Práctica 3 Competición en DrivenData

Paula Villanueva Núñez

49314567Z

pvillanunez@correo.ugr.es

Cuarto Curso del Grado en Ingeniería Informática Curso 2021-2022 Grupo 1

Universidad de Granada

1 Tabla *Submissions*

SUBMISSIONS

Score	Submitted by	Timestamp
0.8517	Paula_Villanueva_UGR_IN	2021-12-27 15:48:41 UTC
0.8530	Paula_Villanueva_UGR_IN	2021-12-27 16:34:21 UTC
	Paula_Villanueva_UGR_IN	2021-12-27 16:40:10 UTC
0.8556	Paula_Villanueva_UGR_IN	2021-12-28 16:44:53 UTC
0.8543	Paula_Villanueva_UGR_IN	2021-12-28 17:13:50 UTC
0.8605	Paula_Villanueva_UGR_IN	2021-12-28 17:30:21 UTC
0.8605	Paula_Villanueva_UGR_IN	2021-12-29 16:53:08 UTC
0.8621	Paula_Villanueva_UGR_IN	2021-12-29 17:19:58 UTC
0.8624	Paula_Villanueva_UGR_IN	2021-12-29 17:33:03 UTC
0.8624	Paula_Villanueva_UGR_IN	2022-01-01 16:35:29 UTC

Figura 1: Tabla *Submissions*

Índice

2 Introducción

Esta práctica ha consistido en una competición real disponible en DrivenData. Para ello, se han utilizado métodos avanzados para aprendizaje supervisado y preprocesado en clasificación. De esta forma, se ha intentado mejorar la capacidad predictiva del modelo.

La competición es la *Flu Shot Learning: Predict H1N1 and Seasonal Flu Vaccines* disponible en https://www.drivendata.org/competitions/66/flu-shot-learning/page/211/. El objetivo es predecir si una persona se vacuna contra la gripe H1N1 y la gripe estacional utilizando información sobre sus antecedentes, opiniones y comportamientos de salud.

El conjunto de entrenamiento consta de 26.707 instancias y 36 atributos (de los cuales, `respondent_id` toma valores únicos y solo sirve para identificar cada ejemplo) categóricos, enteros y binarios. Para este concurso, hay dos variables objetivo:

- `h1n1_vaccine`: si el encuestado recibió la vacuna contra la gripe H1N1.
- `seasonal_vaccine`: si el encuestado recibió la vacuna contra la gripe estacional.

Ambas son variables binarias: 0 = No; 1 = Sí. Se trata de un problema multietiqueta, pues algunos encuestados no recibieron ninguna de las dos vacunas, otros recibieron solo una, y algunos recibieron ambas.

El rendimiento se evaluará según el área bajo la curva ROC (AUC) para cada una de las dos variables objetivo. La media de estas dos puntuaciones será la puntuación global.

3 Exploración de los datos

3.1 Distribución de los datos

Antes de probar los algoritmos, realizaremos un análisis sobre los datos. Esto es, cómo están distribuidos, si hay valores perdidos, si hay correlación entre las variables, etc. De esta forma, podremos realizar un mejor preprocesado o escoger entre varios algoritmos.

La distribución de las etiquetas del conjunto de entrenamiento es el siguiente.

