Siconv Project Report

Jose Leonardo Ribeiro Nascimento

22/11/2019

1 INTRODUCTION

1.1 Overview

Harvard University offers, through *edx.org platform*, a *Professional Certificate in Data Science*. In this program, it is possible to learn:

- Fundamental R programming skills
- Statistical concepts such as probability, inference, and modeling and how to apply them in practice
- Gain experience with the tidyverse, including data visualization with ggplot2 and data wrangling with dplyr
- Become familiar with essential tools for practicing data scientists such as Unix/Linux, git and GitHub, and RStudio
- Implement machine learning algorithms
- In-depth knowledge of fundamental data science concepts through motivating real-world case studies

The program includes nine courses, beginning with R Basics, visualization and probability, going through inference and modeling, wrangling data, linear regression and culminating with machine learning. As a final course, **Data Science: Capstone** presents two challenges, being the second one the object of this report: **Student is requested to apply machine learning techniques that go beyond standard linear regression, developing an "original project"**. For my project, I decided to choose something related to my working area. I'm an auditor at the Controladoria-Geral da União – CGU, a governmental control agency of the federal executive branch. Among other tasks, I audit federal resources applied through agreements between Federal Government, which owns the resources, and states, municipalities and non-governmental organizations. To celebrate an agreement, since the resources are limited, one must submit a proposal of agreement to the federal government. This **proposal of agreement** will be analyzed and can be approved if considered relevant. Despite my brief explanation, this is not a simple process. Sometimes a proposal takes months to be analyzed and yet it's not approved. Thinking about that, I decide to elaborate an algorithm to predict whether a proposal will be approved or not. If you are interested in an agreement with the federal government, this type of information can be critical, as it will give you guidance on how to draft a proposal that is most likely to pass.

1.2 Siconv database

Siconv is a federal government platform to manage agreements and other voluntary transfers. SICONV provides free access to information on Union Voluntary Transfers in order to facilitate access to system data for society and other spheres of government. Files can be downloaded in csv format through the following link: http://plataformamaisbrasil.gov.br/download-de-dados. After downloading all the files, I examined them and realized the following would be useful to my project:

• *siconv_emenda.csv* – Relation of proposals from parliamentary amendments (My guess was that if a proposal comes from parliamentary amendment has more chance of being approved)

• *siconv_justificativas_proposta.csv* – Relation of proposals and fields for proposal justification and to indicate technical capacity to execute the agreement. (My guess was that filling this fields increases the chance approval).

• *siconv_programa_proposta.csv* – Relation of budget program for each proposal (I believe some programs are more likely to be approved than others).

• *siconv_proposta.csv* – Main file. Relation of proposal and several details about it, some useful for my algorithm, some not.

1.3 Goal

My goal is to develop an algorithm capable of predicting, with an acceptable efficiency, whether a proposal of agreement will be approved. As explained before, to this project, I'm required to apply machine learning techniques that go beyond standard linear regression, developing an "original project". Thinking about that, in my project I decided to test different and more complex models, including generalized linear models, but also classification trees, knn and random forest. I will explain this in detail in the method section.

---

2 METHOD

2.1 Data tidying

My first step is loading and installing needed libraries and importing files I will use to prepare my dataset.

```
if(!require(tidyverse)) install.packages("tidyverse", repos =
"http://cran.us.r-project.org")

## Loading required package: tidyverse
```

```
## -- Attaching packages ---------------------------------------------
-----------------------------------------------------------------------
tidyverse 1.2.1 --

## v ggplot2 3.2.1     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0

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

```
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-
project.org")
```

```
## Loading required package: caret
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
if(!require(data.table)) install.packages("data.table", repos =
"http://cran.us.r-project.org")
```

```
## Loading required package: data.table
```

```
##
## Attaching package: 'data.table'
```

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

```
## The following object is masked from 'package:purrr':
##
##     transpose
```

```
if(!require(randomForest)) install.packages("randomForest", repos =
"http://cran.us.r-project.org")
```

```
## Loading required package: randomForest
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':
##
##     combine

## The following object is masked from 'package:ggplot2':
##
##     margin

if(!require(partykit)) install.packages("partykit", repos =
"http://cran.us.r-project.org")

## Loading required package: partykit

## Loading required package: grid

## Loading required package: libcoin

## Loading required package: mvtnorm

dl1 <- tempfile()
download.file("http://plataformamaisbrasil.gov.br/images/docs/CGSIS/csv/s
iconv_proposta.csv.zip", dl1)

siconv_proposta <- read_delim(unzip(dl1),";", escape_double = FALSE,
trim_ws = TRUE)

## Parsed with column specification:
## cols(
##   .default = col_character(),
##   ID_PROPOSTA = col_double(),
##   COD_MUNIC_IBGE = col_double(),
##   COD_ORGAO_SUP = col_double(),
##   DIA_PROP = col_double(),
##   MES_PROP = col_double(),
##   ANO_PROP = col_double(),
##   COD_ORGAO = col_double(),
##   ITEM_INVESTIMENTO = col_logical(),
##   VL_GLOBAL_PROP = col_number()
## )

## See spec(...) for full column specifications.

## Warning: 102720 parsing failures.
##    row               col           expected
actual                   file
## 617211 ITEM_INVESTIMENTO 1/0/T/F/TRUE/FALSE Equipamentos
'./siconv_proposta.csv'
## 617365 ITEM_INVESTIMENTO 1/0/T/F/TRUE/FALSE Equipamentos
'./siconv_proposta.csv'
```

```
## 617372 ITEM_INVESTIMENTO 1/0/T/F/TRUE/FALSE Equipamentos
'./siconv_proposta.csv'
## 617377 ITEM_INVESTIMENTO 1/0/T/F/TRUE/FALSE Equipamentos
'./siconv_proposta.csv'
## 646488 ITEM_INVESTIMENTO 1/0/T/F/TRUE/FALSE Obras e ServiÃ§os de
Engenharia './siconv_proposta.csv'
## ...... ................. .................
.............................. ......................
## See problems(...) for more details.

dl2 <- tempfile()
download.file("http://plataformamaisbrasil.gov.br/images/docs/CGSIS/csv/s
iconv_programa_proposta.csv.zip", dl2)

siconv_programa_proposta <- read_delim(unzip(dl2),";", escape_double =
FALSE, trim_ws = TRUE)

## Parsed with column specification:
## cols(
##   ID_PROGRAMA = col_double(),
##   ID_PROPOSTA = col_double()
## )

dl3 <- tempfile()
download.file("http://plataformamaisbrasil.gov.br/images/docs/CGSIS/csv/s
iconv_emenda.csv.zip", dl3)

siconv_emenda <- read_delim(unzip(dl3),";", escape_double = FALSE,
trim_ws = TRUE)

## Parsed with column specification:
## cols(
##   ID_PROPOSTA = col_double(),
##   QUALIF_PROPONENTE = col_character(),
##   COD_PROGRAMA_EMENDA = col_double(),
##   NR_EMENDA = col_character(),
##   NOME_PARLAMENTAR = col_character(),
##   BENEFICIARIO_EMENDA = col_character(),
##   IND_IMPOSITIVO = col_character(),
##   TIPO_PARLAMENTAR = col_character(),
##   VALOR_REPASSE_PROPOSTA_EMENDA = col_number(),
##   VALOR_REPASSE_EMENDA = col_number()
## )

dl4 <- tempfile()
download.file("http://plataformamaisbrasil.gov.br/images/docs/CGSIS/csv/s
iconv_justificativas_proposta.csv.zip", dl4)

siconv_justificativas_proposta <- read_delim(unzip(dl4),";",
escape_double = FALSE, trim_ws = TRUE)
```

```
## Parsed with column specification:
## cols(
##   ID_PROPOSTA = col_double(),
##   CARACTERIZACAO_INTERESSES_RECI = col_character(),
##   PUBLICO_ALVO = col_character(),
##   PROBLEMA_A_SER_RESOLVIDO = col_character(),
##   RESULTADOS_ESPERADOS = col_character(),
##   RELACAO_PROPOSTA_OBJETIVOS_PRO = col_character(),
##   CAPACIDADE_TECNICA = col_character()
## )
```

Second step is to prepare my data, a process known as data tyding. As I explained before, I had four datasets with useful information. I had to combine what I really needed in one single file, which would be used along the machine learning process.

I began with siconv_proposta, my main file. It's a huge file, as shown below:

```
dim(siconv_proposta)

## [1] 826322     31
```

I only need 7 of those 31 variables. The column SIT_PROPOSTA has information I want to predict (whether a proposal was approved). However, there are intermediary status not useful for my algorithm. For my purpose, I want only approved ("Proposta/Plano de Trabalho Aprovados") and not approved proposals ("Proposta/Plano de Trabalho Rejeitados" and "Proposta/Plano de Trabalho Rejeitados por Impedimento técnico"). I will then filter this situation from the SIT_PROPOSTA column.

The following code do all this at once:

```
tab_proposal<-siconv_proposta%>%
  filter(SIT_PROPOSTA== "Proposta/Plano de Trabalho Aprovados" |
SIT_PROPOSTA== "Proposta/Plano de Trabalho Rejeitados" |
SIT_PROPOSTA=="Proposta/Plano de Trabalho Rejeitados por Impedimento
técnico")%>%
  select(ID_PROPOSTA,SIT_PROPOSTA,
UF_PROPONENTE,COD_ORGAO_SUP,NATUREZA_JURIDICA,ANO_PROP,VL_GLOBAL_PROP)
head(tab_proposal)

## # A tibble: 6 x 7
##   ID_PROPOSTA SIT_PROPOSTA UF_PROPONENTE COD_ORGAO_SUP
NATUREZA_JURIDI~
##         <dbl> <chr>        <chr>                <dbl> <chr>
## 1        1203 Proposta/Pl~ MG                   44000 Administração
P~
## 2        1239 Proposta/Pl~ RS                   54000 Administração
P~
## 3        1282 Proposta/Pl~ PR                   44000 Administração
P~
## 4        1316 Proposta/Pl~ SP                   54000 Administração
```

```
P~
## 5        1324 Proposta/Pl~ RJ                    55000 Organização da
~
## 6        1390 Proposta/Pl~ SP                    20113 Organização da
~
## # ... with 2 more variables: ANO_PROP <dbl>, VL_GLOBAL_PROP <dbl>
```

Since I only want to know if a proposal was approved or no, I will convert the SIT_PROPOSTA column to binary values, being *1 = approved* and *0 = not approved*:

```
tab_proposal$SIT_PROPOSTA<-
ifelse(tab_proposal$SIT_PROPOSTA=="Proposta/Plano de Trabalho
Aprovados",1,0)
dim(tab_proposal)

## [1] 268068      7
```

I will also add information about filling the justification fields of the proposal, because there are many cases of NAs and I believe filling correctly these fields can improve approval chances.

First I will add columns that I want from *siconv_justificativas_proposta* to *tab_justificativa*:

```
tab_justificativa<-siconv_justificativas_proposta%>%
  select(ID_PROPOSTA,CARACTERIZACAO_INTERESSES_RECI,CAPACIDADE_TECNICA)
```

Now I join this columns to tab_proposal:

```
tab_proposal<-left_join(tab_proposal,tab_justificativa,by="ID_PROPOSTA")
head(tab_proposal)

## # A tibble: 6 x 9
##   ID_PROPOSTA SIT_PROPOSTA UF_PROPONENTE COD_ORGAO_SUP
NATUREZA_JURIDI~
##         <dbl>        <dbl> <chr>                <dbl> <chr>
## 1        1203            0 MG                   44000 Administração
P~
## 2        1239            0 RS                   54000 Administração
P~
## 3        1282            0 PR                   44000 Administração
P~
## 4        1316            0 SP                   54000 Administração
P~
## 5        1324            0 RJ                   55000 Organização da
~
## 6        1390            1 SP                   20113 Organização da
~
## # ... with 4 more variables: ANO_PROP <dbl>, VL_GLOBAL_PROP <dbl>,
## #   CARACTERIZACAO_INTERESSES_RECI <chr>, CAPACIDADE_TECNICA <chr>
```

I will check the proportion of NAs for both columns:

```
prop_caracterizacao<-
sum(!is.na(tab_proposal$CARACTERIZACAO_INTERESSES_RECI))/nrow(tab_proposa
l)
prop_caracterizacao
```

```
## [1] 0.02346793
```

```
prop_capacidade<-
sum(!is.na(tab_proposal$CAPACIDADE_TECNICA))/nrow(tab_proposal)
prop_capacidade
```

```
## [1] 0.6854567
```

Only 2.3% of the proposals fill justification fields, while 68% fill techical capacity field. I will check if both fields can work as predictors later, within my models. The final step is to replace *NA* with *0* and any other result with *1*.

