# Machine Learning

## Introducción al problema

Se ha estudiado la concesión de hipotecas entre 1997 y 1998 en Boston de acuerdo a una serie de variables. Los datos provienen de la reserva federal de Boston, y pueden ser encontrados en <https://vincentarelbundock.github.io/Rdatasets/doc/Ecdat/Hdma.html>

La distribución del código está en el anexo

La variable objetivo es “deny”: es una variable categórica binaria, Si el valor es “Yes” se rechaza la hipoteca, en cambio, cuando el valor es “no” se concede la hipoteca. EL objetivo del problema es construir un modelo clasificador binario de la variable deny en función de unas variables independientes.

Hay 12 variables independientes, de ellas 6 son continuas y 6 son categóricas.

Las variables continuas son las siguientes:

* Dir: el ratio entre deuda y el salario
* hir: ratio de los gastos de la casa y el salario
* lvr: ratio entre tamaño de la deuda y el valor de la propiedad
* ccs: puntuación de credito del consumidor del 1 al 6 (cuanto menor sea la puntuación mejor puntuación es)
* mcs: puntuación de crédito de la hipoteca del 1 al 4 (cuanto menor sea la puntuación mejor puntuación es)
* uria: la tasa de desempleo en el sector del solicitante en Massachusetts de 1989

Las variables categóricas son las siguientes:

* Pbcr: ¿El solicitante tiene malos registros crediticios públicos? Las categorías de esta variable son “Yes” o “no”.
* dmi: ¿Se le ha denegado el seguro de hipoteca al solicitante? Las categorías de esta variable son “Yes” o “no”.
* self: ¿Es el solicitante autónomo? Las categorías de esta variable son “Yes” o “no”.
* single: ¿Es el solicitante soltero? Las categorías de esta variable son “Yes” o “no”.
* condominium: ¿La casa es un condominio? Las categorías de esta variable son “Yes” o “no”.
* black ¿El solicitante es negro? Las categorías de esta variable son “Yes” o “no”.

Los datos estaban muy limpios, sólo una observación de las 2381 con dos missings( ha sido eliminada por simplicidad), tampoco se han observado valores erróneos o mal puestos.

## Desbalance de categorías

Las categorías de la variable target están desbalanceadas

# no yes

# 2096 285

Esto, unido a que los algoritmos tienden a entrenar los modelos hacia una tasa de aciertos máxima provoca que intenten maximizar la categoría dominante a costa de la minoritaria.

En este problema esto es especialmente relevante porque un falso negativo significa que estamos concediendo una hipoteca a alguien que probablemente no llegue a pagarla, incurriendo en coste para la entidad.

Las métricas que vamos a utilizar son accuracy, sensitivity y AUC. Usamos accuracy y AUC porque miden de manera distinta como de bien el modelo clasifica. En un dataset desbalanceado los modelos tienen sesgo a favor de la clase dominante, en este caso los “No” (aceptación de hipotecas). Así, sensivity suele ser bastante baja porque gran parte de los casos se predicen como falsos negativos. Como se ha comentado arriba, hay que evitar los falsos negativos lo máximo posible, por lo tanto la estrategia será conseguir modelos con valores de sensitivity lo más altos posibles, mantiendo a la vez, accuracy y AUC con valores lo más altos posibles.

Para solventar el desbalanceo de las clases se va a utilizar un método de generación de datos sintéticos. Esta técnica genera observaciones de la clase minoritaria usando la distribución de esta. Vamos a usar el paquete ROSE.

En las fases siguientes será necesario usar validación cruzada. Sin embargo, no podemos usar simplemente los datos originales. Así que he construido funciones de K-validación cruzada que separan el dataset en k grupos, de estos k-1 grupos de entrenamiento son transformados con el paquete ROSE. El testeo de las predicción se hace con datos sin transformar. Este proceso se repite k-1 veces.

He construido una función para diferentes algoritmos que implantan esta metodología:

* RangerDesbalance: usa una librería llamada ranger para hacer random forest.
* GlmDesbalance: regresión logística.
* RedesDesbalance: se apoya en avnnet.

Estas funciones se han usado en las fases de selección de variables y tuneado de parámetros. Posteriormente aprendí a usar el paquete ROSE junto con los métodos asociados al paquete CARET, usando estos métodos en la comparación de modelos y secciones posteriores.

Así, se ha utilizado las funciones que se utilizaron en clase para los diferentes algoritmos, con pequeñas modificaciones: que muestre también la métrica sensitivity y la instrucción pertinente para que CARET usara el paquete ROSE.

## Selección de variables

En la selección de variables he utilizado dos aproximaciones: lineal y no-lineal. En cada una de las dos aproximaciones obtendremos conjuntos de variables tentativos por distintos métodos y después los compararemos con validación cruzada. Las variables finales se obtendrán de comparar los mejores conjuntos de variables de cada una de las dos aproximaciones usando validación cruzada.

### Aproximación lineal

El método stepwise se utiliza con remuestreo en los datos con ajustes de bondad AIC y BIC. Se realiza en cada uno el remuestreo 10 veces (cada una de las veces con distinta semilla) y con un porcentaje de datos de 0,8. Además modifica levemente la función steprepetidobinaria para que los datos training se balanceen con el paquete ROSE.

Con el ajuste de bondad AIC estos son los conjuntos de variables más frecuentes:

#c("ccs", "dir", "lvr", "mcs", "uria")

#c("ccs", "dir", "lvr", "mcs", "uria","hir")

Con el ajuste de bondad AIC estos son los conjuntos de variables más frecuentes:

#c("ccs", "dir", "lvr", "mcs", "uria")

#c("ccs", "dir", "lvr", "mcs", "uria","hir")

Con el ajuste de bondad BIC este es el conjunto de variables más frecuentes:

#c("ccs", "lvr", "dir", "mcs", "uria")

Este conjunto de variable tiene las mismas variables que el primero con AIC.

