Machine Learning Models for Credit Card Default Risk Assessment

Faccin Giacomo

2026-01-24

```
library(dplyr)
```

```
##
## Caricamento pacchetto: 'dplyr'

## I seguenti oggetti sono mascherati da 'package:stats':
##
##     filter, lag

## I seguenti oggetti sono mascherati da 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
library(tidyr)
```

Data Dawnload

```
credit_card_balanced<-read.csv("credit_card_balance.csv")
bureau_data<-read.csv("bureau.csv")
application_train_data<-read.csv("application_train.csv")

Description_VAR<-read.csv("HomeCredit_columns_description.csv")
```

Data Marging

We've aggregated the observations on the data set *bereau_data* because it collected the observation with more rows for the same client.

We've applied the same logic structure to the data set *credit_card_balanced*. We've aggregate by the variable *SK_ID_CURR*, created a new variable

```
new_balanced_card<-credit_card_balanced%>%
   mutate(
    utilization_ratio = AMT_BALANCE / AMT_CREDIT_LIMIT_ACTUAL
   )%>%
  group_by(SK_ID_CURR)%>%
  summarise(mean_balance = mean(AMT_BALANCE, na.rm = TRUE),
    Max_balance = max(AMT_BALANCE, na.rm = TRUE),
    Mean_utilization = mean(utilization_ratio, na.rm = TRUE),
    Max_dpd = max(SK_DPD, na.rm = TRUE),
    Mean_dpd = mean(SK_DPD, na.rm = TRUE))
```

Here it can be observed the following data sets *new_balanced_card*, *new_bureau* and *application_train_data* have been merged.

```
Data_set<-application_train_data%>%
  left_join(new_bureau, by = "SK_ID_CURR") %>%
  left_join(new_balanced_card, by = "SK_ID_CURR")
```

In this case we do a strong assumption if the observation is NA means there isn't exposition therefore is NA = 0.

we've implemented a little check on the following DF.

```
# To no the number
table(Data_set$TARGET)
```

```
##
##      0      1
## 282686  24825
```

```
#data set dimension

dim(Data_set)
```

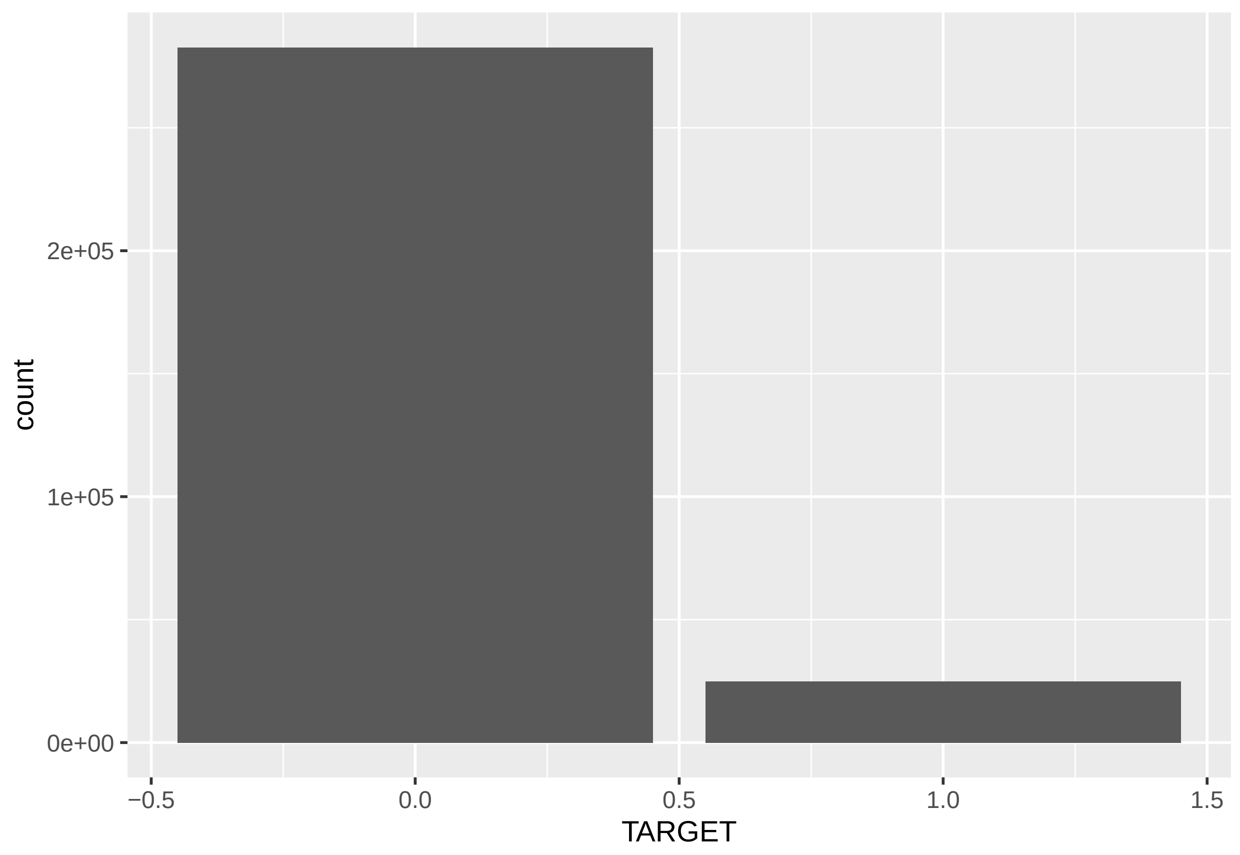
```
## [1] 307511    133
```

```
Data_set<-Data_set%>%
  mutate(across(where(is.numeric), ~ replace_na(., 0)))
```

