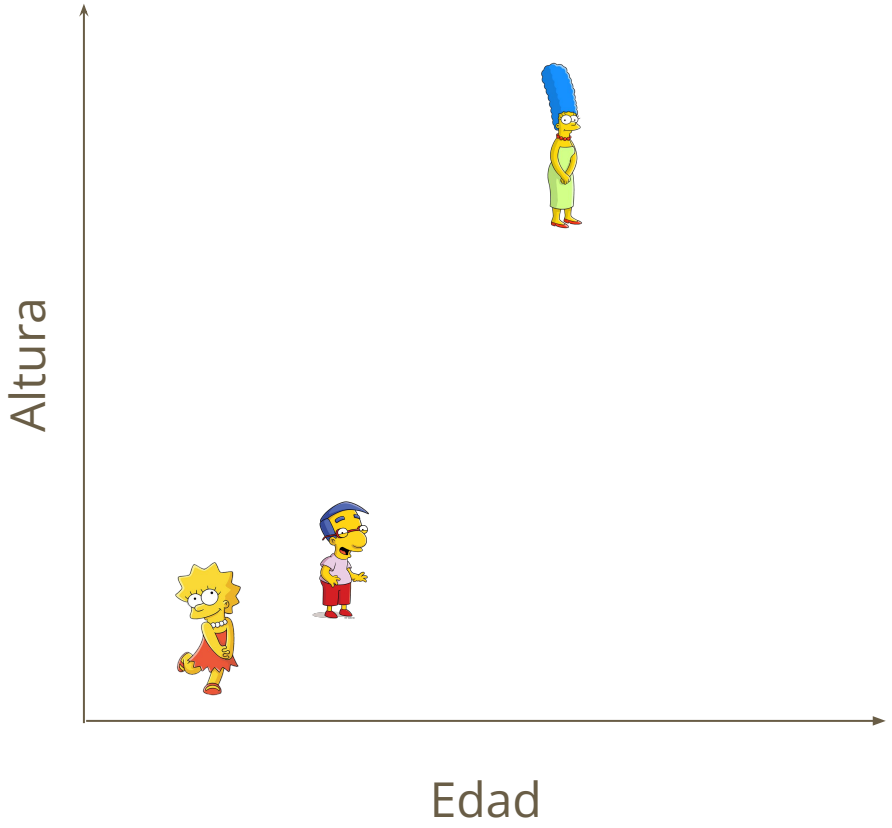
Vecinos más cercanos

Lisa

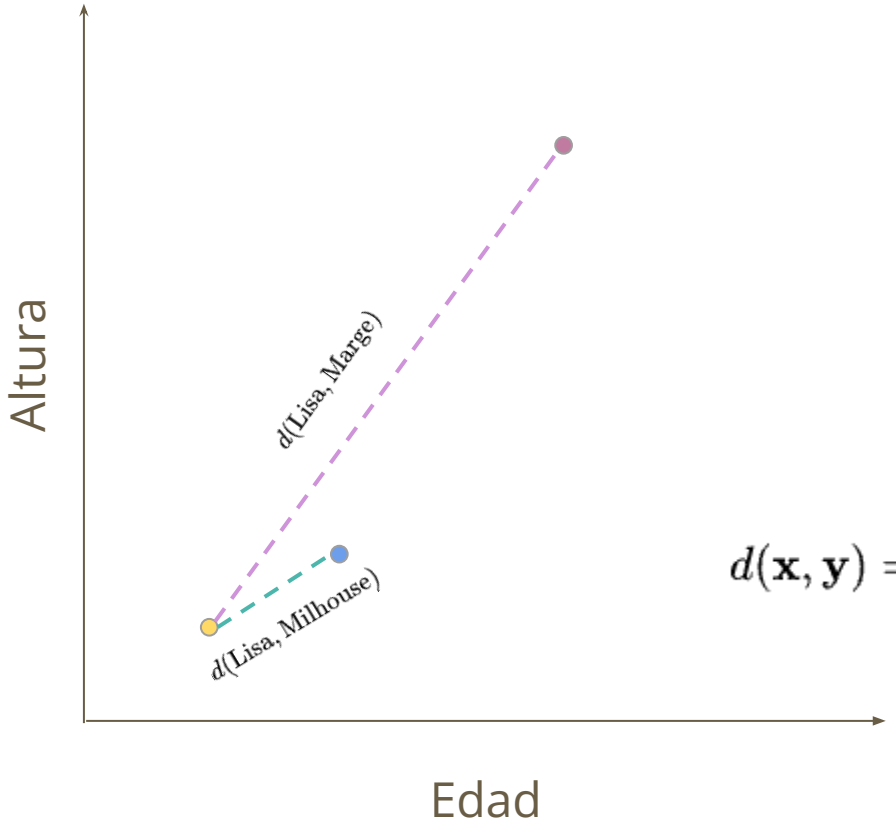
Milhouse

Marge

¿Quién se parece más a Lisa?



