Loan Prediction Project

George Moisescu

Introduction

In this project I will try to use 6 machine learnig algorithm for predicting which individuals included in the test set database will have or will not have access to credit. I will use the data from Kaggle platform: https://www.kaggle.com/altruistdelhite04/loan-prediction-problem-dataset

```
head(loan_data,4)
```

```
##    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
##   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
##   Loan_Status
## 1           Y
## 2           N
## 3           Y
## 4           Y
```

```
str(loan_data)
```

```
## '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" ...
```

The dataset contain 614 observations in 13 variables. I will drop the Loan_ID variable because it is not useful for building the prediction algorithm .

Describing the variables

1. Gender: the gender of applicant(male or female)
2. Married: if the applicant is married or single
3. Dependents: the number of dependents the applicant have(0,1,2,3+)
4. Education: the studies of the applicant(Graduate or Not Graduate)
5. Self_Employed: if the applicant is employed or self_employed
6. ApplicantIncome: integer variable who tell us the applicant income
7. CoapplicantIncome: numeric variable who tell us the coapplicant income
8. LoanAmount: the amount of the loan
9. Loan_Amount_Term: the period in which the loan is given
10. Credit_History: if the applicant has history credit
11. Property_Area: the place where the property of the applicant is situated 12: Loan_Status: status of the application.

Exploratory Data Analysis

As we can see the type of variables are character, integer and numeric. I will transform the data into factors and integer.

```r
loan_data$Gender <- as.factor(loan_data$Gender)
loan_data$Married <- as.factor(loan_data$Married)
loan_data$Dependents <- as.factor(loan_data$Dependents)
loan_data$Education <- as.factor(loan_data$Education)
loan_data$Self_Employed <- as.factor(loan_data$Self_Employed)
loan_data$CoapplicantIncome <- as.integer(loan_data$CoapplicantIncome)
loan_data$Credit_History <- as.factor(loan_data$Credit_History)
loan_data$Property_Area <- as.factor(loan_data$Property_Area)
loan_data$Loan_Status <- as.factor(loan_data$Loan_Status)
loan_data$Loan_Amount_Term <- as.factor(loan_data$Loan_Amount_Term)
```

Summarising the dataset:

```r
summary(loan_data)
```

```
##    Loan_ID             Gender    Married   Dependents        Education
##  Length:614              : 13       :  3     : 15     Graduate    :480
##  Class :character   Female:112   No :213   0 :345     Not Graduate:134
##  Mode  :character   Male  :489   Yes:398   1 :102
##                                            2 :101
##                                            3+: 51
##
##
##  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
##  360    :512      0    : 89       Rural     :179   N:192
##  180    : 44      1    :475       Semiurban:233   Y:422
##  480    : 15      NA's: 50        Urban     :202
##  300    : 13
##  84     :  4
##  (Other): 12
##  NA's   : 14
```

There are missing values in the Gender, Married, Dependents, Self_Employed, Credit_History variables.

Dealing with the missing values in the loan_data

Gender variable

```
gender_data <- loan_data %>% filter(Gender != "") %>% group_by(Gender) %>%
          summarize(n = n()) %>% mutate(percentage = round(n/sum(n),digits = 2))
gender_data
```

```
## # A tibble: 2 x 3
##   Gender     n percentage
##   <fct>  <int>      <dbl>
## 1 Female   112       0.19
## 2 Male     489       0.81
```

We have 19% female applicants and 81% male applicants

```
missing <- loan_data %>% filter(Gender == "")
nrow(missing)
```

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

We have 13 applicants with no value for Gender. 19% of 13 is 2. I will fill 2 of 13 with female and 11 of 13 with male

```
gender_index <- which(loan_data$Gender == "")
gender_index
```

```
##  [1]  24 127 172 189 315 335 461 468 478 508 577 589 593
```

```
loan_data$Gender[c(172,461)] <- "Female"

gender_index_male <-setdiff(gender_index,c(172,461))
gender_index_male
```

```
##  [1]  24 127 189 315 335 468 478 508 577 589 593
```

```r
for (i in gender_index_male) {
  loan_data$Gender[i] <- "Male"
}

loan_data$Gender <- droplevels(loan_data$Gender)
summary(loan_data$Gender)
```

```
## Female   Male
##    114    500
```

```r
levels(loan_data$Gender)
```

```
## [1] "Female" "Male"
```

Married variable

```r
married_data <- loan_data %>% filter(Married != "") %>% group_by(Married) %>%
          summarize(n = n()) %>% mutate(percentage = round(n/sum(n),digits = 2))
married_data
```

```
## # A tibble: 2 x 3
##   Married     n percentage
##   <fct>   <int>      <dbl>
## 1 No        213       0.35
## 2 Yes       398       0.65
```

We have 65% married applicants and 35% single applicants

```r
summary(loan_data$Married)
```

```
##      No Yes
##   3 213 398
```

We have 3 missing values.

```r
married_index <- which(loan_data$Married == "")
married_index
```

```
## [1] 105 229 436
```

```r
round(0.65 * length(married_index))
```

```
## [1] 2
```

I will fill 2 observations with “Yes” and one with “No”

```
loan_data$Married[c(105,229)] <- "Yes"
loan_data$Married[c(436)] <- "No"

loan_data$Married <- droplevels(loan_data$Married)
summary(loan_data$Married)
```

```
##  No Yes
## 214 400
```

Dependents variable

```
summary(loan_data$Dependents)
```

```
##       0   1   2  3+
##  15 345 102 101  51
```

```
dependents_data <- loan_data %>% filter(Dependents != "") %>%
                    group_by(Dependents) %>% summarize(n = n()) %>%
                    mutate(percentage = round(n/sum(n),digits = 2))
dependents_data
```

```
## # A tibble: 4 x 3
##   Dependents     n percentage
##   <fct>      <int>      <dbl>
## 1 0            345       0.58
## 2 1            102       0.17
## 3 2            101       0.17
## 4 3+            51       0.09
```

We have 58% applicants with no dependent, 17% with 1 dependent, 17% with 2 dependents and 9% with 3 or more dependents We have 15 applicants with no value for Dependents.

```
dependents_index <- which(loan_data$Dependents == "")
dependents_index
```

```
##  [1] 103 105 121 227 229 294 302 333 336 347 356 436 518 572 598
```

```
round(0.58 * length(dependents_index))
```

```
## [1] 9
```

I will fill 9 observation with 0, 3 observations with 1 and 3 with 2.

```
loan_data$Dependents[c(103,227,229,302,336,356,436,572,598)] <- 0

round(0.17 * length(dependents_index))
```

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

```
loan_data$Dependents[c(105,121,294)] <- 1
loan_data$Dependents[c(333,347,518)] <- 2
```

```
loan_data$Dependents <- droplevels(loan_data$Dependents)
summary(loan_data$Dependents)
```

```
##   0   1   2  3+
## 354 105 104  51
```

Self_Employed variable

```
summary(loan_data$Self_Employed)
```

```
##      No Yes
##  32 500  82
```

```
selfemp_data <- loan_data %>% filter(Self_Employed != "") %>% group_by(Self_Employed) %>% summarize(n =
selfemp_data
```

```
## # A tibble: 2 x 4
##   Self_Employed     n avg_income percentage
##   <fct>         <int>      <dbl>      <dbl>
## 1 No              500      5050.       0.86
## 2 Yes              82      7381.       0.14
```

We have 86% applicants who are not self_employed and 14% who are self_employed The average income for self_employed applicant is 7381 and for those who employed is 5050 We have 32 applicants with no value for Self_Employed.

```
selfemp_index <- which(loan_data$Self_Employed == "")
selfemp_index
```

```
##  [1]  12  20  25  30  31  96 108 112 115 159 171 219 232 237 269 296 334 337 345
## [20] 375 381 386 412 433 448 464 469 536 543 580 601 602
```

```
loan_data$ApplicantIncome[selfemp_index]
```

```
##  [1]  2500  2600  3717  3750  4166  6782  7333  2929  5050  2980  1820  5000
## [13]  3716  5746  3418  4416 63337  5250  2583  2764  3333  3667  6256 12876
## [25]  3539  5191   210  2550  3652  3182   416  2894
```

```
round(0.14 * length(selfemp_index))
```

```
## [1] 4
```

We choose 4 applicant out of 32 with income close to average income of the category.

```
loan_data$Self_Employed[c(96,108,334,433)] <- "Yes"
```

The rest of 28 observation I fill with “No”

```
selfemp_index_no <-setdiff(selfemp_index,c(96,108,334,433))
selfemp_index_no
```

```
##  [1]  12  20  25  30  31 112 115 159 171 219 232 237 269 296 337 345 375 381 386
## [20] 412 448 464 469 536 543 580 601 602
```

