



UNIVERSIDAD DE GRANADA

Práctica 3

Inteligencia de negocio

Juan Carlos González Quesada
UGR

Posición final en Kaggle:

☰

kaggle

🏠 Home

🏆 Compete

📊 Data

📓 Notebooks

🗨 Communities

🎓 Courses

⌵ More

Recently Viewed

📊

Predicción de precio c...

📊

MLPClassifier with Gri...

📊

[Fraud Detection] Grad...

📊

GradientBoostingClass...

📊

GradientBoostingClass...

🔍 Search

Overview

Data

Notebooks

Discussion

Leaderboard

Rules

Team

My Submissions

Late Submission

#	△pub	Team Name	Notebook	Team Members	Score 🏆	Entries	Last
1	—	JuanHeliosGarcía			0.83002	15	3d
2	—	PATRICIA CORDOBA 77145053			0.82830	13	3d
3	—	José Alberto García 26513007X			0.82484	43	1d
4	—	Alvaro de Rada 49108766V			0.81622	14	20h
5	—	DAVID CABEZAS 20079906			0.81622	34	2d
6	—	OctavioTorres			0.81622	23	19h
7	—	AlejandroAlonso75577394S			0.81363	10	2d
8	—	Mikhail Raudin 531101855			0.80845	11	19h
9	—	Javier Rodríguez 78306251Z			0.80759	12	20h
10	—	David Martín 75931868J			0.80586	28	1d
11	—	JuanCarlosGonQu			0.79982	54	1d
12	—	Pedro Jiménez 76592485R			0.79810	35	1d
13	—	Ilias_Amar_Ceuta			0.79723	15	1d
14	—	Jose Antonio Martín 77561280J			0.79551	14	1d
15	—	Alberto_Postigo_Ceuta			0.79206	21	2d

Para cada subida, indico en que archivo IPYNB se encuentra el código:

Intento n^o X Prueba Y

La subida del intento número X se encuentra en el notebook llamado pruebaY.ipynb.

Contenido

Introducción.....	4
Tabla de resultados.....	4
Subidas	8
Subidas iniciales sin superar el valor de referencia	8
Ajuste de parámetros.....	8
Intento nº 4 Prueba 2.....	9
Intento nº 5 Prueba 2.....	9
Intento nº 6 Prueba 2.....	9
Intento nº 7 Prueba 2.....	9
Intento nº 8 Prueba 2.....	10
Intento nº 9 Prueba 2.....	10
Intento nº 10 Prueba 3.....	10
Intento nº 11 Prueba 3.....	11
Intento nº 12 Prueba 3.....	11
Intento nº 13 Prueba 3.....	11
Intento nº 14 Prueba 4.....	12
Intento nº 15 Prueba 4.....	12
Intento nº 16 Prueba 4.....	12
Intento nº 17 Prueba 4.....	12
Intento nº 18 Prueba 4.....	12
Intento nº 19 Prueba 5.....	12
Intento nº 20 Prueba 6.....	13
Intento nº 21 Prueba 6.....	13
Intento nº 22 Prueba 6.....	14
Intento nº 23 Prueba 6.....	14
Intento nº 24 Prueba 6.....	14
Intento nº 25 Prueba 6.....	14
Intento nº 26 Prueba 6.....	14
Intento nº 27 Prueba 6.....	15
Intento nº 28 Prueba 7.....	15
Intento nº 29 Prueba 7.....	15
Intento nº 30 Prueba 7.....	15
Intento nº 31 Prueba 7.....	15
Intento nº 32 Prueba 7.....	15
Intento nº 33 Prueba 7.....	15

Intento n ^o 34 Prueba 7	16
Intento n ^o 35 Prueba 7	16
Intento n ^o 36 Prueba 9	16
Intento n ^o 37 Prueba 9	16
Intento n ^o 38 Prueba 9	17
Intento n ^o 39 Prueba 11	17
Intento n ^o 40 Prueba 16	18
Intento n ^o 41 Prueba 16	19
Intento n ^o 42 Prueba 16	19
Intento n ^o 43 Prueba 17	19
Intento n ^o 44 Prueba 17	19
Intento n ^o 45 Prueba 19	19
Intento n ^o 46 Prueba 19	20
Intento n ^o 47 Prueba 19	20
Intento n ^o 48 Prueba 20	20
Intento n ^o 49 Prueba 21	20
Intento n ^o 50 Prueba 21	20
Intento n ^o 51 Prueba 24	21
Intento n ^o 52 Prueba 24	21
Intento n ^o 53	21
Intento n ^o 54 Prueba 31	22
Anexo adicional	23
Intento con cambio de codificador	23
Uso de StackingClassifier con distintos modelos y datos	23
Conclusión	24
Bibliografía	25

Introducción

La práctica 3 de inteligencia de negocio se basa en una competición dentro de la plataforma *Kaggle*. Tenemos un conjunto de datos que representan características sobre coches. El objetivo para predecir será el precio de los coches que podrán tomar los valores 1, 2, 3, 4 o 5. Es un problema con clases desbalanceadas.

Tabla de resultados

	Fecha	Hora	Posición	Score_Local	Score_Kaggle	Descripción	Algoritmo	Parámetros
1	10-12	18:35	1	65.87	0.23727	Sustituyo los valores perdidos por el más frecuente	Nayve-Bayes Gaussiano	Ninguno ya que no admite
2	10-12	19:02	1	74.02	0.33735	Lo mismo que la anterior, pero cambio el algoritmo	DecisionTreeClassifier	Ninguno, ya que es así es como mejor resultados me dio en la práctica 1
3	14-12	10:34	1	79.46	0.53062	Lo mismo que la anterior, pero cambio el algoritmo	RandomForest	n_estimators=10, max_depth=None,min_samples_split=2, random_state=0
4	15-12	11:32	1	76.24	0.73770	Me doy cuenta de que tengo un error al realizar las etiquetas, que me está provocando que no se ajuste bien a la plataforma Kaggle.	RandomForest	n_estimators=10, max_depth=None,min_samples_split=2, random_state=0
5	15-12	12:10	1	82.51	0.78343	Utilizo un nuevo algoritmo que he encontrado en un artículo sobre Machine Learning	LGBMClassifier	objective='regression_l1',n_estimators=200,n_jobs=2
6	15-12	12:25	1	74.02	0.71958	Reíso de nuevo el Árbol de decision para comprobar que da mejores resultados	DecisionTreeClassifier	No tiene parámetros
7	16-12	16:54	1	81.79	0.77308	Utilizo un nuevo algoritmo que he encontrado en otro artículo sobre Machine Learning	XGBClassifier	max_depth=15, n_estimators = 2000, learning_rate=0.05, n_jobs=2, num_leaves=30, max_bin=300
8	16-12	17:15	1	80.98	0.77308	Utilizo una mezcla entre RandomForest y Bagging Elimino también al atributo Descuentos.	Bagging (RandomForest ())	A Random forest le añado: n_estimators=10, max_depth=None,min_samples_split=2, random_state=0 A bagging sólo le añado la base del random forest
9	16-12	19:23	1	82.03	0.78257	Utilizo el algoritmo de LGBMClassifier con un cambio en los parámetros	LGBMClassifier	objective='regression_l1',n_estimators=180,n_jobs=2
10	17-12	11:13	1	82.81	0.76100	Aplico la técnica de OverSampling	Bagging (LGBMClassifier ())	A LGBMClassifier le añado: objective='multiclassova',n_estimators=200,n_jobs=2, num_leaves=40, learning_rate=0.05 A Bagging solo le asigno el algoritmo lgbm como base_estimator
11	17-12	12:25	1	83.20	0.77308	Aplico la técnica de UnderSampling	LGBMClassifier	A LGBMClassifier le añado: objective='multiclassova',n_estimators=200,n_jobs=2, num_leaves=40, learning_rate=0.099
12	17-12	13:01	1	82.71	0.77911	Aplico la técnica de UnderSampling	Bagging(LGBMClassifier ())	objective='multiclassova',n_estimators=200,n_jobs=2, num_leaves=40, learning_rate=0.099 A Bagging solo le asigno el algoritmo lgbm como base_estimator
13	18-12	17:24	1	82.81	0.77911	Aplico la técnica de UnderSampling y modifico parámetros	Bagging(LGBMClassifier ())	A LGBMClassifier le añado: objective='regression_l1',n_estimators=200,n_jobs=2, num_leaves=40, learning_rate=0.09 A Bagging solo le asigno el algoritmo lgbm como base_estimator
14	18-12	17:50	1	83.21	0.77911	Añado la columna de Tipo_marchas ya que nos han incorporado su csv.	LGBMClassifier	objective='regression_l1',n_estimators=200,n_jobs=2, num_leaves=40, learning_rate=0.09