Nombre	Edad (años)	Altura (cm)
Lisa	8	132
Marge	36	190
Milhouse	10	135



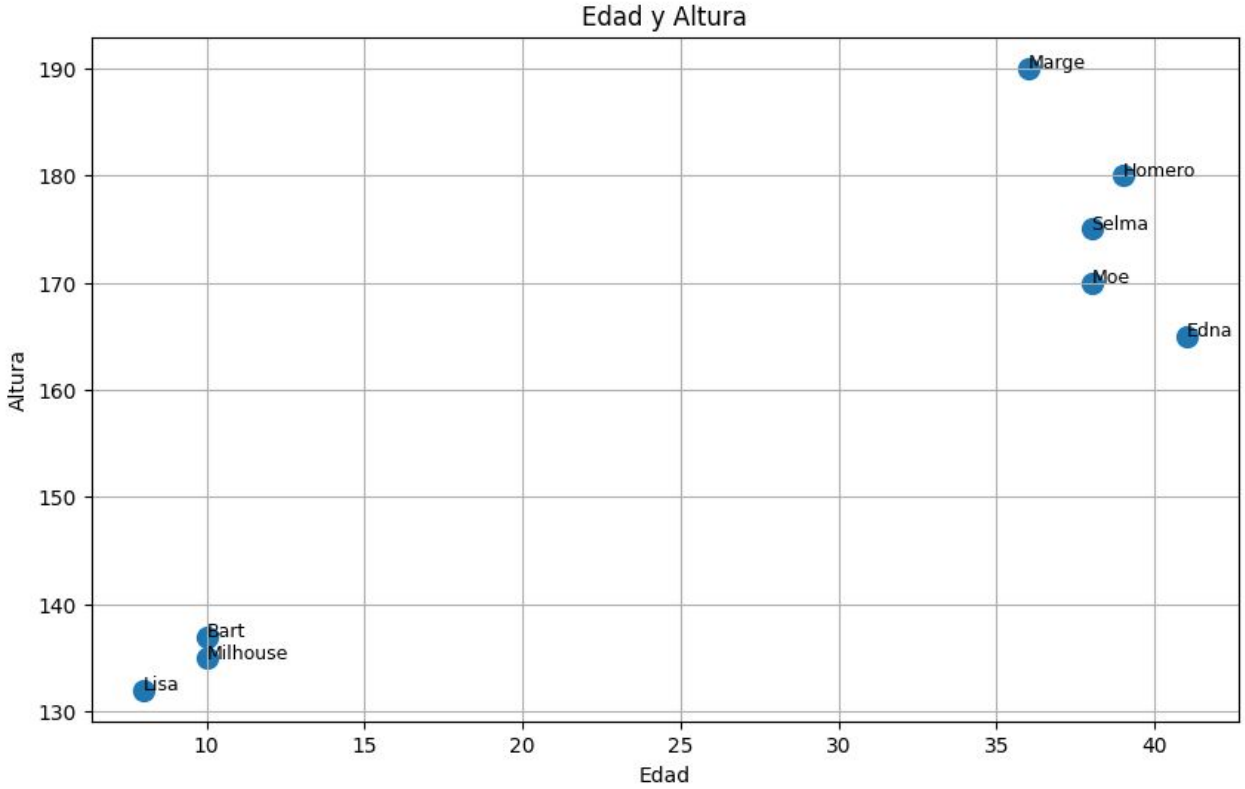
Nombre	Edad (años)	Altura (cm)
Lisa	8	132
Marge	36	190
Milhouse	10	135

