









# Practica 3

Competición en DrivenData

## SUBMISSIONS

Score	Submitted by	Timestamp ⓘ
0.7547	<a href="#">David_Castro_UGR</a> 	2019-01-04 22:23:09 UTC
0.7202	<a href="#">David_Castro_UGR</a> 	2019-01-04 22:52:43 UTC
0.7503	<a href="#">David_Castro_UGR</a> 	2019-01-04 22:55:56 UTC
0.7584	<a href="#">David_Castro_UGR</a> 	2019-01-05 13:34:02 UTC
0.7656	<a href="#">David_Castro_UGR</a> 	2019-01-05 19:33:10 UTC
0.7632	<a href="#">David_Castro_UGR</a> 	2019-01-05 20:23:29 UTC

## Índice

1. Tabla de Datos	4
2. Introducción	5
3. Análisis de datos	5
4. Parámetros de los algoritmos	12
5. Conclusión	14
6. Referencias	15

## Tabla de datos

<u>Algoritmo</u>	<u>Posición</u>	<u>Score DD</u>	<u>Score Local</u>	<u>Preprocesamiento</u>	<u>Configuracion</u>	<u>Fecha</u>
RandomForest	1585	0.7547	0.9612	Eliminación de las variables que se nombran posteriormente, y se usa los valores de las instancias de train a la hora de calcular la media y la moda	Numero de estiamdores = 1000  Cross validation 5 veces	04/01/2019
LightGBM	1585	0.7202	0.9308	Eliminación de las variables que se nombran posteriormente, y se usa los valores de las instancias de train a la hora de calcular la media y la moda	Numero de estiamdores = 2000  Cross validation 5 veces	04/01/2019
RandomForest	1585	0.7503	0.9610	Eliminación de las variables que se nombran posteriormente, y se usa los valores de las instancias de train a la hora de calcular la media y la moda	Numero de estiamdores = 300  Cross validation 5 veces	04/01/2019
RandomForest	1568	0.7584	0.9613	Eliminación de las variables que se nombran posteriormente, y se usa los valores de las instancias de test a la hora de calcular la media y la moda	Numero de estiamdores = 1000  Cross validation 5 veces	05/01/2019
RandomForest	1436	0.7656	0.9613	Recupero algunas de las variables y creo tres nuevas, año, mes y tiempo funcionando	Numero de estiamdores = 1000  Cross validation 5 veces	05/01/2019
RandomForest	1436	0.7632	0.9612	Recupero algunas de las variables y creo tres nuevas, año, mes y tiempo funcionando	Numero de estiamdores = 1000  Cross validation 20 veces	05/01/2019

# Introducción

En esta práctica vamos a participar en una competición de la página <https://www.drivendata.org>. Para esta competición se va a tratar de intentar predecir el estado de la bomba de agua en distintos sitios con determinadas características. El conjunto de datos inicial para el “train” que tenemos contiene 40 variables y la clase, y contiene 59400 instancias. Y el conjunto de datos test que contiene 14850 instancias.

Para esta práctica he usado dos algoritmos a la hora de presentar una solución. Que son:

- Ramdonforest
- LightGBM

A parte de los dos algoritmos anteriores, también he usado otro dos de manera loca para hacer pruebas, pero visto que no han dado buenos resultados no los he tenido en cuenta. Los dos que he usado son:

- K-means
- Xgboost

Además para esta práctica he usado python a la hora de programar y ejecutar los algoritmos de predicción. Pero anteriormente he usado KNIME para analizar los datos.

## Análisis de datos

La parte inicial del análisis de datos la he realizado con el programa KNIME, en el que he usado diferentes forma de tratar los datos para saber la correlación que tenían con la el estado de las bombas de agua.

En primer lugar empecé usando únicamente el fichero de train para saber la correlación que había entre las distintas variables. Pero al tener problemas con la lectura de algunas filas, decidí directamente fusionar el fichero de train con las variables, con el de train de estados de las bombas de uso. Con ello he creado un fichero que he modificado para que los valores NaN se cambiasen por espacios en blanco para que el programa pueda leerlos.

Como se muestra en al siguiente imagen los nodos que he usado para ver la relación que tiene los datos entre ellos son principalmente un filtro inicial de columnas para ir quitando aquellas que no fueran interesantes. Y un pre procesamiento para tratar los valores perdidos de distintas formas, borrándolos, haciendo la media, la moda, etc.

