Autor: Claudia Jara Herrera

_ _

MODELOS DE PREDICCION PARA DETECCION DE ANTIMALWARE

El presente informe presenta los pasos a seguir para definir el mejor modelo de predicción para la detección de antimalware, la data analizada es provista por Kaggle (https://www.kaggle.com/competitions/microsoft-malware-prediction):

La metodología que se tomará como referencia para el análisis de datos será la CRISP-DM (Cross Industry Standard Process for Data Mining). A continuación, se presentan las diferentes etapas que se han tomado en cuenta para el análisis:

i. Business Understanding

El objetivo de este modelado es predecir si una maquina es vulnerable de recibir un ataque malware, considerando las características principales. La variable objetivo es "Hasdetections' y es de tipo binario. Se van a analizar los siguientes modelos: Redes Neuronales, Regresión Logística, Arboles. Random Forest, Bagging, Gradient Boosting, Support Vector Machines. Tambien se realizará una prueba de ensamblado.

ii. Data Understanding & Data Preparation

La data evaluada tiene mas de 8000 registros para el archivo de Training. Para este trabajo solamente se ha tomado la muestra de las primeras 800 líneas. El total de variables es de 83. Se ha realizado un primer análisis para revisar los tipos de cada variable.

a) Validación de tipos de variables:

De acuerdo a la descripción de cada variable y lo analizado de lo cargado por R, se ha colocado el tipo correcto de acuerdo al análisis realizado.

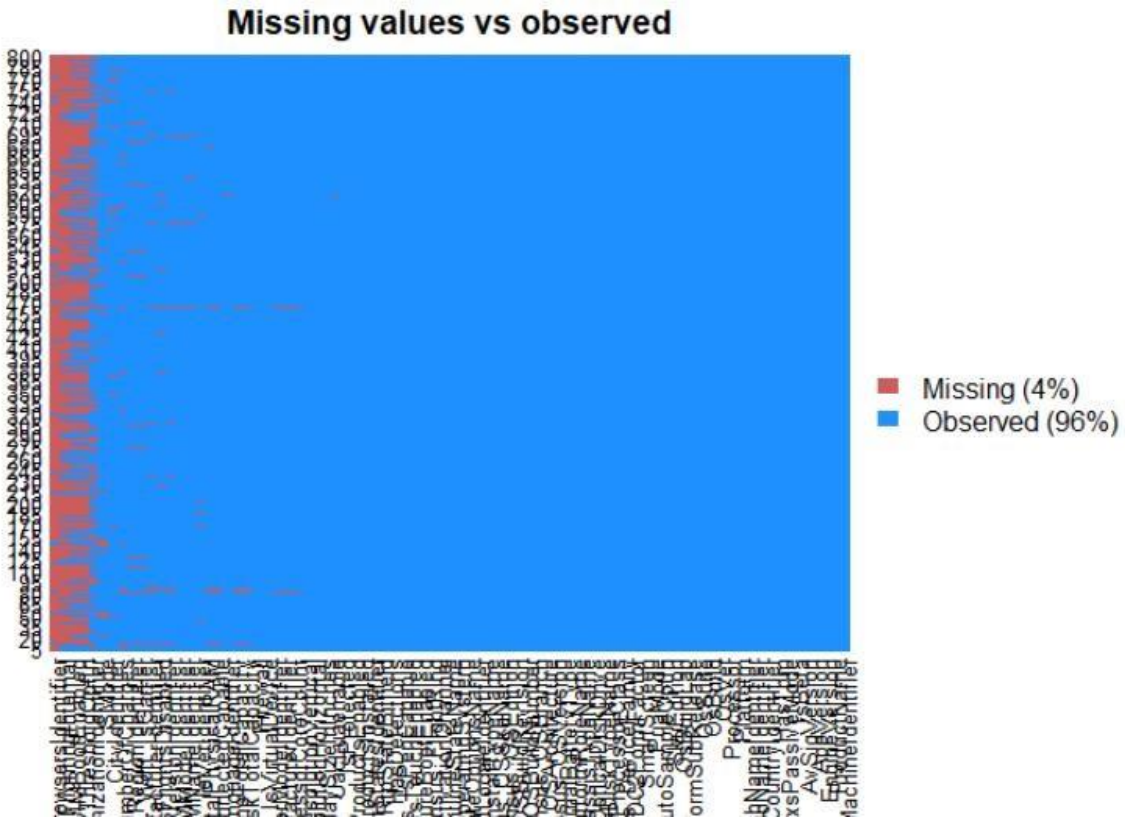
Codigo R
Retipado ---------- ##se va a revisar los datos para validar que esta correctamente categorizados datafile$MachineIdentifier <-toString(datafile$MachineIdentifier) #es del tipo string, no entrara en el analisis datafile$EngineVersion <- toString(datafile$EngineVersion) datafile$AppVersion <- toString(datafile$AppVersion) datafile$AvSigVersion <- toString(datafile$AvSigVersion) datafile$IsBeta <- factor(datafile$IsBeta) datafile$RtpStateBitfield <- factor(datafile$RtpStateBitfield) datafile$IsSxsPassiveMode <- factor(datafile$IsSxsPassiveMode) datafile$AVProductsInstalled <- factor(datafile$AVProductsInstalled)

```
datafile$AVProductsEnabled <-
factor(datafile$AVProductsEnabled) datafile$HasTpm <-
factor(datafile$HasTpm) datafile$OsVer <-
toString(datafile$OsVer) datafile$OsBuildLab <-
toString(datafile$OsBuildLab) datafile$IsProtected <-
factor(datafile$IsProtected)
datafile$AutoSampleOptIn <- factor(datafile$AutoSampleOptIn)
datafile$SMode <- factor(datafile$SMode)
datafile$SmartScreen <- toString(datafile$SmartScreen)
datafile$Firewall <- factor(datafile$Firewall)
datafile$UacLuaenable <-factor(datafile$UacLuaenable)
datafile$Census_MDC2FormFactor <-
toString(datafile$Census_MDC2FormFactor)
datafile$Census_HasOpticalDiskDrive <-
factor(datafile$Census_HasOpticalDiskDrive)
datafile$Census_ChassisTypeName <-
toString(datafile$Census_ChassisTypeName)
datafile$Census_PowerPlatformRoleName     <-
toString(datafile$Census_PowerPlatformRoleName)
datafile$Census_InternalBatteryType <-
toString(datafile$Census_InternalBatteryType) datafile$Census_OSVersion <-
toString(datafile$Census_OSVersion) datafile$Census_OSArchitecture <-
factor(datafile$Census_OSArchitecture) datafile$Census_OSBranch <-
toString(datafile$Census_OSBranch) datafile$Census_OSEdition <-
toString(datafile$Census_OSEdition)
datafile$Census_OSSkuName <- toString(datafile$Census_OSSkuName)
datafile$Census_IsPortableOperatingSystem <-
factor(datafile$Census_IsPortableOperatingSystem)
datafile$Census_GenuineStateName <-
factor(datafile$Census_GenuineStateName) datafile$Census_IsFlightingInternal
<- factor(datafile$Census_IsFlightingInternal) datafile$Census_IsFlightsDisabled
<- factor(datafile$Census_IsFlightsDisabled) datafile$Census_ThresholdOptIn <-
factor(datafile$Census_ThresholdOptIn) datafile$Census_IsSecureBootEnabled
<- factor(datafile$Census_IsSecureBootEnabled)
datafile$Census_IsWIMBootEnabled <-
factor(datafile$Census_IsWIMBootEnabled)
datafile$Census_IsVirtualDevice <- factor(datafile$Census_IsVirtualDevice)
datafile$Census_IsTouchEnabled <- factor(datafile$Census_IsTouchEnabled)
datafile$Census_IsPenCapable <- factor(datafile$Census_IsPenCapable)
datafile$Census_IsAlwaysOnAlwaysConnectedCapable        <-
factor(datafile$Census_IsAlwaysOnAlwaysConnectedCapable)
datafile$Wdft_IsGamer <- factor(datafile$Wdft_IsGamer) datafile$HasDetections
<- factor(datafile$HasDetections
```

Se definieron como string aquellas variables que tenían demasiados factores. En este primer análisis no serán consideradas para los trabajos de análisis.

b) Analisis de Missings:

Se ha validado la cantidad de Missings por cada variable y en total de la data. En este caso se ha visto que no es posible borrar toda las missings, dado que solo el 0.0025 de la data se encuentra completo.
El siguiente grafico muestra el porcentaje de missing values vs observados



Para llegar a estas conclusiones se han ejecutado las sentencias R

Código R
Input

matriznulos<-matrizdenulos(datafile) #se deinio una funcion para obtener una tabla con los nulos por dataset pasado.
 	
nrows <- nrow(datafile)
ncomplete <- sum(complete.cases(datafile)) result <- ncomplete/nrows result
#porcentaje de filas completas, se observa que elos el 0.0025 esta completo por lo que se tendra que analizar cada valor nulo.
 	
#graficas para apoyar visualizacion de datos nulos
# - Análisis de la coexistencia de datos faltantes en varias variables 	
corrplot(cor(is.na(datafile[colnames(datafile)[colSums(
is.na(datafile))>0]])),method = "ellipse",type = "upper")
 	
dfplot(datafile) #grafica para ver data y q nulos
missmap(datafile, main = "Missing values vs observed")

La siguiente tabla muestra la cantidad de missings o desaparecidos:

	Campo	Qnulos	PerNulos
[1,]	"RtpStateBitfield"	"1"	"0.00125"
[2,]	"DefaultBrowsersIdentifier"	"752"	"0.94"
[3,]	"AVProductStatesIdentifier"	"1"	"0.00125"
[4,]	"AVProductsInstalled"	"1"	"0.00125"
[5,]	"AVProductsEnabled"	"1"	"0.00125"
[6,]	"CityIdentifier"	"39"	"0.04875"
[7,]	"OrganizationIdentifier"	"254"	"0.3175"
[8,]	"IsProtected"	"1"	"0.00125"
[9,]	"SMode"	"49"	"0.06125"
[10,]	"IeVerIdentifier"	"10"	"0.0125"
[11,]	"Firewall"	"5"	"0.00625"
[12,]	"UacLuaenable"	"1"	"0.00125"
[13,]	"Census_OEMNameIdentifier"	"10"	"0.0125"
[14,]	"Census_OEMModelIdentifier"	"11"	"0.01375"
[15,]	"Census_ProcessorCoreCount"	"3"	"0.00375"
[16,]	"Census_ProcessorManufacturerIdentifier"	"3"	"0.00375"
[17,]	"Census_ProcessorModelIdentifier"	"3"	"0.00375"
[18,]	"Census_PrimaryDiskTotalCapacity"	"5"	"0.00625"
[19,]	"Census_SystemVolumeTotalCapacity"	"5"	"0.00625"
[20,]	"Census_TotalPhysicalRAM"	"9"	"0.01125"
[21,]	"Census_InternalPrimaryDiagonalDisplaySizeInInches"	"2"	"0.0025"
[22,]	"Census_InternalPrimaryDisplayResolutionHorizontal"	"2"	"0.0025"
[23,]	"Census_InternalPrimaryDisplayResolutionVertical"	"2"	"0.0025"
[24,]	"Census_InternalBatteryNumberOfCharges"	"28"	"0.035"
[25,]	"Census_OSInstallLanguageIdentifier"	"5"	"0.00625"
[26,]	"Census_IsFlightingInternal"	"648"	"0.81"
[27,]	"Census_IsFlightsDisabled"	"18"	"0.0225"
[28,]	"Census_ThresholdOptIn"	"499"	"0.62375"
[29,]	"Census_FirmwareManufacturerIdentifier"	"20"	"0.025"
[30,]	"Census_FirmwareVersionIdentifier"	"17"	"0.02125"
[31,]	"Census_IsWIMBootEnabled"	"499"	"0.62375"
[32,]	"Census_IsVirtualDevice"	"3"	"0.00375"
[33,]	"Census_IsAlwaysOnAlwaysConnectedCapable"	"5"	"0.00625"
[34,]	"Wdft_IsGamer"	"26"	"0.0325"
[35,]	"Wdft_RegionIdentifier"	"26"	"0.0325"

De lo analizado en los valores nulos, se va a eliminar las filas que tengan campos missings menores o iguales a 5, estos representan la minoría, tambien se va a eliminar columnas con % nulos mayor a 50%

Código R
Input

```
matriznulos<-matrizdenulos(datafile)  #se definio una funcion para obtener una tabla con los nulos por dataset pasado.

nrows <- nrow(datafile)
ncomplete <- sum(complete.cases(datafile)) result <- ncomplete/nrows   result
#porcentaje de filas completas, se observa que elos el 0.0025 esta completo por lo que se tendra que analizar cada valor nulo.

 #graficas para apoyar visualizacion de datos nulos
 # - Análisis de la coexistencia de datos faltantes en varias variables
corrplot(cor(is.na(datafile[colnames(datafile)[colSums(
is.na(datafile))>0]])),method = "ellipse",type = "upper")

 dfplot(datafile)  #grafica para ver data y q nulos
 missmap(datafile, main = "Missing values vs observed")

datafile<-datafile[!is.na(datafile$RtpStateBitfield), ]
datafile<-datafile[!is.na(datafile$AVProductStatesIdentifier),  ]  datafile<-
datafile[!is.na(datafile$AVProductsInstalled), ]
datafile<-datafile[!is.na(datafile$AVProductsEnabled), ] datafile<-
datafile[!is.na(datafile$IsProtected), ] datafile<-datafile[!is.na(datafile$UacLuaenable), ]
datafile<-datafile[!is.na(datafile$Census_InternalPrimaryDiagonalDisplaySizeInInches), ]
datafile<-datafile[!is.na(datafile$Census_InternalPrimaryDisplayResolutionHorizontal), ]
datafile<-datafile[!is.na(datafile$Census_InternalPrimaryDisplayResolutionVertical), ]
datafile<-datafile[!is.na(datafile$Census_ProcessorCoreCount), ] datafile<-
datafile[!is.na(datafile$Census_ProcessorManufacturerIdentifier), ] datafile<-
datafile[!is.na(datafile$Census_ProcessorModelIdentifier), ] datafile<-
datafile[!is.na(datafile$Census_IsVirtualDevice), ] datafile<-
datafile[!is.na(datafile$Firewall), ] datafile<-
datafile[!is.na(datafile$Census_PrimaryDiskTotalCapacity), ] datafile<-
datafile[!is.na(datafile$Census_SystemVolumeTotalCapacity), ] datafile<-
datafile[!is.na(datafile$Census_OSInstallLanguageIdentifier), ] datafile<-
datafile[!is.na(datafile$Census_IsAlwaysOnAlwaysConnectedCapable), ]

#tambien se va a eliminar columans con % nulos mayor a 50%

cols.dont.want <- c("DefaultBrowsersIdentifier", "Census_IsFlightingInternal",
            "Census_ThresholdOptIn","Census_IsWIMBootEnabled") # if you want to remove multiple columns
datafile <- datafile[, ! names(datafile) %in% cols.dont.want, drop = F]

matriznulos<-matrizdenulos(datafile)

#se encuentra que Census_TotalPhysicalRAM solo tiene 4 nulos se eliminara datafile<-
datafile[!is.na(datafile$Census_TotalPhysicalRAM), ]

matriznulos<-matrizdenulos(datafile)
```

c) Imputación de la data

Primero se va a analizar las variables representadas como categorías. Se validara en general que los datos del dataset tengan datos válidos, se eliminaran categorías poco representadas.

Código R

Input

```
#validacion de errores en clases
## get mode of all vars
var_mode <- sapply(datafile, mode)
## produce error if complex or raw is found
if (any(var_mode %in% c("complex", "raw"))) stop("complex or raw not allowed!")
## get class of all vars
var_class <- sapply(datafile, class)

## produce error if an "AsIs" object has "logical" or "character"
mode if (any(var_mode[var_class == "AsIs"] %in% c("logical",
"character"))) {   stop("matrix variables with 'AsIs' class must be
'numeric'")
}
```

```
## index of factor columns
fctr <- which(sapply(datafile, is.factor))
## factor variables that have skipped explicit conversion in step 2
# drop unused levels
datafile[fctr] <- lapply(datafile[fctr], droplevels)

for (vari in listclass)
{cat(vari,sep="\n")
  print(table(datafile[,vari],exclude=NULL))
}

#revisar categorias poco representadas, IsBeta, AutoSampleOptIn, PuaMode, SMode,
Census_DeviceFamily,
# Census_IsFlightsDisabled, Census_IsPortableOperatingSystem
#categorias con muchos nulos Census_ProcessorClass,
#eliminar categorias con poca representacion y con muchos nulos
#categoria para arreglar Census_PrimaryDiskTypeName,

cols.dont.want <- c("IsBeta", "AutoSampleOptIn", "PuaMode", "SMode",
"Census_DeviceFamily",
              "Census_IsFlightsDisabled", "Census_IsPortableOperatingSystem",
"Census_ProcessorClass") # if you want to remove multiple columns
datafile <- datafile[, ! names(datafile) %in% cols.dont.want, drop = F]

datafile$Census_PrimaryDiskTypeName<-
replace(datafile$Census_PrimaryDiskTypeName,
                              which(datafile$Census_PrimaryDiskTypeName
%in% c("", "UNKNOWN", "Unspecified")), NA)

listclass <- c("ProductName", "RtpStateBitfield",
"IsSxsPassiveMode","AVProductsInstalled",
          "AVProductsEnabled", "HasTpm","Platform", "Processor", "OsPlatformSubRelease",
"SkuEdition", "IsProtected",  "Firewall", "UacLuaenable",
          "Census_PrimaryDiskTypeName",
          "Census_HasOpticalDiskDrive","Census_OSArchitecture",
"Census_OSInstallTypeName",
          "Census_OSWUAutoUpdateOptionsName",
          "Census_GenuineStateName","Census_ActivationChannel",
          "Census_FlightRing", "Census_IsSecureBootEnabled",
          "Census_IsVirtualDevice", "Census_IsTouchEnabled",
          "Census_IsPenCapable", "Census_IsAlwaysOnAlwaysConnectedCapable",
"Wdft_IsGamer")

## index of factor columns fctr <-
which(sapply(datafile, is.factor))
# drop unused levels
datafile[fctr] <- lapply(datafile[fctr], droplevels)

#validar nuevamente factores, algunos con pocas categorias y mal representadas, podrian
dar error
for (vari in listclass)
{cat(vari,sep="\n")
```

```
  print(table(datafile[,vari],exclude=NULL))
}
#se revisa que ProductName, UacLuaenable pueden eliminarse
cols.dont.want <- c("ProductName", "UacLuaenable") # if you want to remove multiple
columns datafile <- datafile[, ! names(datafile) %in% cols.dont.want, drop = F]
```

Ahora se trabaja con la imputación para cuantitativas y cualitativas, para las cuantitativas se trabajará con la media y para las cualitativas con la moda:

Codigo R

Input

```
#imputacion para las cuantitativas

vectvardep <- datafile$HasDetections
input<-datafile[,-69]

indcont <- which(sapply(input, is.integer))
input[indcont] <- sapply(input[, indcont], as.numeric)

#imputacion cualitativas

input[,as.vector(which(sapply(input, class)=="numeric"))]<-sapply(
Filter(is.numeric, input),function(x) ImputacionCuant(x,"media"))

fctr <- which(sapply(input, is.factor))
#       imputacion       cualitativas
input[fctr] <- sapply(
  Filter(is.factor, input),function(x) ImputacionCuali(x,"moda"))
#se cambia el tipo de factor a character al imputar, as? que hay que indicarle
que es factor
input[,fctr] <- lapply(input[,fctr], factor)

matriznulos <- matrizdenulos(input)



# Estandarizacion-----
means <-apply(input[,listconti],2,mean,na.rm=TRUE) sds<-
sapply(input[,listconti],sd,na.rm=TRUE)
datafile2<-scale(input[,listconti], center = means, scale
= sds) input<-
data.frame(cbind(datafile2,input[,c(listclass)])) input <-
data.frame(cbind(input, vectvardep))
colnames(input)[colnames(input)=="vectvardep"]<-vardep

matriznulos <- matrizdenulos(input)

datafile<-input
```

d) Dummies

Por ultimo se trabajara en la generación de variables dummies y se eliminaran las categorías mal representadas.