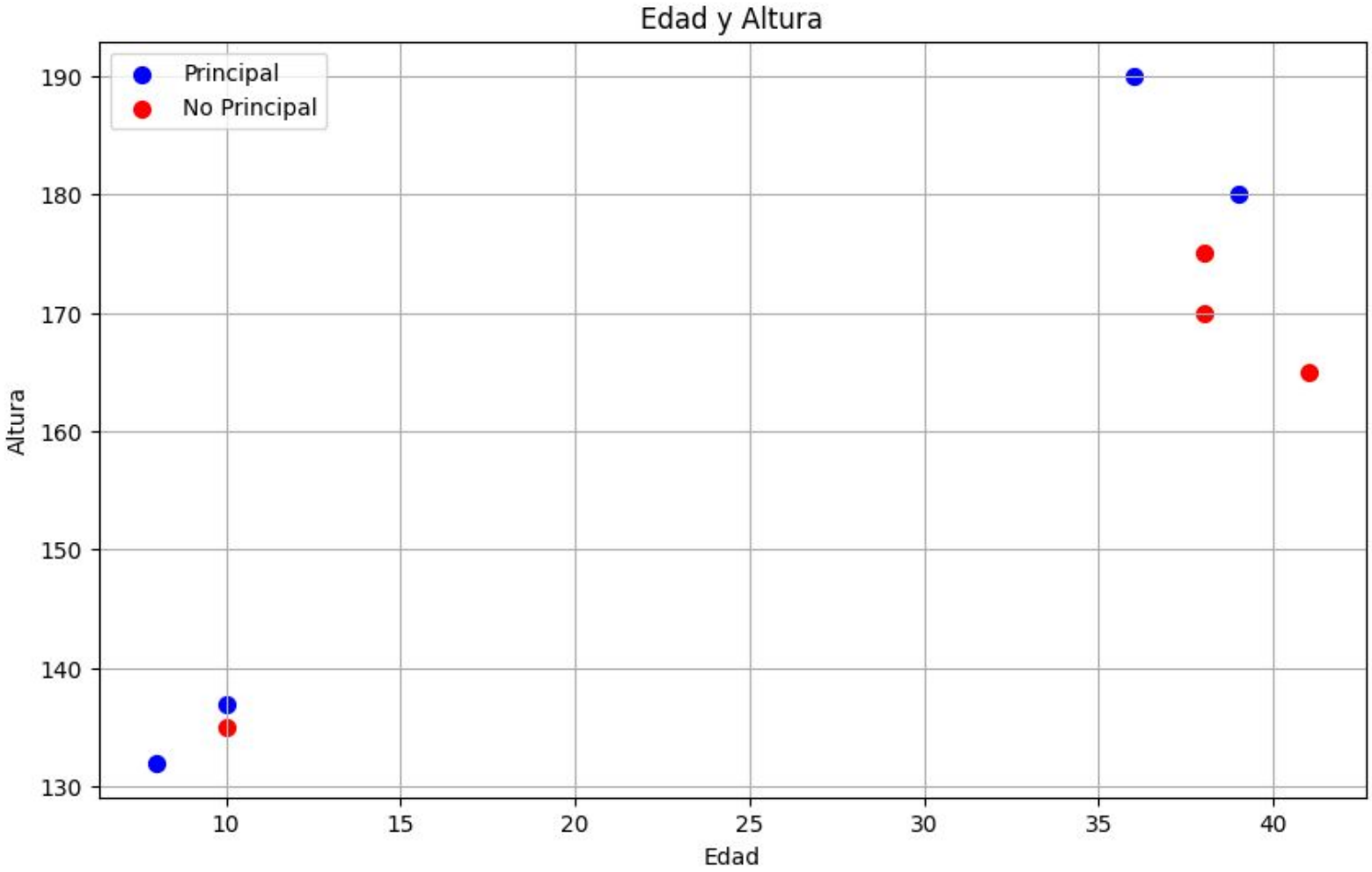
$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{(\text{Edad}_x - \text{Edad}_y)^2 + (\text{Altura}_x - \text{Altura}_y)^2}$$

