SDS-M1-Group-project

Emma, Michael, Signe and Mathias

26. september 2019

```
### Knitr options
knitr::opts_chunk$set(warning=FALSE,
                      message=FALSE,
                      fig.align="center"
                      )

options(warn=-1) # Hides all warnings, as the knitr options only work on local R-Markdown mode.
```

Links:

Link for github: https://github.com/DataEconomistDK/SDS-M1-LendingClub Link for google colab: https://colab.research.google.com/drive/1ElhHlPrcmvlu52jbPgiSRg8CxAA7n-dP

1. Introduction

This project will look into some interesting data from Lending Club. They are an online peer to peer platform that match lenders with investors for small private and business loans.

This project will analyze data from their platform, to predict which lenders who will not be able to pay their full loan in due time defined here as a "bad loan". So this is esssentially a risk analysis. Lending Club also measure the expected risk for each loan and grade them which they publish in their dataset, but this will not be used for this project, as we want to make our own prediction system.

First data will be imported, cleaned and then an explorative data analysis will be made. This will give some insight of how different variables can predict if a loan will be bad. Afterwards unsupervised ML will give more insights into how the data are grouped, which will then be used for superved ML where prediction will be made of different variables. In the end the performance will be evaluated.

1.1 Description of data acquisition

The data used in this project is downloaded from kaggle.com provided by Lending Club. The files contain complete loan data for all loans issued through 2007-2018. There is around 150 variables and 2.26 mio. observations. There is variables on credit scores, number of finance inquiries, address including zip codes and state among others. Data dictionary were provided in a separate file explaining in more detail the meaning of each variable, and when lacking in detail info have been collected from their own webpage or kaggle kernels. As the dataset is very big (around 1.2 GB), very complex and as not everything is usefull or needed filtering have to be done. There is both a practical problem of having big files like this but also a computational problem when running analysis on so much data. So we will only select the most key variables and perform the analysis on these.

The loans can have the following different statuses (Current, Late, Fully Paid, etc.), and as this project will work with supervised ML we need true labels. Therefore this project will discard loans that are current, as these can at some point later become a bad loan and we therefore do not know the true label yet. Also we will only select a few key variables to predict if a loan will be bad. After doing this the data is now only

around 200 MB. This is the data that we will share through a link for loading in the data. A link to the full dataset and code for selecting our subset will also be provided in the "Data preparation" section, but is not run, but availabe if you want to test it out.

2. Setting up the data and packages

First we are installing package

```
rm(list=ls()) # Deletes global evironment
library(tidyverse)
library(lubridate)
library(readr)
library(plyr)
library(dplyr)
library(dummies)

#Data vizualizations
library(FactoMineR) #multivariate Exploratory Data Analysis
library(factoextra) #Extract and Visualize the Results of Multivariate Data Analyses
library(GGally) #extends 'ggplot2'
library(VIM)
library(mice)
library(reshape2)

#Extra
library(ggforce)
library(caret)
library(yardstick)
library(knitr)
library(ggthemes)
library(ggridges)
library(kableExtra)
library(scales)
library(magrittr)
library(rmarkdown)
library(recipes)
library(ranger)
library(rpart)
```

Now we load the data through a dropbox share link. This is the subset which is about 200 MB, where only 10 variables are selected before any data manipulation and cleaning. The total data set can be found on: https://www.kaggle.com/wendykan/lending-club-loan-data, if you want to control. You just need to download the loan.csv file and select the variables we have in this analysis.

```
#loading data
data <- read_csv("https://www.dropbox.com/s/qj57753efkvcd4b/TilpassetData.csv?dl=1")
data.backup <- data
```

3. Data preprocessing

Hereafter we filter on loan status, so that we only have loans that are either paid back fully (good loans) or that is charged off in some way (bad loans). In this way there is no current loans, so we have a label for the

supervised ML. Afterwards we filter on home ownership, so we only have the "strings" that makes sense, and not values like other, don't know ect. We also filter for annual income, so that we only have observation with less than 3 mio. dollars in income. In this case this is only 1 outlier with a income of 6.1 mio. dollars in income that we exclude, as it distorts the data. Then we remove about 50.000 NA observations which is mostly the data from 2012 and earlier. At final we only select observations that have been issued in 2013 and then we have the final dataset.

```
#Filtering the data

data1 <- data.backup %>%
  filter(loan_status %in% c("Charged Off", "Default", "In Grace Period", "Late (16-30 days)", "Late (31
  filter(home_ownership %in% c("RENT", "OWN", "MORTGAGE")) %>% #Filtering for "ANY", "NONE" and "OTHER"
  filter(annual_inc <= 3000000) #Filtering for an annual income over 3 million dollars

data2 <- na.omit(data1) #removing NA's

data <- data2 %>%
  mutate(year = str_detect(issue_d, pattern = "2013")) %>%  #We are only looking at the year 2013
  filter(year == TRUE) %>%
  select(c(-year, -issue_d)) %>% #removing years and issue date
  as.tibble()
```

4. Exploring the data

4.1 Total data set

First, we will explore the total data set.

```
#Dimensions of the data
dim(data)
```

```
## [1] 134804      9
```

Here we can see that the data now contains 9 variables and 134.804 observations after cleaning.

Here we can see the 9 variables we have choosen out of 145 variables in the original dataframe. Here the variables are purpose, term, loan status, loan amount, interest rate, annual income, employment length, home ownership, total bankcard limit. We will analyze the variables individually before continuing with exploring unsupervised and supervised machine learning for the data.

Below you can see an example of one observation.

```
head(data, 1)
```

```
## # A tibble: 1 x 9
##   purpose term  loan_status loan_amnt int_rate annual_inc emp_length
##   <chr>   <chr> <chr>           <dbl>    <dbl>      <dbl> <chr>
## 1 debt_c~ 36 m~ Fully Paid      12000     6.62     105000 10+ years
## # ... with 2 more variables: home_ownership <chr>, total_bc_limit <dbl>
```

4.2. Purpose

The variable purpose indicates which purpose the borrower is lending for. This variable will be used to show which kind of loan is riskier than others. The categories are in the variable is: Credit card, debt consolidation, house, car, vacation, home improvement, small business, major purchase, medical, moving, renewable energy

and wedding. Here we can see that the purpose for taking a loan from Lending Club is first and foremost debt consolidation (80.000), second credit card (30.000), third home improvement (7400) and then the rest being significantly less.

```
table(data$purpose)
```

```
##
##                car        credit_card debt_consolidation
##               1050              32804              80632
##   home_improvement              house     major_purchase
##               7403                675               2299
##            medical             moving              other
##                889                639               5843
##   renewable_energy     small_business           vacation
##                 51               1359                565
##            wedding
##                595
```

4.3. Term

The variable term informs about the length of the loan. Term is divided in two numerical values, which is either 36 months (three years) or 60 months (five years). The purpose with this variable, is to check if the length of the loan is affecting on the chance to repay back the loan.

```
table(data$term)
```

```
##
## 36 months 60 months
##    100421     34383
```

Here we can see the two different kind of loans, here 36 and 60 months, where 100.000 have a 36 months loan and 34.000 have a 60 months loan. We will now transform this variable into a binary variable, where term of 36 months take the value 1 and 60 months take the value 0.

```
data$term = ifelse(data$term == "36 months", 1, 0)
```

4.4. Loan status

The variable loan_status are a categorical variable, with the purpose to inform about the status on the loan by the borrower.

```
table(data$loan_status)
```

