Exploración y Curación de Datos Transformaciones

EyCD 2024

Escalar y Normalizar

Son técnicas de preprocesamiento de datos que se utilizan en el campo del aprendizaje automático y el análisis de datos. Ambas técnicas tienen como objetivo principal ajustar y manipular las escalas y distribuciones de las variables para mejorar la calidad y el rendimiento de los modelos y algoritmos.

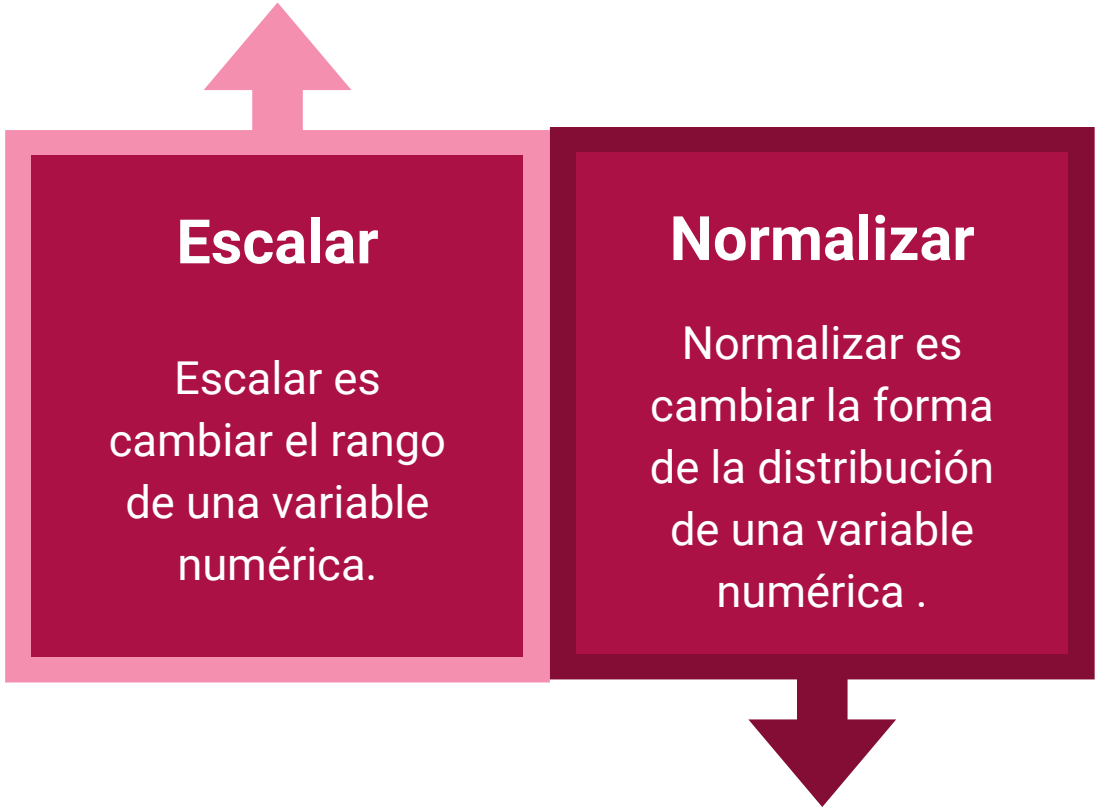
Escalar (Scaling)

Scaling se refiere a la transformación de los valores de una variable para que se encuentren dentro de un rango específico, generalmente entre 0 y 1, o -1 y 1. Al escalar una variable, los valores originales se ajustan proporcionalmente a una nueva escala, manteniendo la relación relativa entre ellos.

Normalizar

La normalización se refiere a ajustar los valores de una variable o conjunto de datos para que cumplan ciertas propiedades estadísticas, como una media de 0 y una desviación estándar de 1. La normalización también puede referirse a la transformación de los datos para que tengan una distribución específica, como una distribución normal estándar (conocida como normalización z-score).

Escalar vs Normalizar, cuál es la diferencia?



Cuando escalar y cuando normalizar

ESCALAR: Si se quiere mantener la forma y la distribución de los datos y ajustarlos a una escala específica, utiliza el escalamiento.