$$d(\text{Lisa}, \text{Marge}) = 64.4$$

$$d(\text{Lisa}, \text{Milhouse}) = 3.61$$

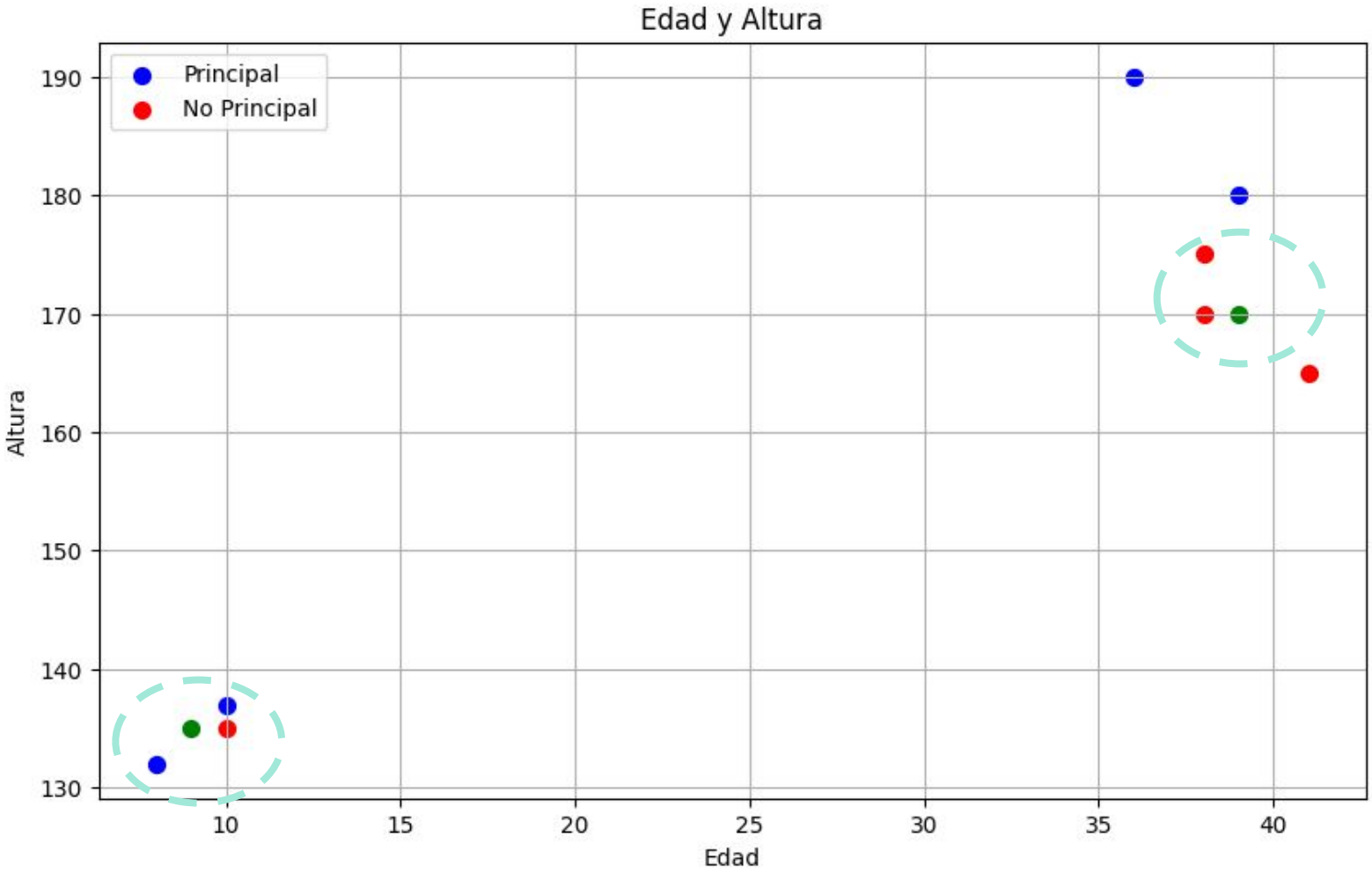

Edad y Altura
Altura
Edad
190
180
170
160
150
140
130
10
15
20
25
30
35
40
Marge
Homero
Selma
Moe
Edna
Bart
Milhouse
Lisa