EDA

The variable target has been investigated with a bar plot, it pretty evident that there is a problem of *Class Imbalance*.

```
ggplot(Data_set)+
  geom_bar(aes(x=TARGET))
```

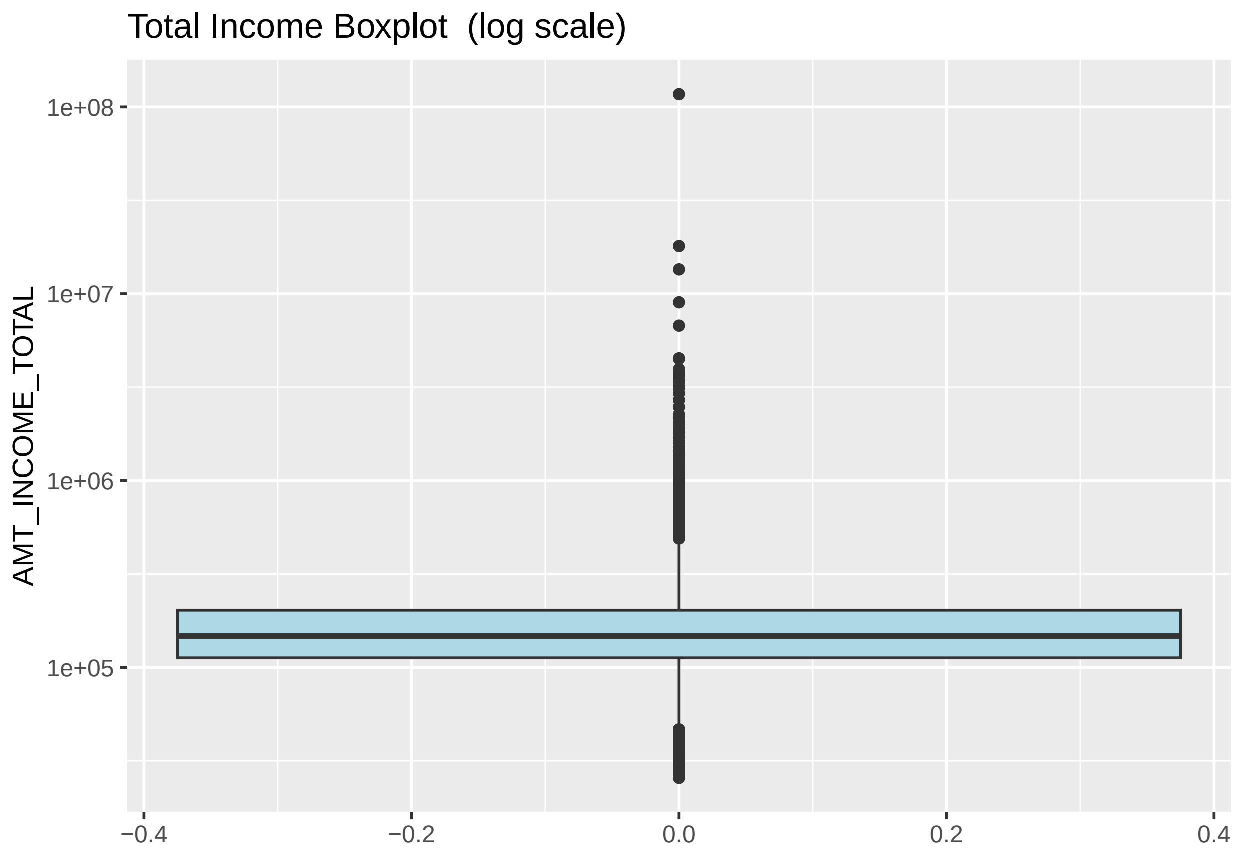


The variable *AMT_INCOME_TOTAL* had been explored before it was analyzed to under how it was distributed.

```r
library(scales)
```

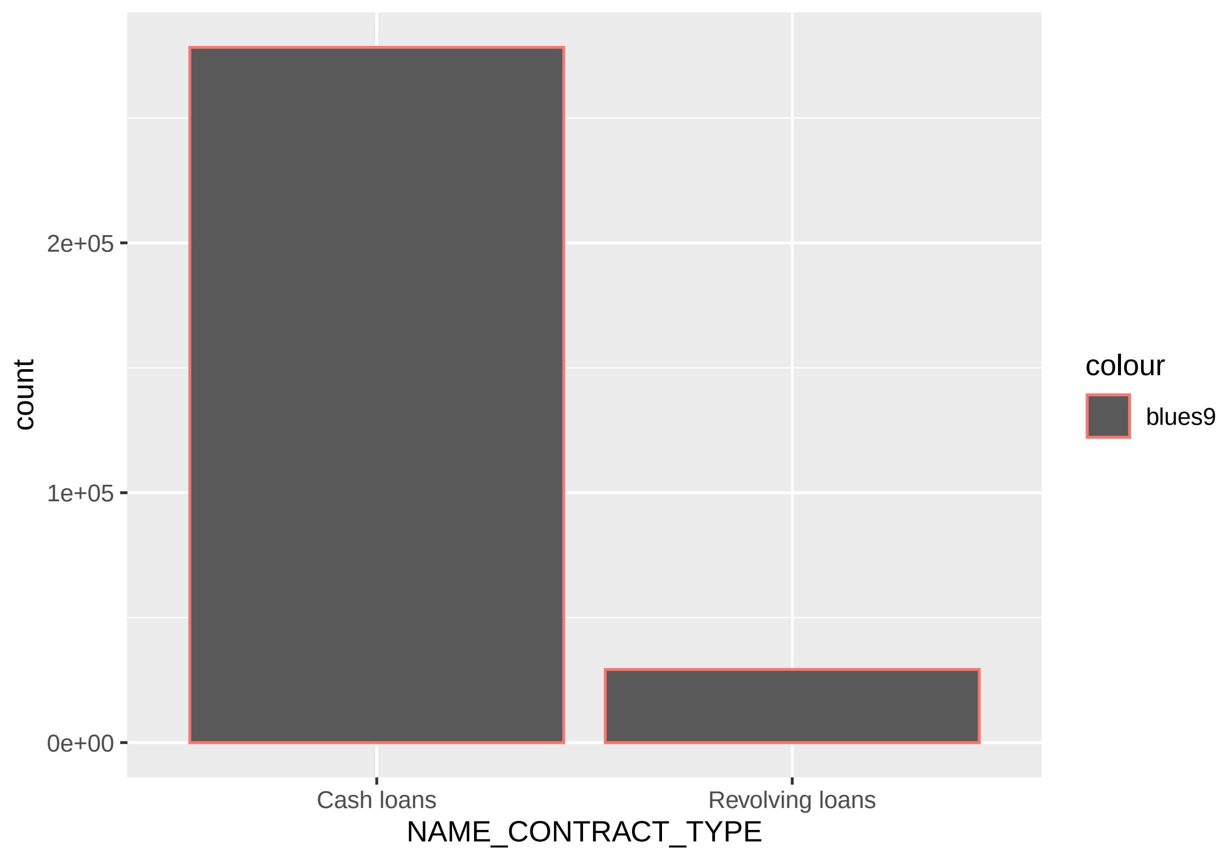
```
## Warning: il pacchetto 'scales' è stato creato con R versione 4.4.3
```

```r
ggplot(Data_set) +
  geom_boxplot(aes(y = AMT_INCOME_TOTAL), fill = "lightblue") +
  scale_y_log10() +
  labs(title = "Total Income Boxplot  (log scale)", y = "AMT_INCOME_TOTAL")
```



