Practico Mentoria - Introduccion al Aprendizaje Automatico

Autor: Melania Omonte

Practico N° 3

Importaciones

In [3]:

```
%matplotlib inline

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import scipy as sp

from collections import OrderedDict
from IPython.display import display

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, LogisticRegression, Perceptron, Ridge
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import accuracy_score, mean_squared_error, confusion_matrix
from sklearn import preprocessing

from ml.visualization import plot_confusion_matrix

import warnings
warnings.filterwarnings('ignore')
```

In [4]:

```
sns.set_style("whitegrid")
sns.set_context('talk')
```

In [5]:

```
# Seteamos una semilla para Reproducibilidad
np.random.seed(1)
```

Carga de los Datesets

In [6]:

```
#player_df = pd.read_csv('./Datesets/football_player.csv')
#team_df = pd.read_csv('./Datesets/football_team.csv')
#match_df = pd.read_csv('./Datesets/football_match.csv')

player_df = pd.read_csv('football_player.csv')
team_df = pd.read_csv('football_team.csv')
match_df = pd.read_csv('football_match.csv')


print("Shape 'player_df' = {}".format(player_df.shape))
```

```
print("Shape 'team_df'   = {}".format(team_df.shape))
print("Shape 'match_df'  = {}".format(match_df.shape))
```

```
Shape 'player_df' = (9925, 44)
Shape 'team_df'   = (288, 22)
Shape 'match_df'  = (25979, 15)
```

Regresion

Vamos a predecir el `overall_rating` (calificacion general) de un jugador

In [7]:

```
# Separamos el "target" del resto del dataset

X = player_df.loc[:, player_df.columns != 'overall_rating']
y = player_df['overall_rating']
```

Verificamos que tipo de datos contiene la columna "overall_rating"

In [8]:

```
print(player_df['overall_rating'].head(10))
```

```
0    63.60
1    66.97
2    67.00
3    69.09
4    73.24
5    77.26
6    60.57
7    79.77
8    48.00
9    67.05
Name: overall_rating, dtype: float64
```

Verificamos los tipos de datos del dataset player_df

In [9]:

```
player_df.dtypes
```

Out[9]:

```
player_name             object
birthday                object
age                      int64
height_m               float64
weight_kg              float64
imc                    float64
overall_rating         float64
potential              float64
preferred_foot          object
attacking_work_rate     object
defensive_work_rate     object
crossing               float64
finishing              float64
heading_accuracy       float64
short_passing          float64
volleys                float64
dribbling              float64
curve                  float64
free_kick_accuracy     float64
long_passing           float64
ball_control           float64
```

```
acceleration           float64
sprint_speed           float64
agility                float64
reactions              float64
balance                float64
shot_power             float64
jumping                float64
stamina                float64
strength               float64
long_shots             float64
aggression             float64
interceptions          float64
positioning            float64
vision                 float64
penalties              float64
marking                float64
standing_tackle        float64
sliding_tackle         float64
gk_diving              float64
gk_handling            float64
gk_kicking             float64
gk_positioning         float64
gk_reflexes            float64
dtype: object
```

Seleccionamos el feature `vision` del dataset player_df, para su entrenamiento. `Vision` es un campo no categórico.

In [10]:

```
# TODO: modificar esta feature por algún otro (o una combinacion de estos) para ver como cambian l
os resultados
X = X[[
    'vision',
]]
```

Realizamos la división de datos en conjuntos de entrenamiento y evaluación

La primer tarea a realizar consiste en dividir el conjunto de datos cargados en el apartado anterior en conjuntos de entrenamiento (o *training*) y evaluación (o *test*).

Utilizamos la función sklearn.model_selection.train_test_split que nos permite dividir un dataset en dos bloques:

1. 70% de los datos para entrenamiento
2. 30% para validación

 La función train_test_split divide el dataset de forma aleatoria.

In [11]:

```
# generamos el dataset de entrenamiento
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
```

Vemos el tamaño de los datos de entrenamiento y evaluacion

In [12]:

```
XT= X_train.shape
yt=y_train.shape
XTest=X_test.shape
ytest=y_test.shape

print("X_train.shape:",XT)
print("y_train.shape:",yt)
print("X_test.shape:",XTest)
print("y_test.shape:",ytest)
```

```
X_train.shape: (6947, 1)
y_train.shape: (6947,)
X_test.shape: (2978, 1)
y_test.shape: (2978,)
```

In [13]:

```
X_train.head(5)
```

Out[13]:

	vision
4339	48.64
6791	76.16
7219	48.67
8598	35.00
1081	66.29

Regresion Lineal

El objetivo de un modelo de regresión lineal es encontrar una relación entre una o más características (variables independientes) y una variable objetivo continua (variable dependiente). Como en este caso tenemos una característica, se llama Regresión lineal univariada y si tuvieramos varias características, se llamaria Regresión lineal múltiple.

In [14]:

```
#from sklearn.linear_model import SGDClassifier
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

Evaluamos el desempeño del clasificador utilizando la media del error cuadrado (MSE o Mean Squared Error) sobre el conjunto de datos de entrenamiento (`X_train` , `y_train`) y lo comparamos con el de validación (`X_test` , `y_test`).
Mientras más cercano a cero mejor

In [15]:

```
from sklearn.metrics import mean_squared_error, r2_score

## Inicializacion del Modelo
model = LinearRegression(n_jobs=-1)

# Ajustar los datos(entrenar el modelo)
model.fit(X_train, y_train)

# Predecir Entrenamiento
y_predicted = model.predict(X_train)

# Evaluacion del Modelo Entrenamiento
rmse = mean_squared_error(y_train, y_predicted)
r2 = r2_score(y_train, y_predicted)

# Ajustar los datos(entrenar el modelo)
model2 = LinearRegression(n_jobs=-1)
model2.fit(X_test, y_test)

# Predecir Validacion
y_predicted_test = model2.predict(X_test)

# Evaluacion del Modelo Validacion
rmse_test = mean_squared_error(y_test, y_predicted_test)
r2_test = r2_score(y_test, y_predicted_test)

# printing values
print('ENTRENAMIENTO:' )
print('Pendiente Entrenamiento:' ,model.coef_)
print('Intercept Entrenamiento:', model.intercept_)
print('Error Cuadrado Medio Entrenamiento: ', rmse)
print('R2 Puntuacion Entrenamiento: ', r2)
print('VALIDACION:' )
```

```
print('Pendiente Validacion:' ,model2.coef_)
print('Intercept Validacion:', model2.intercept_)
print('Error Cuadrado Medio Validacion: ', rmse_test)
print('R2 Puntuacion Validacion: ', r2_test)
```

```
ENTRENAMIENTO:
Pendiente Entrenamiento: [0.21056537]
Intercept Entrenamiento: 55.306231072323676
Error Cuadrado Medio Entrenamiento:  29.18452255010657
R2 Puntuacion Entrenamiento:  0.23116561278084835
VALIDACION:
Pendiente Validacion: [0.21132436]
Intercept Validacion: 55.04666810073566
Error Cuadrado Medio Validacion:  28.52643657329106
R2 Puntuacion Validacion:  0.23904099132359802
```

Resumiento MSE de Entrenamiento y Validacion

In [16]:

