Pr?ctico_3_Intro_learning

August 20, 2019

Universidad Nacional de Córdoba - Facultad de Matemática, Astronomía, Física y Computación

Diplomatura en Ciencia de Datos, Aprendizaje Automático y sus Aplicaciones

Práctico 3 y 4 - Introducción al aprendizaje automático - Aprendizaje Supervisado - 2019

Integrante:

- Tarletta Juan

Trabajaremos con el mismo dataset utilizado en curación de datos (deben utilizar el dataset que se encuentra con los datos modificados o bien aplicar los mismos métodos del trabajo anterior para contar con datos válidos)

El dataset cuenta con X features, siendo las más importantes

- Componente: Indica a que componente pertenece la muestra
- Horas Funcionamiento: Indica la cantidad de horas de funcionamiento del camión (sería como el kilometraje de los camiones)
- Horas del Aceite: Representa la cantidad de horas de utilización del aceite (este dato es importante dado que a medida que, a mayor horas de uso del aceite, el mismo comienza a desgastarse)
- Resultado: (El laboratorio indica si la muestra de aceite está Bien = 1, Regular=2, Mal=3)
- St: Presencia de Hollin en el Aceite
- Al: Presencia de Aluminio en el Aceite
- Fe Presencia de Hierro en el Aceite
- Si Presencia de Silicio en el Aceite
- Na Presencia de Sodio en el Aceite

NOTA: se modifica le dataset y se incluyen el feature * Fecha de Análisis: Indica cuando fué analizada la muestra por el laboratorio

[1]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn
from IPython.display import display, Markdown
```

[2]:
```
seaborn.__version__
```

[2]:
```
'0.9.0'
```

[3]:
```
dataset = pd.read_csv('dataset_intro.csv')
important_cols= ['Equipo', 'Componente','Resultado','Horas␣
 ↪Funcionamiento','Horas del Aceite', 'St', 'Al', 'Fe', 'Si', 'Na' ]

#dataset[important_cols]
```

[4]:
```
dataset[['Na']].count()
```

[4]:
```
Na    21432
dtype: int64
```

El objetivo del práctico es utilizar gridSearch para encontrar los mejores hiperparámetros y función de coste como lo vieron en la notebook Clase 3 - Metricas y validacion de resultados.ipynb

Deben tomar los datos de un solo componente para que los parámetros seán comparables (seleccionar el componente para el que exista la mayor cantidad de muestras en el dataset)

Evaluen diferentes funciones de coste y para cada uno de ellos diferentes hiperparámetros.

Determinar cual es el mejor set de hiperparámetros para clasificar las muestras de aceite

[5]:
```
import numpy as np
import matplotlib.pyplot as plt

from ml.visualization import plot_confusion_matrix, plot_learning_curve
from sklearn.datasets import load_wine
from sklearn.linear_model import SGDClassifier
from sklearn.metrics import accuracy_score, classification_report,␣
 ↪confusion_matrix
from sklearn.model_selection import GridSearchCV, train_test_split

np.random.seed(1234)

%matplotlib inline
```

[6]:
```
#Obtenemos el componente con más apariciones
compo = dataset[['Componente']].mode()
display(compo)
```

```
  Componente
0      Motor
```

[7]:
```
#Redefinimos el dataset a utilizar, en este caso solo con el componente Motor
dataset = dataset[dataset['Componente'] == 'Motor']
dataset.head(5)
```

[7]:

	Unnamed: 0	Equipo	Componente	Id	Resultado	Horas Funcionamiento	\
3	3	1355	Motor	76.084008	3.0	21950.0	
12	12	1357	Motor	41.099310	1.0	27876.0	
29	29	1356	Motor	82.751875	3.0	28295.0	
49	49	1355	Motor	79.207921	3.0	22729.0	
50	50	1355	Motor	82.456346	2.0	23153.0	

