HarvardX project - Classification Model

Luca Aceti

02/10/2022

Introduction

The project is an application of machine learning in the contest of Pharmaceutical sector. The aim is is to understand the persistency of drug, with the objective to gather insights on the factors that are impacting the persistency.For this project we use the database "Classification: Persistent vs Non-Persistent" downloaded from Kaggle datasets (https://www.kaggle.com/datasets/harbhajansingh21/persistent-vs-nonpersistent). First of all a preliminary exploratory analysis is done with a different approach for categorical and non-categorical covariates.Then, after splitting the dataset between training and test set, different models are evaluated in order to choose the best one, considering Accuracy, Area Under Curve ROc and other metrics. After a comparison of the models, we try to improve the results using different techniques (ensemble, stacking, cost-sensitive models).

Data preparation

After the upload of the main R libraries, the first step of the analysis has been the download of the file from Kaggle. For the download it has been necessary to create an account in Kaggle and to achieve the activity after logging in with your credentials. In order to have an easy access to the dataset, I copy it on my Github (https://github.com/Luca-19/HarvardX-project---Classification-Model/blob/main/Persistent_vs_NonPersistent.csv).

```
# suppress the warnings visualization for the readability of the report
options(warn = -1)

# upload of the packages
if (!require(tidyverse)) install.packages('tidyverse')
library(tidyverse)
if (!require(caret)) install.packages('caret')
library(caret)
if (!require(data.table)) install.packages('data.table')
library(data.table)
if (!require(httr)) install.packages('httr')
library(httr)

# dataset url: https://www.kaggle.com/datasets/harbhajansingh21/persistent-vs-nonpersistent
# file download: https://raw.githubusercontent.com/Luca-19/HarvardX-project---Classification-Model/main/

dl <- tempfile()

download.file("https://raw.githubusercontent.com/Luca-19/HarvardX-project---Classification-Model/main/P
dataset<-read.csv(dl)
```

```r
# packages required by caret for the models
if (!require(rpart)) install.packages('rpart') #rpart
library(rpart)
if (!require(randomForest)) install.packages('randomForest') #random forest
library(randomForest)
if (!require(kernlab)) install.packages('kernlab') #svm
library(kernlab)
if (!require(glmnet)) install.packages('glmnet') #glmnet
library(glmnet)
if (!require(pROC)) install.packages('pROC') #pROC package used for ROC curve
library(pROC)
if (!require(caretEnsemble)) install.packages('caretEnsemble') #caretEnsemble package for stacking the
library(caretEnsemble)

#other specific packages
if (!require(fastDummies)) install.packages('fastDummies') # to create dummy variables
library(fastDummies)
if (!require(corrplot)) install.packages('corrplot') # to plot the correlations among predictors
library(corrplot)

# restore the warnings visualization
options(warn=0)
```

Exploratory data analysis

The dataset is composed of 69 variables and 3424 observations. From the summary of the dataset it is possible to see that 67 variables are categorical, while only 2 are numerical.The outcome is categorical: Persistency_Flag, with possible values of Persistent e Non-Persistent. A more detailed description of each variable is contained in the aboved citated link, where you find a distinction among description variables, provider attributes, Clinical Factors and Disease/Treatment Factor.

```r
# dataset structure
str(dataset)
```

```
## 'data.frame':    3424 obs. of  69 variables:
##  $ Ptid                                                                : chr  "P1" "P2" "P3" "P4" ...
##  $ Persistency_Flag                                                    : chr  "Persistent" "Non-Persis
##  $ Gender                                                              : chr  "Male" "Male" "Female" "
##  $ Race                                                                : chr  "Caucasian" "Asian" "Oth
##  $ Ethnicity                                                           : chr  "Not Hispanic" "Not Hisp
##  $ Region                                                              : chr  "West" "West" "Midwest"
##  $ Age_Bucket                                                          : chr  ">75" "55-65" "65-75" ">
##  $ Ntm_Speciality                                                      : chr  "GENERAL PRACTITIONER" "
##  $ Ntm_Specialist_Flag                                                 : chr  "Others" "Others" "Other
##  $ Ntm_Speciality_Bucket                                               : chr  "OB/GYN/Others/PCP/Unkno
##  $ Gluco_Record_Prior_Ntm                                              : chr  "N" "N" "N" "N" ...
##  $ Gluco_Record_During_Rx                                              : chr  "N" "N" "N" "Y" ...
##  $ Dexa_Freq_During_Rx                                                 : int  0 0 0 0 0 0 2 0 0 0 ...
##  $ Dexa_During_Rx                                                      : chr  "N" "N" "N" "N" ...
##  $ Frag_Frac_Prior_Ntm                                                 : chr  "N" "N" "N" "N" ...
##  $ Frag_Frac_During_Rx                                                 : chr  "N" "N" "N" "N" ...
##  $ Risk_Segment_Prior_Ntm                                              : chr  "VLR_LR" "VLR_LR" "HR_VH
```

```
##  $ Tscore_Bucket_Prior_Ntm                                          : chr  ">-2.5" ">-2.5" "<=-2.5"
##  $ Risk_Segment_During_Rx                                           : chr  "VLR_LR" "Unknown" "HR_V
##  $ Tscore_Bucket_During_Rx                                          : chr  "<=-2.5" "Unknown" "<=-2
##  $ Change_T_Score                                                   : chr  "No change" "Unknown" "N
##  $ Change_Risk_Segment                                              : chr  "Unknown" "Unknown" "No
##  $ Adherent_Flag                                                    : chr  "Adherent" "Adherent" "A
##  $ Idn_Indicator                                                    : chr  "N" "N" "N" "N" ...
##  $ Injectable_Experience_During_Rx                                  : chr  "Y" "Y" "Y" "Y" ...
##  $ Comorb_Encounter_For_Screening_For_Malignant_Neoplasms           : chr  "N" "N" "Y" "N" ...
##  $ Comorb_Encounter_For_Immunization                                : chr  "Y" "N" "N" "Y" ...
##  $ Comorb_Encntr_For_General_Exam_W_O_Complaint._Susp_Or_Reprtd_Dx  : chr  "Y" "Y" "Y" "Y" ...
##  $ Comorb_Vitamin_D_Deficiency                                      : chr  "N" "N" "N" "N" ...
##  $ Comorb_Other_Joint_Disorder_Not_Elsewhere_Classified             : chr  "N" "N" "N" "Y" ...
##  $ Comorb_Encntr_For_Oth_Sp_Exam_W_O_Complaint_Suspected_Or_Reprtd_Dx: chr  "Y" "N" "N" "N" ...
##  $ Comorb_Long_Term_Current_Drug_Therapy                            : chr  "N" "N" "N" "N" ...
##  $ Comorb_Dorsalgia                                                 : chr  "Y" "N" "N" "Y" ...
##  $ Comorb_Personal_History_Of_Other_Diseases_And_Conditions         : chr  "Y" "N" "N" "N" ...
##  $ Comorb_Other_Disorders_Of_Bone_Density_And_Structure             : chr  "N" "N" "N" "N" ...
##  $ Comorb_Disorders_of_lipoprotein_metabolism_and_other_lipidemias  : chr  "N" "N" "N" "Y" ...
##  $ Comorb_Osteoporosis_without_current_pathological_fracture        : chr  "N" "N" "N" "N" ...
##  $ Comorb_Personal_history_of_malignant_neoplasm                    : chr  "N" "N" "N" "N" ...
##  $ Comorb_Gastro_esophageal_reflux_disease                          : chr  "N" "N" "N" "Y" ...
##  $ Concom_Cholesterol_And_Triglyceride_Regulating_Preparations      : chr  "N" "N" "Y" "N" ...
##  $ Concom_Narcotics                                                 : chr  "N" "N" "N" "Y" ...
##  $ Concom_Systemic_Corticosteroids_Plain                            : chr  "N" "N" "N" "Y" ...
##  $ Concom_Anti_Depressants_And_Mood_Stabilisers                     : chr  "N" "N" "N" "N" ...
##  $ Concom_Fluoroquinolones                                          : chr  "N" "N" "N" "N" ...
##  $ Concom_Cephalosporins                                            : chr  "N" "N" "N" "N" ...
##  $ Concom_Macrolides_And_Similar_Types                              : chr  "N" "N" "N" "N" ...
##  $ Concom_Broad_Spectrum_Penicillins                                : chr  "N" "N" "N" "N" ...
##  $ Concom_Anaesthetics_General                                      : chr  "N" "N" "N" "N" ...
##  $ Concom_Viral_Vaccines                                            : chr  "N" "N" "N" "Y" ...
##  $ Risk_Type_1_Insulin_Dependent_Diabetes                           : chr  "N" "N" "N" "N" ...
##  $ Risk_Osteogenesis_Imperfecta                                     : chr  "N" "N" "N" "N" ...
##  $ Risk_Rheumatoid_Arthritis                                        : chr  "N" "N" "N" "N" ...
##  $ Risk_Untreated_Chronic_Hyperthyroidism                           : chr  "N" "N" "N" "N" ...
##  $ Risk_Untreated_Chronic_Hypogonadism                              : chr  "N" "N" "N" "N" ...
##  $ Risk_Untreated_Early_Menopause                                   : chr  "N" "N" "N" "N" ...
##  $ Risk_Patient_Parent_Fractured_Their_Hip                          : chr  "N" "N" "Y" "N" ...
##  $ Risk_Smoking_Tobacco                                             : chr  "N" "N" "N" "Y" ...
##  $ Risk_Chronic_Malnutrition_Or_Malabsorption                       : chr  "N" "N" "N" "N" ...
##  $ Risk_Chronic_Liver_Disease                                       : chr  "N" "N" "N" "N" ...
##  $ Risk_Family_History_Of_Osteoporosis                              : chr  "N" "N" "N" "N" ...
##  $ Risk_Low_Calcium_Intake                                          : chr  "N" "N" "Y" "N" ...
##  $ Risk_Vitamin_D_Insufficiency                                     : chr  "N" "N" "N" "N" ...
##  $ Risk_Poor_Health_Frailty                                         : chr  "N" "N" "N" "N" ...
##  $ Risk_Excessive_Thinness                                          : chr  "N" "N" "N" "N" ...
##  $ Risk_Hysterectomy_Oophorectomy                                   : chr  "N" "N" "N" "N" ...
##  $ Risk_Estrogen_Deficiency                                         : chr  "N" "N" "N" "N" ...
##  $ Risk_Immobilization                                              : chr  "N" "N" "N" "N" ...
##  $ Risk_Recurring_Falls                                             : chr  "N" "N" "N" "N" ...
##  $ Count_Of_Risks                                                   : int  0 0 2 1 1 2 1 1 1 1 ...
```

```r
# overview of the dataset
summary(dataset)
```

From the analysis of the dataset we see that there is no missing values. In order to have a differential approach between categorical and numerical features we have decided to split the database into two parts. Within the categorical dataset we decide to eliminate two predictors that seems not to have an information content (ID of the Patient and the speciality of the HCP that prescribed the NTM Rx).

```r
# verify if missing values are present
any(is.na(dataset))
```

```
## [1] FALSE
```

```r
# split between numerical and categorical predictors

datanum<-dataset %>%
  select_if(is.numeric)

# categorical predictors: elimination of categorical predictors without information content
datacat<-dataset %>%
  select_if(negate(is.numeric))
datacat<-datacat %>% select(-c(Ptid,Ntm_Speciality))
```

Categorical predictors

We start the deepening as regard as the categorical features. First of all, we convert the data into a tidy format with 3 columns: Persistency_Flag, predictors and values of predictors (Yes or No, Male or Female,..). This permit us to summarize the percentage of Persistent events, for each predictor and for all its relative value/category. This is a key picture in order to put in evidence the predictors, whose values cause very different percentage of Persistent events.These predictors should have a larger impact on drug persistency. In the following graph it is possible to see for each feature the minimum and maximum value of the persistent rate linked to different categories of the predictors.

```r
### global view of categorical predictors

tidycat<-datacat %>% gather(key=predictors,value=value,-Persistency_Flag)
head(tidycat)
```

```
##   Persistency_Flag predictors  value
## 1       Persistent     Gender   Male
## 2   Non-Persistent     Gender   Male
## 3   Non-Persistent     Gender Female
## 4   Non-Persistent     Gender Female
## 5   Non-Persistent     Gender Female
## 6   Non-Persistent     Gender Female
```

