Laboratorio de datos, clase 12

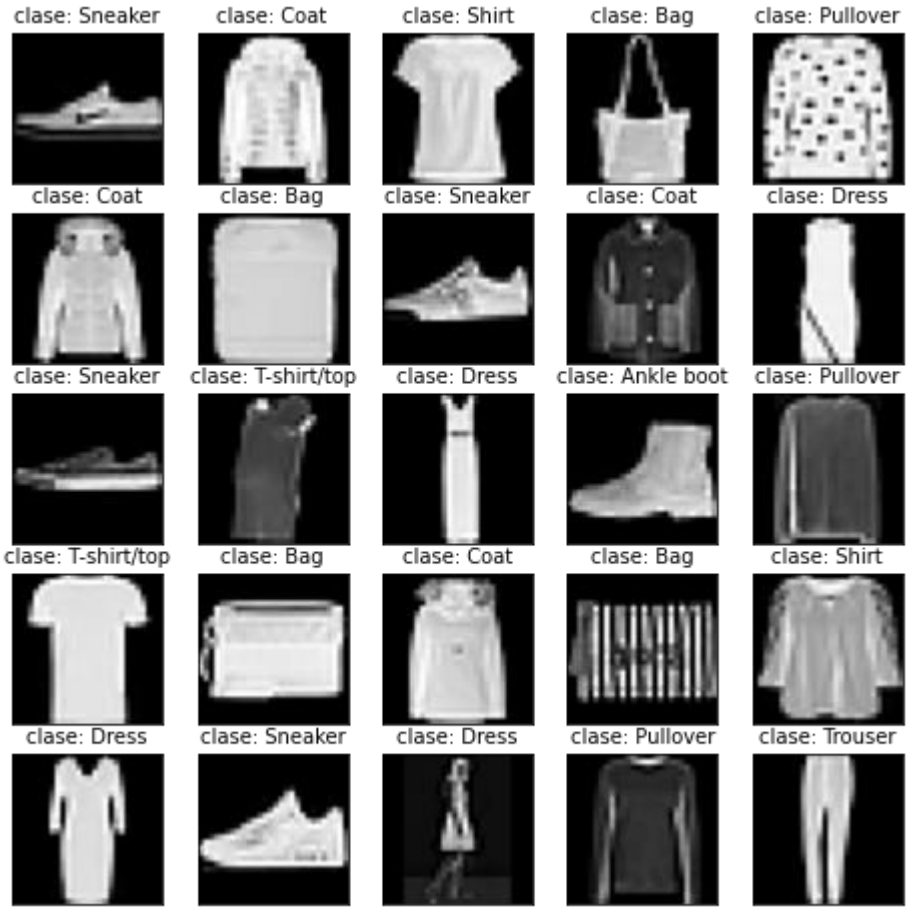
Clasificación y regresión basada en instancias (KNN)

Prof. Enzo Tagliazucchi

tagliazucchi.enzo@googlemail.com
www.cocucolab.org

En la clase pasada vimos...

Dataset “fashion mnist”

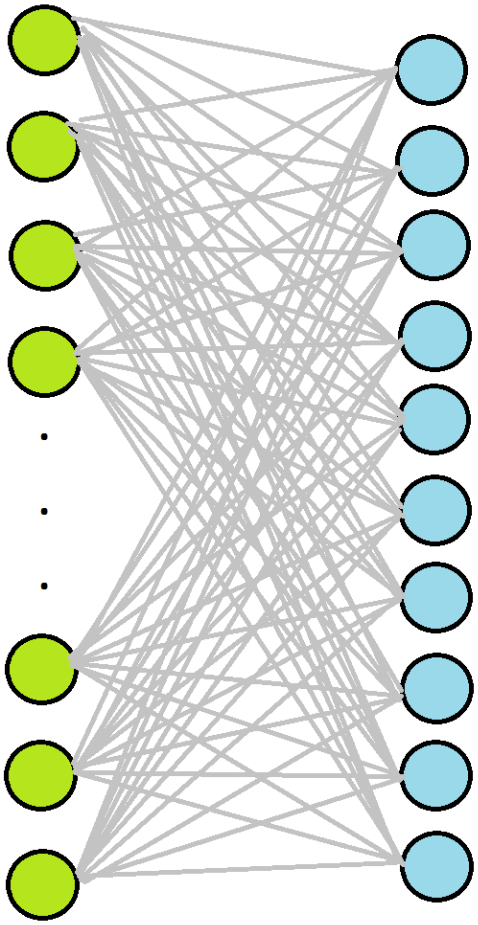


Label	Description
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

En la clase pasada vimos...

