ajuste-de-redes-neuronales

September 11, 2024

Paola Félix Torres A00227869

Ejercicio 1

[ ]:
```
#9 - Variable dependiente MR, variables independientes M, W, H, P y S

#Evalúa con validación cruzada un modelo perceptrón multicapa para las
#variables que se te asignaron para este ejercicio.
```

[ ]:
```
import numpy as np
import pandas as pd
```

[ ]:
```
df = pd.read_csv('/content/sample_data/crime_data.csv')
```

[ ]:
```
X = df[['M', 'W', 'H', 'P', 'S']]
y = df['MR']
df = df[:-1]
```

[ ]:
```
from sklearn.model_selection import KFold
from sklearn.neural_network import MLPRegressor
from sklearn.metrics import mean_squared_error, r2_score

mlp = MLPRegressor(hidden_layer_sizes=(10, 10), max_iter=10000)

kf = KFold(n_splits=5, shuffle=True, random_state=42)

cv_r2_scores = []
cv_mse_scores = []

# Realiza la validación cruzada
for train_index, test_index in kf.split(X):
    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]

    mlp.fit(X_train, y_train)

    y_pred = mlp.predict(X_test)
```

```
    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)

    cv_mse_scores.append(mse)
    cv_r2_scores.append(r2)

# Imprime los resultados
print(f'Average MSE: {np.mean(cv_mse_scores)}')
print(f'Average R^2: {np.mean(cv_r2_scores)}')
```

```
Average MSE: 51.024840336305644
Average R^2: 0.5215673224992711
```

[ ]:
```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import KFold
from sklearn.metrics import mean_squared_error, r2_score

X = df[['M', 'W', 'H', 'P', 'S']]
y = df['MR']

linear_reg = LinearRegression()

kf = KFold(n_splits=5, shuffle=True, random_state=42)

cv_r2_scores = []
cv_mse_scores = []

# Realiza la validación cruzada
for train_index, test_index in kf.split(X):
    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]

    linear_reg.fit(X_train, y_train)

    y_pred = linear_reg.predict(X_test)

    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)

    cv_mse_scores.append(mse)
    cv_r2_scores.append(r2)

# Imprime los resultados promedio
print(f'Average MSE: {np.mean(cv_mse_scores)}')
print(f'Average R^2: {np.mean(cv_r2_scores)}')
```

```
Average MSE: 5.310342632282411
Average R^2: 0.4951098356249871
```

Evalúa con validación cruzada un modelo perceptrón multicapa para las variables que se te asignaron para este ejercicio. Viendo los resultados de regresión, desarrolla una conclusión sobre los siguientes puntos: ¿Consideras que el modelo perceptrón multicapa es efectivo para modelar los datos del problema? ¿Por qué? ¿Qué modelo es mejor para los datos de criminalidad, el lineal o el perceptrón multicapa? ¿Por qué?

Considero que el modelo perceptrón multicapa no es muy efectivo ya que cuenta con una R^2 de 0.52, sin embargo, este modelo se podría mejorar ajustando los hiperparámetros.

Se podría decir que ambos modelos son muy parecidos, pues en cuanto a su R^2, mientras que el MLP tiene una R^2 de 0.52, el modelo lineal tiene una R^2 de 0.49. Pero por otro lado, si comparamos el error cuadrático medio de ambos modelos, el modelo lineal tiene un MSE de 5.31, el cual es significativamente menor al del MLP, el cual es de 78.1. Por lo anteriormente dicho se puede concluir que el modelo lineal es mejor para los datos, ya que ofrece predicciones más precisas con un modelo mucho más fácil y simple.