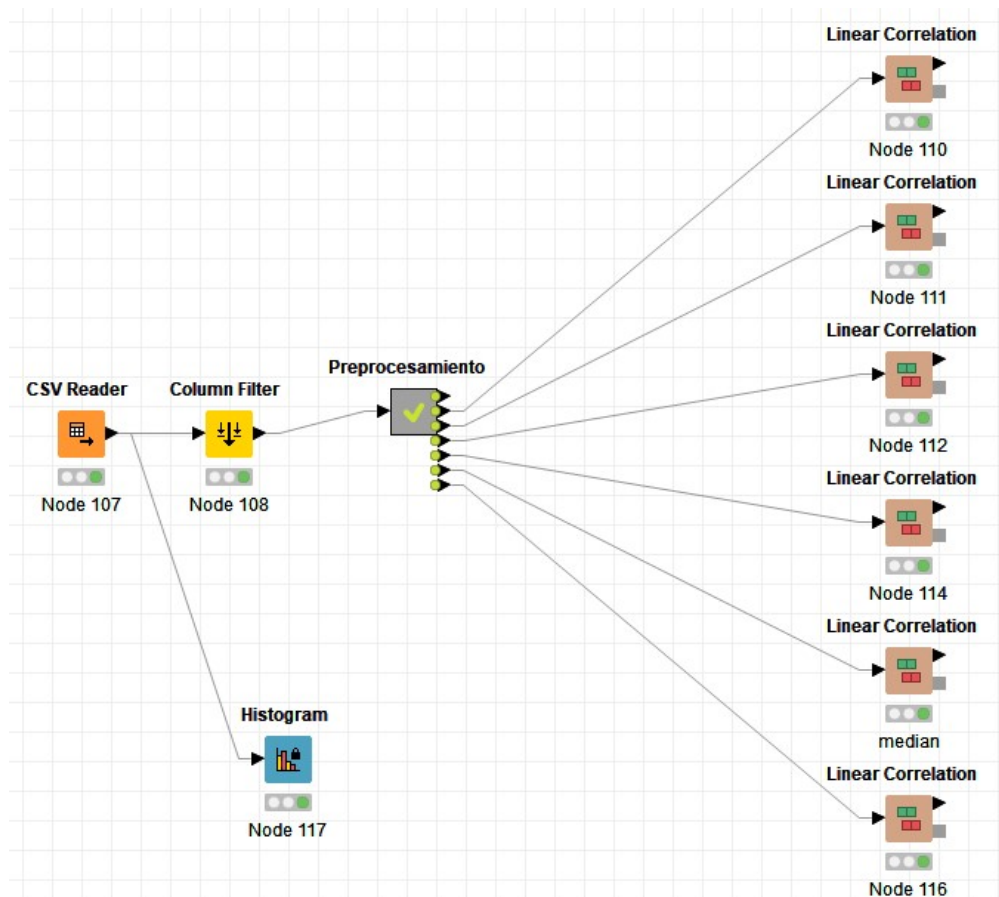


Ilustración 1 Nodo principal KNIME para analisis de datos

En el archivo que voy a subir habrá dos grupos de nodos iguales pero en uno de ellos no habrá columnas borradas para ver cómo era inicialmente la tabla de correlaciones. De ellas se han eliminado distintas variables:

1. Recorder\_by: Esta variable no cambia su valor en ningún momento. Por lo tanto no es una variable útil.
2. Id: No tiene sentido guardar esta variable ya que únicamente indica la instancia que es y no tiene ningún tipo de relación con las otras variables. Esto se pudo observar en la ilustración 2 en la cual podemos ver que la correlación de la columna id como las demás es prácticamente nula.
3. Date\_recorded (únicamente para las primeras 3 ejecuciones): Como en la variable anterior, la tener un gran número de instancias distintas hace que no haya una gran correlación con las demás variables.

