Support Vector Machine con kernlab

Adolfo Sánchez Burón

Algoritmos empleados: Support Vector Machine (SVM)

SVM es un algoritmo de machine learning supervisado que puede emplearse tanto para casos de clasificación (**Support Vector Classifier**) como de predicción (**Support Vector Regressor**). En este post nos dedicaremos a tareas de clasificación binaria. Básicamente consiste en hallar el hiperplano (**hiperplano óptimo de separación**), de las infinitas posibilidades, que posibilite maximizar el límite las observaciones de dos clases.

Para hallar este hiperplano óptimo de separación se calcula la distancia de separación entre las observaciones y diferentes hiperplanos. El SVM da como resultado el hiperplano que reporta el mayor margen (**maximal margin hyperplane**) con respecto a las observaciones de entrenamiento (**vectores soporte**).

Funciones kernel

En este proyecto mostraremos la ejecución de los tres más populares:

- **Kernel lineal**: equivalente a un support vector classifier, segmentación mediante una linea recta.
- **Kernel radial** (RBF kernel, radial basis function kernel): cuyos límites se establecen de forma radial.
- **Kernel polinomial**: con límites más flexibles.

Amat Rodrigo,J. Máquinas de Vector Soporte (Support Vector Machines, SVMs) (https://www.cienciadedatos.net/documentos/34_maquinas_de_vector_soporte_support_vector_machines)

Boehmke,B. Support Vector Machine (http://uc-r.github.io/svm)

Karatzoglou,A., Smola,A. y Hornik,K. kernlab – An S4 Package for Kernel Methods in R (https://cran.r-project.org/web/packages/kernlab/vignettes/kernlab.pdf)

Meyer,D. Support Vector Machines. The Interface to libsvm in package e1071 (https://cran.r-project.org/web/packages/e1071/vignettes/svmdoc.pdf)

Características del caso

El caso empleado en este análisis es el 'German Credit Data', que puede descargarse el dataset original desde UCI (https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data)). Este dataset ha sido previamente trabajado en cuanto a:

- análisis descriptivo
- limpieza de anomalías, missing y outliers
- peso predictivo de las variables mediante random forest
- discretización de las variables continuas para facilitar la interpretación posterior

Por lo que finalmente se emplea en este caso un dataset preparado para iniciar el análisis, que puede descargarse de GitHub (https://github.com/AdSan-R/MachineLearning_R/tree/main/dataset).

El objetivo del caso es predecir la probabilidad de que un determinado cliente puede incluir un crédito bancario. La explicación de esta conducta estará basada en toda una serie de variables predictoras que se explicarán posteriormente.

Proceso

1. Entorno

El primer punto tratará sobre la preparación del entorno, donde se mostrará la descarga de las librerías empleadas y la importación de datos.

2. Análisis descriptivo

Se mostrarán y explicarán las funciones empleadas en este paso, dividiéndolas en tres grupos: Análisis inicial, Tipología de datos y Análisis descriptivo (gráficos).

3. Preparación de la modelización

Particiones del dataset en dos grupos: training (70%) y test (30%)

4. Modelización

Por motivos didácticos, se dividirá la modelización de los dos algoritmos en una sucesión de pasos.

---

1. Entorno

1.1. Instalar librerías

```
library(dplyr)
library(knitr)       # For Dynamic Report Generation in R
library(ROCR)        # Model Performance and ROC curve
library(caret)       # Classification and Regression Training -  for any machine learning
 algorithms
library(kernlab)     # Support Vector Machine
library(DataExplorer) #para realizar el análisis descriptivo con gráficos
```

```
options(scipen=999)
#Desactiva la notación científica
```

1.2. Importar datos

Como el dataset ha sido peviamente trabajado para poder modelizar directamente, si deseas seguir este tutorial, lo puedes descargar de GitHub (https://github.com/AdSan-R).

```
df <- read.csv("CreditBank")
```

---

2. Análisis descriptivo

2.1. Análisis inicial

```
head(df) #ver la estructura de los primeros 6 casos
```

```
##   X chk_ac_status_1 credit_history_3 duration_month_2 savings_ac_bond_6
## 1 1             A11           04.A34            00-06               A65
## 2 2             A12       03.A32.A33              42+               A61
## 3 3             A14           04.A34            06-12               A61
## 4 4             A11       03.A32.A33            36-42               A61
## 5 5             A11       03.A32.A33            12-24               A61
## 6 6             A14       03.A32.A33            30-36               A65
##   purpose_4 property_type_12 age_in_yrs_13 credit_amount_5 p_employment_since_7
## 1       A43             A121           60+          0-1400                  A75
## 2       A43             A121          0-25           5500+                  A73
## 3       A46             A121         45-50       1400-2500                  A74
## 4       A42             A122         40-45           5500+                  A74
## 5       A40             A124         50-60       4500-5500                  A73
## 6       A46             A124         30-35           5500+                  A73
##   housing_type_15 other_instalment_type_14 personal_status_9 foreign_worker_20
## 1            A152                    A143               A93              A201
## 2            A152                    A143               A92              A201
## 3            A152                    A143               A93              A201
## 4            A153                    A143               A93              A201
## 5            A153                    A143               A93              A201
## 6            A153                    A143               A93              A201
##   other_debtors_or_grantors_10 instalment_pct_8 good_bad_21
## 1                         A101               4        Good
## 2                         A101               2         Bad
## 3                         A101               2        Good
## 4                         A103               2        Good
## 5                         A101               3         Bad
## 6                         A101               2        Good
```

2.2. Tipología de datos

```
str(df) #mostrar la estructura del dataset y los tipos de variables
```

```
## 'data.frame':    1000 obs. of  17 variables:
##  $ X                          : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ chk_ac_status_1            : chr  "A11" "A12" "A14" "A11" ...
##  $ credit_history_3           : chr  "04.A34" "03.A32.A33" "04.A34" "03.A32.A33" ...
##  $ duration_month_2           : chr  "00-06" "42+" "06-12" "36-42" ...
##  $ savings_ac_bond_6          : chr  "A65" "A61" "A61" "A61" ...
##  $ purpose_4                  : chr  "A43" "A43" "A46" "A42" ...
##  $ property_type_12           : chr  "A121" "A121" "A121" "A122" ...
##  $ age_in_yrs_13              : chr  "60+" "0-25" "45-50" "40-45" ...
##  $ credit_amount_5            : chr  "0-1400" "5500+" "1400-2500" "5500+" ...
##  $ p_employment_since_7       : chr  "A75" "A73" "A74" "A74" ...
##  $ housing_type_15            : chr  "A152" "A152" "A152" "A153" ...
##  $ other_instalment_type_14   : chr  "A143" "A143" "A143" "A143" ...
##  $ personal_status_9          : chr  "A93" "A92" "A93" "A93" ...
##  $ foreign_worker_20          : chr  "A201" "A201" "A201" "A201" ...
##  $ other_debtors_or_grantors_10: chr  "A101" "A101" "A101" "A103" ...
##  $ instalment_pct_8           : int  4 2 2 2 3 2 3 2 2 4 ...
##  $ good_bad_21                : chr  "Good" "Bad" "Good" "Good" ...
```