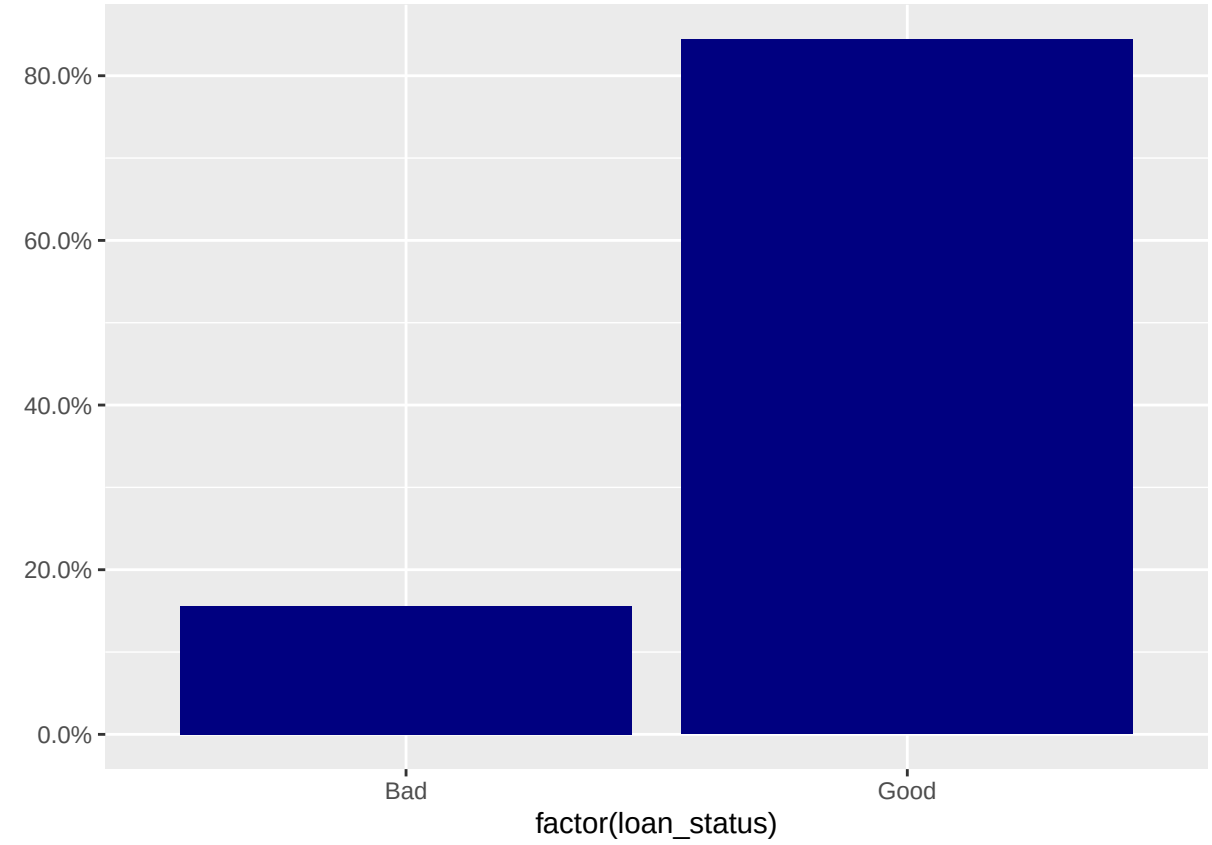
```
##
##        Charged Off         Fully Paid    In Grace Period
##              21022             113770                  2
##  Late (16-30 days) Late (31-120 days)
##                  2                  8
```

In the project, five of the subcategories in the variable will be combined to one, where it will take the value Bad if it includes the following variables: Default, Charged Off, Late (16-30 days), Late(31-120 days) and In Grace Period. It will take the value Good if Fully Paid. Now, we will define a binary variable for loan status.

```
data$loan_status[data$loan_status == "Charged Off"] <- "Bad"
data$loan_status[data$loan_status == "Default"] <- "Bad"
data$loan_status[data$loan_status == "In Grace Period"] <- "Bad"
```

```r
data$loan_status[data$loan_status == "Late (16-30 days)"] <- "Bad"
data$loan_status[data$loan_status == "Late (31-120 days)"] <- "Bad"
data$loan_status[data$loan_status == "Fully Paid"] <- "Good"

ggplot(data, aes(x=factor(loan_status))) +
  geom_bar(aes(y = (..count..)/sum(..count..)), fill="navyblue") +
  scale_y_continuous(labels = scales::percent) +
  ylab("")
```



4.5. Loan amount

The variable loan_amnt determines the size of the loan by the borrower in USD.

```r
summary(data$loan_amnt)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1000    8500   13000   14707   20000   35000
```

Here we can see that the minium loans are for 1000 dollars and the maximum loans are for 35000 dollars. The median is 13000 dollars. This makes sense as Lending Club is for small private and business loans.

4.6. Interest rate

The variable 'int_rate' indicates which interest rate the borrower is going to pay on the loan. Lending Club has a base rate of all borrowers. In addition to this, there will be a percentage be imposed, which are rated on the borrower's adjustment for risk and volatility.

```
summary(data$int_rate)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    6.00   11.14   14.33   14.53   17.56   26.06
```

The variable varies from 6 pct. to 26.06 pct.The median interest rate is 13.36 percent. These are pretty high interest rates, but makes sense since they have way higher risk than normal bank loans. The interest rate are similar to the ones of consumer loans, quick loans, micro loans ect.

4.7. Annual income

The variable 'annual_inc' is describing the yearly income of the borrower, calculated in USD.

```
summary(data$annual_inc)
```

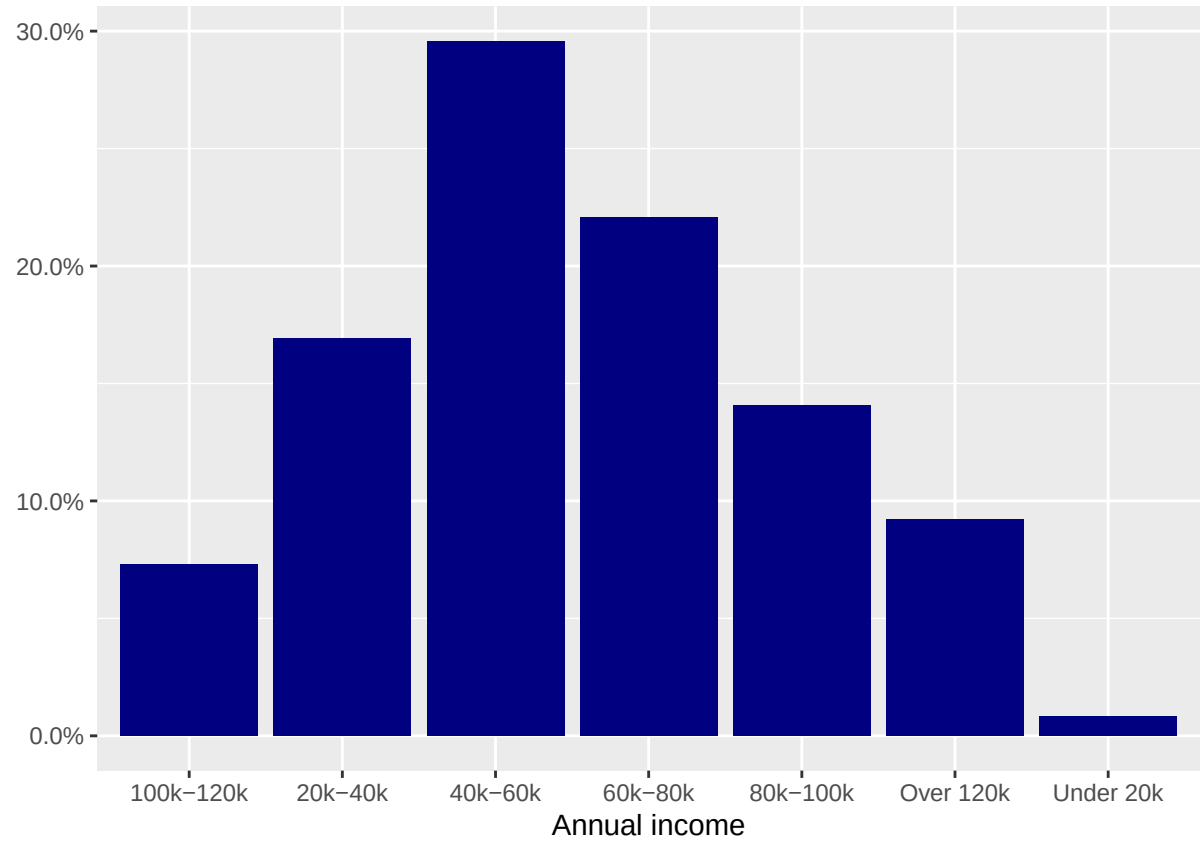
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    6000   45570   64000   73183   89000 2000000
```

There is a big difference between minimum and maximum in annual income. This makes sense as loans can be both for private or business use, and as the purpose can be very different.

Below we define some categories for the income to show in the barplot.