```
print('MSE para entrenamiento: %.2f' %
      mean_squared_error(y_train, model.predict(X_train)))
print('MSE para validación: %.2f' %
      mean_squared_error(y_test, model.predict(X_test)))
```

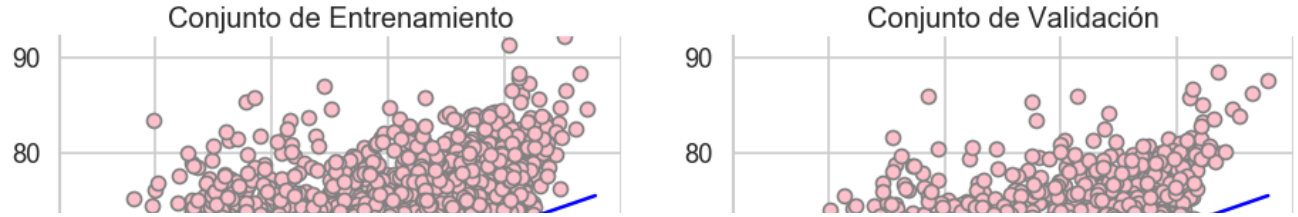
```
MSE para entrenamiento: 29.18
MSE para validación: 28.57
```

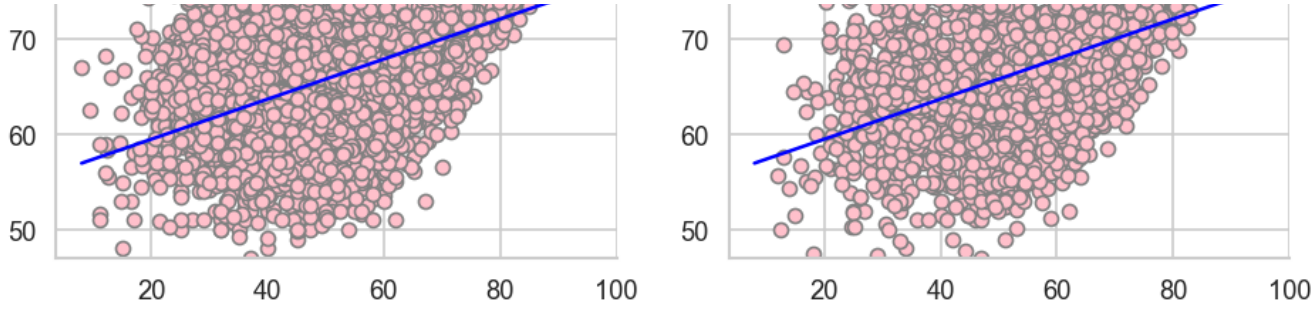
Visualizacion

Mostramos los Valores de Trazado, Puntos de datos tanto para el Entrenamiento como Validacion

In [17]:

```
plt.figure(figsize=(14, 5), dpi= 80, facecolor='w', edgecolor='k')

X_range_start = np.min(np.r_[X_train, X_test])
X_range_stop = np.max(np.r_[X_train, X_test])
y_range_start = np.min(np.r_[y_train, y_test])
y_range_stop = np.max(np.r_[y_train, y_test])
X_linspace = np.linspace(X_range_start, X_range_stop, 200).reshape(-1, 1)

# Conjunto de entrenamiento
plt.subplot(1, 2, 1)
plt.scatter(X_train, y_train, facecolor="pink", edgecolor="grey", label="datos")
plt.plot(X_linspace, model.predict(X_linspace), color="blue", label = 'Pendiente: %.2f' % model.coe
f_)
plt.ylim(y_range_start, y_range_stop)
plt.title("Conjunto de Entrenamiento")


# Conjunto de validación
plt.subplot(1, 2, 2)
plt.scatter(X_test, y_test, facecolor="pink", edgecolor="grey", label="datos")
plt.plot(X_linspace, model.predict(X_linspace), color="blue", label = 'Pendiente: %.2f' % model2.co
ef_)
plt.ylim(y_range_start, y_range_stop)

plt.title("Conjunto de Validación")

sns.despine()
plt.show()
```





Regresión Polinomial

En 'polynomial_degree' definimos el grado de polinomio, lo ideal es comenzar con un valor bajo como el 2 y después ir subiendo poco a poco para ver si se mejora lo resultados de la predicción, pero debemos tener cuidado porque en ocasiones uno puede ajustar de más el modelo ocasionando un sobreajuste u overfitting.

In [18]:

```
polynomial_degree = 2# TODO: Probar distintos grados del polinomio

poly_features = PolynomialFeatures(polynomial_degree)
poly_features.fit(X_train)
X_poly_train = poly_features.transform(X_train)
X_poly_test = poly_features.transform(X_test)

model = LinearRegression()
model.fit(X_poly_train, y_train)

print('ENTRENAMIENTO:' )
print('Valor de la pendiente:' ,model.coef_)
print('Valor de Interseccion:', model.intercept_)
y_predicted=model.predict(X_poly_train)
r2 = r2_score(y_train, y_predicted)
print('Precision del Modelo: ', r2)
print(
    "Media del error cuadrado: {:.2f}"
    .format(mean_squared_error(y_train, model.predict(X_poly_train)))
)
model2 = LinearRegression()
model2.fit(X_poly_test, y_test)
print('VALIDACION:' )
print('Valode de la pendiente:' ,model2.coef_)
print('Valode de la Interseccion:', model2.intercept_)

y_predicted_test=model2.predict(X_poly_test)
r2_test = r2_score(y_test, y_predicted_test)
print('Precision del modelo: ', r2_test)
print(
    "Media del error cuadrado: {:.2f}"
    .format(mean_squared_error(y_test, model.predict(X_poly_test)))
)
```

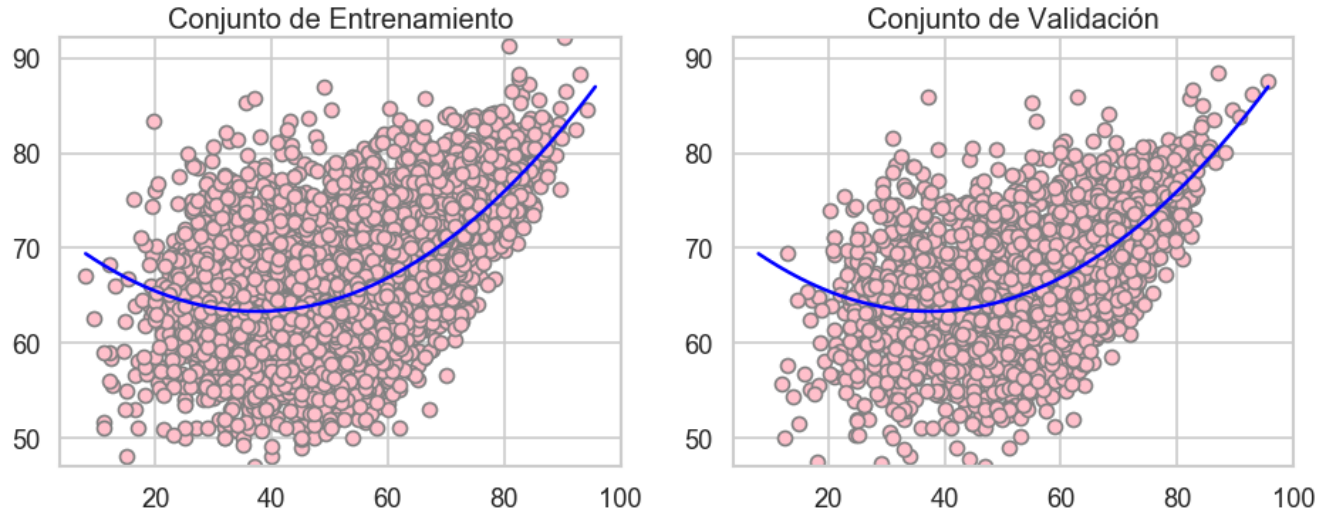
```
ENTRENAMIENTO:
Valor de la pendiente: [ 0.         -0.5242251   0.00699037]
Valor de Interseccion: 73.15233813480533
Precision del Modelo:  0.3112045866605275
Media del error cuadrado: 26.15
VALIDACION:
Valode de la pendiente: [ 0.         -0.40203595  0.00589186]
Valode de la Interseccion: 69.76752589302646
Precision del modelo:  0.29599989124261594
Media del error cuadrado: 26.52
```

Visualizacion

Warning: Tener en cuenta que si son mas de dos features no se va a poder visualizar

In [19]:

```
plt.figure(figsize=(14, 5), dpi=80, facecolor='w', edgecolor='k')

X_range_start = np.min(np.r_[X_train, X_test])
X_range_stop = np.max(np.r_[X_train, X_test])
y_range_start = np.min(np.r_[y_train, y_test])
y_range_stop = np.max(np.r_[y_train, y_test])
X_linspace = np.linspace(X_range_start, X_range_stop, 200).reshape(-1, 1)
X_linspace_poly = poly_features.transform(X_linspace)

# Conjunto de entrenamiento
plt.subplot(1, 2, 1)
plt.scatter(X_train, y_train, facecolor="pink", edgecolor="grey", label="datos")
plt.plot(X_linspace, model.predict(X_linspace_poly), color="blue", label="modelo")
plt.ylim(y_range_start, y_range_stop)
plt.title("Conjunto de Entrenamiento")