```
tab_proposal$CARACTERIZACAO_INTERESSES_RECI<-
ifelse(is.na(tab_proposal$CARACTERIZACAO_INTERESSES_RECI),0,1)
tab_proposal$CAPACIDADE_TECNICA<-
ifelse(is.na(tab_proposal$CAPACIDADE_TECNICA),0,1)
```

Possibly an important predictor is whether the proposal comes from a parliamentary amendment or not. This information can be found at *siconv_emenda.csv* file. I will create a table named *tab_emendas* grouping information I need.

```
tab_emenda<-siconv_emenda%>%
  select(ID_PROPOSTA,NR_EMENDA)
```

Now I add the columns to *tab_proposal*:

```
tab_proposal<-left_join(tab_proposal,tab_emenda,by="ID_PROPOSTA")
```

I replace *NA* with *0* and any other result (meaning proposal comes from parliamentary amendment) with *1*.

```
tab_proposal$NR_EMENDA<-ifelse(is.na(tab_proposal$NR_EMENDA),0,1)
head(tab_proposal)
```

```
## # A tibble: 6 x 10
##   ID_PROPOSTA SIT_PROPOSTA UF_PROPONENTE COD_ORGAO_SUP
NATUREZA_JURIDI~
##         <dbl>        <dbl> <chr>                <dbl> <chr>
## 1        1203            0 MG                   44000 Administração
P~
## 2        1239            0 RS                   54000 Administração
P~
## 3        1282            0 PR                   44000 Administração
P~
## 4        1316            0 SP                   54000 Administração
P~
## 5        1324            0 RJ                   55000 Organização da
```

```
~
## 6        1390            1 SP                    20113 Organização da
~
## # ... with 5 more variables: ANO_PROP <dbl>, VL_GLOBAL_PROP <dbl>,
## #   CARACTERIZACAO_INTERESSES_RECI <dbl>, CAPACIDADE_TECNICA <dbl>,
## #   NR_EMENDA <dbl>
```

I will also include another predictor: *budget program*.

```
tab_proposal<-
left_join(tab_proposal,siconv_programa_proposta,by="ID_PROPOSTA")
```

Other important predictor is proposal year. More specifically, my guess is election years increase the approval chances of a proposal. In Brazil, election occurs in even years, so it is easy to use this as a predictor:

```
tab_proposal<-tab_proposal%>%
      mutate(ELECTION_YEAR=ifelse(ANO_PROP %% 2 == 0, 1,0))%>%
      select(-ANO_PROP)
head(tab_proposal)
```

```
## # A tibble: 6 x 11
##   ID_PROPOSTA SIT_PROPOSTA UF_PROPONENTE COD_ORGAO_SUP
NATUREZA_JURIDI~
##         <dbl>        <dbl> <chr>                <dbl> <chr>
## 1        1203            0 MG                   44000 Administração
P~
## 2        1239            0 RS                   54000 Administração
P~
## 3        1282            0 PR                   44000 Administração
P~
## 4        1316            0 SP                   54000 Administração
P~
## 5        1324            0 RJ                   55000 Organização da
~
## 6        1390            1 SP                   20113 Organização da
~
## # ... with 6 more variables: VL_GLOBAL_PROP <dbl>,
## #   CARACTERIZACAO_INTERESSES_RECI <dbl>, CAPACIDADE_TECNICA <dbl>,
## #   NR_EMENDA <dbl>, ID_PROGRAMA <dbl>, ELECTION_YEAR <dbl>
```

Final steps to tidy *tab_proposal* dataset includes changing some variables to *factor* and changing *SIT_PROPOSTA* from binary values to *factor* ("YES" and "NO"), because some models requires this. Besides this, I will remove *ID_PROPOSTA* column.

```
tab_proposal<-tab_proposal%>%
      mutate(SIT_PROP_YN=ifelse(SIT_PROPOSTA == 1, "YES","NO"))%>%
      select(-ID_PROPOSTA, -SIT_PROPOSTA)

tab_proposal$UF_PROPONENTE=as.factor(tab_proposal$UF_PROPONENTE)
tab_proposal$COD_ORGAO_SUP=as.factor(tab_proposal$COD_ORGAO_SUP)
```

```
tab_proposal$NATUREZA_JURIDICA=as.factor(tab_proposal$NATUREZA_JURIDICA)
tab_proposal$CARACTERIZACAO_INTERESSES_RECI=as.factor(tab_proposal$CARACT
ERIZACAO_INTERESSES_RECI)
tab_proposal$CAPACIDADE_TECNICA=as.factor(tab_proposal$CAPACIDADE_TECNICA
)
tab_proposal$NR_EMENDA=as.factor(tab_proposal$NR_EMENDA)
tab_proposal$ID_PROGRAMA=as.factor(tab_proposal$ID_PROGRAMA)
tab_proposal$ELECTION_YEAR=as.factor(tab_proposal$ELECTION_YEAR)
tab_proposal$SIT_PROP_YN=as.factor(tab_proposal$SIT_PROP_YN)
str(tab_proposal)
```

```
## Classes 'spec_tbl_df', 'tbl_df', 'tbl' and 'data.frame': 271572 obs.
of  10 variables:
##  $ UF_PROPONENTE                 : Factor w/ 27 levels
"AC","AL","AM",..: 11 23 18 26 19 26 18 19 23 18 ...
##  $ COD_ORGAO_SUP                 : Factor w/ 30 levels
"14000","20000",..: 20 25 20 25 26 3 5 26 5 5 ...
##  $ NATUREZA_JURIDICA             : Factor w/ 5 levels "Administração
Pública Estadual ou do Distrito Federal",..: 2 2 2 2 5 5 2 5 2 2 ...
##  $ VL_GLOBAL_PROP                : num  1.01e+05 1.80e+05 2.09e+05
6.00e+05 1.61e+08 ...
##  $ CARACTERIZACAO_INTERESSES_RECI: Factor w/ 2 levels "0","1": 1 1 1 1
1 1 1 1 1 1 ...
##  $ CAPACIDADE_TECNICA            : Factor w/ 2 levels "0","1": 1 1 1 1
1 1 1 1 1 1 ...
##  $ NR_EMENDA                     : Factor w/ 2 levels "0","1": 1 1 1 1
1 1 1 1 1 1 ...
##  $ ID_PROGRAMA                   : Factor w/ 22070 levels
"1203","1206",..: 19 22 19 1 20189 25 63 45 21 21 ...
##  $ ELECTION_YEAR                 : Factor w/ 2 levels "0","1": 2 2 2 2
2 2 2 2 2 2 ...
##  $ SIT_PROP_YN                   : Factor w/ 2 levels "NO","YES": 1 1
1 1 1 2 1 2 2 1 ...
```

2.2 Data analysis

It is time to analyze *tab_proposal* dataset to better understand what challenges I will face in order to predict results. First of all, let's check the final dimmensions of my dataset:

```
dim(tab_proposal)
```

```
## [1] 271572     10
```

Now, let's see what proportion of proposal were accepted:

```
mean(tab_proposal$SIT_PROP_YN=="YES")
```

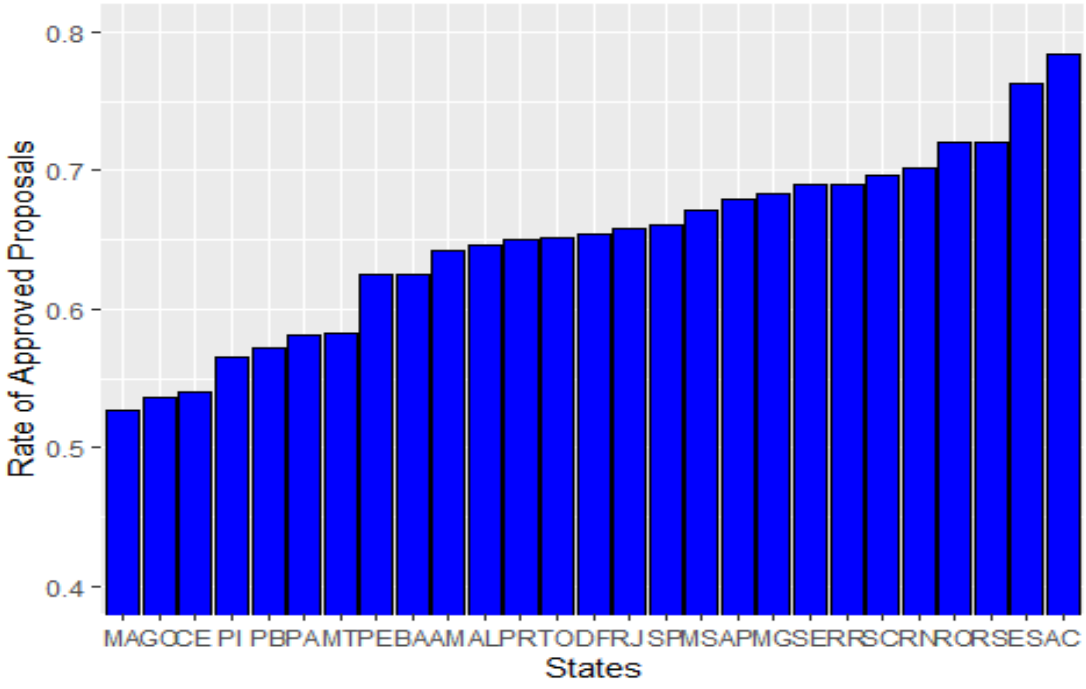
```
## [1] 0.647633
```

Now, let's group data by our variables:

```r
tab_group_by_UF<-tab_proposal%>%
  group_by(UF_PROPONENTE)%>%
  summarise(n=n(), Approved=mean(SIT_PROP_YN=="YES"))

tab_group_by_UF
```

```
## # A tibble: 27 x 3
##    UF_PROPONENTE     n Approved
##    <fct>         <int>    <dbl>
##  1 AC             3349    0.784
##  2 AL             4162    0.646
##  3 AM             2822    0.642
##  4 AP             2237    0.680
##  5 BA            16061    0.626
##  6 CE            12221    0.540
##  7 DF             2696    0.655
##  8 ES             5287    0.763
##  9 GO            13076    0.537
## 10 MA             8314    0.527
## # ... with 17 more rows
```

```r
tab_group_by_UF%>%
  mutate(UF_PROPONENTE = reorder(UF_PROPONENTE, Approved)) %>%
  ggplot(aes(UF_PROPONENTE,Approved)) +
  geom_bar(stat = "identity", fill = "blue", color = "black")+
  xlab("States") +
  ylab("Rate of Approved Proposals")+
  coord_cartesian(ylim = c(0.4, 0.8))
```



We can see the proportion of approved proposals varys from one state to other. While Maranhão (MA) has 53% of approved proposals, Acre (AC) gets 78% of its proposals approved.

Time to check another variable: *COD_ORGAO_SUP*, which discriminates government agency responsible for the transfer of resources.

