Probability of default

Harvard University

Data science capstone project

Yahya Abdelmoaty

12 JANUARY 2022

Probability of default

Yahya Abdelmoaty

12 January 2022

https://www.linkedin.com/in/yahya-kamel-5653b24b/

Harvard University
Data science capstone project

The objective of this project is to use Machine Learning “ML” prediction models to predict the Probability of Default “PD” of retail loans. This prediction process is very vital for companies and institutions dealing with receivables and loans. Through this project, we will get hands on different models and techniques to get more insight from the prediction models.

This project explores the following ML prediction models:

- Logistic regression with its different branches (log, logit, cloglog, probit)
- K Nearest Neighbors non-linear regression
- Vector Support Machines - Radial
- Neural networks
- Naive Bayes
- Decision tree
- Random forest

This paper is divided in the following sections:

1. Data exploration
2. Data wrangling
3. Modeling
4. Final PDs and conclusion

The underlying dataset contains the following data about around 29 thousand of retail borrowers:

- loan_status: 0 if borrower has defaulted and 1 if borrower has not defaulted.
- loan_amt: Total amount of loan
- int_rate: Interest rate applicable to the loan
- grade: Credit risk rating. A is the lowest in risk and G is the highest in risk.
- annual_inc: Annual income of the borrower.
- emp_length: Number of years of work.
- home_ownership: Rent/Mortage/Own/Others.
- age: Borrower's age.

The expected code run time is 25 – 45 minutes, depending on the machine speed.

Disclaimer:

The information, estimates, code, and conclusions mentioned in this paper are only for educational purposes and should not be relied upon for any decisions.

Section 1 - Data exploration

We will explore the dataset in terms of data distribution, outliers, missing data, ..etc, trying to understnad the data and understand the relevant relationships, which can substantially help us building the most relevant PD model.

Import dataset

```
dataset = readRDS(gzcon(url("https://assets.datacamp.com/production/repositories/162/datasets/8f48a2cbb6150e7ae32435e55f271cad5b4b8ecf/loan_data_ch1.rds")))
```

View dataset

```
str(dataset)

## 'data.frame':    29092 obs. of  8 variables:
##  $ loan_status   : int  0 0 0 0 0 0 1 0 1 0 ...
##  $ loan_amnt     : int  5000 2400 10000 5000 3000 12000 9000 3000 10000 1000 ...
##  $ int_rate      : num  10.6 NA 13.5 NA NA ...
##  $ grade         : Factor w/ 7 levels "A","B","C","D",..: 2 3 3 1 5 2 3 2 2 4 ...
##  $ emp_length    : int  10 25 13 3 9 11 0 3 3 0 ...
##  $ home_ownership: Factor w/ 4 levels "MORTGAGE","OTHER",..: 4 4 4 4 4 3 4 4 4 4 ...
##  $ annual_inc    : num  24000 12252 49200 36000 48000 ...
##  $ age           : int  33 31 24 39 24 28 22 22 28 22 ...

head(dataset)

##   loan_status loan_amnt int_rate grade emp_length home_ownership annual_inc age
## 1           0      5000    10.65     B         10           RENT      24000  33
## 2           0      2400       NA     C         25           RENT      12252  31
## 3           0     10000    13.49     C         13           RENT      49200  24
## 4           0      5000       NA     A          3           RENT      36000  39
## 5           0      3000       NA     E          9           RENT      48000  24
## 6           0     12000    12.69     B         11            OWN      75000  28
```

Summarize dataset and NAs

```
head(dataset)

##   loan_status loan_amnt int_rate grade emp_length home_ownership annual_inc age
## 1           0      5000    10.65     B         10           RENT      24000  33
## 2           0      2400       NA     C         25           RENT      12252  31
## 3           0     10000    13.49     C         13           RENT      49200  24
## 4           0      5000       NA     A          3           RENT      36000  39
## 5           0      3000       NA     E          9           RENT      48000  24
## 6           0     12000    12.69     B         11            OWN      75000  28

nrow(dataset) # number of observations to see if NAs are significant enough to be removed/replaced/kept

## [1] 29092

summary(dataset)

##   loan_status       loan_amnt       int_rate     grade      emp_length
##  Min.   :0.0000   Min.   :  500   Min.   : 5.42   A:9649   Min.   : 0.000
##  1st Qu.:0.0000   1st Qu.: 5000   1st Qu.: 7.90   B:9329   1st Qu.: 2.000
```

```
##  Median :0.0000   Median : 8000   Median :10.99   C:5748   Median : 4.000
##  Mean   :0.1109   Mean   : 9594   Mean   :11.00   D:3231   Mean   : 6.145
##  3rd Qu.:0.0000   3rd Qu.:12250   3rd Qu.:13.47   E: 868   3rd Qu.: 8.000
##  Max.   :1.0000   Max.   :35000   Max.   :23.22   F: 211   Max.   :62.000
##                                   NA's   :2776    G:  56   NA's   :809
##   home_ownership    annual_inc          age
##  MORTGAGE:12002   Min.   :   4000   Min.   : 20.0
##  OTHER   :   97   1st Qu.:  40000   1st Qu.: 23.0
##  OWN     : 2301   Median :  56424   Median : 26.0
##  RENT    :14692   Mean   :  67169   Mean   : 27.7
##                   3rd Qu.:  80000   3rd Qu.: 30.0
##                   Max.   :6000000   Max.   :144.0
##
```

```
str(dataset)
```

```
## 'data.frame':    29092 obs. of  8 variables:
##  $ loan_status   : int  0 0 0 0 0 0 1 0 1 0 ...
##  $ loan_amnt     : int  5000 2400 10000 5000 3000 12000 9000 3000 10000 1000 ...
##  $ int_rate      : num  10.6 NA 13.5 NA NA ...
##  $ grade         : Factor w/ 7 levels "A","B","C","D",..: 2 3 3 1 5 2 3 2 2 4 ...
##  $ emp_length    : int  10 25 13 3 9 11 0 3 3 0 ...
##  $ home_ownership: Factor w/ 4 levels "MORTGAGE","OTHER",..: 4 4 4 4 4 3 4 4 4 4 ...
##  $ annual_inc    : num  24000 12252 49200 36000 48000 ...
##  $ age           : int  33 31 24 39 24 28 22 22 28 22 ...
```

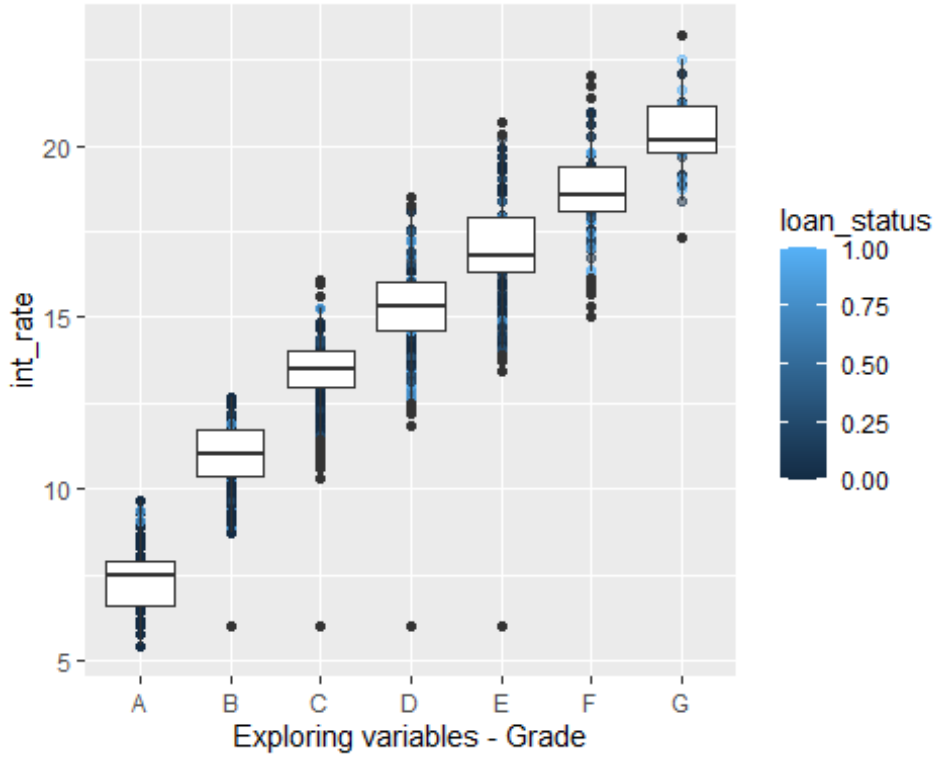
Columns with potential factors

```
cols_to_normalize = c(2,3,5,7,8) #Manual input
```

Visualize variables distribution and locate potential outliers

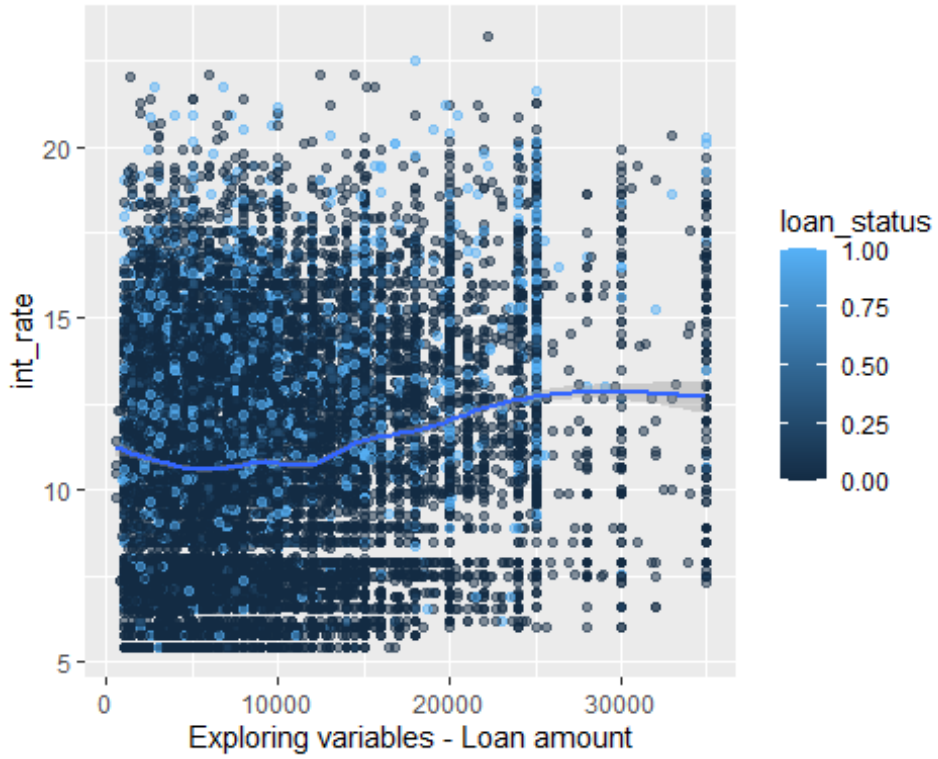
Commentary: There is a relationship between the grade, interest rate and loan defaults. Also, one important observation is that the default data is scattered all around the population of the dataset and it will be hard to predict using linear techniques as well as non-linear techniques without clusttering. Clusterring will be critical for the success of the prediction model.

```
dataset %>%
  ggplot(aes(x = grade,y = int_rate))+
  geom_point(aes(color = loan_status), alpha = 0.5)+
  geom_boxplot()+
# facet_wrap(~loan_status)+ #Add if you wanna split the classification in two tables
  xlab("Exploring variables - Grade")
```



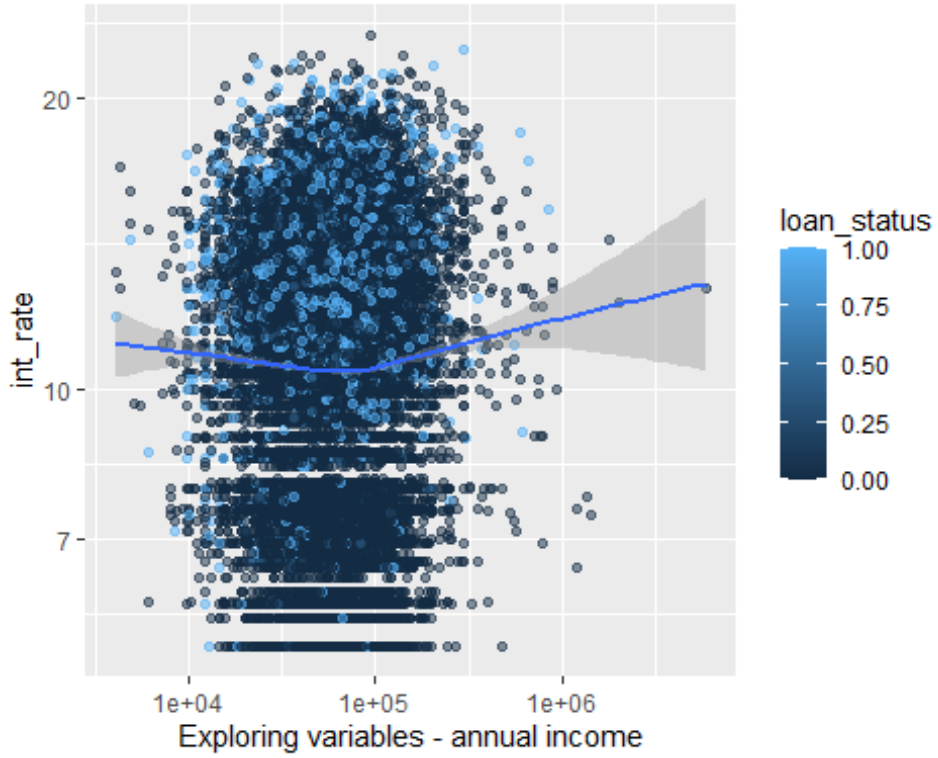
Commentary: There is a relationship between the loan amount, interest rate and loan defaults

```
dataset %>%
  ggplot(aes(x = loan_amnt,y = int_rate))+
  geom_point(aes(color = loan_status), alpha = 0.5)+
  geom_smooth()+
# facet_wrap(~loan_status)+ #Add if you wanna split the classification in two tables
  xlab("Exploring variables - Loan amount")

## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



Commentary: There is a relationship between the annual income, interest rate and loan defaults

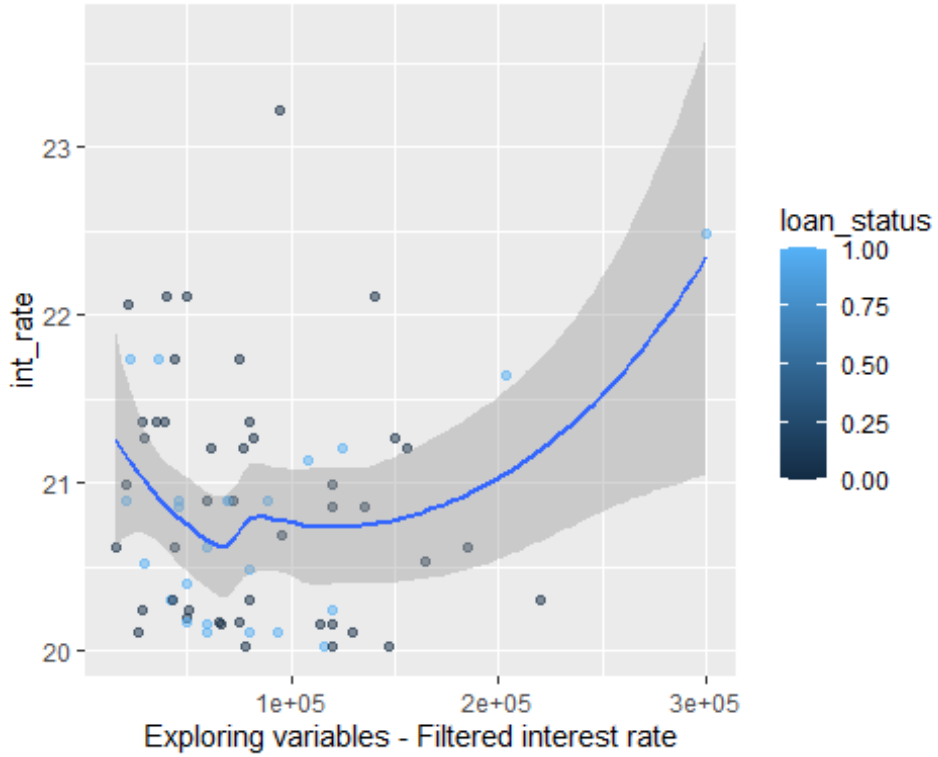
```
dataset %>%
  ggplot(aes(x = annual_inc,y = int_rate))+
  geom_point(aes(color = loan_status), alpha = 0.5)+
  geom_smooth()+
  scale_y_continuous(trans = 'log10')+
  scale_x_continuous(trans = 'log10')+
# facet_wrap(~loan_status)+ #Add if you wanna split the classification in two tables
  xlab("Exploring variables - annual income")

## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



Commentary: Focusing on potential outliers in interest rate.

