Loan Data Analysis

Daniil Azarov

2025-06-20

```r
library(dplyr)          # Data manipulation
library(car)            # Model testing
library(tidyr)          # Data manipulation
library(ggplot2)        # Visualization
library(glmnet)         # Ridge and Lasso
library(DescTools)      # Pseudo R^2 (logistic regression)
library(tibble)         # Data manipulation
library(heplots)        # Homogeneity of variance
library(MASS)           # LDA, QDA
library(klaR)           # Model selection for LDA, QDA, Naive Bayes
library(e1071)          # Naive Bayes
library(pROC)           # ROC, AUC
library(randomForest)   # Random Forest
library(gbm)            # Boosting
```

Pre-Preprocessing

Load the data:

```
df <- read.csv("loan_data.csv")
head(df)
```

```
##   person_age person_gender person_education person_income person_emp_exp
## 1         22        female           Master         71948              0
## 2         21        female      High School         12282              0
## 3         25        female      High School         12438              3
## 4         23        female         Bachelor         79753              0
## 5         24          male           Master         66135              1
## 6         21        female      High School         12951              0
##   person_home_ownership loan_amnt loan_intent loan_int_rate loan_percent_income
## 1                  RENT     35000    PERSONAL         16.02                0.49
## 2                   OWN      1000   EDUCATION         11.14                0.08
## 3              MORTGAGE      5500     MEDICAL         12.87                0.44
## 4                  RENT     35000     MEDICAL         15.23                0.44
## 5                  RENT     35000     MEDICAL         14.27                0.53
## 6                   OWN      2500     VENTURE          7.14                0.19
##   cb_person_cred_hist_length credit_score previous_loan_defaults_on_file
## 1                          3          561                             No
## 2                          2          504                            Yes
## 3                          3          635                             No
## 4                          2          675                             No
## 5                          4          586                             No
## 6                          2          532                             No
##   loan_status
## 1           1
## 2           0
## 3           1
## 4           1
## 5           1
## 6           1
```

```
str(df)
```

```
## 'data.frame':    45000 obs. of  14 variables:
##  $ person_age                   : num  22 21 25 23 24 21 26 24 24 21 ...
##  $ person_gender                : chr  "female" "female" "female" "female" ...
##  $ person_education             : chr  "Master" "High School" "High School" "Bachelor" ...
##  $ person_income                : num  71948 12282 12438 79753 66135 ...
##  $ person_emp_exp               : int  0 0 3 0 1 0 1 5 3 0 ...
##  $ person_home_ownership        : chr  "RENT" "OWN" "MORTGAGE" "RENT" ...
##  $ loan_amnt                    : num  35000 1000 5500 35000 35000 2500 35000 35000 35000 1600 ...
##  $ loan_intent                  : chr  "PERSONAL" "EDUCATION" "MEDICAL" "MEDICAL" ...
##  $ loan_int_rate                : num  16 11.1 12.9 15.2 14.3 ...
##  $ loan_percent_income          : num  0.49 0.08 0.44 0.44 0.53 0.19 0.37 0.37 0.35 0.13 ...
##  $ cb_person_cred_hist_length   : num  3 2 3 2 4 2 3 4 2 3 ...
##  $ credit_score                 : int  561 504 635 675 586 532 701 585 544 640 ...
##  $ previous_loan_defaults_on_file: chr  "No" "Yes" "No" "No" ...
##  $ loan_status                  : int  1 0 1 1 1 1 1 1 1 1 ...
```

```
sum(is.na(df))
```

```
## [1] 0
```

There is no non-available data in the data set, however, some variables need to be reformatted. Also, we will change some variables' names in sake of convenience (e.g., using `age` is easier than `person_age`).

```
df$person_gender <- factor(df$person_gender)
df$person_education <- factor(df$person_education)
df$person_home_ownership <- factor(df$person_home_ownership)
df$loan_intent <- factor(df$loan_intent)
df$previous_loan_defaults_on_file <- factor(df$previous_loan_defaults_on_file)
df$loan_status <- factor(df$loan_status,
                         levels = c(0,1))

df <- df %>%
  rename(age = person_age, sex = person_gender, education = person_education,
         income = person_income, employment_experience = person_emp_exp,
         home_ownership = person_home_ownership, loan_amount = loan_amnt,
         credit_history_length = cb_person_cred_hist_length)
head(df)
```

```
##   age    sex   education income employment_experience home_ownership
## 1  22 female      Master  71948                     0           RENT
## 2  21 female High School  12282                     0            OWN
## 3  25 female High School  12438                     3       MORTGAGE
## 4  23 female    Bachelor  79753                     0           RENT
## 5  24   male      Master  66135                     1           RENT
## 6  21 female High School  12951                     0            OWN
##   loan_amount loan_intent loan_int_rate loan_percent_income
## 1       35000    PERSONAL         16.02                0.49
## 2        1000   EDUCATION         11.14                0.08
## 3        5500     MEDICAL         12.87                0.44
## 4       35000     MEDICAL         15.23                0.44
## 5       35000     MEDICAL         14.27                0.53
## 6        2500     VENTURE          7.14                0.19
##   credit_history_length credit_score previous_loan_defaults_on_file loan_status
## 1                     3          561                             No           1
## 2                     2          504                            Yes           0
## 3                     3          635                             No           1
## 4                     2          675                             No           1
## 5                     4          586                             No           1
## 6                     2          532                             No           1
```

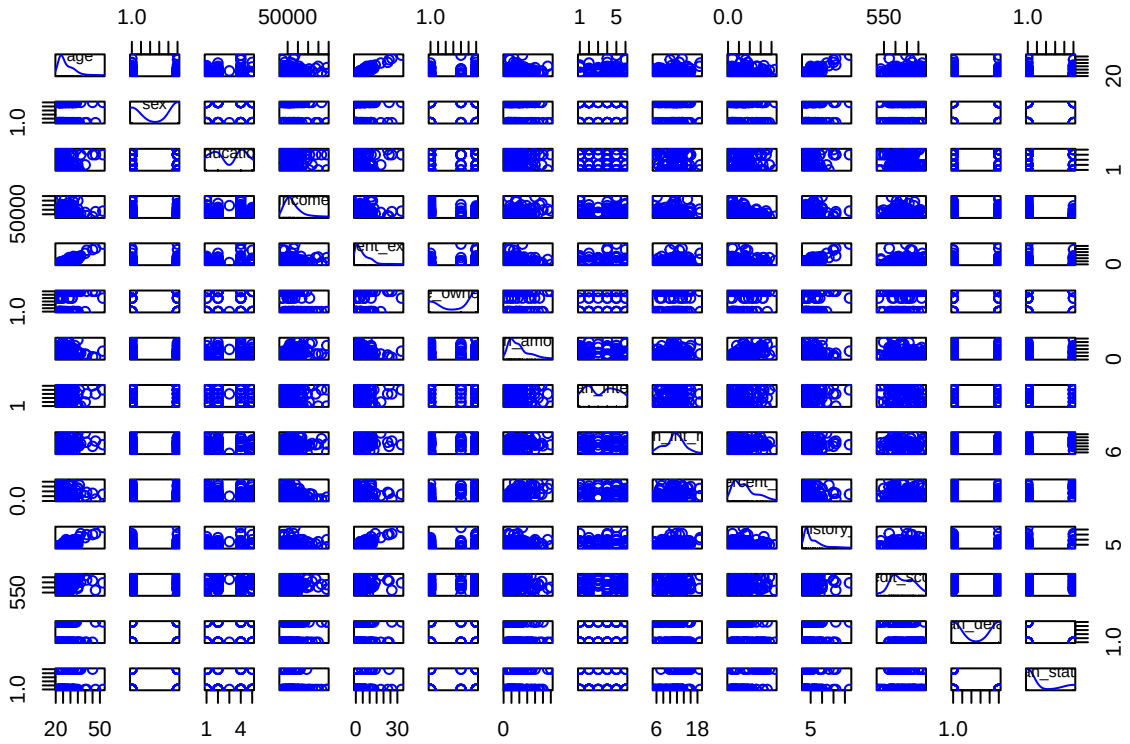
Now, let's specify the training and the test data sets. The training data set is going to be 75% of the full data set, which is $45,000 * 0.75 = 33750$. Therefore, the test data set is going to contain the remaining 25% of the data ($45,000 * 0.25 = 11250$).

```
set.seed(12345)
train_rows <- sample(1:45000,33750,replace = FALSE)
test_rows <- c(1:45000)[-train_rows]

train_df <- df[train_rows,]
rownames(train_df) <- NULL
test_df <- df[test_rows,]
```

Exploratory Data Analysis

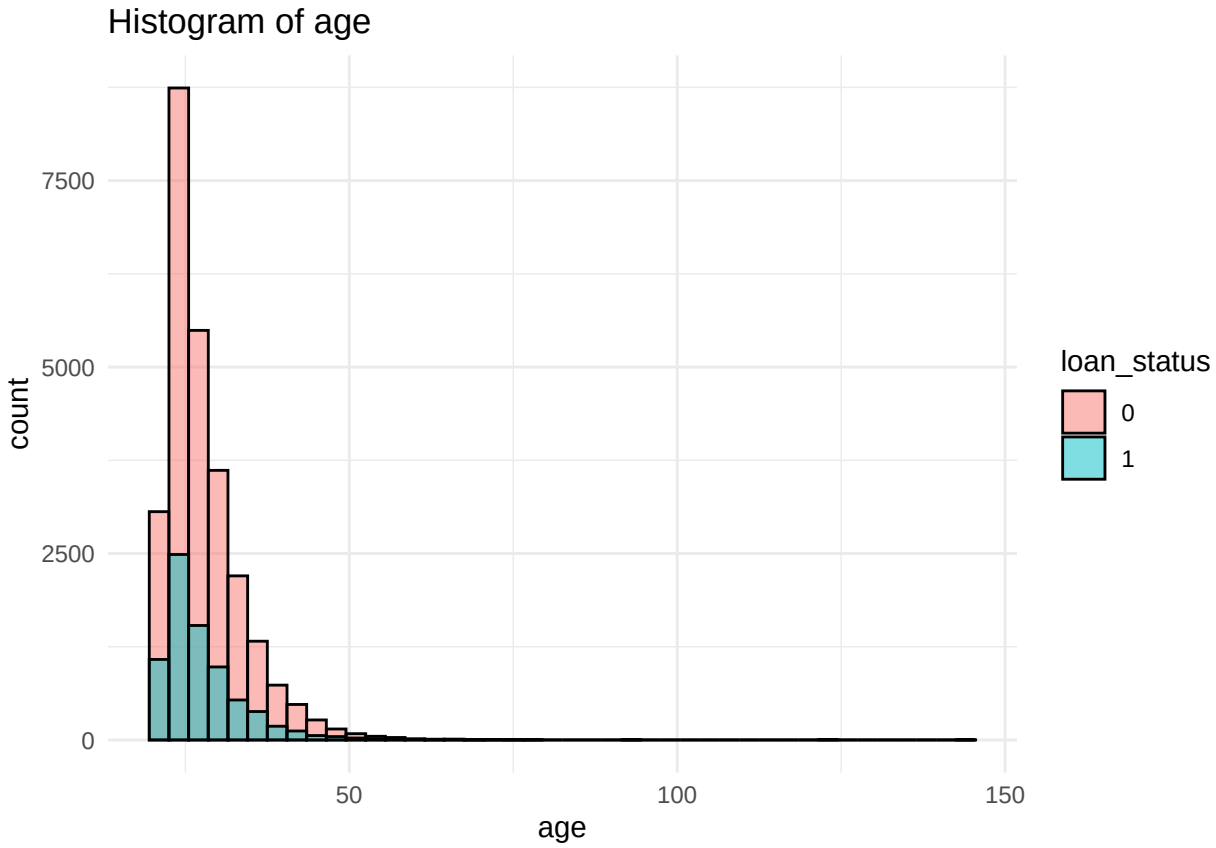
Let's create a plot with 100 random observations. We are using 100 random observations instead of the full data set because it is difficult to see anything when there are 33,750 data points in each square. However, this plot with a small subset of the data will allow us to get a feeling if there are any patterns:

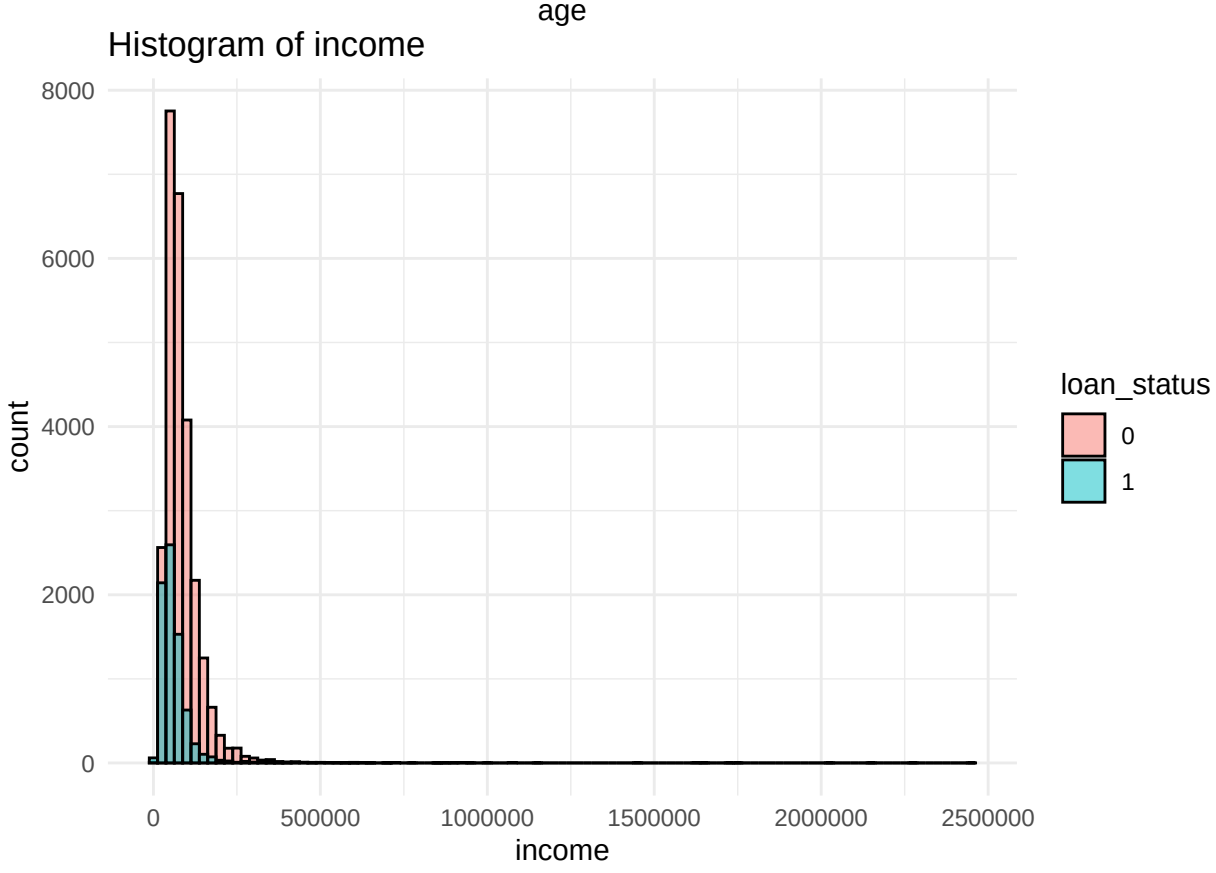
```
set.seed(12)
scatterplotMatrix(train_df[
  sample(1:nrow(train_df), 100),],
                smooth = F, regLine = F, diagonal = T)
```