Edad y Altura
Principal
No Principal
Altura
Edad
190
180
170
160
150
140
130
10
15
20
25
30
35
40

Busco un restaurante para ir a conocer con mi esposa:

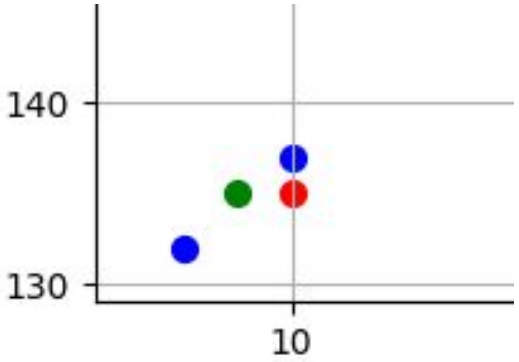
- Le gusta la comida asiática
- No le gusta el buffet

Vecinos más cercanos


Edad y Altura
Principal
No Principal
Altura
Edad
190
180
170
160
150
140
130
10
15
20
25
30
35
40

KNN - K-Nearest-Neighbours

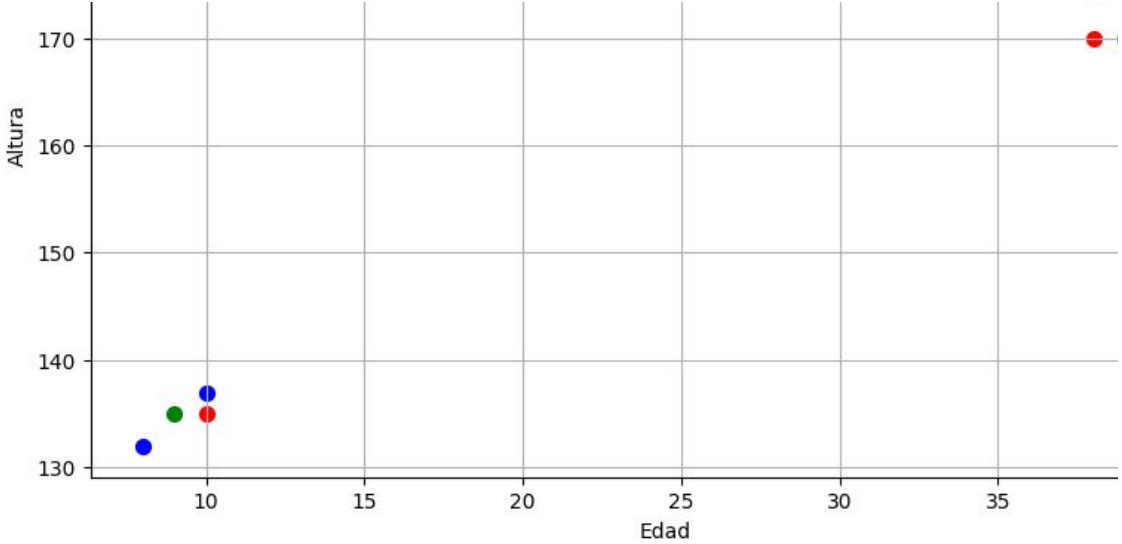
- Es lazy: no construyo un modelo al entrenar, sino que guardo los datos
- Cuando llega una instancia nueva, busco cuales son las K más cercanas
- Retorno la etiqueta mayoritaria en esas K instancias
- Escala a más cantidad de clases
- K es un hiper parámetro que debo elegir:



K=1 -> rojo
K=3 -> azul
k=4 -> ?

KNN - K-Nearest-Neighbours

Veamos qué pasa con K = 4:



- puedo ponderar la votación según la distancia
- puedo definir el vecindario según distancia

KNN - K-Nearest-Neighbours

Vecindario según distancia:

- cómo elijo la distancia?
- Qué pasa si no hay vecinos?