Además, también se ha utilizado el método stepwise con todos las muestras. El conjunto de variable obtenido es

c( "dmi", "ccs", "dir", "pbcr", "black", "lvr", "self", "single", "uria", "mcs")

Comparamos estos tres conjuntos de variables mediante la función GlmDesbalance, como se ha comentado en la sección anterior, esta función realiza validación cruzada usando en el training datos balanceados con la librería ROSE.

|  |  |  |  |
| --- | --- | --- | --- |
| Modelo | Conjunto variables | Método | Numero variables |
| Modelo 1 | c("ccs", "dir", "lvr", "mcs", "uria") | AIC y BIC con remuestreo | 5 |
| Modelo 2 | c("ccs", "dir", "lvr", "mcs", "uria","hir") | AIC con remuestreo | 6 |
| Modelo 3 | c( "dmi", "ccs", "dir", "pbcr", "black", "lvr","self", "single", "uria", "mcs") | #AIC entero | 10 |

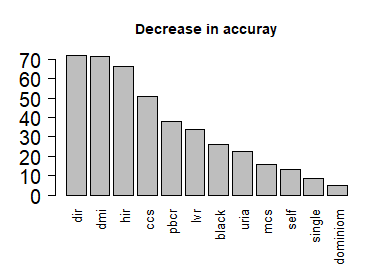
|  |  |
| --- | --- |
| D:\Master_UCM\Machine Learning\Boston\Figuras\V06_seleccion_variables_lineal_Acc.png | D:\Master_UCM\Machine Learning\Boston\Figuras\V06_seleccion_variables_lineal_Sens.png |
| D:\Master_UCM\Machine Learning\Boston\Figuras\V06_seleccion_variables_lineal_ROC.png |  |

En primer lugar descartamos el modelo 3, el aumento en la sensitivity de alrededor de 0.03 que obtenemos con este conjunto de variables no justifica añadir tantas variables al modelo.

Elijo el modelo 1 frente al modelo 2, porque la variable extra no aporta casi nada.

### Aproximación no-lineal

En primer lugar se ha utilizado un random forest para saber la importancia de las variables sobre el dataset balanceado con la librería ROSE.



Diferentes conjuntos de variables se comparan mediante validación cruzada con la función RangerDesbalance. La aproximación será iterativa, se irán añadiendo variables poco a poco, comparando modelos.

Se empezará con un modelo con las tres variables más importantes c("dir", "dmi", "hir"). Además, se ha incluido un modelo con todas las posibles variables para hacernos una idea de cuánto puede mejorar con todas las variables

|  |  |  |
| --- | --- | --- |
| Modelo | Conjunto variables | Numero variables |
| Modelo 1 | c("dir", "dmi", "hir", "ccs", "pbcr", "lvr", "black", "uria",  "mcs", "self", "single", "comdominiom") | 12 |
| Modelo 2 | c("dir", "dmi", "hir") | 3 |
| Modelo 3 | c("dir", "dmi", "hir","ccs") | 4 |
| Modelo 4 | c("dir", "dmi", "hir","pbcr" ) | 4 |
| Modelo 5 | c("dir", "dmi", "hir", "lvr") | 4 |
| Modelo 6 | c("dir", "dmi", "hir","black") | 4 |
| Modelo 7 | c("dir", "dmi", "hir","uria" ) |  |
| Modelo 8 | c("dir", "dmi", "hir","mcs") | 4 |
| Modelo 9 | c("dir", "dmi", "hir","self") | 4 |
| Modelo 10 | c("dir", "dmi", "hir","single" ) | 4 |
| Modelo 11 | c("dir", "dmi", "hir","comdominiom") | 4 |

Desde los modelos 3 al 11 son iguales al modelo 2 salvo que se ha añadido una variable. Entonces, el modelo 2 se usará como referencia para comparar.

|  |  |
| --- | --- |
| C:\Users\Adrián\AppData\Local\Microsoft\Windows\INetCache\Content.Word\V06_seleccion_variables_Nolineal_Rf_Accu(1-11).png | D:\Master_UCM\Machine Learning\Boston\Figuras\V06_seleccion_variables_Nolineal_Rf_Sens(1-11).png |
| C:\Users\Adrián\AppData\Local\Microsoft\Windows\INetCache\Content.Word\V06_seleccion_variables_Nolineal_Rf_ROC(1-11).png |  |

-

Los modelos 3,4,5,6 producen mejoras en las 3 métricas. Especialmente destaca el modelo 3, que es el que más mejora y además reduce la varianza en las métricas, la variable asociada a este modelo es “css”. Además, como las variables asociadas a los modelos del 7 al 11 no dan resultados claros las descartamos. El modelo 1, que contiene todas las variables, tiene muy más sensitivity respecto al 2, parece aconsejable seguir añadiendo variables.

En la siguiente ronda de comparación se añaden dos variables al modelo con respecto al modelo2: combinaciones de a dos de las variables asociadas a los modelos 3,4,5,6; que corresponden a “css”, “pbcr", "lvr", “black” respectivamente.

