# Report

*Aneta Przydrozna*

*25.04.2020*

### The goal of the tasks is to build a model to assess credit risk

**Next steps to complete the goal:**

•Prepare the environment

•Load data

•Exploring the data

•Preparation of features

•Selection type of models that meet the assumptions

•Division of data into training and test sets

•Learning phase on a training set

•Check the result on a test set

•Choose the most efective model

•Model improvement

•Presentation of results

•Model implementation

## Libraries

```
library(stats)  # glm
library(caret)  # prepare data sets (createDataPartition,confusionMatrix)
library(tidyr)  # drop_na
library(dplyr)  # %>%
library(pROC)   # RoC curve and AUC
library(ROSE)   # ovun.sample
library(MASS)   # stepAIC
library(xgboost)# xgb.train, xgb.DMatrix
library(randomcoloR)
library(smbinning)
```

## Preparation of Data

Our goal is to classify, if borrower will repay the loan or not. In Lending Club data we have column with 10 labels describing the repayment status of the loan: “Charged Off”, “Current”, “Default”, “Does not meet the credit policy. Status:Charged Off”, “Does not meet the credit policy. Status:Fully Paid”, “Fully Paid”,“In Grace Period”, “Issued”, “Late (16-30 days)”, “Late (31-120 days)”. But we want to First group contain “Fully paid” status, second contain every others status without “Current” which we can’t define.

To let my computer work I reduced amount of columns, choosing only those that haven't got NA values. I also reduced amount of rows with random undersampling method (randomly chooses observations from majority class which are eliminated until the data set gets balanced).

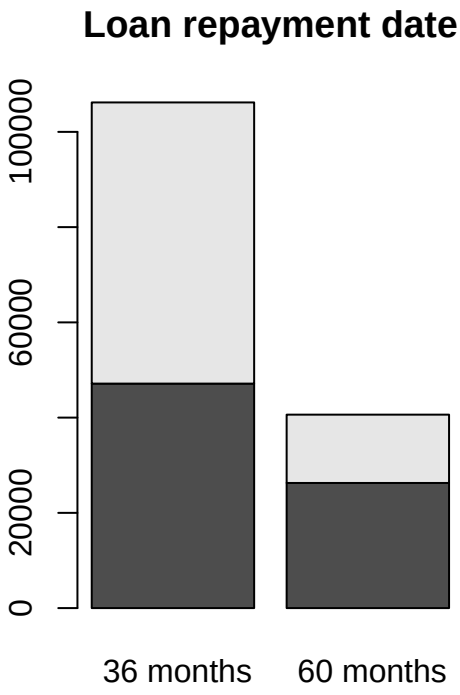
I could use synthetic sampling technique or weight for imbalanced data set but "loan" has got enough samples so I reduced size of data, leaving almost 150 000 rows where half contain "Paid" status and second part contain "Unpaid" status.

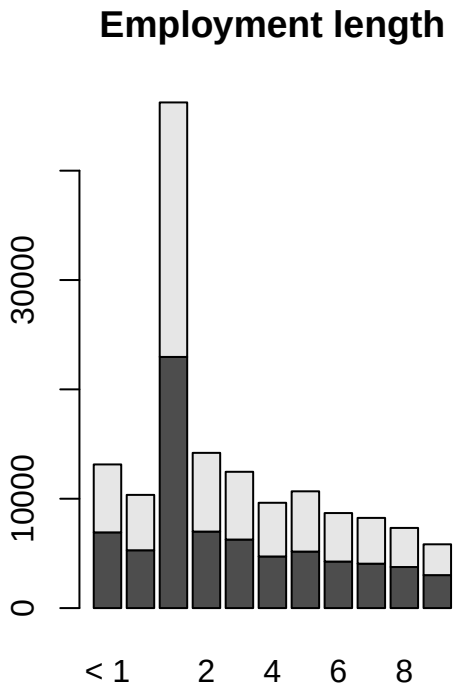
```
loan<-read.csv("loan.csv")
#choosing features from 75 columns
loan<-loan[,c(3,6,7,8,9,10,12,14,17,21,25,28,33,34,39)]
#remove every rows with NA values
loan<-loan %>% drop_na()

#now we can remove rows with loan status= Current
loan<-subset(loan,loan_status!="Current")

# 1= Paid,  0=Unpaid (everythink else)
loan$loan_status<-ifelse(loan$loan_status=="Fully Paid",1,0)

#remove part of rows with status "paid" to reduce size of data to almost 150 000 samples
loan <- ovun.sample(loan_status ~ ., data = loan, method = "under")$data
```