Puede observarse que todas son “chr”, esto es, “character”, por tanto, vamos a pasarlas a Factor. Además, instalment_pct_8 aparece como “entero” cuando es factor. También la transformamos.

```
df <- mutate_if(df, is.character, as.factor) #identifica todas las character y las pasa a
 factores
#Sacamos la esructura

df$instalment_pct_8 <- as.factor(df$instalment_pct_8 )

str(df)
```

```
## 'data.frame':    1000 obs. of  17 variables:
##  $ X                           : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ chk_ac_status_1             : Factor w/ 4 levels "A11","A12","A13",..: 1 2 4 1 1 4 4
2 4 2 ...
##  $ credit_history_3            : Factor w/ 4 levels "01.A30","02.A31",..: 4 3 4 3 3 3 3
3 3 4 ...
##  $ duration_month_2            : Factor w/ 7 levels "00-06","06-12",..: 1 7 2 6 3 5 3 5
2 4 ...
##  $ savings_ac_bond_6           : Factor w/ 5 levels "A61","A62","A63",..: 5 1 1 1 1 5 3
1 4 1 ...
##  $ purpose_4                   : Factor w/ 10 levels "A40","A41","A410",..: 5 5 8 4 1 8
4 2 5 1 ...
##  $ property_type_12            : Factor w/ 4 levels "A121","A122",..: 1 1 1 2 4 4 2 3 1
3 ...
##  $ age_in_yrs_13               : Factor w/ 8 levels "0-25","25-30",..: 8 1 6 5 7 3 7 3
8 2 ...
##  $ credit_amount_5             : Factor w/ 6 levels "0-1400","1400-2500",..: 1 6 2 6 5
6 3 6 3 5 ...
##  $ p_employment_since_7        : Factor w/ 5 levels "A71","A72","A73",..: 5 3 4 4 3 3 5
3 4 1 ...
##  $ housing_type_15             : Factor w/ 3 levels "A151","A152",..: 2 2 2 3 3 3 2 1 2
2 ...
##  $ other_instalment_type_14    : Factor w/ 3 levels "A141","A142",..: 3 3 3 3 3 3 3 3 3
3 ...
##  $ personal_status_9           : Factor w/ 4 levels "A91","A92","A93",..: 3 2 3 3 3 3 3
3 1 4 ...
##  $ foreign_worker_20           : Factor w/ 2 levels "A201","A202": 1 1 1 1 1 1 1 1 1 1
...
##  $ other_debtors_or_grantors_10: Factor w/ 3 levels "A101","A102",..: 1 1 1 3 1 1 1 1 1
1 ...
##  $ instalment_pct_8            : Factor w/ 4 levels "1","2","3","4": 4 2 2 2 3 2 3 2 2
4 ...
##  $ good_bad_21                 : Factor w/ 2 levels "Bad","Good": 2 1 2 2 1 2 2 2 2 1
...
```

Ahora se puede observar que todas las variables son de tipo “Factor”

Para los siguientes análisis: 1º) Eliminamos a la variable X (número de cliente) del df. 2º) Renombraos la la variable good_bad_21 como “target”

```
#Eliminamos x
df <- select(df,-X)

#Creamos la variable "target"
df$target <- as.factor(df$good_bad_21)

#Eliminamos la variable "good_bad_21"
df <- select(df,-good_bad_21)

str(df)
```

```
## 'data.frame':    1000 obs. of  16 variables:
##  $ chk_ac_status_1             : Factor w/ 4 levels "A11","A12","A13",..: 1 2 4 1 1 4 4
2 4 2 ...
##  $ credit_history_3            : Factor w/ 4 levels "01.A30","02.A31",..: 4 3 4 3 3 3 3
3 3 4 ...
##  $ duration_month_2            : Factor w/ 7 levels "00-06","06-12",..: 1 7 2 6 3 5 3 5
2 4 ...
##  $ savings_ac_bond_6           : Factor w/ 5 levels "A61","A62","A63",..: 5 1 1 1 1 5 3
1 4 1 ...
##  $ purpose_4                   : Factor w/ 10 levels "A40","A41","A410",..: 5 5 8 4 1 8
4 2 5 1 ...
##  $ property_type_12            : Factor w/ 4 levels "A121","A122",..: 1 1 1 2 4 4 2 3 1
3 ...
##  $ age_in_yrs_13               : Factor w/ 8 levels "0-25","25-30",..: 8 1 6 5 7 3 7 3
8 2 ...
##  $ credit_amount_5             : Factor w/ 6 levels "0-1400","1400-2500",..: 1 6 2 6 5
6 3 6 3 5 ...
##  $ p_employment_since_7        : Factor w/ 5 levels "A71","A72","A73",..: 5 3 4 4 3 3 5
3 4 1 ...
##  $ housing_type_15             : Factor w/ 3 levels "A151","A152",..: 2 2 2 3 3 3 2 1 2
2 ...
##  $ other_instalment_type_14    : Factor w/ 3 levels "A141","A142",..: 3 3 3 3 3 3 3 3 3
3 ...
##  $ personal_status_9           : Factor w/ 4 levels "A91","A92","A93",..: 3 2 3 3 3 3 3
3 1 4 ...
##  $ foreign_worker_20           : Factor w/ 2 levels "A201","A202": 1 1 1 1 1 1 1 1 1 1
...
##  $ other_debtors_or_grantors_10: Factor w/ 3 levels "A101","A102",..: 1 1 1 3 1 1 1 1 1
1 ...
##  $ instalment_pct_8            : Factor w/ 4 levels "1","2","3","4": 4 2 2 2 3 2 3 2 2
4 ...
##  $ target                      : Factor w/ 2 levels "Bad","Good": 2 1 2 2 1 2 2 2 2 1
...
```