Clasificador lineal

Accuracy (test) = 0.8424



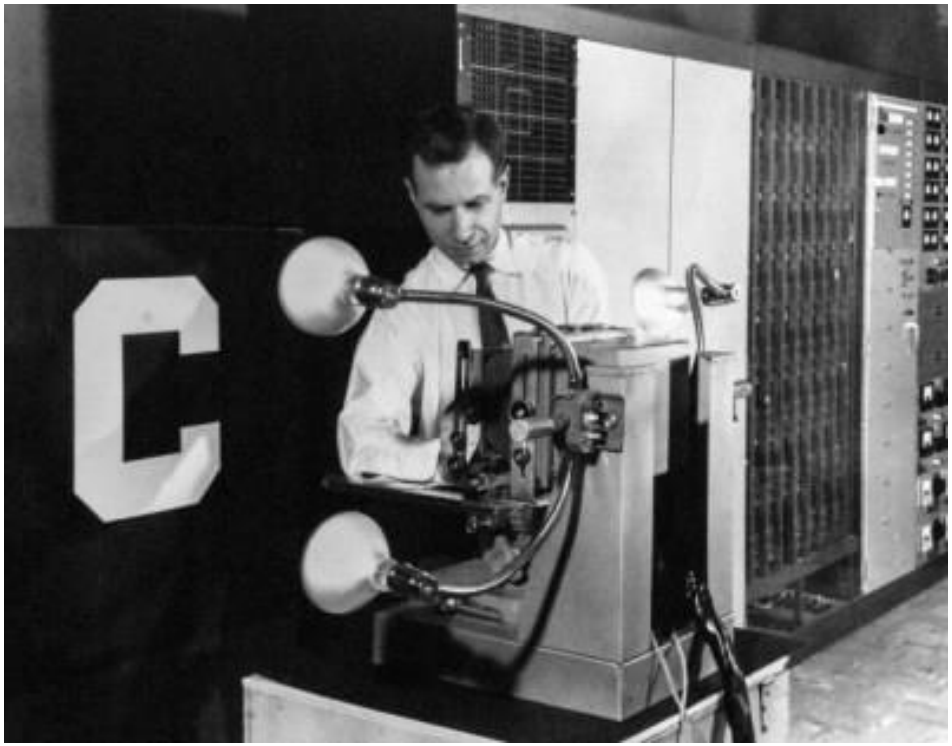
Unidades de entrada
(una por cada pixel en la imagen, total=784)

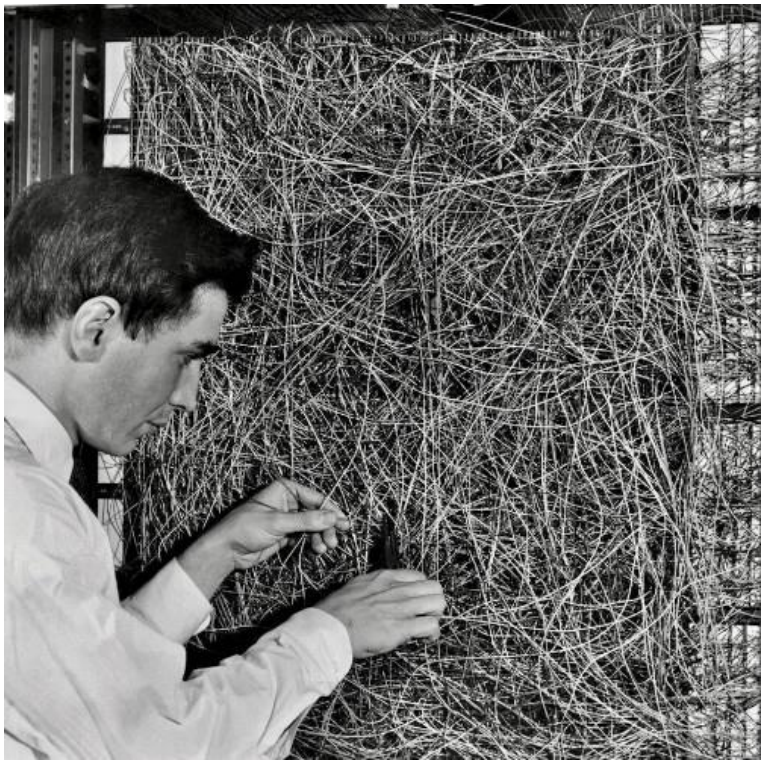
Unidades de salida
(una por cada categoría, total=10)
Función softmax: si x_i y z_i son el input y output de la i-ésima unidad de salida, entonces:

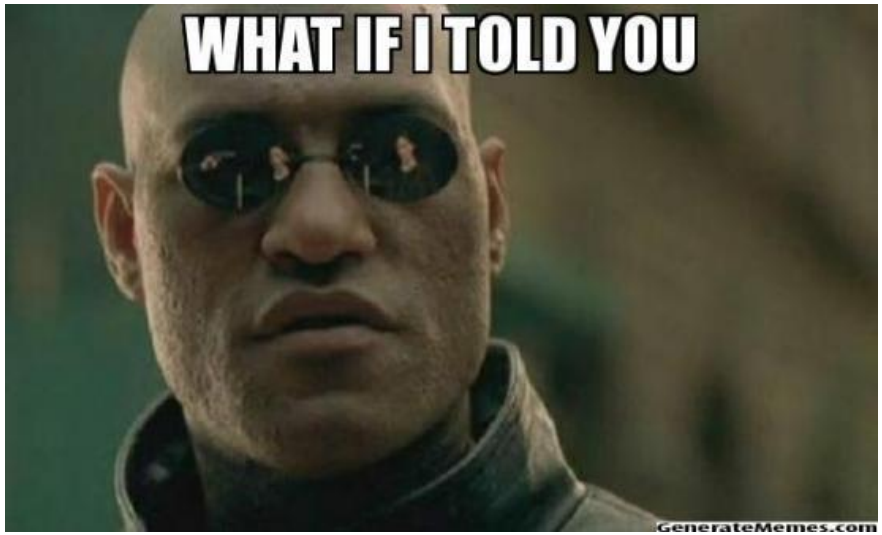
$$z_i = \frac{e^{z_i}}{\sum_{i=1}^{10} e^{z_i}}$$

Parámetros a estimar (total=7840)

Mark I perceptron (Rosenblatt, 1958)
(20 x 20)







Podemos igualar el accuracy del perceptron con un algoritmo sencillo, casi trivial