Ejercicio 2

[1]:
```
#6 y 7 - M_4.txt
```

[2]:
```
import numpy as np
import pandas as pd
```

[3]:
```
data = np.loadtxt('sample_data/M_4.txt')
df = pd.DataFrame(data)
```

[4]:
```
X = df.iloc[:, 2:].values
y = df.iloc[:, 0].values
```

[5]:
```
#Evalúa un modelo perceptrón multicapa con validación cruzada utilizando al␣
 ↪menos 5 capas de 20 neuronas.

import numpy as np
from sklearn import datasets
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import classification_report

clf = MLPClassifier(hidden_layer_sizes=(20, 20, 20, 20, 20), max_iter=10000)
clf.fit(X, y)

# Validación cruzada
n_splits = 5
kf = StratifiedKFold(n_splits=n_splits, shuffle=True, random_state=42)

cv_y_test = []
cv_y_pred = []

for train_index, test_index in kf.split(X, y):
```

```
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]

    clf_i = MLPClassifier(hidden_layer_sizes=(20, 20, 20, 20, 20),␣
 ↪max_iter=10000)
    clf_i.fit(X_train, y_train)

    y_pred = clf_i.predict(X_test)

    cv_y_test.append(y_test)
    cv_y_pred.append(y_pred)

print(classification_report(np.concatenate(cv_y_test), np.
 ↪concatenate(cv_y_pred)))
```

```
              precision    recall  f1-score   support

         1.0       0.93      0.96      0.95        90
         2.0       0.82      0.83      0.82        90
         3.0       0.88      0.89      0.88        90
         4.0       0.90      0.89      0.89        90
         5.0       0.91      0.87      0.89        90
         6.0       0.93      0.90      0.92        90
         7.0       0.87      0.90      0.89        90

    accuracy                           0.89       630
   macro avg       0.89      0.89      0.89       630
weighted avg       0.89      0.89      0.89       630
```

[9]:
```
#Evalúa un modelo perceptrón multicapa con validación cruzada, pero encontrando␣
 ↪el número óptimo de capas y neuronas de la red.
#Prepara el modelo perceptrón multicapa:
#Opten los hiperparámetros óptimos de capas y neuronas de la red.
#Con los hiperparámetros óptimos, ajusta el modelo con todos los datos.

from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import GridSearchCV, cross_val_predict
from sklearn.metrics import classification_report
import numpy as np

# Configurar el modelo perceptrón multicapa con dos capas de 100 neuronas cada␣
 ↪una
clf = MLPClassifier(hidden_layer_sizes=(100, 100), max_iter=10000)
clf.fit(X, y)

# Evaluar el modelo utilizando validación cruzada
```

```
y_pred = cross_val_predict(MLPClassifier(hidden_layer_sizes=(100, 100),␣
 ↪max_iter=10000), X, y)
print("Evaluación del modelo (dos capas de 100 neuronas):")
print(classification_report(y, y_pred))

# Definir los rangos de búsqueda para el número de capas y neuronas
num_layers = np.arange(1, 20, 5)
num_neurons = np.arange(10, 110, 20)

# Crear combinaciones de capas y neuronas
layers = []
for l in num_layers:
    for n in num_neurons:
        layers.append(l * [n])

# Usar GridSearchCV para encontrar el mejor número de capas y neuronas
clf = GridSearchCV(MLPClassifier(max_iter=10000), {'hidden_layer_sizes':␣
 ↪layers}, cv=5)
clf.fit(X, y)

# Mostrar el mejor estimador (modelo óptimo con el número de capas y neuronas)
print("Mejor configuración de capas y neuronas:")
print(clf.best_estimator_)

# Evaluar el modelo utilizando validación cruzada con los mejores␣
 ↪hiperparámetros
y_pred = cross_val_predict(clf, X, y, cv=5)
print("Evaluación del modelo (hiperparámetros óptimos):")
print(classification_report(y, y_pred))
```

```
Evaluación del modelo (dos capas de 100 neuronas):
              precision    recall  f1-score   support

         1.0       0.93      0.96      0.95        90
         2.0       0.78      0.88      0.83        90
         3.0       0.97      0.82      0.89        90
         4.0       0.92      0.92      0.92        90
         5.0       0.92      0.92      0.92        90
         6.0       0.95      0.88      0.91        90
         7.0       0.88      0.96      0.91        90

    accuracy                           0.90       630
   macro avg       0.91      0.90      0.91       630
weighted avg       0.91      0.90      0.91       630

Mejor configuración de capas y neuronas:
MLPClassifier(hidden_layer_sizes=[90], max_iter=10000)
```

```
Evaluación del modelo (hiperparámetros óptimos):
              precision    recall  f1-score   support

         1.0       0.96      0.91      0.94        90
         2.0       0.79      0.83      0.81        90
         3.0       0.85      0.84      0.85        90
         4.0       0.88      0.89      0.88        90
         5.0       0.94      0.93      0.94        90
         6.0       0.91      0.94      0.93        90
         7.0       0.89      0.87      0.88        90

    accuracy                           0.89       630
   macro avg       0.89      0.89      0.89       630
weighted avg       0.89      0.89      0.89       630
```

