Fernando Cerriteño Magaña - A01702790

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

from sklearn.neural_network import MLPRegressor
from sklearn.model_selection import KFold, GridSearchCV,
cross_val_predict
from sklearn.metrics import mean_squared_error, mean_absolute_error

from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import StratifiedKFold, train_test_split
from sklearn.metrics import classification_report
from sklearn.neighbors import KNeighborsClassifier
```

Parte 1

El conjunto de datos de criminalidad Download datos de criminalidadde Estados Unidos publicado en el año 1993 consiste de 51 registros para los que se tienen las siguientes variables:

- VR = Crímenes violentos por cada 100000 habitantes
- MR = Asesinatos por cada 100000 habitantes
- M = Porcentaje de áreas metropolitanas
- W = Porcentaje de gente blanca
- H = Porcentaje de personas con preparatoria terminada
- P = Porcentaje con ingresos por debajo del nivel de pobreza
- S = Porcentaje de familias con solo un miembro adulto como tutor

```
df = pd.read_csv('/content/crime_data.csv')

df = df.drop(['State'],axis=1)
df = df.drop(['MR'],axis=1)
df = df.drop(['S'],axis=1)

y = df.VR
x = df.iloc[:,1:]

df.isnull().sum()
```

```
VR    0
M     0
W     0
H     0
P     0
dtype: int64
```

```
x = x.to_numpy()
y = y.to_numpy()
```

Validación cruzada

```
n_folds = 5
kf = KFold(n_splits=n_folds, shuffle = True)

mse_cv = []
mae_cv = []
for train_index, test_index in kf.split(x):

    # Training phase
    x_train = x[train_index, :]
    y_train = y[train_index]

    regr_cv = MLPRegressor(hidden_layer_sizes=(20, 20),
max_iter=20000)
    regr_cv.fit(x_train, y_train)

    # Test phase
    x_test = x[test_index, :]
    y_test = y[test_index]

    y_pred = regr_cv.predict(x_test)

    # Calculate MSE and MAE

    mse_i = mean_squared_error(y_test, y_pred)
    mse_cv.append(mse_i)

    mae_i = mean_absolute_error(y_test, y_pred)
    mae_cv.append(mae_i)


print('MSE:', np.average(mse_cv), '  MAE:', np.average(mae_cv))
```

```
MSE: 173345.9181168463   MAE: 226.64356831457562
```

Adición de variables independientes

```
df2 = df.copy()

df2['M^2'] = df.M**2
df2['W^2'] = df.W**2
df2['H^2'] = df.H**2
df2['P^2'] = df.P**2
df2['MxW'] = df.M*df.W
df2['MxH'] = df.M*df.H
df2['MxP'] = df.M*df.P
df2['WxH'] = df.W*df.H
df2['WxP'] = df.W*df.P
df2['HxP'] = df.H*df.P
```

```
x = df2.iloc[:,1:].to_numpy()
y = df2.VR.to_numpy()

n_folds = 5
kf = KFold(n_splits=n_folds, shuffle = True)

mse_cv = []
mae_cv = []
for train_index, test_index in kf.split(x):

    # Training phase
    x_train = x[train_index, :]
    y_train = y[train_index]

    regr_cv = MLPRegressor(hidden_layer_sizes=(20, 20),
max_iter=20000)
    regr_cv.fit(x_train, y_train)

    # Test phase
    x_test = x[test_index, :]
    y_test = y[test_index]

    y_pred = regr_cv.predict(x_test)

    # Calculate MSE and MAE

    mse_i = mean_squared_error(y_test, y_pred)
    mse_cv.append(mse_i)

    mae_i = mean_absolute_error(y_test, y_pred)
    mae_cv.append(mae_i)


print('MSE:', np.average(mse_cv), '  MAE:', np.average(mae_cv))
```

```
MSE: 298270.51579787943   MAE: 373.334598698018
```

A. ¿Consideras que el modelo perceptrón multicapa es efectivo para modelar los datos del problema? ¿Por qué?

Opino que el modelo percetrón no es efectivo para el modelado de estos datos, ya que analizando el MSE y el MAE se puede notar como el resultado es exageradamente grande aun con validación cruzada, por lo que no creo que sea una buena opción usar el modelo MLP.

B. ¿Qué modelo es mejor para los datos de criminalidad, el lineal o el perceptrón multicapa? ¿Por qué?

Creo que para estos datos el modelo de regresión lineal es más apto, esto es debido a que el modelo de regresión lineal cuenta con un MSE y MAE más bajo a comparación del modelo perceptron multicapa

Ejercicio 2

En este ejercicio trabajarás con datos que vienen de un experimento en el que se midió actividad muscular con la técnica de la Electromiografía en el brazo derecho de varios participantes cuando éstos realizaban un movimiento con la mano entre siete posible (Flexionar hacia arriba, Flexionar hacia abajo, Cerrar la mano, Estirar la mano, Abrir la mano, Coger un objeto, No moverse).