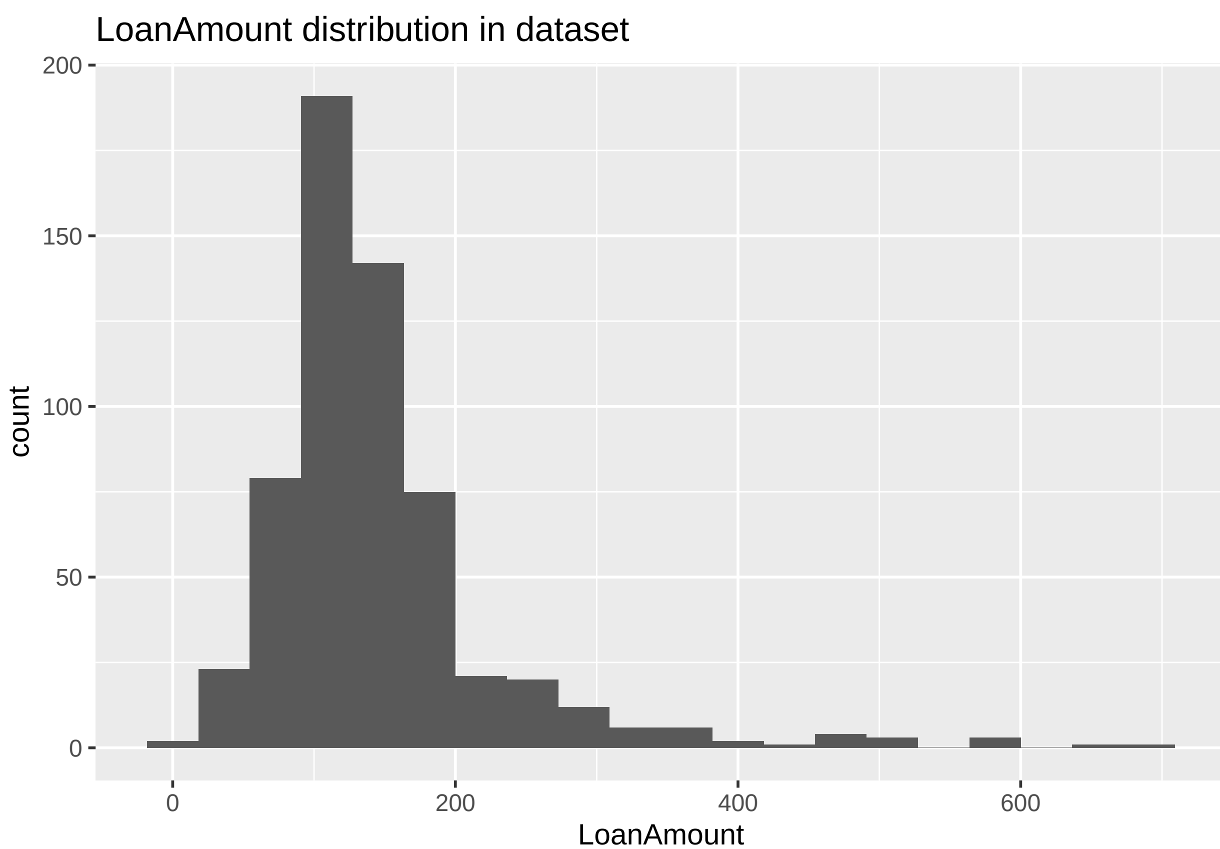
```
for (i in selfemp_index_no) {
  loan_data$Self_Employed[i] <- "No"
}

loan_data$Self_Employed <- droplevels(loan_data$Self_Employed)
summary(loan_data$Self_Employed)
```

```
##  No Yes
## 528  86
```

LoanAmount variable

```
loan_data %>% filter(!is.na(LoanAmount)) %>% ggplot(aes(LoanAmount)) +
      geom_histogram(bins = 20) + ggtitle("LoanAmount distribution in dataset")
```



```
summary(loan_data$LoanAmount)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##     9.0   100.0   128.0   146.4   168.0   700.0      22
```

The distribution is right skewed and we have missing values. I will fill the NA values with median value.

```
median(loan_data$LoanAmount,na.rm = TRUE)
```

```
## [1] 128
```

```
loan_data$LoanAmount[is.na(loan_data$LoanAmount)] <-
                                      median(loan_data$LoanAmount,na.rm = TRUE)
```

Loan_Amount_Term variable

```
loanamountterm_data <- loan_data %>% filter(!is.na(Loan_Amount_Term)) %>%
                    group_by(Loan_Amount_Term) %>% summarize(n = n()) %>%
                    mutate(percentage = round(n/sum(n),digits = 2))
loanamountterm_data
```

```
## # A tibble: 10 x 3
##    Loan_Amount_Term     n percentage
##    <fct>            <int>      <dbl>
##  1 12                   1       0
##  2 36                   2       0
##  3 60                   2       0
##  4 84                   4       0.01
##  5 120                  3       0
##  6 180                 44       0.07
##  7 240                  4       0.01
##  8 300                 13       0.02
##  9 360                512       0.85
## 10 480                 15       0.03
```

I am going to fill the NA's based on the proportion of each category.

```
loan_data %>% filter(is.na(Loan_Amount_Term)) %>%nrow()
```

```
## [1] 14
```

```
loanamountterm_index <- which(is.na(loan_data$Loan_Amount_Term))
loanamountterm_index
```

```
##  [1]  20  37  45  46  74 113 166 198 224 233 336 368 422 424
```

```r
round(0.85 * 14)
```

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

```r
loan_data$Loan_Amount_Term[c(20,37,45,46,74,113,166,198,224,233,336,368)] <- 360

round(0.07 * 14)
```

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

```r
loan_data$Loan_Amount_Term[422] <- 180
loan_data$Loan_Amount_Term[424] <- 480
```

I filled 12 of 14 observations with 360, 1 observation with 180 and 1 with 480

```r
summary(loan_data$Loan_Amount_Term)
```

```
##  12  36  60  84 120 180 240 300 360 480
##   1   2   2   4   3  45   4  13 524  16
```

Credit_History variable

```r
summary(loan_data$Credit_History)
```

```
##    0    1 NA's
##   89  475   50
```

```r
credithistory_data <- loan_data %>% filter(!is.na(Credit_History)) %>%
                      group_by(Credit_History) %>% summarize(n = n()) %>%
                      mutate(percentage = round(n/sum(n),digits = 2))
credithistory_data
```

```
## # A tibble: 2 x 3
##   Credit_History     n percentage
##   <fct>          <int>      <dbl>
## 1 0                 89       0.16
## 2 1                475       0.84
```

We have 50 observations with missing values, 16% of applicants witch don't have credit history and 84% of applicant witch have credit history. I will fill the NA values based on proportion of each category.

```r
credithistory_index <- which(is.na(loan_data$Credit_History))
credithistory_index
```

```
##  [1]  17  25  31  43  80  84  87  96 118 126 130 131 157 182 188 199 220 237 238
## [20] 260 261 280 310 314 318 319 324 349 364 378 393 396 412 445 450 452 461 474
## [39] 491 492 498 504 507 531 534 545 557 566 584 601
```

```
round(0.16 * 50)
```

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

```
credithistory_index_no <- c(17,43,96,130,220,260,318,393)
loan_data$Credit_History[credithistory_index_no] <- 0

credithistory_index_yes <- setdiff(credithistory_index,credithistory_index_no)
credithistory_index_yes
```

```
##  [1]  25  31  80  84  87 118 126 131 157 182 188 199 237 238 261 280 310 314 319
## [20] 324 349 364 378 396 412 445 450 452 461 474 491 492 498 504 507 531 534 545
## [39] 557 566 584 601
```

```
for (i in credithistory_index_yes) {
  loan_data$Credit_History[i] <- 1
}
```

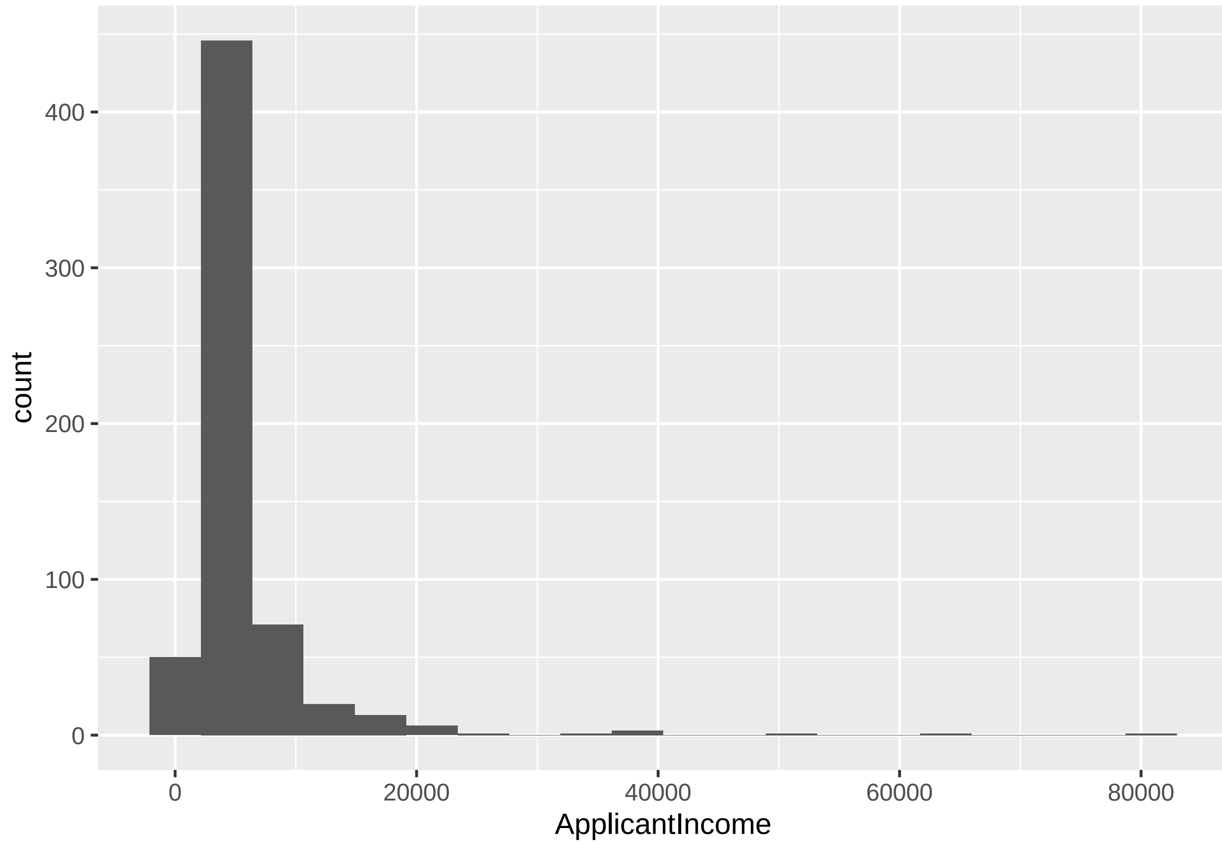
```
loan_data <- loan_data[,-1]
```

Dealing with outliers

I use Tukey's definition of an outlier

Removing outliers for ApplicantIncome

```
loan_data %>% ggplot(aes(ApplicantIncome)) + geom_histogram(bins = 20)
```




```
summary(loan_data$ApplicantIncome)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     150    2878    3812    5403    5795   81000
```