Decision: Remove from the dataset, given their limited number and high Standard Error.

```
dataset %>%
  filter(int_rate >= 20) %>%
  ggplot(aes(x = annual_inc,y = int_rate))+
  geom_point(aes(color = loan_status), alpha = 0.5)+
  geom_smooth()+
# facet_wrap(~loan_status)+ #Add if you wanna split the classification in two tables
  xlab("Exploring variables - Filtered interest rate")

## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



Number of those observations

```
filter_int = dataset %>%  filter(int_rate >= 20)
nrow(filter_int)
```

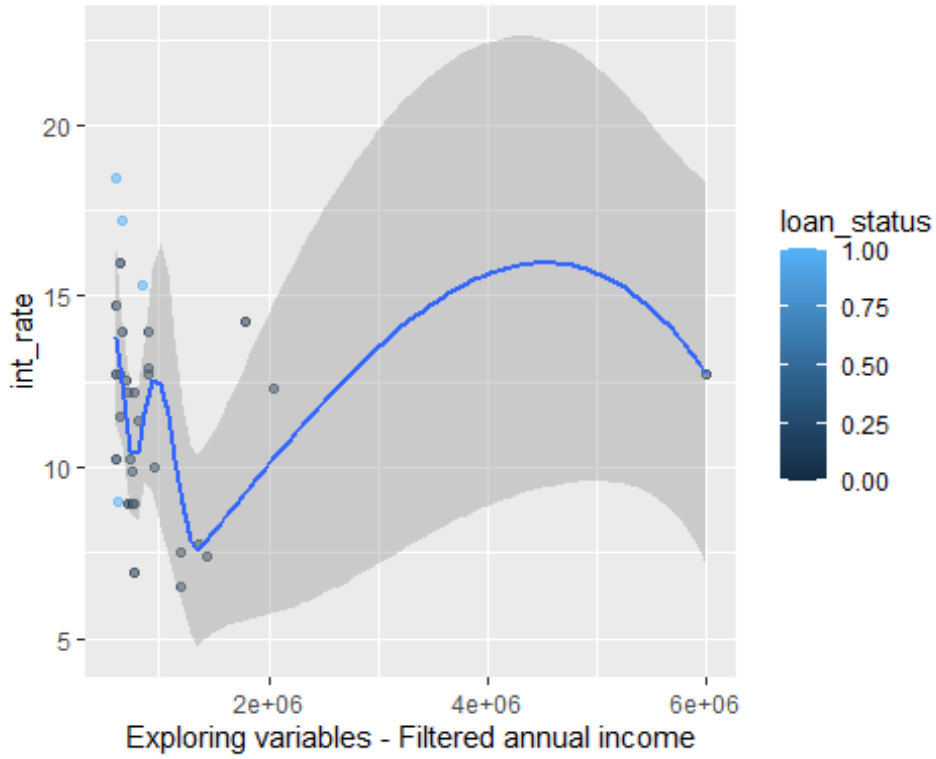
```
## [1] 68
```

Commentary: Focusing on potential outliers in annual income.

Decision: Remove from the dataset, given their limited number and high Standard Error

```
dataset %>%
  filter(annual_inc >= 600000) %>%
  ggplot(aes(x = annual_inc,y = int_rate))+
  geom_point(aes(color = loan_status), alpha = 0.5)+
  geom_smooth()+
# facet_wrap(~loan_status)+ #Add if you wanna split the classification in two tables
  xlab("Exploring variables - Filtered annual income")
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

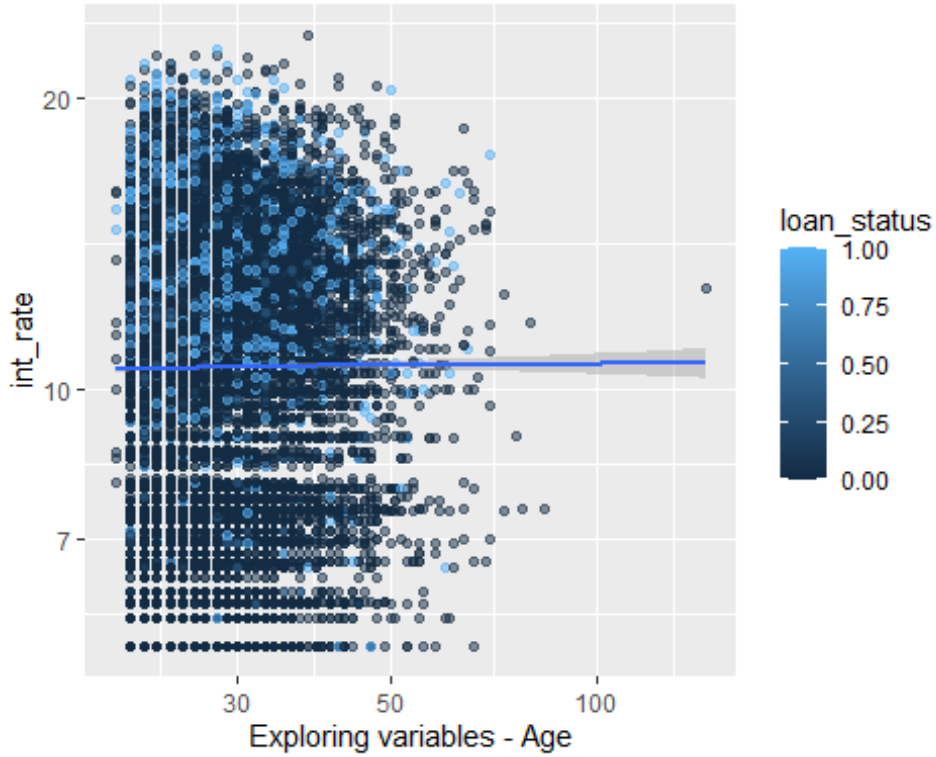


Number of those observations

```
filter_income = dataset %>%  filter(annual_inc >= 600000)
nrow(filter_income)

## [1] 38
```

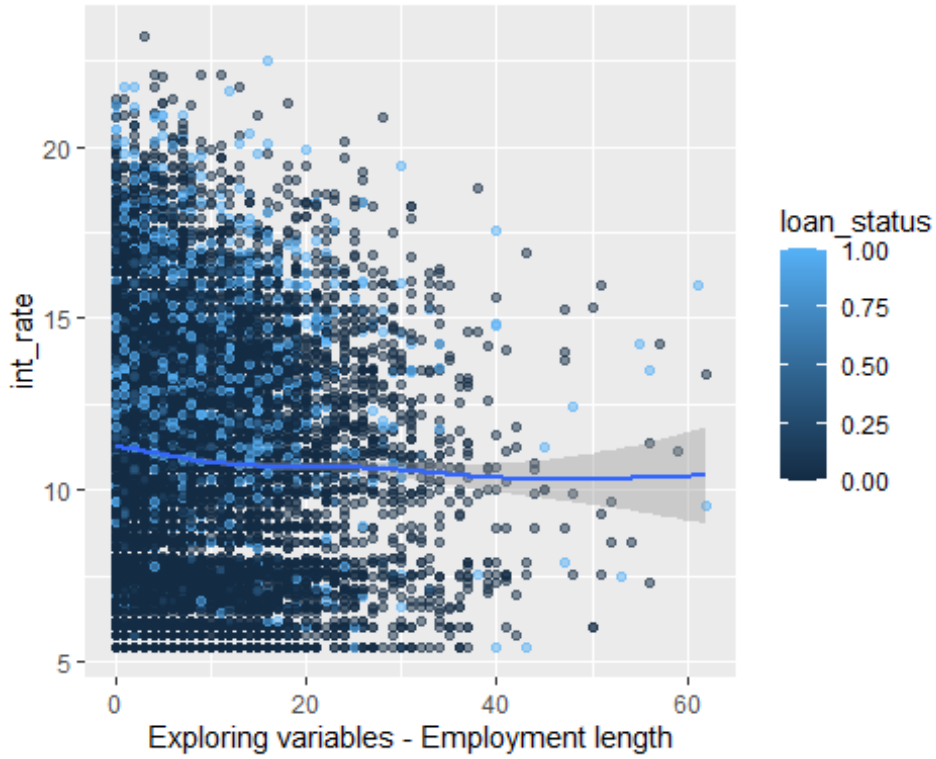
Commentary: There is almost no relationship between the age and loan defaults. One observation above 100 of age, which can be removed as it seems incorrect.

```
dataset %>%
  ggplot(aes(x = age,y = int_rate))+
  geom_point(aes(color = loan_status), alpha = 0.5)+
  scale_y_continuous(trans = 'log10')+
  scale_x_continuous(trans = 'log10')+
  geom_smooth()+
# facet_wrap(~loan_status)+ #Add if you wanna split the classification in two tables
  xlab("Exploring variables - Age")

## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



Commentary: There is almost no relationship between the employment length and loan defaults

```
dataset %>%
  ggplot(aes(x = emp_length,y = int_rate))+
  geom_point(aes(color = loan_status), alpha = 0.5)+
  geom_smooth()+
# facet_wrap(~loan_status)+ #Add if you wanna split the classification in two tables
  xlab("Exploring variables - Employment length")

## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



Section 2 - Data wrangling

Preparing the dataset for modeling

```
dataset_after_NAs = dataset
dataset_after_NAs$int_rate = ifelse(is.na(dataset_after_NAs$int_rate),
                                    ave(dataset_after_NAs$int_rate,FUN = function(x) mean(x, na.rm=TRUE)),
                                    dataset_after_NAs$int_rate)
dataset_after_NAs$emp_length = ifelse(is.na(dataset_after_NAs$emp_length),
                                      ave(dataset_after_NAs$emp_length,FUN = function(x) mean(x, na.rm=TRUE)),
                                      dataset_after_NAs$emp_length)
```

Check if NA values have been replaced for all columns

```
dataset_excluding_outliers = dataset_after_NAs
colSums(is.na(dataset_excluding_outliers))
```

```
##    loan_status      loan_amnt       int_rate          grade     emp_length
##              0              0              0              0              0
## home_ownership     annual_inc            age
##              0              0              0
```

Remove outliers

```
dataset_excluding_outliers = dataset_excluding_outliers %>%
  filter(annual_inc < 600000) %>%
  filter(int_rate < 20) %>%
  filter(age<100)
```

Observations removed - 106 in total, which is not significant for modeling, but this gives our model better chance of correct prediction

```
nrow(dataset)-nrow(dataset_excluding_outliers)
```

```
## [1] 106
```

View overall dataset status and check if there are further data refinement needed

```
summary(dataset_excluding_outliers)
```

```
##   loan_status       loan_amnt       int_rate     grade      emp_length
##  Min.   :0.0000   Min.   :  500   Min.   : 5.42   A:9638   Min.   : 0.000
##  1st Qu.:0.0000   1st Qu.: 5000   1st Qu.: 8.49   B:9317   1st Qu.: 2.000
##  Median :0.0000   Median : 8000   Median :11.00   C:5738   Median : 4.000
##  Mean   :0.1103   Mean   : 9576   Mean   :10.98   D:3228   Mean   : 6.142
##  3rd Qu.:0.0000   3rd Qu.:12019   3rd Qu.:13.11   E: 861   3rd Qu.: 8.000
##  Max.   :1.0000   Max.   :35000   Max.   :19.91   F: 186   Max.   :62.000
##                                                   G:  18
##   home_ownership    annual_inc         age
##  MORTGAGE:11952   Min.   :  4000   Min.   :20.00
##  OTHER   :   96   1st Qu.: 40000   1st Qu.:23.00
##  OWN     : 2292   Median : 56078   Median :26.00
##  RENT    :14646   Mean   : 65849   Mean   :27.68
##                   3rd Qu.: 80000   3rd Qu.:30.00
##                   Max.   :572400   Max.   :94.00
##
```

#Re-visualize the variables after removing the top outliers, following the visualization section above

Encode the target dependent as well as other relevant variables as factor

```
dataset_excluding_outliers$loan_status =  factor(
  dataset_excluding_outliers$loan_status,
  levels = c(0,1),
  labels = c(0,1)) # 0: loan with no default / 1: loan with default
dataset_excluding_outliers$home_ownership = factor(
  dataset_excluding_outliers$home_ownership,
  levels = c("RENT","OWN","MORTGAGE","OTHER"),
  labels = c(1:4))
dataset_excluding_outliers$grade = factor(
  dataset_excluding_outliers$grade,
  levels = c("A","B","C","D","E","F","G"),
  labels = c(1:7))
```

View data after alterations above

```
str(dataset_excluding_outliers)

## 'data.frame':    28986 obs. of  8 variables:
##  $ loan_status   : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 2 1 2 1 ...
##  $ loan_amnt     : int  5000 2400 10000 5000 3000 12000 9000 3000 10000 1000 ...
##  $ int_rate      : num  10.6 11 13.5 11 11 ...
##  $ grade         : Factor w/ 7 levels "1","2","3","4",..: 2 3 3 1 5 2 3 2 2 4 ...
##  $ emp_length    : num  10 25 13 3 9 11 0 3 3 0 ...
##  $ home_ownership: Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 2 1 1 1 1 ...
##  $ annual_inc    : num  24000 12252 49200 36000 48000 ...
##  $ age           : int  33 31 24 39 24 28 22 22 28 22 ...

summary(dataset_excluding_outliers)

##  loan_status   loan_amnt       int_rate     grade      emp_length
##  0:25788     Min.   :  500   Min.   : 5.42   1:9638   Min.   : 0.000
##  1: 3198     1st Qu.: 5000   1st Qu.: 8.49   2:9317   1st Qu.: 2.000
##              Median : 8000   Median :11.00   3:5738   Median : 4.000
##              Mean   : 9576   Mean   :10.98   4:3228   Mean   : 6.142
##              3rd Qu.:12019   3rd Qu.:13.11   5: 861   3rd Qu.: 8.000
##              Max.   :35000   Max.   :19.91   6: 186   Max.   :62.000
##                                              7:  18
##  home_ownership   annual_inc         age
##  1:14646        Min.   :  4000   Min.   :20.00
##  2: 2292        1st Qu.: 40000   1st Qu.:23.00
##  3:11952        Median : 56078   Median :26.00
##  4:   96        Mean   : 65849   Mean   :27.68
##                 3rd Qu.: 80000   3rd Qu.:30.00
##                 Max.   :572400   Max.   :94.00
##

head(dataset_excluding_outliers)

##   loan_status loan_amnt int_rate grade emp_length home_ownership annual_inc age
## 1           0      5000 10.65000     2         10              1      24000  33
## 2           0      2400 11.00457     3         25              1      12252  31
## 3           0     10000 13.49000     3         13              1      49200  24
## 4           0      5000 11.00457     1          3              1      36000  39
```

```
## 5         0     3000 11.00457     5         9             1     48000  24
## 6         0    12000 12.69000     2        11             2     75000  28
```

View correlations between the independent variables

Commentary: No significant correlations between the independent variables. The highest observed is 40% between annual income and the loan amount, which is natural as part of the lending process and is n't significant to impact our models' predictions.

```
round(cor(dataset_excluding_outliers[,cols_to_normalize]),2)

##            loan_amnt int_rate emp_length annual_inc  age
## loan_amnt       1.00     0.13       0.09       0.40 0.05
## int_rate        0.13     1.00      -0.06       0.02 0.01
## emp_length      0.09    -0.06       1.00       0.14 0.01
## annual_inc      0.40     0.02       0.14       1.00 0.12
## age             0.05     0.01       0.01       0.12 1.00
```

Feature Scaling, normalizing the numbers to standard unit

```
dataset_excluding_outliers_normalized = dataset_excluding_outliers
dataset_excluding_outliers_normalized[,cols_to_normalize] = scale(dataset_excluding_outliers[,cols_to_n
ormalize])
```