There are couple of outliers in the range 3 to 6 million of dollars.

```
ggplot(Data_set)+
  geom_bar(aes(x=NAME_CONTRACT_TYPE,col="blues9"))
```



Splitting the the data set in training set and data set

```
set.seed(123)
index<-sample(x=1:nrow(Data_set), size = 0.7*nrow(Data_set))

training_set<-Data_set[index,]

test_set<-Data_set[-index,]
```

check training set and data set

```
nrow(training_set)+nrow(test_set)
```

```
## [1] 307511
```

```
nrow(Data_set)
```

```
## [1] 307511
```

A possible solution for the imbalance problem with the target balance is to created a balanced training set with all default and ad not default quantity of them.

```r
table(training_set$TARGET)
```

```
##
##      0      1
## 197809  17448
```

```r
prop.table(table(training_set$TARGET))
```

```
##
##         0         1
## 0.9189434 0.0810566
```

```r
## we've split the training by target variable =0 or =1
train0<-training_set[training_set$TARGET==0,]

train1<-training_set[training_set$TARGET==1,]

set.seed(123)

index_0<-sample(x=1:nrow(train0), size= nrow(train1))

train_sample_0<-train0[index_0,]

## The official training balanced has been created to solve the imbalance class
training_balanced_set<-rbind(train_sample_0,train1)

# small check on the proportion
nrow(training_balanced_set)
```

```
## [1] 34896
```

```r
prop.table(table(training_balanced_set$TARGET))
```

```
##
##   0   1
## 0.5 0.5
```

Scaling the variable to smooth the effect of the outliers

To address the presence of skewed distributions and negative values in selected numerical variables, a Yeo–Johnson transformation was applied using the caret preprocessing framework. The transformation parameters were estimated exclusively on the training set and subsequently applied to the test set, thereby ensuring consistency and preventing data leakage. This approach allows for variance stabilization and reduces the influence of extreme values while accommodating non-positive observations

```r
library(caret)
```

```
## Warning: il pacchetto 'caret' è stato creato con R versione 4.4.3
## Caricamento del pacchetto richiesto: lattice
```

```r
pp <- preProcess(
  training_balanced_set[, c("Total_debt", "mean_balance")],
  method = "YeoJohnson"
)

training_balanced_set[, c("Total_debt", "mean_balance")] <-
  predict(pp, training_balanced_set[, c("Total_debt", "mean_balance")])

test_set[, c("Total_debt", "mean_balance")] <-
  predict(pp, test_set[, c("Total_debt", "mean_balance")])
```

The following variable, AMT_INCOME_TOTAL, AMT_CREDIT and AMT_ANNUITY, have been scaled to reduce the influence of extreme values, the principal monetary variables were log-transformed using the natural logarithm of one plus the variable (log1p).

```r
# scaling with log for training set
training_balanced_set$AMT_INCOME_TOTAL<-log1p(training_balanced_set$AMT_INCOME_TOTAL)
training_balanced_set$AMT_CREDIT<-log1p(training_balanced_set$AMT_CREDIT)
training_balanced_set$AMT_ANNUITY<-log1p(training_balanced_set$AMT_ANNUITY)
training_balanced_set$Total_credit<-log1p(training_balanced_set$Total_credit)


training_balanced_set$Mean_overdue<-log1p(training_balanced_set$Mean_overdue)
training_balanced_set$Mean_utilization<-log1p(training_balanced_set$Mean_utilization)
training_balanced_set$Mean_dpd<-log1p(training_balanced_set$Mean_dpd)


# scaling with log for test set
test_set$AMT_INCOME_TOTAL<-log1p(test_set$AMT_INCOME_TOTAL)
test_set$AMT_CREDIT<-log1p(test_set$AMT_CREDIT)
test_set$AMT_ANNUITY<-log1p(test_set$AMT_ANNUITY)
test_set$Total_credit<-log1p(test_set$Total_credit)


test_set$Mean_overdue<-log1p(test_set$Mean_overdue)
test_set$Mean_utilization<-log1p(test_set$Mean_utilization)
test_set$Mean_dpd<-log1p(test_set$Mean_dpd)
```