15	18-12	17:59	1	82.95	0.78343	Lo mismo que el anterior, pero tras un previo estudio me doy cuenta de que bajando el learning_rate, obtengo mejor score local.	LGBMClassifier	objective='regression_l1',n_estimators=200,n_jobs=2, num_leaves=40, learning_rate=0.03
16	19-12	18:43	1	82.87	0.78515	Empiezo a ajustar parámetros	LGBMClassifier	objective='regression_l1',n_estimators=300, n_jobs=2, learning_rate=0.09
17	19-12	19:10	2	82.14	0.77739	Cambio de algoritmo a uno similar	XGBClassifier	max_depth=10, n_estimators = 200, n_jobs=-1
18	20-12	20:11	2	83.05	0.78861	Ajuste de parámetros	LGBMClassifier	objective='regression_l1',n_estimators=200, n_jobs=2, learning_rate=0.07
19	21-12	12:15	3	82.24	0.78688	Elimino el atributo Ciudad	LGBMClassifier	n_estimators=200, num_leaves=35, scale_pos_weight=0.1
20	21-12	16:54	3	82.62	0.78257	Añado el atributo Ciudad y hago un nuevo ajuste de parámetros	LGBMClassifier	objective='regression_l1',n_estimators=180, n_jobs=2, learning_rate=0.07, num_leaves=35
21	21-12	17:12	4	81.91	0.76358	Aplico de nuevo la técnica de UnderSampling	Bagging (RandomForest ())	A RandomForest no le añado ningún parámetro A Bagging le añado: n_estimators=180, n_jobs=4
22	22-12	19:15	4	83.55	0.79119	Descubro que mi problema se debe a un overfitting y empiezo a trabajar con el número máximo de profundidad.	LGBMClassifier	learning_rate=0.06, objective='regression_l1', n_estimators=410, n_jobs=2, num_leaves=10, max_depth=5
23	22-12	20:11	4	83.75	0.79292	Sigo trabajando con la profundidad	LGBMClassifier	learning_rate=0.06, objective='regression_l1', n_estimators=420, n_jobs=2, num_leaves=10, max_depth=5
24	22-12	20:35	5	83.75	0.77480	Aplicando UnderSampling	LGBMClassifier	learning_rate=0.06, objective='regression_l1', n_estimators=420, n_jobs=2, num_leaves=10, max_depth=5
25	23-12	13:12	5	83.37	0.79810	Ajusto la estimación, la profundidad y el learning_rate	LGBMClassifier	learning_rate=0.055, objective='binary', n_estimators=460, n_jobs=2, num_leaves=10, max_depth=-1,seed=46000
26	23-12	13:20	5	83.37	0.79723	Ajusto el número de interacciones	LGBMClassifier	learning_rate=0.045, objective='binary', n_estimators=400, n_jobs=2, num_leaves=10, max_depth=-1,seed=46000
27	23-12	13:34	5	83.54	0.79896	Ajusto el número de interacciones	LGBMClassifier	learning_rate=0.055, objective='binary', n_estimators=450, n_jobs=2, num_leaves=10, max_depth=-1,seed=46000
28	24-12	12:01	5	82.87	0.79206	Sin Nombre ni Asientos	LGBMClassifier	objective='regression_l1',n_estimators=300, n_jobs=2, learning_rate=0.04
29	24-12	12:30	5	82.34	0.79119	Elimino la columna Asientos	LGBMClassifier	objective='regression_l1',n_estimators=300, n_jobs=2, learning_rate=0.04
30	24-12	12:54	5	83.14	0.79723	Elimino la columna Nombre	LGBMClassifier	objective='regression_l1',n_estimators=300, n_jobs=2, learning_rate=0.04
31	25-12	10:10	5	83.63	0.79119	Cambio en LGBM el regression_l1 por binary	LGBMClassifier	learning_rate=0.055, objective='binary', n_estimators=460, n_jobs=2, num_leaves=10, max_depth=-1,seed=46000
32	25-12	10:14	5	81.65	0.78429	LGBM con binary y oversampling	LGBMClassifier	learning_rate=0.055, objective='binary', n_estimators=460, n_jobs=2, num_leaves=10, max_depth=-1,seed=46000
33	25-12	11:56	5	83.14	0.78947	LGMB con binary, Learning y oversampling	LGBMClassifier	learning_rate=0.06, objective='binary', n_estimators=460, n_jobs=2, num_leaves=10, max_depth=-1,seed=46000
34	26-12	9:32	5	83.43	0.79206	Ajuste de parámetros correspondientes con el valor mayor obtenido	LGBMClassifier	learning_rate=0.055, objective='binary', n_estimators=455, n_jobs=2, num_leaves=10, max_depth=-1,seed=46000
35	26-12	9:45	5	83.57	0.78688	Ajuste de parámetros correspondientes con el valor mayor obtenido	LGBMClassifier	learning_rate=0.06, objective='binary', n_estimators=455, n_jobs=2, num_leaves=10, max_depth=-1,seed=46000
36	26-12	10:05	5	82.56	0.79292	uso del algoritmo stackingClassifier con LGBM y RandomForest	StackingClassifier(LGBM y RandomForest)	LGBMClassifier(learning_rate=0.06, objective='binary', n_estimators=350, n_jobs=2, num_leaves=10, max_depth=-1,seed=46000) RandomForestClassifier(n_estimators = 300, criterion = "gini", max_depth = 10, max_features = "auto", min_samples_leaf =0.005,min_samples_split = 0.005, n_jobs = -1, random_state = 1000)