Splitting the dataset into the Training set and Test set

```
set.seed(11111)
split = sample.split(dataset_excluding_outliers_normalized$loan_status, SplitRatio = 0.75)
training_set = subset(dataset_excluding_outliers_normalized, split == TRUE)
test_set = subset(dataset_excluding_outliers_normalized, split == FALSE)
test_set_unnormalized = subset(dataset_excluding_outliers, split == FALSE)
```

Section 3 - Modeling

Fitting GLM Logistic Regression to dataset

#A polynomial relationship is reflected into interest rate

```
set.seed(1)
classifier_log = glm(loan_status ~  poly(int_rate,3) + grade + annual_inc,family=binomial,data=training_se
t) #exclude variables with no significance
 summary(classifier_log)

##
## Call:
## glm(formula = loan_status ~ poly(int_rate, 3) + grade + annual_inc,
##     family = binomial, data = training_set)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.9549  -0.5327  -0.4424  -0.3305   3.1586
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -2.56264    0.07851 -32.642  < 2e-16 ***
## poly(int_rate, 3)1  32.74995    8.01741   4.085 4.41e-05 ***
## poly(int_rate, 3)2 -13.27761    5.12526  -2.591  0.00958 **
## poly(int_rate, 3)3   8.97276    3.91675   2.291  0.02197 *
## grade2               0.30132    0.10014   3.009  0.00262 **
## grade3               0.62252    0.12751   4.882 1.05e-06 ***
## grade4               0.83253    0.14602   5.702 1.19e-08 ***
## grade5               0.93617    0.18155   5.157 2.51e-07 ***
## grade6               1.19396    0.26109   4.573 4.81e-06 ***
## grade7               1.17727    0.63049   1.867  0.06187 .
## annual_inc          -0.28075    0.02781 -10.094  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 15094  on 21738  degrees of freedom
## Residual deviance: 14471  on 21728  degrees of freedom
## AIC: 14493
##
## Number of Fisher Scoring iterations: 5

classifier_logit = glm(loan_status ~  poly(int_rate,3) + grade + annual_inc,family = binomial(link = logit), d
ata = training_set)
 summary(classifier_logit)

##
## Call:
## glm(formula = loan_status ~ poly(int_rate, 3) + grade + annual_inc,
##     family = binomial(link = logit), data = training_set)
##
## Deviance Residuals:
```

```
##     Min       1Q   Median       3Q      Max
## -0.9549  -0.5327  -0.4424  -0.3305   3.1586
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -2.56264    0.07851 -32.642  < 2e-16 ***
## poly(int_rate, 3)1  32.74995    8.01741   4.085 4.41e-05 ***
## poly(int_rate, 3)2 -13.27761    5.12526  -2.591  0.00958 **
## poly(int_rate, 3)3   8.97276    3.91675   2.291  0.02197 *
## grade2               0.30132    0.10014   3.009  0.00262 **
## grade3               0.62252    0.12751   4.882 1.05e-06 ***
## grade4               0.83253    0.14602   5.702 1.19e-08 ***
## grade5               0.93617    0.18155   5.157 2.51e-07 ***
## grade6               1.19396    0.26109   4.573 4.81e-06 ***
## grade7               1.17727    0.63049   1.867  0.06187 .
## annual_inc          -0.28075    0.02781 -10.094  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 15094  on 21738  degrees of freedom
## Residual deviance: 14471  on 21728  degrees of freedom
## AIC: 14493
##
## Number of Fisher Scoring iterations: 5

classifier_cloglog =  glm(loan_status ~ poly(int_rate,3) + grade + annual_inc,family = binomial(link = clogl
og), data = training_set)
 summary(classifier_cloglog)

##
## Call:
## glm(formula = loan_status ~ poly(int_rate, 3) + grade + annual_inc,
##     family = binomial(link = cloglog), data = training_set)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.9654  -0.5324  -0.4422  -0.3308   3.1244
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -2.60146    0.07478 -34.789  < 2e-16 ***
## poly(int_rate, 3)1  31.14671    7.56941   4.115 3.87e-05 ***
## poly(int_rate, 3)2 -13.25069    4.84404  -2.735  0.00623 **
## poly(int_rate, 3)3   8.47135    3.59635   2.356  0.01850 *
## grade2               0.28633    0.09577   2.990  0.00279 **
## grade3               0.58643    0.12023   4.877 1.07e-06 ***
## grade4               0.77986    0.13649   5.714 1.11e-08 ***
## grade5               0.87234    0.16699   5.224 1.75e-07 ***
## grade6               1.08717    0.23311   4.664 3.10e-06 ***
## grade7               1.08950    0.53538   2.035  0.04185 *
## annual_inc          -0.26485    0.02625 -10.088  < 2e-16 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 15094  on 21738  degrees of freedom
## Residual deviance: 14472  on 21728  degrees of freedom
## AIC: 14494
##
## Number of Fisher Scoring iterations: 6

classifier_probit = glm(loan_status ~ poly(int_rate,3) + grade + annual_inc,family = binomial(link = probit),
data = training_set)
 summary(classifier_probit)

##
## Call:
## glm(formula = loan_status ~ poly(int_rate, 3) + grade + annual_inc,
##     family = binomial(link = probit), data = training_set)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.9366  -0.5339  -0.4444  -0.3300   3.3331
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)       -1.46139    0.03942 -37.071  < 2e-16 ***
## poly(int_rate, 3)1 16.28068    4.10709   3.964 7.37e-05 ***
## poly(int_rate, 3)2 -5.89202    2.58676  -2.278  0.02274 *
## poly(int_rate, 3)3  4.48087    2.07428   2.160  0.03076 *
## grade2             0.15547    0.04973   3.126  0.00177 **
## grade3             0.32591    0.06573   4.958 7.13e-07 ***
## grade4             0.44044    0.07687   5.730 1.01e-08 ***
## grade5             0.49901    0.09863   5.059 4.21e-07 ***
## grade6             0.65337    0.14804   4.414 1.02e-05 ***
## grade7             0.64301    0.37399   1.719  0.08555 .
## annual_inc        -0.13967    0.01386 -10.076  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 15094  on 21738  degrees of freedom
## Residual deviance: 14471  on 21728  degrees of freedom
## AIC: 14493
##
## Number of Fisher Scoring iterations: 5
```

Step-wise AIC analysis, which presents AIC score for different models with different variables

Commentary: StepAIC is suggesting to further include “emp_length”, as a significant variable. However, due to the earlier analysis, we won’t include it in the model.

```
stepAIC(glm(loan_status ~ . ,family=binomial,data=training_set),direction="both") #Look for model with lo
west AIC

## Start:  AIC=14501.52
## loan_status ~ loan_amnt + int_rate + grade + emp_length + home_ownership +
##     annual_inc + age
##
##                Df Deviance   AIC
## - loan_amnt       1    14472 14500
## - home_ownership  3    14476 14500
## - age             1    14472 14500
## <none>                 14472 14502
## - emp_length      1    14475 14503
## - int_rate        1    14487 14515
## - grade           6    14512 14530
## - annual_inc      1    14570 14598
##
## Step:  AIC=14499.59
## loan_status ~ int_rate + grade + emp_length + home_ownership +
##     annual_inc + age
##
##                Df Deviance   AIC
## - home_ownership  3    14476 14498
## - age             1    14472 14498
## <none>                 14472 14500
## - emp_length      1    14475 14501
## + loan_amnt       1    14472 14502
## - int_rate        1    14487 14513
## - grade           6    14512 14528
## - annual_inc      1    14592 14618
##
## Step:  AIC=14498.11
## loan_status ~ int_rate + grade + emp_length + annual_inc + age
##
##                Df Deviance   AIC
## - age             1    14477 14497
## <none>                 14476 14498
## + home_ownership  3    14472 14500
## + loan_amnt       1    14476 14500
## - emp_length      1    14480 14500
## - int_rate        1    14491 14511
## - grade           6    14516 14526
## - annual_inc      1    14597 14617
##
## Step:  AIC=14497.04
## loan_status ~ int_rate + grade + emp_length + annual_inc
##
##                Df Deviance   AIC
## <none>                 14477 14497
## + age             1    14476 14498
## + home_ownership  3    14472 14498
## + loan_amnt       1    14477 14499
## - emp_length      1    14481 14499
## - int_rate        1    14492 14510
```

```
## - grade          6    14517 14525
## - annual_inc     1    14602 14620

##
## Call:  glm(formula = loan_status ~ int_rate + grade + emp_length + annual_inc,
##     family = binomial, data = training_set)
##
## Coefficients:
## (Intercept)     int_rate       grade2       grade3       grade4       grade5
##     -2.6185       0.1945       0.4451       0.6885       0.8327       0.9388
##      grade6       grade7   emp_length   annual_inc
##      1.2415       1.2534       0.0461      -0.2916
##
## Degrees of Freedom: 21738 Total (i.e. Null);  21729 Residual
## Null Deviance:       15090
## Residual Deviance: 14480     AIC: 14500
```

Predicting using test_set

```
prob_pred_test_set_log = predict(classifier_log, type = 'response', newdata = test_set[,-1])
prob_pred_test_set_logit = predict(classifier_logit, type = 'response', newdata = test_set[,-1])
prob_pred_test_set_cloglog = predict(classifier_cloglog, type = 'response', newdata = test_set[,-1])
prob_pred_test_set_probit = predict(classifier_probit, type = 'response', newdata = test_set[,-1])
```

View the probabilities range

Low range means bad model prediction. Wide range would presume the opposite.

```
prediction_range_table=data.frame(
range(prob_pred_test_set_log),
range(prob_pred_test_set_logit),
range(prob_pred_test_set_cloglog),
range(prob_pred_test_set_probit)
)
prediction_range_table
```

```
##   range.prob_pred_test_set_log. range.prob_pred_test_set_logit.
## 1                   0.002245982                     0.002245982
## 2                   0.380815609                     0.380815609
##   range.prob_pred_test_set_cloglog. range.prob_pred_test_set_probit.
## 1                       0.002604686                       0.000666377
## 2                       0.390791812                       0.365647769
```

Find cut-off threshold through ROC curve

Commentary: ROC in this test is irrelevant because we are more interested in the specificity "unpredicted defaults". A lower cut-off threshold means higher prediction of default and higher cut-off threshold, lower prediction of default. The code below is for demonstration purposes only.

```
# pred=prediction(prob_pred_test_set_probit,test_set[,1]) #Manual choice of the model
# perf=performance(pred,measure = "acc")
# plot(perf)
```

Find best accuracy point on ROCR curve

```
# max=which.max(slot(perf,"y.values")[[1]])
# acc=slot(perf,"y.values")[[1]][max]
# cut=slot(perf,"x.values")[[1]][max]
# ROC_cut-off threshold=round(c(Accuracy_value=acc,Cutoff_value=cut),4)
# ROC_cut-off threshold

cut-off threshold_pred=0.15 #Manual input
```

Convert prediction probabilities to binary - GLM models

```
y_pred_log_test_set = unname(as.factor( ifelse(prob_pred_test_set_log > cut-off threshold_pred, 1, 0)))
y_pred_logit_test_set = unname(as.factor( ifelse(prob_pred_test_set_logit > cut-off threshold_pred, 1, 0))
)
y_pred_cloglog_test_set = unname(as.factor( ifelse(prob_pred_test_set_cloglog > cut-off threshold_pred,
1, 0)))
y_pred_probit_test_set = unname(as.factor( ifelse(prob_pred_test_set_probit > cut-off threshold_pred, 1,
0)))
```

Run the Confusion Matrix

```
cm_log_test_set = confusionMatrix(data = y_pred_log_test_set,reference = test_set$loan_status)
cm_logit_test_set = confusionMatrix(data = y_pred_logit_test_set,reference = test_set$loan_status)
cm_cloglog_test_set = confusionMatrix(data = y_pred_cloglog_test_set,reference = test_set$loan_status)
cm_probit_test_set = confusionMatrix(data = y_pred_probit_test_set,reference = test_set$loan_status)
```

Display confusion matrix table

Commentary: GLM models demonstrate reasonable prediction accuracy rate for the overall data, but not enough for borrowers' defaulting, when looked at in isolation. There should be better models for capturing default rates more accurately without compromising the overall prediction accuracy.

```
cm_log_test_set$table

##          Reference
## Prediction    0    1
##          0 5013  492
##          1 1434  308

cm_logit_test_set$table

##          Reference
## Prediction    0    1
##          0 5013  492
##          1 1434  308

cm_cloglog_test_set$table

##          Reference
## Prediction    0    1
##          0 5022  492
##          1 1425  308

cm_probit_test_set$table

##          Reference
## Prediction    0    1
##          0 4997  490
##          1 1450  310
```

Caret - GLM models with cross validation, seeking the optimal GLM model outcome

Train the model

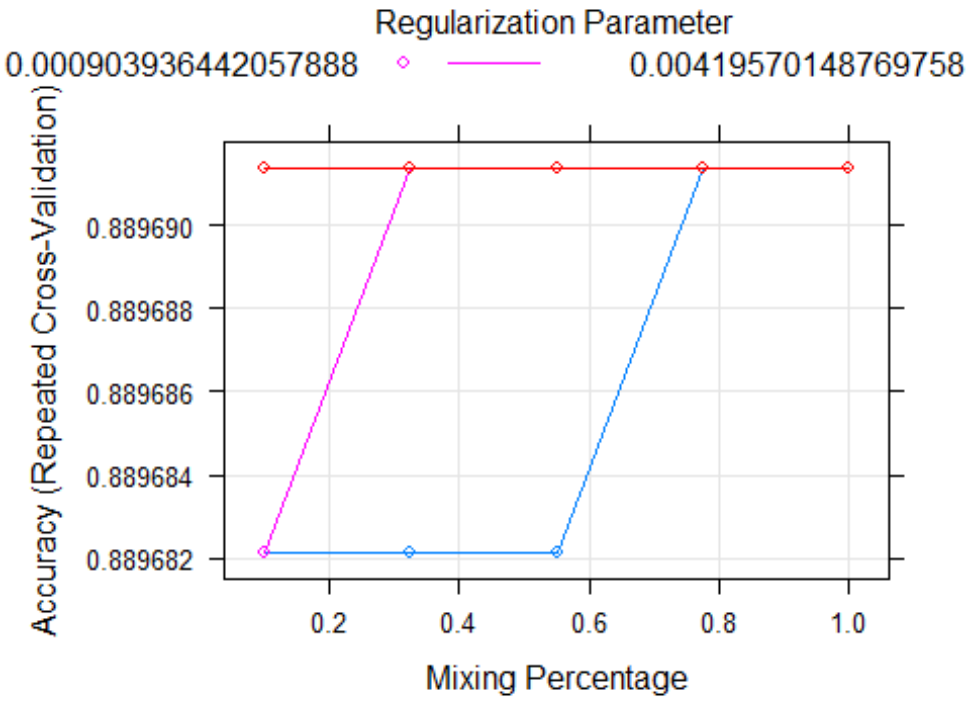
```
trctrl = trainControl(method="repeatedcv",
                      number=5,
                      repeats = 5)
```

Fitting the model - caret GLM

```
classifier_caret_GLM = train(loan_status ~int_rate + grade + annual_inc,
                      data = training_set,
                      method = "glmnet",
                      trControl = trctrl,
                      tuneLength=5)
```

Accuracy curve and confusion matrix

```
plot(classifier_caret_GLM)
```



Accuracy and confusion matrix

Comment: Overall high accuracy. However, low default prediction accuracy, which is the focus of this project.

```
y_pred_caret_GLM_test_set = predict(classifier_caret_GLM, newdata = test_set, type = "raw")
y_pred_caret_GLM_test_set_P = predict(classifier_caret_GLM, newdata = test_set, type = "prob")[,2] #selecting column 2, which is probability of 1, comparable to GLM probabilities
y_pred_caret_GLM_test_set = unname(as.factor(ifelse(y_pred_caret_GLM_test_set_P > 0.2, 1, 0))) #Manual input
cm_caret_GLM_test_set = confusionMatrix(reference = test_set$loan_status, data = y_pred_caret_GLM_test_set)
cm_caret_GLM_test_set$table
```

```
##          Reference
## Prediction    0    1
##          0 6254  748
##          1  193   52
```

Visualize GLM model performance

Commentary: Comparing the actual to predicted defaults, the cross-validated GLM prediction model is showing most of the inaccuracies in prediction in the data of interest rates above 15%. This could be modeled with higher accuracy through models relying more on clustering data, which we will explore in later sections.

