

**Práctica 3: Competición en Kaggle.
Predicción de precio coche usado
(multiclase):**



**UNIVERSIDAD
DE GRANADA**

Luis González Romero, XXXXXXXXXX, luisgonromero@correo.ugr.es

Escuela Técnica Superior de Ingeniería informática y Telecomunicaciones

15 de enero de 2021

Práctica 3: Competición en Kaggle. Predicción de precio coche usado (multiclase)

Memoria sobre la tercera práctica de la asignatura Inteligencia de Negocio
cursada en la ETSIIT, UGR.

Luis González Romero, XXXXXXXXXX, luisgonromero@correo.ugr.es

Índice

1. Leaderboard	7
2. Predicción de precios de coches usados	8
2.1. Descripción del problema	8
3. Subidas a la plataforma Kaggle	11
3.1. Resumen de las subidas	11
3.2. Selección de columnas	13
3.3. Imputación de valores perdidos	13
3.4. Oversamplers	14
3.5. Undersamplers	14
3.6. subida 1	14
3.7. subida 2	14
3.8. subida 3	14
3.9. subida 4	15
3.10. subida 5	15
3.11. subida 6	15
3.12. subida 7	15
3.13. subida 8	16
3.14. subida 9	16
3.15. subida 10	16
3.16. subida 11	17
3.17. subida 14	18
3.18. subida 15	18
3.19. subida 16	18
3.20. subida 17	19
3.21. subida 18	19
3.22. subida 20	19

Índice de figuras

1.	Fila de la competición correspondiente	7
2.	Presencia de cada variable en las distintas clases representadas en Precio_cat	8
3.	Valores perdidos en el dataset de entrenamiento.	9
4.	Ocurrencias de las clases	10
5.	Representación gráfica de los score obtenidos en Kaggle.	11
7.	Obteniendo el valor de las columnas <i>Consumo</i> , <i>Motor_CC</i> y <i>Potencia</i> . . .	13
8.	Selección de columnas usado	13
9.	Hyperparameter tuning(XGB) - sub5	15
10.	<i>sklearn.preprocessing.StandardScaler</i>	16
11.	Classifier ranking - sub10	16
12.	GridSearchCV(RFC) - sub10	17
13.	RFC - sub10	17
14.	StackingClassifier - sub11	18
15.	SMOTETomek- sub16	18
16.	Aumento de instancias de la clase 1	19

Índice de tablas

1.	Subidas a la plataforma Kaggle	11
2.	Subidas con dos medidas	12

1. Leaderboard

En Kaggle me di de alta como LuisUGR, ya que en la primera clase en la que se explicó el guión se dijo que usaramos: -DNI o -UGR. Dias después, intenté cambiarlo pero siempre salía que se había cambiado con éxito, pero realmente no se aplicaba.


30	—	LuisUGR		0.76100	20	4d
----	---	---------	---	---------	----	----

Figura 1: Fila de la competición correspondiente

2. Predicción de precios de coches usados

2.1. Descripción del problema

Tenemos un conjunto de datos de vehículos usados, clasificados estos en 5 categorías: del 1 al 5 dependiendo del precio(desde los más baratos a más caros, respectivamente). Debemos de poder predecir la categoría a la que pertenece un coche(clasificación multiclase).