```
data$annual_inc1[data$annual_inc < 20000] = "Under 20k"
data$annual_inc1[data$annual_inc >= 20000 & data$annual_inc <= 40000] = "20k-40k"
data$annual_inc1[data$annual_inc >= 40001 & data$annual_inc <= 60000] = "40k-60k"
data$annual_inc1[data$annual_inc >= 60001 & data$annual_inc <= 80000] = "60k-80k"
data$annual_inc1[data$annual_inc >= 80001 & data$annual_inc <= 100000] = "80k-100k"
data$annual_inc1[data$annual_inc >= 100001 & data$annual_inc <= 120000] = "100k-120k"
data$annual_inc1[data$annual_inc > 120000] = "Over 120k"

ggplot(data, aes(x=factor(annual_inc1))) + geom_bar(aes(y = (..count..)/sum(..count..)), fill="navyblue
  scale_y_continuous(labels = scales::percent) +
  xlab("Annual income") + ylab("")
```




```
data = data %>% select(-annual_inc1)
```

We can see that most loaners are around middle income class.

4.8. Employment length

The variable 'emp_length' indicates the duration of the borrower's employment in years. If the value is given as 'n/a', the individual is currently unemployment. The values between 1 and 9 indicates the years the been the same employment and 10+ indicates if the duration is over ten years.

```
table(data$emp_length)
```

```
##
##  < 1 year    1 year 10+ years   2 years   3 years   4 years   5 years
##      9084      7782     45798     11240     10095      6884      9727
##   6 years   7 years   8 years   9 years       n/a
##      8175      8173      6667      5218      5961
```

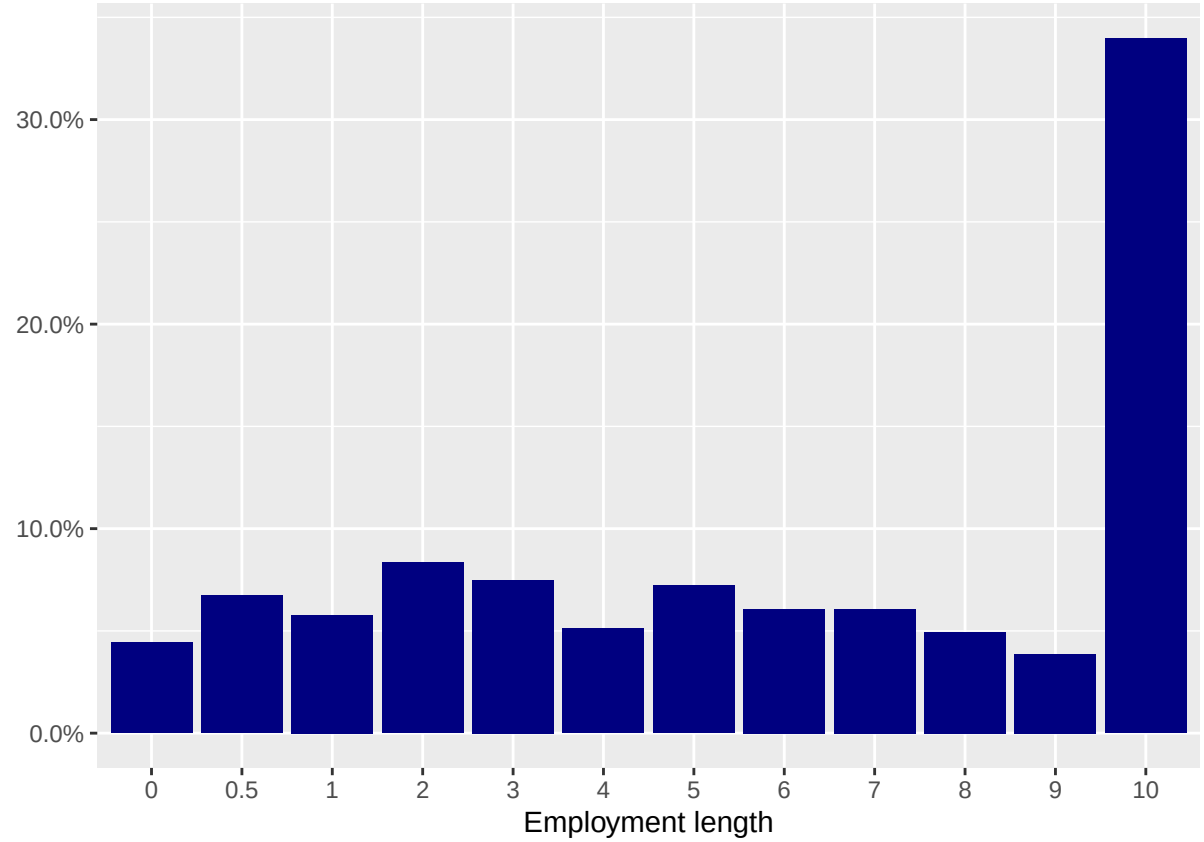
The categorical data is first transformed into a numerican and in the below plot you can see the frequency of the employment length.

```
data$emp_length[data$emp_length == "n/a"] = 0
data$emp_length[data$emp_length == "< 1 year"] = 0.5
data$emp_length[data$emp_length == "1 year"] = 1
data$emp_length[data$emp_length == "2 years"] = 2
data$emp_length[data$emp_length == "3 years"] = 3
data$emp_length[data$emp_length == "4 years"] = 4
data$emp_length[data$emp_length == "5 years"] = 5
```

```
data$emp_length[data$emp_length == "6 years"] = 6
data$emp_length[data$emp_length == "7 years"] = 7
data$emp_length[data$emp_length == "8 years"] = 8
data$emp_length[data$emp_length == "9 years"] = 9
data$emp_length[data$emp_length == "10+ years"] = 10

data$emp_length <- as.numeric(data$emp_length)

ggplot(data, aes(x=factor(emp_length, level=c("0","0.5","1","2","3","4","5","6","7","8","9","10")))) + g
  scale_y_continuous(labels = scales::percent) +
  xlab("Employment length") + ylab("")
```



Most people appear to be in jobs at least a few years, and about a third have been in a jobs in 10 or more years. So there is only very few unemployed people who actually get a loan.

4.9. Home ownership

The 'home_ownership' variable take the following categories: Rent, Own and Mortgage. First, we plot the different home ownership statuses. We have already filtered away any, none and other, so they will not show up.