Código R

Input
library(dummies) datadummies<- dummy.data.frame(datafile, listclass, sep = ".")

```
datadummies$RtpStateBitfield.3<-NULL datadummies$RtpStateBitfield.5 <-NULL
datadummies$RtpStateBitfield.8 <- NULL datadummies$AVProductsInstalled.5 <- NULL
datadummies$AVProductsEnabled.0 <- NULL datadummies$AVProductsEnabled.4 <- NULL
datadummies$Platform.windows7 <- NULL datadummies$Processor.arm64 <- NULL
datadummies$OsPlatformSubRelease.windows7 <- NULL datadummies$SkuEdition.Invalid <- NULL
datadummies$Census_OSArchitecture.arm64 <- NULL
datadummies$Census_OSInstallTypeName.CleanPCRefresh <- NULL
datadummies$Census_GenuineStateName.UNKNOWN <- NULL
datadummies$Census_FlightRing.RP <- NULL

datafile<- datadummies

datafile$HasDetections<-ifelse(datafile$HasDetections==1,"Yes","No")
```

iii. Modelado y evaluación

Como se mencionó inicialmente se va a revisar los siguientes modelos para ver cuál es el que funciona mejor para las predicciones: Redes Neuronales, Regresión Logística, Arboles. Random Forest, Bagging, Gradient Boosting, Support Vector Machines. También se realizará una prueba de ensamblado.

a) Regresión Logística

Ante de iniciar la regresión logística se realizar la selección de variables con criterio stepwise, de lo revisado, se ha escogido un grupo para AIC y otro para BIC. Ambos grupos serán evaluados en la regresión logística, pero para las siguientes aplicaciones se trabajara con el grupo de variables que tenga mejores resultados.

Codigo R
Input

```
#seleccion de variables
full<-glm(factor(HasDetections)~.,data=datafile,family = binomial(link="logit"))
null<-glm(factor(HasDetections)~1,data=datafile,family = binomial(link="logit"))

#    ** # APLICANDO steprepetidobinaria # **  lista<-
steprepetidobinaria(data=datafile,
vardep=vardep,listconti=listconti,sinicio=12345,
sfinal=12355, porcen=0.8, criterio="AIC")
lista<-steprepetidobinaria(data=datafile,
vardep=vardep,listconti=listconti,sinicio=1111,
                  sfinal=1122,porcen=0.8,criterio="BIC")
```

Ahora se aplicará la regresión logística considerando los dos grupos de variables:

Grupo1:

```
listconti=c("AVProductsInstalled.1", "Census_OSArchitecture.amd64",

"Census_IsTouchEnabled.0","Census_OSWUAutoUpdateOptionsName.Notify",
                        "RtpStateBitfield.0",
                        "Census_IsVirtualDevice.0", "Wdft_IsGamer.0",
                        "AVProductStatesIdentifier",
"Census_OSInstallTypeName.Update",
                        "Census_FlightRing.Retail","Census_OEMNameIdentifier",
                        "Census_InternalBatteryNumberOfCharges", "Firewall.0",
                        "Census_PrimaryDiskTotalCapacity"),
```

Grupo2:

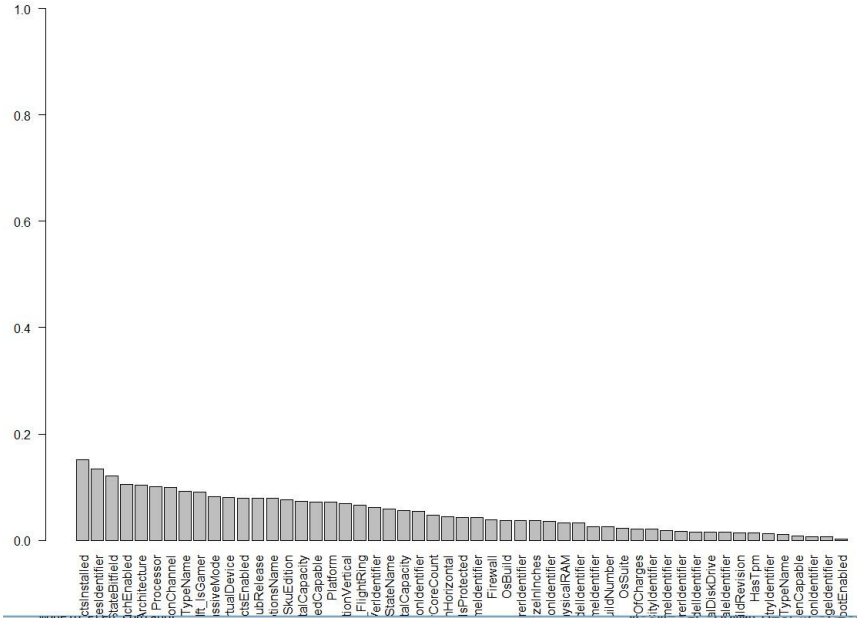
```
listconti=c("AVProductStatesIdentifier", "Census_TotalPhysicalRAM")
```

Tambien se ha seleccionado un Grupo3, escogiendo variables de acuerdo a la importancia de variables dado por la VCrammer:

Grupo3:

```
listconti=c("AVProductStatesIdentifier","Census_ProcessorCoreCount","IeVerIdentifi
er",
"Census_PrimaryDiskTotalCapacity",
"Census_InternalPrimaryDisplayResolutionVertical",
"Census_OEMNameIdentifier", "Census_FirmwareVersionIdentifier"),
```

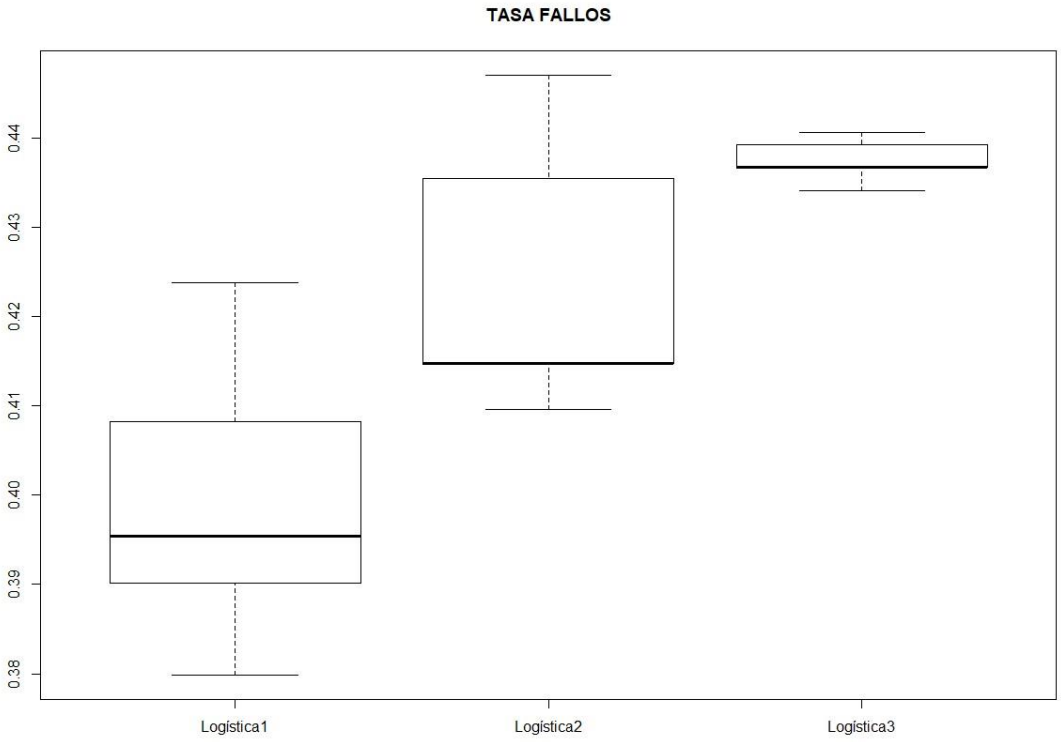
El siguiente grafico muestra la importancia de variables

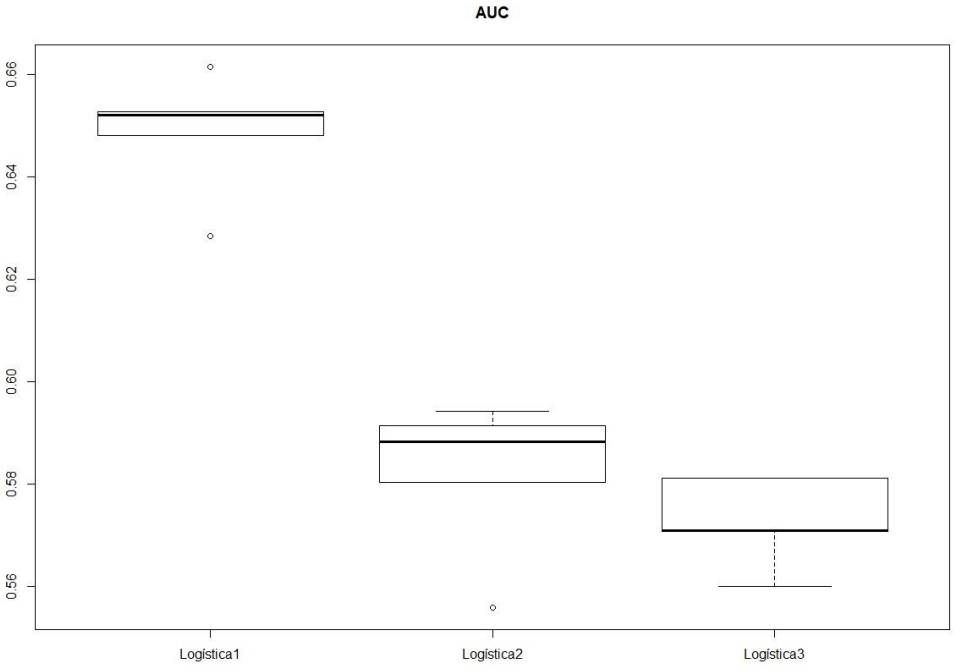


El siguiente es un trozo de la salida de la evaluación de la VCramer:

AVProductsInstalled	AVProductStatesIdentifier
0.151683818	0.134752580
RtpStateBitfield	Census_IsTouchEnabled
0.120514908	0.105530202
Census_OSArchitecture	Processor
0.103870396	0.100280847
Census_ActivationChannel	Census_OSInstallTypeName
0.099247094	0.091909706
Wdft_IsGamer	IsSxsPassiveMode
0.090933510	0.082520606
Census_IsVirtualDevice	AVProductsEnabled
0.080396028	0.079684166
OsPlatformSubRelease	Census_OSWUAutoUpdateOptionsName
0.079503429	0.078376171
SkuEdition	Census_PrimaryDiskTotalCapacity
0.076086600	0.072997215
Census_IsAlwaysOnAlwaysConnectedCapable	Platform
0.072055398	0.071300625
Census_InternalPrimaryDisplayResolutionVertical	Census_FlightRing
0.068412302	0.065443422

Resultado de la aplicación del modelo de Red Logística para los tres grupos de variables, se obtiene los siguientes resultados. Por lo observado en los gráficos, el modelo ganador es Logistica 1, que tiene como input las variables del stepwise AIC. Este será tomado en cuenta para las siguientes evaluaciones.





Codigo R
Input

```
medias1<-cruzadalogistica(data=datafile, vardep=vardep,
listconti=c("AVProductsInstalled.1", "Census_OSArchitecture.amd64",
"Census_IsTouchEnabled.0","Census_OSWUAutoUpdateOptionsName.Notify",
"RtpStateBitfield.0", "Census_IsVirtualDevice.0", "Wdft_IsGamer.0",
"AVProductStatesIdentifier", "Census_OSInstallTypeName.Update",
"Census_FlightRing.Retail","Census_OEMNameIdentifier",
"Census_InternalBatteryNumberOfCharges", "Firewall.0",
"Census_PrimaryDiskTotalCapacity"), listclass=c(""), grupos=4,sinicio=1234,repe=5)
medias1$modelo="Logística1"

medias2<-cruzadalogistica(data=datafile, vardep=vardep,
listconti=c("AVProductStatesIdentifier","Census_ProcessorCoreCount","IeVerIdentifier"
,"Census_PrimaryDiskTotalCapacity",
"Census_InternalPrimaryDisplayResolutionVertical", "Census_OEMNameIdentifier",
"Census_FirmwareVersionIdentifier"), listclass=c(""), grupos=4,sinicio=1234,repe=5)
medias2$modelo="Logística2"

medias3<-cruzadalogistica(data=datafile, vardep=vardep,
listconti=c("AVProductStatesIdentifier", "Census_TotalPhysicalRAM"),
listclass=c(""), grupos=4,sinicio=1234,repe=5) medias3$modelo="Logística3"

#esto es con valores del arbol, pero no supera al primer modelo
medias3<-cruzadalogistica(data=datafile, vardep=vardep,  #con lo obtenido del arbol
listconti=c("AVProductStatesIdentifier","Census_OEMNameIdentifier,
"Census_TotalPhysicalRAM","Census_FirmwareVersionIdentifier",
"Census_OEMModelIdentifier"),  listclass=c(""), grupos=4,sinicio=1234,repe=5)
medias3$modelo="Logística3"

union1<-rbind(medias1,medias2, medias3)
par(cex.axis=0.5)
boxplot(data=union1,tasa~modelo,main="TASA FALLOS")
boxplot(data=union1,auc~modelo,main="AUC")
```

a) Redes Neuronales

Antes de aplicar la validación cruzada con las redes neuronales, se va tunear al modelo para obtener mejor resultados.

Al aplicar el tuneo con el criterio Accuracy, se obtiene que el size debería ser y el decay 0.1 . Estos son los parámetros que se colocaran para realizar la validación cruzada. Los siguientes son los resultados aplicando el train con todas las variables

size	decay	Accuracy	Kappa
5	0.001	0.5322808	0.06183406
5	0.010	0.5356433	0.06868864
5	0.100	0.5431110	0.08432447

Los siguientes son los resultados aplicando el train con las variables escogidas con el stepwise aic.

size	decay	Accuracy	Kappa
5	0.001	0.5793762	0.1553717
5	0.010	0.5803658	0.1572337
5	0.100	0.5821660	0.1600160
10	0.001	0.5698000	0.1367004

Se valida que ambos casos el size es 5 y el decay 0.1. El ACC es mejor cuando se tiene el grupo de variables seleccionadas por stepwise aic.