```
    Horas del Aceite Fecha de Análisis      B   Nit  ...      Zn   Ag   Ti  \
3              409.0        2019-02-20   51.0   9.0  ...  1089.0  0.0  0.0
12             263.0        2017-11-05    NaN   6.0  ...  1246.0  0.0  0.0
29               NaN        2019-05-17   47.0  10.0  ...  1122.0  0.0  0.0
49             380.0        2019-04-03  100.0   6.0  ...  1334.0  0.0  0.0
50             424.0        2019-05-12  105.0   8.0  ...  1331.0  0.0  0.0

    V40  V100   TBN  TAN  ISO14  ISO4  ISO6
3   NaN  13.7   9.1  NaN    NaN   NaN   NaN
12  NaN  14.0   NaN  NaN    NaN   NaN   NaN
29  NaN  14.2  10.4  NaN    NaN   NaN   NaN
49  NaN  14.1   9.8  NaN    NaN   NaN   NaN
50  NaN  14.3   9.6  NaN    NaN   NaN   NaN

[5 rows x 45 columns]
```

[8]:
```
dataset.columns
```

[8]:
```
Index(['Unnamed: 0', 'Equipo', 'Componente', 'Id', 'Resultado',
       'Horas Funcionamiento', 'Horas del Aceite', 'Fecha de Análisis', 'B',
       'Nit', 'Oxi', 'Sul', 'St', 'V', 'Al', 'Cr', 'Cu', 'Fe', 'Pb', 'Mo',
       'Ni', 'Sn', 'Si', 'K', 'Na', 'W', 'F', 'A', 'ISO', 'PQI', 'Ba', 'Ca',
       'Mg', 'Mn', 'P', 'Zn', 'Ag', 'Ti', 'V40', 'V100', 'TBN', 'TAN', 'ISO14',
       'ISO4', 'ISO6'],
      dtype='object')
```

[9]:
```
#utilizamos los datos que estan completos
dataset_reduce = dataset[['Horas Funcionamiento','Horas del Aceite','Sul', 'V',␣
 ↪'Cr','Cu','Pb','Mo','Ni','K','St','PQI','Al', 'Fe', 'Si','Ca','Na','Mg',␣
 ↪'Zn','V100','B','Oxi','P']].dropna()
index_dataset_reduce = dataset_reduce.index.values.astype(int)
index_dataset_reduce.shape
```

[9]:
```
(1982,)
```

[10]:
```
#Separamos entre Datos y objetivo

X = dataset_reduce[['Horas Funcionamiento','Horas del Aceite', 'Sul', 'V',␣
 ↪'Cr','Cu','Pb','Mo','Ni','K','St','PQI','Al', 'Fe', 'Si','Ca','Na','Mg',␣
 ↪'Zn','V100','B','Oxi','P']]
y = dataset[['Resultado']].ix[index_dataset_reduce]
#display(X.shape,y.shape,X,y)
```

```
C:\Users\Juan\Anaconda3\envs\diplodatos\lib\site-
packages\ipykernel_launcher.py:4: DeprecationWarning:
.ix is deprecated. Please use
.loc for label based indexing or
.iloc for positional indexing

See the documentation here:
```

```
http://pandas.pydata.org/pandas-docs/stable/indexing.html#ix-indexer-is-
deprecated
  after removing the cwd from sys.path.
```

[11]:
```
#Vemos la cantidad total de cada resultado
dataset[dataset['Resultado']==1.0].shape[0], dataset[dataset['Resultado']==2.0].
 ↪shape[0], dataset[dataset['Resultado']==3.0].shape[0]
```

[11]:
```
(1543, 509, 435)
```

[12]:
```
#Vemos la cantidad de cada resultado luego del dropna
y[y['Resultado']==1.0].shape[0],y[y['Resultado']==2.0].
 ↪shape[0],y[y['Resultado']==3.0].shape[0]
```

[12]:
```
(1173, 488, 321)
```

[13]:
```
#Separamos los datos en test y train
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.27)