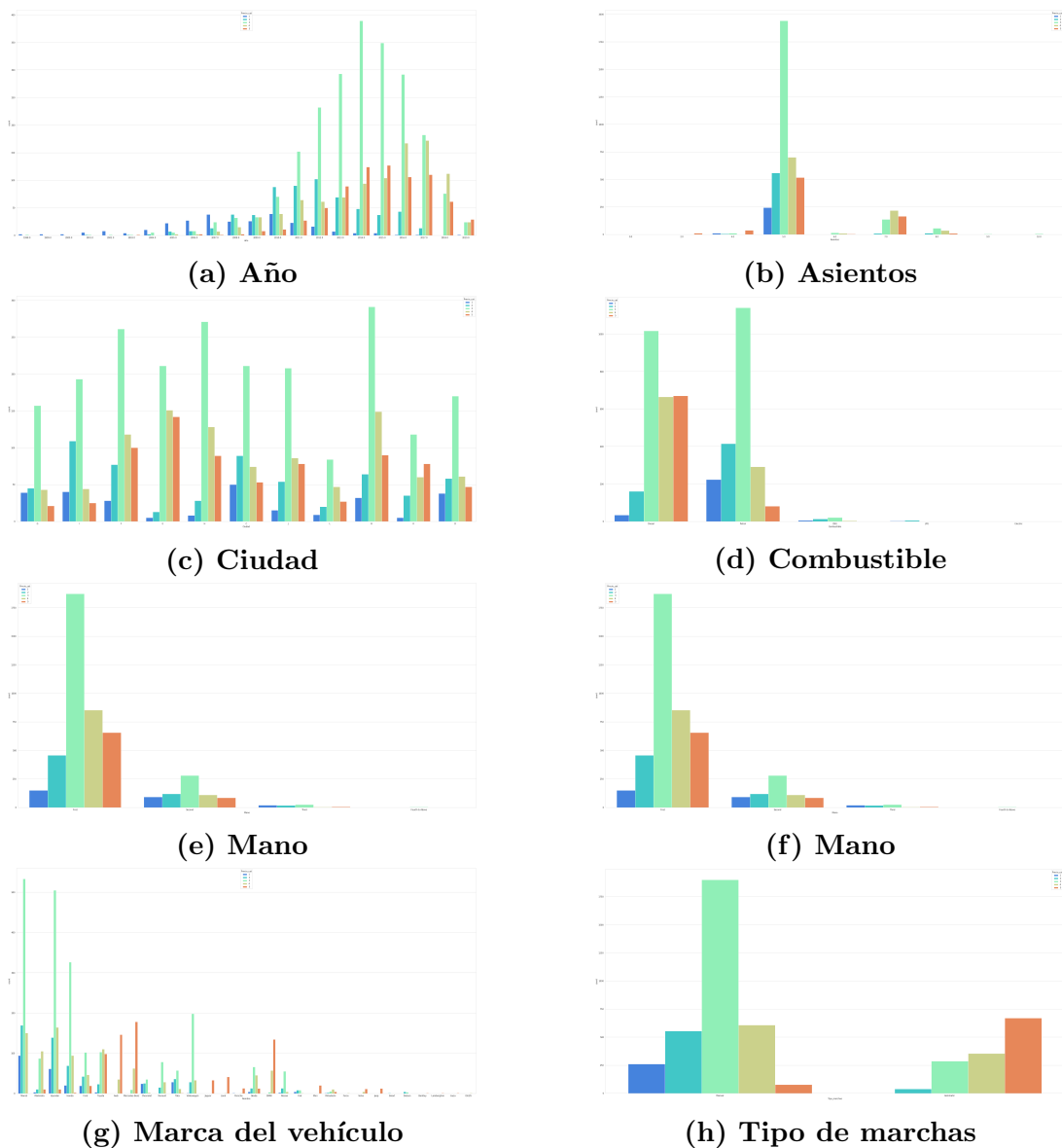


Figura 2: Presencia de cada variable en las distintas clases representadas en Precio_cat


```
train.isnull().sum()
```

```
id          72
Nombre      0
Ciudad      72
Año         72
Kilometros  72
Combustible 72
Tipo_marchas 0
Mano        0
Consumo     73
Motor_CC    101
Potencia    175
Asientos    106
Descuento   0
Precio_cat  0
dtype: int64
```

```
fig = sns.heatmap(train.isnull(), cbar=False, cmap='hot_r')
```



Figura 3: Valores perdidos en el dataset de entrenamiento.

