Loan Approval Project

Katarina I

28 7 2021

1. Introduction

This is a capstone project of my choice for the final course in the HarvardX Professional Certificate in Data Science. It uses a publicly available Kaggle dataset from a company that wants to automate the loan eligibility process based on customer details provided while filling online application form. To automate this process, they have provided a partial dataset in order to identify the customers segments eligible for loan. The train set consists of 614 rows and 13 columns being:

Variable	Description
Loan_ID	Unique Loan ID
Gender	Male/ Female
Married	Applicant married (Y/N)
Dependents	Number of dependents
Education	Applicant Education (Graduate/ Under Graduate)
Self_Employed	Self employed (Y/N)
ApplicantIncome	Applicant income
CoapplicantIncome	Coapplicant income
LoanAmount	Loan amount in thousands
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)

For this binary classification problem (loan granted/not granted) four different machine learning algorithms will be used and model performance will be tested.

This report is organized as follows:

1. An introduction that summarizes the goal of the project.
2. The exploratory analysis section that includes data exploration and visualization, insights gained upon which different modeling approaches will be discussed
3. Data preprocessing and feature engineering
4. Modeling section that presents the models and discusses the model performance
5. A conclusion section that gives a brief summary of the report, its limitations and future work

2. Exploratory analysis

2.1 Libraries and dowloading data

```
if (!require(tidyverse)) install.packages('tidyverse')
library(tidyverse)
if (!require(caret)) install.packages('caret')
library(caret)
if (!require(corrplot)) install.packages('corrplot')
library(corrplot)
if (!require(RColorBrewer)) install.packages('RColorBrewer')
library(RColorBrewer)
if (!require(rpart)) install.packages('rpart')
library(rpart)
```

Dataset has been dowloaded from Kaggle through the following links:

https://www.kaggle.com/altruistdelhite04/loan-prediction-problem-dataset?select=train_u6lujuX_CVtuZ9i.csv train data

https://www.kaggle.com/altruistdelhite04/loan-prediction-problem-dataset?select=test_Y3wMUE5_7gLdaTN.csv test data

Automated download is provided through GitHub:

```
test <- read.csv("https://raw.githubusercontent.com/eKInomics/Loan_prediction/main/test_Y3wMUE5_7gLdaTN
train <- read.csv("https://raw.githubusercontent.com/eKInomics/Loan_prediction/main/train_u6lujuX_CVtuZ
```

2.2 First glimpse of the dataset

```
dim(train)
```

```
## [1] 614  13
```

```
dim(test)
```

```
## [1] 367  12
```

```
head(train)
```

```
##    Loan_ID Gender Married Dependents    Education Self_Employed ApplicantIncome
## 1 LP001002   Male      No          0     Graduate            No             5849
## 2 LP001003   Male     Yes          1     Graduate            No             4583
## 3 LP001005   Male     Yes          0     Graduate           Yes             3000
## 4 LP001006   Male     Yes          0 Not Graduate            No             2583
## 5 LP001008   Male      No          0     Graduate            No             6000
## 6 LP001011   Male     Yes          2     Graduate           Yes             5417
##   CoapplicantIncome LoanAmount Loan_Amount_Term Credit_History Property_Area
## 1                 0         NA              360              1         Urban
```

```
## 2              1508        128              360              1         Rural
## 3                 0         66              360              1         Urban
## 4              2358        120              360              1         Urban
## 5                 0        141              360              1         Urban
## 6              4196        267              360              1         Urban
##   Loan_Status
## 1           Y
## 2           N
## 3           Y
## 4           Y
## 5           Y
## 6           Y
```

```
head (test)
```

```
##    Loan_ID Gender Married Dependents    Education Self_Employed ApplicantIncome
## 1 LP001015   Male     Yes          0     Graduate            No            5720
## 2 LP001022   Male     Yes          1     Graduate            No            3076
## 3 LP001031   Male     Yes          2     Graduate            No            5000
## 4 LP001035   Male     Yes          2     Graduate            No            2340
## 5 LP001051   Male      No          0 Not Graduate            No            3276
## 6 LP001054   Male     Yes          0 Not Graduate           Yes            2165
##   CoapplicantIncome LoanAmount Loan_Amount_Term Credit_History Property_Area
## 1                 0        110              360              1         Urban
## 2              1500        126              360              1         Urban
## 3              1800        208              360              1         Urban
## 4              2546        100              360             NA         Urban
## 5                 0         78              360              1         Urban
## 6              3422        152              360              1         Urban
```

There si no outcome (Status_Loan) in the test set, so model performance will be evaluated on partitioned train set. We will rename the train dataset to "data"

```
rm(test)
data <- train
summary(data)
```

```
##      Loan_ID       Gender    Married   Dependents        Education
##  LP001002:  1         : 13      :  3     : 15     Graduate    :480
##  LP001003:  1   Female:112   No :213   0 :345     Not Graduate:134
##  LP001005:  1   Male  :489   Yes:398   1 :102
##  LP001006:  1                          2 :101
##  LP001008:  1                          3+: 51
##  LP001011:  1
##  (Other) :608
##  Self_Employed ApplicantIncome CoapplicantIncome   LoanAmount
##     : 32       Min.   :  150   Min.   :    0     Min.   :  9.0
##  No :500       1st Qu.: 2878   1st Qu.:    0     1st Qu.:100.0
##  Yes: 82       Median : 3812   Median : 1188     Median :128.0
##                Mean   : 5403   Mean   : 1621     Mean   :146.4
##                3rd Qu.: 5795   3rd Qu.: 2297     3rd Qu.:168.0
##                Max.   :81000   Max.   :41667     Max.   :700.0
##                                                  NA's   :22
```

```
##  Loan_Amount_Term Credit_History     Property_Area Loan_Status
##  Min.   : 12      Min.   :0.0000   Rural    :179   N:192
##  1st Qu.:360      1st Qu.:1.0000   Semiurban:233   Y:422
##  Median :360      Median :1.0000   Urban    :202
##  Mean   :342      Mean   :0.8422
##  3rd Qu.:360      3rd Qu.:1.0000
##  Max.   :480      Max.   :1.0000
##  NA's   :14       NA's   :50
```

Our dataset consists of 614 rows and 13 columns: outcome (Loan_Status) and 12 features: Loan_ID, Gender, Married, Dependents, Education, Self_Employed, ApplicantIncome, CoapplicantIncome, LoanAmount, Loan_Amount_Term, Credit_History, Property_Area)

It appears to have some empty cells and NA's that we will have to deal with.

```
data[data==""] <- NA # replace empty cells with NA's

colSums(is.na(data))
```

```
##           Loan_ID            Gender           Married        Dependents
##                 0                13                 3                15
##         Education     Self_Employed   ApplicantIncome CoapplicantIncome
##                 0                32                 0                 0
##        LoanAmount  Loan_Amount_Term    Credit_History     Property_Area
##                22                14                50                 0
##       Loan_Status
##                 0
```

```
sum(is.na(data))
```

```
## [1] 149
```

There are 149 missing values in our train dataset, the most in the credit history, followed by self employment, loan amounts, dependents...

```
str(data)
```

```
## 'data.frame':    614 obs. of  13 variables:
##  $ Loan_ID          : Factor w/ 614 levels "LP001002","LP001003",..: 1 2 3 4 5 6 7 8 9 10 ...
##  $ 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       : 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    : 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 ...
```

There are 8 variables catogorized as categorical (some of them ordinal) and 5 numerical variables in the train set. A numerical Credit_History variable (1/0) appears to have categorical characteristics hence will be factorized for the purposes of easier exploratory analyses.

```
data$Credit_History <- as.factor(data$Credit_History)
```

2.3 Data exploration and visualization

Outcome

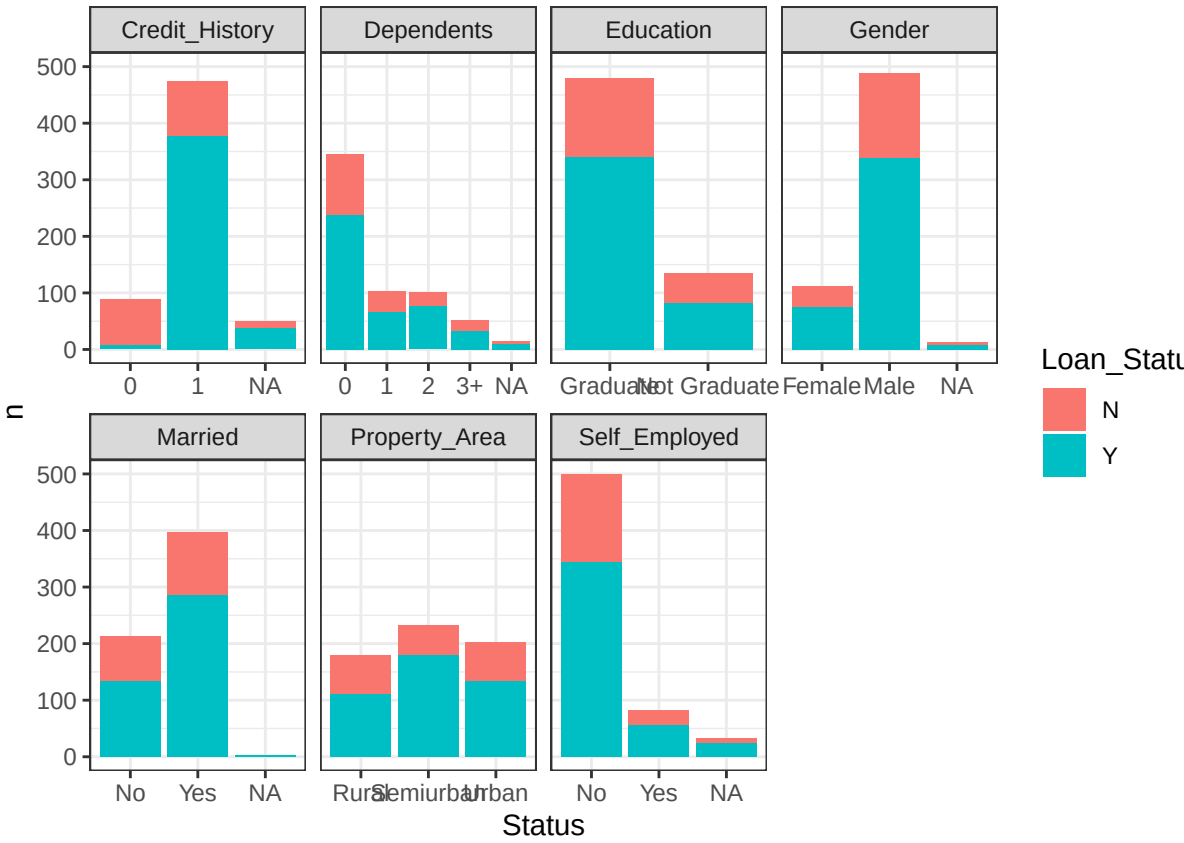
```
prop.table(table(data$Loan_Status))%>%round(3)
```

```
##
##     N     Y
## 0.313 0.687
```

