PONTIFICIA
UNIVERSIDAD
CATÓLICA
DE CHILE

IIC2433 - Minería de datos

K-Nearest Neighbors

Ricardo Schilling
reschilling@uc.cl

K-Nearest Neighbors

- Puede usarse tanto para clasificación como para regresión fácilmente.
- Se basa en tomar los puntos más cercanos al dato de interés.
- No requiere gran poder computacional para entrenarse, pero si para realizar predicciones, dependiendo del dataset.
- Los mejores resultados se obtienen con datos normalizados.

Descripción del *dataset*

Si bien kNN permite datasets de múltiples dimensiones, usaremos uno de sólo 3 para ayudar a la visualización.

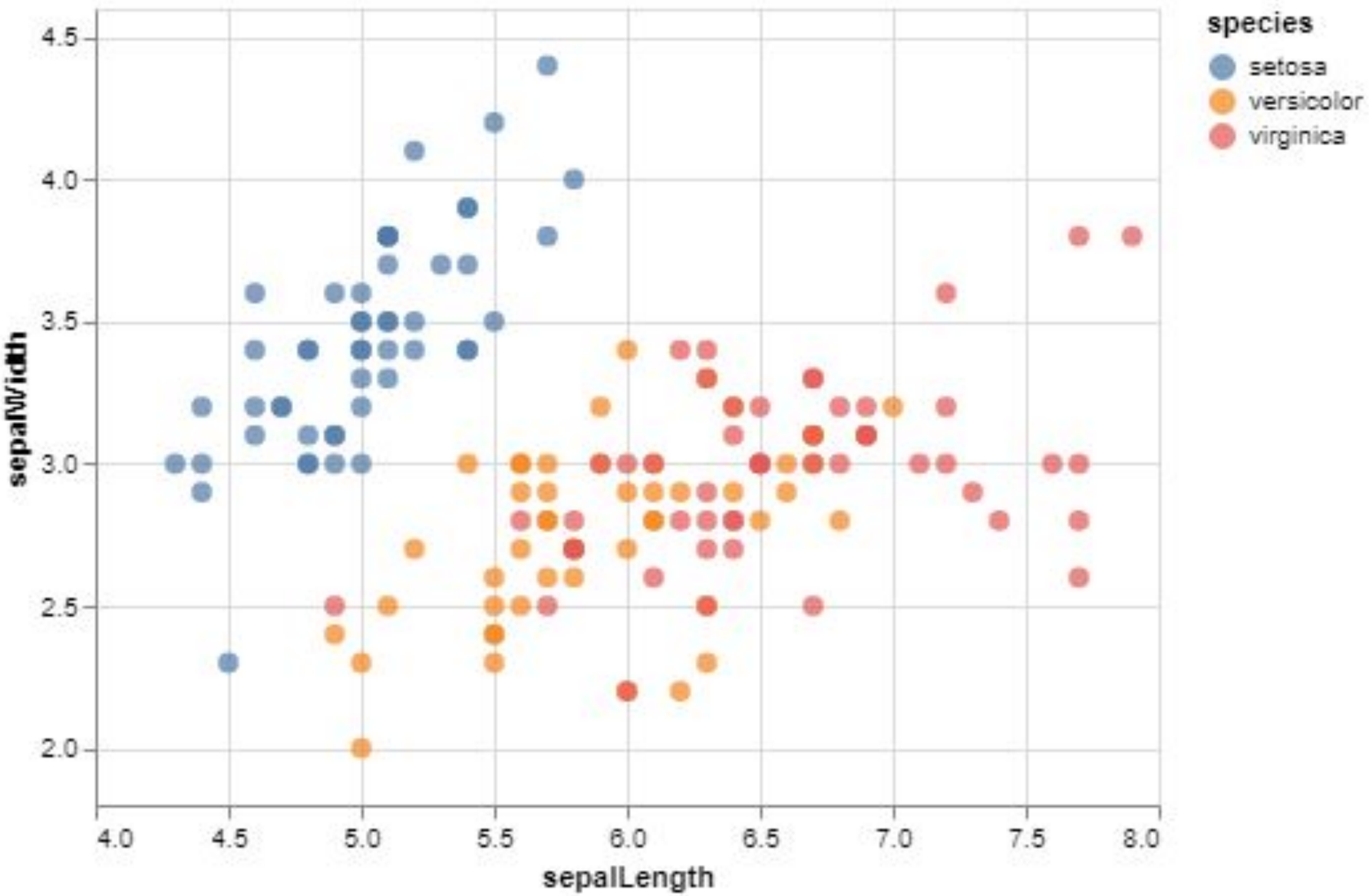
El siguiente dataset es el dataset **iris,** la información de 150 flores.

