Tarea

‘Machine learning’

Autor: Javier Portela

Actualizado Enero 2020

Trabajo de Clasificación binaria.

Este trabajo está orientado a predecir una variable **binaria** a través de diferentes algoritmos de clasificación.

Normas para la realización del trabajo

1) El trabajo es individual.
2) Se entregará el trabajo en pdf en una copia digital enviada al Campus Virtual.
3) El trabajo deberá estar explicado (no basta con responder a las cuestiones), indicando, si es necesario, el código utilizado. Se valora la claridad de exposición en el informe y la estructura. Puede contener Anexos de datos y gráficos o no, todo según vuestro criterio. En principio no hay límite de páginas en ese sentido, pero la parte explicada debe ser razonable. El trabajo es libre, con lo cual se agradece sentido común.
4) El trabajo se debería recuperar en septiembre si se da al menos una de las siguientes circunstancias:

- La presentación y explicaciones son escasas.
- Los modelos comprobados son demasiado limitados.
- Se observan signos de copia de otros trabajos de otros alumnos.

5) Se debe responder obligatoriamente a cada una de las cuestiones a) hasta la f) expuestas más abajo. *Si falta una sola de las cuestiones a-f, o algoritmos, el trabajo está para septiembre*.

Se construirán los modelos sobre **una** matriz de datos a elegir voluntariamente. Se aconseja aproximadamente más de 300 observaciones y 5 variables input posibles, de las cuales al menos una debe ser categórica.

Como mínimo, para poder elaborar modelos mínimamente complejos, se deberían tener 100 observaciones en la clase minoritaria de la variable dependiente.

En el ANEXO hay información sobre datos a utilizar. Podéis utilizar datos vuestros, de otras webs, etc. siempre que se cumplan las condiciones. También en cada artículo de referencia del tema de Comparación y Estrategias avanzadas hay más información sobre datasets que pueden utilizarse.

La selección de los datos a tratar forma parte del trabajo. El profesor no dará más indicaciones sobre la conveniencia de utilizar o no cualquier archivo de datos.

Cuestiones generales a responder

Se trata de conseguir obtener el mejor método/algoritmo para predecir, estable en cuanto a su performance, y en comparación con un modelo de regresión logística con selección de variables stepwise.

a) Se deben realizar pruebas suficientes para obtener una buena selección de variables, obteniendo uno o varios conjuntos de variables tentativos
b) Se requiere la comparación entre los mejores algoritmos y regresión logística ;
c) Se comprobará el efecto de la variación de los parámetros básicos de cada algoritmo (tuneado) (número de nodos en redes, shrink en gradient boosting, etc.).
d) Los algoritmos a utilizar son obligatoriamente y como mínimo:
    - Redes Neuronales
    - Regresión Logística
    - Bagging
    - Random Forest
    - Gradient Boosting
    - Support Vector Machines

- También si se quiere y para comprender los datos se puede probar con un simple árbol pero no es obligatorio.

e) Es necesario utilizar validación cruzada, validación cruzada repetida o como mínimo training/test repetido.

f) Es necesario hacer alguna prueba de ensamblado.

Se puede utilizar el programa R aportado por el profesor y cualquier modificación o uso de código. También se puede utilizar python y/o cualquier otro paquete, siempre que se cumplan los apartados (a)-(f).

Se ruega leer el apartado de FAQ y errores típicos de los alumnos que aparece al final del documento.

Nota aproximada del trabajo según lo que se haga:

0) Falta algún apartado a)-f) (por ejemplo ensamblado): **suspenso**

1) Todos los apartados a) –f) correctamente, proceso básico de depuración y selección de variables, trabajo realizado solo con tuneo y remuestro simple (training test repetido por ejemplo): **nota 5-6**

2) Todos los apartados correctamente, proceso básico de depuración y selección de variables, trabajo realizado con validación cruzada repetida, estudio sesgo-varianza, **nota 6-7**

3) Todos los apartados correctamente, buen proceso de depuración y selección de variables, validación cruzada repetida, estudio sesgo-varianza, , variación de parámetros no considerados en el grid de caret como maxit en redes o sampsize en random forest, **nota 7-8**