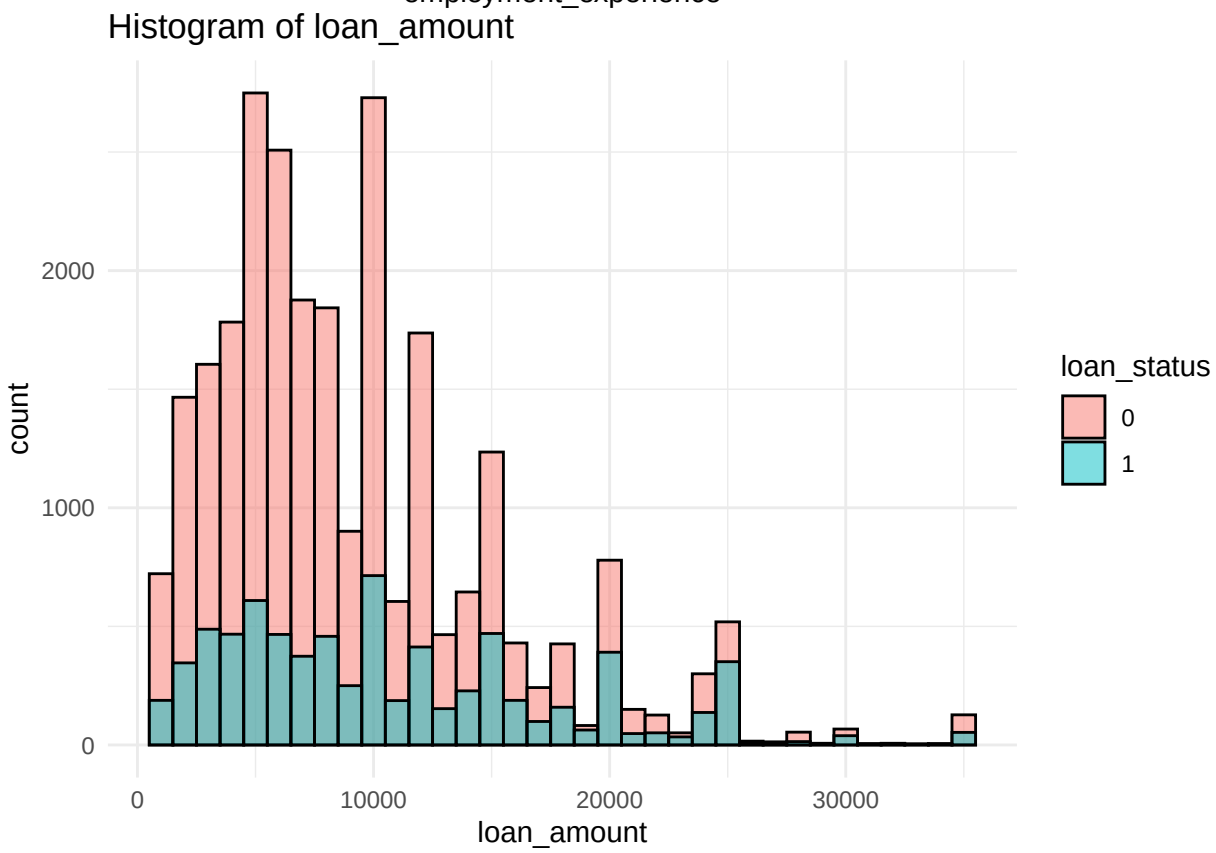
Let's also see histograms of numerical variables:

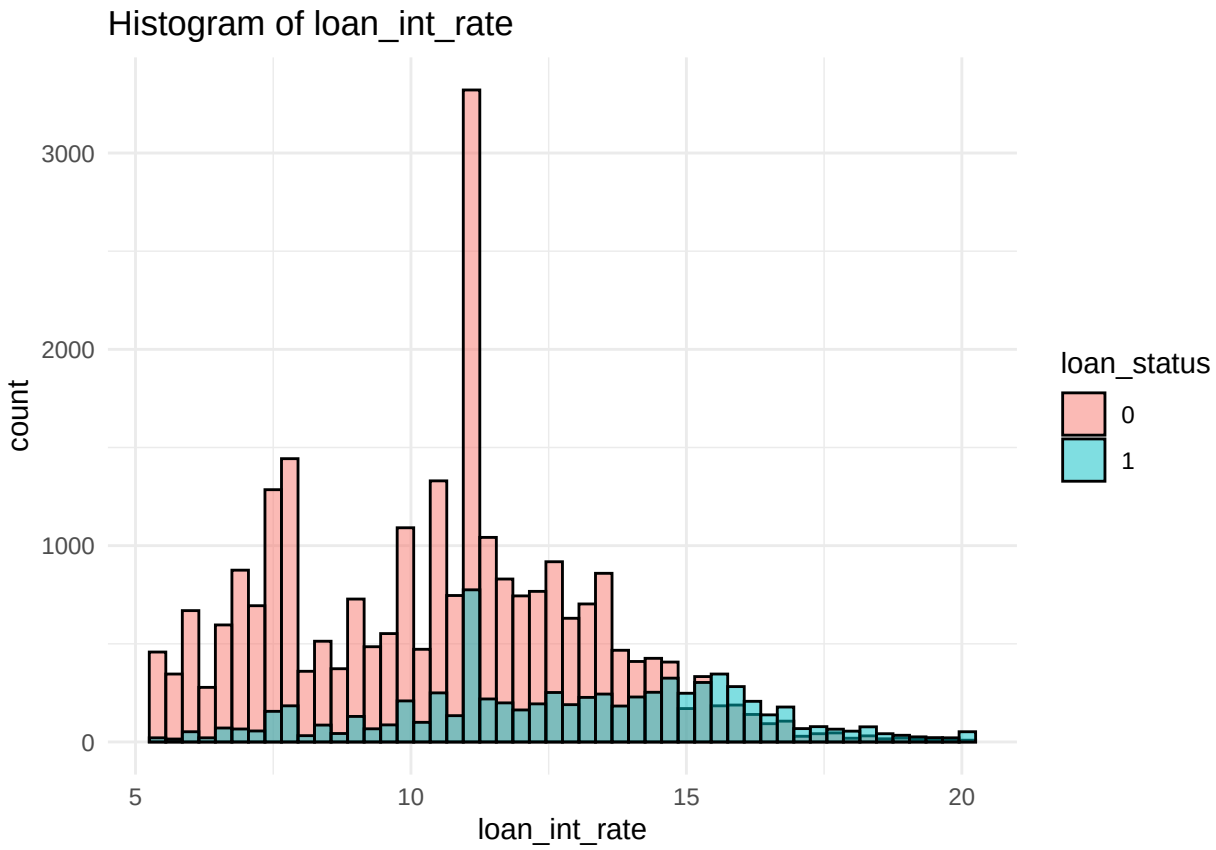
```
numerical_vars <- c("age","income","employment_experience",
                    "loan_amount","loan_int_rate","loan_percent_income",
                    "credit_history_length","credit_score")
binwidth_i <- c(3,25000,2,1000,0.3,0.02,1,10)
i = 1
for (var in numerical_vars) {
  p <- ggplot(train_df, aes(x = .data[[var]],
                            fill = loan_status)) +
    geom_histogram(binwidth = binwidth_i[i], color = "black",
                   alpha = .5, position="identity") +
    labs(title = paste("Histogram of", var), x = var) +
    theme_minimal()
  print(p)
  i = i+1
}
```

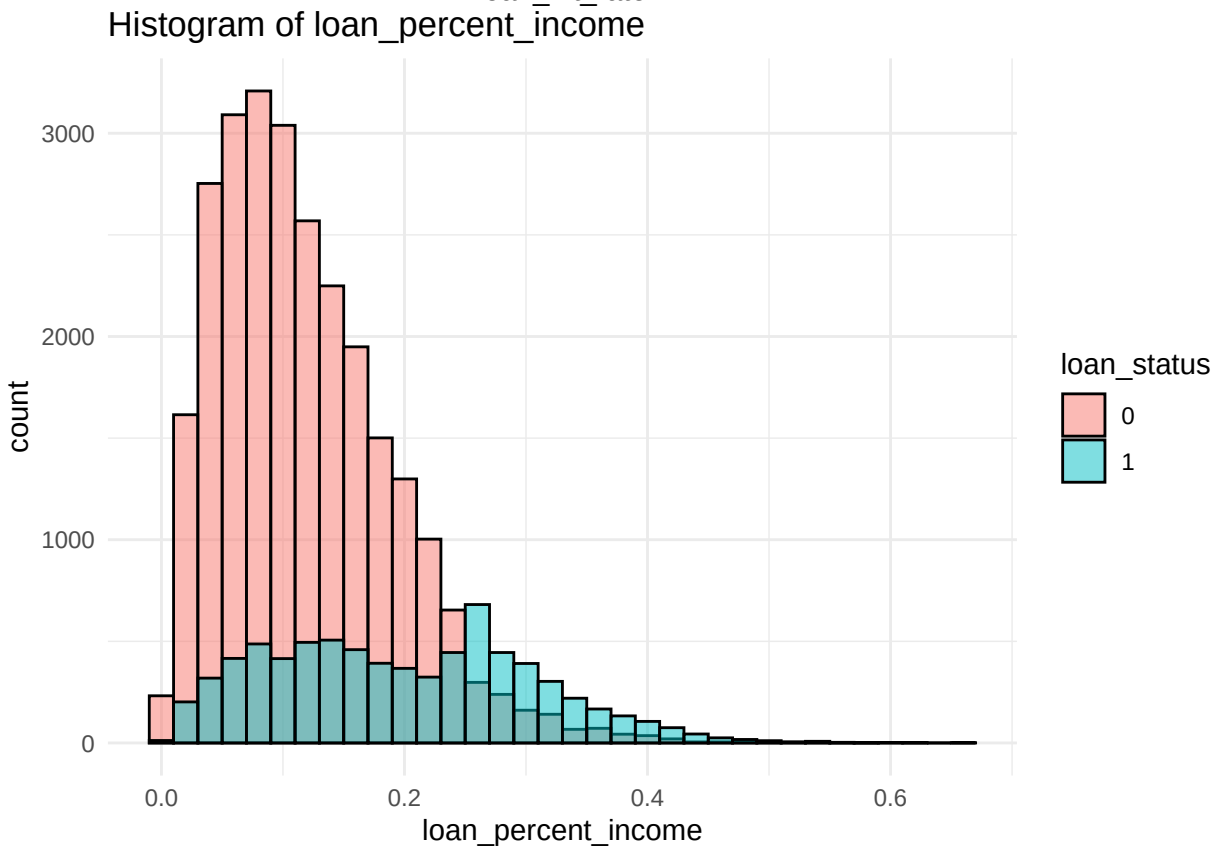

Histogram of age
count
7500
5000
2500
0
50
100
150
age
loan_status
0
1

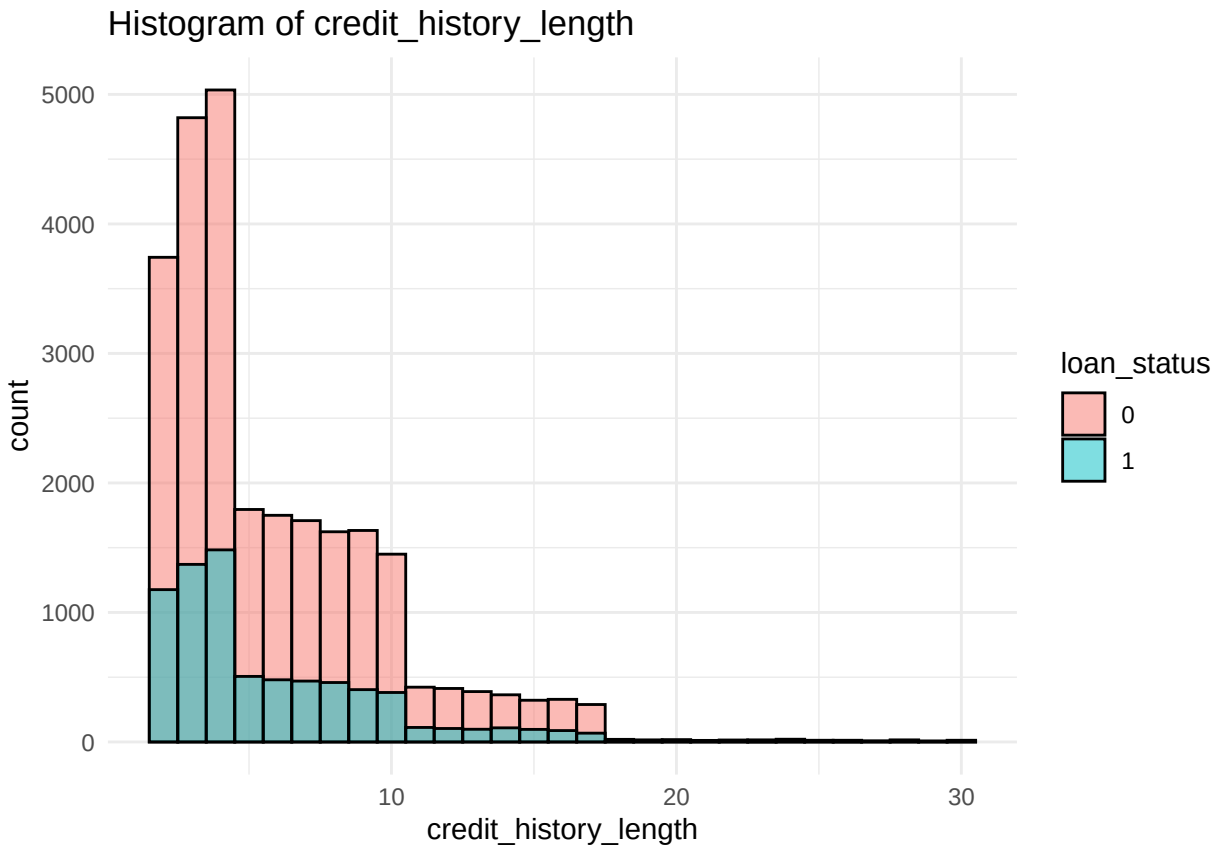

Histogram of income
count
8000
6000
4000
2000
0
0
500000
1000000
1500000
2000000
2500000
income
loan_status
0
1

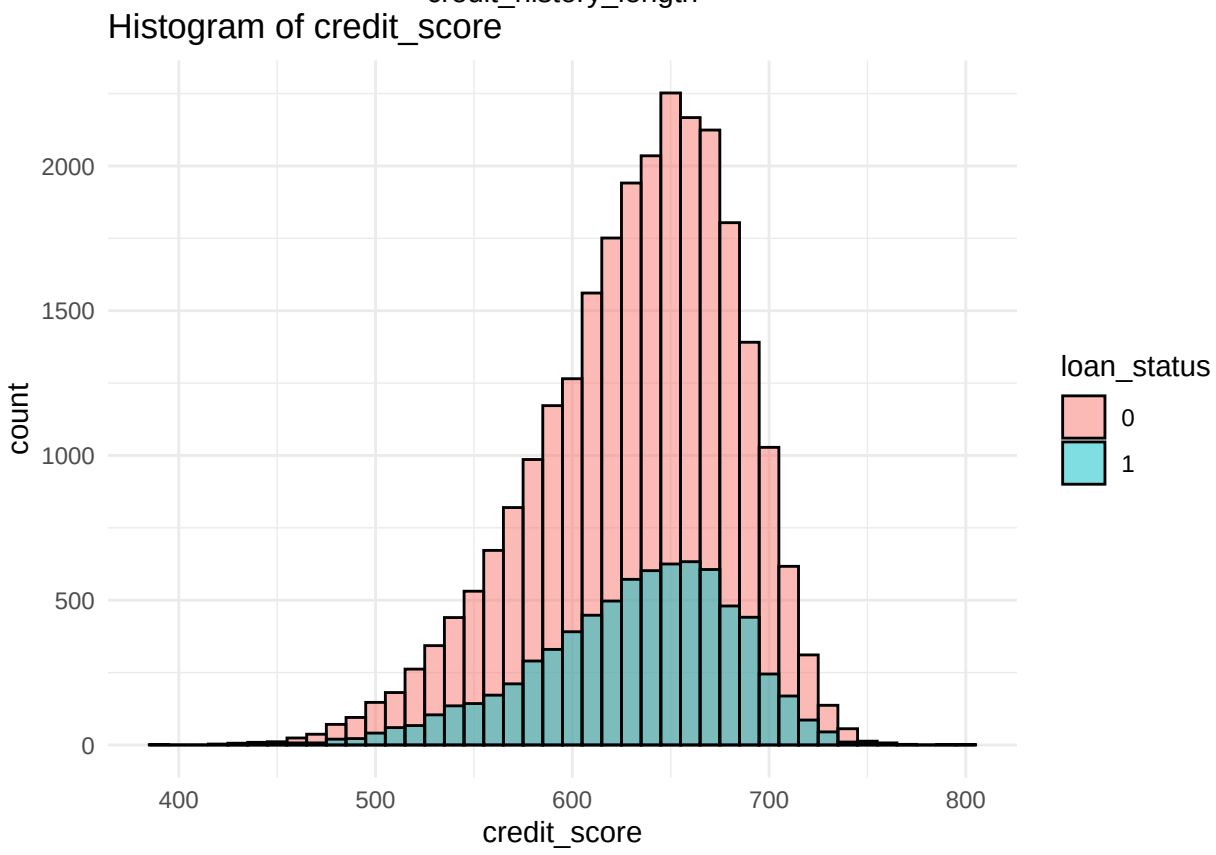

Histogram of employment_experience
8000
6000
4000
2000
0
count
0
40
80
120
employment_experience
loan_status
0
1


Histogram of loan_amount
2000
1000
0
count
0
10000
20000
30000
loan_amount
loan_status
0
1


Histogram of loan_int_rate
count
3000
2000
1000
0
5
10
15
20
loan_int_rate
loan_status
0
1


Histogram of loan_percent_income
count
3000
2000
1000
0
0.0
0.2
0.4
0.6
loan_percent_income
loan_status
0
1





Based on what we can see here, there are no obvious reasons that might stand in the way. However, some

variables are highly correlated with each other (e.g., `age` and `employment_experience`) which can be a problem in the future. For now, we will use all the predictors and rely on model selection and regularization.

