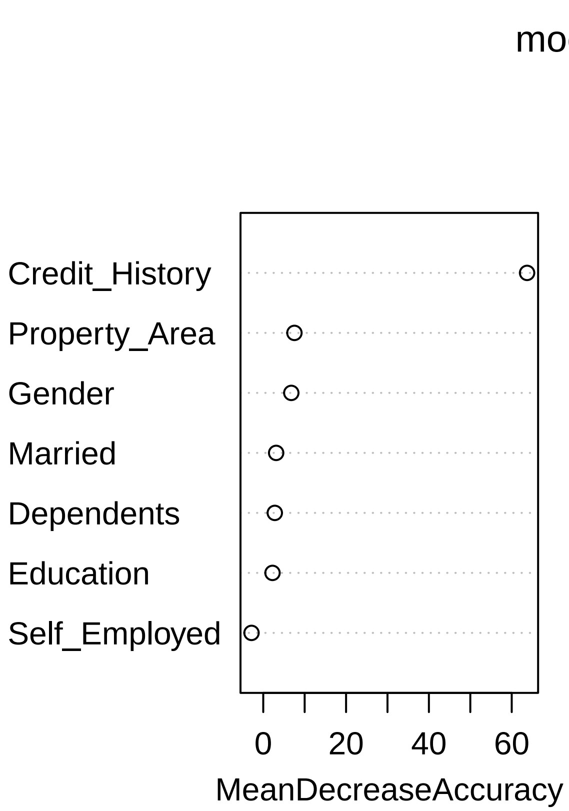
```
model3 <- randomForest(Loan_Status ~., data = train1, importance = TRUE, ntree=500)
print(model3)
```

```
##
## Call:
##  randomForest(formula = Loan_Status ~ ., data = train1, importance = TRUE,      ntree = 500)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 21.21%
## Confusion matrix:
##    N   Y class.error
## N 62  84   0.5753425
## Y 14 302   0.0443038
```

```
varImp(model3)
```

```
##                         N          Y
## Gender          2.9838443  2.9838443
## Married         0.6227917  0.6227917
## Dependents      0.5938169  0.5938169
## Education       1.8119032  1.8119032
## Self_Employed  -2.0796632 -2.0796632
## Property_Area   5.1007809  5.1007809
## Credit_History 58.0467064 58.0467064
```

```
varImpPlot(model3)
```



```r
#importance(model3, type = 2)

pred3 <- predict(model3, test1)
model3.accuracy <- table(test1$Loan_Status, pred3, dnn = c("actual", "predicted"))
model3.accuracy
```

```
##       predicted
## actual   N   Y
##      N  21  25
##      Y   4 102
```

```r
conf_matrix_RF <- confusionMatrix(pred3, test1$Loan_Status)
conf_matrix_RF
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   N   Y
##          N  21   4
##          Y  25 102
##
##                Accuracy : 0.8092
##                  95% CI : (0.7376, 0.8683)
##     No Information Rate : 0.6974
##     P-Value [Acc > NIR] : 0.0012442
##
##                   Kappa : 0.4809
##
##  Mcnemar's Test P-Value : 0.0002041
##
##             Sensitivity : 0.4565
##             Specificity : 0.9623
##          Pos Pred Value : 0.8400
##          Neg Pred Value : 0.8031
##              Prevalence : 0.3026
##          Detection Rate : 0.1382
##    Detection Prevalence : 0.1645
##       Balanced Accuracy : 0.7094
##
##        'Positive' Class : N
##
```