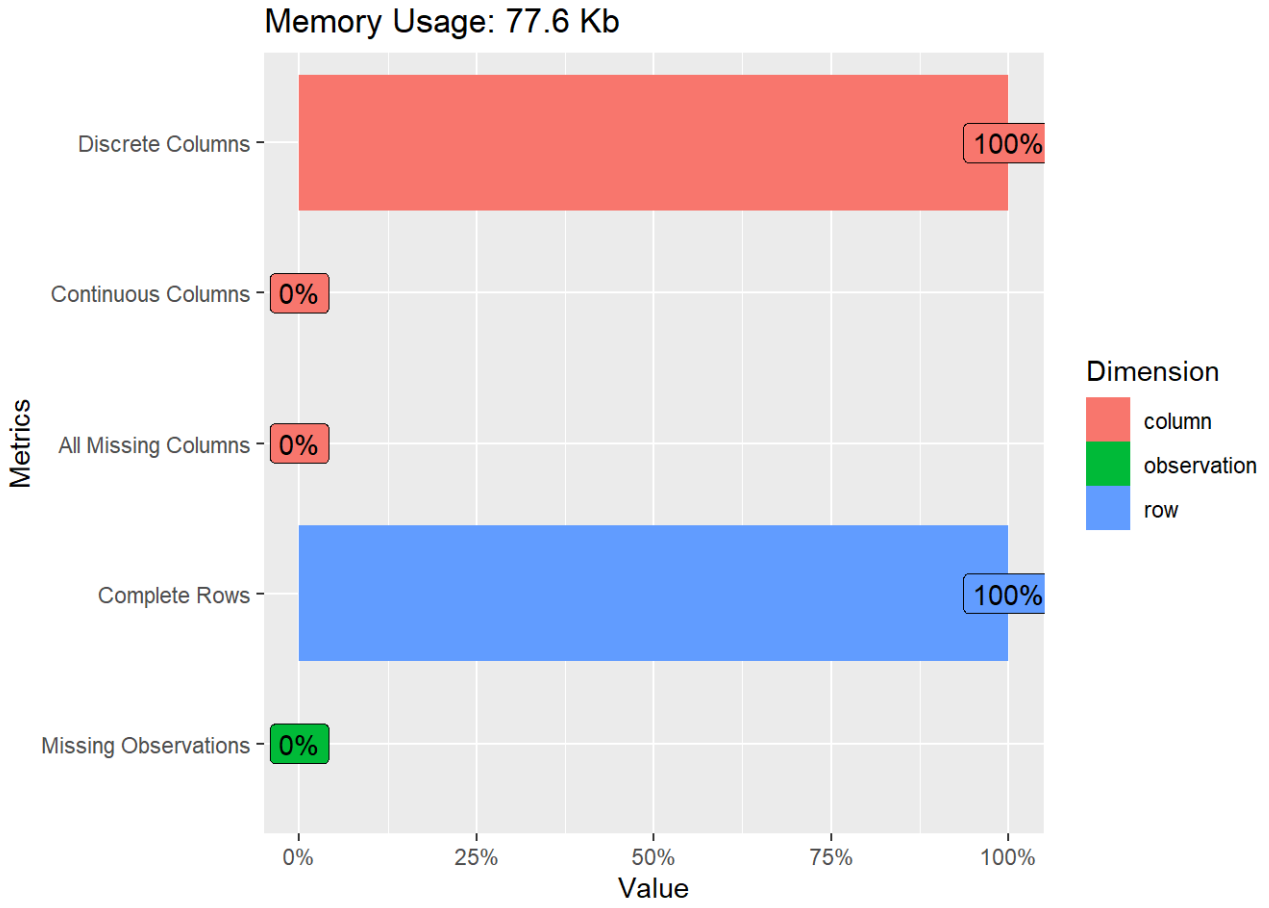
```
lapply(df,summary) #mostrar la distribución de frecuencias en cada categoría de todas las
 variables
```

```
## $chk_ac_status_1
## A11 A12 A13 A14
## 274 269  63 394
##
## $credit_history_3
##     01.A30     02.A31 03.A32.A33     04.A34
##         40         49        618        293
##
## $duration_month_2
## 00-06 06-12 12-24 24-30 30-36 36-42   42+
##    82   277   411    57    86    17    70
##
## $savings_ac_bond_6
## A61 A62 A63 A64 A65
## 603 103  63  48 183
##
## $purpose_4
##  A40  A41 A410  A42  A43  A44  A45  A46  A48  A49
##  234  103   12  181  280   12   22   50    9   97
##
## $property_type_12
## A121 A122 A123 A124
##  282  232  332  154
##
## $age_in_yrs_13
##  0-25 25-30 30-35 35-40 40-45 45-50 50-60   60+
##   190   221   177   138    88    73    68    45
##
## $credit_amount_5
##    0-1400 1400-2500 2500-3500 3500-4500 4500-5500     5500+
##       267       270       149        98        48       168
##
## $p_employment_since_7
## A71 A72 A73 A74 A75
##  62 172 339 174 253
##
## $housing_type_15
## A151 A152 A153
##  179  713  108
##
## $other_instalment_type_14
## A141 A142 A143
##  139   47  814
##
## $personal_status_9
## A91 A92 A93 A94
##  50 310 548  92
##
## $foreign_worker_20
## A201 A202
##  963   37
##
## $other_debtors_or_grantors_10
```

```
## A101 A102 A103
##  907   41   52
##
## $instalment_pct_8
##   1   2   3   4
## 136 231 157 476
##
## $target
##  Bad Good
##  300  700
```

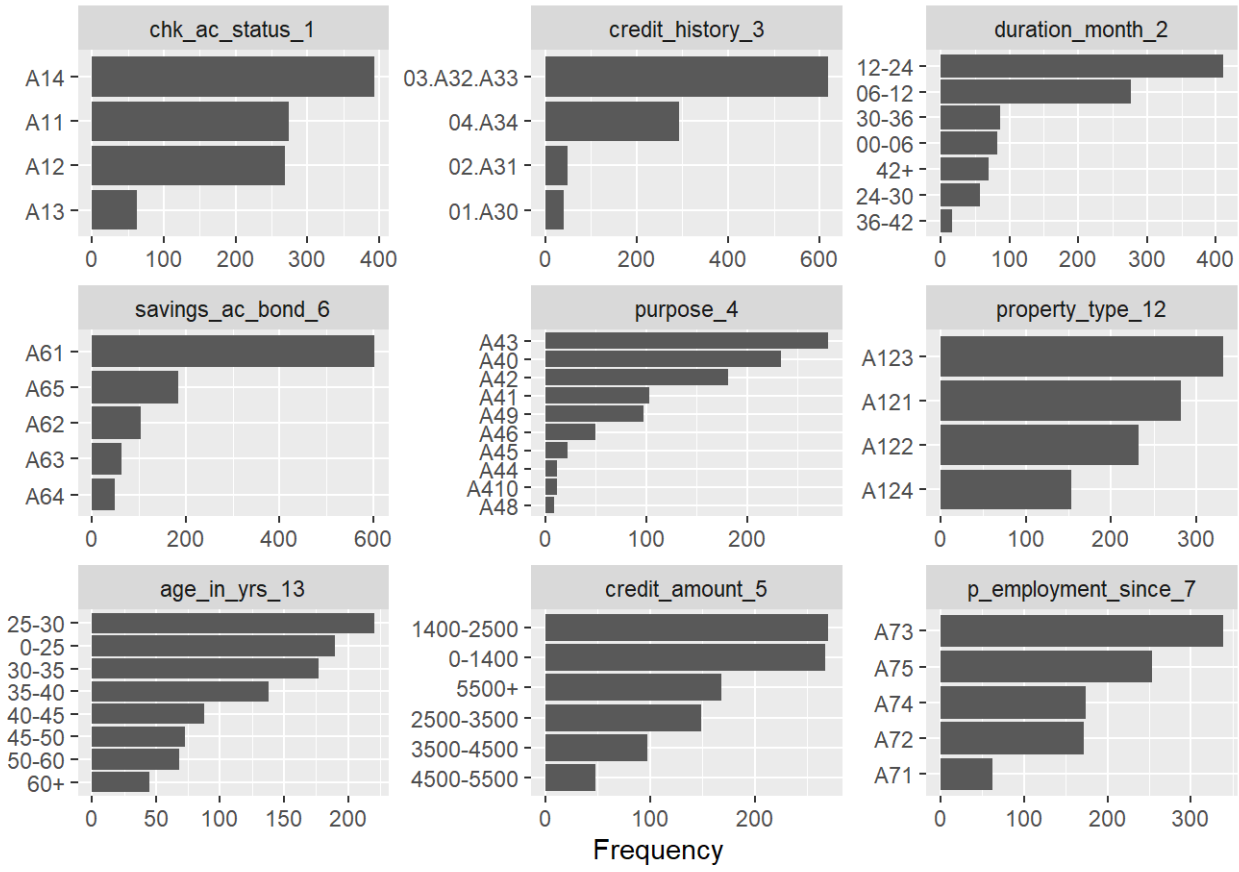
2.3. Análisis descriptivo (gráficos)

```
plot_intro(df) #gráfico para observar la distribución de variables y los casos missing por columnas, observaciones y filas
```