```
#First a table of the ditribution
table(data$home_ownership)
```

```
##
## MORTGAGE      OWN     RENT
##    72057    11250    51497
```

Here we can see that mortgage has the biggest value (72.000). Rent is second biggest (51.000) and third one (11.000) which is only a little part of the data.

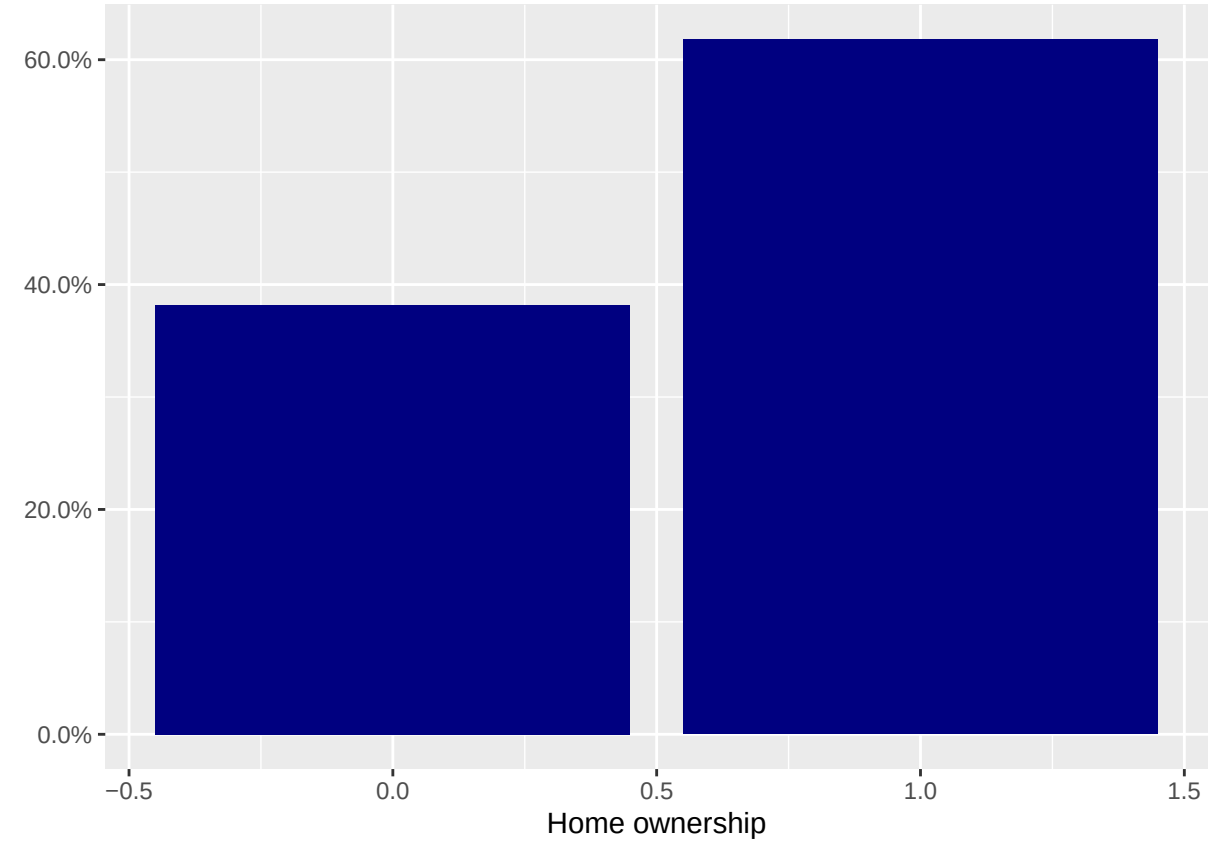
Unfortunately, the explanations for the data set is not very explicit and it is hard to get more information on the variables. Here we run into a problem with how to define home ownership. We want to create a binary variable, but how do you categorize being a home owner? There is three categories left; mortgage, own and rent. Here we categories rent as not a home owner. But what about mortgage? Do you own a home if you are still paying a mortgage. In Denmark you would still say that you own a house even though you have mortgage, so we combine mortage and own into one category. So the new variable is called house and takes the value 1(TRUE) if you have a house and opposite it takes the value 0(FALSE) if you rent.

Below we se the final distribution in the binary variable.

```
#First binary varible
data$house[data$home_ownership == "MORTGAGE"] <- 1
data$house[data$home_ownership == "OWN"] <- 1
data$house[data$home_ownership == "RENT"] <- 0

data = data %>% select(-home_ownership)

#Plot of distribution
ggplot(data, aes(x=house)) + geom_bar(aes(y = (..count..)/sum(..count..)), fill="navyblue") +
   scale_y_continuous(labels = scales::percent) +
  xlab("Home ownership") + ylab("")
```



4.10. Total bankcard limit

The variable 'total_bc_limit' describes the amount of credit the borrower is allowed on the bank account, calculated in USD.

```
summary(data$total_bc_limit)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       0    7941   14800   20453   26800  297800
```

4.11. Variables

The variables selected from the whole dataset for this analysis:

```
# Variables to select
vars <- c("purpose", "term", "issue_d",  "loan_status",
          "loan_amnt", "int_rate", "annual_inc", "emp_length", "home_ownership", "total_bc_limit")
```

The numeric variables after transforming.

```
vars_num <- c("loan_amnt", "int_rate", "annual_inc", "total_bc_limit", "term", "emp_length", "house")
```

Now we can show the same first observation again after being formatted:

```
data %>%
  head(1)
```

```
## # A tibble: 1 x 9
##   purpose  term loan_status loan_amnt int_rate annual_inc emp_length
##   <chr>   <dbl> <chr>           <dbl>    <dbl>      <dbl>      <dbl>
## 1 debt_c~     1 Good            12000     6.62     105000         10
## # ... with 2 more variables: total_bc_limit <dbl>, house <dbl>
```

4.12. Scaling

We now scale all the numerical values so they can be used for further analysis. Below we also test to see if the mean is actually 0 and SD 1, which they are.

```
data.scaled <- data[,vars_num] %>%
  scale()

# Test of scale data
colMeans(data.scaled) #Mean is 0.
```

```
##      loan_amnt       int_rate     annual_inc total_bc_limit           term
##  -7.919123e-17   9.368209e-16  -8.320192e-17  -3.736818e-17  -1.876122e-18
##     emp_length          house
##   3.292653e-17   1.258839e-16
```

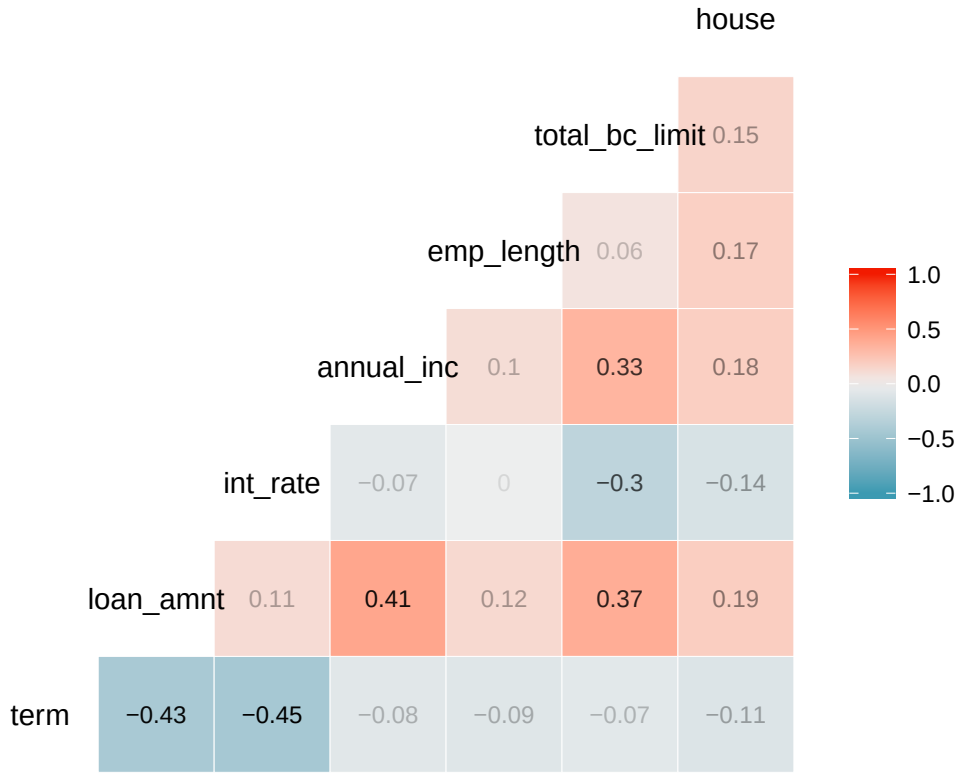
```
apply(data.scaled, 2, sd) #SD is 1.
```