Las columnas que usaremos del dataset son:

- sepalLength: El largo de los sépalos de la flor en centímetros
- sepalWidth: El ancho de los sépalos de la flor en centímetros
- target: El tipo de flor

sepalLength	sepalWidth	target
5.1	3.5	Setosa
4.9	3	Setosa
6.1	2.9	Versicolor
6.7	3.1	Versicolor
6.2	3.4	Virginica
5.9	3	Virginica

Visualización del dataset



Clasificación de dato nuevo

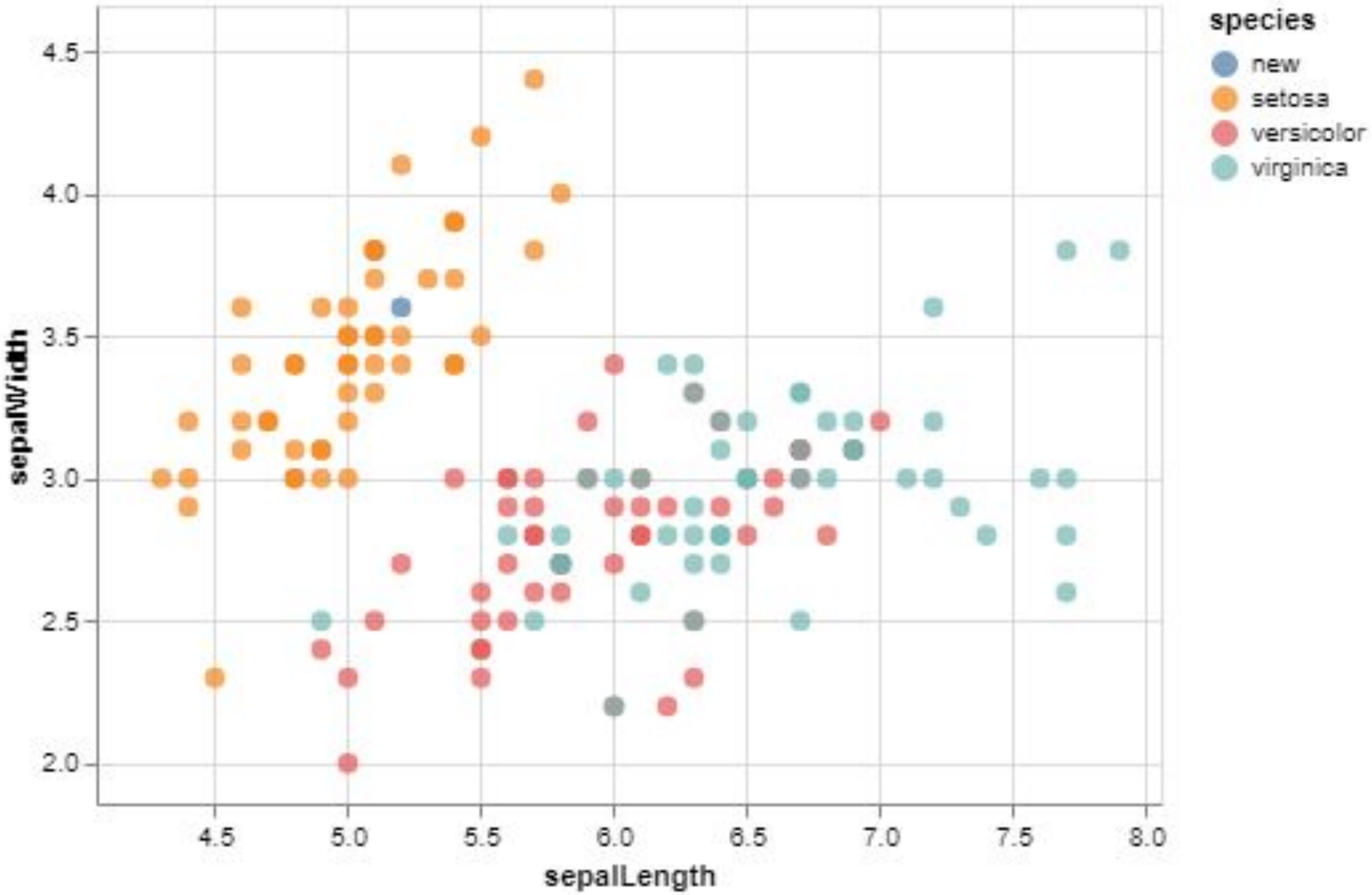
Usualmente no es necesario ningún tipo de “entrenamiento” para kNN, sólo basta con entregarle los datos.