The implemantation of the model

```r
library(randomForest)
```

```
## Warning: il pacchetto 'randomForest' è stato creato con R versione 4.4.3
```

```
## randomForest 4.7-1.2
```

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

```
##
## Caricamento pacchetto: 'randomForest'

## Il seguente oggetto è mascherato da 'package:ggplot2':
##
##     margin

## Il seguente oggetto è mascherato da 'package:dplyr':
##
##     combine
```

```
library(xgboost)
```

```
## Warning: il pacchetto 'xgboost' è stato creato con R versione 4.4.3
```

```
# WE remove all NA from training set and test set





# Solo per training e test set
training_balanced_set <- training_balanced_set[!is.na(training_balanced_set$TARGET), ]
test_set <- test_set[!is.na(test_set$TARGET), ]


# Converti in fattore dopo
training_balanced_set$TARGET <- as.factor(training_balanced_set$TARGET)
test_set$TARGET <- as.factor(test_set$TARGET)


table(test_set$TARGET)
```

```
##
##     0     1
## 84877  7377
```

```
table(training_balanced_set$TARGET)
```

```
##
##     0     1
## 17448 17448
```

```
## here we implement the random forest

RF_MODEL<-randomForest(TARGET ~ EXT_SOURCE_1 + EXT_SOURCE_2 + EXT_SOURCE_3 +
           AMT_INCOME_TOTAL + AMT_CREDIT + AMT_ANNUITY +
           DAYS_BIRTH + DAYS_EMPLOYED +
           NAME_CONTRACT_TYPE + CODE_GENDER + NAME_EDUCATION_TYPE + NAME_FAMILY_STATUS +
           FLAG_OWN_CAR + FLAG_OWN_REALTY + CNT_CHILDREN +
           REGION_POPULATION_RELATIVE,
  data = training_balanced_set,
  ntree = 500,
  mtry = 3,)
```

```
y_hat_RF<-predict(RF_MODEL,test_set, type = "response")

confusionMatrix(as.factor(y_hat_RF),test_set$TARGET, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 57678  2443
##          1 27199  4934
##
##                Accuracy : 0.6787
##                  95% CI : (0.6757, 0.6817)
##     No Information Rate : 0.92
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1376
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.66884
##             Specificity : 0.67955
##          Pos Pred Value : 0.15355
##          Neg Pred Value : 0.95937
##              Prevalence : 0.07996
##          Detection Rate : 0.05348
##    Detection Prevalence : 0.34831
##       Balanced Accuracy : 0.67419
##
##        'Positive' Class : 1
##
```

```
MISCLASSRATE<-mean(y_hat_RF!=test_set$TARGET)
MISCLASSRATE
```

```
## [1] 0.3213086
```

```
# Lista delle colonne da controllare
vars_to_check <- c("Total_credit", "Total_debt", "Mean_dpd", "Mean_utilization", "Mean_overdue")

# Controllo NAs
sapply(training_balanced_set[ , vars_to_check], function(x) sum(is.na(x)))
```

```
##     Total_credit       Total_debt         Mean_dpd Mean_utilization
##                0                0                0                0
##     Mean_overdue
##                0
```

```
# Controllo Inf o -Inf
sapply(training_balanced_set[ , vars_to_check], function(x) sum(!is.finite(x)))
```

```
##     Total_credit       Total_debt         Mean_dpd Mean_utilization
##                0                0                0               86
##     Mean_overdue
##                0
```

second model RAndom forest

```
RF_MODEL_2 <- randomForest(
  TARGET ~ EXT_SOURCE_1 + EXT_SOURCE_2 + EXT_SOURCE_3 +
    AMT_INCOME_TOTAL + AMT_CREDIT + AMT_ANNUITY + Total_credit + Total_debt + Mean_dpd + Mean_overdue +
    DAYS_BIRTH + DAYS_EMPLOYED +
    NAME_CONTRACT_TYPE + CODE_GENDER + NAME_EDUCATION_TYPE + NAME_FAMILY_STATUS +
    FLAG_OWN_CAR + FLAG_OWN_REALTY + CNT_CHILDREN +
    REGION_POPULATION_RELATIVE,
  data = training_balanced_set,
  ntree = 500,
  mtry = 3
)
```

