Actividad: Ajuste de redes neuronales

Mi nombre es **Isai Ambrocio** con matrícula **A01625101** con lo cual me corresponde: Variable dependiente VR, variables independientes M, W, H y S.

```
import pandas as pd
import numpy as np
from sklearn.neural_network import MLPRegressor, MLPClassifier
from sklearn.model_selection import cross_val_score, GridSearchCV,
train_test_split
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import make_pipeline
```

Problema 1

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

df.head()
```

	State	VR	MR	M	W	H	P	S
0	AK	761	9.0	41.8	75.2	86.6	9.1	14.3
1	AL	780	11.6	67.4	73.5	66.9	17.4	11.5
2	AR	593	10.2	44.7	82.9	66.3	20.0	10.7
3	AZ	715	8.6	84.7	88.6	78.7	15.4	12.1
4	CA	1078	13.1	96.7	79.3	76.2	18.2	12.5

```
df.isna().sum()
```

```
State    0
VR       0
MR       0
M        0
W        0
H        0
P        0
S        0
dtype: int64
```

```
columnas_a_mantener = ["M", "W", "H", "S"]
X_original = df[columnas_a_mantener]
y = df["VR"]
```

1. Evalúa con validación cruzada un modelo pereceptrón multicapa para las variables que se te asignaron para este ejercicio.

```
mlp_model = MLPRegressor(random_state=1, max_iter=1000)
```

Validación cruzada.

```
scores = cross_val_score(mlp_model, X_original, y,
                         cv=5, scoring="neg_mean_squared_error")

mse_mean = -scores.mean()
print("Error cuadrático medio promedio (Originales):", mse_mean)
```

```
Error cuadrático medio promedio (Originales): 139813.3065454982
```

2. Agrega al conjunto de datos columnas que representen los cuadrados de las variables predictoras (por ejemplo, M2, W2), así como los productos entre pares de variables (por ejemplo, PxS, MxW). Evalúa un modelo perceptrón multicapa para este nuevo conjunto de datos.

```
poly = PolynomialFeatures(degree=2, include_bias=False)
X_extended = poly.fit_transform(X_original)

mlp_model_extended = MLPRegressor(random_state=1, max_iter=1000)
```

Validación cruzada.

```
scores_extended = cross_val_score(mlp_model_extended, X_extended, y,
cv=5,
                                  scoring="neg_mean_squared_error")

mse_mean_extended = -scores_extended.mean()
print("Error cuadrático medio promedio (Ejercicio 2):",
mse_mean_extended)
```

```
Error cuadrático medio promedio (Ejercicio 2): 108633.82545056226
```

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

El MLP parece ser efectivo para modelar los datos del problema, ya que tiene un error cuadrático medio promedio (MSE) más bajo en comparación con el modelo lineal utilizado en el ejercicio 1.

Un MSE promedio de 108633.83 en el ejercicio 2 indica que el MLP con variables predictoras extendidas tiene un rendimiento relativamente mejor que el modelo lineal con variables predictoras originales, que tiene un MSE promedio de 139813.31 en el ejercicio 1.

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

El modelo MLP con variables predictoras extendidas (Ejercicio 2) parece ser mejor para los datos de criminalidad en comparación con el modelo lineal (Ejercicio 1). Esto se debe a que el modelo MLP es más flexible y puede capturar relaciones no lineales entre las variables predictoras y la variable dependiente.

Al agregar cuadrados de variables y productos entre pares de variables, el MLP puede aprender patrones más complejos en los datos, lo que conduce a un MSE promedio más bajo. Por lo tanto,

en este contexto, el MLP es preferible al modelo lineal para modelar la relación entre las variables predictoras y la tasa de criminalidad.

Problema 2

Cargamos los datos

```
data = np.loadtxt("M_1.txt")

data

X = data[:, 1:]
y = data[:, 0]
```

1. Evalúa un modelo perceptrón multicapa con validación cruzada utilizando al menos 5 capas de 20 neuronas.

```
mlp_model_1 = MLPClassifier(hidden_layer_sizes=(20, 20, 20, 20, 20),
random_state=1)
```

Realizamos validación cruzada

```
scores_1 = cross_val_score(mlp_model_1, X, y, cv=5)

accuracy_mean_1 = scores_1.mean()
print("Precisión promedio (Paso 1):", accuracy_mean_1)
```

1. Evalúa un modelo perceptrón multicapa con validación cruzada, pero encontrando el número óptimo de capas y neuronas de la red.

```
param_grid = {
    "hidden_layer_sizes": [(20, 20), (30, 30), (40, 40), (50, 50)],
}

mlp_model_2 = MLPClassifier(random_state=1)

grid_search = GridSearchCV(mlp_model_2, param_grid, cv=5,
scoring="accuracy")
grid_search.fit(X, y)
```

Mejores hiperparametros

```
best_params = grid_search.best_params_
print("Mejores hiperparámetros (Paso 2):", best_params)

Mejores hiperparámetros (Paso 2): {'hidden_layer_sizes': (40, 40)}
```

1. 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.

```
best_hidden_layer_sizes = best_params['hidden_layer_sizes']
mlp_model_final =
MLPClassifier(hidden_layer_sizes=best_hidden_layer_sizes,
                                 random_state=1)
```

Ajustamos el modelo con todos los datos

```
mlp_model_final.fit(X, y)

MLPClassifier(hidden_layer_sizes=(40, 40), random_state=1)
```

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

Sí, se observa una mejora importante al optimizar el tamaño de la red. En el Paso 1, se utilizó un modelo MLP con 5 capas de 20 neuronas cada una, lo que da un total de 100 neuronas en la red.

En el Paso 2, se encontraron los mejores hiperparámetros, que resultaron en una red con 2 capas ocultas, cada una con 40 neuronas, lo que da un total de 80 neuronas en la red. Esto representa una reducción en el número de neuronas en comparación con el modelo inicial.

El resultado es el esperado en el sentido de que, en muchos casos, una red neuronal más grande no necesariamente conduce a un mejor rendimiento. En este caso, la reducción en el número de neuronas en la red condujo a un modelo más eficiente *(Princibio de Parsimonia).*

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

Uno de los inconvenientes más notorios fue el tiempo de ejecución, ya que fue un poco más lento. Del mismo modo, puede ser costoso desde el punto de vista computacional, especialmente si se exploran muchas combinaciones de hiperparámetros, como el número de capas y neuronas. Por el hecho de que conlleva entrenar y evaluar múltiples modelos, lo que puede llevar tiempo y recursos.

Isai Ambrocio - A01625101