Summary of model performance

```r
library(kableExtra)
```

```
##
## Attaching package: 'kableExtra'
```

	decision_tree_model	randomForest_model
Accuracy	0.8092105	0.8092105
Sensitivity	0.4565217	0.4565217
Specificity	0.9622642	0.9622642
Pos Pred Value	0.8400000	0.8400000
Neg Pred Value	0.8031496	0.8031496
Precision	0.8400000	0.8400000
Recall	0.4565217	0.4565217
F1	0.5915493	0.5915493
Prevalence	0.3026316	0.3026316
Detection Rate	0.1381579	0.1381579
Detection Prevalence	0.1644737	0.1644737
Balanced Accuracy	0.7093929	0.7093929

```
## The following object is masked from 'package:dplyr':
##
##     group_rows
```

```
decision_tree_model <- confusionMatrix(table(model2.pred, test1$Loan_Status))$byClass
decision_tree_accuracy <- confusionMatrix(table(model2.pred, test1$Loan_Status))$overall['Accuracy']
decision_tree_model <- data.frame(decision_tree_model)
decision_tree_model <- rbind("Accuracy" = decision_tree_accuracy, decision_tree_model)


randomForest_model <- confusionMatrix(table(pred3, test1$Loan_Status))$byClass
randomforest_accuracy <- confusionMatrix(table(pred3, test1$Loan_Status))$overall['Accuracy']
randomForest_model <- data.frame(randomForest_model)
randomForest_model <- rbind("Accuracy" = randomforest_accuracy, randomForest_model)

summary_dt_rf <- data.frame(decision_tree_model, randomForest_model)

summary_dt_rf %>%
              kable() %>%
                     kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"))
```

The performance of the decision trees and random forest models appears to be about the same. We wonder if we didn't assign the same variable twice. Nonetheless, the code looks good and we calling the random forest and decision trees function. Perhaps the explanation is on the rpart() function . . . meaning we get the same result with RandomForest(). Let's switch the target variable and see if we still get the same result.

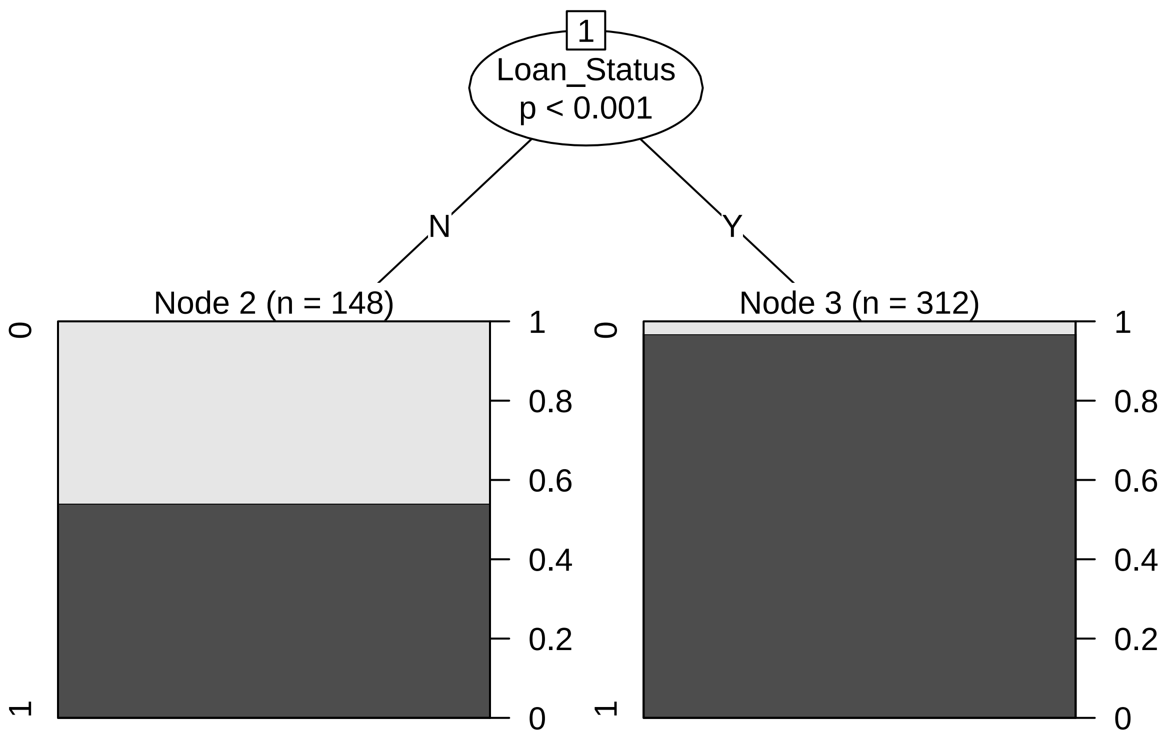
```
str(loanDF2)
```

```
## 'data.frame':    614 obs. of  8 variables:
##  $ Gender        : Factor w/ 3 levels "","Female","Male": 3 3 3 3 3 3 3 3 3 3 ...
##  $ Married       : Factor w/ 3 levels "","No","Yes": 2 3 3 3 2 3 3 3 3 3 ...
##  $ Dependents    : Factor w/ 5 levels "","0","1","2",..: 2 3 2 2 2 4 2 5 4 3 ...
##  $ Education     : Factor w/ 2 levels "Graduate","Not Graduate": 1 1 1 2 1 1 2 1 1 1 ...
##  $ Self_Employed : Factor w/ 3 levels "","No","Yes": 2 2 3 2 2 3 2 2 2 2 ...
##  $ Property_Area : Factor w/ 3 levels "Rural","Semiurban",..: 3 1 3 3 3 3 3 2 3 2 ...
##  $ Credit_History: Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 1 2 2 ...
##  $ Loan_Status   : Factor w/ 2 levels "N","Y": 2 1 2 2 2 2 2 1 2 1 ...
```

