SAA - ACTIVIDAD 2.19 - ÁLVARO FERNANDEZ BECERRA

Escoge cualquiera de los ejercicios anteriores y utiliza la clase MinMaxScaler de sklearn.preprocessing para normalizar el conjunto de predictores.

En mi caso utilizo la actividad anterior que carga y limpia el dataset de enfermedades del corazon, normaliza y muestra el score 80% entrenamiento 20% validacion.

```
import pandas as pd
import numpy as np

from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

# IMPORTAR MinMaxScaler
from sklearn.preprocessing import MinMaxScaler

url = "https://archive.ics.uci.edu/ml/machine-learning-databases/heart-disease/processed.cleveland.data"

# Nombres de columnas
columnas = [
    'age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg',
    'thalach', 'exang', 'oldpeak', 'slope', 'ca', 'thal', 'num'
]

df = pd.read_csv(url, header=None, names=columnas, na_values='?')
df.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	num
0	63.0	1.0	1.0	145.0	233.0	1.0	2.0	150.0	0.0	2.3	3.0	0.0	6.0	0
1	67.0	1.0	4.0	160.0	286.0	0.0	2.0	108.0	1.0	1.5	2.0	3.0	3.0	2
2	67.0	1.0	4.0	120.0	229.0	0.0	2.0	129.0	1.0	2.6	2.0	2.0	7.0	1
3	37.0	1.0	3.0	130.0	250.0	0.0	0.0	187.0	0.0	3.5	3.0	0.0	3.0	0
4	41.0	0.0	2.0	130.0	204.0	0.0	2.0	172.0	0.0	1.4	1.0	0.0	3.0	0

Pasos siguientes: Generar código con df | New interactive sheet

```
#Comprobar valores nulos a sustituir:
df.isna().sum()
```

	0
age	0
sex	0
cp	0
trestbps	0
chol	0
fbs	0
restecg	0
thalach	0
exang	0
oldpeak	0
slope	0
ca	4
thal	2
num	0

dtype: int64

```
#1 - Sustituir valores nulos :
# Sustituir '?' por NaN
df = df.replace('?', np.nan)
df = df.dropna()

# Convertir todas las columnas a numericas
for col in columnas:
    df[col] = pd.to_numeric(df[col], errors='coerce')

df.info()
#Convertir target a binario 0-> sano 1-4(1)-> enfermo. Sin usar lambda!
df['num'] = (df['num'] > 0).astype(int)
# Categoricas a binario
df= pd.get_dummies(df, columns=["sex", "cp", "fbs", "restecg", "exang", "slope", "thal"])
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 297 entries, 0 to 301
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       297 non-null    float64
 1   sex       297 non-null    float64
 2   cp        297 non-null    float64
```

```
 3   trestbps  297 non-null    float64
 4   chol      297 non-null    float64
 5   fbs       297 non-null    float64
 6   restecg   297 non-null    float64
 7   thalach   297 non-null    float64
 8   exang     297 non-null    float64
 9   oldpeak   297 non-null    float64
 10  slope     297 non-null    float64
 11  ca        297 non-null    float64
 12  thal      297 non-null    float64
 13  num       297 non-null    int64
dtypes: float64(13), int64(1)
memory usage: 34.8 KB
```

```
#2. NORMALIZACIÓN
x = df.drop('num', axis=1)
y = df['num']
x_num = x.select_dtypes(include=['number'])

# uso de Minmaxscaler
escalador = MinMaxScaler()
x_norm = escalador.fit_transform(x)

#Slit 80% - 20%
X_entrenamiento, X_validacion, y_entrenamiento, y_validacion = train_test_split(
    x_norm, y, test_size=0.2, random_state=42, stratify=y)

clf = KNeighborsClassifier(n_neighbors=5)

clf.fit(X_entrenamiento, y_entrenamiento)

# Resultados
y_pred_entrenamiento = clf.predict(X_entrenamiento)
y_pred_validacion = clf.predict(X_validacion)
```

```
# Accuracy
accuracy_entrenamiento = accuracy_score(y_entrenamiento, y_pred_entrenamiento)
accuracy_validacion    = accuracy_score(y_validacion, y_pred_validacion)

print(f"Accuracy en entrenamiento: {accuracy_entrenamiento:.4f}")
print(f"Accuracy en validación   : {accuracy_validacion:.4f}")
```

```
Accuracy en entrenamiento: 0.8397
Accuracy en validación   : 0.8667
```