#Vemos como queda la distribucion de objetivos
y_train[y_train==1.0].shape[0],y_train[y_train['Resultado']==2.0].
 ↪shape[0],y_train[y_train['Resultado']==3.0].shape[0],y_test[y_test==1.0].
 ↪shape[0],y_test[y_test['Resultado']==2.0].
 ↪shape[0],y_test[y_test['Resultado']==3.0].shape[0]
```

[13]:
```
(1446, 349, 218, 536, 139, 103)
```

[14]:
```
#Realizamos las iteraciones como en el práctico de intro al aprendizaje
plt.figure(figsize=(14, 4), dpi= 80, facecolor='w', edgecolor='k')

for idx, loss in enumerate(('hinge', 'log', 'perceptron'), start=1):
    exploring_params = {
        'learning_rate': ['constant'],
        'eta0': [0.1, 0.01, 0.001],  # Tasa de entrenamiento
        'alpha': [0.1, 0.01, 0.001]  # Tasa de regularización
    }
    m = SGDClassifier(loss=loss, tol=1e-3)
    model = GridSearchCV(m, exploring_params, cv=7, scoring='accuracy')
    model.fit(X_train, y_train.values.ravel())

    print("# Exploración de hiperparámetros para función de coste \"%s\"" %␣
 ↪loss, end="\n\n")

    print("Mejor conjunto de parámetros:")
    print(model.best_params_, end="\n\n")

    print("Puntajes de la grilla:", end="\n\n")
    means = model.cv_results_['mean_test_score']
    stds = model.cv_results_['std_test_score']
    for mean, std, params in zip(means, stds, model.cv_results_['params']):
```

```
        print("Exactitud: %0.3f (+/-%0.03f) para los parámetros %r" % (mean,␣
↪std ** 2, params))
    print()

    print("Reporte de clasificación para el mejor clasificador (sobre conjunto␣
↪de evaluación):", end="\n\n")
    y_true, y_pred = y_test, model.predict(X_test)
    print(classification_report(y_true, y_pred), end="\n\n")

    print("================================================", end="\n\n")

    plt.subplot(1, 3, idx)
    plot_confusion_matrix(confusion_matrix(y_true, y_pred),
                          classes=['Bueno','Regular','Malo'], title="Matriz de␣
↪confusión para %s" % loss)
```

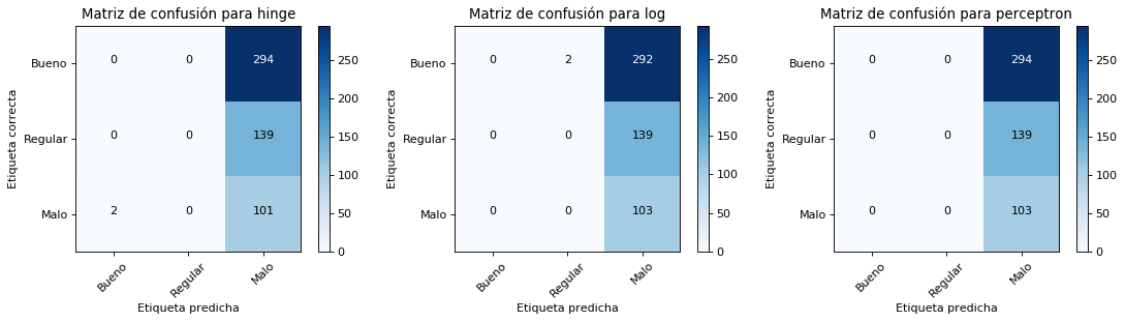
```
C:\Users\Juan\Anaconda3\envs\diplodatos\lib\site-
packages\sklearn\model_selection\_search.py:841: DeprecationWarning: The default
of the `iid` parameter will change from True to False in version 0.22 and will
be removed in 0.24. This will change numeric results when test-set sizes are
unequal.
  DeprecationWarning)
C:\Users\Juan\Anaconda3\envs\diplodatos\lib\site-
packages\sklearn\metrics\classification.py:1143: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples.
  'precision', 'predicted', average, warn_for)
C:\Users\Juan\Anaconda3\envs\diplodatos\lib\site-
packages\sklearn\metrics\classification.py:1143: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples.
  'precision', 'predicted', average, warn_for)
C:\Users\Juan\Anaconda3\envs\diplodatos\lib\site-
packages\sklearn\metrics\classification.py:1143: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples.
  'precision', 'predicted', average, warn_for)

