Otros Clasificadores

Naive Bayes, KNN

Introducción

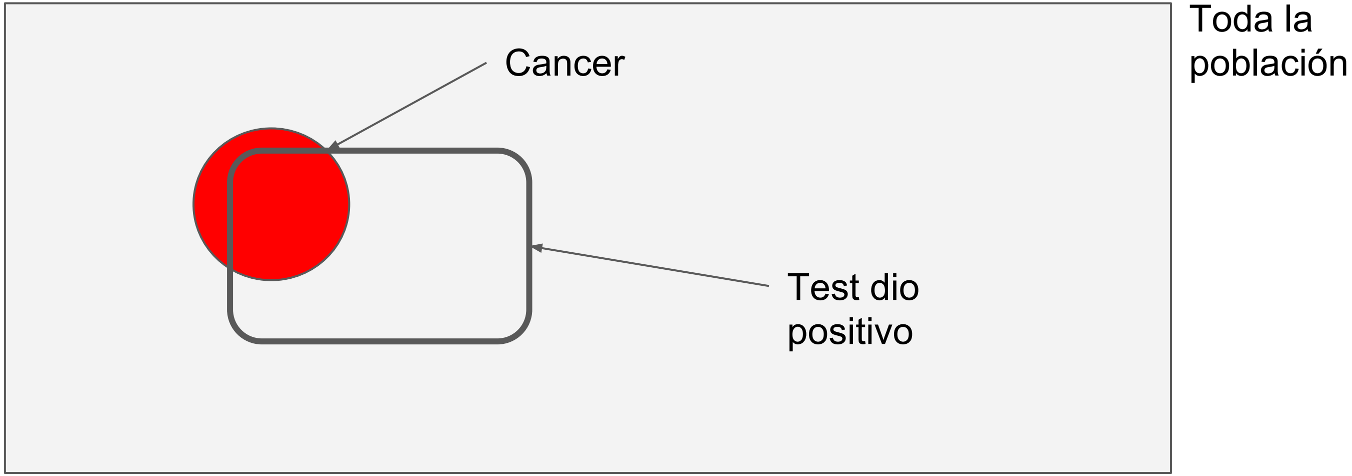
- Thomas Bayes => Teorema de Bayes.

- Método de clasificación supervisada que aplica el ***teorema de Bayes***.

- Método bastante utilizado en el problema de ***clasificación de textos.***

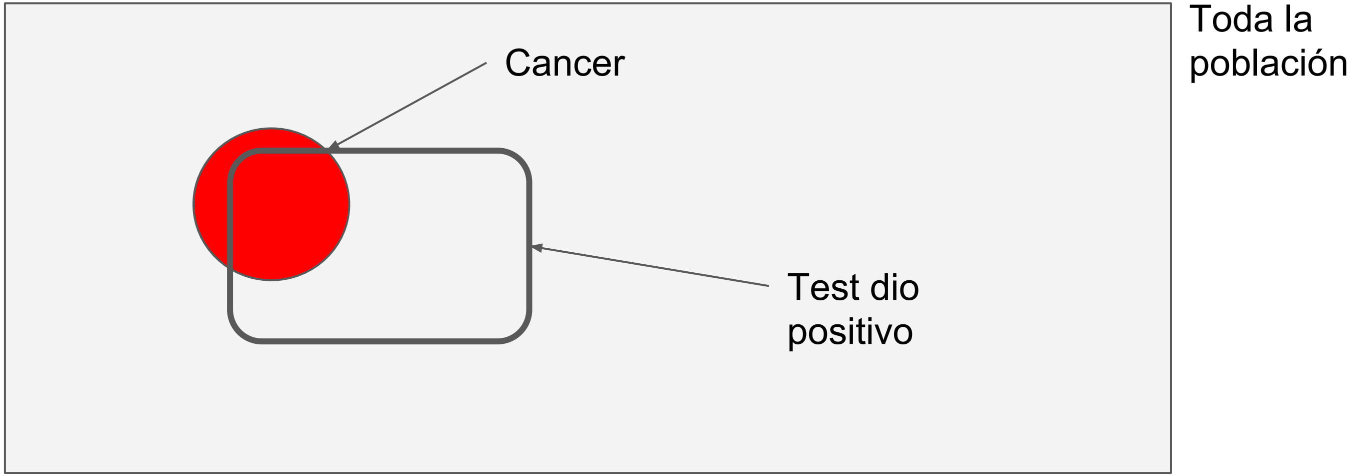
Teorema de bayes - Ejemplo

- $P(C) = 0.01$ → Probabilidad de tener un tipo de cáncer específico.
- $P(+ \mid C) = 0.9$ → Probabilidad de un test dar positivo dado que si hay cáncer.
- $P(- \mid \sim C) = 0.9$ → Probabilidad de un test dar negativo dado que no hay cáncer.
- Pregunta $P(C \mid +)$ =???