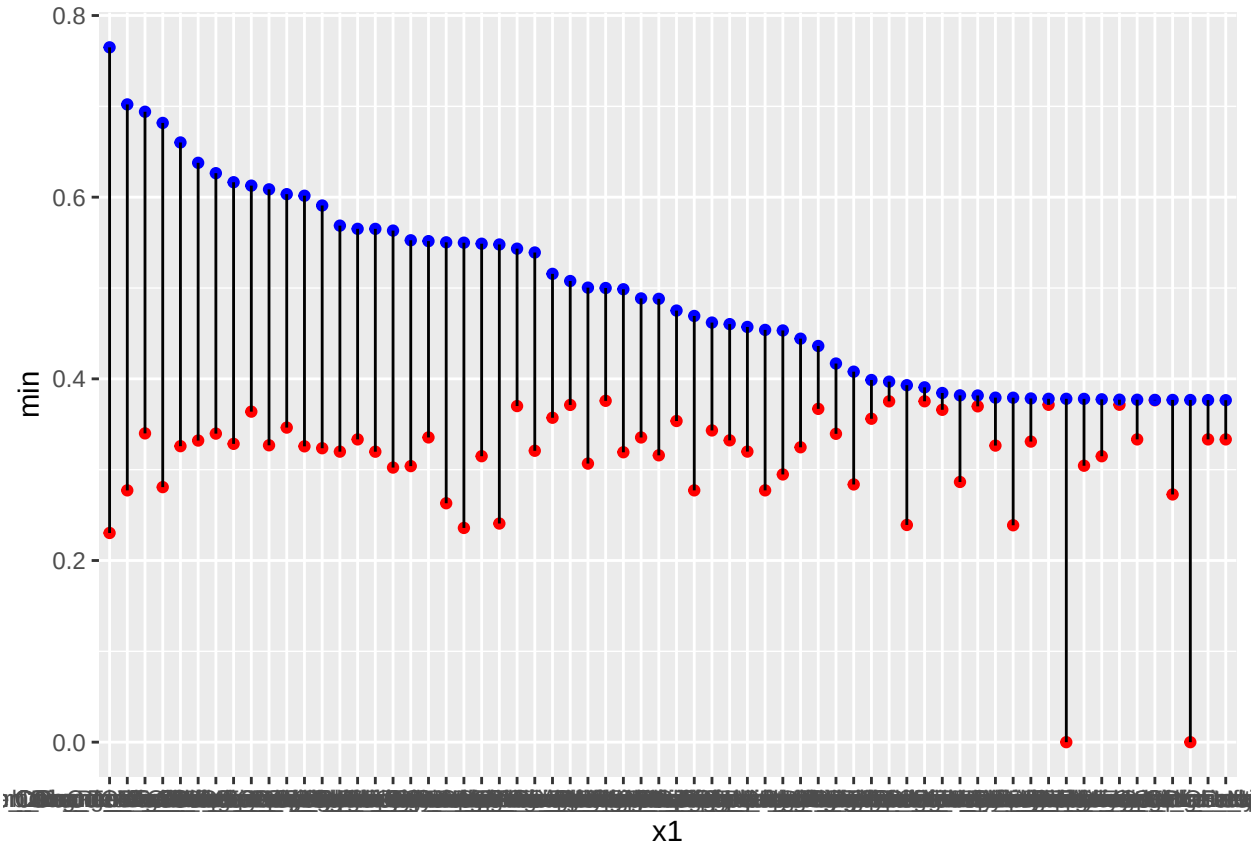
```r
# summarize for each predictor and category within predictors the percentage of Persistent cases

table<-tidycat %>% group_by(predictors,value) %>% summarize(Persistent_rate=mean(Persistency_Flag=="Pers
  group_by(predictors) %>% summarize(min=min(Persistent_rate),max=max(Persistent_rate))
```

```
## `summarise()` has grouped output by 'predictors'. You can override using the
## `.groups` argument.
```

```
# the percentage of Persistent cases across different categories and predictors
# in order to put in evidence the features that influence more the outcome

table %>% arrange(desc(max))
```

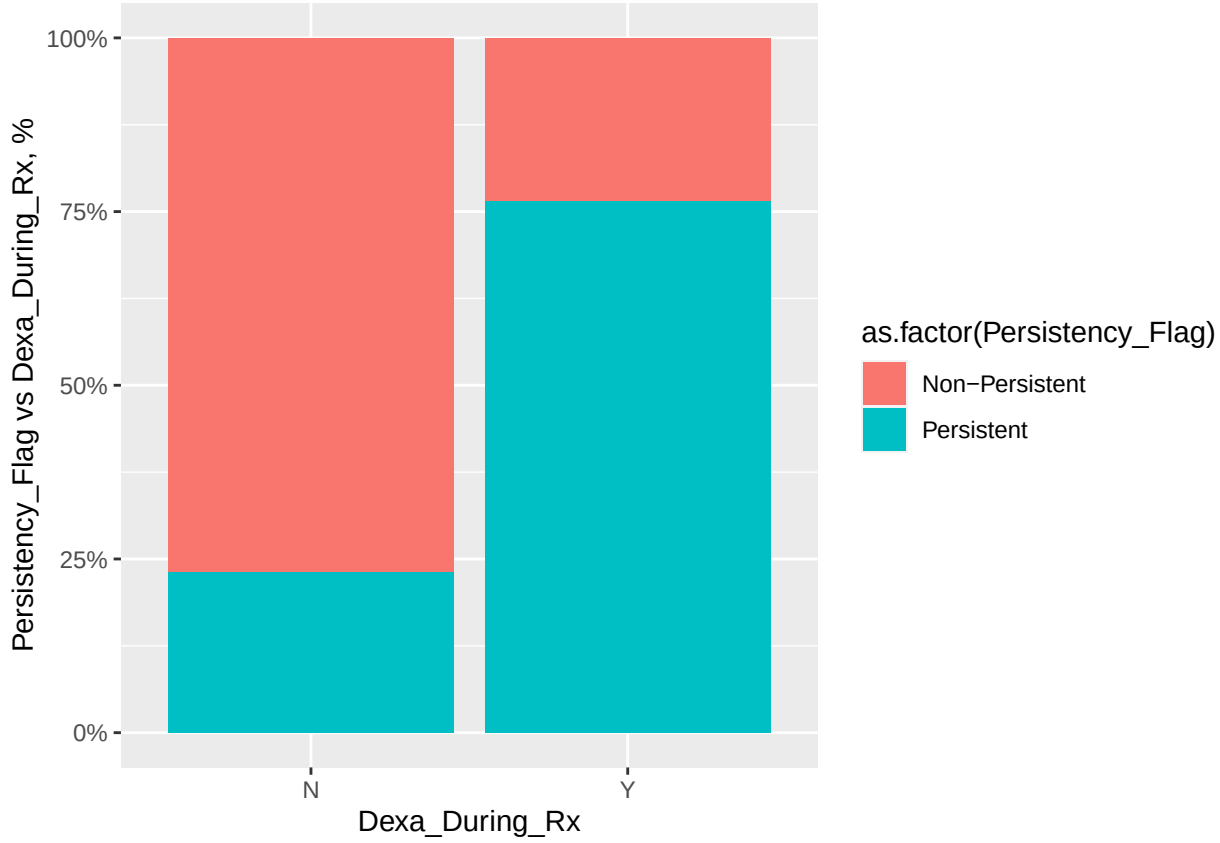
```
## # A tibble: 64 x 3
##    predictors                                              min   max
##    <chr>                                                 <dbl> <dbl>
##  1 Dexa_During_Rx                                        0.230 0.765
##  2 Change_T_Score                                        0.277 0.702
##  3 Concom_Viral_Vaccines                                 0.340 0.694
##  4 Comorb_Long_Term_Current_Drug_Therapy                 0.281 0.682
##  5 Comorb_Other_Disorders_Of_Bone_Density_And_Structure 0.326 0.660
##  6 Concom_Anaesthetics_General                           0.332 0.638
##  7 Concom_Broad_Spectrum_Penicillins                     0.340 0.626
##  8 Concom_Macrolides_And_Similar_Types                   0.328 0.616
##  9 Adherent_Flag                                         0.364 0.613
## 10 Concom_Cephalosporins                                 0.327 0.609
## # ... with 54 more rows
```

```
table %>% mutate(x1 = fct_reorder(predictors, desc(max))) %>% ggplot(aes(x=x1,y=min))+geom_point(color=
  geom_segment(aes(x = x1,y = min, xend = x1,yend = max))
```



Choosing among the first 10 features with a larger difference between maximum and minimum percentage of Persistent rate, we want to analyze better these situations.It is clear, for example, that for the patients with a positive Dexa_During_Rx, the percentage of Persistent case is really higher (76%).

```
# Categorical Predictor analysis: Dexa_During_Rx

datacat %>% ggplot(aes(x=Dexa_During_Rx,fill=as.factor(Persistency_Flag)))+
  geom_bar(aes( y=..count../tapply(..count.., ..x.. ,sum)[..x..]))+
  ylab('Persistency_Flag vs Dexa_During_Rx, %') +
  scale_y_continuous(labels = scales::percent)
```



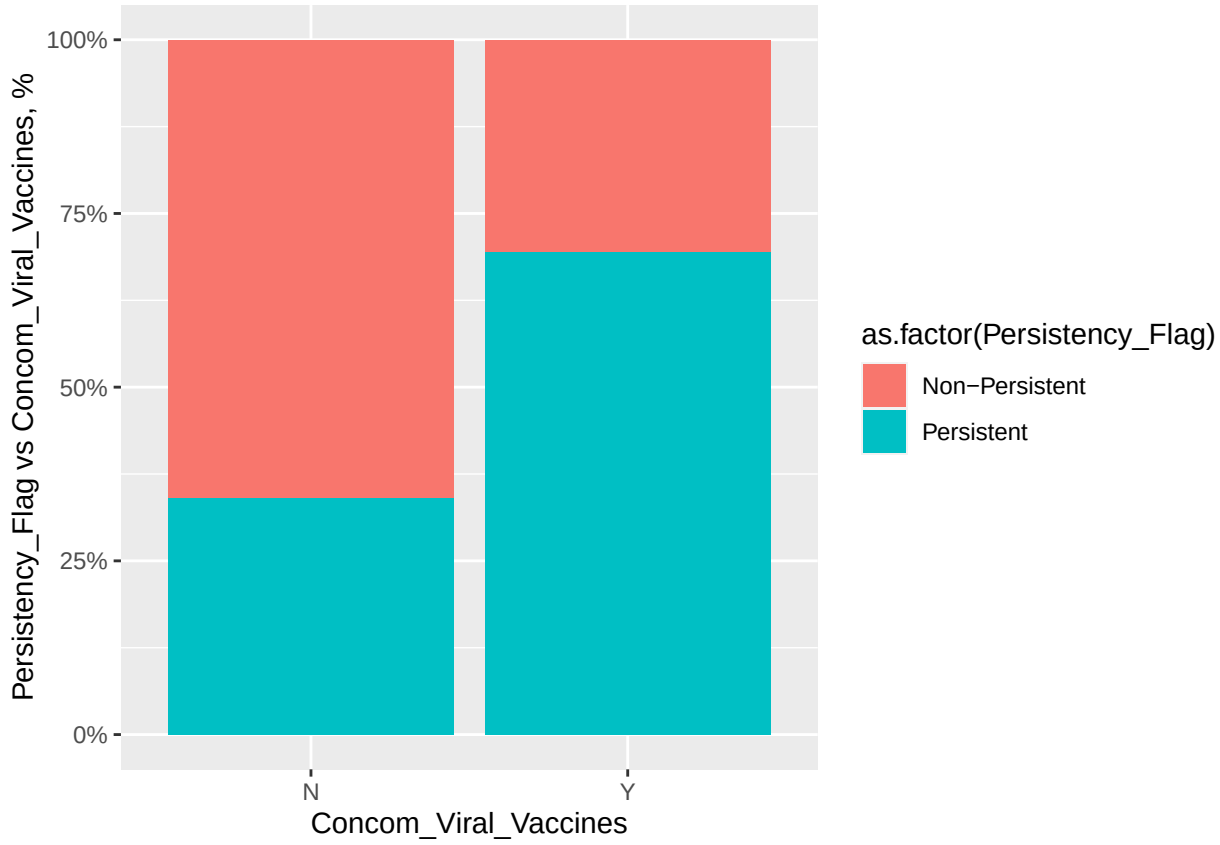

```
datacat %>% group_by(Dexa_During_Rx) %>% summarize(Persistent_rate=mean(Persistency_Flag=="Persistent"))
```

```
## # A tibble: 2 x 2
##   Dexa_During_Rx Persistent_rate
##   <chr>                   <dbl>
## 1 N                       0.230
## 2 Y                       0.765
```

In the same way we analyze the predictor "Concom_Viral_Vaccines": in the presence of comorbidity with viral vaccines (yes) the persistent cases are about 70%, in the absence the percentage goes down to 30%.

```
# Categorical Predictor analysis: Concom_Viral_Vaccines
```

```
datacat %>% ggplot(aes(x=Concom_Viral_Vaccines,fill=as.factor(Persistency_Flag)))+
  geom_bar(aes( y=..count../tapply(..count.., ..x.. ,sum)[..x..]))+
  ylab('Persistency_Flag vs Concom_Viral_Vaccines, %') +
  scale_y_continuous(labels = scales::percent)
```