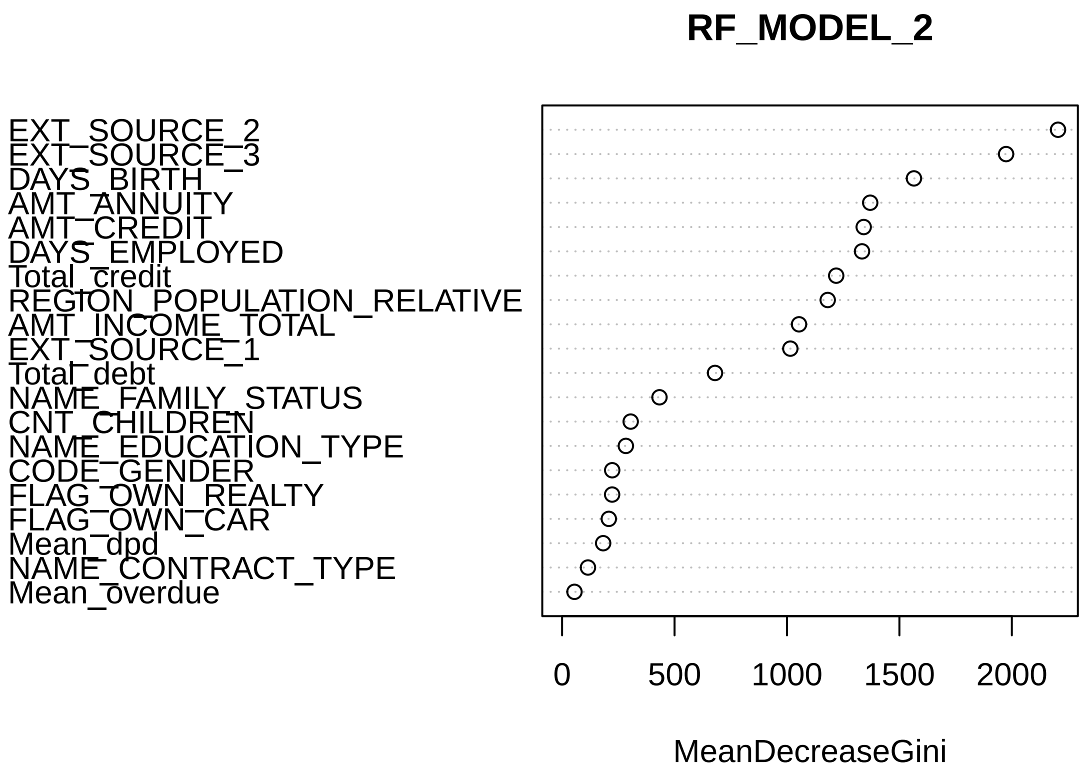
```
# Tutti i nomi delle colonne
all_vars <- colnames(training_balanced_set)

# Stampa per controllare
print(all_vars)
```

```
##   [1] "SK_ID_CURR"                   "TARGET"
##   [3] "NAME_CONTRACT_TYPE"           "CODE_GENDER"
##   [5] "FLAG_OWN_CAR"                 "FLAG_OWN_REALTY"
##   [7] "CNT_CHILDREN"                 "AMT_INCOME_TOTAL"
##   [9] "AMT_CREDIT"                   "AMT_ANNUITY"
##  [11] "AMT_GOODS_PRICE"              "NAME_TYPE_SUITE"
##  [13] "NAME_INCOME_TYPE"             "NAME_EDUCATION_TYPE"
##  [15] "NAME_FAMILY_STATUS"           "NAME_HOUSING_TYPE"
##  [17] "REGION_POPULATION_RELATIVE"   "DAYS_BIRTH"
##  [19] "DAYS_EMPLOYED"                "DAYS_REGISTRATION"
##  [21] "DAYS_ID_PUBLISH"              "OWN_CAR_AGE"
##  [23] "FLAG_MOBIL"                   "FLAG_EMP_PHONE"
##  [25] "FLAG_WORK_PHONE"              "FLAG_CONT_MOBILE"
##  [27] "FLAG_PHONE"                   "FLAG_EMAIL"
##  [29] "OCCUPATION_TYPE"              "CNT_FAM_MEMBERS"
##  [31] "REGION_RATING_CLIENT"         "REGION_RATING_CLIENT_W_CITY"
##  [33] "WEEKDAY_APPR_PROCESS_START"   "HOUR_APPR_PROCESS_START"
##  [35] "REG_REGION_NOT_LIVE_REGION"   "REG_REGION_NOT_WORK_REGION"
##  [37] "LIVE_REGION_NOT_WORK_REGION"  "REG_CITY_NOT_LIVE_CITY"
##  [39] "REG_CITY_NOT_WORK_CITY"       "LIVE_CITY_NOT_WORK_CITY"
##  [41] "ORGANIZATION_TYPE"            "EXT_SOURCE_1"
##  [43] "EXT_SOURCE_2"                 "EXT_SOURCE_3"
##  [45] "APARTMENTS_AVG"               "BASEMENTAREA_AVG"
##  [47] "YEARS_BEGINEXPLUATATION_AVG"  "YEARS_BUILD_AVG"
##  [49] "COMMONAREA_AVG"               "ELEVATORS_AVG"
##  [51] "ENTRANCES_AVG"                "FLOORSMAX_AVG"
##  [53] "FLOORSMIN_AVG"                "LANDAREA_AVG"
##  [55] "LIVINGAPARTMENTS_AVG"         "LIVINGAREA_AVG"
##  [57] "NONLIVINGAPARTMENTS_AVG"      "NONLIVINGAREA_AVG"
##  [59] "APARTMENTS_MODE"              "BASEMENTAREA_MODE"
##  [61] "YEARS_BEGINEXPLUATATION_MODE" "YEARS_BUILD_MODE"
##  [63] "COMMONAREA_MODE"              "ELEVATORS_MODE"
##  [65] "ENTRANCES_MODE"               "FLOORSMAX_MODE"
##  [67] "FLOORSMIN_MODE"               "LANDAREA_MODE"
##  [69] "LIVINGAPARTMENTS_MODE"        "LIVINGAREA_MODE"
##  [71] "NONLIVINGAPARTMENTS_MODE"     "NONLIVINGAREA_MODE"
##  [73] "APARTMENTS_MEDI"              "BASEMENTAREA_MEDI"
##  [75] "YEARS_BEGINEXPLUATATION_MEDI" "YEARS_BUILD_MEDI"
##  [77] "COMMONAREA_MEDI"              "ELEVATORS_MEDI"
##  [79] "ENTRANCES_MEDI"               "FLOORSMAX_MEDI"
##  [81] "FLOORSMIN_MEDI"               "LANDAREA_MEDI"
##  [83] "LIVINGAPARTMENTS_MEDI"        "LIVINGAREA_MEDI"
##  [85] "NONLIVINGAPARTMENTS_MEDI"     "NONLIVINGAREA_MEDI"
##  [87] "FONDKAPREMONT_MODE"           "HOUSETYPE_MODE"
##  [89] "TOTALAREA_MODE"               "WALLSMATERIAL_MODE"
##  [91] "EMERGENCYSTATE_MODE"          "OBS_30_CNT_SOCIAL_CIRCLE"
##  [93] "DEF_30_CNT_SOCIAL_CIRCLE"     "OBS_60_CNT_SOCIAL_CIRCLE"
##  [95] "DEF_60_CNT_SOCIAL_CIRCLE"     "DAYS_LAST_PHONE_CHANGE"
##  [97] "FLAG_DOCUMENT_2"              "FLAG_DOCUMENT_3"
##  [99] "FLAG_DOCUMENT_4"              "FLAG_DOCUMENT_5"
## [101] "FLAG_DOCUMENT_6"              "FLAG_DOCUMENT_7"
## [103] "FLAG_DOCUMENT_8"              "FLAG_DOCUMENT_9"
## [105] "FLAG_DOCUMENT_10"             "FLAG_DOCUMENT_11"
## [107] "FLAG_DOCUMENT_12"             "FLAG_DOCUMENT_13"
## [109] "FLAG_DOCUMENT_14"             "FLAG_DOCUMENT_15"
## [111] "FLAG_DOCUMENT_16"             "FLAG_DOCUMENT_17"
## [113] "FLAG_DOCUMENT_18"             "FLAG_DOCUMENT_19"
## [115] "FLAG_DOCUMENT_20"             "FLAG_DOCUMENT_21"
## [117] "AMT_REQ_CREDIT_BUREAU_HOUR"   "AMT_REQ_CREDIT_BUREAU_DAY"
## [119] "AMT_REQ_CREDIT_BUREAU_WEEK"   "AMT_REQ_CREDIT_BUREAU_MON"
## [121] "AMT_REQ_CREDIT_BUREAU_QRT"    "AMT_REQ_CREDIT_BUREAU_YEAR"
## [123] "n_loans"                      "Total_credit"
## [125] "na.rm"                        "Total_debt"
## [127] "Max_overdue"                  "Mean_overdue"
## [129] "mean_balance"                 "Max_balance"
## [131] "Mean_utilization"             "Max_dpd"
## [133] "Mean_dpd"
```

```r
y_hat_RF2<-predict(RF_MODEL_2, test_set, type = "response")

CFM_RF_2<-confusionMatrix(as.factor(y_hat_RF2),test_set$TARGET)

CFM_RF_2
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 57896  2401
##          1 26981  4976
##
##                Accuracy : 0.6815
##                  95% CI : (0.6785, 0.6845)
##     No Information Rate : 0.92
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1415
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.6821
##             Specificity : 0.6745
##          Pos Pred Value : 0.9602
##          Neg Pred Value : 0.1557
##              Prevalence : 0.9200
##          Detection Rate : 0.6276
##    Detection Prevalence : 0.6536
##       Balanced Accuracy : 0.6783
##
##        'Positive' Class : 0
##
```

```
varImpPlot(RF_MODEL_2)
```



```
training_balanced_set$debt_ratio <- training_balanced_set$Total_debt / training_balanced_set$Total_credi
training_balanced_set$annuity_ratio <- training_balanced_set$AMT_ANNUITY / training_balanced_set$AMT_CR
training_balanced_set$age_years <- -training_balanced_set$DAYS_BIRTH / 365
training_balanced_set$employment_years <- training_balanced_set$DAYS_EMPLOYED / 365

# 2 managing the NA
num_vars <- c("Total_credit", "Total_debt", "AMT_CREDIT", "AMT_ANNUITY",
              "debt_ratio", "annuity_ratio", "age_years", "employment_years")

training_balanced_set[num_vars] <- lapply(training_balanced_set[num_vars], function(x) {
  x[is.na(x) | !is.finite(x)] <- median(x, na.rm = TRUE)
  x
})

#  feature sul test set
test_set$debt_ratio <- test_set$Total_debt / test_set$Total_credit
test_set$annuity_ratio <- test_set$AMT_ANNUITY / test_set$AMT_CREDIT
test_set$age_years <- -test_set$DAYS_BIRTH / 365
test_set$employment_years <- test_set$DAYS_EMPLOYED / 365

# new features
num_vars_test <- c("Total_credit", "Total_debt", "AMT_CREDIT", "AMT_ANNUITY",
                   "debt_ratio", "annuity_ratio", "age_years", "employment_years")

test_set[num_vars_test] <- lapply(test_set[num_vars_test], function(x) {
  x[is.na(x) | !is.finite(x)] <- median(x, na.rm = TRUE)  # mediana calcolata sul test set
  x
})


if("TARGET" %in% colnames(test_set)){
  test_set$TARGET <- as.factor(test_set$TARGET)
}
```

```
RF_MODEL_3<- randomForest(TARGET  ~ EXT_SOURCE_1 + EXT_SOURCE_2 + EXT_SOURCE_3 +
    AMT_INCOME_TOTAL + AMT_CREDIT + AMT_ANNUITY + Total_credit + Total_debt + Mean_dpd + Mean_overdue +
    DAYS_BIRTH +NAME_CONTRACT_TYPE + CODE_GENDER + NAME_EDUCATION_TYPE + NAME_FAMILY_STATUS +
    FLAG_OWN_CAR + FLAG_OWN_REALTY + CNT_CHILDREN +debt_ratio+annuity_ratio+age_years+employment_years+
    REGION_POPULATION_RELATIVE,
  data = training_balanced_set,
  ntree= 500,
  mtry= 3,
  )
```

