

Actividad: Ajuste de redes neuronales

Link del Colab:

https://colab.research.google.com/drive/1IGXjttUQ_Fxul3oqWi_qGLNvOhhT1fPi?usp=sharing

▼ Ejercicio 1 (50 puntos)

El conjunto de datos de criminalidad de 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

Para este conjunto de datos:



Nota: Las variables con las que vas a trabajar depende del último número de tu matrícula de acuerdo a la siguiente lista:

- 0 - Variable dependiente VR, variables independientes M, W, H y P
- 1 - Variable dependiente VR, variables independientes M, W, H y S
- 2 - Variable dependiente VR, variables independientes M, W, S y P
- 3 - Variable dependiente VR, variables independientes M, H, S y P
- 4 - Variable dependiente MR, variables independientes M, W, H y P
- 5 - Variable dependiente MR, variables independientes M, W, H y S
- 6 - Variable dependiente MR, variables independientes M, W, S y P
- 7 - Variable dependiente MR, variables independientes M, H, S y P
- 8 - Variable dependiente VR, variables independientes M, W, H, P y S
- 9 - Variable dependiente MR, variables independientes M, W, H, P y S



```
import numpy as np
import pandas as pd
```

```
from sklearn.neural_network import MLPRegressor, MLPClassifier
from sklearn.model_selection import KFold, StratifiedKFold, GridSearchCV, cross_val_predict
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score, classification_report
```

```
df = pd.read_csv('/content/drive/MyDrive/7mo Semestre/Colab Notebooks/DataSources/crime_data.csv')
df
```

	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	
5	CO	567	5.8	81.8	92.5	84.4	9.9	12.1	
6	CT	456	6.3	95.7	89.0	79.2	8.5	10.1	
7	DE	686	5.0	82.7	79.4	77.5	10.2	11.4	
8	FL	1206	8.9	93.0	83.5	74.4	17.8	10.6	
9	GA	723	11.4	67.7	70.8	70.9	13.5	13.0	
10	HI	261	3.8	74.7	40.9	80.1	8.0	9.1	
11	IA	326	2.3	43.8	96.6	80.1	10.3	9.0	
12	ID	282	2.9	30.0	96.7	79.7	13.1	9.5	
13	IL	960	11.4	84.0	81.0	76.2	13.6	11.5	
14	IN	489	7.5	71.6	90.6	75.6	12.2	10.8	
15	KS	496	6.4	54.6	90.9	81.3	13.1	9.9	
16	KY	463	6.6	48.5	91.8	64.6	20.4	10.6	
17	LA	1062	20.3	75.0	66.7	68.3	26.4	14.9	
18	MA	805	3.9	96.2	91.1	80.0	10.7	10.9	
19	MD	998	12.7	92.8	68.9	78.4	9.7	12.0	

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

	MR	M	W	H	P	
0	9.0	41.8	75.2	86.6	9.1	
1	11.6	67.4	73.5	66.9	17.4	
2	10.2	44.7	82.9	66.3	20.0	
3	8.6	84.7	88.6	78.7	15.4	
4	13.1	96.7	79.3	76.2	18.2	
5	5.8	81.8	92.5	84.4	9.9	
6	6.3	95.7	89.0	79.2	8.5	
7	5.0	82.7	79.4	77.5	10.2	
8	8.9	93.0	83.5	74.4	17.8	
9	11.4	67.7	70.8	70.9	13.5	
10	3.8	74.7	40.9	80.1	8.0	
11	2.3	43.8	96.6	80.1	10.3	
12	2.9	30.0	96.7	79.7	13.1	
13	11.4	84.0	81.0	76.2	13.6	
14	7.5	71.6	90.6	75.6	12.2	
15	6.4	54.6	90.9	81.3	13.1	
16	6.6	48.5	91.8	64.6	20.4	
17	20.3	75.0	66.7	68.3	26.4	
18	3.9	96.2	91.1	80.0	10.7	
19	12.7	92.8	68.9	78.4	9.7	
20	1.6	35.7	98.5	78.8	10.7	
21	9.8	82.7	83.1	76.8	15.4	
22	3.4	69.3	94.0	82.4	11.6	
23	11.3	68.3	87.6	73.9	16.1	
24	13.5	30.7	63.3	64.3	24.7	
25	3.0	24.0	92.6	81.0	14.9	
26	11.3	66.3	75.2	70.0	14.4	
27	1.7	41.6	94.2	76.7	11.2	
28	3.9	50.6	94.3	81.8	10.3	
29	2.0	59.4	98.0	82.2	9.9	
30	5.3	100.0	80.8	76.7	10.9	
31	8.0	56.0	87.1	75.1	17.4	

