Predict Grant Applications

Predict Grant Applications

Some information on the dataset:

> This task requires participants to predict the outcome of grant applications for the University of Melbourne.

Load R packages

```
# library(Amelia)
library(randomForest)
library(pROC)
library(caret)
library(stringi)
library(dplyr)
library(tidyr)
```

Reading-in the data

```
### Load Data
rawdata <- read.csv("unimelb_training.csv")

### Look at roles (in column names of 'rawdata')
roles  <- vector("character")

for (i in 1:15) {
  colstr = paste("Role.", as.character(i), sep = "")
  roles <- c(roles, as.vector(unique(rawdata[, colstr])))
}

### Get unique roles
roles <- unique(roles)

# Display 'roles'
sort(roles)
```

```
## [1] ""                          "CHIEF_INVESTIGATOR"
## [3] "DELEGATED_RESEARCHER"      "EXT_CHIEF_INVESTIGATOR"
## [5] "EXTERNAL_ADVISOR"          "HONVISIT"
## [7] "PRINCIPAL_SUPERVISOR"      "STUD_CHIEF_INVESTIGATOR"
## [9] "STUDRES"
```

Initial data cleaning of rawdata

```
### Look at missmap
Amelia::missmap(rawdata, rank.order = FALSE, x.cex = 0.01, y.cex = 0.01)
```


```
### Change long column name
names(rawdata)[names(rawdata) == "Contract.Value.Band...see.note.A"] <-
  "Contract.Value.Band"

### Convert columns from factors to character vectors for easier manipulation
rawdata$Sponsor.Code        <- as.character(rawdata$Sponsor.Code)
rawdata$Grant.Category.Code <- as.character(rawdata$Grant.Category.Code)
rawdata$Contract.Value.Band <- as.character(rawdata$Contract.Value.Band)


### Deal with missing values in these columns and re-transform to factors
# Sponsor.Code
rawdata$Sponsor.Code[rawdata$Sponsor.Code == ""] <- "Unk"
rawdata$Sponsor.Code <- factor(paste0("Sponsor", rawdata$Sponsor.Code))

# Grant.Category.Code
rawdata$Grant.Category.Code[rawdata$Grant.Category.Code == ""] <- "Unk"
rawdata$Grant.Category.Code <- factor(paste0("Grant.Category", rawdata$Grant.Category.Code))

# Contract.Value.Band
rawdata$Contract.Value.Band[rawdata$Contract.Value.Band == ""] <- "Unk"
rawdata$Contract.Value.Band <- factor(paste0("Contract.Value.Band", rawdata$Contract.Value.Band))
```

Create a data frame with vertical information, i.e. for each person in each grant

```
### Create list in global environment
temp.list <- vector("list", length = 15)

### Here we loop over all columns with numbers in their name, e.g. "RFCD.Code.1",
### and save the respective coumns for each number in a list of length 15.

for (i in 1:15) {

  # Get data for all columns for one person, e.g. "RFCD.Code.1", "RFCD.Percentage.1",
  # "Role.1", "With.PHD.1", etc. for the 'temp' data frame.
  # Additionally, the "Grant.Application.ID" is saved as well to preserve the identify
  # of the grants.
  temp <- rawdata[, c("Grant.Application.ID", grep(paste0("\\.", i, "$"),
                                                   names(rawdata), value = TRUE))]

  # Clean name of columns of temp
  names(temp) <- gsub(paste0("\\.", i, "$"), "", names(temp))

  # Initial step
  if (i == 1) temp.names <- names(temp)

  # This is needed as certain information is available only for the first five
  # researchers
  if (i > 5) {
    temp$RFCD.Code <- NA
    temp$RFCD.Percentage <- NA
    temp$SEO.Code <- NA
    temp$SEO.Percentage <- NA
  }
  # Put resulting data frame in the temp.list list elements
  # Use the column order of temp.names
  temp.list[[i]] <- temp[, temp.names]
}
```

Now, we have 15 data frames inside the temp.list which each data frame / list element containing the information about the following columns:

```
names(temp.list[[1]])
```

```
##  [1] "Grant.Application.ID"
##  [2] "RFCD.Code"
##  [3] "RFCD.Percentage"
##  [4] "SEO.Code"
##  [5] "SEO.Percentage"
##  [6] "Person.ID"
##  [7] "Role"
##  [8] "Year.of.Birth"
##  [9] "Country.of.Birth"
## [10] "Home.Language"
```

```
## [11] "Dept.No."
## [12] "Faculty.No."
## [13] "With.PHD"
## [14] "No..of.Years.in.Uni.at.Time.of.Grant"
## [15] "Number.of.Successful.Grant"
## [16] "Number.of.Unsuccessful.Grant"
## [17] "A."
## [18] "A"
## [19] "B"
## [20] "C"
```

Next, we want to build a data frame where the previously created data frames / list elements are situated below each other, i.e. in a vertical structure.

```
### Create vertical data frame in global environment
vertical <- data.frame()

### Loop through temp.list and create a data frame through rbind
### (using 'do.call' here will be faster than the loop)
for (i in 1:15) {
  vertical <- rbind(vertical, temp.list[[i]])
}

### Delete temp.list as all its information is now preserved in vertical
rm(temp.list)

# Look at resulting 'vertical' data frame
vertical %>%
  filter(Grant.Application.ID == 3) %>%
  .[1:10, 1:10]
```