Ahora el modelo de referencia será el modelo 11, que está formado por las mismas variables que el anterior modelo 3. Así podremos decir sin compensa seguir añadiendo variables o, si por el contrario, es mejor tener un modelo más parsimonioso.

|  |  |  |
| --- | --- | --- |
| Modelo | Conjunto variables | Numero variables |
| Modelo 11 | c("dir", "dmi", "hir","ccs") | 4 |
| Modelo 12 | c("dir", "dmi", "hir","ccs","pbcr") | 5 |
| Modelo 13 | c("dir", "dmi", "hir","ccs","lvr") | 5 |
| Modelo 14 | c("dir", "dmi", "hir","ccs","black") | 5 |
| Modelo 15 | c("dir", "dmi", "hir","pbcr","lvr" ) | 5 |
| Modelo 16 | c("dir", "dmi", "hir","pbcr","black" ) | 5 |
| Modelo 17 | c("dir", "dmi", "hir", "lvr","black") | 5 |

|  |  |
| --- | --- |
| D:\Master_UCM\Machine Learning\Boston\Figuras\V06_seleccion_variables_Nolineal_Rf_AC(11-17).png | D:\Master_UCM\Machine Learning\Boston\Figuras\V06_seleccion_variables_Nolineal_Rf_Sens(11-17).png |
|  |  |

Destacan los modelos 12, 13, 14. Todos ellos tienen en común la variable “css”.

Vamos a realizar una nueva iteración, esta vez teniendo como referencia el modelo 18, que es igual al 11. Esta vez se van a añadir a este modelo las variables “pbcr”,”lvr”,”black”, y sus distintas combinaciones.

|  |  |  |
| --- | --- | --- |
| Modelo | Conjunto variables | Numero variables |
| Modelo 18 | c("dir", "dmi", "hir","ccs") | 4 |
| Modelo 19 | c("dir", "dmi", "hir","ccs","pbcr") | 5 |
| Modelo 20 | c("dir", "dmi", "hir","ccs","lvr") | 5 |
| Modelo 21 | c("dir", "dmi", "hir","ccs", "black") | 5 |
| Modelo 22 | c("dir", "dmi", "hir","ccs","pbcr","lvr") | 6 |
| Modelo 23 | c("dir", "dmi", "hir","ccs","pbcr","black") | 6 |
| Modelo 24 | c("dir", "dmi", "hir","ccs","lvr","black") | 6 |
| Modelo 25 | c("dir", "dmi", "hir","ccs","pbcr","lvr","black") | 7 |

|  |  |
| --- | --- |
| C:\Users\Adrián\AppData\Local\Microsoft\Windows\INetCache\Content.Word\V06_seleccion_variables_Nolineal_Rf_Ac(18-25).png | C:\Users\Adrián\AppData\Local\Microsoft\Windows\INetCache\Content.Word\V06_seleccion_variables_Nolineal_Rf_Sens(18-25).png |
| C:\Users\Adrián\AppData\Local\Microsoft\Windows\INetCache\Content.Word\V06_seleccion_variables_Nolineal_Rf_ROC(18-25).png |  |

Las figuras muestran que cuanto más variables más mejoran lasa métricas de AUC y sensitivity. Como modelos tentativos he escogido los modelos 23 y 25. El modelo 25 supera la barrera 0.6 en sensitivity y es modelo AUC más alto. El modelo 24 ha sido descartado por su accuracy. Se ha escogido el 23 frente al 22 porque tiene algo más de sensitivity y valor AUC.

### Selección entre conjuntos de variables

Los modelos escogidos en las secciones anteriores se comparan vía validación cruzada usando el mismo algoritmo (random forest). Uso este modo de proceder porque busco es un conjunto de variables, no un modelo ya entrenado. Esa parte se dará en fases posteriores.

|  |  |  |
| --- | --- | --- |
| Modelo | Variables | número variables |
| Lineal | c("ccs", "dir", "lvr", "mcs", "uria") | 5 |
| No lineal 1 (antiguo modelo23) | c("dir", "dmi", "hir","ccs","pbcr","black") | 6 |
| No lineal 2  (antiguo modelo25) | c("dir", "dmi", "hir","ccs","pbcr","lvr","black") | 7 |

|  |  |
| --- | --- |
| C:\Users\Adrián\AppData\Local\Microsoft\Windows\INetCache\Content.Word\V06_Comparacion_modelos-Acc.png | C:\Users\Adrián\AppData\Local\Microsoft\Windows\INetCache\Content.Word\V06_Comparacion_modelos-Sens.png |
| C:\Users\Adrián\AppData\Local\Microsoft\Windows\INetCache\Content.Word\V06_Comparacion_modelos-ROC.PNG |  |

El modelo escogido a través de la aproximación lineal es claramente inferior a los otros dos. Escojo el modelo no lineal 2 porque tiene valores ROC y de sensitivity superiores a modelo no lineal 1.

El conjunto de variables ganador es:

c("dir", "dmi", "hir","ccs","pbcr","lvr","black")

Como cabía esperar variables que miden económicamente al solicitante como dir dmi, hir,ccs. Además, no únicamente importa el estado económico actual del solicitante sino se toma en cuenta la historia crediticia del solicitante (“Pbcr”), aunque no tanto como el estado actual.

Contrario a la intuición la tasa de desempleo en el sector del solicitante (uria), o si es autónomo no importan tanto. Probablemente, esto se deba a que 1997 coincide con un periodo de bonanza económica en el que la tasa de desempleo descendía continuamente (alrededor de 4.5% en 1997, <https://es.investing.com/economic-calendar/unemployment-rate-300>).

Además llama la atención que la variable black tenga importancia.

## Tuneado de parámetros

Un parámetro se considerara significativo si con su variación cambia las métricas un 5%, de otro modo se considera que el modelo solo mejora marginalmente. En este caso se elegirá modelos lo más simples y robustos posibles para evitar el sobreajuste.