Credit history sounds appropriate for a target variable, let's say the bank want to predict if a customer requesting for a new loan based on the pre-existing conditions as described in the dataset met the minimum years loan qualification.

Model4 Decision Tree

```
library(caTools)
library(party)

loanDF2 <- loanDF1 %>%
                   dplyr::select(Gender, Married, Dependents, Education, Self_Employed, Property_Area, C

data2 = sample.split(loanDF2, SplitRatio = 0.80)
train2 <- subset(loanDF2, data2 == TRUE)
test2 <- subset(loanDF2, data2 == FALSE)
model4 <- ctree(Credit_History ~ ., train2)
plot(model4)
```



Prediction of model4

```
pred4 <- predict(model4, test2)
classifier2 <- table(test2$Credit_History, pred4)
classifier2
```

```
##    pred4
##       0   1
##   0   0  21
##   1   0 133
```

Accuracy of Model4

```
accuracy1 <- sum(diag(classifier2))/sum(classifier2)
accuracy1
```

```
## [1] 0.8636364
```

```
#str(loanDF2)

# # load package
# #install.packages("ggstatsplot")
# library(ggstatsplot)
#
# # correlogram
# ggstatsplot::ggcorrmat(
#   data = data1000R1,
#   type = "parametric", # parametric for Pearson, nonparametric for Spearman's correlation
#   colors = c("darkred", "white", "steelblue") # change default colors
# )
```