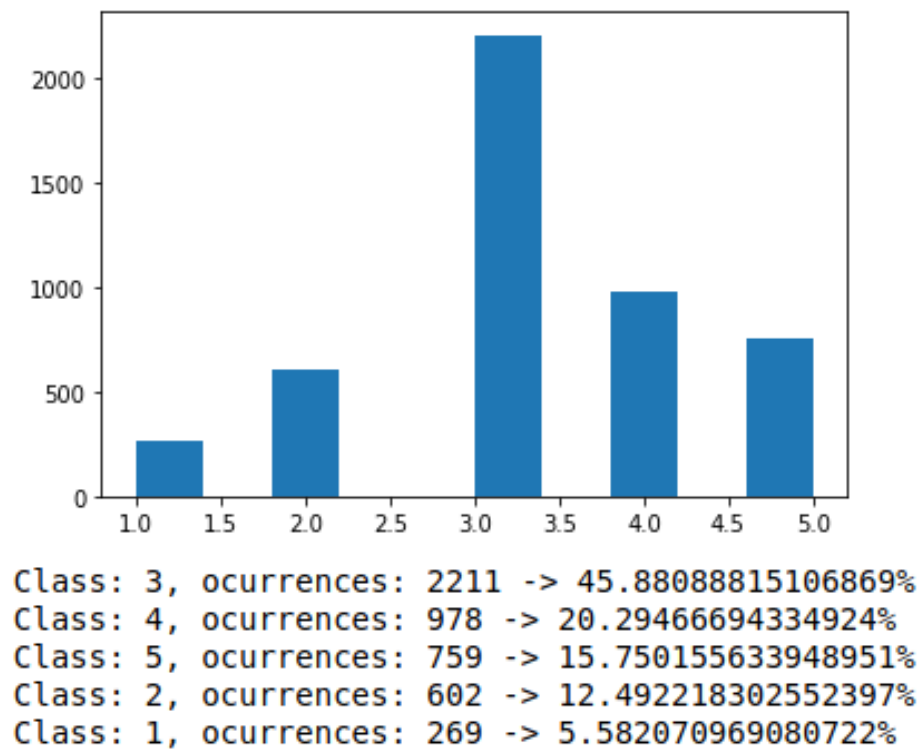


Figura 4: Ocurrencias de las clases

Como bien sabíamos, se trata de un problema con múltiples clases y en este caso, desbalanceadas.

3. Subidas a la plataforma Kaggle

3.1. Resumen de las subidas

Las subidas 13 y 19 no aparecen porque subí un csv donde la columna de etiquetas tenía formato erróneo (equivalentes a las posteriores a estas), por lo que obtuve score 0,00.

Submission	Date	Private Score	Local Score	Description
1	17-12	0,73943	0,801064	$prep_{sin_mv}$, rfc y $features_1$
2	17-12	0,72562	0,885873	$prep_{sin_mv}$, rfc, $features_1$ y $over_1$
3	17-12	0,72562	0,805373	$prep_{sin_mv}$, xgb y $features_1$
4	18-12	0,74547	0,901131	$prep_2$, xgb, $features_1$ y $over_1$
5	19-12	0,75496	0,889281	$prep_2$, xgb-tuning, $features_1$ y $over_1$
6	22-12	0,75754	0,800581	$prep_2$, lgb, $features_1$
7	22-12	0,64106	0,948383	$prep_2$, lgb, $features_1$, $over_1$, $under_1$
8	22-12	0,76100	0,970485	$prep_2$, lgb, $features_1$, $over_3$
9	23-12	0,62381	0,907092	$prep_2$, lgb, $features_1$, $over_4$
10	23-12	0,72303	0,800374	$prep_2$, rfc-gridsearch, $features_1$
11	24-12	0,73856	0,761826	$prep_2$, stacking, $features_1$
12	24-12	0,73770	0,712209	$prep_2$, stacking, $features_1$, $under_2$
14	25-12	0,72389	-	$prep_2$, catboost, $features_1$
15	25-12	0,72821	0,840249	$prep_2$, catboost, $features_1$
16	27-12	0,73425	0,747925	$prep_2$, lgb-tuning, smote + tomek_links
17	27-12	0,71613	0,718880	$prep_2$, lgb, smote + tomek_links
18	27-12	0,73684	0,786492	$prep_2$, lgb, smote + tomek_links
20	31-12	0,75323	0,738589	$prep_{knn}$, lgb, $features_2$