### Redes

Los parámetros de la red size (número de nodos) y decay, se han tuneado mediante validación cruzada. Se usado la siguiente grid para la búsqueda de los parámetros óptimos:

* Size toma valores 5,10,15,20
* Decay toma valores 0.1, 0.01, 0.001

|  |
| --- |
| size decay Accuracy Sensitivity auc parametros  1 5 0.100 0.5007295 0.8634460 0.7842921 71  2 10 0.100 0.5339091 0.7777667 0.7498721 141  3 15 0.100 0.5538362 0.7403716 0.7289378 211  4 20 0.100 0.5743132 0.7993255 0.7627940 281  5 5 0.010 0.5102437 0.8524133 0.7768831 71  6 10 0.010 0.5328066 0.8041311 0.7506513 141  7 15 0.010 0.5470482 0.7810033 0.7336366 211  8 20 0.010 0.5239966 0.8410504 0.7634015 281  9 5 0.001 0.4719835 0.8649992 0.7839825 71  10 10 0.001 0.4842651 0.8243040 0.7511846 141  11 15 0.001 0.5101128 0.7998167 0.7414134 211  12 20 0.001 0.5584428 0.8028957 0.7584928 281 |
|  |
| |  | | --- | |  | |

Modelos con size 5 consiguen unos valores superiores en las métricas sensitivity y auc, además tiene la ventaja que el modelo será menos proclive al sobreajuste porque tiene menos parámetros. Se podría elegir decay 0.1 o 0.001, se elige el primero porque así el tiempo de optimización de la red será más rápido. Elegimos size 5 decay 0.1 parametros 71.

### Random Forest

Los parámetros de un modelo de random forest se han tuneado. Los parámetros de la grid son los siguientes: número de árboles, número de variable que se eligen en cada nodo (mtry) y mínimo tamaño de nodo.

La estrategia es tunear secuencialmente cada una de estas, primero tuneado las relacionadas con el algoritmo (número de arboles y mtry) y después la propia de los arboles (mínimo tamaño de nodos)

* Número de árboles: se han tomado los valores 50,100,200,300,500,800,1000

|  |  |
| --- | --- |
| C:\Users\Adrián\AppData\Local\Microsoft\Windows\INetCache\Content.Word\V08_tuneadoRF_Auc.png | C:\Users\Adrián\AppData\Local\Microsoft\Windows\INetCache\Content.Word\V08_tuneadoRF_Accuracy.png |
| C:\Users\Adrián\AppData\Local\Microsoft\Windows\INetCache\Content.Word\V08_tuneadoRF_Sensitivity.png |  |

El número de árboles no parece afectar demasiado a las métricas. Elijo un número de árboles de 800 para asegurarme de que el modelo sea robusto

* Mtry: número de variables que se usan para el Split en cada nodo. Se han utilizado valores 2,3,4

|  |  |
| --- | --- |
| C:\Users\Adrián\AppData\Local\Microsoft\Windows\INetCache\Content.Word\V08_tuneadoRF_mtry_Auc.png | C:\Users\Adrián\AppData\Local\Microsoft\Windows\INetCache\Content.Word\V08_tuneadoRF_mtry_Accuracy.png |
| C:\Users\Adrián\AppData\Local\Microsoft\Windows\INetCache\Content.Word\V08_tuneadoRF_mtry_Sensitivity.png |  |

La métrica que más varía es accuracy, según esta, el mejor valor es 2. En cualquier caso, un cambio en mtry tampoco produce un cambio significativo en las métricas. Así qué nos quedamos con mtry=2 para intentar que el modelo tienda al sobreajuste lo mínimo posible.

* Se ha tuneado el tamaño mínimo de los nodos. Ha tomado unos valores 1,2,3,5,8,10

|  |  |
| --- | --- |
| C:\Users\Adrián\AppData\Local\Microsoft\Windows\INetCache\Content.Word\V08_tuneadoRF_tamañondo_Auc.png | C:\Users\Adrián\AppData\Local\Microsoft\Windows\INetCache\Content.Word\V08_tuneadoRF_tamañondo_Accuracy.png |
| C:\Users\Adrián\AppData\Local\Microsoft\Windows\INetCache\Content.Word\V08_tuneadoRF_tamañondo_Sensivity.png |  |

Con este parámetro las métricas varían de forma marginal. Se ha escogido un valor 10 para tener un modelo robusto, al igual que con los parámetros anteriores.

### Bagging

Los parámetros de un modelo Bagging se han tuneado. Los parámetros de la grid son los siguientes: número de árboles y tamaño mínimo de nodo. Se tunean secuencialmente cada una de ellas, empezando por el número de árboles.

* Número de árboles: se han tomado los valores 50,100,200,300,500,800,1000

|  |  |
| --- | --- |
|  | C:\Users\Adrián\AppData\Local\Microsoft\Windows\INetCache\Content.Word\V10_tuneadoBaggin_arboles_Accuracy.png |
| C:\Users\Adrián\AppData\Local\Microsoft\Windows\INetCache\Content.Word\V10_tuneadoBaggin_arboles_Sensitivity.png |  |

Al igual que en random forest, el número de árboles produce efectos marginales en las métricas. Se ha elegido 800 para tener un modelo robusto.