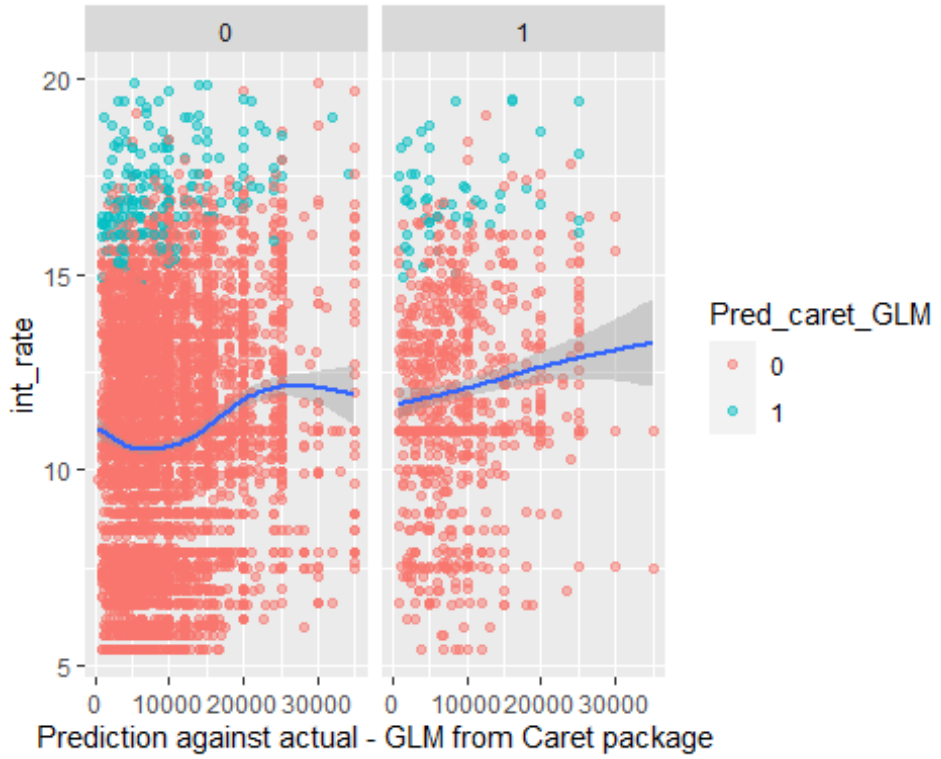
Add predictions to test set

```
test_set_pred_binary = data.frame(test_set_unnormalized,
                                  Pred_caret_GLM = y_pred_caret_GLM_test_set)
```

Borrower's loan amount, interest rate projected against the actual vs predicted defaults.

```
test_set_pred_binary %>%
  ggplot(aes(x = loan_amnt,y = int_rate))+
  geom_point(aes(color = Pred_caret_GLM ), alpha = 0.5)+ #Prediction
  geom_smooth()+
  facet_wrap(~loan_status)+ # Actual
  xlab("Prediction against actual - GLM from Caret package")
```

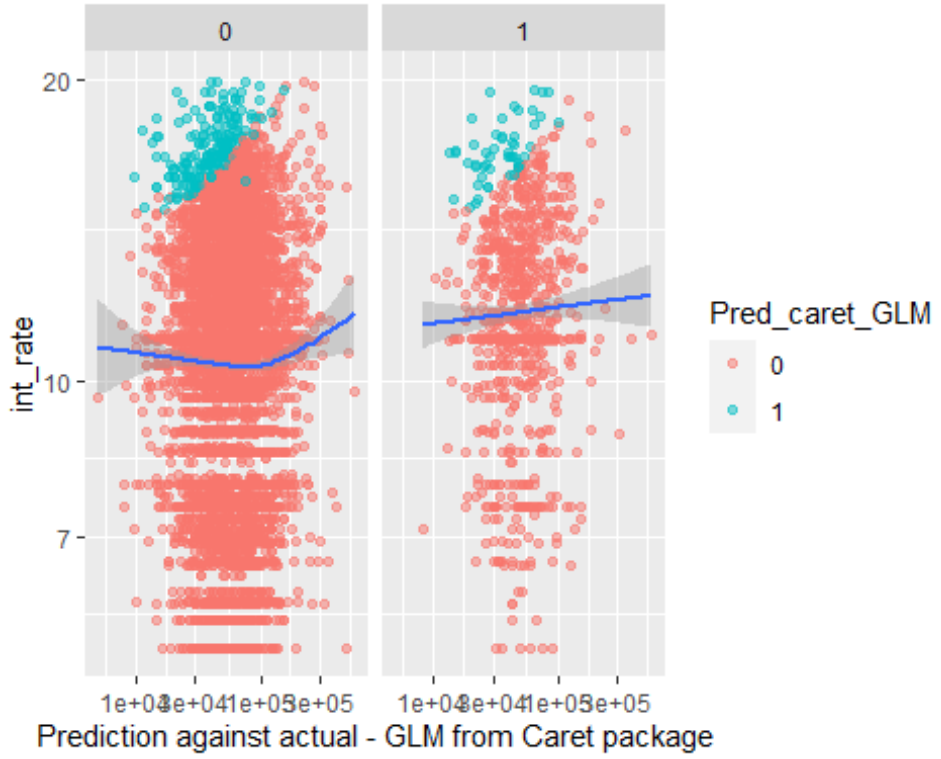
```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



Borrower's annual income, interest rate projected against the actual vs predicted defaults

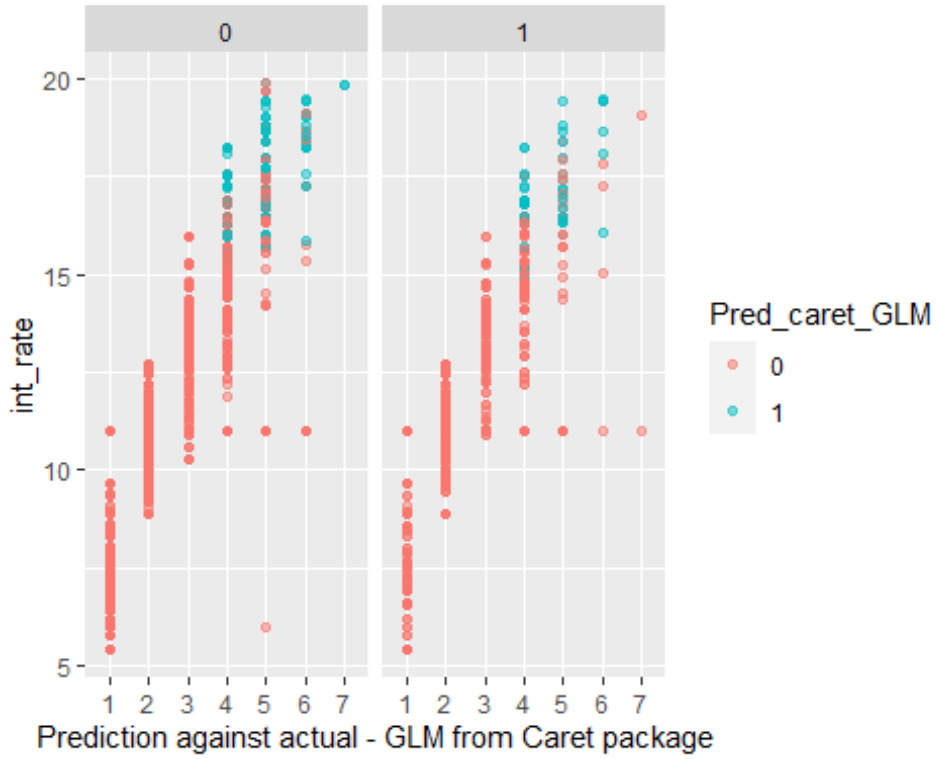
```
test_set_pred_binary %>%
  ggplot(aes(x = annual_inc,y = int_rate))+
  geom_point(aes(color = Pred_caret_GLM), alpha = 0.5)+ #Prediction
  scale_y_continuous(trans = 'log10')+
  scale_x_continuous(trans = 'log10')+
  geom_smooth()+
  facet_wrap(~loan_status)+ # Actual
  xlab("Prediction against actual - GLM from Caret package")
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



Borrower's grade, interest rate projected against the actual vs predicted defaults

```
test_set_pred_binary %>%
  ggplot(aes(x = grade,y = int_rate))+
  geom_point(aes(color = Pred_caret_GLM), alpha = 0.5)+ #Prediction
  facet_wrap(~loan_status)+ # Actual
  xlab("Prediction against actual - GLM from Caret package")
```



K-Fold cross validation

```
folds=createFolds(training_set$loan_status, #dependent variable
                  k=10)                 # number of folds.
cv = lapply(folds,function(x){          # X = each fold
  training_fold=training_set[-x,]     #-x: except for the test fold. Assign a training dataset to each fold
  test_fold=training_set[x,]          #x: test fold
  #To each fold apply the following:
  classifier_log = glm(loan_status~.-home_ownership,family=binomial,data=training_set)
  y_pred = predict(classifier_log, newdata = test_fold[-1]) #Test on the test fold
  cm=table(test_fold[,1],y_pred)     #Summarize the confusion matrix for test folds
  accuracy = (cm[1,1]+cm[2,2]) / (cm[1,1]+cm[1,2]+cm[2,1]+cm[2,2]) #calculate the accuracy for each fold
  return(accuracy)
})
```

Return list of accuracies on folds above

Commentary: Similar to our observation from the analysis above, the default prediction can be enhanced by looking at other modeling techniques.

```
min_accuracy=min(as.numeric(cv))
max_accuracy=max(as.numeric(cv))
average_accuracy=mean(as.numeric(cv))
data.frame(min_accuracy,max_accuracy,average_accuracy)

##   min_accuracy max_accuracy average_accuracy
## 1            0         0.5            0.45
```

Model: K-Nearest neighbors regression “KNN”

Configure train() with cross validation and parallel processing

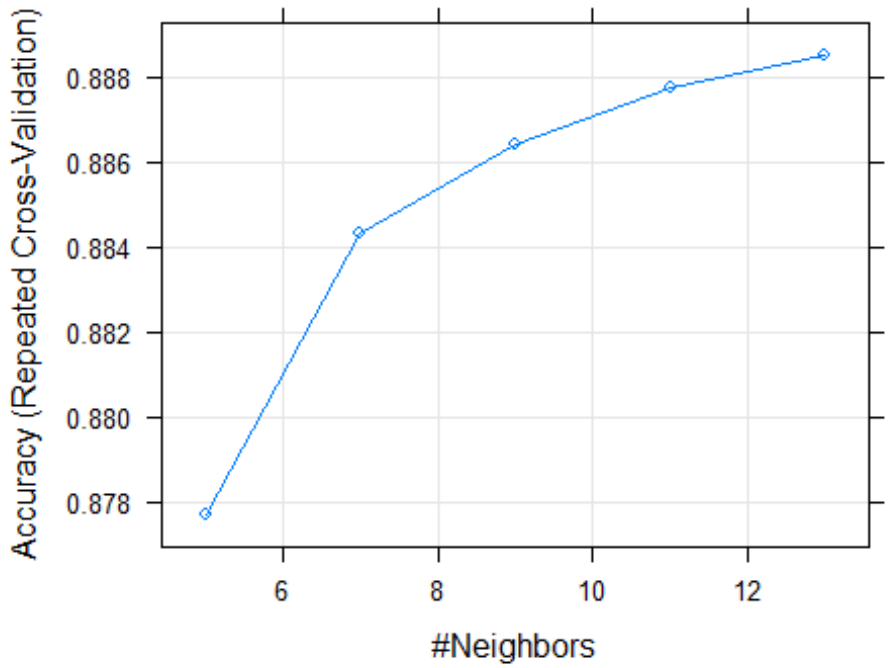
```
trctrl = trainControl(method = "repeatedcv", repeats = 5, number = 5)
```

Fitting KNN model

```
set.seed(1)
classifier_knn = train(loan_status ~. -emp_length - loan_amnt - home_ownership -age, data = training_set
,
                       method = "knn",
                       trControl=trctrl,
                       tuneLength = 5 #maximum grid size
                       )
classifier_knn

## k-Nearest Neighbors
##
## 21739 samples
##     7 predictor
##     2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 5 times)
## Summary of sample sizes: 17391, 17391, 17391, 17391, 17392, 17392, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    5  0.8776945  0.0132702647
##    7  0.8843001  0.0132028672
##    9  0.8864161  0.0034561064
##   11  0.8877501  0.0024314464
##   13  0.8885045  0.0003553564
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 13.
```

Accuracy curve - KNN

```
plot(classifier_knn)
```



Accuracy curve and confusion matrix

Commentary: The KNN model is demonstrating better accuracy in prediction compared to GLM models, with the errors more spread across the data. However, there are better models that can demonstrate higher accuracy rates, which we are exploring next.

```
y_pred_knn_test_set = predict(classifier_knn, newdata = test_set, type = "raw")
y_pred_knn_test_set_P = predict(classifier_knn, newdata = test_set, type = "prob")[,2] #selecting column 2, which is probability of 1, comparable to GLM probabilities
y_pred_knn_test_set = unname(as.factor(ifelse(y_pred_knn_test_set_P > 0.17, 1, 0))) #Manual input
cm_knn_test_set = confusionMatrix(reference = test_set$loan_status, data = y_pred_knn_test_set)
cm_knn_test_set$table

##          Reference
## Prediction    0    1
##          0 5162  566
##          1 1285  234
```

Visualize the model – KNN

Add predictions to test set

```
test_set_pred_binary = data.frame(test_set_unnormalized,
                                  Pred_KNN = y_pred_knn_test_set)
```

Borrower's loan amount, interest rate projected against the actual vs predicted defaults