Tabla 1: Subidas a la plataforma Kaggle

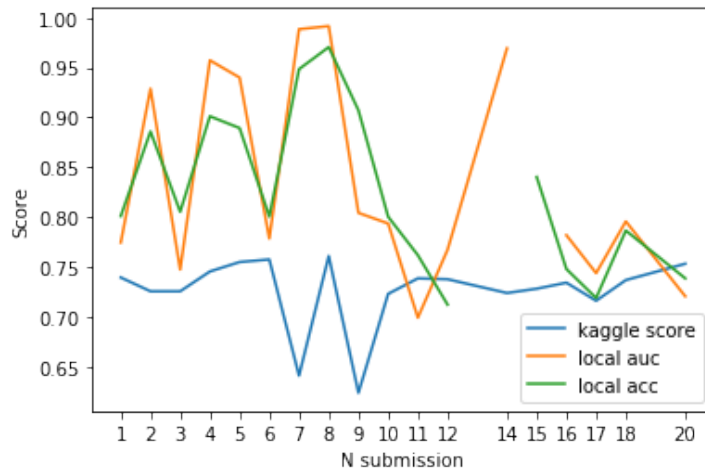


Figura 5: Representación gráfica de los score obtenidos en Kaggle.

Submission	Private Score	Local AUC	Local Acc
<i>1</i>	0,73943	0,774507	0,801064
<i>2</i>	0,72562	0,928768	0,885873
<i>3</i>	0,72562	0,747393	0,805373
<i>4</i>	0,74547	0,957258	0,901131
<i>5</i>	0,75496	0,939908	0,889281
<i>6</i>	0,75754	0,778514	0,800581
<i>7</i>	0,64106	0,988479	0,948383
<i>8</i>	0,76100	0,991525	0,970485
<i>9</i>	0,62381	0,804244	0,907092
<i>10</i>	0,72303	0,793589	0,800374
<i>11</i>	0,73856	0,699132	0,761826
<i>12</i>	0,73770	0,767308	0,712209
<i>14</i>	0,72389	0,969275	-
<i>15</i>	0,72821	-	0,840249
<i>16</i>	0,73425	0,781858	0,747925
<i>17</i>	0,71613	0,743823	0,718880
<i>18</i>	0,73684	0,795716	0,786492
<i>20</i>	0,75323	0,720588	0,738589

Tabla 2: Subidas con dos medidas

3.2. Selección de columnas

```
columns = ['Año',  
           'Kilometros',  
           'Combustible',  
           'Mano',  
           'Consumo',  
           'Motor_CC',  
           'Potencia',  
           'Asientos',  
           'Descuento']
```

(a) $features_1$

```
columns_train = ['Nombre',  
                 'Año',  
                 'Kilometros',  
                 #'Combustible',  
                 'Mano',  
                 'Consumo',  
                 'Motor_CC',  
                 'Potencia',  
                 'Tipo_marchas',  
                 #'Asientos',  
                 'Descuento']
```

(b) $features_2$

Se transformaron las columnas *Consumo*, *Motor_CC* y *Potencia* de cadena a *real*, haciendo uso de expresiones regulares.

```
import re  
import math  
  
for index, row in train.iterrows():  
    consumo = row.Consumo  
    motor = row.Motor_CC  
    potencia = row.Potencia  
  
    if type(consumo) == str:  
        train.loc[index, 'Consumo'] = float(re.findall(r"^\d+\.\d+", consumo)[0])  
    if type(motor) == str:  
        train.loc[index, 'Motor_CC'] = float(re.findall(r"^\d+", motor)[0])  
    if type(potencia) == str:  
        train.loc[index, 'Potencia'] = float(re.findall(r"^\d+", potencia)[0])  
  
for index, row in test.iterrows():  
    consumo = row.Consumo  
    motor = row.Motor_CC  
    potencia = row.Potencia  
  
    if type(consumo) == str:  
        test.loc[index, 'Consumo'] = float(re.findall(r"^\d+\.\d+", consumo)[0])  
    if type(motor) == str:  
        test.loc[index, 'Motor_CC'] = float(re.findall(r"^\d+", motor)[0])  
    if type(potencia) == str:  
        test.loc[index, 'Potencia'] = float(re.findall(r"^\d+", potencia)[0])
```