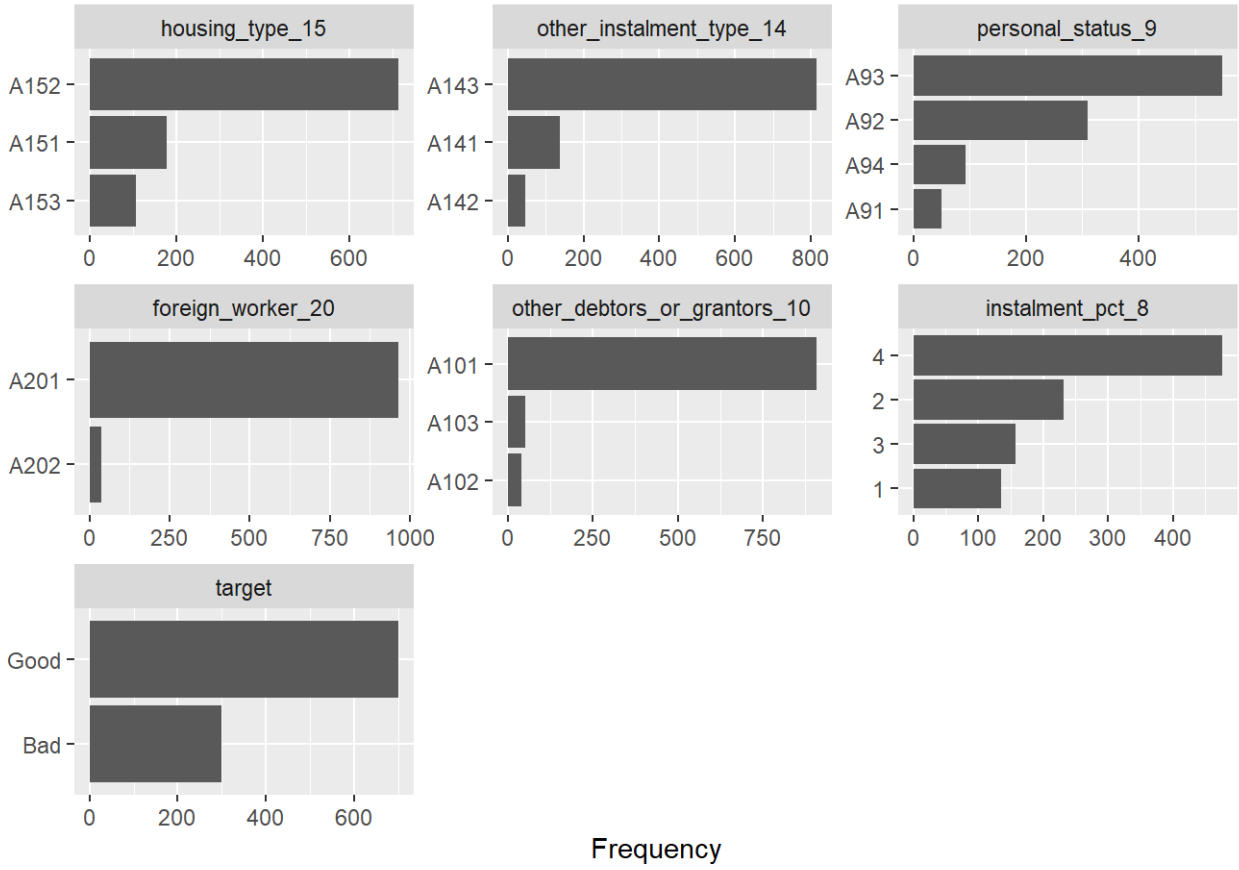


Como se ha trabajado previamente, no existen casos missing, por lo que podemos seguir el análisis descriptivo.

```
plot_bar(df) #gráfico para observar la distribución de frecuencias en variables categóricas
```



Page 1



Page 2

3. Modelización

3.1. Preparar funciones

Tomadas del curso de Machine Learning Predictivo (https://www.datascience4business.com/08_mlc-salespage-b) de DS4B) :

- Matriz de confusión
- Métricas
- Umbrales

Función para la matriz de confusión

En esta función se prepara la matriz de confusión (ver en otro post), donde se observa qué casos coinciden entre la puntuación real (obtenida por cada sujeto) y la puntuación predicha ("scoring") por el modelo, estableciendo previmente un límite ("umbral") para ello.

```
confusion<-function(real,scoring,umbral){
  conf<-table(real,scoring>=umbral)
  if(ncol(conf)==2) return(conf) else return(NULL)
}
```

Funcion para métricas de los modelos

Los indicadores a observar serán:

- Acierto (accuracy) = (TRUE POSITIVE + TRUE NEGATIVE) / TODA LA POBLACIÓN
- Precisión = TRUE POSITIVE / (TRUE POSITIVE + FALSE POSITIVE)
- Cobertura (recall, sensitivity) = TRUE POSITIVE / (TRUE POSITIVE + FALSE NEGATIVE)
- F1 = 2* (precisión * cobertura) (precisión + cobertura)

```
metricas<-function(matriz_conf){
  acierto <- (matriz_conf[1,1] + matriz_conf[2,2]) / sum(matriz_conf) *100
  precision <- matriz_conf[2,2] / (matriz_conf[2,2] + matriz_conf[1,2]) *100
  cobertura <- matriz_conf[2,2] / (matriz_conf[2,2] + matriz_conf[2,1]) *100
  F1 <- 2*precision*cobertura/(precision+cobertura)
  salida<-c(acierto,precision,cobertura,F1)
  return(salida)
}
```

Función para probar distintos umbrales

Con esta función se analiza el efecto que tienen distintos umbrales sobre los indicadores de la matriz de confusión (precisión y cobertura). Lo que buscaremnos será aquél que maximice la relación entre cobertura y precisión (F1).

```
umbrales<-function(real,scoring){
  umbrales<-data.frame(umbral=rep(0,times=19),acierto=rep(0,times=19),precision=rep(0,time
s=19),cobertura=rep(0,times=19),F1=rep(0,times=19))
  cont <- 1
  for (cada in seq(0.05,0.95,by = 0.05)){
    datos<-metricas(confusion(real,scoring,cada))
    registro<-c(cada,datos)
    umbrales[cont,]<-registro
    cont <- cont + 1
  }
  return(umbrales)
}
```

3.2. Particiones de training (70%) y test (30%)

Se segmenta la muestra en dos partes (train y test) empleando el programa Caret.