NORMALIZACIÓN: Si se desea llevar los datos en una escala común y eliminar la influencia de la magnitud o la unidad de medida, utiliza la normalización.

Ventajas de Escalar y Normalizar

Evitar la dominancia de variables: cuando los datos tienen diferentes escalas, una variable con valores más grandes puede dominar sobre las variables con valores más pequeños

Ejemplo:

Supongamos que tenemos un conjunto de datos que contiene dos variables: "Ingresos mensuales" (en dólares) y "Edad" (en años), y queremos predecir si una persona es propensa a comprar un determinado producto. Los valores de "Ingresos mensuales" varían entre 1000 y 50000 dólares, mientras que los valores de "Edad" varían entre 18 y 70 años.

Si no escalamos los datos y entrenamos un modelo de regresión logística sin ninguna transformación, la variable "Ingresos mensuales" podría dominar sobre la variable "Edad" debido a su mayor escala. Esto puede llevar a que el modelo otorgue una importancia excesiva a los ingresos mensuales y subestime la influencia de la edad en la predicción.

Ventajas de Escalar y Normalizar

Mejorar la eficiencia de los algoritmos: Algunos algoritmos de aprendizaje automático, como los algoritmos basados en distancias (por ejemplo, k-means o k vecinos más cercanos), pueden verse afectados negativamente por las diferencias de escala entre las variables. Al escalar o normalizar los datos, se mejora la comparabilidad y la eficiencia de estos algoritmos.

Acelerar la convergencia del modelo: Al escalar o normalizar los datos, se puede lograr una convergencia más rápida durante el entrenamiento del modelo. Esto es particularmente útil en algoritmos que utilizan métodos iterativos, como el descenso de gradiente, donde una escala inadecuada de los datos puede relentizar la convergencia.

Manejar características con diferentes unidades: Cuando se tienen características con diferentes unidades de medida (por ejemplo, peso en kilogramos y altura en centímetros), escalar o normalizar los datos permite que todas las características estén en la misma escala, lo que facilita su comparación y análisis.

Mejorar la interpretación de los resultados: Al escalar o normalizar los datos, se pueden obtener resultados más fácilmente interpretables. Las variables estarán en la misma escala, lo que facilita la comparación de sus efectos y contribuciones en el modelo.

Notebook

04_Transformaciones-2023.ipynb

Escalado

- El escalado implica transformar los datos para que ajusten a una escala determinada, como 0-100 o [0,1].
- Es importante escalar los datos antes de usar métodos basados en distancias, como Support Vector Machines o K nearest neighbors.
- Por ejemplo, si se tienen precios de productos en yens y en dolares, un dolar son más de 100 yens, pero estos algoritmos consideran el cambio en una unidad igual para ambas características.
- Escalando, las variables pasan a ser pesadas de la misma forma
- Sklearn tiene los métodos MinMaxScaler y MaxAbsScaler

Escalado

- El escalado implica transformar los datos para que ajusten a una escala determinada, como 0-100 o [0,1].
- Es importante escalar los datos antes de usar métodos basados en distancias, como Support Vector Machines o K nearest neighbors.
- Por ejemplo, si se tienen precios de productos en yens y en dolares, un dolar son más de 100 yens, pero estos algoritmos consideran el cambio en una unidad igual para ambas características.
- Escalando, las variables pasan a ser pesadas de la misma forma
- Sklearn tiene los métodos MinMaxScaler y MaxAbsScaler

Escalado

- El escalado implica transformar los datos para que ajusten a una escala determinada, como 0-100 o [0,1].
- Es importante escalar los datos antes de usar métodos basados en distancias, como Support Vector Machines o K nearest neighbors.
- Por ejemplo, si se tienen precios de productos en yens y en dolares, un dolar son más de 100 yens, pero estos algoritmos consideran el cambio en una unidad igual para ambas características.
- Escalando, las variables pasan a ser pesadas de la misma forma
- Sklearn tiene los métodos MinMaxScaler y MaxAbsScaler