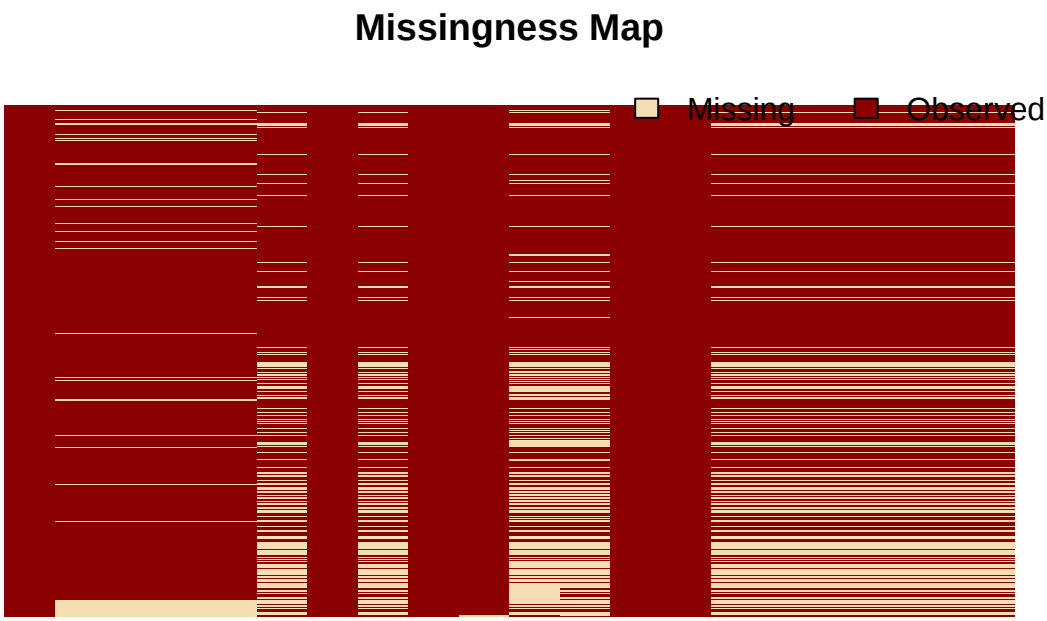
```
##    Grant.Application.ID RFCD.Code RFCD.Percentage SEO.Code SEO.Percentage
## 1                     3    321004              60   730105             60
## 2                     3    321216              40   730207             40
## 3                     3         0               0        0              0
## 4                     3         0               0        0              0
## 5                     3         0               0        0              0
## 6                     3        NA              NA       NA             NA
## 7                     3        NA              NA       NA             NA
## 8                     3        NA              NA       NA             NA
## 9                     3        NA              NA       NA             NA
## 10                    3        NA              NA       NA             NA
##    Person.ID                   Role Year.of.Birth Country.of.Birth
## 1       5967     CHIEF_INVESTIGATOR          1955        Australia
## 2      27307     CHIEF_INVESTIGATOR          1950        Australia
## 3      79652     CHIEF_INVESTIGATOR          1950     Asia Pacific
## 4      11667   DELEGATED_RESEARCHER          1950        Australia
## 5         NA EXT_CHIEF_INVESTIGATOR            NA
## 6         NA EXT_CHIEF_INVESTIGATOR            NA
## 7         NA EXT_CHIEF_INVESTIGATOR            NA
## 8         NA                                   NA
## 9         NA                                   NA
```

```
## 10        NA                                      NA
##    Home.Language
## 1
## 2
## 3
## 4
## 5
## 6
## 7
## 8
## 9
## 10
```

```
### Look only at subset with existing role, i.e. actual researchers
vertical <- subset(vertical, subset = Role != "")
```

Now, we can have a look at the amount of missing values again in a missmap. We can see that we are already dealing with a lot less missing values as we did before with `rawdata`:

```
### Look at missmap again
Amelia::missmap(vertical, rank.order = FALSE, x.cex = 0.01, y.cex = 0.01)
```



Remove spaces and set missing values to NA

```
table(vertical$Role)
```

```
##
##                             CHIEF_INVESTIGATOR    DELEGATED_RESEARCHER
##                          0                12136                     315
##  EXT_CHIEF_INVESTIGATOR        EXTERNAL_ADVISOR                HONVISIT
##                    3732                       5                      29
##    PRINCIPAL_SUPERVISOR STUD_CHIEF_INVESTIGATOR                 STUDRES
##                     536                     586                     157
```

```
table(vertical$Year.of.Birth)
```

```
##
## 1900 1925 1930 1935 1940 1945 1950 1955 1960 1965 1970 1975 1980 1985
##    6    3   16   49  289  734 1578 1758 2487 2361 1900 1393  445   11
```

Set "Role" and "Year.of.Birth" to factor, thereby also removing the obsolute "" level:

```
### Role
# Remove unwanted level ""
vertical$Role <- factor(as.character(vertical$Role))

### Year.of.Birth
# Convert to factor
vertical$Year.of.Birth <- factor(vertical$Year.of.Birth)
```