4) Todos los apartados correctamente, buen proceso de depuración y selección de variables, validación cruzada repetida, estudio con diferentes sets de variables, variación de parámetros no considerados en el grid de caret como maxit en redes o sampsize en random forest y bagging, usar xgboost, datos interesantes o complicados, uso de estrategias avanzadas como diferentes estrategias de tuneo (random o BSO), AutoML de h2o, etc., uso de python en algún algoritmo para comparar, buena descripción del problema, reflexiones como en el tema Decisión en clasificación binaria, y conclusiones (añadir a las conclusiones tabla básica del modelo de regresión y un árbol básico con comentarios). **nota 8-10**

En este apartado 4 no es necesario hacer todo lo que pone para obtener un 10; son ideas ilustrativas para la mejora, normalmente con algunas cosas más que en el apartado 3 es suficiente; pero queda al criterio del profesor.

Se ruega leer el apartado de FAQ y errores típicos de los alumnos que aparece al final del documento.

ANEXO

WEBS DE DATASETS PARA APLICAR TÉCNICAS DE MACHINE LEARNING

RECOMENDADOS PARA EMPEZAR, MUY BIEN ESTRUCTURADOS
****
https://archive.ics.uci.edu/ml/datasets.html
https://sci2s.ugr.es/keel/datasets.php
https://vincentarelbundock.github.io/Rdatasets/datasets.html
****
1) En uci y keel están los archivos ordenados por clasificación o regresión.
****
2) En la web Rdatasets_
https://vincentarelbundock.github.io/Rdatasets/datasets.html
Archivos de más de 1200 observaciones, lista elaborada por mí (para clasificación binaria, aparte de los datasets denotados como clasificación, también se pueden utilizar los multiclase escogiendo una clase objetivo de entre las múltiples clases):

Item	Rows	Cols	modelo
BEPS	1525	10	clasificación
Caravan	5822	86	clasificación
Gunnels	1592	10	clasificación
Hdma	2381	13	clasificación
Hmda	2381	13	clasificación
VerbAgg	7584	9	clasificación
WVS	5381	6	clasificación
Wells	3020	5	clasificación
YouthRisk2007	13387	6	clasificación
azcabgptca	1959	6	clasificación
dengue	2000	13	clasificación
flchain	7874	11	clasificación
mexico	1359	33	clasificación
mifem	1295	10	clasificación
monica	6367	12	clasificación
ohio	2148	4	clasificación
spam7	4601	7	clasificación
student	9679	13	clasificación
turnout	2000	5	clasificación
voteincome	1500	7	clasificación
Car	4654	70	clasificación multiclase
Chile	2700	8	clasificación multiclase
Kakadu	1827	22	clasificación multiclase
msqR	6411	79	correspondencias
colon	1858	16	cox
cricketer	5960	8	cox
mgus2	1384	10	cox
nwtco	4028	9	cox
BudgetFood	23972	6	regresión
BudgetItaly	1729	11	regresión
BudgetUK	1519	10	regresión
Computers	6259	10	regresión
DoctorContacts	20186	15	regresión
HI	22272	13	regresión
InstInnovation	6208	25	regresión
Males	4360	12	regresión

Males	4360	12	regresión
MathPlacement	2696	16	regresión
MedExp	5574	15	regresión
PatentsRD	1629	7	regresión
SLID	7425	5	regresión
SaratogaHouses	1728	16	regresión
Schooling	3010	28	regresión
Snmesp	5904	8	regresión
Star	5748	8	regresión
VietNamH	5999	11	regresión
Vocab	30351	4	regresión
Wage	3000	11	regresión
Wages	4165	12	regresión
Wages	4165	12	regresión
Workinghours	3382	12	regresión
azpro	3589	6	regresión
baseball	21699	22	regresión
diamonds	53940	10	regresión
mdvis	2227	13	regresión
medpar	1495	10	regresión
nlschools	2287	6	regresión
rwm5yr	19609	17	regresión
science	1385	7	regresión
DoctorAUS	5190	15	regresión muchos dep posibles ver web
OFP	4406	19	regresión varias dependientes
VietNamI	27765	12	regresión y clasi
Gestation	1236	23	regresión y clasificación
NCbirths	1450	15	regresión y clasificación