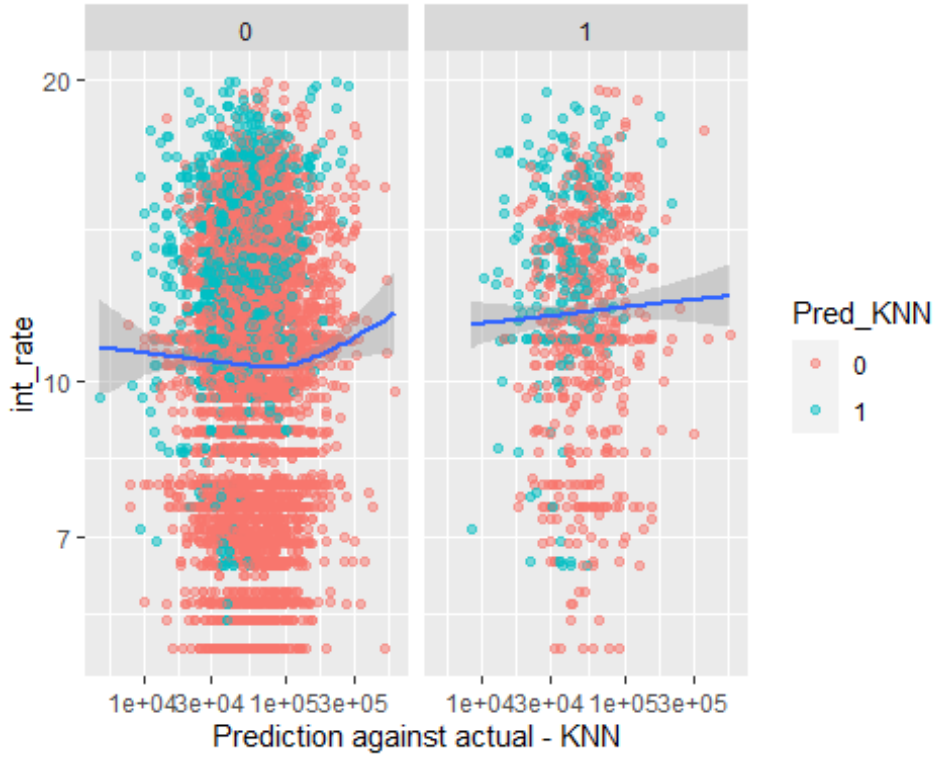
```
test_set_pred_binary %>%
  ggplot(aes(x = loan_amnt,y = int_rate))+
  geom_point(aes(color = Pred_KNN ), alpha = 0.5)+ #Prediction
  geom_smooth()+
  facet_wrap(~loan_status)+ # Actual
  xlab("Prediction against actual - KNN")
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

Borrower's annual income, interest rate projected against the actual vs predicted defaults

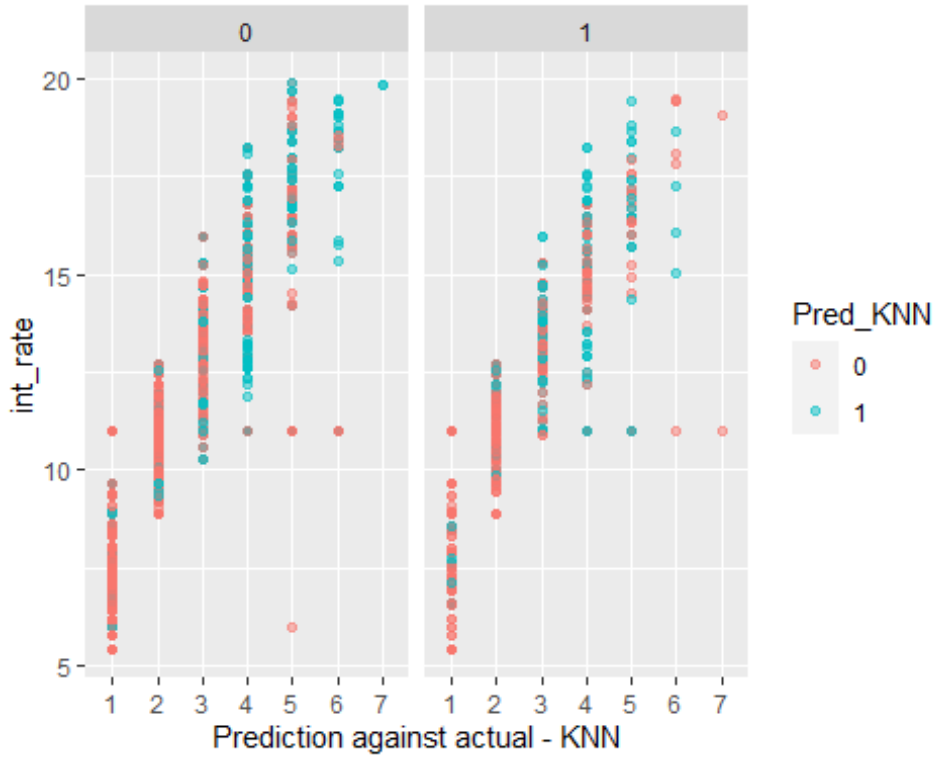
```
test_set_pred_binary %>%
  ggplot(aes(x = annual_inc,y = int_rate))+
  geom_point(aes(color = Pred_KNN), alpha = 0.5)+ #Prediction
  scale_y_continuous(trans = 'log10')+
  scale_x_continuous(trans = 'log10')+
  geom_smooth()+
  facet_wrap(~loan_status)+ # Actual
  xlab("Prediction against actual - KNN")
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



Borrower's grade, interest rate projected against the actual vs predicted defaults

```
test_set_pred_binary %>%
  ggplot(aes(x = grade,y = int_rate))+
  geom_point(aes(color = Pred_KNN), alpha = 0.5)+ #Prediction
  facet_wrap(~loan_status)+ # Actual
  xlab("Prediction against actual - KNN")
```



Model: Support Vector Machines with Radial Basis “SVM”

Fitting SVM-Radial for non-linear relationship

```
classifier_svm = svm(formula = loan_status ~ .,
          data = training_set,
          type = 'C-classification',
          kernel = 'radial',
          probability = TRUE)

# SVM with cross validation, but very slow on caret package
# classifier_svm = train(loan_status ~., data = training_set,
#                        method = "svmRadial",
#                        trControl=trctrl,
#                        tuneLength = 5) #maximum grid size
summary(classifier_svm)

##
## Call:
## svm(formula = loan_status ~ ., data = training_set, type = "C-classification",
##     kernel = "radial", probability = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  6750
##
##  ( 4352 2398 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

Accuracy curve and confusion matrix

Commentary: Similar to preceding models, the 'default' accuracy outcome from SVM is not that better. However, the observed inaccuracies seem to improve, being spread across the data and not scattered around certain clusters similar to earlier models.

Therefore, we will look into further models that can better sub-group the data, and better predict defaults.

```
y_pred_svm_test_set = predict(classifier_svm, newdata = test_set)
y_pred_svm_test_set_P = predict(classifier_svm, newdata = test_set, probability = TRUE ) #selecting col
umn 2, which is probability of 1, comparable to GLM probabilities
y_pred_svm_test_set_P = as.data.frame(
                    attr(y_pred_svm_test_set_P, "probabilities") # added step to extract probabilities
                    )[,2] #Extract column 2 for probabilities of 1
y_pred_svm_test_set_P_x = y_pred_svm_test_set_P * 1000 / 2
y_pred_svm_test_set = unname(as.factor(ifelse(y_pred_svm_test_set_P_x > 55.18, 1, 0))) #Manual input
cm_svm_test_set = confusionMatrix(reference = test_set$loan_status, data = y_pred_svm_test_set)
cm_svm_test_set$table

##          Reference
## Prediction    0    1
##          0 2981  322
##          1 3466  478
```

Visualize the model - SVM

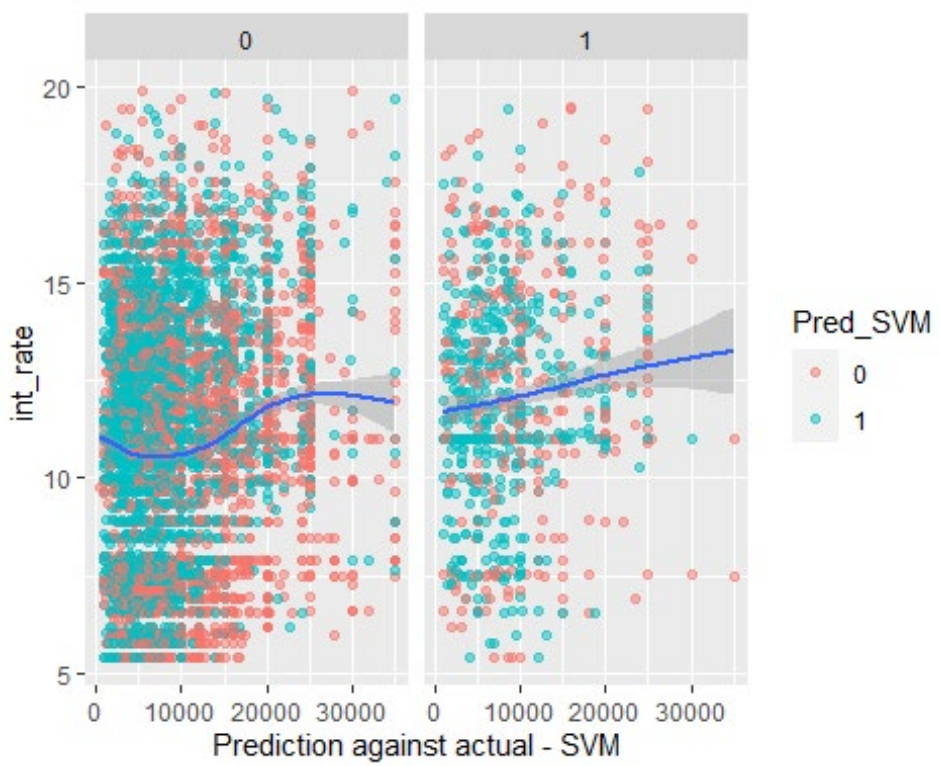
Commentary: SVM model is demonstrating better ability to spread across the data, but didn't do better either in terms of default prediction.

Add predictions to test set

```
test_set_pred_binary = data.frame(test_set_unnormalized,
                                  Pred_SVM = y_pred_svm_test_set)
```

Borrower's loan amount, interest rate projected against the actual vs predicted defaults

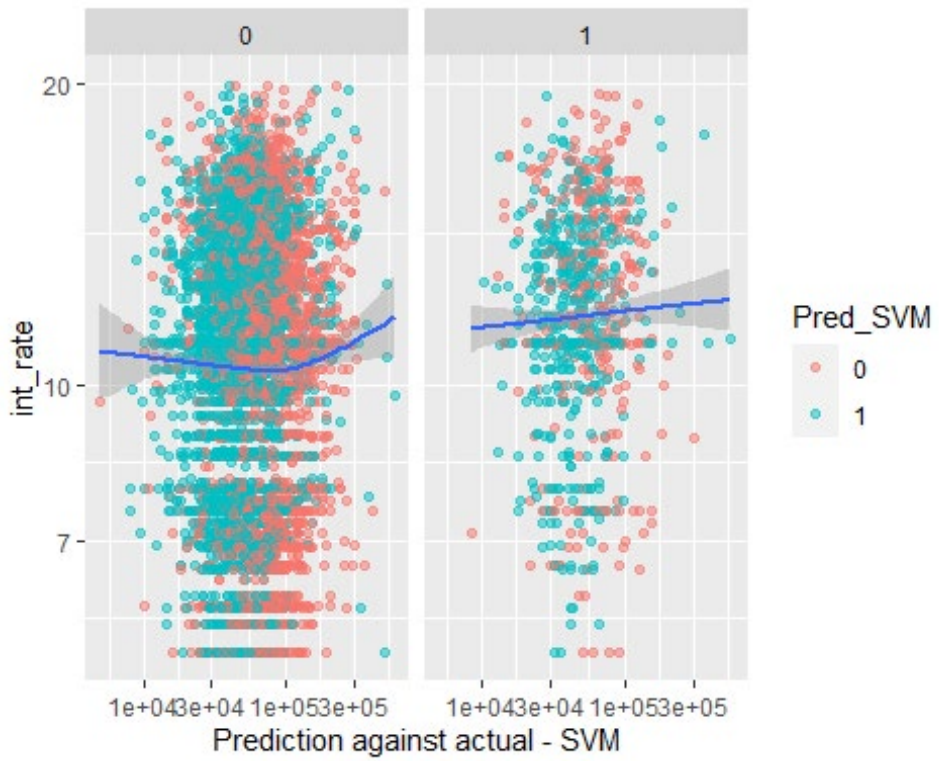
```
test_set_pred_binary %>%
  ggplot(aes(x = loan_amnt,y = int_rate))+
  geom_point(aes(color = Pred_SVM ), alpha = 0.5)+ #Prediction
  geom_smooth()+
  facet_wrap(~loan_status)+ # Actual
  xlab("Prediction against actual - SVM")

## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



Borrower's annual income, interest rate projected against the actual vs predicted defaults

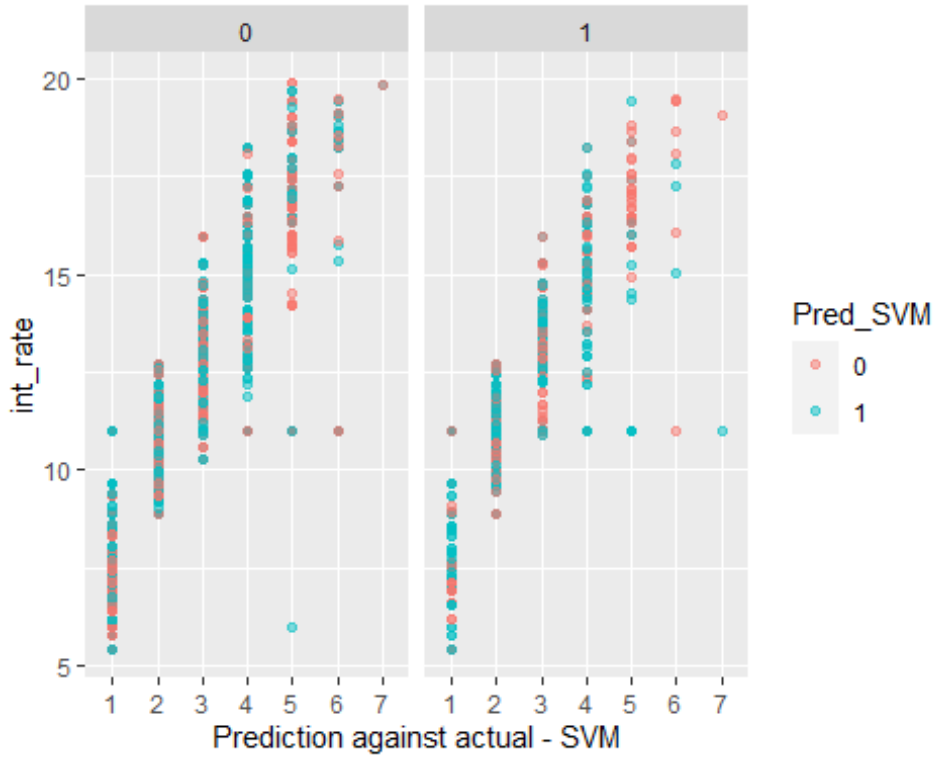
```
test_set_pred_binary %>%
  ggplot(aes(x = annual_inc,y = int_rate))+
  geom_point(aes(color = Pred_SVM), alpha = 0.5)+ #Prediction
  scale_y_continuous(trans = 'log10')+
  scale_x_continuous(trans = 'log10')+
  geom_smooth()+
  facet_wrap(~loan_status)+ # Actual
  xlab("Prediction against actual - SVM")
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



Borrower's grade, interest rate projected against the actual vs predicted defaults

```
test_set_pred_binary %>%
  ggplot(aes(x = grade,y = int_rate))+
  geom_point(aes(color = Pred_SVM), alpha = 0.5)+ #Prediction
  facet_wrap(~loan_status)+ # Actual
  xlab("Prediction against actual - SVM")
```



Model: Neural Networks

Fit model - NNET

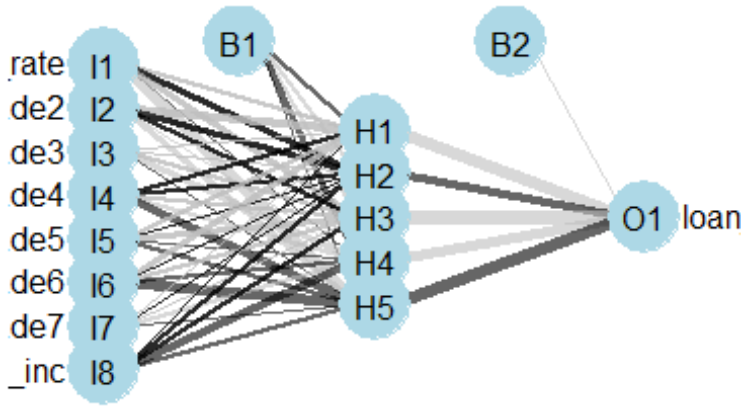
```
# very slow, more accurate than the following model, but still below expectations with around 20% specifi
city rate.
# data_nnt = data.matrix(training_set) #convert data to numeric matrix, so neuralnet can accept it.
# classifier_nnet = neuralnet(loan_status~., data = data_nnt,
#               algorithm = "rprop+", #manual input
#               hidden=c(10,3),
#               cut-off threshold=0.1,
#               stepmax = 1e+06)
# plot(classifier_nnet)
```

Caret NNET - more efficient model from caret package with cross validation

```
myGrid = expand.grid(.decay=c(0.5, 0.1), .size=c(4,5,6))
classifier_nnet = train(loan_status~. -emp_length - loan_amnt - home_ownership -age,
                        data = training_set,
                        method = "nnet",
                        tuneGrid = myGrid,
                        preProcess = c("center"),
                        linout=0, #for classification
                        maxit=300,
                        trace=F)
classifier_nnet

