



**UNIVERSIDAD  
DE GRANADA**

Departamento de Ciencias de la  
Computación e Inteligencia Artificial



















# Inteligencia de Negocio

Práctica 3:  
Competición en Kaggle

Curso 2020-2021

Cuarto curso del grado de Ingeniería Informática

Luis Miguel Aguilar González  
laguilarg99@correo.ugr.es

25	—	Carlos Díaz_Ceuta		0.77135	14	17h
26	—	Sandro Sanchez 53588054D		0.77049	20	2d
27	—	Francisco Amor 460704509Z		0.76962	10	18h
28	—	Sohaib Mohamed Ceuta		0.76617	9	16h
29	—	NajibSaadouliArco77022855H		0.76358	13	13h
30	—	LuisUGR		0.76100	20	2d
31	—	AlonsoBueno76067525Q		0.76100	22	2d
32	—	Ana Maria Romero		0.75754	12	13h
33	—	FERNANDO DE LA HOZ		0.75323	4	3d
34	—	Luis Miguel Aguilar Gonzalez 2...		0.74805	16	21h
35	—	Alberto Garcia Valero 7665415...		0.74633	12	2d
36	—	Juan Manuel Consigliere Picco		0.74460	9	3d
37	—	Elena_Ceuta		0.73684	7	14h
38	—	Alberto_Garcia_Paño_Ceuta		0.72562	10	2d
39	—	Ayoub_Filali_Mohamed_Ceuta		0.71958	22	1d
40	—	OUMNIA_CHAARA_CHAARA_C...		0.71268	12	7d
41	—	David Joaquín González-Vene...		0.70491	8	1d
42	—	JAIROCIDGARZON 15471975		0.66954	2	4d

# Índice

<b>1.- Introducción</b>	<b>4</b>
<b>2.- Proceso seguido</b>	<b>4</b>
<b>3.- Algoritmos Utilizados</b>	<b>6</b>
<b>4.- Tabla resumen</b>	<b>7</b>

# 1.- Introducción

En esta práctica nos enfrentamos principalmente a un problema de clasificación de los precios de vehículos en 5 categorías (1 para aquellos con un menor precio mientras que 5 es el mayor precio) dados los valores de una serie de atributos.

Para el desarrollo de la práctica, cuyo objetivo es maximizar la precisión en la predicción del conjunto de test para ganar la competición se han utilizado tres algoritmos, concretamente, *Random Forest*, *Gradient Boosting Classifier* y *XGBClassifier*. Sin embargo, el preprocesado de los datos y la forma en que estos han sido dispuestos para su aprendizaje ha sido variada dependiendo de distintas observaciones que he realizado y que desglosaré a lo largo de la práctica.

## 2.- Proceso seguido

La elaboración de la práctica la dividiría en tres partes en función de las decisiones que he tomado y del consecuente preprocesado independientemente del algoritmo utilizado que como ya he dicho han sido los 3 nombrados anteriormente desde el primer momento siendo *XGBClassifier* el que mejores valores ha conseguido por regla general aunque ha sido finalmente *Random Forest* el que ha conseguido mi mejor puntuación:

- ➔ Primera Parte: Cometí el error de confiar que el principal problema era simplemente el formato de los datos en el conjunto y cómo resolver los valores nulos de las columnas. Aparte de decidir si realizar oversampling o undersampling. Respecto a esto último siempre he hecho oversampling y aunque probé a usar *RandomOverSampler*, para todas las subidas realizadas en la competición he usado **SMOTE** para aumentar el dataset a un total de 11035. Tal y como indico en la siguiente línea. Respecto al preprocesado siempre he mantenido la mayor cantidad de los atributos posibles despreciando *Motor\_CC* (me parece redundante del atributo *Potencia* del vehículo solo que con otra unidad de medida) y el *Consumo* del vehículo.

```
oversample = SMOTE(sampling_strategy='not majority', random_state=5)
X_res, y_res = oversample.fit_resample(X, y)
data_train = pd.concat([pd.DataFrame(X_res), pd.DataFrame(y_res)],
axis=1)
```

Los valores nulos en esta primera parte los resolvía usando el método `ffill` en la función `fillna`

```
data_train = data_train.fillna(method='ffill')
```

Que simplemente toma el valor no nulo justo por encima de la fila actual y lo inserta.