ZIPEADOS POCO EXPLICADOS

http://biostat.mc.vanderbilt.edu/wiki/Main/DataSets?CGISESSID=10713f6d891653ddcbb7ddbdd9cffb79
https://www.cs.waikato.ac.nz/ml/weka/datasets.html

DATOS COMPLICADOS PERO INTERESANTES

https://www.nature.com/sdata/
http://www.inf.ed.ac.uk/teaching/courses/dme/html/datasets0405.html

CONCURSOS

https://www.kaggle.com/
https://www.drivendata.org/competitions/
http://www.chalearn.org/challenges.html
https://www.kdd.org/kdd-cup

Frequently Asked Questions (FAQ) sobre el Trabajo

1) Estructura

1) ¿Hay un esquema concreto para el trabajo?

En principio hay un esquema básico, sobre él se puede trabajar y alterar cosas:

1) Descripción de los datos

2) Depuración y codificación, estandarización y creación de dummies

3) Estudio de selección de variables; selección primaria del mejor set o mejores sets bajo logística

4) Tuneo-optimización comentado de cada algoritmo con el mejor set de variables obtenido anteriormente

5) Comparación final

6) Ensamblado. Conclusiones

7) Refinamiento, si procede. Algunas ideas (depende de los casos, no todas son necesarias) :

a) cambiar-añadir variables y/o tuneo en los mejores algoritmos,

b) cambiar ponderaciones en los mejores ensamblados si procede

c) añadir técnicas de reducción de varianza como sortear observaciones/variables en xgboost

d) si hay dudas sobre el sobreajuste o qué algoritmo es mejor, probar a repetir toda la comparación final con más semillas u otro tamaño de grupos

e) comprobar resultados y los mejores algoritmos en h2o, usar h2o AutoML y/o usar python

f) variar el punto de corte para clasificación, tema Decisión en Clasificación Binaria

2) Presentación y comentarios

¿Hay algunas normas concretas sobre presentación y comentarios?

Normalmente se deja al alumno actuar según su sentido común. Se agradece que se muestren pequeños scripts y partes de código intercaladas entre el texto y gráficos. También que se explique suficientemente bien el proceso y toma de decisiones en los tuneados, selección de variables, etc. Se espera que el trabajo pueda servir como

documento de referencia al autor en un futuro, para recordar conceptos o técnicas olvidadas.

Respecto a la presentación, solo se pide un pdf. Se pueden añadir anexos a voluntad.

3) Selección de variables

1) ¿Puedo usar otros métodos de selección de variables como rfe, Boruta, modelo escogido arbitrariamente, cortar por grafico de importancia, etc.?

Sí, pero al menos tienes que compararlo con el set de variables obtenido con un método clásico estándar como stepwise AIC o BIC. Esto es obligatorio.

Por otra parte, no solo hay que tener en cuenta el resultado empírico que observamos por CV, sino que si un set de variables tiene muchas más variables que otro pero la ganancia en error es muy pequeña, estaremos posiblemente sobreajustando a pesar de que empíricamente en nuestras pruebas parezca mejor. Como en el ejemplo ozono del tema Decisión en Clasificación Binaria.

2) Para algoritmos determinados puede ser mejor un set de variables y para otros otro set . ¿Puedo usar diferentes sets en cada algoritmo?

Es cierto que por ejemplo los algoritmos basados en árboles pueden beneficiarse de añadir más variables porque las usan de manera diferente.

Pero recomiendo:

a) siempre comparar con el set obtenido en la primera fase general de selección bajo logística.

b) No fiarse demasiado de lo empírico; usar demasiadas variables si no hay mucha diferencia en el error puede llevar al sobreajuste.

3) Tengo tres sets de variables tentativos, ¿puedo aplicarlos y tunearlos sobre todos los algoritmos y comparar?

Sí, es correcto; todo vale, teniendo cuidado lo que se comenta en el punto 2. Por otro lado, hay que tener claro que también se puede seguir el esquema original con un solo set de variables A y después, al final, añadir pequeñas modificaciones y/o tuneo con otros sets de variables, en los algoritmos que quedaron mejores en las comparaciones finales, que usaban el set A de variables.

