Estaciones Valenbisi: Machine Learning para Predecir Bicicletas Disponibles

Fermín Leal Payá, Rafael Vallejo Antich (equipo *CucumberLearners*)

***Resumen*— La capacidad de conocer cuántas bicicletas hay disponibles en las diferentes estaciones Valenbisi repartidas por la ciudad es una prestación indispensable que esta empresa debe aportar a sus clientes para incrementar el valor añadido de su servicio. Sin embargo, es posible ir un paso más allá y ser capaces, no sólo de indicar el número de bicicletas disponibles, si no de predecir cuántas puede haber disponibles en las próximas horas mediante el uso de algoritmos de machine learning. Esta memoria incluye la discusión de análisis de datos, exploración, optimización y comparación de errores entre las diferentes técnicas utilizadas por nuestro equipo.**

I. INTRODUCCIÓN

El estudio que se pretende abordar en esta memoria estriba en la predicción del uso de bicicletas para 4 nuevas estaciones instaladas en la ciudad. El objetivo principal será el de determinar el número de dispositivos libres y usados en cada una de estas nuevas estaciones para las próximas 3 horas durante el mes de julio.

Para ello se dispone de un dataset de muestras etiquetadas, correspondientes a 20 estaciones (entre las que se encuentran las 4 para las que se desea realizar la predicción). Por lo que nos encontramos ante un problema de aprendizaje supervisado.

Así pues, el objetivo final será, una vez creado el modelo, el de completar un fichero de test. En él, dados un ID de estación y una fecha en formato *timestamp* deberemos asociar el número de bicicletas que habrá disponibles.

Haciendo un análisis exploratorio básico inicial, vimos que las 4 estaciones sobre las que realizar la predicción (con IDs 7, 8, 106 y 110) observamos que éstas disponían de 20-30 dispositivos. Por lo tanto, creímos conveniente recurrir a técnicas de regresión para resolver este problema.

Como veremos en detalle en los siguientes puntos, el algoritmo que mejores prestaciones proporcionó en nuestro caso fue el Random Forest Regressor, entrenado con un 70% del dataset total y que nos proporcionó un MAE del 2.54.

II. EL DATASET Y SUS CARACTERÍSTICAS

Los datasets con los que contamos para la generación de un modelo se dividen en 2 partes. Por un lado, disponemos de la información asociada al comportamiento de estas 4 estaciones objetivo durante los meses de abril y mayo (éste será el dataset de *deploy*). Y, por otro lado, se tiene también la información asociada al comportamiento de otras 16 estaciones para los 2 últimos años (éste será el dataset de *historic*). Toda esta información será susceptible de utilizarse para entrenar nuestro modelo, como veremos más adelante.

También cabe destacar que el dataset asociado a cada estación se tiene en ficheros csv independientes, por lo que el primer paso de nuestra labor fue cargar todos estos ficheros en diferentes variables para posteriormente unirlas con el fin de crear los datasets de train y test

correspondientes para entrenar nuestro modelo.

Uno de los primeros pasos es realizar una vista preliminar de las diferentes variables de los datasets, de este modo podemos comprobar de cuántas variables disponemos para la elaboración del modelo, así como el tipo de datos que contiene:

```
str(df)
'data.frame':   207183 obs. of  25 variables:
 $ station                    : int  104 104 104 104 104 104 104 104 104 104 ...
 $ latitude                   : num  39.5 39.5 39.5 39.5 39.5 ...
 $ longitude                  : num  -0.347 -0.347 -0.347 -0.347 -0.347 ...
 $ numDocks                   : int  20 20 20 20 20 20 20 20 20 20 ...
 $ timestamp                  : num  1.34e+09 1.34e+09 1.34e+09 1.34e+09 1.34e+09 ...
 $ year                       : int  2012 2012 2012 2012 2012 2012 2012 2012 2012 2012 ...
 $ month                      : int  6 6 6 6 6 6 6 6 6 6 ...
 $ day                        : int  8 8 8 8 8 8 8 8 8 8 ...
 $ hour                       : int  3 4 5 6 7 8 9 10 11 12 ...
 $ weekday                    : chr  "Friday" "Friday" "Friday" "Friday" ...
 $ weekhour                   : int  100 101 102 103 104 105 106 107 108 109 ...
 $ isHoliday                  : chr  "No" "No" "No" "No" ...
 $ windMaxSpeed.m.s           : num  2.1 1.3 4.1 2.5 1.7 5 6.1 6.3 8.3 8.7 ...
 $ windMeanSpeed.m.s          : num  0.6 0.4 1.2 0.6 0.7 2.7 3.5 3.3 4.1 4.7 ...
 $ windDirection.grades       : num  5 314 318 348 69 92 99 104 112 104 ...
 $ temperature.C              : num  25.6 24.7 24.8 24.8 25.2 23.7 23.9 24.4 24.9 24.8 ...
 $ relHumidity.HR             : int  50 55 42 44 62 85 82 77 72 69 ...
 $ airPressure.mb             : int  1008 1008 1001 1002 1002 1003 1004 1004 1004 1004 ...
 $ precipitation.l.m2         : num  0 0 0 0 0 0 0 0 0 0 ...
 $ bikes_3h_ago               : int  4 3 4 6 4 6 5 6 7 8 ...
 $ full_profile_3h_diff_bikes : num  8.5 8.47 8.44 8.43 8.4 ...
 $ full_profile_bikes         : num  8.43 8.4 8.39 8.37 8.36 ...
 $ short_profile_3h_diff_bikes: num  8.5 8.47 8.44 8.43 8.4 ...
 $ short_profile_bikes        : num  8.43 8.4 8.39 8.37 8.36 ...
 $ bikes                      : int  6 4 6 5 6 7 8 7 3 2 ...
```