```
tab_group_by_COD<-tab_proposal%>%
  group_by(COD_ORGAO_SUP)%>%
  summarise(n=n(), Approved=mean(SIT_PROP_YN=="YES"))%>%
  arrange(desc(n))

tab_group_by_COD
```

```
## # A tibble: 30 x 3
##    COD_ORGAO_SUP     n Approved
##    <fct>         <int>    <dbl>
##  1 22000         37127    0.890
##  2 54000         36221    0.551
##  3 55000         33609    0.376
##  4 36000         31441    0.965
##  5 53000         28865    0.730
##  6 56000         26921    0.885
##  7 51000         22898    0.560
##  8 49000         10907    0.284
##  9 30000          8041    0.437
## 10 20000          6453    0.190
## # ... with 20 more rows
```

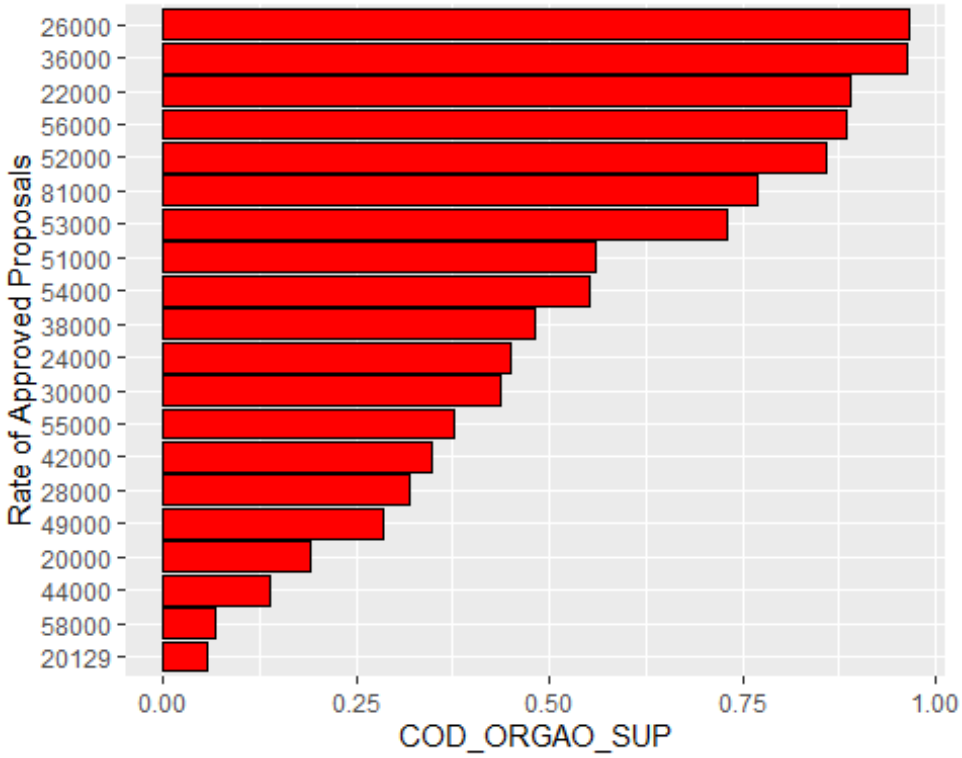
We can see there are 30 government agencies, with different number of proposals (n) and rates of approval. Analyzing *n*, 8 agencies have more then 10,000 proposals, while 10 have 100 or less. Difference in *n* may lead to a bias in the predictor. A simple way to minimize this is to establish a threshold. In this case, we can see *n* drops from 925 (COD_ORGAO_SUP = 58000) to 100 (COD_ORGAO_SUP = 32000). Therefore, I will keep only agencies with number of proposals greater than 100:

```
tab_proposal<-tab_proposal%>%
       group_by(COD_ORGAO_SUP)%>%
       mutate(n_ORGAO=sum(n()))%>%
       filter(n_ORGAO>100)%>%
       ungroup()
```

Now, let's group again and plot the results:

```
tab_group_ORGAO<-tab_proposal%>%
  group_by(COD_ORGAO_SUP)%>%
  summarise(n=n(), Approved=mean(SIT_PROP_YN=="YES"))%>%
  arrange(desc(n))

tab_group_ORGAO%>%
  mutate(COD_ORGAO_SUP = reorder(COD_ORGAO_SUP, Approved)) %>%
```

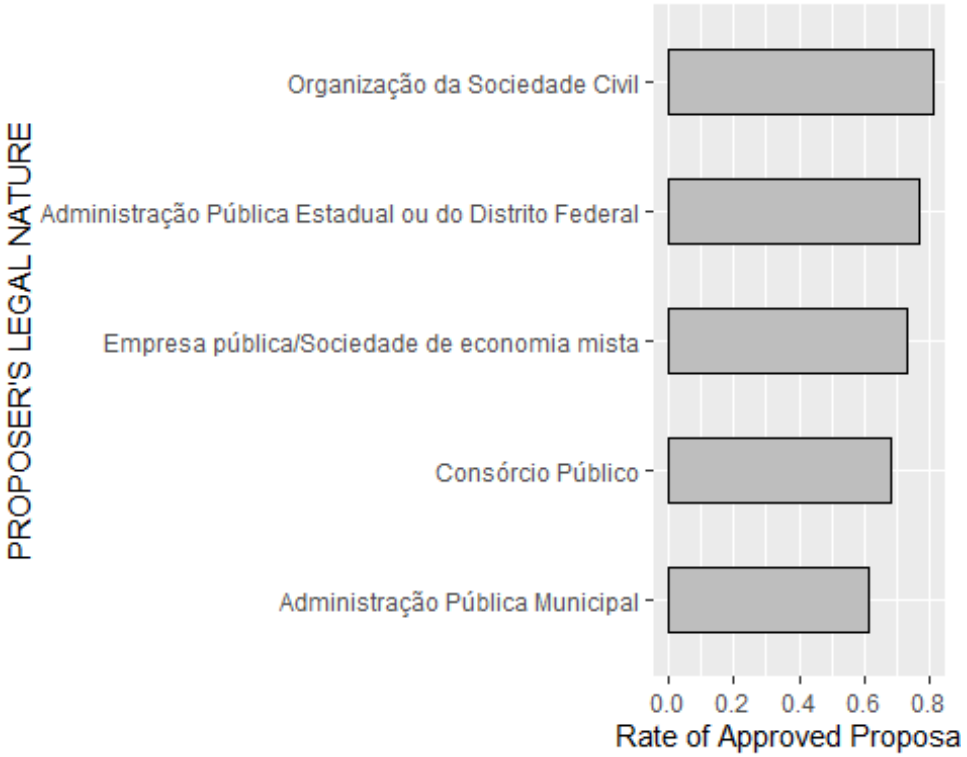
```
  ggplot(aes(COD_ORGAO_SUP, Approved)) +
  geom_bar(stat = "identity", fill = "red", color = "black")+
  coord_flip()+
  xlab("Rate of Approved Proposals")+
  ylab("COD_ORGAO_SUP")
```



It's possible to see that rate of approval enormously varies among *COD_ORGAO_SUP*, meaning this variable will possibly work as a predictor.

Now I will check the variable *NATUREZA_JURIDICA*, which shows who is proposing the agreement: municipalities, states or non-governmental entities.

```
tab_group_NATUREZA<-tab_proposal%>%
  group_by(NATUREZA_JURIDICA)%>%
  summarise(n=n(), Approved=mean(SIT_PROP_YN=="YES"))

tab_group_NATUREZA%>%
  mutate(NATUREZA_JURIDICA = reorder(NATUREZA_JURIDICA, Approved)) %>%
  ggplot(aes(NATUREZA_JURIDICA, Approved)) +
  geom_bar(stat = "identity", fill = "grey", color = "black", width =
0.5)+
  coord_flip()+
  ylab("Rate of Approved Proposals")+
  xlab("PROPOSER'S LEGAL NATURE")
```

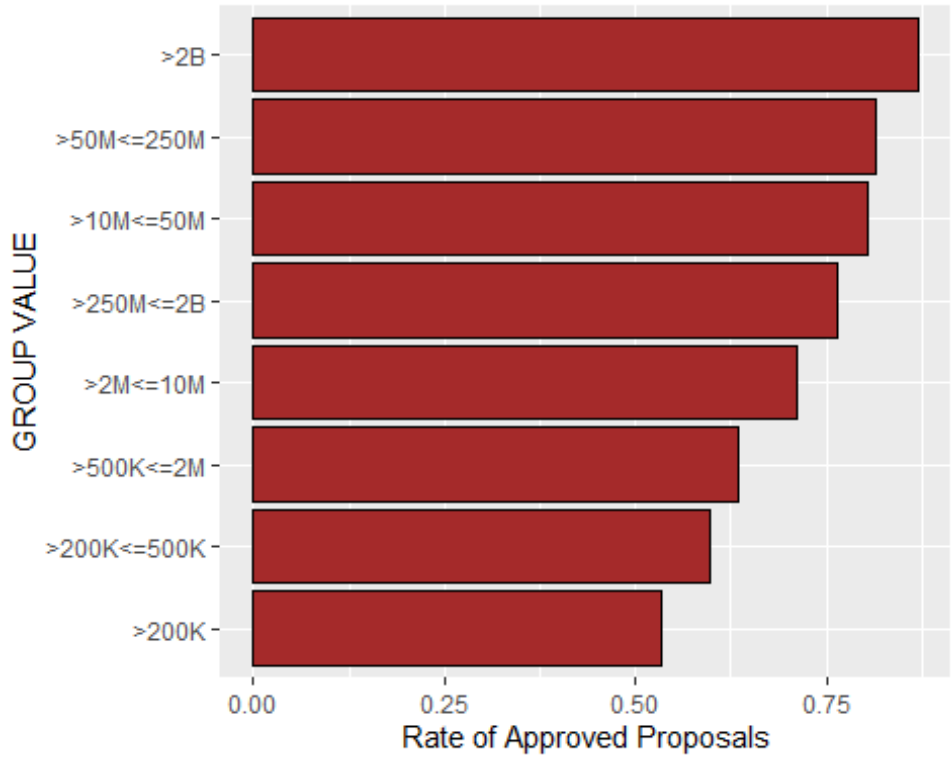


In this case, we can see the variability is smaller, being the smallest rate of approval municipalities (Administração Pública Municipal), with 61%, and the highest, non-governmental entities, with 80%.

To check approval rate by value, I have to define some intervals, since variance is huge. There are proposals from *0* to almost *$50 bilions*.