Nombre	Edad (años)	Altura (cm)
Lisa	8	132
Marge	36	190
Milhouse	10	135

Nombre	Edad (años)	Altura (m)
Lisa	8	1.32
Marge	36	1.90
Milhouse	10	1.35

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{(\text{Edad}_x - \text{Edad}_y)^2 + (\text{Altura}_x - \text{Altura}_y)^2}$$

x21

$$d(\text{Lisa}, \text{Marge}) = 64.4$$
$$d(\text{Lisa}, \text{Milhouse}) = 3.61$$

$$d(\text{Lisa}, \text{Marge}) = 28.0$$
$$d(\text{Lisa}, \text{Milhouse}) = 2.0$$

x14

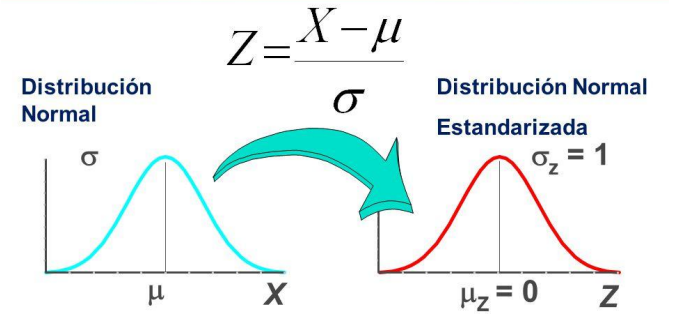
Es fundamental normalizar atributos

Para evitar que uno domine sobre los demás sólo porque esta en otra escala

MinMaxScaler: llevar cada feature al rango (0,1)

$$x_{\text{norm}} = \frac{x - x_{\min}}{x_{\max} - x_{\min}}$$

StandardScaler: llevar cada feature a media 0 y std 1



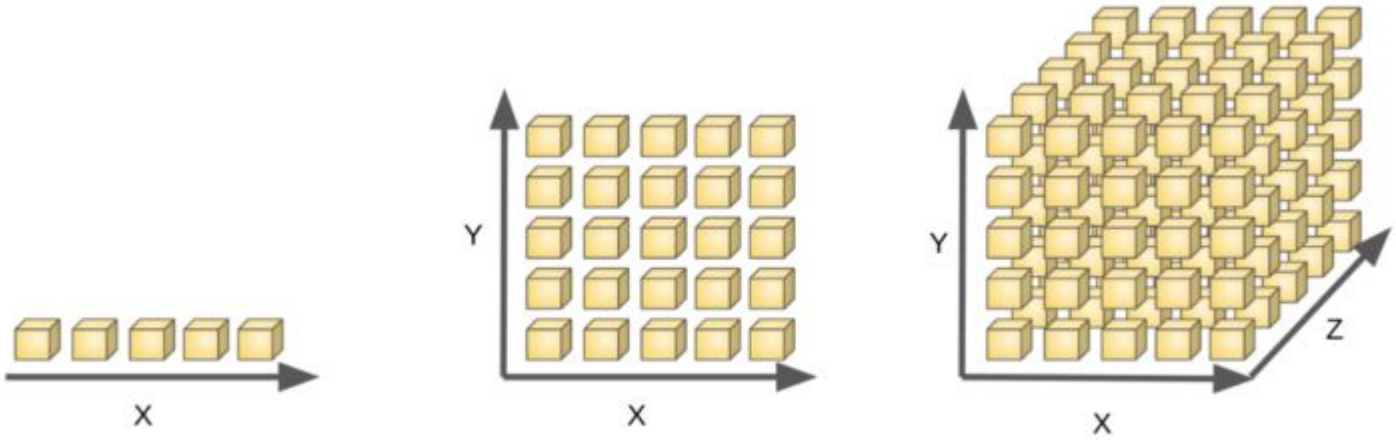
Más es mejor?

Hasta ahora venimos con 2 features, que no tienen mucha relación con el target.

En el práctico construimos más de 40.

Es mejor tener más features?