1. Training o entrenamiento (70% de la muestra): servirá para entrenar al modelo de clasificación.
2. Test (30%): servirá para validar el modelo. La característica fundamental es que esta muestra no debe haber tenido contacto previamente con el funcionamiento del modelo.

```
set.seed(100)  # Para reproducir los mismos resultados
partition <- createDataPartition(y = df$target, p = 0.7, list = FALSE)
train <- df[partition,]
test <- df[-partition,]
```

```
#Distribución de la variable TARGET
table(train$target)
```

```
##
##  Bad Good
##  210  490
```

```
table(test$target)
```

```
##
##  Bad Good
##   90  210
```

4. Modelización con Support Vector Machine

A continuación se realizará una demostración de los tres kernel más empleados:

- Linear kernel function (kernel = vanilladot)
- Radial Basis Function (RBF) Kernel (“Gaussian”) (kernel = rbfdot)
- Polynomial kernel function (kernel = polydot)

4.1. Linear kernel function

Paso 1. Entrenamiento del modelo

```
# Entrenamos el modelo de SVM lineal
svm_model_l <- ksvm(target ~ .,
             data = train,
             kernel = "vanilladot",
             type = "C-svc",
             prob.model = TRUE)
```

```
##  Setting default kernel parameters
```

```
svm_model_l
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 1
##
## Linear (vanilla) kernel function.
##
## Number of Support Vectors : 366
##
## Objective Function Value : -331.1183
## Training error : 0.195714
## Probability model included.
```

Paso 2. Predict y matriz de confusión

```
svm_score_l_Response <- predict(svm_model_l, test, type="response")

MC_svm_l <- confusionMatrix(svm_score_l_Response, test$target , positive = 'Good')
MC_svm_l
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Bad Good
##       Bad   44   26
##       Good  46  184
##
##                Accuracy : 0.76
##                  95% CI : (0.7076, 0.8072)
##     No Information Rate : 0.7
##     P-Value [Acc > NIR] : 0.01249
##
##                   Kappa : 0.3898
##
##  Mcnemar's Test P-Value : 0.02514
##
##             Sensitivity : 0.8762
##             Specificity : 0.4889
##          Pos Pred Value : 0.8000
##          Neg Pred Value : 0.6286
##              Prevalence : 0.7000
##          Detection Rate : 0.6133
##    Detection Prevalence : 0.7667
##       Balanced Accuracy : 0.6825
##
##        'Positive' Class : Good
##
```

En este paso se puede observar la matriz de confusión y los indicadores fundamentales.

Paso 3. Predict con probabilidades y umbrales

1º) En este paso sacamos la probabilidad de cada cliente de devolver el crédito.

```
# Compute at the prediction scores
svm_score_l <- predict(svm_model_l,test, type="probabilities")[,2]

head(svm_score_l)
```

```
## [1] 0.9473509 0.9041024 0.9670356 0.6823252 0.4870095 0.7629345
```

2º) Ahora transformamos la probabilidad obtenida en una decisión binaria de si conceder el crédito (Sí lo va a devolver) o no (No lo va a devolver).

Con la función umbrales probamos diferentes cortes

```
umb_svm_l<-umbrales(test$target,svm_score_l)
umb_svm_l
```

```
##    umbral  acierto precision cobertura        F1
## 1    0.05  0.05000   0.05000  0.050000  0.050000
## 2    0.10 69.66667  69.89967 99.523810 82.121807
## 3    0.15 69.33333  69.79866 99.047619 81.889764
## 4    0.20 69.66667  70.03367 99.047619 82.051282
## 5    0.25 70.33333  70.64846 98.571429 82.306163
## 6    0.30 70.33333  71.08014 97.142857 82.092555
## 7    0.35 72.66667  73.02158 96.666667 83.196721
## 8    0.40 73.66667  74.34944 95.238095 83.507307
## 9    0.45 75.00000  76.06178 93.809524 84.008529
## 10   0.50 76.00000  78.51240 90.476190 84.070796
## 11   0.55 76.33333  79.82833 88.571429 83.972912
## 12   0.60 76.00000  82.54717 83.333333 82.938389
## 13   0.65 73.66667  83.58974 77.619048 80.493827
## 14   0.70 71.00000  84.74576 71.428571 77.519380
## 15   0.75 66.66667  85.71429 62.857143 72.527473
## 16   0.80 59.66667  88.03419 49.047619 62.996942
## 17   0.85 53.33333  93.75000 35.714286 51.724138
## 18   0.90 42.33333  97.43590 18.095238 30.522088
## 19   0.95 33.00000 100.00000  4.285714  8.219178
```

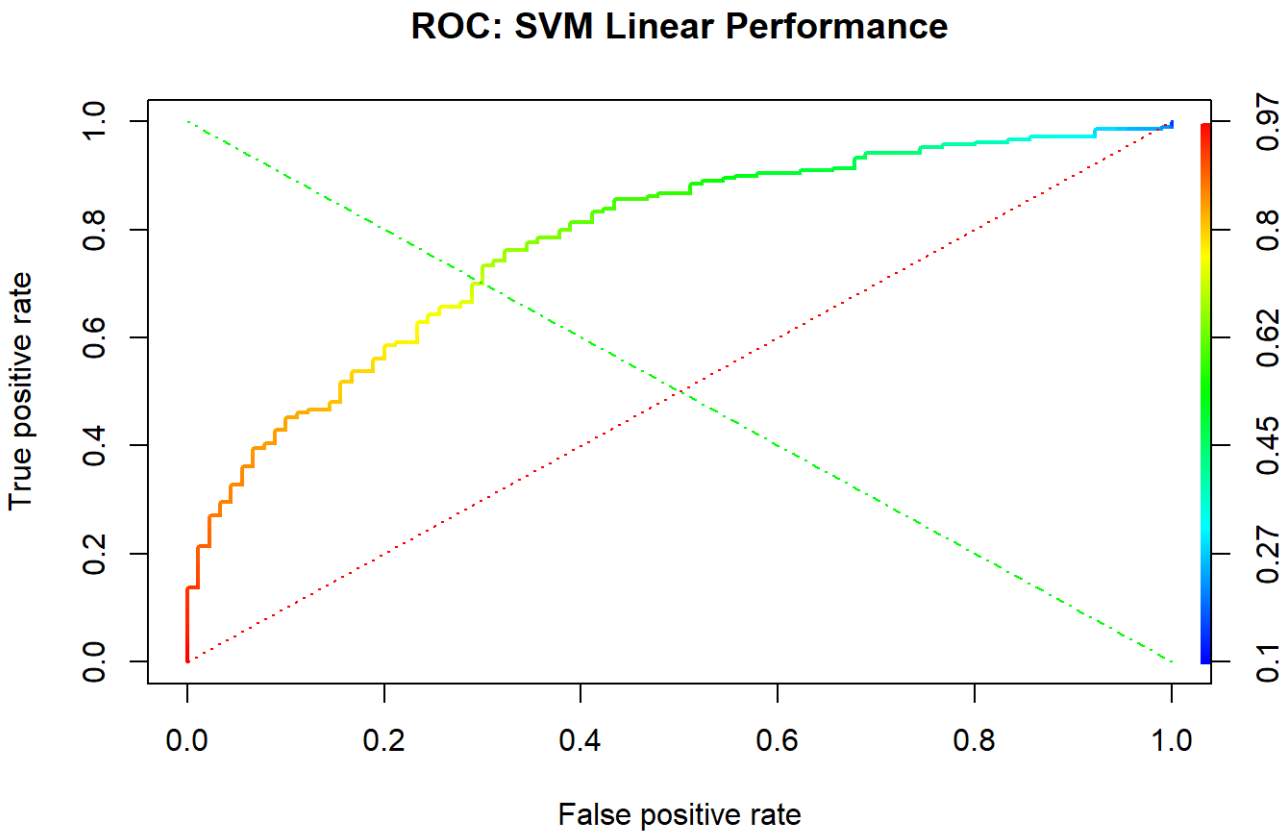
Seleccionamos el umbral que maximiza la F1 (cuando empieza a decaer)

```
umbfinal_svm_l<-umb_svm_l[which.max(umb_svm_l$F1),1]
umbfinal_svm_l
```

```
## [1] 0.5
```