```
tab_proposal<-tab_proposal%>%
  mutate(GROUP_VALUE=ifelse(VL_GLOBAL_PROP<=200000,">200K",
ifelse(VL_GLOBAL_PROP<=500000,">200K<=500K",
               ifelse(VL_GLOBAL_PROP<=2000000,">500K<=2M",
               ifelse(VL_GLOBAL_PROP<=10000000,">2M<=10M",
               ifelse(VL_GLOBAL_PROP<=50000000,">10M<=50M",
               ifelse(VL_GLOBAL_PROP<=250000000,">50M<=250M",
ifelse(VL_GLOBAL_PROP<=2000000000,">250M<=2B",">2B")))))))%>%
  select(-n_ORGAO,-VL_GLOBAL_PROP)

tab_group_VALUE<-tab_proposal%>%
  group_by(GROUP_VALUE)%>%
  summarise(n=n(), Approved=mean(SIT_PROP_YN=="YES"))

tab_group_VALUE%>%
  mutate(GROUP_VALUE = reorder(GROUP_VALUE, Approved)) %>%
  ggplot(aes(GROUP_VALUE, Approved)) +
  geom_bar(stat = "identity", fill = "brown", color = "black")+
  coord_flip()+
```

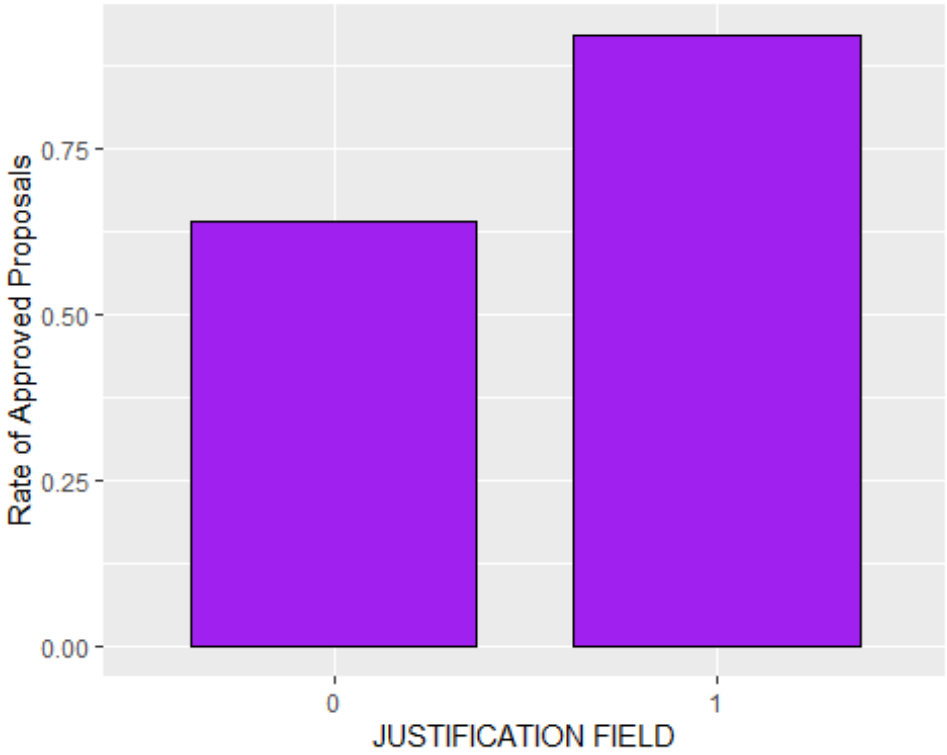
```
  ylab("Rate of Approved Proposals")+
  xlab("GROUP VALUE")
```



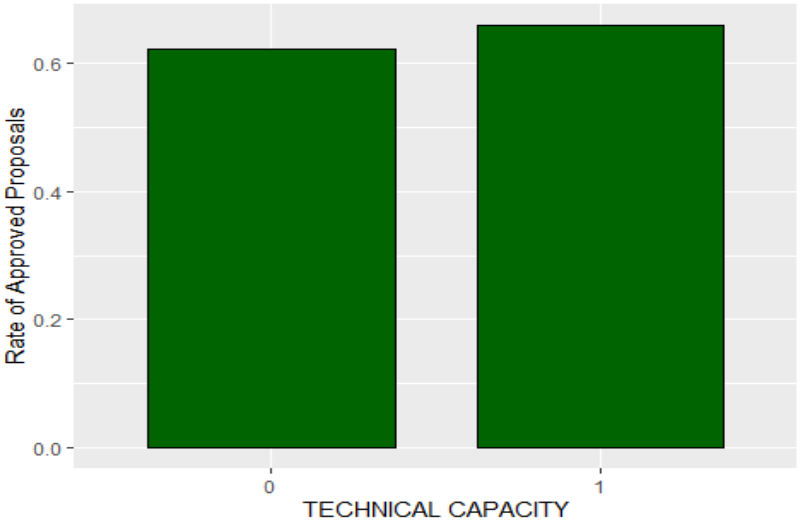
Generally speaking, as proposal value increases, rate of approval increases too.

Two related variables are *CARACTERIZACAO_INTERESSES_RECI* and *CAPACIDADE_TECNICA*, both related to filling justification fields.

```
tab_group_CARACT<-tab_proposal%>%
  group_by(CARACTERIZACAO_INTERESSES_RECI)%>%
  summarise(n=n(), Approved=mean(SIT_PROP_YN=="YES"))

tab_group_CARACT%>%
  ggplot(aes(CARACTERIZACAO_INTERESSES_RECI, Approved)) +
  geom_bar(stat = "identity", fill = "purple", color = "black", width =
0.75)+
  ylab("Rate of Approved Proposals")+
  xlab("JUSTIFICATION FIELD")
```



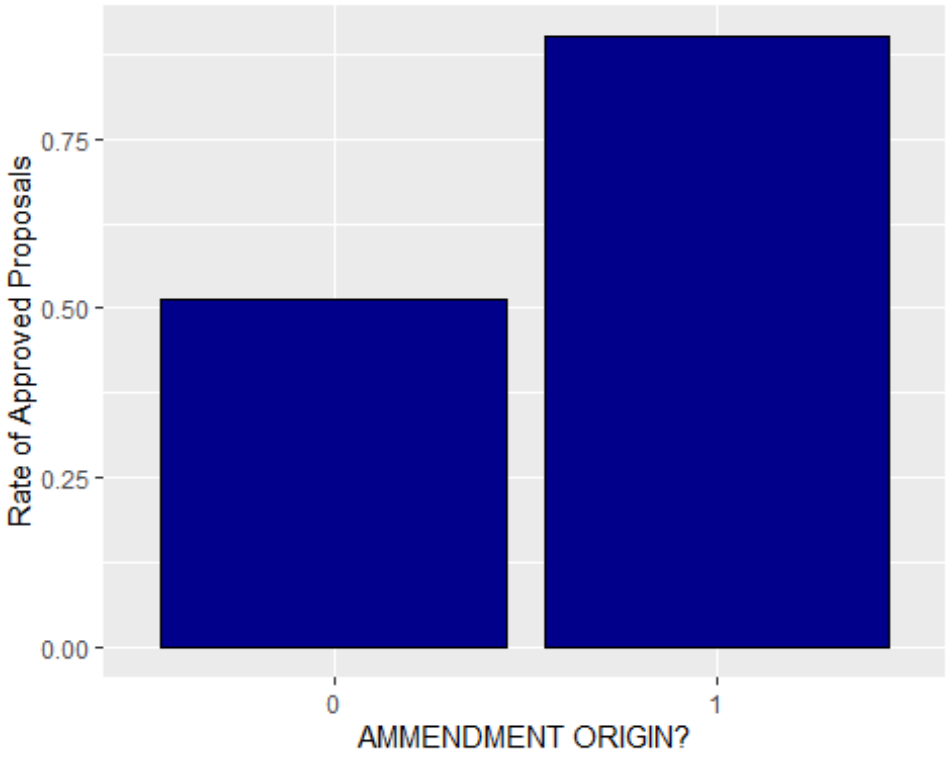

```
tab_group_TECH<-tab_proposal%>%
  group_by(CAPACIDADE_TECNICA)%>%
  summarise(n=n(), Approved=mean(SIT_PROP_YN=="YES"))

tab_group_TECH%>%
  ggplot(aes(CAPACIDADE_TECNICA, Approved)) +
  geom_bar(stat = "identity", fill = "dark green", color = "black", width
= 0.75)+
  ylab("Rate of Approved Proposals")+
  xlab("TECHNICAL CAPACITY")
```



As observed, filling justification field *CARACTERIZACAO_INTERESSES_RECI* increases rate of approval more than filling *CAPACIDADE_TECNICA* does.

Now let's check how rate of approval changes if proposal comes from parliamentary ammendment:

```
tab_group_AMMENDMENT<-tab_proposal%>%
  group_by(NR_EMENDA)%>%
  summarise(Approved=mean(SIT_PROP_YN=="YES"))

tab_group_AMMENDMENT%>%
  ggplot(aes(NR_EMENDA, Approved)) +
  geom_bar(stat = "identity", fill = "dark blue", color = "black")+
  ylab("Rate of Approved Proposals")+
  xlab("AMMENDMENT ORIGIN?")
```

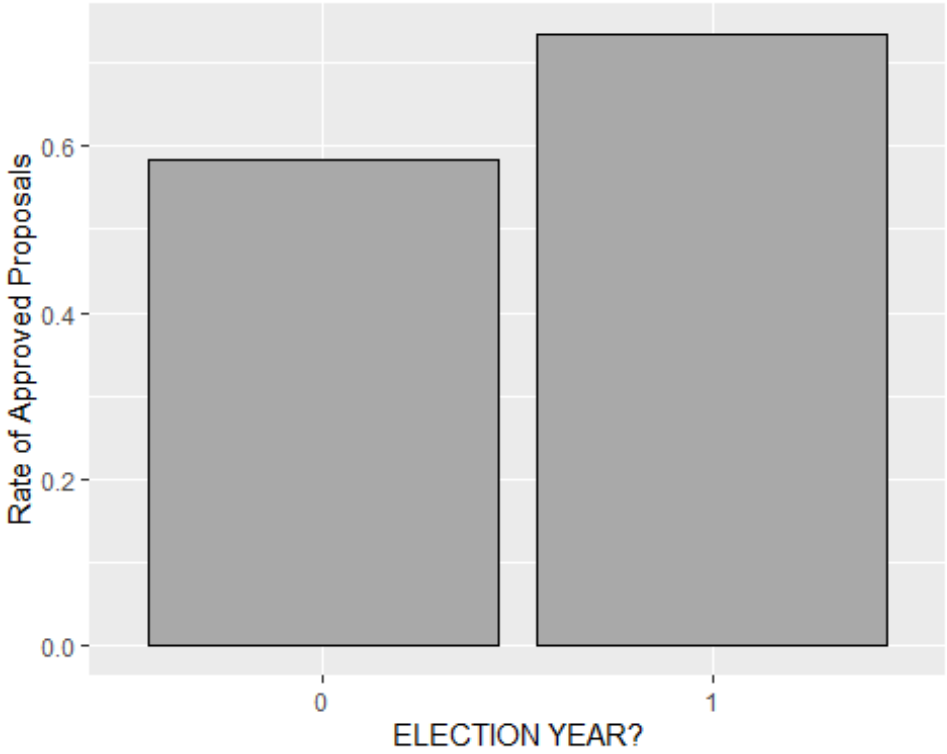


90% of proposals origined from parliamentary ammendment are approved, whilst only 51% of "independent proposals" are approved.

The following plot shows if election years increase approval rate:

```
tab_group_election<-tab_proposal%>%
  group_by(ELECTION_YEAR)%>%
  summarise(Approved=mean(SIT_PROP_YN=="YES"))

tab_group_election%>%
  ggplot(aes(ELECTION_YEAR, Approved)) +
```

```
  geom_bar(stat = "identity", fill = "dark grey", color = "black")+
  ylab("Rate of Approved Proposals")+
  xlab("ELECTION YEAR?")
```



As plot shows, in election years, 73% of proposals are approved, against 58% in non-election years.

The last variable is ID_PROGRAMA.