Maldición de la dimensionalidad



Duda, Hart: en formulaciones “clasicas” $N \geq 5 \cdot d$ o idealmente $N \geq 10 \cdot d$

Feature selection

Debo elegir las mejores:

- Las que me den más información respecto al target
- Evitar tener features redundantes
- Features de poca varianza contienen poca información

Estrategias:

- Ver correlaciones, entre si y con el target; ver varianzas
- Entrenar algún modelo y ver el feature_importance
- Recursive Feature Elimination
- Sequential Feature Elimination (backward/forward)

Feature selection: RFE

Recursive Feature Elimination

1. Entreno con todas las features
2. Las ordeno por importancia
3. Elimino la menos importante. Vuelvo a 1.

Necesito algo que les asigne importancia a las features

Puedo graficar como cambia la performance en la medida que quito features

Feature selection: Sequential Feature Elimination

En cada paso tengo N features disponibles, entreno N modelos (uno por cada feature que voy a remover)

Luego elimino la feature que cambió menos la preformance.

Repito ahora con las N-1 features restantes

Opcional: puedo partir sin features e ir agregando.

Tengo 100 features, quiero 10 vs tengo 100 quiero 90

No necesita ningun estimador para la calidad de las features

Es mas caro

Reducción de dimensiones: PCA

Principal Component Analysis

Método de reducción de dimensionalidad

Sirve para visualización

Técnica exploratoria

La veremos luego en profundidad

PCA

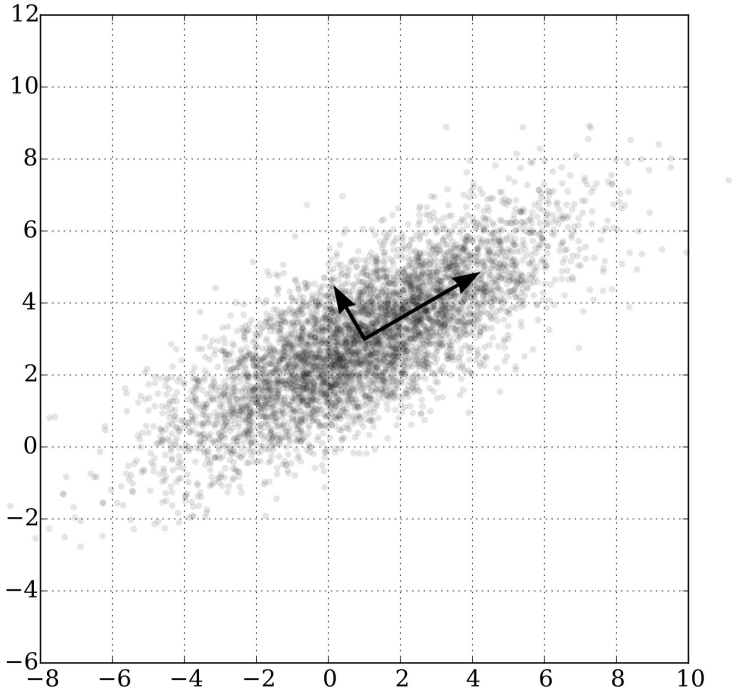
El objetivo es encontrar un conjunto de ejes sobre los cuales los datos tengan la mayor varianza

De esta forma al proyectar los datos quedan “separados” lo más posible

Los datos pasan a una nueva representación de menor dimensionalidad con features independientes

Es una transformación lineal

PCA



KNN: la distancia a utilizar

Es un hiper parámetro clave

Modificar la distancia puede modificar todos los resultados

Elegir la correcta depende de nuestro problema

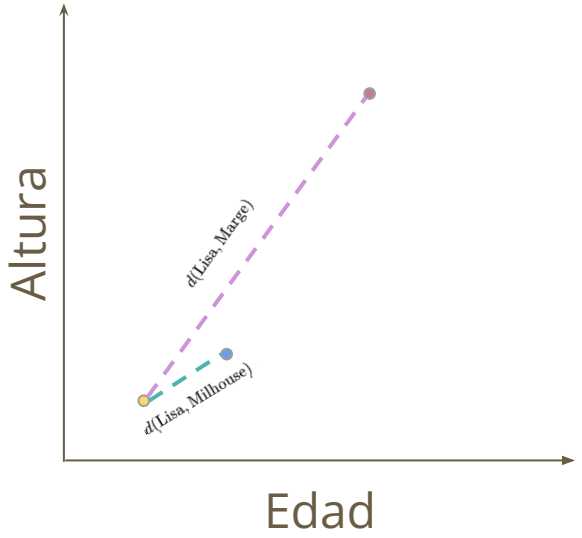
Distancia a utilizar: Euclídea (L2)

La más común.