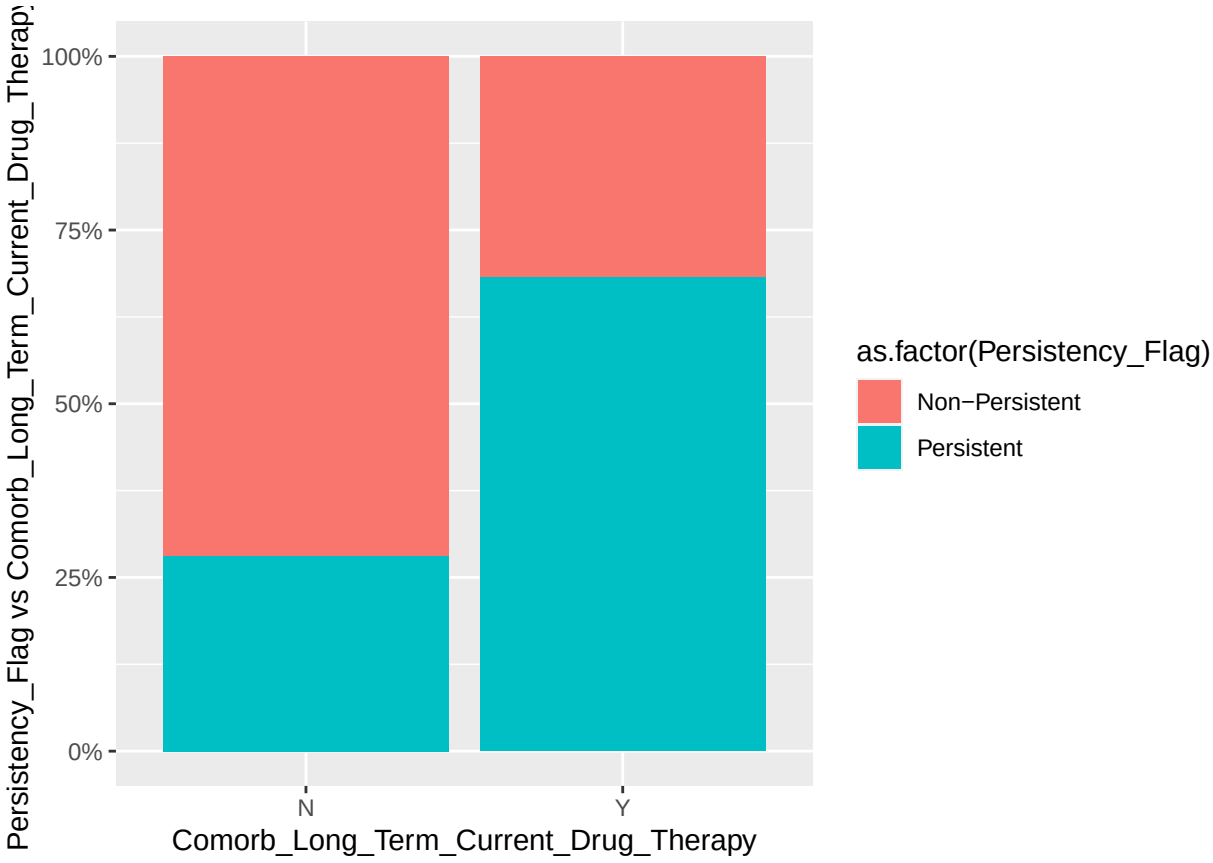



```
datacat %>% group_by(Concom_Viral_Vaccines) %>% summarize(Persistent_rate=mean(Persistency_Flag=="Persi
```

```
## # A tibble: 2 x 2
##   Concom_Viral_Vaccines Persistent_rate
##   <chr>                           <dbl>
## 1 N                               0.340
## 2 Y                               0.694
```

We see more or less the same results also for the predictor Comorb_Long_Term_Current_Drug_Therapy.

```
# Categorical Predictor analysis: Comorb_Long_Term_Current_Drug_Therapy

datacat %>% ggplot(aes(x=Comorb_Long_Term_Current_Drug_Therapy,fill=as.factor(Persistency_Flag)))+
  geom_bar(aes( y=..count../tapply(..count.., ..x.. ,sum)[..x..]))+
  ylab('Persistency_Flag vs Comorb_Long_Term_Current_Drug_Therapy, %') +
  scale_y_continuous(labels = scales::percent)
```



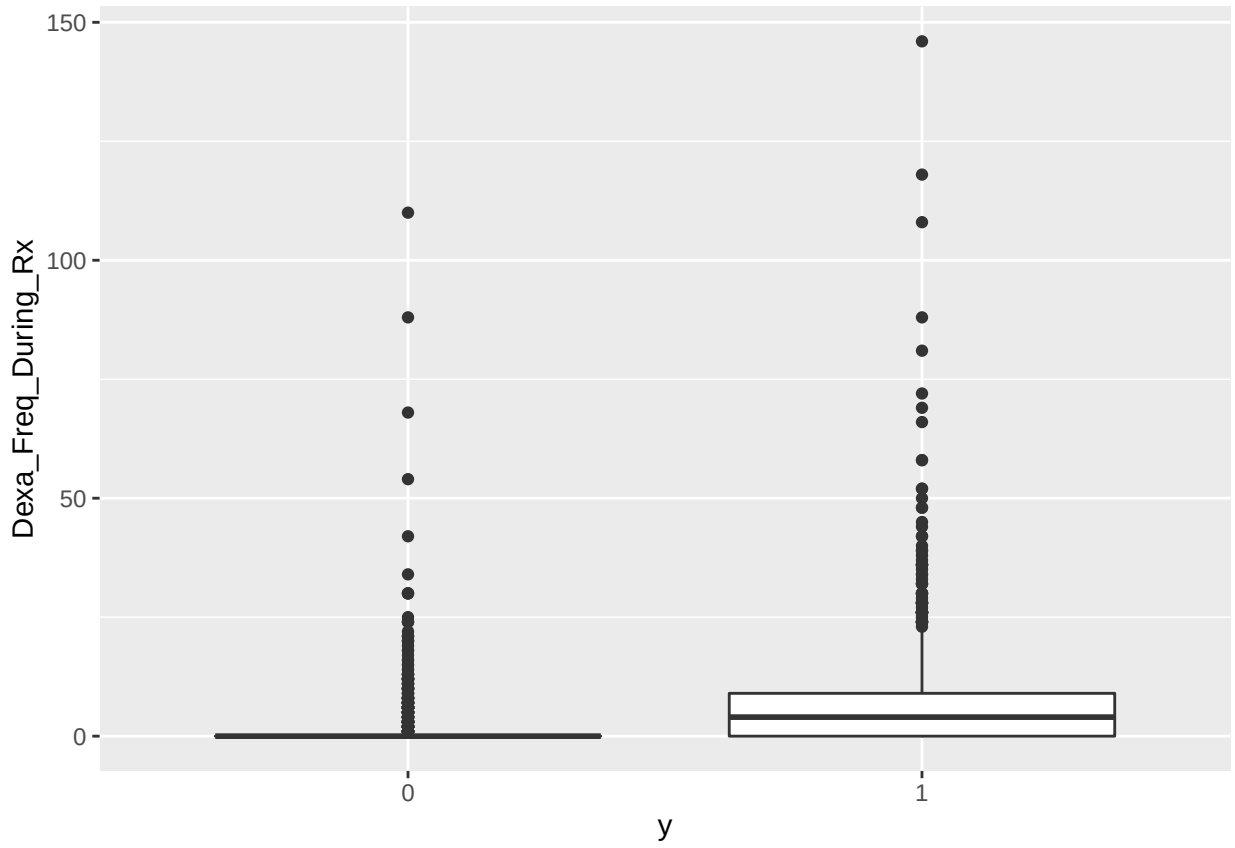

```
datacat %>% group_by(Comorb_Long_Term_Current_Drug_Therapy) %>% summarize(Persistent_rate=mean(Persister
```

```
## # A tibble: 2 x 2
##   Comorb_Long_Term_Current_Drug_Therapy Persistent_rate
##   <chr>                                          <dbl>
## 1 N                                              0.281
## 2 Y                                              0.682
```

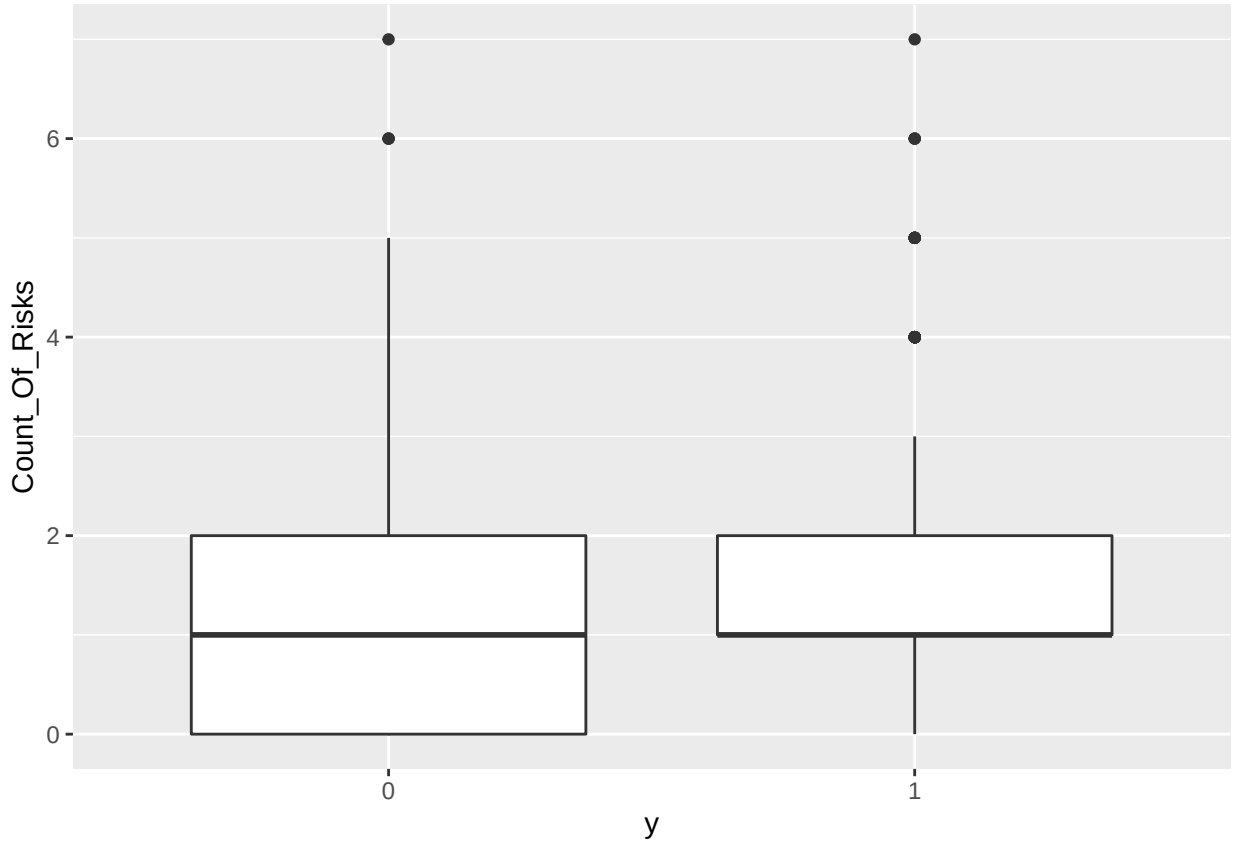
Numerical predictors

The next step is the analysis of the two numerical features, for which we use boxplot to see the distribution of the two predictors considering the two categories of the outcome. Only for the first predictor (Dexa_Freq_During_Rx) we see a significant difference between the two distributions. Using density plot, we have a confirmation of the previous remarks.

```
# global view of numerical predictors
y<-as.factor(ifelse(datacat$Persistency_Flag=="Persistent",1,0))
relevance<-cbind(y,datanum)

# boxplot
relevance %>% ggplot(aes(x=y,y=Dexa_Freq_During_Rx))+geom_boxplot()
```

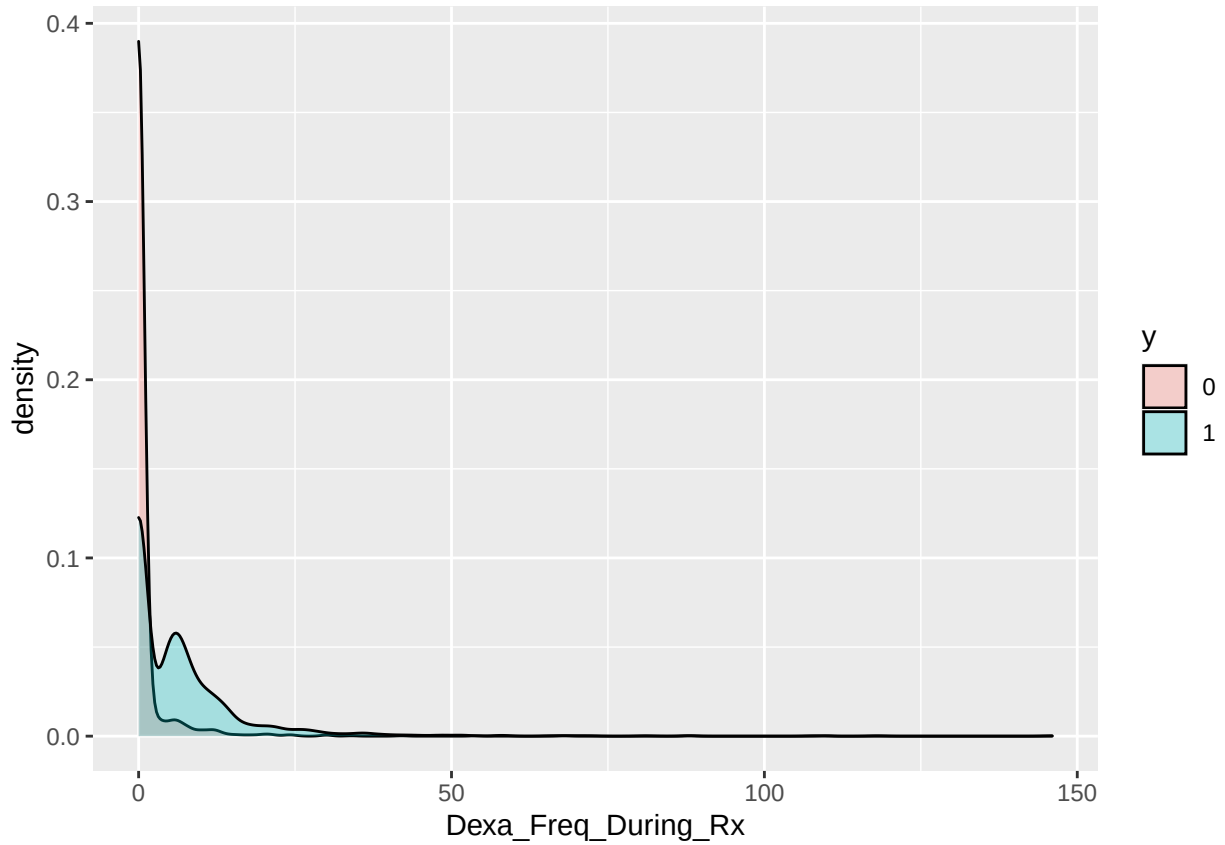



```
relevance %>% ggplot(aes(x=y,y=Count_Of_Risks))+geom_boxplot()
```