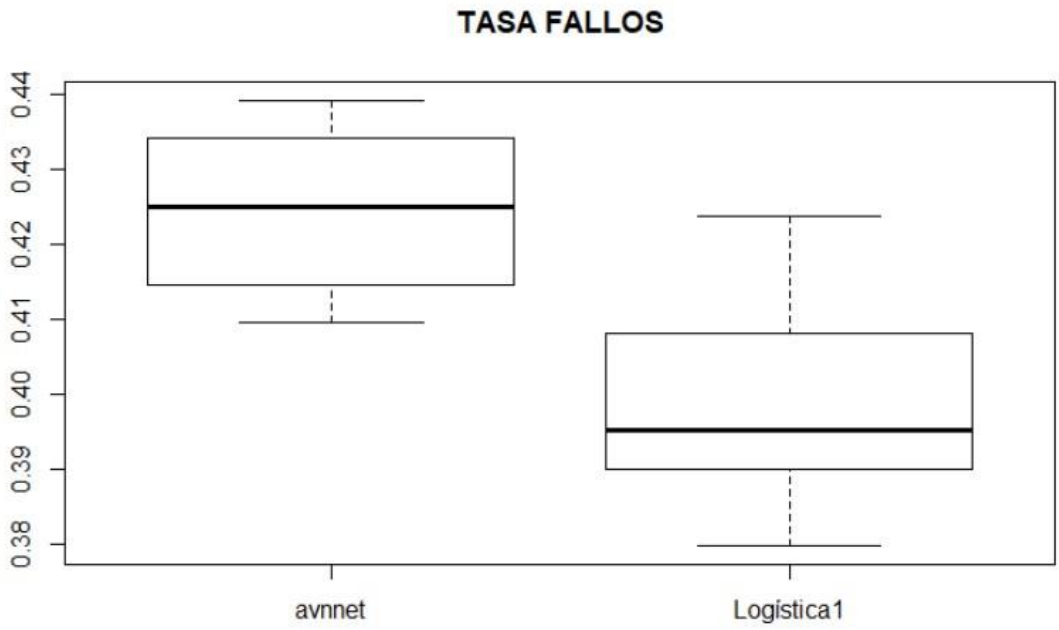
Codigo R

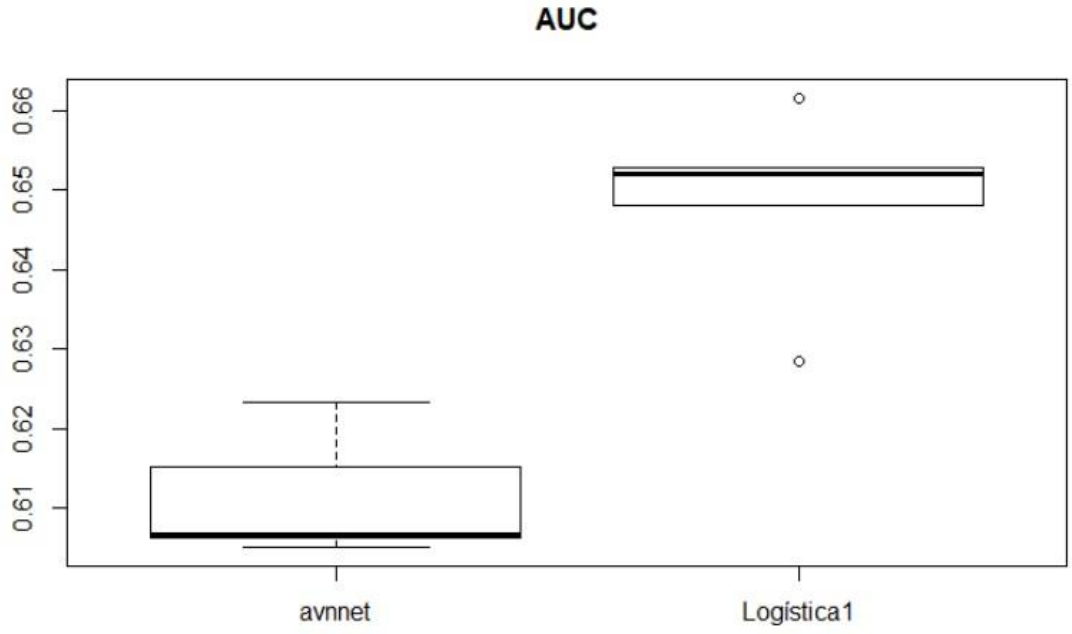
Input

```
#Tuneado de la red, con criterio Accuracy (tasa de aciertos)
# Validación cruzada repetida
control<-trainControl(method = "repeatedcv",number=4,repeats=5,
savePredictions = "all",classProbs=TRUE)
# **** avNNet: parámetros
# Number of Hidden Units (size, numeric)
# Weight Decay (decay, numeric)
# Bagging (bag, logical)
# ****************************************************
avnnetgrid <-expand.grid(size=c(5,10,15,20),
decay=c(0.01,0.1,0.001),bag=FALSE)
```

```
        redavnnet<- train(HasDetections~.,data=datafile, method="avNNet",
        linout = FALSE,maxit=100,
                trControl=control,tuneGrid=avnnetgrid, repeats=5)
redavnnet  #nos quedamos con size 5 y decay 0.1 acc 0.54

redavnnet2 <- train(HasDetections ~ AVProductsInstalled.1 +
Census_OSArchitecture.amd64 +  Census_IsTouchEnabled.0 +
Census_OSWUAutoUpdateOptionsName.Notify +  RtpStateBitfield.0
+Census_IsVirtualDevice.0 + Wdft_IsGamer.0 + AVProductStatesIdentifier +
Census_OSInstallTypeName.Update + Census_FlightRing.Retail +
Census_OEMNameIdentifier + Census_InternalBatteryNumberOfCharges +
Firewall.0 +
Census_PrimaryDiskTotalCapacity,data=datafile,  method="avNNet",  linout =
FALSE,maxit=100,  trControl=control,tuneGrid=avnnetgrid, repeats=5)

redavnnet2  #en este gana el size 5 y decay 0.1 se obtuvo eun acc=0.58
```

Con los parámetros identificados, se procedió a ejecutar la validación cruzada de la red neuronal, al compararla con la regresión logística, siguió ganando la red logística, el ACC obtenido por la red neuronal es de 0.57, en la regresión logística se obtuvo 0.66, los gráficos ayudan mejor en la comparación de ambos modelos:





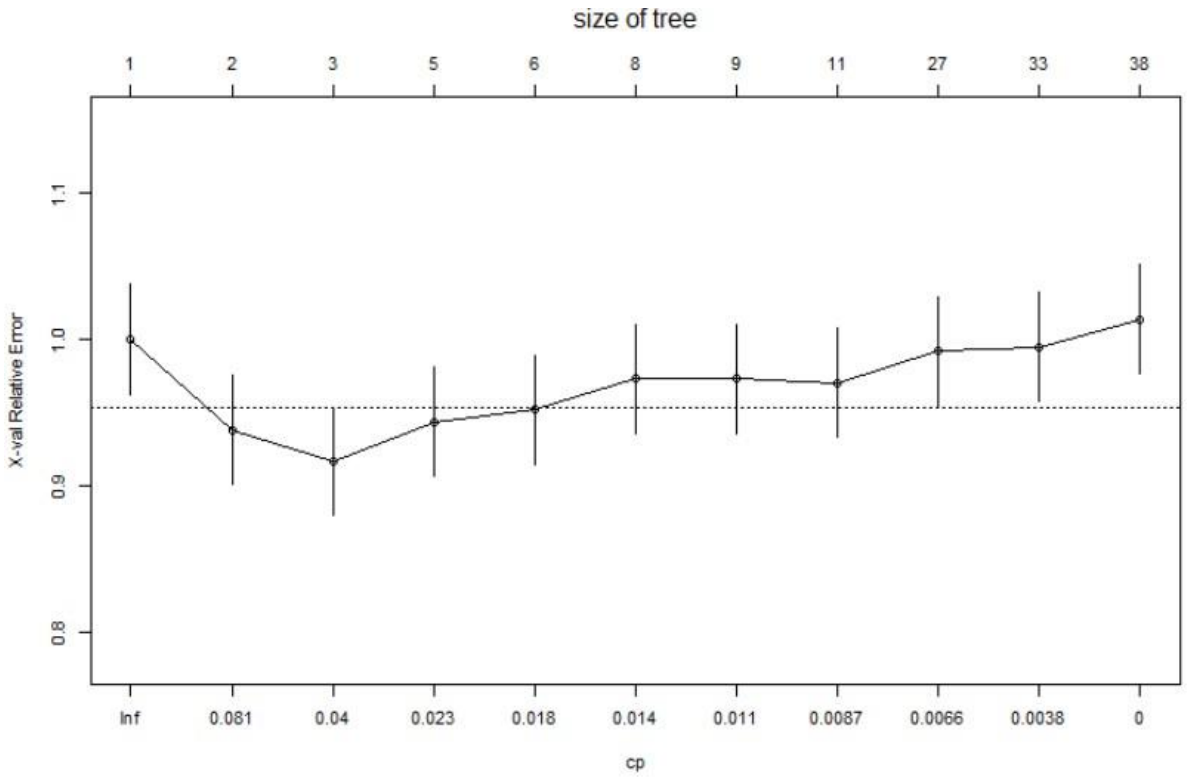
Código R
Input

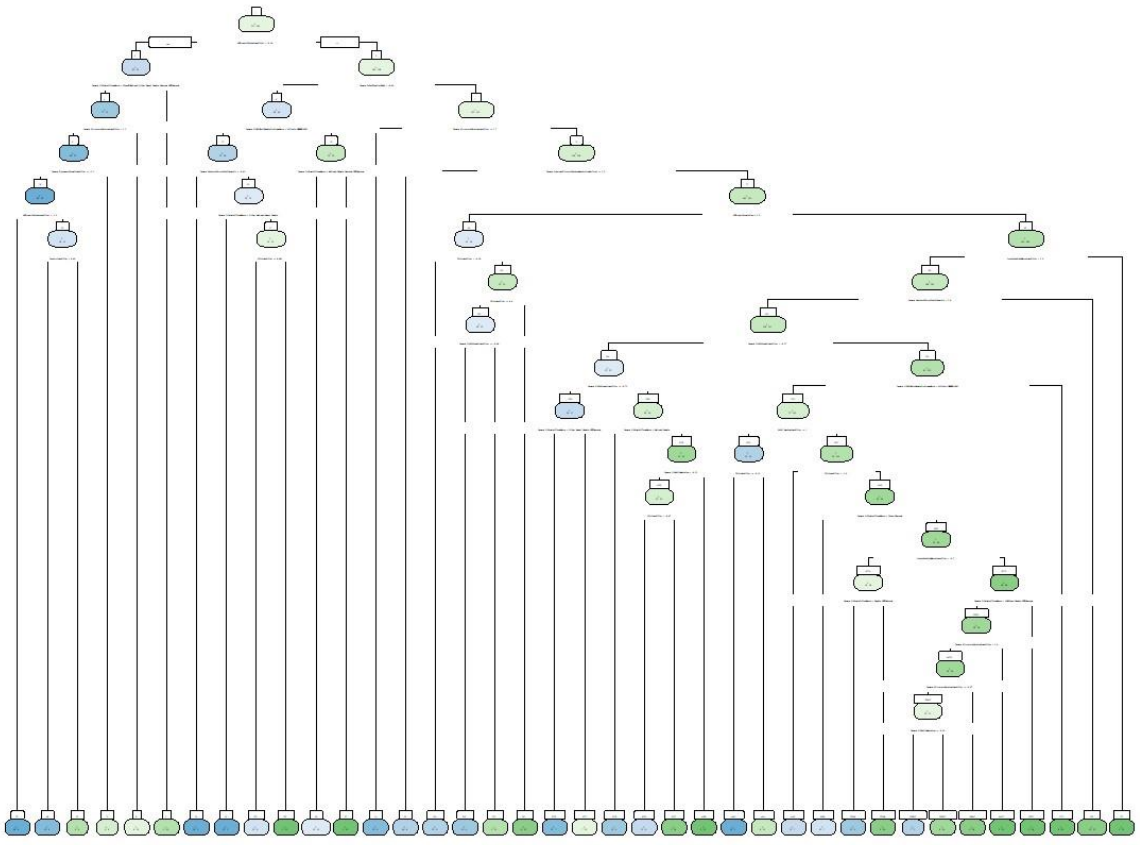
```
medias4<-cruzadaavnnetbin(data=datafile,
vardep=vardep,listconti=c("AVProductsInstalled.1",
"Census_OSArchitecture.amd64",
"Census_IsTouchEnabled.0","Census_OSWUAutoUpdateOptionsName.Notify",
"RtpStateBitfield.0",  "Census_IsVirtualDevice.0", "Wdft_IsGamer.0",
"AVProductStatesIdentifier", "Census_OSInstallTypeName.Update",
"Census_FlightRing.Retail","Census_OEMNameIdentifier",
"Census_InternalBatteryNumberOfCharges", "Firewall.0",
"Census_PrimaryDiskTotalCapacity"),
listclass=c(""),grupos=4,sinicio=1234,repe=5,
size=c(5),decay=c(0.1),repeticiones=5,itera=200) #parametros se colocan de
acuerdo a tuneado con auc
medias4$modelo="avnnet"

union1<-rbind(medias1,medias4)
par(cex.axis=0.5)
boxplot(data=union1,tasa~modelo,main="TASA FALLOS", cex=6)
boxplot(data=union1,auc~modelo,main="AUC", cex=1)
```

b) Arboles

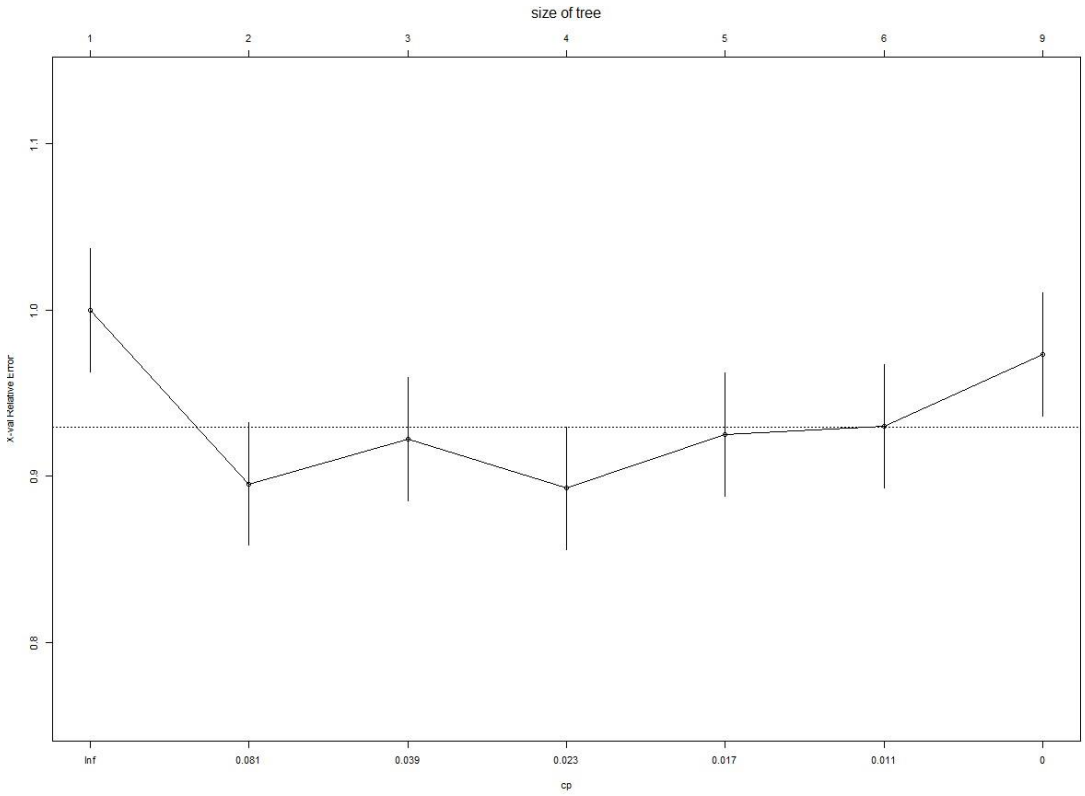
Antes de ejecutar la validación cruzada con el árbol, se va a ejecutar la selección de variables que realizan los arboles y se va a ajustar el parámetro cp y el minbucket. Se comparará la selección de variables del árbol, contra la del stepwise aic y nos quedaremos con el mejor modelo.
Se han construido dos alternativas variando el minbucket, primero 10 y luego 30, los resultado son los siguientes

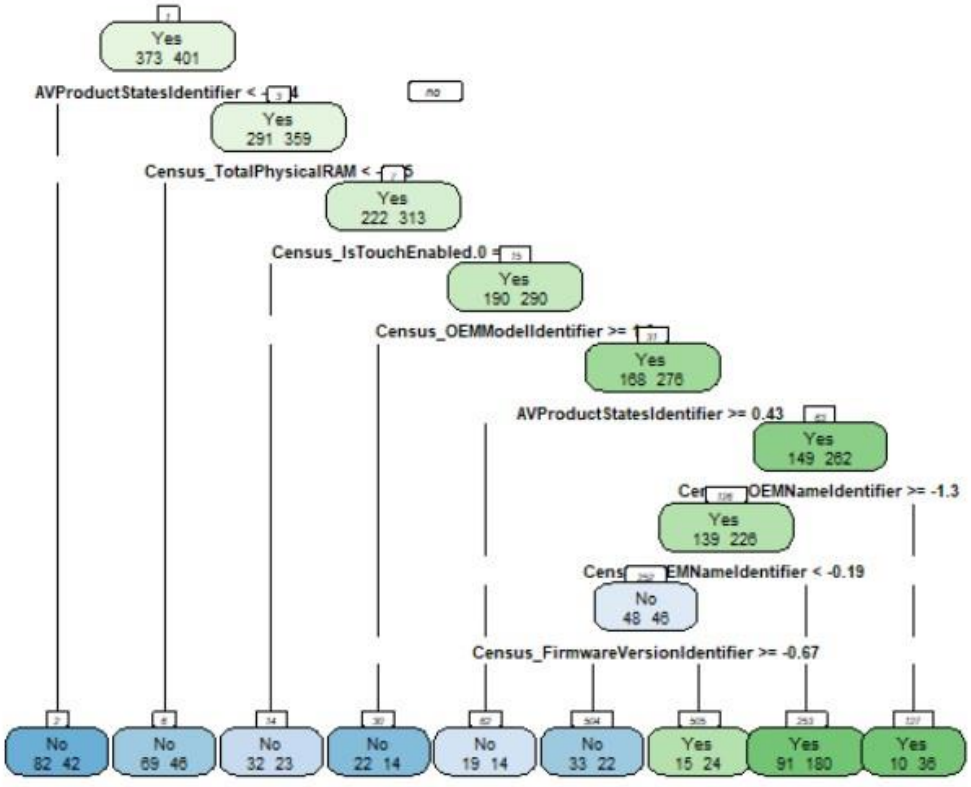
- Para minbucket 10 se obtuvo lo siguiente:



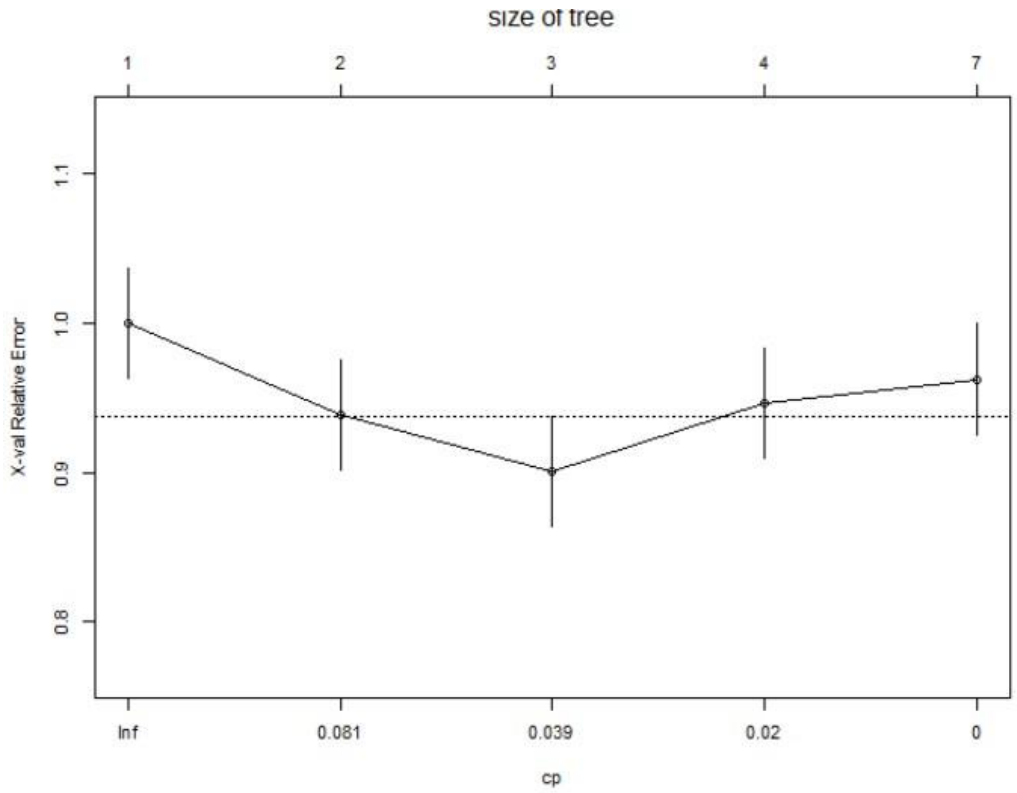


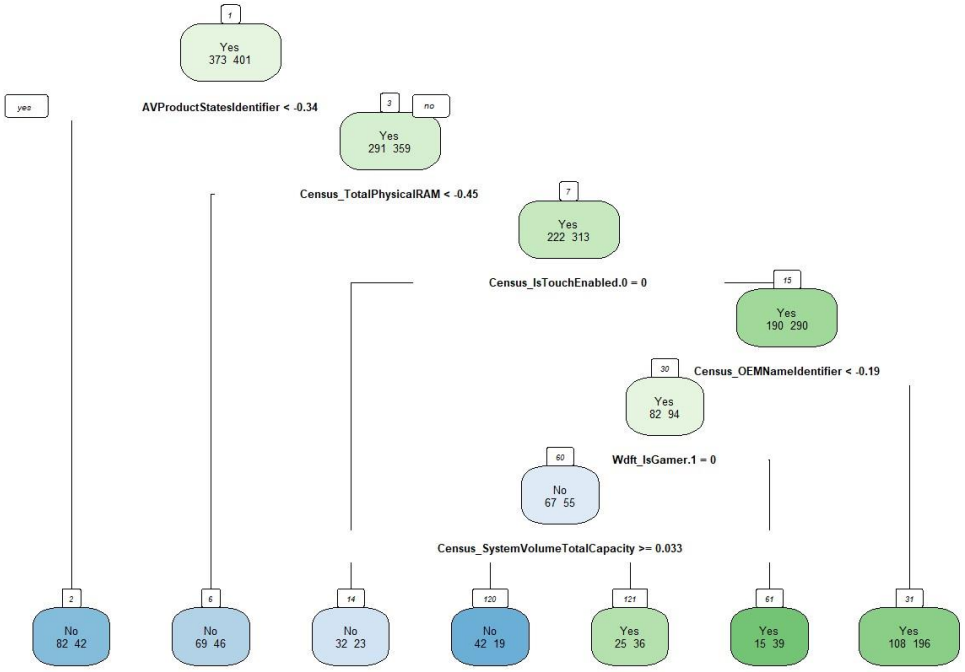
- Para minbucket 30 se obtienen los siguientes resultados





Las variables escogidas para este modelos serian AVProductStatesIdentifier, Census_TotalPhysicalRAM,Census_IsTouchEnabled, Census_OEMModelIdentifier,AVProductStatesIdentifier, Census_OEMNameIdentifier,Census_FirmwareVersionIdentifier El CP seria 0.017





Las variables escogidas para este modelos serian: AVProductStatesIdentifier, Census_TotalPhysicalRAM, Census_IsTouchEnabled.0, Census_OEMNameIdentifier, Wdft_IsGamer.1,
Census_SystemVolumeTotalCapacity El CP seria
0.02

Se va a probar el train con minbucket 30 y 40 y se elegirá el modelo como mejores auc

Para minbucket 30 y cp 0.07 se obtienen los siguientes resultados:

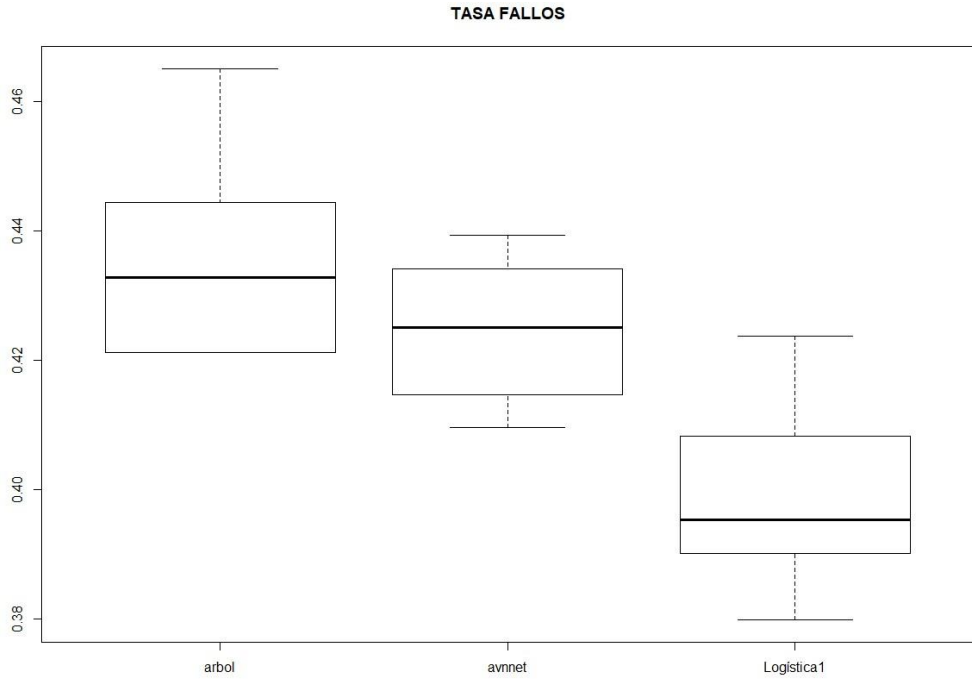
cp	Accuracy	Kappa
0.000	0.5294391	0.05791901
0.001	0.5296982	0.05834286
0.005	0.5418105	0.08082504
0.017	0.5759486	0.14425433
0.100	0.5392744	0.05393489

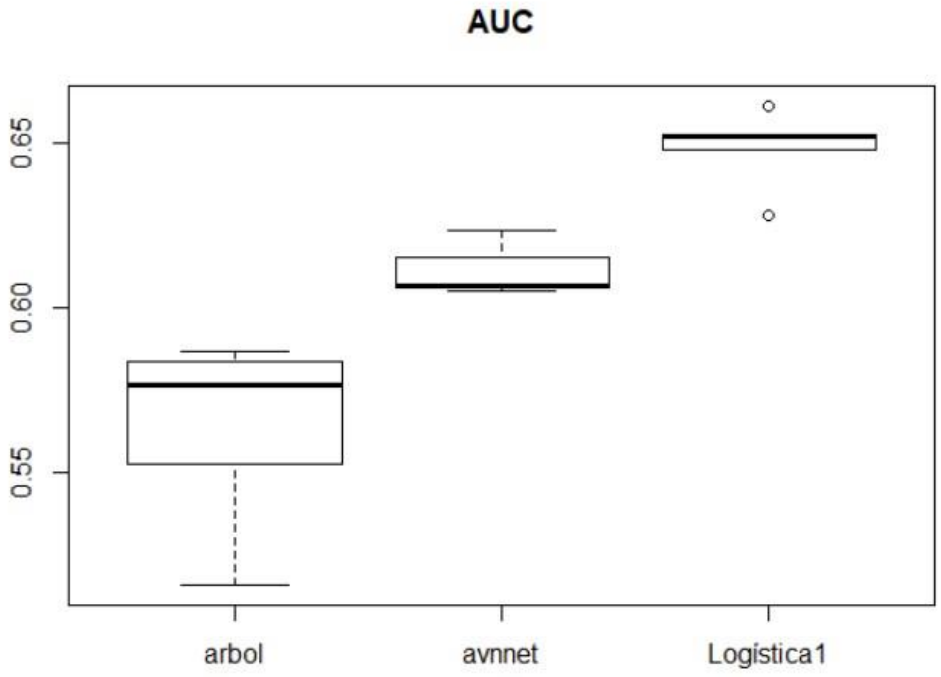
Para minbucket 40 y cp 0.02 se obtiene los siguientes resultados:

cp	Accuracy	Kappa
0.000	0.5206880	0.03874135
0.001	0.5206880	0.03874135
0.020	0.5772597	0.14535804
0.050	0.5705373	0.12707801
0.100	0.5447092	0.06545296

De acuerdo a los resultados observados, se hará la validación cruzada con minbucket 40 y cp 0.02, dado que se tiene mejor ACC.

Al realizar la validación cruzada con el árbol de decisión y comparar con los modelos anteriormente ejecutados, la regresión logística sigue siendo el mejor modelo





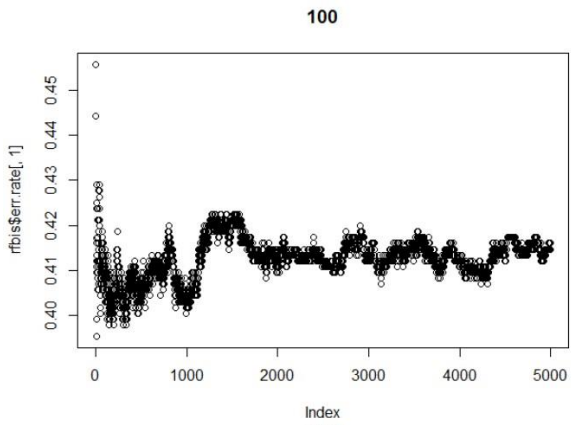
Codigo R
Input
arbol <- rpart(HasDetections~.,data = datafilebck,minbucket =10,cp=0,method = "class") summary(arbol) par(cex=0.7)

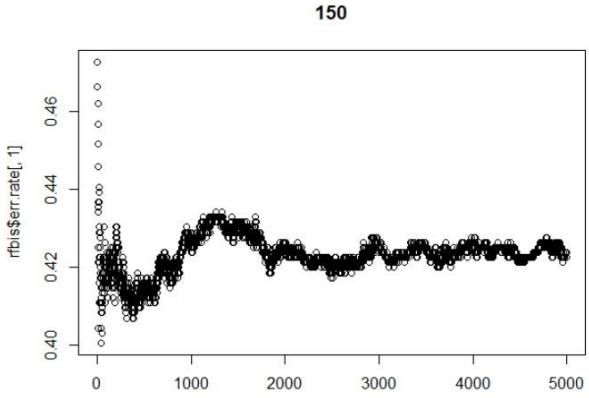
```
arbol$variable.importance
#Census_OSInstallTypeName,CityIdentifier,AVProductStatesIdentifier,Census_
SystemVolumeTotalCapacity,Census_OSBuildRevision,Census_FirmwareVersio
nIdentifier
plotcp(arbol) #cp 0.0087
rpart.plot(arbol,extra=1,tweak=1.1,nn=TRUE) par(cex=0.7)
barplot(arbol$variable.importance,col="orange")

arbol2 <- rpart(HasDetections~., data = datafile,minbucket =30, method =
"class",cp=0) summary(arbol2) arbol2$variable.importance
#AVProductStatesIdentifier,
Census_TotalPhysicalRAM,Census_IsTouchEnabled,
Census_OEMModelIdentifier,AVProductStatesIdentifier,
Census_OEMNameIdentifier,Census_FirmwareVersionIdentifier      plotcp(arbol2) #cp
0.017
rpart.plot(arbol2,extra=1,tweak=1.1,nn=TRUE) par(cex=0.7)
barplot(arbol2$variable.importance,col="orange")



arbol3 <- rpart(HasDetections~., data = datafile,minbucket =40, method =
"class",cp=0) summary(arbol3)
arbol3$variable.importance #AVProductStatesIdentifier,
Census_TotalPhysicalRAM,
Census_IsTouchEnabled.0,Census_OEMNameIdentifier,Wdft_IsGamer.1,
Census_SystemVolumeTotalCapacity
plotcp(arbol3) #cp 0.02
rpart.plot(arbol3,extra=1,tweak=1.1,nn=TRUE) par(cex=0.7)
barplot(arbol3$variable.importance,col="orange")



# CON ARBOL: con rpart SE PUEDE TUNEAR EL CP arbolgrid <-
expand.grid(cp=c(0,0.001,0.05,0.017,0.1)) arbolcaret<-
train(factor(HasDetections)~AVProductStatesIdentifier+Census_TotalPhysicalR
AM+Census_IsTouchEnabled.0+
        Census_OEMModelIdentifier+AVProductStatesIdentifier+
Census_OEMNameIdentifier+
        Census_FirmwareVersionIdentifier,          data=datafile,
method="rpart",minbucket=30,trControl=control,tuneGrid=arbolgrid)
arbolcaret sal<-arbolcaret$pred
salconfu<-confusionMatrix(sal$pred,sal$obs) salconfu
curvaroc<-roc(response=sal$obs,predictor=sal$Yes)  auc<-curvaroc$auc
```

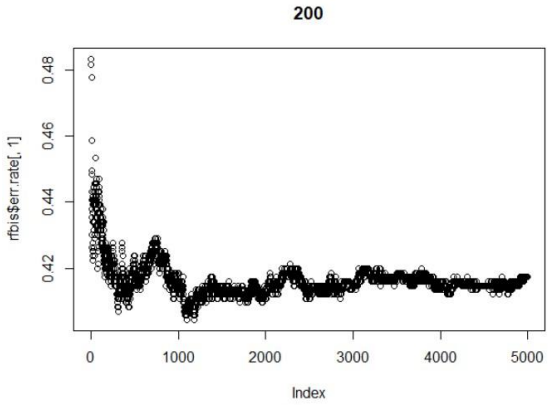
```
auc
plot(roc(response=sal$obs,predictor=sal$Yes))

arbolgrid <- expand.grid(cp=c(0,0.001,0.02,0.05,0.1)) arbolcaret2<-
train(factor(HasDetections)~AVProductStatesIdentifier+Census_TotalPhysicalR
AM+

Census_IsTouchEnabled.0+Census_OEMNameIdentifier+Wdft_IsGamer.1+Cen
sus_SystemVolumeTotalCapacity,
        data=datafile,
method="rpart",minbucket=40,trControl=control,tuneGrid=arbolgrid)
arbolcaret2 sal<-arbolcaret2$pred
salconfu<-confusionMatrix(sal$pred,sal$obs) salconfu
curvaroc<-roc(response=sal$obs,predictor=sal$Yes)  auc<-
curvaroc$auc
auc
plot(roc(response=sal$obs,predictor=sal$Yes))


medias7<-cruzadaarbolbin(data=datafile,
    vardep=vardep,
    listconti=c("AVProductStatesIdentifier","Census_TotalPhysicalRAM",

"Census_IsTouchEnabled.0","Census_OEMNameIdentifier","Wdft_IsGamer.1",
        "Census_SystemVolumeTotalCapacity"),
    listclass=c(""),grupos=4,sinicio=1234,repe=5, cp=c(0.02),minbucket =40)

medias7$modelo="arbol"

union1<-rbind(medias1,medias4, medias7)
#par(cex.axis=0.5)
boxplot(data=union1,tasa~modelo,main="TASA FALLOS", cex=6)
boxplot(data=union1,auc~modelo,main="AUC", cex=1)
```

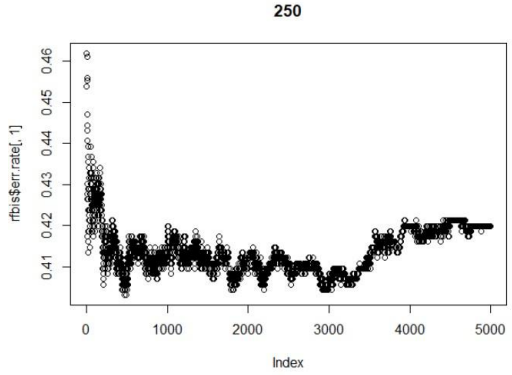
c) Bagging

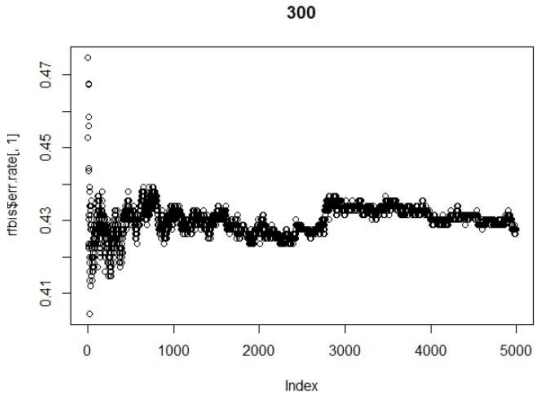
Antes de realizar la validación cruzada con bagging, se va a realizar el tuneo del sampsize