```
##      loan_amnt       int_rate     annual_inc total_bc_limit           term
##              1              1              1              1              1
##     emp_length          house
##              1              1
```

##4.11. Correlation matrix

Here we can see the correlation between the different numeric variables. The correlation goes from -1 to 1, where 1 is a totally positive correlation and -1 is a totally negative correlation.

```
#Correlation
data$emp_length = as.numeric(data$emp_length)
ggcorr(data, label = TRUE, label_size = 3, label_round = 2, label_alpha = TRUE)
```



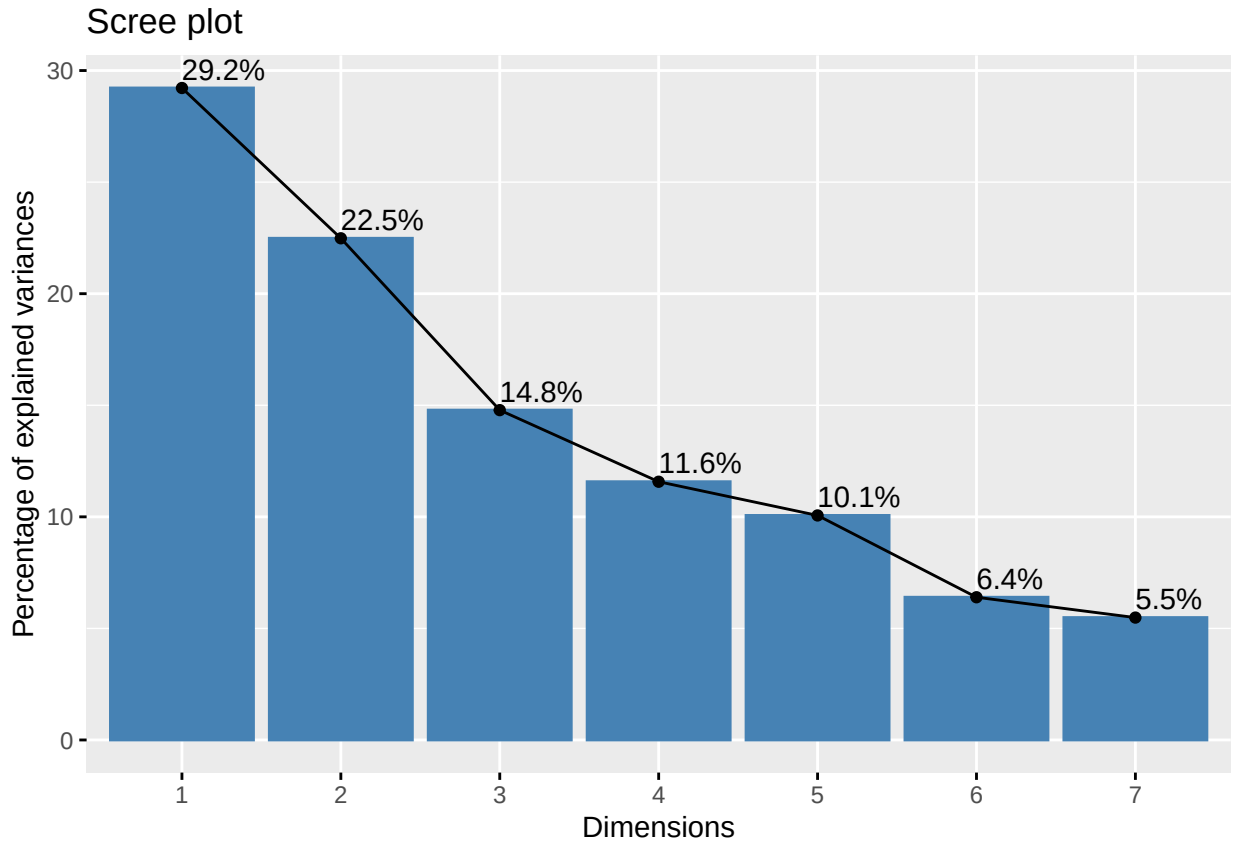
5. PCA

To provide the Principal Component Analysis we'll use, the previously loaded packages FactoMineR to do the job. And the package factoextra and visualizing the results.

```
data.pca <- as.data.frame(data.scaled) %>%
  PCA(scale.unit = TRUE, graph = FALSE)
```

The function 'fviz_screeplot' creates a Scree plot - which is a diagnostic visualization that displays the variance explained by every component.

```
data.pca %>%
  fviz_screeplot(addlabels = TRUE,
                 ncp = 10,
                 ggtheme = theme_gray())
```



As we can see the fist component captures the main variance. This is supported by the corrosponding eigenvalues:

```
data.pca$eig %>%
as_tibble()
```

```
## # A tibble: 7 x 3
##   eigenvalue `percentage of variance` `cumulative percentage of variance`
##        <dbl>                    <dbl>                                <dbl>
## 1      2.05                     29.2                                 29.2
## 2      1.57                     22.5                                 51.7
## 3      1.03                     14.8                                 66.5
## 4      0.810                    11.6                                 78.1
## 5      0.704                    10.1                                 88.1
## 6      0.448                     6.40                                94.5
## 7      0.384                     5.48                               100
```

The rule of thumb is to only include an eigenvalue larger than 1. According to the results this means we can reduce our dataset to 3 dimensions. In the next step we project them onto a 2-dimensional space.