We are also going to introduce some **interactions** in the models: `home_ownership:credit_score`, `home_ownership:employment_experience`, and `education:employment_experience`. These variables are correlated based on the scatter plot, and introducing these interactions should also make logical sense (e.g., if a person has very little employment experience but has a PhD or Masters, it may mean that he just didn't have time to work but he/she is actually very responsible and is able to pay off the loan).

Models

Logistic Regression

```
m1_logistic <- glm(loan_status ~ age + sex + education +
                 income + employment_experience +
                 home_ownership +
                 loan_amount + loan_intent + loan_int_rate +
                 loan_percent_income + credit_history_length +
                 credit_score + previous_loan_defaults_on_file +
                 education:employment_experience +
                 (employment_experience +
                 credit_score):home_ownership,
                 data = train_df,
                 family = binomial)
```

Since the data is not grouped (binomial), let's make functions to check for the goodness-of-fit:

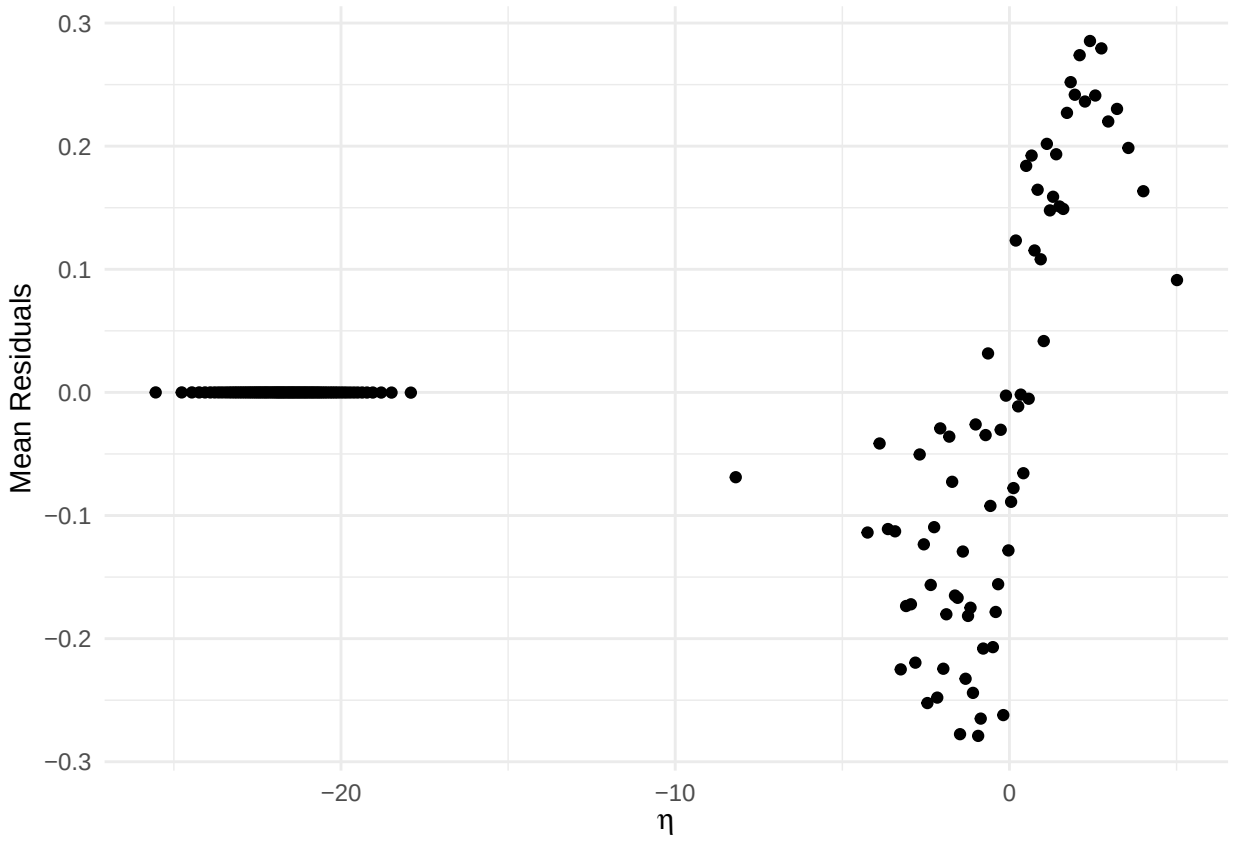
- Binning technique to check for heteroscedasticity
- Hosmer-Lemeshow test

```
# Binning
binning_bernoulli <- function(model,n_bins = 5){
  grouped.df <- model$data %>%
  mutate(residuals=residuals(model), eta=predict(model)) %>%
  dplyr::group_by(ntile(eta, n_bins)) %>%
  summarise(residuals=mean(residuals),
            eta=mean(eta),
            n=n())
  gr <- ggplot(grouped.df, aes(x = eta, y = residuals)) +
    geom_point() +
    labs(x = expression(eta), y = "Mean Residuals") +
    theme_minimal()

  out <- list(df = grouped.df, graph = gr)
  return(out)
}


# Hosmer-Lemeshow test
hosmer_lemeshow <- function(model,n_bins = 5){
  full_df <- model$data %>%
  mutate(residuals=residuals(model),
         p.hat=predict(model,
                       type="response"),
         eta = predict(model),
         n=n())

  hl_df <- full_df %>%
  dplyr::group_by(ntile(eta, n_bins)) %>%
  summarise(y=sum(as.numeric(loan_status)-1),
            ppred=mean(p.hat),
            count=n()) %>%
  mutate(se.fit=sqrt( ppred*(1-ppred)/count ) )
```

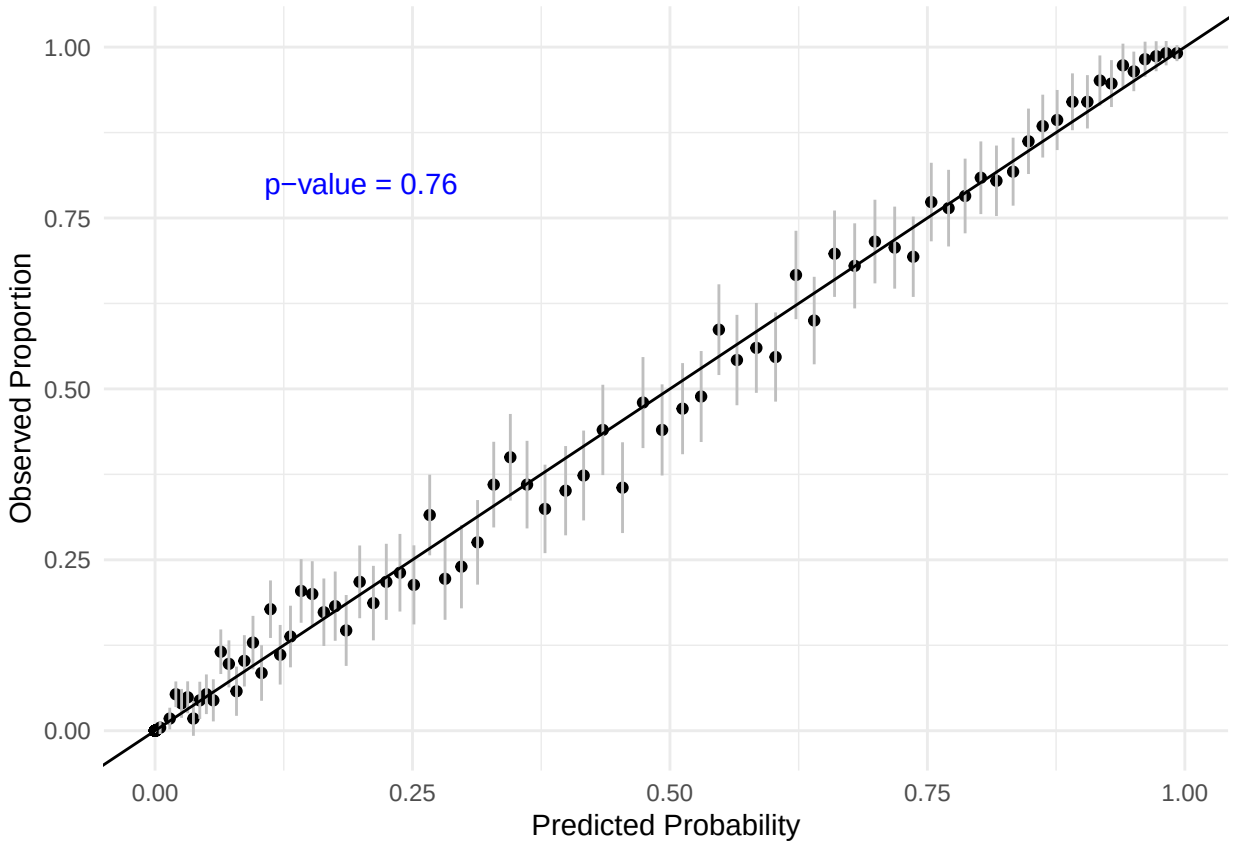
```
  hl_stat <- with(hl_df,
                  sum( (y-count*ppred)^2 / (count*ppred*(1-ppred)) )
                  )
  p.value <- 1-pchisq(hl_stat, nrow(hl_df)-2)

  gr <- ggplot(hl_df,aes(x=ppred,
                   y=y/count,
                   ymin=y/count-2*se.fit,
                   ymax=y/count+2*se.fit))+
    geom_point()+
    geom_linerange(color=grey(0.75))+
    geom_abline(intercept=0,slope=1)+
    xlab("Predicted Probability")+
    ylab("Observed Proportion")+
    theme_minimal()+
    annotate("text", x = 0.2, y = 0.8,
             label = paste0("p-value = ", round(p.value,3)),
             size = 4, color = "blue")


  out <- list(hl_stat = hl_stat,
              deg.freed = nrow(hl_df)-2,
              p.value = p.value,
              min.count.in.bin = min(hl_df$count),
              max.count.in.bin = max(hl_df$count),
              mean.count.in.bin = mean(hl_df$count),
              graph = gr)
  return(out)
}
```

Let's check goodness-of-fit for the initial model:

```
binning_bernoulli(m1_logistic,150)$graph
```




```
hosmer_lemeshow(m1_logistic,150)$graph
```



The model is a good fit (p-value = .76). However, there is heteroscedasticity in the data. This is most likely due to the fact that `previous_loan_defaults_on_file` is almost a perfect predictor:

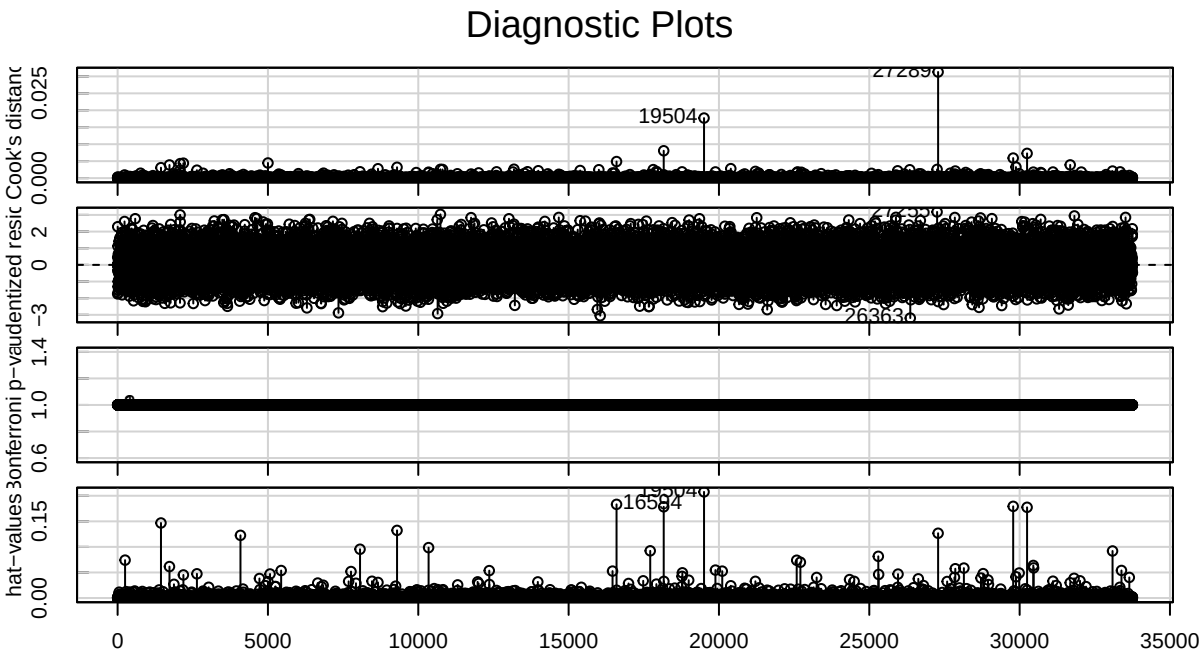
```
with(
  df,
  table(loan_status,previous_loan_defaults_on_file)
)
```

```
##             previous_loan_defaults_on_file
## loan_status    No   Yes
##           0 12142 22858
##           1 10000     0
```

In other words, if a person had previously defaulted, there is no chance that he or she will pay off a loan. It introduces a problem for logistic regression (perfect separation between classes).

However, there are no individual influential observations (largest Cook's distance is below 0.03, no outliers, leverage values are below 0.20):

```
influenceIndexPlot(m1_logistic)
```



We might need to add polynomials to reduce the influence of heteroscedasticity - let's check for this:

```
residualPlots(m1_logistic, type = "rstudent", plot = F)
```

```
##                               Test stat Pr(>|Test stat|)
## age                              0.2207          0.63852
## sex
## education
## income                           2.2161          0.13658
## employment_experience            1.0013          0.31698
## home_ownership
## loan_amount                    318.3916        < 2.2e-16 ***
## loan_intent
## loan_int_rate                  186.7507        < 2.2e-16 ***
## loan_percent_income             90.7922        < 2.2e-16 ***
## credit_history_length            3.2700          0.07056 .
## credit_score                    22.4309        2.178e-06 ***
## previous_loan_defaults_on_file
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

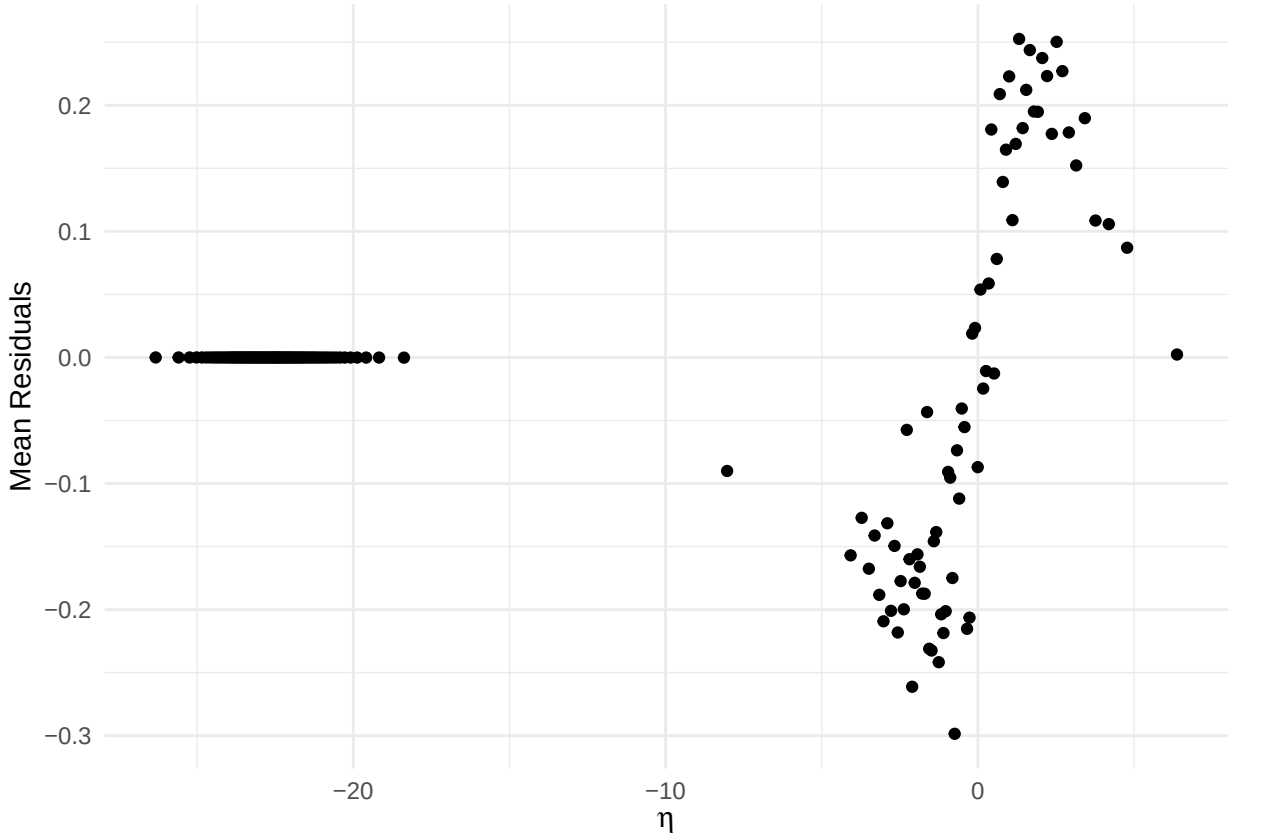
Residual plots indicate that we need to add some polynomials:

```
m2_logistic <- glm(loan_status ~ age + sex + education +
                 income + employment_experience +
                 home_ownership +
                 poly(loan_amount,2) + loan_intent + poly(loan_int_rate,2) +
                 poly(loan_percent_income,2) + credit_history_length +
                 poly(credit_score,2) + previous_loan_defaults_on_file +
                 education:employment_experience +
                 (employment_experience +
                 credit_score):home_ownership,
```

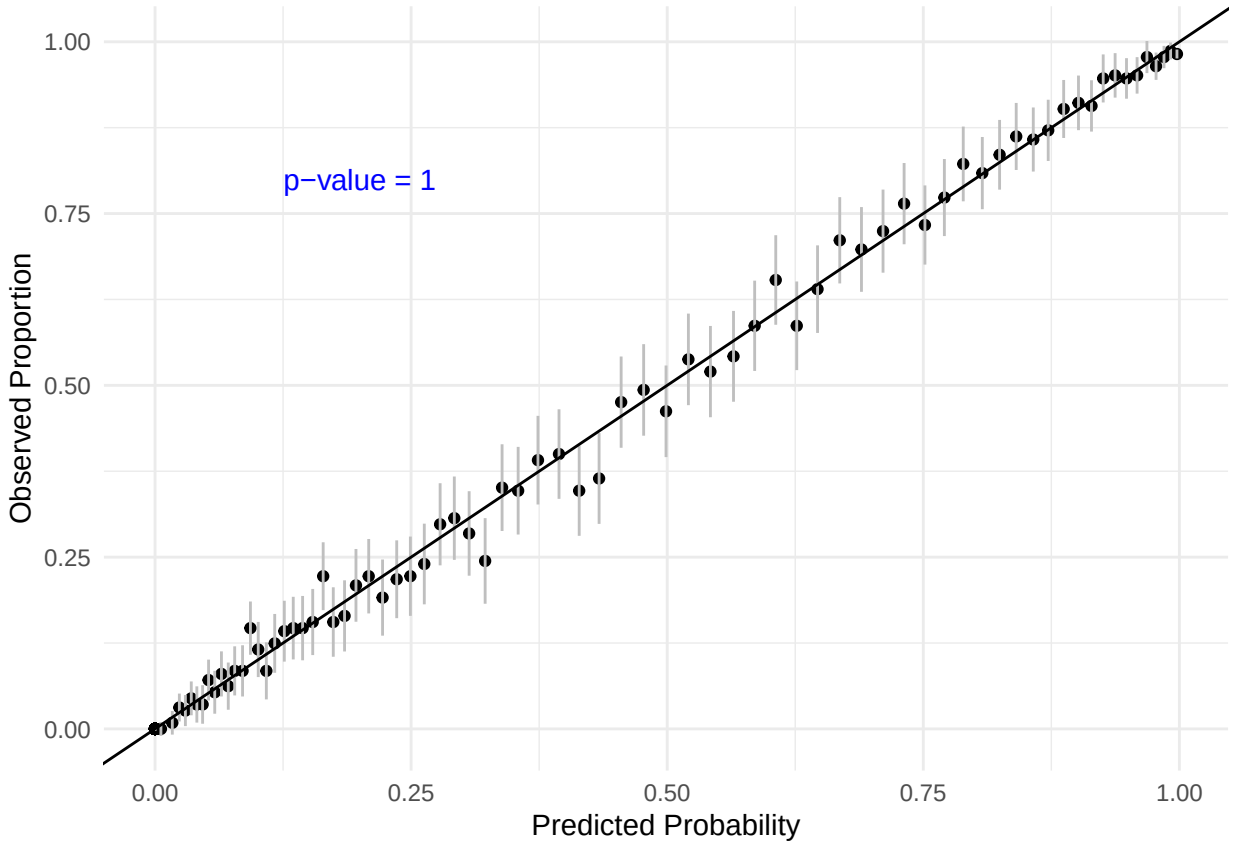
```
                data = train_df,
                family = binomial)
```

Let's now check the goodness-of-fit:

```
binning_bernoulli(m2_logistic,150)$graph
```




```
hosmer_lemeshow(m2_logistic,150)$graph
```



P-value is large, the predicted probabilities (for bins) are well aligned with the observed values. So, we are going to use this model. Note that heteroscedasticity is not a major problem since we are not performing inference here, rather just trying to come up with a model that provides the most accurate results.

Model selection

Now, let's use model selection techniques to come up with the best model:

```
m0_logistic <- glm(loan_status ~ 1,
              family = binomial,train_df)
m1_logistic_aic1 <- step(m0_logistic,
                  scope=list(lower=m0_logistic, upper=m2_logistic),
                  direction = "both", trace = F)
m1_logistic_aic2 <- step(m2_logistic,
                  scope=list(lower=m0_logistic, upper=m2_logistic),
                  direction = "both", trace = F)
```

```
m1_logistic_aic1$aic
```

```
## [1] 14289.2
```

```
m1_logistic_aic2$aic
```

```
## [1] 14289.2
```

```
m1_logistic_aic1$formula
```

```
## loan_status ~ previous_loan_defaults_on_file + poly(loan_percent_income,
##     2) + poly(loan_int_rate, 2) + poly(loan_amount, 2) + home_ownership +
##     loan_intent + poly(credit_score, 2) + income + employment_experience +
```

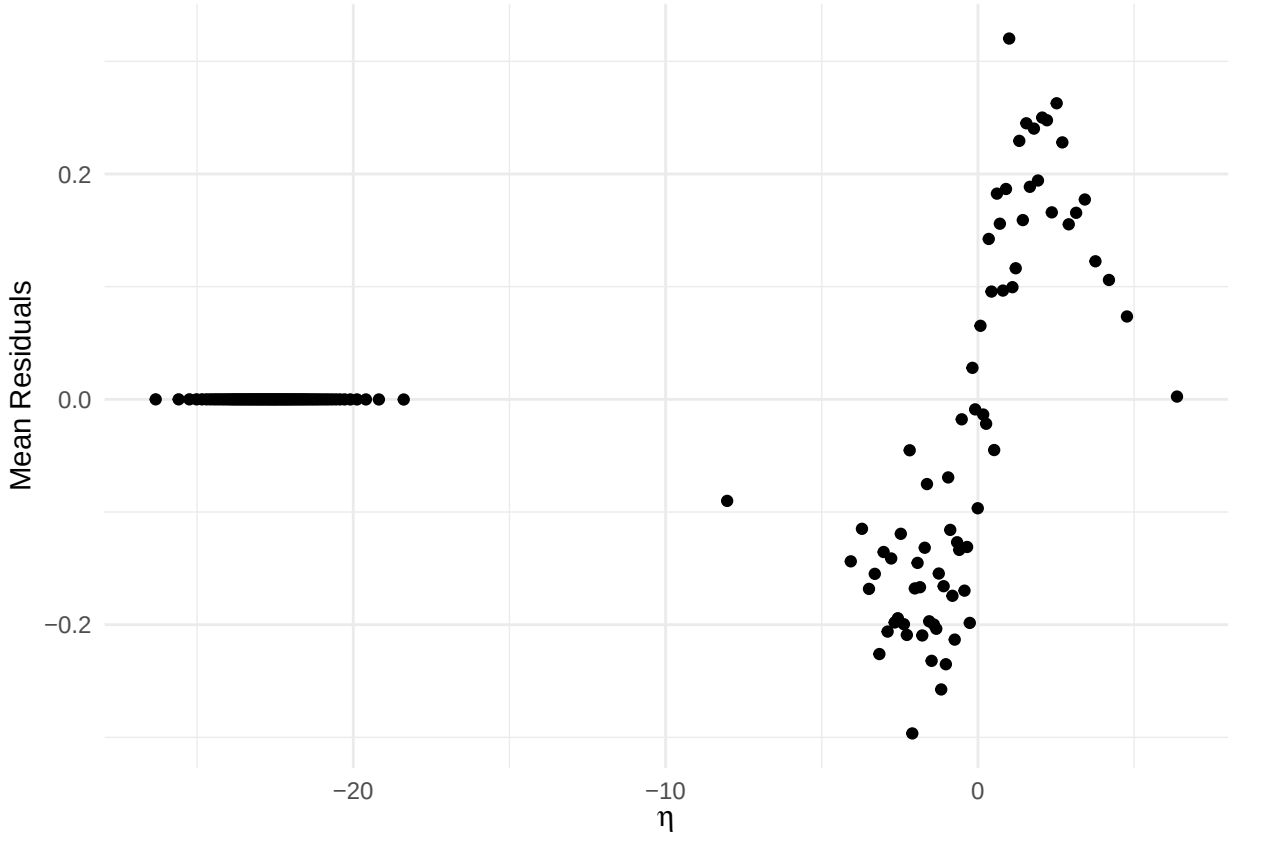
```
##     age + home_ownership:credit_score + home_ownership:employment_experience
```

```
m1_logistic_aic2$formula
```

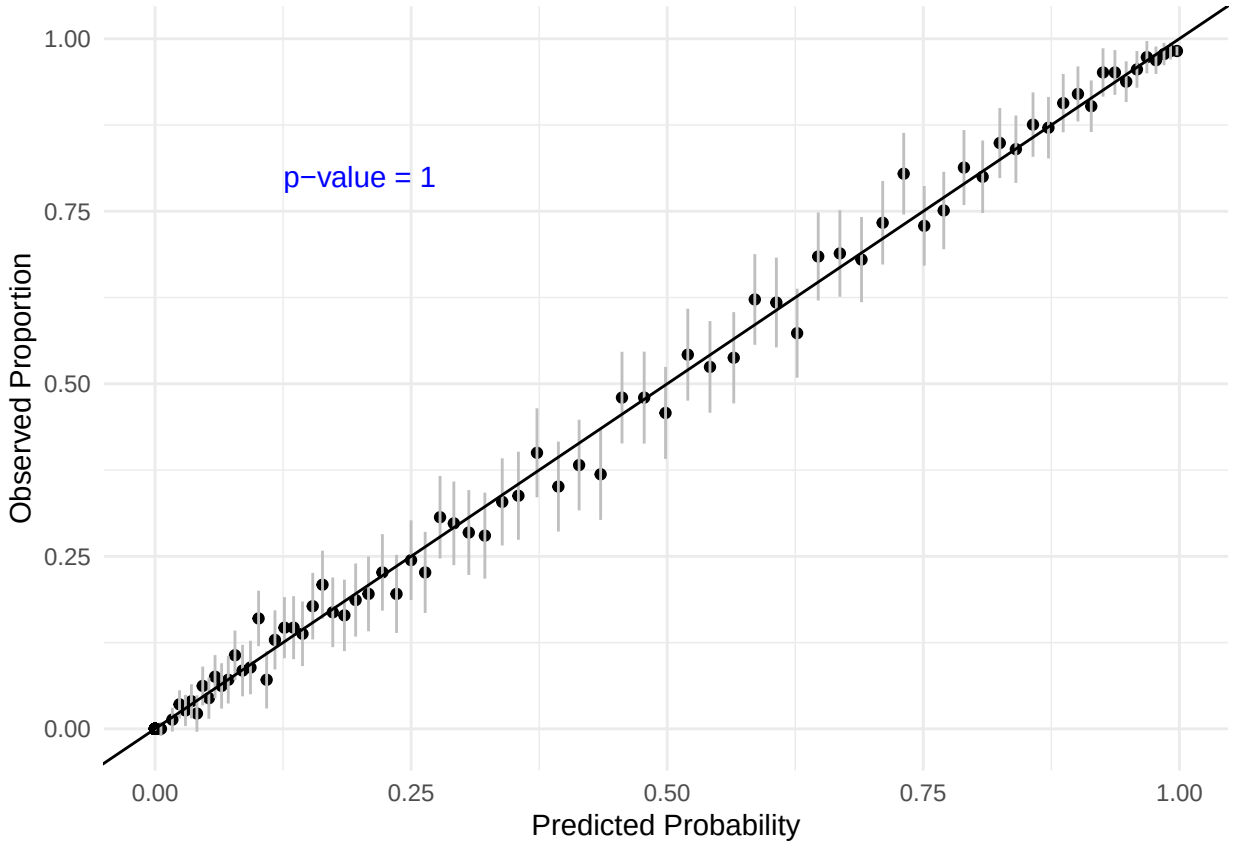
```
## loan_status ~ age + income + employment_experience + home_ownership +
##     poly(loan_amount, 2) + loan_intent + poly(loan_int_rate,
##     2) + poly(loan_percent_income, 2) + poly(credit_score, 2) +
##     previous_loan_defaults_on_file + employment_experience:home_ownership +
##     home_ownership:credit_score
```

Despite the starting point, we ended up with the same model. Let's check its goodness-of-fit:

```
binning_bernoulli(m1_logistic_aic1,150)$graph
```




```
hosmer_lemeshow(m1_logistic_aic1,150)$graph
```



Heteroscedasticity is still there, however, p-value is large.

Let's have a look at pseudo R^2 and successful prediction rate (for the training data set):

```
# Pseudo R^2
PseudoR2(m1_logistic_aic1, which = "Nagelkerke")
```

```
## Nagelkerke
##  0.7208142
```

```
# Contingency table
# (for now use a basic threshold of 0.5)
preds.logistic <- as.numeric(m1_logistic_aic1$fitted.values >= .5)
cont.table.logistic <- table(train_df$loan_status, preds.logistic)
(sum(diag(cont.table.logistic))) / sum(cont.table.logistic)
```

```
## [1] 0.9040593
```

Pseudo $R^2 = .72$, which is a decent number. If we use a basic threshold level (0.5), then the model shows a very good performance rate - 90.4% (note, that it is the performance for the data that the model was trained on).

This is the first model that we are going to work with and compare with other models (`m1_logistic_aic`).

Regularization

Now, let's use **Ridge and Lasso** regularization to come up with potentially better solutions. For both Ridge and Lasso, we'll start with the full model that was built before (with polynomials).