68.7% of applicants in our dataset have been granted a loan

Features

```
data %>%
  select(where(is.factor), -Loan_ID) %>%
  pivot_longer(-Loan_Status,names_to="Variable", values_to="Status")%>%
  group_by(Variable, Status, Loan_Status)%>%tally()%>%
  ggplot()+
  geom_col(aes(x=Status, y=n, fill=Loan_Status))+
  theme_bw()+
  facet_wrap(~Variable, scales="free_x", ncol=4)
```

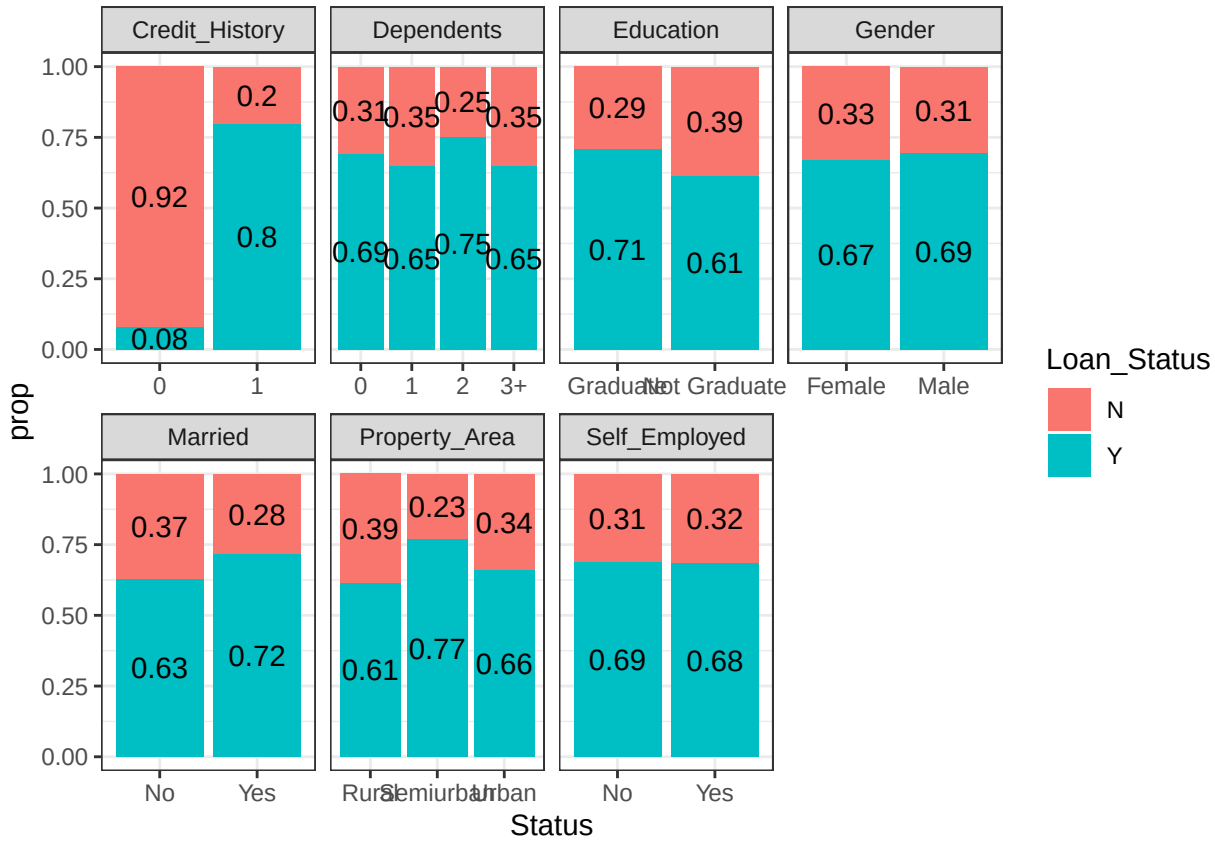


Categorical data

Visual inspection of the dataset shows that majority of applicants have credit history that meets the guidelines (1), have 0 dependents, graduated, are male, married and employed (not self-employed) and live in a semiurban area. Mode (most frequent value) will be used to replace NA's in this case.

Some features make difference in probability of getting a loan. Let's have a closer look.

```
data %>%
  select(where(is.factor), -Loan_ID) %>%
  pivot_longer(-Loan_Status,names_to="Variable", values_to="Status")%>%
  group_by(Variable, Status)%>%count(Loan_Status)%>%
  filter(!is.na(Status))%>%mutate(prop = prop.table(n))%>%
  ggplot(aes(x=Status, y=prop,fill=Loan_Status))+
  geom_col()+
  geom_text(aes(label = round(prop,2)),
            position = position_stack(vjust = .5))+
  theme_bw()+
  facet_wrap(~Variable, scales="free_x", ncol=4)
```



It appears that applicants with appropriate credit history (1) are much more likely to get a loan. More likely to be granted a loan are also those who have 2 dependants, graduates, married and with a semiurban property. At first glimpse gender and employment status do not seem to be important in decision for loan approval

```
data %>%
  select(where(is.numeric))%>%summary()
```

Numerical data

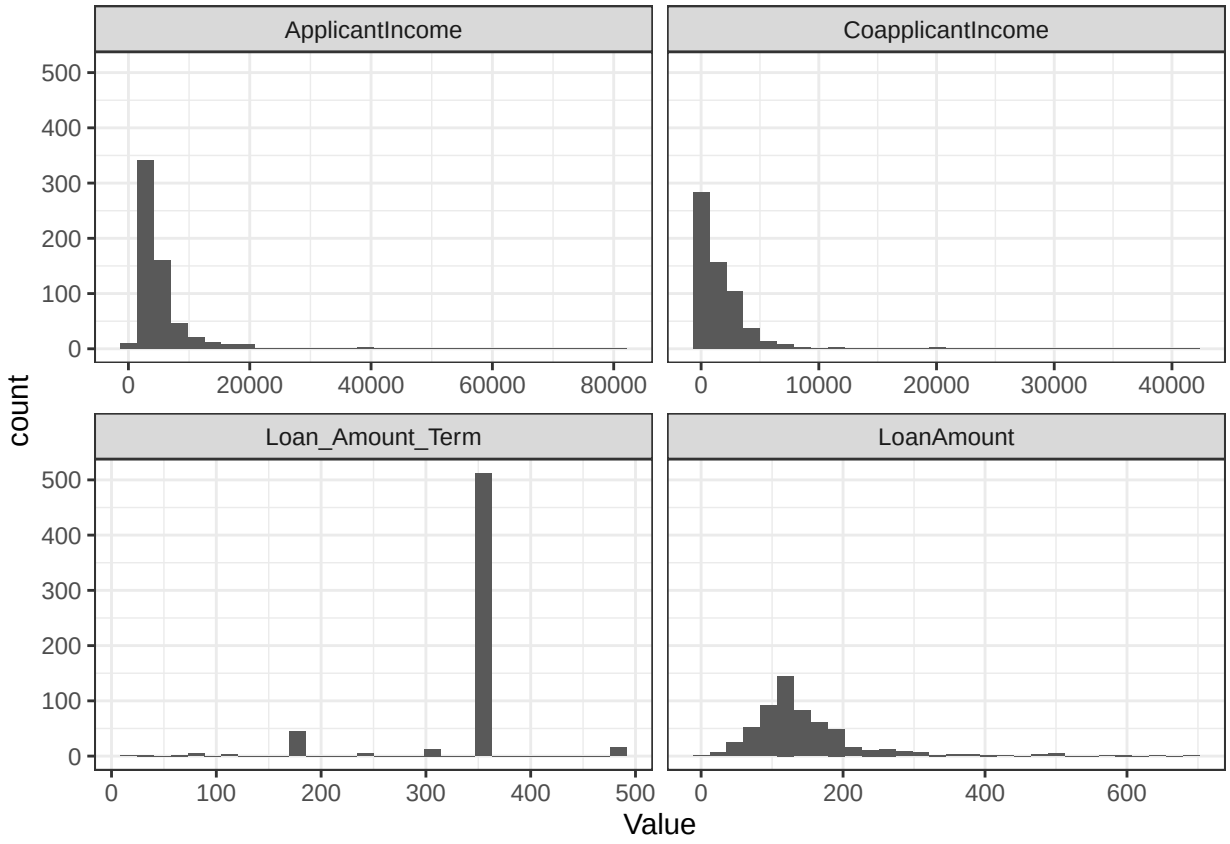
```
##  ApplicantIncome CoapplicantIncome   LoanAmount    Loan_Amount_Term
##  Min.   :  150   Min.   :    0     Min.   :  9.0   Min.   : 12
##  1st Qu.: 2878   1st Qu.:    0     1st Qu.:100.0   1st Qu.:360
##  Median : 3812   Median : 1188     Median :128.0   Median :360
##  Mean   : 5403   Mean   : 1621     Mean   :146.4   Mean   :342
##  3rd Qu.: 5795   3rd Qu.: 2297     3rd Qu.:168.0   3rd Qu.:360
##  Max.   :81000   Max.   :41667     Max.   :700.0   Max.   :480
##                                    NA's   :22      NA's   :14
```