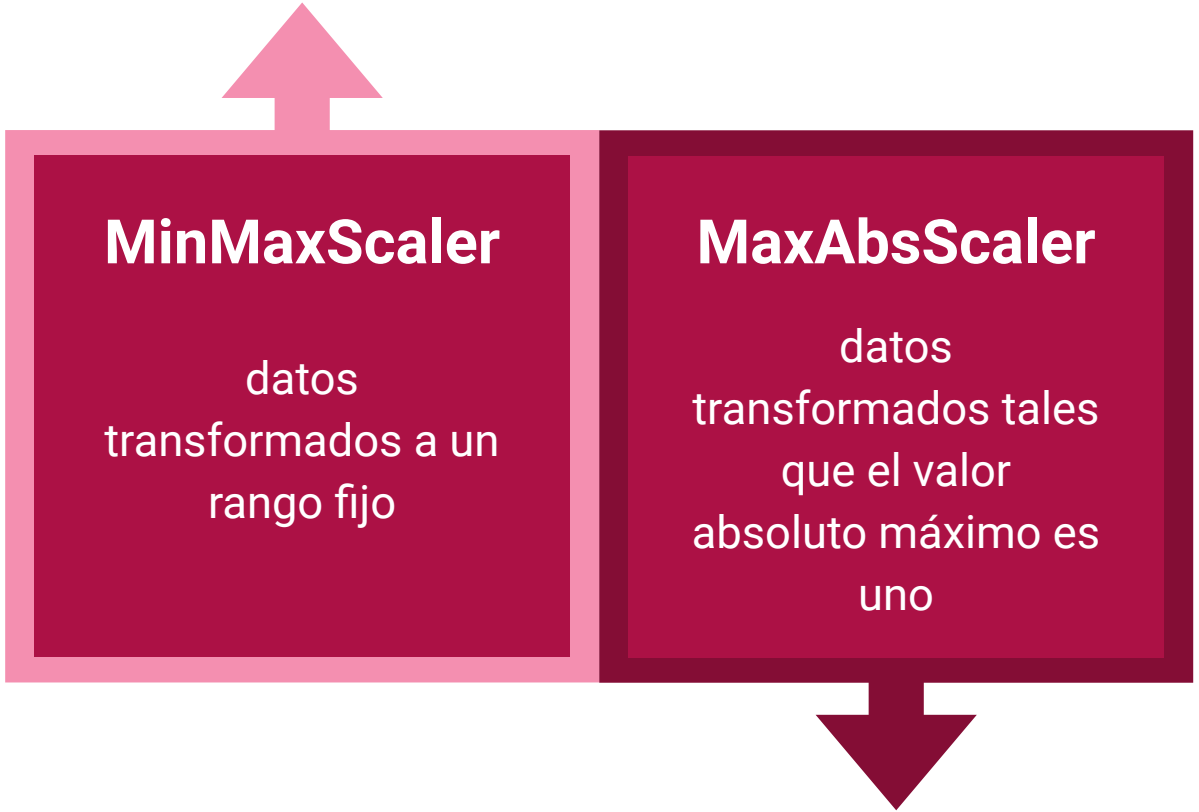
Escalado

- El escalado implica transformar los datos para que ajusten a una escala determinada, como 0-100 o [0,1].
- Es importante escalar los datos antes de usar métodos basados en distancias, como Support Vector Machines o K nearest neighbors.
- Por ejemplo, si se tienen precios de productos en yens y en dolares, un dolar son más de 100 yens, pero estos algoritmos consideran el cambio en una unidad igual para ambas características.
- Escalando, las variables pasan a ser pesadas de la misma forma
- Sklearn tiene los métodos MinMaxScaler y MaxAbsScaler

Escalado

- El escalado implica transformar los datos para que ajusten a una escala determinada, como 0-100 o [0,1].
- Es importante escalar los datos antes de usar métodos basados en distancias, como Support Vector Machines o K nearest neighbors.
- Por ejemplo, si se tienen precios de productos en yens y en dolares, un dolar son más de 100 yens, pero estos algoritmos consideran el cambio en una unidad igual para ambas características.
- Escalando, las variables pasan a ser pesadas de la misma forma
- Sklearn tiene los métodos MinMaxScaler y MaxAbsScaler

MinMaxScaler vs MaxAbsScaler



MinMaxScaler vs MaxAbsScaler

`MinMaxScaler` La motivación para usar esta escala incluye la solidez contra las desviaciones estándar muy pequeñas en las variables y la preservación de entradas cero en datos ralos.