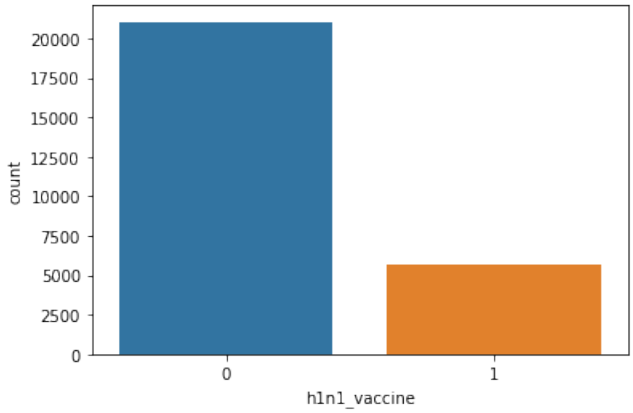

Figura 2: Tabla *Submissions*

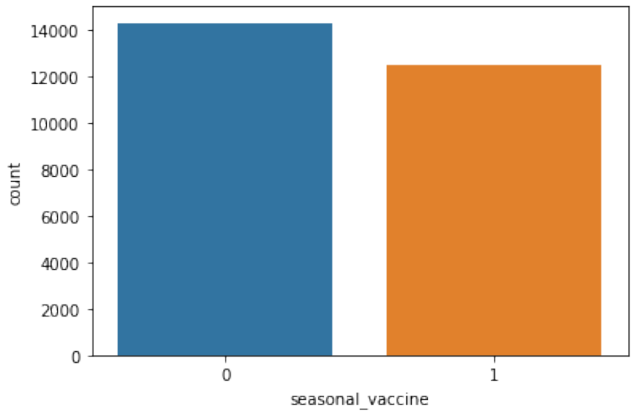

Figura 3: Tabla *Submissions*

Como podemos observar, existe un desbalanceo de clases. La mayoría de las personas no se vacunaron contra la gripe H1N1. Sin embargo, con la vacuna de la gripe estacional no notamos tanta diferencia.

3.2 Preprocesado

Los ficheros .csv se han preparado previamente para sustituir los valores desconocidos por NaN (**valores perdidos**).

También se ha **eliminado las columnas** que no se usan, como `respondent_id`, ya que solo identifica cada ejemplo.

```
data_x.drop(labels=['respondent_id'], axis=1,inplace = True)
data_x_tst.drop(labels=['respondent_id'], axis=1,inplace = True)
data_y.drop(labels=['respondent_id'], axis=1,inplace = True)
data_all_features = pd.concat([data_x, data_x_tst])
```

Otra técnica que se ha utilizado es la eliminación de **valores nulos** con `SimpleImputer`.

```
imp = SimpleImputer(missing_values=np.nan, strategy='
    most_frequent')
data_x_tmp = data_x.astype(str)
# Uso SimpleImputer para "eliminar los nulos", como elimina los
    nombres de las columnas los vuelvo a poner
cols = data_x_tmp.columns
data_x_tmp = pd.DataFrame(imp.fit_transform(data_x_tmp))
data_x_tmp.columns = cols
```

Por último, se ha realizado un **etiquetado** de los datos, pues aunque la mayoría de los atributos ya están etiquetados, no pasa con todos.

```
# Aprendo las etiquetas
for col in data_all_features.columns:
  labels[col] = LabelEncoder()
  labels[col].fit(data_all_features[col].astype(str))

# Aplico el etiquetado
for col in data_x_tmp.columns:
  data_x_tmp[col] = labels[col].transform(data_x_tmp[col])

# Conjunto final de aprendizaje
X = data_x_tmp
y = data_y.values

data_x_tmp = data_x_tst.astype(str)
data_x_tmp = pd.DataFrame(imp.transform(data_x_tmp))
data_x_tmp.columns = cols

# Aplico el mismo etiquetado con los valores de test
for col in data_x_tmp.columns:
  data_x_tmp[col] = labels[col].transform(data_x_tmp[col])

# Conjunto final de test
X_tst = data_x_tmp.values
```

Se ha intentado hacer más preprocesamientos, como el tratamiento de outliers, o normalizar valores en un intervalo, etc. pero los resultados obtenidos han resultado ser peores que sin el preprocesamiento.

4 Soluciones subidas a DrivenData

En este apartado se muestran las soluciones subidas a DrivenData resumiendo cada experimento.