4. Amount\_tsh: Esta variable la quite debido a la gran cantidad de ceros que tenía. Para esta variable no use KNIME, en este caso comprobé que cantidad de ceros tenía la variable.
5. Num\_private: La misma razón que la variable anterior. Debido a la cantidad de ceros que tiene apenas se relaciona con las demás variables.
6. Distric\_code(al igual que Date\_recored la elimine para las primeras 3 ejecuciones): En este caso al elimine debido a que era muy similar a región\_code, y además, región\_code tenía una mejor correlación con el estado de las bombas de agua.
7. wpt\_name: No era una variable que dijera nada importante. Además como podemos ver en la ilustración 2, no tenía apenas correlación con ninguna otra variable a excepción de región.
8. Región: Esta variable la quite después ya que creí que era similar a región\_code
9. Scheme\_name: Esta variable la quite por error. Debido ha parecido que tiene el nombre con Scheme\_management. Quería quitar Scheme\_management ya que tenía baja correlación tanto con el estado de las bombas, como con las demás variables.
10. Extraction\_type\_group: Era una variable duplicada de Extraction\_type.
11. Extraction\_type\_class: También era una variable duplicada de Extraction\_type.
12. management\_group: era prácticamente una variable duplicada de management (tenía un 0,8 de correlación).
13. payment\_type: Era una variable duplicada de payment.
14. water\_quality: Era una variable duplicada de quality\_group.
15. quantity\_group: Era una variable duplicada de quantity.
16. source\_class: Era una variable duplicada de source.
17. waterpoint\_type\_group: Era una variable duplicada de waterpoint\_type.

En general se puede ver la decisión de borrar variables se debe o a que haya una gran cantidad de valores que no podemos tratar ya que son la mayoría de la variable, como num\_private. O que tiene una baja correlación con las demás variables o en especial con la variable que dice el estado de las bombas de agua. O simplemente porque sea una variable duplicada.