4) ¿Es necesario crear dummies antes de la selección de variables?

Yo lo recomiendo porque hacerlo así da modelos más finos. En particular, por este orden:

1) Crear dummies

2) Eliminar dummies con pocas observaciones con valor 1 (menos de 20-30 por ejemplo).

Eventualmente crear nuevas dummies a partir de uniones de dummies y usar éstas ("si Valencia=1 o Almería=1 entonces Valmeria=1"). Esto último si el conocimiento del contexto lo hace lógico.

4) Redes

¿ Es estricto calcular como mínimo 30 observaciones por parámetro?

Es simplemente una protección, 20 observaciones es menos seguro. Por otro lado, existe lo que se llama la maldición de la dimensionalidad: a más variables input, exponencialmente más observaciones se necesitan para poder representar la relación funcional entre la variable dependiente y las variables input.

Por lo tanto, si tienes 30 variables input en tu modelo no necesitas solo 20 o 30 observaciones por parámetro, posiblemente sea necesario tener 50 para protegerse más, por la alta dimensión del espacio de variables input. No hay fórmulas exactas, se podría decir que 20 observaciones por parámetro para un tamaño moderado de 5 variables input, 30 para 10 variables input, etc. Y se evaluará la ganancia en error. Como se comenta en otras cuestiones, se considera lo empírico (CV) pero también la protección y cuánto ganamos en bajar el error haciendo el modelo más complicado. Si no ganamos lo suficiente, tomamos el modelo más sencillo.

5) Bagging y random forest, gradient boosting

En algunos ejemplos del curso pones random forest o gbm sobre todas las variables, ¿es necesario para el trabajo o hay que tunearlo sobre el set optimo de variables obtenido en la primera fase?

Esos ejemplos son simplemente para aprovechar la posibilidad de obtener un orden de importancia de variables y obtener más información de cómo los modelos basados en árboles las utilizan. Pero estos modelos gbm y random forest no son inmunes al sobreajuste y es mejor usar los algoritmos rf y gbm sobre un set de variables bien seleccionado que dejarles hacer el trabajo. Es decir, en general es mejor aplicarlos sobre el set óptimo (obtenido al principio en selección de variables) o como mucho añadir alguna variable más si parece interesante, que aplicarlos de manera indiscriminada con todas las variables del dataset. También se puede simplemente comparar vía cv repetida el funcionamiento de TODAS las variables respecto al del set óptimo, pero siempre teniendo cuidado con el sobreajuste porque ceñirse solo a lo empírico implica a menudo meter variables que no tienen ninguna justificación real.

6) SVM

Me tarda demasiado el SVM polinomial, ¿es normal?

En el paquete de R a veces sí, es una cuestión del algoritmo de optimización numérica utilizado para construir el SVM . Recomiendo en este caso las siguientes actuaciones:

a) probar con doParallel, poniendo la cabecera doParallel antes y después del tuneado (y de la función de CV repetida que uso para boxplot).

b) tunearlo con una muestra pequeña

c) No tunearlo ,o fijar grado=2 y solo tunear la escala, y a la hora de comparar con otros algoritmos usar menos semillas; la razón es que muy pocas veces SVM polinomial es la mejor opción: si la separación es lineal, es mejor SVM lineal; si no es lineal, SVM RBF se suele adaptar mejor que el SVM polinomial.

7) Ensamblado

¿Puedo usar Caretensemble en lugar de las funciones adaptadas al ensamblado?

Desgraciadamente el Caretensemble no funciona bien: no deja elegir las ponderaciones, sino que las estima mediante regresión. En la mayor parte de los casos esto da lugar a que se ponderen algunos algoritmos con coeficientes negativos, lo que no tiene sentido teórico ni práctico. Otra posibilidad es construir ensamblados a mano, combinando las salidas de predict, cambiar las ponderaciones a mano en las funciones que yo os doy (en lugar de promediar estrictamente), o usar el h2o que sí pondera siempre con números positivos.

8) Velocidad

Tengo problemas de velocidad con R, ¿Qué hago?