I will keep the values witch are between lower and higher values.

```
cutt_off <- 1.5 * IQR(loan_data$ApplicantIncome)
cutt_off
```

```
## [1] 4376.25
```

```
lower_value <- quantile(loan_data$ApplicantIncome,0.25) - cutt_off
higher_Value <- quantile(loan_data$ApplicantIncome,0.75) + cutt_off
lower_value
```

```
##      25%
## -1498.75
```

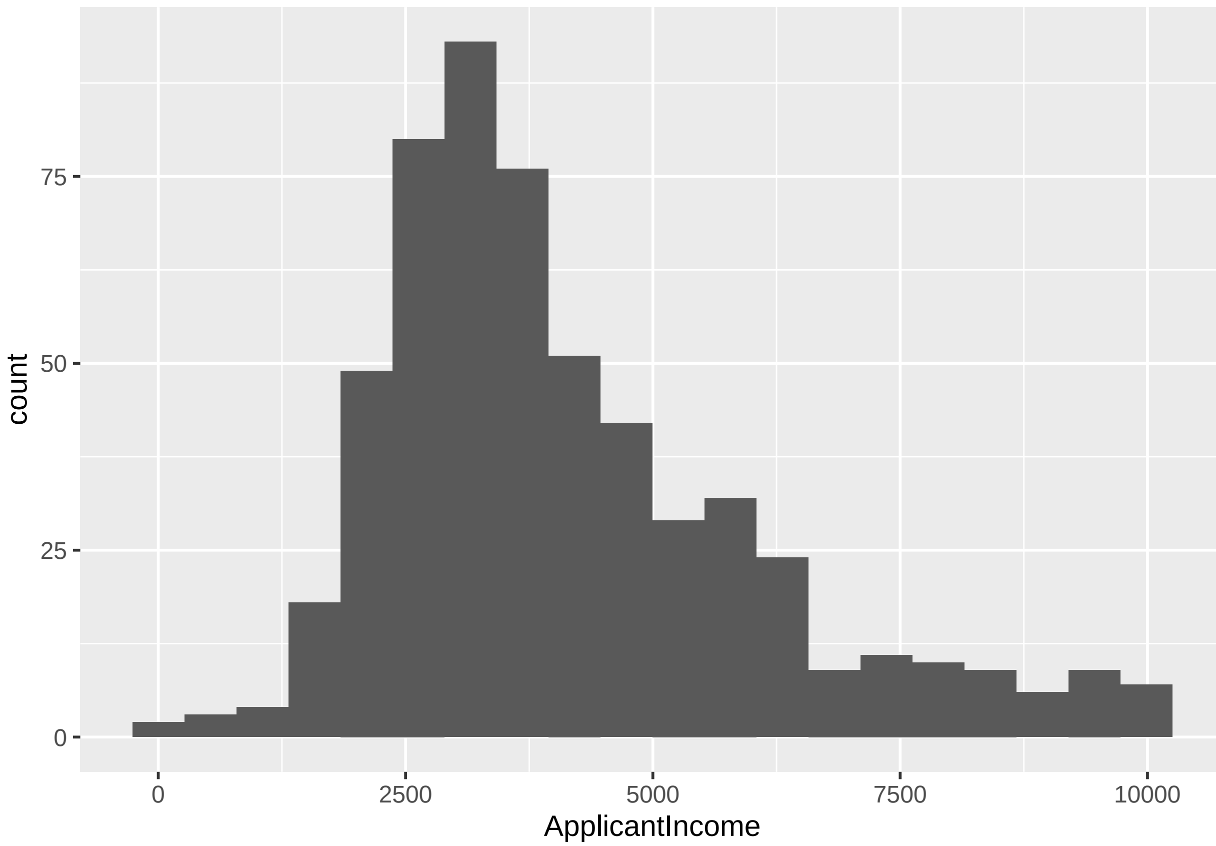
```
higher_Value
```

```
##      75%
## 10171.25
```

```r
appinc_dropindex <- which(loan_data$ApplicantIncome > higher_Value)
appinc_dropindex
```

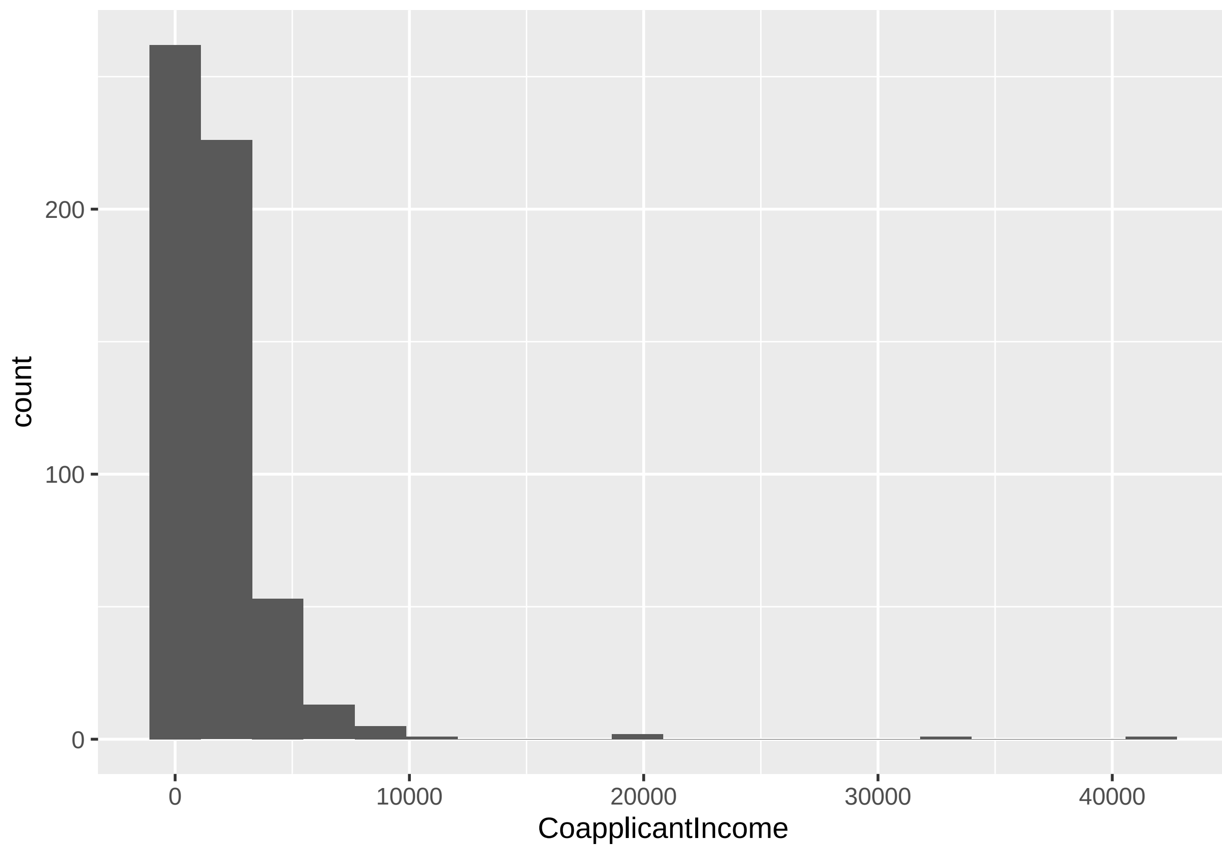
```
##  [1]  10  35  55  68 103 107 116 120 127 129 131 139 145 147 156 172 184 186 192
## [20] 200 255 259 272 279 285 309 325 334 370 371 410 425 433 439 444 468 476 479
## [39] 484 488 494 507 510 526 534 535 562 573 595 605
```

```r
loan_data <- loan_data[-appinc_dropindex,]
loan_data %>% ggplot(aes(ApplicantIncome)) + geom_histogram(bins = 20)
```



Removing outliers for CoapplicantIncome

```r
loan_data %>% ggplot(aes(CoapplicantIncome)) + geom_histogram(bins = 20)
```




```
summary(loan_data$CoapplicantIncome)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       0       0    1406    1692    2337   41667
```

I will keep the values witch are between lower and higher values.

```
cutt_off <- 1.5 * IQR(loan_data$CoapplicantIncome)
cutt_off
```

```
## [1] 3505.5
```

```
lower_value <- quantile(loan_data$CoapplicantIncome,0.25) - cutt_off
higher_Value <- quantile(loan_data$CoapplicantIncome,0.75) + cutt_off
lower_value
```

```
##     25%
## -3505.5
```