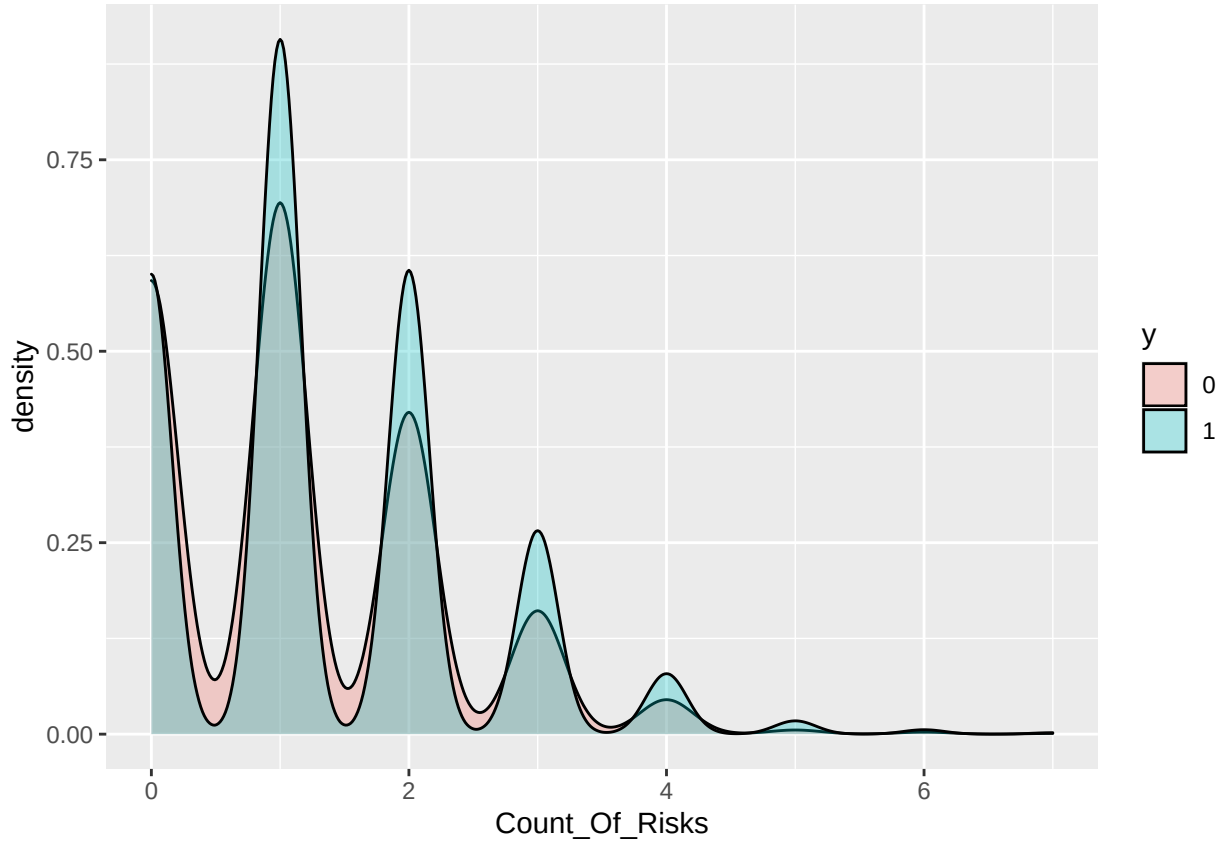



```
#density plot
relevance %>% ggplot(aes(x=Dexa_Freq_During_Rx,fill=y))+geom_density(alpha=0.3,xlim=c(0,50))
```

```
## Warning: Ignoring unknown parameters: xlim
```




```
relevance %>% ggplot(aes(x=Count_Of_Risks,fill=y))+geom_density(alpha=0.3)
```



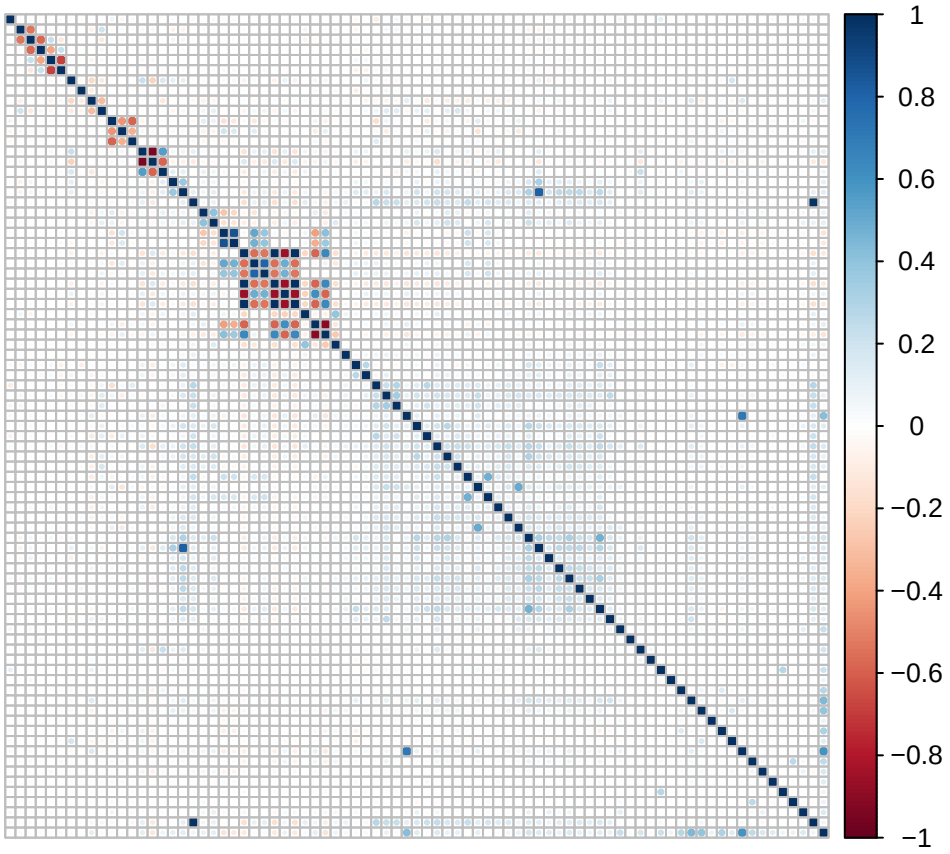
Predictors correlation

In order to have a complete overview of the dataset, it is also important to have an idea of the correlation of the different predictors. Before doing this, it could be useful to transform the categorical features into dummy variables. The aim is to have an easier interpretation of the variables, in particular for those ones which have different levels.This is also useful for the further correlation and model analysis. We use the function dummy_cols within fastDummies library, with this option: remove_first_dummy = TRUE, in order to be sure not to have duplicated variables. It means that a binary predictor is replaced by only a variable with 0 or 1 values.The variables are now 79.

```
# dummy vars creation
datacat1<-datacat %>% select(-c(Persistency_Flag))
dataf <- dummy_cols(datacat1,remove_selected_columns = TRUE,remove_first_dummy = TRUE)

# re-build of the dataset
cor_set<-cbind(dataf,datanum)
```

Then we use the re-built dataset to see the correlation among predictors. Since the categorical variables are binary, for the correlation the method "spearman" is more appropriate, in comparison with the default one, which implies the normal distribution of the covariates. From the corrplot there is no evidence of huge correlation or collinearity across the dataset, even though a group of high correlated variables exists.

```
# correlation plot: evidence of the correlation inside the structure
correlation<-cor(cor_set,method="spearman")
corrplot(correlation,tl.pos='n')
```




```
# selection of the correlated predictors
highCorr<-findCorrelation(correlation,cutoff=.6)
highCorr_list<-cor_set[,highCorr]
str(highCorr_list)
```

```
## 'data.frame':    3424 obs. of  12 variables:
##  $ Risk_Segment_During_Rx_Unknown                 : int  0 1 0 0 1 1 1 0 1 1 ...
##  $ Tscore_Bucket_During_Rx_Unknown                : int  0 1 0 0 1 1 1 0 1 1 ...
##  $ Change_T_Score_Unknown                         : int  0 1 0 0 1 1 1 0 1 1 ...
##  $ Concom_Systemic_Corticosteroids_Plain_Y        : int  0 0 0 1 1 0 0 1 0 0 ...
##  $ Change_T_Score_No change                       : int  1 0 1 1 0 0 0 1 0 0 ...
##  $ Change_Risk_Segment_Unknown                    : int  1 1 0 0 1 1 1 0 1 1 ...
##  $ Dexa_Freq_During_Rx                            : int  0 0 0 0 0 0 2 0 0 0 ...
##  $ Tscore_Bucket_During_Rx_>-2.5                  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Risk_Segment_Prior_Ntm_VLR_LR                  : int  1 1 0 0 0 0 0 0 1 0 ...
##  $ Ntm_Speciality_Bucket_OB/GYN/Others/PCP/Unknown: int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Comorb_Vitamin_D_Deficiency_Y                  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Ethnicity_Unknown                              : int  0 0 0 0 0 0 0 0 0 0 ...
```

Data partition

For the further model analysis we have to do the partition of the dataset between training and test set. Inside the training set we do a split between features and outcome and we change one category name of the outcome, because the term Non-Persistent is not correctly read in some R packages. Therefore we replace it with Non.Persistent.

```r
# data partition
set.seed(100, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(100)
```

```
## Warning in set.seed(100, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

```r
Persistency_Flag<-datacat$Persistency_Flag

# re-build of the dataset with outcome
rev_set<-cbind(Persistency_Flag,dataf,datanum)
rev_set$Persistency_Flag<-factor(rev_set$Persistency_Flag)

# Validation set will be 20% of dataset
split<-createDataPartition(rev_set$Persistency_Flag,p=.8,times=1,list=FALSE)
training<-rev_set[split,]
test<-rev_set[-split,]

# split between outcome and features in the training set
outcome<-ifelse(training$Persistency_Flag=="Non-Persistent","Non.Persistent","Persistent")
features<-training %>% select(-c(Persistency_Flag))

#replace the term Non-Persistent with Non.Persistent
test$Persistency_Flag<-factor(ifelse(test$Persistency_Flag=="Non-Persistent","Non.Persistent","Persiste
```

Model Analysis

Considering that we are facing a classification problem, we test the logistic regression as first model, then we try with classification trees (rpart and random forest). Given the structure of the dataset, these models should be more appropriate in comparison with linear discriminant models, for which the assumption is the normal distribution of the indipendent variables.With this dataset the assumption can't be satisfied. Finally other algoritms (support vector machines and glm with penalty) are implemented. From the bibliography we assume in fact that the glm with penalty model should stabilize the logistic regression coefficients in a situation with a large number of predictors, while SVM could have good performance thanks to the flexibility in the boundary calculation used for the classification. We use library caret for all the models, setting up the trainControl parameters, in order to have not only the class predictions, but also the class probabilities. For cross-validation we maintain the default option of the library. In this section we suppress the warnings for better readability of the report.