Luego, para clasificar un nuevo dato, basta con:

1. Obtener la distancia de todos los puntos al nuevo dato
2. Obtener los k datos más cercanos al nuevo dato
3. Usando las clases de estos k datos, clasificamos

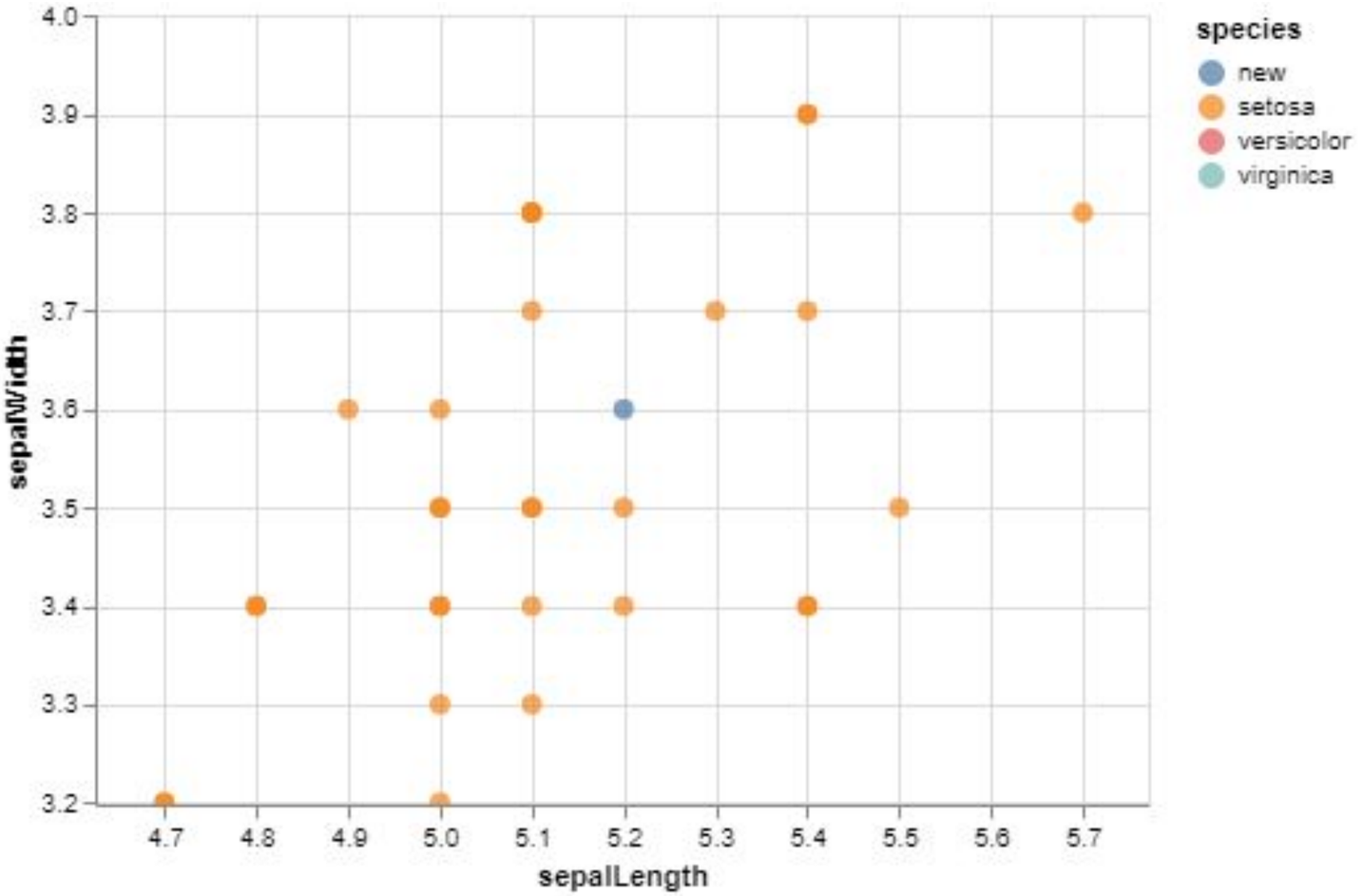