```
data.pca %>% get_pca_var()
```

```
## Principal Component Analysis Results for variables
##  ===================================================
##   Name       Description
## 1 "$coord"   "Coordinates for the variables"
## 2 "$cor"     "Correlations between variables and dimensions"
## 3 "$cos2"    "Cos2 for the variables"
## 4 "$contrib" "contributions of the variables"
```

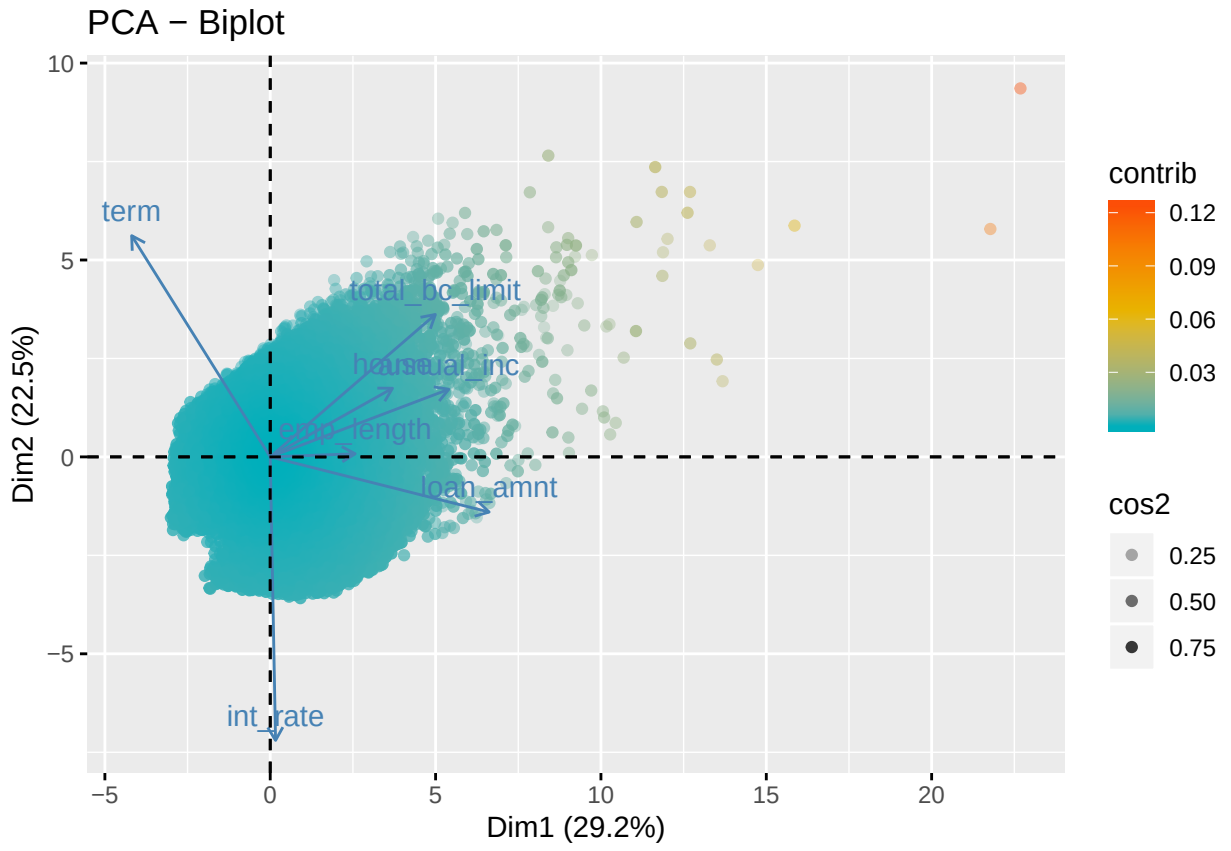
In the next steps we take a closer look at the numeric values:

```
data.pca$var$coord %>% as_tibble() %>%
head()
```

```
## # A tibble: 6 x 5
##     Dim.1    Dim.2    Dim.3  Dim.4   Dim.5
##     <dbl>    <dbl>    <dbl>  <dbl>   <dbl>
## 1  0.813  -0.172   -0.198   0.0416 -0.0511
## 2  0.0197 -0.884   -0.0144  0.0266  0.172
## 3  0.663   0.211   -0.205   0.149   0.637
## 4  0.613   0.444   -0.289   0.113  -0.431
## 5 -0.514   0.690    0.00950 0.129   0.267
## 6  0.315   0.00979  0.775   0.545  -0.0569
```

By the function 'fviz_pca_biplot' we visualize our observations and the variables. By the plots of the observations we can see, there's some outliers in the first quadrant that has impact on the vectors (in the same quadrant)

```
data.pca %>%
  fviz_pca_biplot(alpha.ind = "cos2",
                  col.ind = "contrib",
                  gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
                  geom = "point",
                  ggtheme = theme_gray())
```



From this plot we can see how the variables correlates.

6. Unsupervised machine learning

6.1. K-means clustering

We want to use clustering with different methods to get insights into how the data is clustered. In this project we will use K-means. We want to see if there is some groups that are more likely to have bad loans.

As K-means have a random element when assigning the different datapoints to a center in the first iteration we want to use a seed for reproduciability, so that you get the same results.

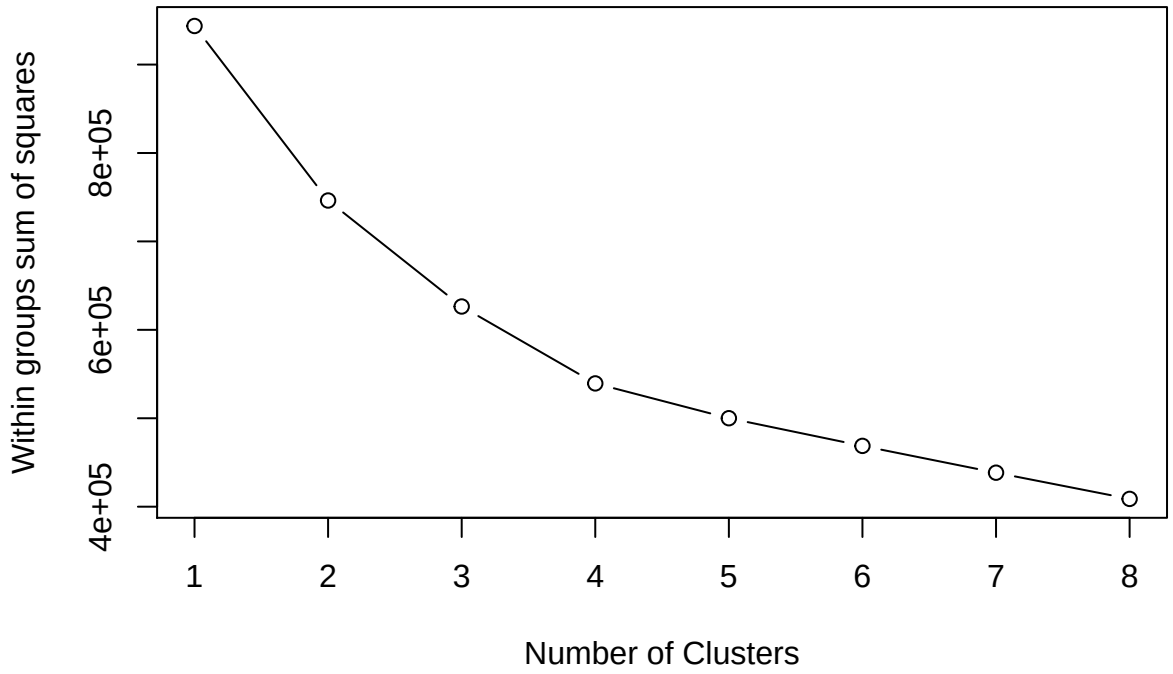
```
set.seed(1)
```

When using K-means we want to figure out the optimal amounts of clusters and this can be done by making a scree plot which plot the "total within sum of squares" (tot.withinss) on the y-axis and the number of clusters on the x-axis. To do so we run a for loop 8 each with the K-means model with the amount of clusters being set from 1 to 8.

```
# Initialize total within sum of squares error: wss
wss <- 0

# For 1 to 15 cluster centers
for (i in 1:8) {
  km.out <- kmeans(data.scaled, centers = i, nstart = 20)
  # Save total within sum of squares to wss variable
  wss[i] <- km.out$tot.withinss
}

# Plot total within sum of squares vs. number of clusters
plot(1:8, wss, type = "b",
     xlab = "Number of Clusters",
     ylab = "Within groups sum of squares")
```



The optimal amount of clusters can be decided by using the elbow rule, where we want to choose the amount of clusters where the line breaks, so that choosing more clusters would not increase the tot.withinss significantly.

In this case we can see that 3-5 clusters would be good, and in this case we go with 4.

We now run our K-means model with 4 clusters.

```
# Set k equal to the number of clusters corresponding to the elbow location
k <- 4

data.scaled.df <- as.data.frame(data.scaled)

# Making the final km model.
km <- data.scaled %>%
  kmeans(centers = 4, nstart = 20)
```

6.2. Summary of clustering

First we want a summary of how the clusters seperate the different variables so below is a summary of the mean of each cluster.