# Conjunto de validación
plt.subplot(1, 2, 2)
plt.scatter(X_test, y_test, facecolor="pink", edgecolor="grey", label="datos")
plt.plot(X_linspace, model.predict(X_linspace_poly), color="blue", label="modelo")
plt.ylim(y_range_start, y_range_stop)
plt.title("Conjunto de Validación")

plt.show()
```



Grado de Polinomio mayor a 2

In [20]:

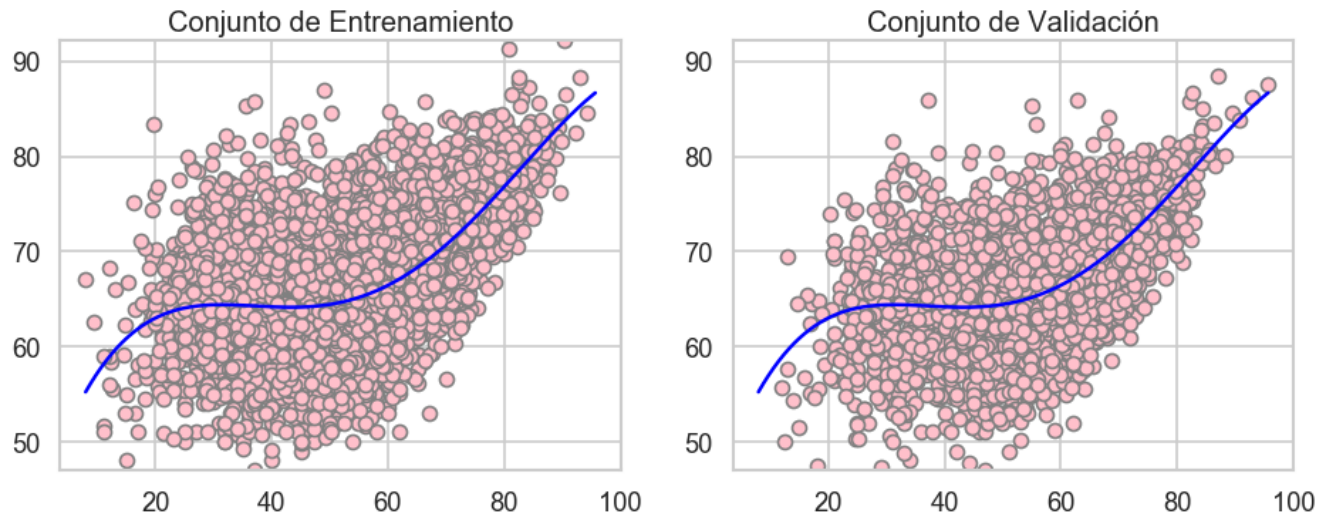
```
polynomial_degree = 5# TODO: Probar distintos grados del polinomio

poly_features = PolynomialFeatures(polynomial_degree)
poly_features.fit(X_train)
X_poly_train = poly_features.transform(X_train)
X_poly_test = poly_features.transform(X_test)

model = LinearRegression()
model.fit(X_poly_train, y_train)

print('ENTRENAMIENTO:' )
print('Valor de la Pendiente:' ,model.coef_)
print('Valor de la interseccion:', model.intercept_)
y_predicted=model.predict(X_poly_train)
r2 = r2_score(y_train, y_predicted)
print('Precision del modelo R2: ', r2)
print(
    "Media del error cuadrado: {:.2f}"
    .format(mean_squared_error(y_train, model.predict(X_poly_train)))
)
model2 = LinearRegression()
model2.fit(X_poly_test, y_test)
print('VALIDACION:' )
print('Valor de la Pendiente:' ,model2.coef_)
```

```
print('Valor de la interseccion:', model2.intercept_)

y_predicted_test=model2.predict(X_poly_test)
r2_test = r2_score(y_test, y_predicted_test)
print('Precision del modelo R2: ', r2_test)
print(
    "Media del error cuadrado: {:.2f}"
    .format(mean_squared_error(y_test, model.predict(X_poly_test)))
)
```

```
ENTRENAMIENTO:
Valor de la Pendiente: [ 0.00000000e+00  1.88499271e+00 -5.92593562e-02  7.25806777e-04
 -2.34455123e-06 -4.24230081e-09]
Valor de la interseccion: 43.584681122265536
Precision del modelo R2:  0.3237218081077009
Media del error cuadrado: 25.67
VALIDACION:
Valor de la Pendiente: [ 0.00000000e+00  2.61846223e+00 -8.98677146e-02  1.41083948e-03
 -9.94689600e-06  2.80067937e-08]
Valor de la interseccion: 35.26349533197008
Precision del modelo R2:  0.30928076354362577
Media del error cuadrado: 26.05
```

In [21]:

```
plt.figure(figsize=(14, 5), dpi=80, facecolor='w', edgecolor='k')

X_range_start = np.min(np.r_[X_train, X_test])
X_range_stop = np.max(np.r_[X_train, X_test])
y_range_start = np.min(np.r_[y_train, y_test])
y_range_stop = np.max(np.r_[y_train, y_test])
X_linspace = np.linspace(X_range_start, X_range_stop, 200).reshape(-1, 1)
X_linspace_poly = poly_features.transform(X_linspace)

# Conjunto de entrenamiento
plt.subplot(1, 2, 1)
plt.scatter(X_train, y_train, facecolor="pink", edgecolor="grey", label="datos")
plt.plot(X_linspace, model.predict(X_linspace_poly), color="blue", label="modelo")
plt.ylim(y_range_start, y_range_stop)
plt.title("Conjunto de Entrenamiento")

# Conjunto de validación
plt.subplot(1, 2, 2)
plt.scatter(X_test, y_test, facecolor="pink", edgecolor="grey", label="datos")
plt.plot(X_linspace, model.predict(X_linspace_poly), color="blue", label="modelo")
plt.ylim(y_range_start, y_range_stop)
plt.title("Conjunto de Validación")

plt.show()
```



In [22]:

```
Grado de Polinomio mayor a 10
```

```
  File "<ipython-input-22-ecbd6115f7e3>", line 1
    Grado de Polinomio mayor a 10
           ^
SyntaxError: invalid syntax
```

In [23]:

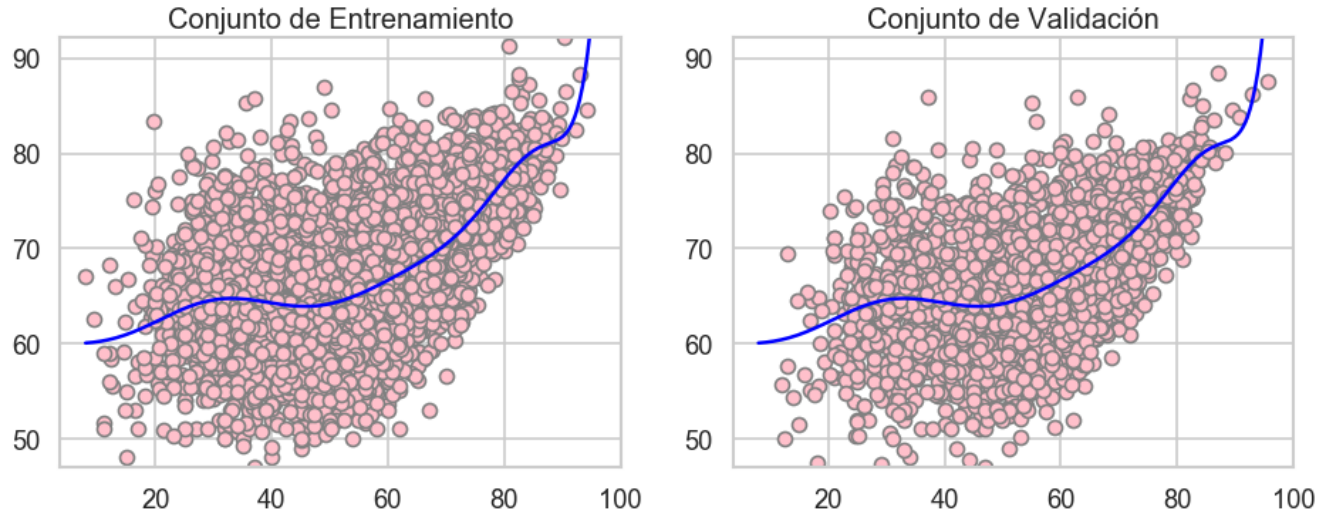
```
polynomial_degree = 11# TODO: Probar distintos grados del polinomio