Clasificación de dato nuevo

Queremos clasificar el punto (5.2, 3.6) y revisar los 7 vecinos más cercanos.



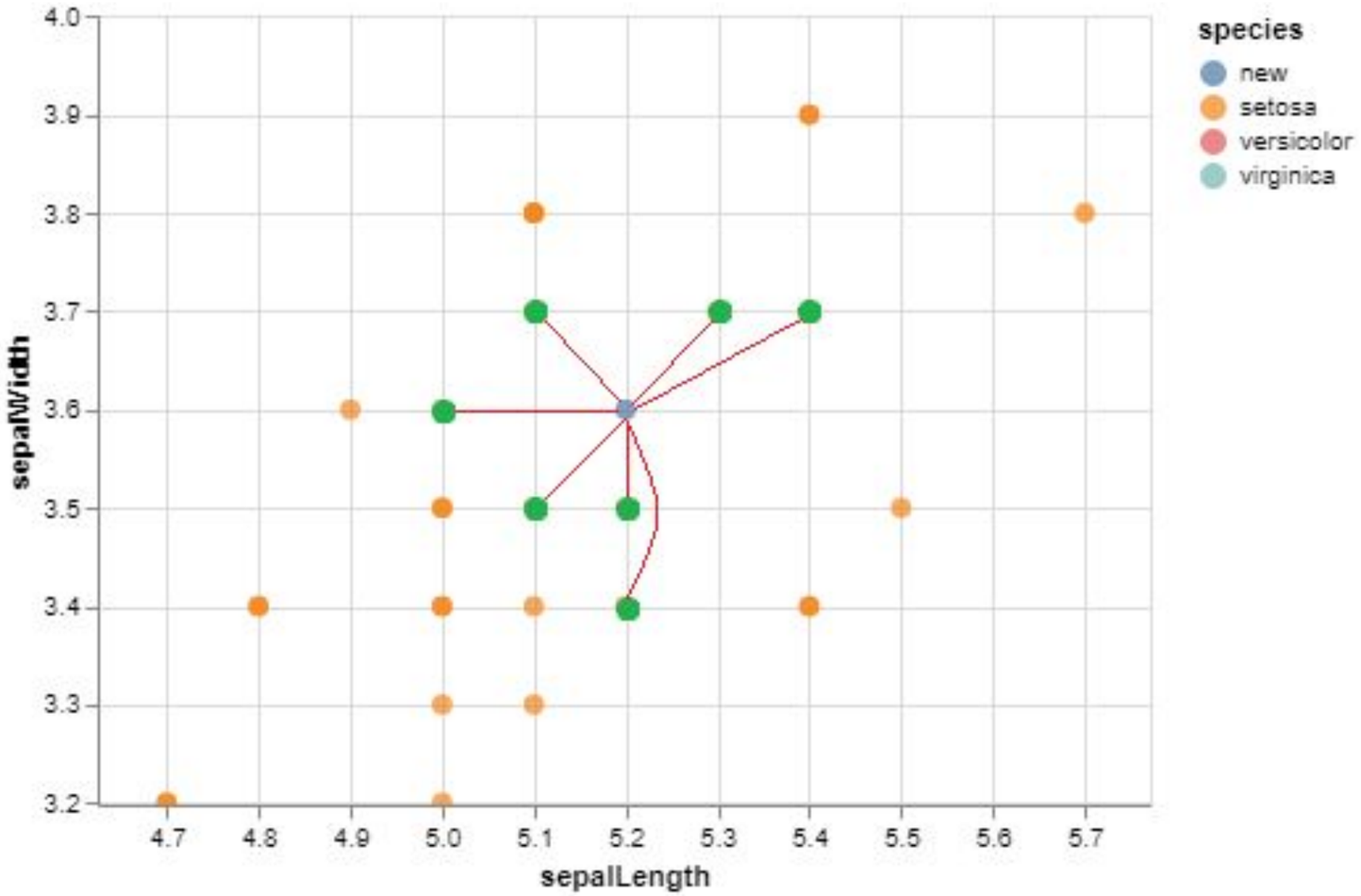
Clasificación de dato nuevo

Queremos clasificar el punto (5.2, 3.6) y revisar los 7 vecinos más cercanos.