¿Observas alguna mejora importante al optimizar el tamaño de la red? ¿Es el resultado que esperabas? Argumenta tu respuesta. ¿Qué inconvenientes hay al encontrar el tamaño óptimo de la red? ¿Por qué?

Realmente no observé ninguna mejora importante al optimizar el tamaño de red, esto no era lo que esperaba pues aunque la precisión media ya era bastante buena antes de la optimización, con una precisión de 0.89, almenos esperaba una mejoría de unas cuantas decimas, pero esto no fue así, lo que puede significar que el modelo original ya estaba bastante bien ajustado.

Entre los inconvenientes que se encuentran al buscar el tamaño optimo de red están el hecho que es muy costoso computacionalmente, pues tanto el tiempo de entrenamiento y el uso de recursos pueden ser un poco limitantes si queremos explorar muchas combinaciones de capas y neuronas, asimismo existe el riesgo de sobreajuste.

Ejercicio 3

[ ]:
```
#8 y 9 - P1_5.txt
```

[10]:
```
import numpy as np
import pandas as pd
```

[11]:
```
data = np.loadtxt('sample_data/P1_5.txt')
df = pd.DataFrame(data)
```

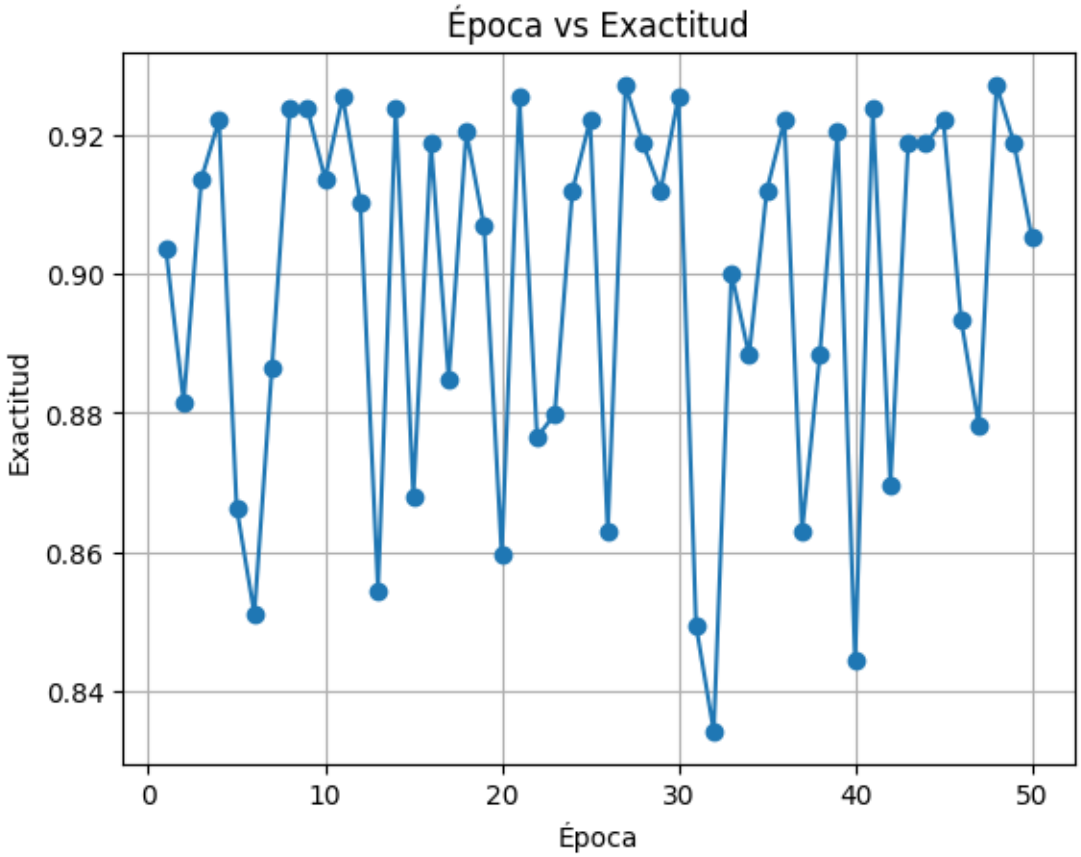
[14]:
```
x = df.iloc[:, 2:].values
y = df.iloc[:, 0].values
```