37	27-12	01:01	5	83.86	0.00000	uso del algoritmo stackingClassifier con SVM, RandomForest y LGBM	StackingClassifier(LGBM , SVC y RandomForest)	LGBMClassifier(learning_rate=0.06, objective='binary', n_estimators=350, n_jobs=2, num_leaves=10, max_depth=-1, seed=46000) RandomForestClassifier(n_estimators = 300, criterion = "gini", max_depth = 10, max_features = "auto", min_samples_leaf = 0.005, min_samples_split = 0.005, n_jobs = -1, random_state = 1000) SVC(C = 50, degree = 1, gamma = "auto", kernel = "rbf", probability = True)
38	27-12	01:05	5	83.80	0.79465	uso del algoritmo stackingClassifier con RandomForest	StackingClassifier(LGBM y RandomForest)	RandomForestClassifier(n_estimators = 300, criterion = "gini", max_depth = 2, min_samples_leaf = 0.005, min_samples_split = 0.005, n_jobs = -1, random_state = 1000) LGBMClassifier(learning_rate=0.055, objective='binary', n_estimators=460, n_jobs=2, num_leaves=10, max_depth=-1, seed=46000)
39	27-12	01:10	5	83.12	0.78084	Usamos un nuevo algoritmo	GradientClassifier	n_estimators=400, learning_rate=0.2
40	28-12	01:03	5	83.69	0.78774	Aplico un pre-procesado de normalización y no uso etiquetas para las variables Kilómetros, motor y potencia	LGBMClassifier	learning_rate=0.06, objective='binary', n_estimators=350, n_jobs=2, num_leaves=10, max_depth=-1, seed=46000
41	28-12	01:15	5	83.42	0.79206	Aplico un pre-procesado de normalización, pero uso las etiquetas del profesor.	LGBMClassifier	learning_rate=0.06, objective='binary', n_estimators=350, n_jobs=2, num_leaves=10, max_depth=-1, seed=46000
42	28-12	01:20	5	82.31	0.79292	Uso de nuevo el algoritmo con el que obtuve los mejores resultados y normalización de los datos	LGBMClassifier	learning_rate=0.055, objective='binary', n_estimators=450, n_jobs=2, num_leaves=10, max_depth=-1, seed=46000
43	29-12	9:26	6	92.75	0.78429	Uso de nuevo stakingClassifier con lgbm SVC y RandomForest con los datos normalizados y uso de la técnica de OverSampling	StackingClassifier(LGBM , SVC y RandomForest)	LGBMClassifier(learning_rate=0.055, objective='binary', n_estimators=740, n_jobs=2, num_leaves=12, max_depth=-1, seed=46000) RandomForestClassifier(n_estimators = 300, criterion = "gini", max_depth = 10, max_features = "auto", min_samples_leaf = 0.005, min_samples_split = 0.005, n_jobs = -1, random_state = 1000) SVC(C = 50, degree = 1, gamma = "auto", kernel = "rbf", probability = True) StackingCVClassifier(classifiers = [classifier1, classifier2, classifier3], shuffle = False, use_proba = True, cv = 5, meta_classifier = SVC(probability = True))
44	29-12	9:30	6	92.22	0.79723	Uso de nuevo del algoritmo lgbm con los datos normalizados y uso de la técnica de OverSampling	LGBMClassifier	learning_rate=0.055, objective='binary', n_estimators=740, n_jobs=2, num_leaves=12, max_depth=-1, seed=46000
45	29-12	9:45	6	92.37	0.79986	Uso de nuevo del algoritmo lgbm con los datos normalizados y uso de la técnica de OverSampling	LGBMClassifier	learning_rate=0.055, objective='binary', n_estimators=640, n_jobs=2, num_leaves=20, max_depth=-1, seed=46000
46	30-12	01:02	7	92.42	0.79033	Ajuste de parámetros con los datos normalizados y uso de la técnica de OverSampling	LGBMClassifier	learning_rate=0.055, objective='binary', n_estimators=640, n_jobs=2, num_leaves=22, max_depth=-1, seed=46000
47	30-12	01:10	7	92.72	0.79292	Ajuste de parámetros con los datos normalizados y uso de la técnica de OverSampling	LGBMClassifier	learning_rate=0.055, objective='binary', n_estimators=740, n_jobs=2, num_leaves=12, max_depth=-1, seed=46000
48	30-12	01:16	7	92.54	0.78774	Ajuste de parámetros con los datos normalizados y uso de la técnica de OverSampling	LGBMClassifier	learning_rate=0.054, objective='binary', n_estimators=620, n_jobs=2, num_leaves=22, max_depth=-1, seed=46000

49	31-12	01:05	8	83.77	0.78861	Ajuste de parámetros y sin la normalización de los datos	LGBMClassifier	learning_rate=0.055, objective='binary', n_estimators=440, n_jobs=2, num_leaves=11, max_depth=-1, reg_alpha=0.1
50	31-12	01:12	8	83.42	0.78429	Ajuste de parámetros y normalización de los datos	LGBMClassifier	learning_rate=0.055, objective='binary', n_estimators=440, n_jobs=2, num_leaves=11, max_depth=-1, reg_alpha=0.9
51	31-12	12:48	9	89.56	0.79292	Uso de nuevo del algoritmo de GradientClassifier con la técnica de OverSampling y el uso del randomState y eliminando el atributo Kilometros	GradientBoostingClassifier	learning_rate=0.9, n_estimators=750, max_depth=2
52	1-1	1:50	10	91.84	0.78774	Uso de nuevo del algoritmo de GradientClassifier con la técnica de OverSampling y el uso del randomState y eliminando el atributo Kilómetros con un nuevo ajuste de parámetros	GradientBoostingClassifier	learning_rate=0.5, n_estimators=70, max_depth=5, random_state=42
53	1-1	2:10	10	83.16	0.77653	Utilizo LGBMClassifier con un nuevo ajuste de parámetros	LGBMClassifier	learning_rate=0.3, objective=' multi:softmax', n_estimators=70, n_jobs=2, num_leaves=10, max_depth=6
54	1-1	16:59	10	84.03	0.79119	Utilizo LGBMClassifier sin kilómetros y con un nuevo parámetro ajustado	LGBMClassifier	random_state=0, objetive='multi:softmax', silent=True, colsample_bytree = 0.6, learning_rate= 0.1, max_depth = 10, n_estimators=70, scales_pos_weight=0.8, subsample=0.6

Subidas

Subidas iniciales sin superar el valor de referencia

Las primeras subidas las hice para poder tener el esquema de la práctica hecho. Por tanto, realicé Intentos basándome en mis algoritmos de la primera práctica. Las dos primeras subidas me dieron error porque no tenía la estructura del csv igual, y la segunda porque tenía 115 filas y debe haber 1159. Estas subidas se encuentran dentro del archivo “Intento1.ipynb”

- Primer intento Nayve-Bayes

Una vez que ya arreglé los csv, puede realizar la primera subida con score en Kaggle probando el algoritmo de *Nayve-Bayes* con el subtipo *Gaussiano*. De forma local me salía un score del 72.12.% que para empezar no estaba mal, sin embargo, cuando lo subí se me quedó en un 0.23727. Decidí que quizás este algoritmo no servía para este problema.

- Segundo intento DecisionTreeClassifier

El segundo algoritmo fue un árbol de decisión debido a que leyendo en artículos he visto que suele dar buenos resultados. Aunque superó el resultado del algoritmo anterior, no solucionó los problemas de un score en *Kaggle* bajo.

- Tercer intento RandomForest

El tercer algoritmo fue un árbol de decisión debido a que leyendo en artículos ¹he visto que suele dar buenos resultados. Aunque superó de nuevo el resultado del algoritmo anterior, no solucionó los problemas de un score en *Kaggle* bajo.

Ajuste de parámetros

Para realizar el ajuste, me basaré en dos técnicas, la de Intento y error, y con la ayuda del algoritmo *GridSearch*, con el que obtengo los mejores valores para cada parámetro dado un rango.