Paso 4. Curva ROC

```
pred_svm_l <- prediction(svm_score_l, test$target)
perf_svm_l <- performance(pred_svm_l,"tpr","fpr")
#library(ROCR)
plot(perf_svm_l, lwd=2, colorize=TRUE, main="ROC: SVM Linear Performance")
lines(x=c(0, 1), y=c(0, 1), col="red", lwd=1, lty=3);
lines(x=c(1, 0), y=c(0, 1), col="green", lwd=1, lty=4)
```



Paso 5. Métricas definitivas

```
#Matriz de confusión con umbral final
score <- ifelse(svm_score_l > umbfinal_svm_l, "Good", "Bad")
MC <- table(test$target, score)
Acc_svm_l <- round((MC[1,1] + MC[2,2]) / sum(MC) *100, 2)
Sen_svm_l <- round(MC[2,2] / (MC[2,2] + MC[1,2]) *100, 2)
Pr_svm_l <- round(MC[2,2] / (MC[2,2] + MC[2,1]) *100, 2)
F1_svm_l <- round(2*Pr_svm_l*Sen_svm_l/(Pr_svm_l+Sen_svm_l), 2)

#AUC
AUROC_svm_l <- round(performance(pred_svm_l, measure = "auc")@y.values[[1]]*100, 2)

#Métricas finales del modelo
cat("Acc_svm_l: ", Acc_svm_l,"\tSen_svm_l: ", Sen_svm_l, "\tPr_svm_l:", Pr_svm_l, "\tF1_sv
m_l:", F1_svm_l, "\tAUROC_svm_l: ", AUROC_svm_l)
```

```
## Acc_svm_l:  76   Sen_svm_l:  78.51   Pr_svm_l: 90.48     F1_svm_l: 84.07     AUROC_svm_
l:  77.82
```

Se obtiene una AUC de 77.88, lo que indica un modelo moderadamente aceptable.

4.2. Radial Basis Function (RBF) Kernel ("Gaussian")

Paso 1. Entrenamiento del modelo

```
#Entrenamos el modelo RBF
svm_model_r <- ksvm(target ~ .,
                    data = train,
                    kernel = "rbfdot",
                    prob.model = TRUE)
svm_model_r
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 1
##
## Gaussian Radial Basis kernel function.
##  Hyperparameter : sigma =  0.0777777777777778
##
## Number of Support Vectors : 448
##
## Objective Function Value : -327.7512
## Training error : 0.151429
## Probability model included.
```

Paso 2. Predict y matriz de confusión

```
svm_score_r_Response <- predict(svm_model_r, test, type="response")

MC_svm_r <- confusionMatrix(svm_score_r_Response, test$target , positive = 'Good')
MC_svm_r
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Bad Good
##       Bad   24   18
##       Good  66  192
##
##                Accuracy : 0.72
##                  95% CI : (0.6655, 0.7701)
##     No Information Rate : 0.7
##     P-Value [Acc > NIR] : 0.2456
##
##                   Kappa : 0.2135
##
##  Mcnemar's Test P-Value : 0.0000002926
##
##             Sensitivity : 0.9143
##             Specificity : 0.2667
##          Pos Pred Value : 0.7442
##          Neg Pred Value : 0.5714
##              Prevalence : 0.7000
##          Detection Rate : 0.6400
##    Detection Prevalence : 0.8600
##       Balanced Accuracy : 0.5905
##
##        'Positive' Class : Good
##
```

En este paso se puede observar la matriz de confusión y los indicadores fundamentales.

Paso 3. Predict con probabilidades y umbrales

1º) En este paso sacamos la probabilidad de cada cliente de devolver el crédito.

```
# Compute at the prediction scores
svm_score_r <- predict(svm_model_r,test, type="probabilities")[,2]

head(svm_score_r)
```

```
## [1] 0.7676239 0.8935069 0.9637798 0.7787240 0.6055292 0.7963661
```

2º) Ahora transformamos la probabilidad obtenida en una decisión binaria de si conceder el crédito (Sí lo va a devolver) o no (No lo va a devolver).

Con la función umbrales probamos diferentes cortes.

```
umb_svm_r <- umbrales(test$target,svm_score_r)
umb_svm_r
```

```
##    umbral  acierto precision  cobertura        F1
## 1    0.05  0.05000   0.05000   0.050000  0.050000
## 2    0.10 70.66667  70.46980 100.000000 82.677165
## 3    0.15 71.00000  70.98976  99.047619 82.703777
## 4    0.20 70.00000  71.12676  96.190476 81.781377
## 5    0.25 70.33333  71.37809  96.190476 81.947262
## 6    0.30 72.00000  73.16176  94.761905 82.572614
## 7    0.35 71.66667  73.40824  93.333333 82.180294
## 8    0.40 71.66667  73.94636  91.904762 81.953291
## 9    0.45 72.00000  75.20000  89.523810 81.739130
## 10   0.50 72.00000  76.25000  87.142857 81.333333
## 11   0.55 73.66667  78.60262  85.714286 82.004556
## 12   0.60 73.66667  79.63801  83.809524 81.670534
## 13   0.65 72.33333  80.97561  79.047619 80.000000
## 14   0.70 71.66667  82.38342  75.714286 78.908189
## 15   0.75 68.66667  83.72093  68.571429 75.392670
## 16   0.80 61.00000  84.96241  53.809524 65.889213
## 17   0.85 57.00000  90.09901  43.333333 58.520900
## 18   0.90 46.66667  96.29630  24.761905 39.393939
## 19   0.95 32.33333 100.00000   3.333333  6.451613
```