Further data cleaning:

```
### Country.of.Birth
# Replace space with dots
vertical$Country.of.Birth <- stri_replace_all_regex(as.character(vertical$Country.of.Birth),
                                                    pattern = " ", replacement = "_")

# Set proper NA values
vertical$Country.of.Birth[vertical$Country.of.Birth == ""] <- NA

# Previous step made column a character vector, but we need a factor
vertical$Country.of.Birth <- factor(vertical$Country.of.Birth)

### For Home.Language
# Set proper NA values
vertical$Home.Language[vertical$Home.Language == ""] <- NA

# Previous step made column a character vector, but we need a factor
vertical$Home.Language <- factor(vertical$Home.Language)

### Dept.No.
# Replace integers with characters, set NAs and the vector as a factor
```

```
vertical$Dept.No.                                  <- paste0("Dept", vertical$Dept.No.)
vertical$Dept.No.[vertical$Dept.No. == "DeptNA"] <- NA
vertical$Dept.No.                                  <- factor(vertical$Dept.No.)

### Faculty.No.
# Same as for Dept.No.
vertical$Faculty.No.                                        <- paste0("Faculty",
                                                                      vertical$Faculty.No.)
vertical$Faculty.No.[vertical$Faculty.No. == "FacultyNA"] <- NA
vertical$Faculty.No.                                        <- factor(vertical$Faculty.No.)

### RFCD.Code
# Same as previous steps.
# Also replace certain frequent codes with NA
vertical$RFCD.Code                                            <- paste0("RFCD",
                                                                        vertical$RFCD.Code)
vertical$RFCD.Percentage[vertical$RFCD.Code == "RFCDNA"]     <- NA
vertical$RFCD.Code[vertical$RFCD.Code == "RFCDNA"]           <- NA
vertical$RFCD.Percentage[vertical$RFCD.Code == "RFCD0"]      <- NA
vertical$RFCD.Code[vertical$RFCD.Code == "RFCD0"]            <- NA
vertical$RFCD.Percentage[vertical$RFCD.Code == "RFCD999999"] <- NA
vertical$RFCD.Code[vertical$RFCD.Code == "RFCD999999"]       <- NA
vertical$RFCD.Code                                            <- factor(vertical$RFCD.Code)

### SEO.Code
# As the previous step
vertical$SEO.Code                                          <- paste0("SEO",
                                                                     vertical$SEO.Code)
vertical$SEO.Percentage[vertical$SEO.Code == "SEONA"]     <- NA
vertical$SEO.Code[vertical$SEO.Code == "SEONA"]           <- NA
vertical$SEO.Percentage[vertical$SEO.Code == "SEO0"]      <- NA
vertical$SEO.Code[vertical$SEO.Code  == "SEO0"]           <- NA
vertical$SEO.Percentage[vertical$SEO.Code == "SEO999999"] <- NA
vertical$SEO.Code[vertical$SEO.Code == "SEO999999"]       <- NA
vertical$SEO.Code                                          <- factor(vertical$SEO.Code)

### No..of.Years.in.Uni.at.Time.of.Grant
# The levels used here are full of space and hence unwieldy -
# Change to more appropiate levels
table(vertical$No..of.Years.in.Uni.at.Time.of.Grant)
```

```
##
##                 >=0 to 5    >10 to 15     >5 to 10  Less than 0
##         6196         4191         1490         2472         1546
## more than 15
##         1601
```

```
vertical$No..of.Years.in.Uni.at.Time.of.Grant <-
  as.character(vertical$No..of.Years.in.Uni.at.Time.of.Grant)

vertical$No..of.Years.in.Uni.at.Time.of.Grant[
  vertical$No..of.Years.in.Uni.at.Time.of.Grant == ""] <- "DurationUnk"

vertical$No..of.Years.in.Uni.at.Time.of.Grant[
```

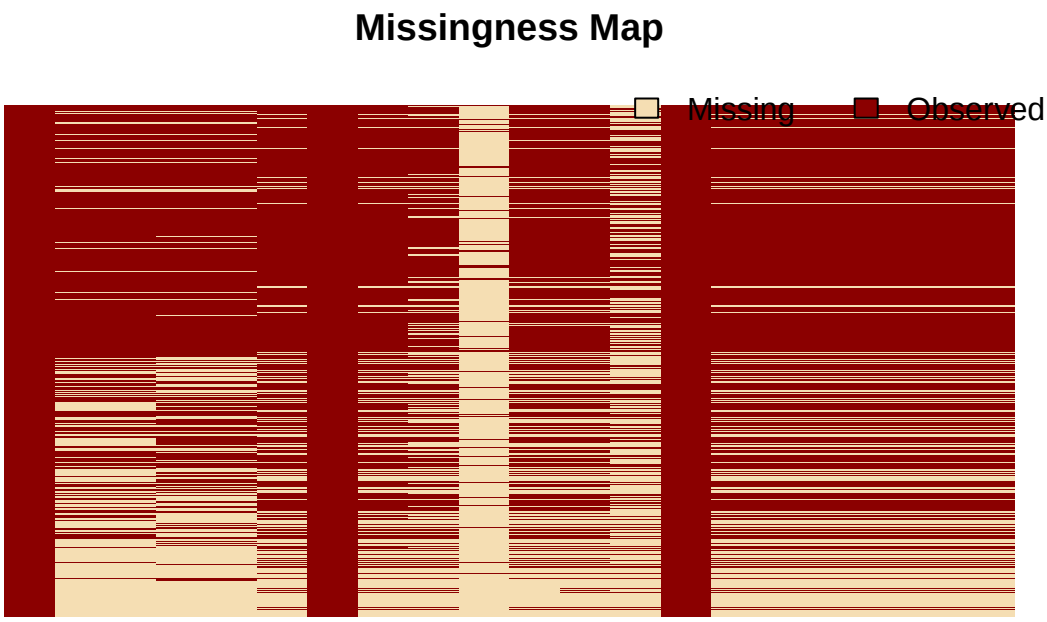
```
  vertical$No..of.Years.in.Uni.at.Time.of.Grant == ">=0 to 5"] <- "Duration0to5"

vertical$No..of.Years.in.Uni.at.Time.of.Grant[
  vertical$No..of.Years.in.Uni.at.Time.of.Grant == ">5 to 10"] <- "Duration5to10"

vertical$No..of.Years.in.Uni.at.Time.of.Grant[
  vertical$No..of.Years.in.Uni.at.Time.of.Grant == ">10 to 15"] <- "Duration10to15"

vertical$No..of.Years.in.Uni.at.Time.of.Grant[
  vertical$No..of.Years.in.Uni.at.Time.of.Grant == "more than 15"] <- "DurationGT15"

vertical$No..of.Years.in.Uni.at.Time.of.Grant[
  vertical$No..of.Years.in.Uni.at.Time.of.Grant == "Less than 0"] <- "DurationLT0"

# Set vector to factors again
vertical$No..of.Years.in.Uni.at.Time.of.Grant <-
  factor(vertical$No..of.Years.in.Uni.at.Time.of.Grant)

### With.PHD
# Again, NAs and factor
vertical$With.PHD[vertical$With.PHD == ""]  <- NA
vertical$With.PHD <- factor(vertical$With.PHD)
```

