Assignment 1A - IDS 572

Prathamesh Bapat . Reza Amini . So Hee Choi

14 February, 2021

```
#Ignore this - To knit the related output to min 10 lines
library(knitr)
hook_output <- knit_hooks$get("output")
knit_hooks$set(output = function(x, options) {
  lines <- options$output.lines
  if (is.null(lines)) {
    return(hook_output(x, options))  # pass to default hook
  }
  x <- unlist(strsplit(x, "\n"))
  more <- "..."
  if (length(lines)==1) {        # first n lines
    if (length(x) > lines) {
      # truncate the output, but add ....
      x <- c(head(x, lines), more)
    }
  } else {
    x <- c(more, x[lines], more)
  }
  # paste these lines together
  x <- paste(c(x, ""), collapse = "\n")
  hook_output(x, options)
})
```

Phase A

1) Describe the business model for online lending platforms like Lending Club. Consider the stakeholders and their roles, and what advantages Lending Club offers. What is the attraction for investors? How does the platform make money? (Not more than 1.5 pages, single spaced, 11 pt font. Please cite your sources).

If you get a loan from a bank, the bank will use some of its assets, which are securities made through accounts by other customers, to finance the loan. In peer lending, lenders are indirectly connected compared investors via a lending digital medium. Investors get to know and choose exactly what kind of loans they want to finance/invest. P2P loans in general are personal loans or small business loans.

P2P lending via digital medium is attractive to investors because it offers better interest on their investments and also make it their personal choice in what to invest into.

Market lenders make money by lending money to borrowers and taking a percentage of the interest earned on the loan. Usually, lenders will charge a start-up fee, usually 1% to 7% of the total loan amount, and late payments to lenders. On the investment side, lenders will take a percentage of the interest earned on the loan. Hence, this way they make money and run a big profitable business.

2) Data exploration

(a) Some questions to consider:

(i) What is the proportion of defaults ('charged off' vs 'fully paid' loans) in the data? How does default rate vary with loan grade? Does it vary with sub-grade? And is this what you would expect, and why?

```r
data <- read.csv("lcDataSample5m.csv")
tbl <- table(data$loan_status)
res <- cbind(tbl,round(prop.table(tbl)*100,2))
colnames(res) <- c("status","grade")
res
```

```
##             status grade
## Charged Off  11827  14.6
## Current          1   0.0
## Fully Paid   69195  85.4
```

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
df <- data[,c("loan_status","grade")]
df2 <- add_count(df, loan_status, grade)
colnames(df2) <- c("status","grade","total count")
df2 <- df2[!duplicated(t(apply(df2, 1, sort))),]
df2
```

```
##           status grade total count
## 1      Fully Paid     C       18461
## 3      Fully Paid     A       19294
## 4      Fully Paid     B       20717
## 8     Charged Off     B        2682
## 10    Charged Off     C        4116
## 15    Charged Off     D        2647
## 18     Fully Paid     D        8155
## 23    Charged Off     E        1045
```

```
## 31     Charged Off      A         1108
## 42      Fully Paid      E         2146
## 424     Fully Paid      F          369
## 805    Charged Off      F          191
## 1796   Charged Off      G           38
## 1987    Fully Paid      G           53
## 15013      Current      C            1
```

The proportion is roughly around 85% for fully paid and 15% for charged off. The status does vary with the grade and sub-grade. Grades and sub grades with lower values have a high proportion of loans getting charged off in comparison to the higher values in a good first glance. And, yes we expect this because this is the very reason why the grades were given in the first place.

(ii) How many loans are there in each grade? And do loan amounts vary by grade? Does interest rate for loans vary with grade, subgrade? Look at the average, standard-deviation, min and max of interest rate by grade and subgrade. Is this what you expect, and why?

```
df <- data %>% add_count(grade)
df2 <- df[,c("grade","n")]
df2 <- df2[!duplicated(t(apply(df2, 1, sort))),]
df2
```

```
##       grade     n
## 1         C 22578
## 3         A 20402
## 4         B 23399
## 15        D 10802
## 23        E  3191
## 424       F   560
## 1796      G    91
```

As you can see via the output of chunk3 all grade values have certain frequency with "C" grade having the largest value and "G" being the lowest.