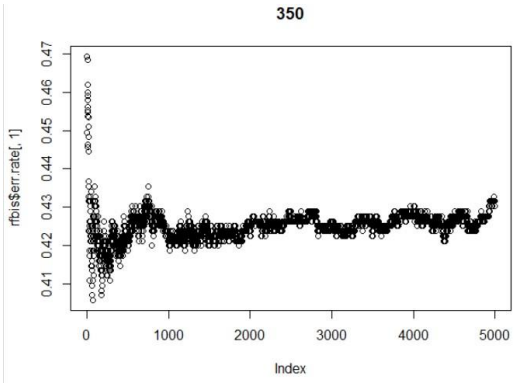


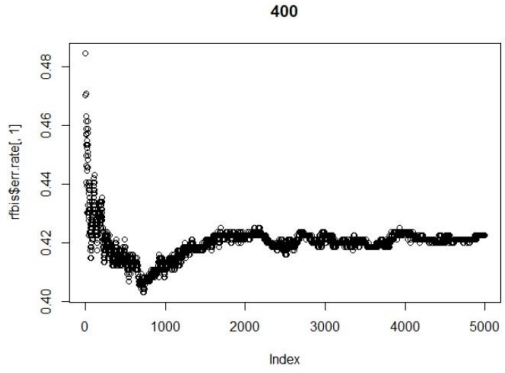


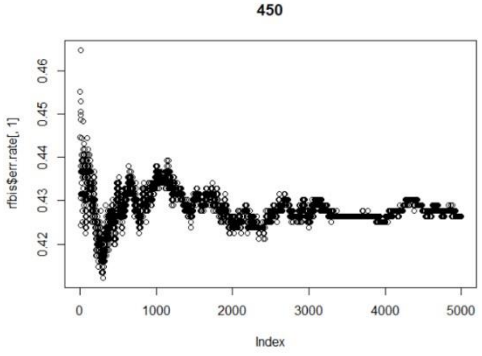






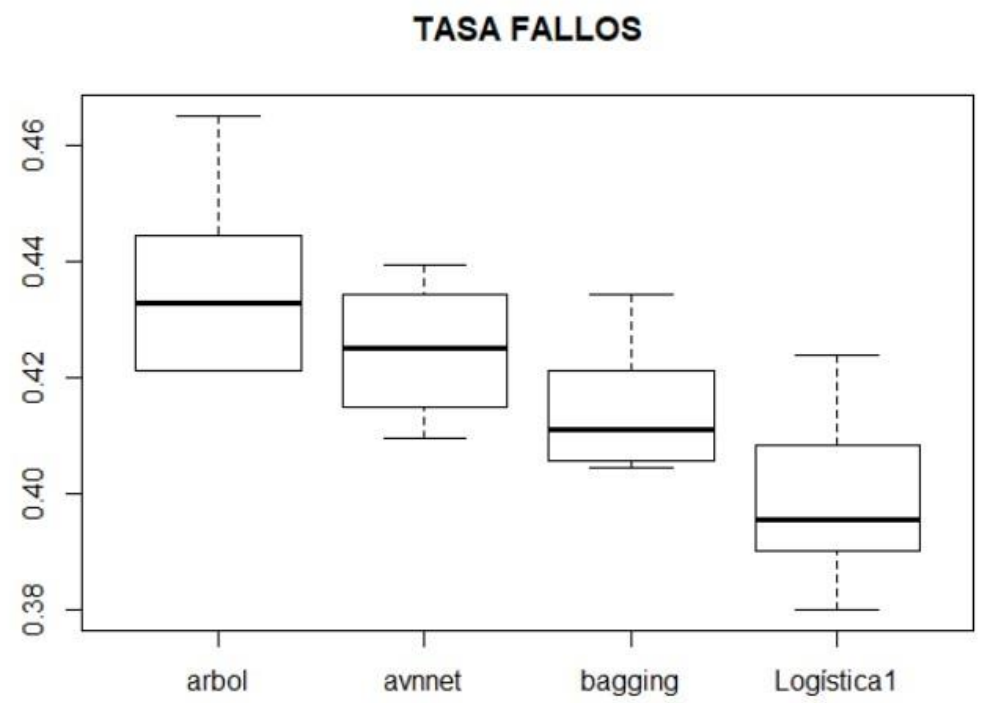


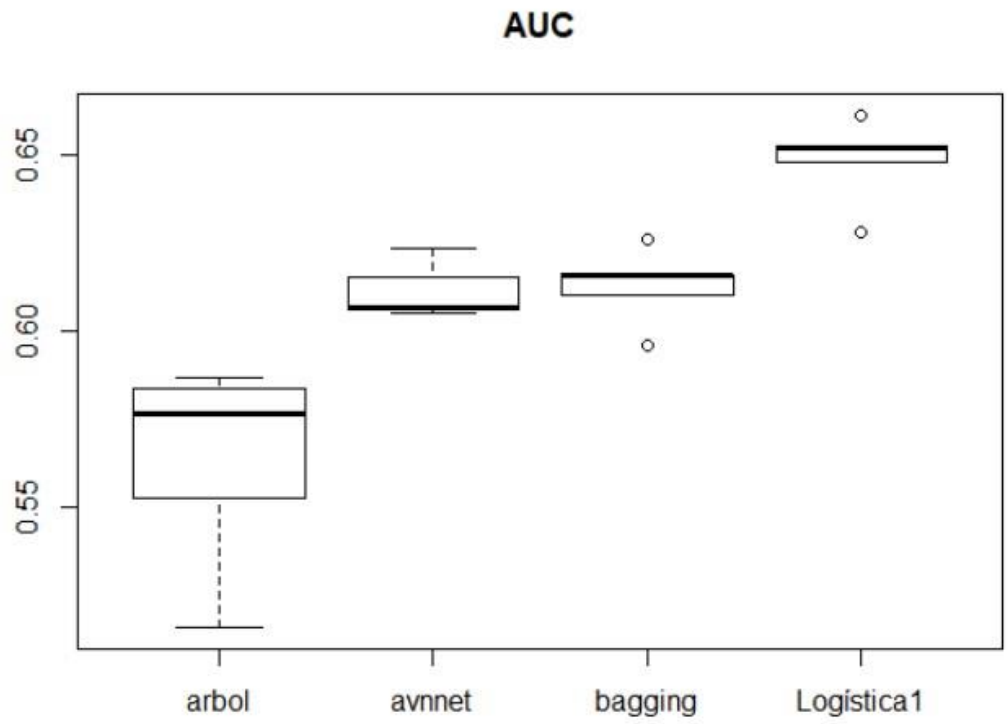




El valor escogido para sampsize será 250

Al ejecutar la validación cruzada se comprueba que la regresión logística sigue siendo el mejor modelo.





Codigo R
Input

```
set.seed(12345)

for (muestra in seq(100,450,50)) { # controlamos la semilla pues bagging depende de
ella set.seed(12345)    rfbis<-
randomForest(factor(HasDetections)~AVProductsInstalled.1+Census_OSArchitecture.a
md64+Census_IsTouchEnabled.0+
              Census_OSWUAutoUpdateOptionsName.Notify+RtpStateBitfield.0+
Census_IsVirtualDevice.0+

Wdft_IsGamer.0+AVProductStatesIdentifier+Census_OSInstallTypeName.Update+

Census_FlightRing.Retail+Census_OEMNameIdentifier+Census_InternalBatteryNumber
OfCharges+
              Firewall.0+ Census_PrimaryDiskTotalCapacity, data=datafile,
              mtry=14,ntree=5000,sampsize=muestra,nodesize=10,replace=TRUE)

  plot(rfbis$err.rate[,1],main=muestra)   print(rfbis)
}

rfgrid<-expand.grid(mtry=14) #Para bagging en mtry se pone el número total de var
inddependientes del modelo
control<-trainControl(method = "cv",number=4,savePredictions = "all",
classProbs=TRUE)
rf<-
train(factor(HasDetections)~AVProductsInstalled.1+Census_OSArchitecture.amd64+Ce
nsus_IsTouchEnabled.0+
         Census_OSWUAutoUpdateOptionsName.Notify+RtpStateBitfield.0+
Census_IsVirtualDevice.0+

Wdft_IsGamer.0+AVProductStatesIdentifier+Census_OSInstallTypeName.Update+

Census_FlightRing.Retail+Census_OEMNameIdentifier+Census_InternalBatteryNumber
OfCharges+
         Firewall.0+ Census_PrimaryDiskTotalCapacity
          ,data=datafile, method="rf",
          trControl=control,tuneGrid=rfgrid, linout = FALSE,ntree=200,
sampsize=250,nodesize=10,replace=TRUE)  rf

medias8<-cruzadarfbin(data=datafile, vardep=vardep,
              listconti=c("AVProductsInstalled.1", "Census_OSArchitecture.amd64",

"Census_IsTouchEnabled.0","Census_OSWUAutoUpdateOptionsName.Notify",
               "RtpStateBitfield.0",
               "Census_IsVirtualDevice.0", "Wdft_IsGamer.0",
```

```
            "AVProductStatesIdentifier", "Census_OSInstallTypeName.Update",
            "Census_FlightRing.Retail","Census_OEMNameIdentifier",
            "Census_InternalBatteryNumberOfCharges", "Firewall.0",
            "Census_PrimaryDiskTotalCapacity"),
listclass=c(""),
          grupos=4,sinicio=1234,repe=5,nodesize=10,
mtry=14,ntree=200,replace=TRUE)
#
medias8$modelo="bagging"

union1<-rbind(medias1,medias4, medias7, medias8)
#par(cex.axis=0.5)
boxplot(data=union1,tasa~modelo,main="TASA FALLOS", cex=6)
boxplot(data=union1,auc~modelo,main="AUC", cex=1)
```

d) Random Forest

Se procederá a tunear el parámetro mtry con caret, al ejecutar el código, se obtiene que el valor debería ser 3.

mtry	Accuracy	Kappa
3	0.5736205	0.1414763
4	0.5606805	0.1170120
5	0.5684525	0.1346069
6	0.5593852	0.1145618
7	0.5645465	0.1266876
8	0.5658819	0.1284576
9	0.5568279	0.1107847
10	0.5619759	0.1212575
11	0.5645665	0.1267636

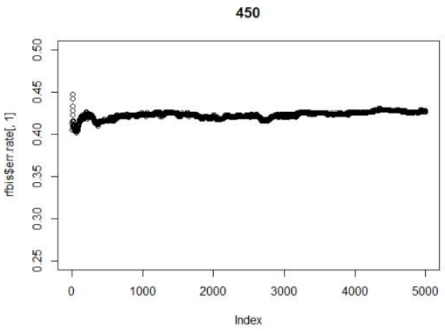
También se puede ejecutar selección de variables, al ejecutar el código se obtiene que las variables propuestas son las siguientes:

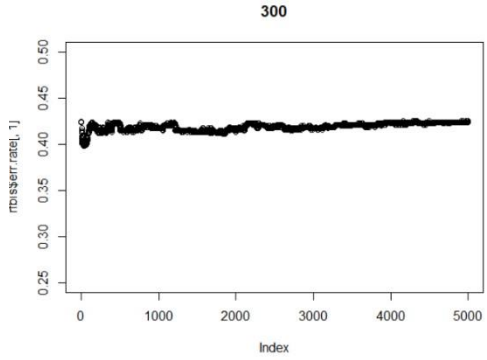
Variables: AVProductStatesIdentifier, AVProductsInstalled.2, AVProductsInstalled.1, RtpStateBitfield.0, RtpStateBitfield.7, IsSxsPassiveMode.0, Census_IsTouchEnabled.1, Census_IsTouchEnabled.0

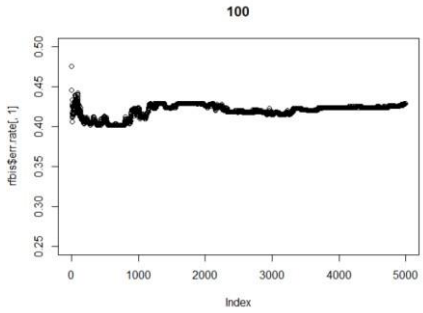
El siguiente grafico muestra los valores obtenidos en R (solo se muestra parte de las variables)

	No	Yes	MeanDecreaseAccuracy
AVProductStatesIdentifier	8.0893630	5.47564400	11.061446462
AVProductsInstalled.2	4.5783926	4.57100237	7.946340165
AVProductsInstalled.1	5.0482670	5.03636218	7.715344950
RtpStateBitfield.0	4.6827319	5.82567759	6.128870956
RtpStateBitfield.7	2.9090020	5.62234566	5.407748321
IsSxsPassiveMode.0	3.4887462	5.31461647	5.256455268
Census_IsTouchEnabled.1	3.2370258	3.61463487	5.132796723
Census_IsTouchEnabled.0	3.5429834	3.47000118	4.968018025

Ahora se tuneará el valor de la muestra, considerando las variables escogidas y el mtry tuneado. De lo observado en los gráficos, se trabajo con sampsize 300:

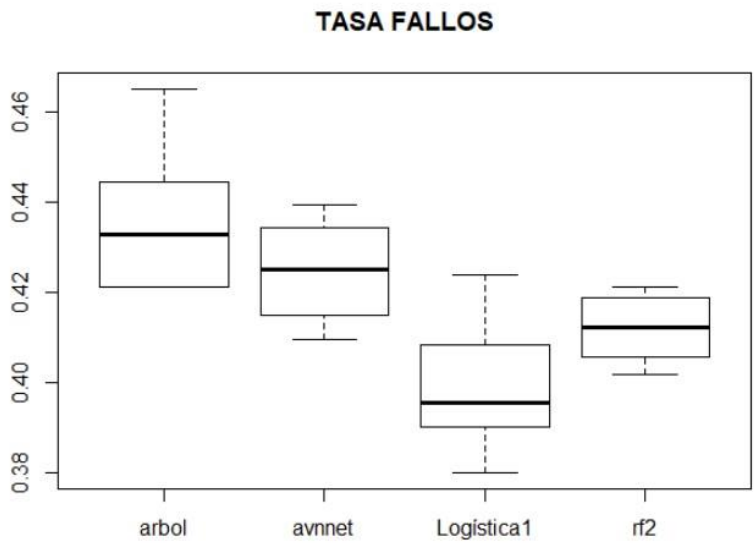


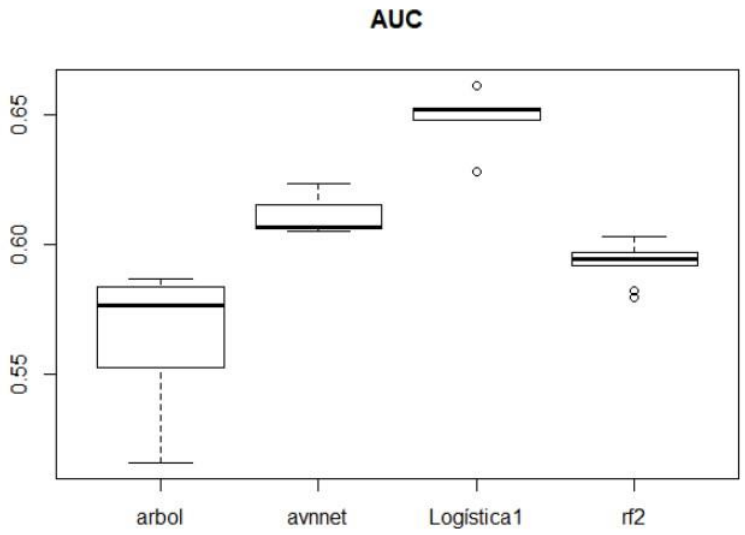




Con los datos tuneados, se ha procedido a ejecutar la validación cruzada con las variables escogidas por random forest (también se cruzo con las variables escogidas por stepwise aic, sin embargo, se obtienen mejores resultados con las escogidas por random forest)

Al comparar la validación cruzada contra los anteriores modelos, la regresión logística sigue siendo el mejor modelo:





Codigo R
Input

#para random forest se tone el mtry
# TUNEADO DE MTRY CON CARET 	
set.seed(12345)
rfgrid<-expand.grid(mtry=c(3,4,5,6,7,8,9,10,11))
control<-trainControl(method = "cv",number=4,savePredictions = "all",
classProbs=TRUE) rf<- train(HasDetections~.,data=datafile,
method="rf",trControl=control, tuneGrid=rfgrid, linout =
FALSE,ntree=1000,sampsize=250,nodesize=10, replace=TRUE,
importance=TRUE) rf #mtry = 3 es el escogido