Un ejemplo con *GridSearch* para el algoritmo *XGBClassifier* sería:

```
parametros={
    'max_depth':range(2,10,2),
    'n_estimators':range(20,80,10),
    'learning_rate': [0.5,0.15,0.3,0.5,0.7]
}
from sklearn.model_selection import GridSearchCV
xgbModel = xgb.XGBClassifier(random_state=0)

gsearch = GridSearchCV(xgbModel, param_grid=parametros, scoring='accuracy', n_jobs=-1, cv=5)
gsearch.fit(data_trainCopia, target_train)

print(gsearch.best_params_)
print(gsearch.best_score_)
```

¹ <https://www.aprendemachinelearning.com/random-forest-el-poder-del-ensamble/>

Se establecen los tres parámetros que quiero estudiar, y le doy un rango de valores. Luego consulto los mejores parámetros junto al score local que obtengo:

```
{'learning_rate': 0.5, 'max_depth': 4, 'n_estimators': 70}  
0.8384247129942877
```

Intento nº 4 Prueba 2

Me di cuenta de que en el `data_train` estaba aplicando un etiquetado diferente al utilizado por el `data_test`, eso es lo que me estaba haciendo tener un score tan bajo en la plataforma. Por tanto, lo arreglé leyendo los csv que el profesor nos proporcionó y asignando las etiquetas a cada columna, sin embargo, para el atributo `Tipo_marchas` seguí usando el anterior procedimiento porque no hay csv asociado.

El algoritmo que utilizo es el *RandomForest* con los mismos parámetros que para el ejemplo anterior, y aunque el score local bajó, en la plataforma Kaggle saqué un 0.73, superando en 10 puntos el valor de referencia.

Intento nº 5 Prueba 2

Para la Intento de este nuevo algoritmo que es el que mejores resultados nos dará de cara al futuro nos basamos en otro informe² sobre “*Machine Learning*”. Será el uso del algoritmo *LGBMClassifier*, al que le asigno unos parámetros similares a los que el autor del blog le otorga. Cuando subo la plataforma su predicción, nos da un score 5 puntos superior al obtenido anteriormente.

Intento nº 6 Prueba 2

Reviso con el *DecisionTreeClassifier* que efectivamente su mal score era debido a un mal etiquetado. El resultado es un 0.71 en Kaggle, lo que supone una subida de 20 puntos con respecto a su primera ejecución.

Intento nº 7 Prueba 2

Para esta subida, uso otro algoritmo del que me documenté siguiendo el hilo del anterior blog³ citado. Será un algoritmo similar al *LGBM*, que se llama *XGBClassifier*. De nuevo, volveré a asignarle los parámetros que el autor recomienda. Nos da un score alto, pero no supera al *lgbm*.

² <https://dev.to/ruizleandro/introduccion-a-lightgbm-ij7>

³ <https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/>

Intento nº 8 Prueba 2

Leyendo otro artículo⁴ descubro que se pueden utilizar la unión de dos algoritmos para abordar este problema. Para ello usaré *Bagging* y *RandomForest*:

```
bagging = BaggingClassifier(RandomForestClassifier(n_estimators=10, max_depth=None, min_samples_split=2, random_state=0))
```

Parametrización de los algoritmos

Sin embargo, no obtengo un resultado relevante en *Kaggle*, por lo que decido dejar a un lado el uso de este tipo de algoritmos.

Intento nº 9 Prueba 2

Leyendo en una página web⁵, veo un ajuste, bajar el número de estimadores, que puede ayudar a *LGBMClassifier* a obtener mejores resultados, pero no supera nuestro 0.78343.

Es a partir de este momento, cuando decido usar las técnicas para tratar las clases desbalanceadas como es nuestro caso. Primeramente, probaré la técnica de *OverSampling* aunque tiene un inconveniente que es añadir datos fabricados sin saber si son correctos.

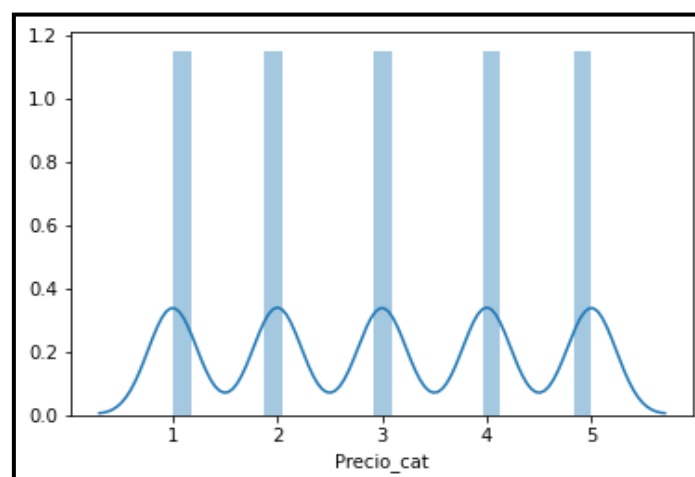
Intento nº 10 Prueba 3

Aplico a los datos *OverSampling* de la siguiente manera:

```
Xo, yo = SMOTE().fit_resample(data_trainCopia, target_train)
```

Aplicando la técnica de OverSampling

Las clases quedarían de esta forma:



Clases tras el OverSampling

⁴ <https://www.iartificial.net/random-forest-bosque-aleatorio/>

⁵ <https://lightgbm.readthedocs.io/en/latest/Parameters.html>

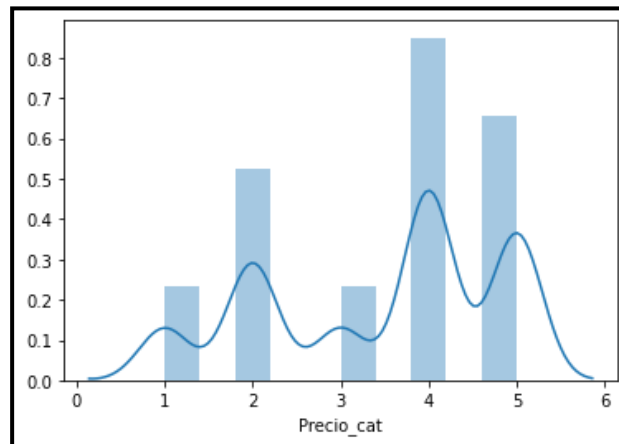
Utilizo los algoritmos que usé en la quinta Intento. Lejos de mejorar, conseguimos la peor Intento desde que empezamos a probar con algoritmos que parecen mejorar los resultados.

Intento nº 11 Prueba 3

Para esta Intento, realicé la técnica del UnderSampling y utilicé el mismo algoritmo.

```
rus = RandomUnderSampler(random_state=40, sampling_strategy='majority', replacement=False)
Xu, yu = rus.fit_resample(data_trainCopia, target_train)
```

El resultado de las clases tras la aplicación de estas técnicas es la siguiente:



Clases tras el UnderSampling

Como podemos observar, el resultado no es muy bueno y también lo observamos en el score obtenido en Kaggle.