```
tab_group_PROGRAM<-tab_proposal%>%
  group_by(ID_PROGRAMA)%>%
  summarise(n=n(), Approved=mean(SIT_PROP_YN=="YES"))%>%
  arrange(desc(n))
head(tab_group_PROGRAM)
```

```
## # A tibble: 6 x 3
##   ID_PROGRAMA     n Approved
##   <fct>       <int>    <dbl>
## 1 36304        5426   0.0302
## 2 36434        4433   0.0244
## 3 2406         4130   0.993
## 4 35210        4028   0.248
## 5 14501        3588   0.277
## 6 5853         3134   0.382
```

```
tab_group_PROGRAM<-tab_proposal%>%
  group_by(ID_PROGRAMA)%>%
  summarise(n=n(), Approved=mean(SIT_PROP_YN=="YES"))%>%
```

```
  arrange(n)
head(tab_group_PROGRAM)

## # A tibble: 6 x 3
##   ID_PROGRAMA     n Approved
##   <fct>       <int>    <dbl>
## 1 1219            1        1
## 2 1228            1        1
## 3 1237            1        1
## 4 1281            1        0
## 5 1297            1        1
## 6 1355            1        1

dim(tab_group_PROGRAM)

## [1] 21835     3
```

We can see we have a problem. There are 21,770 unique values for ID_PROGRAMA, meaning there are a lot of programs with small value of *n*. In fact, 15,435 programs have only one proposal (n = 1). This situation represents two problems: first, small values of *n* generates unintended bias. Second, a very large number of levels in a variable can't be handled by most of models I want to use.

Again, a simple way to minimize this problem is to establish a threshold. In this case, I will keep only programs with $n > 100$:

```
tab_proposal<-tab_proposal%>%
  group_by(ID_PROGRAMA)%>%
  mutate(n_PROG=sum(n()))%>%
  filter(n_PROG>100)%>%
  ungroup()
n_distinct(tab_proposal$n_PROG)

## [1] 270
```

Now I have 271 unique values for ID_PROGRAM. This number is still too large to be handled by random forest models, for example. Because of this, I will group ID_PROGRAM according with approval rate. To do that, first I will add a column with this information:

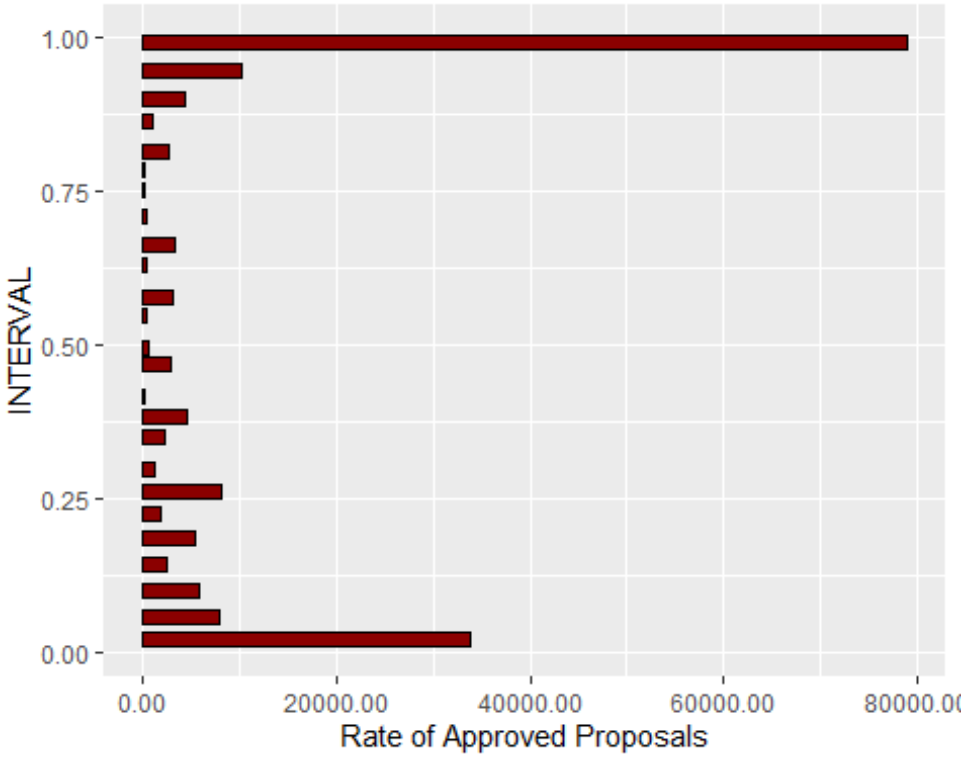
```
tab_proposal<-tab_proposal%>%
  group_by(ID_PROGRAMA)%>%
  mutate(PROP_PROG=sum(SIT_PROP_YN=="YES")/sum(n()))%>%
  ungroup()

tab_proposal$INTERVAL<-findInterval(tab_proposal$PROP_PROG,
seq(0,1,0.04), rightmost.closed = TRUE)

tab_proposal<-select(tab_proposal,-ID_PROGRAMA,-n_PROG,-PROP_PROG)
tab_proposal$INTERVAL<-as.factor(tab_proposal$INTERVAL)
tab_proposal$GROUP_VALUE<-as.factor(tab_proposal$GROUP_VALUE)
```

```
tab_proposal$COD_ORGAO_SUP<-factor(tab_proposal$COD_ORGAO_SUP)

str(tab_proposal)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    183018 obs. of  10
variables:
##  $ UF_PROPONENTE                 : Factor w/ 27 levels
"AC","AL","AM",..: 18 7 19 12 16 18 16 6 14 7 ...
##  $ COD_ORGAO_SUP                 : Factor w/ 19 levels
"20000","20129",..: 3 4 15 3 3 3 15 15 15 15 ...
##  $ NATUREZA_JURIDICA             : Factor w/ 5 levels "Administração
Pública Estadual ou do Distrito Federal",..: 2 5 5 2 2 2 5 1 4 5 ...
##  $ CARACTERIZACAO_INTERESSES_RECI: Factor w/ 2 levels "0","1": 1 1 1 1
1 1 1 1 1 1 ...
##  $ CAPACIDADE_TECNICA            : Factor w/ 2 levels "0","1": 1 1 1 1
1 1 1 1 1 1 ...
##  $ NR_EMENDA                     : Factor w/ 2 levels "0","1": 1 1 1 1
1 1 1 1 1 1 ...
##  $ ELECTION_YEAR                 : Factor w/ 2 levels "0","1": 2 2 2 2
2 2 2 2 2 2 ...
##  $ SIT_PROP_YN                   : Factor w/ 2 levels "NO","YES": 1 2
1 2 2 1 2 2 2 2 ...
##  $ GROUP_VALUE                   : Factor w/ 8 levels
">10M<=50M",">200K",..: 2 3 2 7 1 2 2 3 4 3 ...
##  $ INTERVAL                      : Factor w/ 25 levels
"1","2","3","4",..: 5 25 10 5 5 5 10 10 15 15 ...
```

Now I can see how approval rates vary according to program.

```
tab_group_INTERVAL<-tab_proposal%>%
  group_by(INTERVAL)%>%
  summarise(n=n(), Approved=mean(SIT_PROP_YN=="YES"))

scaleFUN <- function(x) sprintf("%.2f", x)

tab_group_INTERVAL%>%
  mutate(INTERVAL = reorder(INTERVAL, Approved)) %>%
  ggplot(aes(Approved, n)) +
  geom_bar(stat = "identity", fill = "dark red", color = "black")+
  coord_flip()+
  scale_y_continuous(labels=scaleFUN)+
  ylab("Rate of Approved Proposals")+
  xlab("INTERVAL")
```



More than 30,000 programs had 0% approval rate, whilst 80,000 were 100% approved. As we can see, the approval rate distribution among programs is not normal.

After this extensive analysis, finally I have my dataset ready to begin the development process.

2.2 Creating train and test sets

In order to test the predictors, I've decided to split the *tab_proposal* set into *train_set* and *test_set*. First, I must load *caret package*, which includes tools for data splitting.

```
library(caret)

#I set the seed to 1:

set.seed(1, sample.kind="Rounding")

## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform
'Rounding'
## sampler used

#Finally, I split the tab_proposal set, with 80% of the data to train_set
and 20% for test_set:

test_index <- createDataPartition(y = tab_proposal$SIT_PROP_YN, times =
1, p = 0.2, list = FALSE)
```

```
train_set <- tab_proposal[-test_index,]
test_set <- tab_proposal[test_index,]
```

While I tested the different models, I had huge problems with calculation time. Some models, without tune, took 12+ hours to run each test. This proved impractical. I decided then to create smaller train and test sets to run the first versions of my models.

```
set.seed(1, sample.kind="Rounding")

## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform
'Rounding'
## sampler used

train_small<-sample_n(tab_proposal, 10000)
test_small<-sample_n(tab_proposal, 10000)
```

2.3 Testing models

2.3.1 First model - GLM

The first model I will test will be *glm*, or *generalized linear model*. In project description, students are required to "go beyond standard linear regression". I will apply this to this project in a way that my starting point, my reference will be results obtained with this model.

```
set.seed(1, sample.kind="Rounding")

## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform
'Rounding'
## sampler used

glm_model = glm(SIT_PROP_YN ~ ., family = binomial,
                data = train_small)

p_glm<-predict(glm_model,newdata = test_small)

y_glm<-ifelse(p_glm>0.5,"YES","NO")
y_glm<-factor(y_glm)


confMatrix.glm <-
confusionMatrix(table(as.factor(y_glm),as.factor(test_small$SIT_PROP_YN))
)

confMatrix.glm$overall["Accuracy"]

## Accuracy
##   0.9114

confMatrix.glm$byClass["F1"]
```

```
##        F1
## 0.8934584
```

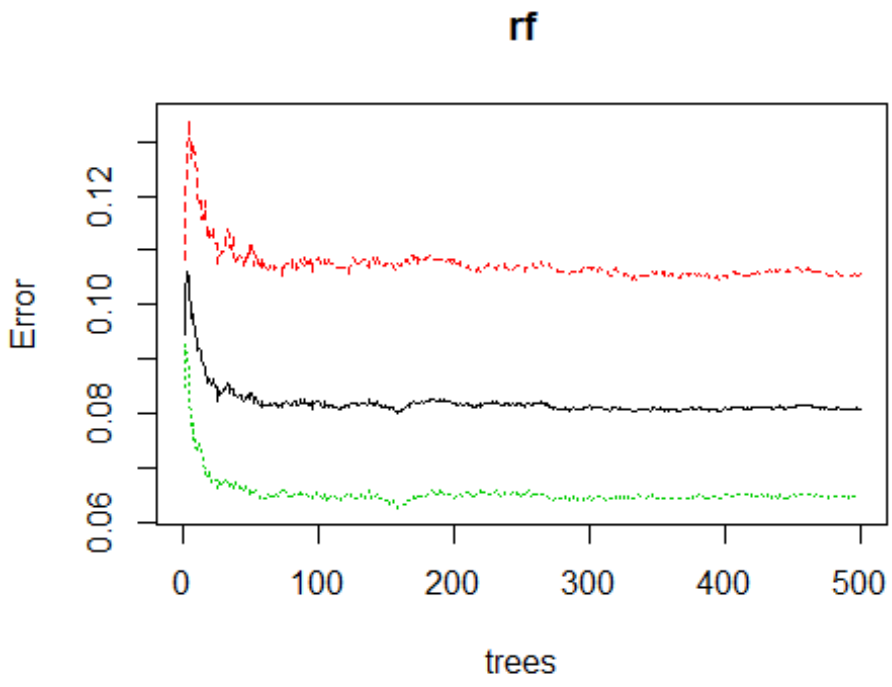
My start is already good, with Accuracy of 0.9114 and Accuracy of 0.8934.

I will try to improve this numbers using other models.

2.3.2 Second model - Random Forest

This is a very simple and "untuned" randomForest test. I will start with *ntree = 500*.