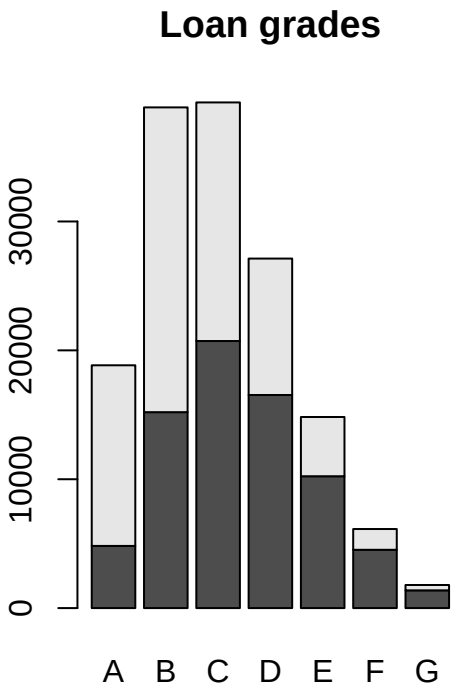
## Data exploration

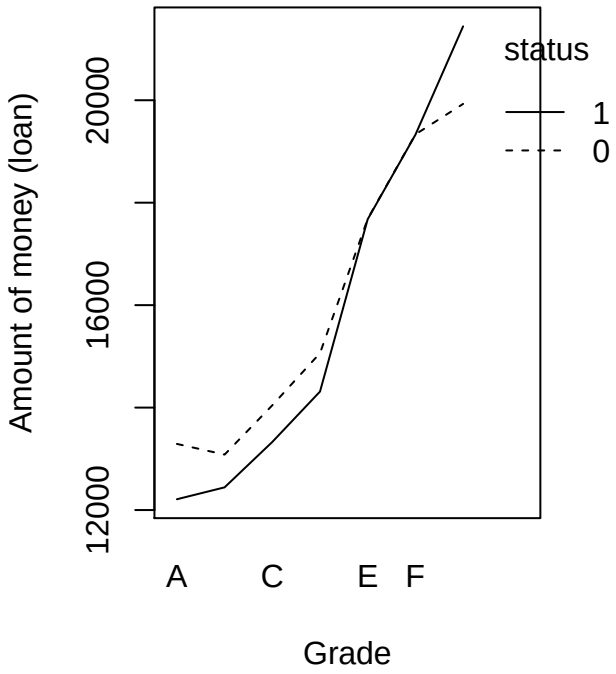
Saved features have bigger or smaller influence on loan status. For example number of months allocated for loan repayment. Shorter term is better option (56% paid for 36 monts and 35% for 60 monts). Percentage of repaid in different length of employment have frequency between 47% and 52% so we could suppose that this feature is neutral for loan status. Credit grade is character (A-G). The worse the credit grade, the higher the return percentage, and the bigger loan.