## Neural Network
##
## 21739 samples
##     7 predictor
##     2 classes: '0', '1'
##
## Pre-processing: centered (8)
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 21739, 21739, 21739, 21739, 21739, 21739, ...
## Resampling results across tuning parameters:
##
##   decay  size  Accuracy   Kappa
##   0.1    4     0.8899672  2.556991e-04
##   0.1    5     0.8899171  8.523799e-04
##   0.1    6     0.8898469  1.309310e-03
##   0.5    4     0.8900573  0.000000e+00
##   0.5    5     0.8900624  8.225041e-05
##   0.5    6     0.8900573  0.000000e+00
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were size = 5 and decay = 0.5.
```

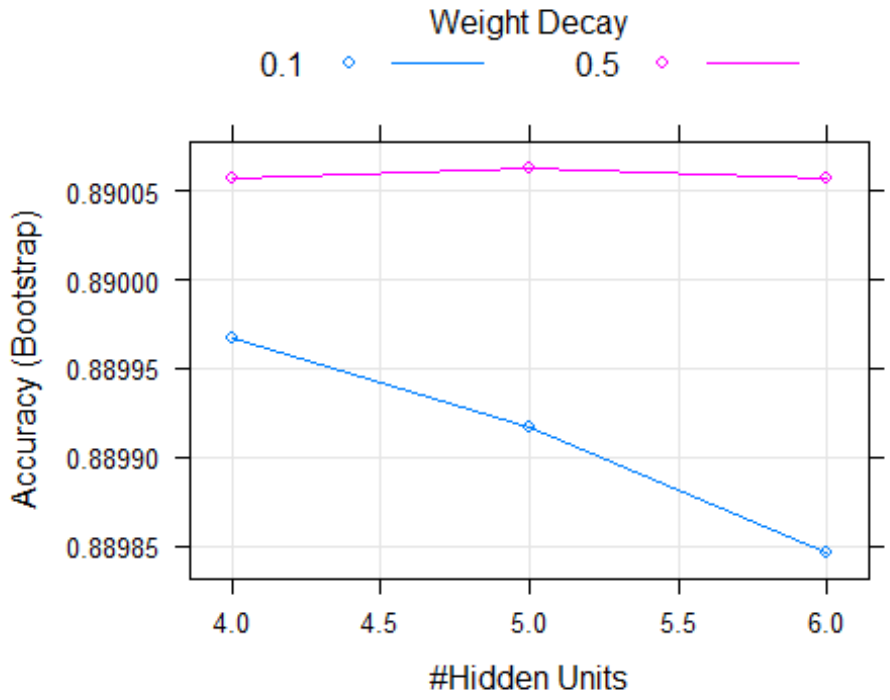
Visualize the network and confusion matrix

```
library(NeuralNetTools)
plotnet(classifier_nnet, alpha = 0.6)
```



Accuracy curve

```
plot(classifier_nnet)
```



Prediction - binary (0,1) before adjustment

```
y_pred_nnet_test_set = predict(classifier_nnet, newdata = test_set)
```

Prediction - probabilities

```
y_pred_nnet_test_set_P = predict(classifier_nnet, newdata = test_set, type = "prob")[,2]
```

Predictions in binary (0,1) form after adjustment through manual input

```
y_pred_nnet_test_set = unname(as.factor(ifelse(y_pred_nnet_test_set_P > 0.15, 1, 0))) #Manual input
```

Run the Confusion Matrix

```
cm_nnet_test_set = confusionMatrix(reference = test_set$loan_status, data = y_pred_nnet_test_set)
```

Display confusion matrix table

```
cm_nnet_test_set$table
```

```
##          Reference
## Prediction    0    1
##          0 4959  484
##          1 1488  316
```

Visualize the model - Neural networks

Commentary: Neural networks model seem to get to a prediction outcome similar to GLM models.

Add predictions to test set

```
test_set_pred_binary = data.frame(test_set_unnormalized,
                                  Pred_NNET = y_pred_nnet_test_set)
```

Borrower's loan amount, interest rate projected against the actual vs predicted defaults

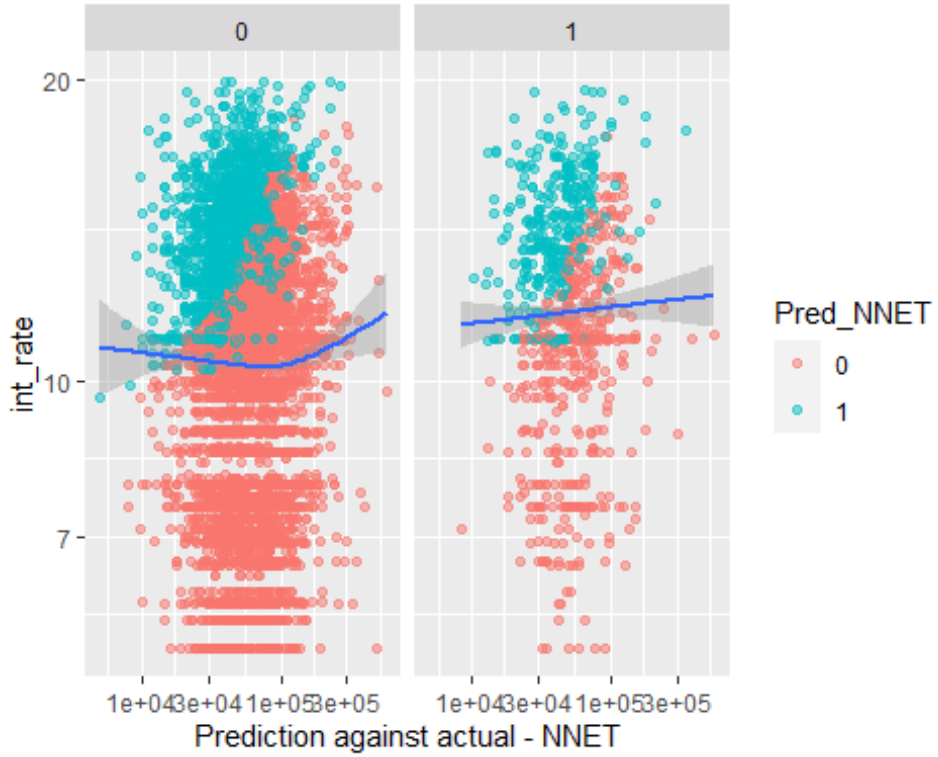
```
test_set_pred_binary %>%
  ggplot(aes(x = loan_amnt,y = int_rate))+
  geom_point(aes(color = Pred_NNET ), alpha = 0.5)+ #Prediction
  geom_smooth()+
  facet_wrap(~loan_status)+ # Actual
  xlab("Prediction against actual - NNET")
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

Borrower's annual income, interest rate projected against the actual vs predicted defaults

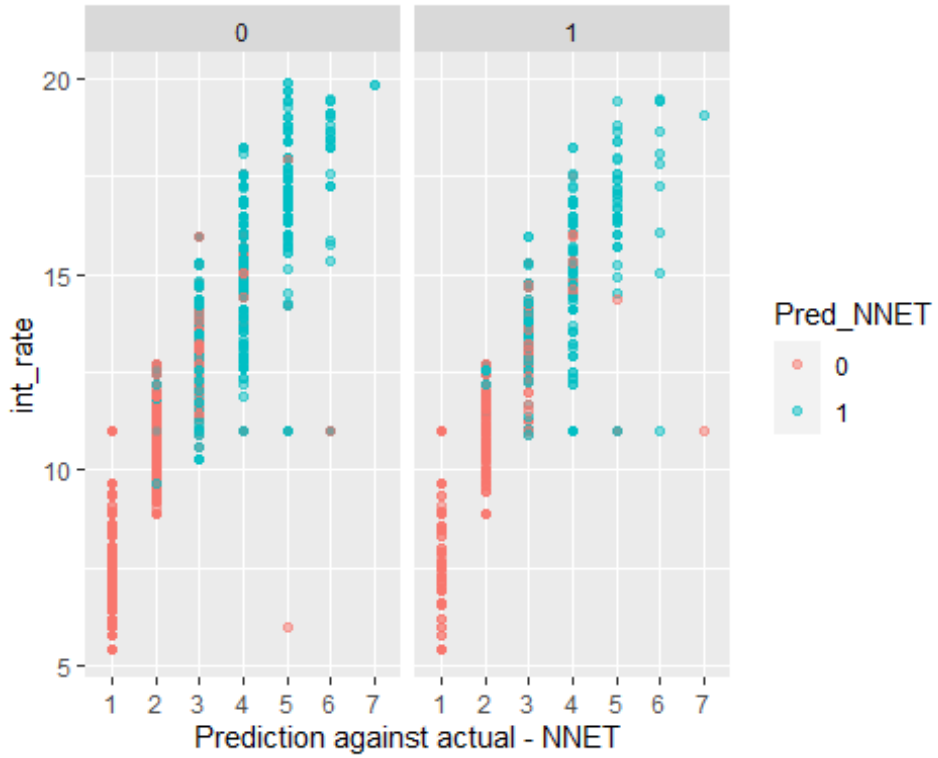
```
test_set_pred_binary %>%
  ggplot(aes(x = annual_inc,y = int_rate))+
  geom_point(aes(color = Pred_NNET), alpha = 0.5)+ #Prediction
  scale_y_continuous(trans = 'log10')+
  scale_x_continuous(trans = 'log10')+
  geom_smooth()+
  facet_wrap(~loan_status)+ # Actual
  xlab("Prediction against actual - NNET")
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



Borrower's grade, interest rate projected against the actual vs predicted defaults

```
test_set_pred_binary %>%
  ggplot(aes(x = grade,y = int_rate))+
  geom_point(aes(color = Pred_NNET), alpha = 0.5)+ #Prediction
  facet_wrap(~loan_status)+ # Actual
  xlab("Prediction against actual - NNET")
```



Model: Naive Bayes

Fit model - Naive Bayes

```
# Caret - time consuming model
#classifier_nb = train(loan_status~. , data = training_set, method = "nb", metric = "Accuracy", maxit = 150,
trControl = trctrl, tuneLength = 5, preProcess = c("center"))
# classifier_nb
```

Caret Naive Bayes

```
classifier_nb = naiveBayes(x=training_set[,-1], #Manually remove dependent variable
                           y=training_set$loan_status, #It should be a vector, so no brackets used
                           usekernel = TRUE)
```

Prediction - binary (0,1) before adjustment

```
y_pred_nb_test_set = predict(classifier_nb, newdata = test_set)
```

Prediction - probabilities

```
y_pred_nb_test_set_P = predict(classifier_nb, newdata = test_set, type = "raw")[,2]
```

Predictions in binary (0,1) form after adjustment through manual input

```
y_pred_nb_test_set = unname(as.factor(ifelse(y_pred_nb_test_set_P > 0.2, 1, 0))) #Manual input
```

Run the Confusion Matrix

```
cm_nb_test_set = confusionMatrix(reference = test_set$loan_status, data = y_pred_nb_test_set)
cm_nb_test_set$table
```

```
##          Reference
## Prediction    0    1
##          0 4988  511
##          1 1459  289
```

Visualize the model - Naive Bayes

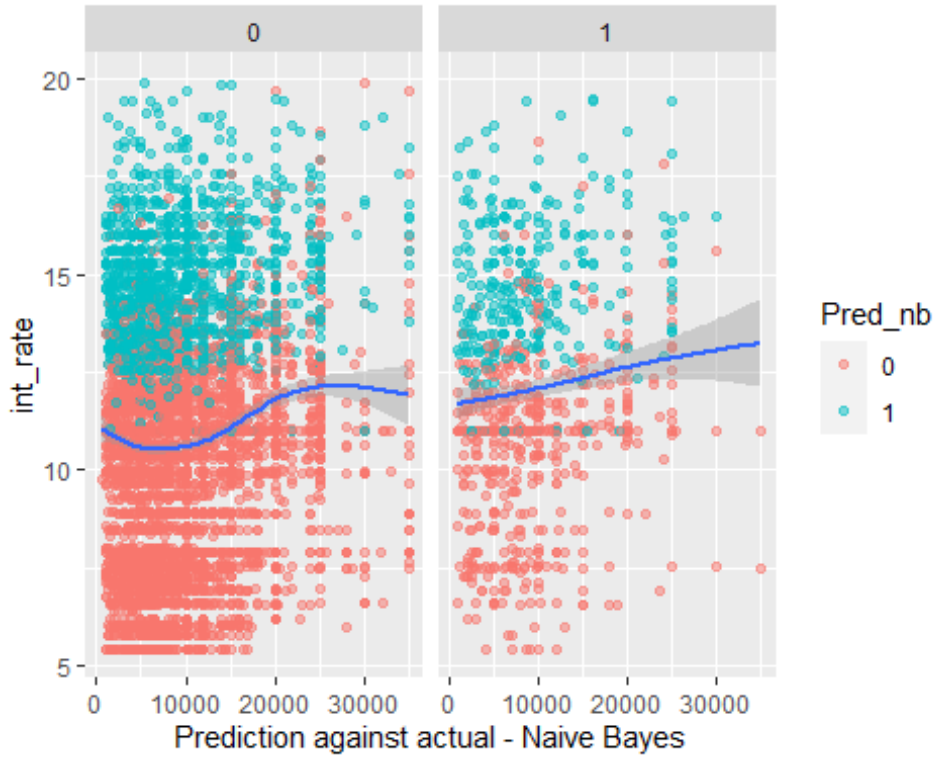
Commentary: Naïve Bayes model, again gives prediction accuracy level similar to GLM models.

Add predictions to test set

```
test_set_pred_binary = data.frame(test_set_unnormalized,
                                  Pred_nb = y_pred_nb_test_set)
```

Borrower's loan amount, interest rate projected against the actual vs predicted defaults

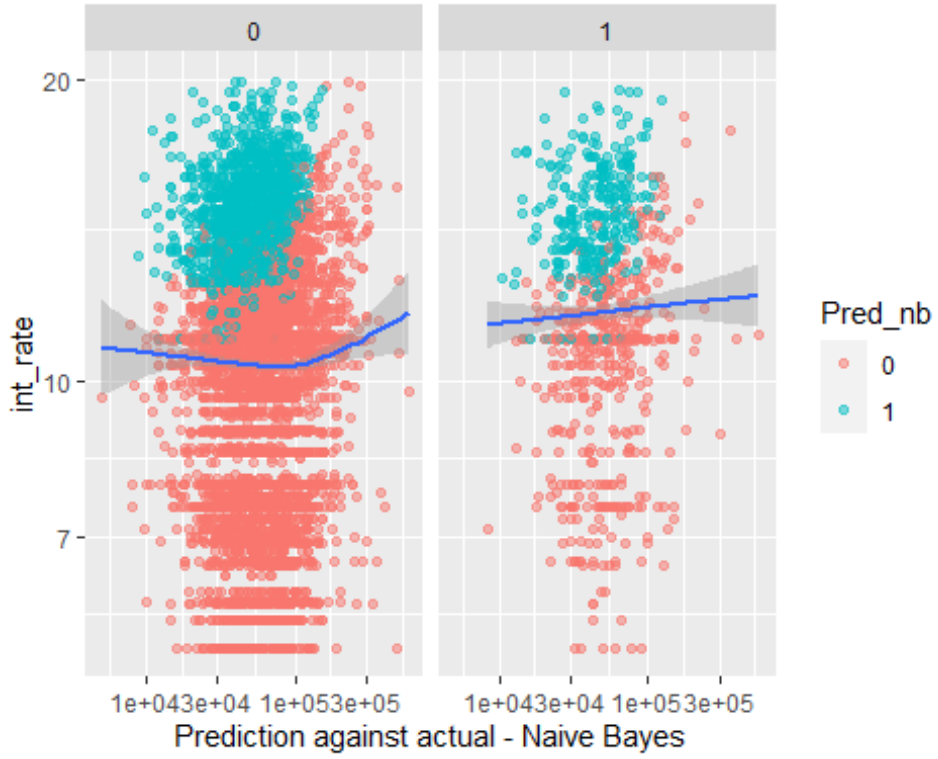
```
test_set_pred_binary %>%
  ggplot(aes(x = loan_amnt,y = int_rate))+
  geom_point(aes(color = Pred_nb ), alpha = 0.5)+ #Prediction
  geom_smooth()+
  facet_wrap(~loan_status)+ # Actual
  xlab("Prediction against actual - Naive Bayes")
```



Borrower's annual income, interest rate projected against the actual vs predicted defaults

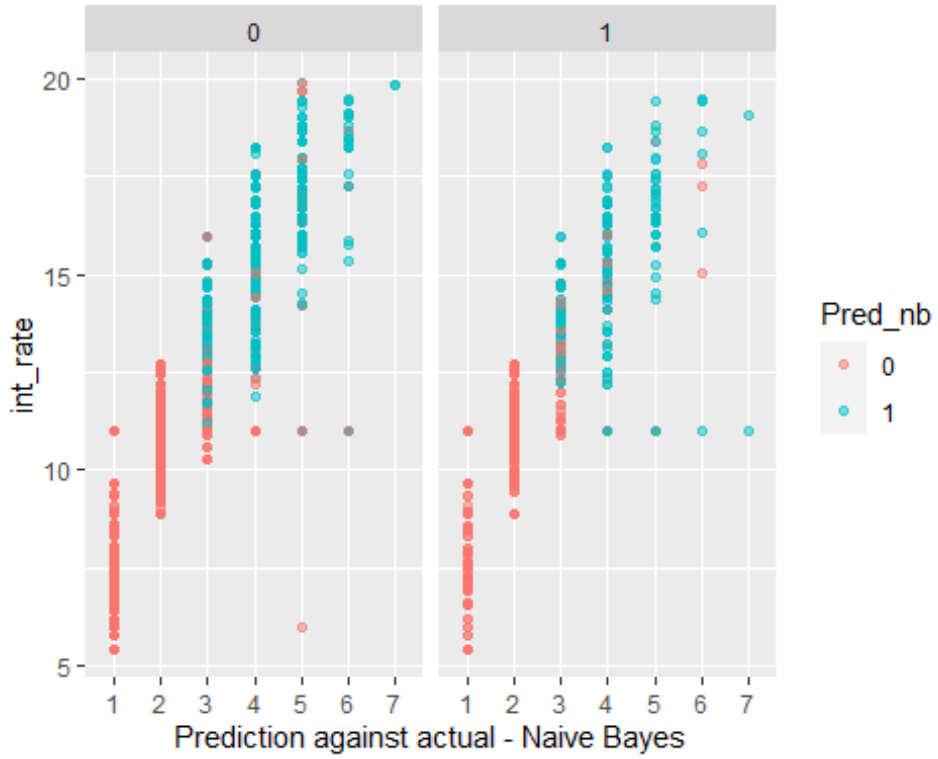
```
test_set_pred_binary %>%
  ggplot(aes(x = annual_inc,y = int_rate))+
  geom_point(aes(color = Pred_nb), alpha = 0.5)+ #Prediction
  scale_y_continuous(trans = 'log10')+
  scale_x_continuous(trans = 'log10')+
  geom_smooth()+
  facet_wrap(~loan_status)+ # Actual
  xlab("Prediction against actual - Naive Bayes")
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