```
set.seed(1, sample.kind="Rounding")

## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform
'Rounding'
## sampler used

rf = randomForest(SIT_PROP_YN ~ .,data=train_small,ntree= 500,
importance=TRUE)

plot(rf)
```




```
p_rf<-predict(rf,newdata = test_small)

confMatrix.rf<-confusionMatrix(table(p_rf,test_small$SIT_PROP_YN))
confMatrix.rf$overall["Accuracy"]
```

```
## Accuracy
##   0.9205

confMatrix.rf$byClass["F1"]

##        F1
## 0.8989706
```

As seen, I've got a little improvement in my second model. Time to test other options.

2.3.3 Third model - Random Forest with "Rborist" method

```
set.seed(1, sample.kind="Rounding")



rf_rborist <- train(SIT_PROP_YN ~ .,

                    method = "Rborist",

                    tuneGrid = data.frame(predFixed = 2, minNode = 1),

                    data = train_small,

                    nTree = 300,

                    classWeight = "balance",

                    minInfo=0.01)

## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform
'Rounding'
## sampler used

p_rf_rborist<-predict(rf_rborist,newdata = test_small)

confMatrixRF_rborist<-confusionMatrix(table(p_rf_rborist,
test_small$SIT_PROP_YN))
confMatrixRF_rborist$overall["Accuracy"]

## Accuracy
##   0.8936

confMatrixRF_rborist$byClass["F1"]

##        F1
## 0.8762791
```

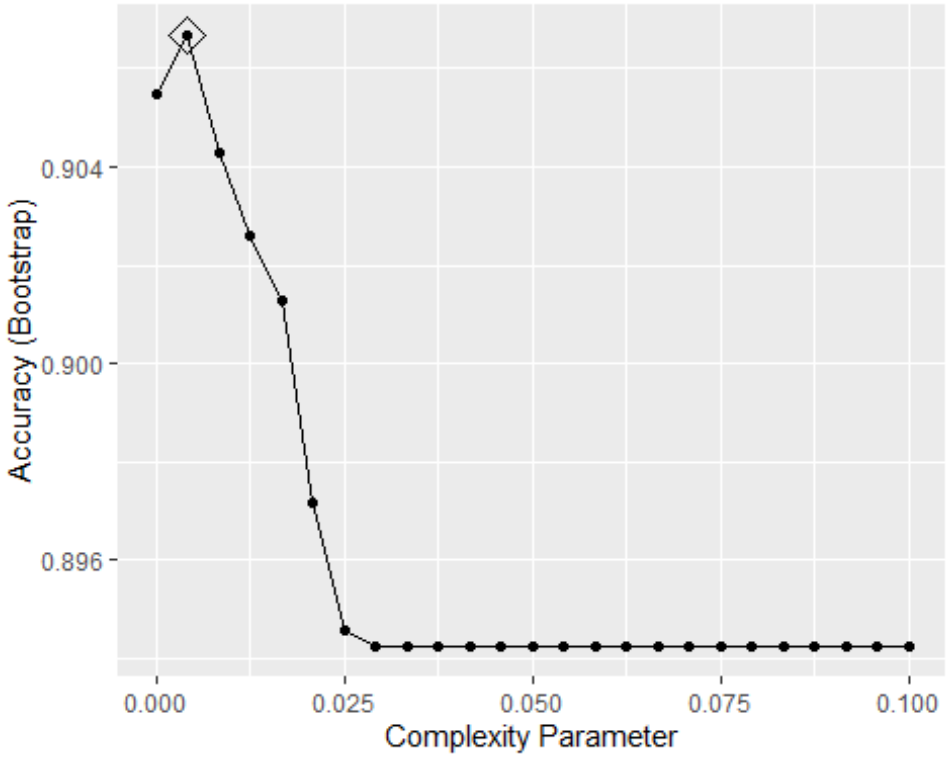
I tuned parameters for this method, but didn't get good results. No combination could improve Accuracy up to 0.90.

2.3.4 Fourth model - Classification (Decision) Trees

```r
set.seed(1, sample.kind="Rounding")

train_rpart <- train(SIT_PROP_YN ~ .,

                     method = "rpart",

                     tuneGrid = data.frame(cp = seq(0.0, 0.1, len = 25)),

                     data = train_small)
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform
'Rounding'
## sampler used
```

```r
ggplot(train_rpart, highlight = TRUE)
```




```r
p_rpart<-predict(train_rpart,newdata = test_small)

confMatrix.rpart<-confusionMatrix(table(p_rpart, test_small$SIT_PROP_YN))

confMatrix.rpart$overall["Accuracy"]
```

```
## Accuracy
##   0.9048
```

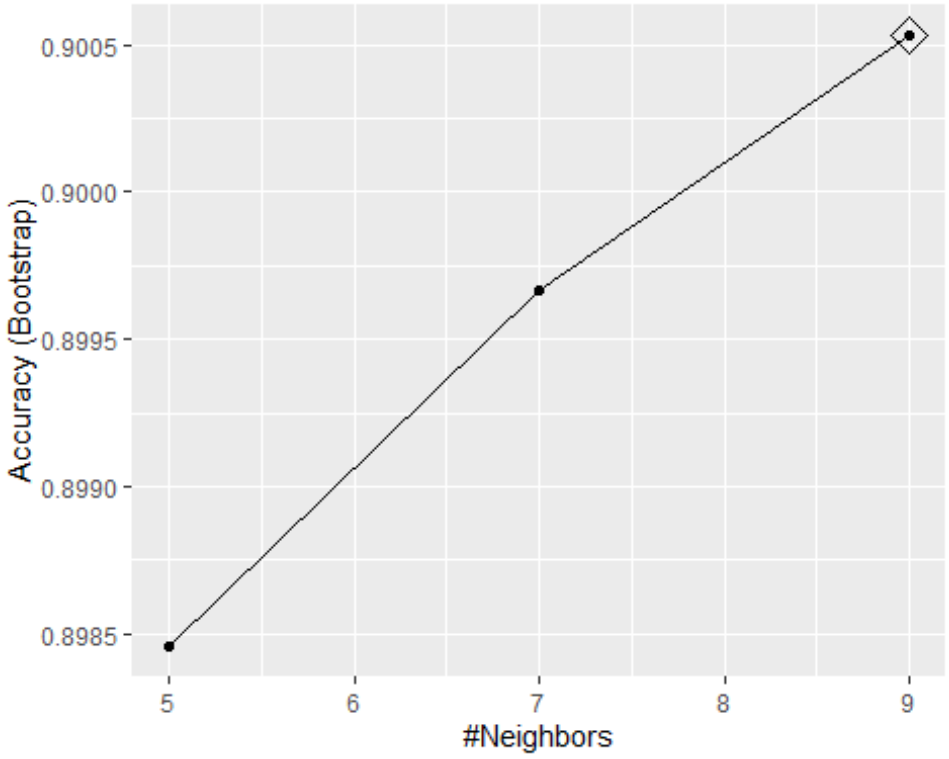
```r
confMatrix.rpart$byClass["F1"]
```

```
##        F1
## 0.8844099
```

2.3.5 Fifth model - KNN

The next method is *kNN*, or *k-nearest neighbors*:

```
set.seed(1, sample.kind="Rounding")

train_knn <- train(SIT_PROP_YN ~ .,

                   method = "knn",

                   data = train_small)

## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform
'Rounding'
## sampler used

ggplot(train_knn, highlight = TRUE)
```




```
p_knn<-predict(train_knn,newdata = test_small)

confMatrix.knn<-confusionMatrix(table(p_knn, test_small$SIT_PROP_YN))

confMatrix.knn$overall["Accuracy"]

## Accuracy
##   0.9016

confMatrix.knn$byClass["F1"]
```

```
##        F1
## 0.8793822
```

2.3.6 Sixth model - Conditional Tree

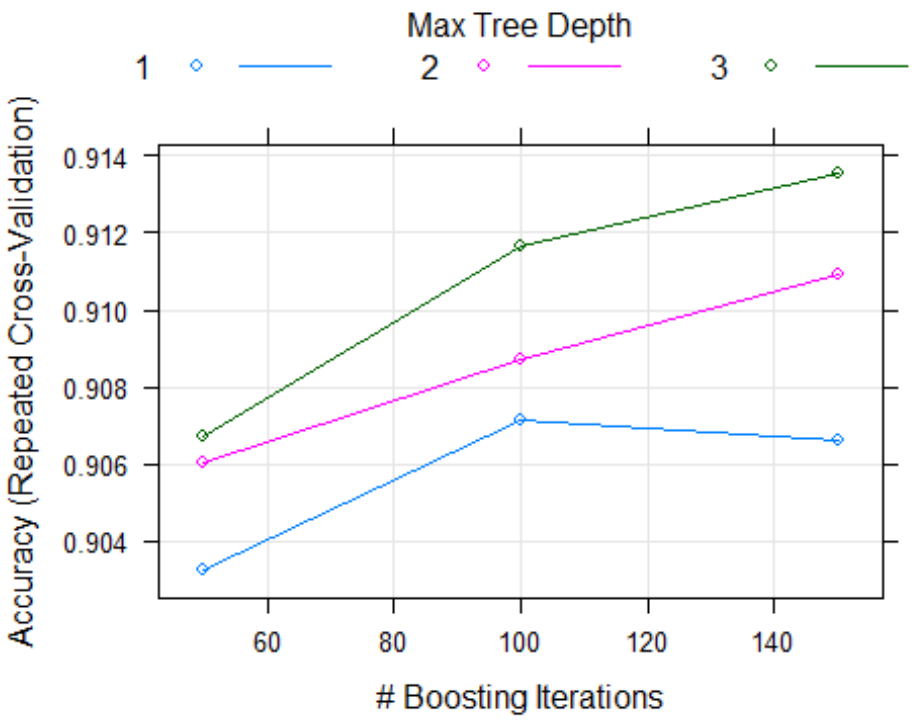
```
set.seed(1, sample.kind="Rounding")

## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform
'Rounding'
## sampler used

fitControl <- trainControl(method = "cv", number = 10)

ctree = ctree(SIT_PROP_YN ~ .,
                        data=train_small)

p_ctree<-predict(ctree,newdata = test_small)

confMatrix.ctree<-confusionMatrix(table(p_ctree, test_small$SIT_PROP_YN))

confMatrix.ctree$overall["Accuracy"]

## Accuracy
##    0.911

confMatrix.ctree$byClass["F1"]

##        F1
## 0.8869411
```

2.3.7 Seventh model - GBM

This method is called *Generalized Boosted Regression Model*. I'll set it up to a 10-fold cross-validation with 10 times repetition.

```
fitControl <- trainControl(## 10-fold CV
  method = "repeatedcv",
  number = 10,
  ## repeated ten times
  repeats = 10)

set.seed(1, sample.kind="Rounding")

gbmFit1 <- train(SIT_PROP_YN ~ .,
                 data = train_small,
                 method = "gbm",
                 trControl = fitControl,
                 verbose = FALSE)
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform
'Rounding'
## sampler used
```

```
p_gbmFit1.sm<-predict(gbmFit1,newdata = test_small)

confMatrix.gbmfit1.sm<-confusionMatrix(table(p_gbmFit1.sm,
test_small$SIT_PROP_YN))

plot(gbmFit1)
```

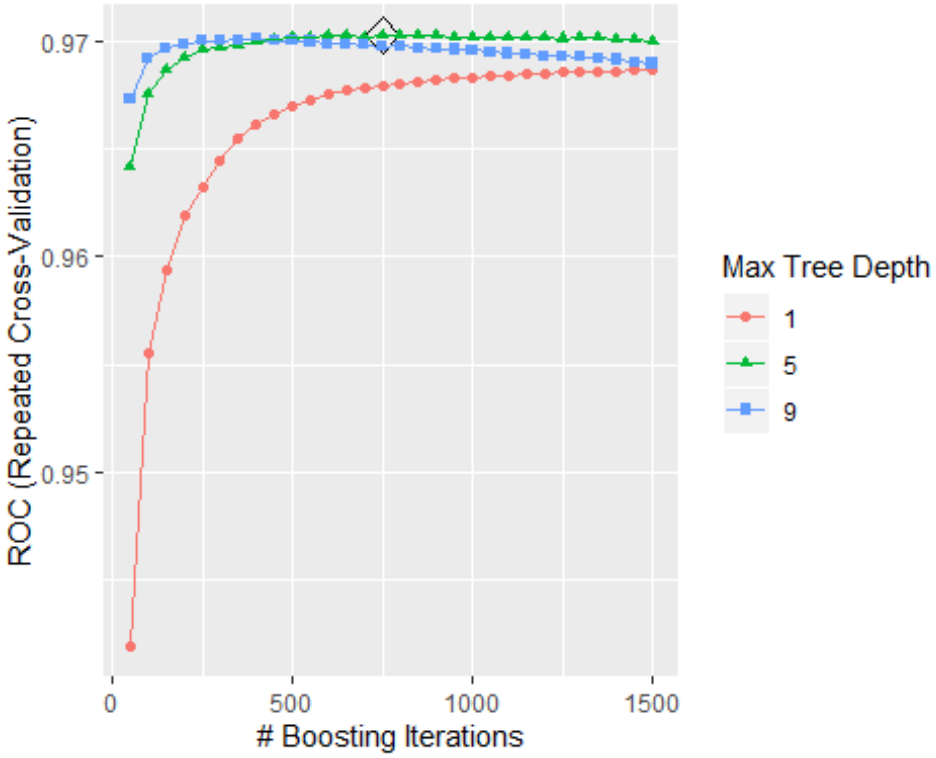



```
confMatrix.gbmfit1.sm$overall["Accuracy"]