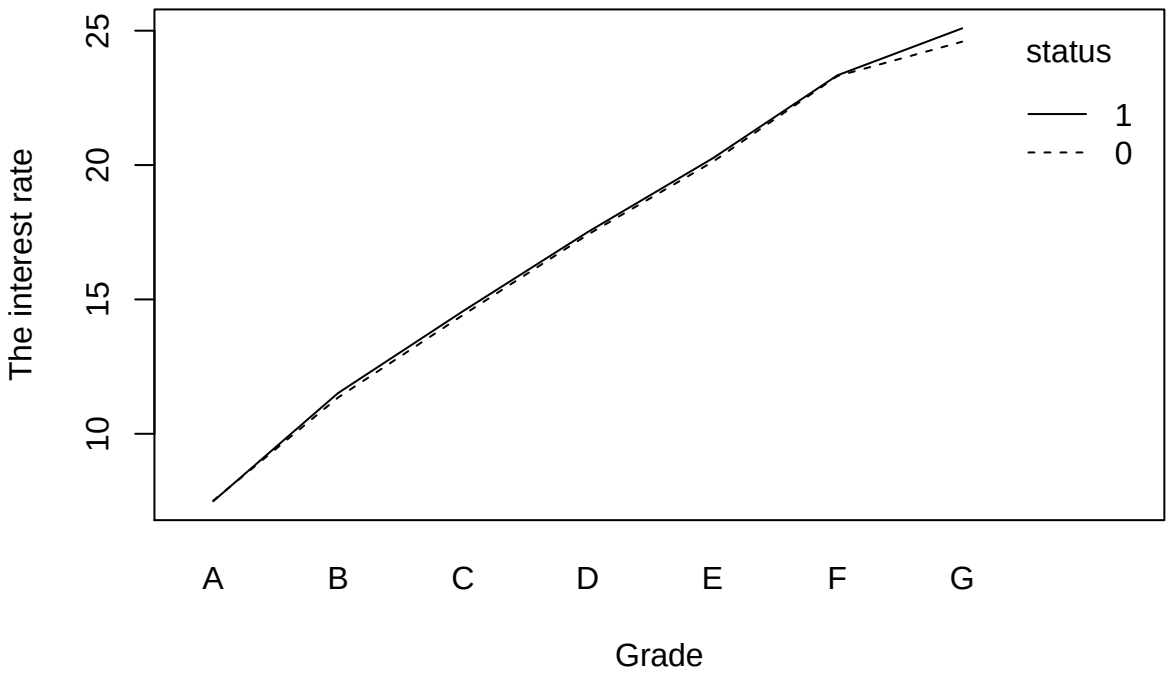
These are some examples of interactions. White section show how much loan status have label "Paid" and the dark one tell us about "Unpaid" status.

