6-escalado-normalizacion

October 22, 2024

1 Escalado y Normalización

Al escalar, está cambiando el rango de sus datos, mientras que en la normalización, está cambiando la forma de la distribución de sus datos.

[ ]:
```
import pandas as pd
import numpy as np
from scipy import stats
from mlxtend.preprocessing import minmax_scaling
import seaborn as sns
import matplotlib.pyplot as plt
```

1.1 Escalada

Esto significa que está transformando sus datos para que se ajusten a una escala específica, como 0-100 o 0-1. Desea escalar los datos cuando utiliza métodos basados en medidas de la distancia entre los puntos de datos, como las máquinas de vectores de soporte (SVM) o los k-vecinos más cercanos (KNN). Con estos algoritmos, se le da la misma importancia a un cambio de "1" en cualquier característica numérica.

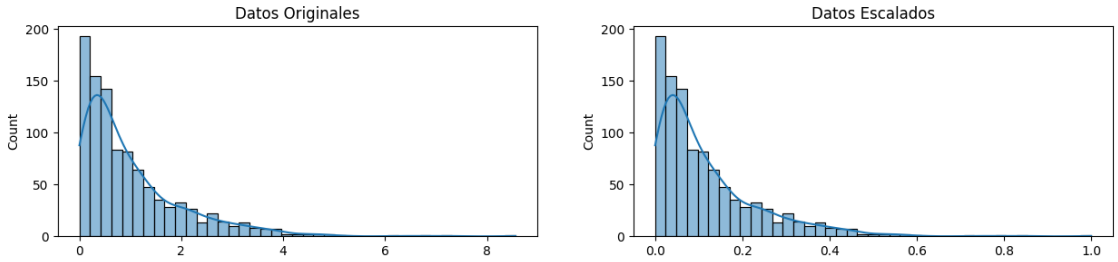
Al escalar sus variables, puede ayudar a comparar diferentes variables en igualdad de condiciones. ### Min Max Scalling

[ ]:
```
# Semilla.
np.random.seed(0)
# Mil puntos de datos aleatorios de una distribución exponencial.
original_data = np.random.exponential(size=1000)

# Mix-max escala los datos entre 0 y 1.
scaled_data = minmax_scaling(original_data, columns=[0])
```

[ ]:
```
# Graficar ambos juntos para comparar.
fig, ax = plt.subplots(1, 2, figsize=(15, 3))
# Datos originales.
sns.histplot(original_data, ax=ax[0], kde=True, legend=False)
ax[0].set_title("Datos Originales")
# Datos escalados.
sns.histplot(scaled_data, ax=ax[1], kde=True, legend=False)
ax[1].set_title("Datos Escalados")
```

```
plt.show()
```



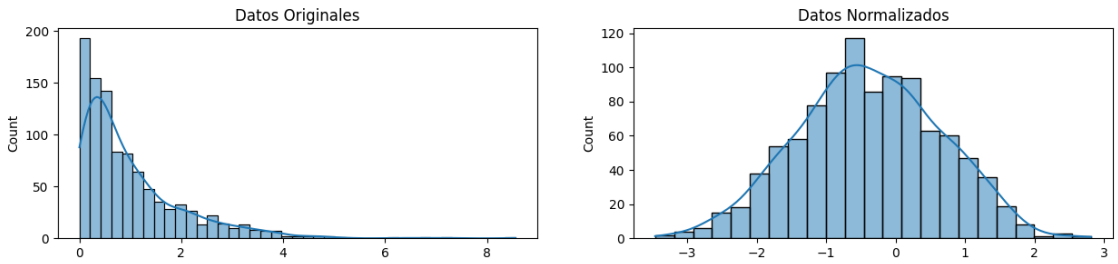
1.2 Normalización

El objetivo de la normalización es cambiar sus observaciones para que puedan describirse como una distribución normal.

Distribución normal: también conocida como "curva de campana", se trata de una distribución estadística específica en la que aproximadamente las mismas observaciones caen por encima y por debajo de la media, la media y la mediana son iguales y hay más observaciones más cercanas a la media. La distribución normal también se conoce como distribución gaussiana. ### Box-Cox

[ ]:
```
# Normalizar los datos exponenciales con boxcox.
normalized_data = stats.boxcox(original_data)
```

[ ]:
```
# Graficar ambos juntos para comparar.
fig,ax = plt.subplots(1,2,figsize=(15,3))
sns.histplot(original_data, ax=ax[0],kde=True, legend=False)
ax[0].set_title("Datos Originales")
sns.histplot(normalized_data[0], ax=ax[1], kde=True, legend=False)
ax[1].set_title("Datos Normalizados")
plt.show()
```



Observe que la forma de nuestros datos ha cambiado. Antes de normalizarse tenía casi forma de L. Pero después de normalizar se parece más al contorno de una campana (de ahí la "curva de campana").