```
x_train <- model.matrix(m2_logistic)[,-1] # no intercept
y_train <- train_df$loan_status
```

```
n_folds <- 10 # in this case each fold contains 33,750 / 10 = 3,375 observations

set.seed(123)

# Ridge
# Use cross-validation to find the best lambda (tuning parameter)
cv.m1_ridge <- cv.glmnet(x_train, y_train,
                         alpha = 0, # 0 - Ridge, 1 - Lasso
                         nfolds = n_folds,
                         family = binomial)
m1_ridge <- glmnet(x_train, y_train,
                   alpha = 0, # 0 - Ridge, 1 - Lasso
                   lambda = cv.m1_ridge$lambda.min,
                   family = binomial)

# Lasso
cv.m1_lasso <- cv.glmnet(x_train, y_train,
                         alpha = 1, # 0 - Ridge, 1 - Lasso
                         nfolds = n_folds,
                         family = binomial)
m1_lasso <- glmnet(x_train, y_train,
                   alpha = 1, # 0 - Ridge, 1 - Lasso
                   lambda = cv.m1_lasso$lambda.min,
                   family = binomial)
```

These are the λ's that we are using:

```
cv.m1_ridge$lambda.min # for Ridge
```

```
## [1] 0.02252542
```

```
cv.m1_lasso$lambda.min # for Lasso
```

```
## [1] 6.878937e-05
```

Let's compare the regressors' coefficients for the three models (this is just to have look):

```
m <- cbind(
  round(coef(m1_ridge),3),
  round(ifelse(coef(m1_lasso)==0,NA,coef(m1_lasso)),3) # put NAs for 0s
  )
colnames(m) <- c("ridge","lasso")
m <- m %>%
  as.matrix(.) %>%
  as.data.frame(.)
m <- rownames_to_column(m, "predictor")
m1 <- as.data.frame(round(coef(m1_logistic_aic1),3))
colnames(m1) <- "aic_step"
m1 <- rownames_to_column(m1, "predictor")
full_join(m1,m,by = "predictor")
```

```
##                                   predictor aic_step   ridge    lasso
## 1                               (Intercept)    0.373  -0.401   -0.621
## 2          previous_loan_defaults_on_fileYes  -21.334  -2.826   -9.558
## 3              poly(loan_percent_income, 2)1  311.143 137.430  308.125
## 4              poly(loan_percent_income, 2)2   15.443  21.901   13.629
```

```
## 5                     poly(loan_int_rate, 2)1  176.694 114.548  176.022
## 6                     poly(loan_int_rate, 2)2   57.648  39.299   57.124
## 7                      poly(loan_amount, 2)1 -160.451 -26.958 -158.693
## 8                      poly(loan_amount, 2)2   69.316  23.811   68.905
## 9                        home_ownershipOTHER   -5.319   0.245       NA
## 10                         home_ownershipOWN   -1.674  -0.505   -1.374
## 11                       home_ownershipRENT   -0.910   0.314       NA
## 12                     loan_intentEDUCATION   -0.871  -0.430   -0.864
## 13                loan_intentHOMEIMPROVEMENT    0.031   0.160    0.028
## 14                       loan_intentMEDICAL   -0.270   0.017   -0.264
## 15                     loan_intentPERSONAL   -0.698  -0.302   -0.690
## 16                      loan_intentVENTURE   -1.262  -0.644   -1.250
## 17                    poly(credit_score, 2)1  -78.617 -38.474  -84.738
## 18                    poly(credit_score, 2)2   21.808   0.925   20.932
## 19                                    income    0.000   0.000    0.000
## 20                    employment_experience   -0.018  -0.004   -0.018
## 21                                       age    0.027   0.000    0.024
## 22       home_ownershipMORTGAGE:credit_score   -0.003   0.000   -0.001
## 23          home_ownershipOTHER:credit_score    0.007   0.000    0.000
## 24            home_ownershipOWN:credit_score   -0.003  -0.001   -0.001
## 25           home_ownershipRENT:credit_score       NA   0.000    0.000
## 26  home_ownershipOTHER:employment_experience   -0.022      NA       NA
## 27    home_ownershipOWN:employment_experience   -0.009      NA       NA
## 28   home_ownershipRENT:employment_experience   -0.020      NA       NA
## 29                                   sexmale       NA   0.006    0.011
## 30                       educationBachelor       NA  -0.003   -0.031
## 31                      educationDoctorate       NA  -0.009   -0.119
## 32                    educationHigh School       NA  -0.002   -0.042
## 33                        educationMaster       NA  -0.014   -0.023
## 34                  credit_history_length       NA   0.004       NA
## 35    educationBachelor:employment_experience       NA  -0.004   -0.003
## 36   educationDoctorate:employment_experience       NA   0.003    0.007
## 37 educationHigh School:employment_experience       NA   0.002    0.007
## 38      educationMaster:employment_experience       NA   0.002    0.011
## 39  employment_experience:home_ownershipOTHER       NA   0.021    0.001
## 40    employment_experience:home_ownershipOWN       NA  -0.016   -0.009
## 41   employment_experience:home_ownershipRENT       NA  -0.004   -0.017
```

Obviously, all the models produced very different coefficients. For example, for `previous_loan_defaults_on_fileYes` AIC based model's coefficient is -21.3, while Ridge model's coefficient is -2.8, and Lasso's is -9.6. Another extreme example is for `home_ownershipRENT`: AIC based model suggest a negative coefficient (-0.91), Ridge regression suggests a positive coefficient (0.31), and Lasso regression suggests to exclude this parameter.

Let's also have a look at the prediction rate for the training dataset:

```
preds_ridge <- as.numeric(predict(m1_ridge,
        s = m1_ridge$lambda,
        newx = x_train, type = "response") >= 0.5)
preds_lasso <- as.numeric(predict(m1_lasso,
        s = m1_lasso$lambda,
        newx = x_train, type = "response") >= 0.5)

contig_ridge <- xtabs( ~ preds_ridge + train_df$loan_status)
contig_lasso <- xtabs( ~ preds_lasso + train_df$loan_status)
```

```r
sum(diag(contig_ridge))/ nrow(train_df)
```

```
## [1] 0.9004148
```

```r
sum(diag(contig_lasso))/ nrow(train_df)
```

```
## [1] 0.904237
```

The models show similar prediction rates, though Lasso is slightly better. Both Ridge and Lasso are slightly worse than a simple logistic regression (AIC-based).

Summary

Later, we are going to use the following models:

- `m1_logistic_aic1` (AIC-based)
- `m1_ridge` (Ridge)
- `m1_lasso` (Lasso)

LDA, QDA, NB assumptions

Note these assumptions:

1. Predictors are normally distributed within each level of the grouping variable.
2. Independence of observations.
3. Homogeneity of variance/covariance (only LDA).
4. Feature independence given the class (only Naive Bayes)

Because of the first one, it is impossible to use factors and polynomials for LDA and QDA. The third assumption is not held here:

```
boxM <- boxM( cbind(age, sex, education, income, employment_experience,
                home_ownership, loan_amount,loan_intent, loan_int_rate,
                loan_percent_income, credit_history_length,
                credit_score,previous_loan_defaults_on_file) ~ loan_status,
              train_df)
boxM
```

```
##
##  Box's M-test for Homogeneity of Covariance Matrices
##
## data:  Y
## Chi-Sq (approx.) = Inf, df = 91, p-value < 2.2e-16
```

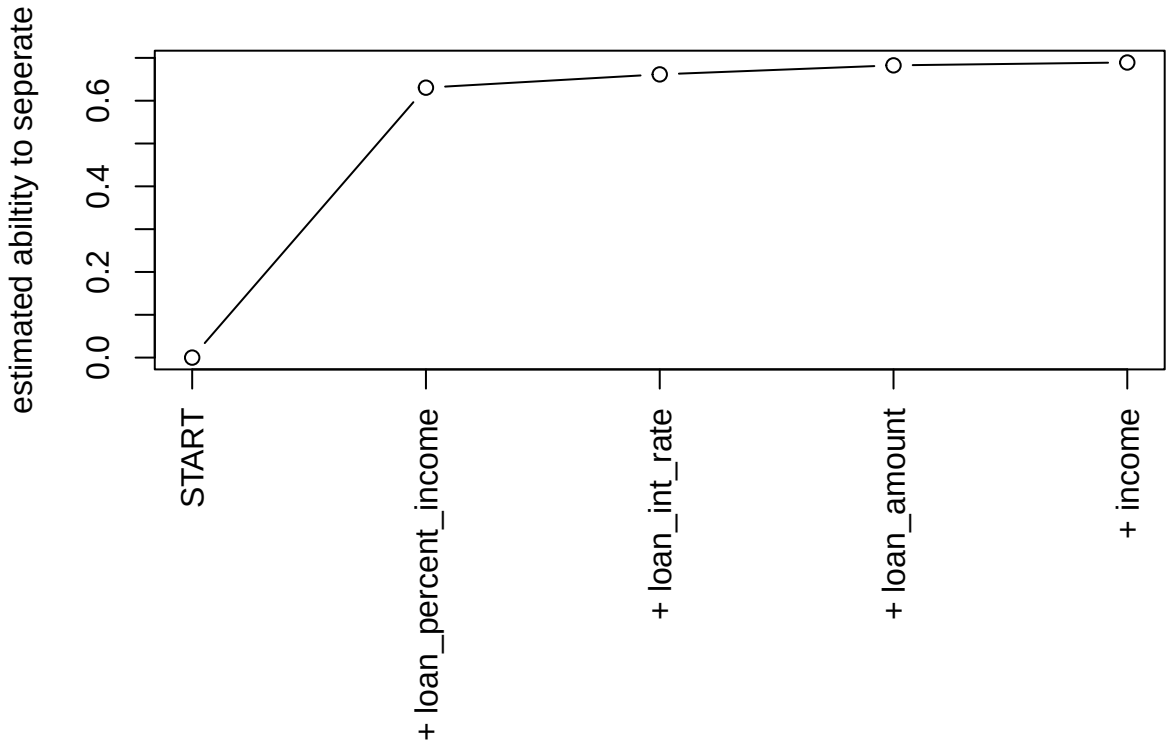
Finally, the last assumption is definitely not met here (e.g.,`loan_percent_income / loan_amount = income`).

Linear Discriminant Analysis

Model selection:

```
set.seed(12345)

lda_m_select <- stepclass(

  loan_status ~ age + sex + education + income +
                employment_experience + home_ownership +
                loan_amount + loan_intent + loan_int_rate +
                loan_percent_income + credit_history_length +
                credit_score + previous_loan_defaults_on_file,
  data = train_df,
  method = "lda",
  direction = "both",
  criterion = "AS", # ability to separate (aka ability to classify correctly)
  improvement = .0001, # min improvement
  cv.groups = rep(sample(1:10),3375) # 10 folds

  )
```

```
plot(lda_m_select)
```




```
m_lda_best <- lda(loan_status ~ loan_percent_income +
                    loan_int_rate + loan_amount + income,
                  train_df)
```

Let's have a look at the prediction rate for the training data set (again, using a basic threshold, 0.5):

```
preds_lda <- xtabs( ~ predict(m_lda_best)$class + train_df$loan_status)
sum(diag(preds_lda))/ nrow(train_df)
```

```
## [1] 0.8286815
```

The prediction rate is decent, though it's worse than for all the three logistic models.

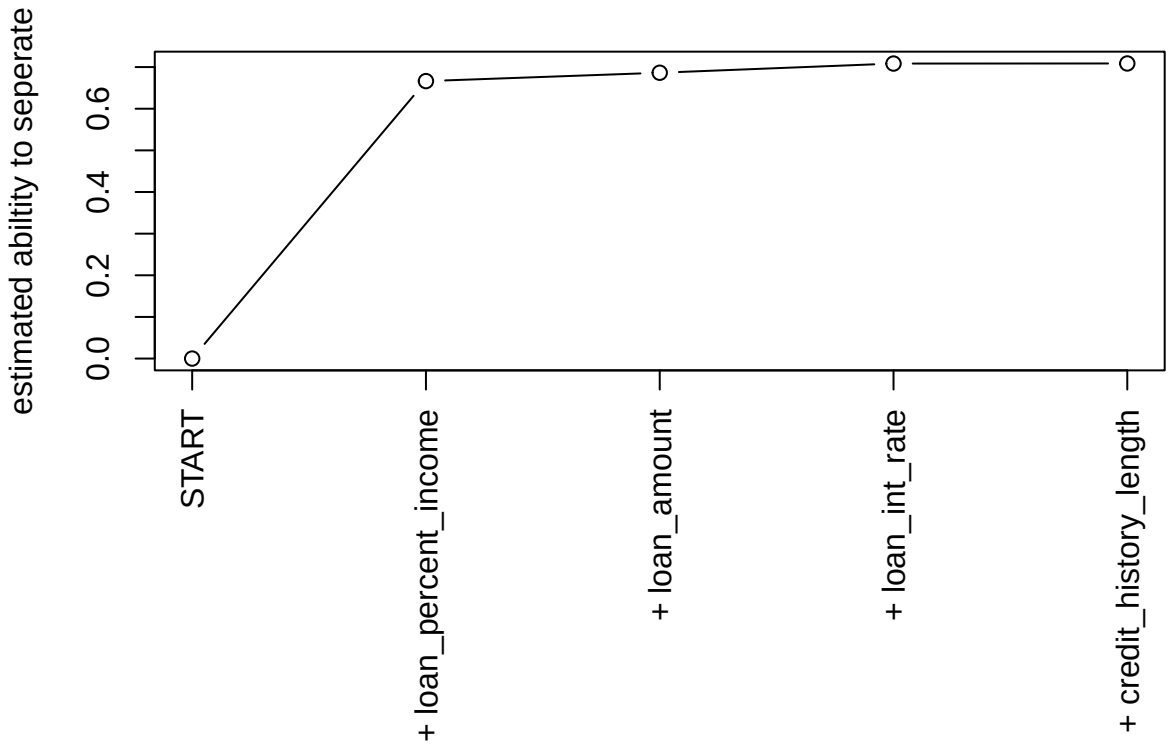
Quadratic Discriminant Analysis

Note that we are not going to use `previous_loan_defaults_on_file` here because of rank deficiency.

Model selection:

```
set.seed(12345)

qda_m_select <- stepclass(

  loan_status ~ age + sex + education + income +
                employment_experience + home_ownership +
                loan_amount + loan_intent + loan_int_rate +
                loan_percent_income + credit_history_length +
                credit_score,
  data = train_df,
  method = "qda",
  direction = "both",
  criterion = "AS", # ability to separate (aka ability to classify correctly)
  improvement = .0001, # min improvement
  cv.groups = rep(sample(1:10),3375) # 10 folds

  )
```

```
plot(qda_m_select)
```




```
m_qda_best <- qda(loan_status ~ loan_percent_income +
                    loan_amount + loan_int_rate +
                    credit_history_length,
                  train_df)
```

Let's have a look at the prediction rate for the training data set (again, using a basic threshold, 0.5):

```
preds_qda <- xtabs( ~ predict(m_qda_best)$class + train_df$loan_status)
sum(diag(preds_qda))/ nrow(train_df)
```

```
## [1] 0.835763
```

The prediction rate for the training data set is slightly better here than for LDA. Though this is expected because of heterogeneity of variance.

Naive Bayes

`naiveBayes()` can't predict new values when using `poly()`, so we are going to use `I()` and create new columns in the data sets. So, for example, instead of using `poly(loan_percent_income,2)` in the model, we are going to use `loan_percent_income + I(loan_percent_income^2) = loan_percent_income + loan_percent_income_sq`.

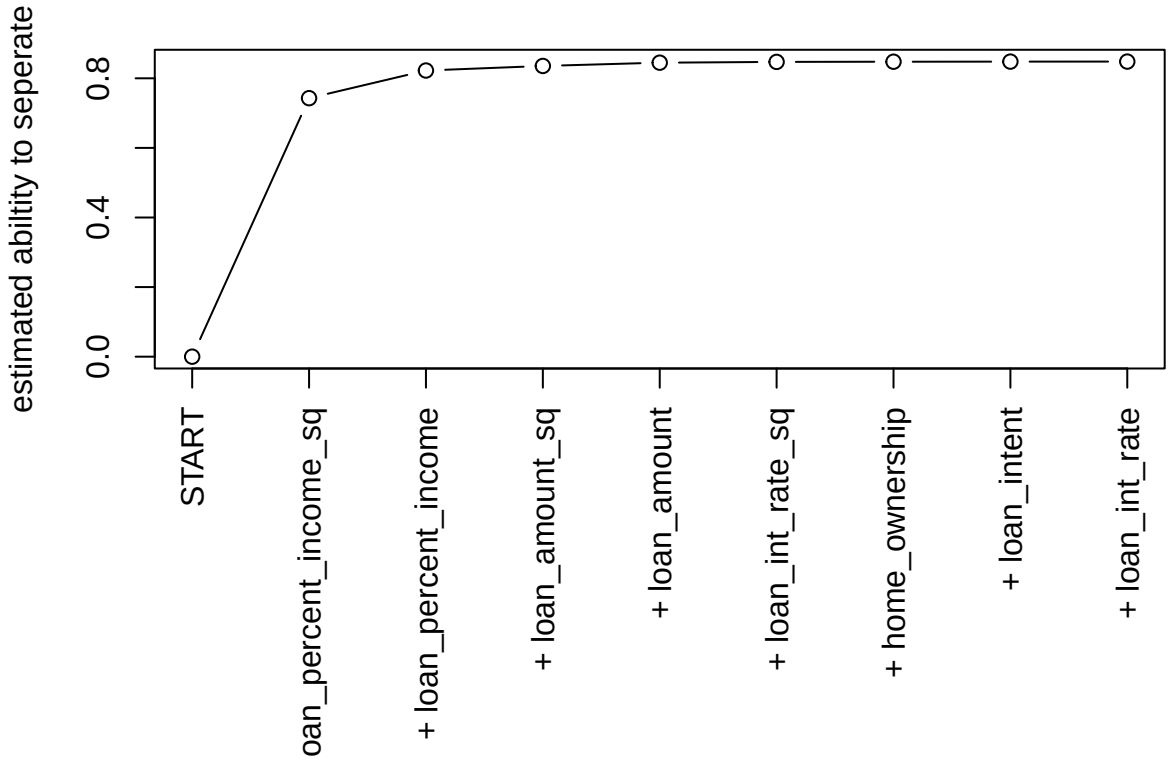
Also, please note that we are going to use `klaR::NaiveBayes()` (starts with capital "N") for model selection and `e1071::naiveBayes()` (starts with lowercase "n") for actual prediction because of how they work with model selection. But the mechanism under the hood is absolutely the same.