Tabla 1: Soluciones subidas a DrivenData

ID	Fecha y hora	Posición	Score (entrenamiento)	Score (test)	Preprocesado	Algoritmo	Parámetros
00	2021-12-27 15:48:41 UTC	436	0.8551	0.8517	Valores perdidos, eliminación de columnas, valores nulos, etiquetado	*Random Forest*	
01	2021-12-27 16:34:21 UTC	376	0.8593	0.8530	Valores perdidos, eliminación de columnas, valores nulos, etiquetado	*Random Forest*	`n_estimators=500` `max_depth=10` `max_features="sqrt"` `random_state=123456`
02	2021-12-28 16:44:53 UTC	336	0.8603	0.8556	Valores perdidos, eliminación de columnas, valores nulos, etiquetado	*Random Forest*	`n_jobs=-1` `n_estimators=600` `max_depth=15` `max_features="sqrt"` `random_state=123456`
03	2021-12-28 17:13:50 UTC	349	0.8614	0.8543	Valores perdidos, eliminación de columnas, valores nulos, etiquetado	*Ada Boost*	`n_estimators=600` `learning_rate=0.5` `random_state=123456`
04	2021-12-28 17:30:21 UTC	238	0.8670	0.8605	Valores perdidos, eliminación de columnas, valores nulos, etiquetado	*Gradient Boosting*	`n_estimators=300` `criterion="squared_error"` `random_state=123456`
05	2021-12-29 16:53:08 UTC	238	0.8675	0.8605	Valores perdidos, eliminación de columnas, valores nulos, etiquetado	*Gradient Boosting*	`n_splits=15` `n_estimators=300` `criterion="squared_error"` `random_state=123456`
06	2021-12-29 17:19:58 UTC	134	0.8690	0.8621	Valores perdidos, eliminación de columnas, valores nulos, etiquetado	*LightGBM*	`num_leaves=20` `n_estimators=1000` `min_child_samples=30` `colsample_bytree=0.3` `reg_alpha=8` `reg_lambda=8`
07	2021-12-29 17:33:03 UTC	108	0.8689	0.8624	Valores perdidos, eliminación de columnas, valores nulos, etiquetado	*LightGBM*	`n_estimators=1000` `colsample_bytree=0.3` `reg_alpha=8` `reg_lambda=15`
08	2022-01-01 16:35:29 UTC	112	0.8689	0.8624	Valores perdidos, eliminación de columnas, valores nulos, etiquetado	*LightGBM*	`n_estimators=400` `colsample_bytree=0.3` `reg_alpha=8` `reg_lambda=15`

5 Estrategias seguidas y progreso

5.1 Validación cruzada

Se ha realizado una validación cruzada con KFold y control de aleatoriedad fijando la semilla.

5.2 Estructura del programa

Una vez realizado el preprocesamiento, en esta sección indicaremos cómo se han utilizado los algoritmos.

```
print("------ RandomForestClassifier...")

# Creo el modelo normal
rf = RandomForestClassifier() # Algoritmo utilizado
# Multioutput classifier, para indicar que tiene que aprender varias etiquetas a la vez
multi = MultiOutputClassifier(rf)
# Aplica validación cruzada y devuelve el modelo
clf1 = validacion_cruzada(multi,X,data_y,skf)

# Aprendo con todos los ejemplos
clf1 = clf1.fit(X,data_y)
# Aplico probabilidad
preds = clf1.predict_proba(X_tst)
df_submission = pd.read_csv('submission_format.csv')

# Reformateo la salida
y_test_preds = pd.DataFrame(
      {
        "h1n1_vaccine": preds[0][:, 1],
        "seasonal_vaccine": preds[1][:, 1],
      },
      index = df_submission.index
    )

# Modifico las columnas con las predichas
df_submission['h1n1_vaccine'] = y_test_preds.h1n1_vaccine
df_submission['seasonal_vaccine'] = y_test_preds.seasonal_vaccine
# Escribo el fichero de salida
df_submission.to_csv("submission_ejemplo_flu_rf_00_0.csv", index=False)
```

5.3 flu_00

En este primer intento, se ha subido la plantilla proporcionada por los profesores para tener una referencia desde dónde partir. En este caso, se ha utilizado el algoritmo **Random Forest** sin parámetros:

```
1   rf = RandomForestClassifier()
```

El score obtenido en DrivenData es de 0.8517, bastante elevado pues actualmente el primer puesto tiene un score de 0.8658.

5.4 flu_01

En este segundo intento, se ha realizado un **análisis de parámetros** del algoritmo Random Forest aplicado a este problema. La siguiente gráfica representa cómo varía el AUC con respecto a `n_estimators` y `max_depth`.