Figura 7: Obteniendo el valor de las columnas *Consumo*, *Motor_CC* y *Potencia*

Figura 8: Selección de columnas usado

3.3. Imputación de valores perdidos

- $prep_{sin_mv} \rightarrow data_without_nan = train.copy()$
 $data_without_nan = data_without_nan.dropna()$
- $prep_2 \rightarrow$ Imputación de valores nulos: categóricos-most_freq y numéricos-mean
- $prep_{knn} \rightarrow KNNImputer(n_neighbors = 5, weights = 'uniform', metric = 'nan_euclidean')$

3.4. Oversamplers

Para los *smote_variants*, he usado *MulticlassOversampling*, pasándole la variante que especifico como parámetro(oversampler=variante).

- *over₁* → *imblearn.over_sampling.SMOTE*
- *over₂* → *smote_variants.MDO*
- *over₃* → *smote_variants.DSRBF*
- *over₄* → *smote_variants.MSMOTE*

3.5. Undersamplers

- *under₁* → *smote_variants EditedNearestNeighbors()*
- *under₂* → *smote_variants.CondensedNearestNeighbour()*
- *under₃* → *Tomek links*

3.6. subida 1

La primera subida consite en usar Random Forest como clasificador, pasándole como parámetro la semilla que esta en todos los notebooks igual.

Para la validación cruzada se usará: *cv_ = StratifiedKFold(n_splits = 5, shuffle = True)*

```
rfc = RandomForestClassifier(random_state = random_seed)  
y_pred = cross_val_predict(rfc, X, y, cv = cv_, n_jobs = -1)
```

3.7. subida 2

Igual que la anterior pero haciendo oversampling.

```
from imblearn.over_sampling import SMOTE  
oversample = SMOTE()  
X, y = oversample.fit_resample(X, y)
```

3.8. subida 3

Igual que la primera pero usando *XGBClassifier* con parámetros por defecto.

3.9. subida 4

Igual que la anterior pero haciendo $prep_2$ y $SMOTE_{imblearn}$.

3.10. subida 5

Igual que el anterior pero haciendo tuning de parámetros.

```
params

{'max_depth': 19,
 'min_child_weight': 2,
 'eta': 0.05,
 'subsample': 1.0,
 'colsample_bytree': 1.0,
 'eval_metric': 'mae'}

from xgboost.sklearn import XGBClassifier
model = XGBClassifier(
    max_depth=params['max_depth'],
    learning_rate=params['eta'],
    njobs=-1,
    min_child_weight=params['min_child_weight'],
    subsample=params['subsample'],
    colsample_bytree=params['colsample_bytree'],
    random_state=seed
)
```

Figura 9: Hyperparameter tuning(XGB) - sub5

3.11. subida 6

Igual que sub3 pero usando *lgb* como clasificador.

$clf = \text{lgb.LGBMClassifier}(\text{random_state} = \text{seed}, n_jobs = -1)$

3.12. subida 7

Igual que la anterior pero haciendo uso de $over_2$ y $under_1$.

3.13. subida 8

Igual que sub6 pero haciendo uso de *over*₃.

3.14. subida 9

Igual que la anterior pero empleando *over*₄.

3.15. subida 10

A partir de esta subida, realizo una estandarización de los datos de entrenamiento y test.