Hay varias alternativas que cito más o menos por el orden en que hay que probar:

i) Usar doParallel en el proceso que da problemas. Hay ejemplos en el último tema de estrategias avanzadas. Por ejemplo se puede poner doparallel antes de mis funciones de construcción de datos en Cv repetida y cerrarlo después. También se puede usar en cualquier tuneado.

ii) Si se tienen muchas observaciones, realizar el trabajo con una muestra de ellas.

Se pueden tener en cuenta las pautas para seleccionar el tamaño de muestra que aparecen en el tema de estrategias avanzadas. Una vez decidido el tamaño se puede escoger la muestra por muestreo estratificado o aleatorio.

Una vez realizado el trabajo, si se tiene tiempo, se puede hacer una última comparación de los mejores algoritmos ya tuneados, con los datos completos para observar si el orden entre algoritmos se conserva.

iii) Eventualmente se puede usar CV con 2 o tres repeticiones, o CV sin repetir (en cuyo caso no hay mucho control sobre la varianza), o training test repetido, o por último training test sin repetir. Todas estas opciones las estoy enumerando de mejor a peor.

9) Decisiones en tuneado y modelos

A veces no eliges los valores que te recomienda caret. Por ejemplo, eliges 5 nodos en una red cuando caret prefiere 15 nodos, pues le dan menor error en sus pruebas de validación cruzada . ¿Cuáles son los criterios para decidirse?

En este caso, seguramente la diferencia de error entre 5 nodos y 15 sea tan pequeña que no merezca la pena usar el modelo más complicado (15 nodos) frente al más sencillo (5 nodos).

En principio los criterios empíricos (es decir, seleccionar directamente el hiperparámetro con menor valor de error promedio en validación cruzada) no son suficientes.

A menudo un tuneado nos da un error por CV relativamente más bajo con una cantidad de parámetros o nodos demasiado alto. Eso ocurre porque las técnicas de remuestreo tampoco son perfectas, y las debemos considerar simultáneamente con medidas de protección ante el sobreajuste. Por lo tanto, los criterios que se deben de tener en cuenta en el proceso de tuneado y selección de modelos deben ser varios:

1) Lo empírico, teniendo en cuenta sesgo y también varianza

2) Aspectos numéricos de sentido común. Por ejemplo, en las redes considerar modelos con al menos 20-30 observaciones por parámetro. .

3) Ante diferencias muy pequeñas entre resultados empíricos, escoger el modelo más sencillo (con menos parámetros).

4) Ante diferencias muy pequeñas entre resultados empíricos, escoger los modelos clásicos (logística, regresión lineal) ante los algoritmos de machine learning.

5) Otras consideraciones de índole práctica sobre el modelo construido:

a) Ante la duda, en clasificación eligiremos el modelo con mejor AUC si hay discrepancia entre tasa de fallos y AUC.

b) Ante la duda, eligiremos el modelo en el que las variables introducidas sean más lógicas, tengan más sentido para el investigador. Igualmente se debería verificar que el signo del parámetro asociado tenga sentido, en la tabla de la logística (o regresión lineal en caso de predicción de v. continua).

c) Ante la duda, eligiremos el modelo en que las variables introducidas tengan una medición más lógica y directa (suelen ser variables continuas) o más fáciles o baratas de medir, y siempre que esté claro que en sucesivas campañas y tomas de datos van a poder ser medidas con seguridad con los mismos criterios. Y ante la duda las que tengan menos missing.

Por último, a menudo la performance de un algoritmo no difiere significativamente de otro. Aunque para fortalecer la intuición y la visión gráfica no hemos tenido en cuenta la comparación de la performance con contrastes de hipótesis estadísticos, se podrían también tener en cuenta.

10) Otras medidas de performance para clasificación aparte de tasa de fallos, AUC

Mis datos están desequilibrados y además la tasa de fallos no me interesa demasiado, porque por ejemplo los algoritmos clasifican todas las observaciones como 0. Me interesa comparar los algoritmos con una medida que equilibre sensibilidad y especificidad, puede que manipulando el punto de corte. Pero solo das las funciones para Tasa de Fallos y AUC. ¿Qué hago?