We write a short function to shorten the role titles:

```
### A short function to replace role titles
shortNames <- function(x, pre = "")
{
  x <- gsub("EXT_CHIEF_INVESTIGATOR",  "ECI", x)
  x <- gsub("STUD_CHIEF_INVESTIGATOR", "SCI", x)
  x <- gsub("CHIEF_INVESTIGATOR",      "CI", x)
  x <- gsub("DELEGATED_RESEARCHER",    "DR", x)
  x <- gsub("EXTERNAL_ADVISOR",        "EA", x)
  x <- gsub("HONVISIT",                "HV", x)
  x <- gsub("PRINCIPAL_SUPERVISOR",    "PS", x)
  x <- gsub("STUDRES",                 "SR", x)
  x <- gsub("Unk",                     "UNK", x)
  # Contains all names that are not Grant.Application.ID
  other <- x[x != "Grant.Application.ID"]
  # Returns all replaced Names
  c("Grant.Application.ID", paste(pre, other, sep = ""))
}
```

After some more data cleaning, we can again look at the missmap of `vertical`:

```
### Look at the missmap again
Amelia::missmap(vertical, rank.order = FALSE, x.cex = 0.01, y.cex = 0.01)
```



Creation of additional features of existing non-grant specific data

Now we can create (engineer) additional columns (features) which contain aggregate information on data which is not directly grant-specific:

Role

```
### Calculate how many people worked on each grant
people <-
  vertical %>%
  group_by(Grant.Application.ID) %>%
  summarise(people.count = n())

head(people)
```

```
## # A tibble: 6 × 2
##   Grant.Application.ID people.count
##                  <int>        <int>
## 1                    1            1
## 2                    2            1
## 3                    3            7
## 4                    4            6
```

```
## 5                    5            1
## 6                    6            1
```

```
### Calculate how many people of each role participated in grant request
role.count <-
  vertical %>%
  group_by(Grant.Application.ID, Role) %>%
  summarise(role.count = n())

head(role.count)
```

```
## Source: local data frame [6 x 3]
## Groups: Grant.Application.ID [4]
##
##   Grant.Application.ID                 Role role.count
##                  <int>               <fctr>      <int>
## 1                    1     CHIEF_INVESTIGATOR          1
## 2                    2     CHIEF_INVESTIGATOR          1
## 3                    3     CHIEF_INVESTIGATOR          3
## 4                    3   DELEGATED_RESEARCHER          1
## 5                    3 EXT_CHIEF_INVESTIGATOR          3
## 6                    4     CHIEF_INVESTIGATOR          1
```

```
role.count <-
  role.count %>%
  tidyr::spread(data = ., key = Role, value = role.count, fill = 0)

### Shorten the names
names(role.count) <- shortNames(names(role.count))

head(role.count)
```

```
## Source: local data frame [6 x 9]
## Groups: Grant.Application.ID [6]
##
##   Grant.Application.ID    CI    DR   ECI    EA    HV    PS   SCI    SR
##                  <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1                    1     1     0     0     0     0     0     0     0
## 2                    2     1     0     0     0     0     0     0     0
## 3                    3     3     1     3     0     0     0     0     0
## 4                    4     1     0     3     0     0     1     1     0
## 5                    5     1     0     0     0     0     0     0     0
## 6                    6     1     0     0     0     0     0     0     0
```