```r
# fit control for the function train
fit.control <- trainControl(summaryFunction = twoClassSummary, classProbs = TRUE)

# suppress the warnings for better readability of the report
options(warn=-1)
```

1. Logistic Regression

We train the model and then select the variables that are significant in order to put in evidence the factors that impact more on the outcome.

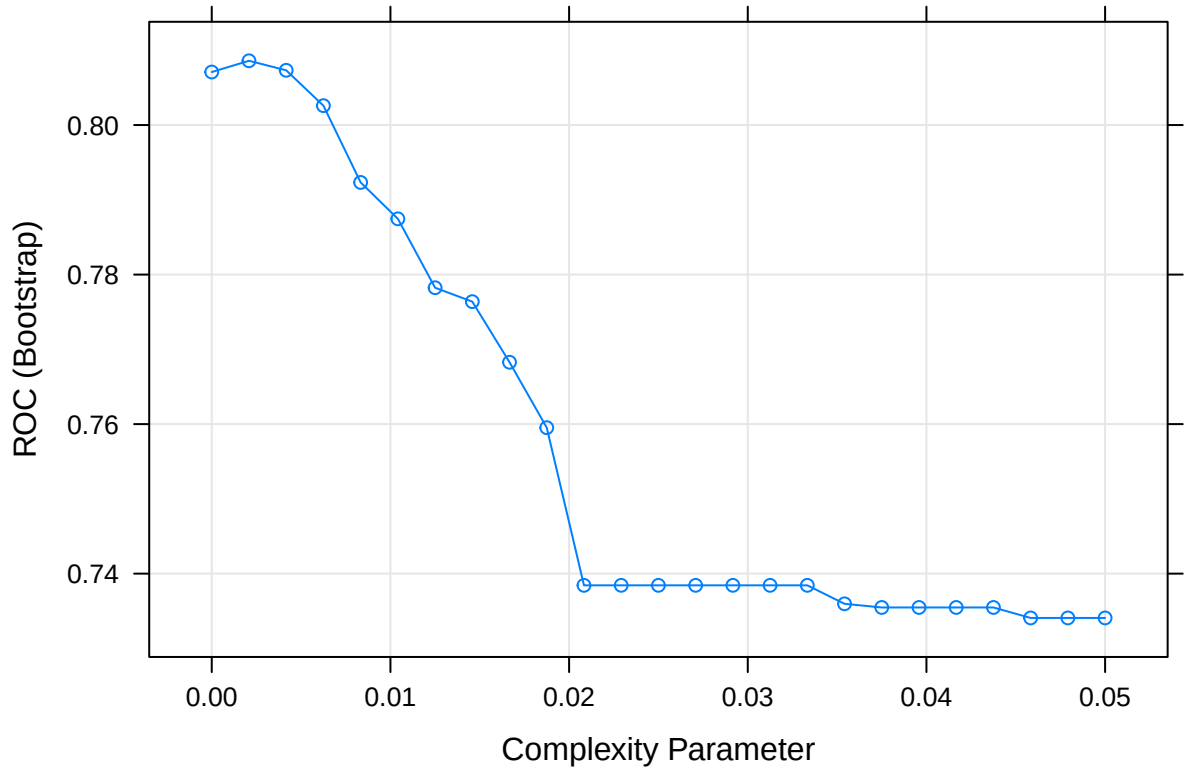
```r
# 1. model GLM

set.seed(7, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(7)
# model GLM training on the train set
model_glm<-train(x=features,y=outcome,method="glm",family="binomial",trControl = fit.control)

# prediction on the test set
y_pred_glm<-factor(predict(model_glm,test))
# accuracy calculation
accuracy_glm<-confusionMatrix(y_pred_glm,test$Persistency_Flag) $overall["Accuracy"]

# select sig. variables
toselect.x <- summary(model_glm)$coeff[-1,4] < 0.05
relevant.x <- names(toselect.x)[toselect.x == TRUE]
# show sig. variables
relevant.x
```

```
##  [1] "Race_Asian"
##  [2] "Region_South"
##  [3] "Ntm_Speciality_Bucket_Rheum"
##  [4] "Gluco_Record_Prior_Ntm_Y"
##  [5] "Dexa_During_Rx_Y"
##  [6] "Frag_Frac_Prior_Ntm_Y"
##  [7] "Risk_Segment_During_Rx_Unknown"
##  [8] "Idn_Indicator_Y"
##  [9] "Comorb_Encounter_For_Screening_For_Malignant_Neoplasms_Y"
## [10] "Comorb_Encounter_For_Immunization_Y"
## [11] "Comorb_Encntr_For_General_Exam_W_O_Complaint._Susp_Or_Reprtd_Dx_Y"
## [12] "Comorb_Vitamin_D_Deficiency_Y"
## [13] "Comorb_Other_Joint_Disorder_Not_Elsewhere_Classified_Y"
## [14] "Comorb_Long_Term_Current_Drug_Therapy_Y"
## [15] "Comorb_Personal_History_Of_Other_Diseases_And_Conditions_Y"
## [16] "Comorb_Other_Disorders_Of_Bone_Density_And_Structure_Y"
## [17] "Comorb_Personal_history_of_malignant_neoplasm_Y"
## [18] "Comorb_Gastro_esophageal_reflux_disease_Y"
## [19] "Concom_Narcotics_Y"
## [20] "Concom_Systemic_Corticosteroids_Plain_Y"
## [21] "Concom_Macrolides_And_Similar_Types_Y"
## [22] "Concom_Broad_Spectrum_Penicillins_Y"
## [23] "Concom_Anaesthetics_General_Y"
## [24] "Concom_Viral_Vaccines_Y"
## [25] "Risk_Type_1_Insulin_Dependent_Diabetes_Y"
## [26] "Risk_Untreated_Chronic_Hypogonadism_Y"
## [27] "Risk_Vitamin_D_Insufficiency_Y"
## [28] "Risk_Poor_Health_Frailty_Y"
## [29] "Risk_Recurring_Falls_Y"
```

2. Classification trees - rpart

With rpart we have the possibility to make a tuning of cp parameter. The plot shows that the best tune for complexity parameter cp. ROC values are used to choose the optimal cp: from the graph we see that greater values of ROC are obtained with low cp and then the curve drops rapidly.

```r
#2. model RPART

set.seed(15, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(15)
# model RPART training on the train set - cross-validation to choose cp
model_rpart<-train(x=features,y=outcome,method="rpart",
                   tuneGrid=data.frame(cp=seq(0,0.05,len=25)),
                   trControl = fit.control)

# plot cp optimization
plot(model_rpart)
```




```r
# summary model rpart
model_rpart
```

```
## CART
##
## 2740 samples
##   81 predictor
##    2 classes: 'Non.Persistent', 'Persistent'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 2740, 2740, 2740, 2740, 2740, 2740, ...
## Resampling results across tuning parameters:
##
```

```
##    cp           ROC        Sens       Spec
##    0.000000000  0.8070920  0.8183670  0.6511208
##    0.002083333  0.8085964  0.8439948  0.6573786
##    0.004166667  0.8073231  0.8527871  0.6598778
##    0.006250000  0.8025953  0.8663059  0.6513712
##    0.008333333  0.7923256  0.8740945  0.6388925
##    0.010416667  0.7874571  0.8780981  0.6328452
##    0.012500000  0.7782390  0.8792368  0.6233219
##    0.014583333  0.7763805  0.8816350  0.6194596
##    0.016666667  0.7682744  0.8912653  0.6033693
##    0.018750000  0.7595204  0.8961196  0.5939217
##    0.020833333  0.7384319  0.9173518  0.5542177
##    0.022916667  0.7384319  0.9173518  0.5542177
##    0.025000000  0.7384319  0.9173518  0.5542177
##    0.027083333  0.7384319  0.9173518  0.5542177
##    0.029166667  0.7384319  0.9173518  0.5542177
##    0.031250000  0.7384319  0.9173518  0.5542177
##    0.033333333  0.7384319  0.9173518  0.5542177
##    0.035416667  0.7359620  0.9194571  0.5499050
##    0.037500000  0.7354717  0.9217007  0.5477050
##    0.039583333  0.7354717  0.9217007  0.5477050
##    0.041666667  0.7354717  0.9217007  0.5477050
##    0.043750000  0.7354717  0.9217007  0.5477050
##    0.045833333  0.7340710  0.9241483  0.5439937
##    0.047916667  0.7340710  0.9241483  0.5439937
##    0.050000000  0.7340710  0.9241483  0.5439937
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.002083333.
```

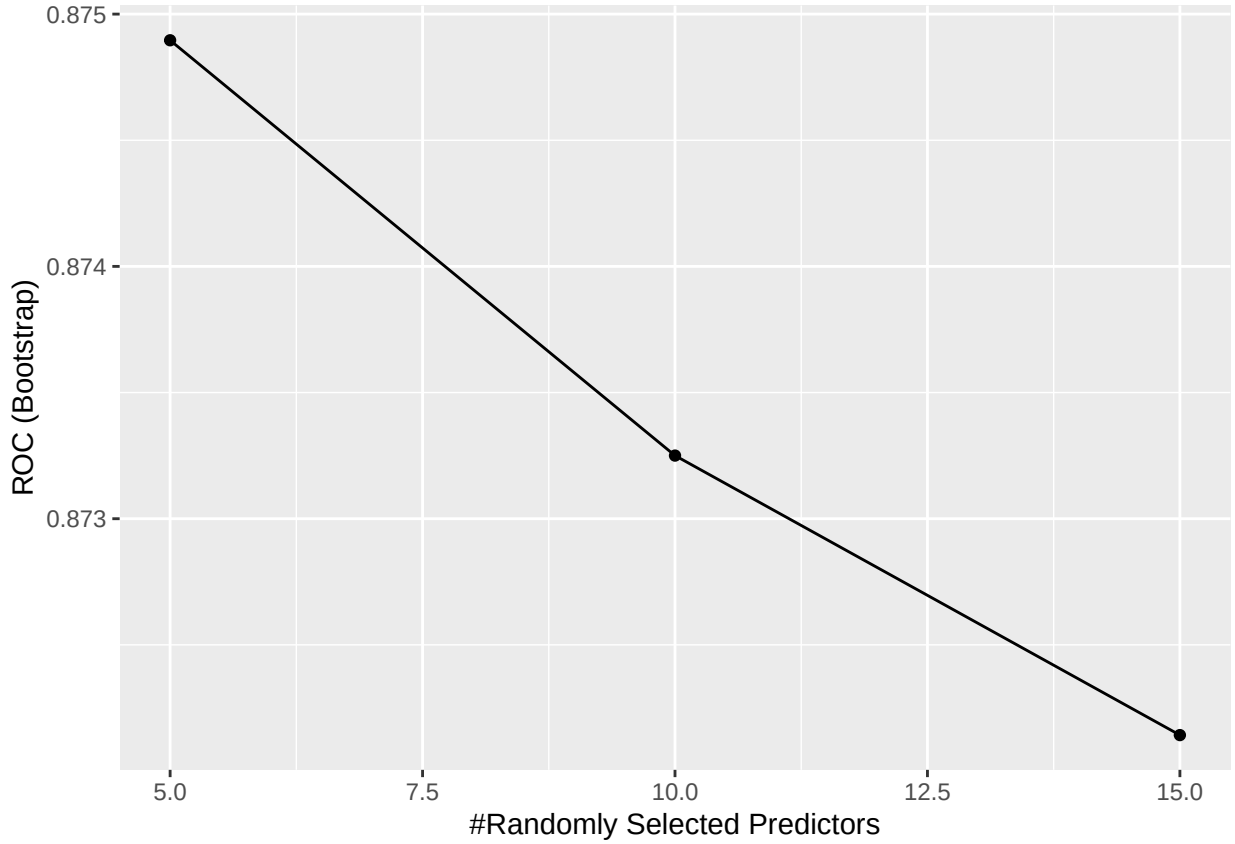
```
# prediction on the test set
y_pred_rpart<-predict(model_rpart,test)
# accuracy calculation
accuracy_rpart<-confusionMatrix(y_pred_rpart,test$Persistency_Flag) $overall["Accuracy"]
```

3. Classification trees - random forest

Then we train random forest, which allows to put in evidence the features that have an higher impact on the outcome. The computation time of this model is quite long, therefore the use of a tuning grid involves a significant computational effort,considering also the number of variables in the dataset. After some attemps, we have chosen to tune the model using a limited customized grid, with only three values of the parameter mtry (number of the variables randomly sampled as candidates at each split).