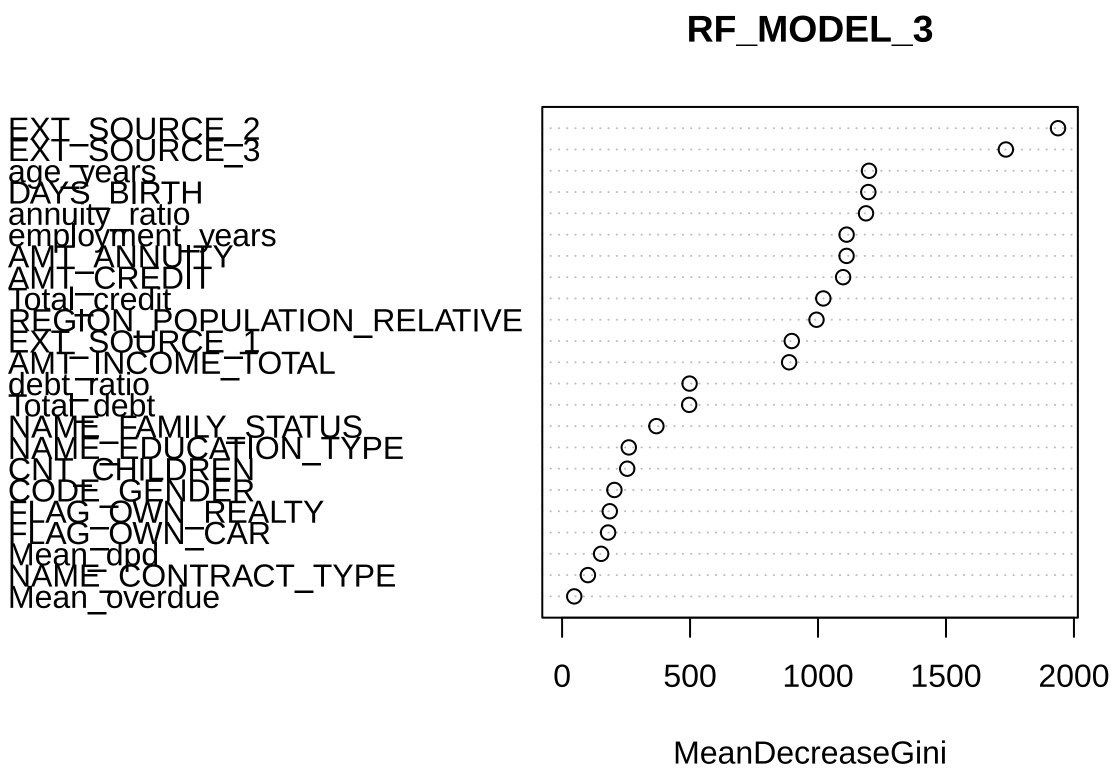
```
y_hat_RF3<-predict(RF_MODEL_3,test_set, type="response")

CFM_RF_3<-confusionMatrix(as.factor(y_hat_RF3),test_set$TARGET)
```

```
CFM_RF_3
```

```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction     0     1
##          0 58357  2430
##          1 26520  4947
##
##                Accuracy : 0.6862
##                  95% CI : (0.6832, 0.6892)
##     No Information Rate : 0.92
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1438
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.6875
##             Specificity : 0.6706
##          Pos Pred Value : 0.9600
##          Neg Pred Value : 0.1572
##              Prevalence : 0.9200
##          Detection Rate : 0.6326
##    Detection Prevalence : 0.6589
##       Balanced Accuracy : 0.6791
##
##        'Positive' Class : 0
##
```

```r
varImpPlot(RF_MODEL_3)
```