Year.of.Birth

```
### Calcuate amount of people on grant according to (only) DOB
people.DOB <-
  vertical %>%
  group_by(Grant.Application.ID, Year.of.Birth) %>%
  summarise(count = n()) %>%
  tidyr::spread(key = Year.of.Birth, value = count, fill = 0)

names(people.DOB)[-1] <- paste("year", names(people.DOB)[-1], sep = "_")

head(people.DOB)
```

```
## Source: local data frame [6 x 16]
## Groups: Grant.Application.ID [6]
##
##   Grant.Application.ID year_1900 year_1925 year_1930 year_1935 year_1940
##                  <int>     <dbl>     <dbl>     <dbl>     <dbl>     <dbl>
## 1                    1         0         0         0         0         0
## 2                    2         0         0         0         0         0
## 3                    3         0         0         0         0         0
## 4                    4         0         0         0         0         0
## 5                    5         0         0         0         0         0
## 6                    6         0         0         0         0         0
## # ... with 10 more variables: year_1945 <dbl>, year_1950 <dbl>,
## #   year_1955 <dbl>, year_1960 <dbl>, year_1965 <dbl>, year_1970 <dbl>,
## #   year_1975 <dbl>, year_1980 <dbl>, year_1985 <dbl>, `year_<NA>` <dbl>
```

```
### Calcuate amount of people on grant according to role and DOB
role.DOB <-
  vertical %>%
  group_by(Grant.Application.ID, Role, Year.of.Birth) %>%
  summarise(people.count = n())

head(role.DOB)
```

```
## Source: local data frame [6 x 4]
## Groups: Grant.Application.ID, Role [5]
##
##   Grant.Application.ID                  Role Year.of.Birth people.count
##                  <int>                <fctr>        <fctr>        <int>
## 1                    1     CHIEF_INVESTIGATOR          1965            1
## 2                    2     CHIEF_INVESTIGATOR          1960            1
## 3                    3     CHIEF_INVESTIGATOR          1950            2
## 4                    3     CHIEF_INVESTIGATOR          1955            1
## 5                    3   DELEGATED_RESEARCHER          1950            1
## 6                    3 EXT_CHIEF_INVESTIGATOR            NA            3
```

```
role.DOB <-
  role.DOB %>%
  tidyr::unite(col = YOB_Role, Role, Year.of.Birth, sep = "_") %>%
  tidyr::spread(key = YOB_Role, value = people.count, fill = 0)

### Shorten names
names(role.DOB) <- shortNames(names(role.DOB))

head(role.DOB)[, 1:7]
```

```
## Source: local data frame [6 x 7]
## Groups: Grant.Application.ID [6]
##
##   Grant.Application.ID CI_1900 CI_1925 CI_1930 CI_1935 CI_1940 CI_1945
##                  <int>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
## 1                    1       0       0       0       0       0       0
## 2                    2       0       0       0       0       0       0
## 3                    3       0       0       0       0       0       0
## 4                    4       0       0       0       0       0       0
```

```
## 5                   5       0       0       0       0       0       0
## 6                   6       0       0       0       0       0       0
```

Country.of.Birth

```
### Calculate amount of people on grant according to role and Country of birth
role.COB <-
  vertical %>%
  group_by(Grant.Application.ID, Role, Country.of.Birth) %>%
  summarise(count = n()) %>%
  tidyr::unite(col = COB_Role, Role, Country.of.Birth, sep = "_") %>%
  tidyr::spread(key = COB_Role, value = count, fill = 0)

### Shorten names
names(role.COB) <- shortNames(names(role.COB))
```

Home.Language

```
### Calculate amount of people on grant according to role and Home.Language
role.lang <-
  vertical %>%
  group_by(Grant.Application.ID, Role, Home.Language) %>%
  summarise(count = n()) %>%
  tidyr::unite(col = Lang_Role, Role, Home.Language, sep = "_") %>%
  tidyr::spread(key = Lang_Role, value = count, fill = 0)

### Shorten names
names(role.lang) <- shortNames(names(role.lang))

### Calculate amount of people on grant according to (only) Home.Language
people.lang <-
  vertical %>%
  group_by(Grant.Application.ID, Home.Language) %>%
  summarise(count = n()) %>%
  tidyr::spread(key = Home.Language, value = count, fill = 0) %>%
  # Don't select 'NA' column
  select(Grant.Application.ID, English, Other)
```

Dept.No.

```
### Calculate amount of people on grant according to role and Dept.No.
role.dept <-
  vertical %>%
  group_by(Grant.Application.ID, Role, Dept.No.) %>%
  summarise(count = n()) %>%
  tidyr::unite(col = Dept_Role, Role, Dept.No., sep = "_") %>%
  tidyr::spread(key = Dept_Role, value = count, fill = 0)

### Shorten names
names(role.dept) <- shortNames(names(role.dept))

### Calculate amount of people on grant according to (only) Dept.No.
people.dept <-
```

```
  vertical %>%
  group_by(Grant.Application.ID, Dept.No.) %>%
  summarise(count = n()) %>%
  tidyr::spread(key = Dept.No., value = count, fill = 0)
```

Faculty.No.

```
### Calculate amount of people on grant according to role and Faculty.No.
role.faculty <-
  vertical %>%
  group_by(Grant.Application.ID, Role, Faculty.No.) %>%
  summarise(count = n()) %>%
  tidyr::unite(col = Fact_Role, Role, Faculty.No., sep = "_") %>%
  tidyr::spread(key = Fact_Role, value = count, fill = 0)

### Shorten names
names(role.faculty) <- shortNames(names(role.faculty))

### Calculate amount of people on grant according to (only) Faculty.No.
people.faculty <-
  vertical %>%
  group_by(Grant.Application.ID, Faculty.No.) %>%
  summarise(count = n()) %>%
  tidyr::spread(key = Faculty.No., value = count, fill = 0)
```