→ Segunda Parte: En esta segunda parte me centré sobre todo en cómo reemplazar los valores nulos. Sin embargo, al hacer:

```
print(data_train.isnull().sum())
```

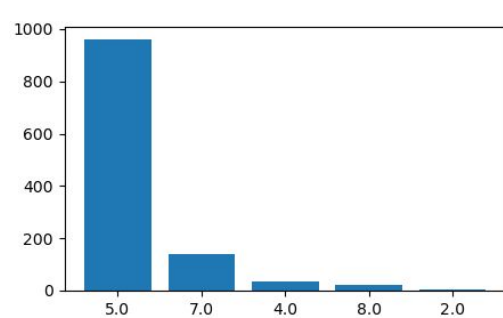
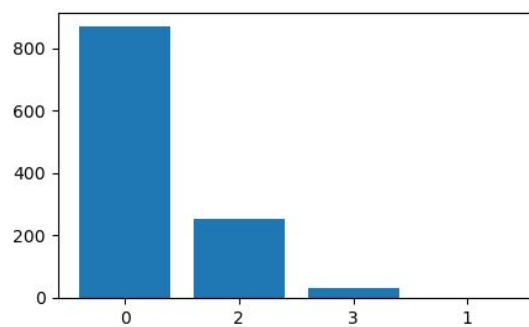
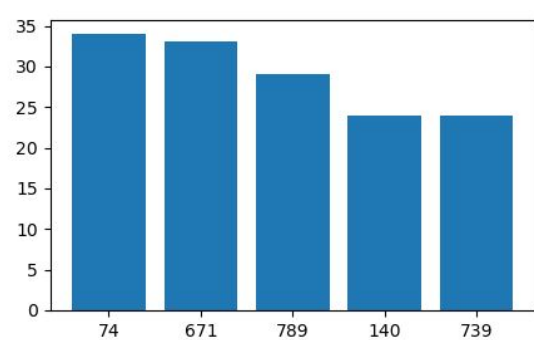
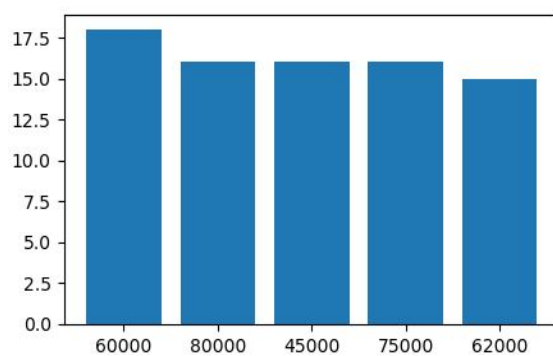
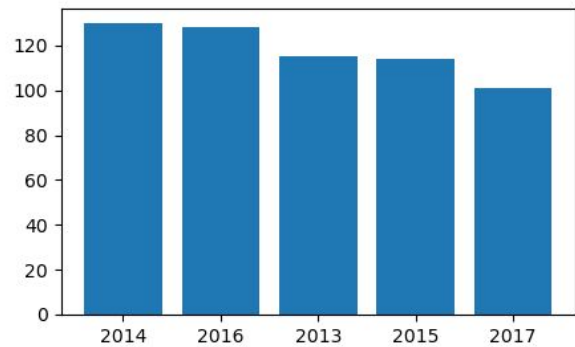
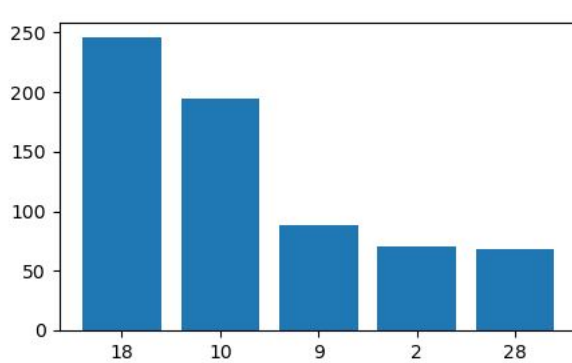
Nombre	72
Ciudad	72
Año	72
Kilometros	72
Combustible	72
Tipo_marchas	72
Potencia	175
Mano	72
Asientos	106

Se puede observar que para el total del conjunto los valores nulos son pocos y no debería de marcar una gran diferencia la forma en que los eliminamos, sin embargo, en un principio pensé que realmente podría marcar una diferencia grande la forma de reemplazar dichos valores, por ello, para reemplazar los kilómetros, por ejemplo, usaba la media de la columna. Es en este momento cuando los outliers (valores atípicos) comienzan a alterar el conjunto insertando valores que hacen que la precisión empeore, en el caso de los kilómetros hay un valor de 6.5 millones que aumenta el valor de la media y por lo tanto haciendo una inserción de mala calidad.