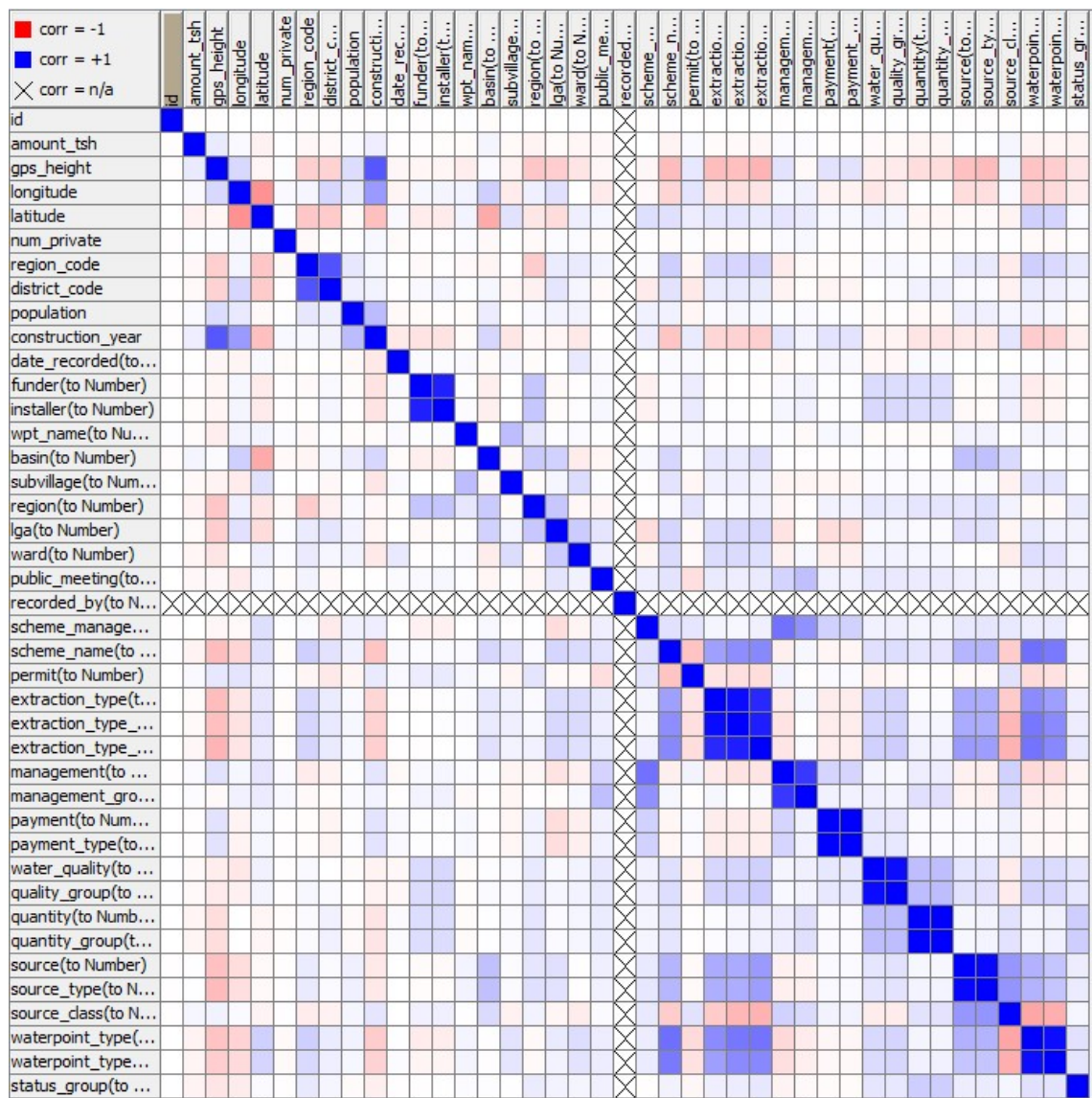


Ilustración 2 Tabla de correlaciones con todas las variables que había inicialmente en la práctica

Después de eliminar las columnas que no se van a usar he sacado las dos tablas principales por las que me he guiado. Esas tablas muestran la correlación entre las distintas variables. La principal variable en la que me he fijado ha sido en la de status\_group, que nos dice como está la bomba. Principalmente he intentado dejar las variables que tuviesen mayor correlación. También hay variables como extraction\_type que aunque la correlación no sea especialmente alta con status\_group se dejan porque si que tienen correlación alta con otras variables que son interesantes.

Las dos ilustraciones siguientes muestran la tabla de correlación que hay entre cada variable. La ilustración 3 es la tabla que elimina cualquier instancia que tenga un valor perdido. Como elimina una gran cantidad de instancias la correlación entre las variables que quedan se muestran mejor. Por ejemplo la variable population se relaciona con más variables cuando se eliminan los valores perdidos. Al final he decidido no escoger esta opción ya que los datos se quedaban en menos de la mitad (dejando únicamente 25384 instancias).



Por el otro lado está la ilustración 4 que representa la tabla de la correlación entre las variable, pero sustituyendo los valores perdidos por la media. Al final de entre todas las tablas creí que esta era la mejor opción.