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_ = sc.fit_transform(X)
test_ = sc.transform(test)
```

Figura 10: *sklearn.preprocessing.StandardScaler*

Para esta subida hice un ranking previo de clasificadores para escoger uno al que realizar un *GridSearchCV*, en este caso mirando F1-score.

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean
RandomForestClassifier	0.802905	0.821577	0.780083	0.796680	0.807892	0.801827
LGBMClassifier	0.799793	0.811203	0.786307	0.792531	0.818276	0.801622
CatBoostClassifier	0.804979	0.804979	0.782158	0.797718	0.806854	0.799338
XGBoostClassifier	0.795643	0.804979	0.776971	0.790456	0.802700	0.794150
GradientBoostingClassifier	0.792531	0.800830	0.774896	0.779046	0.801661	0.789793
Radial SVM	0.748963	0.786307	0.756224	0.751037	0.744548	0.757416
KNeighborsClassifier	0.753112	0.768672	0.730290	0.738589	0.753894	0.748912
Linear SVM	0.754149	0.759336	0.731328	0.731328	0.744548	0.744138
LogisticRegression	0.750000	0.752075	0.727178	0.725104	0.732087	0.737289
AdaBoostClassifier	0.612033	0.622407	0.545643	0.625519	0.563863	0.593893

Figura 11: Classifier ranking - sub10


```

'''param_grid = {
    'n_estimators': [ 200,300,500],
    'max_features': ['auto', 'sqrt'],
    'max_depth' : [6,7,8,9,10,11,12],
    'criterion' :['gini', 'entropy']
}

rfc = RandomForestClassifier(random_state=seed)
from sklearn.model_selection import GridSearchCV
CV_rfc = GridSearchCV(estimator=rfc, param_grid=param_grid, cv= 5)
CV_rfc.fit(X_,y)
CV_rfc.best_params_'''

```

Figura 12: GridSearchCV(RFC) - sub10

Tras eso, realizo un ajuste de parámetros en base a lo obtenido.

```

from sklearn.ensemble import RandomForestClassifier
rfc_ = RandomForestClassifier(random_state=seed,
                             n_estimators= 200,
                             criterion = 'entropy',max_features = 'auto',max_depth=10)

```

Figura 13: RFC - sub10

3.16. subida 11

A partir de esta subida, comencé a crear un dataset de validación (parte pequeña de train como test). Haciendo fit de la parte de entrenamiento restante y predicción solo de la parte de test.

Para esta subida hice muchas pruebas variando *estimators* y *final_estimator*. Dejando esta para la subida a kaggle.

```

'''estimators = [
    ('rf1', RandomForestClassifier(random_state=seed, n_estimators=200, criterion='entropy')),
    ('knn', KNeighborsClassifier()),
    ('nn', MLPClassifier(random_state=seed)),
    ('lgb', lgb.LGBMClassifier(random_state=seed, n_jobs=-1))
]'''
estimators = [
    ('lr', LogisticRegression(random_state=seed, multi_class='ovr', solver='liblinear', C=1)),
    ('gbc', GradientBoostingClassifier(random_state=seed)),
    ('rf1', RandomForestClassifier(random_state=seed, n_estimators=200, criterion='gini')),
    ('rf2', RandomForestClassifier(random_state=seed, n_estimators=200, criterion='entropy')),
    ('et1', ExtraTreesClassifier(random_state=seed, n_estimators=200, criterion='gini')),
    ('et2', ExtraTreesClassifier(random_state=seed, n_estimators=200, criterion='entropy')),
    ('lgb', lgb.LGBMClassifier(random_state=seed, n_jobs=-1)),
    ('nn', MLPClassifier(random_state=seed))
]

#final_est = RandomForestClassifier(random_state=seed)
final_est = CatBoostClassifier(random_state=seed)
#final_est = lgb.LGBMClassifier(random_state=seed, n_jobs=-1)
clf = StackingClassifier(
    estimators=estimators, final_estimator=final_est, cv=cv_
)

```

Figura 14: StackingClassifier - sub11

3.17. subida 14

Usé como clasificador, CatBoostClassifier con los siguientes parámetros: *loss_function = 'MultiClass'*, *eval_metric = 'AUC'*.

3.18. subida 15

CatBoostClassifier con los siguientes parámetros: *loss_function = 'MultiClass'*, *eval_metric = 'Accuracy'*.

3.19. subida 16

Uso de SMOTETomek (combinando over y undersampling), con lgb como clasificador con tuning de parámetros. Viendo la baja accuracy que siempre tiene la clase 1, intento balancear algo las instancias de esa clase pasando de un 5.58% a un 10.09%. Siendo este realizado solo a la parte de entrenamiento de la parte de validación.

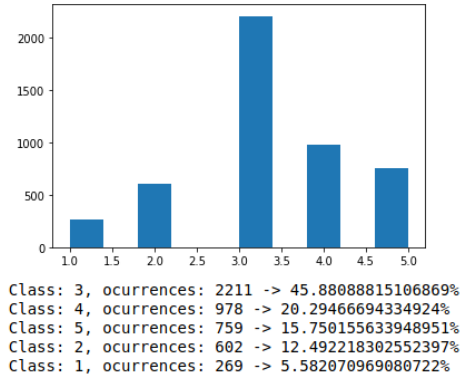
clf = lgb.LGBMClassifier(random_state = seed, n_jobs = -1, colsample_bytree = 0,5, num_leaves = 10, subsample = 0,5)