```
library(dplyr)
df <- data[,c("loan_amnt","grade")]
df <- df %>% group_by(grade) %>% summarise(median_loan_amnt = median(loan_amnt), mean_loan_amnt = mean(l
df
```

```
## # A tibble: 7 x 3
##   grade median_loan_amnt mean_loan_amnt
## * <chr>            <dbl>          <dbl>
## 1 A               12000          14146.
## 2 B               10000          12458.
## 3 C                9800          11466.
## 4 D                9650          12150.
## 5 E                9775          12558.
## 6 F                8162.         10169.
## 7 G               10000          12509.
```

The loan amount doesn't vary dpending on the grade. Only a slight variation is visible.

```r
library(dplyr)
df <- data[,c("grade","int_rate")]
df_A <- df[df$grade == "A", "int_rate"]
print("For grade A")
```

```
## [1] "For grade A"
```

```r
summary(df_A)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    6.00    6.49    7.49    7.25    8.19    8.39
```

```r
df_B <- df[df$grade == "B", "int_rate"]
print("For grade B")
```

```
## [1] "For grade B"
```

```r
summary(df_B)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    6.00    9.49   10.99   10.71   11.67   12.49
```

```r
df_C <- df[df$grade == "C", "int_rate"]
print("For grade C")
```

```
## [1] "For grade C"
```

```r
summary(df_C)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    6.00   12.99   13.66   13.67   14.31   14.99
```

```r
df_D <- df[df$grade == "D", "int_rate"]
print("For grade D")
```

```
## [1] "For grade D"
```

```r
summary(df_D)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    6.00   15.61   16.29   16.53   17.14   18.24
```

```r
df_E <- df[df$grade == "E", "int_rate"]
print("For grade E")
```

```
## [1] "For grade E"
```

```r
summary(df_E)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    6.00   18.99   19.52   19.77   20.20   22.15
```

```r
df_F <- df[df$grade == "F", "int_rate"]
print("For grade F")
```

```
## [1] "For grade F"
```

```r
summary(df_F)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   22.99   23.43   24.08   24.12   24.50   25.57
```

```r
df_G <- df[df$grade == "G", "int_rate"]
print("For grade G")
```

```
## [1] "For grade G"
```

```r
summary(df_G)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   25.80   25.80   25.83   25.84   25.83   26.06
```

The loan interest rate increases with the value of your grade. Lower the grade higher the interest. Yes, this was expected because that's how these lending club works by charging higher interest rates to more unsecured loans.

(iii)What are people borrowing money for (purpose)? Examine how many loans, average amounts, etc. by purpose? And within grade? Do defaults vary by purpose?

```r
df <- data[,c("purpose")]
df <- unique(df)
print("Different reasons people are borrowing money for as follows:-")
```

```
## [1] "Different reasons people are borrowing money for as follows:-"
```

```r
df
```

```
##  [1] "debt_consolidation" "medical"            "credit_card"
##  [4] "home_improvement"   "moving"             "major_purchase"
##  [7] "other"              "vacation"           "small_business"
## [10] "car"                "renewable_energy"   "house"
## [13] "wedding"
```

```
#code to find sum
df <- data[,c("purpose","loan_amnt")]
df <- df%>% add_count(purpose) %>% group_by(purpose) %>% do({
        sum_value = sum(distinct(., purpose, loan_amnt)$loan_amnt);
        mutate(., sum_value = sum_value)
    })
df2 <- df[,c("purpose","n","sum_value")]
df2 <- df2[!duplicated(t(apply(df2, 1, sort))),]
df2 <- transform(df2, avg = sum_value / n)
df2
```