Model4 Random Forest

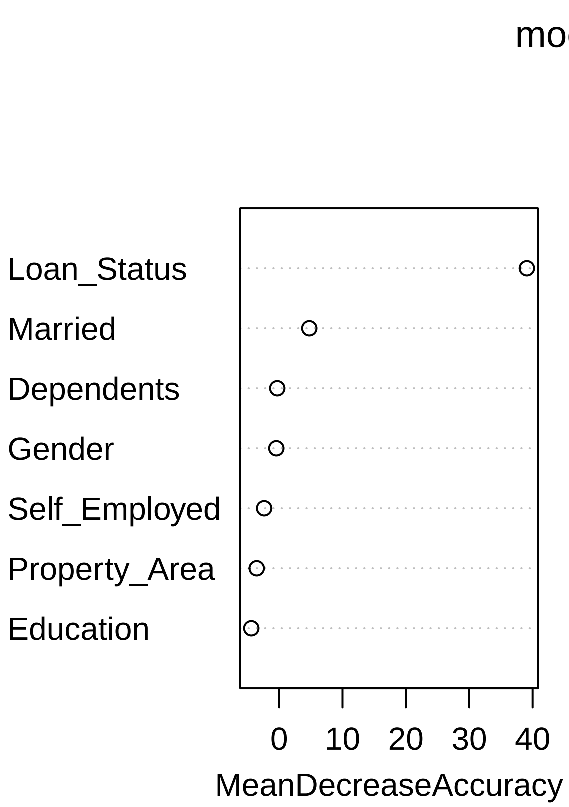
```
library(randomForest)

model5 <- randomForest(Credit_History ~., data = train2, importance = TRUE, ntree=500)
print(model5)
```

```
##
## Call:
##  randomForest(formula = Credit_History ~ ., data = train2, importance = TRUE,      ntree = 500)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 19.78%
## Confusion matrix:
##    0   1 class.error
## 0 13  65  0.83333333
## 1 26 356  0.06806283
```

```r
varImp(model5)
```

```
##                        0          1
## Gender        -0.0597435 -0.0597435
## Married        2.0295887  2.0295887
## Dependents    -0.7149852 -0.7149852
## Education     -2.9101691 -2.9101691
## Self_Employed -1.5031505 -1.5031505
## Property_Area -2.8970870 -2.8970870
## Loan_Status   35.0951505 35.0951505
```

```r
varImpPlot(model5)
```




```r
#importance(model3, type = 2)

pred5 <- predict(model5, test2)
model5.accuracy <- table(test2$Credit_History, pred5, dnn = c("actual", "predicted"))
model5.accuracy
```

```
##        predicted
## actual   0   1
##      0   5  16
##      1  11 122
```

```r
conf_matrix_RF <- confusionMatrix(pred5, test2$Credit_History)
conf_matrix_RF
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0   5  11
##          1  16 122
##
##                Accuracy : 0.8247
##                  95% CI : (0.7553, 0.8812)
##     No Information Rate : 0.8636
##     P-Value [Acc > NIR] : 0.9325
##
##                   Kappa : 0.1727
##
##  Mcnemar's Test P-Value : 0.4414
##
##             Sensitivity : 0.23810
##             Specificity : 0.91729
##          Pos Pred Value : 0.31250
##          Neg Pred Value : 0.88406
##              Prevalence : 0.13636
##          Detection Rate : 0.03247
##    Detection Prevalence : 0.10390
##       Balanced Accuracy : 0.57769
##
##        'Positive' Class : 0
##
```

Summary of model (Credit History as a target) performance

```r
library(kableExtra)
decision_tree_model <- confusionMatrix(table(pred4, test2$Credit_History))$byClass
decision_tree_accuracy <- confusionMatrix(table(pred4, test2$Credit_History))$overall['Accuracy']
decision_tree_model <- data.frame(decision_tree_model)
decision_tree_model <- rbind("Accuracy" = decision_tree_accuracy, decision_tree_model)


randomForest_model <- confusionMatrix(table(pred5, test2$Credit_History))$byClass
randomforest_accuracy <- confusionMatrix(table(pred5, test2$Credit_History))$overall['Accuracy']
randomForest_model <- data.frame(randomForest_model)
randomForest_model <- rbind("Accuracy" = randomforest_accuracy, randomForest_model)

summary_dt_rf <- data.frame(decision_tree_model, randomForest_model)

summary_dt_rf %>%
              kable() %>%
                     kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"))
```

	decision_tree_model	randomForest_model
Accuracy	0.8636364	0.8246753
Sensitivity	0.0000000	0.2380952
Specificity	1.0000000	0.9172932
Pos Pred Value	NaN	0.3125000
Neg Pred Value	0.8636364	0.8840580
Precision	NA	0.3125000
Recall	0.0000000	0.2380952
F1	NA	0.2702703
Prevalence	0.1363636	0.1363636
Detection Rate	0.0000000	0.0324675
Detection Prevalence	0.0000000	0.1038961
Balanced Accuracy	0.5000000	0.5776942

This time based on model accuracy , decision tree wins over random forest. We wonder if the different in the performance between the two models is not due to the fact we used ctree() function for the decision tree model. In addition, there is also the possibility of some biais because the dataset is not all clean(blank values present)