With.PHD

```
### Calculate amount of people on grant according to role and With.PHD
role.phd <-
  vertical %>%
  group_by(Grant.Application.ID, Role, With.PHD) %>%
  summarise(count = n()) %>%
  tidyr::unite(col = PHD_Role, Role, With.PHD, sep = "_") %>%
  tidyr::spread(key = PHD_Role, value = count, fill = 0)

### Shorten names
names(role.phd) <- shortNames(names(role.phd))

### Remove columns with NA in title
role.phd <- role.phd[, !names(role.phd) %in% grep("_NA", names(role.phd), value = TRUE)]

### Calculate amount of people on grant according with.PHD
people.phd <-
  vertical %>%
  group_by(Grant.Application.ID, With.PHD) %>%
  summarise(count = n()) %>%
  tidyr::spread(key = With.PHD, value = count, fill = 0)

### Remove columns with NA in title
people.phd <- people.phd[, c(1, 2)]

### Set "Yes " to "PHD"
names(people.phd)[2]  <- "PhD"
```

Number.of.Successful.Grant & Number.of.Unsuccessful.Grant

```
### Calculate sum of successfull grants per grant ID
people.grants.success <-
  vertical %>%
  group_by(Grant.Application.ID) %>%
  summarise(Sum.Number.of.Successful.Grant = sum(Number.of.Successful.Grant, na.rm = TRUE))

### Calculate sum of unsuccessfull grants per grant ID (same as before)
people.grants.fails <-
  vertical %>%
  group_by(Grant.Application.ID) %>%
  summarise(Sum.Number.of.Unsuccessful.Grant = sum(Number.of.Unsuccessful.Grant, na.rm = TRUE))

### Calculate sum of successfull grants per grant ID and
### role (as before but with role)
role.grants.success <-
  vertical %>%
  group_by(Grant.Application.ID, Role) %>%
  summarise(Sum.Number.of.Successful.Grant = sum(Number.of.Successful.Grant, na.rm = TRUE)) %>%
  tidyr::spread(key = Role, value = Sum.Number.of.Successful.Grant, fill = 0)

### Shorten names
names(role.grants.success) <- shortNames(names(role.grants.success))

### Calculate sum of unsuccessfull grants per grant ID and
### role (as before but with role)
role.grants.fails <-
  vertical %>%
  group_by(Grant.Application.ID, Role) %>%
  summarise(Sum.Number.of.Unsuccessful.Grant = sum(Number.of.Unsuccessful.Grant, na.rm = TRUE)) %>%
  tidyr::spread(key = Role, value = Sum.Number.of.Unsuccessful.Grant, fill = 0)

### Shorten names
names(role.grants.fails) <- shortNames(names(role.grants.fails))
```

No..of.Years.in.Uni.at.Time.of.Grant

```
### Calculate No..of.Years.in.Uni.at.Time.of.Grant per role & grant
role.uniyears <-
  vertical %>%
  group_by(Grant.Application.ID, Role, No..of.Years.in.Uni.at.Time.of.Grant) %>%
  summarise(count = n()) %>%
  tidyr::unite(col = Role_Years, Role, No..of.Years.in.Uni.at.Time.of.Grant, sep = "_") %>%
  tidyr::spread(key = Role_Years, value = count, fill = 0)


### Shorten names
names(role.uniyears) <- shortNames(names(role.uniyears))

### Calculate No..of.Years.in.Uni.at.Time.of.Grant per grant
### (as before)
people.uniyears <-
  vertical %>%
```

```
  group_by(Grant.Application.ID, No..of.Years.in.Uni.at.Time.of.Grant) %>%
  summarise(count = n()) %>%
  tidyr::spread(key = No..of.Years.in.Uni.at.Time.of.Grant, value = count, fill = 0)
```

RFCD.Code

```
### Calculate amount of successfull grants per role & RFDC Code
role.rfcd.code <-
  vertical %>%
  mutate(RFCD.Code = substr(RFCD.Code, 1, 6)) %>%
  group_by(Grant.Application.ID, Role, RFCD.Code) %>%
  summarise(count = n()) %>%
  unite(Role_RFCD.Code, Role, RFCD.Code, sep = "_") %>%
  tidyr::spread(key = Role_RFCD.Code, value = count, fill = 0)

### Shorten names
names(role.rfcd.code) <- shortNames(names(role.rfcd.code))

### Calculate amount of successfull grants per RFDC Code
people.rfcd.code <-
  vertical %>%
  mutate(RFCD.Code = substr(RFCD.Code, 1, 6)) %>%
  group_by(Grant.Application.ID, RFCD.Code) %>%
  summarise(count = n()) %>%
  tidyr::spread(key = RFCD.Code, value = count, fill = 0)
```

SEO.Code

```
### Calculate amount of successfull grants per role AND SEO Code (as before)
role.seo.code <-
  vertical %>%
  mutate(SEO.Code = substr(SEO.Code, 1, 6)) %>%
  group_by(Grant.Application.ID, Role, SEO.Code) %>%
  summarise(count = n()) %>%
  unite(Role_SEO.Code, Role, SEO.Code, sep = "_") %>%
  tidyr::spread(key = Role_SEO.Code, value = count, fill = 0)

### Shorten names
names(role.seo.code) <- shortNames(names(role.seo.code))

### Calculate amount of successfull grants per SEO Code
people.seo.code <-
  vertical %>%
  mutate(SEO.Code = substr(SEO.Code, 1, 6)) %>%
  group_by(Grant.Application.ID, SEO.Code) %>%
  summarise(count = n()) %>%
  tidyr::spread(key = SEO.Code, value = count, fill = 0)
```