```
##               purpose     n sum_value       avg
## 1                 car   719   1702500 2367.8720
## 2         credit_card 18780  13998050  745.3701
## 3  debt_consolidation 48647  18427900  378.8086
## 4    home_improvement  3942   5924500 1502.9173
## 5               house   254   1488125 5858.7598
## 6      major_purchase  1402   2925225 2086.4658
## 7             medical   900   1663250 1848.0556
## 8              moving   604    954100 1579.6358
## 9               other  4455   5888350 1321.7396
## 10   renewable_energy    65    402175 6187.3077
## 11     small_business   760   3345925 4402.5329
## 12           vacation   492    794825 1615.4980
## 13            wedding     3     23600 7866.6667
```

```
library(dplyr)
df <- data[,c("loan_status","purpose")]
df2 <- df %>% group_by(loan_status) %>% add_count(loan_status, purpose)
df2 <- df2[!duplicated(t(apply(df2, 1, sort))),]
df2
```

```
## # A tibble: 26 x 3
## # Groups:   loan_status [3]
##    loan_status purpose                n
##    <chr>       <chr>              <int>
##  1 Fully Paid  debt_consolidation 41224
##  2 Fully Paid  medical              759
##  3 Fully Paid  credit_card        16454
##  4 Charged Off credit_card         2326
##  5 Charged Off home_improvement     503
##  6 Charged Off debt_consolidation  7423
##  7 Fully Paid  moving               472
##  8 Charged Off medical              141
##  9 Charged Off major_purchase       220
## 10 Fully Paid  other               3753
## # ... with 16 more rows
```

People are borrowing money for various different reasons as it can be seen in chunk6 output. For each specific purpose all the amount loaned out is shown in the output. And as we can see from the last output of the chunk6 the default number varies in accordance with that of the purpose.

For loans which are fully paid back, how does the time-to-full-payoff vary? For this, calculate the 'actual term' (issue-date to last-payment-date) for all loans. How does this actual-term vary by loan grade (a box-plot can help visualize this).

```
df <- data[,c("loan_status","issue_d","grade")]
library(xts)
```

```
## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

##
## Attaching package: 'xts'

## The following objects are masked from 'package:dplyr':
##
##     first, last
```

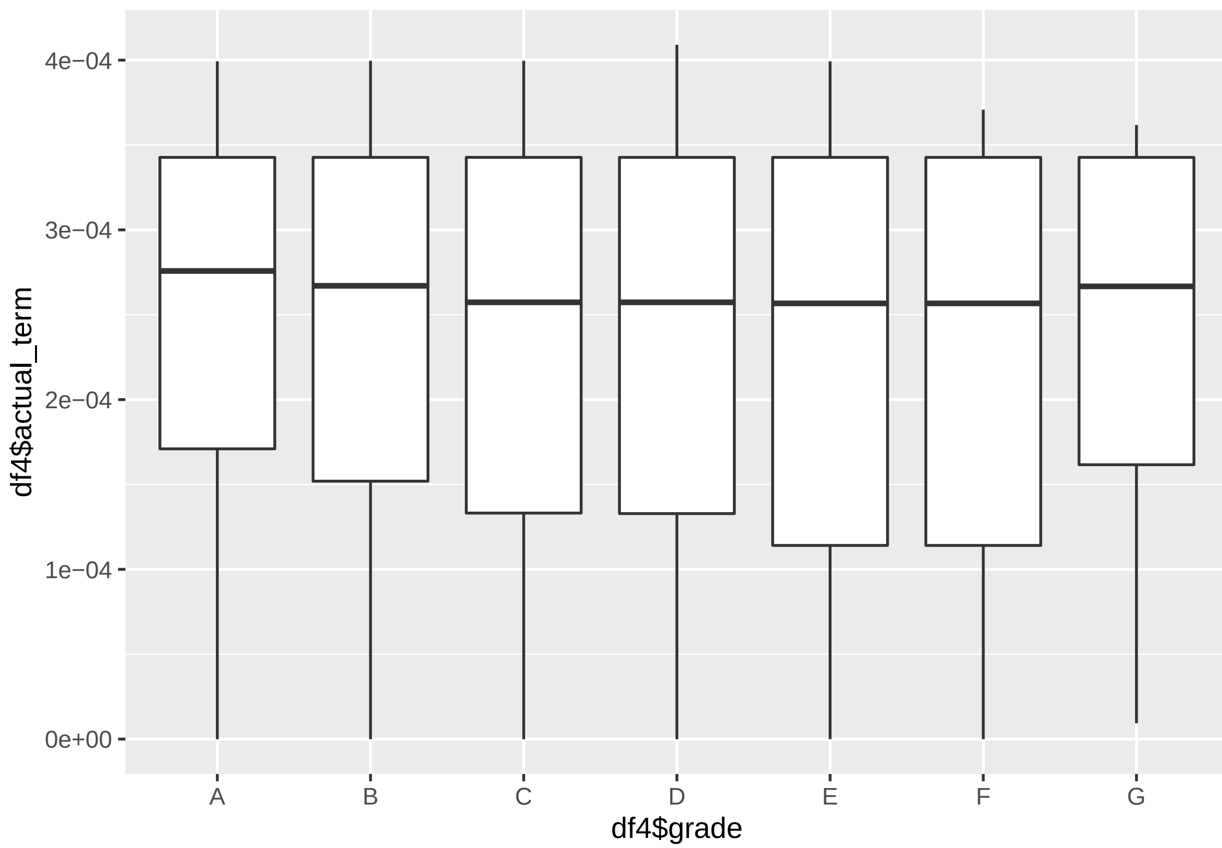
```
#convert month-year to year-month-day format
df2 <- as.Date(as.yearmon(paste0('01-',data$last_pymnt_d), "%d-%b-%Y"))
df4 <- cbind(df,df2)