`MaxAbsScaler` funciona de manera muy similar, pero escala forzando los datos al rango [-1, 1], dividiendo por el valor máximo (absoluto) en cada característica. Está destinado a datos que ya están centrados en cero o datos ralos.

Centrar datos ralos destruiría la estructura de dispersión de los datos y, por lo tanto, rara vez es algo sensato. Sin embargo, puede tener sentido escalar entradas ralas, especialmente si las variables están en escalas diferentes.

`MaxAbsScaler` fue diseñado específicamente para escalar datos ralos y es la forma recomendada de hacerlo

MinMaxScaler vs MaxAbsScaler

`MinMaxScaler` La motivación para usar esta escala incluye la solidez contra las desviaciones estándar muy pequeñas en las variables y la preservación de entradas cero en datos ralos.

`MaxAbsScaler` funciona de manera muy similar, pero escala forzando los datos al rango [-1, 1], dividiendo por el valor máximo (absoluto) en cada característica. Está destinado a datos que ya están centrados en cero o **datos ralos**.

Centrar datos ralos destruiría la estructura de dispersión de los datos y, por lo tanto, rara vez es algo sensato. Sin embargo, puede tener sentido escalar entradas ralas, especialmente si las variables están en escalas diferentes.

`MaxAbsScaler` fue diseñado específicamente para escalar datos ralos y es la forma recomendada de hacerlo

Standarization

- La estandarización de conjuntos de datos es un requisito común para muchos estimadores de aprendizaje automático implementados en scikit-learn; podrían comportarse mal si las características individuales no se parecen más o menos a datos distribuidos normal patrón: **Gaussiano con media cero y varianza unitaria.**
- En la práctica, a menudo ignoramos la forma de la distribución y simplemente transformamos los datos para centrarlos eliminando el valor medio de cada característica, luego la escalamos dividiendo las características no constantes por su desviación estándar usando `StandardScaler`
- Si sus datos contienen muchos valores atípicos, es probable que el escalado utilizando la media y la varianza de los datos no funcione muy bien. En estos casos, puede usar `RobustScaler` como reemplazo directo. Utiliza estimaciones más sólidas para el centro y rango de sus datos.

Standarization

- La estandarización de conjuntos de datos es un requisito común para muchos estimadores de aprendizaje automático implementados en scikit-learn; podrían comportarse mal si las características individuales no se parecen más o menos a datos distribuidos normal patrón: Gaussiano con media cero y varianza unitaria.
- En la práctica, a menudo ignoramos la forma de la distribución y simplemente transformamos los datos para centrarlos eliminando el valor medio de cada característica, luego la escalamos dividiendo las características no constantes por su desviación estándar usando `StandardScaler`
- Si sus datos contienen muchos valores atípicos, es probable que el escalado utilizando la media y la varianza de los datos no funcione muy bien. En estos casos, puede usar `RobustScaler` como reemplazo directo. Utiliza estimaciones más sólidas para el centro y rango de sus datos.

Standarization

- La estandarización de conjuntos de datos es un requisito común para muchos estimadores de aprendizaje automático implementados en scikit-learn; podrían comportarse mal si las características individuales no se parecen más o menos a datos distribuidos normal patrón: Gaussiano con media cero y varianza unitaria.
- En la práctica, a menudo ignoramos la forma de la distribución y simplemente transformamos los datos para centrarlos eliminando el valor medio de cada característica, luego la escalamos dividiendo las características no constantes por su desviación estándar usando `StandardScaler`
- Si sus datos contienen muchos valores atípicos, es probable que el escalado utilizando la media y la varianza de los datos no funcione muy bien. En estos casos, puede usar `RobustScaler` como reemplazo directo. Utiliza estimaciones más sólidas para el centro y rango de sus datos.