* Se ha tuneado el tamaño mínimo de los nodos. Ha tomado unos valores 1,2,3,5,8,10

|  |  |
| --- | --- |
| C:\Users\Adrián\AppData\Local\Microsoft\Windows\INetCache\Content.Word\V10_tuneadoBaggin_Auc_tamaño_nodo.png | C:\Users\Adrián\AppData\Local\Microsoft\Windows\INetCache\Content.Word\V10_tuneadoBaggin_tamaño_nodo_Accuracy.png |
| C:\Users\Adrián\AppData\Local\Microsoft\Windows\INetCache\Content.Word\V10_tuneadoBaggin_tamaño_nodo_sensivity.png |  |

Se ha elegido un número mínimo de nodo 10 mejora las tres métricas y además un valor alto del número mínimo del nodo hace al modelo menos proclive a sobreajuste.

### GBM

La estrategia para tunear los parámetros del modelo Gradient Boosting es la siguiente:

Se hace una rejilla de parámetros:

* Shrinkage con valores 0.1,0.05,0.03,0.01,0.001
* tamaño de nodo mínimo con valores 5,10,20
* n.trees con valores 100,500,1000,5000

En primer lugar, se elegirá shrinkage y el tamaño de nodo mínimo. Posteriormente, se usará una rejilla para determinar n.trees. Sigo este método secuencial porque n.trees está interrelacionado con shrinkage.

|  |  |
| --- | --- |
| C:\Users\Adrián\AppData\Local\Microsoft\Windows\INetCache\Content.Word\V11_tuneado_GBM_sizeNodo_Sensitivity.png | C:\Users\Adrián\AppData\Local\Microsoft\Windows\INetCache\Content.Word\V11_tuneado_GBM_sizeNodo_Auc.png |
| C:\Users\Adrián\AppData\Local\Microsoft\Windows\INetCache\Content.Word\V11_tuneado_GBM_sizeNodo_Accuracy.png |  |

Elegimos un tamaño mínimo de nodo de 5 porque es el mayor en accuracy y auc, con la sensivity es el peor pero los valores apenas varían.

|  |  |
| --- | --- |
| C:\Users\Adrián\AppData\Local\Microsoft\Windows\INetCache\Content.Word\V11_tuneado_GBM_shrinkage_Sensivity.png | C:\Users\Adrián\AppData\Local\Microsoft\Windows\INetCache\Content.Word\V11_tuneado_GBM_shrinkage_Auc.png |
| C:\Users\Adrián\AppData\Local\Microsoft\Windows\INetCache\Content.Word\V11_tuneado_GBM_shrinkage_Accuracy.png |  |

La métrica que más varía es auc, según esta los mejores valores son 0.03 y 0.05. Elijo el último valor para alcanzar un modelo más robusto.

|  |  |
| --- | --- |
| D:\Master_UCM\Machine Learning\Boston\Figuras\V11_tuneado_GBM_numArboles_Accuracy.png | C:\Users\Adrián\AppData\Local\Microsoft\Windows\INetCache\Content.Word\V11_tuneado_GBM_numArboles_Sensivity.png |
| C:\Users\Adrián\AppData\Local\Microsoft\Windows\INetCache\Content.Word\V11_tuneado_GBM_numArboles_Auc.png |  |

Escogemos para el parámetro número de árboles el valor 500, esta en la cima en las tres métricas.

### SVM

#### SVM lineal

El único parámetro que hay que tunear en SVM lineal es C.

|  |  |
| --- | --- |
| C:\Users\Adrián\AppData\Local\Microsoft\Windows\INetCache\Content.Word\V12_tuneado_SVMlineal_C_accuracy.png | C:\Users\Adrián\AppData\Local\Microsoft\Windows\INetCache\Content.Word\V12_tuneado_SVMlineal_C_Auc.png |
|  |  |

En las tres métricas se puede observar un incremento rápido para valores pequeños de C y para después pasar a una región constante. Elijo C=0.05 porque es el menor valor de C que está en la región constante. De esta forma, el modelo tenderá menos al sobreajuste.

#### SVM polinomial

Los parámetros que se han tuneado son : C, grado, y scala. Se ha usado la siguiente rejilla de parámetros

* C: 0.01,0.05,0.1,0.2,0.5,1,2,5,10
* Degree: 2,3
* Scale: 0.1,0.5,1,2,5

Esta vez los parámetros tienen influencia en las métricas. Entonces la estrategia será en cada parámetro ir descartando combinaciones. Al final de la criba podremos elegir más claramente entre un conjunto muy inferior de parámetros.

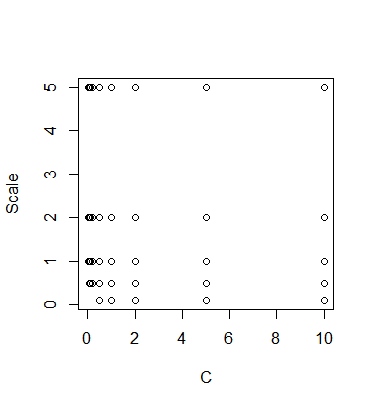
|  |  |
| --- | --- |
| C:\Users\Adrián\AppData\Local\Microsoft\Windows\INetCache\Content.Word\V12_tuneado_SVMpoli_degree_accuracy.png | C:\Users\Adrián\AppData\Local\Microsoft\Windows\INetCache\Content.Word\V12_tuneado_SVMpoli_degree_auc.png |
| C:\Users\Adrián\AppData\Local\Microsoft\Windows\INetCache\Content.Word\V12_tuneado_SVMpoli_degree_sensi.png |  |

Con los dos valores se pueden conseguir valores altos en todas las métricas. Elijo degree=2 para tener modelo robusto.

|  |  |
| --- | --- |
| C:\Users\Adrián\AppData\Local\Microsoft\Windows\INetCache\Content.Word\V12_tuneado_SVMpoli_C_sensi.png | C:\Users\Adrián\AppData\Local\Microsoft\Windows\INetCache\Content.Word\V12_tuneado_SVMpoli_C_accuracy.png |
| C:\Users\Adrián\AppData\Local\Microsoft\Windows\INetCache\Content.Word\V12_tuneado_SVMpoli_C_auc.png |  |

Hay modelos con sensitivity muy alta. Pongo como condición modelos con sensitivity mayor de 0.95, y a partir de aquí elijo aquel que pueda darme la mayor sensitivity sin que se resienta demasiado AUC.