```
#3. model RANDOM FOREST

set.seed(70, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(70)

# tune grid with mtry=5,10,15
grid<-expand.grid(.mtry=c(5,10,15))

# model RANDOM FOREST training on the train set
# this line takes a long time to run
model_rf<-train(x=features,y=outcome,method="rf",trControl = fit.control,tuneGrid=grid)
```

```r
# plot the tuning of mtry (number of the variables randomly sampled as candidates at each split)
ggplot(model_rf)
```




```r
#best tune model
model_rf$bestTune
```

```
##   mtry
## 1    5
```

```r
# prediction on the test set
y_pred_rf<-predict(model_rf,test)
# accuracy calculation
accuracy_rf<-confusionMatrix(y_pred_rf,test$Persistency_Flag) $overall["Accuracy"]

# main important features
important_var<-varImp(model_rf)
main_var<-data.frame(important_var$importance) %>% arrange(desc(Overall)) %>% top_n(30)
```

```
## Selecting by Overall
```

```r
relevant.xrf<-row.names(main_var)
```

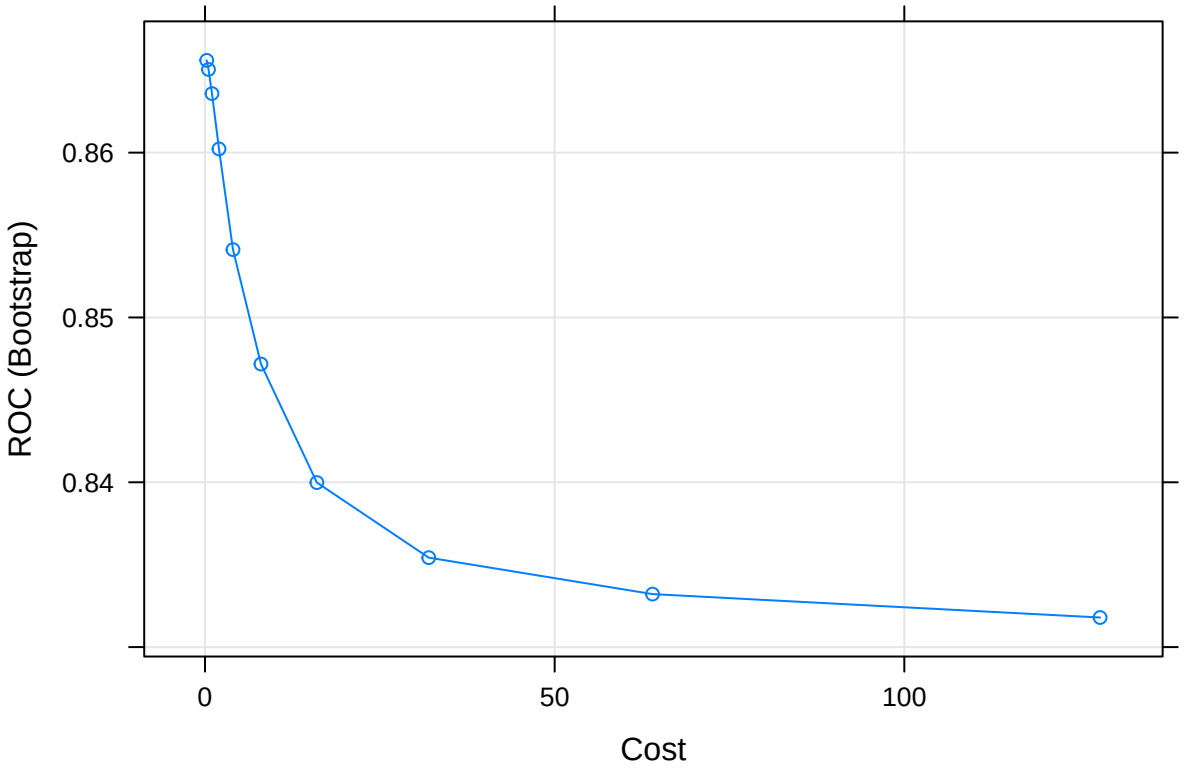
4. Support vector machines

Support vector machine algorithm has the objective to find a hyperplane in an N-dimensional space(N — the number of features) that distinctly classifies the data points. In practice, we are looking to maximize the margin between the data points and the hyperplane. Within the family of SVM algorithms, we choose to use the model with radial-basis-functions, that allows to have non-linear decision boundaries. The hyperparameters associated with this model are: sigma and cost. But, also in this case, we face computational problems to tune parameters. Therefore we train this model tuning only the cost parameter, using the tuneLength argument, that consider a default grid search of 10 cost values between:

$$\frac{1}{2^2}, \frac{1}{2^1}, 2^0, 2^1, 2^2, 2^3, 2^4, ..., 2^7$$

while sigma is estimated analytically by default. The cost parameter control the complexity of the model: the tuning phase should permit to find a balance between under and over-fitting.

```
#4. model SVM

set.seed(87, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(87)
training_set<-cbind(outcome,features)

# model SVM training on the train set
# this line takes a long time to run
# tuning cost parameter with tuneLength argument
model_svm<-train(outcome~.,data=training_set,method="svmRadial",trControl = fit.control,tuneLength=10)

# plot cost parameter optimization
plot(model_svm)
```




```
#best tune model
model_svm$bestTune
```

```
##         sigma    C
## 1 0.008001509 0.25
```

```
# prediction on the test set
y_pred_svm<-predict(model_svm,test,type="raw")
# accuracy calculation
accuracy_svm<-confusionMatrix(y_pred_svm,test$Persistency_Flag) $overall["Accuracy"]
```

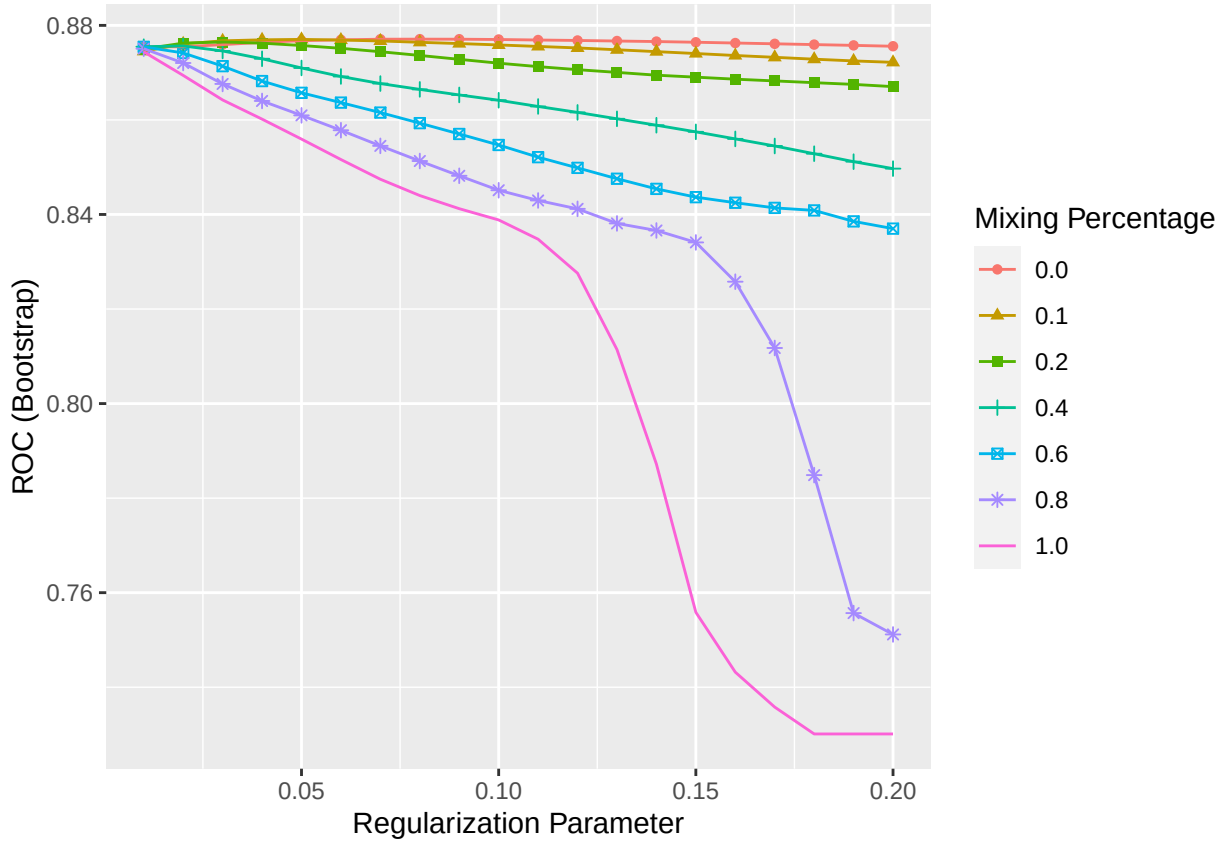
5. Glm with penalty - glmnet

For the model optimization, it is minimized the negative binomial log-likelihood, penalized adding this term:

$$\lambda[(1-\alpha)||\beta||_2^2/2 + \alpha||\beta||_1]$$

The penalty term is the combination of two parameters: alpha and lambda. The tuning parameter lambda controls the overall strength of the penalty, while alpha can have values between 0 and 1: with alpha=0 we have the ridge regression with alpha=1 the lasso regression. From the introduction of glmnet, it is possible to have a wider overview of the model (https://glmnet.stanford.edu/articles/glmnet.html). We tune the model using a grid of both parameters. The best model is chosen by train function with alpha=0 and lambda=0.08.

```r
#5. model GLMNET

set.seed(300, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(300)

# grid for tuning parameters (alpha and lambda)
glmnGrid<-expand.grid(.alpha=c(0,0.1,0.2,0.4,0.6,0.8,1),
                      .lambda=seq(.01,.2,length=20))

# model GLMNET training on the train set
model_glmnet<-train(x=features,y=outcome,method="glmnet",trControl = fit.control,tuneGrid=glmnGrid,famil

# plot the result of tuning
ggplot(model_glmnet)
```




```r
#best tune model
model_glmnet$bestTune
```

```
##   alpha lambda
## 8     0   0.08
```

```r
# prediction on the test set
y_pred_glmnet<-predict(model_glmnet,test)
# accuracy calculation
accuracy_glmnet<-confusionMatrix(y_pred_glmnet,test$Persistency_Flag) $overall["Accuracy"]
```

Model Evaluation

1. Accuracy

First of all we consider accuracy to evaluate the model: logistic regression, svm and glmnet have accuracy over 82%, while the other models presents results very closed to 80%.

```
# restore the warnings visualization
options(warn=0)

### Model evaluation - Accuracy
accuracy_results<-data.frame(method=c("glm","rpart",
                                      "random forest",
                                      "svm","glmnet"),
                             Accuracy=c(accuracy_glm,accuracy_rpart,accuracy_rf,accuracy_svm,accuracy_glmnet

knitr::kable(accuracy_results)
```

method	Accuracy
glm	0.8230994
rpart	0.7894737
random forest	0.8055556
svm	0.8274854
glmnet	0.8318713

In parallel it is important to understand if the models (logistic regression and random forest) select the same features. The evidence is that, comparing the chosen features, 18 variables are selected in both cases, so it confirms their strong impact on the outcome.

```
# join the main factors
relevant.xrf<-data.frame(relevant.xrf)
colnames(relevant.xrf)<-"relevant.x"

# intersect the main factors for glm and random forest
inner_join(data.frame(relevant.x),relevant.xrf,by="relevant.x")
```

```
##                                                    relevant.x
## 1                                                Region_South
## 2                                              Dexa_During_Rx_Y
## 3          Comorb_Encounter_For_Screening_For_Malignant_Neoplasms_Y
## 4                                Comorb_Encounter_For_Immunization_Y
## 5  Comorb_Encntr_For_General_Exam_W_O_Complaint._Susp_Or_Reprtd_Dx_Y
## 6                                     Comorb_Vitamin_D_Deficiency_Y
## 7           Comorb_Other_Joint_Disorder_Not_Elsewhere_Classified_Y
## 8                          Comorb_Long_Term_Current_Drug_Therapy_Y
## 9       Comorb_Personal_History_Of_Other_Diseases_And_Conditions_Y
## 10          Comorb_Other_Disorders_Of_Bone_Density_And_Structure_Y
## 11                Comorb_Personal_history_of_malignant_neoplasm_Y
## 12                      Comorb_Gastro_esophageal_reflux_disease_Y
## 13                                           Concom_Narcotics_Y
## 14                        Concom_Systemic_Corticosteroids_Plain_Y
```

```
## 15                                Concom_Macrolides_And_Similar_Types_Y
## 16                                  Concom_Broad_Spectrum_Penicillins_Y
## 17                                        Concom_Anaesthetics_General_Y
## 18                                             Concom_Viral_Vaccines_Y
```

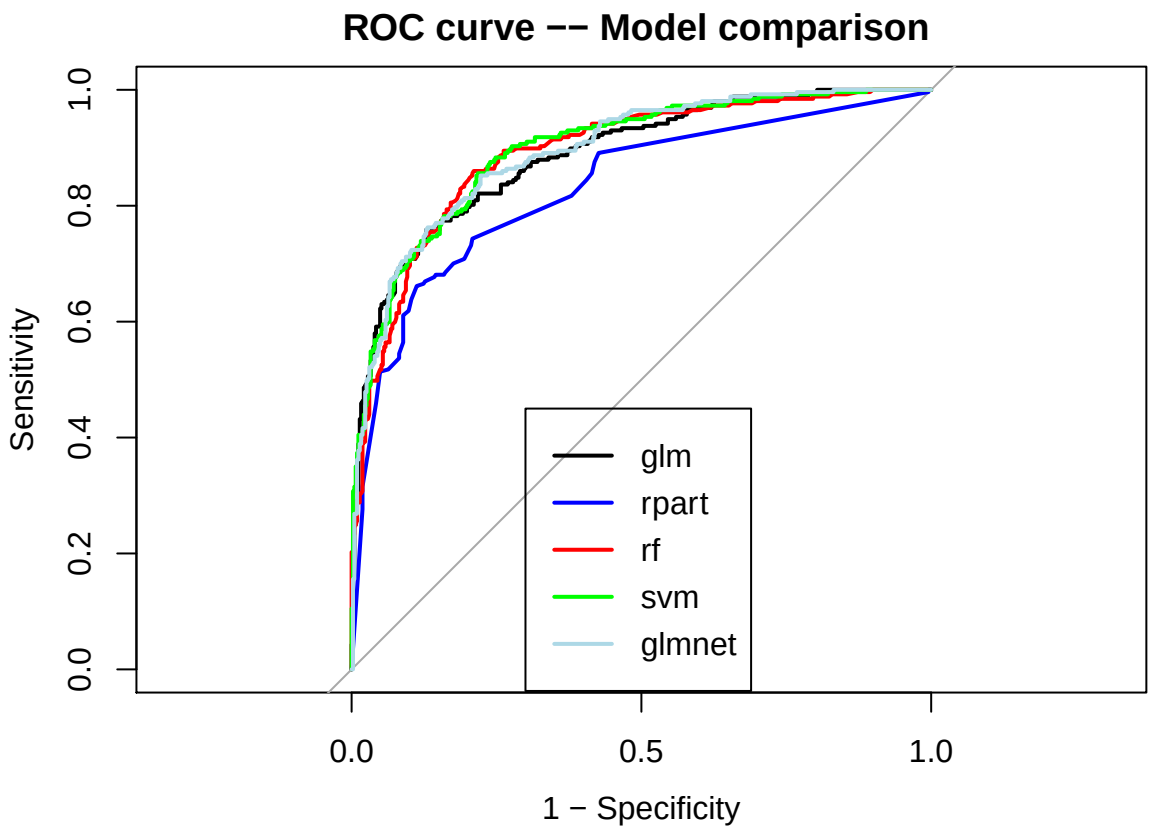
2. AUC

Then the second metric chosen for the model evaluation is the AUC (area under curve ROC). As first step we plot the ROC curve with the package pROC. Then the AUC for every model is calculated and the results are placed together with accuracy to have a complete overview for the evaluation. As concern as the AUC results, we see that, apart from rpart, the other models exhibit very similar values close to 89%-90%.