Teorema de bayes - Ejemplo

- $P(C) = 0.01$ → Probabilidad a priori.
- $P(+ \mid C) = 0.9$ → Evidencia 1.
- $P(- \mid \sim C) = 0.9$ → Evidencia 2.
- Pregunta $P(C \mid +) = ???$ → Probabilidad a posteriori.



Regla de Bayes

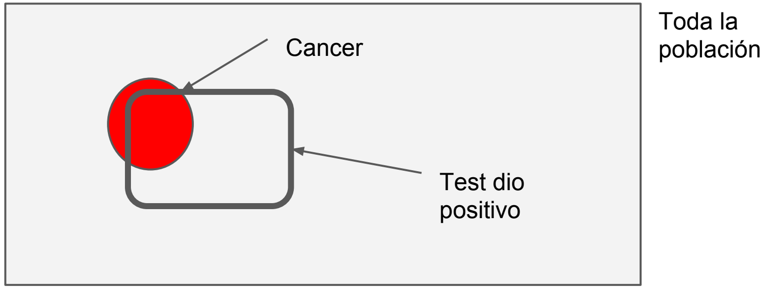
Probabilidad a posteriori = Probabilidad a priori X Evidencia

Regla de Beyes

Probabilidad a posteriori = Probabilidad a priori X Evidencia

$$P(y \mid x_1, \dots, x_n) = \frac{P(y) P(x_1, \dots x_n \mid y)}{P(x_1, \dots, x_n)}$$

Teorema de bayes - Ejemplo



- $P(C) = 0.01$ → Probabilidad a priori.
- $P(+ \mid C) = 0.9$ → Evidencia 1.
- $P(- \mid \sim C) = 0.9$ → Evidencia 2.
- Pregunta $P(C \mid +) =$ 0.08333 → Probabilidad a posteriori.

- $P(C \mid +) = P(C)\, P(+ \mid C) = 0.009$ (.... Sin Normalizar)
- $P(\sim C \mid +) = P(\sim C)\, P(+ \mid \sim C) = 0.099$ (.... Sin Normalizar)

Normalizando:

- $P(+) = P(+ \mid C) + P(+ \mid \sim C) = 0.009 + 0.099 = 0.108$
- $P(C \mid +) = 0.08333$
- $P(\sim C \mid +) = 0.9167$

Naive Bayes

Teorema de Bayes

$$P(y \mid x_1, \dots, x_n) = \frac{P(y)P(x_1, \dots x_n \mid y)}{P(x_1, \dots, x_n)}$$

Suposición de independencia

$$P(x_i | y, x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n) = P(x_i | y)$$

Naive Bayes

$$P(y \mid x_1, \dots, x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \dots, x_n)}$$

Naive Bayes

- Como $P(x_1, \dots, x_n)$ puede considerarse constante, tenemos

$$P(y \mid x_1, \dots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

$$\Downarrow$$

$$\hat{y} = \arg\max_y P(y) \prod_{i=1}^{n} P(x_i \mid y),$$

Calculando los términos...

- *P(y)* es a la proporción del número de veces que la clase y aparece en el conjunto de entrenamiento sobre el total de ejemplos.
- $P(x_i \mid y)$ puede tomar varias formas dependiendo del tipo de distribución que se asume que tiene, por ejemplo:

Naive Bayes Gaussiano

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

Naive Bayes Multinomial - Ejemplo

- Usualmente utilizado en Clasificación de textos (método Bag of words)
- Ejemplo: Tamaño del vocabulario (|V| = 6) → chinese, beijing, shanghai, Macao, Tokyo, Japan.

	Doc	Words	Class
Training	1	Chinese Beijing Chinese	c
	2	Chinese Chinese Shanghai	c
	3	Chinese Macao	c
	4	Tokyo Japan Chinese	j
Test	5	Chinese Chinese Chinese Tokyo Japan	?

Naive Bayes Multinomial - Formulas

- $|V| = 6$

	Doc	Words	Class
Training	1	Chinese Beijing Chinese	c
	2	Chinese Chinese Shanghai	c
	3	Chinese Macao	c
	4	Tokyo Japan Chinese	j
Test	5	Chinese Chinese Chinese Tokyo Japan	?

- Prob. a priori: $\hat{P}(c) = \frac{N_c}{N}$

$P(c) = \frac{3}{4}$

$P(j) = \frac{1}{4}$

- Evidencia: $\hat{P}(w \mid c) = \frac{count(w,c)+1}{count(c)+|V|}$

$P(Chinese|c) = (5+1) / (8+6) = 6/14 = 3/7$

$P(Tokyo|c) = (0+1) / (8+6) = 1/14$

$P(Japan|c) = (0+1) / (8+6) = 1/14$

$P(Chinese|j) = (1+1) / (3+6) = 2/9$

$P(Tokyo|j) = (1+1) / (3+6) = 2/9$

$P(Japan|j) = (1+1) / (3+6) = 2/9$

Naive Bayes Multinomial - Escogiendo una clase

$|V| = 6$

$P(c) = \frac{3}{4}$

$P(j) = \frac{1}{4}$

	Doc	Words	Class
Training	1	Chinese Beijing Chinese	c
	2	Chinese Chinese Shanghai	c
	3	Chinese Macao	c
	4	Tokyo Japan Chinese	j
Test	5	Chinese Chinese Chinese Tokyo Japan	?