First, one can notice there are missing values in the variables Loan_Amount and Loan_Amount_Term. Summary statistics also shows that the average applicant's income is 5403, much higher than the average coapplicants income (1621). Average loan amount is 146.4 (in thousands) and average Loan amount term is

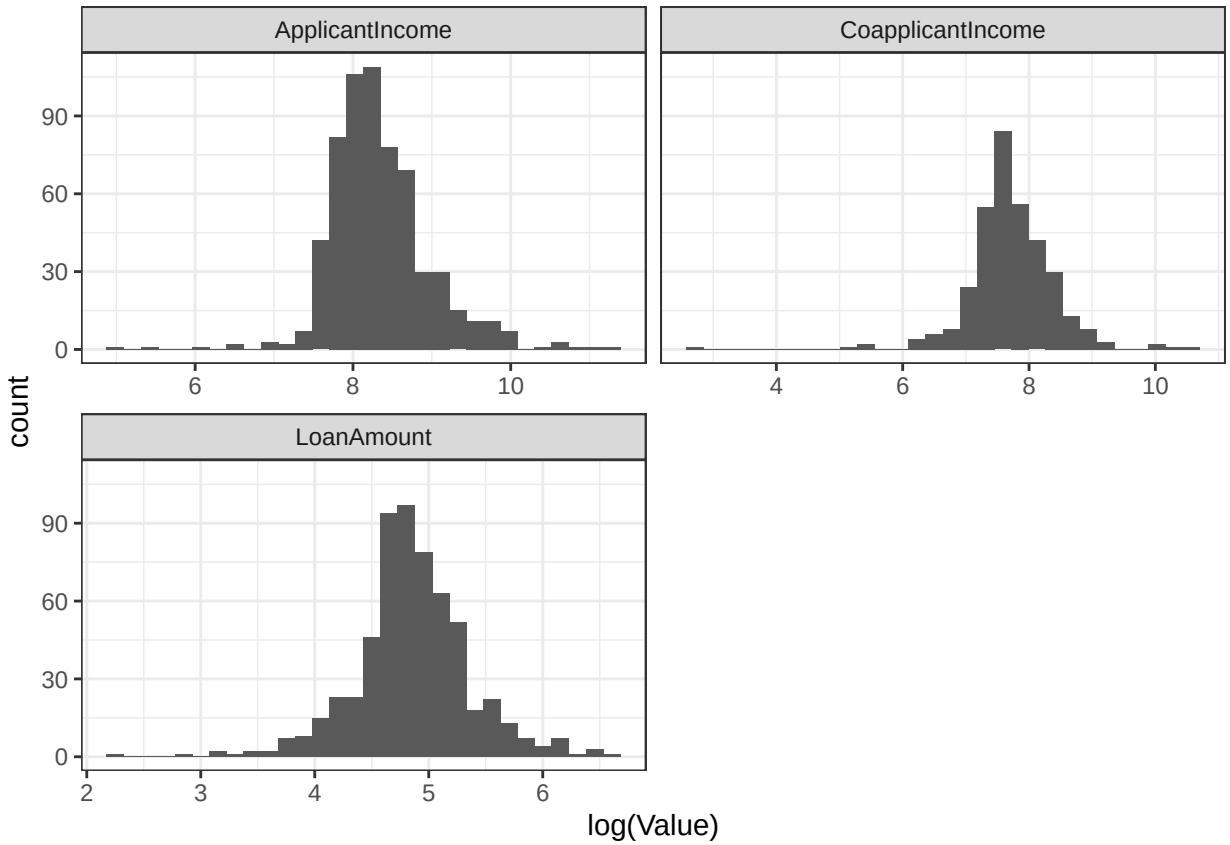
342 months. For the first three numerical features, mean is higher than the median, suggesting right-skewed distributions.

Let's look at the histograms (before NA's replaced):

```
data%>%select(ApplicantIncome, CoapplicantIncome, LoanAmount, Loan_Amount_Term, Loan_Status)%>%
  pivot_longer(-Loan_Status,names_to="Variable", values_to="Value")%>%
  ggplot()+
  geom_histogram(aes(x=Value))+
  facet_wrap(~Variable, scales="free_x", ncol=2)+
  theme_bw()
```



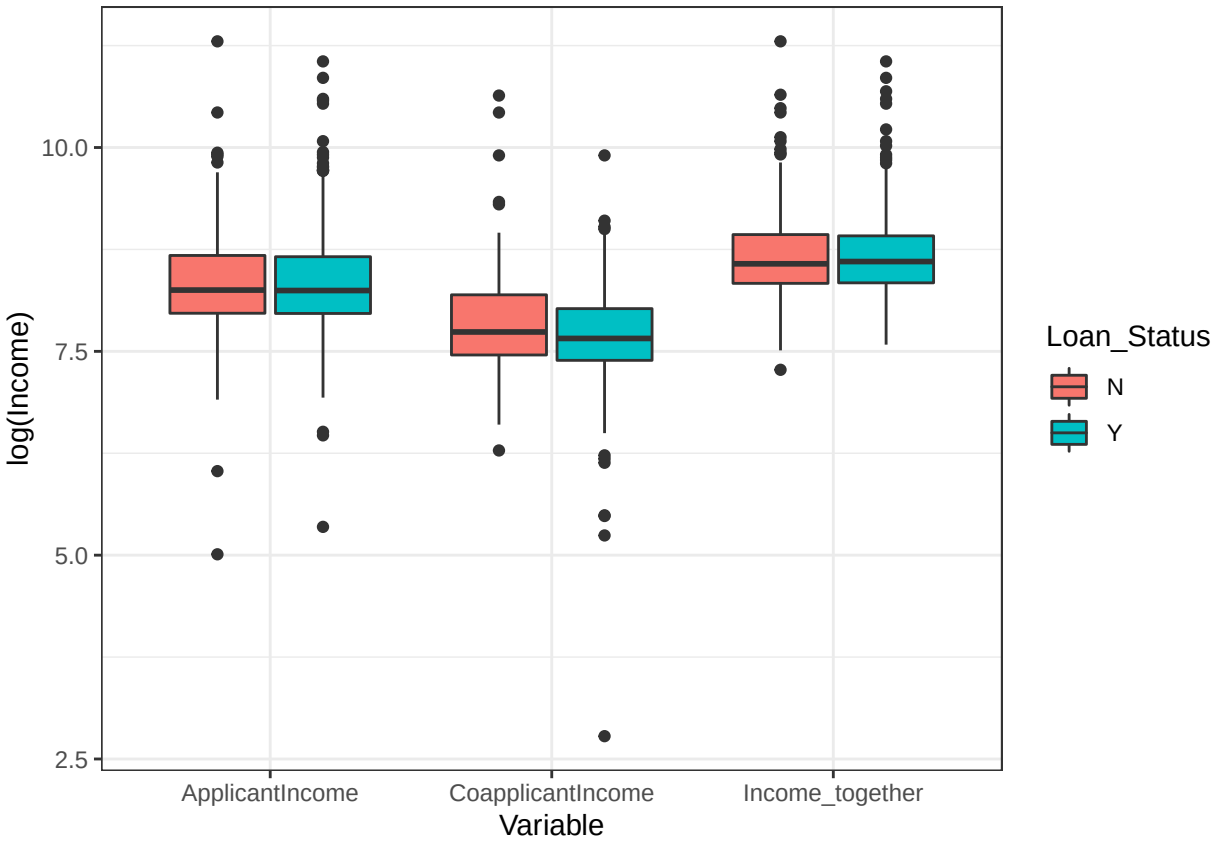

```
# log-transformations
data%>%select(ApplicantIncome, CoapplicantIncome, LoanAmount,  Loan_Status)%>%
  pivot_longer(-Loan_Status,names_to="Variable", values_to="Value")%>%
  ggplot()+
  geom_histogram(aes(x=log(Value)))+
  facet_wrap(~Variable, scales="free_x", ncol=2)+
  theme_bw()
```



Histograms confirm right-skewed distributions of applicant and co-applicant income as well as loan amount. There are also quite a lot of outliers in the features. Median (less susspective to extreme values than mean) will be used to replace NA's in Loan_Amount (recall, there are no missing values in case of applicant's and coapplicant's income). In case of Loan_Amount_Term 360 is by far the most frequent value and coincides with the median value.