Fig. 1: Estructura del dataset

Una vez explorado cómo se estructuran los datos de nuestro dataset, analizamos la distribución de la variable objetivo *$bikes*. Para ello representamos en un histograma los valores que comprende esta variable, tanto para el dataset de *historic* como el de *deploy*:

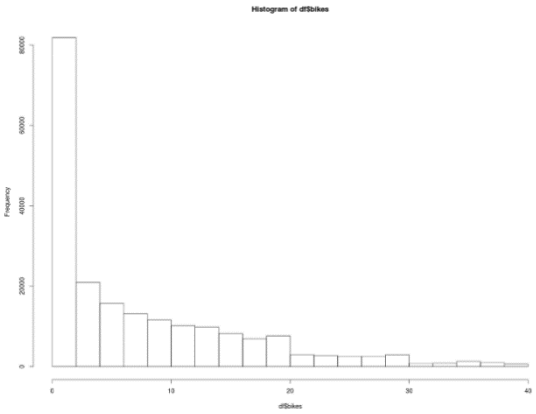

Fig. 2: Distribución de bicicletas para el *historic*

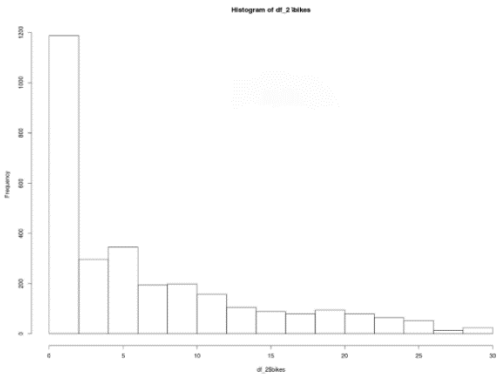

Fig. 3: Distribución de bicicletas para el *deploy*

III. PREPROCESADO

Uno de los apartados que más tiempo consumen en las tareas de ciencia de datos consiste en el preproceso de los mismos y, en este caso de estudio, no fue una excepción.

En primer lugar disponíamos de los datos de las estaciones en ficheros csv independientes, por lo que en primer lugar tuvimos que ser capaces de cargar estos datos en RStudio, asignarlos a variables y posteriormente unirlos componiendo una única tabla con el mismo número de columnas. Esto lo hicimos mediante el comando *rbind*.

Una vez cargados los datos decidimos asignarlos a 2 dataframes distintos. Al igual que se ha comentado anteriormente, se creó una variable que incluyese los datos históricos de las 4 estaciones a predecir y, otra, para las restantes 16 estaciones.

En cuanto a los valores cargados, observamos un grupo muy reducido de muestras que tenían valores faltantes. Puesto que se trataba de un porcentaje ínfimo en comparación con el número de muestras totales, se optó por eliminarlas para la implementación del modelo.