Number of publications per journal

```
### Role + type
role.pub <-
```

```
  vertical %>%
  group_by(Grant.Application.ID, Role) %>%
  summarise(sum_A1 = sum(A., na.rm = TRUE),
            sum_A  = sum(A, na.rm = TRUE),
            sum_B  = sum(B, na.rm = TRUE),
            sum_C  = sum(C, na.rm = TRUE))

role.pub.A1 <-
  role.pub %>%
  select(Grant.Application.ID, Role, sum_A1) %>%
  tidyr::spread(key = Role, value = sum_A1, fill = 0)

role.pub.A <-
  role.pub %>%
  select(Grant.Application.ID, Role, sum_A) %>%
  tidyr::spread(key = Role, value = sum_A, fill = 0)

role.pub.B <-
  role.pub %>%
  select(Grant.Application.ID, Role, sum_B) %>%
  tidyr::spread(key = Role, value = sum_B, fill = 0)

role.pub.C <-
  role.pub %>%
  select(Grant.Application.ID, Role, sum_C) %>%
  tidyr::spread(key = Role, value = sum_C, fill = 0)

### Shorten names and add prefix of publication type
names(role.pub.A1) <- shortNames(names(role.pub.A1))
names(role.pub.A1)[-1] <- paste("Astar", names(role.pub.A1)[-1], sep="_")

names(role.pub.A) <- shortNames(names(role.pub.A))
names(role.pub.A)[-1] <- paste("A", names(role.pub.A)[-1], sep="_")

names(role.pub.B) <- shortNames(names(role.pub.B))
names(role.pub.B)[-1] <- paste("B", names(role.pub.B)[-1], sep="_")

names(role.pub.C) <- shortNames(names(role.pub.C))
names(role.pub.C)[-1] <- paste("C", names(role.pub.C)[-1], sep="_")

### Number of publication just per type and grant
people.pub.A1 <- data.frame(Grant.Application.ID = role.pub.A1$Grant.Application.ID,
                            A1 = rowSums(role.pub.A1[, -1]))

people.pub.A <- data.frame(Grant.Application.ID = role.pub.A$Grant.Application.ID,
                           A = rowSums(role.pub.A[, -1]))

people.pub.B <- data.frame(Grant.Application.ID = role.pub.B$Grant.Application.ID,
                           B = rowSums(role.pub.B[, -1]))

people.pub.C <- data.frame(Grant.Application.ID =  role.pub.C$Grant.Application.ID,
                           C = rowSums(role.pub.C[, -1]))
```

```
### Create a DF with all publication data
people.pub <- left_join(people.pub.A1, people.pub.A, by = "Grant.Application.ID")
people.pub <- left_join(people.pub, people.pub.B, by = "Grant.Application.ID")
people.pub <- left_join(people.pub, people.pub.C, by = "Grant.Application.ID")
```

RFCD & SEO percentages

```
### Calculate sum of percentages
RFCD.Percentage <-
  vertical %>%
  group_by(Grant.Application.ID) %>%
  summarise(RFCD.Percentage = sum(RFCD.Percentage, na.rm = TRUE))

### Same as RFCF for SEO percentage
SEO.Percentage <-
  vertical %>%
  group_by(Grant.Application.ID) %>%
  summarise(SEO.Percentage = sum(SEO.Percentage, na.rm = TRUE))
```

Engineering of additional features for grant-specific data

```
### Create grant data
grantData <- rawdata[, c("Contract.Value.Band", "Grant.Category.Code")]

### Make a lubridate object for the time, then derive the day, week, month
### and year info
startTime <- lubridate::dmy(rawdata$Start.date)

grantData$Month <- factor(as.character(lubridate::month(startTime, label = TRUE)))
grantData$Weekday <- factor(as.character(lubridate::wday(startTime, label = TRUE)))
grantData$Day <- lubridate::day(startTime)
grantData$Year <- lubridate::year(startTime)

### Create dummies
dummies <- dummyVars(~., data = grantData, levelsOnly = TRUE)

### Recreate grant data with dummies
grantData <- as.data.frame(predict(dummies, grantData))

### Remove spaces from column names
names(grantData) <- gsub(" ", "", names(grantData))

### Create class depending on grant status
grantData$Class <- factor(ifelse(rawdata$Grant.Status, "successful",
                                 "unsuccessful"))

### Get Application.ID from rawdata
grantData$Grant.Application.ID <- rawdata$Grant.Application.ID

## Create function to find and remove zero-variance ("ZV") predictors
noZV <- function(x)
{
```

```
  keepers <- unlist(lapply(x, function(x) length(unique(x)) > 1))
  x[, keepers, drop = FALSE]
}

### Remove zero variance predictors from grantData
grantData <- noZV(grantData)
```

Join predictors together

```
### Join predictors together in separate steps
summarized <- dplyr::left_join(people, role.count, by = "Grant.Application.ID")
summarized <- dplyr::left_join(summarized, people.DOB[, -length(people.DOB)],
                               by = "Grant.Application.ID")
summarized <- dplyr::left_join(summarized, people.lang, by = "Grant.Application.ID")
summarized <- dplyr::left_join(summarized, people.phd, by = "Grant.Application.ID")
summarized <- dplyr::left_join(summarized, people.grants.success, by = "Grant.Application.ID")
summarized <- dplyr::left_join(summarized, people.grants.fails, by = "Grant.Application.ID")
summarized <- dplyr::left_join(summarized, people.dept[, -length(people.dept)],
                               by = "Grant.Application.ID")
summarized <- dplyr::left_join(summarized, people.uniyears, by = "Grant.Application.ID")
summarized <- dplyr::left_join(summarized, people.pub, by = "Grant.Application.ID")
summarized <- dplyr::left_join(summarized, people.rfcd.code[, -length(people.rfcd.code)],
                               by = "Grant.Application.ID")
summarized <- dplyr::left_join(summarized, people.seo.code[, -length(people.seo.code)],
                               by = "Grant.Application.ID")
summarized <- dplyr::left_join(summarized, RFCD.Percentage, by = "Grant.Application.ID")
summarized <- dplyr::left_join(summarized, SEO.Percentage, by = "Grant.Application.ID")

summarized <- dplyr::left_join(summarized, grantData, by="Grant.Application.ID")

### Remove Grant.Application.ID (not needed anymore)
summarized$Grant.Application.ID <- NULL
```