Clasificación de dato nuevo

Queremos clasificar el punto (5.2, 3.6) y revisar los 7 vecinos más cercanos.



Clasificación de dato nuevo

Los 7 vecinos más cercanos son setosa, por lo tanto nuestro modelo predecirá que el nuevo dato también lo es.

¿Qué sucede si los vecinos no son iguales?

- Escogemos la clase que más se repita

¿Y si hay un empate? Dependiendo de la cantidad de clases, podemos:

- Escoger un k distinto
- Elegir al azar
- Asignar pesos a cada vecino según su distancia

Regresión con kNN

Para hacer una regresión con kNN, simplemente tomamos los k vecinos mas cercanos y el resultado será nada más que el promedio (asumiendo que cada vecino tiene el mismo peso en la decisión) de los valores obtenidos.

Datasets de múltiples dimensiones

Lo visto anteriormente puede extenderse para un dataset con una cantidad arbitraria de dimensiones, sin embargo no es fácil visualizarlos.

Para cada punto, la **distancia euclidiana** entre el dato que queremos clasificar y el resto de los datos se puede calcular como:

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_i - q_i)^2 + \cdots + (p_n - q_n)^2} = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}.$$

Sin embargo, existen varias métricas distintas como distancia **minkowsky** o **manhattan**.

Pesos desiguales

Dependiendo del problema, a veces nos interesan más los vecinos más cercanos al dato por predecir, que los que se encuentran más lejos.

Si bien esto se usa más para regresión con kNN, también se puede usar para clasificación.

Asignamos un peso a cada vecino, entre mas lejano menor peso, luego en el caso de la clasificación sumamos todos los pesos y la clase con mayor peso es la ganadora.

Para la regresión, el resultado será el promedio ponderado según los pesos obtenidos.

Datos categóricos

Hasta ahora, todo lo que hemos visto funciona perfecto, pero sólo para datos numéricos, no categóricos (excepto por la clase a predecir).

¿Como trabajamos con datos categóricos?

Existen varios métodos distintos.

Distancia discreta

Ya vimos que para calcular la distancia euclidiana podemos usar la función

$$d(\mathbf{p},\mathbf{q}) = \sqrt{(p_1-q_1)^2 + (p_2-q_2)^2 + \cdots + (p_i-q_i)^2 + \cdots + (p_n-q_n)^2} = \sqrt{\sum_{i=1}^{n} (p_i-q_i)^2}.$$

Para variables categóricas, podemos reemplazar la resta de los valores por 1 o 0, tal que:

$$d(x,y) = \begin{cases} 0 & \text{si } x = y \\ 1 & \text{si } x \neq y \end{cases}$$

Encoding

Otra opción consiste en cambiar directamente los datos, de manera que los datos de la misma categoría tengan el mismo valor.

color
Verde
Azul
Rojo
Rojo
Rojo
Verde

Verde = 0
Azul = 1
Rojo = 2

color
0
1
2
2
2
0

Encoding

Problema de este encoding, Verde está muy lejos de Rojo, sin razón alguna.
Pueden generar sesgos debido al ordenamiento ficticio que estamos creando.

color
Verde
Azul
Rojo
Rojo
Rojo
Verde

Verde = 0
Azul = 1
Rojo = 2

color
0
1
2
2
2
0

OneHotEncoding

Cada categoría genera su propia columna. Funciona bien pero para columnas con muchas categorías, el tamaño del dataset puede aumentar considerablemente.

color
Verde
Azul
Rojo
Rojo
Rojo
Verde

colorVerde	colorAzul	colorRojo
1	0	0
0	1	0
0	0	1
0	0	1
0	0	1
1	0	0

PONTIFICIA
UNIVERSIDAD
CATÓLICA
DE CHILE

IIC2433 - Minería de datos

K-Nearest Neighbors

Ricardo Schilling
reschilling@uc.cl