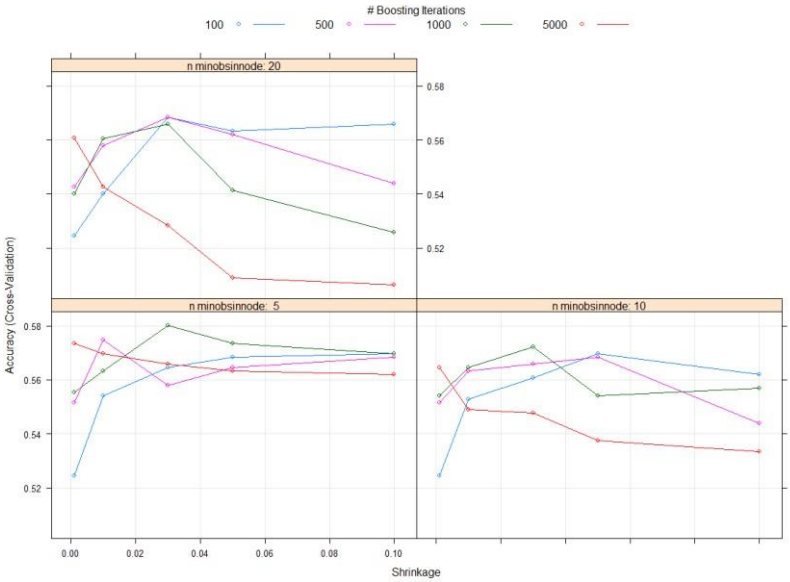
```
#importancia de las variables final<-rf$finalModel
tabla<-as.data.frame(importance(final))
tabla<-tabla[order(-tabla$MeanDecreaseAccuracy),]  tabla
#AVProductStatesIdentifier,AVProductsInstalled.1,RtpStateBitfield.7,AVProductsInstall
ed.2,RtpStateBitfield.0.,Census_OSArchitecture.x86,

barplot(tabla$MeanDecreaseAccuracy,names.arg=rownames(tabla))

#TUNEADO BÁSICO DEL TAMAÑO DE MUESTRA A SORTEAR
# NOTA: ESTE TRUCO SE PUEDE USAR EN CUALQUIER FUNCIÓN, ÚTIL #
PARA TUNEAR PARÁMETROS QUE CARET NO PERMITE EN EL GRID
for (muestra in seq(100,450,50)) { # controlamos la semilla pues bagging depende de
ella, pendiente de correr    set.seed(12345)    rfbis<-
randomForest(factor(HasDetections)~AVProductStatesIdentifier+AVProductsInstalled.1
+

RtpStateBitfield.7+AVProductsInstalled.2+RtpStateBitfield.0+Census_OSArchitecture.x8
6,
              data=datafile, mtry=3,
              ntree=5000,sampsize=muestra,nodesize=10,replace=TRUE)
plot(rfbis$err.rate[,1],main=muestra,ylim=c(0.25,0.5))   print(muestra)
print(rfbis)

}
# Ahora se comprueba con validación cruzada con caret rfgrid<-expand.grid(mtry=c(3))
rf<- train(factor(HasDetections)~AVProductStatesIdentifier+AVProductsInstalled.1+

RtpStateBitfield.7+AVProductsInstalled.2+RtpStateBitfield.0+Census_OSArchitecture.x8
6,
       data=datafile, method="rf",
       trControl=control,tuneGrid=rfgrid, linout = FALSE,
ntree=1000,sampsize=300,nodesize=10,replace=TRUE) rf

#ahora si se calcula con el mtry tuneado
medias9<-cruzadarfbin(data=datafile, vardep=vardep,
              listconti=c("AVProductsInstalled.1", "Census_OSArchitecture.amd64",

"Census_IsTouchEnabled.0","Census_OSWUAutoUpdateOptionsName.Notify",
                     "RtpStateBitfield.0",
                     "Census_IsVirtualDevice.0", "Wdft_IsGamer.0",
                     "AVProductStatesIdentifier", "Census_OSInstallTypeName.Update",
                     "Census_FlightRing.Retail","Census_OEMNameIdentifier",
                     "Census_InternalBatteryNumberOfCharges", "Firewall.0",
"Census_PrimaryDiskTotalCapacity"),
```

```
            listclass=c(""), grupos=4,sinicio=1234,repe=10,nodesize=10,
mtry=3,ntree=100,replace=TRUE,sampsize = 300)
medias9$modelo="rf1" #acc 0.5847579

medias10<-cruzadarfbin(data=datafile, vardep=vardep,
            listconti=c("AVProductStatesIdentifier","AVProductsInstalled.1",
"RtpStateBitfield.7","AVProductsInstalled.2","RtpStateBitfield.0",
                  "Census_OSArchitecture.x86"),
            listclass=c(""), grupos=4,sinicio=1234,repe=10,nodesize=10,
mtry=3,ntree=100,replace=TRUE, sampsize = 300)
medias10$modelo="rf2" #acc 0.5876017


union1<-rbind(medias1,medias4, medias7, medias10)
#par(cex.axis=0.5)
boxplot(data=union1,tasa~modelo,main="TASA FALLOS", cex=6)
boxplot(data=union1,auc~modelo,main="AUC", cex=1)
```

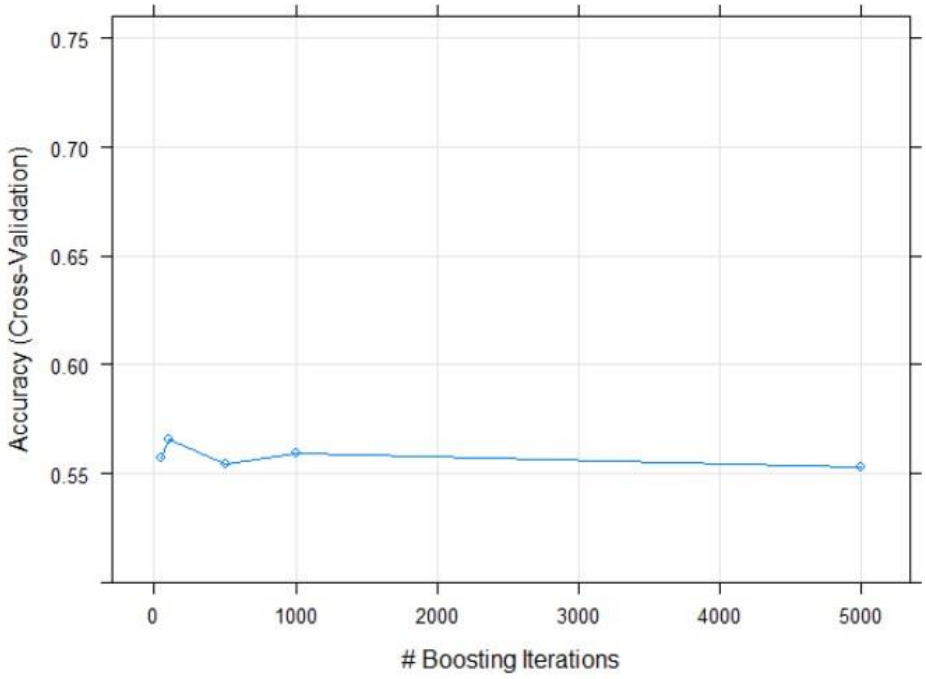
e) Gradient Boosting

Se procederá a tunear los parámetros shrinkage, n.trees y n.minobsinnode. De acuerdo a lo observado se tendrán como parámetros shrinkage 0.03 , n.trees 1000 y n.minobsinnode = 5

shrinkage	n.minobsinnode	n.trees	Accuracy	Kappa
0.001	5	100	0.5245647	0.017296955
0.001	5	500	0.5516265	0.087428799
0.001	5	1000	0.5555058	0.097968907
0.001	5	5000	0.5736005	0.138634723
0.001	10	100	0.5245647	0.017296955
0.001	10	500	0.5516265	0.087797074
0.001	10	1000	0.5542105	0.095421084
0.001	10	5000	0.5645599	0.120790930
0.001	20	100	0.5245647	0.017296955
0.001	20	500	0.5426059	0.073133453
0.001	20	1000	0.5400286	0.067658982
0.001	20	5000	0.5606672	0.113146954
0.010	5	100	0.5542172	0.095446697
0.010	5	500	0.5749025	0.140980787
0.010	5	1000	0.5632178	0.118646249
0.010	5	5000	0.5696878	0.134490378
0.010	10	100	0.5529219	0.092882365
0.010	10	500	0.5632645	0.118095001
0.010	10	1000	0.5645131	0.122759073
0.010	10	5000	0.5490492	0.096692634
0.010	20	100	0.5400286	0.067840252
0.010	20	500	0.5580832	0.107782416
0.010	20	1000	0.5606271	0.114994740
0.010	20	5000	0.5425725	0.080877839
0.030	5	100	0.5645599	0.118802690
0.030	5	500	0.5580298	0.108983326
0.030	5	1000	0.5800705	0.154398039
0.030	5	5000	0.5658485	0.128395375
0.030	10	100	0.5606805	0.110568468
0.030	10	500	0.5658218	0.126595178
0.030	10	1000	0.5723185	0.141043537
0.030	10	5000	0.5477806	0.093727729
0.030	20	100	0.5683991	0.127406784
0.030	20	500	0.5683858	0.131214987



Si se ejecuta el gbm con los parámetros fijos, se observa la siguiente evolución del ACC



Se observa que el ACC tiene su mejor momento con 100 n.trees.

También se puede escoger las variables con mayor importancia. De acuerdo con lo observado estas serían las más importantes:

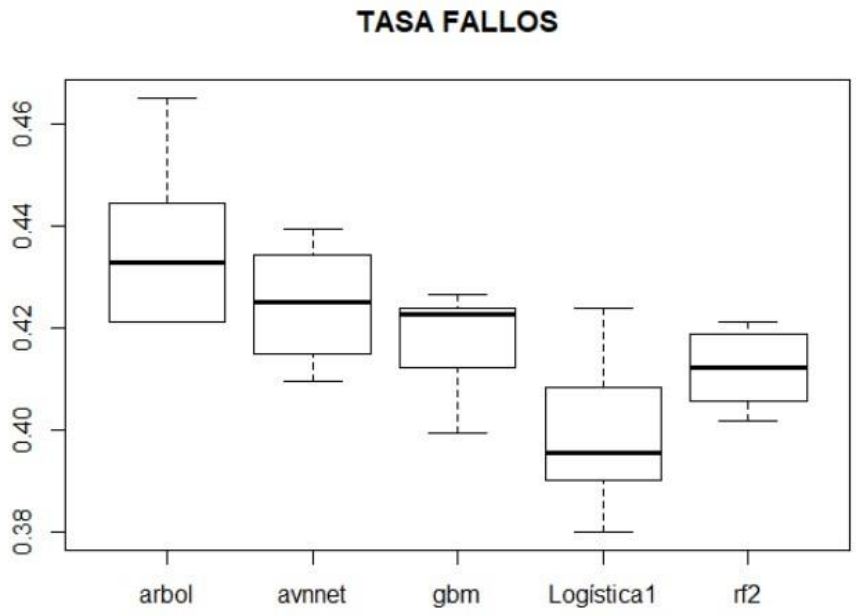
El siguiente grafico se observa una muestra de la importancia de las variables:
AVProductStatesIdentifier,Census_SystemVolumeTotalCapacity,AVProductsInstalled.1,Census_TotalPhysicalRA

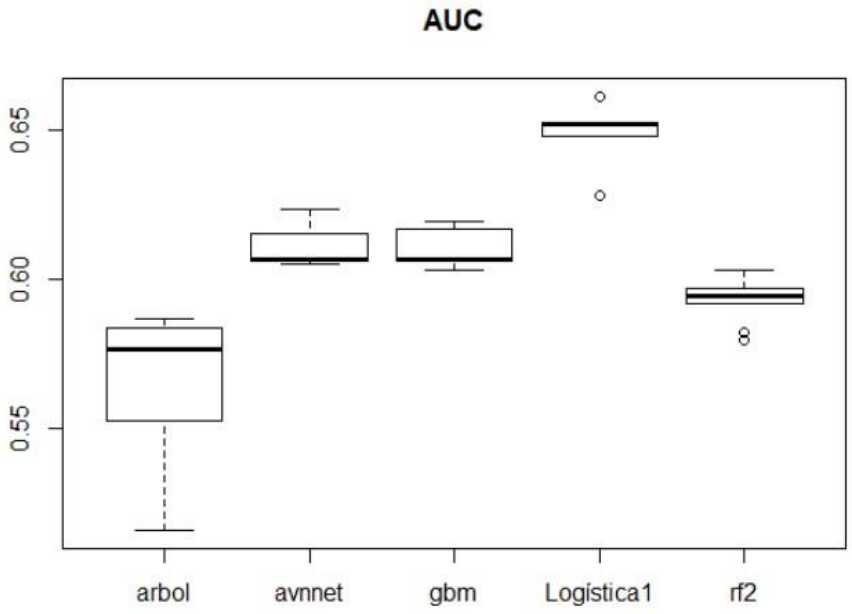
M,Census_FirmwareVersionIdentifier,CityIdentifier,Census_ActivationChannel.Retail,Census_FirmwareManufacturerIdentifier

	rel.inf
AVProductStatesIdentifier	16.0602562
Census_SystemVolumeTotalCapacity	9.8727340
AVProductsInstalled.1	9.5536417
Census_TotalPhysicalRAM	9.2303707
Census_FirmwareVersionIdentifier	8.8239629
CityIdentifier	7.7100275
Census_ActivationChannel.Retail	5.4016271
Census_FirmwareManufacturerIdentifier	4.3110894

Se ha comparado una ejecución de entrenamiento de gbm con las variables escogidas anteriormente y con las variables escogidas por stepwise. Mejores resultados se obtienen con stewise. Con estas variables se trabajar la validación cruzada:

Al ejecutar la validación cruzada, se observa que la regresión logística sigue siendo el mejor modelo:





Codigo R
Input

```
# Gradient Boosting ----
#tuneo de parametros shrinkage y n.trees

gbmgrid<-expand.grid(shrinkage=c(0.1,0.05,0.03,0.01,0.001),
                     n.minobsinnode=c(5,10,20), n.trees=c(100,500,1000,5000),
interaction.depth=c(2)) control<-trainControl(method = "cv",number=4,savePredictions =
"all", classProbs=TRUE) gbm<- train(factor(HasDetections)~.,data=datafile,
method="gbm",          trControl=control,tuneGrid=gbmgrid, distribution="bernoulli",
bag.fraction=1,verbose=FALSE)
gbm   #shrinkage 0.03 y n.trees 1000, n.minobsinnode = 5
plot(gbm)


# ESTUDIO DE EARLY STOPPING
# Probamos a fijar algunos parámetros para ver como evoluciona # en función de las
iteraciones gbmgrid<-expand.grid(shrinkage=c(0.03), n.minobsinnode=c(5),
                     n.trees=c(50,100,500,1000,5000), interaction.depth=c(2))
control<-trainControl(method = "cv",number=4,savePredictions = "all",
classProbs=TRUE) gbm<- train(factor(HasDetections)~.,data=datafile,
method="gbm",trControl=control,          tuneGrid=gbmgrid, distribution="bernoulli",
bag.fraction=1,verbose=FALSE) gbm
plot(gbm,ylim=c(0.50,0.6))

# IMPORTANCIA DE VARIABLES
summary(gbm) tabla<-
summary(gbm) par(cex=1.5,las=2)
```

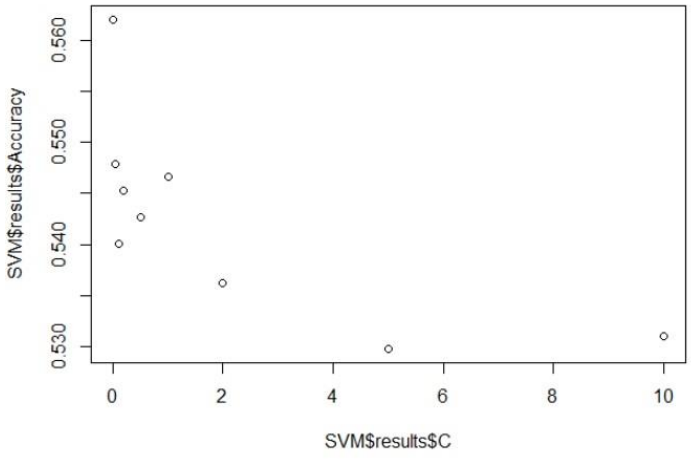
```
barplot(tabla$rel.inf,names.arg=row.names(tabla))
#AVProductStatesIdentifier,Census_SystemVolumeTotalCapacity,AVProductsInstalled.1,Census_TotalPhysicalRAM,Census_FirmwareVersionIdentifier,CityIdentifier,Census_ActivationChannel.Retail,Census_FirmwareManufacturerIdentifier


#se prueba con las variables de gbm

gbmgrid<-expand.grid(shrinkage=c(0.03), n.minobsinnode=c(5),
            n.trees=c(50,100,500,1000,5000), interaction.depth=c(2)) control<-trainControl(method = "cv",number=4,savePredictions = "all", classProbs=TRUE) gbm<-
train(factor(HasDetections)~AVProductStatesIdentifier+
Census_SystemVolumeTotalCapacity+AVProductsInstalled.1+
         Census_TotalPhysicalRAM+Census_FirmwareVersionIdentifier+
         CityIdentifier+Census_ActivationChannel.Retail+Census_FirmwareManufacturerIdentifier,
data=datafile, method="gbm",trControl=control,
       tuneGrid=gbmgrid, distribution="bernoulli", bag.fraction=1,verbose=FALSE) gbm

#se prueba con las variables de stepwise gbmgrid<-expand.grid(shrinkage=c(0.03),
n.minobsinnode=c(5),
            n.trees=c(50,100,500,1000,5000), interaction.depth=c(2))

control<-trainControl(method = "cv",number=4,savePredictions = "all", classProbs=TRUE) gbm2<-
train(factor(HasDetections)~AVProductsInstalled.1+Census_OSArchitecture.amd64+

Census_IsTouchEnabled.0+Census_OSWUAutoUpdateOptionsName.Notify+RtpStateBitfield.0+

Census_IsVirtualDevice.0+Wdft_IsGamer.0+AVProductStatesIdentifier+Census_OSInstallTypeName.Update+

Census_FlightRing.Retail+Census_OEMNameIdentifier+Census_InternalBatteryNumberOfCharges+
Firewall.0+Census_PrimaryDiskTotalCapacity,          data=datafile,
method="gbm",trControl=control,
       tuneGrid=gbmgrid, distribution="bernoulli", bag.fraction=1,verbose=FALSE) gbm2

medias11<-cruzadagbmbin(data=datafile, vardep=vardep,
             listconti=c("AVProductsInstalled.1", "Census_OSArchitecture.amd64",
                   "Census_IsTouchEnabled.0","Census_OSWUAutoUpdateOptionsName.Notify",
                   "RtpStateBitfield.0",
                   "Census_IsVirtualDevice.0", "Wdft_IsGamer.0",
                   "AVProductStatesIdentifier", "Census_OSInstallTypeName.Update",
                   "Census_FlightRing.Retail","Census_OEMNameIdentifier",
                   "Census_InternalBatteryNumberOfCharges", "Firewall.0",
                   "Census_PrimaryDiskTotalCapacity"),
             listclass=c(""), grupos=4,sinicio=1234,repe=5, n.minobsinnode=5,
shrinkage=0.03,n.trees=1000,interaction.depth=2) medias11$modelo="gbm"

union1<-rbind(medias1,medias4, medias7, medias10, medias11)
```

```
#par(cex.axis=0.5)
boxplot(data=union1,tasa~modelo,main="TASA FALLOS", cex=6)
boxplot(data=union1,auc~modelo,main="AUC", cex=1)
```

f) SVM

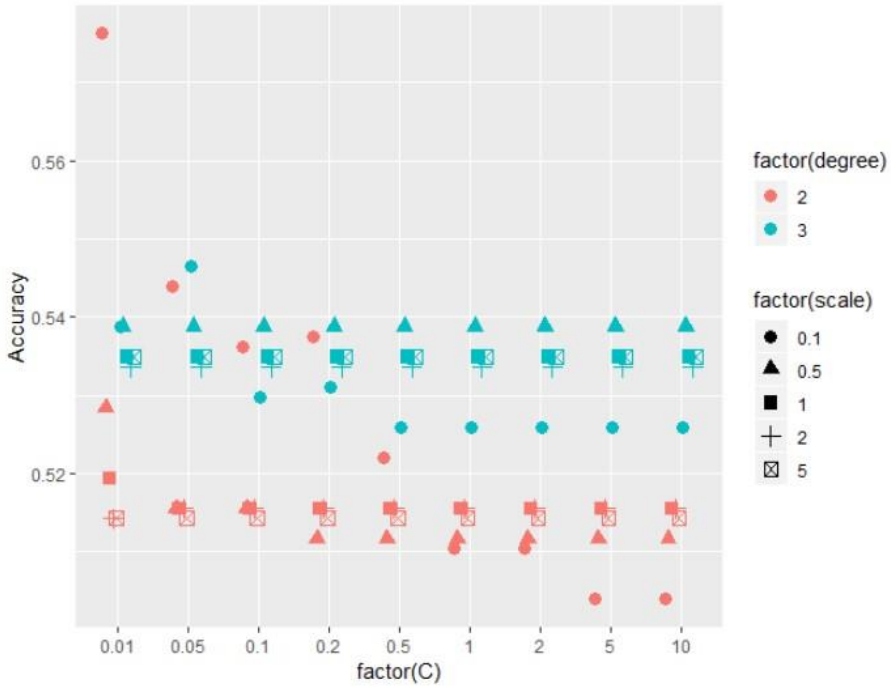
Al igual que en los anteriores modelos, se procedera con el tuneo de los parámetros:

Para SVM Lineal, se obtiene como c=0.01

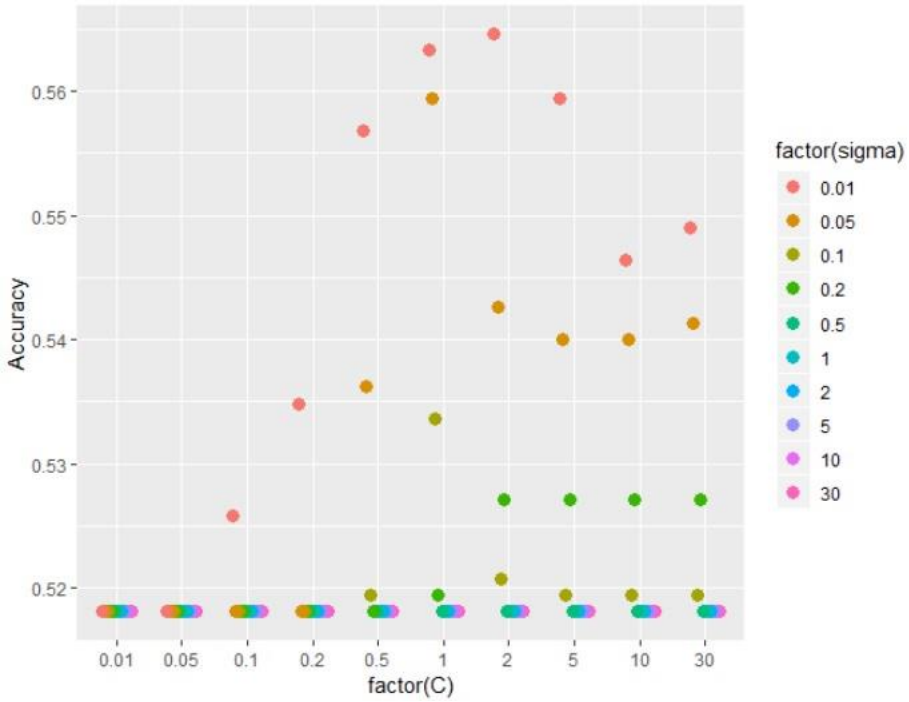



```
> SVM$results
      C  Accuracy      Kappa AccuracySD    KappaSD
1  0.01 0.5620426 0.11504341 0.02648760 0.05433973
2  0.05 0.5478473 0.08969455 0.03613504 0.07402455
3  0.10 0.5401087 0.07498858 0.04195378 0.08571874
4  0.20 0.5452633 0.08547554 0.03207937 0.06489577
5  0.50 0.5426793 0.07995054 0.03302072 0.06673649
6  1.00 0.5465720 0.08856545 0.03075034 0.06264583
7  2.00 0.5362160 0.06796235 0.02130605 0.04332051
8  5.00 0.5297527 0.05481549 0.02243295 0.04542074
9 10.00 0.5310480 0.05717160 0.02681596 0.05439224
```

Para SVM polinomial, se obtiene degree = 2, scale = 0.1 and C = 0.01

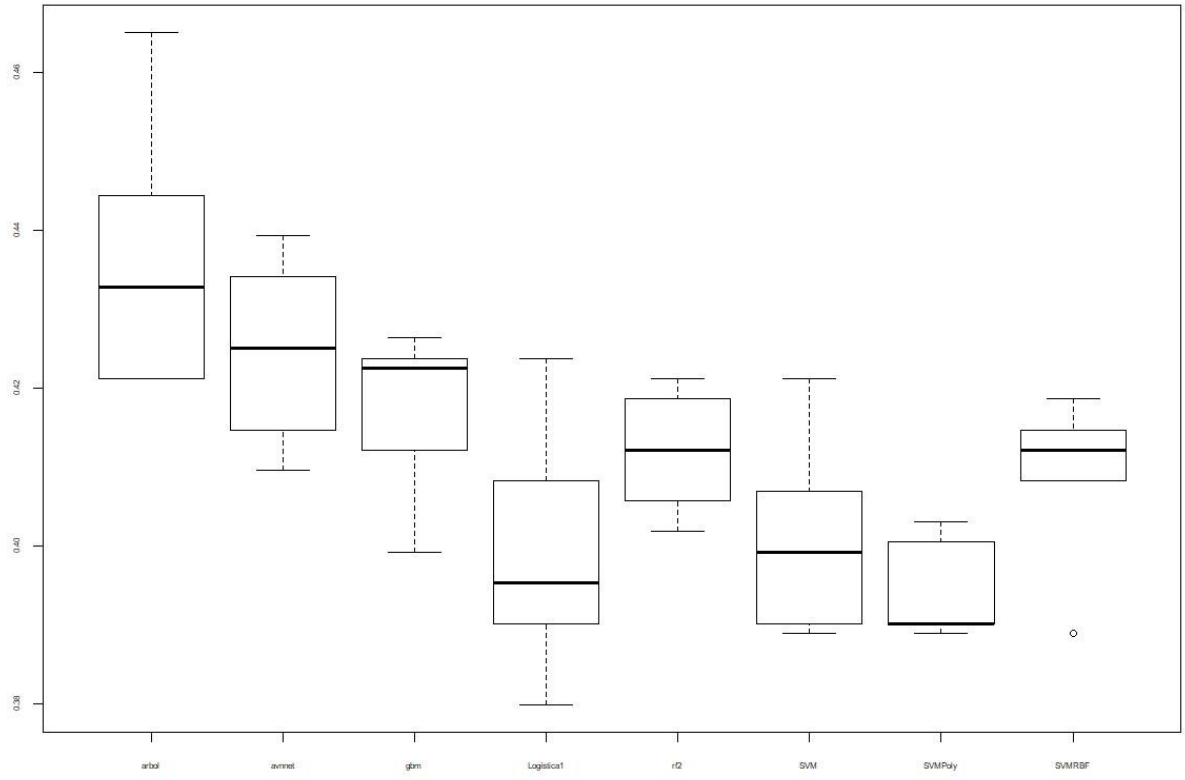


Para SVM Radial se obtiene sigma = 0.01 and C = 2

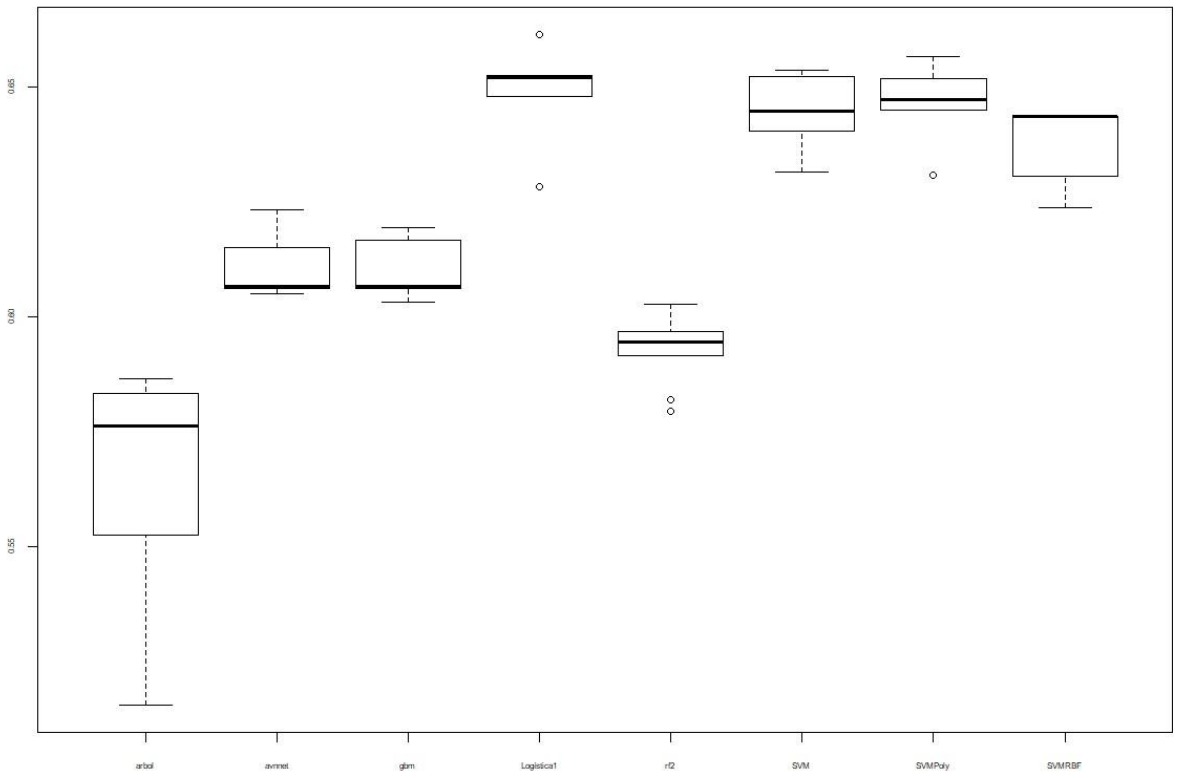


Al aplicar la validación cruzada para los tres tipos de SVM se obtiene que la logística sigue superando, sin embargo, SVM Polinomial se acerca bastante:

TASA FALLOS



AUC



Codigo R

Input

```
#tuneado con caret

SVMgrid<-expand.grid(C=c(0.01,0.05,0.1,0.2,0.5,1,2,5,10))
control<-trainControl(method = "cv",number=4,savePredictions = "all") SVM<-
train(data=datafile,factor(HasDetections)~.,
      method="svmLinear",trControl=control,   tuneGrid=SVMgrid,verbose=FALSE)     SVM
#c=0.01 SVM$results
plot(SVM$results$C,SVM$results$Accuracy)



SVMgrid<-expand.grid(C=c(0.01,0.05,0.1,0.2,0.5,1,2,5,10),
degree=c(2,3),scale=c(0.1,0.5,1,2,5))
control<-trainControl(method = "cv", number=4,savePredictions = "all") SVM<-
train(data=datafile,factor(HasDetections)~.,
      method="svmPoly",trControl=control, tuneGrid=SVMgrid,verbose=FALSE) SVM #
degree = 2, scale = 0.1 and C = 0.01

SVM$results
dat<-as.data.frame(SVM$results)

# PLOT DE DOS VARIABLES CATEGÓRICAS, UNA CONTINUA
ggplot(dat, aes(x=factor(C), y=Accuracy, color=factor(degree),pch=factor(scale))) +
geom_point(position=position_dodge(width=0.5),size=3)



SVMgrid<-expand.grid(C=c(0.01,0.05,0.1,0.2,0.5,1,2,5,10,30),
            sigma=c(0.01,0.05,0.1,0.2,0.5,1,2,5,10,30))
control<-trainControl(method = "cv", number=4,savePredictions = "all") SVM<-
train(data=datafile,factor(HasDetections)~.,
      method="svmRadial",trControl=control, tuneGrid=SVMgrid,verbose=FALSE) SVM
#sigma = 0.01 and C = 2

dat<-as.data.frame(SVM$results)
ggplot(dat, aes(x=factor(C), y=Accuracy, color=factor(sigma))) +
geom_point(position=position_dodge(width=0.5),size=3)



medias13<-cruzadaSVMbin(data=datafile, vardep=vardep,
 listconti=c("AVProductsInstalled.1", "Census_OSArchitecture.amd64",
      "Census_IsTouchEnabled.0","Census_OSWUAutoUpdateOptionsName.Notify",
      "RtpStateBitfield.0",
      "Census_IsVirtualDevice.0", "Wdft_IsGamer.0",
      "AVProductStatesIdentifier", "Census_OSInstallTypeName.Update",
      "Census_FlightRing.Retail","Census_OEMNameIdentifier",
      "Census_InternalBatteryNumberOfCharges", "Firewall.0",
      "Census_PrimaryDiskTotalCapacity"),   listclass=c(""),
```

```
  grupos=4,sinicio=1234,repe=5,
  C=0.01)

 medias13$modelo="SVM"
#
# # degree = 2, scale = 0.1 and C = 0.01
medias14<-cruzadaSVMbinPoly(data=datafile, vardep=vardep,
   listconti=c("AVProductsInstalled.1", "Census_OSArchitecture.amd64",
          "Census_IsTouchEnabled.0","Census_OSWUAutoUpdateOptionsName.Notify",
          "RtpStateBitfield.0",
          "Census_IsVirtualDevice.0", "Wdft_IsGamer.0",
          "AVProductStatesIdentifier", "Census_OSInstallTypeName.Update",
          "Census_FlightRing.Retail","Census_OEMNameIdentifier",
          "Census_InternalBatteryNumberOfCharges", "Firewall.0",
          "Census_PrimaryDiskTotalCapacity"),
listclass=c(""),
  grupos=4,sinicio=1234,repe=5,
  C=0.01,degree=2,scale=0.1)

 medias14$modelo="SVMPoly"

 source("cruzadaSVMbinRBF.R")
#para radial sigma = 0.01 and C = 2
 medias15<-cruzadaSVMbinRBF(data=datafile, vardep=vardep,
                  listconti=c("AVProductsInstalled.1", "Census_OSArchitecture.amd64",

"Census_IsTouchEnabled.0","Census_OSWUAutoUpdateOptionsName.Notify",
                           "RtpStateBitfield.0",
                           "Census_IsVirtualDevice.0", "Wdft_IsGamer.0",
                           "AVProductStatesIdentifier", "Census_OSInstallTypeName.Update",
                           "Census_FlightRing.Retail","Census_OEMNameIdentifier",
                           "Census_InternalBatteryNumberOfCharges", "Firewall.0",
                           "Census_PrimaryDiskTotalCapacity"),
                  listclass=c(""), grupos=4,sinicio=1234,repe=5, C=2,sigma=0.01)
medias15$modelo="SVMRBF"


 union1<-rbind(medias1,medias4, medias7, medias10,
medias11,medias13,medias14,medias15)
 #par(cex.axis=0.5)
 boxplot(data=union1,tasa~modelo,main="TASA FALLOS", cex=1)
boxplot(data=union1,auc~modelo,main="AUC", cex=1)
```

g) Ensamblado

Por último, se va a validar el método ensamblado para validar si se mejora la predicción de los modelos anteriores combinándolos. Los parámetros que se toman para cada modelo son los que han sido previamente tuneados. Cada una de las funciones de validación cruzada luego del train tienen unos comandos adicionales que servirán para validar los ensamblados, este código se ejecuta luego de cada train en las validaciones cruzadas:
Por ejemplo en la cruzada SVMbinPoly, se observa la sección de código Para ensamblado y train, lo que se agrega luego del train son los comandos necesarios para obtener las predicciones que se usaran en los ensamblados:

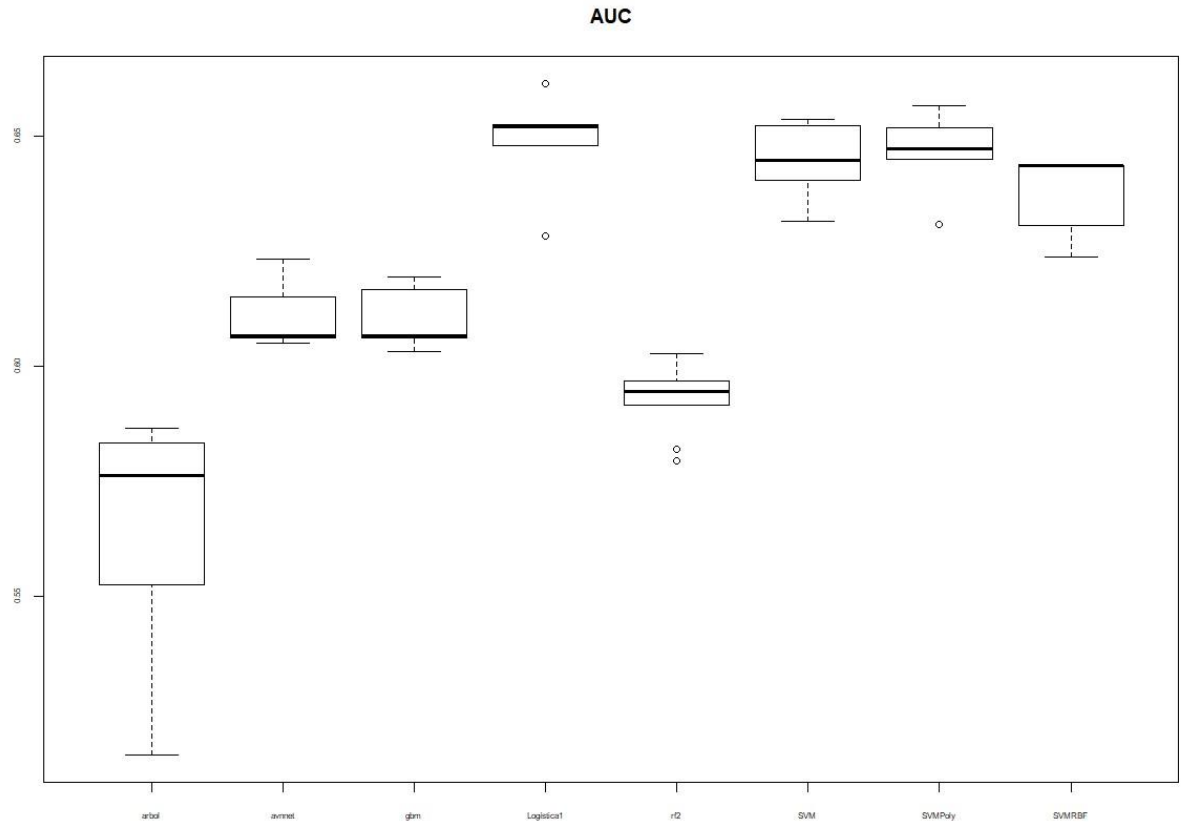
```
cruzadaSVMbinPoly<-
  function(data=data,vardep="vardep",
           listconti="listconti",listclass="listclass",
           grupos=4,sinicio=1234,repe=5,
           C=1,degree=2,scale=1)
  {

    # Preparación del archivo
    # Para ensamblados y train -----
    SVM<- train(formu,data=databis,
                method="svmPoly",trControl=control,
                tuneGrid=SVMgrid,replace=replace)


    preditest<-SVM$pred

    preditest$prueba<-strsplit(preditest$Resample,"[.]")
    preditest$Fold <- sapply(preditest$prueba, "[", 1)
    preditest$Rep <- sapply(preditest$prueba, "[", 2)
    preditest$prueba<-NULL

    tasafallos<-function(x,y) {
      confu<-confusionMatrix(x,y)
      tasa<-confu[[3]][1]
      return(tasa)
    }
    # Aplicamos función sobre cada Repetición
      medias<-preditest %>%
      group_by(Rep) %>%
      summarize(tasa=1-tasafallos(pred,obs))

    # CalculamoS AUC  por cada Repetición de cv
    # Definimnos función

    auc<-function(x,y) {
      curvaroc<-roc(response=x,predictor=y)
      auc<-curvaroc$auc
      return(auc)
    }
    # Aplicamos función sobre cada Repetición

    mediasbis<-preditest %>%
      group_by(Rep) %>%
      summarize(auc=auc(obs,Yes))

    # Unimos la info de auc y de tasafallos

    medias$auc<-mediasbis$auc

    return(medias)
```

Luego de cada llamada a las funciones de validación cruzada, se mapean los valores generados de predicciones

Luego de la llamada a cada función se han mapeado los vectores de predicción, esto se ha realizado sobre los modelos con los valores tuneados: Por ejemplo se tiene el siguiente código luego de cada llamada a las validaciones cruzadas:

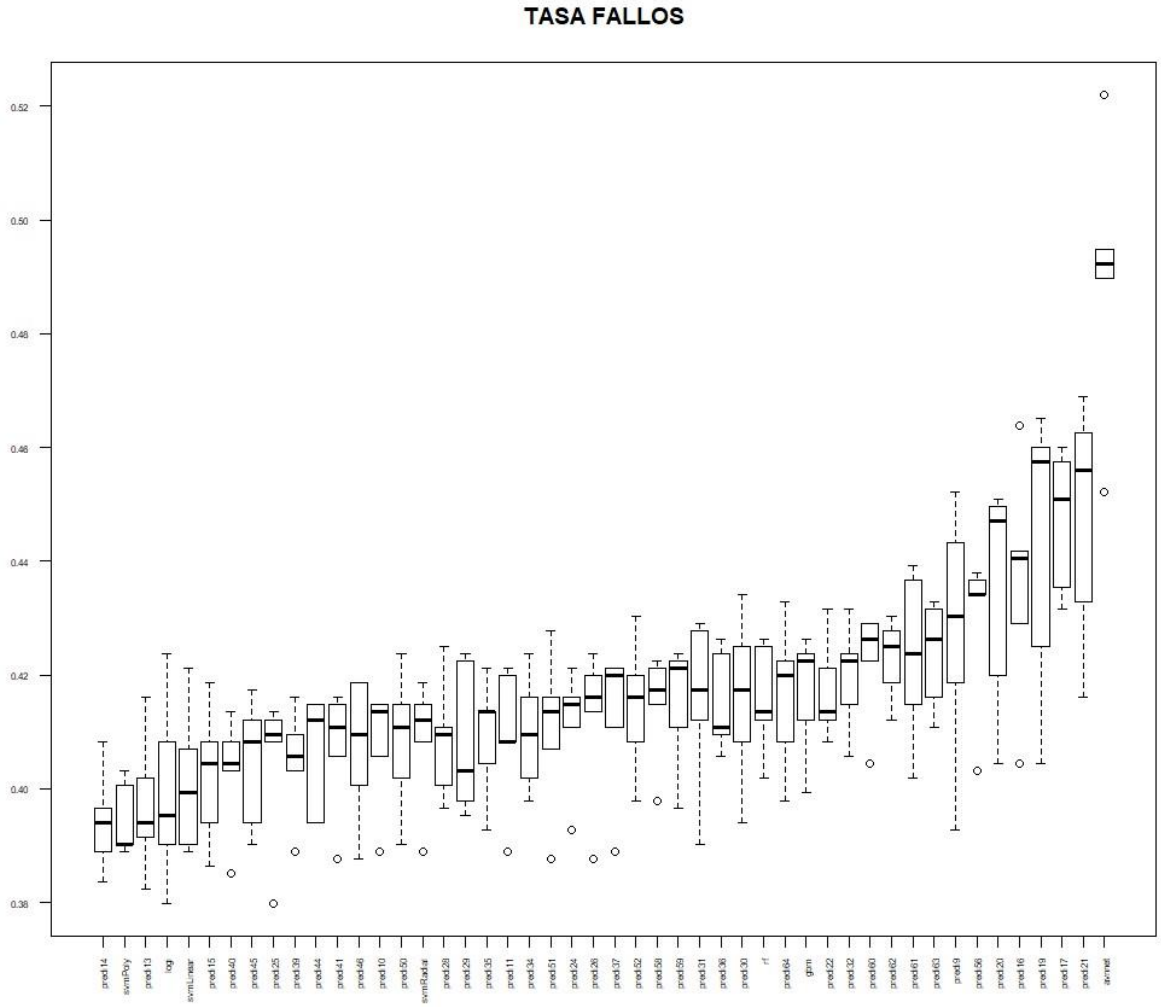
```
medias1bis<-as.data.frame(medias1[1])
medias1bis$modelo<-"Logistica" predi1<-
as.data.frame(medias1[2]) predi1$logi<-predi1$Yes
```

En el resumen general se observa que el modelo que mejor predecía es la regresión logisitica:



Se va a validar a continuación si el ensamblado mejora esta predicción o la estabiliza.

Luego de calcular la media de la tasa del error se obtiene lo siguiente:



Esta información se confirma con la tabla de medias de la tasa de error, se observa que los modelos predi14 (Logística + SVM Polinomial), SVMPoly (SVM Polinomial) y predi13(Logística + SVM Lineal) también mejoran la predicción.

	modelo	tasa
	<fct>	<dbl>
1	predi14	0.394
2	svmPoly	0.395
3	predi13	0.397
4	logi	0.399
5	svmLinear	0.401
6	predi15	0.402
7	predi40	0.403
8	predi45	0.404
9	predi25	0.405
10	predi39	0.405

Podría valer quedarse únicamente con el modelo logístico, dado que los otros modelos ensamblado mejoran de manera mínima el modelo.

Codigo R

Input

```
# CONSTRUCCIÓN DE TODOS LOS ENSAMBLADOS
# SE UTILIZARÁN LOS ARCHIVOS SURGIDOS DE LAS FUNCIONES LLAMADOS predi1,...

unipredi<-cbind(predi1,predi2,predi3,predi4,predi6,predi7,predi8)

# Esto es para eliminar columnas duplicadas
unipredi<- unipredi[, !duplicated(colnames(unipredi))]

# Construccion de ensamblados, cambiar al gusto

unipredi$predi9<-(unipredi$logi+unipredi$avnnet)/2 unipredi$predi10<-
(unipredi$logi+unipredi$rf)/2 unipredi$predi11<-(unipredi$logi+unipredi$gbm)/2
#unipredi$predi12<-(unipredi$logi+unipredi$xgbm)/2 unipredi$predi13<-
(unipredi$logi+unipredi$svmLinear)/2 unipredi$predi14<-(unipredi$logi+unipredi$svmPoly)/2
unipredi$predi15<-(unipredi$logi+unipredi$svmRadial)/2 unipredi$predi16<-
(unipredi$avnnet+unipredi$rf)/2 unipredi$predi17<-(unipredi$avnnet+unipredi$gbm)/2
#unipredi$predi18<-(unipredi$avnnet+unipredi$xgbm)/2 unipredi$predi19<-
(unipredi$avnnet+unipredi$svmLinear)/2 unipredi$predi20<-
(unipredi$avnnet+unipredi$svmPoly)/2 unipredi$predi21<-
(unipredi$avnnet+unipredi$svmRadial)/2 unipredi$predi22<-(unipredi$rf+unipredi$gbm)/2
#unipredi$predi23<-(unipredi$rf+unipredi$xgbm)/2 unipredi$predi24<-
(unipredi$rf+unipredi$svmLinear)/2 unipredi$predi25<-(unipredi$rf+unipredi$svmPoly)/2
unipredi$predi26<-(unipredi$rf+unipredi$svmRadial)/2 #unipredi$predi27<-
(unipredi$gbm+unipredi$xgbm)/2 unipredi$predi28<-(unipredi$gbm+unipredi$svmLinear)/2
unipredi$predi29<-(unipredi$gbm+unipredi$svmPoly)/2 unipredi$predi30<-
(unipredi$gbm+unipredi$svmRadial)/2

unipredi$predi31<-(unipredi$logi+unipredi$avnnet+unipredi$rf)/3 unipredi$predi32<-
(unipredi$logi+unipredi$avnnet+unipredi$gbm)/3 #unipredi$predi33<-
(unipredi$logi+unipredi$avnnet+unipredi$xgbm)/3 unipredi$predi34<-
(unipredi$logi+unipredi$avnnet+unipredi$svmLinear)/3 unipredi$predi35<-
(unipredi$logi+unipredi$avnnet+unipredi$svmPoly)/3 unipredi$predi36<-
(unipredi$logi+unipredi$avnnet+unipredi$svmRadial)/3 unipredi$predi37<-
(unipredi$logi+unipredi$rf+unipredi$gbm)/3 #unipredi$predi38<-
(unipredi$logi+unipredi$rf+unipredi$xgbm)/3 unipredi$predi39<-
(unipredi$logi+unipredi$rf+unipredi$svmLinear)/3
```

```
unipredi$predi40<-(unipredi$logi+unipredi$rf+unipredi$svmPoly)/3 unipredi$predi41<-
(unipredi$logi+unipredi$rf+unipredi$svmRadial)/3 #unipredi$predi42<-
(unipredi$logi+unipredi$gbm+unipredi$xgbm)/3 #unipredi$predi43<-
(unipredi$logi+unipredi$gbm+unipredi$xgbm)/3 unipredi$predi44<-
(unipredi$logi+unipredi$gbm+unipredi$svmLinear)/3 unipredi$predi45<-
(unipredi$logi+unipredi$gbm+unipredi$svmPoly)/3 unipredi$predi46<-
(unipredi$logi+unipredi$gbm+unipredi$svmRadial)/3 #unipredi$predi47<-
(unipredi$logi+unipredi$xgbm+unipredi$svmLinear)/3
#unipredi$predi48<-(unipredi$logi+unipredi$xgbm+unipredi$svmPoly)/3
#unipredi$predi49<-(unipredi$logi+unipredi$xgbm+unipredi$svmRadial)/3

unipredi$predi50<-(unipredi$rf+unipredi$gbm+unipredi$svmLinear)/3 unipredi$predi51<-
(unipredi$rf+unipredi$gbm+unipredi$svmPoly)/3 unipredi$predi52<-
(unipredi$rf+unipredi$gbm+unipredi$svmRadial)/3

#unipredi$predi53<-(unipredi$rf+unipredi$xgbm+unipredi$svmLinear)/3
#unipredi$predi54<-(unipredi$rf+unipredi$xgbm+unipredi$svmPoly)/3
#unipredi$predi55<-(unipredi$rf+unipredi$xgbm+unipredi$svmRadial)/3

unipredi$predi56<-(unipredi$rf+unipredi$avnnet+unipredi$gbm)/3 #unipredi$predi57<-
(unipredi$rf+unipredi$avnnet+unipredi$xgbm)/3 unipredi$predi58<-
(unipredi$rf+unipredi$avnnet+unipredi$svmLinear)/3 unipredi$predi59<-
(unipredi$rf+unipredi$avnnet+unipredi$svmPoly)/3 unipredi$predi60<-
(unipredi$rf+unipredi$avnnet+unipredi$svmRadial)/3

unipredi$predi61<-(unipredi$avnnet+unipredi$gbm+unipredi$svmLinear)/3
unipredi$predi62<-(unipredi$avnnet+unipredi$gbm+unipredi$svmPoly)/3 unipredi$predi63<-
(unipredi$avnnet+unipredi$gbm+unipredi$svmRadial)/3

unipredi$predi64<-(unipredi$logi+unipredi$rf+unipredi$gbm+unipredi$avnnet)/4
#unipredi$predi65<-(unipredi$logi+unipredi$rf+unipredi$xgbm+unipredi$avnnet)/4
#unipredi$predi66<-(unipredi$logi+unipredi$rf+unipredi$xgbm+unipredi$avnnet)/4

#unipredi$predi67<-
(unipredi$logi+unipredi$rf+unipredi$xgbm+unipredi$avnnet+unipredi$svmLinear)/5
#unipredi$predi68<-
(unipredi$logi+unipredi$rf+unipredi$xgbm+unipredi$avnnet+unipredi$svmPoly)/5
#unipredi$predi69<-
(unipredi$logi+unipredi$rf+unipredi$xgbm+unipredi$avnnet+unipredi$svmRadial)/5



# Listado de modelos a considerar, cambiar al gusto

dput(names(unipredi))

listado<-c( "logi", "avnnet", "rf",
"gbm", "svmLinear", "svmPoly",
        "svmRadial", "predi9", "predi10", "predi11", "predi13", "predi14",
        "predi15", "predi16", "predi17", "predi19", "predi20", "predi21",
        "predi22", "predi24", "predi25", "predi26", "predi28", "predi29",
```

"predi30", "predi31", "predi32", "predi34", "predi35", "predi36",

```
          "predi37", "predi39", "predi40", "predi41", "predi44", "predi45",
          "predi46", "predi50", "predi51", "predi52", "predi56", "predi58",
"predi59", "predi60", "predi61", "predi62", "predi63", "predi64"
)

# Cambio a Yes, No, todas las predicciones

for (prediccion in listado)
{
 unipredi[,prediccion]<-ifelse(unipredi[,prediccion]>0.5,"Yes","No")
}

# Defino funcion tasafallos

tasafallos<-function(x,y) {   confu<-
confusionMatrix(x,y)
 tasa<-confu[[3]][1]
return(tasa)
}

auc<-function(x,y) {
 curvaroc<-roc(response=x,predictor=y)   auc<-
curvaroc$auc
 return(auc)
}

# Se obtiene el número de repeticiones CV y se calculan las medias por
repe en # el data frame medias0

repeticiones<-nlevels(factor(unipredi$Rep)) unipredi$Rep<-
as.factor(unipredi$Rep) unipredi$Rep<-as.numeric(unipredi$Rep)


# Se obtiene el número de repeticiones CV y se calculan las medias por repe en # el data
frame medias0

repeticiones<-nlevels(factor(unipredi$Rep))  unipredi$Rep<-
as.factor(unipredi$Rep)  unipredi$Rep<-
as.numeric(unipredi$Rep) medias0<-data.frame(c())
for (prediccion in listado)
{
 for (repe in 1:repeticiones)
 {
   paso <- unipredi[(unipredi$Rep==repe),]
pre<-factor(paso[,prediccion])     obs<-
factor(paso[,c("obs")])     tasa=1-
tasafallos(pre,obs)     t<-as.data.frame(tasa)
t$modelo<-prediccion
```

```
   medias0<-rbind(medias0,t)
  }
}


#ORDENACIÓN POR LA MEDIA
medias0$modelo <- with(medias0, reorder(modelo,tasa, mean))
par(cex.axis=0.5,las=2)
boxplot(data=medias0,tasa~modelo,main="TASA FALLOS")


tablamedias2<-medias0 %>%
group_by(modelo) %>%
  summarize(tasa=mean(tasa))
```