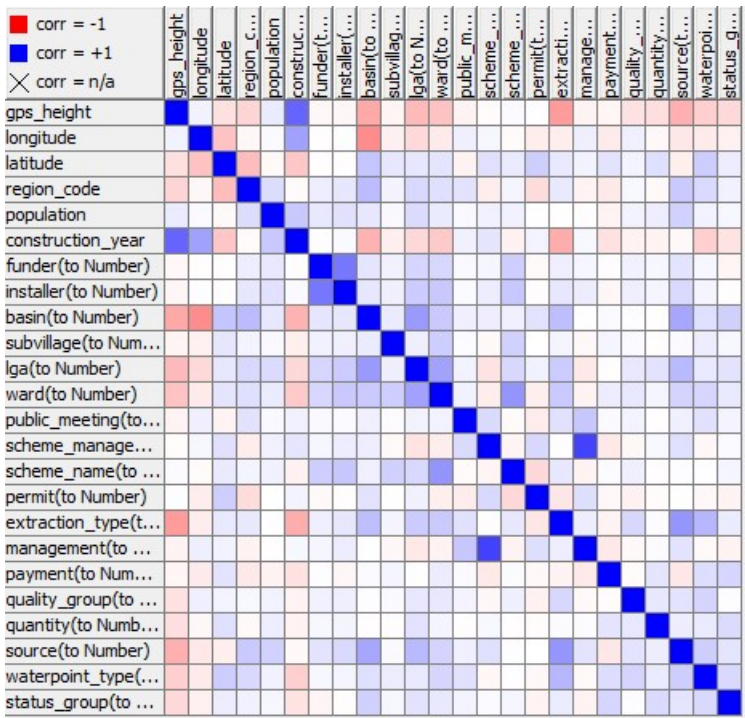


Ilustración 3 Tabla de las correlaciones quitando los valores perdidos

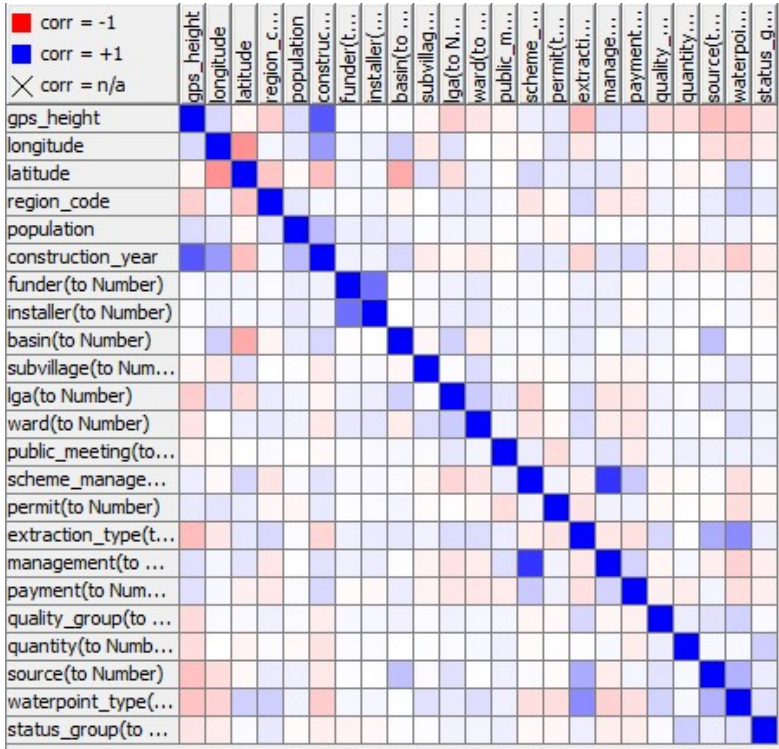


Ilustración 4 Tablas de las correlaciones poniendo los valores perdidos como la media

Ya una vez trabajando en el fichero de python para ajustar las variables para mostrar las cosas observe que podía ser interesante que a la hora de calcular la media se hiciera en vez de una general para todos, se buscara una específica por región para rellenarla de forma que sea más parecido a como es la región. Con ello busco los valores que sean '0' en las variables enteras como population o construction\_year y voy realizando las medias por región. Esto mismo se hacía a la hora de rellenar la columnas de datos del tst pero cogiendo la media de la región del train. También decidí usar la moda para las variables de tipo string, ya que la que más valores nulos tenía eran 3877 instancias como se muestra en la ilustración 5.

```
In [2]: data_x.isnull().sum()
Out[2]:
funder          3662
gps_height      0
installer       3696
longitude       0
latitude        0
basin           0
subvillage      371
region_code     0
lga             0
ward            0
population      0
public_meeting  3334
scheme_management 3877
permit          3056
construction_year 0
extraction_type 0
management      0
payment         0
quality_group   0
quantity        0
source          0
source_type     0
waterpoint_type 0
dtype: int64
```

Ilustración 5 Numero de variables nulas

Una vez escogidas la variables para la primera ejecución, tenía que escoger entres distintos algoritmos. Tanto por mis compañeros como por internet interprete que los dos algoritmos que mejores resultados estaban dando eran ramdonforest y LightGBM. Haciendo pruebas de manera local fueron los que mejor resultado me dieron. Como se muestra en la ilustración 6, los algoritmos que mejor puntuación dieron de manera local son el ramdonforest y LightGBM. Una vez escogidos ajuste los parámetros según lo que leí en internet, que decía que en general que el número de estimadores más optimo era 1000.

Después de haber ejecutado todos los algoritmos y haberlos ajustados decidí realizar las primeras subidas. Una con el ramdonForest con 1000 estimadores y otra con LightGBM con 2000 estimadores. Los resultados no fueron los esperados sacando 0.7547 en ramdonForest y 0.7202 en LightGBM con el cual deje de hacer ejecuciones después del resultado obtenido. Y debido a la hora la única modificación que realice era cambiar el número de estimadores de

ramdonForest a 300. Al ser un resultado también bajo, sacando 0.7503, decidí replantearme porque razón podía haber pasado eso.

```
----- XGB...
Score: 0.7760, tiempo: 39.37 segundos
Score: 0.7751, tiempo: 40.23 segundos
Score: 0.7718, tiempo: 39.60 segundos
Score: 0.7695, tiempo: 44.84 segundos
Score: 0.7683, tiempo: 43.51 segundos

Score: 0.7805
----- RandomForest...
Score: 0.8105, tiempo: 20.64 segundos
Score: 0.8099, tiempo: 20.60 segundos
Score: 0.8060, tiempo: 21.36 segundos
Score: 0.8075, tiempo: 21.82 segundos
Score: 0.8068, tiempo: 20.09 segundos

Score: 0.9612
----- LightGBM...
Score: 0.8109, tiempo: 18.86 segundos
Score: 0.8055, tiempo: 18.12 segundos
Score: 0.8003, tiempo: 18.13 segundos
Score: 0.8031, tiempo: 18.72 segundos
Score: 0.8029, tiempo: 17.30 segundos

Score: 0.9308
----- RandomForest n_est=300...
Score: 0.8101, tiempo: 6.02 segundos
Score: 0.8099, tiempo: 5.98 segundos
Score: 0.8055, tiempo: 6.16 segundos
Score: 0.8065, tiempo: 5.82 segundos
Score: 0.8057, tiempo: 5.73 segundos

Score: 0.9610
```