```

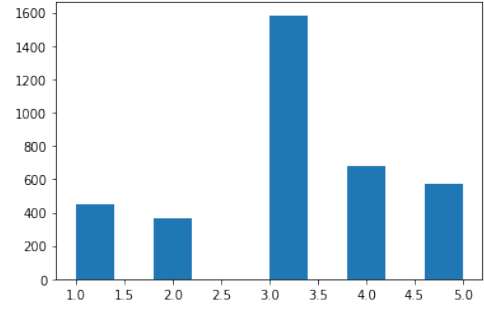
smt = SMOTETomek(sampling_strategy={1: 482, 2: 482, 3: 1764, 4: 792, 5: 600}, n_jobs=-1)
X_smote, y_smote = smt.fit_sample(X_train, y_train)

```

Figura 15: SMOTETomek- sub16



(a) Dataset original



(b) Dataset tras SMOTETomek

Figura 16: Aumento de instancias de la clase 1

3.20. subida 17

Igual que la anterior, pero en este caso {1: 471, 2: 482, 3: 1764, 4: 792, 5: 600} como diccionario y *lgb* con configuración de parámetros por defecto.

3.21. subida 18

Igual que el anterior, pero intentando subir la precisión de la clase 1, *sample_strategy* = {1 : 600, 2 : 602, 3 : 2211, 4 : 978, 5 : 759}

3.22. subida 20

A la vista de no mejorar mucho el score, cambié las columnas usadas y la imputación de valores perdidos. Descarté el combustible porque no veía que pudiera ayudar a mejorar la precisión de las clases minoritarias y los asientos porque casi todos pertenecen a 5 asientos. Y añadí la marca del coche porque puede haber marcas destinadas a ciertos tipo de clientes.

Guardé en la columna *Nombre*, solo la marca del coche dejando como 'mv' los valores perdidos. Al hacer el LabelEncoder, mediante la transformación inversa obtuve de nuevo los valores perdidos para imputarlos con *KNNImputer*.

4. Conclusiones

Claramente se ha producido overfitting en muchos casos de intento de balanceo de clases y ajuste de parámetros, pero no conseguí que pudiera generalizar el modelo sin dejar de lado la precisión de este y así obtener mejores resultados en conjuntos de datos distintos a los de entrenamiento.

Durante y después de la realización de la práctica lo que más he notado es lo difícil que es llevar lo teórico a la parte práctica. Al final he obtenido el mejor resultado de un modelo configurado por defecto y un oversampling específico.

Debería haber probado más preprocesamientos así como un mejor estudio de las columnas a elegir. Quizás también más clasificadores, pero el ultimo que probé OneVersusOne, me seguía yendo peor en local así que pensé que podría estar limitandome el preprocesamiento de los datos y/o elección de las columnas.

Referencias

- [1] PyPI SMOTE-variants. smote-variants 0.4.0.
- [2] Shihab Shahriar Khan. Introduction to Scikit-clean.
- [3] PyPI. scikit-clean 0.1.2.
- [4] V. López. A. Fernandez. S. Garcia. V. Palade and F. Herrera. An Insight into Classification with Imbalanced Data: Empirical Results and Current Trends on Using Data Intrinsic Characteristics.
- [5] Soft Computing and Intelligent Information Systems. A University of Granada research group. Noisy Data in Data Mining.