Inteligencia de Negocio

Miguel Morales Castillo

miguemc@correo.ugr.es
Grupo Lunes

Enero 2019

Índice

1. Tabla submissions

Submissions

BEST	CURRENT RANK	# COMPETITORS	SUBS. TODAY
0.7765	1386	6184	0 / 3

SUBMISSIONS

Score	Submitted by	Timestamp
0.7585	Miguel_Morales_UGR	2018-12-31 12:37:00 UTC
0.7721	Miguel_Morales_UGR	2018-12-31 14:52:52 UTC
0.7765	Miguel_Morales_UGR	2018-12-31 15:14:21 UTC
0.7580	Miguel_Morales_UGR	2019-01-01 06:35:10 UTC
0.7603	Miguel_Morales_UGR	2019-01-02 00:03:31 UTC
0.7669	Miguel_Morales_UGR	2019-01-02 00:17:49 UTC
0.7671	Miguel_Morales_UGR	2019-01-02 01:24:40 UTC
0.7675	Miguel_Morales_UGR	2019-01-03 00:47:40 UTC
0.7531	Miguel_Morales_UGR	2019-01-03 02:00:14 UTC
0.7523	Miguel_Morales_UGR	2019-01-04 01:45:22 UTC
0.7666	Miguel_Morales_UGR	2019-01-05 01:33:02 UTC
0.7653	Miguel_Morales_UGR	2019-01-05 01:33:20 UTC
0.7661	Miguel_Morales_UGR	2019-01-05 03:13:48 UTC

2. Introducción

La práctica consiste en aplicar minería de datos con los datos de Taarifa y el Ministerio de Agua de Tanzania, el objetivo es predecir qué bombas de extracción de agua funcionan, cuáles necesitan algunas reparaciones y cuáles no funcionan.
Para predecir una de estas tres clases se dispone de 39 variables (numéricas y categóricas) sobre qué tipo de bomba se está usando, cuándo se instaló, cómo se administra, su ubicación, datos sobre la cuenca geográfica, tipo de extracción, coste del agua, etc. Una comprensión inteligente de qué puntos de agua fallarán puede mejorar las operaciones de mantenimiento y garantizar que el agua potable y limpia esté disponible para las comunidades de Tanzania. El conjunto de entrenamiento consta de 59.400 instancias. En esta competición, se emplea el porcentaje de acierto como score para valorar la calidad de la solución aportada.

3. Desarrollo del experimento

Para este experimento hemos seleccionado 4 algoritmos con los que trabajar e intentar obtener resultados, estos algoritmos son LGBM, XGBoosted, Gradient Boosted y Random Forest. En el comienzo, trabajamos con el preprocesado mínimo necesario para que los algoritmos funcionen, ejecutamos cada uno y la primera subida la realizaremos usando LGBM con 200 estimadores como punto de partida. Como veremos mas adelante en la tabla los resultados no son malos, pero tampoco especialmente buenos, por lo que probamos con Random Forest, con 1000 estimadores y los demás parámetros por defecto, obteniendo el segundo resultado de la tabla.

A partir de aqui, introduciremos para todos los experimentos el mismo preprocesado, que explicaremos a continuación:

3.1. Preprocesado para el experimento

Para el preprocesado nos hemos basado en el método groupBy() que nos permite agrupar las instancias según los atributos que le pasamos como parámetro, de esta forma podemos eliminar aquellos atributos que sean "iguales" o que expresen la misma circunstancia, un ejemplo de el uso de esta función lo vemos en la siguiente imagen:

```
source               source_type          source_class
dam                  dam                  surface           656
hand dtw             borehole             groundwater       874
lake                 river/lake           surface           765
machine dbh          borehole             groundwater     11075
other                other                unknown           212
rainwater harvesting rainwater harvesting surface          2295
river                river/lake           surface           9612
shallow well         shallow well         groundwater     16824
spring               spring               groundwater     17021
unknown              other                unknown             66
dtype: int64
```

Como podemos observar, aplicando groupBy sobre source, source_type y source_class vemos que las tres tienen opciones iguales o muy similares, por lo que podemos quitar dos de ellas.Siguiendo el mismo procedimiento, eliminamos los atributos:
(date_recorded, wpt_name, num_private, subvillage, region_code, recorded_by, management_group, source_type, source_class, extraction_type_group, extraction_type_class, scheme_name, payment_type, quality_group, quantity_group, waterpoint_type_group, installer, public_meeting, permit)

Una vez eliminados estos atributos volvemos a aplicar Random Forest con 1000 estimadores y el resto de valores por defecto, y obtenemos nuestro mejor resultado obtenido en todo el experimento.