[15]:
```
# Implementa un modelo perceptrón de una neurona entrenado con descenso de␣
 ↪gradiente estocástico, y evalúalo con validación cruzada.
#Para esta caso, es necesario que encuentres la gráfica de Época Vs Exactitud.

import numpy as np
import matplotlib.pyplot as plt
```

```
from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,␣
 ↪random_state=42)

perceptron = SGDClassifier(loss='perceptron', max_iter=1,␣
 ↪learning_rate='constant', eta0=0.01, random_state=42, warm_start=True)

accuracies = []
epochs = 50

for epoch in range(epochs):
    perceptron.fit(x_train, y_train)
    y_pred = perceptron.predict(x_test)
    acc = accuracy_score(y_test, y_pred)
    accuracies.append(acc)
    perceptron.max_iter += 1

plt.plot(range(1, epochs + 1), accuracies, marker='o')
plt.title('Época vs Exactitud')
plt.xlabel('Época')
plt.ylabel('Exactitud')
plt.grid(True)
plt.show()

skf = StratifiedKFold(n_splits=5)
cross_val_scores = cross_val_score(perceptron, x, y, cv=skf)
print(f"Exactitud media en validación cruzada: {np.mean(cross_val_scores):.4f}")
```

```
/usr/local/lib/python3.10/dist-
packages/sklearn/linear_model/_stochastic_gradient.py:713: ConvergenceWarning:
Maximum number of iteration reached before convergence. Consider increasing
max_iter to improve the fit.
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/linear_model/_stochastic_gradient.py:713: ConvergenceWarning:
Maximum number of iteration reached before convergence. Consider increasing
max_iter to improve the fit.
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/linear_model/_stochastic_gradient.py:713: ConvergenceWarning:
Maximum number of iteration reached before convergence. Consider increasing
max_iter to improve the fit.
```

```
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/linear_model/_stochastic_gradient.py:713: ConvergenceWarning:
Maximum number of iteration reached before convergence. Consider increasing
max_iter to improve the fit.
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/linear_model/_stochastic_gradient.py:713: ConvergenceWarning:
Maximum number of iteration reached before convergence. Consider increasing
max_iter to improve the fit.
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/linear_model/_stochastic_gradient.py:713: ConvergenceWarning:
Maximum number of iteration reached before convergence. Consider increasing
max_iter to improve the fit.
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/linear_model/_stochastic_gradient.py:713: ConvergenceWarning:
Maximum number of iteration reached before convergence. Consider increasing
max_iter to improve the fit.
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/linear_model/_stochastic_gradient.py:713: ConvergenceWarning:
Maximum number of iteration reached before convergence. Consider increasing
max_iter to improve the fit.
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/linear_model/_stochastic_gradient.py:713: ConvergenceWarning:
Maximum number of iteration reached before convergence. Consider increasing
max_iter to improve the fit.
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/linear_model/_stochastic_gradient.py:713: ConvergenceWarning:
Maximum number of iteration reached before convergence. Consider increasing
max_iter to improve the fit.
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/linear_model/_stochastic_gradient.py:713: ConvergenceWarning:
Maximum number of iteration reached before convergence. Consider increasing
max_iter to improve the fit.
  warnings.warn(
```




```
Exactitud media en validación cruzada: 0.8690
```