#calculate the difference in dates to find time taken.
colnames(df4)[4] <- "last_pymnt_d"
df4 <- cbind(df4,as.POSIXct(df4$last_pymnt_d) - as.POSIXct(df4$issue_d))
colnames(df4)[5] <- "actual_term"
df4$actual_term = df4$actual_term/(24*365)
df5<-df4
#convert hours to days
library(dplyr)
df4 <- df4 %>% subset(df$loan_status == "Fully Paid")

#building the box plot
df4 <- df4[c("actual_term","grade")]
df4$actual_term = df4$actual_term/(24*365)
library(ggplot2)
ggplot(stack(df4), aes(x = df4$grade, y = df4$actual_term)) + geom_boxplot()
```

```
## Warning in stack.data.frame(df4): non-vector columns will be ignored

## Don't know how to automatically pick scale for object of type difftime. Defaulting to continuous.
```



For the loans fully paid back the time to pay doesn't vary much to considerable difference. As you can see from the output of the box-plot for all the different grades there's not much difference considering the total time taken to repay the loan. In between grades have a slightly higher time taken to repay if you look at the extremes but other than that it's mostly the same for all the particulars.

(v) Calculate the annual return. Show how you calculate the percentage annual return. Is there any return from loans which are 'charged off'? Explain. How does return from charged - off loans vary by loan grade? Compare the average return values with the average interest_rate on loans – do you notice any differences, and how do you explain this? How do returns vary by grade, and by sub-grade. If you wanted to invest in loans based on this data exploration, which loans would you invest in?

```
df <- data
df6 <- as.numeric(df5$actual_term)
df$annRet <- ifelse(df6>0, ((df$total_pymnt -df$funded_amnt)/df$funded_amnt)*(1/df6),0)

df$perc_rt = (df$annRet/df$funded_amnt) * 100
df2 <- df %>% group_by(grade) %>% summarise(nLoans=n(), defaults=sum(loan_status=="Charged Off"), avgIn
df2
```

```
## # A tibble: 7 x 11
##   grade nLoans defaults avgInterest stdInterest avgLoanAMt avgPmnt avgRet stdRet
## * <chr>  <int>    <int>       <dbl>       <dbl>      <dbl>   <dbl>  <dbl>  <dbl>
## 1 A      20402     1108        7.25       0.796     14146.  15188. NA      NA
## 2 B      23399     2682       10.7        1.22      12458.  13549. NA      NA
```