Tras este intento hemos intentado mejorar el resultado en un principio aplicando clasificador por voto combinando los algoritmos seleccionados, lamentablemente sin mucho éxito, a continuación, aplicamos grid search a los 2 algoritmos mas prometedores, Random Forest y LGBM, estas son las llamadas a dicha función y su salida.

```
print("Empieza Lgbm")
params_lgbm = {'learning_rate':[i/100 for i in range(5,70,5) ],'num_leaves':[30,50,80], 'n_estimators':[100,200,500]}
grid1 = GridSearchCV(clf4, params_lgbm, cv=3, n_jobs=1, verbose=1, scoring=make_scorer(accuracy_score))
grid1.fit(X,y)
grid1=test_less_train(grid1,X,X_tst,y)
print("Mejores parámetros grid 1:")
print(grid1.best_params_)


print("Empieza RF")
params_rf = {'n_estimators':[200,500,1000],'min_samples_leaf':[5,30,50], 'criterion':['gini','entropy']}
grid2 = GridSearchCV(clf2, params_rf, cv=3, n_jobs=1, verbose=1, scoring=make_scorer(accuracy_score))
grid2.fit(X,y)
grid2=test_less_train(grid2,X,X_tst,y)
print("Mejores parámetros grid 2:")
print(grid2.best_params_)

print("Empieza RF")
params_rf = {'n_estimators':[200,300,500,700],'min_samples_leaf':[5,15,30,50], 'criterion':['gini','entropy']}
grid2 = GridSearchCV(clf2, params_rf, cv=5, n_jobs=1, verbose=1, scoring=make_scorer(accuracy_score))
grid2.fit(X,y)
grid2=test_less_train(grid2,X,X_tst,y)
print("Mejores parámetros grid 2:")
print(grid2.best_params_)'''
```

```
Empieza Lgbm
Fitting 3 folds for each of 117 candidates, totalling 351 fits
[Parallel(n_jobs=1)]: Done 351 out of 351 | elapsed: 65.1min finished
Mejores parámetros grid 1:
{'learning_rate': 0.15, 'n_estimators': 200, 'num_leaves': 80}
Empieza RF
Fitting 3 folds for each of 18 candidates, totalling 54 fits
[Parallel(n_jobs=1)]: Done  54 out of  54 | elapsed: 39.0min finished
Mejores parámetros grid 2:
{'criterion': 'entropy', 'min_samples_leaf': 5, 'n_estimators': 500}

Fitting 5 folds for each of 32 candidates, totalling 160 fits
[Parallel(n_jobs=1)]: Done 160 out of 160 | elapsed: 176.7min finished
Mejores parámetros grid 2:
{'criterion': 'gini', 'min_samples_leaf': 5, 'n_estimators': 300}
```

Tras probar los algoritmos con estos parámetros, obtuvimos resultados similares al mejor pero un poco por debajo. A continuación mostramos la tabla resumen de las subidas a Driven Data. Omitiremos el apartado de preprocesamiento ya que se ha comentado en la sección anterior.

Score	Posicion	Fecha y hora	ScoreTrain	Algoritmo	Parámetros
0.7585	1200	2018-12-31 12:37:00 UTC	0.82	LGBM	200 estimadores
0.7721	1185	2018-12-31 14:52:52 UTC	0.856	RF	1000 estimadores
0.7765	1114	2018-12-31 15:14:21 UTC	0.87	RF	1000 estimadores
0.7580	1364	2019-01-01 6:35:10 UTC	0.872	Voto	LGBM, RF, XGB (250 estimadores LGBM, XGB y 1000 RF)
0.7603	1364	2019-01-02 00:03:31 UTC	0.884	Voto	LGBM, RF, XGB (300 estimadores LGBM, XGB y 800 RF)
0.7669	1364	2019-01-02 00:17:49 UTC	0.883	RF	1000 estimadores, min_samples_leaf=5
0.7671	1364	2019-01-02 1:24:40 UTC	0.879	RF	1000 estimadores, min_samples_leaf=30
0.7675	1364	2019-01-03 00:47:40 UTC	0.893	RF	800 estimadores, min_samples_leaf=15
0.7531	1364	2019-01-03 02:00:14 UTC	0.92	RF	800 estimadores, min_samples_leaf=50
0.7523	1364	2019-01-04 01:45:22 UTC	0.91	RF	1000 estimadores, min_samples_leaf=50
0.7666	1364	2019-01-05 01:33:02 UTC	0.884	RF	criterion=entropy, min_samples_leaf=5,n_estimators=500
0.7653	1386	2019-01-05 01:33:20 UTC	0.876	LGBM	learning_rate=0.15, n_estimators=200,num_leaves= 80
0.7661	1386	2019-01-05 03:13:48 UTC	0.885	Voto	LGBM, RF1, RF2: (LGBM y RF1 son los dos anteriores, siguientes parámetros: criterion=gini, min_samples_leaf=5, n_estimators=300