Normalization

- Hay dos tipos de transformaciones disponibles: **transformaciones de cuantiles y transformaciones de potencia.** Tanto las transformaciones de cuantiles como las de potencia se basan en transformaciones monótonas de las características y, por lo tanto, preservan el rango de los valores a lo largo de cada característica.
- Las transformadas de cuantiles colocan todas las características en la misma distribución deseada según la fórmula G ^ {- 1} (F (X)) donde F es la función de distribución acumulativa de la característica y G ^{− 1} la función de cuantiles de la distribución de salida deseada G

Normalization

- Esta fórmula utiliza los dos hechos siguientes: (i) si X es una variable aleatoria con una función de distribución acumulativa continua F, entonces F (X) se distribuye uniformemente en [0,1] (ii) si U es una variable aleatoria con distribución en [0,1] entonces G ^ {- 1} (U) tiene distribución G
- Al realizar una transformación de rango, una transformación de cuantiles suaviza distribuciones inusuales y está menos influenciada por valores atípicos que los métodos de escala. Sin embargo, distorsiona las correlaciones y distancias dentro y entre entidades.
- `QuantileTransformer` proporciona una transformación no paramétrica para asignar los datos a una distribución uniforme con valores entre 0 y 1:

Normalization

- Esta fórmula utiliza los dos hechos siguientes: (i) si X es una variable aleatoria con una función de distribución acumulativa continua F, entonces F (X) se distribuye uniformemente en [0,1] (ii) si U es una variable aleatoria con distribución en [0,1] entonces G ^ {- 1} (U) tiene distribución G
- Al realizar una transformación de rango, una transformación de cuantiles suaviza distribuciones inusuales y está menos influenciada por valores atípicos que los métodos de escala. Sin embargo, distorsiona las correlaciones y distancias dentro y entre entidades.
- `QuantileTransformer` proporciona una transformación no paramétrica para asignar los datos a una distribución uniforme con valores entre 0 y 1:

Normalization

- Esta fórmula utiliza los dos hechos siguientes: (i) si X es una variable aleatoria con una función de distribución acumulativa continua F, entonces F (X) se distribuye uniformemente en [0,1] (ii) si U es una variable aleatoria con distribución en [0,1] entonces G ^ {- 1} (U) tiene distribución G
- Al realizar una transformación de rango, una transformación de cuantiles suaviza distribuciones inusuales y está menos influenciada por valores atípicos que los métodos de escala. Sin embargo, distorsiona las correlaciones y distancias dentro y entre entidades.
- `QuantileTransformer` proporciona una transformación no paramétrica para asignar los datos a una distribución uniforme con valores entre 0 y 1:

Power transformations

- Las transformaciones de potencia son una familia de transformaciones paramétricas que tienen como objetivo mapear datos de cualquier distribución lo más cerca posible de una distribución gaussiana.
- La transformación de Yeo-Johnson y la de Box-Cox son

$$x_i^{(\lambda)} = \begin{cases} [(x_i+1)^\lambda - 1]/\lambda & \text{if } \lambda \neq 0, x_i \geq 0, \\ \ln(x_i+1) & \text{if } \lambda = 0, x_i \geq 0 \\ -[(-x_i+1)^{2-\lambda} - 1]/(2-\lambda) & \text{if } \lambda \neq 2, x_i < 0, \\ -\ln(-x_i+1) & \text{if } \lambda = 2, x_i < 0 \end{cases}$$