Como aun así hay muchas opciones posibles grafico los parámetros de los modelos



Cuanto menor sean estos parámetros menos tenderán a sobreajustar. La estrategia es escoger un modelo con parámetros que estén los más cerca posible del origen de coordenadas pero que aun así tenga buenas métricas.

Escojo C=0.05 degree=2 y scale =0.5. Este caso tiene las siguientes métricas: Accuracy 0.902, Auc 0.806, sensitivity 0.972

#### SVM RBF

Los parámetros que se han tuneado son: C y sigma. En ambos un valor alto significa que aumenta la varianza del modelo pudiendo incurrir en sobreajuste. Por lo tanto dentro de las métricas se escoge los valores de los parámetros lo menores posible.

La rejilla que he utilizado es C=0.01,0.05,0.1,0.2,0.5. Sigma = 0.01,0.05,0.1,0.2,0.5,1,2,5,10,30

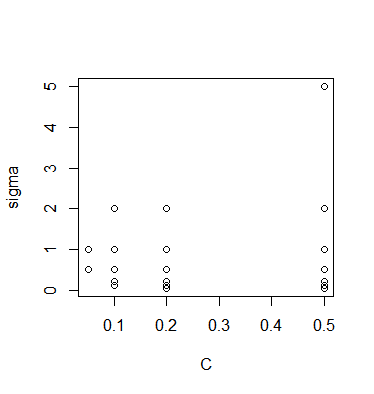
|  |  |
| --- | --- |
| C:\Users\Adrián\AppData\Local\Microsoft\Windows\INetCache\Content.Word\V12_tuneado_SVM_RBF_C_accuracy.png | C:\Users\Adrián\AppData\Local\Microsoft\Windows\INetCache\Content.Word\V12_tuneado_SVM_RBF_C_auc.png |
| C:\Users\Adrián\AppData\Local\Microsoft\Windows\INetCache\Content.Word\V12_tuneado_SVM_RBF_C_sens.png |  |

Al igual que en el caso anterior, vamos a poner condiciones para ir quitando casos. Posteriormente elegiremos de entre el subconjunto resultante. Con los gráficos vamos cogiendo las condiciones de manera que descartamos el mayor número de casos.

Descartamos todos los casos que no cumplan las siguientes condiciones:

Accuracy>0.85, auc>0.75, sensitivity>0.85

De los 50 casos iniciales de la rejilla tenemos 20 resultantes, que he graficado en función de sus parámetros sigma y C.



Ahora intentaremos coger aquel caso que tenga los parámetros menores con las métricas mayores posibles. La elección final es:

C=0.20, Sigma=0.5, Accuracy=0.889, Auc=0.8027, Sensitivity=0.9026

## Comparación modelos

Se comparan todos los modelos anteriormente tuneados. Se usará como método de remuestreo K-validacion cruzada para todos los algoritmos, donde K es 4, y se hace 20 repeticiones.

|  |  |
| --- | --- |
| C:\Users\Adrián\AppData\Local\Microsoft\Windows\INetCache\Content.Word\V13_Comparacion_Accuracy.png | C:\Users\Adrián\AppData\Local\Microsoft\Windows\INetCache\Content.Word\V13_Comparacion_Auc.png |
| C:\Users\Adrián\AppData\Local\Microsoft\Windows\INetCache\Content.Word\V13_Comparacion_Sensitivity.png |  |

De entre todos estos modelos elijo GBM, porque es el que tiene la sensitivity mayor y sus otras métricas son parecidas a las de los demás modelos.

## Construcción Ensemble

Con el objetivo de conseguir mejores modelos, se va a construir modelos ensemble usando combinaciones lineales predicciones del apartado anterior.

