Práctica 2 – Competición en Kaggle sobre Clasificación Multiclase

Sistemas Inteligentes para la Gestión en la Empresa

Máster Universitario en Ingeniería Informática

Curso 2015 - 2016

Germán Martínez Maldonado

54097023B

germanm@correo.ugr.es

Equipo Kaggle: Germán Martínez Maldonado

Mejor puntuación: 0.80233

Puesto final: 275

Fecha y hora de la solución: 31 May 2016 19:01:08

1 Trabajo realizado

El trabajo realizado consiste en predecir el destino de un animal que se encuentra en una protectora de animales: adopción, muerte, eutanasia, devolución a su propietario o transferencia. Para esto tenemos una serie de datos sobre cada uno de los animales: identificador, nombre, fecha de llegada, tipo de animal, sexo resultante (esterilización/castración), edad a la salida, raza y color.

Para esta tarea, al igual que en la práctica anterior voy a utilizar el software RStudio para tratar los datos y realizar las predicciones utilizando sobretodo los algoritmos implementados en el paquete eXtreme Gradient Boosting, "xgboost" (aunque también hiciera unas pruebas iniciales con Random Forest), que incluye eficientes modelos de resolución lineal y algoritmos de aprendizaje en árbol.

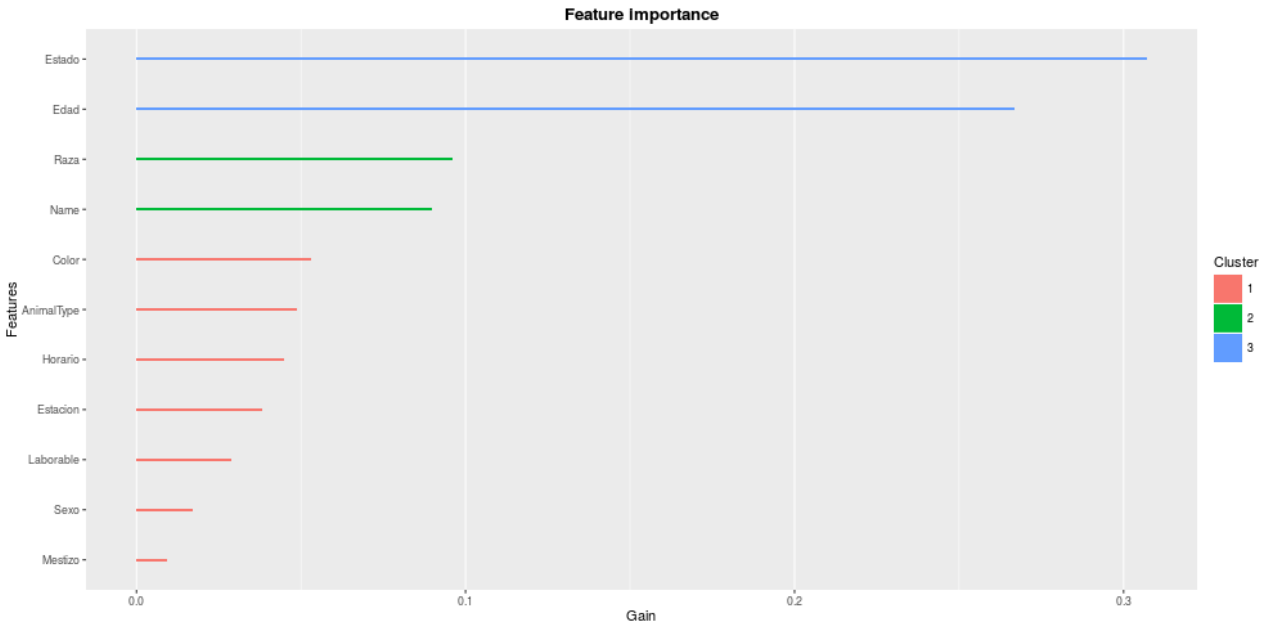
Antes de empezar con las predicciones he manipulado varias variables, además de crear otras nuevas:

- El animal tiene o no un nombre.
- Llegó en horario de mañana, tarde o noche.
- Llegó en día laboral o no laboral.
- En qué estación del año llegó.
- Se indica el sexo del animal: macho, hembra o desconocido.
- Si está castrado, esterilizado o intacto.
- Se simplifica la raza para considerar solo la principal en caso de ser mestizo.
- Distinguir si el perro es o no es mestizo.
- Se simplifica su color para considerar solo el principal en caso de no ser de un único color.
- Homogeneizamos la edad de todos los animales en días.
- Hacemos una predicción de las edades faltantes de los animales.