## Accuracy
##   0.9113

confMatrix.gbmfit1.sm$byClass["F1"]

##        F1
## 0.8914719
```

Tuning some parameters of the model:

```
fitControl <- trainControl(method = "repeatedcv",
                           number = 10,
                           repeats = 10,
                           ## Estimate class probabilities
                           classProbs = TRUE,
                           ## Evaluate performance using
                           ## the following function
                           summaryFunction = twoClassSummary)
```

```r
gbmGrid <-  expand.grid(interaction.depth = c(1, 5, 9),
                        n.trees = (1:30)*50,
                        shrinkage = 0.1,
                        n.minobsinnode = 20)

set.seed(1, sample.kind="Rounding")

gbmFit3 <- train(SIT_PROP_YN ~ .,
                 data = train_small,
                 method = "gbm",
                 trControl = fitControl,
                 verbose = FALSE,
                 tuneGrid = gbmGrid,
                 ## Specify which metric to optimize
                 metric = "ROC")


## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform
'Rounding'
## sampler used

p_gbmFit3<-predict(gbmFit3,newdata = test_small)

confMatrix.gbmfit3<-confusionMatrix(table(p_gbmFit3,
test_small$SIT_PROP_YN))

ggplot(gbmFit3, highlight = TRUE)
```



```
confMatrix.gbmfit3$overall["Accuracy"]

## Accuracy
##   0.9148

confMatrix.gbmfit3$byClass["F1"]

##        F1
## 0.8939771
```

2.4 Comparing model performance on small train set

After testing seven models (and several tuning parameters for each one of them not described here), it's time to compare them and choose which one gets better performance.

```
model.summary <- data.frame(Model = c("GLM", "randomForest",
"RF_Rborist", "Rpart", "kNN", "ctree", "GBM"),
                            Accuracy =
c(confMatrix.glm$overall["Accuracy"],
                      confMatrix.rf$overall["Accuracy"],
                       confMatrixRF_rborist$overall["Accuracy"],
                       confMatrix.rpart$overall["Accuracy"],
                       confMatrix.knn$overall["Accuracy"],
                       confMatrix.ctree$overall["Accuracy"],
                       confMatrix.gbmfit3$overall["Accuracy"]),
                      Sensitivity =
c(confMatrix.glm$byClass["Sensitivity"],
                      confMatrix.rf$byClass["Sensitivity"],
                       confMatrixRF_rborist$byClass["Sensitivity"],
                       confMatrix.rpart$byClass["Sensitivity"],
                       confMatrix.knn$byClass["Sensitivity"],
                       confMatrix.ctree$byClass["Sensitivity"],
                       confMatrix.gbmfit3$byClass["Sensitivity"]),
                      Specificity =
c(confMatrix.glm$byClass["Specificity"],
                      confMatrix.rf$byClass["Specificity"],
                       confMatrixRF_rborist$byClass["Specificity"],
                       confMatrix.rpart$byClass["Specificity"],
                       confMatrix.knn$byClass["Specificity"],
                       confMatrix.ctree$byClass["Specificity"],
                       confMatrix.gbmfit3$byClass["Specificity"]),
                      Precision = c(confMatrix.glm$byClass["Precision"],
                      confMatrix.rf$byClass["Precision"],
                       confMatrixRF_rborist$byClass["Precision"],
                       confMatrix.rpart$byClass["Precision"],
                       confMatrix.knn$byClass["Precision"],
                       confMatrix.ctree$byClass["Precision"],
                       confMatrix.gbmfit3$byClass["Precision"]),
                      Recall = c(confMatrix.glm$byClass["Recall"],
                      confMatrix.rf$byClass["Recall"],
```

```
                      confMatrixRF_rborist$byClass["Recall"],
                      confMatrix.rpart$byClass["Recall"],
                      confMatrix.knn$byClass["Recall"],
                      confMatrix.ctree$byClass["Recall"],
                      confMatrix.gbmfit3$byClass["Recall"]),
                    F1 = c(confMatrix.glm$byClass["F1"],
                    confMatrix.rf$byClass["F1"],
                      confMatrixRF_rborist$byClass["F1"],
                      confMatrix.rpart$byClass["F1"],
                      confMatrix.knn$byClass["F1"],
                      confMatrix.ctree$byClass["F1"],
                      confMatrix.gbmfit3$byClass["F1"]))



model.summary[order(-model.summary$Accuracy),]
```

```
##          Model Accuracy Sensitivity Specificity Precision    Recall
## 2 randomForest   0.9205   0.8981717   0.9350049 0.8997711 0.8981717
## 7          GBM   0.9148   0.9121381   0.9165292 0.8765251 0.9121381
## 1          GLM   0.9114   0.9433723   0.8906302 0.8485610 0.9433723
## 6        ctree   0.9110   0.8864906   0.9269218 0.8873920 0.8864906
## 4        Rpart   0.9048   0.9248349   0.8917849 0.8473709 0.9248349
## 5          kNN   0.9016   0.9108685   0.8955790 0.8500000 0.9108685
## 3   RF_Rborist   0.8936   0.9568309   0.8525239 0.8082368 0.9568309
##          F1
## 2 0.8989706
## 7 0.8939771
## 1 0.8934584
## 6 0.8869411
## 4 0.8844099
## 5 0.8793822
## 3 0.8762791
```

As we can see, randomForest gets a small advantage over the other models.

2.5 Trying to improve randomForest model

I will try to improve *randomForest* results through *train* function, from *caret* package, tuning some parameters. I tried to run these tests on the *train set* this time, but each run was taking 15-20 hours, so I decided to keep the test on *train_small and test_small*.

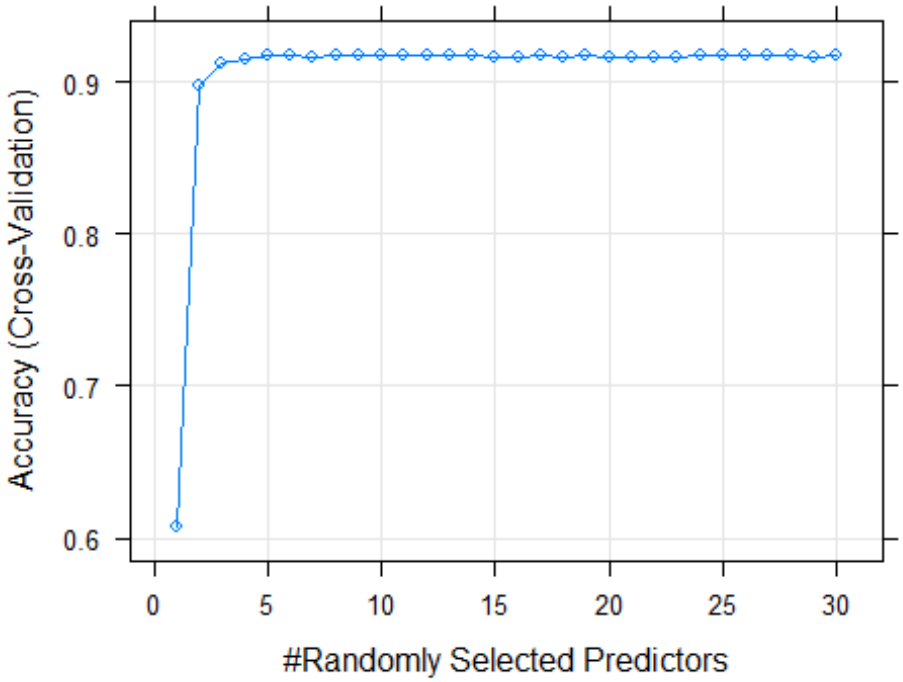
2.5.1 First try: testing different values for mtry

First try I will use *ten fold cross validation* and repeat it 3 times. I will test *mtry* values from 1 to 30.

```
numFolds <- trainControl(method = "cv", number = 10)

cpGrid <- expand.grid(.mtry = seq(1, 30, 1))
```

```r
set.seed(1, sample.kind="Rounding")

## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform
'Rounding'
## sampler used

print(rf_cvt)

## Random Forest
##
## 10000 samples
##     9 predictor
##     2 classes: 'NO', 'YES'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 8999, 9000, 9001, 9000, 9000, 9001, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    1    0.6069002  0.00784001
##    2    0.8974998  0.78594293
##    3    0.9123984  0.81883087
##    4    0.9146990  0.82361110
##    5    0.9170993  0.82857283
##    6    0.9176995  0.82974327
##    7    0.9166992  0.82761921
##    8    0.9174994  0.82924066
##    9    0.9181995  0.83059718
##   10    0.9176990  0.82950856
##   11    0.9172992  0.82861073
##   12    0.9173993  0.82869450
##   13    0.9174984  0.82881514
##   14    0.9171987  0.82814589
##   15    0.9165987  0.82674434
##   16    0.9168984  0.82731621
##   17    0.9172980  0.82806290
##   18    0.9167973  0.82697038
##   19    0.9169980  0.82721597
##   20    0.9167972  0.82675729
##   21    0.9164975  0.82610765
##   22    0.9164971  0.82605260
##   23    0.9167978  0.82662926
##   24    0.9173970  0.82785432
##   25    0.9171976  0.82739896
##   26    0.9171976  0.82745077
##   27    0.9172979  0.82761217
##   28    0.9169985  0.82690740
##   29    0.9164982  0.82588373
##   30    0.9170977  0.82711526
##
```

```
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 9.

plot(rf_cvt)
```



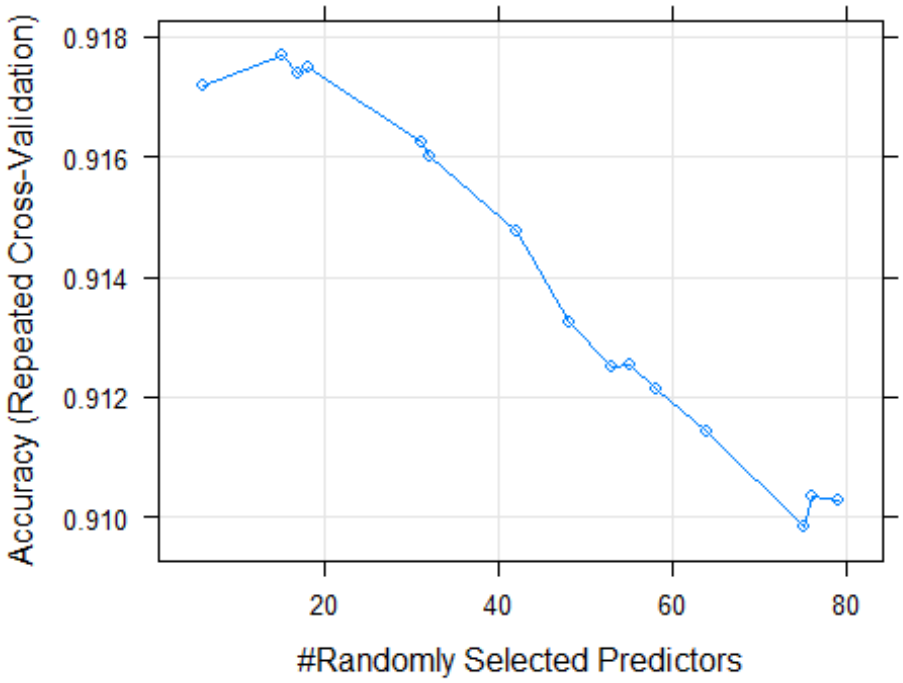

```
p_rf_cvt<-predict(rf_cvt,newdata = test_small)

confMatrix.rf_cvt<-
confusionMatrix(table(p_rf_cvt,test_small$SIT_PROP_YN))
confMatrix.rf_cvt$overall["Accuracy"]

## Accuracy
##   0.9174

confMatrix.rf_cvt$byClass["F1"]

##       F1
## 0.897187
```

2.5.2 Second try: Random Search

In this second attempt, I will add *random* mode to search paramater within *trainControl*. It will take longer, but *train* will test some random values of *mtry* and point the one with best results.