Otros puntos que son comunes en las tareas de preproceso y las medidas que decidimo adoptar fueron las siguientes:

- **Valores atípicos:** Puesto que la variable objetivo viene acotada superiormente por el número máximo de bicicletas en una única estación, el caso de los valores atípicos era un problema que *a priori* no debía darse en nuestro caso de estudio. Además, analizando el valor de la variable objetivo sobre las muestras etiquetadas de las que disponíamos, vimos que ningún registro superaba este valor. Esto puede comprobarse simplemente observando los histogramas de las Figuras 1 y 2. En conclusión, en nuestro análisis del problema, no fue necesario realizar ninguna eliminación de valores atípicos.
- **Escalado:** Para evitar el impacto o peso que determinadas variables pueden establecer erróneamente en el modelo debido al orden de sus magnitudes, decidimos escalar todas las variables numéricas (excepto la variable a predecir) para asignarles un valor en el rango [0, 1]. Para ello hicimos uso de la función *scale*.
- **Eliminación de variables:** La única variable que eliminamos de forma genérica fue *isHoliday*, puesto que para todas las muestras a predecir se cumple que este valor será siempre False, por lo que no debería influir en absoluto a la hora de generar el modelo.

Una vez realizadas las tareas de preproceso, sólo faltaba por generar los subsets de train y test, con los que entrenar y validar nuestros modelos. Para ello, optamos por hacer selecciones del 70% de los datos para el train y el 30% restante para el test.

Esta generación de subsets la realizamos mediante la función createDataPartition() en R del siguiente modo:

```
createDataPartition(df$bikes, p = 0.7, list = FALSE, times = 1)
```

IV. MÉTODOS

A. Baseline

Con el objetivo de tener un punto de partida se generó un algoritmo sencillo como referencia que debía ser el método a batir. En este caso de estudio el Baseline establecido fue el de asignar el número de bicicletas disponibles en las próximas 3 horas el valor de bicicletas disponibles en el momento actual. Es decir, asumimos que este número no va a variar en las siguientes 3 horas.

Como cabe esperar, no se trata de un método de gran precisión pero que puede proporcionarnos una idea generalizada de los resultados que deberemos obtener aplicando métodos de regresión.

Aplicando este Baseline obteníamos un MAE del 4.5. Por lo que consideramos este valor como error a batir de cara a la implementación de los siguientes modelos.

B. Linear Regressor

El primer modelo que decidimos aplicar fue la Regresión Lineal, algoritmo que se basa en la asignación de pesos sobre cada una de las variables de las muestras de entrada. Estas muestras, como se ha comentado en el punto anterior, se normalizaron previamente. Además, se decidió inicialmente incluir todas las variables para la generación del modelo.

El modelo de regresión lineal se llevó a cabo por medio de la función *lm()* de la librería **caret**.

```
lm(bikes ~., data=train)
```

Donde train es el dataset con el que entrenamos y se refiere al dataset con el 70% de las muestras totales.

Aplicando este método obtuvimos un MAE del 4.27 que, a pesar de mejorar el resultado del baseline, sigue siendo un error medio relativamente elevado.

Por otra parte, creímos que existía la posibilidad de que el mes del año podía influir en el comportamiento de las

estaciones. Así pues, y en vistas de que estábamos entrenando con todos los meses del año sobre el dataset de *histórico* decidimos crear un subset para seleccionar únicamente los meses de mayo, junio y julio. El objetivo era excluir para el entrenamiento aquellos meses para los que no tuviéramos registro sobre las estaciones del *deploy*.

De este modo, entrenando nuestro modelo de regresión lineal únicamente con muestras comprendidas entre mayo y julio conseguimos reducir el MAE al 4.21.

Por otra parte, había un par de problemas con el modelo generado:

1. Este modelo asignaba **valores negativos** sobre la variable objetivo, lo cual carece de sentido. Asignando el valor 0 a estos casos conseguíamos que el MAE descendiese hasta el 4.19.
2. Al tratarse de un modelo de regresión se asignan **valores continuos** sobre la variable a predecir, por lo que estábamos estableciendo números reales en lugar de enteros. Aplicando un redondeo mediante la función *round()* conseguimos bajar el MAE a los 4.18.

C. Random Forest

El siguiente algoritmo que aplicamos para la creación de nuestro modelo de regresión fue el Random Forest, por medio de la librería **randomForest**.

Para entrenar el modelo hicimos uso de la función *randomForest* como se muestra a continuación:

```
randomForest(train$bikes ~ ., data = train, ntree = 20)
```

Cabe destacar que la versión Random Forest Regressor de R no permitía entrenar con variables de tipo String. Esto nos supuso un problema a la hora de entrenar con la variable *weekday*. Para poder crear el modelo se optó por eliminar dicha variable. Otra alternativa sería convertirla en variable numérica, aunque debido a la existencia ya de la variable *timestamp* decidimos que no era imprescindible mantenerla.