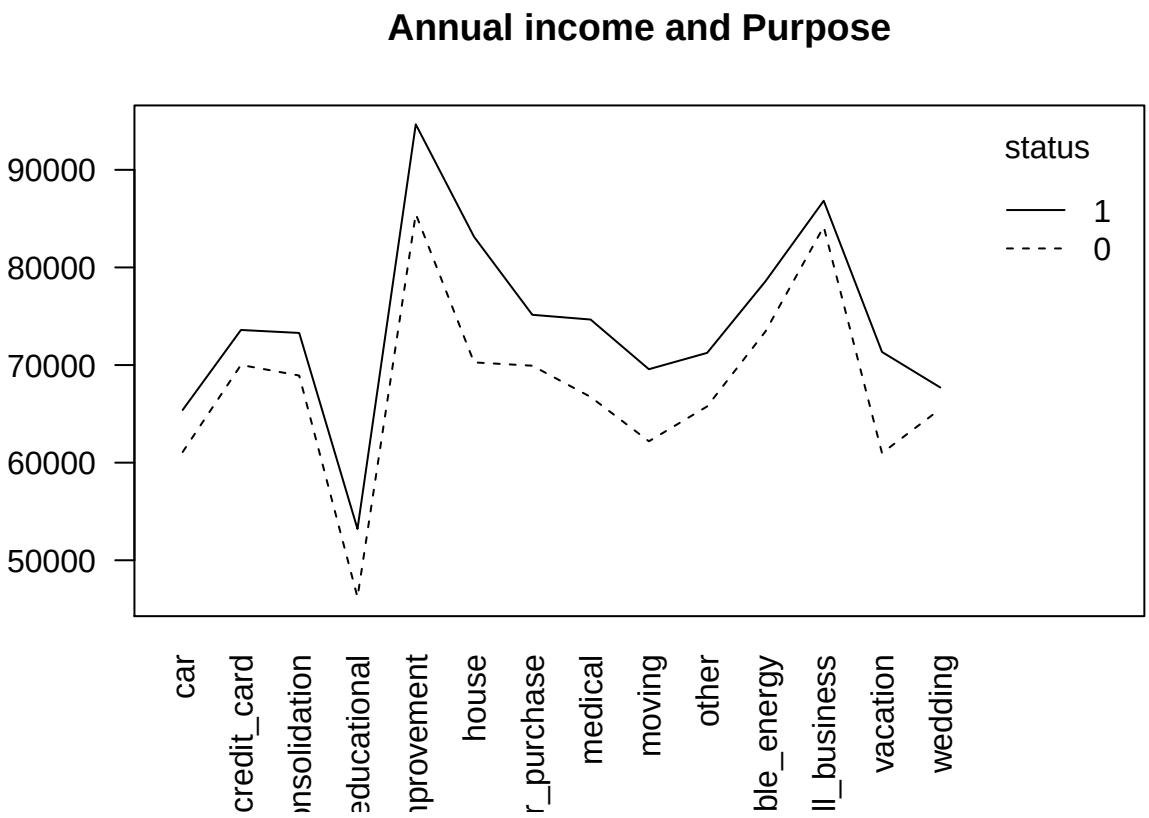

Loan repayment date
0
20000
60000
100000
36 months
60 months


Employment length
0
10000
30000
< 1
2
4
6
8


Loan grades
0
10000
20000
30000
A
B
C
D
E
F
G


Amount of money (loan)
12000
16000
20000
A
C
E
F
Grade
status
1
0


The interest rate
10
15
20
25
A
B
C
D
E
F
G
Grade
status
1
0




## Training set and test set

```
loan$loan_status<-as.factor(loan$loan_status)
train <- createDataPartition(loan$loan_status, p = 0.8)[[1]] #1 for take integer from list  # from care
loan_test <- loan[-train,]
loan_train <- loan[train,]
```

## Logistic regression with stepwise selection of features

```
model_glm<-glm(loan$loan_status~., family = binomial,data=loan) #link="logit" default in binomial
step_glm<-stepAIC(model_glm ,steps =500)
model_glm<-glm(formula=step_glm$formula, family = "binomial",data=loan_train)
```

## Logistic regression with selection of features based on IV

```
factor_vars <- c ("term", "grade", "emp_length")  # other data have too many labels

continuous_vars <- c("loan_amnt", "int_rate","installment", "annual_inc", "dti", "inq_last_6mths","revol
```

```
iv_df <- data.frame(VARS=c(factor_vars, continuous_vars), IV=numeric(13))  # init for IV results

# compute IV for categoricals
for(factor_var in factor_vars){
  smb <- smbinning.factor(loan_train, y="loan_status", x=factor_var)  # WOE table
  if(class(smb) != "character"){ # heck if some error occured
    iv_df[iv_df$VARS == factor_var, "IV"] <- smb$iv
  }
}

# compute IV for continuous vars
for(continuous_var in continuous_vars){
  smb <- smbinning(loan_train, y="loan_status",  x=continuous_var)
  if(class(smb) != "character"){  # any error while calculating scores.
    iv_df[iv_df$VARS == continuous_var, "IV"] <- smb$iv
  }
}

model_iv<-glm( loan_status~grade+int_rate+term+dti, family = "binomial",data=loan_train)
```