Los candidatos a modelos con mejor AUC, incluyendo ensamblado, pueden servir como punto de partida para posteriores refinamientos. Para abordar un problema más fino en cuanto a la decisión sobre un modelo, se pueden tomar las siguientes medidas, de manera exploratoria:

1) Obtener las medidas básicas en cuanto a sensibilidad , especificidad, etc. con punto de corte 0.5 de los mejores modelos obtenidos y compararlas a través de una tabla.

Esto con pROC se puede hacer, y con los cripts "función resultados…" del tema Decisión en Clasificación Binaria.

2) Con esos modelos y el paquete pROC o bien el paquete optimalcutpoints se pueden encontrar el mejor punto de corte para el equilibrio sensibilidad-especificidad (por ejemplo indice de Youden) y también comparar como funcionan los diferentes algoritmos bajo ese criterio o bajo diferentes puntos de corte o bajo criterios diferentes. El paquete ROCR también puede calcular el F1 Score por ejemplo.

Para los puntos 1 y 2 se puede abordar una perspectiva de training test sencilla, obteniendo los resultados de los algoritmos con caret sobre datos test y calculando todas las medidas de 1)y 2) sobre esos datos test. Se debe repetir varias veces con diferentes semillas para observar la estabilidad de los resultados.

También es interesante utilizar los razonamientos ad-hoc sobre el punto de corte y los gráficos visualpred que aparecen en el tema Decisión en Clasificación Binaria, y los scripts "función resultados…" que aparecen en ese tema.

11) Algunos Errores en R

a) En clasificación binaria, mis funciones están programadas bajo la base de que la variable dependiente está categorizada como Yes, No. Respetando las mayúsculas. Si no se hace así puede haber errores.

b) A veces el rpart no da las medidas de importancia, o no funciona bien. A veces se arregla poniendo cp=-1.

c) Tarda demasiado, o se cuelga. Ver apartado 6).

Algunos errores típicos en los trabajos

1) Considerar las semillas y/o número de grupos de CV (validación cruzada) como un parámetro de los modelos.

Este es un error grave que ocurre en un 5% del alumnado. Las semillas de aleatorización y el número de grupos son solamente un marco, esquemas de muestreo para comparación.

a) En principio todos las pruebas de algoritmos y tuneados deben llevarse a cabo con el mismo esquema (mismas semillas, mismo número de grupos) para EVITAR que la diferencia que observamos entre un algoritmo y otro sea debida a la reestructuración de la muestra sobre la que se prueban las predicciones, y no al buen o mal funcionamiento de cada algoritmo.

b) Otro error típico relacionado es cuando un alumno, para comprobar la estabilidad de una solución-algoritmo, representa varias cajas (box plot) en paralelo, cada una construida con diferentes semillas. Intento muy loable, pero ahí realmente lo que se está viendo es la variabilidad o varianza de los errores al variar la semilla. En realidad se está representando esa variabilidad del error en paralelo en varias cajas, en vez de representarla en vertical que es lo más apropiado.

c) Más grave es cuando se presenta la comparación de un solo algoritmo o varios en varias cajas, cada caja construida con diferente número de grupos de validación cruzada.

En este caso no se está llegando a buenas conclusiones. Pues salvo fluctuaciones aleatorias, el error de CV es menor cuanto mayor sea el número de grupos, pues al aumentar los grupos, se está construyendo el modelo con más observaciones y por lo tanto será más eficiente al predecir el grupo excluido.

d) Lo que sí se puede hacer es, una vez terminado todo el proceso del trabajo (por ejemplo después de construir los ensamblados) y obtenido un gráfico de cajas A, realizar aparte una última comparación de los mejores modelos, bajo un número alto y diferente de semillas y un número mayor de grupos de CV (por ejemplo k=10, muy recomendado en la teoría pero arbitrario igualmente). Obviamente en este caso lo que se busca es que en el gráfico de cajas B construido bajo este segundo esquema, el orden entre cajas y más o menos la varianza se mantiene similar a lo que aparece en la gráfica A de cajas que habíamos obtenido al final del trabajo. Si no es así, es que no había suficiente diferencia entre los algoritmos (observar siempre los valores numéricos del error en el eje Y) y/o bien estábamos sobreajustando y/o eran datos muy sencillos o con pocas observaciones.