Let's check for any apparent relations between loan approval status and numerical variables

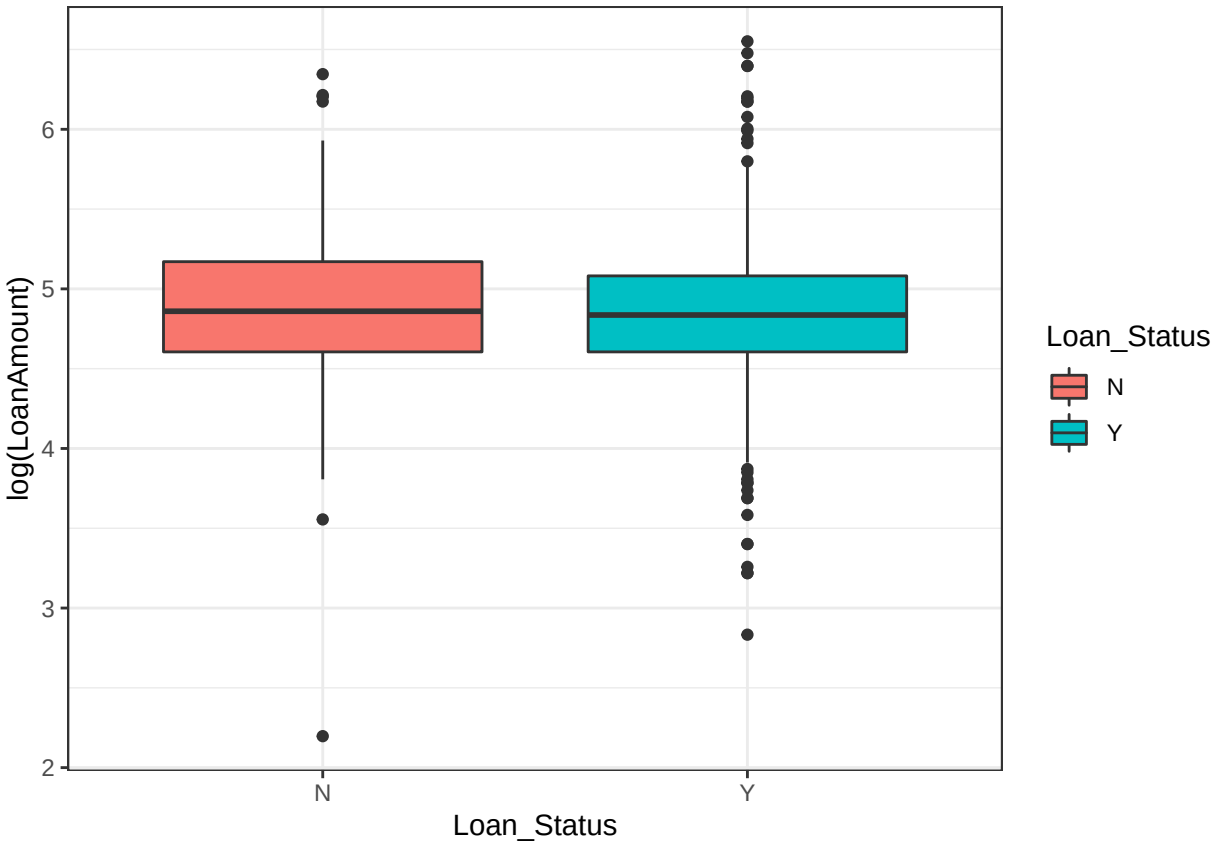
```
# Loan status vs incomes
data%>%select(ApplicantIncome, CoapplicantIncome, Loan_Status)%>%
  mutate(Income_together=ApplicantIncome+CoapplicantIncome)%>%
  pivot_longer(-Loan_Status,names_to="Variable", values_to="Income")%>%
  ggplot(aes(x=Variable, y=log(Income), fill=Loan_Status))+
  geom_boxplot()+
  theme_bw()
```



Suprisingly, at first glimpse it seems that applicant's income does not affect chances of loan approvals. Even more suprisingly, it appears that median coapplicants income is lower in cases where loan has been granted. We have therefore created a variable income_together, where incomes of both, applicant and co-applicant, are combined. Yet, not even their income combined gives a strightforward evidence of its importance for prediction of loan status.

Let's explore the loan amount

```
data%>%select(LoanAmount, Loan_Status)%>% filter(!is.na(LoanAmount))%>%
  ggplot(aes(x=Loan_Status, y=log(LoanAmount), fill=Loan_Status))+
  geom_boxplot()+
  theme_bw()
```



There seems to be no straightforward relationship between loan amount and loan status. This might be due to higher loan amounts requested from those with higher income and banks might be cautious about their disposable income after loan payment. We will explore correlations between pairs of variables and try to overcome this issues later

3. Data Preprocessing and feature engineering

3.1 Imputing missing values

Visual inspection in the prevoius section led us to believe that it would be a good call to replace NA's with their mode for categorical and with their median for numerical variables.

```
# Mode function defined
Mode <- function(x) {
  ux <- unique(x)
  ux[which.max(tabulate(match(x, ux)))]
  }

# replace NA's with their mode for categorical and with their median for numerical var.

data<-data %>% select(-Loan_ID)%>%
  mutate_if(is.numeric, list(~replace(.,is.na(.), median(., na.rm = TRUE)))) %>%
  mutate_if(is.factor, list(~replace(.,is.na(.), Mode(na.omit(.)))))
```

```r
colSums(is.na(data)) # no more NA's
```

```
##            Gender           Married        Dependents         Education
##                 0                 0                 0                 0
##     Self_Employed   ApplicantIncome CoapplicantIncome        LoanAmount
##                 0                 0                 0                 0
##  Loan_Amount_Term    Credit_History     Property_Area       Loan_Status
##                 0                 0                 0                 0
```

3.2 Encoding categorical features

Let's transform all categorical variables to numerical for easier use

```r
data <- data %>%
  mutate(Dependents = ifelse(as.character(Dependents) == "3+", 3, as.numeric(as.character(Dependents)))
  Gender_male=ifelse(Gender=="Male", 1, 0), # (Male=1, Female=0)
  Married_yes=ifelse(Married=="Yes", 1,0), # (Married =1, Not married =0)
  Education_graduate=ifelse(Education=="Graduate",1,0), #(Graduates = 1, Undergraduates=0)
  Self_Employed=ifelse(Self_Employed=="Yes", 1,0), # (Self-employed=1, Employed=0)
  Propery_Urban=ifelse(Property_Area=="Urban", 2, # (Urban=2, Semiurban=1, Rural=0)
                       ifelse(Property_Area=="Semiurban",1,0)),
  Loan_Status=ifelse(Loan_Status=="Y", 1, 0))%>% # (Loan approved= 1, Not approved=0)
  mutate(Credit_History=as.numeric(as.character(Credit_History)))%>%
  select(3, 5:10, 12:16)
```

```r
str(data)
```

```
## 'data.frame':    614 obs. of  12 variables:
##  $ Dependents        : num  0 1 0 0 0 2 0 3 2 1 ...
##  $ Self_Employed     : num  0 0 1 0 0 1 0 0 0 0 ...
##  $ ApplicantIncome   : num  5849 4583 3000 2583 6000 ...
##  $ 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 ...
##  $ Loan_Status       : num  1 0 1 1 1 1 1 0 1 0 ...
##  $ Gender_male       : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ Married_yes       : num  0 1 1 1 0 1 1 1 1 1 ...
##  $ Education_graduate: num  1 1 1 0 1 1 0 1 1 1 ...
##  $ Propery_Urban     : num  2 0 2 2 2 2 2 1 2 1 ...
```

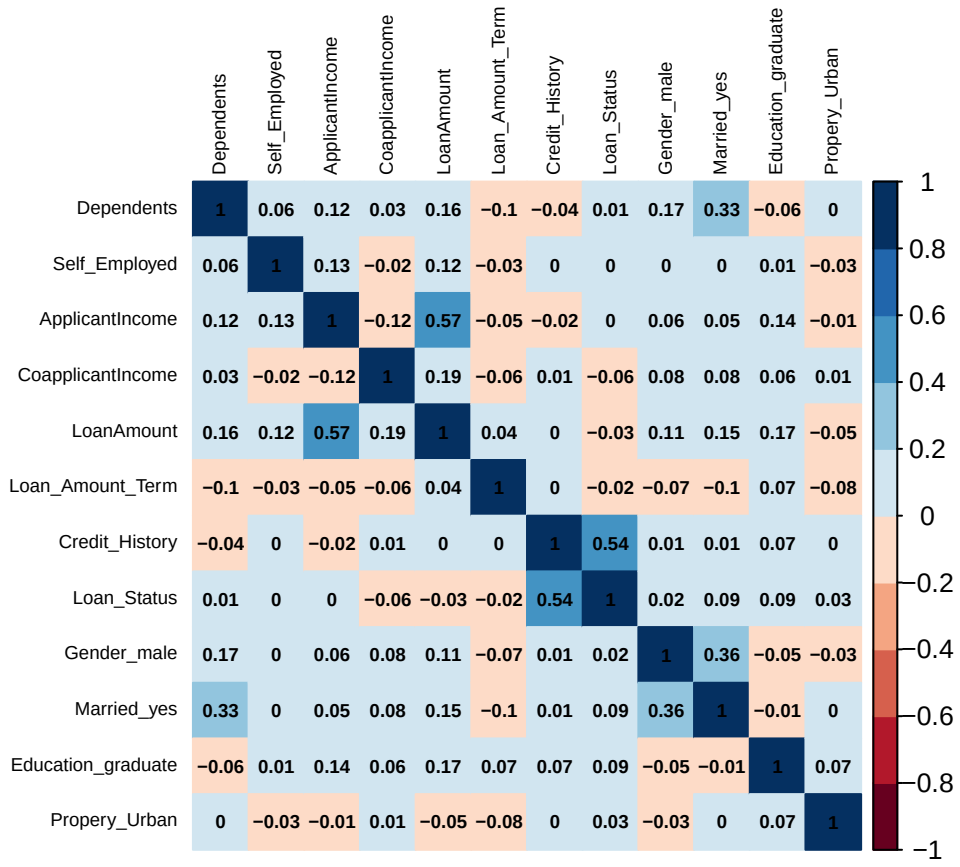
3.3 Correlation matrices

Now let's look at the correlations between the pairs of variables.

```r
cor_mat <- round(cor(data),2)

corrplot(cor_mat,       #  corrplot and RColorBrewer libraries used
         method="color",
         col=brewer.pal(n=10, name="RdBu"),
```

```
        addCoef.col = "black", # Add correlation coeff.
        tl.col="black", #Text label color
        tl.cex=0.6,
        number.cex=0.6)
```




```
rm(cor_mat)
```