unipredi$predi9 <- (unipredi$net + unipredi$Rf)/2

unipredi$predi10 <- (unipredi$net + unipredi$Log)/2

unipredi$predi11 <- (unipredi$net + unipredi$GBM)/2

unipredi$predi12 <- (unipredi$net + unipredi$SVM\_lineal)/2

unipredi$predi13 <- (unipredi$net + unipredi$SVM\_Poli )/2

unipredi$predi14 <- (unipredi$net + unipredi$SVM\_RBF)/2

unipredi$predi15 <- (unipredi$Rf + unipredi$Log)/2

unipredi$predi16 <- (unipredi$Rf + unipredi$Bagging)/2

unipredi$predi17 <- (unipredi$Rf + unipredi$GBM)/2

unipredi$predi18 <- (unipredi$Rf + unipredi$SVM\_lineal)/2

unipredi$predi19 <- (unipredi$Rf + unipredi$SVM\_Poli )/2

unipredi$predi20 <- (unipredi$Rf + unipredi$SVM\_RBF)/2

unipredi$predi21 <- (unipredi$Log + unipredi$Bagging)/2

unipredi$predi22 <- (unipredi$Log + unipredi$GBM)/2

unipredi$predi23 <- (unipredi$Log + unipredi$SVM\_lineal)/2

unipredi$predi24 <- (unipredi$Log + unipredi$SVM\_Poli )/2

unipredi$predi25 <- (unipredi$Log + unipredi$SVM\_RBF)/2

unipredi$predi26 <- (unipredi$Bagging + unipredi$GBM)/2

unipredi$predi27 <- (unipredi$Bagging + unipredi$SVM\_lineal)/2

unipredi$predi28 <- (unipredi$Bagging + unipredi$SVM\_Poli)/2

unipredi$predi29 <- (unipredi$Bagging + unipredi$SVM\_RBF)/2

unipredi$predi30 <- (unipredi$SVM\_lineal + unipredi$SVM\_Poli)/2

unipredi$predi31 <- (unipredi$SVM\_lineal + unipredi$SVM\_RBF)/2

unipredi$predi32 <- (unipredi$SVM\_Poli + unipredi$SVM\_RBF)/2

unipredi$predi33 <- (unipredi$net + unipredi$Rf+unipredi$Log)/3

unipredi$predi34 <- (unipredi$net + unipredi$Rf+unipredi$GBM)/3

unipredi$predi35 <- (unipredi$net + unipredi$Rf+unipredi$SVM\_lineal)/3

unipredi$predi36 <- (unipredi$net + unipredi$Rf+unipredi$SVM\_Poli)/3

unipredi$predi37 <- (unipredi$net + unipredi$Rf+unipredi$SVM\_RBF)/3

unipredi$predi38 <- (unipredi$net + unipredi$Log+unipredi$GBM)/3

unipredi$predi39 <- (unipredi$net + unipredi$Log+unipredi$SVM\_lineal)/3

unipredi$predi40 <- (unipredi$net + unipredi$Log+unipredi$SVM\_Poli)/3

unipredi$predi41 <- (unipredi$net + unipredi$Log+unipredi$SVM\_RBF)/3

unipredi$predi42 <- (unipredi$net + unipredi$SVM\_lineal+unipredi$SVM\_Poli)/3

unipredi$predi43 <- (unipredi$net + unipredi$SVM\_lineal+unipredi$SVM\_RBF)/3

unipredi$predi44 <- (unipredi$net + unipredi$SVM\_Poli+unipredi$SVM\_RBF)/3

unipredi$predi45 <- (unipredi$Rf+unipredi$Log+unipredi$GBM)/3

unipredi$predi46 <- (unipredi$Rf+unipredi$Log+unipredi$SVM\_lineal)/3

unipredi$predi47 <- (unipredi$Rf+unipredi$Log+unipredi$SVM\_Poli)/3

unipredi$predi48 <- (unipredi$Rf+unipredi$Log+unipredi$SVM\_RBF)/3

unipredi$predi49 <- (unipredi$Log+unipredi$GBM+unipredi$SVM\_lineal)/3

unipredi$predi50 <- (unipredi$Log+unipredi$GBM+unipredi$SVM\_Poli)/3

unipredi$predi51 <- (unipredi$Log+unipredi$GBM+unipredi$SVM\_RBF)/3

unipredi$predi52 <- (unipredi$GBM+unipredi$SVM\_lineal+unipredi$SVM\_Poli)/3

unipredi$predi53 <- (unipredi$GBM+unipredi$SVM\_lineal+unipredi$SVM\_RBF)/3

unipredi$predi54 <- (unipredi$SVM\_lineal+unipredi$SVM\_Poli+unipredi$SVM\_RBF)/3

unipredi$predi55 <- (unipredi$net + unipredi$Rf + unipredi$Log + unipredi$GBM)/4

unipredi$predi56 <- (unipredi$net + unipredi$Rf + unipredi$Log +unipredi$SVM\_lineal)/4

unipredi$predi57 <- (unipredi$net + unipredi$Rf + unipredi$Log +unipredi$SVM\_Poli)/4

unipredi$predi58 <- (unipredi$net + unipredi$Rf + unipredi$Log +unipredi$SVM\_RBF)/4

unipredi$predi59 <- ( unipredi$Rf + unipredi$Log + unipredi$GBM+unipredi$SVM\_lineal)/4

unipredi$predi60 <- ( unipredi$Rf + unipredi$Log + unipredi$GBM+unipredi$SVM\_Poli)/4

unipredi$predi61 <- ( unipredi$Rf + unipredi$Log + unipredi$GBM+unipredi$SVM\_RBF)/4

unipredi$predi62 <- ( unipredi$Log + unipredi$GBM+unipredi$SVM\_lineal+unipredi$SVM\_Poli)/4

unipredi$predi63 <- ( unipredi$Log + unipredi$GBM+unipredi$SVM\_lineal+unipredi$SVM\_RBF)/4

unipredi$predi64 <- ( unipredi$GBM+unipredi$SVM\_lineal+unipredi$SVM\_Poli+unipredi$SVM\_RBF)/4

unipredi$predi65 <- (unipredi$net + unipredi$Rf + unipredi$Log + unipredi$GBM+unipredi$SVM\_lineal)/5