El paquete "xgboost" utiliza matrices de valores numéricas así que transformamos todas las variables con las que vamos a trabajar en sus equivalentes clases numéricas (que además deben empezar por el valor 0) y generamos las particiones de entrenamiento y validación. Después de un par de pruebas iniciales en las que generé unos archivos de resultado erróneos (de ahí el alto score tan alto en las primeras soluciones), sabiendo que los datos estaban mezclados es importante realizar validación cruzada, en el caso de "xgboost" este divide los datos de entrenamiento en tantas partes como folds le hayamos indicado, quedándose a continuación con la primera parte para utilizarla como datos de prueba, después vuelve a introducir la primera parte en el conjunto de entrenamiento y se queda con la segunda parte para hacer el entrenamiento y así sucesivamente con todas las partes. Gracias a esto podemos asegurarnos una mayor integridad en los resultados.

Realicé de forma iterativa la validación cruzada con diferentes pruebas buscando los parámetros que me dieran un mejor resultado en la métrica de evaluación de rendimiento de la competición (multi class log loss) que en este caso será el valor mínimo de dicho valor, almacenando los parámetros "max_depth", "eta", "gamma", "subsample", "colsample_bytree", "min_child_weight", "max_delta_step", así como el número de rondas (árboles) para llegar a ese valor óptimo; utilizando después eso parámetros en el entrenamiento del modelo.

Además, para conocer si las variables que se van incluyendo son realmente influyentes en el resultado. El funcionamiento básico de la predicción es que el dataset es dividido de forma recursiva varias veces, pero no todas las divisiones son igual de importantes, divisiones a niveles más altos tienen una mayor importancia que divisiones a nivel más profundo, debido a que una mal clasificación en niveles superiores repercute en los conjuntos más pequeños generados de datos que se usarán para la predicción. Este valor se puede cuantificar para calcular la mejora aportada por cada división, el propio paquete "xgboost" puede calcular dichos valores y representarlos mediante una gráfica.



El color de la barra está indicado según los valores de clasificación obtenidos mediante k-medias, en este caso, vemos que tenemos dos variables muy importantes (estado y edad), dos un poco menos importantes (raza y tener nombre) y el resto son mucho menos importantes; generalmente me ha dado mejores resultados trabajar más la interacción con las variables más importantes que eliminar las menos importantes.

Finalmente, solos nos queda realizar la predicción y almacenar los resultados teniendo en cuenta que en la matriz de predicción se han calculado las probabilidades para cada una de las 5 clases por cada elemento de test.

2 Resumen de soluciones intentadas

N.º solución	**Descripción de la manipulación de los datos aplicada**	**Resumen de los algoritmos y software empleados**	**% de aciertos sobre conjunto de validación**	**% de acierto en Kaggle**	**Posición / Fecha / Hora**
1	• Calcula la edad en días de los animales y predice las edades faltantes. • Simplifica la raza de los animales quedándose con la principal. • Saca las 30 razas con más animales y agrupa el resto en un raza común (“Otros”). • Simplifica el color de los animales quedándose con el principal. • Partición de entrenamiento/validación con 60/40 de los datos.	• (Edad) Rpart: ◦ Edad ~ AnimalType + SexuponOutcome + Breed + Color ◦ method=anova • (Predicción) Random Forest: ◦ OutcomeType ~ AnimalType + Edad + SexuponOutcome + Raza + Color ◦ method=cv ◦ number=5 ◦ ntree=2000 • Rstudio	0.6007	13.46458	629 / 27 May 2016 / 23:37:39
2	• Igual que la anterior.	• (Edad) Rpart: Igual que la anterior • (Pre.) eXtreme Gradient Boosting: ◦ OutcomeType ~ AnimalType + Edad + SexuponOutcome + Raza + Color ◦ method=cv ◦ number=5 ◦ nrounds=1000 ◦ eta=c(0.01, 0.001, 0.0001) ◦ max_depth=c(2, 4, 6, 8, 10)	0.6382	12.32193	606 / 28 May 2016 / 15:49:49