```
# training data set
train_df$loan_percent_income_sq <- train_df$loan_percent_income^2
train_df$loan_int_rate_sq <- train_df$loan_int_rate^2
train_df$loan_amount_sq <- train_df$loan_amount^2
train_df$credit_score_sq <- train_df$credit_score^2

# test data set (we'll use it later)
test_df$loan_percent_income_sq <- test_df$loan_percent_income^2
test_df$loan_int_rate_sq <- test_df$loan_int_rate^2
test_df$loan_amount_sq <- test_df$loan_amount^2
test_df$credit_score_sq <- test_df$credit_score^2
```

Model selection:

```
set.seed(12345)

nb_m_select <- stepclass(

  loan_status ~ loan_percent_income +
                               loan_percent_income_sq +
                               loan_int_rate + loan_int_rate_sq +
                               loan_amount + loan_amount_sq +
                               home_ownership + loan_intent +
                               credit_score + credit_score_sq +
                               income + employment_experience + age,
  data = train_df,
  method = "NaiveBayes",
  direction = "both",
  criterion = "AS", # ability to separate (aka ability to classify correctly)
  improvement = .0001, # min improvement
  cv.groups = rep(sample(1:10),3375) # 10 folds

  )
```

```
plot(nb_m_select)
```




```
m_nb_best <- naiveBayes(loan_status ~ loan_percent_income +
                          loan_percent_income_sq +
                          loan_amount + loan_amount_sq +
                          loan_int_rate + loan_int_rate_sq +
                          home_ownership + loan_intent,
              train_df)
```

Note, that even though polynomials were considered as absolutely independent variables, both levels of each of them was selected in the model.

```
preds_nb <- xtabs( ~ predict(m_nb_best, newdata = train_df) + train_df$loan_status)
sum(diag(preds_nb))/ nrow(train_df)
```

```
## [1] 0.8259852
```

The prediction rate for the training data set is worse than for LDA and QDA. Though it's still decent.

Random Forest

For random forest we are going to use all the regressors - not only those chosen by `step()`. However, we are not going to use polynomials.

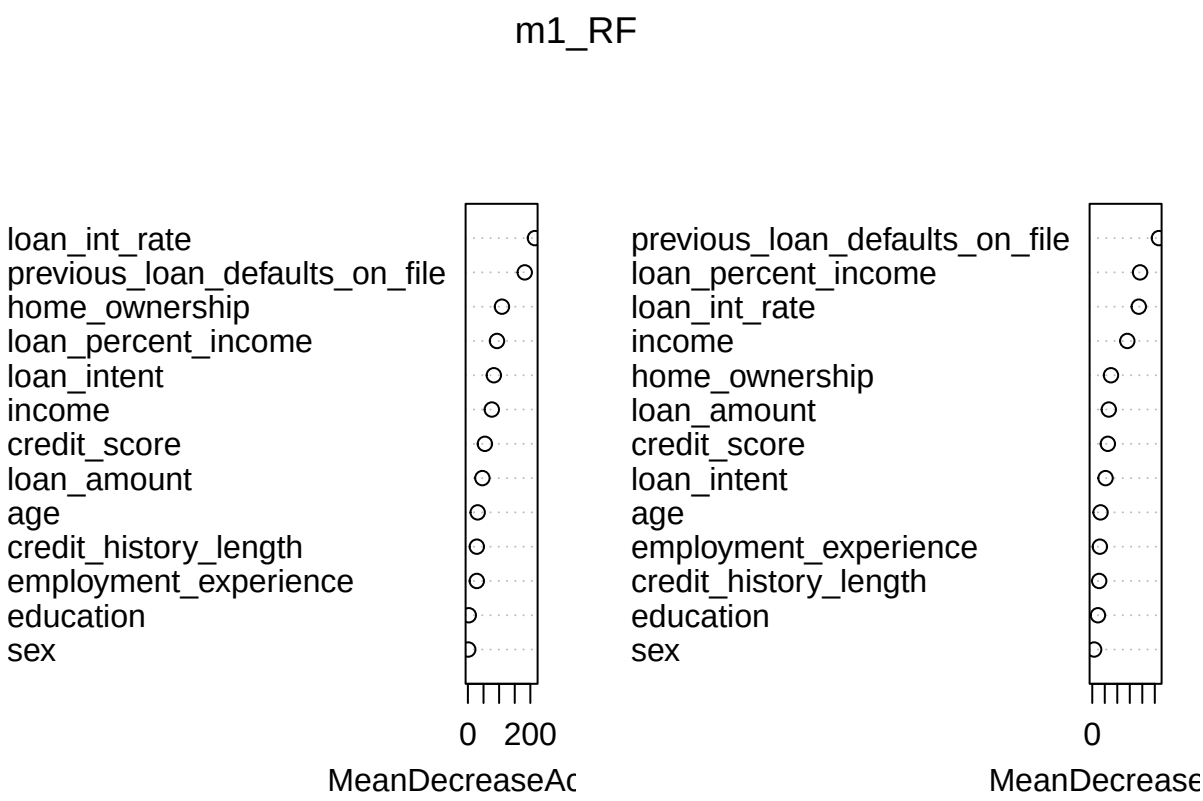
Note that we are using the default number of predictors, $m \approx \sqrt{p} = \sqrt{13} \approx 3.61$, at each split.

```
set.seed(123)
m1_RF <- randomForest(loan_status ~ age + sex + education +
                                    income + employment_experience +
                                    home_ownership + loan_amount + loan_intent +
                                    loan_int_rate + loan_percent_income +
                                    credit_history_length + credit_score +
                                    previous_loan_defaults_on_file,
                                  data = train_df,
```

```
                                    importance = TRUE)
```

Let's have a look at feature importance:

```
varImpPlot(m1_RF)
```



Based on the Gini criterion `previous_loan_defaults_on_file` and `loan_percent_income` are the most important predictors, while `sex` and `education` are the least important.

```
preds_RF <- m1_RF$confusion[,-3]
sum(diag(preds_RF))/ nrow(train_df)
```

```
## [1] 0.9303704
```

Prediction rate is 93% which is the largest so far.

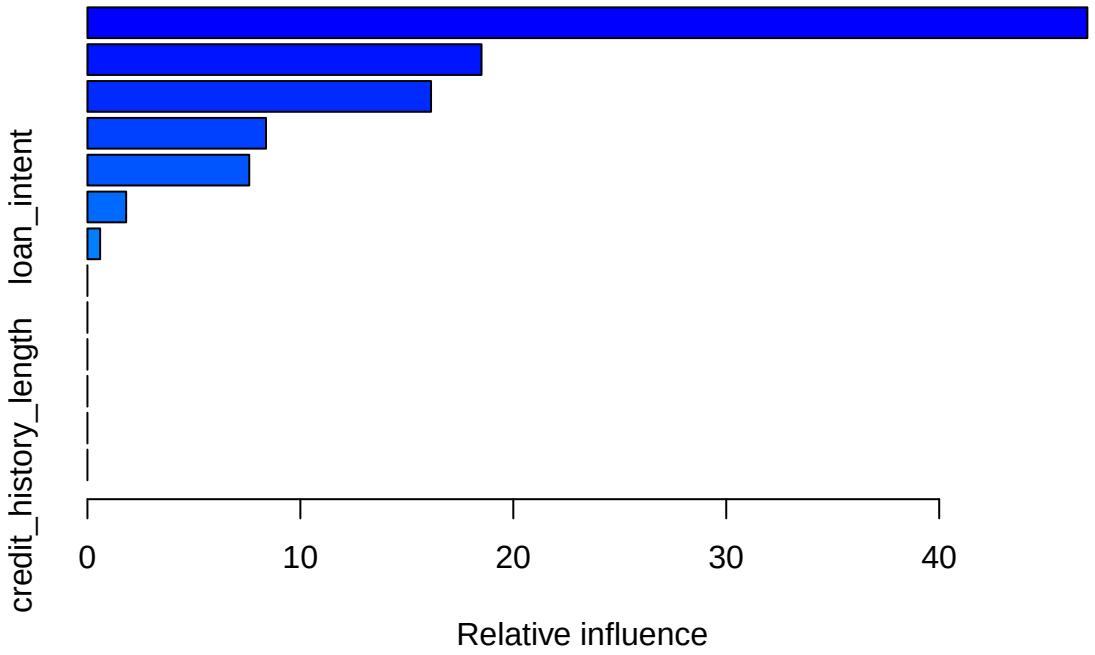
Boosting

Boosting works only with numeric 0 and 1 outcomes, so we make sure this rule is followed.

```
set.seed(123)
m1_boost <- gbm((as.numeric(loan_status)-1) ~ age + sex + education +
                                  income + employment_experience +
                                  home_ownership + loan_amount + loan_intent +
                                  loan_int_rate + loan_percent_income +
                                  credit_history_length + credit_score +
                                  previous_loan_defaults_on_file,
                                data = train_df,
                distribution = "bernoulli",
                interaction.depth = 2)
```

Let's have a look at feature importance:

```r
summary(m1_boost)
```




```
##                                                          var    rel.inf
## previous_loan_defaults_on_file previous_loan_defaults_on_file 46.9583578
## loan_percent_income                       loan_percent_income 18.5064724
## loan_int_rate                                   loan_int_rate 16.1359509
## income                                                 income  8.3859375
## home_ownership                                 home_ownership  7.5995897
## loan_intent                                       loan_intent  1.8178099
## credit_score                                     credit_score  0.5958817
## age                                                       age  0.0000000
## sex                                                       sex  0.0000000
## education                                           education  0.0000000
## employment_experience                 employment_experience  0.0000000
## loan_amount                                       loan_amount  0.0000000
## credit_history_length                 credit_history_length  0.0000000
```

Predictions for the training data set based on the baseline threshold:

```r
preds_boost <- as.numeric(predict(m1_boost, type = "response") >= 0.5)
contig_boost <- xtabs( ~ preds_boost + train_df$loan_status )
sum(diag(contig_boost))/ nrow(train_df)
```

```
## [1] 0.9215704
```

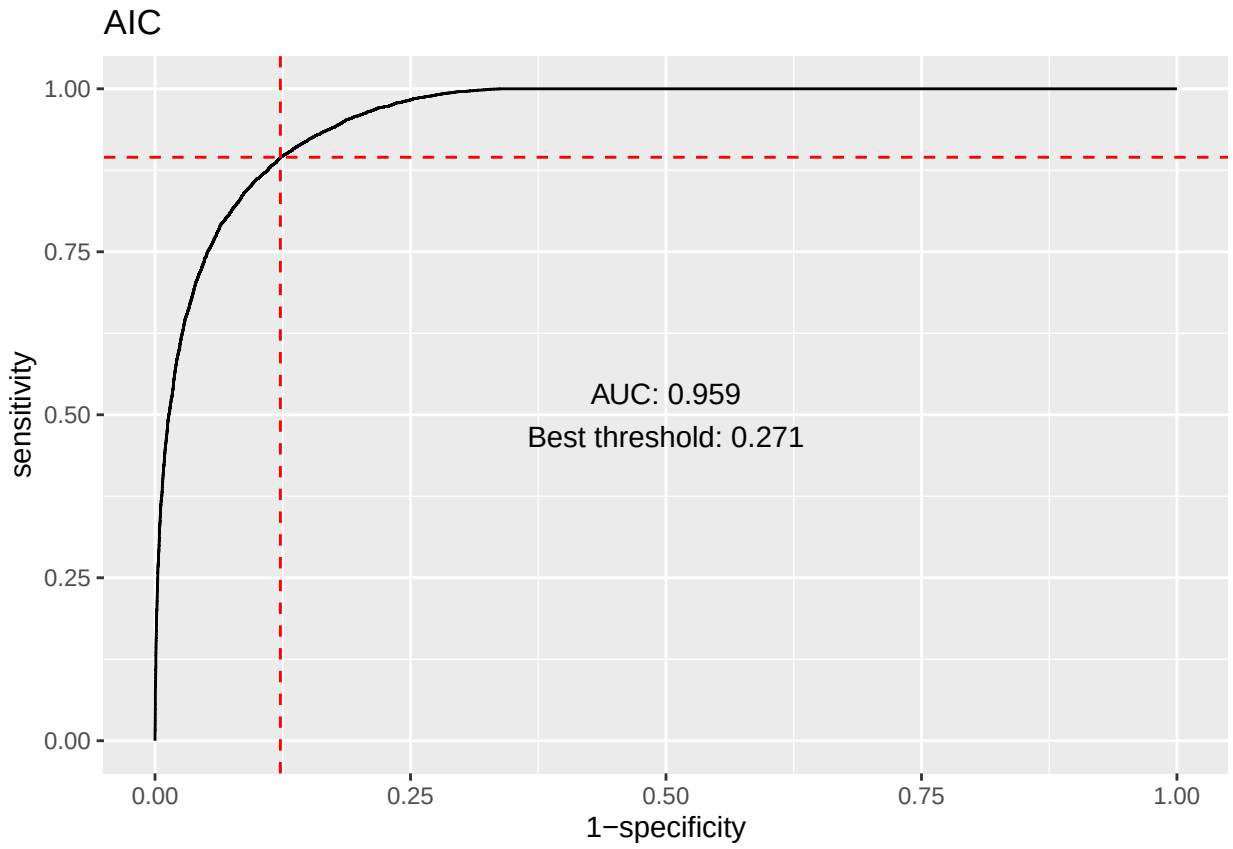
The prediction rate is 92.2%.

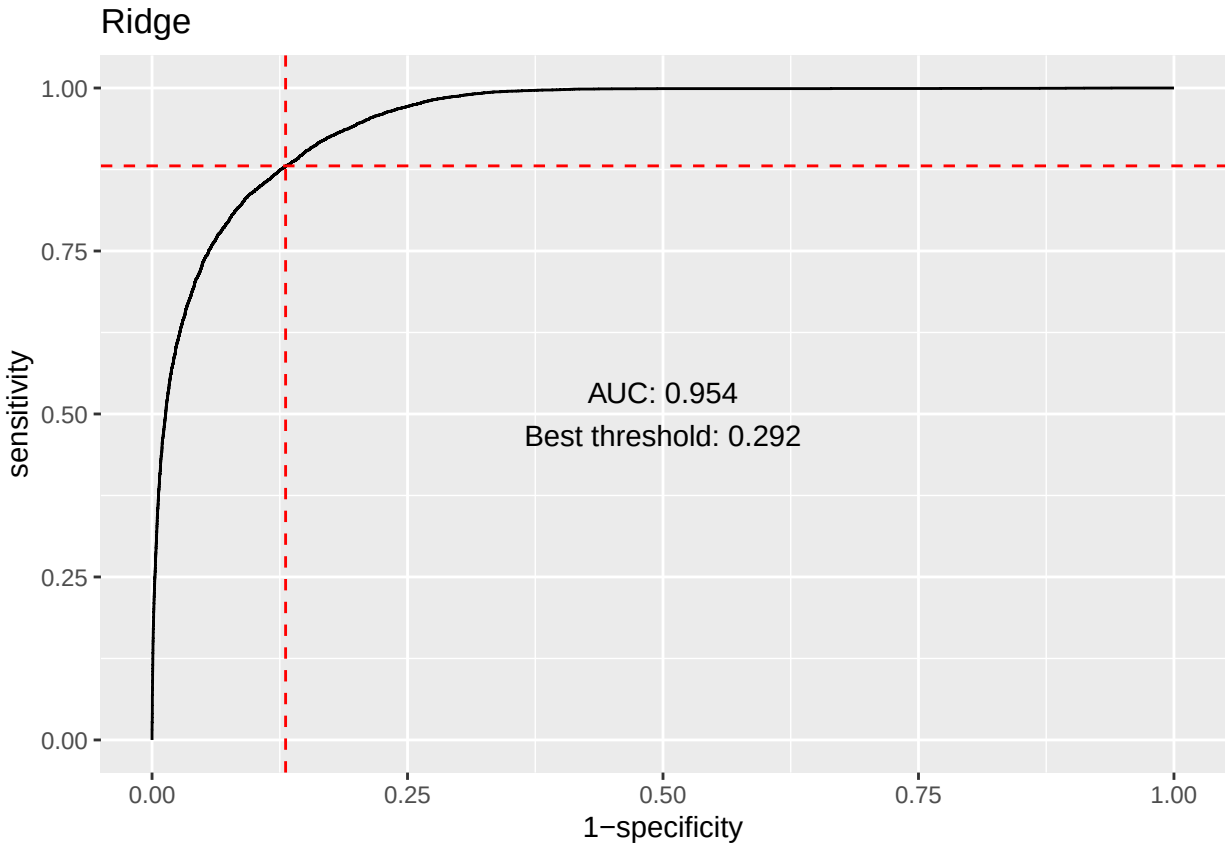
Thresholds for Models

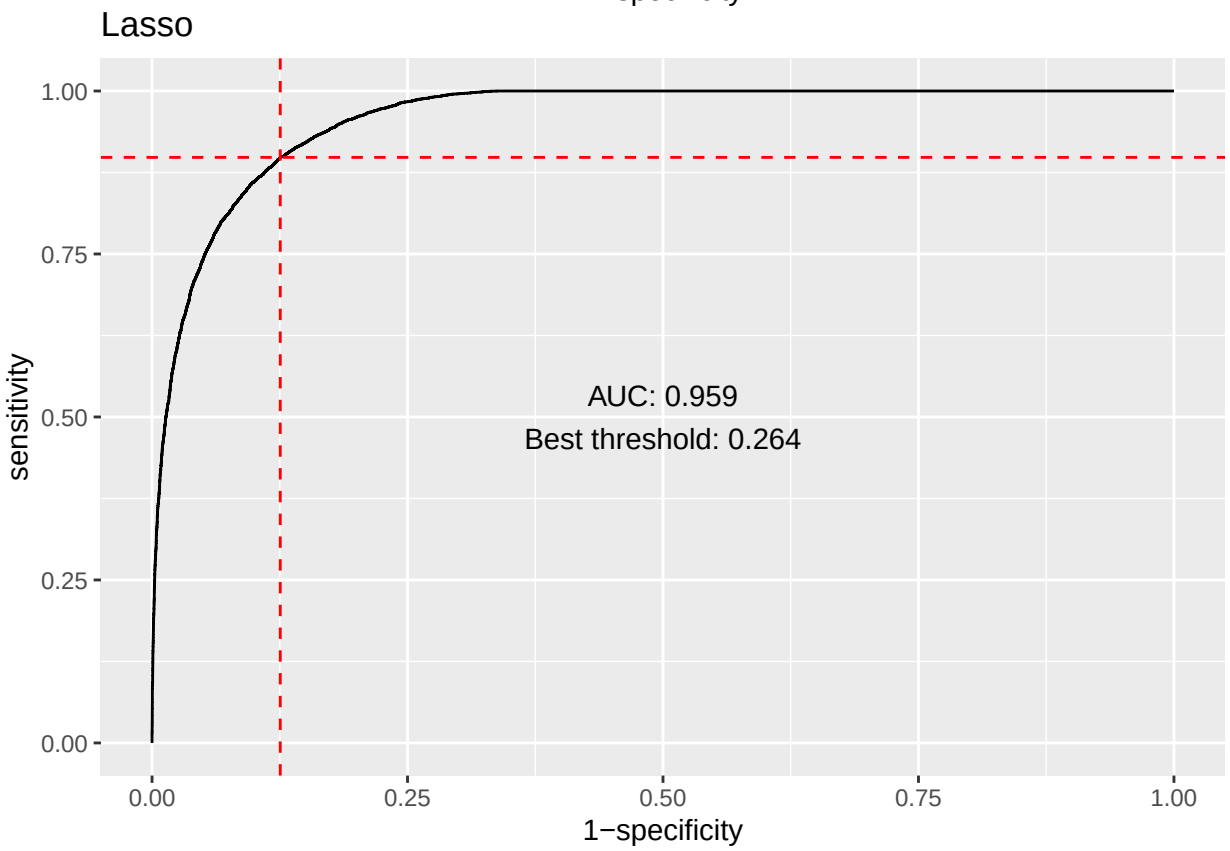
Before comparing different models on the test data set, we need to find the best thresholds. To do this we'll find the threshold that corresponds to the most top-left corner on the AUC-curve.

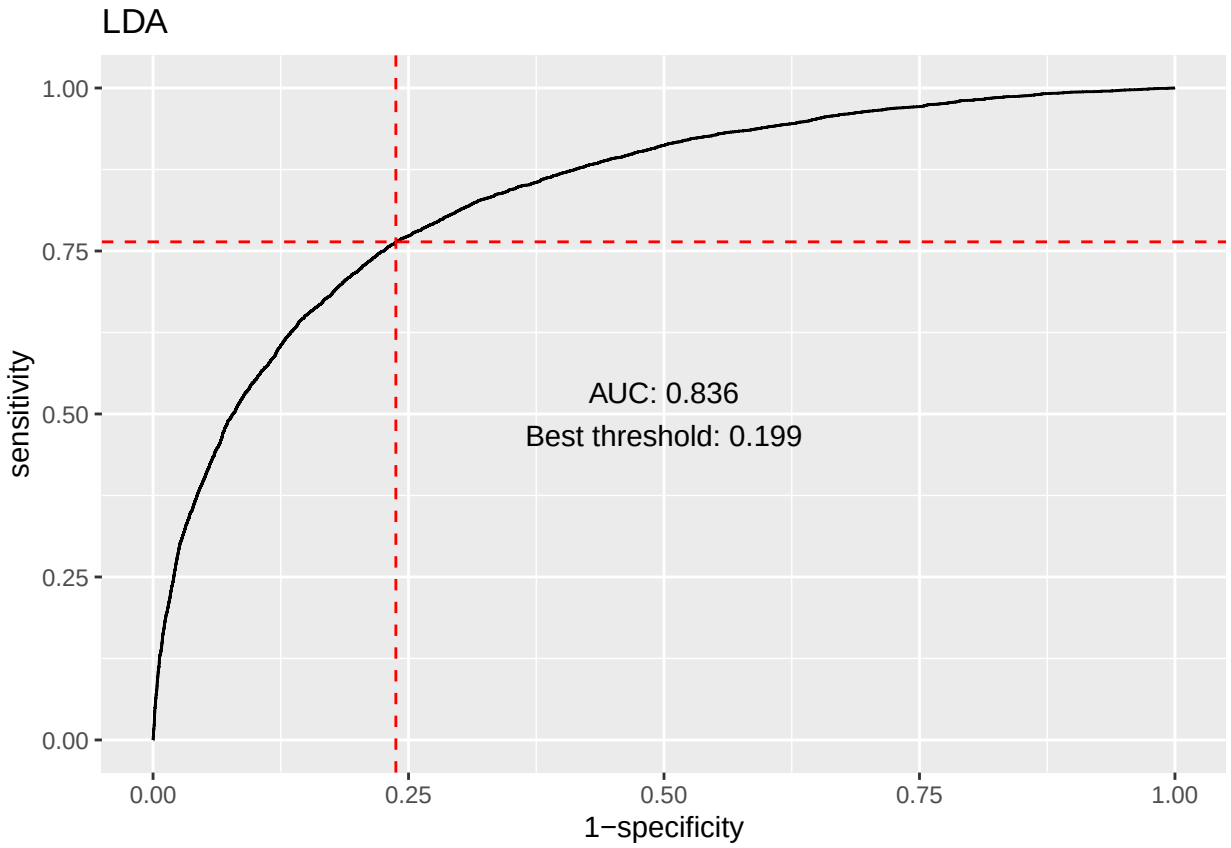
```
x_train <- model.matrix(m2_logistic)[,-1] # for Ridge and Lasso