## XGBoost

we try to use this algoritmic

```
library(Matrix)
```

```
##
## Caricamento pacchetto: 'Matrix'

## I seguenti oggetti sono mascherati da 'package:tidyr':
##
##     expand, pack, unpack
```

```
feature_vars <- c("EXT_SOURCE_1", "EXT_SOURCE_2", "EXT_SOURCE_3",
                  "AMT_INCOME_TOTAL", "AMT_CREDIT", "AMT_ANNUITY",
                  "Total_credit", "Total_debt", "Mean_dpd", "Mean_overdue",
                  "DAYS_BIRTH", "NAME_CONTRACT_TYPE", "CODE_GENDER",
                  "NAME_EDUCATION_TYPE", "NAME_FAMILY_STATUS",
                  "FLAG_OWN_CAR", "FLAG_OWN_REALTY", "CNT_CHILDREN",
                  "debt_ratio", "annuity_ratio", "age_years", "employment_years",
                  "REGION_POPULATION_RELATIVE")
train_matrix <- sparse.model.matrix(
  TARGET ~ .,
  data = training_balanced_set[, c(feature_vars, "TARGET")]
)
```

```r
train_label <- as.numeric(training_balanced_set$TARGET) - 1  # 0/1

# Se hai il test set:
test_matrix <- sparse.model.matrix(
  ~ .,
  data = test_set[, feature_vars]
)
if("TARGET" %in% colnames(test_set)){
  test_label <- as.numeric(test_set$TARGET) - 1
}

dtrain <- xgb.DMatrix(data = train_matrix, label = train_label)
if(exists("test_label")){
  dtest <- xgb.DMatrix(data = test_matrix, label = test_label)
}else{
  dtest <- xgb.DMatrix(data = test_matrix)
}

params <- list(
  booster = "gbtree",
  objective = "binary:logistic",
  eval_metric = "auc",
  eta = 0.05,
  max_depth = 6,
  min_child_weight = 1,
  subsample = 0.8,
  colsample_bytree = 0.8
)

set.seed(123)
xgb_model <- xgb.train(
  params = params,
  data = dtrain,
  nrounds = 1000,
  watchlist = list(train = dtrain),
  early_stopping_rounds = 50,
  print_every_n = 20
)
```