```
## 3 C      22578     4116       13.7        0.850      11466.  12364. NA      NA
## 4 D      10802     2647       16.5        0.895      12150.  12957. NA      NA
## 5 E       3191     1045       19.8        1.10       12558.  13079. NA      NA
## 6 F        560      191       24.1        0.798      10169.  10588. NA      NA
## 7 G         91       38       25.8        0.0593     12509.  13576. -0.201   1.30
## # ... with 2 more variables: minRet <dbl>, maxRet <dbl>
```

The new column named "perc_rt" gives us the value of annualized percentage returns. So, basically what we do is divide the anuual return with the funded amount and multiply it with 100. The average return is higher for grade A,B and at the tail with G. The average return stays a bit unaffected with high increase in interest rate even though if increased not too much there's decrease in return. It can be explained by the lesser number of loans given out for E and F grades with also considerably lesser amount. Solely based on this I will invest in the grade B with highest annual returns, sufficient number of loans issued and a average to less defaults by proportion.

Generate some (at least 3) new derived attributes which you think may be useful for predicting default., and explain what these are.

```r
df<-data
df$total_intrest <- df$loan_amnt*df$int_rate*3
df$total_uti <- df$revol_bal*df$revol_util
df$total_rec_rate <- df$total_rec_late_fee/df$recoveries
df
```

```
##        X id member_id loan_amnt funded_amnt funded_amnt_inv      term int_rate
## 1      1 NA        NA      5000        5000            5000 36 months    12.39
## 2      2 NA        NA     17000       17000           17000 36 months    12.39
## 3      3 NA        NA      3500        3500            3500 36 months     7.49
## 4      4 NA        NA     14000       14000           14000 36 months    11.99
## 5      5 NA        NA      1400        1400            1400 36 months    12.99
## 6      6 NA        NA      9000        9000            9000 36 months    14.31
## 7      7 NA        NA     19200       19200           19200 36 months    10.49
## 8      8 NA        NA      3600        3600            3600 36 months    11.44
## 9      9 NA        NA     12000       12000           12000 36 months    11.99
## 10    10 NA        NA      9000        9000            9000 36 months    14.99
## 11    11 NA        NA      3000        3000            3000 36 months    12.99
## 12    12 NA        NA     14000       14000           14000 36 months    14.31
## 13    13 NA        NA      9500        9500            9500 36 months    12.99
## 14    14 NA        NA     11000       11000           11000 36 months    14.99
## 15    15 NA        NA     10125       10125           10125 36 months    15.59
## 16    16 NA        NA      3500        3500            3500 36 months    14.31
## 17    17 NA        NA     24000       24000           23900 36 months     9.49
## 18    18 NA        NA      8000        8000            8000 36 months    15.59
## 19    19 NA        NA     20000       20000           20000 36 months     9.49
## 20    20 NA        NA      4125        4125            4125 36 months    13.66
## 21    21 NA        NA      6400        6400            6400 36 months    12.39
## 22    22 NA        NA     24000       24000           23900 36 months     9.49
## 23    23 NA        NA      9450        9450            9450 36 months    20.99
## 24    24 NA        NA     10000       10000           10000 36 months    12.99
## 25    25 NA        NA      3200        3200            3200 36 months    11.99
## 26    26 NA        NA      7000        7000            7000 36 months    13.66
## 27    27 NA        NA      4000        4000            4000 36 months    13.66
```

```
## 28   28 NA         NA      5750         5750             5750 36 months     14.99
## 29   29 NA         NA     12500        12500            12500 36 months      7.49
...
```

The first attribute is the total interest someone has to pay regardless of the fact that they be paying it or not. Based on this you can guess about the default rate. The second attribute talks about the total balance someone is using from the available one which can give you information about the money they are in need of which can be a better indicator if they have a default rather than they amount they loaned. The third attribute talks about the late fee rate which can help you the guess if they will be able to pay off the loan or not.

(b) Are there missing values? What is the proportion of missing values in different variables? Explain how you will handle missing values for different variables. You should consider what is the variable is about, and what missing values may arise from – for example, a variable monthsSinceLastDeliquency may have no value for someone who has not yet had a delinquency; what is a sensible value to replace the missing values in this case? Are there some variables you will exclude from your model due to missing values?