Una vez realizados estos pasos y generado el modelo, tras validarlo contra el dataset de test obtuvimos un MAE de 2.55.

A diferencia del LinearRegressor, mediante el Random Forest no tuvimos el problema de asignar valores negativos. Por lo que, tras aplicar únicamente el redondeo sobre el valor predicho *bikes*, coseguimos bajar el MAE hasta un valor del 2.54.

Tras generar este modelo, se optó, al igual que en el Linear Regressor, generar un modelo únicamente con las muestras comprendidas entre mayo y julio. Sin embargo, a diferencia del modelo de regresión, el error absoluto medio generado, lejos de reducirse, se vio incrementado.

V. RESULTADOS

El resultado del análisis sobre la predicción de bicicletas disponibles en las estaciones de Valenbisi fue que el modelo que mejor nos ajustó los resultados fue el Random Forest. Este algoritmo mejoró notablemente el accuracy obtenido por medio del Linear Regressor. Posiblemente el error obtenido podría reducirse si dispusiéramos de datos de más meses para las 4 estaciones a predecir.

Por otra parte, analizando el histograma de los valores predichos con el Random Forest vemos que la distribución es similar a la de las muestras de entrada:

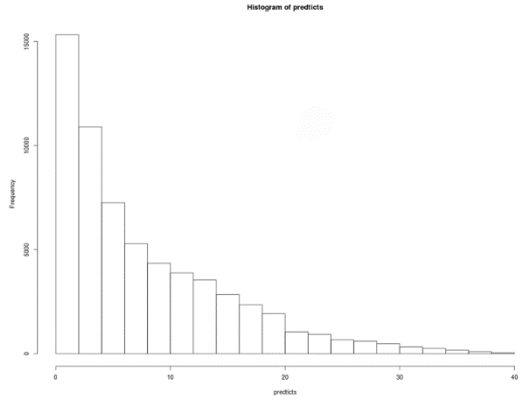

Fig. 4: Histograma de la variable predicha

Ya que la mayoría de valores para la variable bikes se sitúan en el 0 o en valores próximos a él.

En conclusión, con RF conseguimos una distribución que parece ser la habitual para las estaciones de Valenbisi y donde conseguimos el mínimo Mean Absolute Error:

```
> calc_mae(test$bikes,predticts)
[1] 2.540257
```

VI. TRABAJO FUTURO

En caso de disponer de más tiempo podríamos seguir trabajando en otras ideas para mejorar la precisión del modelo. La primera de ellas sería aplicar **otros modelos de regresión** como árboles de decisión (muy rápidos de generar) o, si buscamos precisión, que es lo que se pretende principalmente en este estudio, recurrir a boosting o ensembles para combinar varios modelos. El uso de redes neuronales sería también interesante de estudiar para este caso.

Asimismo, tan importante como aplicar diferentes modelos, sería conveniente también estudiar los resultados obtenidos en función de los parámetros con los que construimos los modelos, haciendo uso de **optimización bayesiana**.

Otras ideas que sería posible a aplicar sería comprobar la existencia de **correlaciones entre variables,** pudiendo generar nuevas variables que puedan influir sobre la variable objetivo y, por tanto, permitan aumentar el accuracy de nuestro modelo.

Otra idea interesante en la que no hemos tenido tiempo para trabajar y que sería interesante, sería analizar el comportamiento de las estaciones en base a su **localización geográfica**. De este modo, se podrían clasificar las estaciones por su ubicación haciendo más preciso el modelo.

Más trabajo adicional consistiría en **cruzar los datos con otros datasets** que sean fácilmente adquiribles y que puedan proporcionarnos información acerca del uso de este medio de transporte por parte de los usuarios. Por ejemplo, sería interesante estudiar la variación del tráfico en función de la hora y día de la semana y año. Puesto que es muy probable que en momentos de alta intensidad del tráfico el uso de bicicletas sea mayor.

VII. CONCLUSIÓN

Para concluir, hemos construido un modelo relativamente preciso para predecir el número de bicicletas disponibles en 4 estaciones para las siguientes 3 horas. Procesando los datos de otras 16 estaciones, además de las 4 en cuestión, y aplicando el modelo de Random Forest fuimos capaces de obtener un MAE del 2.54 sobre nuestro conjunto de test. Pasos como generar modelos mediante otros algoritmos de regresión, trabajar más en el preprocesado de datos para encontrar correlación entre variables, o cruzar nuestros datos con otros que puedan dar valor añadido a nuestro data set, pueden potencialmente mejorar los resultados en el futuro.