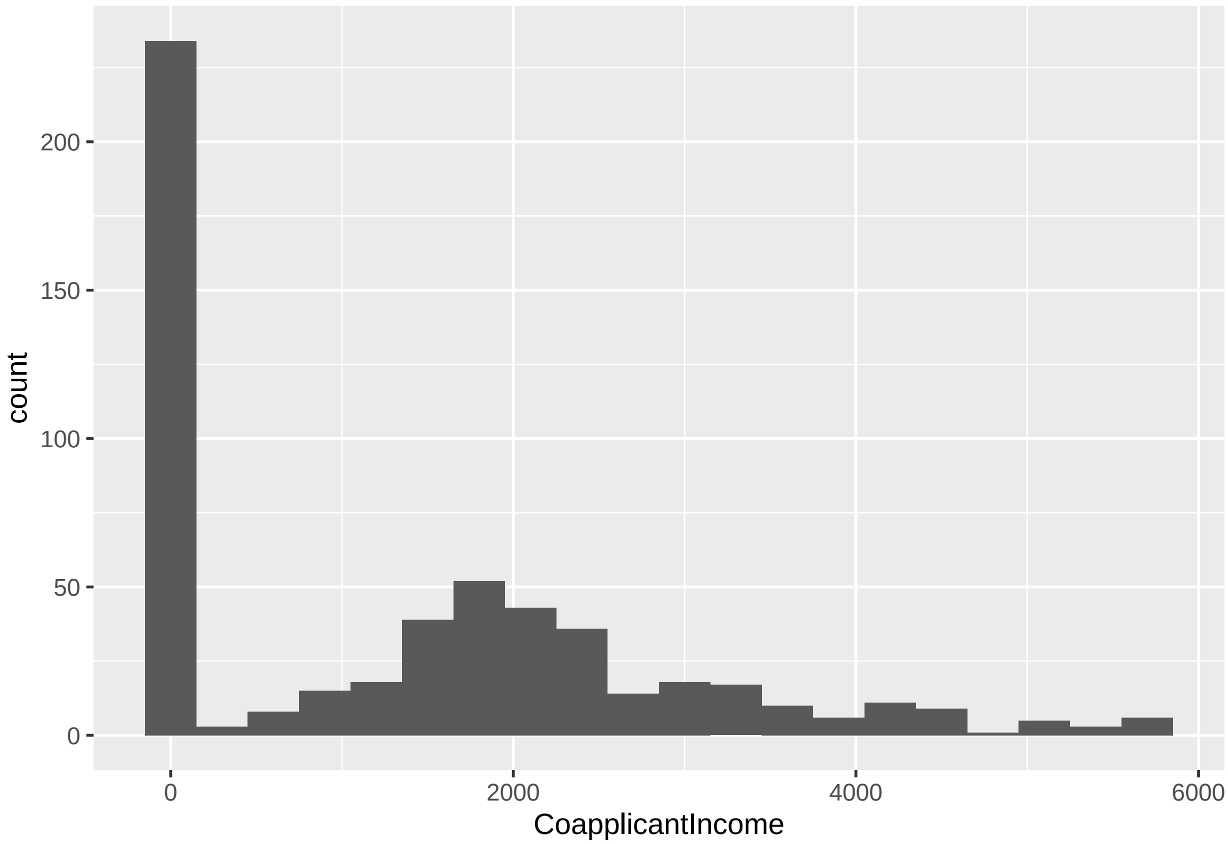
```
higher_Value
```

```
##    75%
## 5842.5
```

```r
coappinc_dropindex <- which(loan_data$CoapplicantIncome > higher_Value)
coappinc_dropindex
```

```
##  [1]  12  37 115 125 162 165 234 322 343 373 387 410 471 481 534 552
```

```r
loan_data <- loan_data[-coappinc_dropindex,]
loan_data %>% ggplot(aes(CoapplicantIncome)) + geom_histogram(bins = 20)
```



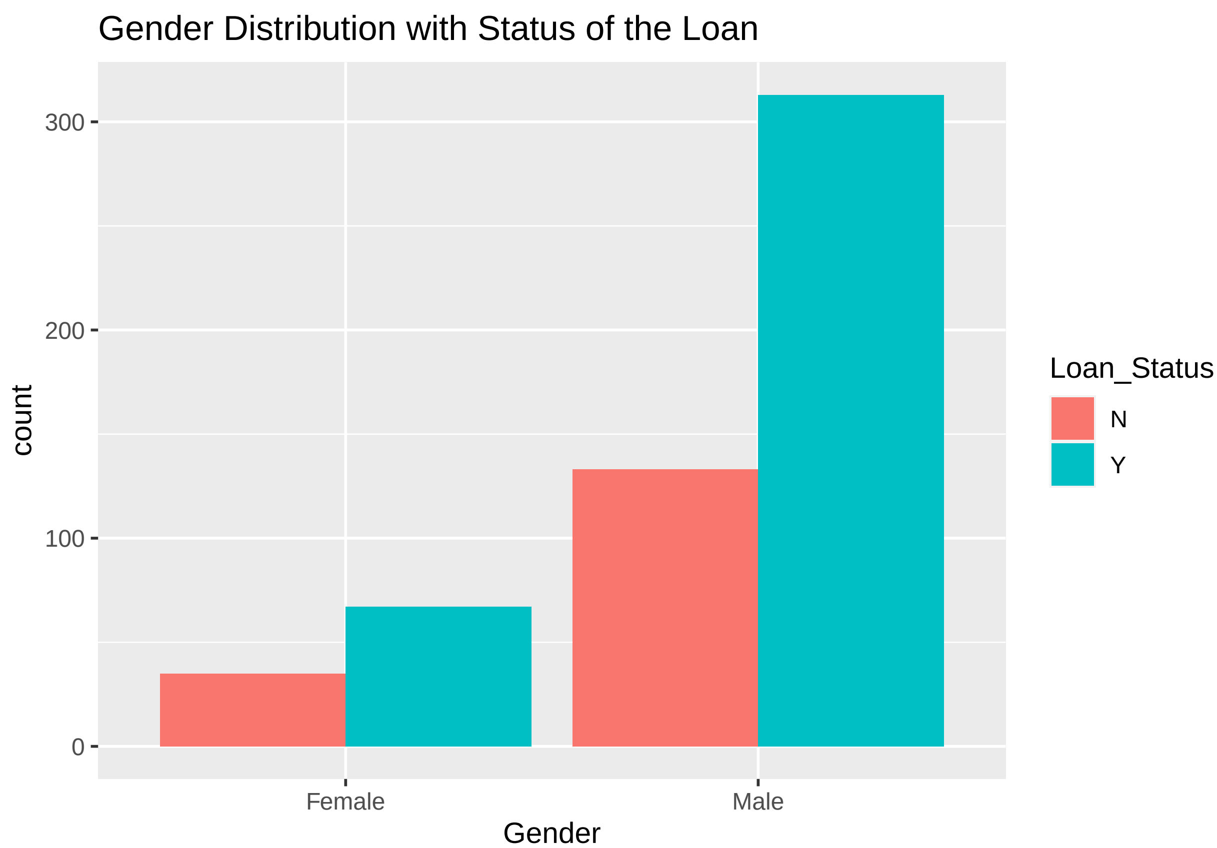
Grouping the ApplicantIncome and CoapplicantIncome

I will group the ApplicantIncome and CoapplicantIncome in a single variable TotalIncome.

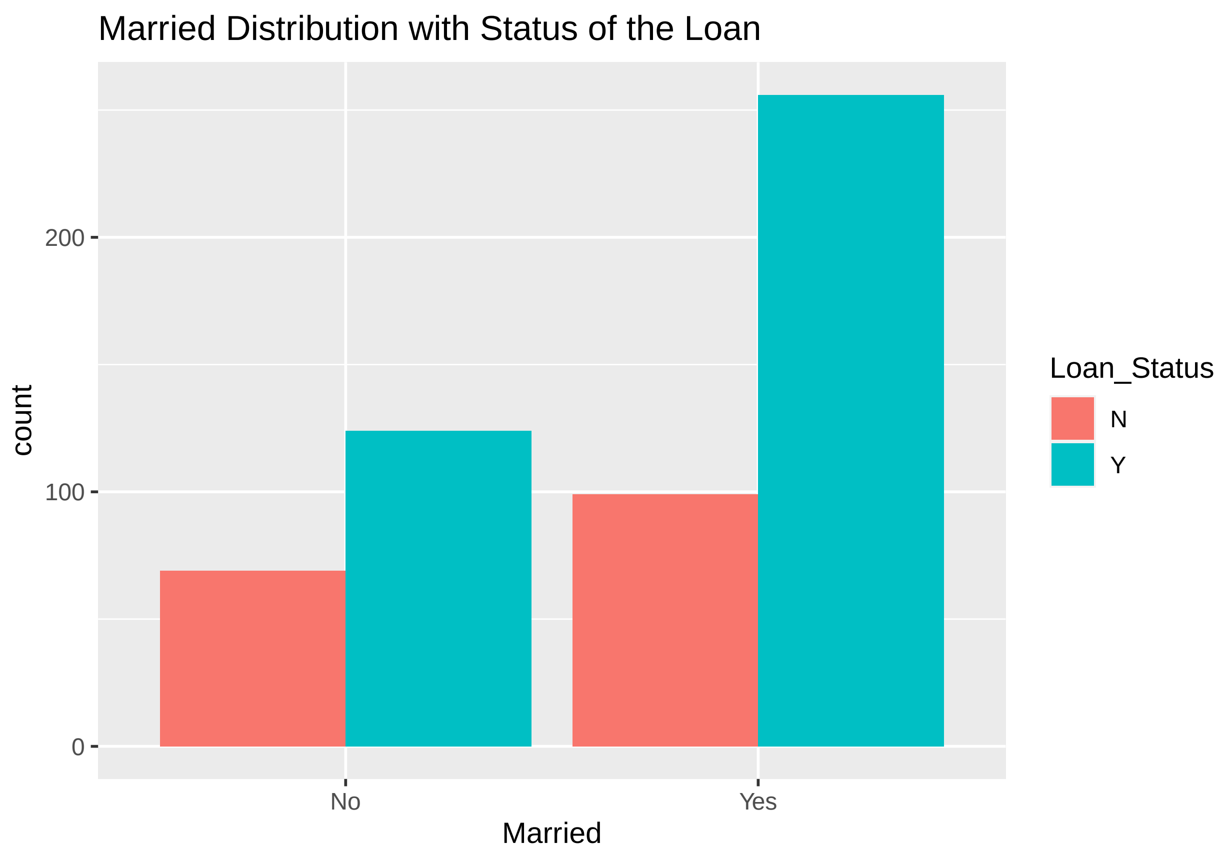
```r
loan_data <- loan_data %>% mutate(TotalIncome = ApplicantIncome + CoapplicantIncome)
loan_data <- loan_data[,-c(6,7)]
loan_data <- loan_data[,c(1:9,11,10)]
```