```
table.summary <- data %>%
  bind_cols(cluster = km$cluster) %>%
  select(vars_num, cluster) %>%
  group_by(cluster) %>%
  summarise_all(funs(mean))

table.summary$n <- km$cluster %>%
  count() %>%
  select(freq) %>%
  as_vector()

table.summary
```

```
## # A tibble: 4 x 9
##   cluster loan_amnt int_rate annual_inc total_bc_limit   term emp_length
##     <int>     <dbl>    <dbl>      <dbl>          <dbl>  <dbl>      <dbl>
## 1       1    20668.     18.3     76284.         19825. 0.0123       6.59
## 2       2    11106.     13.2     64578.         14673. 0.999        6.19
## 3       3    10370.     14.6     55546.         13585. 0.993        4.93
## 4       4    22550.     11.0    127853.         51238. 0.882        6.59
## # ... with 2 more variables: house <dbl>, n <int>
```

First we see that the amount of observation in each cluster range from around 18.000 to 46.000, so they are not equal in size, but they are not far from each other. Cluster 1 and 2 have about the same loan amount of around 20.000 dollars, but cluster 2 have loaners with significantly higher income and bank limit and they also have way lower interest rate which makes great sense as they must have lower risk of being a bad loaner. Cluster 1 is the only cluster where term have a value close to 0. As terms of 36 months take the value 1 and 60 months take the value 0, this means that cluster 1 is the only one that have long loans of 60 months.

Cluster 3 and 4 have about the same loan amount of around 10.000 dollars, and are similar in most variables except that cluster 4 have about 1 year longer employment length and then that the clusters are split 100% on the house variable. This means that the primary difference is that cluster 4 have houses, but cluster 3 does not.

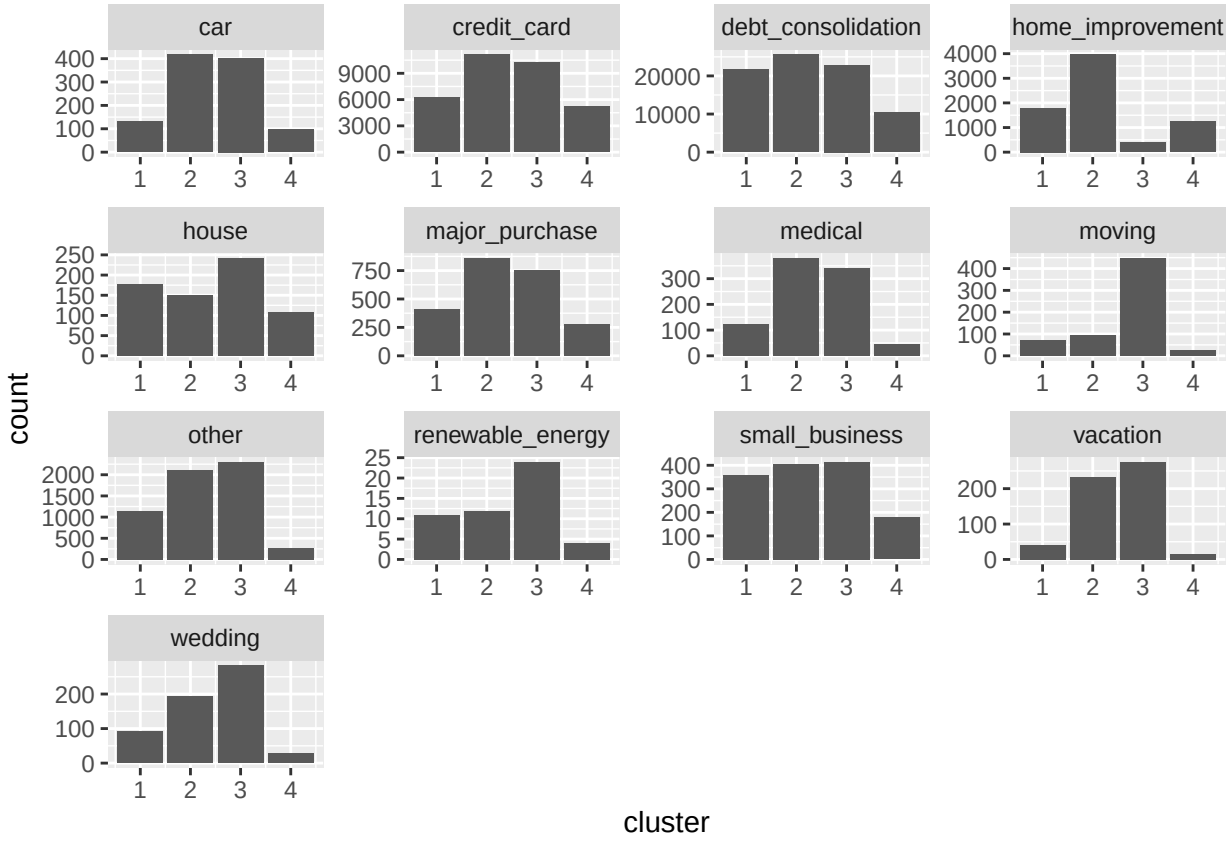
If we also want to get insight into how the clustering seperates some of the categorical variables, f. ex. Does one category have a higher amount of one specifc loan type ect. We can make a barplot, where each bar is a cluster within that category.

6.3. Clustering of purpose

Below we see this for the purpose variable.

```
plotdata <- data %>%
  bind_cols(cluster = km$cluster) %>%
  select(purpose, cluster)

ggplot(data = plotdata, aes(x = cluster)) +
geom_bar() +
facet_wrap(~ purpose, scales = "free")
```



We can here see that there is significant differences between the clusters on the reason why they take a loan. Just remember that cluster 2 is about half size of the others cluster. Cluster 1 and 2 with high loan amounts is not used as often for purposes as vacation, weddings, moving ect. Cluster 3 and 4 are quite similar, but as only cluster 4 have their own houses they also often loan for home improvement. Cluster 3 instead often loans with the purpose of moving.

#7. Supervised Machine Learning First we start by splitting our data in a training and a test sample. To do this we use the package 'caret', which have a feature that takes care of the outcome are proportionally distrubted.

##7.1. Test and train preparation

```
#Transforming data into factor
data$loan_status = as.factor(data$loan_status)

#Splitting into training and test sample, here 75 and 25 percent.
index <- createDataPartition(y=data$loan_status, p = 0.75, list=FALSE)
```

```
training <- data[index,]
test <- data[-index,]
```

Before we can start tuning the models, we are forced to some more preprocessing. To this part we will use the package 'recipe', which provides us with the opportunity define a recipe of ML preprocessing tasks. Afterwards can we use the recipe to 'bake' the data.

We will make a few but simple transformations, such as normalizing all numeric data by centering (subtracting the mean) and scaling (dividing by the standard deviation)

```
#Data is already scaled, so now we prepare traning data
reci = recipe(loan_status ~ ., data = training) %>%
    step_center(all_numeric(), -all_outcomes()) %>% # Centers all numeric variables to mean = 0
    step_scale(all_numeric(), -all_outcomes()) # scales all numeric variables to sd = 1

reci %<>% prep(data = training)

reci
```

```
## Data Recipe
##
## Inputs:
##
##       role #variables
##    outcome          1
##  predictor          8
##
## Training data contained 101104 data points and no missing data.
##
## Operations:
##
## Centering for term, loan_amnt, int_rate, ... [trained]
## Scaling for term, loan_amnt, int_rate, ... [trained]
```

Here there is 1 outcome, here loan status, and eigth predictors.

Now its time to execute the recipe and 'bake' our data. Here x is all our predictors and y is our outcome.