## XGBoost on default parameters

```
test_watchlist = list(
  test = xgb.DMatrix(
    data = as.matrix(loan_train_num),
    label = as.numeric(loan_train$loan_status)-1
  )
)

model_xgb = xgb.train(
  data= xgb.DMatrix(
    data = as.matrix(loan_train_num),
    label = as.numeric(loan_train$loan_status)-1
  ),
  objective = "binary:logistic",
  nrounds = 350,
  watchlist = test_watchlist,
  eval_metric = "auc",
  early.stop.round = 10)
```

## XGBoost grid search

```
etas = c(0.1,0.3)
alphas = c(0,0.5,1)
lambdas = c(0,0.5,1)


gbm_perf = data.frame(eta=numeric(0),alpha=numeric(0),lambda=numeric(0),auc=numeric(0))
for(eta in etas){
```

```
  for(alpha in alphas){
    for(lambda in lambdas){
      model = xgb.train(
        data= xgb.DMatrix(
          data = as.matrix(loan_train_num),
          label = as.numeric(loan_train$loan_status)-1
        ),
        objective = "binary:logistic",
        nrounds = 350,
        watchlist = test_watchlist,
        eval_metric = "auc",
        alpha = alpha,
        early.stop.round = 10,
        lambda = lambda,
        eta = eta)
      gbm_perf[nrow(gbm_perf)+1,] = c(eta,alpha,lambda,model$best_score)
    }
  }
}

gbm<-gbm_perf[which(gbm_perf[,4]>=max(gbm_perf[,4])) ,]

model_xgb_grid = xgb.train(
  data= xgb.DMatrix(
    data = as.matrix(loan_train_num),
    label = as.numeric(loan_train$loan_status)-1
  ),
  objective = "binary:logistic",
  nrounds = 350,
  watchlist = test_watchlist,
  eval_metric = "auc",
  early.stop.round = 10,
  alpha = gbm$alpha,
  lambda = gbm$lambda,
  eta = gbm$eta)
```

## Models comparison

```
comparison<-function(model,loan_test_status,loan_test,name){

  pred = predict(model,loan_test,type="response")
  roc= roc(response = loan_test_status,predictor = pred)
  AUC = auc(roc)

  threshold<-coords(roc, "best", "threshold", transpose = TRUE)
  pred<-ifelse(pred> threshold[1],1,0)
  cm <- confusionMatrix(as.factor(pred), loan_test_status)

  result = c(name,
             round(AUC,3),
             round(cm$overall["Accuracy"],3),
```

```
            round(cm$byClass["Sensitivity"],3),
            round(cm$byClass["Specificity"],3),
            round(2*AUC-1,3 )
  )

  plot<-plot(roc,legacy.axes = TRUE,print.auc = TRUE,col=randomColor(luminosity = "dark"),main=name)
  return(result)
}
```

```
                        Model   AUC Accuracy Sensitivity Specificity
1           logistic regression 0.699    0.647       0.660       0.633
2 logistic regression- Stepwise 0.699    0.647       0.656       0.637
3       logistic regression- IV 0.686    0.636       0.643       0.629
4   xgboost- default parameters 0.711    0.652       0.682       0.623
5          xgboost- grid search 0.711    0.651       0.684       0.619
     GC
1 0.398
2 0.398
3 0.371
4 0.423
5 0.422
```

Above we can see table with all used methods of prediction and some of statistics that show diffrends between every of them. The closer to 1 are statistics values for each model, the better it is. If (Area Under The Curve) AUC=0.5 our model predict as good as random predict . Gini coefficient is depend on AUC (2*AUC-1) and if GC=0 method work also as good as random predict. We want choose model with balanced values. Table show us that for loan datasets, XGBoost with default parameters predict most accurately, but these with grid search parameters has very similar statistics. Logistic regression on subinning package works the worst, but it could be depend with levels of features (I didn't work on them too much). Generally models' statistics are even close to each others but not very good to predict loan repairment.