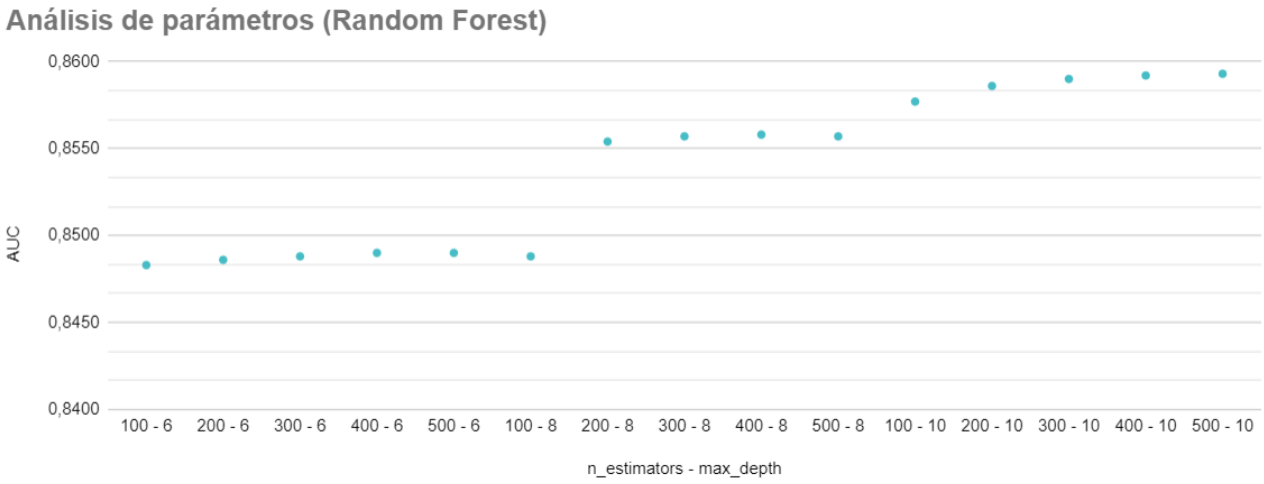

Figura 4: Tabla *Análisis de parámetros de Random Forest*

De esta forma, se ha ejecutado este algoritmo con los mejores parámetros obtenidos, que son los siguientes.

```
1   rf = RandomForestClassifier(n_estimators=500, max_depth=10,
      max_features="sqrt", random_state=123456)
```

Así, se ha conseguido incrementar el score obtenido en DrivenData y se ha obtenido un 0.8530.

5.5 flu_02

Al igual que en el intento anterior, se ha continuado analizando los parámetros de Random Forest y se ha conseguido mejorar todavía mas la puntuación.

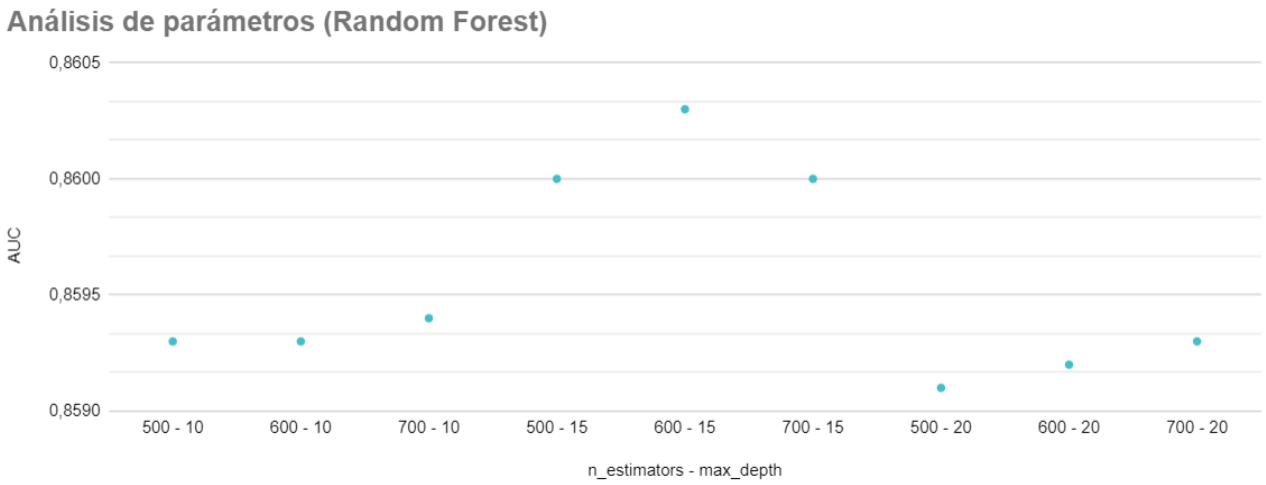

Figura 5: Tabla *Análisis de parámetros de Random Forest*

```
1   rf = RandomForestClassifier(n_jobs=-1, n_estimators=600,
      max_depth=15, max_features="sqrt", random_state=123456)
```

Con estos parámetros, se ha obtenido en DrivenData un score de 0.8556.

5.6 flu_03

En este intento, se ha decidido cambiar de algoritmo y se ha utilizado **AdaBoost**. Se ha realizado el siguiente análisis de parámetros .