```
### Model evaluation - ROC

test_y<-ifelse(test$Persistency_Flag=="Persistent",1,0)

#1. ROC glm
test_pred_glm<-predict(model_glm,test,type="prob")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```
# plot ROC curve for glm model
roc_mod_glm = roc(test_y, test_pred_glm$Persistent,levels = c(0, 1), direction = "<")
plot(roc_mod_glm, main="ROC curve -- Model comparison ",legacy.axes=TRUE)

#2. ROC rpart
test_pred_rpart<-predict(model_rpart,test,type="prob")
# plot ROC curve for rpart model
roc_mod_rpart = roc(test_y, test_pred_rpart$Persistent,levels = c(0, 1), direction = "<")
lines(roc_mod_rpart,col="blue")

#3. ROC random forest
test_pred_rf<-predict(model_rf,test,type="prob")
# plot ROC curve for random forest
roc_mod_rf<- roc(test_y, test_pred_rf$Persistent,levels = c(0, 1), direction = "<")
lines(roc_mod_rf,col="red")

#4. ROC svm
test_pred_svm<-predict(model_svm,test,type="prob")
# plot ROC curve for svm model
roc_mod_svm<- roc(test_y, test_pred_svm$Persistent,levels = c(0, 1), direction = "<")
lines(roc_mod_svm,col="green")

#5. ROC glmnet
test_pred_glmnet<-predict(model_glmnet,test,type="prob")
# plot ROC curve for glmnet model
roc_mod_glmnet<- roc(test_y, test_pred_glmnet$Persistent,levels = c(0, 1), direction = "<")
lines(roc_mod_glmnet,col="light blue")

# add legend
legend(0.7,0.45, c('glm','rpart','rf','svm','glmnet'),lty=c(1,1),
       lwd=c(2,2),col=c('black','blue','red','green','light blue'))
```




```
# Area under curve ROC
AUC_glm<-auc(roc_mod_glm)
AUC_rpart<-auc(roc_mod_rpart)
AUC_rf<-auc(roc_mod_rf)
AUC_svm<-auc(roc_mod_svm)
AUC_glmnet<-auc(roc_mod_glmnet)
AUC_results<-c(AUC_glm,AUC_rpart,AUC_rf,AUC_svm,AUC_glmnet)

# recap of AUC and accuracy results
results<-cbind(accuracy_results,AUC_results)
knitr::kable(results)
```

method	Accuracy	AUC_results
glm	0.8230994	0.8872415
rpart	0.7894737	0.8317508
random forest	0.8055556	0.8910825
svm	0.8274854	0.8975569
glmnet	0.8318713	0.8941853

Recap: AUC, accuracy results and other metrics

Considering together both the metrics, there is not a model with significantly better figures: glmnet has the best accuracy, while svm has the best AUC. But the performances for glmnet,svm and glm are very similar.

```
# recap of AUC and accuracy results
results<-cbind(accuracy_results,AUC_results)
knitr::kable(results)
```

method	Accuracy	AUC_results
glm	0.8230994	0.8872415
rpart	0.7894737	0.8317508
random forest	0.8055556	0.8910825
svm	0.8274854	0.8975569
glmnet	0.8318713	0.8941853

In order to have a complete overview of the model results we add also the performances in terms of sensitivity and specificity. We calculate the sensitivity, considering as positive outcome (Y=1) the class "Persistent". Assuming this, all the models put in evidence an high specificity, while sensitivity is low.

```
# Calculation of Sensitivity and Specificity

glm<-confusionMatrix(y_pred_glm,test$Persistency_Flag,
                     positive="Persistent")$byClass[(c("Sensitivity","Specificity"))]
rpart<-confusionMatrix(y_pred_rpart,test$Persistency_Flag,
                       positive="Persistent")$byClass[(c("Sensitivity","Specificity"))]
rf<-confusionMatrix(y_pred_rf,test$Persistency_Flag,
                    positive="Persistent")$byClass[(c("Sensitivity","Specificity"))]
svm<-confusionMatrix(y_pred_svm,test$Persistency_Flag,
                     positive="Persistent")$byClass[(c("Sensitivity","Specificity"))]
glmnet<-confusionMatrix(y_pred_glmnet,test$Persistency_Flag,
                        positive="Persistent")$byClass[(c("Sensitivity","Specificity"))]


cm_results<-data.frame(rbind(glm,rpart,rf,svm,glmnet))

# final table
results<-cbind(results[,2:3],cm_results)
knitr::kable(results)
```

	Accuracy	AUC_results	Sensitivity	Specificity
glm	0.8230994	0.8872415	0.7120623	0.8899297
rpart	0.7894737	0.8317508	0.6770428	0.8571429
rf	0.8055556	0.8910825	0.6186770	0.9180328
svm	0.8274854	0.8975569	0.7081712	0.8992974
glmnet	0.8318713	0.8941853	0.6848249	0.9203747

Ensemble and further steps

Considering the results, we try to build an ensemble prediction using all the trained models, using the average of the class probabilities estimation on the test set. There is an improvement in the results, as regard as accuracy, but not really significant.

```
### Ensemble the results using the average of the class probabilities
y_ensemble<-ifelse((test_pred_glm$Persistent+test_pred_rpart$Persistent+test_pred_rf$Persistent+
                      test_pred_svm$Persistent+test_pred_glmnet$Persistent)/5>0.5,1,0)
y_pred_agg<-factor(ifelse(y_ensemble==1,"Persistent","Non.Persistent"))

# Accuracy calculation
accuracy_agg<-confusionMatrix(y_pred_agg,test$Persistency_Flag) $overall["Accuracy"]
accuracy_agg
```

```
##  Accuracy
## 0.8347953
```

```
# Area under curve ROC
test_pred_agg<-(test_pred_glm$Persistent+test_pred_rpart$Persistent+test_pred_rf$Persistent+
  test_pred_svm$Persistent+test_pred_glmnet$Persistent)/5
AUC_mod_agg<- auc(roc(test_y, test_pred_agg,levels = c(0, 1), direction = "<"))

# Calculation of Sensitivity and Specificity
sens_agg<-confusionMatrix(y_pred_agg,test$Persistency_Flag,
                        positive="Persistent")$byClass[(c("Sensitivity","Specificity"))]

# performance for simple ensemble using the average of the results of the models
ensemble_avg_mod<-c(accuracy_agg,AUC_mod_agg,sens_agg)
```

Model ensemble with stacking

Exploring the potentials of the package caretEnsemble, we try also to combine the models chosen before via stacking.The idea behind this method is to handle a machine learning problem using different types of models, using which we can make intermediate predictions and then add a new model that can learn from the intermediate predictions. It is quite a challenging issue, but we try in order to improve the achieved results. We have to define a set of parameters for training and cross-validation, a list of algoritms and then we can stake the models. We choose three models already tuned (svm,glmnet and random forest) and we use also glm as the method used for stacking. The reason is to have a "meta model" easy to be interpreted. By dividing the coefficients of the meta-model with the sum of them, it is possible to see the different contribution of the single models, that, in this case, seems to be quite balanced.

```
### caret ensemble: stacking

set.seed(77, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(77)
```

```
## Warning in set.seed(77, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

```
# suppress the warnings for better readability of the report
options(warn=-1)

# train parameters:
# resampling method = cv
# classProbs = TRUE: class probabilities are computed for classification models
# twoClassSummary computes sensitivity, specificity and the area under the ROC curve
```

```
my_control <- trainControl(method = 'cv', # for "cross-validation"
                           number = 20, # number of k-folds
                           savePredictions = 'final',
                           summaryFunction = twoClassSummary, classProbs = TRUE,
                           allowParallel = TRUE)

set.seed(123, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(123)

# list of the base algoritm - training on the training set
# we use the models with the parameters already tuned before

model_list <- caretList(outcome~.,data=training_set,
                        trControl = my_control,
                        methodList = NULL,
                        tuneList = list(
                          glmnet1=caretModelSpec(method='glmnet', tuneGrid=data.frame(alpha=0,lambda=0.
                          svm1=caretModelSpec(method='svmRadial', tuneGrid=data.frame(sigma=0.008001509
                          rf1=caretModelSpec(method="rf", tuneGrid=data.frame(.mtry=5))))

set.seed(85, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(85)

# model stacking with GLM ###############

glm_ensemble <- caretStack(model_list,
                           method = 'glm',
                           metric = 'ROC',
                           trControl = my_control)
# model summary
summary(glm_ensemble)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.6331  -0.5881  -0.3624   0.4834   2.5753
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   4.1504     0.2198  18.881  < 2e-16 ***
## glmnet1      -5.4942     0.9335  -5.886 3.96e-09 ***
## svm1          2.9060     0.8869   3.277  0.00105 **
## rf1          -4.9391     0.6779  -7.286 3.19e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 3629.9  on 2739  degrees of freedom
## Residual deviance: 2262.3  on 2736  degrees of freedom
## AIC: 2270.3
##
```

```
## Number of Fisher Scoring iterations: 5
```

```
# prediction on the test set and calculation of the accuracy of the model
predict_ens_glm <- predict(glm_ensemble, newdata = test)

accuracy_ens<-confusionMatrix(predict_ens_glm,test$Persistency_Flag) $overall["Accuracy"]
accuracy_ens
```

```
##  Accuracy
## 0.8260234
```

```
# coefficient for the base models
CF <- coef(glm_ensemble$ens_model$finalModel)[-1]
CF/sum(CF)
```

```
##    glmnet1       svm1        rf1
##  0.7298979 -0.3860525  0.6561547
```

```
# Area under curve ROC
model_preds <- predict(glm_ensemble, newdata=test, type="prob")
auc_mod_ens<- auc(roc(test_y, model_preds))
```

```
## Setting levels: control = 0, case = 1
```

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

```
# Calculation of Sensitivity and Specificity on the test set
sens_ens<-confusionMatrix(predict_ens_glm,test$Persistency_Flag,
                          positive="Persistent")$byClass[(c("Sensitivity","Specificity"))]

# performance for ensemble via stacking glm
stacking_glm<-c(accuracy_ens,auc_mod_ens,sens_ens)
```

We try also with another meta-model in order to have a comparison. For this issue we choose random forest.

```
set.seed(72, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(72)

# model stacking with random forest - caretStack
rf_ensemble <- caretStack(model_list,
                          method = 'rf',
                          metric = 'ROC',
                          trControl = my_control)
```

```
## note: only 2 unique complexity parameters in default grid. Truncating the grid to 2 .
```

```
# prediction on the test set and calculation of the accuracy of the model
predict_ens_rf <- predict(rf_ensemble, newdata = test)
accuracy_ens_rf<-confusionMatrix(predict_ens_rf,test$Persistency_Flag) $overall["Accuracy"]
accuracy_ens_rf
```

```
##  Accuracy
## 0.8304094
```

```
# Area under curve ROC
model_preds_rf <- predict(rf_ensemble, newdata=test, type="prob")
auc_mod_ens_rf<- auc(roc(test_y, model_preds_rf))
```

```
## Setting levels: control = 0, case = 1
```

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

```
# Calculation of Sensitivity and Specificity on the test set
sens_ens_rf<-confusionMatrix(predict_ens_rf,test$Persistency_Flag,
                             positive="Persistent")$byClass[(c("Sensitivity","Specificity"))]

# performance for ensemble via stacking rf
stacking_rf<-c(accuracy_ens_rf,auc_mod_ens_rf,sens_ens_rf)
```