```
data = np.loadtxt('/content/M_5.txt')
df = pd.DataFrame(data)

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

x = x.to_numpy()
y = y.to_numpy()
```

Validación cruzada

```
kf = StratifiedKFold(n_splits=5, shuffle = True)

cv_y_test = []
cv_y_pred = []

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

    # Training phase
    x_train = x[train_index, :]
    y_train = y[train_index]

    clf_cv = MLPClassifier(hidden_layer_sizes=((20, 20, 20, 20, 20)),
max_iter=10000)
    clf_cv.fit(x_train, y_train)

    # Test phase
    x_test = x[test_index, :]
    y_test = y[test_index]
    y_pred = clf_cv.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       0.98      0.92      0.95        90
         2.0       0.68      0.72      0.70        90
         3.0       0.99      0.93      0.96        90
         4.0       0.94      0.92      0.93        90
```

```
         5.0       0.95      0.96      0.95        90
         6.0       0.71      0.72      0.71        90
         7.0       0.97      1.00      0.98        89

    accuracy                           0.88       629
   macro avg       0.89      0.88      0.88       629
weighted avg       0.89      0.88      0.88       629
```

Número optimo de capas y neuronas

```
num_layers = np.arange(1, 20, 5)
num_neurons = np.arange(10, 110, 20)
layers = []
for l in num_layers:
  for n in num_neurons:
    layers.append(l*[n])
clf = GridSearchCV(MLPClassifier(max_iter=10000),
{'hidden_layer_sizes': layers},
cv = 5)
clf.fit(x, y)
print(clf.best_estimator_)
```

```
MLPClassifier(hidden_layer_sizes=[90], max_iter=10000)
```

```
layers
```

```
[[10],
 [30],
 [50],
 [70],
 [90],
 [10, 10, 10, 10, 10, 10],
 [30, 30, 30, 30, 30, 30],
 [50, 50, 50, 50, 50, 50],
 [70, 70, 70, 70, 70, 70],
 [90, 90, 90, 90, 90, 90],
 [10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10],
 [30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30],
 [50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50],
 [70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70],
 [90, 90, 90, 90, 90, 90, 90, 90, 90, 90, 90],
 [10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10],
 [30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30],
 [50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50],
 [70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70],
 [90, 90, 90, 90, 90, 90, 90, 90, 90, 90, 90, 90, 90, 90, 90]]
```

```
clf = GridSearchCV(MLPClassifier(max_iter=10000),
{'hidden_layer_sizes': layers}, cv = 5)
```

```
y_pred = cross_val_predict(clf, x, y, cv = 5)
print(classification_report(y, y_pred))
```

```
              precision    recall  f1-score   support

         1.0       0.95      0.90      0.93        90
         2.0       0.66      0.71      0.68        90
         3.0       0.99      0.96      0.97        90
         4.0       0.97      0.99      0.98        90
         5.0       0.90      0.96      0.92        90
         6.0       0.74      0.68      0.71        90
         7.0       0.97      0.98      0.97        89

    accuracy                           0.88       629
   macro avg       0.88      0.88      0.88       629
weighted avg       0.88      0.88      0.88       629
```

Obtención de los hiperparámetros óptimos de capas y neuronas de la red

```
print("----- Production model -----")
clf = GridSearchCV(MLPClassifier(max_iter=10000),
{'hidden_layer_sizes': layers},cv = 5)
clf.fit(x, y)
print(clf.best_estimator_)
```

```
----- Production model -----
MLPClassifier(hidden_layer_sizes=[90], max_iter=10000)
```

Ajuste del modelo con todos los datos e hipeparametros óptimos

```
clf_cv = MLPClassifier(hidden_layer_sizes=(50), max_iter=10000)
clf_cv.fit(x, y)
```

```
MLPClassifier(hidden_layer_sizes=50, max_iter=10000)
```

A. ¿Observas alguna mejora importante al optimizar el tamaño de la red? ¿Es el resultado que esperabas? Argumenta tu respuesta.

Creo que una forma de mejorar el modelo podria ser encontrando el número optimo de caracteristicas y descartando las caracteristicas que no sirvan para el modelo.

B. ¿Qué inconvenientes hay al encontrar el tamaño óptimo de la red? ¿Por qué?

El mayor inconveniente con el que me tope fue el timepo que tarda la red en ajustarse, debido a que se esta probando con diferentes hiperparametros y debido a la cantidad de datos con los que se cuenta.