# predictions from different models
all_models <- list(

  # AIC based
  m1_logistic_aic1$fitted.values,
  # Ridge
  as.vector(predict(m1_ridge,
          s = m1_ridge$lambda,
          type = "response",
          newx = x_train)),
  # Lasso
  as.vector(predict(m1_lasso,
          s = m1_lasso$lambda,
          type = "response",
          newx = x_train)),
  # LDA
  predict(m_lda_best)$posterior[,2],
  # QDA
  predict(m_qda_best)$posterior[,2],
  # Naive Bayes
  predict(m_nb_best,
          newdata = train_df,
          type = "raw")[,2],
  # Random Forest
  predict(m1_RF, type = "prob")[,2],
  # Boosting
  predict(m1_boost, type = "response")
  )
models_names <- c("AIC", "Ridge",
                  "Lasso", "LDA",
                  "QDA", "NB","RF","Boost")
# store AUC parameters here
AUCs <- matrix(ncol = 5, nrow = length(all_models))
colnames(AUCs) <- c("threshold", "sensitivity", "specificity",
                    "AUC","model")

# Plots
for (i in 1:nrow(AUCs)){
  roc_m <- roc(
    response = train_df$loan_status,
    predictor = all_models[[i]])

  best_coords <- coords(roc_m,
                        x = "best",
                        best.method = "closest.topleft",
                        ret = c("threshold",
                                "sensitivity",
```

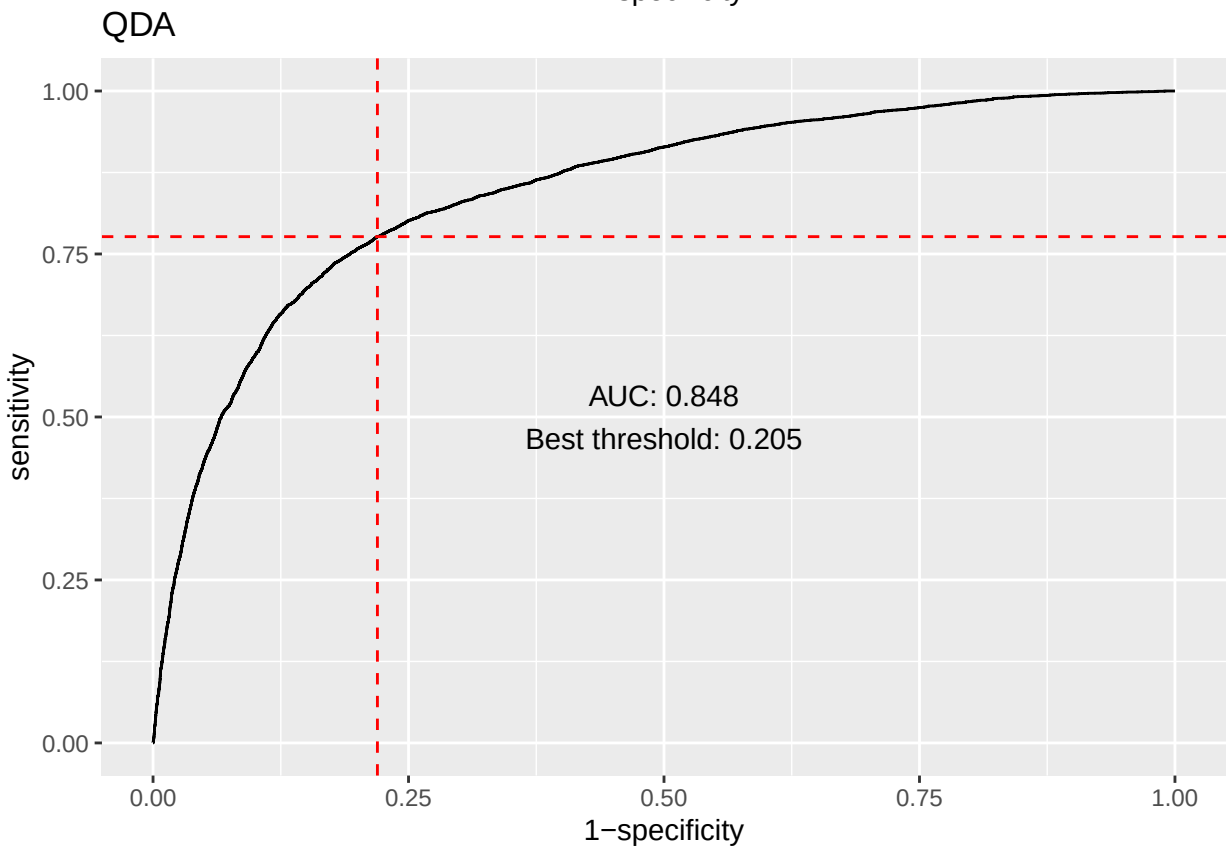
```
                                "specificity")) %>%
  dplyr::mutate(AUC = as.numeric(roc_m$auc),
                model = models_names[i])

  AUCs[i,] <- as.vector(unlist(best_coords))

  p <- ggroc(roc_m, legacy.axes = TRUE) +
    labs(title = models_names[i]) +
    annotate('text', x = .5, y = .5,
           label = paste0('AUC: ', round(best_coords["AUC"], 3),
                          '\nBest threshold: ',
                          round(best_coords["threshold"], 3))) +
    geom_vline(xintercept = as.numeric(1 - best_coords["specificity"]),
               linetype = "dashed", color = "red") +
    geom_hline(yintercept = as.numeric(best_coords["sensitivity"]),
               linetype = "dashed", color = "red")
  print(p)
}
```

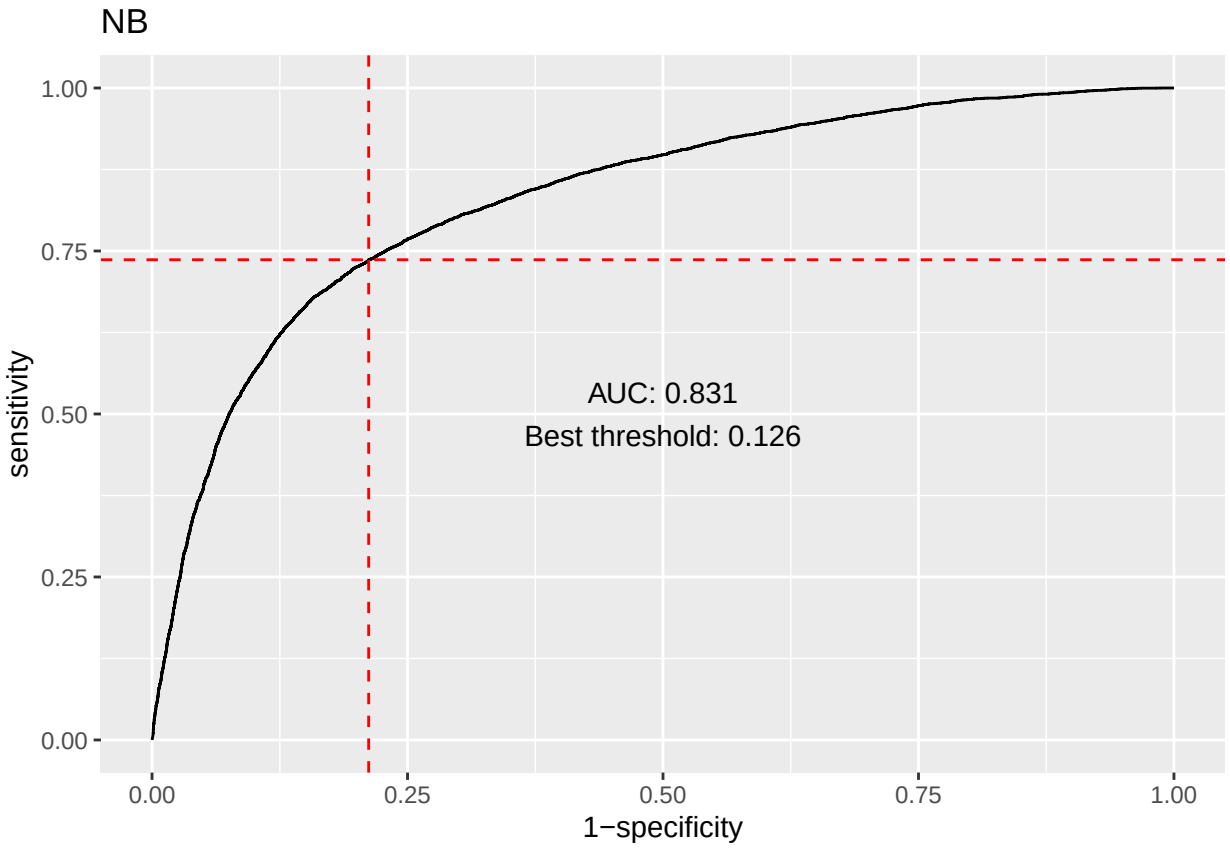


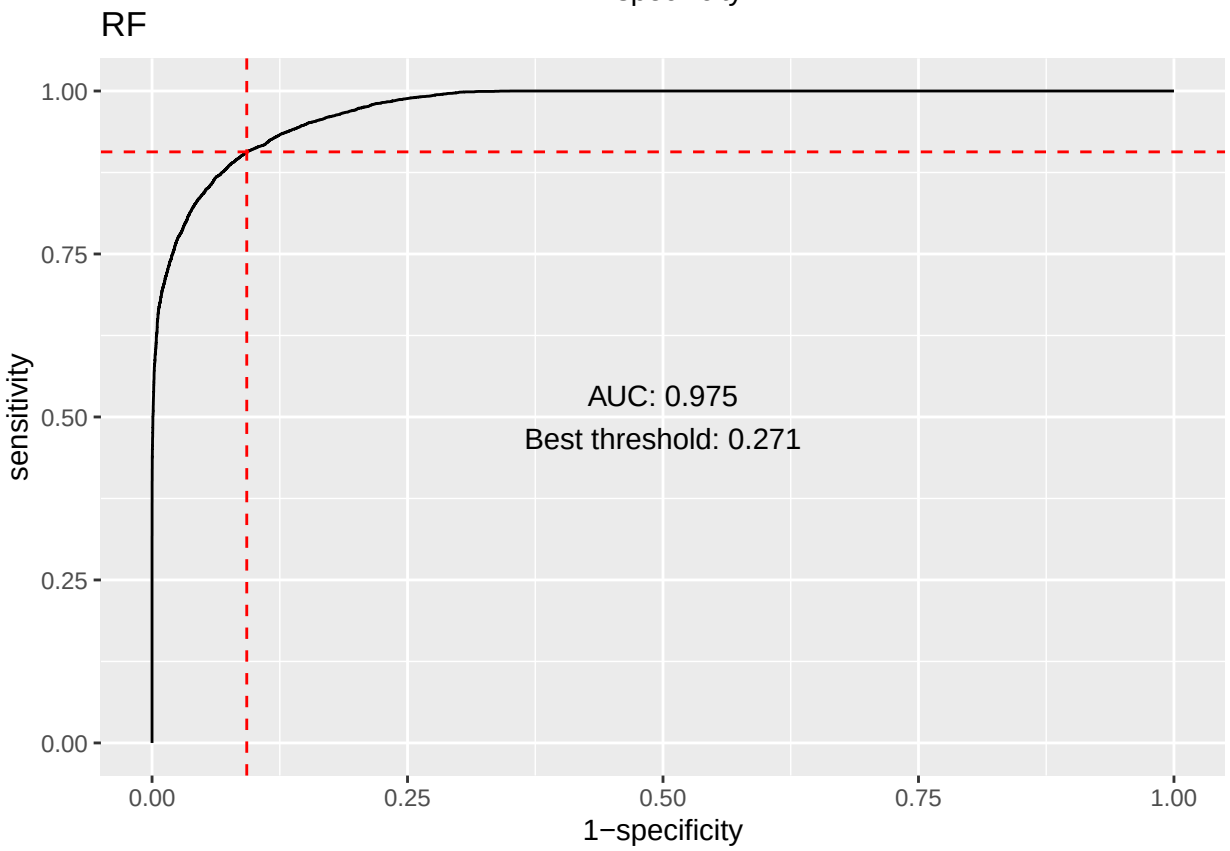

Ridge
1.00
0.75
0.50
0.25
0.00
sensitivity
AUC: 0.954
Best threshold: 0.292
0.00
0.25
0.50
0.75
1.00
1−specificity

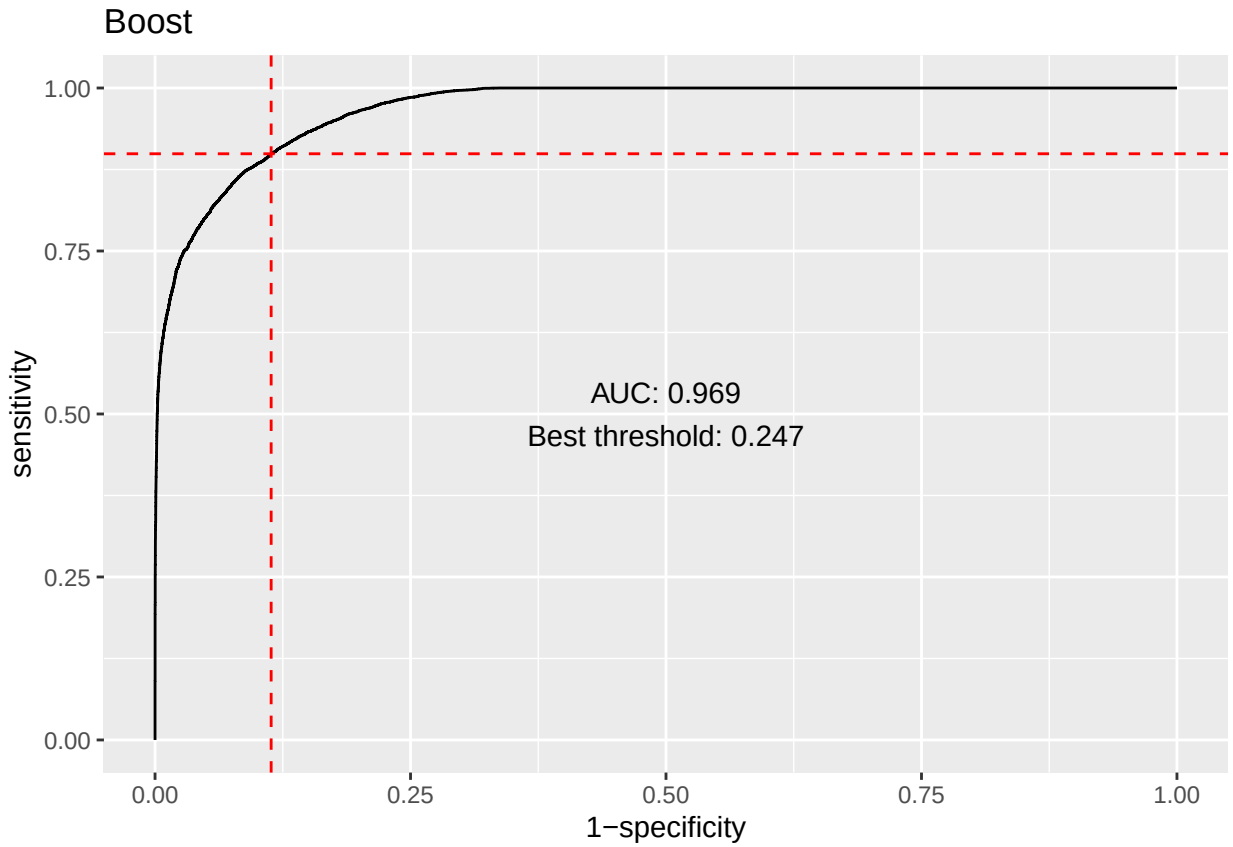

Lasso
1.00
0.75
0.50
0.25
0.00
sensitivity
AUC: 0.959
Best threshold: 0.264
0.00
0.25
0.50
0.75
1.00
1−specificity


LDA
1.00
0.75
0.50
0.25
0.00
sensitivity
AUC: 0.836
Best threshold: 0.199
0.00
0.25
0.50
0.75
1.00
1−specificity


QDA
1.00
0.75
0.50
0.25
0.00
sensitivity
AUC: 0.848
Best threshold: 0.205
0.00
0.25
0.50
0.75
1.00
1−specificity


NB
1.00
0.75
0.50
0.25
0.00
sensitivity
AUC: 0.831
Best threshold: 0.126
0.00
0.25
0.50
0.75
1.00
1−specificity


RF
1.00
0.75
0.50
0.25
0.00
sensitivity
AUC: 0.975
Best threshold: 0.271
0.00
0.25
0.50
0.75
1.00
1−specificity