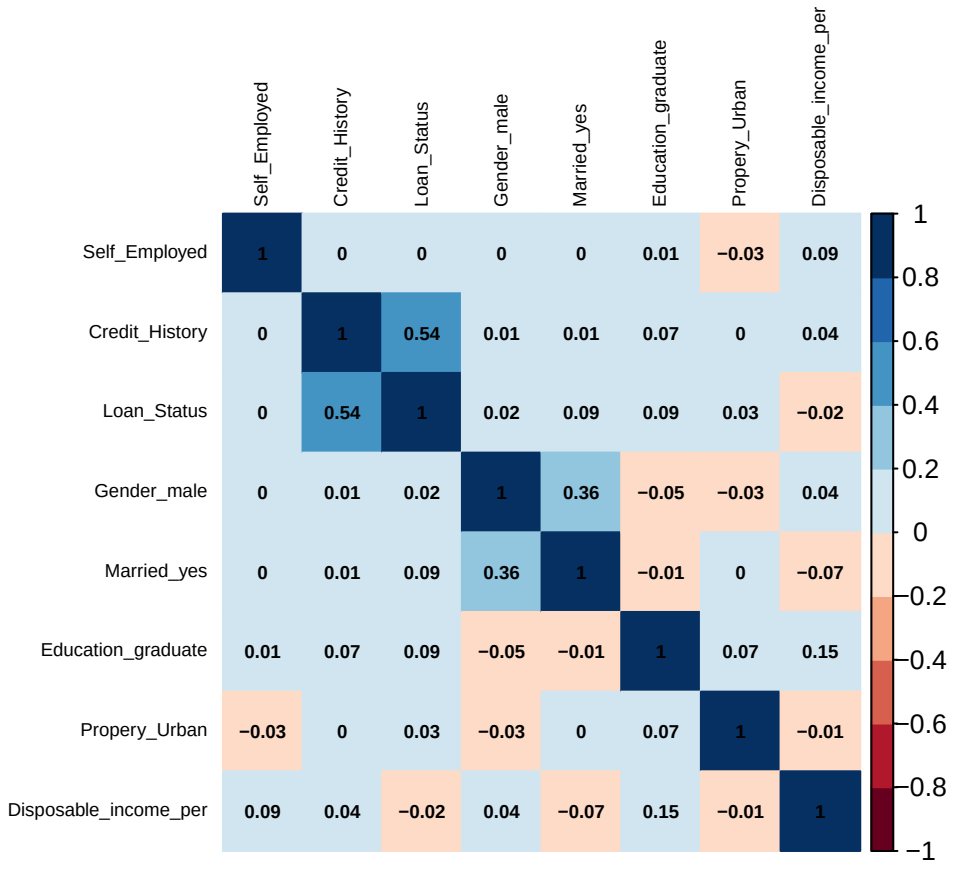
As we have suspected from visual inspection, in terms of our target variable (Loan_Status) Credit_history has by far the largest coefficient of correlation. It also proves that applicant's income doesn't correlate with loan status, while coapplicants's income even shows negative correlation with loan status. That might be because applicant's income is highly correlated to loan amount which in turn is negatively correlated with loan status. This implies that creating new features that take this relations into account might be useful.

3.4 Creating new features

```
data_new <- data%>%
  mutate(Income_together=ApplicantIncome+CoapplicantIncome,
         Monthly_payment=LoanAmount*1000/Loan_Amount_Term, # Loan amount is in thousands (multiply by 1
         Disposable_income=Income_together-Monthly_payment, # Income after monthly payments
         Disposable_income_per=Disposable_income/(Dependents+1))%>%   # Income per dependent person +1
  select(-LoanAmount, -Loan_Amount_Term, -ApplicantIncome, -CoapplicantIncome, -Dependents,
         - Income_together, -Monthly_payment, -Disposable_income) # get rid off varibales incorporated
```

```
cor_mat_2 <- round(cor(data_new),2)

corrplot(cor_mat_2, method="color", col=brewer.pal(n=10, name="RdBu"),
         addCoef.col = "black", # Add correlation coeff.
         tl.col="black",
         tl.cex=0.6,
         number.cex=0.6)
```

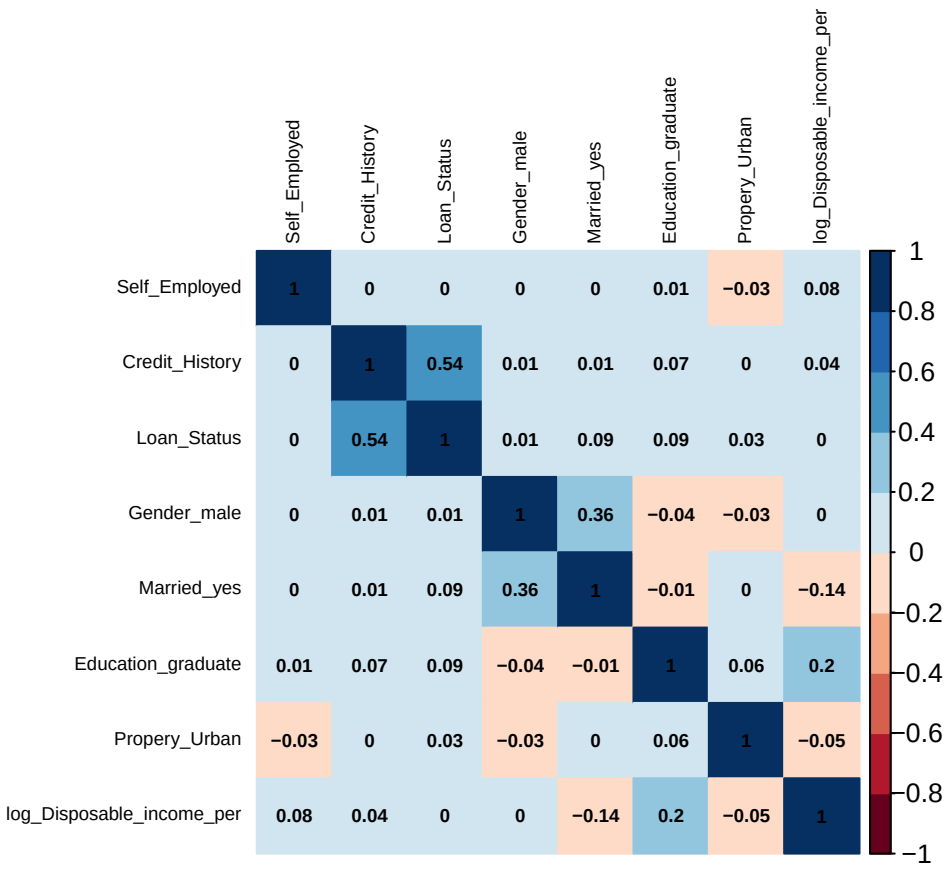


We will proceed by performing log transformations on income variables

```
data_new_log <- data_new%>%
  filter(Disposable_income_per>=0)%>% # Get rid of negative values before log
  mutate(log_Disposable_income_per=log(Disposable_income_per))%>%
  select(-Disposable_income_per)

cor_mat_log <- round(cor(data_new_log),2)

corrplot(cor_mat_log, method="color", col=brewer.pal(n=10, name="RdBu"),
         addCoef.col = "black", # Add correlation coeff.
         tl.col="black",
         tl.cex=0.6,
         number.cex=0.6)
```




```
rm(list=c("cor_mat_2","cor_mat_log", "data_new"))
```

After all the transformations, disposable income per person, which shows how much income per person a household is left after paying for loan, still shows no correlation with loan status.

4. Modeling

We will use 2 different datasets for modelling:

```
a) data, where only necessary imputations of missing values were performed

b) data_new_log, where more feature engineering has been done. We have created new variables and made l
```

Models will be evaluated on both datasets.

4.1 Partitioning

Because of the lack of target (outcome) variable in the original Keggle test set, we will partition the data which was originally tagged as the train. 20% of data will be used as test set and remaining 80% as train set

a) "Data" : with only necessary imputations of missing values performed

```r
# change target back to more intuitive (1=Y, 0=N)
data<-data%>%
  mutate(Loan_Status=ifelse(Loan_Status==1,"Y","N"))

set.seed(3, sample.kind = "Rounding")

#partition on Loan_Status
test_index <- createDataPartition(data$Loan_Status, times = 1, p = 0.2, list = FALSE)
test_set <- data[test_index, ]# Assign the 20% partition to test_set and
train_set <- data[-test_index, ] # the remaining 80% partition to train_set

dim(test_set) #check dimensions
```

```
## [1] 124  12
```

```r
dim(train_set)
```

```
## [1] 490  12
```

```r
mean(test_set$Loan_Status=="Y") # check means
```

```
## [1] 0.6854839
```

```r
mean(train_set$Loan_Status=="Y")
```

```
## [1] 0.6877551
```

b) "Data_new_log" : new variable out of existing (which were omitted) + log transformation

```r
data_new_log<-data_new_log%>%
  mutate(Loan_Status=ifelse(Loan_Status=="1","Y","N"))

set.seed(9, sample.kind = "Rounding")
#partition on Loan_Status
test_index <- createDataPartition(data_new_log$Loan_Status, times = 1, p = 0.2, list = FALSE)

test_set_2 <- data_new_log[test_index, ]# Assign the 20% partition to test_set and
train_set_2 <- data_new_log[-test_index, ] # the remaining 80% partition to train_set

dim(test_set_2) #check dimensions
```

```
## [1] 124   8
```

```r
dim(train_set_2)
```

```
## [1] 488   8
```

```r
mean(test_set_2$Loan_Status=="Y") # check means
```

```
## [1] 0.6854839
```

```r
mean(train_set_2$Loan_Status=="Y")
```

```
## [1] 0.6885246
```

One can notice that datasets are of somewhat different dimensions. We have created new variables out of existing ones in the data_new_log, hence those were no longer needed. Additionally, we have lost 2 observations due to log transformation.