Intento nº 12 Prueba 3

Para intentar comprobar si esta técnica nos puede ayudar, realizo con los mismos datos una nueva Intento, utilizando una mezcla de algoritmos: *BaggingClassifier* con *LGBMClassifier*:

```
BaggingClassifier(lgb.LGBMClassifier(objective='regression_l1',
n_estimators=200,n_jobs=2, num_leaves=40, learning_rate=0.099))
```

No obtengo buenos resultados a pesar de tener un score local de 82.59

Intento nº 13 Prueba 3

Realizo la misma Intento anterior, pero esta vez modifico un poco los parámetros:

```
BaggingClassifier(lgb.LGBMClassifier(objective='regression_l1',
n_estimators=200,n_jobs=2, num_leaves=40, learning_rate=0.09))
```

Intento n° 14 Prueba 4

Estos Intentos los había realizado sin las etiquetas que utiliza el profesor en la plataforma de *Kaggle*, debido a que el csv no se encontraba en los archivos a descargar. En prado, me di cuenta de que ya estaba el archivo y decidí realizar la Intento con el algoritmo *LGBMClassifier* que es con el que obtengo los mejores resultados:

```
LGBMClassifier(objective='regression_l1',n_estimators=200,n_jobs=2, num_leaves=40, learning_rate=0.09)
```

Intento n° 15 Prueba 4

Realizo otra subida ajustando los parámetros:

```
LGBMClassifier(objective='regression_l1',n_estimators=200,n_jobs=2, num_leaves=40, learning_rate=0.03)
```

Intento n° 16 Prueba 4

Realizo otra subida ajustando los parámetros y de esta forma, vuelvo a superar mi score actual en *Kaggle*:

```
LGBMClassifier(objective='regression_l1',n_estimators=300,n_jobs=2, learning_rate=0.09)
```

Intento n° 17 Prueba 4

Vuelvo a utilizar el algoritmo *XGBClassifier* puesto que las primeras Intentos con él, las realicé sin las etiquetas de Tipo_marchas.

```
XGBClassifier(max_depth=10, n_estimators = 200, n_jobs=-1)
```

Sin embargo, no obtengo un buen score en *Kaggle*.

Intento n° 18 Prueba 4

Regreso al *LGBMClassifier* y vuelvo a realizar un ajuste de los parámetros con el que vuelvo a obtener una mejora:

```
objective='regression_l1',n_estimators=200, n_jobs=2, learning_rate=0.07
```

Intento n° 19 Prueba 5

De forma local, realizo varias Intentos ajustando los parámetros, pero no consigo ningún buen resultado. Ahora me planteo que quizás sea necesario un pre-procesado de

los datos. Leyendo en Internet descubro que el hecho de que haya una variable con una correlación muy pequeña puede estar haciendo que la predicción de los algoritmos empeore.

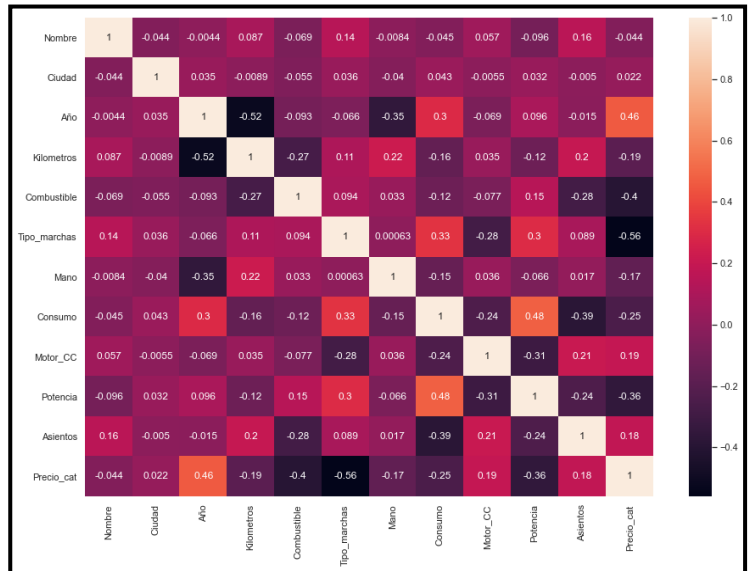
Por tanto, realizo la matriz de correlación con respecto al *Precio_cat*.

```
matriz_correlacion = data_trainCopia.corr()
sns.set(rc={'figure.figsize':(15,10)})
sns.heatmap(matriz_correlacion, annot=True)
plt.show()
```

Código para visualizar la matriz de correlación

Al visualizarla descubro que hay dos atributos que tienen una muy baja correlación con el objetivo a predecir: Nombre y Ciudad.

Para la realización de esta Intento, eliminaré el atributo Ciudad.



Matriz de correlación

La Intento la realizaré con *LGBMClassifier* con el que obtengo unos buenos resultados anteriormente.

```
n_estimators=200, num_leaves=35, scale_pos_weight=0.1
```

Pero obtuve un mal resultado al subirlo a la plataforma.

Intento nº 20 Prueba 6

Añado de nuevo el atributo y realizo un nuevo ajuste de parámetros:

```
objective='regression_l1', n_estimators=180, n_jobs=2, learning_rate=0.07, num_leaves=35
```

Intento nº 21 Prueba 6

Viendo que vuelvo a obtener buenos resultados, decido aplicar de nuevo la técnica de *UnderSampling* y usar *BaggingClassifier* con *RandomForest*, pero esta vez añado los parámetros al algoritmo de *Bagging*:

```
BaggingClassifier(RandomForestClassifier(), n_estimators=180, n_jobs=4)
```

Pero vuelvo a alejarme del objetivo.

Intento nº 22 Prueba 6

Leyendo por internet y extrañado de obtener un buen resultado en score y malo en *Kaggle*, descubro que es por un problema de *Overfitting*, así que para solucionarlo empiezo a trabajar con el número de profundidad del algoritmo.

```
learning_rate=0.06, objective='regression_l1',  
n_estimators=410, n_jobs=2, num_leaves=10, max_depth=5
```

Después de varios intentos, vuelvo a subir mi score y ya alcanzo el 79%.

Intento nº 23 Prueba 6

Realizo un nuevo intento con la profundidad.

```
objective='regression_l1', n_estimators=300, n_jobs=2, learning_rate=0.06, num_leaves=10, max_depth=5
```

De esta forma, vuelvo a superar mi score y subo hasta el 0.79292.

Intento nº 24 Prueba 6

Tras la Intento realizada anteriormente, vuelvo a aplicar la técnica de *UnderSampling*. Uso los mismos parámetros que para el caso anterior, pero no obtengo buenos resultados.

Intento nº 25 Prueba 6

Elimino la técnica de *UnderSampling* y realizo unos ajustes de los parámetros:

```
learning_rate=0.055, objective='binary', n_estimators=460, n_jobs=2, num_leaves=10, max_depth=-1, seed=46000
```

Esta vez, obtengo de nuevo otra subida de score en *Kaggle*: 0.79810.

Intento nº 26 Prueba 6

Realizo otra Intento ajustando el número de estimadores:

```
learning_rate=0.055, objective='binary', n_estimators=400, n_jobs=2, num_leaves=10, max_depth=-1, seed=46000
```

Pero no consigo una subida.

Intento n° 27 Prueba 6

Pruebo a realizar la última subida obteniendo un score intermedio con otro ajuste de parámetros:

```
learning_rate=0.055, objective='binary', n_estimators=450, n_jobs=2, num_leaves=10, max_depth=-1, seed=46000
```

Consigo una nueva subida que se aproxima al 80% en *Kaggle*: 0.79896

Intento n° 28 Prueba 7

Vuelvo a intentar ver si hay algún atributo correlacionado. Para ello, elimino las etiquetas *Nombre* y *Asientos* y realizo el siguiente ajuste de parámetros:

```
learning_rate=0.04, objective='regression_l1', n_estimators=300, n_jobs=2
```

Pero vuelvo a alejarme del objetivo.

Intento n° 29 Prueba 7

Añado *Nombre* y elimino *Asientos* y aplico los parámetros anteriores, pero obtengo un resultado anterior al anterior.

Intento n° 30 Prueba 7

Añado *Asientos* y elimino *Nombre* y aplico los parámetros anteriores, pero vuelvo a obtener un resultado anterior al anterior.