32 10.4 84.8 86.7 78.8 9.8

33 13.3 91.7 77.2 74.8 16.4

34 6.0 81.3 87.5 75.7 13.0

35 8.4 60.1 82.5 74.6 19.9

36 4.6 70.0 93.2 81.5 11.8

37 6.8 84.8 88.7 74.7 13.2

38 3.9 93.6 92.6 72.0 11.2

39 10.3 69.8 68.6 68.3 18.7

40 3.4 32.6 90.2 77.1 14.2

41 10.0 87.7 88.0 87.1 10.0

```
x = df.iloc[:, 1:].values
```

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

43 3.1 77.5 94.8 85.1 10.7

1. Evalúa con validación cruzada un modelo perceptrón

- ▼ multicapa para las variables que se te asignaron para este ejercicio.

45 10.0 87.7 88.0 87.1 10.0

```
n_folds = 5
```

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

```
mse_cv = []
```

```
mae_cv = []
```

```
r2_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)
```

```
    print('mse = ', mse_i)
```

```
    mse_cv.append(mse_i)
```

```

mae_i = mean_absolute_error(y_test, y_pred)
print('mae = ', mae_i)
mae_cv.append(mae_i)

r2_i = r2_score(y_test, y_pred)
print('r^2= ', r2_i)
r2_cv.append(r2_i)

print('\nPromedios:')
print('MSE:', np.average(mse_cv), ' MAE:', np.average(mae_cv), ' R^2:', np.average(r2_cv))

mse = 3.1398140137808963
mae = 1.3691213288298802
r^2= 0.7642489726050633
mse = 3.8899111325139564
mae = 1.6668928333098278
r^2= 0.8574189255037568
mse = 118.67207354787729
mae = 5.0645093754220145
r^2= -8.045058616007292
mse = 50.01810766493709
mae = 5.905111128330297
r^2= -5.945030222846026
mse = 546.7824764436017
mae = 9.591302958230788
r^2= -0.18333537800012434

Promedios:
MSE: 144.50047656054218 MAE: 4.719387524824562 R^2: -2.510351263748925

```

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.

```

df2 = pd.DataFrame({'M2':df['M']**2,
                    'W2':df['W']**2,
                    'H2':df['H']**2,
                    'P2':df['P']**2,
                    'MxW':df['M']*df['W'],
                    'MxH':df['M']*df['H'],
                    'MxP':df['M']*df['P'],
                    'WxH':df['W']*df['H'],
                    'WxP':df['W']*df['P'],
                    'HxP':df['H']*df['P']})

df2 = pd.concat([df, df2], axis = 1)
df2