```r
library(dplyr)
df <- data
#remove entirely empty columns
df <- df %>% select_if(function(x){!all(is.na(x))})
df <- df[colSums(is.na(df))>0]
#For proportion of missing values in a column with missing values(In percentage, higher means more null
nm <- colMeans(is.na(df))>0.7
summary(df)
```

```
##   emp_title          mths_since_last_delinq mths_since_last_record
##  Length:81023       Min.   :  0.00         Min.   :  0.00
##  Class :character   1st Qu.: 15.00         1st Qu.: 50.00
##  Mode  :character   Median : 30.00         Median : 66.00
##                     Mean   : 33.64         Mean   : 68.15
##                     3rd Qu.: 49.00         3rd Qu.: 86.00
##                     Max.   :133.00         Max.   :120.00
##                     NA's   :38752          NA's   :66265
##    revol_util     last_pymnt_d       next_pymnt_d       last_credit_pull_d
##  Min.   :  0.00   Length:81023       Length:81023       Length:81023
##  1st Qu.: 36.50   Class :character   Class :character   Class :character
##  Median : 54.40   Mode  :character   Mode  :character   Mode  :character
##  Mean   : 54.22
##  3rd Qu.: 72.20
##  Max.   :184.60
##  NA's   :34
##  mths_since_last_major_derog bc_open_to_buy      bc_util
##  Min.   :  0.0               Min.   :     0   Min.   :  0.00
##  1st Qu.: 26.0               1st Qu.:  1025   1st Qu.: 43.30
##  Median : 43.0               Median :  3656   Median : 67.30
##  Mean   : 42.9               Mean   :  8854   Mean   : 63.46
##  3rd Qu.: 59.0               3rd Qu.: 10377   3rd Qu.: 87.40
##  Max.   :150.0               Max.   :264424   Max.   :318.20
##  NA's   :56987               NA's   :977      NA's   :1028
##  mo_sin_old_il_acct mths_since_recent_bc mths_since_recent_bc_dlq
```

```
##  Min.   :  1.0      Min.   :  0.00       Min.   :  0.00
##  1st Qu.: 96.0      1st Qu.:  6.00       1st Qu.: 21.00
##  Median :128.0      Median : 13.00       Median : 39.00
##  Mean   :124.8      Mean   : 23.92       Mean   : 39.96
##  3rd Qu.:152.0      3rd Qu.: 29.00       3rd Qu.: 59.00
...
```

```
name <- names(df)[nm]
df <- df %>% select(-name)
```

```
## Note: Using an external vector in selections is ambiguous.
## i Use `all_of(name)` instead of `name` to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.
```

```
#replacing the missing values
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v tibble  3.0.4     v purrr   0.3.4
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x xts::first()    masks dplyr::first()
## x dplyr::lag()    masks stats::lag()
## x xts::last()     masks dplyr::last()
```

```
df<- df %>% replace_na(list(mths_since_last_delinq=500, revol_util=median(df$revol_util, na.rm=TRUE), b
```

The proportions can be seen via the output of chunk ten. We should remove all the variable columns which have missing variables more than 60-70 percent as that's a lot of missing values hence it would be inappropriate to use the given values to predict the null ones as it might sway our predictions. As far as the missing values go we can replace them appropriate values which can be either the max, median or the mean depending upon it's meaning.

Consider the potential for data leakage. You do not want to include variables in your model which may not be available when applying the model; that is, some data may not be available for new loans before they are funded. Leakage may also arise from variables in the data which may have been updated during the loan period (ie., after the loan is funded). Identify and explain which variables will you exclude from the model.