Este algoritmo ni siquiera se entrena

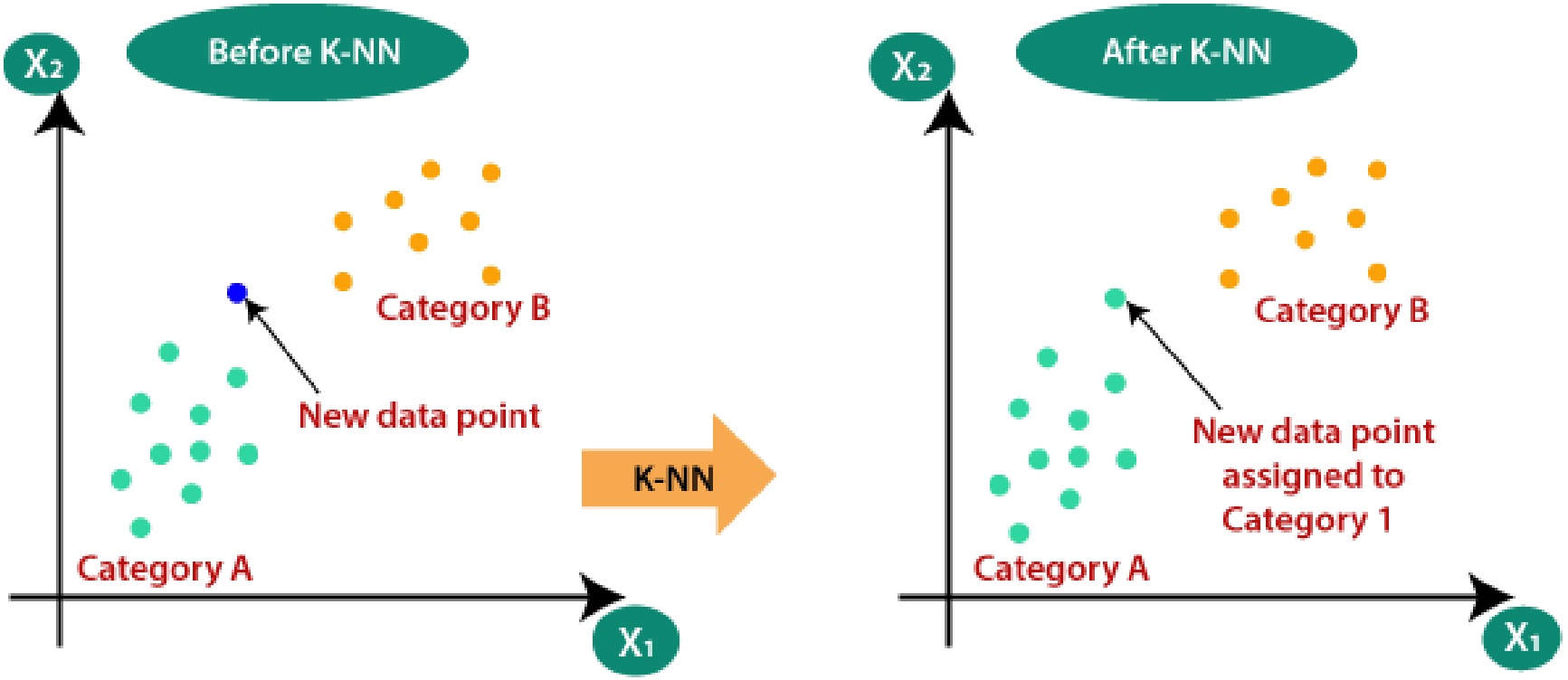
Como mucho, este algoritmo tiene un único hiperparámetro para ajustar

El algortimo KNN (básico)

Tenemos un conjunto de datos y etiquetas (matrices **X**, **y**).

Queremos predecir la etiqueta de un nuevo sample, **x**

1. Calculamos la proximidad del sample **x** a todos los que están en **X** (usando alguna función de distancia para la proximidad, típicamente norma 1, 2, infinito)
2. Seleccionamos los K que estén más próximos a **X** (K es hiperparámetro)
3. Nos fijamos la etiqueta que tienen esos K y predecimos que **x** tiene la etiqueta más frecuente



Observaciones:

1. Puede usarse para predecir la probabilidad de cada clase (cantidad de veces que cada etiqueta aparece entre los K más cercanos, dividido K)
2. Se puede usar para problemas multiclase
3. Se puede usar para regresión, por ejemplo, devuelvo el promedio de los valores asociados a los K más cercanos

Ventajas de KNN:

1. Fácil de usar e interpretar (podemos entender por qué obtuvimos la predicción que obtuvimos)
2. Depende de un único hiperparámetro
3. Multiclase, devuelve probabilidades
4. Sirve para regresión
5. Es robusto al ruido en los features
6. No paramétrico
7. Entrenamiento súper rápido
8. Útil para sistemas de recomendaciones
9. Buena performance

Desventajas de KNN:

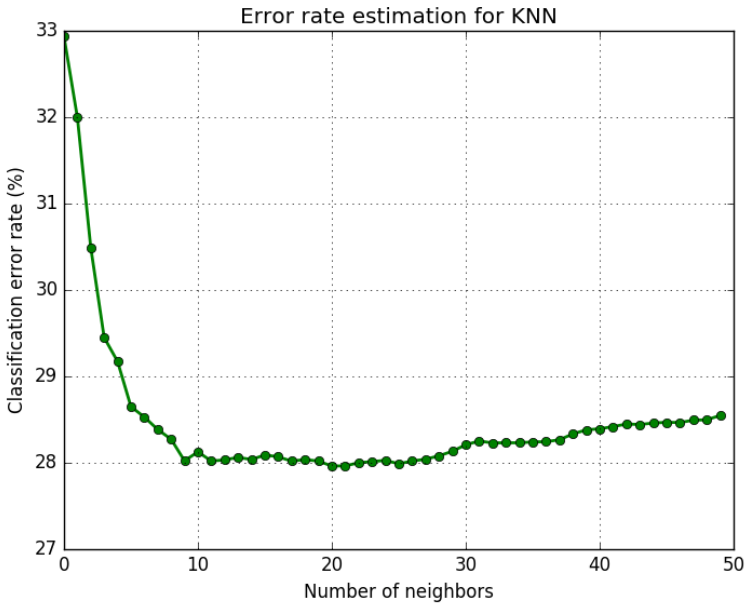
1. Es lo que se llama un “lazy learner”: no se estiman los parámetros de una $f(x, \beta)$ que pueda ser aplicada rápidamente a nuevos datos; cada nueva predicción necesita potencialmente todos los datos.
2. Por el ítem anterior, se vuelve lento a la hora de predecir muchos labels
3. Como todos los clasificadores que dependen de distancias, es sensible a disparidades en la escala de los features
4. Puede tener requerimientos altos de memoria

¿Cómo elijo K?

K=1 implica predicciones poco estables y sensibles al ruido (“overfittear”)

K muy grande implica que empiezan a votar ejemplos que están muy lejos del dato cuya etiqueta quiero predecir, y la performance empeora

En el medio, tiene que haber algún K óptimo



¿Qué tan lento es predecir una nueva etiqueta?

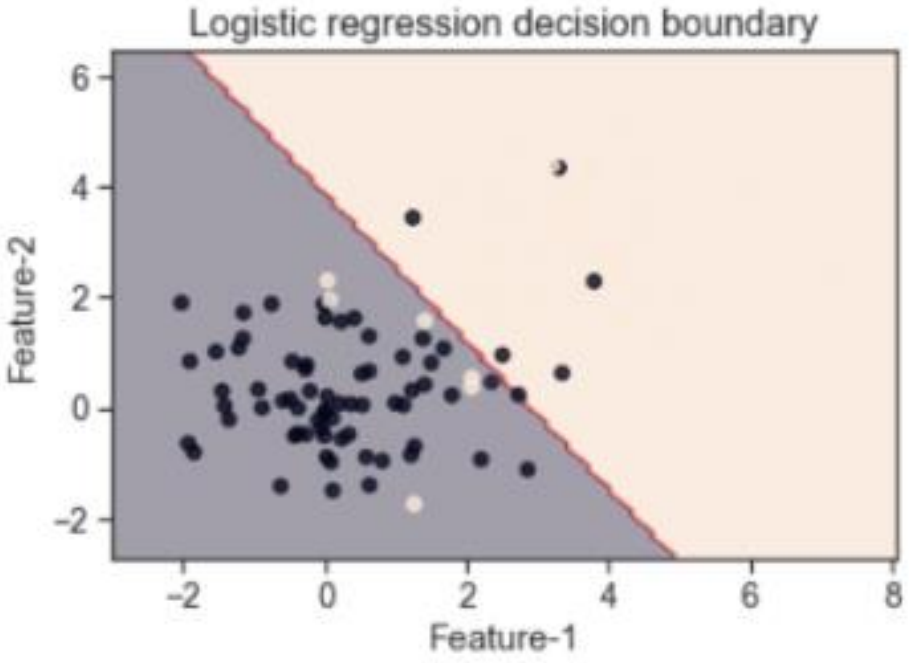
Depende del algoritmo que uso para calcular la distancias.