poly_features = PolynomialFeatures(polynomial_degree)
poly_features.fit(X_train)
X_poly_train = poly_features.transform(X_train)
X_poly_test = poly_features.transform(X_test)

model = LinearRegression()
model.fit(X_poly_train, y_train)

print('ENTRENAMIENTO:' )
print('Valor de la Pendiente:' ,model.coef_)
print('Valor de la interseccion:', model.intercept_)
y_predicted=model.predict(X_poly_train)
r2 = r2_score(y_train, y_predicted)
print('Precision del modelo R2: ', r2)
print(
    "Media del error cuadrado: {:.2f}"
    .format(mean_squared_error(y_train, model.predict(X_poly_train)))
)
model2 = LinearRegression()
model2.fit(X_poly_test, y_test)
print('VALIDACION:' )
print('Valor de la Pendiente:' ,model2.coef_)
print('Valor de la interseccion:', model2.intercept_)

y_predicted_test=model2.predict(X_poly_test)
r2_test = r2_score(y_test, y_predicted_test)
print('Precision del modelo R2: ', r2_test)
print(
    "Media del error cuadrado: {:.2f}"
    .format(mean_squared_error(y_test, model.predict(X_poly_test)))
)
```

```
ENTRENAMIENTO:
Valor de la Pendiente: [ 0.00000000e+00  2.05411389e-07  2.04611777e-09  3.72132944e-08
  6.04200302e-07  5.47635310e-06 -4.77758964e-07  1.72842883e-08
 -3.31556057e-10  3.55308654e-12 -2.01552002e-14  4.72620409e-17]
Valor de la interseccion: 59.971263497020715
Precision del modelo R2:  0.3249039789001864
Media del error cuadrado: 25.63
VALIDACION:
Valor de la Pendiente: [ 0.00000000e+00 -8.49142321e-09  2.04811727e-09  3.38250652e-08
  5.58607767e-07  5.14016291e-06 -4.39528754e-07  1.56374468e-08
 -2.95459936e-10  3.12061612e-12 -1.74474301e-14  4.03128267e-17]
Valor de la interseccion: 58.803885733564314
Precision del modelo R2:  0.30968476410871604
Media del error cuadrado: 26.05
```

In [24]:

```
plt.figure(figsize=(14, 5), dpi=80, facecolor='w', edgecolor='k')

X_range_start = np.min(np.r_[X_train, X_test])
X_range_stop = np.max(np.r_[X_train, X_test])
y_range_start = np.min(np.r_[y_train, y_test])
y_range_stop = np.max(np.r_[y_train, y_test])
X_linspace = np.linspace(X_range_start, X_range_stop, 200).reshape(-1, 1)
X_linspace_poly = poly_features.transform(X_linspace)

# Conjunto de entrenamiento
plt.subplot(1, 2, 1)
plt.scatter(X_train, y_train, facecolor="pink", edgecolor="grey", label="datos")
plt.plot(X_linspace, model.predict(X_linspace_poly), color="blue", label="modelo")
plt.ylim(y_range_start, y_range_stop)
plt.title("Conjunto de Entrenamiento")
```

```
plt.title('Conjunto de Entrenamiento')

# Conjunto de validación
plt.subplot(1, 2, 2)
plt.scatter(X_test, y_test, facecolor="pink", edgecolor="grey", label="datos")
plt.plot(X_linspace, model.predict(X_linspace_poly), color="blue", label="modelo")
plt.ylim(y_range_start, y_range_stop)
plt.title("Conjunto de Validación")

plt.show()
```



Grado de Polinomio mayor a 20

In [25]:

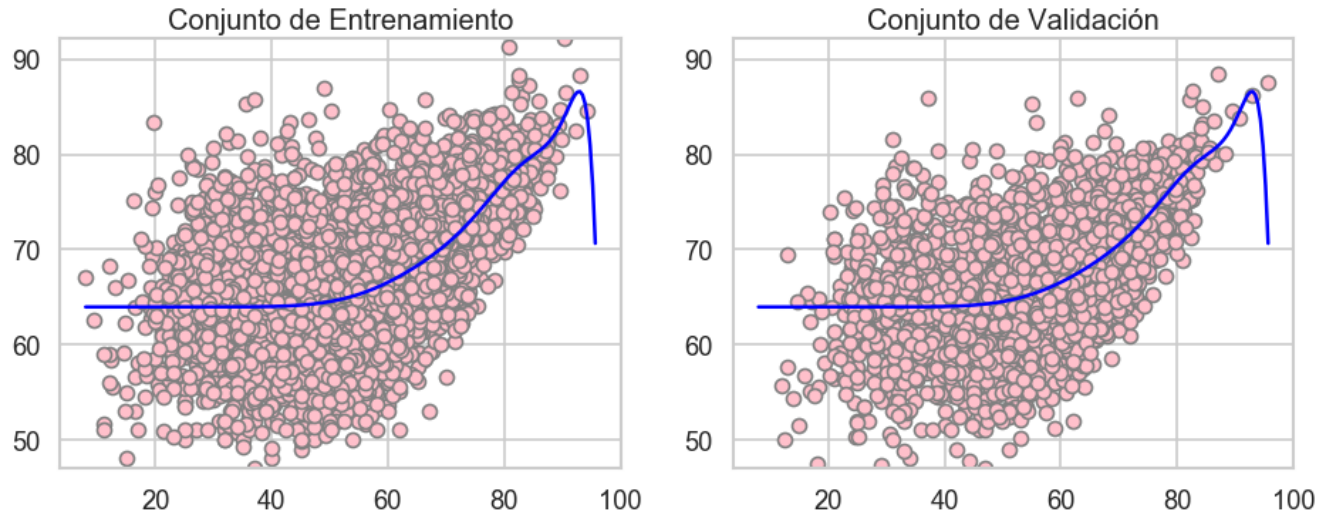
```
polynomial_degree = 21# TODO: Probar distintos grados del polinomio

poly_features = PolynomialFeatures(polynomial_degree)
poly_features.fit(X_train)
X_poly_train = poly_features.transform(X_train)
X_poly_test = poly_features.transform(X_test)

model = LinearRegression()
model.fit(X_poly_train, y_train)

print('ENTRENAMIENTO:' )
print('Valor de la Pendiente:' ,model.coef_)
print('Valor de la interseccion:', model.intercept_)
y_predicted=model.predict(X_poly_train)
r2 = r2_score(y_train, y_predicted)
print('Precision del modelo R2: ', r2)
print(
    "Media del error cuadrado: {:.2f}"
    .format(mean_squared_error(y_train, model.predict(X_poly_train)))
)
model2 = LinearRegression()
model2.fit(X_poly_test, y_test)
print('VALIDACION:' )
print('Valor de la Pendiente:' ,model2.coef_)
print('Valor de la interseccion:', model2.intercept_)

y_predicted_test=model2.predict(X_poly_test)
r2_test = r2_score(y_test, y_predicted_test)
print('Precision del modelo R2: ', r2_test)
print(
    "Media del error cuadrado: {:.2f}"
    .format(mean_squared_error(y_test, model.predict(X_poly_test)))
)
```

```
ENTRENAMIENTO:
Valor de la Pendiente: [ 0.00000000e+00  1.24921261e-28  8.40384328e-32 -1.34769950e-34
 -1.70307487e-37  3.57162953e-40  8.65001677e-42  5.74533904e-40
  2.80501381e-38  1.31157382e-36  5.82263490e-35  2.42342329e-33
  9.27591823e-32  3.16730351e-30  9.15925509e-29  2.02545073e-27
```

```
  2.61999473e-26 -1.53428245e-27  3.60356042e-29 -4.24337182e-31
  2.50329783e-33 -5.91440114e-36]
Valor de la interseccion: 63.95464391582996
Precision del modelo R2:  0.32140706868287194
Media del error cuadrado: 25.76
VALIDACION:
Valor de la Pendiente: [ 0.00000000e+00 -1.44721953e-28 -8.48481995e-32  1.43969692e-34
 -8.72725433e-38 -9.96435312e-41  9.04906315e-42  6.65102974e-40
  3.27707103e-38  1.54612815e-36  6.92809415e-35  2.91134358e-33
  1.12540927e-31  3.88190104e-30  1.13428318e-28  2.53506959e-27
  3.31493890e-26 -1.93706702e-27  4.53011039e-29 -5.30201364e-31
  3.10411144e-33 -7.26911198e-36]
Valor de la interseccion: 63.6049950417957
Precision del modelo R2:  0.30429760431617126
Media del error cuadrado: 26.25
```

In [26]:

```
plt.figure(figsize=(14, 5), dpi=80, facecolor='w', edgecolor='k')

X_range_start = np.min(np.r_[X_train, X_test])
X_range_stop = np.max(np.r_[X_train, X_test])
y_range_start = np.min(np.r_[y_train, y_test])
y_range_stop = np.max(np.r_[y_train, y_test])
X_linspace = np.linspace(X_range_start, X_range_stop, 200).reshape(-1, 1)
X_linspace_poly = poly_features.transform(X_linspace)

# Conjunto de entrenamiento
plt.subplot(1, 2, 1)
plt.scatter(X_train, y_train, facecolor="pink", edgecolor="grey", label="datos")
plt.plot(X_linspace, model.predict(X_linspace_poly), color="blue", label="modelo")
plt.ylim(y_range_start, y_range_stop)
plt.title("Conjunto de Entrenamiento")

# Conjunto de validación
plt.subplot(1, 2, 2)
plt.scatter(X_test, y_test, facecolor="pink", edgecolor="grey", label="datos")
plt.plot(X_linspace, model.predict(X_linspace_poly), color="blue", label="modelo")
plt.ylim(y_range_start, y_range_stop)
plt.title("Conjunto de Validación")

plt.show()
```



Conclusion

Vemos que a medida que agregamos grados al polinomio, se produce overfiting o sobreajuste. El grado de Polinomio optimo para este polinomio, consideramos que esta entre 2 y 5.

Clasificacion Multiclase

Vamos a predecir el `attacking_work_rate` de un jugador

In [27]:

```
# Separamos el "target" del resto del dataset

X = player_df.loc[:, player_df.columns != 'attacking_work_rate']
y = player_df['attacking_work_rate']
```

Validamos el tipo de valor

In [28]:

```
player_df['attacking_work_rate'].dtypes
```

Out[28]:

```
dtype('O')
```

Muestro los distintos valores incluidos en 'attacking_work_rate' junto con el número de apariciones de cada uno.

In [29]:

```
y.value_counts()
```

Out[29]:

```
medium    7350
high      2112
low        463
Name: attacking_work_rate, dtype: int64
```

Codificar la variable categorica `attacking_work_rate` como una variable numerica

La función LabelEncoder nos permite codificar etiquetas de una característica categórica en valores numéricos entre 0 y el número de clases menos 1.

In [30]:

```
le = preprocessing.LabelEncoder()
```

Una vez instanciado LabelEncoder, el método fit lo entrena (creando el mapeado entre las etiquetas y los números) y el método transform transforma las etiquetas que se incluyan como argumento en los números correspondientes. El método fit_transform realiza ambas acciones simultáneamente.

In [31]:

```
y[:] = le.fit_transform(y)
y.value_counts()
```

Out[31]:

```
2    7350
0    2112
1     463
Name: attacking_work_rate, dtype: int64
```

Resultado: Podemos validar que que la variable categorica `attacking_work_rate`, se ha convertido en variable numerica.

In [32]:

```
le.classes_
```

Out[32]:

```
array(['high', 'low', 'medium'], dtype=object)
```

In [33]:

```
player_df['attacking_work_rate'].dtypes
```

Out[33]:

```
dtype('int32')
```

Resultado: validamos que se cambio el tipo de dato

Seleccionamos un feature de los listados en la descripción que no sea categórico, por ejemplo `vision`

In [34]:

```
# TODO: modificar esta feature por algún otro (o una combinacion de estos) para ver como cambian l
os resultados
X = X[[
    'penalties',
]]
```

División de datos en conjuntos de entrenamiento y evaluación

La primer tarea consiste en dividir el conjunto de datos cargados en el apartado anterior en conjuntos de entrenamiento (o *training*) y evaluación (o *test*).

Utilizar aproximadamente 70% de los datos para entrenamiento y 30% para validación.

In [35]:

```
# TODO
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
```

Regresion Logistica

In [36]:

```
from sklearn.metrics import accuracy_score, mean_squared_error, confusion_matrix,
balanced_accuracy_score
penalty ='l1' # TODO: Tipo de regularización: l1 (valor absoluto), l2 (cuadrados).
alpha = 2# TODO: Parámetro de regularización. También denominado como parámetro `lambda`. Debe ser
mayor que 0.

model = LogisticRegression(penalty=penalty, C=1./alpha, multi_class='ovr')
model.fit(X_train, y_train)

print("Accuracy para entrenamiento: {:.2f}".format(balanced_accuracy_score(y_train, model.predict(
X_train))))
print("Accuracy para validación   : {:.2f}".format(balanced_accuracy_score(y_test, model.predict(X_
test))))
```

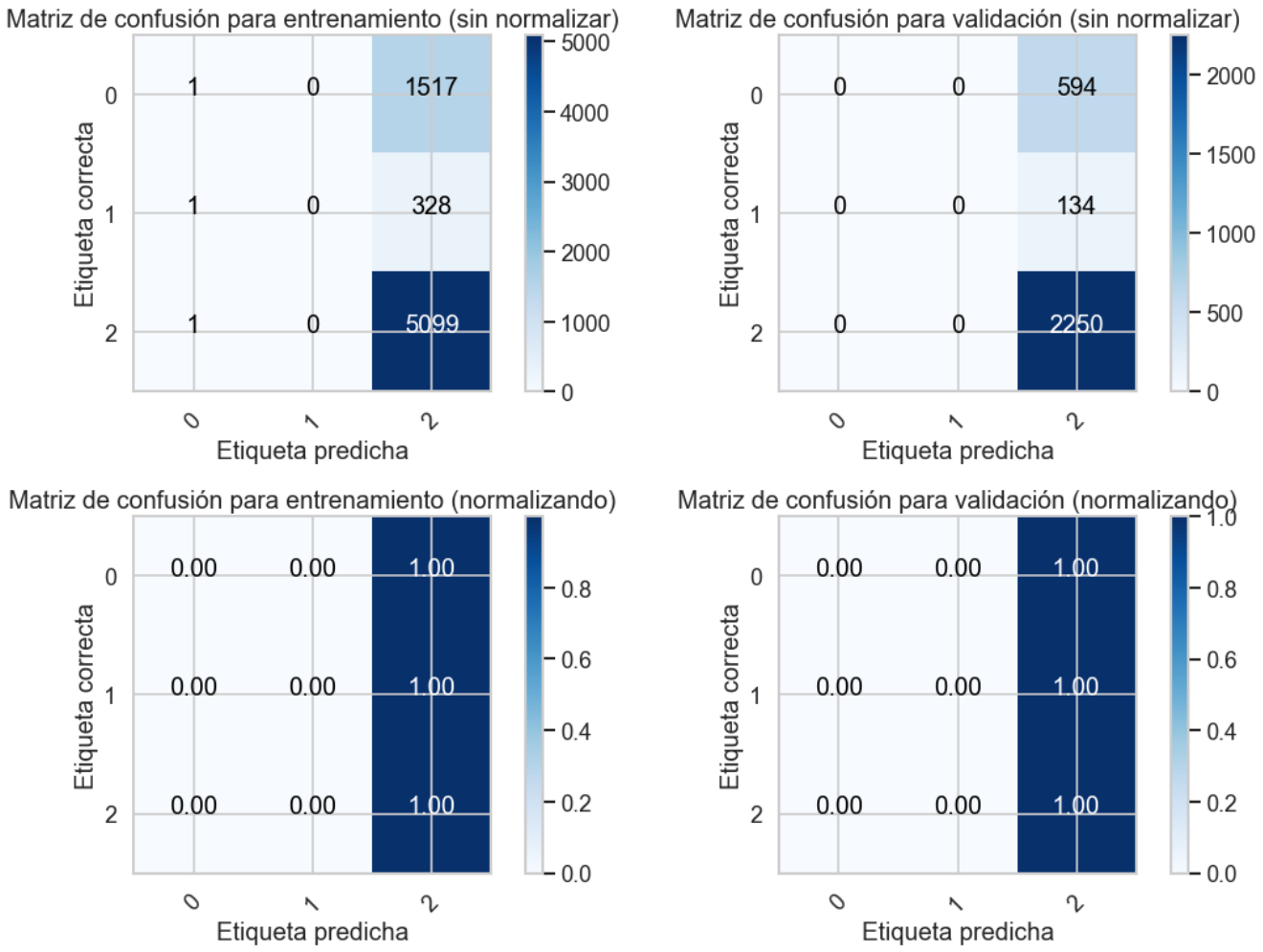
```
Accuracy para entrenamiento: 0.33
Accuracy para validación   : 0.33
```

Matriz de Confusion

In [37]:

```
plt.figure(figsize=(14, 10), dpi= 80, facecolor='w', edgecolor='k')
```

```
plt.subplot(2, 2, 1)
plot_confusion_matrix(
    confusion_matrix(y_train, model.predict(X_train)),
    classes=np.unique(y),
    title='Matriz de confusión para entrenamiento (sin normalizar)'
)

plt.subplot(2, 2, 3)
plot_confusion_matrix(
    confusion_matrix(y_train, model.predict(X_train)),
    classes=np.unique(y),
    normalize=True,
    title='Matriz de confusión para entrenamiento (normalizando)'
)

plt.subplot(2, 2, 2)
plot_confusion_matrix(
    confusion_matrix(y_test, model.predict(X_test)),
    classes=np.unique(y),
    title='Matriz de confusión para validación (sin normalizar)'
)

plt.subplot(2, 2, 4)
plot_confusion_matrix(
    confusion_matrix(y_test, model.predict(X_test)),
    classes=np.unique(y),
    normalize=True,
    title='Matriz de confusión para validación (normalizando)'
)

plt.show()
```



Visualización de la frontera de decisión

Seleccion de Hiperparametros

Utilizando búsqueda exhaustiva (*grid search*) con *n-fold cross-validation* y utilizando como métrica la **Accuracy**, realizamos una selección de los mejores hiperparámetros para su conjunto de datos.

In [38]:

```
from sklearn.linear_model import SGDClassifier
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

In [43]:

```
np.random.seed(123)

plt.figure(figsize=(10, 4), dpi= 80, facecolor='w', edgecolor='k')

model = SGDClassifier()
model.fit(X_train, y_train)

# Clasificación para Conjunto Entrenamiento.
print(" MODELO SGDClassifier ")
print("= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = ")
print("Reporte de clasificación para el mejor clasificador (sobre Conjunto de Entrenamiento):", en
d="\n\n")
print(classification_report(y_train, model.predict(X_train)), end="\n\n")


# Clasificación para Conjunto Evaluación.
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_test, model.predict(X_test)), end="\n\n")
# para ambos casos utilizamos como hiperparametros los valores predeterminados del SGD class.

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

```
 MODELO SGDClassifier
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
Reporte de clasificación para el mejor clasificador (sobre Conjunto de Entrenamiento):

              precision    recall  f1-score   support

           0       0.00      0.00      0.00      1518
           1       0.04      0.74      0.08       329
           2       0.85      0.24      0.38      5100

   micro avg       0.21      0.21      0.21      6947
   macro avg       0.30      0.33      0.15      6947
weighted avg       0.63      0.21      0.28      6947


Reporte de clasificación para el mejor clasificador (sobre Conjunto de Evaluación):

              precision    recall  f1-score   support

           0       0.00      0.00      0.00       594
           1       0.04      0.73      0.08       134
           2       0.88      0.26      0.40      2250

   micro avg       0.23      0.23      0.23      2978
   macro avg       0.31      0.33      0.16      2978
weighted avg       0.66      0.23      0.31      2978


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



<Figure size 800x320 with 0 Axes>
```

La Regresión Logística es un modelo para clasificación, no para regresión. Para este ejemplo usamos las funciones: hinge, log, perceptron y el siguiente conjunto de parametros:C,multi_class, tol, max_iter Usamos GridSearchCV para evaluar y seleccionar los parámetros del modelo elegido (LogisticRegression)

parámetros del modelo elegido (LogisticRegression).

In [42]:

```
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn import metrics
np.random.seed(123)


plt.figure(figsize=(10, 4), dpi= 80, facecolor='w', edgecolor='k')

for idx, loss in enumerate(('hinge', 'log', 'perceptron'), start=1):
    parametros = {
        'C': (1e+1, 1, 1e-1),
        'multi_class': ('ovr', 'multinomial'),
        'tol': (1e-3, 1e-4, 1e-5),
        'max_iter': (10, 100, 1000)}

    clasif = LogisticRegression(n_jobs=1, random_state=1, penalty='l1')
    model = GridSearchCV(clasif, parametros, cv=5, scoring='accuracy', iid=False, error_score=0.0,
n_jobs=-1)
    model.fit(X_train, y_train)

    print("# Función \"%s\"" % loss, end="\n\n")

    print("Mejor conjunto de parámetros:")
    print(model.best_params_, end="\n\n")
    # Model Accuracy, how often is the classifier correct?



    print("Puntajes de la GridSearchCV:", 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()
## Reporte clasificación para Conjunto Entrenamiento
    print("Clasificación sobre Conjunto de Entrenamiento:", end="\n\n")
    print(classification_report(y_train, model.predict(X_train)), end="\n\n")

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


## Reporte clasificación para Conjunto Evaluación
    print("Clasificación sobre Conjunto de Evaluación:", end="\n\n")
    y_true, y_pred = y_test, model.predict(X_test)
    print(classification_report(y_test, model.predict(X_test)), end="\n\n")


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

```
# Función "hinge"

Mejor conjunto de parámetros:
{'C': 10.0, 'max_iter': 100, 'multi_class': 'ovr', 'tol': 0.001}

Puntajes de la GridSearchCV:

Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 10.0, 'max_iter': 10, 'multi_class': 'ovr',
'tol': 0.001}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 10.0, 'max_iter': 10, 'multi_class': 'ovr',
'tol': 0.0001}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 10.0, 'max_iter': 10, 'multi_class': 'ovr',
'tol': 1e-05}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 10.0, 'max_iter': 10, 'multi_class':
'multinomial', 'tol': 0.001}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 10.0, 'max_iter': 10, 'multi_class':
'multinomial', 'tol': 0.0001}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 10.0, 'max_iter': 10, 'multi_class':
'multinomial', 'tol': 1e-05}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 10.0, 'max_iter': 100, 'multi_class': 'ovr',
'tol': 0.001}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 10.0, 'max_iter': 100, 'multi_class': 'ovr',
'tol': 0.0001}
```