# Exploración de hiperparámetros para función de coste "hinge"

Mejor conjunto de parámetros:
{'alpha': 0.1, 'eta0': 0.001, 'learning_rate': 'constant'}

Puntajes de la grilla:

Exactitud: 0.362 (+/-0.046) para los parámetros {'alpha': 0.1, 'eta0': 0.1,
'learning_rate': 'constant'}
```

```
Exactitud: 0.467 (+/-0.040) para los parámetros {'alpha': 0.1, 'eta0': 0.01,
'learning_rate': 'constant'}
Exactitud: 0.546 (+/-0.016) para los parámetros {'alpha': 0.1, 'eta0': 0.001,
'learning_rate': 'constant'}
Exactitud: 0.371 (+/-0.042) para los parámetros {'alpha': 0.01, 'eta0': 0.1,
'learning_rate': 'constant'}
Exactitud: 0.421 (+/-0.043) para los parámetros {'alpha': 0.01, 'eta0': 0.01,
'learning_rate': 'constant'}
Exactitud: 0.448 (+/-0.032) para los parámetros {'alpha': 0.01, 'eta0': 0.001,
'learning_rate': 'constant'}
Exactitud: 0.450 (+/-0.032) para los parámetros {'alpha': 0.001, 'eta0': 0.1,
'learning_rate': 'constant'}
Exactitud: 0.335 (+/-0.033) para los parámetros {'alpha': 0.001, 'eta0': 0.01,
'learning_rate': 'constant'}
Exactitud: 0.387 (+/-0.039) para los parámetros {'alpha': 0.001, 'eta0': 0.001,
'learning_rate': 'constant'}

Reporte de clasificación para el mejor clasificador (sobre conjunto de
evaluación):

              precision    recall  f1-score   support

         1.0       0.00      0.00      0.00       294
         2.0       0.00      0.00      0.00       139
         3.0       0.19      0.98      0.32       103

   micro avg       0.19      0.19      0.19       536
   macro avg       0.06      0.33      0.11       536
weighted avg       0.04      0.19      0.06       536


==============================================


C:\Users\Juan\Anaconda3\envs\diplodatos\lib\site-
packages\sklearn\model_selection\_search.py:841: DeprecationWarning: The default
of the `iid` parameter will change from True to False in version 0.22 and will
be removed in 0.24. This will change numeric results when test-set sizes are
unequal.
  DeprecationWarning)
C:\Users\Juan\Anaconda3\envs\diplodatos\lib\site-
packages\sklearn\metrics\classification.py:1143: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples.
  'precision', 'predicted', average, warn_for)
C:\Users\Juan\Anaconda3\envs\diplodatos\lib\site-
packages\sklearn\metrics\classification.py:1143: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
```

```
predicted samples.
  'precision', 'predicted', average, warn_for)
C:\Users\Juan\Anaconda3\envs\diplodatos\lib\site-
packages\sklearn\metrics\classification.py:1143: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples.
  'precision', 'predicted', average, warn_for)

# Exploración de hiperparámetros para función de coste "log"

Mejor conjunto de parámetros:
{'alpha': 0.1, 'eta0': 0.01, 'learning_rate': 'constant'}

Puntajes de la grilla:

Exactitud: 0.490 (+/-0.035) para los parámetros {'alpha': 0.1, 'eta0': 0.1,
'learning_rate': 'constant'}
Exactitud: 0.552 (+/-0.016) para los parámetros {'alpha': 0.1, 'eta0': 0.01,
'learning_rate': 'constant'}
Exactitud: 0.418 (+/-0.045) para los parámetros {'alpha': 0.1, 'eta0': 0.001,
'learning_rate': 'constant'}
Exactitud: 0.437 (+/-0.039) para los parámetros {'alpha': 0.01, 'eta0': 0.1,
'learning_rate': 'constant'}
Exactitud: 0.389 (+/-0.039) para los parámetros {'alpha': 0.01, 'eta0': 0.01,
'learning_rate': 'constant'}
Exactitud: 0.444 (+/-0.031) para los parámetros {'alpha': 0.01, 'eta0': 0.001,
'learning_rate': 'constant'}
Exactitud: 0.371 (+/-0.042) para los parámetros {'alpha': 0.001, 'eta0': 0.1,
'learning_rate': 'constant'}
Exactitud: 0.504 (+/-0.028) para los parámetros {'alpha': 0.001, 'eta0': 0.01,
'learning_rate': 'constant'}
Exactitud: 0.323 (+/-0.029) para los parámetros {'alpha': 0.001, 'eta0': 0.001,
'learning_rate': 'constant'}

Reporte de clasificación para el mejor clasificador (sobre conjunto de
evaluación):

             precision    recall  f1-score   support

        1.0       0.00      0.00      0.00       294
        2.0       0.00      0.00      0.00       139
        3.0       0.19      1.00      0.32       103

   micro avg       0.19      0.19      0.19       536
   macro avg       0.06      0.33      0.11       536
weighted avg       0.04      0.19      0.06       536
```

```
================================================

C:\Users\Juan\Anaconda3\envs\diplodatos\lib\site-
packages\sklearn\model_selection\_search.py:841: DeprecationWarning: The default
of the `iid` parameter will change from True to False in version 0.22 and will
be removed in 0.24. This will change numeric results when test-set sizes are
unequal.
  DeprecationWarning)
C:\Users\Juan\Anaconda3\envs\diplodatos\lib\site-
packages\sklearn\metrics\classification.py:1143: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples.
  'precision', 'predicted', average, warn_for)
C:\Users\Juan\Anaconda3\envs\diplodatos\lib\site-
packages\sklearn\metrics\classification.py:1143: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples.
  'precision', 'predicted', average, warn_for)
C:\Users\Juan\Anaconda3\envs\diplodatos\lib\site-
packages\sklearn\metrics\classification.py:1143: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples.
  'precision', 'predicted', average, warn_for)

# Exploración de hiperparámetros para función de coste "perceptron"

Mejor conjunto de parámetros:
{'alpha': 0.001, 'eta0': 0.01, 'learning_rate': 'constant'}

Puntajes de la grilla:

Exactitud: 0.498 (+/-0.026) para los parámetros {'alpha': 0.1, 'eta0': 0.1,
'learning_rate': 'constant'}
Exactitud: 0.482 (+/-0.033) para los parámetros {'alpha': 0.1, 'eta0': 0.01,
'learning_rate': 'constant'}
Exactitud: 0.487 (+/-0.034) para los parámetros {'alpha': 0.1, 'eta0': 0.001,
'learning_rate': 'constant'}
Exactitud: 0.355 (+/-0.046) para los parámetros {'alpha': 0.01, 'eta0': 0.1,
'learning_rate': 'constant'}
Exactitud: 0.421 (+/-0.044) para los parámetros {'alpha': 0.01, 'eta0': 0.01,
'learning_rate': 'constant'}
Exactitud: 0.490 (+/-0.025) para los parámetros {'alpha': 0.01, 'eta0': 0.001,
'learning_rate': 'constant'}
Exactitud: 0.494 (+/-0.026) para los parámetros {'alpha': 0.001, 'eta0': 0.1,
'learning_rate': 'constant'}
Exactitud: 0.611 (+/-0.000) para los parámetros {'alpha': 0.001, 'eta0': 0.01,
'learning_rate': 'constant'}
```