Intento n° 31 Prueba 7

Vuelvo al momento donde obtuve el mejor score e intento ajustar los parámetros tras un previo estudio:

```
learning_rate=0.055, objective='binary', n_estimators=460, n_jobs=2, num_leaves=10, max_depth=-1, seed=46000
```

Pero vuelvo a no acercarme al score deseado.

Intento n° 32 Prueba 7

Como anteriormente había probado con la técnica de *UnderSampling*, decido aplicar *OverSampling* y utilizar los mismos atributos que me dieron el buen score.

Sin embargo, vuelvo a alejarme mucho volviendo a un score de 0.78

Intento n° 33 Prueba 7

Vuelvo a realizar la misma Intento con otro ajuste de parámetros que parecen mejorar el score local:

```
learning_rate=0.06, objective='binary', n_estimators=460, n_jobs=2, num_leaves=10, max_depth=-1, seed=46000
```


Sin embargo, parece que lo que obtengo es *Overfitting* y decido abandonar la técnica del *OverSampling*.

Intento n° 34 Prueba 7

Pruebo a cambiar el pre-procesado de los datos y decido utilizar la técnica del más frecuente. Utilizo el mismo algoritmo con los parámetros iguales. Pero vuelvo a no tener buenos resultados.

Intento n° 35 Prueba 7

Realizo mi última Intento de ajustes de parámetros porque empiezo a encontrarme en un túnel sin salida.

```
learning_rate=0.06, objective='binary', n_estimators=455, n_jobs=2, num_leaves=10, max_depth=-1, seed=46000
```

Efectivamente, no consigo una mejora y vuelvo al score de 0.78, por tanto, decido empezar a cambiar de técnicas de algoritmos

Intento n° 36 Prueba 9

Ahora hago uso del algoritmo de *StackingClassifier* con *LGBMClassifier* y *RandomForest* y el *meta_classifier* que utilizo será el *SupportVectorMachines*. Los parámetros que utilizo son los siguientes:

```
LGBMClassifier(learning_rate=0.06, objective='binary', n_estimators=350, n_jobs=2, num_leaves=10, max_depth=-1, seed=46000)
RandomForestClassifier(n_estimators = 300, criterion = "gini", max_depth = 10, max_features = "auto", min_samples_leaf =
0.005, min_samples_split = 0.005, n_jobs = -1, random_state = 1000)
StackingCVClassifier(classifiers = [classifier1, classifier4], shuffle = False, use_proba = True, cv = 5,
meta_classifier = SVC(probability = True))
```

Aunque obtengo un score del 0.79292 no creo obtener mejores resultados, porque en local no conseguía mejorar. A lo que se añade que probar este algoritmo provocaba un alto tasa de espera y ejecución.

Intento n° 37 Prueba 9

Sin embargo, en un foro de internet descubro un ajuste de parámetros que me puede ayudar. A lo anterior, añado a los clasificadores el propio SVC con los siguientes parámetros:

```
SVC(C = 50, degree = 1, gamma = "auto", kernel = "rbf", probability = True)
```

Para mi sorpresa obtengo un score del 0.00000. Luego me he dado cuenta de que al crear el csv en vez de asignar la predicción había asignado el modelo y este era el csv tan bonito que subí a Kaggle.

	A	B	C	D	E	F	G
46							
47							
48							
49							
50							
51							
52							
53							
54							
55							
56							
57							
58							
59							
60							
61							
62							
63							
64							
65							
66							
67							
68							
69							
70							
71							
72							
73							

```

46 use_probas=True)"
47 4823,"StackingCVClassifier(classifiers=[LGBMClassifier(learning_rate=0.06,
48     n_estimators=350, n_jobs=2,
49     num_leaves=10,
50     objective='binary',
51     seed=46000),
52     RandomForestClassifier(max_depth=10,
53     min_samples_leaf=0.005,
54     min_samples_split=0.005,
55     n_estimators=300,
56     n_jobs=-1,
57     random_state=1000),
58     SVC(C=50, degree=1, gamma='auto',
59     probability=True)],
60     cv=5, meta_classifier=SVC(probability=True), shuffle=False,
61     use_probas=True)"
62 4824,"StackingCVClassifier(classifiers=[LGBMClassifier(learning_rate=0.06,
63     n_estimators=350, n_jobs=2,
64     num_leaves=10,
65     objective='binary',
66     seed=46000),
67     RandomForestClassifier(max_depth=10,
68     min_samples_leaf=0.005,
69     min_samples_split=0.005,
70     n_estimators=300,
71     n_jobs=-1,
72     random_state=1000),
73     SVC(C=50, degree=1, gamma='auto',
stackingClassifierlgbmSVCRandomF

```

demonstración del error que provocó el 0.00000 en Kaggle

Intento nº 38 Prueba 9

Voy a utilizar de nuevo la técnica de *StackingClassifier* pero con *RandomForest* y *LGBMClassifier*:

```

RandomForestClassifier(n_estimators = 300, criterion = "gini", max_depth = 2,
    min_samples_leaf = 0.005, min_samples_split = 0.005, n_jobs = -1,
    random_state = 1000)

LGBMClassifier(learning_rate=0.055, objective='binary', n_estimators=460, n_jobs=2,
    num_leaves=10, max_depth=-1, seed=46000)

```

Pero vuelvo a no obtener buenos resultados.

Intento nº 39 Prueba 11

En otro foro de internet, leo que hay un algoritmo que ya tenía visto y que parece obtener buenos resultados en problemas de clasificación. Empiezo a realizar Intentos con él y viendo los buenos resultados en local, decido probar en *Kaggle*.

```
GradientClassifier(n_estimators=440, learning_rate=0.1)
```

Sin embargo, parece que no he dado con la tecla por que bajo demasiado en la plataforma. Por ahora, decido abandonar este algoritmo y seguir probando nuevas técnicas.

Intento nº 40 Prueba 16

Viendo que ajustando los parámetros del algoritmo no obtengo buenos resultados, y que algoritmos que en foros de *Machine Learning* parecen ayudar, no me aportan resultados relevantes, decido aplicar una técnica de normalización de los datos.

Lo primero que realizo es una transformación de los datos categóricos, y evito usar las etiquetas con ellos. Esto es debido a que cuando un coche ha recorrido 10.000 kilómetros si lo codifico como 2 y un coche que recorre 50 lo hago como 10, no estoy dando una importancia verdadera a los valores de esos atributos.

Una vez que tenga los datos de forma numérica, le aplico una normalización que consistirán en train entre la diferencia de medias entre la desviación típica y en test, lo mismo, pero con respecto a train.