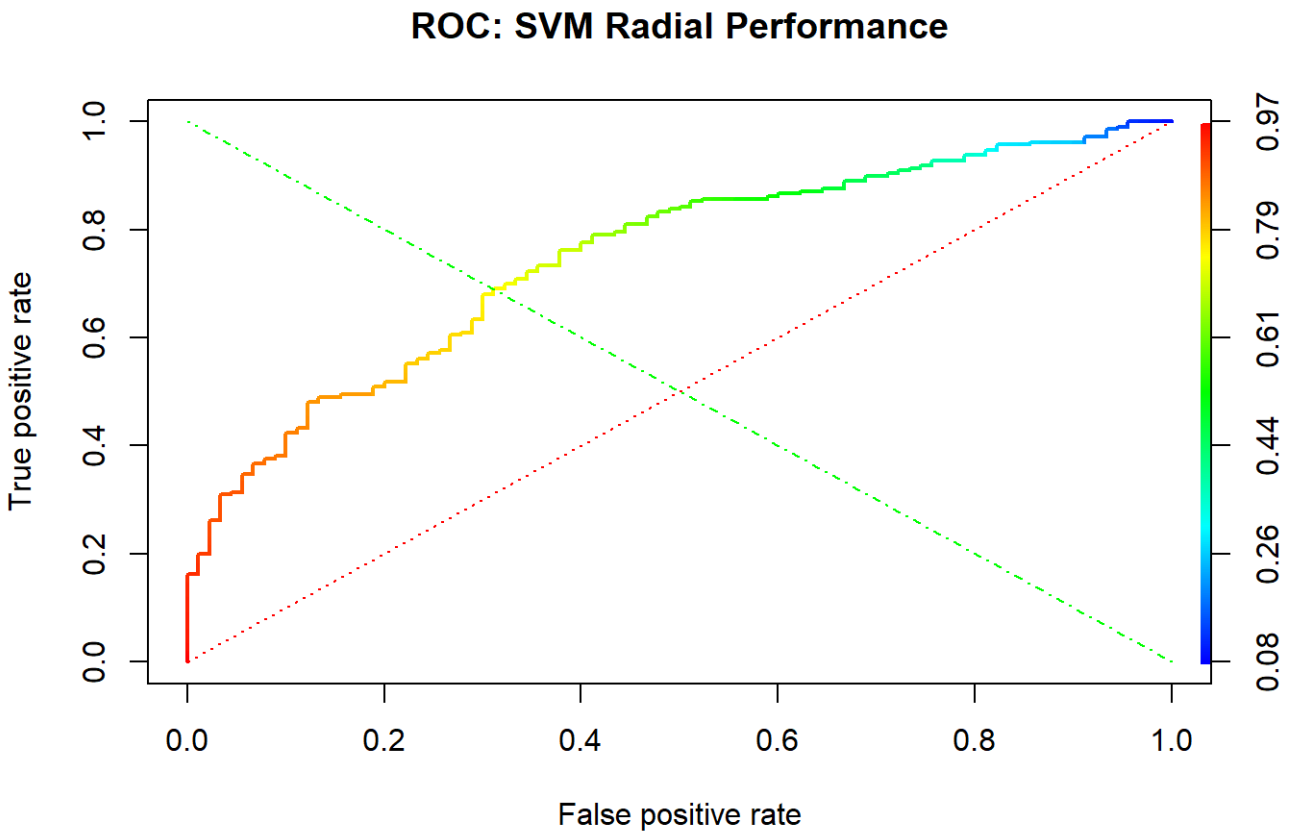
Seleccionamos el umbral que maximiza la F1 (cuando empieza a decaer)

```
umbfinal_svm_r <- umb_svm_r[which.max(umb_svm_r$F1),1]
umbfinal_svm_r
```

```
## [1] 0.15
```

Paso 4. Curva ROC

```
pred_svm_r <- prediction(svm_score_r, test$target)
perf_svm_r <- performance(pred_svm_r,"tpr","fpr")
#library(ROCR)
plot(perf_svm_r, lwd=2, colorize=TRUE, main="ROC: SVM Radial Performance")
lines(x=c(0, 1), y=c(0, 1), col="red", lwd=1, lty=3);
lines(x=c(1, 0), y=c(0, 1), col="green", lwd=1, lty=4)
```



Paso 5. Métricas definitivas

```
#Matriz de confusión con umbral final
score <- ifelse(svm_score_r > umbfinal_svm_r, "Good", "Bad")
MC <- table(test$target, score)
Acc_svm_r <- round((MC[1,1] + MC[2,2]) / sum(MC) *100, 2)
Sen_svm_r <- round(MC[2,2] / (MC[2,2] + MC[1,2]) *100, 2)
Pr_svm_r <- round(MC[2,2] / (MC[2,2] + MC[2,1]) *100, 2)
F1_svm_r <- round(2*Pr_svm_r*Sen_svm_r/(Pr_svm_r+Sen_svm_r), 2)

#AUC
AUROC_svm_r <- round(performance(pred_svm_r, measure = "auc")@y.values[[1]]*100, 2)

#Métricas finales del modelo
cat("Acc_svm_r: ", Acc_svm_r,"\tSen_svm_r: ", Sen_svm_r, "\tPr_svm_r:", Pr_svm_r, "\tF1_sv
m_r:", F1_svm_r, "\tAUROC_svm_r: ", AUROC_svm_r)
```

```
## Acc_svm_r:  71   Sen_svm_r:  70.99   Pr_svm_r: 99.05     F1_svm_r: 82.7  AUROC_svm_r:
74.79
```

Se obtiene una AUC de 74.79, lo que indica un modelo moderadamente aceptable.

4.3. Polynomial kernel function

Paso 1. Entrenamiento del modelo

```
svm_model_p <- ksvm(target ~ .,
                    data = train,
                    kernel = "polydot",
                    prob.model = TRUE)
```

```
##  Setting default kernel parameters
```

```
svm_model_p
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 1
##
## Polynomial kernel function.
##  Hyperparameters : degree =  1  scale =  1  offset =  1
##
## Number of Support Vectors : 366
##
## Objective Function Value : -331.1183
## Training error : 0.195714
## Probability model included.
```

Paso 2. Predict y matriz de confusión

```
svm_score_p_Response <- predict(svm_model_p, test, type="response")

MC_svm_p <- confusionMatrix(svm_score_p_Response, test$target , positive = 'Good')
MC_svm_p
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Bad Good
##       Bad   44   26
##       Good  46  184
##
##                Accuracy : 0.76
##                  95% CI : (0.7076, 0.8072)
##     No Information Rate : 0.7
##     P-Value [Acc > NIR] : 0.01249
##
##                   Kappa : 0.3898
##
##  Mcnemar's Test P-Value : 0.02514
##
##             Sensitivity : 0.8762
##             Specificity : 0.4889
##          Pos Pred Value : 0.8000
##          Neg Pred Value : 0.6286
##              Prevalence : 0.7000
##          Detection Rate : 0.6133
##    Detection Prevalence : 0.7667
##       Balanced Accuracy : 0.6825
##
##        'Positive' Class : Good
##
```

En este paso se puede observar la matriz de confusión y los indicadores fundamentales.

Paso 3. Predict con probabilidades y umbrales

1º) En este paso sacamos la probabilidad de cada cliente de devolver el crédito.

```
# Compute at the prediction scores
svm_score_p <- predict(svm_model_p,test, type="probabilities")[,2]

head(svm_score_p)
```

```
## [1] 0.9482704 0.9044489 0.9678749 0.6766536 0.4762462 0.7596934
```

2º) Ahora transformamos la probabilidad obtenida en una decisión binaria de si conceder el crédito (Sí lo va a devolver) o no (No lo va a devolver).

Con la función umbrales probamos diferentes cortes.

```
umb_svm_p <- umbrales(test$target,svm_score_p)
umb_svm_p
```

```
##    umbral  acierto precision cobertura        F1
## 1    0.05  0.05000   0.05000  0.050000  0.050000
## 2    0.10 69.66667  69.89967 99.523810 82.121807
## 3    0.15 69.33333  69.79866 99.047619 81.889764
## 4    0.20 69.66667  70.03367 99.047619 82.051282
## 5    0.25 70.66667  70.89041 98.571429 82.470120
## 6    0.30 71.00000  71.57895 97.142857 82.424242
## 7    0.35 72.33333  72.92419 96.190476 82.956879
## 8    0.40 74.33333  74.90637 95.238095 83.857442
## 9    0.45 74.66667  75.96899 93.333333 83.760684
## 10   0.50 75.66667  78.42324 90.000000 83.813747
## 11   0.55 76.33333  80.08658 88.095238 83.900227
## 12   0.60 76.00000  82.54717 83.333333 82.938389
## 13   0.65 73.00000  83.76963 76.190476 79.800499
## 14   0.70 70.33333  84.57143 70.476190 76.883117
## 15   0.75 66.66667  86.18421 62.380952 72.375691
## 16   0.80 59.33333  88.59649 48.095238 62.345679
## 17   0.85 53.00000  93.67089 35.238095 51.211073
## 18   0.90 42.33333  97.43590 18.095238 30.522088
## 19   0.95 33.00000 100.00000  4.285714  8.219178
```