Sensible a la escala de los atributos.

Representa la “distancia en línea recta”.

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$



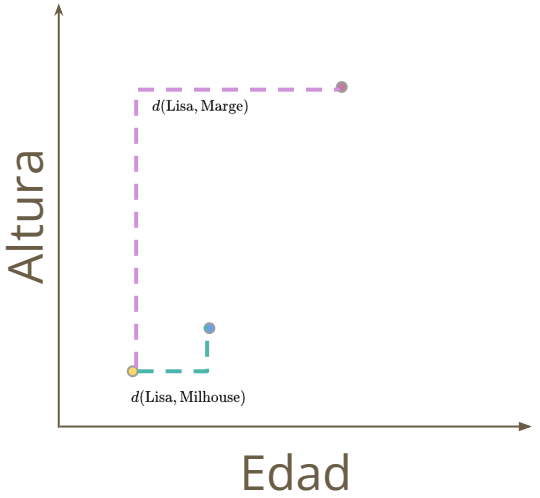
Distancia a utilizar: Manhattan (L1)

Mide la distancia como en una grilla de calles.

Más robusta a outliers.

Puede funcionar mejor cuando los datos son dispersos o poco densos.

$$d(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} |x_i - y_i|$$



Distancia a utilizar: Coseno

Mide el ángulo entre los vectores.

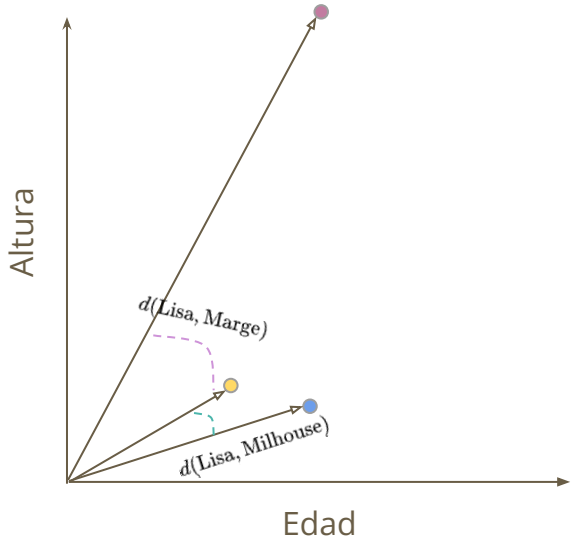
Ignora la magnitud: se enfoca en la dirección.

Muy útil en datos de texto, recomendaciones o alta dimensión.

Acotada: 0<= distancia <= 2

$$\text{similaridad} = \cos(\theta) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\|\|\mathbf{y}\|}$$

$$\text{distancia} = 1 - \cos(\theta)$$



Los puntos cambiaron un poco de lugar para facilitar visualización

Otras aplicaciones de KNN

- **Regresión:**

  En lugar de predecir la clase más votada, se predice el valor promedio entre los vecinos más cercanos.

- **Probabilidades:**

  Interpretando el vecindario como una estimación local de la densidad de probabilidad, se puede retornar la frecuencia relativa de cada clase como una estimación de su probabilidad.

$$P(c \mid \vec{x}) = \frac{\#\text{ vecinos de la clase } c}{\#\text{ total de vecinos}}$$

ANN

En problemas con grandes volúmenes de datos o alta dimensionalidad, las búsquedas exactas pueden volverse muy lentas.

Para ganar eficiencia, se utilizan métodos aproximados: sacrifican algo de recall a cambio de mayor velocidad.

Usan estructuras especializadas (como grafos, árboles o particiones del espacio) para acelerar la búsqueda.

Ejemplos de frameworks: FAISS, Annoy, HNSW