```
control <- trainControl(method="repeatedcv", number=10, repeats=3,
search="random")
```

```r
set.seed(1, sample.kind="Rounding")

mtry <- sqrt(ncol(train_small))

rf_random <- train(SIT_PROP_YN~., data=train_small, method="rf",
metric="Accuracy", tuneLength=15, trControl=control)
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform
'Rounding'
## sampler used

print(rf_random)

## Random Forest
##
## 10000 samples
##     9 predictor
##     2 classes: 'NO', 'YES'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 8999, 9000, 9001, 9000, 9000, 9001, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    6    0.9171660  0.8286692
##   15    0.9176993  0.8290459
##   17    0.9173659  0.8282136
##   18    0.9174657  0.8282957
##   31    0.9162323  0.8253394
##   32    0.9159993  0.8248319
##   42    0.9147655  0.8222320
##   48    0.9132322  0.8189672
##   53    0.9124991  0.8173910
##   55    0.9125320  0.8174280
##   58    0.9121322  0.8165950
##   64    0.9114324  0.8151351
##   75    0.9098322  0.8117839
##   76    0.9103655  0.8129092
##   79    0.9102989  0.8127848
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 15.

plot(rf_random)
```




```
p_rf_random<-predict(rf_random,newdata = test_small)

confMatrix.rf_random<-confusionMatrix(table(p_rf_random,
test_small$SIT_PROP_YN))

confMatrix.rf_random$overall["Accuracy"]

## Accuracy
##   0.9188

confMatrix.rf_random$byClass["F1"]

##        F1
## 0.8975265
```

2.5.3 Third try: different ntree with best mtry

We got *14* and *17* as best value for *mtry* in the two previous methods. Now I will test if increasing *ntree* I improve results for both *mtry* values.

```
set.seed(1, sample.kind="Rounding")

tunegrid <- expand.grid(.mtry=14)

rf_mtry_14 <- train(SIT_PROP_YN~., data=train_small, method="rf",
metric="Accuracy", tuneGrid=tunegrid, ntree = 1000)
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform
'Rounding'
## sampler used
```

```
print(rf_mtry_14)
```

```
## Random Forest
##
## 10000 samples
##     9 predictor
##     2 classes: 'NO', 'YES'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 10000, 10000, 10000, 10000, 10000, 10000, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.9144452  0.8227154
##
## Tuning parameter 'mtry' was held constant at a value of 14
```

```
p_rf_mtry_14<-predict(rf_mtry_14,newdata = test_small)
```

```
confMatrix.rf_mtry_14<-confusionMatrix(table(p_rf_mtry_14,
test_small$SIT_PROP_YN))
```

```
confMatrix.rf_mtry_14$overall["Accuracy"]
```

```
## Accuracy
##   0.9191
```

```
confMatrix.rf_mtry_14$byClass["F1"]
```

```
##        F1
## 0.8983541
```

```
set.seed(1, sample.kind="Rounding")
```

```
tunegrid <- expand.grid(.mtry=17)
```

```
rf_mtry_14 <- train(SIT_PROP_YN~., data=train_small, method="rf",
metric="Accuracy", tuneGrid=tunegrid, ntree = 1000)
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform
'Rounding'
## sampler used
```

```
print(rf_mtry_17)
```

```
## Random Forest
##
```

```
## 10000 samples
##     9 predictor
##     2 classes: 'NO', 'YES'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 10000, 10000, 10000, 10000, 10000, 10000, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.9140641  0.8216754
##
## Tuning parameter 'mtry' was held constant at a value of 17
```

```
p_rf_mtry_17<-predict(rf_mtry_17,newdata = test_small)

confMatrix.rf_mtry_17<-confusionMatrix(table(p_rf_mtry_17,
test_small$SIT_PROP_YN))

confMatrix.rf_mtry_17$overall["Accuracy"]
```

```
## Accuracy
##   0.9185
```

```
confMatrix.rf_mtry_17$byClass["F1"]
```

```
##        F1
## 0.8972387
```

2.5.4 Random Forest summary

The following data frame shows how each *random forest* try performed:

```
model.summary <- data.frame(Model = c("RF", "RF_CV", "RF_random_search",
"RF_mtry_14", "RF_mtry_17"),
                            Accuracy =
c(confMatrix.rf$overall["Accuracy"],
                              confMatrix.rf_cvt$overall["Accuracy"],
                              confMatrix.rf_random$overall["Accuracy"],
                              confMatrix.rf_mtry_14$overall["Accuracy"],
                              confMatrix.rf_mtry_17$overall["Accuracy"]),
                            Sensitivity =
c(confMatrix.rf$byClass["Sensitivity"],
                              confMatrix.rf_cvt$byClass["Sensitivity"],
                              confMatrix.rf_random$byClass["Sensitivity"],
                              confMatrix.rf_mtry_14$byClass["Sensitivity"],
                              confMatrix.rf_mtry_17$byClass["Sensitivity"]),
                            Specificity =
c(confMatrix.rf$byClass["Specificity"],
                              confMatrix.rf_cvt$byClass["Specificity"],
                              confMatrix.rf_random$byClass["Specificity"],
```

```
                      confMatrix.rf_mtry_14$byClass["Specificity"],
                      confMatrix.rf_mtry_17$byClass["Specificity"]),
                    Precision = c(confMatrix.rf$byClass["Precision"],
                      confMatrix.rf_cvt$byClass["Precision"],
                      confMatrix.rf_random$byClass["Precision"],
                      confMatrix.rf_mtry_14$byClass["Precision"],
                      confMatrix.rf_mtry_17$byClass["Precision"]),
                    Recall = c(confMatrix.rf$byClass["Recall"],
                      confMatrix.rf_cvt$byClass["Recall"],
                      confMatrix.rf_random$byClass["Recall"],
                      confMatrix.rf_mtry_14$byClass["Recall"],
                      confMatrix.rf_mtry_17$byClass["Recall"]),
                    F1 = c(confMatrix.rf$byClass["F1"],
                      confMatrix.rf_cvt$byClass["F1"],
                      confMatrix.rf_random$byClass["F1"],
                      confMatrix.rf_mtry_14$byClass["F1"],
                      confMatrix.rf_mtry_17$byClass["F1"]))



model.summary[order(-model.summary$Accuracy),]
```

```
##             Model Accuracy Sensitivity Specificity Precision    Recall
## 1              RF   0.9205   0.8981717   0.9350049 0.8997711 0.8981717
## 4      RF_mtry_14   0.9191   0.9078212   0.9264269 0.8890823 0.9078212
## 3RF_random_search   0.9188   0.9029964   0.9290663 0.8921224 0.9029964
## 5      RF_mtry_17   0.9185   0.9035043   0.9282415 0.8910594 0.9035043
## 2           RF_CV   0.9174   0.9151854   0.9188387 0.8798828 0.9151854
##          F1
## 1 0.8989706
## 4 0.8983541
## 3 0.8975265
## 5 0.8972387
## 2 0.8971870
```

As we can see, all Random Forest models performed very similarly. In the following section, **Results**, I will present the best model.

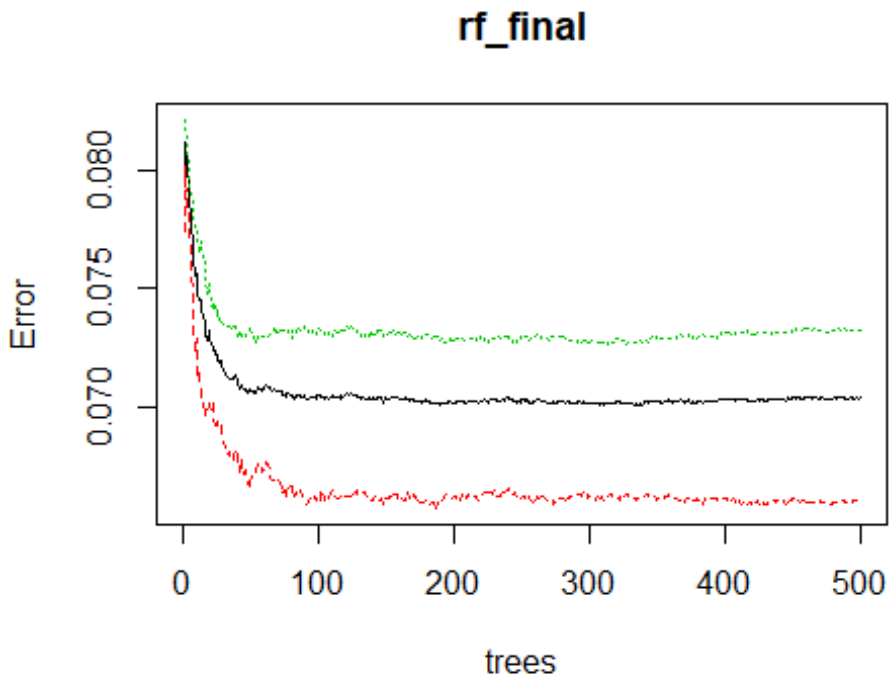
3 RESULTS

Applying chosen model to train_set and test_set

The model with best result in the *train_small* set was *randomForest without tuning.* I ran this model in train and test sets:

```
set.seed(1, sample.kind="Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform
'Rounding'
## sampler used

rf_final = randomForest(SIT_PROP_YN ~ .,data=train_set,ntree= 500,
importance=TRUE)

plot(rf_final)
```




```
p_rf_final<-predict(rf_final,newdata = test_set)

confMatrix.rf_final<-
confusionMatrix(table(p_rf_final,test_set$SIT_PROP_YN))

confMatrix.rf_final$overall["Accuracy"]

##  Accuracy
## 0.9288876

confMatrix.rf_final$byClass["Sensitivity"]

## Sensitivity
##   0.9320843

confMatrix.rf_final$byClass["Specificity"]

## Specificity
##   0.9267862
```

```
confMatrix.rf_final$byClass["Precision"]

## Precision
## 0.8932603

confMatrix.rf_final$byClass["Recall"]

##    Recall
## 0.9320843

confMatrix.rf_final$byClass["F1"]

##        F1
## 0.9122594
```

The proportion of approved proposals in the main set is *0.646*. So, quite simply, if I randomly choose a proposal from the list, the chance that the chosen proposal has been approved is *0.646*. The algorithm I developed increases this chance to *0.9288*, which I consider to be a satisfactory number after all the adjustments and tests performed. Besides that, I've got balanced numbers of *Sensitivity*, *Specificity*, *Precision* and *Recall*.

4 CONCLUSION

I chose my own project based on what I do at work. I didn't want something useless. After several weeks of hard work, I was able to get an *Accuracy* result of *0.8288*. To reach this, I had to learn a lot of things about R, including how to tidy data correctly to run models like *randomForest*.

Importantly, however, the technical specifications of my computer represented a limiting factor for my project. I've spent many hours waiting for a code to run. This got in the way of the project so much that I decided to create smaller training and testing sets to run my models.

In the end, I spent almost two weeks simply trying to run the model with higher results on the train and test sets, but my RStudio awlays crashed after some days. I ended up giving up and running a simpler code (with slightly worse performance).

Probably there are ways of optimize the execution of more advanced models even with a limited laptop as mine, but I don't have enough R knowledge to do this. As I practice using this powerful language, I am likely to learn this and produce even better results.

My conclusion, however, is that I developed a very good project, within the limits of my knowledge, that can be used as a decision making tool by anyone that uses **Siconv** and want to know which proposals are more likely to be approved.