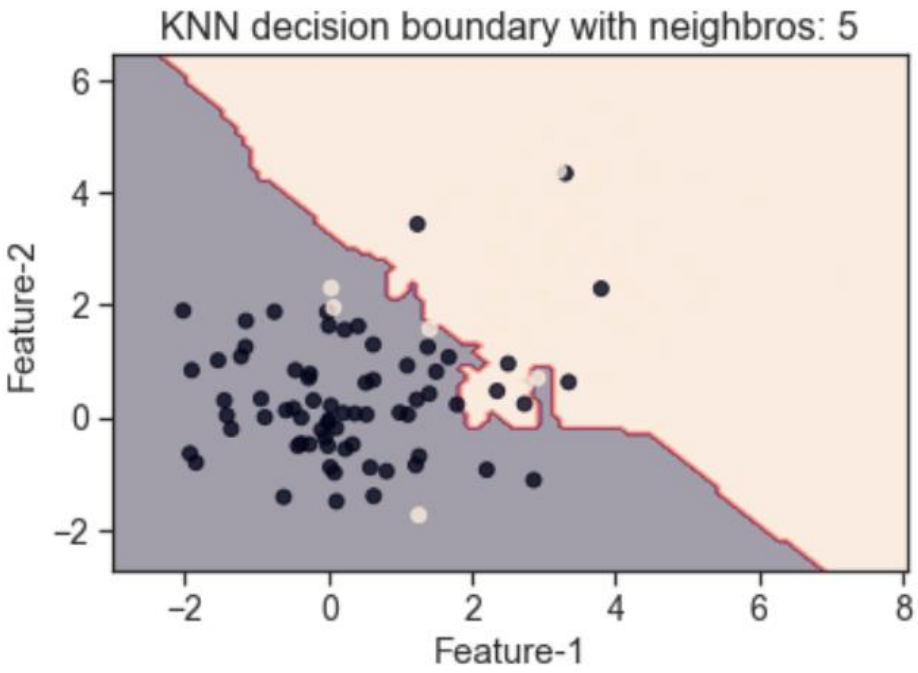
Fuerza bruta (distancia contra todos):
O(ND), donde N es la cantidad de ejemplos de entrenamiento y D la cantidad de dimensiones (independiente de K).
Recomendable solo si N es pequeño.

Otros más sofisticados (KDtree, Balltree) pueden ser más eficientes si hay muchos datos o dimensiones.

Idea: los datos pueden vivir en una variedad de dimensión menor a D.

Por qué KNN es *no paramétrico*:





sklearn.neighbors.KNeighborsClassifier¶

class sklearn.neighbors.**KNeighborsClassifier**(*n_neighbors=5, *, weights='uniform', algorithm='auto', leaf_size=30, p=2, metric='minkowski', metric_params=None, n_jobs=None, **kwargs*) [source]

Classifier implementing the k-nearest neighbors vote.

Parámetros:

n_neighbours: el K de KNN

weights: 'uniform' si el peso de los votos es el mismo para todos los K, 'distance' si el peso del voto decae con la distancia.

algorithm: 'auto', 'brute', 'kd_tree', 'ball_tree'

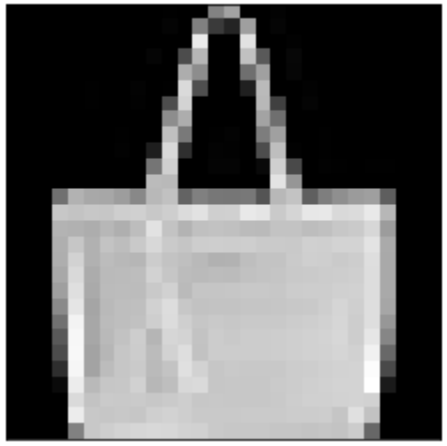
leaf_size: parámetro relevante para 'kd_tree' y 'ball_tree'

p: potencia para la métrica de Minkowski. p=2 resulta en distancia Euclidea

metric_params: si quiero probar otra métrica

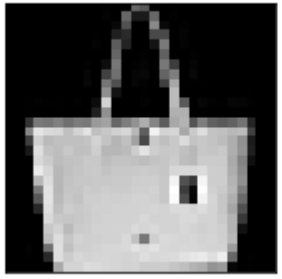
Por qué tengo la etiqueta que tengo:

Nuevo sample, clase: Bag



Training set, clase: Bag

Training set, clase: Bag



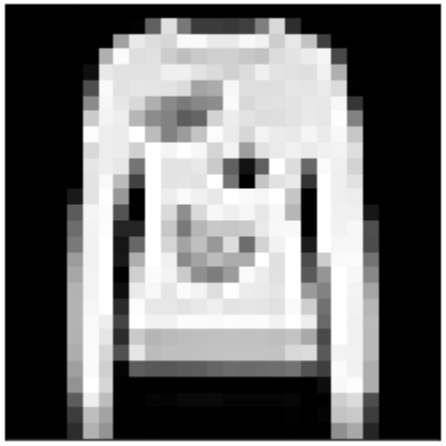
Training set, clase: Bag

Training set, clase: Bag

Training set, clase: Bag

Por qué tengo la etiqueta que tengo:

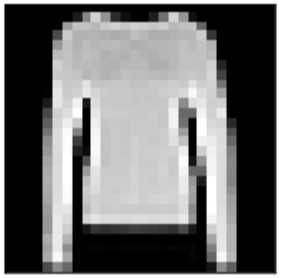
Nuevo sample, clase: Pullover



Training set, clase: Pullover

Training set, clase: Pullover

Training set, clase: Pullover

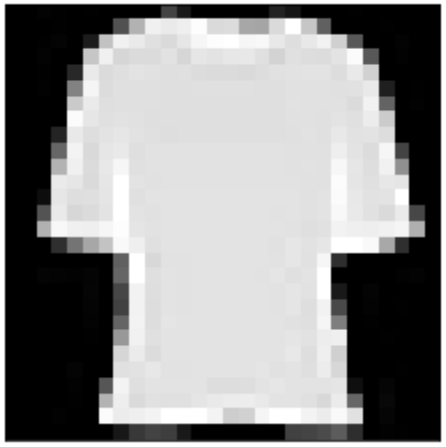


Training set, clase: Pullover

Training set, clase: Shirt

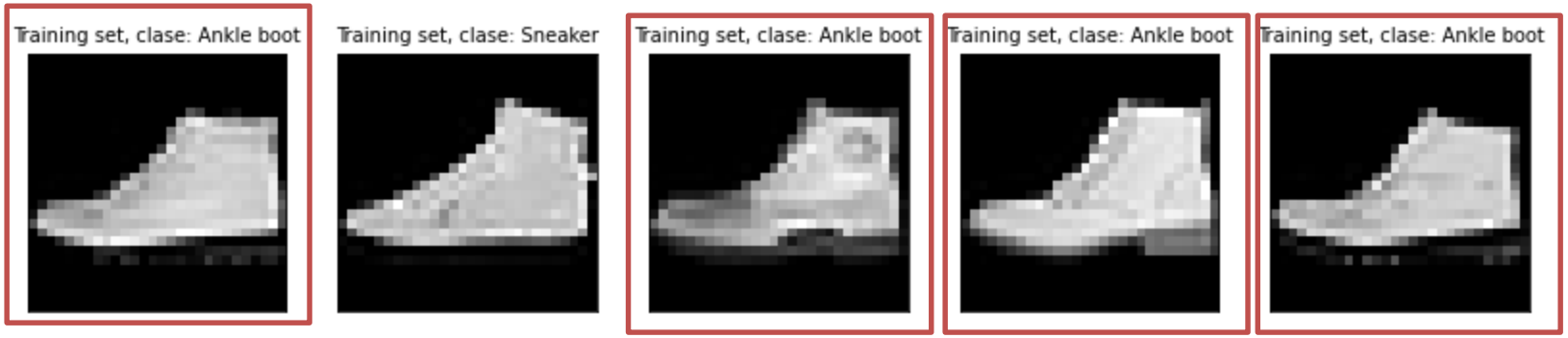
¿Por qué me equivoco?