```
#Drop some other columns which are not useful and those which will cause 'leakage'
df <- data
df <- df %>% select(-c(funded_amnt_inv, term, emp_title, pymnt_plan, title, zip_code, addr_state, out_p

varsToRemove <- c("last_pymnt_d", "last_pymnt_amnt","id","member_id")
df <- df %>% select(-varsToRemove)
```

```
## Note: Using an external vector in selections is ambiguous.
## i Use `all_of(varsToRemove)` instead of `varsToRemove` to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.
```

These variables as seen in chunk 11 are removed as they won't be available to us not before the loan is finished or is in the process so we remove them.

Do a uni-variate analyses to determine which variables (from amongst those you decide to consider for the next stage prediction task) will be individually useful for predicting the dependent variable (loan_status). For this, you need a measure of relationship between the dependent variable and each of the potential predictor variables. Given loan-status as a binary dependent variable, which measure will you use? From your analyses using this measure, which variables do you think will be useful for predicting loan_status? (Note – if certain variables on their own are highly predictive of the outcome, it is good to ask if this variable has a leakage issue).

```
df <- data
#create training set
Trn_frac = 0.75
nr<-nrow(df)
trnIndex<- sample(1:nr, size = round(Trn_frac * nr), replace=FALSE)
dfTrn <- df[trnIndex, ]
dfTst <- df[-trnIndex, ]
dfTrn <- dfTrn [!( grepl("Current", dfTrn$loan_status)) , ]
sum(dfTrn$loan_status=="Current")
```

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

```
#univariate test
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
#delete the current value from loan_status
aucAll<- sapply(dfTrn %>% mutate_if(is.factor, as.numeric) %>% select_if(is.numeric), multiclass.roc, r
```

```
## Setting direction: controls > cases
```

```
## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases

## Setting direction: controls > cases

## Setting direction: controls < cases
## Setting direction: controls < cases

## Setting direction: controls > cases

## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases

## Setting direction: controls > cases
## Setting direction: controls > cases

## Setting direction: controls < cases
## Setting direction: controls < cases

## Setting direction: controls > cases

## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases

## Setting direction: controls > cases
## Setting direction: controls > cases

## Setting direction: controls < cases
## Setting direction: controls < cases

## Setting direction: controls > cases

## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases

## Setting direction: controls > cases

## Setting direction: controls < cases
## Setting direction: controls < cases
```

```
## Setting direction: controls > cases
```

```
## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases
```

```
## Setting direction: controls > cases
```

```
## Setting direction: controls < cases
## Setting direction: controls < cases
```

```
## Setting direction: controls > cases
```

```
## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases
```

```
aucAll
```

```
##           X               loan_amnt       funded_amnt     funded_amnt_inv
## response  Character,60766 Character,60766 Character,60766 Character,60766
## predictor Integer,60766   Integer,60766   Integer,60766   Integer,60766
## percent   FALSE           FALSE           FALSE           FALSE
## levels    Character,2     Character,2     Character,2     Character,2
## rocs      List,1          List,1          List,1          List,1
## auc       0.5034194       0.5137309       0.5137309       0.5136852
## call      Expression      Expression      Expression      Expression
##           int_rate        installment     annual_inc      dti
## response  Character,60766 Character,60766 Character,60766 Character,60766
```

```
## predictor Numeric,60766   Numeric,60766   Numeric,60766   Numeric,60766
## percent   FALSE           FALSE           FALSE           FALSE
## levels    Character,2     Character,2     Character,2     Character,2
## rocs      List,1          List,1          List,1          List,1
## auc       0.6732648       0.4984402       0.5753937       0.5738335
## call      Expression      Expression      Expression      Expression
##           delinq_2yrs     inq_last_6mths  mths_since_last_delinq
## response  Character,60766 Character,60766 Character,60766
## predictor Integer,60766   Integer,60766   Integer,60766
## percent   FALSE           FALSE           FALSE
## levels    Character,2     Character,2     Character,2
## rocs      List,1          List,1          List,1
## auc       0.4955136       0.4534234       0.5041433
## call      Expression      Expression      Expression
##           mths_since_last_record open_acc        pub_rec
## response  Character,60766        Character,60766 Character,60766
## predictor Integer,60766          Integer,60766   Integer,60766
## percent   FALSE                  FALSE           FALSE
## levels    Character,2            Character,2     Character,2
## rocs      List,1                 List,1          List,1
...
```

We will use variables whose auc value is more than 0.5 so they will be able to give good prediction