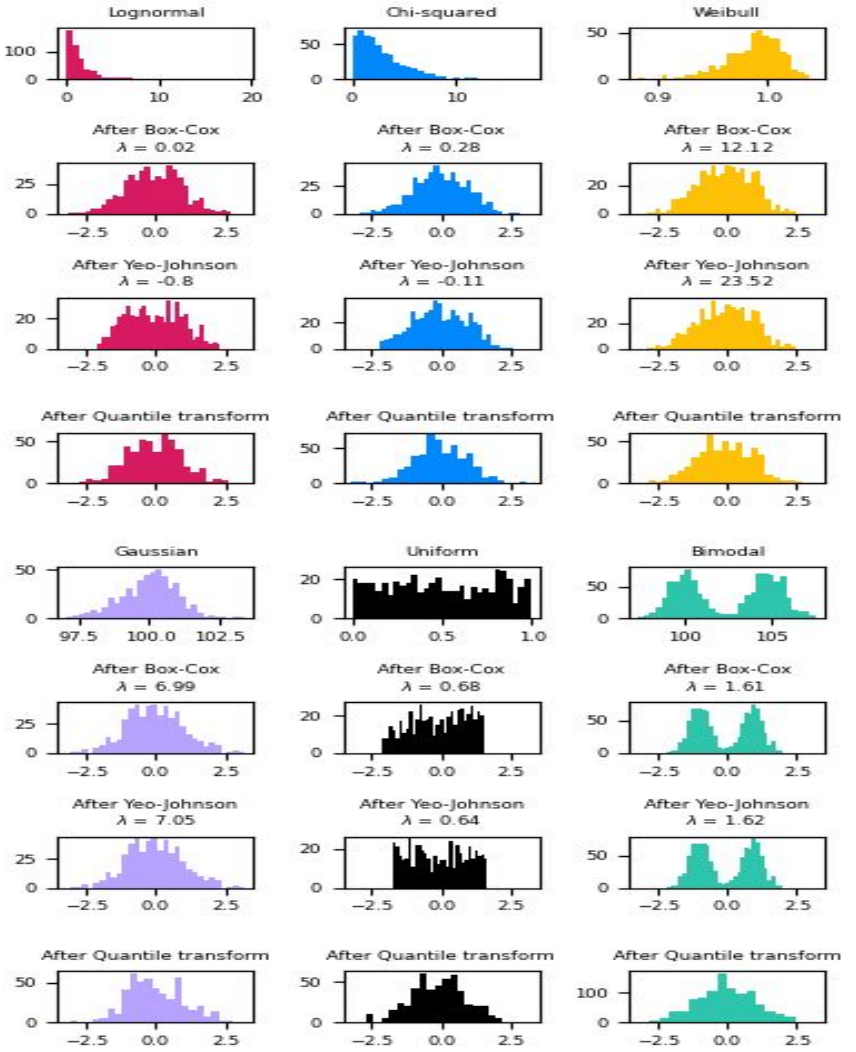
$$x_i^{(\lambda)} = \begin{cases} \dfrac{x_i^\lambda - 1}{\lambda} & \text{if } \lambda \neq 0, \\ \ln(x_i) & \text{if } \lambda = 0, \end{cases}$$

Power transformations

```
bc = PowerTransformer(method='box-cox')

yj = PowerTransformer(method='yeo-johnson')

qr = QuantileTransformer(output_distribution='normal'
, random_state=0)
```



Escalar vs. Normalizar

Veamos algunos gráficos!

https://www.kaggle.com/code/alexisbcook/scaling-and-normalization

Normalization (llevar a norma unitaria)

- La normalización es el proceso de escalar muestras individuales para tener una norma unitaria.
- Este proceso puede ser útil si planea utilizar una forma cuadrática como el producto punto o cualquier otro núcleo para cuantificar la similitud de cualquier par de muestras.
- La función `normalize` proporciona una manera rápida y fácil de realizar esta operación en un único conjunto de datos similar a una matriz, ya sea usando las normas l1, l2 o max

Normalization (llevar a norma unitaria)

- La normalización es el proceso de escalar muestras individuales para tener una norma unitaria.
- Este proceso puede ser útil si planea utilizar una forma cuadrática como el producto punto o cualquier otro núcleo para cuantificar la similitud de cualquier par de muestras.
- La función `normalize` proporciona una manera rápida y fácil de realizar esta operación en un único conjunto de datos similar a una matriz, ya sea usando las normas l1, l2 o max

Normalization (llevar a norma unitaria)

- La normalización es el proceso de escalar muestras individuales para tener una norma unitaria.
- Este proceso puede ser útil si planea utilizar una forma cuadrática como el producto punto o cualquier otro núcleo para cuantificar la similitud de cualquier par de muestras.
- La función `normalize` proporciona una manera rápida y fácil de realizar esta operación en un único conjunto de datos similar a una matriz, ya sea usando las normas l1, l2 o max