```
# x spit into training and test
x_train <-bake(reci, new_data = training) %>% select(-loan_status)
x_test <- bake(reci, new_data = test) %>% select(-loan_status)

# y spit into training and test
y_train <- pull(training, loan_status) %>% as.factor()
y_test <- pull(test, loan_status) %>% as.factor()
```

Now we will split the data up again by using n-fold crossvalidation. On which we both will fit and test every hyperparameter configurations. This will help us in the way that our result isn't driven by a particular samplin

Now we can define the number of crossvalidations which will be chosen to be 10.

```
#number of crossvalidations
ctrl <- trainControl(method = "cv", # repeatedcv, boot, cv, LOOCV, timeslice OR adaptive etc.
                     number = 10, # Number of CV's
                     classProbs = TRUE, # Include probability of class prediction
                     savePredictions = TRUE, # Save the prediction results
                     summaryFunction = defaultSummary,
```

```
                    verboseIter = FALSE,
                    # add adaptive resampling for hyperparamether search
                    adaptive = list(min = 3,
                                    alpha = 0.05,
                                    method = "gls",
                                    complete = TRUE),
                    search = "random" )


metric = "Accuracy" #
n_tune = 10 # number of tuning rounds
```

Hyperparameter tuning is finally ready to start.

##7.2 Three different algorithms ###7.2.1. Logistic regression model

We will start with using a logistic regression:

```
#Logistic regression model
fit_log <- train(x = x_train,
                 y = y_train,
                 trControl = ctrl,
                 metric = metric,
                 method = "glm", family = "binomial")

fit_log
```

```
## Generalized Linear Model
##
## 101104 samples
##      8 predictor
##      2 classes: 'Bad', 'Good'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 90994, 90993, 90993, 90993, 90993, 90994, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.8437154  0.001246011
```

###7.2.2. Decision tree Secondly, a decision tree. It splits the population or sample into two or more sub-populations.

We here use rpart, but there is a huge variety available. This one has only one tunable parameter. Here we are able to restrict the complexity via a hyperparameter. This parameter represents the complexity costs of every split, and allows further splits only if it leads to an decrease in model loss below this threshold.

```
#Decision tree
fit_dt <- train(x = x_train,
                y = y_train,
                trControl = ctrl,
                metric = metric,
                tuneLength = n_tune,
                method = "rpart")
fit_dt
```

```
## CART
##
## 101104 samples
##      8 predictor
##      2 classes: 'Bad', 'Good'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 90993, 90994, 90993, 90994, 90994, 90993, ...
## Resampling results across tuning parameters:
##
##   cp            Accuracy   Kappa
##   5.282285e-06  0.7997112  0.07372828
##   5.762493e-06  0.7997805  0.07384232
##   2.377028e-05  0.8079897  0.07499563
##   2.535497e-05  0.8079897  0.07499563
##   5.070994e-05  0.8193444  0.07209494
##   5.704868e-05  0.8216391  0.07136284
##   7.606491e-05  0.8293934  0.06076459
##   8.874239e-05  0.8327167  0.05403994
##   1.373394e-04  0.8401250  0.03132317
##   1.648073e-04  0.8423109  0.02219246
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.0001648073.
```

###7.2.3. Random forest Last we will show a random forest, which aims at reducing overfitting by introducing randomness. It is a type of ensemble learning method, where a group of weak models are combined to form a powerful model. The idea is closely linked to the Monte Carlo simulation approaches.

```
#Random forest
fit_rf <- train(x = x_train,
                y = y_train,
                trControl = ctrl,
                metric = metric,
                tuneLength = n_tune,
                method = "ranger",
                importance = "impurity",
                num.trees = 25
                )

fit_rf
```

```
## Random Forest
##
## 101104 samples
##      8 predictor
##      2 classes: 'Bad', 'Good'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 90994, 90993, 90994, 90993, 90994, 90994, ...
## Resampling results across tuning parameters:
##
##   min.node.size  mtry  splitrule   Accuracy   Kappa
```

```
##    3             3     gini        0.8359610  0.04477830
##    5             3     gini        0.8377117  0.03966225
##    7             3     extratrees  0.8419845  0.01855947
##    7             8     gini        0.8339729  0.04475548
##    8             6     gini        0.8351598  0.03972298
##   10             3     gini        0.8396206  0.03379957
##   11             8     gini        0.8355753  0.03861330
##   14             6     extratrees  0.8414009  0.02183476
##   16             5     extratrees  0.8422911  0.01659697
##   17             7     extratrees  0.8417471  0.02014257
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were mtry = 5, splitrule =
##  extratrees and min.node.size = 16.
```

##7.3 Predictions

Now we can make predictions from the models based on the test data.

```
#Predictions based on the test data
pred_log <- predict(fit_log, newdata = x_test)
pred_dt <- predict(fit_dt, newdata = x_test)
pred_rf <- predict(fit_rf, newdata = x_test)
```

We will not print the predicts.

##7.4 Performance evaluation

The confusion matrix is a 2x2 matrix that predict true positives (TP), true negative (TN), false positives (FP) and false negatives (FN). Accuracy is defined as:

$$ACC = (TP + TN)/(TP + TN + FP + FN)$$

Another factor we would like to investigate is Specificity or True Negative Rate, which is defined as:

$$TNR = TN/(TN + FP)$$

```
#Confusion matrix of logistic regression
confusionMatrix(pred_log, y_test, positive="Good")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   Bad  Good
##       Bad      9    13
##       Good  5249 28429
##
##                Accuracy : 0.8439
##                  95% CI : (0.8399, 0.8477)
##     No Information Rate : 0.844
##     P-Value [Acc > NIR] : 0.5276
##
##                   Kappa : 0.0021
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.999543
```

```
##             Specificity : 0.001712
##          Pos Pred Value : 0.844142
##          Neg Pred Value : 0.409091
##              Prevalence : 0.843976
##          Detection Rate : 0.843591
##    Detection Prevalence : 0.999347
##       Balanced Accuracy : 0.500627
##
##        'Positive' Class : Good
##
```

```
#Confusion matrix of decision tree
confusionMatrix(pred_dt, y_test, positive="Good")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   Bad  Good
##       Bad     98   129
##       Good  5160 28313
##
##                Accuracy : 0.8431
##                  95% CI : (0.8391, 0.8469)
##     No Information Rate : 0.844
##     P-Value [Acc > NIR] : 0.6823
##
##                   Kappa : 0.0231
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.99546
##             Specificity : 0.01864
##          Pos Pred Value : 0.84585
##          Neg Pred Value : 0.43172
##              Prevalence : 0.84398
##          Detection Rate : 0.84015
##    Detection Prevalence : 0.99326
##       Balanced Accuracy : 0.50705
##
##        'Positive' Class : Good
##
```

```
#Confusion matrix of random forest
confusionMatrix(pred_rf, y_test, positive="Good")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   Bad  Good
##       Bad     65   112
##       Good  5193 28330
##
##                Accuracy : 0.8426
##                  95% CI : (0.8386, 0.8465)
##     No Information Rate : 0.844
##     P-Value [Acc > NIR] : 0.7624
```

```
##
##                   Kappa : 0.0139
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.99606
##             Specificity : 0.01236
##          Pos Pred Value : 0.84509
##          Neg Pred Value : 0.36723
##              Prevalence : 0.84398
##          Detection Rate : 0.84065
##    Detection Prevalence : 0.99475
##       Balanced Accuracy : 0.50421
##
##        'Positive' Class : Good
##
```

This is some quiet interesting results. The three models perform almost excately alike, with a prediction accuracy around 84 percent, which is alright though not better than predicting all outcomes as Good.

The three models almost all have the same sum of False Negative and False Positive, and all predict som True Negative values. Therefore, we will actually prefer of the models which predict some True Negatives. We would say that it is more important to predict some True Negative and avoid Bad loaners than not predict any at all. Therefore, Speicificity may be better to look at than Accuracy.