Both train and test set seem to have similar proportions of those who have been granted loans

4.2 Models

We will make predictions based on the following models:

1. Simple model using credit history only
2. Logistic regression
3. KNN
4. Decision tree
5. Random forest

First, we will train the algorithms on the train_set, a partion of the "data" that was treated only for the missing values. Values will be predicted by different ML algorithms on the test_set and model performance discussed (a)

Then everything will be repeated on the train_set_2 and test_set_2, partions of the "data_new_log" dataset where additional feature engineering has been performed (b)

```r
train_set$Loan_Status <- as.factor(train_set$Loan_Status) #convert outcome to factor
test_set$Loan_Status <- as.factor(test_set$Loan_Status)

train_set_2$Loan_Status <- as.factor(train_set_2$Loan_Status)
test_set_2$Loan_Status <- as.factor(test_set_2$Loan_Status)
```

Before we start, let's create matrix for performance metrics

```r
# a)
metric = matrix(rep(0), nrow=4, ncol=4)
colnames(metric )<- c('Accuracy', 'Sensitivity', 'Specificity', 'F1')
rownames(metric) <- c('Simple model with credit_history', 'Logistic regression',
                      'kNN', "Decision tree")

# b)

metric_2 = matrix(rep(0), nrow=4, ncol=4)
colnames(metric_2)<- c('Accuracy', 'Sensitivity', 'Specificity', 'F1')
rownames(metric_2) <- c('Simple model with credit_history', 'Logistic regression',
                        'kNN', "Decision tree")
```

4.2.1. Simple model using credit history only

Previously we have seen that credit history seems to be an important feature in loan approval prediction. We will therefore train a very simple model, where credit history is the only predictor.

a)

```
train_simple <- train(Loan_Status ~ Credit_History,
                      method="glm",
                      family = "binomial",
                      data= train_set)

cm <- confusionMatrix(data = predict(train_simple, test_set),
                reference = test_set$Loan_Status)

cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  N  Y
##          N 20  0
##          Y 19 85
##
##                Accuracy : 0.8468
##                  95% CI : (0.7711, 0.9052)
##     No Information Rate : 0.6855
##     P-Value [Acc > NIR] : 3.157e-05
##
##                   Kappa : 0.5907
##
##  Mcnemar's Test P-Value : 3.636e-05
##
##             Sensitivity : 0.5128
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.8173
##              Prevalence : 0.3145
##          Detection Rate : 0.1613
##    Detection Prevalence : 0.1613
##       Balanced Accuracy : 0.7564
##
##        'Positive' Class : N
##
```

```
metric[1,2:4] <-round(cm$byClass[c("Sensitivity","Specificity","F1")],3)
metric[1,1] <-round(cm$overall["Accuracy"],3)
metric[1,]
```

```
##    Accuracy Sensitivity Specificity          F1
##       0.847       0.513       1.000       0.678
```

b)

```
train_simple_2 <- train(Loan_Status ~ Credit_History,
                        method="glm",
                        family = "binomial",
                        data= train_set_2)

cm_2 <- confusionMatrix(data = predict(train_simple_2, test_set_2),
                reference = test_set_2$Loan_Status)

metric_2[1,2:4] <-round(cm_2$byClass[c("Sensitivity","Specificity","F1")],3)
metric_2[1,1] <-round(cm_2$overall["Accuracy"],3)
metric_2[1,]
```

```
##    Accuracy Sensitivity Specificity          F1
##       0.798       0.410       0.976       0.561
```

This simple model that uses credit history as the only predictor shows relatively high accuracy. This is possible despite low sensitivity and is due to low prevalence: the proportion of loans not granted (Loan_Status=N, N being our Positive class) is relatively low. Failing to predict loan granted when it actually hasn't been granted (low sensitivity) does not lower the accuracy as much as failing to predict loans granted when actual loans were granted (low specificity). There are 2 measures of model performance that overcome the issues of highly imbalanced data (balanced accuracy and F1 score). We will pay extra attention to sensitivity and F1 score in the rest of this project.

4.2.2. Logistic regression

Now we will try logistic regression using all the predictors in the dataset. Logistic regression provides an alternative to linear regression for binary classification problems.

a)

```
train_glm <- train(Loan_Status ~ .,
                   method="glm",
                   family = "binomial",
                   trControl = trainControl(method = "cv", number = 5),
                   data= train_set)

summary(train_glm)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0467  -0.4430   0.5828   0.7112   2.4505
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -2.762e+00  8.436e-01  -3.274  0.00106 **
## Dependents         -1.953e-02  1.285e-01  -0.152  0.87925
## Self_Employed      -1.460e-01  3.381e-01  -0.432  0.66585
```

```
## ApplicantIncome     2.550e-05  2.300e-05   1.109  0.26745
## CoapplicantIncome  -4.328e-05  3.733e-05  -1.159  0.24636
## LoanAmount         -2.460e-03  1.743e-03  -1.411  0.15822
## Loan_Amount_Term    1.921e-04  1.827e-03   0.105  0.91626
## Credit_History      3.529e+00  4.276e-01   8.252  < 2e-16 ***
## Gender_male        -3.082e-02  3.057e-01  -0.101  0.91971
## Married_yes         7.842e-01  2.707e-01   2.897  0.00377 **
## Education_graduate  3.655e-01  2.784e-01   1.313  0.18921
## Propery_Urban       4.708e-02  1.474e-01   0.319  0.74938
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 608.47  on 489  degrees of freedom
## Residual deviance: 469.08  on 478  degrees of freedom
## AIC: 493.08
##
## Number of Fisher Scoring iterations: 4
```

```
cm <- confusionMatrix(data = predict(train_glm, test_set),
                reference = test_set$Loan_Status)

metric[2,2:4] <-round(cm$byClass[c("Sensitivity","Specificity","F1")],3)
metric[2,1] <-round(cm$overall["Accuracy"],3)
metric[1:2,]
```

```
##                                 Accuracy Sensitivity Specificity    F1
## Simple model with credit_history    0.847       0.513           1 0.678
## Logistic regression                 0.847       0.513           1 0.678
```

b)

```
train_glm_2 <- train(Loan_Status ~ .,
                     method="glm",
                     family = "binomial",
                     trControl = trainControl(method = "cv", number = 5),
                     data= train_set_2)

cm_2 <-confusionMatrix(data = predict(train_glm_2, test_set_2),
                reference = test_set_2$Loan_Status)

metric_2[2,2:4] <-round(cm_2$byClass[c("Sensitivity","Specificity","F1")],3)
metric_2[2,1] <-round(cm_2$overall["Accuracy"],3)
metric_2[1:2,]
```

```
##                                 Accuracy Sensitivity Specificity    F1
## Simple model with credit_history    0.798        0.41       0.976 0.561
## Logistic regression                 0.798        0.41       0.976 0.561
```

Logistic regression performance didn't improve w.r.t simple model, Credit_History is in both cases decisive feature