[16]:
```
#Repite el paso anterior, pero utilizando descenso de gradiente de lote y de␣
 ↪mini-lote para entrenar el modelo.

import numpy as np
import matplotlib.pyplot as plt
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,␣
 ↪random_state=42)

epochs = 50

batch_accuracies = []
minibatch_accuracies = []
```

```
#␣
 ↪-----------------------------------------------------------------------------------------
# Descenso de Gradiente de Lote
#␣
 ↪-----------------------------------------------------------------------------------------

perceptron_batch = MLPClassifier(hidden_layer_sizes=(1,), max_iter=1,␣
 ↪solver='sgd', batch_size=len(x_train), learning_rate_init=0.01,␣
 ↪warm_start=True)

for epoch in range(epochs):
    perceptron_batch.fit(x_train, y_train)
    y_pred_batch = perceptron_batch.predict(x_test)
    batch_acc = accuracy_score(y_test, y_pred_batch)
    batch_accuracies.append(batch_acc)
    perceptron_batch.max_iter += 1

plt.plot(range(1, epochs + 1), batch_accuracies, marker='o', label='Lote␣
 ↪Completo')

#␣
 ↪-----------------------------------------------------------------------------------------
# Descenso de Gradiente de Mini-Lote
#␣
 ↪-----------------------------------------------------------------------------------------
perceptron_minibatch = MLPClassifier(hidden_layer_sizes=(1,), max_iter=1,␣
 ↪solver='sgd', batch_size=32, learning_rate_init=0.01, warm_start=True)

for epoch in range(epochs):
    perceptron_minibatch.fit(x_train, y_train)
    y_pred_minibatch = perceptron_minibatch.predict(x_test)
    minibatch_acc = accuracy_score(y_test, y_pred_minibatch)
    minibatch_accuracies.append(minibatch_acc)
    perceptron_minibatch.max_iter += 1

plt.plot(range(1, epochs + 1), minibatch_accuracies, marker='o',␣
 ↪label='Mini-Lote (Batch Size = 32)')

plt.title('Época vs Exactitud')
plt.xlabel('Época')
plt.ylabel('Exactitud')
plt.legend()
plt.grid(True)
plt.show()

batch_cv_scores = cross_val_score(perceptron_batch, x, y, cv=5)
```

```python
print(f"Exactitud media en validación cruzada (lote completo): {np.
  ↪mean(batch_cv_scores):.4f}")

minibatch_cv_scores = cross_val_score(perceptron_minibatch, x, y, cv=5)
print(f"Exactitud media en validación cruzada (mini-lote): {np.
  ↪mean(minibatch_cv_scores):.4f}")
```

```
/usr/local/lib/python3.10/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1) reached and the
optimization hasn't converged yet.
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (2) reached and the
optimization hasn't converged yet.
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (3) reached and the
optimization hasn't converged yet.
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (4) reached and the
optimization hasn't converged yet.
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (5) reached and the
optimization hasn't converged yet.
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (6) reached and the
optimization hasn't converged yet.
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (7) reached and the
optimization hasn't converged yet.
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (8) reached and the
optimization hasn't converged yet.
  warnings.warn(
```