		◦ gamma=1 ◦ colsample_bytree=0.7 ◦ min_child_weight=1 • Rstudio			
3	• Igual que la anterior.	• (Pre.) eXtreme Gradient Boosting: ◦ OutcomeType ~ AnimalType + Edad + SexuponOutcome + Raza + Color ◦ objective=multi:softprob ◦ eval_metric=mlogloss ◦ num_class=5 ◦ eta=0.005 ◦ gamma=0.5 ◦ max.depth=10 ◦ min_child_weight=4 ◦ subsample=0.9 ◦ colsample_bytree=0.8 ◦ nthread=3 ◦ nfold=5 ◦ nrounds=min(xgb.cv$test.mlogloss.mean: nrounds=1000)	0.727935	2.94248	563 / 29 May 2016 / 11:27:05
4	• Igual que la anterior.	• Todo igual que la anterior menos nrounds=min(xgb.cv$test.mlogloss.mean: nrounds=10000)	0.691954	3.44006	563 / 29 May 2016 / 12:23:11
5	• Clasifica animales por con nombre y sin nombre. • Clasifica animales por mestizos o no mestizos.	• Mismos parámetros que #3, pero: OutcomeType ~ AnimalType + Edad + SexuponOutcome + Raza + Color + Name + Mestizo	0.782631	0.94366	452 / 29 May 2016 / 19:21:03

6	• Clasifica los animales por su sexo: macho, hembra, desconocido. • Clasifica los animales por su estado de esterilización: esterilizado, castrado, intacto.	• Mismos parámetros que anterior, pero: OutcomeType ~ AnimalType + Edad + Sexo + Estado + Raza + Color + Name + Mestizo	0.688707	0.84411	339 / 29 May 2016 / 20:12:38
7	• Clasifica por horario de llegada: mañana, tarde, noche. • Clasifica por día de llegada: laborable, no laborable. • Clasifica por estación de llegada: primavera, verano, otoño, invierno.	• Mismos parámetros que anterior, pero: OutcomeType ~ AnimalType + Edad + Sexo + Estado + Raza + Color + Name + Mestizo + Horario + Laborable + Estación	0.592959	0.81233	287 / 31 May 2016 / 01:41:14
8	• Igual que la anterior.	• Mismos parámetros que anterior, pero: OutcomeType ~ Estado + Edad + Raza + Name + Color + AnimalType	0.716082	0.84677	- / 31 May 2016 / 10:15:16
9	• Igual que la anterior.	• (Pre.) eXtreme Gradient Boosting: ◦ OutcomeType ~ AnimalType + Edad + Sexo + Estado + Raza + Color + Name + Mestizo + Horario + Laborable + Estación ◦ objective=multi:softprob ◦ eval_metric=mlogloss ◦ num_class=5 ◦ max_depth=sample(6:10, 1) ◦ eta=runif(1, .01, .3) ◦ gamma=runif(1, 0.0, 0.2) ◦ subsample=runif(1, .6, .9) ◦ colsample_bytree=runif(1, .5, .8)	0.737258	0.80233	275 / 31 May 2016 / 11:14:33

		◦ min_child_weight=sample(1:40, 1) ◦ max_delta_step=sample(1:10, 1) ◦ nthread=6 ◦ 5 iteraciones ◦ cv: nfold=5, nrounds=1000			
10	• Igual que la anterior.	• Mismos parámetros que anterior, pero: ◦ 10 iteraciones ◦ cv: nfold=5, nrounds=2000	0.679565	0.79936	273 / 31 May 2016 / 13:42:37
11	• Igual que la anterior.	• Mismos parámetros que anterior, pero: ◦ 20 iteraciones ◦ cv: nfold=5, nrounds=2000	0.653082	0.79907	275 / 31 May 2016 / 19:01:08