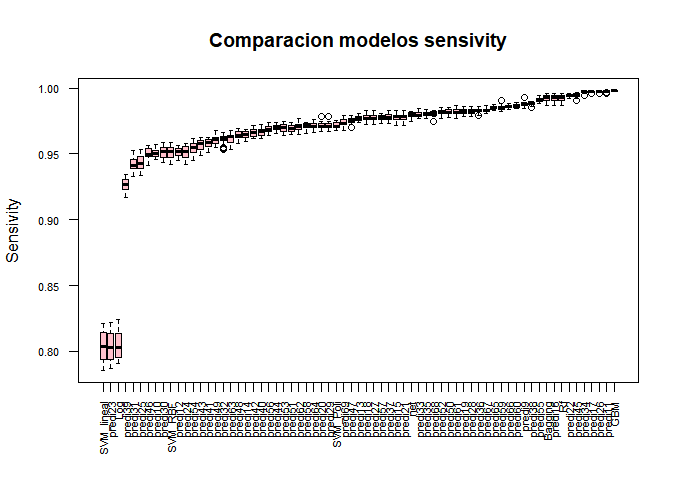
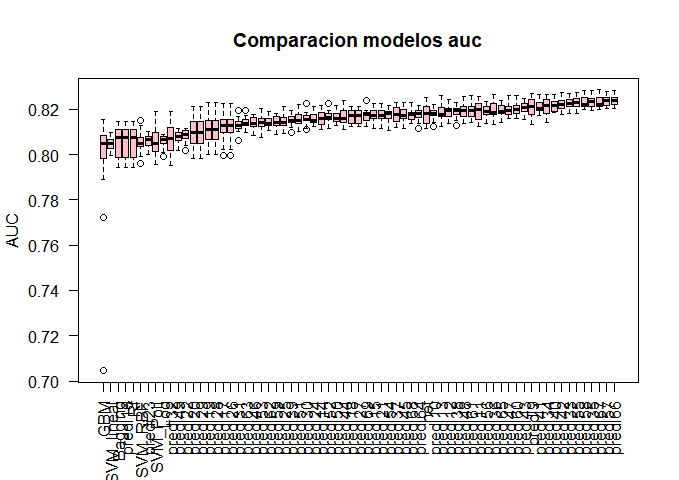
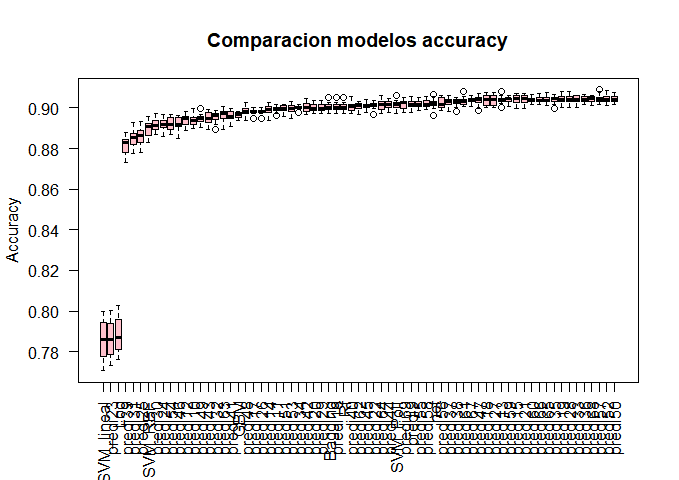
unipredi$predi66 <- (unipredi$net + unipredi$Rf + unipredi$Log + unipredi$GBM+unipredi$SVM\_Poli)/5

unipredi$predi67 <- (unipredi$net + unipredi$Rf + unipredi$Log + unipredi$GBM+unipredi$SVM\_RBF)/5

unipredi$predi68 <- ( unipredi$Rf + unipredi$Log + unipredi$GBM+unipredi$SVM\_lineal+unipredi$SVM\_Poli)/5

unipredi$predi69 <- ( unipredi$Rf + unipredi$Log + unipredi$GBM+unipredi$SVM\_lineal+unipredi$SVM\_RBF)/5

Comparamos gráficamente todos los modelos:



Como modelo final se ha elegido GBM. Este algoritmo tiene relativamente un valor auc bajo (el más bajo de hecho), pero es irrelevante porque de todas formas no mucha diferencia en esta métrica entre los valores mayores o menores. Además, este modelo es el que tiene un valor de la sensivity mayor, métrica que considero vital.

No he elegido los ensamblados porque considero que no compensa el aumento en las métricas con el aumento de la complejidad.

## Conclusiones

El modelo escogido finalmente tiene las siguientes características:

* Su algoritmo es GBM.
* Sus variables son dir, dmi, hir, ccs, pbcr , lvr, black.
* Sus parámetros son shrinkage= 0.05, n.minobsinnode= 5 y n.trees= 500.

Los valores medios de las métricas de este modelo son:

* Accuracy 0.896
* Auc 0.797
* Sensitivity 0.997

Como se puede apreciar, gracias a su alto valor de sensitivity corremos bajo riesgo de conceder hipotecas a quien no se debería. Entonces, gracias a este modelo la entidad puede conceder créditos con más seguridad porque el número de impagos será mínimo.

Los ensamblados no procuran mucha mejoría respecto a los modelo más sencillos como Random Forest o GBM.

# Anexo: Distribución archivos

* Carpeta Código:
  + v6\_base: selección de variables
  + v7\_Tunear\_Redes: tuneo parámetros de algoritmo de redes
  + v8\_Tunear\_RF :tuneo parámetros de algoritmo de redes
  + v10\_Tunear\_Bagging tuneo parámetros de algoritmo de Bagging
  + v11\_Tunear\_GradientBoosting tuneo parámetros de algoritmo de GBM
  + v12\_Tunear\_SVM tuneo parámetros de algoritmo de SVM: lineal, polinomial y radial
  + v13\_Comparacion\_Modelos y ensembles: se comparan los modelos tuneados, y se crean modelo ensemble y se comparan.
* Carpeta código:
  + Hdma: datos originales
  + Datos: datos originales limpios
  + datos\_originales\_limpios: datos originales limpios