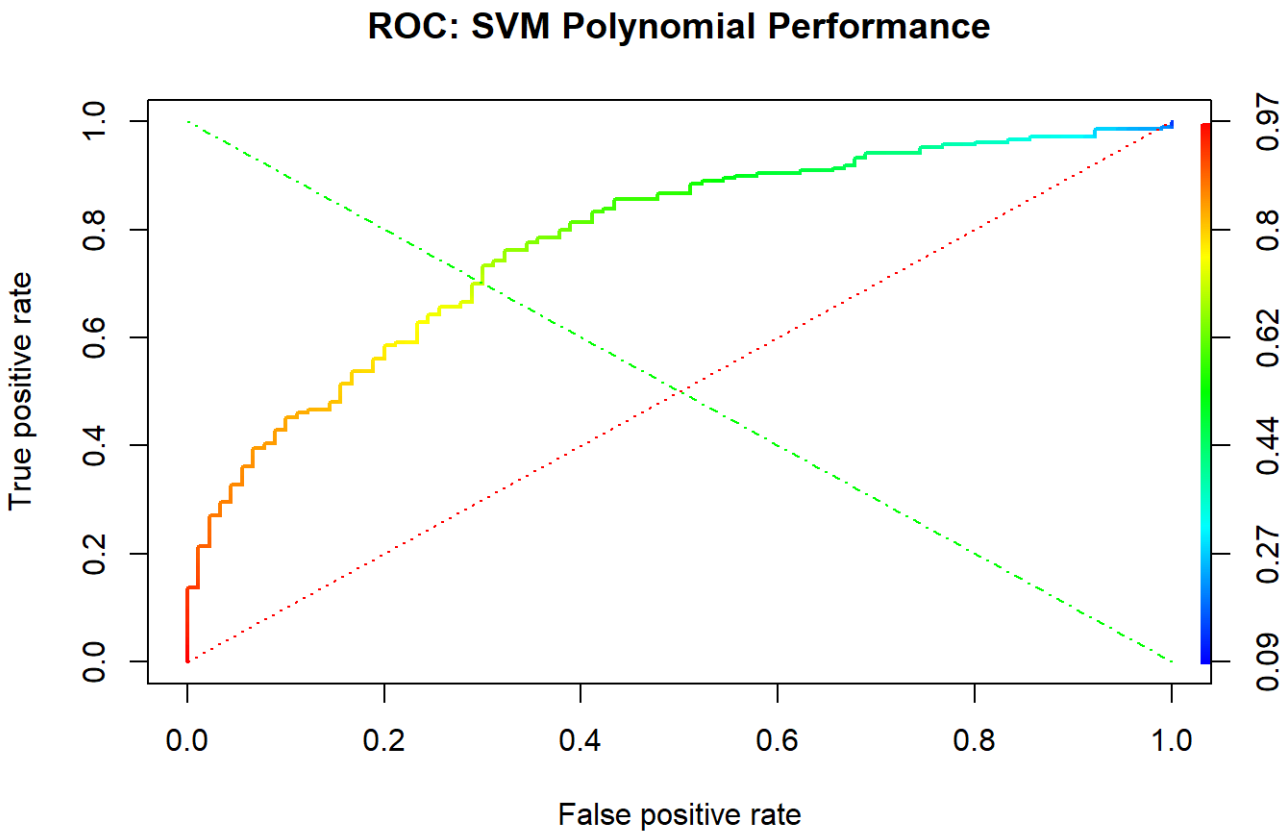
Seleccionamos el umbral que maximiza la F1 (cuando empieza a decaer)

```
umbfinal_svm_p <- umb_svm_p[which.max(umb_svm_p$F1),1]
umbfinal_svm_p
```

```
## [1] 0.55
```

Paso 4. Curva ROC

```
pred_svm_p <- prediction(svm_score_p, test$target)
perf_svm_p <- performance(pred_svm_p,"tpr","fpr")
#library(ROCR)
plot(perf_svm_p, lwd=2, colorize=TRUE, main="ROC: SVM Polynomial Performance")
lines(x=c(0, 1), y=c(0, 1), col="red", lwd=1, lty=3);
lines(x=c(1, 0), y=c(0, 1), col="green", lwd=1, lty=4)
```



Paso 5. Métricas definitivas

```
#Matriz de confusión con umbral final
score <- ifelse(svm_score_p > umbfinal_svm_p, "Good", "Bad")
MC <- table(test$target, score)
Acc_svm_p <- round((MC[1,1] + MC[2,2]) / sum(MC) *100, 2)
Sen_svm_p <- round(MC[2,2] / (MC[2,2] + MC[1,2]) *100, 2)
Pr_svm_p <- round(MC[2,2] / (MC[2,2] + MC[2,1]) *100, 2)
F1_svm_p <- round(2*Pr_svm_p*Sen_svm_p/(Pr_svm_p+Sen_svm_p), 2)

#AUC
AUROC_svm_p <- round(performance(pred_svm_p, measure = "auc")@y.values[[1]]*100, 2)

#Métricas finales del modelo
cat("Acc_svm_p: ", Acc_svm_p,"\tSen_svm_p: ", Sen_svm_p, "\tPr_svm_p:", Pr_svm_p, "\tF1_sv
m_p:", F1_svm_p, "\tAUROC_svm_p: ", AUROC_svm_p)
```

```
## Acc_svm_p:  76.33    Sen_svm_p:  80.09    Pr_svm_p: 88.1  F1_svm_p: 83.9  AUROC_svm_p:
77.81
```

Se obtiene una AUC de 77.89, lo que indica un modelo moderadamente aceptable.

5. Comparación de los tres modelos

```
# Etiquetas de filas
models <- c('SVM_l', 'SVM_r', 'SVM_p')

#Accuracy
models_Acc <- c(Acc_svm_l, Acc_svm_r, Acc_svm_p)

#Sensibilidad
models_Sen <- c(Sen_svm_l, Sen_svm_r, Sen_svm_p)

#Precisión
models_Pr <- c(Pr_svm_l, Pr_svm_r, Pr_svm_p)

#F1
models_F1 <- c(F1_svm_l, F1_svm_r, F1_svm_p)

# AUC
models_AUC <- c(AUROC_svm_l, AUROC_svm_r, AUROC_svm_p)
```

```
# Combinar métricas
metricas <- as.data.frame(cbind(models, models_Acc, models_Sen, models_Pr, models_F1, mode
ls_AUC))
```

```
# Colnames
colnames(metricas) <- c("Model", "Acc", "Sen", "Pr", "F1", "AUC")
```

```
# Tabla final de métricas
kable(metricas, caption ="Comparision of Model Performances")
```

Comparision of Model Performances

Model	Acc	Sen	Pr	F1	AUC
SVM_l	76	78.51	90.48	84.07	77.82
SVM_r	71	70.99	99.05	82.7	74.79
SVM_p	76.33	80.09	88.1	83.9	77.81

Se observa que los modelos SVM lineal (AUC = 77.88) y polinómico (AUC = 77.89) tienen un rendimiento superior al radial (AUC = 74.79).