#### Ilustración 6 Prueba en local de los primeros algoritmos

La primera idea para solucionar el problema de las puntuaciones que se me ocurrió fue cambiar como se hacían las medias. A las que en vez de usar la media de las regiones del dataset de train se usaban las medias del dataset de tst. Haciendo ese cambio la puntuación mejor un poco, hasta llegar a 0.7584.

Tras este cambio decidí en primer lugar volver a revisar las variables. Entre ellas me di cuenta que inicialmente deje una de las variables duplicadas, la variable `source_type`. Y también decidí devolver la variable `district_code` ya que era una variable que tenia alta correlación y podía ser interesante.

Por otro lado encontré que en Tanzania había distintas épocas de lluvia, y de ahí se me ocurrió que podía ser interesante crear una nueva variable a través de `data_recorded`. Se puede coger la variable `data_recorded` y dividirla en días, meses y años. Los días no es una variable interesante, ya que si alguna bomba de agua está rota en especial en un día en particular se debe a una casualidad. Por otro lado tanto la variable año como la variable mes si que pueden ser interesantes. Se puede usar las variables mes para saber si en una

determinada época del año en la que ha habido muchas lluvias las pozas se han podido estropear. También por otro lado el año podía ser interesante para dos cosas. La primera al igual que en los meses, se ha podido dar el caso de que un año por cualquier tipo de problema, como una larga época de lluvias o algún problema interno del país, las bombas de agua ese año hayan tenido problemas. También se puede usar el año para calcular cuánto tiempo lleva funcionando esa bomba. Usando la variable `construction_year` con el año en el que se recogió la información se puede obtener aproximadamente el tiempo que lleva funcionando la bomba de agua. Es posible que una bomba de agua que lleve funcionando mucho tiempo se pueda romper con más facilidad. Una vez terminado de organizar las variables pasamos a realizar la penúltima puntuación la cual fue la más alta de la práctica, sacando 0,7650

Tras la última puntuación. No se me ocurrían más cambios de variables que fueran realmente interesantes. Así que con una última ejecución decidí reajustar alguno de los parámetros. En este caso al hacer distintas pruebas en local reajustando el `cross validation`, la puntuación e manera local aumentaba considerablemente. Hice la pruebas con 5, 10 y 20. Al final decidió hacer una última subida haciendo el cambio del `cross validation` 20 veces. La puntuación fue de 0.7632.

## Parámetros de los algoritmos

Para los distintos algoritmos he hecho varias pruebas de forma local para así determinar cuál podía dar mejor resultado. Los algoritmos con los que he hechos pruebas son los mencionados anteriormente. Como podemos apreciar en la ilustración 6 el `xgboost` fue directamente descartado debido a la baja puntuación de forma local.

Estas son las pruebas que hice para cada uno de los algoritmos restantes. Para todas las pruebas use un `cross validation` de 5:

-`RamdonForest`, para el `ramdonForest` el único cambio que hice fue en el número de estimadores. Poniéndolo en 300, 500 y 1000 estimadores como se muestra en la ilustración 7. Como las mejores puntuaciones fueron las de 300 y las de 1000, fueron las que subí. Como finalmente la de 300 dio una puntuación más baja que la de 1000. Las ejecuciones posteriores las deje únicamente en 1000.

-`LightGMB`, Al igual que para el `ramdonforest` el parámetro que cambie fue el del numero de estimadores. En este caso 500, 1000, 2000. Como de forma local el que mayor puntuación saco fue el de 2000. Tras ver que su puntuación era menor que la del `ramdonforest` decidí no hacer mas subidas con este algoritmo.