```

	MR	M	W	H	P	M2	W2	H2	P2	MxW	MxH	
0	9.0	41.8	75.2	86.6	9.1	1747.24	5655.04	7499.56	82.81	3143.36	3619.88	
1	11.6	67.4	73.5	66.9	17.4	4542.76	5402.25	4475.61	302.76	4953.90	4509.06	1
2	10.2	44.7	82.9	66.3	20.0	1998.09	6872.41	4395.69	400.00	3705.63	2963.61	
3	8.6	84.7	88.6	78.7	15.4	7174.09	7849.96	6193.69	237.16	7504.42	6665.89	1
4	13.1	96.7	79.3	76.2	18.2	9350.89	6288.49	5806.44	331.24	7668.31	7368.54	1
5	5.8	81.8	92.5	84.4	9.9	6691.24	8556.25	7123.36	98.01	7566.50	6903.92	
6	6.3	95.7	89.0	79.2	8.5	9158.49	7921.00	6272.64	72.25	8517.30	7579.44	
7	5.0	82.7	79.4	77.5	10.2	6839.29	6304.36	6006.25	104.04	6566.38	6409.25	
8	8.9	93.0	83.5	74.4	17.8	8649.00	6972.25	5535.36	316.84	7765.50	6919.20	1
9	11.4	67.7	70.8	70.9	13.5	4583.29	5012.64	5026.81	182.25	4793.16	4799.93	
10	3.8	74.7	40.9	80.1	8.0	5580.09	1672.81	6416.01	64.00	3055.23	5983.47	
11	2.3	43.8	96.6	80.1	10.3	1918.44	9331.56	6416.01	106.09	4231.08	3508.38	
12	2.9	30.0	96.7	79.7	13.1	900.00	9350.89	6352.09	171.61	2901.00	2391.00	
13	11.4	84.0	81.0	76.2	13.6	7056.00	6561.00	5806.44	184.96	6804.00	6400.80	1
14	7.5	71.6	90.6	75.6	12.2	5126.56	8208.36	5715.36	148.84	6486.96	5412.96	
15	6.4	54.6	90.9	81.3	13.1	2981.16	8262.81	6609.69	171.61	4963.14	4438.98	
16	6.6	48.5	91.8	64.6	20.4	2352.25	8427.24	4173.16	416.16	4452.30	3133.10	
17	20.3	75.0	66.7	68.3	26.4	5625.00	4448.89	4664.89	696.96	5002.50	5122.50	1
18	3.9	96.2	91.1	80.0	10.7	9254.44	8299.21	6400.00	114.49	8763.82	7696.00	1
19	12.7	92.8	68.9	78.4	9.7	8611.84	4747.21	6146.56	94.09	6393.92	7275.52	
20	1.6	35.7	98.5	78.8	10.7	1274.49	9702.25	6209.44	114.49	3516.45	2813.16	
21	9.8	82.7	83.1	76.8	15.4	6839.29	6905.61	5898.24	237.16	6872.37	6351.36	1
22	3.4	69.3	94.0	82.4	11.6	4802.49	8836.00	6789.76	134.56	6514.20	5710.32	
23	11.3	68.3	87.6	73.9	16.1	4664.89	7673.76	5461.21	259.21	5983.08	5047.37	1
24	13.5	30.7	63.3	64.3	24.7	942.49	4006.89	4134.49	610.09	1943.31	1974.01	
25	3.0	24.0	92.6	81.0	14.9	576.00	8574.76	6561.00	222.01	2222.40	1944.00	
26	11.3	66.3	75.2	70.0	14.4	4395.69	5655.04	4900.00	207.36	4985.76	4641.00	
27	1.7	41.6	94.2	76.7	11.2	1730.56	8873.64	5882.89	125.44	3918.72	3190.72	
28	3.9	50.6	94.3	81.8	10.3	2560.36	8892.49	6691.24	106.09	4771.58	4139.08	
29	2.0	59.4	98.0	82.2	9.9	3528.36	9604.00	6756.84	98.01	5821.20	4882.68	
30	5.3	100.0	80.8	76.7	10.9	10000.00	6528.64	5882.89	118.81	8080.00	7670.00	1
31	8.0	56.0	87.1	75.1	17.4	3136.00	7586.41	5640.01	302.76	4877.60	4205.60	