Results for stacking

Considering the results of the new models, the stacking with glm doesn't increase the performance of the previous trial (ensemble using the average of class prediction probabilities). Stacking with random forest could be another possible choice, but the results are not better.

```
# table of results updated ################

# results summary
results<-rbind(results,ensemble_avg_mod,stacking_glm,stacking_rf)
rownames(results)<-NULL
final_results<-data.frame(model=c("glm","rpart","rf","svm","glmnet",
                                  "ensemble (average)","stacking with glm","stacking with rf"),
                          results)
knitr::kable(final_results)
```

model	Accuracy	AUC_results	Sensitivity	Specificity
glm	0.8230994	0.8872415	0.7120623	0.8899297
rpart	0.7894737	0.8317508	0.6770428	0.8571429
rf	0.8055556	0.8910825	0.6186770	0.9180328
svm	0.8274854	0.8975569	0.7081712	0.8992974
glmnet	0.8318713	0.8941853	0.6848249	0.9203747
ensemble (average)	0.8347953	0.8939301	0.7003891	0.9156909
stacking with glm	0.8260234	0.8924721	0.6964981	0.9039813
stacking with rf	0.8304094	0.8821249	0.7120623	0.9016393

Further steps: sensitivity improvement

All the models have a low performance as regard as sensitivity, therefore we try to improve the results using cost-sensitive models. It is possible to find information about these models in: Kuhn, M., and K. Johnson. 2016. Applied Predictive Modeling, chapter 16. One possible way for implementing these models is to use rpart package with the parms argument, that offers different options, among which loss matrix. The loss matrix is structured with actuals on the rows and predictions on the columns. In corrispondence to false negative (FN) position we put 2.2, while we put 1 for false positive (FP). So in this case it's 2.2 times

worse to generate a false negative than a false positive. We choose these parameters, in order to balance the results of sensitivity and specificity: we find in fact similar values between 75%-80%. The accuracy, using this cost-sensitive model, has a little variation (less than 1% of decrease).It's interesting to observe that if we choose a cost of 3 for FN, the sensitivity reaches 90%, but the specificity drops to 56%.

```
# function to define a different set of performance measures
# Persistent is the level 2, while Non.Persistent is the level 1

fourStats<-function(data,lev=levels(data$obs),model=NULL)
  {
  accKapp<-postResample(data[,"pred"],data[,"obs"])
  out<-c(accKapp,
         sensitivity(data[,"pred"],data[,"obs"],lev[2]),
         specificity(data[,"pred"],data[,"obs"],lev[1]))
  names(out)[3:4]<-c("Sens","Spec")
  out
}

# different trainControl setting coherent with
# the new set of performance measures

ctrl_sens<-trainControl(method="cv",
                    classProbs=TRUE,
                    summaryFunction=fourStats,
                    verboseIter=FALSE)

# cost sensitive training matrix (FN=2.2,FP=1)
# reference value on the rows and predictions on the columns

costMatrix<-matrix(c(0,2.2,1,0),ncol=2)
rownames(costMatrix)<-levels(as.factor(outcome))
colnames(costMatrix)<-levels(as.factor(outcome))

set.seed(1103, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1103)

# model rpart with costMatrix training on the train set
model_rpart_sens<-train(x=features,y=outcome,method="rpart",
                    metric="Kappa",
                    trControl = ctrl_sens,
                    parms=list(loss=costMatrix))

model_rpart_sens
```

```
## CART
##
## 2740 samples
##   81 predictor
##    2 classes: 'Non.Persistent', 'Persistent'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 2466, 2466, 2467, 2466, 2466, 2466, ...
## Resampling results across tuning parameters:
##
```

```
##   cp          Accuracy   Kappa      Sens       Spec
##   0.01162791  0.7617295  0.5064657  0.7510269  0.7682043
##   0.01744186  0.7240887  0.4482686  0.7917196  0.6832405
##   0.42635659  0.3766417  0.0000000  1.0000000  0.0000000
##
## Kappa was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.01162791.
```

```
# prediction on the test set
y_pred_rpart_sens<-predict(model_rpart_sens,test)
# accuracy calculation
accuracy_rpart_sens<-confusionMatrix(y_pred_rpart_sens,test$Persistency_Flag) $overall["Accuracy"]
accuracy_rpart_sens
```

```
##  Accuracy
## 0.7821637
```

```
# Area under curve ROC
model_preds_rpart_sens <- predict(model_rpart_sens, newdata=test, type="prob")
auc_mod_rpart_sens<- auc(roc(test_y, model_preds_rpart_sens$Persistent))
```

```
## Setting levels: control = 0, case = 1

## Setting direction: controls < cases
```

```
# sensitivity-specificity calculation
rpart_sens<-confusionMatrix(y_pred_rpart_sens,test$Persistency_Flag,
                            positive="Persistent")$byClass[(c("Sensitivity","Specificity"))]
rpart_with_cost_matrix<-c(accuracy_rpart_sens,auc_mod_rpart_sens,rpart_sens)


# trial with a different cost matrix ############
# change the cost matrix with FN=3 ##############

costMatrix2<-matrix(c(0,3,1,0),ncol=2)
rownames(costMatrix2)<-levels(as.factor(outcome))
colnames(costMatrix2)<-levels(as.factor(outcome))

set.seed(1109, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1109)

# model rpart with costMatrix training on the train set
model_rpart_sens_2<-train(x=features,y=outcome,method="rpart",
                          metric="Kappa",
                          trControl = ctrl_sens,
                          parms=list(loss=costMatrix2))

# prediction on the test set
y_pred_rpart_sens_2<-predict(model_rpart_sens_2,test)

# sensitivity-specificity calculation
rpart_sens2<-confusionMatrix(y_pred_rpart_sens_2,test$Persistency_Flag,
                            positive="Persistent")$byClass[(c("Sensitivity","Specificity"))]
rpart_sens2
```

```
## Sensitivity Specificity
##   0.8949416   0.5620609
```

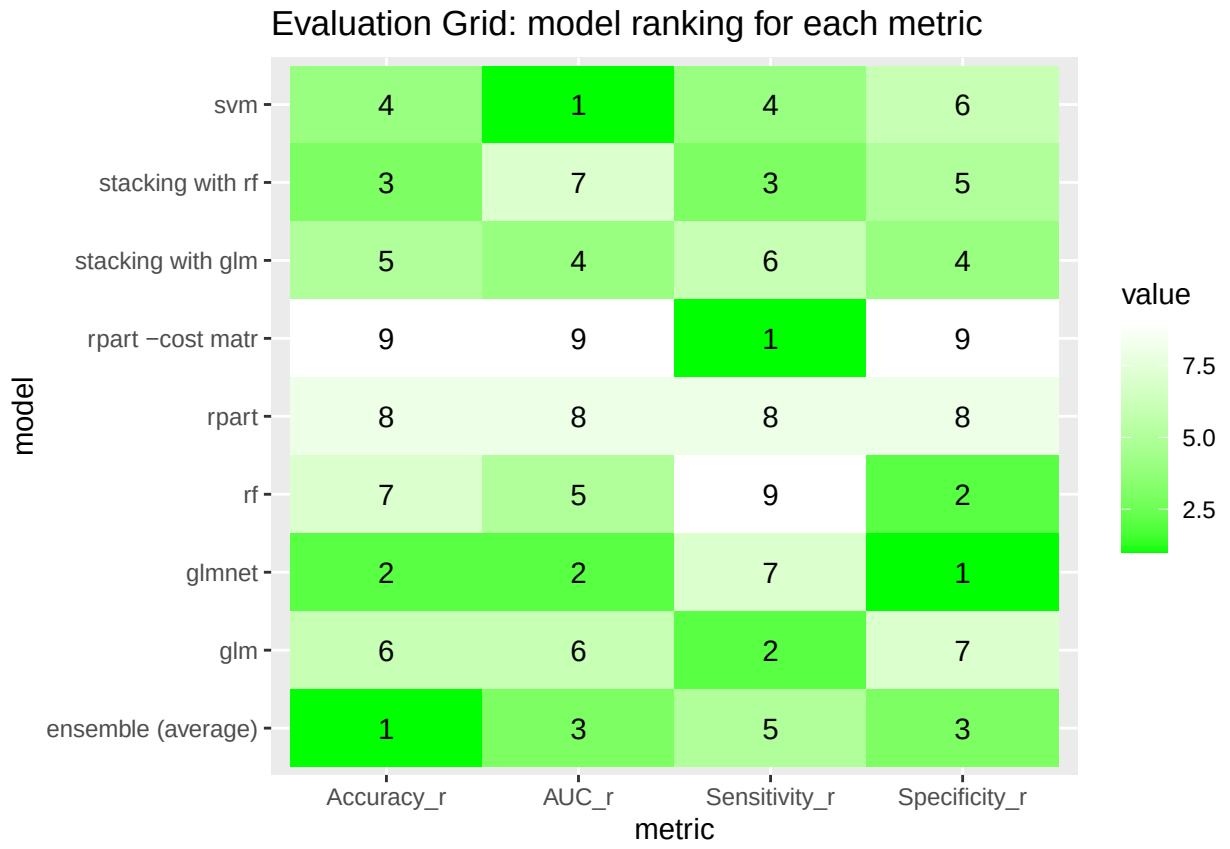
Final recap

Ultimately we put together all the results and then we plot the model ranking for each metric into an evaluation grid. It appears evident that the introduction of new models has produced, in any case, selective improvement of one metric, but at the same time others metrics are not impacted positively. The choice of the metric to be tuned is therefore critical for the model selection.

```
# final recap
last<-final_results %>% add_row(model = "rpart -cost matr", Accuracy = accuracy_rpart_sens,
                                AUC_results=auc_mod_rpart_sens[1],Sensitivity=rpart_sens[1],
                                Specificity=rpart_sens[2])

knitr::kable(last)
```

model	Accuracy	AUC_results	Sensitivity	Specificity
glm	0.8230994	0.8872415	0.7120623	0.8899297
rpart	0.7894737	0.8317508	0.6770428	0.8571429
rf	0.8055556	0.8910825	0.6186770	0.9180328
svm	0.8274854	0.8975569	0.7081712	0.8992974
glmnet	0.8318713	0.8941853	0.6848249	0.9203747
ensemble (average)	0.8347953	0.8939301	0.7003891	0.9156909
stacking with glm	0.8260234	0.8924721	0.6964981	0.9039813
stacking with rf	0.8304094	0.8821249	0.7120623	0.9016393
rpart -cost matr	0.7821637	0.8313453	0.7665370	0.7915691

```
# substitute the results with ranking of the relative column/metric

rank_last<-last %>% mutate(Accuracy_r = rank(-Accuracy, ties.method = 'first')) %>%
  mutate(AUC_r = rank(-AUC_results, ties.method = 'first')) %>%
  mutate(Sensitivity_r = rank(-Sensitivity, ties.method = 'first')) %>%
  mutate(Specificity_r = rank(-Specificity, ties.method = 'first')) %>%
  select(model,Accuracy_r,AUC_r,Sensitivity_r,Specificity_r)

# transformation of the table in a tidy format in order to make a heatmap
# with the ranking of the results for each metric
tidylast<-rank_last %>% gather(key=metric,value=value,-c(model))

# plot the heatmap of the ranking for each metric
ggplot(tidylast, aes(x = metric, y = model, fill = value)) + geom_tile()+
  geom_text(aes(label = value))+
  scale_fill_gradient(high = "white", low = "green")+
  ggtitle("Evaluation Grid: model ranking for each metric")
```



Conclusion

The analyzed project is an example of classification problem, in presence of a complex dataset with an high number of predictors. For this reason we have tried first to get a rough idea of the features that impact more on the outcome using the tools of exploratory analysis. With a different approach for categorical and non categorical predictors, we put in evidence some insights from the data in terms of variable importance and correlations across the dataset. Then we train five models (logistic regression, classification trees with rpart and random forest, svm and glmnet) and we evaluate them using two metrics (accuracy and AUC). During the training phase we have faced critical issues as regard as computational time for tuning parameters of the models. This could have limited the results achieved. More over from the evaluation of the models, it doesn't appear clearly a best model: the winner is different considering the two metrics (accuracy: glmnet, AUC: SVM), but the performances for logistic regression, glm with penalty and SVM are quite similar.Broadening the analysis to other metrics, all the models put in evidence an high specificity and low results for sensitivity. Then we try to improve accuracy with ensemble techniques. First we try with a simple ensemble of the results of the models that produces an improvement on the class prediction, but not really significant. Therefore we combine three algorithms (svm, glmnet and random forest) using staking, but the results are not better than the previous aggregated ones. In order to improve sensitivity we test also an option of rpart package, that assigns different costs to model errors (false negative and false positive). In this way sensitivity and specificity can be balanced. I think that, for further steps of improvement, it could be possible to try a wider range of models and to implement sophisticated techniques, but also an in-depth study of predictors and outcome from a medical perspective could help in order to choose the best mixture of metrics to be tuned.

References

- Project Dataset from Kaggle
- The caret package
- Kuhn, M., and K. Johnson. 2016. Applied Predictive Modeling. New York: Springer
- An Introduction to glmnet
- A Brief Introduction to caretEnsemble