Borrower's grade, interest rate projected against the actual vs predicted defaults

```
test_set_pred_binary %>%
  ggplot(aes(x = grade,y = int_rate))+
  geom_point(aes(color = Pred_nb), alpha = 0.5)+ #Prediction
  facet_wrap(~loan_status)+ # Actual
  xlab("Prediction against actual - Naive Bayes")
```



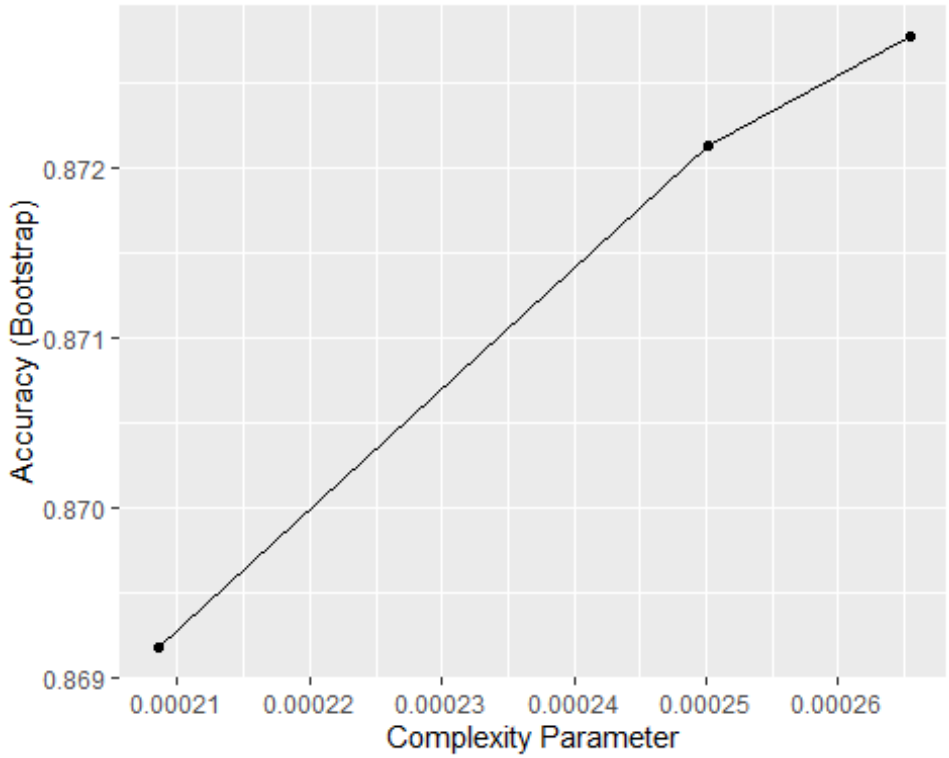
Model: Decision Tree

Fit model - Decision Tree

```
# less accurate
# library(rpart)
# rpt1 = rpart(loan_status~., data = training_set,
#              method = "class") #classification regression
# classifier_rf = rpt1
#
# rpart.plot(classifier_dt)
# plot(classifier_dt)
```

Fit the decision tree model

```
classifier_dt = train(loan_status ~ int_rate + grade + annual_inc,
                      method = "rpart",
                      data = training_set)

ggplot(classifier_dt) # y = "RMSE - bootstrap": minimum error
```




```
confusionMatrix(predict(classifier_dt, test_set), test_set$loan_status)

## Confusion Matrix and Statistics
##
##          Reference
## Prediction    0    1
##          0 6447  800
##          1    0    0
```

```
##
##               Accuracy : 0.8896
##                 95% CI : (0.8822, 0.8967)
##    No Information Rate : 0.8896
##    P-Value [Acc > NIR] : 0.5094
##
##                  Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 1.0000
##             Specificity : 0.0000
##          Pos Pred Value : 0.8896
##          Neg Pred Value :    NaN
##              Prevalence : 0.8896
##          Detection Rate : 0.8896
##    Detection Prevalence : 1.0000
##       Balanced Accuracy : 0.5000
##
##        'Positive' Class : 0
##
```

Accuracy and confusion matrix

```
y_pred_dt_test_set = predict(classifier_dt, newdata = test_set)
y_pred_dt_test_set_P = predict(classifier_dt, newdata = test_set, type = "prob")[,2]
y_pred_dt_test_set = unname(as.factor(ifelse(y_pred_dt_test_set_P > 0.1, 1, 0))) #Manual input
cm_dt_test_set = confusionMatrix(reference = test_set$loan_status, data = y_pred_dt_test_set)

## Warning in confusionMatrix.default(reference = test_set$loan_status, data =
## y_pred_dt_test_set): Levels are not in the same order for reference and data.
## Refactoring data to match.

cm_dt_test_set$table

##          Reference
## Prediction    0    1
##          0    0    0
##          1 6447  800
```

Visualize the model - Decision Tree

Commentary: Decision tree model doesn't sound to be appropriate for the sake of this exercise.

Add predictions to test set

```
test_set_pred_binary = data.frame(test_set_unnormalized,
                                  Pred_dt = y_pred_dt_test_set)
```

Borrower's loan amount, interest rate projected against the actual vs predicted defaults

Commentary: Decision tree with the cross validation version has split the data into two halves, giving more weight to the non-defaults, which is not suitable for our prediction.

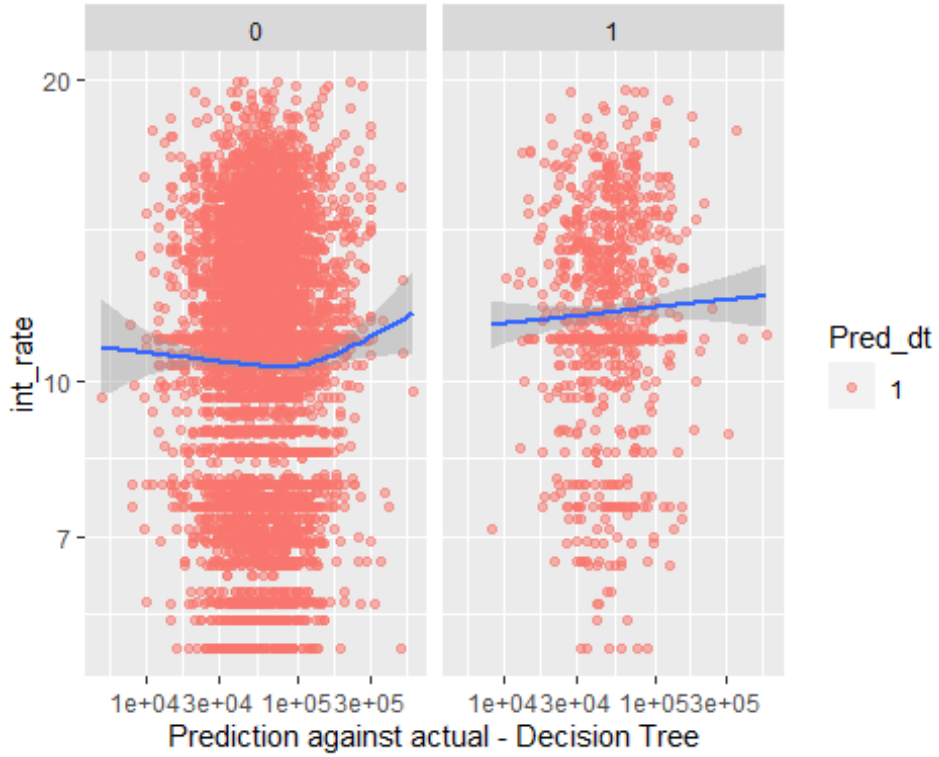
```
test_set_pred_binary %>%
  ggplot(aes(x = loan_amnt,y = int_rate))+
  geom_point(aes(color = Pred_dt ), alpha = 0.5)+ #Prediction
  geom_smooth()+
  facet_wrap(~loan_status)+ # Actual
  xlab("Prediction against actual - Decision Tree")
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

Borrower's annual income, interest rate projected against the actual vs predicted defaults

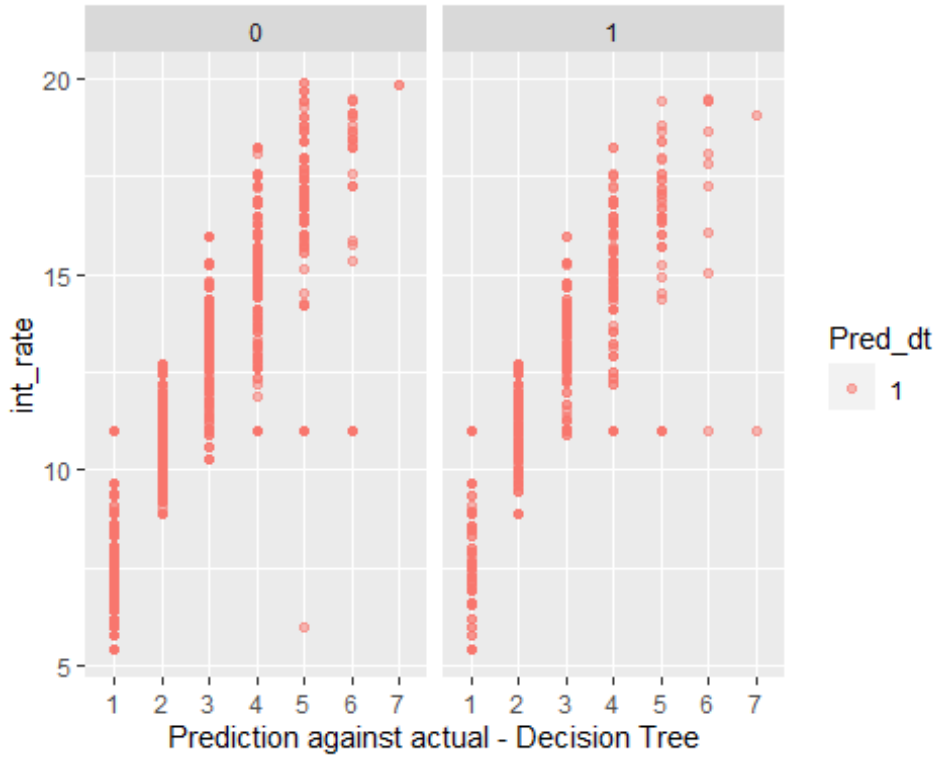
```
test_set_pred_binary %>%
  ggplot(aes(x = annual_inc,y = int_rate))+
  geom_point(aes(color = Pred_dt), alpha = 0.5)+ #Prediction
  scale_y_continuous(trans = 'log10')+
  scale_x_continuous(trans = 'log10')+
  geom_smooth()+
  facet_wrap(~loan_status)+ # Actual
  xlab("Prediction against actual - Decision Tree")
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



Borrower's grade, interest rate projected against the actual vs predicted defaults

```
test_set_pred_binary %>%
  ggplot(aes(x = grade,y = int_rate))+
  geom_point(aes(color = Pred_dt), alpha = 0.5)+ #Prediction
  facet_wrap(~loan_status)+ # Actual
  xlab("Prediction against actual - Decision Tree")
```



Model: Random forest

Fit model - Random Forest

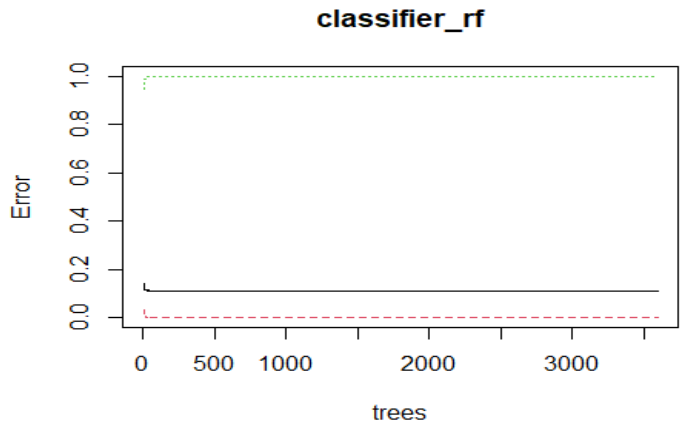
```
# set.seed(1)
## Fit model with cross validation - not perfect, as it's focused with accuracy and not with specific
ity
# classifier_rf = train(loan_status ~. -emp_length - loan_amnt - home_ownership -age,
#                       method = "Rborist",
#                       tuneGrid = data.frame(predFixed = 2, minNode = c(3, 50)),
#                       data = training_set)
```

Fit model - even better outcome, using the number of trees option

```
library(randomForest)
classifier_rf = randomForest(loan_status ~ int_rate + grade + annual_inc,
                             data=test_set,
                             ntree = 3600) #Manual input
```

Accuracy curve and confusion matrix

```
plot(classifier_rf)
```




```
y_pred_rf_test_set = predict(classifier_rf, newdata = test_set)
y_pred_rf_test_set_P = predict(classifier_rf, newdata = test_set, type = "prob")[,2]
y_pred_rf_test_set = unname(as.factor(ifelse(y_pred_rf_test_set_P > 0.025, 1, 0))) #Manual input
cm_rf_test_set = confusionMatrix(reference = test_set$loan_status, data = y_pred_rf_test_set)
cm_rf_test_set$table

##          Reference
## Prediction    0    1
##          0 5588  114
##          1  859  686
```

Visualize the model - Random Forest

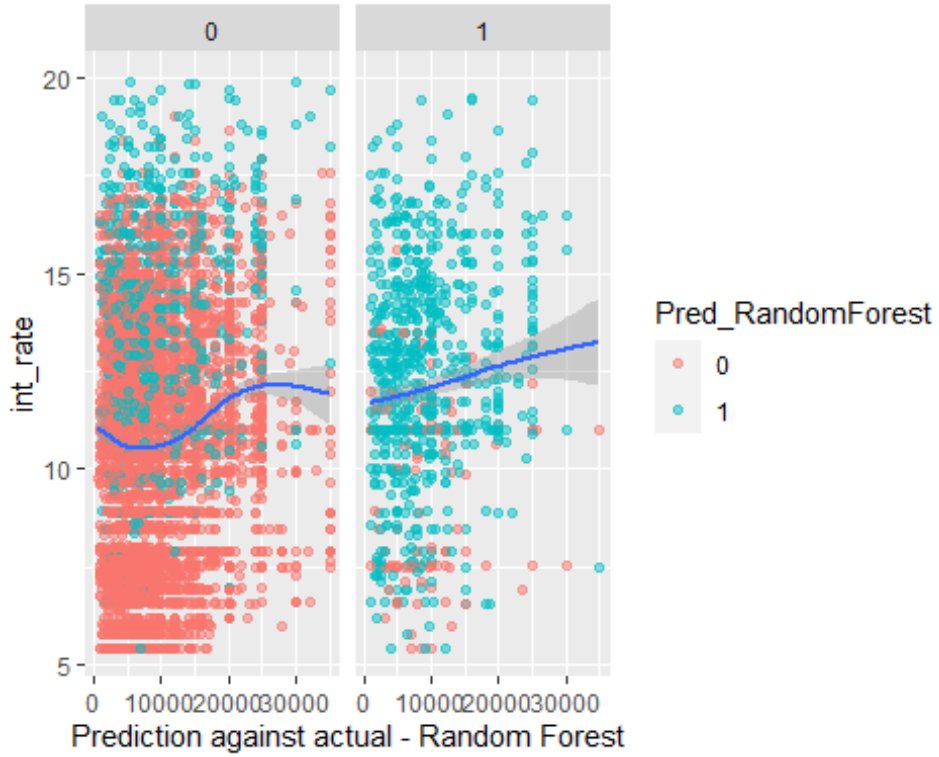
Commentary: The Random Forest model demonstrates improved pattern recognition, and therefore better accuracy, compared to earlier models, especially with default prediction.