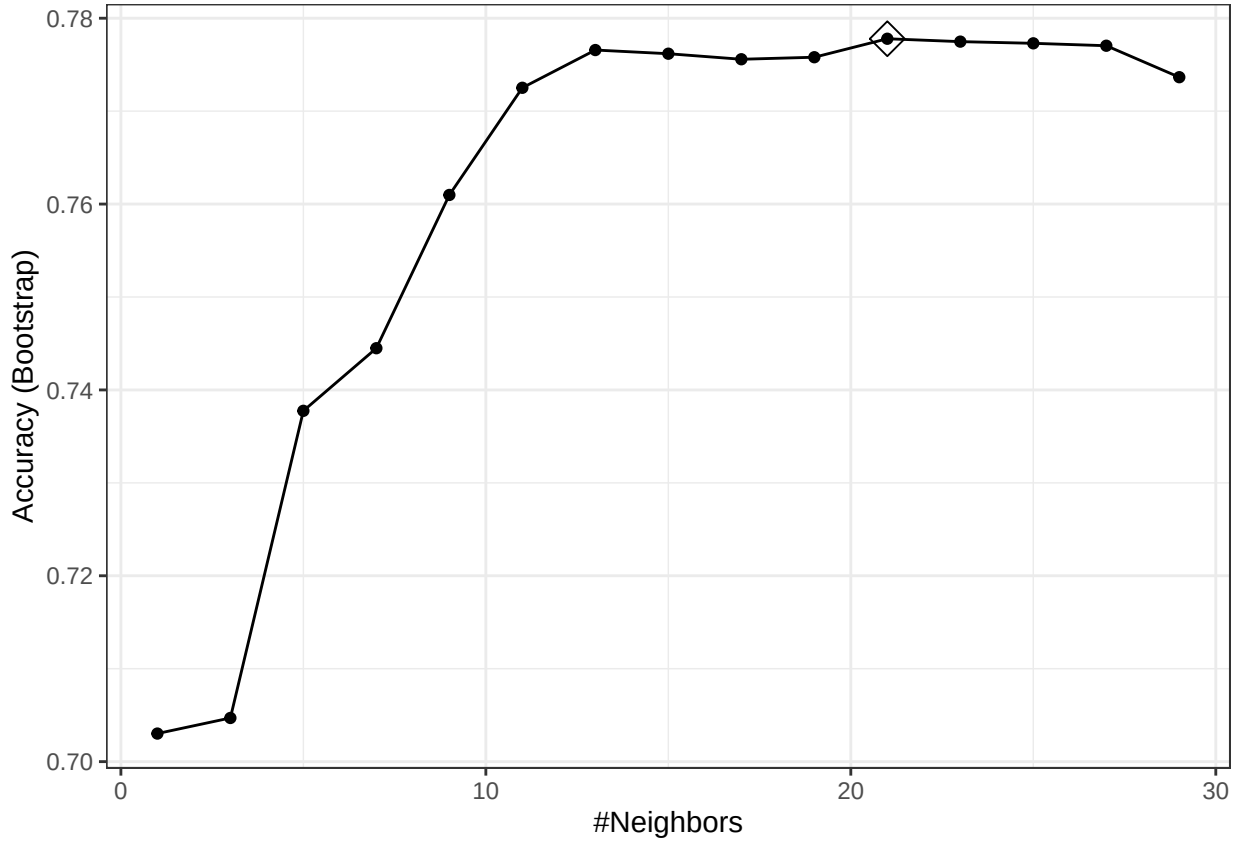
4.2.3. K-nearest neighbors

a)

```
set.seed(9, sample.kind = "Rounding")
train_knn <- train(Loan_Status ~ ., method="knn",
                   preProcess = c("center","scale"),
                   data= train_set,
                   tuneGrid=data.frame(k = seq(1, 30, 2)))
train_knn$bestTune
```

```
##     k
## 11 21
```

```
ggplot(train_knn, highlight = T)+
  theme_bw()
```




```
cm <- confusionMatrix(data = predict(train_knn, test_set, type="raw"),
                      reference = test_set$Loan_Status)

metric[3,2:4] <-round(cm$byClass[c("Sensitivity","Specificity","F1")],3)
metric[3,1] <-round(cm$overall["Accuracy"],3)
metric[1:3,]
```

```
##                                  Accuracy Sensitivity Specificity    F1
## Simple model with credit_history    0.847       0.513       1.000 0.678
## Logistic regression                 0.847       0.513       1.000 0.678
## kNN                                 0.831       0.513       0.976 0.656
```

b)

```
set.seed(9, sample.kind = "Rounding")
```

```
## Warning in set.seed(9, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

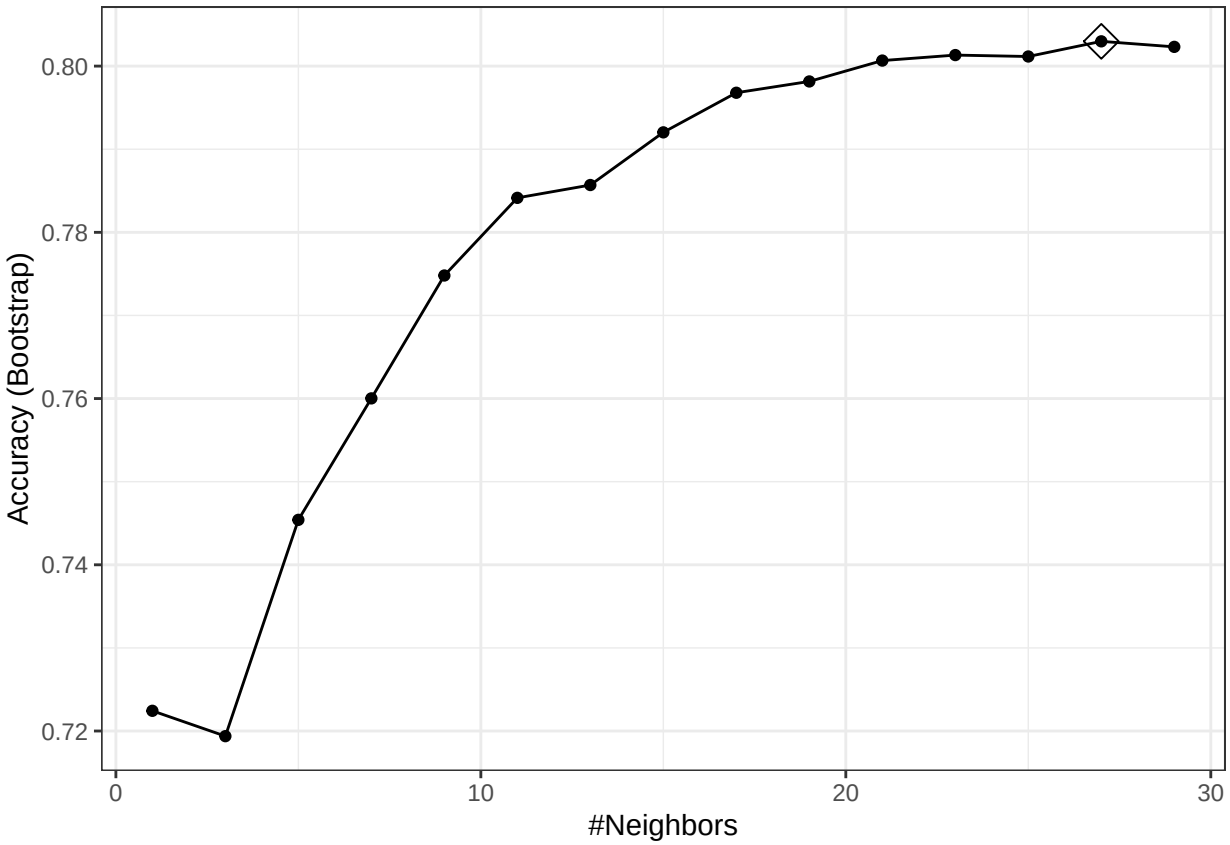
```
train_knn_2 <- train(Loan_Status ~ .,
                     method="knn",
                     data= train_set_2,
                     preProcess = c("center","scale"),
                     tuneGrid=data.frame(k = seq(1, 30, 2)))
```

```
## Warning in (function (kind = NULL, normal.kind = NULL, sample.kind = NULL) :
## non-uniform 'Rounding' sampler used
```

```
train_knn_2$bestTune
```

```
##     k
## 14 27
```

```
ggplot(train_knn_2, highlight = T)+
  theme_bw()
```




```
#The best performing model is 27-nearest neighbor model
```

```
cm_2 <- confusionMatrix(data = predict(train_knn_2, test_set_2, type="raw"),
                reference = test_set_2$Loan_Status)

metric_2[3,2:4] <-round(cm_2$byClass[c("Sensitivity","Specificity","F1")],3)
metric_2[3,1] <-round(cm_2$overall["Accuracy"],3)
metric_2[1:3,]
```

```
##                                  Accuracy Sensitivity Specificity    F1
## Simple model with credit_history    0.798       0.410       0.976 0.561
## Logistic regression                 0.798       0.410       0.976 0.561
## kNN                                 0.790       0.385       0.976 0.536
```

In both cases kNN models perform somewhat worse than logistic regression. Even after tuning we haven't been able to increase none of our selected performance measures (accuracy, sensitivity, specificity, F1) above those of logistic regression.

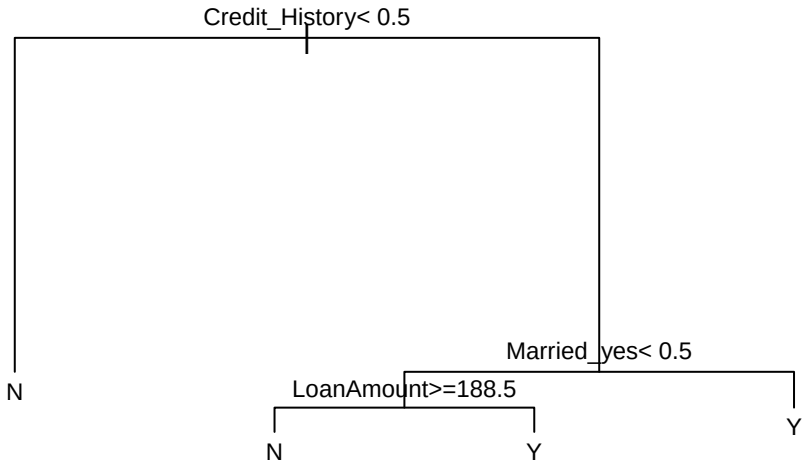
4.2.4. Decision tree

Decision trees form predictions by calculating which class is the most common among the training set observations within the partition. They are easy to interpret and visualize and they can model human decision processes, which loan approval could be seen as. However the approach via recursive partitioning can easily over-train.

a)

```
fit <- rpart(Loan_Status ~ ., data = train_set)

# visualize the splits
plot(fit, margin=0.1)
text(fit, cex = 0.75)
```




```
pred <- predict(fit, test_set, type="class")%>%factor()
cm<-confusionMatrix(pred, test_set$Loan_Status)

metric[4,2:4] <- round(cm$byClass[c("Sensitivity","Specificity","F1")],3)
metric[4,1]<-round(cm$overall["Accuracy"],3)
metric[1:4,]
```

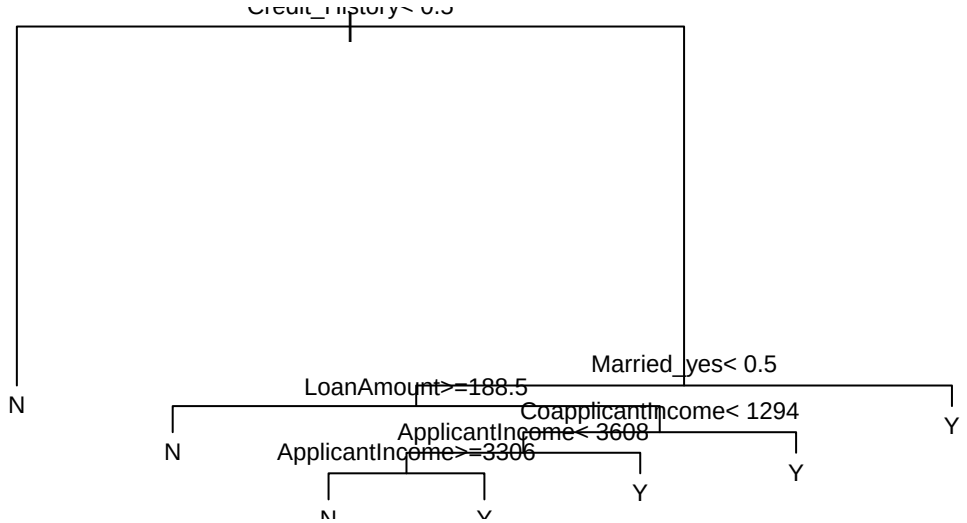
```
##                                 Accuracy Sensitivity Specificity    F1
## Simple model with credit_history    0.847       0.513       1.000 0.678
## Logistic regression                 0.847       0.513       1.000 0.678
## kNN                                 0.831       0.513       0.976 0.656
## Decision tree                       0.823       0.513       0.965 0.645
```

First decision tree splits at marriage status and loan amount, in addition to credit history. However, it's performance is lower compared to previous models.

We will now use cross validation to choose cp. We allow for very marginal improvement