Data Visualisation

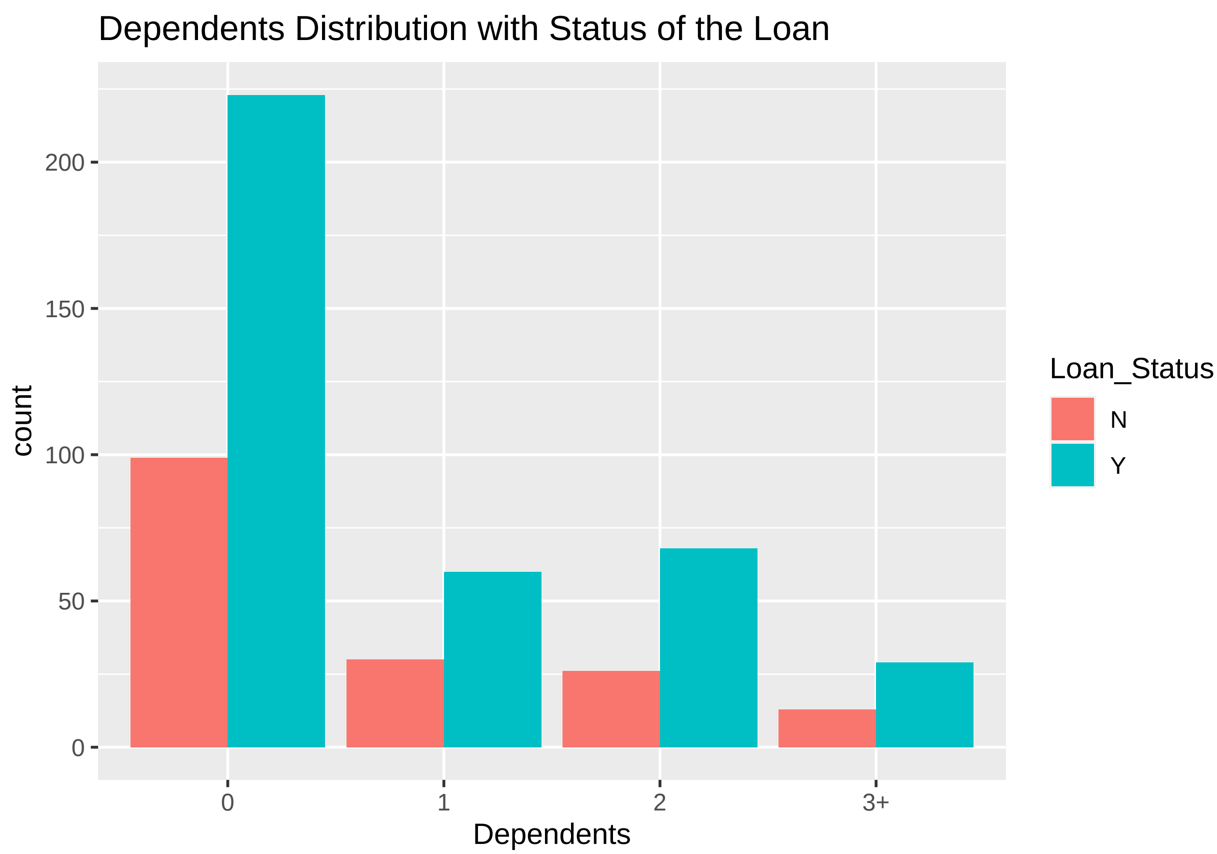
```r
loan_data %>% ggplot(aes(Gender, fill = Loan_Status)) +
                        geom_bar(position=position_dodge()) +
                        ggtitle("Gender Distribution with Status of the Loan")
```



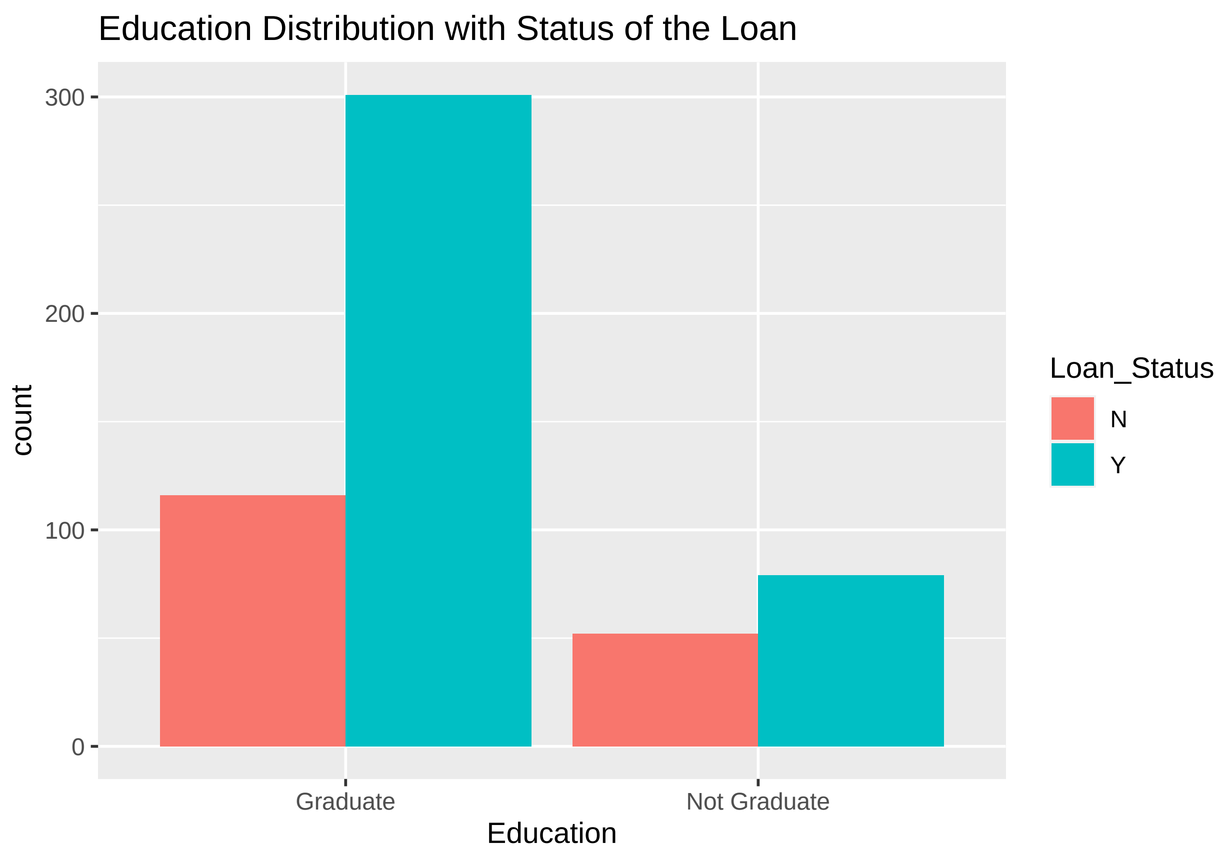

```
loan_data %>% ggplot(aes(Married, fill = Loan_Status)) +
                        geom_bar(position=position_dodge()) +
                        ggtitle("Married Distribution with Status of the Loan")
```




```
loan_data %>% ggplot(aes(Dependents, fill = Loan_Status)) +
                    geom_bar(position=position_dodge()) +
                    ggtitle("Dependents Distribution with Status of the Loan")
```