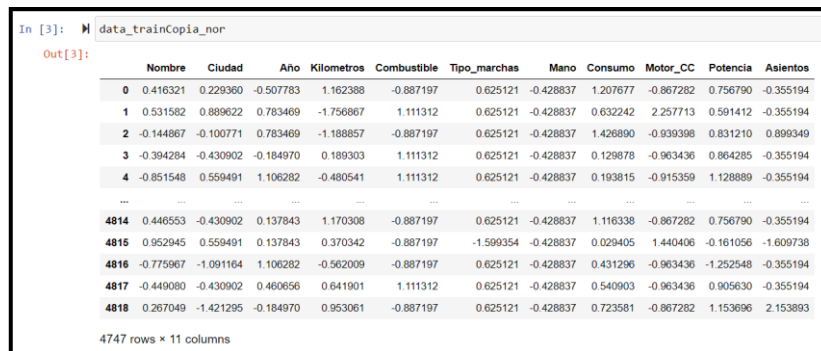
```
data_trainCopia["Consumo"]=data_trainCopia["Consumo"].str.replace('kmpl','')
data_trainCopia["Motor_CC"]=data_trainCopia["Motor_CC"].str.replace('CC','')
data_trainCopia["Potencia"]=data_trainCopia["Potencia"].str.replace('bhp','')
data_testCopia["Consumo"]=data_testCopia["Consumo"].str.replace('kmpl','')
data_testCopia["Motor_CC"]=data_testCopia["Motor_CC"].str.replace('CC','')
data_testCopia["Potencia"]=data_testCopia["Potencia"].str.replace('bhp','')

target_train=data_trainCopia['Precio_cat']
data_trainCopia=data_trainCopia.drop(['Precio_cat'], axis=1)

data_trainCopia=data_trainCopia.astype(float)
data_testCopia=data_testCopia.astype(float)

data_testCopia_nor = (data_testCopia-data_trainCopia.mean(0))/data_trainCopia.std(0)
data_trainCopia_nor = (data_trainCopia-data_trainCopia.mean(0))/data_trainCopia.std(0)
```

El dataFrame del train tendrá el siguiente aspecto:



	Nombre	Ciudad	Año	Kilometros	Combustible	Tipo_marchas	Mano	Consumo	Motor_CC	Potencia	Asientos
0	0.416321	0.229360	-0.507783	1.162388	-0.887197	0.625121	-0.428837	1.207677	-0.867282	0.756790	-0.355194
1	0.531582	0.889622	0.783469	-1.756867	1.111312	0.625121	-0.428837	0.632242	2.257713	0.591412	-0.355194
2	-0.144867	-0.100771	0.783469	-1.188857	-0.887197	0.625121	-0.428837	1.426890	-0.939398	0.831210	0.899349
3	-0.394284	-0.430902	-0.184970	0.189303	1.111312	0.625121	-0.428837	0.129878	-0.963436	0.864285	-0.355194
4	-0.851548	0.559491	1.106282	-0.480541	1.111312	0.625121	-0.428837	0.193815	-0.915359	1.128889	-0.355194
...
4814	0.446553	-0.430902	0.137843	1.170308	-0.887197	0.625121	-0.428837	1.116338	-0.867282	0.756790	-0.355194
4815	0.952945	0.559491	0.137843	0.370342	-0.887197	-1.599354	-0.428837	0.029405	1.440406	-0.161056	-1.609738
4816	-0.775967	-1.091164	1.106282	-0.962009	-0.887197	0.625121	-0.428837	0.431296	-0.963436	-1.252548	-0.355194
4817	-0.449080	-0.430902	0.460656	0.641901	1.111312	0.625121	-0.428837	0.540903	-0.963436	0.905630	-0.355194
4818	0.267049	-1.421295	-0.184970	0.953061	-0.887197	0.625121	-0.428837	0.723581	-0.867282	1.153696	2.153893

DataFrame de train normalizado

El algoritmo que volveré a utilizar será el *LGBMClassifier*:

```
LGBMClassifier(learning_rate=0.06, objective='binary', n_estimators=350, n_jobs=2,
               num_leaves=10, max_depth=-1, seed=46000)
```

Sin embargo, no obtengo buen resultado.

Intento n° 41 Prueba 16

Supongo que esto es debido de nuevo a no utilizar las etiquetas como el profesor las ha clasificado, por tanto, elimino el proceso de las variables categóricas y vuelvo a ejecutar el algoritmo. Aunque obtengo un peor score local, subo en *Kaggle*, lo que demuestra que estaba en razón.

Intento n° 42 Prueba 16

Ahora pruebo con los parámetros con los que obtuve la última gran subida y la normalización de los datos. No obstante, no obtengo un buen score.

Intento n° 43 Prueba 17

Como parece que no la normalización obtengo buenos resultados, pruebo a utilizar aquel algoritmo que obtuve en 0.0000 por culpa de no crear bien en csv. Utilizo los mismos parámetros y lo pruebo en *Kaggle*, pero bajo al 0.78, lo que me aleja y mucho del objetivo.

Intento n° 44 Prueba 17

Volviendo a la lectura de nuevos ⁶artículos⁷, siguen refiriéndose que, contra las clases desbalanceadas, el mejor tratamiento es la técnica de *OverSampling*. Así que decido de nuevo probar con mi algoritmo y otro ajuste de parámetros y la normalización de datos.

```
LGBMClassifier(learning_rate=0.055, objective='binary', n_estimators=740, n_jobs=2,
               num_leaves=12, max_depth=-1, seed=46000)
```

Para mi sorpresa, aunque no consigo superar el score, veo que se acerca mucho al actual.

Intento n° 45 Prueba 19

En mi último intento del día, y viendo que el anterior ajuste se acercaba a poder superar por fin el score en *Kaggle*, realizo un ajuste de parámetros que me dan una subida en local, tras subirlo en *Kaggle*, me doy cuenta tras 24 intentos, que he conseguido mejorar el score, pero no logro mi objetivo. ¡Me he quedado a 0.00016 del 0.8!

```
learning_rate=0.055, objective='binary', n_estimators=640, n_jobs=2, num_leaves=20, max_depth=-1, seed=4600
```

En las siguientes Intentos, realizaré un estudio y ajuste de parámetros para conseguir un score del 80%

⁶ <https://www.doctormetrics.com/datos-no-balanceados/>

⁷ https://www.researchgate.net/publication/297698675_Tecnicas_de_muestreo_para_mejorar_el_rendimiento_del_algoritmo_back-propagation_en_problemas_de_desbalance_de_clases_Un_estudio_empirico_sobre_la_clasificacion_en_imagenes_de_percepcion_r_emota

Intento nº 46 Prueba 19

Primer ajuste de parámetros:

```
learning_rate=0.055, objective='binary', n_estimators=640, n_jobs=2, num_leaves=22, max_depth=-1, seed=46000
```

Intento nº 47 Prueba 19

Segundo ajuste de parámetros:

```
learning_rate=0.055, objective='binary', n_estimators=740, n_jobs=2, num_leaves=12, max_depth=-1, seed=46000
```

Intento nº 48 Prueba 20

Tercer ajuste de parámetros:

```
learning_rate=0.054, objective='binary', n_estimators=620, n_jobs=2, num_leaves=22, max_depth=-1, seed=46000
```

En ninguna de las tres Intentos, a pesar de obtener mejores scores, consigo ni siquiera acercarme al buen resultado en Kaggle. En un foro de internet, descubro que mi problema ha sido no usar o haber almacenado la semilla con la que *SMOTE* creó mis datos. Por tanto, cada vez que ejecuté la técnica de *OverSampling* o reinicié el *Notebook*, obtenía datos diferentes. Mi frustración empieza a convertirse en una carga importante.

Empiezo a plantarme que no estoy por buen camino, y decido eliminar la técnica de *OverSampling*. Por tanto, realizaré dos Intentos con mi algoritmo y la normalización sin la técnica ya mencionada.

Intento nº 49 Prueba 21

Ajustando los parámetros y sin la normalización de datos

```
learning_rate=0.055, objective='binary', n_estimators=440, n_jobs=2, num_leaves=11, max_depth=-1, seed=46000, reg_alpha=0.1
```

Intento nº 50 Prueba 21

Ajustando los parámetros y con la normalización de datos

```
learning_rate=0.055, objective='binary', n_estimators=440, n_jobs=2, num_leaves=11, max_depth=-1, seed=46000, reg_alpha=0.9
```

En ninguno de los dos casos obtengo buenos resultados.