Prediction & evaluation

```
### We want to use the date before 2008 to predict the outcomes in 2008:

### Create training data before 2008
train.data <- summarized[summarized$Year < 2008, ]

### Remaining data from 2008
summarized.2008 <- summarized[summarized$Year == 2008, ]


###### Create test and hold out set
### Test set: 25% of 2008 data
### Hold-out set: 75% of 2008 data

### Setting the seed
set.seed(75)
test.data <- summarized.2008[sample(nrow(summarized.2008),
```

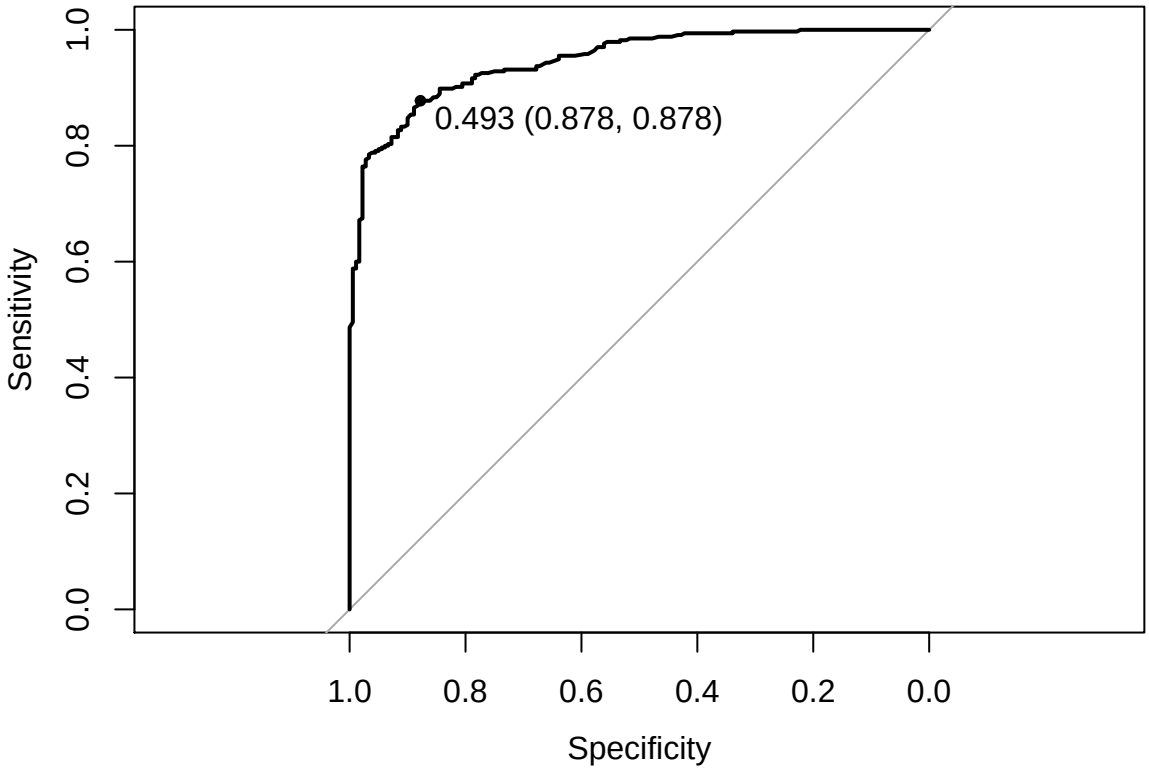
```
                                    size = round(nrow(summarized.2008) *
                                                   0.25)), ]
hold.out <- setdiff(summarized.2008, test.data)

### Remove previous 2008 data
rm(summarized.2008)

### Remove year data
train.data$Year <- NULL
test.data$Year <- NULL
hold.out$Year <- NULL
```

Train Model

```
### Train random forest model on train.data
rf <- randomForest(formula = Class ~ ., data = train.data, importance = TRUE)

### Save importance of random forest
vaImp.rf <- varImp(rf, verbose = FALSE)

### Prediction on test data
result.predicted.test <- predict(rf, test.data, type = "prob")

### Get ROC result
result.roc <-  roc(test.data$Class, result.predicted.test[, 1])

#### Result of model & plot

### Result
# Depending on the seed the result (area under the curve) should be around 0.93.
result.roc
```

```
##
## Call:
## roc.default(response = test.data$Class, predictor = result.predicted.test[,     1])
##
## Data: result.predicted.test[, 1] in 180 controls (test.data$Class successful) > 335 cases (test.data$
## Area under the curve: 0.9467
```

```
### Plot result
plot(result.roc, print.thres = "best", print.thres.best.method = "closest.topleft")
```




```
##
## Call:
## roc.default(response = test.data$Class, predictor = result.predicted.test[,     1])
##
## Data: result.predicted.test[, 1] in 180 controls (test.data$Class successful) > 335 cases (test.data$
## Area under the curve: 0.9467
```