'tol': 0.0001}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 10.0, 'max_iter': 100, 'multi_class': 'ovr',
'tol': 1e-05}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 10.0, 'max_iter': 100, 'multi_class':
'multinomial', 'tol': 0.001}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 10.0, 'max_iter': 100, 'multi_class':
'multinomial', 'tol': 0.0001}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 10.0, 'max_iter': 100, 'multi_class':
'multinomial', 'tol': 1e-05}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 10.0, 'max_iter': 1000, 'multi_class':
'ovr', 'tol': 0.001}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 10.0, 'max_iter': 1000, 'multi_class':
'ovr', 'tol': 0.0001}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 10.0, 'max_iter': 1000, 'multi_class':
'ovr', 'tol': 1e-05}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 10.0, 'max_iter': 1000, 'multi_class':
'multinomial', 'tol': 0.001}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 10.0, 'max_iter': 1000, 'multi_class':
'multinomial', 'tol': 0.0001}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 10.0, 'max_iter': 1000, 'multi_class':
'multinomial', 'tol': 1e-05}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 1, 'max_iter': 10, 'multi_class': 'ovr', 'to
l': 0.001}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 1, 'max_iter': 10, 'multi_class': 'ovr', 'to
l': 0.0001}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 1, 'max_iter': 10, 'multi_class': 'ovr', 'to
l': 1e-05}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 1, 'max_iter': 10, 'multi_class':
'multinomial', 'tol': 0.001}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 1, 'max_iter': 10, 'multi_class':
'multinomial', 'tol': 0.0001}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 1, 'max_iter': 10, 'multi_class':
'multinomial', 'tol': 1e-05}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 1, 'max_iter': 100, 'multi_class': 'ovr', 't
ol': 0.001}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 1, 'max_iter': 100, 'multi_class': 'ovr', 't
ol': 0.0001}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 1, 'max_iter': 100, 'multi_class': 'ovr', 't
ol': 1e-05}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 1, 'max_iter': 100, 'multi_class':
'multinomial', 'tol': 0.001}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 1, 'max_iter': 100, 'multi_class':
'multinomial', 'tol': 0.0001}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 1, 'max_iter': 100, 'multi_class':
'multinomial', 'tol': 1e-05}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 1, 'max_iter': 1000, 'multi_class': 'ovr', '
tol': 0.001}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 1, 'max_iter': 1000, 'multi_class': 'ovr', '
tol': 0.0001}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 1, 'max_iter': 1000, 'multi_class': 'ovr', '
tol': 1e-05}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 1, 'max_iter': 1000, 'multi_class':
'multinomial', 'tol': 0.001}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 1, 'max_iter': 1000, 'multi_class':
'multinomial', 'tol': 0.0001}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 1, 'max_iter': 1000, 'multi_class':
'multinomial', 'tol': 1e-05}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 0.1, 'max_iter': 10, 'multi_class': 'ovr', '
tol': 0.001}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 0.1, 'max_iter': 10, 'multi_class': 'ovr', '
tol': 0.0001}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 0.1, 'max_iter': 10, 'multi_class': 'ovr', '
tol': 1e-05}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 0.1, 'max_iter': 10, 'multi_class':
'multinomial', 'tol': 0.001}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 0.1, 'max_iter': 10, 'multi_class':
'multinomial', 'tol': 0.0001}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 0.1, 'max_iter': 10, 'multi_class':
'multinomial', 'tol': 1e-05}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 0.1, 'max_iter': 100, 'multi_class': 'ovr',
'tol': 0.001}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 0.1, 'max_iter': 100, 'multi_class': 'ovr',
'tol': 0.0001}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 0.1, 'max_iter': 100, 'multi_class': 'ovr',
'tol': 1e-05}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 0.1, 'max_iter': 100, 'multi_class':
'multinomial', 'tol': 0.001}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 0.1, 'max_iter': 100, 'multi_class':

```
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 0.1, 'max_iter': 100, 'multi_class':
'multinomial', 'tol': 0.0001}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 0.1, 'max_iter': 100, 'multi_class':
'multinomial', 'tol': 1e-05}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 0.1, 'max_iter': 1000, 'multi_class': 'ovr',
'tol': 0.001}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 0.1, 'max_iter': 1000, 'multi_class': 'ovr',
'tol': 0.0001}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 0.1, 'max_iter': 1000, 'multi_class': 'ovr',
'tol': 1e-05}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 0.1, 'max_iter': 1000, 'multi_class':
'multinomial', 'tol': 0.001}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 0.1, 'max_iter': 1000, 'multi_class':
'multinomial', 'tol': 0.0001}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 0.1, 'max_iter': 1000, 'multi_class':
'multinomial', 'tol': 1e-05}

Clasificación sobre Conjunto de Entrenamiento:

              precision    recall  f1-score   support

           0       0.00      0.00      0.00      1518
           1       0.00      0.00      0.00       329
           2       0.73      1.00      0.85      5100

   micro avg       0.73      0.73      0.73      6947
   macro avg       0.24      0.33      0.28      6947
weighted avg       0.54      0.73      0.62      6947


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

Clasificación sobre Conjunto de Evaluación:

              precision    recall  f1-score   support

           0       0.00      0.00      0.00       594
           1       0.00      0.00      0.00       134
           2       0.76      1.00      0.86      2250

   micro avg       0.76      0.76      0.76      2978
   macro avg       0.25      0.33      0.29      2978
weighted avg       0.57      0.76      0.65      2978


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

# Función "log"

Mejor conjunto de parámetros:
{'C': 10.0, 'max_iter': 100, 'multi_class': 'ovr', 'tol': 0.001}

Puntajes de la GridSearchCV:

Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 10.0, 'max_iter': 10, 'multi_class': 'ovr',
'tol': 0.001}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 10.0, 'max_iter': 10, 'multi_class': 'ovr',
'tol': 0.0001}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 10.0, 'max_iter': 10, 'multi_class': 'ovr',
'tol': 1e-05}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 10.0, 'max_iter': 10, 'multi_class':
'multinomial', 'tol': 0.001}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 10.0, 'max_iter': 10, 'multi_class':
'multinomial', 'tol': 0.0001}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 10.0, 'max_iter': 10, 'multi_class':
'multinomial', 'tol': 1e-05}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 10.0, 'max_iter': 100, 'multi_class': 'ovr',
'tol': 0.001}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 10.0, 'max_iter': 100, 'multi_class': 'ovr',
'tol': 0.0001}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 10.0, 'max_iter': 100, 'multi_class': 'ovr',
'tol': 1e-05}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 10.0, 'max_iter': 100, 'multi_class':
'multinomial', 'tol': 0.001}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 10.0, 'max_iter': 100, 'multi_class':
'multinomial', 'tol': 0.0001}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 10.0, 'max_iter': 100, 'multi_class':
'multinomial', 'tol': 1e-05}
```