-`k-meas`, En este último caso las puntuaciones en local eran muy bajas como se muestra en la ilustración 9, a excepción de si el número de vecinos que miraba era 1. Pero poner un único vecino no tenía mucho sentido así que al igual que el algoritmo `Xgboost` debido a las puntuaciones decidió no hacer ninguna subida con él.

```

----- RandomForest n_est=300...
Score: 0.8101, tiempo: 5.73 segundos
Score: 0.8099, tiempo: 6.00 segundos
Score: 0.8055, tiempo: 5.80 segundos
Score: 0.8065, tiempo: 5.79 segundos
Score: 0.8057, tiempo: 5.78 segundos

Score: 0.9610
----- RandomForest n_est=500...
Score: 0.8111, tiempo: 9.56 segundos
Score: 0.8086, tiempo: 9.50 segundos
Score: 0.8056, tiempo: 9.85 segundos
Score: 0.8060, tiempo: 9.57 segundos
Score: 0.8068, tiempo: 9.61 segundos

Score: 0.9608
----- RandomForest n_est=1000...
Score: 0.8101, tiempo: 5.93 segundos
Score: 0.8099, tiempo: 5.76 segundos
Score: 0.8055, tiempo: 5.80 segundos
Score: 0.8065, tiempo: 6.02 segundos
Score: 0.8057, tiempo: 5.69 segundos

Score: 0.9610

```

Ilustración 7 Pruebas ramdonForest

```

----- LightGBM n_estimators=500...
Score: 0.8089, tiempo: 6.42 segundos
Score: 0.8057, tiempo: 6.20 segundos
Score: 0.8033, tiempo: 6.38 segundos
Score: 0.8029, tiempo: 6.25 segundos
Score: 0.8006, tiempo: 6.25 segundos

Score: 0.8592
----- LightGBM n_estimators=1000...
Score: 0.8108, tiempo: 10.99 segundos
Score: 0.8105, tiempo: 9.67 segundos
Score: 0.8040, tiempo: 9.72 segundos
Score: 0.8061, tiempo: 9.61 segundos
Score: 0.8022, tiempo: 9.58 segundos

Score: 0.8968
----- LightGBM n_estimators=2000...
Score: 0.8109, tiempo: 18.31 segundos
Score: 0.8055, tiempo: 18.23 segundos
Score: 0.8003, tiempo: 18.08 segundos
Score: 0.8031, tiempo: 18.02 segundos
Score: 0.8029, tiempo: 17.95 segundos

Score: 0.9308

```

Ilustración 8 Pruebas LightGBM



```

----- Knn n=10...
Score: 0.6451, tiempo: 0.12 segundos
Score: 0.6462, tiempo: 0.12 segundos
Score: 0.6403, tiempo: 0.11 segundos
Score: 0.6367, tiempo: 0.11 segundos
Score: 0.6480, tiempo: 0.11 segundos

Score: 0.6944
----- Knn n=5...
Score: 0.6511, tiempo: 0.11 segundos
Score: 0.6522, tiempo: 0.12 segundos
Score: 0.6492, tiempo: 0.11 segundos
Score: 0.6503, tiempo: 0.12 segundos
Score: 0.6559, tiempo: 0.11 segundos

Score: 0.7396
----- Knn n=1...
Score: 0.6555, tiempo: 0.12 segundos
Score: 0.6550, tiempo: 0.12 segundos
Score: 0.6550, tiempo: 0.12 segundos
Score: 0.6488, tiempo: 0.12 segundos
Score: 0.6532, tiempo: 0.12 segundos

Score: 0.9305

```

Ilustración 9 Prueba k-means

## Conclusión

Creo que en mi caso que para este problema en vez de dedicar tanto tiempo para realizar lo que yo consideraba un buen análisis de datos debía de haber intentado hacer más pruebas con distintas variables para así encontrar que valores podía ajustarse para dar mejores resultados.

Por otro lado también creo que las pruebas en local para esta práctica no han sido tan importantes. Y aunque las puntuaciones fuesen altas, no significaba que fuesen mejores. En general en esta práctica no se han obtenido los resultados esperados.

## Referencias

- [1] <https://community.drivendata.org/c/pump-it-up-data-mining-the-water-table>
- [2] <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>
- [3] <https://medium.com/@vaibhavshukla182/pump-it-up-data-mining-the-water-table-f903d4cfc7a8>
- [4] <https://jitpaul.blog/2017/07/12/pump-it-up/>
- [5] <https://www.tripsavvy.com/tanzania-weather-and-average-temperatures-4071465>