```
train_rpart <- train(Loan_Status ~ .,
                     method = "rpart",
                     tuneGrid = data.frame(cp = seq(0.0, 0.007, len = 25)),
                     data = train_set)

plot(train_rpart$finalModel)
text(train_rpart$finalModel, cex = 0.75)
```




```
cm <- confusionMatrix(data = predict(train_rpart, test_set),
                      reference = test_set$Loan_Status)

metric[4,2:4] <- round(cm$byClass[c("Sensitivity","Specificity","F1")],3)
metric[4,1] <- round(cm$overall["Accuracy"],3)
metric[1:4,]
```

```
##                                 Accuracy Sensitivity Specificity    F1
## Simple model with credit_history    0.847       0.513       1.000 0.678
## Logistic regression                 0.847       0.513       1.000 0.678
## kNN                                 0.831       0.513       0.976 0.656
## Decision tree                       0.815       0.538       0.941 0.646
```

Adding complexity (additional splitting by coapplicant and applicant income) improves one important aspect of performance (sensitivity), while F1 stays similar and accuracy even lower. Among decision trees this one is chosen. Apart from sensitivity, it underperforms with respect to prevoius models (Simple, Logistic regression, kNN).

```
cm_tr <-confusionMatrix(data = predict(train_rpart, train_set),
                reference = train_set$Loan_Status)

round(cm_tr$byClass[c("Sensitivity","Specificity","F1")],3)
```

```
## Sensitivity Specificity          F1
##       0.529       0.947       0.643
```

```
round(cm_tr$overall["Accuracy"],3)
```

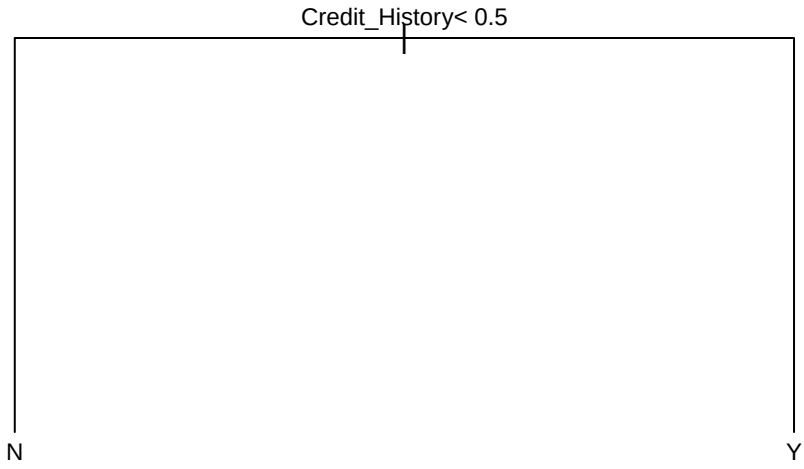
```
## Accuracy
##    0.816
```

```
# no signs of over-training
```

b)

```
fit <- rpart(Loan_Status ~ ., data = train_set_2)

# visualize the splits
plot(fit, margin=0.1)
text(fit, cex = 0.75)
```



First model gives us credit history as the only split. Not suprisingly, performance is similar to simple model using credit history only and logistic regression.

```
pred <- predict(fit, test_set_2, type="class")%>%factor()
cm_2<-confusionMatrix(pred, test_set_2$Loan_Status)

metric_2[4,2:4] <- round(cm_2$byClass[c("Sensitivity","Specificity","F1")],3)
metric_2[4,1] <- round(cm_2$overall["Accuracy"],3)
metric_2[1:4,]
```

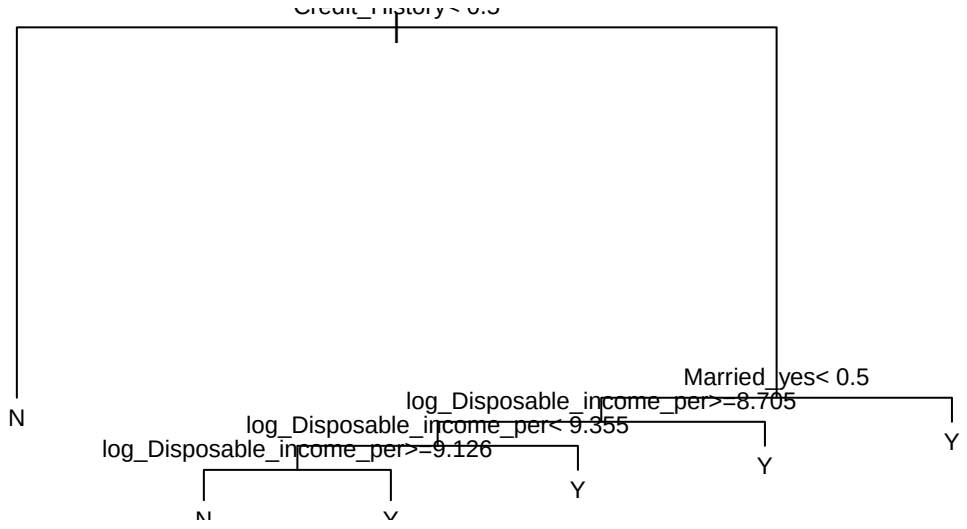
```
##                                  Accuracy Sensitivity Specificity    F1
## Simple model with credit_history    0.798       0.410       0.976 0.561
## Logistic regression                 0.798       0.410       0.976 0.561
## kNN                                 0.790       0.385       0.976 0.536
## Decision tree                       0.798       0.410       0.976 0.561
```

We have tried the most common parameters used for partition decision: the complexity parameter (cp) and the minimum number of observations required in a partition before partitioning it further (minsplit in the rpart package). The latter shows no difference compared to the baseline so only cp tuning is presented

We will now use cross validation to choose cp. We allow for very marginal improvement

```
train_rpart_2 <- train(Loan_Status ~ .,
                       method = "rpart",
                       tuneGrid = data.frame(cp = seq(0.0, 0.007, len = 25)),
                       data = train_set_2)

plot(train_rpart_2$finalModel)
text(train_rpart_2$finalModel, cex = 0.75)
```



```r
cm_2 <- confusionMatrix(data = predict(train_rpart_2, test_set_2),
                reference = test_set_2$Loan_Status)

round(cm_2$byClass[c("Sensitivity","Specificity","F1")],3)
```

```
## Sensitivity Specificity          F1
##       0.410       0.953       0.542
```

```r
round(cm_2$overall["Accuracy"],3)
```

```
## Accuracy
##    0.782
```

Even by increasing it's complexity, model performance is not good.

5. Conclusion

This project is based on publicly available Kaggle dataset from a company that wants to automate the loan eligibility process. Dataset consists of 614 observations, outcome (Loan_Status) and 12 features. Some of them (apart from loanID, also gender, employment status . . .) seem not to be decisive for the loan approval, while on the other hand credit history is the most important feature across all the models analyzed.

Data was trained with four different models:

1. simple model using credit history as the only predictor
2. logistic regression
3. kNN
4. decision tree

All the accuracies of predicted outcomes w.r.t. true values in the test set were well above 80% for not engineered dataset - cases (a). However, all the models were somehow weak in sensitivity (predicted loan granted when true value is loan not granted). Due to low prevalance this didn't affect accuracies as much, while computed balanced accuracies and F1 scores were much lower.

Feature engineering has also been performed and models based on the modified dataset evaluated, but their performance didn't improve compared to the original dataset, where only imputations of missing values have been treated.

	Accuracy	Sensitivity	Specificity	F1
Simple model with credit_history	0.847	0.513	1.000	0.678
Logistic regression	0.847	0.513	1.000	0.678
kNN	0.831	0.513	0.976	0.656
Decision tree	0.815	0.538	0.941	0.646

Final choice of model isn't staightforward. Decision tree improves on sensitivity compared to logistic regression. Some types of errors are more costly than the others, and granting a loan to someone that is unable to repay is one of them. On the other hand, decision trees are not very flexible and highly unstable to changes in training data. Given that and the fact that logistic regression outperformed decision tree in terms of

accuracy and F1, I have finally chosen logistic regression. Its performance metrics are identical to those of the simple model that uses credit history as the only predictor, as credit history is by far the most predictive variable out of all the available features, but marriage status seems to have statistically significant effect as well and its flexiblity could prove important in case additional observations could be obtained.

In the future random forest, which improves some of the shortcomings of decision trees, should be evaluated, as well as some tunings of logistic regression in direction of putting more weight to sensitivity could be performed. Yet, it is also quite likely that not all the important features are available in the dataset. Since by far the most frequent loan amount term in the dataset is 30 years (360 months), applicants' age could be an important feature, besides other properties owned by applicants, type of job contract (permanent/temporary), to name just few. There are also numerous outliers that might be useful to treat, eventhough our dataset isn't large. Generally speaking, having been able to have more observations would most probably improve model performance.