Empecé a usar la función *classification\_report* que devuelve información muy útil para ver cómo de buena es la clasificación dándome cuenta de que no estoy haciendo bien la inserción y es motivo de los outliers, que además son los culpables de la mala precisión, por ello intenté automatizar una función que calcule máximos y mínimos en un rango de valores aceptable usando los cuartiles del conjunto ajustando los valores usados a este rango. Sin embargo, esta operación reducía el tamaño de la muestra drásticamente a pesar de hacer oversampling posteriormente (por eso los valores tan bajos entre el día 29 y el 31).

Al final, opté por ajustar el rango de forma manual eliminando dichos outliers, pero reduciendo la muestra, es ahora, al terminar la competición, cuando me doy cuenta, que en lugar de eliminar estas muestras, reemplazar dichos valores por algún valor más próximo a la media del conjunto, hubiese sido mejor en lugar de eliminar directamente la muestra.

Para ajustar ese rango representé los distintos atributos del conjunto test para mantener aquellos valores importantes en el conjunto de entrenamiento y no que al ajustar para eliminar los outliers se eliminen muestras que pueden ser importantes, como se representa en las siguientes páginas en el siguiente orden Nombre, Año, Kilometros, Potencia, Mano, Asientos, localización, combustible y la transmisión.



Ciudad\_C

0 1032

1 127

Name: Ciudad\_C, dtype: int64

Ciudad\_D

0 1056

1 103

Name: Ciudad\_D, dtype: int64

Ciudad\_E

0 1054

1 105

Name: Ciudad\_E, dtype: int64

Ciudad\_F

0 1014

1 145

Name: Ciudad\_F, dtype: int64

Ciudad\_G  
 0 1061  
 1 98  
 Name: Ciudad\_G, dtype: int64

Ciudad\_H  
 0 1041  
 1 118  
 Name: Ciudad\_H, dtype: int64

Ciudad\_I  
 0 1042  
 1 117  
 Name: Ciudad\_I, dtype: int64

Ciudad\_J  
 0 1053  
 1 106  
 Name: Ciudad\_J, dtype: int64

Ciudad\_K  
 0 1101  
 1 58  
 Name: Ciudad\_K, dtype: int64

Ciudad\_L  
 0 1126  
 1 33  
 Name: Ciudad\_L, dtype: int64

Combustible\_Diesel  
 1 609  
 0 550  
 Name: Combustible\_Diesel, dtype: int64

Combustible\_LPG  
 0 1156  
 1 3  
 Name: Combustible\_LPG, dtype: int64

Combustible\_Petrol  
 0 625  
 1 534  
 Name: Combustible\_Petrol, dtype: int64

Transmission  
 1 813  
 0 346  
 Name: Transmission, dtype: int64

	precision	recall	f1-score	support
1	0.94	0.92	0.93	648
2	0.83	0.86	0.84	628
3	0.81	0.80	0.81	676
4	0.83	0.82	0.82	667
5	0.91	0.93	0.92	692
accuracy			0.87	3311
macro avg	0.87	0.87	0.87	3311
weighted avg	0.87	0.87	0.87	3311

Como se puede observar en el mejor resultado la clasificación está balanceada

- Tercera parte: Encarando la recta final del proyecto, leyendo algunos notebooks de kaggle descubrí una idea que me gustó bastante, trataba de convertir atributos como la ciudad, el combustible y la transmisión en binarias cómo se puede observar a continuación:

```
var = 'Ciudad'
Location = data_train[[var]]
Location = pd.get_dummies(Location, drop_first=True)

var = 'Combustible'
Fuel_t = data_train[[var]]
Fuel_t = pd.get_dummies(Fuel_t, drop_first=True)
var = 'Tipo_marchas'
Transmission = data_train[[var]]
Transmission = pd.get_dummies(Transmission, drop_first=True)
Transmission.columns=['Transmission']

precio = data_train['Precio_cat']
del data_train['Precio_cat']
data_train = pd.concat([data_train, Location, Fuel_t, Transmission,
precio], axis=1)
data_train.drop(['Ciudad', 'Combustible',
'Tipo_marchas'], axis=1, inplace=True)
```

Fue en este momento, haciendo uso de una inserción adecuada, de la eliminación de outliers y de la conversión de estos atributos a un carácter binario junto con el algoritmo Random Forest cuando *classification\_report* en la columna support se podía observar un resultado casi perfectamente balanceado, consiguiendo mi mejor resultado de precisión. Los otros resultados del día 1 de Enero corresponden con la utilización de los otros algoritmos que a nivel local con el conjunto de entrenamiento conseguían una mejor precisión pero sin resultados con un buen balanceo de forma que al aplicarlos al conjunto de test obtenían finalmente peores resultados.