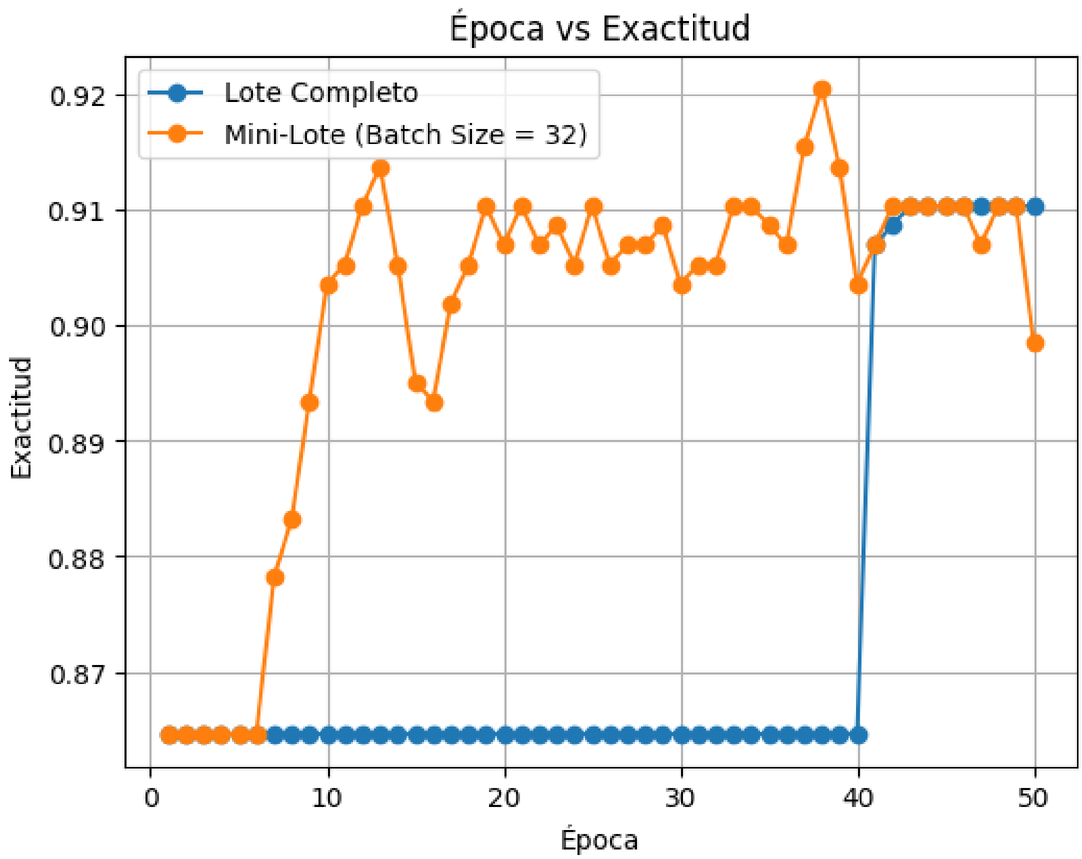
```
/usr/local/lib/python3.10/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (9) reached and the
optimization hasn't converged yet.
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (10) reached and
the optimization hasn't converged yet.
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (11) reached and
the optimization hasn't converged yet.
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (12) reached and
the optimization hasn't converged yet.
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (13) reached and
the optimization hasn't converged yet.
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (14) reached and
the optimization hasn't converged yet.
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (15) reached and
the optimization hasn't converged yet.
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (16) reached and
the optimization hasn't converged yet.
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (17) reached and
the optimization hasn't converged yet.
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (18) reached and
```

```
the optimization hasn't converged yet.
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (19) reached and
the optimization hasn't converged yet.
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (20) reached and
the optimization hasn't converged yet.
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (21) reached and
the optimization hasn't converged yet.
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (22) reached and
the optimization hasn't converged yet.
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (23) reached and
the optimization hasn't converged yet.
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (24) reached and
the optimization hasn't converged yet.
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (25) reached and
the optimization hasn't converged yet.
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (26) reached and
the optimization hasn't converged yet.
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (27) reached and
the optimization hasn't converged yet.
  warnings.warn(
/usr/local/lib/python3.10/dist-
```

```
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (28) reached and
the optimization hasn't converged yet.
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (29) reached and
the optimization hasn't converged yet.
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (30) reached and
the optimization hasn't converged yet.
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (31) reached and
the optimization hasn't converged yet.
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (32) reached and
the optimization hasn't converged yet.
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (33) reached and
the optimization hasn't converged yet.
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (34) reached and
the optimization hasn't converged yet.
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (35) reached and
the optimization hasn't converged yet.
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (36) reached and
the optimization hasn't converged yet.
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (37) reached and
the optimization hasn't converged yet.
```

```
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (38) reached and
the optimization hasn't converged yet.
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (39) reached and
the optimization hasn't converged yet.
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (40) reached and
the optimization hasn't converged yet.
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (41) reached and
the optimization hasn't converged yet.
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (42) reached and
the optimization hasn't converged yet.
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1) reached and the
optimization hasn't converged yet.
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (2) reached and the
optimization hasn't converged yet.
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (3) reached and the
optimization hasn't converged yet.
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (4) reached and the
optimization hasn't converged yet.
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
```

```
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (5) reached and the
optimization hasn't converged yet.
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (6) reached and the
optimization hasn't converged yet.
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (7) reached and the
optimization hasn't converged yet.
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (8) reached and the
optimization hasn't converged yet.
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (10) reached and
the optimization hasn't converged yet.
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (11) reached and
the optimization hasn't converged yet.
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (12) reached and
the optimization hasn't converged yet.
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (13) reached and
the optimization hasn't converged yet.
  warnings.warn(
```




```
/usr/local/lib/python3.10/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (51) reached and
the optimization hasn't converged yet.
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (51) reached and
the optimization hasn't converged yet.
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (51) reached and
the optimization hasn't converged yet.
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (51) reached and
the optimization hasn't converged yet.
  warnings.warn(
```

```
/usr/local/lib/python3.10/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (51) reached and
the optimization hasn't converged yet.
  warnings.warn(

Exactitud media en validación cruzada (lote completo): 0.8665

/usr/local/lib/python3.10/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (51) reached and
the optimization hasn't converged yet.
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (51) reached and
the optimization hasn't converged yet.
  warnings.warn(

Exactitud media en validación cruzada (mini-lote): 0.9107

/usr/local/lib/python3.10/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (51) reached and
the optimization hasn't converged yet.
  warnings.warn(
```