32	10.4	84.8	86.7	78.8	9.8	7191.04	7516.89	6209.44	96.04	7352.16	6682.24	
33	13.3	91.7	77.2	74.8	16.4	8408.89	5959.84	5595.04	268.96	7079.24	6859.16	1
34	6.0	81.3	87.5	75.7	13.0	6609.69	7656.25	5730.49	169.00	7113.75	6154.41	1
35	8.4	60.1	82.5	74.6	19.9	3612.01	6806.25	5565.16	396.01	4958.25	4483.46	1
36	4.6	70.0	93.2	81.5	11.8	4900.00	8686.24	6642.25	139.24	6524.00	5705.00	
37	6.8	84.8	88.7	74.7	13.2	7191.04	7867.69	5580.09	174.24	7521.76	6334.56	1
38	3.9	93.6	92.6	72.0	11.2	8760.96	8574.76	5184.00	125.44	8667.36	6739.20	1
39	10.3	69.8	68.6	68.3	18.7	4872.04	4705.96	4664.89	349.69	4788.28	4767.34	1
40	3.4	32.6	90.2	77.1	14.2	1062.76	8136.04	5944.41	201.64	2940.52	2513.46	
41	10.2	67.7	82.8	67.1	19.6	4583.29	6855.84	4502.41	384.16	5605.56	4542.67	1
42	11.9	83.9	85.1	72.1	17.4	7039.21	7242.01	5198.41	302.76	7139.89	6049.19	1
43	3.1	77.5	94.8	85.1	10.7	6006.25	8987.04	7242.01	114.49	7347.00	6595.25	
44	8.3	77.5	77.1	75.2	9.7	6006.25	5944.41	5655.04	94.09	5975.25	5828.00	
45	3.6	27.0	98.4	80.8	10.0	729.00	9682.56	6528.64	100.00	2656.80	2181.60	
46	5.2	83.0	89.4	83.8	12.1	6889.00	7992.36	7022.44	146.41	7420.20	6955.40	1
47	4.4	68.1	92.1	78.6	12.6	4637.61	8482.41	6177.96	158.76	6272.01	5352.66	
48	6.9	41.8	96.3	66.0	22.2	1747.24	9273.69	4356.00	492.84	4025.34	2758.80	
49	3.4	29.7	95.9	83.0	13.3	882.09	9196.81	6889.00	176.89	2848.23	2465.10	
50	78.5	100.0	31.8	73.1	26.4	10000.00	1011.24	5343.61	696.96	3180.00	7310.00	2

```
x = df.iloc[:, 1:].values
y = df.iloc[:, 0].values
```

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

```
mse_cv = []
mae_cv = []
r2_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)
print('mse = ', mse_i)
mse_cv.append(mse_i)

mae_i = mean_absolute_error(y_test, y_pred)
print('mae = ', mae_i)
mae_cv.append(mae_i)

r2_i = r2_score(y_test, y_pred)
print('r^2= ', r2_i)
r2_cv.append(r2_i)

print('\nPromedios:')
print('MSE:', np.average(mse_cv), ' MAE:', np.average(mae_cv), ' R^2:', np.average(r2_cv))

mse = 8.875119768646503
mae = 2.0610640986618565
r^2= 0.1572710570460435
mse = 7.1496297649837075
mae = 2.0745242389787224
r^2= 0.7435027582959196
mse = 56.87852235256317
mae = 6.358454883834243
r^2= -1.9135602065650632
mse = 51.102995285141404
mae = 5.7973182981766795
r^2= -4.804783870818915
mse = 611.6188059656437
mae = 10.708382524231297
r^2= -0.3081554782794129

Promedios:
MSE: 147.1250146273957 MAE: 5.39994880877656 R^2: -1.2251451480642856

```

3. 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?

No mucho, al menos en este conjunto de datos generó un MSE entre 90 y 130, así que hay una medida de error considerablemente alta en cuanto a sus predicciones. Sin mencionar que su R^2 es muy mala.

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

El lineal. El modelo lineal ofrece predicciones con menos márgenes de error, aunque tampoco es perfecto, también tiene una medida de error bastante considerable, y su R^2 tampoco es la mejor; pero presenta mejores resultados que el modelo perceptrón multicapa.

▼ Ejercicio 2 (50 puntos)

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 posibles (Flexionar hacia arriba, Flexionar hacia abajo, Cerrar la mano, Estirar la mano, Abrir la mano, Coger un objeto, No moverse). A su vez, la primera columna corresponde a la clase (1, 2, 3, 4, 5, 6, y 7), la segunda columna se ignora, y el resto de las columnas indican las variables que se calcularon de la respuesta muscular. El archivo de datos con el que trabajarás depende de tu matrícula.