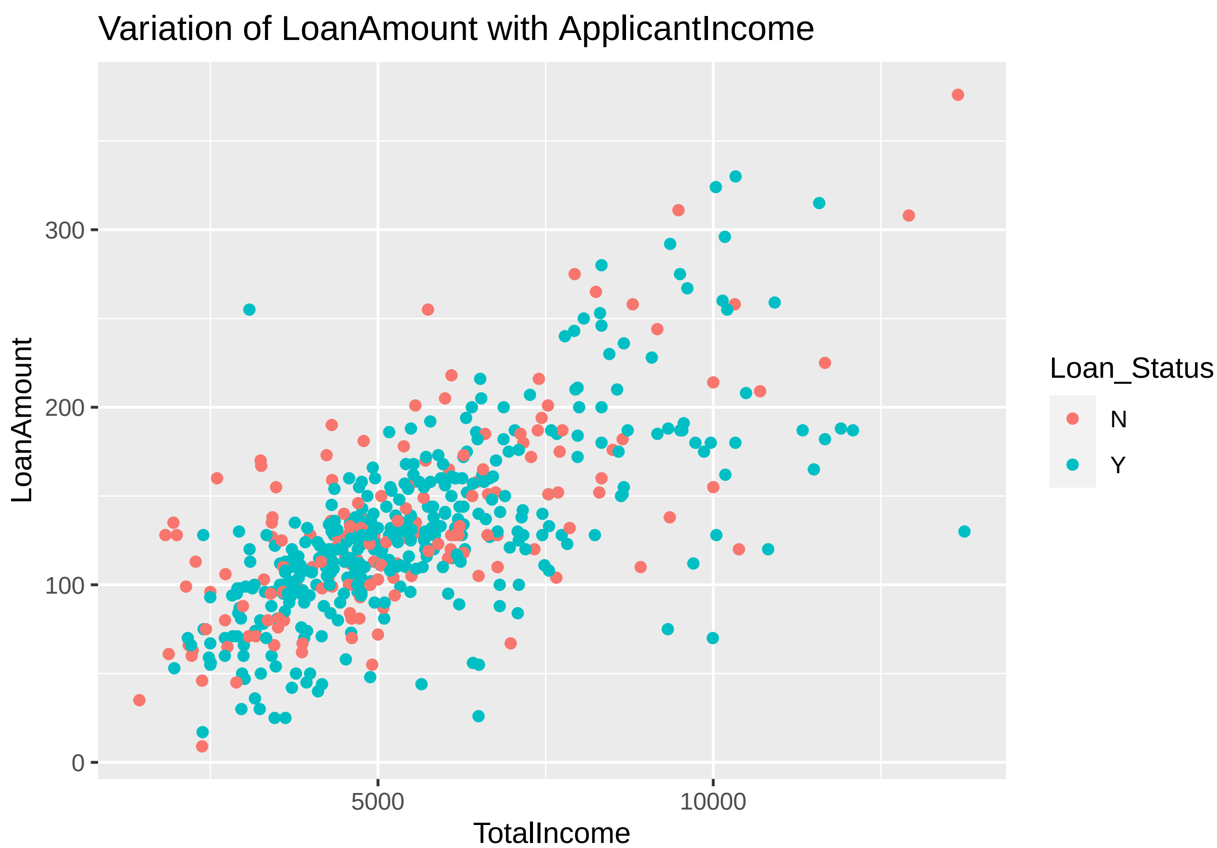



```
loan_data %>% ggplot(aes(Education, fill = Loan_Status)) +
                    geom_bar(position=position_dodge()) +
                    ggtitle("Education Distribution with Status of the Loan")
```

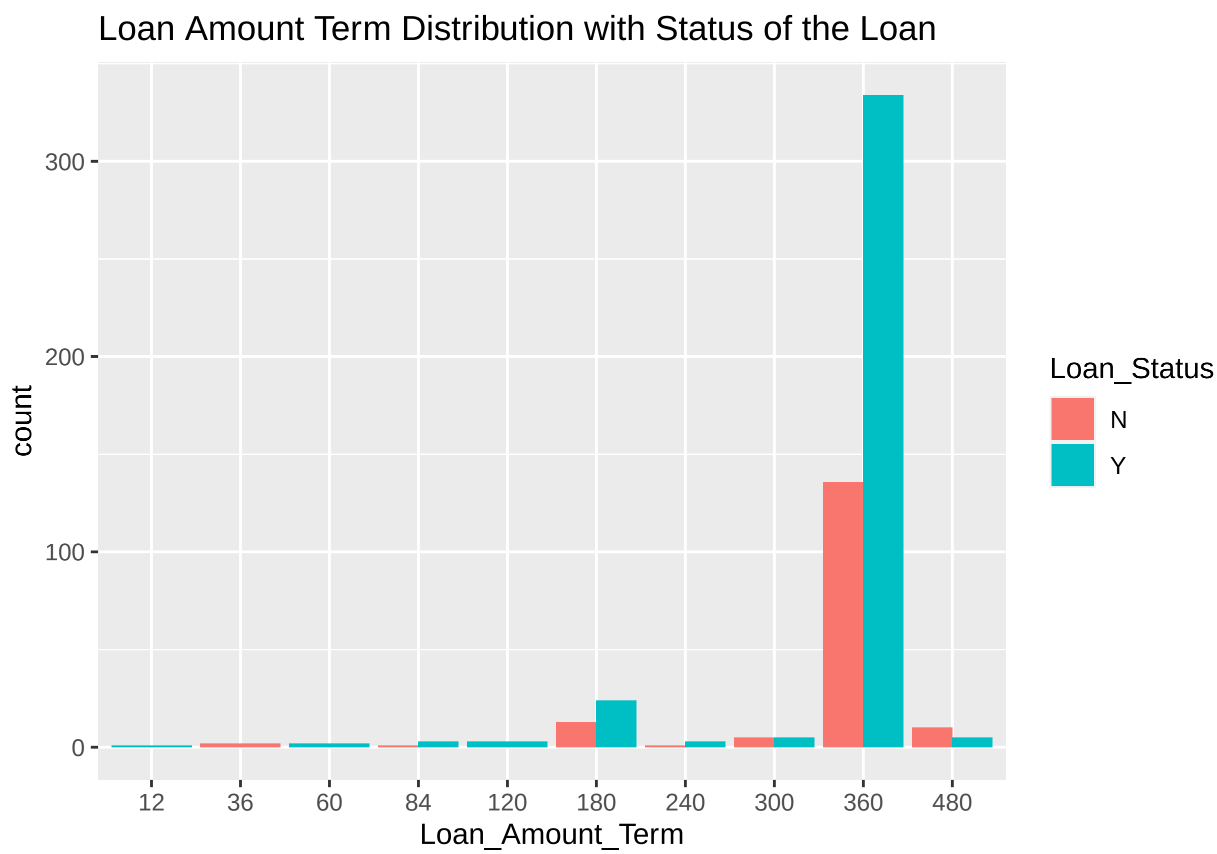



```r
loan_data %>% ggplot(aes(Self_Employed, fill = Loan_Status)) +
                  geom_bar(position=position_dodge()) +
                  ggtitle("Self Employed Distribution with Status of the Loan")
```

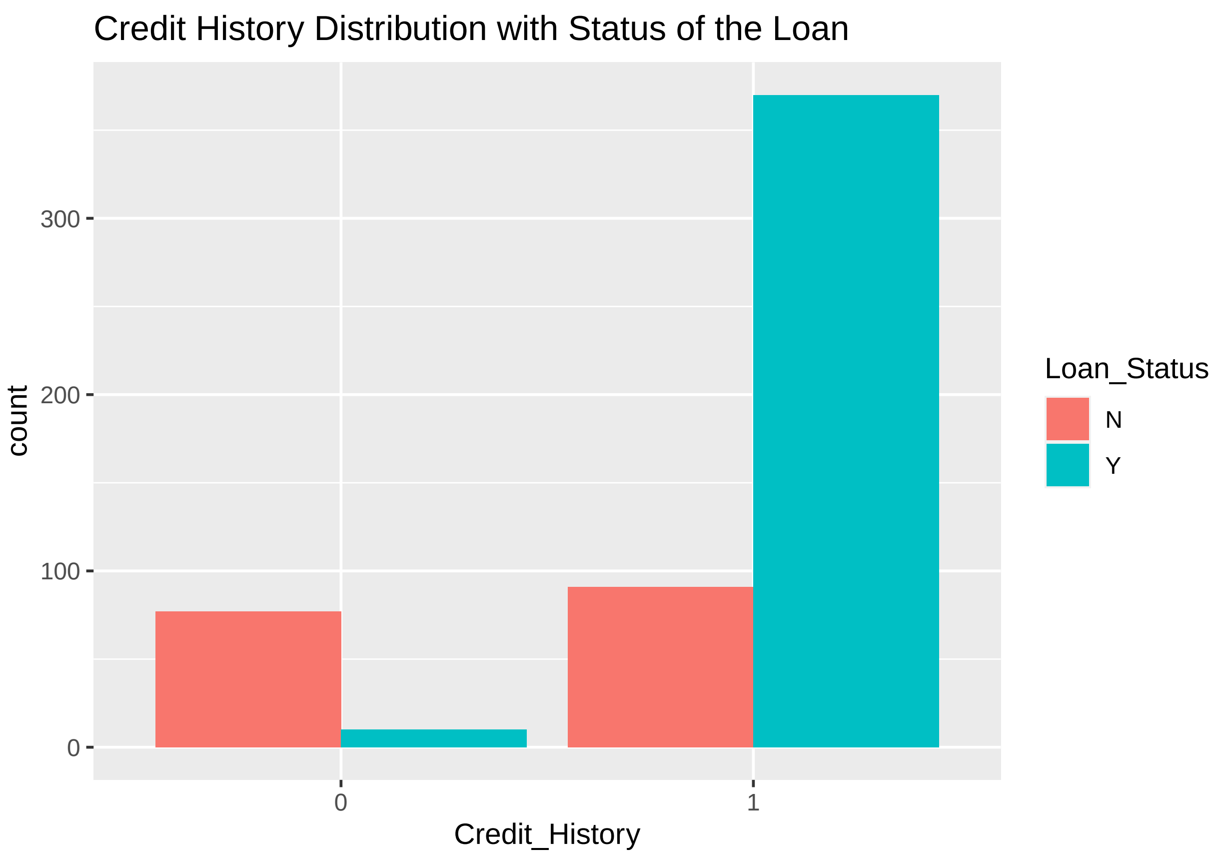

```
loan_data %>% ggplot(aes(TotalIncome,LoanAmount, color = Loan_Status)) +
          geom_point() + ggtitle("Variation of LoanAmount with ApplicantIncome")
```