### 3.- Algoritmos Utilizados

Realmente aunque son tres algoritmos las técnicas de clasificación son solamente 2, *Random Forest* y *Gradient Boosting* pues el tercero *XGBClassifier* es una optimización del segundo.

- Random Forest: un modelo basado en árboles de decisión, el método genera el árbol de clasificación como un conjunto de árboles de decisión generados a partir de tomar varias muestras de forma aleatoria del conjunto total de datos, con el fin de mejorar la precisión y controlar el sobreajuste. La fortaleza de este método reside en que cada árbol protege al otro de sus errores individuales



- Gradient Boosting: Es una técnica de aprendizaje automático utilizado para el análisis de la regresión y para problemas de clasificación estadística. Construye el modelo de forma escalonada como lo hacen otros métodos de boosting, estos son meta-algoritmos que reducen el sesgo y varianza. En conclusión, el boosting consiste en combinar los resultados de varios clasificadores débiles para obtener un clasificador robusto. Dichos clasificadores tienen distintos pesos en función de la exactitud de sus predicciones.

## 4.- Tabla resumen

Los preprocesados a la hora de ponerlos en la tabla los enumeraré del 1 al 3 según lo descrito en el apartado de los preprocesados, si se repite el mismo en varias ocasiones seguidas puede ser por dos motivos, el primero, que el algoritmo sea distinto o probar distintos conjuntos de atributos o inserciones:

Id	Fecha/Hora (Aprox.)	Score (Test)	Preprocesado	Algoritmo
1	23/12/2020 12:00 am	0.70405	1	XGBClassifier
2	23/12/2020 17:00 pm	0.73339	1 (Algoritmo distinto)	RandomForest
3	23/12/2020 19:00 pm	0.71440	1 (Algoritmo distinto)	GradientBoosting
4	24/12/2020 2:00 am	0.68334	2 (Inserción simple)	RandomForest
5	29/12/2020 16:00 pm	0.39516	2 (Uso del rango de valores)	XGBClassifier
6	29/12/2020 18:00 pm	0.45729	2 (Distintos atributos)	XGBClassifier
7	29/12/2020 19:00 pm	0.59361	2 (Distintos Atributos)	XGBClassifier
8	30/12/2020 11:00 am	0.57808	2 (Distintos Atributos)	XGBClassifier
9	30/12/2020 13:00 pm	0.52804	2 (Distinta inserción)	XGBClassifier
10	30/12/2020 16:00 pm	0.54098	2 (Distintos algoritmos)	RandomForest
11	31/12/2020 12:00 am	0.48576	2 (Distintos algoritmos)	GradientBoosting
12	31/12/2020 13:00 pm	0.59361	2 (Distintos algoritmos)	XGBClassifier
13	31/12/2020 16:00 pm	0.74719	2 (Rango de valores manual)	XGBClassifier
14	01/01/2021 10:00 am	0.74805	3 (Otro Algoritmo)	RandomForest
15	01/01/2021 12:00 am	0.72562	3 (Cambio atributos)	XGBClassifier
16	01/01/2021 16:00 pm	0.71182	3 (Cambio Algoritmo)	GradientBoosting

Para cada algoritmo se ha usado una función previa o bucle que optimiza los parámetros, en el caso de *XGBClassifier* compruebo cual es la mejor semilla, para *GradientBoosting* uso distintos `learning_rate` que controlan el peso de cada uno de los árboles añadidos. Y para *RandomForest* calculo la profundidad óptima.

## 5.- Bibliografía

<https://xgboost.readthedocs.io/en/latest/>

[https://es.wikipedia.org/wiki/Gradient\\_boosting](https://es.wikipedia.org/wiki/Gradient_boosting)

<https://es.wikipedia.org/wiki/Boosting>

[https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification\\_report.html](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html)

<https://machinelearningmastery.com/random-oversampling-and-undersampling-for-imbalanced-classification/>

<https://www.kaggle.com/ddmngml/trying-to-predict-used-car-value>

<https://stackabuse.com/gradient-boosting-classifiers-in-python-with-scikit-learn/>

<https://rubikscore.net/2020/11/15/top-9-feature-engineering-techniques/>