Intento n° 51 Prueba 24

En varios⁸ artículos⁹, descubro que el uso del *GradientClassifier*, algoritmo que ya había utilizado y descartado anteriormente, puede ayudarme a conseguir ese esperado 0.8. Por tanto, vuelvo a probarlo realizando un ajuste de parámetros y aplicando de nuevo la técnica de *OverSampling* puesto que en los algoritmos leemos que cuantos más datos poseamos mejor resultamos ofrece y realizo la Intento con los siguientes valores:

```
GradientBoostingClassifier(learning_rate=0.9, n_estimators=750, max_depth=2)
```

Nos acercamos de nuevo al score que queremos superar, pero habrá que realizar de nuevo un cambio en los parámetros.

En las últimas tres Intentos, voy a realizar una selección de las mejores ideas que he utilizado para conseguir llegar al 0.8 en *Kaggle*.

Intento n° 52 Prueba 24

Observando que *GradientBoosting* tiene un buen resultado, junto a la técnica de *OverSampling*, decido eliminar el atributo Kilómetros debido a que, como expliqué anteriormente, puede estar perjudicando a la predicción por la pérdida de información a la hora de la codificación.

Los parámetros que utilizo son:

```
GradientBoostingClassifier(learning_rate=0.5, n_estimators=70, max_depth=5, random_state=42)
```

Pero a pesar del buen score en local y haber intentado prevenir el *Overfitting*, no obtengo un buen resultado en *Kaggle*.

Intento n° 53

Quiero darle una oportunidad a mi último algoritmo, esta vez rehago un nuevo ajuste de parámetros y añado el atributo kilómetros, puesto que puede ser que no tenga razón en que sea perjudicial.

Pruebo un ajuste de parámetros para prevenir el *Overfitting*:

```
learning_rate=0.3, objective='multi:softmax', n_estimators=70, n_jobs=2, num_leaves=10, max_depth=6
```

Pero me encuentro que es el peor resultado en *Kaggle* de este algoritmo con un 0.77 en la plataforma.

⁸ <https://towardsdatascience.com/understanding-gradient-boosting-machines-9be756fe76ab>

⁹ <https://www.displayr.com/gradient-boosting-the-coolest-kid-on-the-machine-learning-block/>

Intento nº 54 Prueba 31

Para mi último intento, quise intentarlo hasta el final con mi algoritmo. Eliminé la etiqueta Kilómetros ya que vi que me daba mejor score local. Realicé un ajuste de parámetros con la técnica *GridSearch* y conseguí un score local de 84.03 que nunca había conseguido sin utilizar *OverSampling*:

```
LGBMClassifier(random_state=0, objective='multi:softmax', silent=True, colsample_bytree = 0.6,  
learning_rate= 0.1, max_depth = 10, n_estimators=70, scales_pos_weight=0.8, subsample=0.6
```

Sin embargo, sólo obtuve un 0.79119 en Kaggle.

Anexo adicional

Aquí voy a comentar algunas de las Intentos que realicé porque no tenías más intentos, no sabía si estaban realizadas de forma correcta o no me fiaba del resultado que podía obtener en *Kaggle*.

Intento con cambio de codificador

En internet en otro de los artículos que leí, hablaban del uso de *HotVectorEncode* en sustitución del *LabelEncode*, para las variables categóricas, en mi caso para Kilómetros, Motor_CC... Sin embargo, numerosos errores, warnings y mi no convencimiento de que lo que estaba haciendo iba a ser fructífero me llevaron a no utilizar la técnica.

Uso de *StackingClassifier* con distintos modelos y datos

Según iba avanzando en la práctica, con este algoritmo que parece dar buenos resultados, se me ocurrieron numerosas ideas que podía haber tenido un buen resultado en *Kaggle*, debido a que en local sí era sí.

Primera Intento

Uso de *GradientClassifier* con *LGBMClassifier*, sin normalizar los datos y sin ninguna técnica.

Segunda Intento

Uso de *GradientClassifier* con *LGBMClassifier*, normalizando los datos y sin ninguna técnica.

Tercera Intento

Uso de *GradientClassifier* con *LGBMClassifier*, normalizando los datos y sin con la técnica de *OverSampling*.

Cuarta Intento

Uso de *GradientClassifier* con *LGBMClassifier*, sin normalizar los datos y con la técnica de *OverSampling*.

Para no alargar más este apartado, quedarían estas técnicas con la eliminación de atributos con poca correlación que podían estar afectando a la predicción como Nombre, Ciudad, Kilómetros o Mano, juntado a el uso de otros algoritmos como *SVM*, *RandomForest* o *XGBClassifier*. Sin embargo, para realizar las Intentos necesitaría cerca de 15 intentos, de los cuales no poseía.

Conclusión

Al ser una competición, tratas de hacerlo lo mejor posible, sin embargo, creo que no ha sido mi mejor práctica. Mi score final ha sido un 0.79986, quedándome a 0.00014 milésimas de poder estar en el 0.8. He encontrado numerosos artículos, foros, pdf's y páginas web, que hablan de la cantidad de posibilidades que hay en el mundo del Machine Learning, y que me han hecho entender de una forma más práctica el tratamiento de algunos algoritmos y procesado de datos que facilitan su estudio y clasificación.

No obstante, esta práctica hubiese podido ser más “divertida” como, para mí, fueron las anteriores, si no se tratara de una competición e intentar superar un score, por que sino me voy a quedar fuera de un ranking. He usado prácticamente todos mis intentos, porque al principio no sabíamos que se podría valorar positivamente el uso de muchas técnicas y subidas.

Otro de los problemas han sido las Intentos en local, y es que en general para esta práctica no han ido nada bien ni han sido tan importantes. Esto es debido a que aunque las puntuaciones en local fueran altas, no significaban que fuesen mejores. En general en esta práctica no he obtenido los resultados esperados.

Por otra parte, esta práctica y a diferencia de las anteriores, me ha permitido conocer en profundidad como funciona el aprendizaje supervisado. Durante la práctica me he dado cuenta de que quizás tenía que haberme centrado más en un pre-procesado de los datos antes que en la configuración de los algoritmos, pero es posible que este error lo tenga, porque es la primera vez que me enfrento a un problema de estas características.

No obstante, aunque estoy contento con el resultado obtenido, no lo estoy tanto con mi posición, lo cual ha llegado a ser muy frustrante de cara a la nota y recompensas que consiga, pues depende de lo que haga el resto aunque yo haya cumplido con mis objetivos.

Bibliografía

https://programacion.net/articulo/introduccion_a_pandas_1632

https://xgboost.readthedocs.io/en/latest/python/python_api.html

[https://scikit-](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html)

[learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html)

<https://towardsdatascience.com/understanding-gradient-boosting-machines-9be756fe76ab>

https://www.researchgate.net/publication/297698675_Tecnicas_de_muestreo_para_mejorar_el_rendimiento_del_algoritmo_back-propagation_en_problemas_de_desbalance_de_clases_Un_estudio_empirico_sobre_la_clasificacion_en_imagenes_de_percepcion_remota

<https://www.doctormetrics.com/datos-no-balanceados/>

<https://www.displayr.com/gradient-boosting-the-coolest-kid-on-the-machine-learning-block/>

<https://machinelearningmastery.com/hyperparameter-optimization-with-random-search-and-grid-search/>

[https://scikit-](https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html)

[learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html](https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html)

<https://es.stackoverflow.com/questions/386685/obtener-lista-de-valores-mas-repetidos-en-una-lista-anidada-en-python>

<https://stackoverflow.com/es/q/4104780>

<https://aprendeconalf.es/docencia/python/manual/pandas/>