```
Exactitud: 0.424 (+/-0.044) para los parámetros {'alpha': 0.001, 'eta0': 0.001,
'learning_rate': 'constant'}

Reporte de clasificación para el mejor clasificador (sobre conjunto de
evaluación):

              precision    recall  f1-score   support

         1.0       0.00      0.00      0.00       294
         2.0       0.00      0.00      0.00       139
         3.0       0.19      1.00      0.32       103

   micro avg       0.19      0.19      0.19       536
   macro avg       0.06      0.33      0.11       536
weighted avg       0.04      0.19      0.06       536


================================================
```



A continuación realizaremos predicciones en base a lo que hemos visto de Aprendizaje Supervisado e intentaremos mejorar el pronóstico

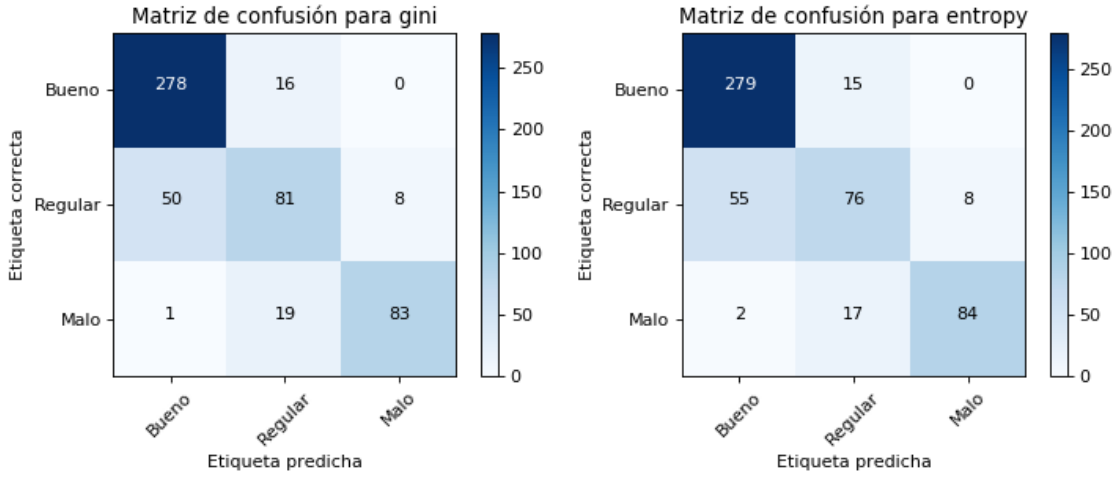
[15]:
```
#Realizamos las iteraciones como en el práctico de intro pero utilizando un␣
 ↪modelo de ensamble
from sklearn.ensemble import RandomForestClassifier as RFC


plt.figure(figsize=(14, 4), dpi= 80, facecolor='w', edgecolor='k')

for idx, criterion in enumerate(('gini', 'entropy'), start=1):
    exploring_params = {
        'n_estimators': [100,200,300], #Cantidad de arboles
        'max_depth': [10,15,20],  # Profundidad del arbol
    }
    m = RFC(criterion=criterion)
    model = GridSearchCV(m, exploring_params, cv=7, scoring='accuracy')
```

```
    model.fit(X_train, y_train.values.ravel())

    print("# Exploración de hiperparámetros para función de medida \"%s\"" %␣
↪criterion, end="\n\n")

    print("Mejor conjunto de parámetros:")
    print(model.best_params_, end="\n\n")

    print("Puntajes de la grilla:", end="\n\n")
    means = model.cv_results_['mean_test_score']
    stds = model.cv_results_['std_test_score']
    for mean, std, params in zip(means, stds, model.cv_results_['params']):
        print("Exactitud: %0.3f (+/-%0.03f) para los parámetros %r" % (mean,␣
↪std ** 2, params))
    print()

    print("Reporte de clasificación para el mejor clasificador (sobre conjunto␣
↪de evaluación):", end="\n\n")
    y_true, y_pred = y_test, model.predict(X_test)
    print(classification_report(y_true, y_pred), end="\n\n")

    print("================================================", end="\n\n")

    plt.subplot(1, 3, idx)
    plot_confusion_matrix(confusion_matrix(y_true, y_pred),
                          classes=['Bueno','Regular','Malo'], title="Matriz de␣
↪confusión para %s" % criterion)
```