'multinomial', 'tol': 1e-05}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 10.0, 'max_iter': 1000, 'multi_class': 'ovr', 'tol': 0.001}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 10.0, 'max_iter': 1000, 'multi_class': 'ovr', 'tol': 0.0001}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 10.0, 'max_iter': 1000, 'multi_class': 'ovr', 'tol': 1e-05}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 10.0, 'max_iter': 1000, 'multi_class': 'multinomial', 'tol': 0.001}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 10.0, 'max_iter': 1000, 'multi_class': 'multinomial', 'tol': 0.0001}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 10.0, 'max_iter': 1000, 'multi_class': 'multinomial', 'tol': 1e-05}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 1, 'max_iter': 10, 'multi_class': 'ovr', 'tol': 0.001}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 1, 'max_iter': 10, 'multi_class': 'ovr', 'tol': 0.0001}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 1, 'max_iter': 10, 'multi_class': 'ovr', 'tol': 1e-05}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 1, 'max_iter': 10, 'multi_class': 'multinomial', 'tol': 0.001}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 1, 'max_iter': 10, 'multi_class': 'multinomial', 'tol': 0.0001}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 1, 'max_iter': 10, 'multi_class': 'multinomial', 'tol': 1e-05}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 1, 'max_iter': 100, 'multi_class': 'ovr', 'tol': 0.001}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 1, 'max_iter': 100, 'multi_class': 'ovr', 'tol': 0.0001}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 1, 'max_iter': 100, 'multi_class': 'ovr', 'tol': 1e-05}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 1, 'max_iter': 100, 'multi_class': 'multinomial', 'tol': 0.001}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 1, 'max_iter': 100, 'multi_class': 'multinomial', 'tol': 0.0001}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 1, 'max_iter': 100, 'multi_class': 'multinomial', 'tol': 1e-05}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 1, 'max_iter': 1000, 'multi_class': 'ovr', 'tol': 0.001}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 1, 'max_iter': 1000, 'multi_class': 'ovr', 'tol': 0.0001}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 1, 'max_iter': 1000, 'multi_class': 'ovr', 'tol': 1e-05}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 1, 'max_iter': 1000, 'multi_class': 'multinomial', 'tol': 0.001}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 1, 'max_iter': 1000, 'multi_class': 'multinomial', 'tol': 0.0001}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 1, 'max_iter': 1000, 'multi_class': 'multinomial', 'tol': 1e-05}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 0.1, 'max_iter': 10, 'multi_class': 'ovr', 'tol': 0.001}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 0.1, 'max_iter': 10, 'multi_class': 'ovr', 'tol': 0.0001}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 0.1, 'max_iter': 10, 'multi_class': 'ovr', 'tol': 1e-05}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 0.1, 'max_iter': 10, 'multi_class': 'multinomial', 'tol': 0.001}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 0.1, 'max_iter': 10, 'multi_class': 'multinomial', 'tol': 0.0001}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 0.1, 'max_iter': 10, 'multi_class': 'multinomial', 'tol': 1e-05}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 0.1, 'max_iter': 100, 'multi_class': 'ovr', 'tol': 0.001}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 0.1, 'max_iter': 100, 'multi_class': 'ovr', 'tol': 0.0001}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 0.1, 'max_iter': 100, 'multi_class': 'ovr', 'tol': 1e-05}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 0.1, 'max_iter': 100, 'multi_class': 'multinomial', 'tol': 0.001}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 0.1, 'max_iter': 100, 'multi_class': 'multinomial', 'tol': 0.0001}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 0.1, 'max_iter': 100, 'multi_class': 'multinomial', 'tol': 1e-05}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 0.1, 'max_iter': 1000, 'multi_class': 'ovr', 'tol': 0.001}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 0.1, 'max_iter': 1000, 'multi_class': 'ovr', 'tol': 0.0001}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 0.1, 'max_iter': 1000, 'multi_class': 'ovr',

```
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 0.1, 'max_iter': 1000, 'multi_class': 'ovr',
'tol': 1e-05}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 0.1, 'max_iter': 1000, 'multi_class':
'multinomial', 'tol': 0.001}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 0.1, 'max_iter': 1000, 'multi_class':
'multinomial', 'tol': 0.0001}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 0.1, 'max_iter': 1000, 'multi_class':
'multinomial', 'tol': 1e-05}

Clasificación sobre Conjunto de Entrenamiento:

              precision    recall  f1-score   support

           0       0.00      0.00      0.00      1518
           1       0.00      0.00      0.00       329
           2       0.73      1.00      0.85      5100

   micro avg       0.73      0.73      0.73      6947
   macro avg       0.24      0.33      0.28      6947
weighted avg       0.54      0.73      0.62      6947


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

Clasificación sobre Conjunto de Evaluación:

              precision    recall  f1-score   support

           0       0.00      0.00      0.00       594
           1       0.00      0.00      0.00       134
           2       0.76      1.00      0.86      2250

   micro avg       0.76      0.76      0.76      2978
   macro avg       0.25      0.33      0.29      2978
weighted avg       0.57      0.76      0.65      2978


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

# Función "perceptron"

Mejor conjunto de parámetros:
{'C': 10.0, 'max_iter': 100, 'multi_class': 'ovr', 'tol': 0.001}

Puntajes de la GridSearchCV:

Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 10.0, 'max_iter': 10, 'multi_class': 'ovr',
'tol': 0.001}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 10.0, 'max_iter': 10, 'multi_class': 'ovr',
'tol': 0.0001}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 10.0, 'max_iter': 10, 'multi_class': 'ovr',
'tol': 1e-05}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 10.0, 'max_iter': 10, 'multi_class':
'multinomial', 'tol': 0.001}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 10.0, 'max_iter': 10, 'multi_class':
'multinomial', 'tol': 0.0001}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 10.0, 'max_iter': 10, 'multi_class':
'multinomial', 'tol': 1e-05}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 10.0, 'max_iter': 100, 'multi_class': 'ovr',
'tol': 0.001}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 10.0, 'max_iter': 100, 'multi_class': 'ovr',
'tol': 0.0001}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 10.0, 'max_iter': 100, 'multi_class': 'ovr',
'tol': 1e-05}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 10.0, 'max_iter': 100, 'multi_class':
'multinomial', 'tol': 0.001}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 10.0, 'max_iter': 100, 'multi_class':
'multinomial', 'tol': 0.0001}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 10.0, 'max_iter': 100, 'multi_class':
'multinomial', 'tol': 1e-05}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 10.0, 'max_iter': 1000, 'multi_class':
'ovr', 'tol': 0.001}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 10.0, 'max_iter': 1000, 'multi_class':
'ovr', 'tol': 0.0001}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 10.0, 'max_iter': 1000, 'multi_class':
'ovr', 'tol': 1e-05}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 10.0, 'max_iter': 1000, 'multi_class':
'multinomial', 'tol': 0.001}
```

'multinomial', 'tol': 0.001}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 10.0, 'max_iter': 1000, 'multi_class':
'multinomial', 'tol': 0.0001}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 10.0, 'max_iter': 1000, 'multi_class':
'multinomial', 'tol': 1e-05}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 1, 'max_iter': 10, 'multi_class': 'ovr', 'to
l': 0.001}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 1, 'max_iter': 10, 'multi_class': 'ovr', 'to
l': 0.0001}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 1, 'max_iter': 10, 'multi_class': 'ovr', 'to
l': 1e-05}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 1, 'max_iter': 10, 'multi_class':
'multinomial', 'tol': 0.001}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 1, 'max_iter': 10, 'multi_class':
'multinomial', 'tol': 0.0001}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 1, 'max_iter': 10, 'multi_class':
'multinomial', 'tol': 1e-05}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 1, 'max_iter': 100, 'multi_class': 'ovr', 't
ol': 0.001}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 1, 'max_iter': 100, 'multi_class': 'ovr', 't
ol': 0.0001}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 1, 'max_iter': 100, 'multi_class': 'ovr', 't
ol': 1e-05}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 1, 'max_iter': 100, 'multi_class':
'multinomial', 'tol': 0.001}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 1, 'max_iter': 100, 'multi_class':
'multinomial', 'tol': 0.0001}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 1, 'max_iter': 100, 'multi_class':
'multinomial', 'tol': 1e-05}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 1, 'max_iter': 1000, 'multi_class': 'ovr', '
tol': 0.001}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 1, 'max_iter': 1000, 'multi_class': 'ovr', '
tol': 0.0001}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 1, 'max_iter': 1000, 'multi_class': 'ovr', '
tol': 1e-05}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 1, 'max_iter': 1000, 'multi_class':
'multinomial', 'tol': 0.001}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 1, 'max_iter': 1000, 'multi_class':
'multinomial', 'tol': 0.0001}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 1, 'max_iter': 1000, 'multi_class':
'multinomial', 'tol': 1e-05}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 0.1, 'max_iter': 10, 'multi_class': 'ovr', '
tol': 0.001}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 0.1, 'max_iter': 10, 'multi_class': 'ovr', '
tol': 0.0001}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 0.1, 'max_iter': 10, 'multi_class': 'ovr', '
tol': 1e-05}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 0.1, 'max_iter': 10, 'multi_class':
'multinomial', 'tol': 0.001}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 0.1, 'max_iter': 10, 'multi_class':
'multinomial', 'tol': 0.0001}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 0.1, 'max_iter': 10, 'multi_class':
'multinomial', 'tol': 1e-05}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 0.1, 'max_iter': 100, 'multi_class': 'ovr',
'tol': 0.001}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 0.1, 'max_iter': 100, 'multi_class': 'ovr',
'tol': 0.0001}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 0.1, 'max_iter': 100, 'multi_class': 'ovr',
'tol': 1e-05}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 0.1, 'max_iter': 100, 'multi_class':
'multinomial', 'tol': 0.001}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 0.1, 'max_iter': 100, 'multi_class':
'multinomial', 'tol': 0.0001}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 0.1, 'max_iter': 100, 'multi_class':
'multinomial', 'tol': 1e-05}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 0.1, 'max_iter': 1000, 'multi_class': 'ovr',
'tol': 0.001}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 0.1, 'max_iter': 1000, 'multi_class': 'ovr',
'tol': 0.0001}
Exactitud: 0.734 (+/-0.000) para los parámetros {'C': 0.1, 'max_iter': 1000, 'multi_class': 'ovr',
'tol': 1e-05}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 0.1, 'max_iter': 1000, 'multi_class':
'multinomial', 'tol': 0.001}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 0.1, 'max_iter': 1000, 'multi_class':
'multinomial', 'tol': 0.0001}
Exactitud: 0.000 (+/-0.000) para los parámetros {'C': 0.1, 'max_iter': 1000, 'multi_class':
'multinomial', 'tol': 1e-05}

```
Clasificación sobre Conjunto de Entrenamiento:

              precision    recall  f1-score   support

           0       0.00      0.00      0.00      1518
           1       0.00      0.00      0.00       329
           2       0.73      1.00      0.85      5100

   micro avg       0.73      0.73      0.73      6947
   macro avg       0.24      0.33      0.28      6947
weighted avg       0.54      0.73      0.62      6947


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

Clasificación sobre Conjunto de Evaluación:

              precision    recall  f1-score   support

           0       0.00      0.00      0.00       594
           1       0.00      0.00      0.00       134
           2       0.76      1.00      0.86      2250

   micro avg       0.76      0.76      0.76      2978
   macro avg       0.25      0.33      0.29      2978
weighted avg       0.57      0.76      0.65      2978


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


<Figure size 800x320 with 0 Axes>
```