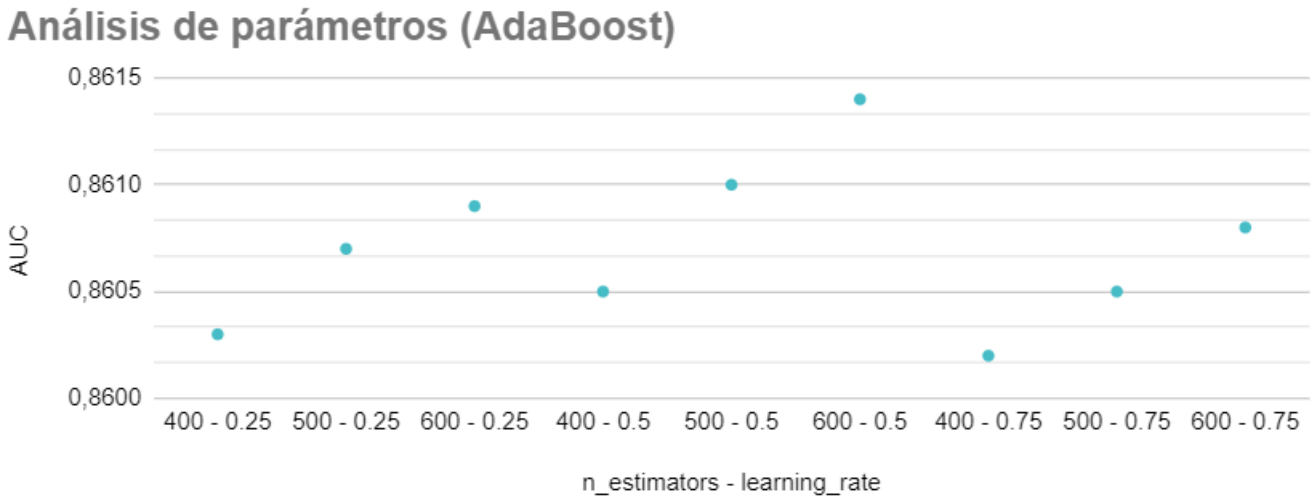

Figura 6: Tabla *Análisis de parámetros de AdaBoost*

De esta forma con la siguiente configuración, se ha obtenido en DrivenData un score de 0.8543, algo menor al intento anterior.

```
1   ab = AdaBoostClassifier(n_estimators=600, learning_rate=0.5,
      random_state=123456)
```

5.7 flu_04

En este intento, se ha utilizado el algoritmo **GradientBoost** y también se ha realizado un análisis de parámetros sobre él, siendo la mejor configuración la que se muestra a continuación.

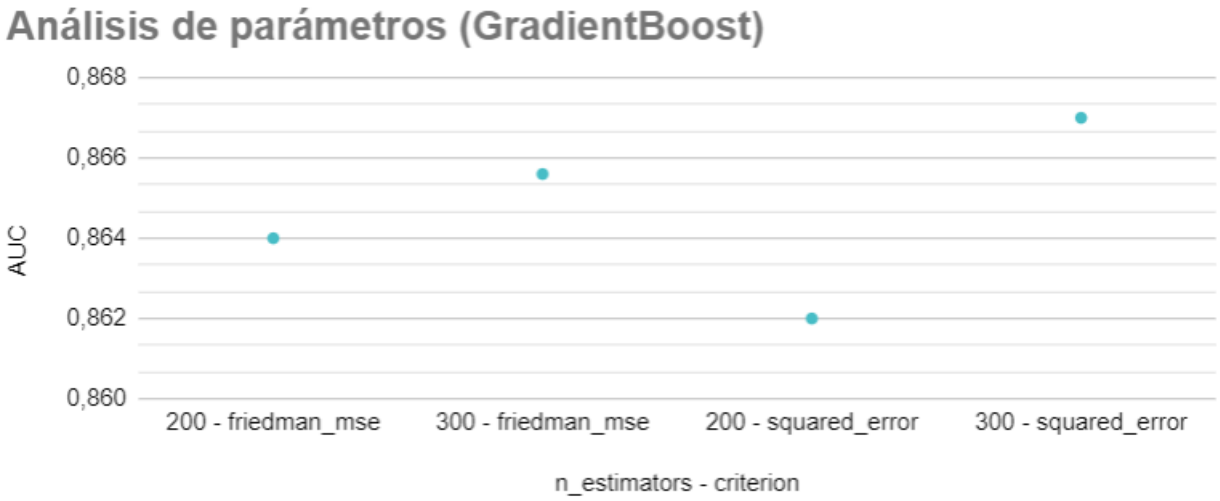

Figura 7: Tabla *Análisis de parámetros de GradientBoost*

```
1   gb = GradientBoostingClassifier(n_estimators=300, criterion="
      squared_error", random_state=123456) # 0.8670
```

En DrivenData se ha obtenido un score de 0.8605.