```
# Exploración de hiperparámetros para función de medida "gini"

Mejor conjunto de parámetros:
{'max_depth': 15, 'n_estimators': 200}

Puntajes de la grilla:

Exactitud: 0.846 (+/-0.001) para los parámetros {'max_depth': 10,
'n_estimators': 100}
Exactitud: 0.842 (+/-0.001) para los parámetros {'max_depth': 10,
'n_estimators': 200}
Exactitud: 0.840 (+/-0.001) para los parámetros {'max_depth': 10,
'n_estimators': 300}
Exactitud: 0.844 (+/-0.001) para los parámetros {'max_depth': 15,
'n_estimators': 100}
Exactitud: 0.848 (+/-0.001) para los parámetros {'max_depth': 15,
'n_estimators': 200}
Exactitud: 0.843 (+/-0.001) para los parámetros {'max_depth': 15,
'n_estimators': 300}
```

```
Exactitud: 0.839 (+/-0.001) para los parámetros {'max_depth': 20,
'n_estimators': 100}
Exactitud: 0.840 (+/-0.001) para los parámetros {'max_depth': 20,
'n_estimators': 200}
Exactitud: 0.846 (+/-0.001) para los parámetros {'max_depth': 20,
'n_estimators': 300}

Reporte de clasificación para el mejor clasificador (sobre conjunto de
evaluación):

              precision    recall  f1-score   support

         1.0       0.84      0.95      0.89       294
         2.0       0.70      0.58      0.64       139
         3.0       0.91      0.81      0.86       103

   micro avg       0.82      0.82      0.82       536
   macro avg       0.82      0.78      0.79       536
weighted avg       0.82      0.82      0.82       536


================================================

# Exploración de hiperparámetros para función de medida "entropy"

Mejor conjunto de parámetros:
{'max_depth': 20, 'n_estimators': 200}

Puntajes de la grilla:

Exactitud: 0.840 (+/-0.001) para los parámetros {'max_depth': 10,
'n_estimators': 100}
Exactitud: 0.833 (+/-0.001) para los parámetros {'max_depth': 10,
'n_estimators': 200}
Exactitud: 0.840 (+/-0.001) para los parámetros {'max_depth': 10,
'n_estimators': 300}
Exactitud: 0.840 (+/-0.001) para los parámetros {'max_depth': 15,
'n_estimators': 100}
Exactitud: 0.842 (+/-0.001) para los parámetros {'max_depth': 15,
'n_estimators': 200}
Exactitud: 0.846 (+/-0.001) para los parámetros {'max_depth': 15,
'n_estimators': 300}
Exactitud: 0.844 (+/-0.001) para los parámetros {'max_depth': 20,
'n_estimators': 100}
Exactitud: 0.849 (+/-0.001) para los parámetros {'max_depth': 20,
'n_estimators': 200}
Exactitud: 0.844 (+/-0.001) para los parámetros {'max_depth': 20,
'n_estimators': 300}
```

```
Reporte de clasificación para el mejor clasificador (sobre conjunto de
evaluación):

              precision    recall  f1-score   support

         1.0       0.83      0.95      0.89       294
         2.0       0.70      0.55      0.62       139
         3.0       0.91      0.82      0.86       103

   micro avg       0.82      0.82      0.82       536
   macro avg       0.82      0.77      0.79       536
weighted avg       0.81      0.82      0.81       536


================================================
```



Podemos observar que mejoramos las predicciónes enormemente