Nuevo sample, clase: Shirt



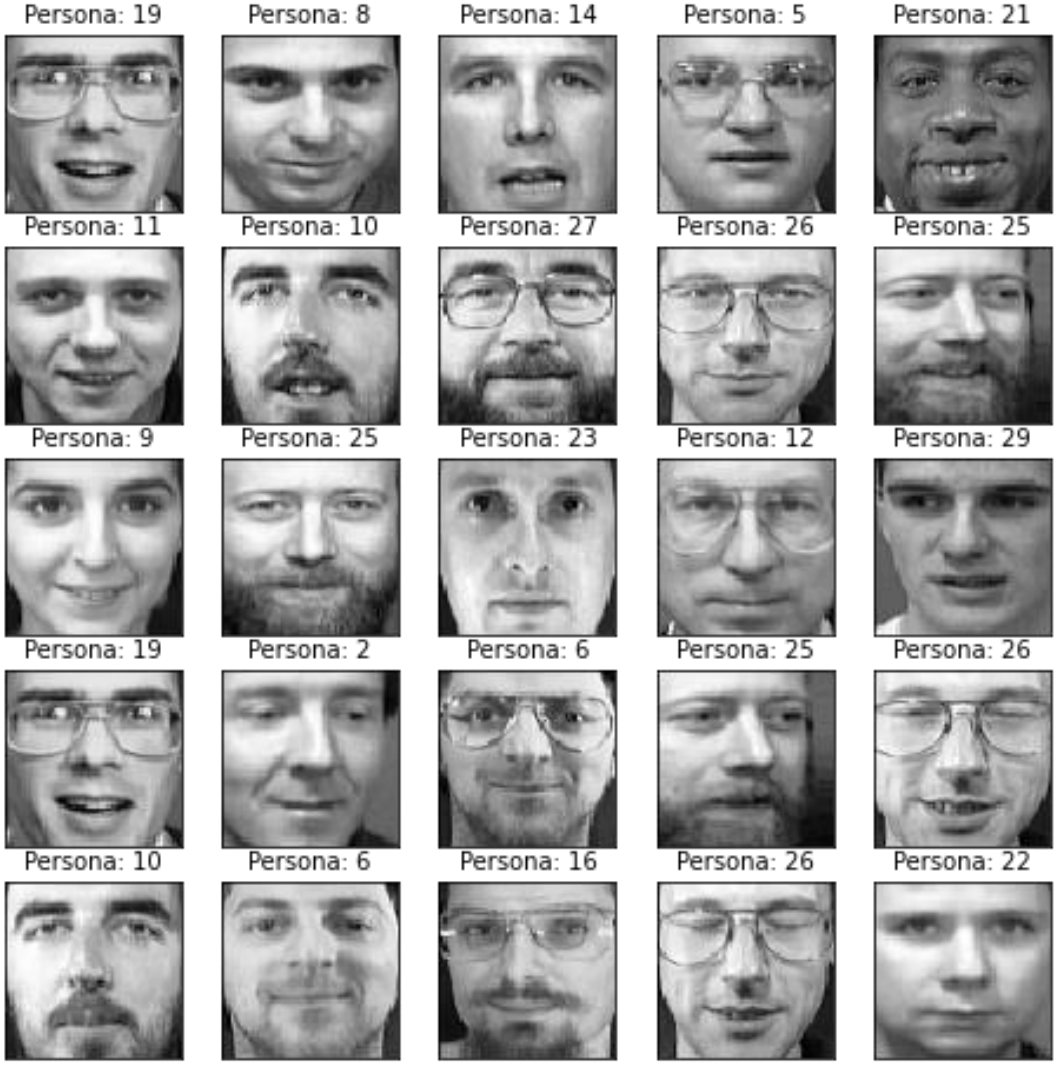
¿Por qué me equivoco?

Nuevo sample, clase: Sneaker



Regresión con KNN

Problema: completar caras


Persona: 19
Persona: 8
Persona: 14
Persona: 5
Persona: 21
Persona: 11
Persona: 10
Persona: 27
Persona: 26
Persona: 25
Persona: 9
Persona: 25
Persona: 23
Persona: 12
Persona: 29
Persona: 19
Persona: 2
Persona: 6
Persona: 25
Persona: 26
Persona: 10
Persona: 6
Persona: 16
Persona: 26
Persona: 22

Caras completas, originales

Caras completas (KNN)

Caras completas (regresión lineal)