5.8 flu_05

En este caso, tomando el fichero del intento anterior, se han cambiado los parámetros de KFold:

```
1   skf = KFold(n_splits=15, shuffle=True, random_state=123456)
```

Sin embargo, al subirlo a DrivenData, hemos obtenido la misma puntuación, 0.8605.

5.9 flu_06

En este intento, se ha utilizado otro algoritmo, **LGBM**. Se ha realizado un análisis de sus parámetros y la mejor configuración obtenida hasta el momento ha sido la siguiente.

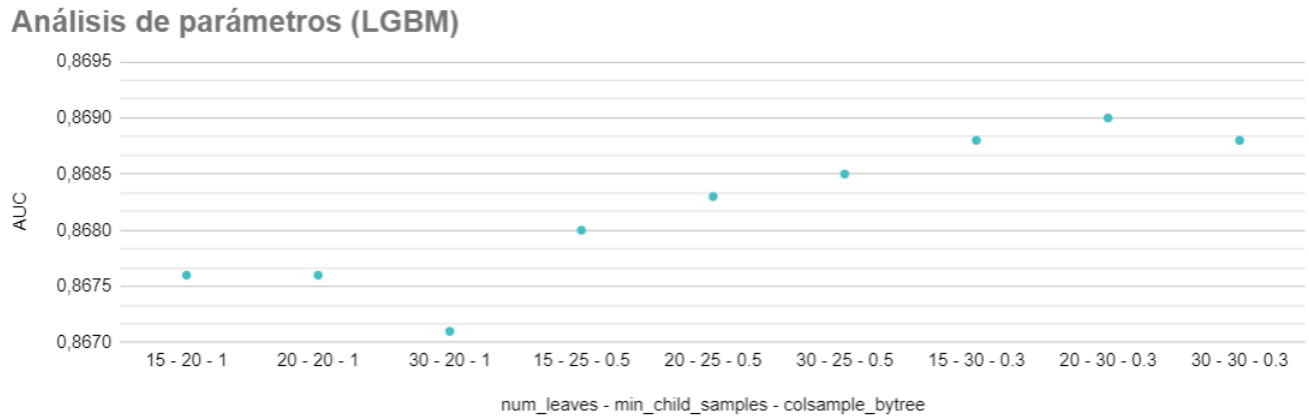

Figura 8: Tabla *Análisis de parámetros de LGBM*

```
lgbm = LGBMClassifier(num_leaves=20, learning_rate=0.1,
   n_estimators=1000, min_child_samples=30, colsample_bytree=0.3,
    reg_alpha=8, reg_lambda=8)
```

Al subirlo a DrivenData, hemos obtenido un score más elevado, 0.8621.

5.10 flu_07

Al obtener una puntuación tan elevada, se ha intentado realizar un análisis de parámetros más exhaustivo sobre el algoritmo LGBM y se ha llegado a la conclusión de que los siguientes parámetros aportan mejora.

```
lgbm = LGBMClassifier(n_estimators=1000, colsample_bytree=0.3,
   reg_alpha=8, reg_lambda=15)
```

En DrivenData hemos subido la puntuación hasta 0.8624.

5.11 flu_08

Como el intento anterior tardaba demasiado, hemos intentado bajar el número de estimadores pero sin empeorar el resultado. De esta forma, con los siguientes parámetros hemos seguido obteniendo la misma puntuación, 0.8624, pero con un tiempo de ejecución mucho menor.

```
lgbm = LGBMClassifier(n_estimators=400, colsample_bytree=0.3,
   reg_alpha=8, reg_lambda=15)
```

5.12 Otros intentos

Se ha intentado probar los algoritmos HistGradient, DecisionTree, Bagging, NaiveBayes, etc. pero no se han obtenido buenos resultados por lo que no ha sido necesario subir estos resultados a DrivenData.

6 Bibliografía

- Material proporcionado por los profesores sobre la asignatura.
  https://pradogrado2122.ugr.es/
- Métodos Ensemble
  https://scikit-learn.org/stable/modules/classes.html#module-sklearn.ensemble