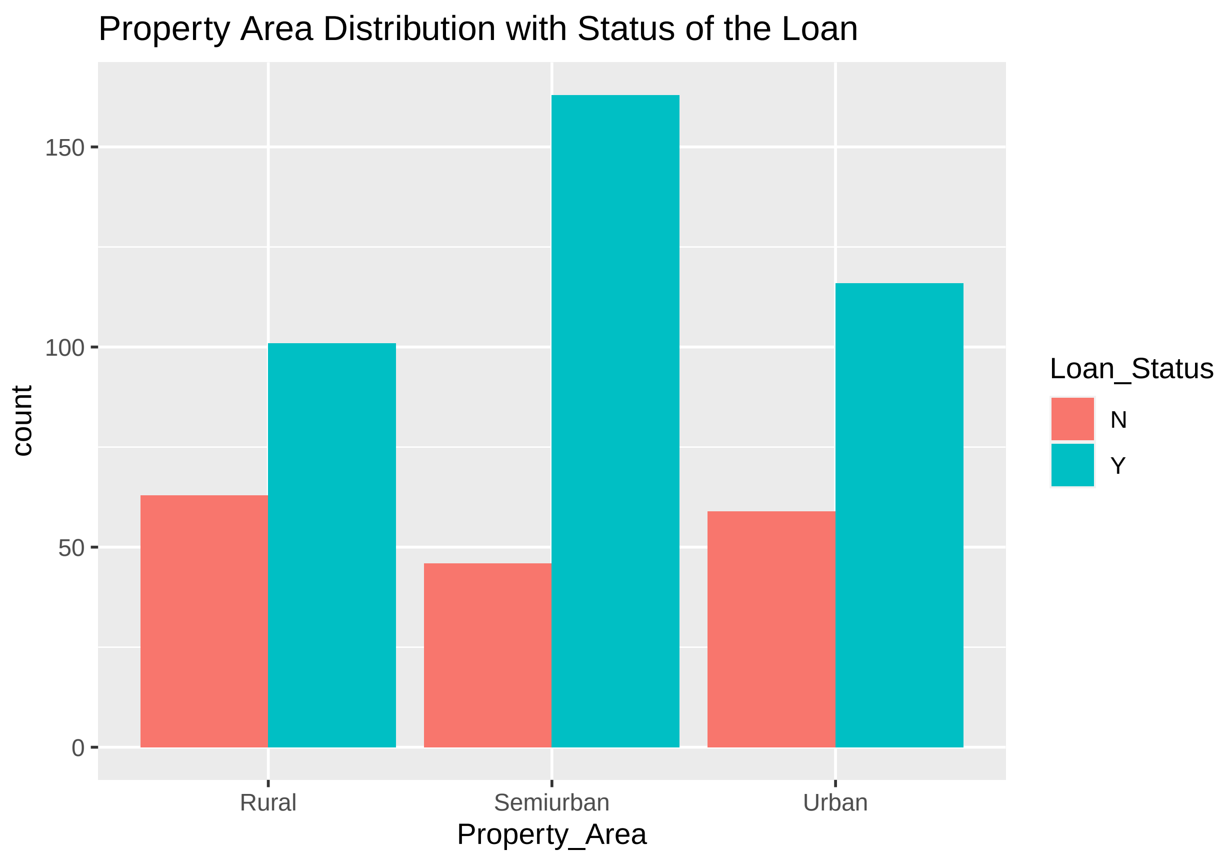

```
loan_data %>% ggplot(aes(Loan_Amount_Term, fill = Loan_Status)) +
                geom_bar(position=position_dodge()) +
                ggtitle("Loan Amount Term Distribution with Status of the Loan")
```




```r
loan_data %>% ggplot(aes(Credit_History, fill = Loan_Status)) +
                geom_bar(position=position_dodge()) +
                ggtitle("Credit History Distribution with Status of the Loan")
```




```
loan_data %>% ggplot(aes(Property_Area, fill = Loan_Status)) +
                geom_bar(position=position_dodge()) +
                ggtitle("Property Area Distribution with Status of the Loan")
```



Scaling the data

I will transform the TotalIncome variable so that it will fit into the scale of 0-1

```
loan_data$TotalIncome <- (loan_data$TotalIncome - min(loan_data$TotalIncome)) /
                      (max(loan_data$TotalIncome) - min(loan_data$TotalIncome))
```

```
summary(loan_data$TotalIncome)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.2022  0.2933  0.3289  0.4134  1.0000
```

Features encoding

I will convert categorical data into numbers because it lead us to a better model and most of the algorithm cannot handle categorical data.

```
loan_data$Gender <- ifelse(loan_data$Gender == "Male",1,0)
loan_data$Married <- ifelse(loan_data$Married == "Yes",1,0)


loan_data$Dependents <- as.integer(loan_data$Dependents)
```

```r
indexs <- which(loan_data$Dependents == 1)
loan_data$Dependents[indexs] <- 0
indexs <- which(loan_data$Dependents == 2)
loan_data$Dependents[indexs] <- 1
indexs <- which(loan_data$Dependents == 3)
loan_data$Dependents[indexs] <- 2
indexs <- which(loan_data$Dependents == 4)
loan_data$Dependents[indexs] <- 3

loan_data$Education <- ifelse(loan_data$Education == "Graduate",1,0)
loan_data$Self_Employed <- ifelse(loan_data$Self_Employed == "Yes",1,0)
loan_data$LoanAmount <- (loan_data$LoanAmount - min(loan_data$LoanAmount)) /
                      (max(loan_data$LoanAmount) - min(loan_data$LoanAmount))
loan_data$Loan_Amount_Term<-as.numeric(as.character(loan_data$Loan_Amount_Term))
loan_data$Loan_Amount_Term <- (loan_data$Loan_Amount_Term -
          min(loan_data$Loan_Amount_Term)) / (max(loan_data$Loan_Amount_Term) -
          min(loan_data$Loan_Amount_Term))
loan_data$Credit_History <- as.numeric(as.character(loan_data$Credit_History))

levels(loan_data$Property_Area)
```

```
## [1] "Rural"     "Semiurban" "Urban"
```

```r
loan_data$Property_Area <- as.character(loan_data$Property_Area)
indexs <- which(loan_data$Property_Area == "Rural")
loan_data$Property_Area[indexs] <- 0
indexs <- which(loan_data$Property_Area == "Semiurban")
loan_data$Property_Area[indexs] <- 1
indexs <- which(loan_data$Property_Area == "Urban")
loan_data$Property_Area[indexs] <- 2
loan_data$Property_Area <- as.numeric(loan_data$Property_Area)
```

```r
head(loan_data,7)
```

```
##   Gender Married Dependents Education Self_Employed LoanAmount Loan_Amount_Term
## 1      1       0          0         1             0  0.3242507        0.7435897
## 2      1       1          1         1             0  0.3242507        0.7435897
## 3      1       1          0         1             1  0.1553134        0.7435897
## 4      1       1          0         0             0  0.3024523        0.7435897
## 5      1       0          0         1             0  0.3596730        0.7435897
## 6      1       1          2         1             1  0.7029973        0.7435897
## 7      1       1          0         0             0  0.2343324        0.7435897
##   Credit_History Property_Area TotalIncome Loan_Status
## 1              1             2   0.3581762           Y
## 2              1             0   0.3778446           N
## 3              1             2   0.1266255           Y
## 4              1             2   0.2843791           Y
## 5              1             2   0.3704486           Y
## 6              1             2   0.6640930           Y
## 7              1             2   0.1956274           Y
```

Modeling approach

For this project I will use 6 machine learning algorithms:

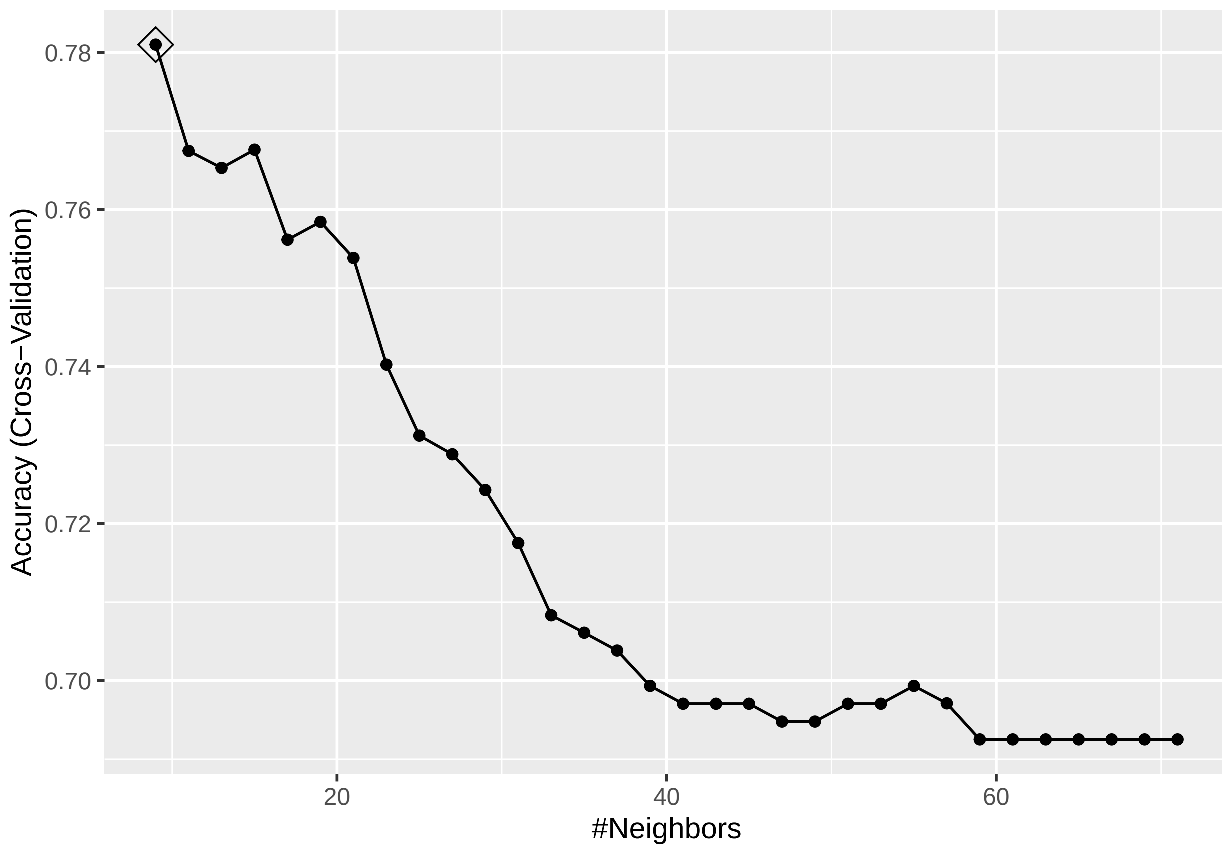
- k-nearest neighbors
- Generalized Linear Model
- Quadratic Discriminant Analysis
- Linear discriminant analysis
- Decision Trees
- Random forest

Splitting the loan_data in train data and test data:

```
set.seed(1)
index <- createDataPartition(loan_data$Loan_Status,times = 1,p = 0.8,
                             list = FALSE)
loan_data_train <- loan_data[index,]
loan_data_test <-loan_data[-index,]
```

K-nearest neighbors

```
control <- trainControl(method = "cv", number = 10, p = .9)
train_knn <- train(Loan_Status ~ ., method = "knn", data = loan_data_train,
                   tuneGrid = data.frame(k = seq(9, 71, 2)),trControl = control)
ggplot(train_knn, highlight = TRUE)
```




```
train_knn$bestTune
```

```
##   k
## 1 9
```

```
pred_LoanStatus_knn <- predict(train_knn,loan_data_test)
```

Accuracy of the model:

```
knn_accuracy <- confusionMatrix(pred_LoanStatus_knn,loan_data_test$Loan_Status)$
                                                          overall["Accuracy"]
knn_accuracy
```

```
##  Accuracy
## 0.7889908
```

Generalized Linear Model

```
train_glm <- train(Loan_Status ~ ., method = "glm", data = loan_data_train)
pred_LoanStatus_glm <- predict(train_glm,loan_data_test)
```

Accuracy of the model:

```r
glm_accuracy <- confusionMatrix(pred_LoanStatus_glm,loan_data_test$Loan_Status)$
                                                          overall["Accuracy"]
glm_accuracy
```

```
##  Accuracy
## 0.8348624
```

Quadratic Discriminant Analysis

```r
train_qda <- train(Loan_Status ~ ., method = "qda", data = loan_data_train)
pred_LoanStatus_qda <- predict(train_qda,loan_data_test)
```

Accuracy of the model:

```r
qda_accuracy <- confusionMatrix(pred_LoanStatus_qda,loan_data_test$Loan_Status)$
                                                          overall["Accuracy"]
qda_accuracy
```

```
##  Accuracy
## 0.8165138
```

Linear discriminant analysis

```r
train_lda <- train(Loan_Status ~ ., method = "lda", data = loan_data_train)
pred_LoanStatus_lda <- predict(train_lda,loan_data_test)
```

Accuracy of the model:

```r
lda_accuracy <- confusionMatrix(pred_LoanStatus_lda,loan_data_test$Loan_Status)$
                                                      overall["Accuracy"]
lda_accuracy
```

```
##  Accuracy
## 0.8348624
```

Decision Trees

```r
train_rpart <- train(Loan_Status ~ ., method = "rpart",data = loan_data_train)
pred_LoanStatus_rpart <- predict(train_rpart,loan_data_test)
```

Accuracy of the model:

```r
rpart_accuracy <- confusionMatrix(pred_LoanStatus_rpart,loan_data_test$
                                          Loan_Status)$overall["Accuracy"]
rpart_accuracy
```

```
##  Accuracy
## 0.8348624
```

Random forest

```
train_rf <- train(Loan_Status ~ ., method = "rf", data = loan_data_train)
pred_LoanStatus_rf <- predict(train_rf,loan_data_test)
```

Accuracy of the model:

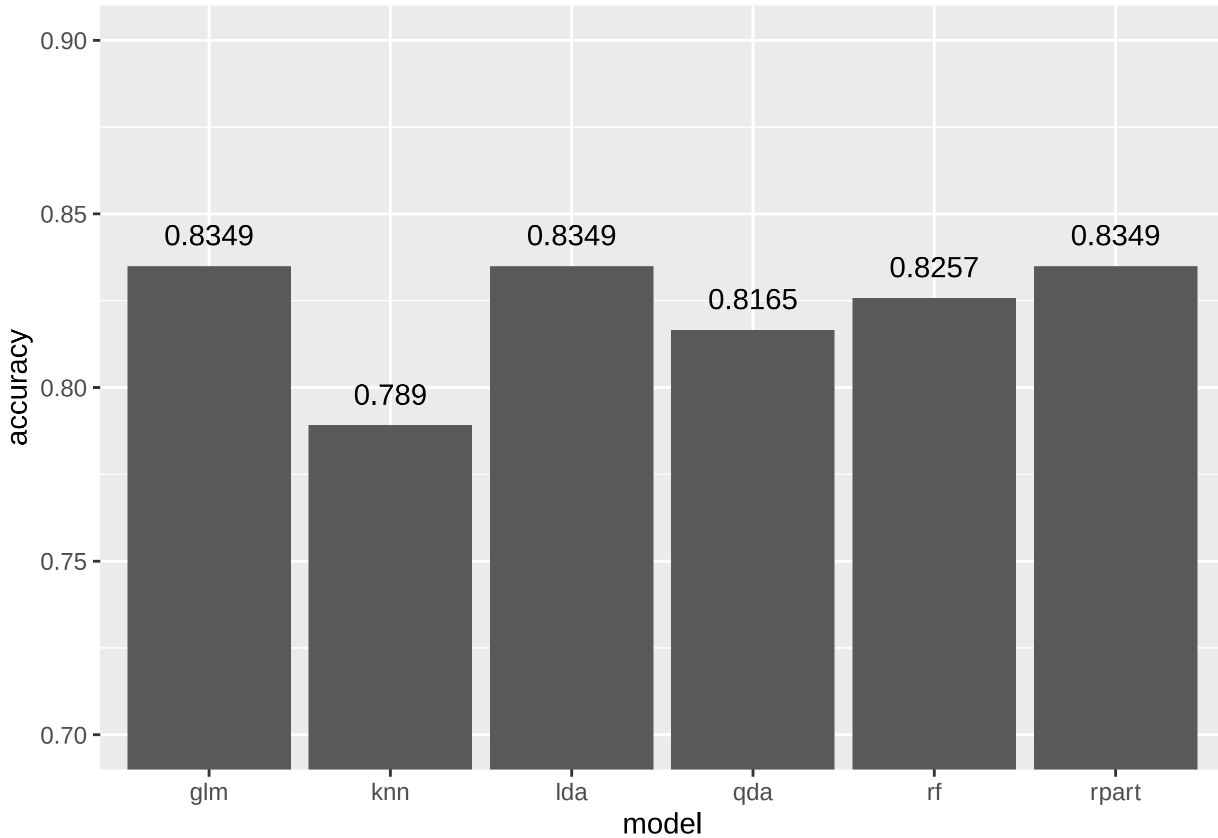
```
rf_accuracy <- confusionMatrix(pred_LoanStatus_rf,loan_data_test$Loan_Status)$
                                                        overall["Accuracy"]
rf_accuracy
```

```
##  Accuracy
## 0.8256881
```

Models performance

```
model <- rep(0,6)
accuracy <- rep(0,6)
accuracy_summary <- data.frame(model, accuracy)
accuracy_summary$model[1] <- "knn"
accuracy_summary$accuracy[1] <- round(knn_accuracy,4)
accuracy_summary$model[2] <- "glm"
accuracy_summary$accuracy[2] <- round(glm_accuracy,4)
accuracy_summary$model[3] <- "qda"
accuracy_summary$accuracy[3] <- round(qda_accuracy,4)
accuracy_summary$model[4] <- "lda"
accuracy_summary$accuracy[4] <- round(lda_accuracy,4)
accuracy_summary$model[5] <- "rpart"
accuracy_summary$accuracy[5] <- round(rpart_accuracy,4)
accuracy_summary$model[6] <- "rf"
accuracy_summary$accuracy[6] <- round(rf_accuracy,4)

accuracy_summary %>% ggplot(aes(x = model,y = accuracy)) +
  geom_bar(stat = "identity") + scale_y_continuous(limits=c(0.7,0.9),oob =
              rescale_none) + geom_text(aes(label = accuracy), vjust = -1)
```



Conclusions

We can see that the most accurate models for our dataset is GLM, LDA and LPART with an accuracy of 0.8349.