En esta última comparación el valor promedio general del error en las cajas del grafico B va a ser menor que en el grafico A pues el numero de grupos de CV se ha puesto

mayor en B que en A. Pero lo que importa en estos gráficos es la comparación entre cajas.

2) No poner conclusiones .

Ocurre poco frecuentemente. Se debe tomar una decisión sobre uno o varios modelos adecuados y dar alguna explicación o reflexión personal.

3) No hacer depuración ni buena selección de variables.

Ocurre poco frecuentemente. No pido explicaciones exhaustivas en la depuración, pero sí que hagáis un mínimo esfuerzo en la selección de variables.

4) Automatizar todo el proceso, dejando a R el tuneado estricto.

Ocurre poco frecuentemente. Como podéis ver en las FAQ, la idea es combinar lo empírico con sentido común.

En el caso de los hiperparámetros en tuneado, aparte de la protección contra el sobreajuste, hay que buscar patrones en los gráficos y resultados. Cuando se fija un hiperparámetro al valor 0.0345 como óptimo uno tiene que estar más o menos seguro de que en un entorno de ese valor se tendrían resultados similares. En general es mejor buscar un entorno estable en un error aceptable en un rango de valores del hiperparámetro, que un punto exacto de valor del hiperparámetro que nos da óptimo error, porque este punto realmente depende de nuestra muestra y estructura de remuestreo.

Igualmente en cualquier estudio de hiperparámetros y comparación de algoritmos nunca hay que perder de vista los valores numéricos del auc o tasa de fallos, pues las diferencias de error pueden ser mínimas en la práctica aunque los gráficos, al estar a escala, nos parezcan muy diferentes.

5) *Tunear exageradamente el número de árboles en Bagging o Random Forest.*

El número de árboles en estos algoritmos es un valor mínimo que hay que buscar. A partir de un cierto valor, se va estabilizando el error y todas las bajadas o subidas son fluctuaciones aleatorias.

Por tanto no tiene sentido decir que es mejor 200 árboles que 300. Simplemente se busca-tunea un valor a partir del cual el error se estabiliza en media. Muchas veces con 100 o 200 árboles es suficiente.

Sí es mucho más importante tunear el sampsize (número de observaciones que se sortean antes de construir cada árbol). Como se ve en la documentación, tiene gran influencia, tanto en bagging como en Random Forest y se debería tunear en ambos algoritmos. Otro error que cometen los alumnos es tunear en bagging pero no en rf, cuando en rf es igualmente importante pues es el mismo algoritmo.

También es importante no confundir con gbm. En gradient boosting sí que hay que tunear el número de árboles pues según la teoría a partir de cierto número se puede sobreajustar (aunque en muchos ejemplos prácticos no detectemos siquiera ese sobreajuste).

6) *Tunear el sampsize en bagging pero no en Random Forest.*

Ocurre a menudo, porque en los ejemplos de la documentación tengo tuneado sampsize en Bagging pero no en Random Forest. Bagging es un caso particular de Random Forest. Es necesario tunear sampsize en ambos algoritmos por separado.

7) *En la comparación numérica con otros paquetes como h2o, o python, no tener en cuenta que el remuestreo es diferente.*

Cuando se comparan paquetes entre sí se buscan más bien generalidades: cuál parece el mejor sistema-algoritmo, cual es el nivel de error aproximado al que se puede aspirar.

Aunque se puede estructurar la muestra para comparar fielmente los resultados entre dos paquetes (por ejemplo , con una simple división training-test que se aplica en ambos paquetes) no lo hemos realizado en el curso, y las pruebas con CV usan en general diferente reordenación aleatoria en los diferentes paquetes.

Por lo tanto las comparaciones numéricas deben tener en cuenta este hecho, y si en caret el auc es de 0.87 y en h2o es de 0.89 esa diferencia puede ser debida a la estructura del remuestreo, y no a los diferentes paquetes-algoritmos. Otra cosa es cuando la diferencia es grande y entonces habrá que profundizar más en la comparación.