Nota: El conjunto de datos con el que trabajarás en este ejercicio depende del penúltimo número de tu matrícula de acuerdo a la siguiente lista:

- 0 y 1 - M_1.txt
- 2 y 3 - M_2.txt
- 4 y 5 - M_3.txt
- 6 y 7 - M_4.txt
- 8 y 9 - M_5.txt

```
data = np.loadtxt('/content/drive/MyDrive/7mo Semestre/Colab Notebooks/DataSources/M_5.txt')
df = pd.DataFrame(data)
df
```

	0	1	2	3	4	5	6	7	8
0	1.0	1.0	0.159910	0.829038	-0.236322	-1.137015	0.049065	-1.331090	0.081879
1	1.0	1.0	-1.039646	0.061581	-0.372804	-0.315868	0.351879	-1.399993	-0.981714
2	1.0	1.0	-1.411644	-1.090915	-1.164213	-1.041624	0.055639	-2.163669	-1.410827

```
data = np.delete(data, 1, axis=1)
df = pd.DataFrame(data)
df
```

	0	1	2	3	4	5	6	7
0	1.0	0.159910	0.829038	-0.236322	-1.137015	0.049065	-1.331090	0.081879
1	1.0	-1.039646	0.061581	-0.372804	-0.315868	0.351879	-1.399993	-0.981714
2	1.0	-1.411644	-1.090915	-1.164213	-1.041624	0.055639	-2.163669	-1.410827
3	1.0	-2.645974	0.129788	2.822250	-1.345104	-0.196804	3.303073	-3.014899
4	1.0	-1.692860	-1.597632	-2.690969	-0.413617	-1.163711	-2.757666	-1.719198
...
624	7.0	-7.042298	-6.180162	-4.875976	-7.048198	-7.099642	-4.785359	-6.666241
625	7.0	-6.826647	-5.981072	-4.807238	-7.361130	-6.843612	-4.523671	-6.360528
626	7.0	-7.717987	-5.548352	-4.399172	-6.311186	-7.481042	-4.227248	-6.762767
627	7.0	-7.447476	-5.223852	-4.073834	-6.553809	-7.143856	-3.894904	-6.507190
628	7.0	-8.059398	-5.362028	-4.429245	-6.694931	-7.674423	-4.078131	-6.847052

629 rows × 631 columns

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

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

```
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	0.97	0.93	0.95	90
2.0	0.73	0.67	0.70	90
3.0	0.98	0.96	0.97	90
4.0	0.97	0.96	0.96	90
5.0	0.95	0.96	0.95	90
6.0	0.73	0.80	0.76	90
7.0	0.95	0.99	0.97	89
accuracy			0.89	629
macro avg	0.89	0.89	0.89	629
weighted avg	0.89	0.89	0.89	629

2. Evalúa un modelo perceptrón multicapa con validación cruzada,

- ▼ pero encontrando el número óptimo de capas y neuronas de la red.

```

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=[30], max_iter=10000)

```

- ▼ 3. 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.

```
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	0.98	0.92	0.95	90
2.0	0.63	0.60	0.61	90
3.0	0.95	0.98	0.96	90
4.0	1.00	0.98	0.99	90
5.0	0.91	0.96	0.93	90
6.0	0.66	0.68	0.67	90
7.0	0.98	0.99	0.98	89
accuracy			0.87	629
macro avg	0.87	0.87	0.87	629
weighted avg	0.87	0.87	0.87	629

4. Contesta lo siguientes:

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

No es el resultado que esperaba, esperaba mayor precisión y recall, pero en el entrenamiento con validación cruzada y con 5 capas de 20 neuronas, dió mejores resultados que la red con el número óptimo de neuronas.

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

Que el procesamiento puede ser mucho, al probar tantas combinaciones distintas de neuronas y capas, termina necesitando mucho esfuerzo computacional.

✓ 4s completed at 1:58 PM

● ✕