Add predictions to test set

```
test_set_pred_binary = data.frame(test_set_unnormalized,
                                  Pred_RandomForest = y_pred_rf_test_set)
```

Borrower's loan amount, interest rate projected against the actual vs predicted defaults

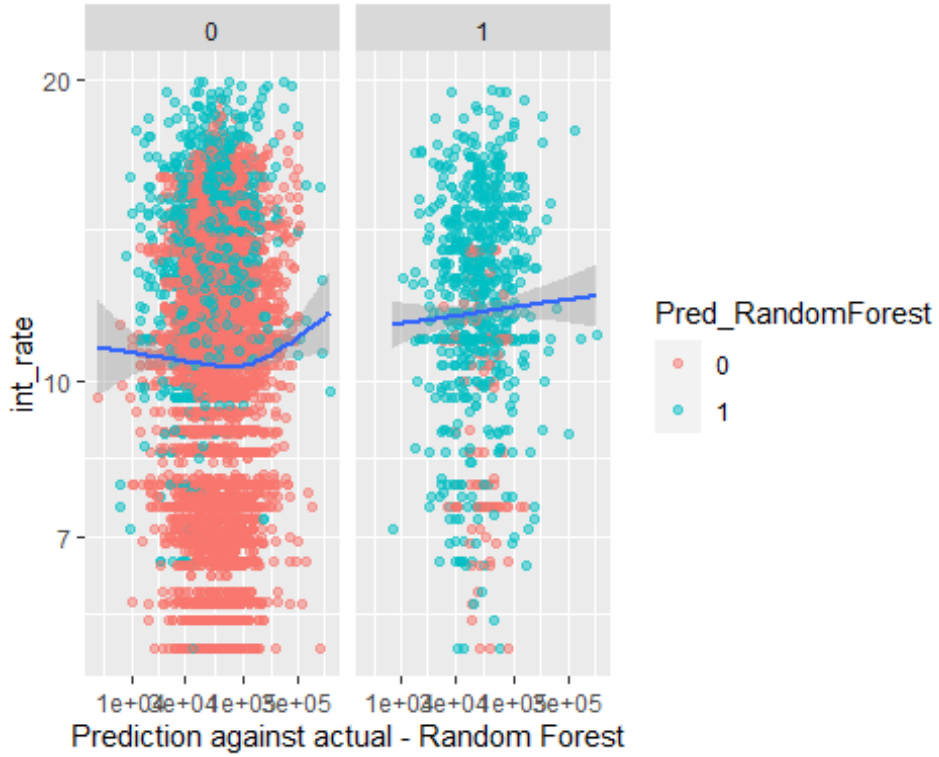
```
test_set_pred_binary %>%
  ggplot(aes(x = loan_amnt,y = int_rate))+
  geom_point(aes(color = Pred_RandomForest ), alpha = 0.5)+ #Prediction
  geom_smooth()+
  facet_wrap(~loan_status)+ # Actual
  xlab("Prediction against actual - Random Forest")

## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



Borrower's annual income, interest rate projected against the actual vs predicted defaults

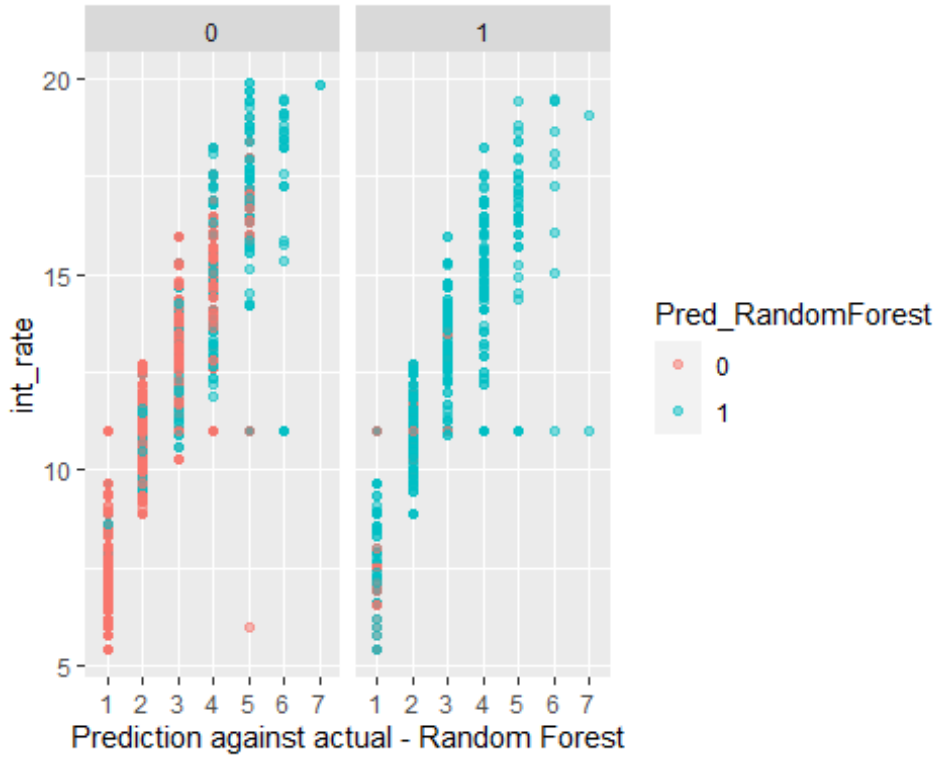
```
test_set_pred_binary %>%
  ggplot(aes(x = annual_inc,y = int_rate))+
  geom_point(aes(color = Pred_RandomForest), alpha = 0.5)+ #Prediction
  scale_y_continuous(trans = 'log10')+
  scale_x_continuous(trans = 'log10')+
  geom_smooth()+
  facet_wrap(~loan_status)+ # Actual
  xlab("Prediction against actual - Random Forest")
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



Borrower's grade, interest rate projected against the actual vs predicted defaults

```
test_set_pred_binary %>%
  ggplot(aes(x = grade,y = int_rate))+
  geom_point(aes(color = Pred_RandomForest), alpha = 0.5)+ #Prediction
  facet_wrap(~loan_status)+ # Actual
  xlab("Prediction against actual - Random Forest")
```



Comparing different models' accuracies and final PDs

Commentary: Comparing the preceding models in terms of overall accuracy and specificity rates, random forest model stands out with 86% overall accuracy rate and similar outcome for specificity rate.

Specificity rate is of higher concern for the sake of this project, given the focus is on more on predicting defaults, rather than just the overall model accuracy.

The accuracy level of (GLMs, KNN, SVM, NB, Neural Networks) prediction models lifts off by at least +10%, reaching out to 83-88% of accuracy in the test environment. This is achievable by lowering the cut-off threshold, which lowers the PDs. This accuracy outcome is reasonable, given these models ability to predict the non-defaults more accurately, which is more prevalent in this dataset with the low default rates.

Add predictions to test set - all models update

```
test_set_pred_binary = data.frame(test_set_unnormalized,
                                  Pred_GLM_log = y_pred_log_test_set,
                                  Pred_GLM_logit = y_pred_logit_test_set,
                                  Pred_GLM_cloglog = y_pred_cloglog_test_set,
                                  Pred_GLM_probit = y_pred_probit_test_set,
                                  Pred_KNN = y_pred_knn_test_set,
                                  Pred_RandomForest = y_pred_rf_test_set,
                                  Pred_NNET = y_pred_nnet_test_set,
                                  Pred_nb = y_pred_nb_test_set,
                                  Pred_dt = y_pred_dt_test_set)
```

Summarize table of the confusion matrix accuracy

```
Model_names = c("GLM_Log","GLM_Logit","GLM_Cloglog","GLM_Probit", "KNN", "SVM", "Neural_Networks", "Naive Bayes", "Decision Tree", "Random Forest")
Accuracy_ = c(cm_log_test_set$overall["Accuracy"],
              cm_logit_test_set$overall["Accuracy"],
              cm_cloglog_test_set$overall["Accuracy"],
              cm_probit_test_set$overall["Accuracy"],
              cm_knn_test_set$overall["Accuracy"],
              cm_svm_test_set$overall["Accuracy"],
              cm_nnet_test_set$overall["Accuracy"],
              cm_nb_test_set$overall["Accuracy"],
              cm_dt_test_set$overall["Accuracy"],
              cm_rf_test_set$overall["Accuracy"])

Specificity_ = c(cm_log_test_set$byClass["Specificity"],
                 cm_logit_test_set$byClass["Specificity"],
                 cm_cloglog_test_set$byClass["Specificity"],
                 cm_probit_test_set$byClass["Specificity"],
                 cm_knn_test_set$byClass["Specificity"],
                 cm_svm_test_set$byClass["Specificity"],
                 cm_nnet_test_set$byClass["Specificity"],
                 cm_nb_test_set$byClass["Specificity"],
                 cm_dt_test_set$byClass["Specificity"],
                 cm_rf_test_set$byClass["Specificity"])
```

```
Confusion_matrix_table = data.frame(Model_name = Model_names,
                                    Accuracy_overall = Accuracy_,
                                    Specificity_Actual_1_Predicted_0 = Specificity_)
Confusion_matrix_table
```

```
##         Model_name Accuracy_overall Specificity_Actual_1_Predicted_0
## 1          GLM_Log        0.7342349                          0.38500
## 2        GLM_Logit        0.7342349                          0.38500
## 3      GLM_Cloglog        0.7354767                          0.38500
## 4       GLM_Probit        0.7323030                          0.38750
## 5              KNN        0.7445840                          0.29250
## 6              SVM        0.4773010                          0.59750
## 7  Neural_Networks        0.7278874                          0.39500
## 8      Naive Bayes        0.7281634                          0.36125
## 9    Decision Tree        0.1103905                          1.00000
## 10   Random Forest        0.8657375                          0.85750
```

Changing relevant variables to factors for the whole dataset

```
dataset_after_NAs_Factored = dataset_after_NAs
dataset_after_NAs_Factored$loan_status =  factor(
  dataset_after_NAs_Factored$loan_status,
  levels = c(0,1),
  labels = c(0,1)) # 0: loan with no default / 1: loan with default
dataset_after_NAs_Factored$home_ownership = factor(
  dataset_after_NAs_Factored$home_ownership,
  levels = c("RENT","OWN","MORTGAGE","OTHER"),
  labels = c(1:4))
dataset_after_NAs_Factored$grade = factor(
  dataset_after_NAs_Factored$grade,
  levels = c("A","B","C","D","E","F","G"),
  labels = c(1:7))
```

Table data

```
All_set_pred_binary = data.frame(test_set,
                                 PD_RandomForest = y_pred_rf_test_set,
                                 PD_Probit = y_pred_probit_test_set)
```

PDs results for the test set - GLM: Probit

```
CrossTable(All_set_pred_binary$grade,
           All_set_pred_binary$PD_Probit,
           prop.r = TRUE, prop.c = FALSE, prop.t = FALSE, prop.chisq = FALSE)

##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |-------------------------|
##
##
## Total Observations in Table:  7247
##
##
##                          | All_set_pred_binary$PD_Probit
## All_set_pred_binary$grade |         0 |         1 | Row Total |
## --------------------------|-----------|-----------|-----------|
##                         1 |      2446 |         0 |      2446 |
##                           |     1.000 |     0.000 |     0.338 |
## --------------------------|-----------|-----------|-----------|
##                         2 |      2288 |         0 |      2288 |
##                           |     1.000 |     0.000 |     0.316 |
## --------------------------|-----------|-----------|-----------|
##                         3 |       628 |       828 |      1456 |
##                           |     0.431 |     0.569 |     0.201 |
## --------------------------|-----------|-----------|-----------|
##                         4 |        99 |       693 |       792 |
##                           |     0.125 |     0.875 |     0.109 |
## --------------------------|-----------|-----------|-----------|
##                         5 |        23 |       198 |       221 |
##                           |     0.104 |     0.896 |     0.030 |
## --------------------------|-----------|-----------|-----------|
##                         6 |         3 |        37 |        40 |
##                           |     0.075 |     0.925 |     0.006 |
## --------------------------|-----------|-----------|-----------|
##                         7 |         0 |         4 |         4 |
##                           |     0.000 |     1.000 |     0.001 |
## --------------------------|-----------|-----------|-----------|
##              Column Total |      5487 |      1760 |      7247 |
## --------------------------|-----------|-----------|-----------|
##
##
```

PDs results for the test set - Random Forest

```
CrossTable(All_set_pred_binary$grade,
           All_set_pred_binary$PD_RandomForest,
           prop.r = TRUE, prop.c = FALSE, prop.t = FALSE, prop.chisq = FALSE)

##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |-------------------------|
##
##
## Total Observations in Table:  7247
##
##
##                          | All_set_pred_binary$PD_RandomForest
## All_set_pred_binary$grade |         0 |         1 | Row Total |
## --------------------------|-----------|-----------|-----------|
##                         1 |      2286 |       160 |      2446 |
##                           |     0.935 |     0.065 |     0.338 |
## --------------------------|-----------|-----------|-----------|
##                         2 |      1933 |       355 |      2288 |
##                           |     0.845 |     0.155 |     0.316 |
## --------------------------|-----------|-----------|-----------|
##                         3 |      1021 |       435 |      1456 |
##                           |     0.701 |     0.299 |     0.201 |
## --------------------------|-----------|-----------|-----------|
##                         4 |       402 |       390 |       792 |
##                           |     0.508 |     0.492 |     0.109 |
## --------------------------|-----------|-----------|-----------|
##                         5 |        60 |       161 |       221 |
##                           |     0.271 |     0.729 |     0.030 |
## --------------------------|-----------|-----------|-----------|
##                         6 |         0 |        40 |        40 |
##                           |     0.000 |     1.000 |     0.006 |
## --------------------------|-----------|-----------|-----------|
##                         7 |         0 |         4 |         4 |
##                           |     0.000 |     1.000 |     0.001 |
## --------------------------|-----------|-----------|-----------|
##              Column Total |      5702 |      1545 |      7247 |
## --------------------------|-----------|-----------|-----------|
##
##
```

Conclusion

However, the Random Forest model in the analysis above seem to show more accurate results in terms of predicting defaults, it's not necessarily the same objective of the prediction exercise. Without knowing the purpose of the prediction and its ultimate use, it's not reasonably possible to conclude on the ultimate prediction model and therefore the most accurate PDs.

The above analysis can help with loan granting decisions, understanding the health status of a receivable/loan portfolio, quantifying credit losses. Each circumstance and business purpose will drive the relevant prediction model and its respective cut-off threshold. Hence, the purpose of the prediction will dictate its use, and therefore the choice of the model and the cut-off threshold.

For instance, in this project, we focused on predicting PDs for the purposes of quantifying expected credit losses. Therefore, we used lower cut-off thresholds to remain conservative in default predictions.

To conclude, depending on the predictor's risk appetite in market-share strategy as well as the profitability aim, the selection of the prediction model, cut-off thresholds will differ, and so do the PDs.