```
# make it a data frame to refer to it later
AUCs <- as.data.frame(AUCs)
```

Based on the AUC-curves for training data sets, we can hypothesize that the boosting and random forest models will perform better than other models.

Predictions

Let's create vectors with class predictions from each model for the test data set:

```
# Model matrix for test df (for Ridge and Lasso)
x_test <- model.matrix(loan_status ~ age + sex + education +
                income + employment_experience +
                home_ownership +
                poly(loan_amount,2) + loan_intent + poly(loan_int_rate,2) +
                poly(loan_percent_income,2) + credit_history_length +
                poly(credit_score,2) + previous_loan_defaults_on_file +
                education:employment_experience +
                (employment_experience +
                credit_score):home_ownership,
                test_df)[,-1]
# AIC based
class.preds.AIC.logistic <- as.numeric(predict(m1_logistic_aic1,
                                               newdata = test_df,
                                               type = "response") > AUCs$threshold[1])

# Ridge
class.preds.Ridge <- as.numeric(as.vector(
  predict(m1_ridge,
          s = m1_ridge$lambda,
          type = "response",
          newx = x_test)) > AUCs$threshold[2])

# Lasso
class.preds.Lasso <- as.numeric(as.vector(
  predict(m1_lasso,
          s = m1_lasso$lambda,
          type = "response",
          newx = x_test)) > AUCs$threshold[3])

# LDA
class.preds.LDA <- as.numeric(as.vector(
  predict(m_lda_best,
          newdata = test_df)$posterior[,2]) > AUCs$threshold[4])

# QDA
class.preds.QDA <- as.numeric(as.vector(
  predict(m_qda_best,
          newdata = test_df)$posterior[,2]) > AUCs$threshold[5])

# Naive Bayes
class.preds.NB <- as.numeric(as.vector(
  predict(m_nb_best,
          newdata = test_df,
          type = "raw")[,2]) > AUCs$threshold[6])

# Random Forest
class.preds.RF <- as.vector(as.numeric(predict(m1_RF,
                                   newdata = test_df,
                                   type = "prob")[,2] > AUCs$threshold[7]))
```

```r
# Boosting
class.preds.Boost <- as.numeric(predict(m1_boost, #n.trees = best_n.trees,
                                        newdata = test_df,
                                        type = "response") > AUCs$threshold[8])
```

Now let's build contingency tables and calculate the performance rate:

```r
test.preds <- list(class.preds.AIC.logistic,
                   class.preds.Ridge,
                   class.preds.Lasso,
                   class.preds.LDA,
                   class.preds.QDA,
                   class.preds.NB,
                   class.preds.RF,
                   class.preds.Boost)
methods.preds <- c("AIC", "Ridge", "Lasso", "LDA", "QDA", "NB", "RF", "Boost")

for (i in 1:length(test.preds)){
  cont.table <- xtabs( ~ test.preds[[i]] + test_df$loan_status)
  prop.cor <- sum(diag(cont.table))/ nrow(test_df)

  print(paste0(methods.preds[i],": ",round(prop.cor,3)))
  print(cont.table)
}
```

```
## [1] "AIC: 0.372"
##                 test_df$loan_status
## test.preds[[i]]    0    1
##               0 1944  281
##               1 6781 2244
## [1] "Ridge: 0.868"
##                 test_df$loan_status
## test.preds[[i]]    0    1
##               0 7599  356
##               1 1126 2169
## [1] "Lasso: 0.479"
##                 test_df$loan_status
## test.preds[[i]]    0    1
##               0 3192  325
##               1 5533 2200
## [1] "LDA: 0.763"
##                 test_df$loan_status
## test.preds[[i]]    0    1
##               0 6700  636
##               1 2025 1889
## [1] "QDA: 0.783"
##                 test_df$loan_status
## test.preds[[i]]    0    1
##               0 6901  615
##               1 1824 1910
## [1] "NB: 0.777"
##                 test_df$loan_status
## test.preds[[i]]    0    1
##               0 6913  695
```

```
##               1 1812 1830
## [1] "RF: 0.903"
##                test_df$loan_status
## test.preds[[i]]    0    1
##               0 7898  262
##               1  827 2263
## [1] "Boost: 0.884"
##                test_df$loan_status
## test.preds[[i]]    0    1
##               0 7689  267
##               1 1036 2258
```

It turns out that random forest provides the most accurate results, while boosting and Ridge regression provide slightly worse results. Contrary, AIC-based and Lasso logistic regressions show results which are even worse than the random guessing rate which is:

```
(preds <- (table(test_df$loan_status) / nrow(test_df) ))
```

```
##
##         0         1
## 0.7755556 0.2244444
```

```
(rand.guess <- sum(preds^2))
```

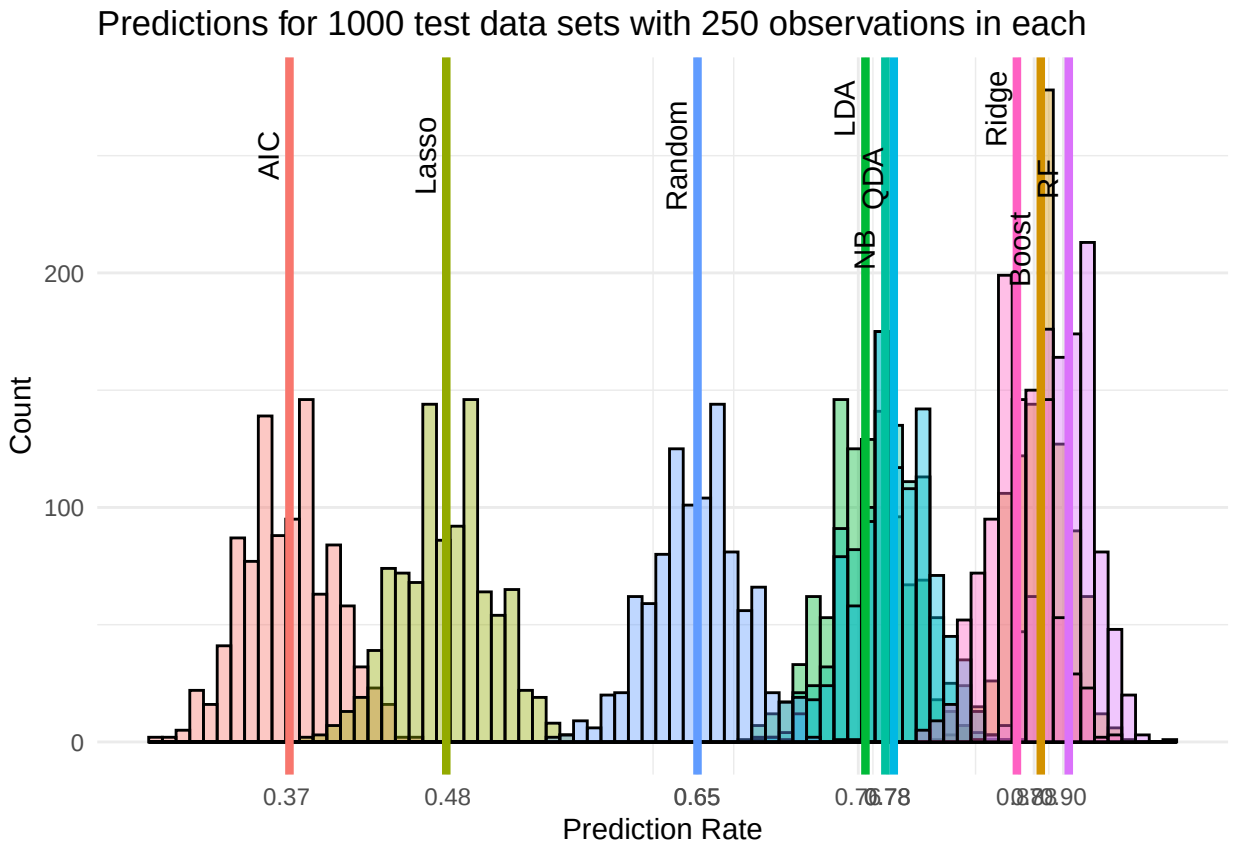
```
## [1] 0.6518617
```

Permutations

Finally, to show the stability of the results, let's compare the predictions for smaller data sets. Let's create 1,000 test data sets with 250 observations in each and compare the results:

```
set.seed(12345)
N_new_test_dfs <- 1000
small_test_df_size <- 250
preds_small_size <- matrix(ncol = length(test.preds)+1, nrow = N_new_test_dfs)

for (test_df_N in 1:N_new_test_dfs){
  # random 250 observations
  rand.rows <- sample(1:nrow(test_df), small_test_df_size)
  resp <- test_df$loan_status

  # for each model
  for (i in 1:length(test.preds)){
    cont.table <- xtabs( ~ test.preds[[i]][rand.rows] + resp[rand.rows])
    prop.cor <- sum(diag(cont.table))/ small_test_df_size
    preds_small_size[test_df_N,i] <- prop.cor
  }

  # for random guessing
  rang.guess.preds <- sample(c(0,1), 250, replace = T, prob = preds)
  cont.table <- xtabs( ~ rang.guess.preds + resp[rand.rows])
  prop.cor <- sum(diag(cont.table))/ small_test_df_size
  preds_small_size[test_df_N,length(test.preds)+1] <- prop.cor
}

# Visualization
```

```r
preds_small_size_df_wide <- as.data.frame(preds_small_size)
colnames(preds_small_size_df_wide) <- c(methods.preds, "Random")
preds_small_size_df <- preds_small_size_df_wide %>%
  pivot_longer(cols = everything(),
               names_to = "Method",values_to = "Pred.rate")
preds_small_size_df$Method <- as.factor(preds_small_size_df$Method)
vline_df <- data.frame(
  Method = c(methods.preds, "Random"),
  Mean = colMeans(preds_small_size),
  y = c(250,270,250,270,240,210,240,210,250)
)

ggplot(preds_small_size_df, aes(x=Pred.rate,
                                fill = Method))+
  geom_histogram(position = "identity", alpha = .4,bins = 75, color="black")+
  geom_vline(data = vline_df, aes(xintercept = Mean, color = Method), linewidth = 1.5) +
  geom_text(data = vline_df, aes(x = Mean, y = y, label = Method),
          color = "black", angle = 90, vjust = -0.5, size = 4)+
  scale_x_continuous(breaks = round(c(vline_df$Mean,
                                      rand.guess),2))+
  labs(title = paste0("Predictions for ",N_new_test_dfs,
                      " test data sets with ",small_test_df_size,
                      " observations in each"),
       x = "Prediction Rate",
       y = "Count")+
  theme_minimal()+
  theme(legend.position = "none")
```



Summary statistics:

```
apply(preds_small_size_df_wide, 2, function(x) c(mean=mean(x), sd = sd(x)))
```

```
##            AIC      Ridge      Lasso        LDA        QDA         NB
## mean 0.3719160 0.86822400 0.47889600 0.76488800 0.78420400 0.77849600
## sd   0.0309566 0.02168733 0.03160513 0.02721971 0.02655605 0.02593613
##              RF      Boost     Random
## mean 0.90361200 0.88450800 0.65029200
## sd   0.01798302 0.01882084 0.03155767
```

Summary

The best results were provided by random forest (90.3% accuracy), while the worst results were given by a simple logistic regression, acquired as a result of model selection (AIC-based), with 37.2% accuracy.

All the models are robust and show good prediction rates not only for big data sets (11,250 observations), but also for small (250 observations).