[17]:
```python
#Evalúa un modelo perceptrón multicapa con validación cruzada. Para este caso,␣
 ↪puedes utilizar un modelo dado por scikit-learn, Keras o Pytorch.

import numpy as np
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import cross_val_score
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,␣
 ↪random_state=42)

mlp = MLPClassifier(hidden_layer_sizes=(100, 100), max_iter=500, solver='adam',␣
 ↪random_state=42)

cv_scores = cross_val_score(mlp, x, y, cv=5, scoring='accuracy')

print(f"Exactitud media en validación cruzada: {np.mean(cv_scores):.4f}")
print(f"Desviación estándar de la exactitud: {np.std(cv_scores):.4f}")

mlp.fit(x_train, y_train)
y_pred = mlp.predict(x_test)
```

```
print("Informe de clasificación en el conjunto de prueba:")
print(classification_report(y_test, y_pred))
```

```
Exactitud media en validación cruzada: 0.9234
Desviación estándar de la exactitud: 0.0132
Informe de clasificación en el conjunto de prueba:
              precision    recall  f1-score   support

         1.0       0.77      0.66      0.71        80
         2.0       0.95      0.97      0.96       511

    accuracy                           0.93       591
   macro avg       0.86      0.82      0.83       591
weighted avg       0.92      0.93      0.92       591
```

¿El modelo de una neurona es suficiente para modelar el conjunto de datos de este problema?

Sí, en este caso el modelo de una neurona es suficiente y eficiente. Dado que solo hay dos clases y el problema no es muy complicado, el perceptrón de una sola neurona, tiene exactitud media de 0.8690. Aunque el perceptrón multicapa tiene una exactitud mayor de 0.9234, la diferencia no es tan grande.