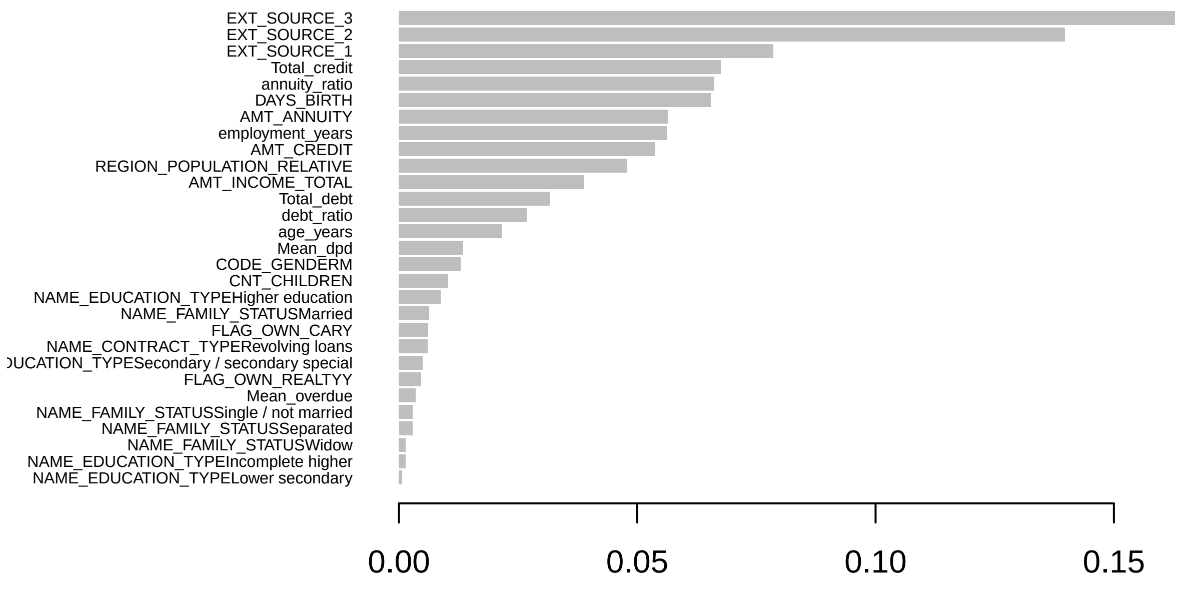
```
## Warning in throw_err_or_depr_msg("Parameter '", match_old, "' has been renamed
## to '", : Parameter 'watchlist' has been renamed to 'evals'. This warning will
## become an error in a future version.
```

```
## Will train until train_auc hasn't improved in 50 rounds.
##
## [1]  train-auc:0.726311
## [21] train-auc:0.763014
## [41] train-auc:0.775299
## [61] train-auc:0.784773
## [81] train-auc:0.793473
## [101]    train-auc:0.801372
## [121]    train-auc:0.807348
## [141]    train-auc:0.813013
## [161]    train-auc:0.818604
## [181]    train-auc:0.823739
## [201]    train-auc:0.828828
## [221]    train-auc:0.833322
## [241]    train-auc:0.838015
## [261]    train-auc:0.842629
## [281]    train-auc:0.847188
## [301]    train-auc:0.851657
## [321]    train-auc:0.856358
## [341]    train-auc:0.860482
## [361]    train-auc:0.864479
## [381]    train-auc:0.867925
## [401]    train-auc:0.872006
## [421]    train-auc:0.876138
## [441]    train-auc:0.879024
## [461]    train-auc:0.882773
## [481]    train-auc:0.885984
## [501]    train-auc:0.889300
## [521]    train-auc:0.892566
## [541]    train-auc:0.895687
## [561]    train-auc:0.898539
## [581]    train-auc:0.901796
## [601]    train-auc:0.905150
## [621]    train-auc:0.908045
## [641]    train-auc:0.910953
## [661]    train-auc:0.913510
## [681]    train-auc:0.916387
## [701]    train-auc:0.918780
## [721]    train-auc:0.921127
## [741]    train-auc:0.923857
## [761]    train-auc:0.926263
## [781]    train-auc:0.928544
## [801]    train-auc:0.930651
## [821]    train-auc:0.932900
## [841]    train-auc:0.935055
## [861]    train-auc:0.937257
## [881]    train-auc:0.939342
## [901]    train-auc:0.941239
## [921]    train-auc:0.943196
## [941]    train-auc:0.944745
## [961]    train-auc:0.946607
## [981]    train-auc:0.948218
## [1000]   train-auc:0.949982
```

```r
pred_prob <- predict(xgb_model, dtest)

if(exists("test_label")){
  pred_class <- ifelse(pred_prob > 0.5, 1, 0)
  accuracy <- mean(pred_class == test_label)
  print(paste("Accuracy:", round(accuracy, 4)))
}
```

```
## [1] "Accuracy: 0.6833"
```

```r
importance <- xgb.importance(model = xgb_model)
xgb.plot.importance(importance)
```



```r
pred_class <- ifelse(pred_prob > 0.5, 1, 0)
pred_class <- factor(pred_class, levels = c(0,1))
test_label_factor <- factor(test_label, levels = c(0,1))


conf_matrix <- confusionMatrix(pred_class, test_label_factor, positive = "1")
print(conf_matrix)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 58086  2423
##          1 26791  4954
##
##                Accuracy : 0.6833
##                  95% CI : (0.6803, 0.6863)
##     No Information Rate : 0.92
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1419
##
```

```
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.67155
##             Specificity : 0.68436
##          Pos Pred Value : 0.15606
##          Neg Pred Value : 0.95996
##              Prevalence : 0.07996
##          Detection Rate : 0.05370
##    Detection Prevalence : 0.34410
##       Balanced Accuracy : 0.67795
##
##        'Positive' Class : 1
##
```