$P(\text{Chinese}|c) = (5+1) / (8+6) = 6/14 = 3/7$

$P(\text{Tokyo}|c) = (0+1) / (8+6) = 1/14$

$P(\text{Japan}|c) = (0+1) / (8+6) = 1/14$

$P(\text{Chinese}|j) = (1+1) / (3+6) = 2/9$

$P(\text{Tokyo}|j) = (1+1) / (3+6) = 2/9$

$P(\text{Japan}|j) = (1+1) / (3+6) = 2/9$

$P(c|d5) \propto 3/4 * (3/7)^3 * 1/14 * 1/14 \approx 0.0003$

$P(j|d5) \propto 1/4 * (2/9)^3 * 2/9 * 2/9 \approx 0.0001$

Naive Bayes Bernoulli

$$P(x_i \mid y) = P(i \mid y)x_i + (1 - P(i \mid y))(1 - x_i)$$

Naive Bayes - Consideraciones finales

- Principales ventajas:
  - Facil de implementar
  - No necesita de ningun parámetro

Scikit-Learn

- Es una libreria de Machine Learning en Python
- http://scikit-learn.org/stable/index.html
- Esta muy bien documentado, con referencias a los algoritmos que tiene implementado.
- Para Naive Bayes: http://scikit-learn.org/stable/modules/naive_bayes.html

K-vecinos más cercanos (KNN)

- KNN es un algoritmo muy simple que almacena todos los casos de entrenamiento en una estructura y luego clasifica nuevos casos utilizando medidas de similaridad (por ejemplo: distancia Euclidiana).
- Se ha utilizado en el reconocimiento de patrones desde los años 70.

Clasificación con KNN - Algoritmo

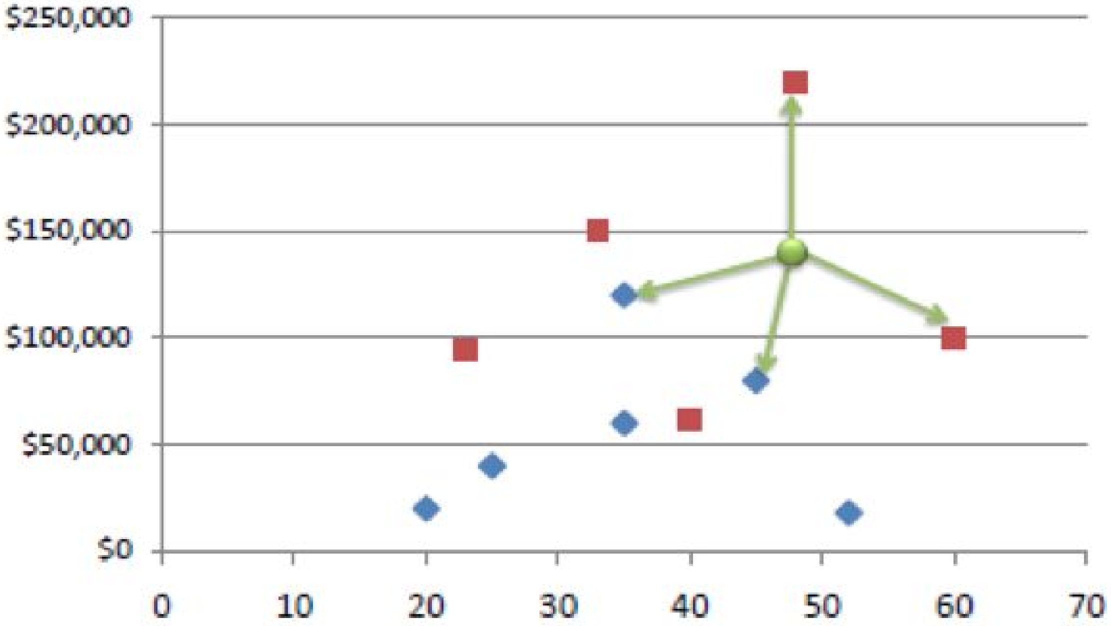
1. Toda la data de entrenamiento es almacenada en una estructura de datos métrica *n*-dimensional (*n* = número de variables o características).
2. Un caso nuevo es clasificado por mayoría de votos de sus k -vecinos más cercanos. Es decir, la clase más votada es la ganadora.

La cercanía es medida por funciones de distancia.

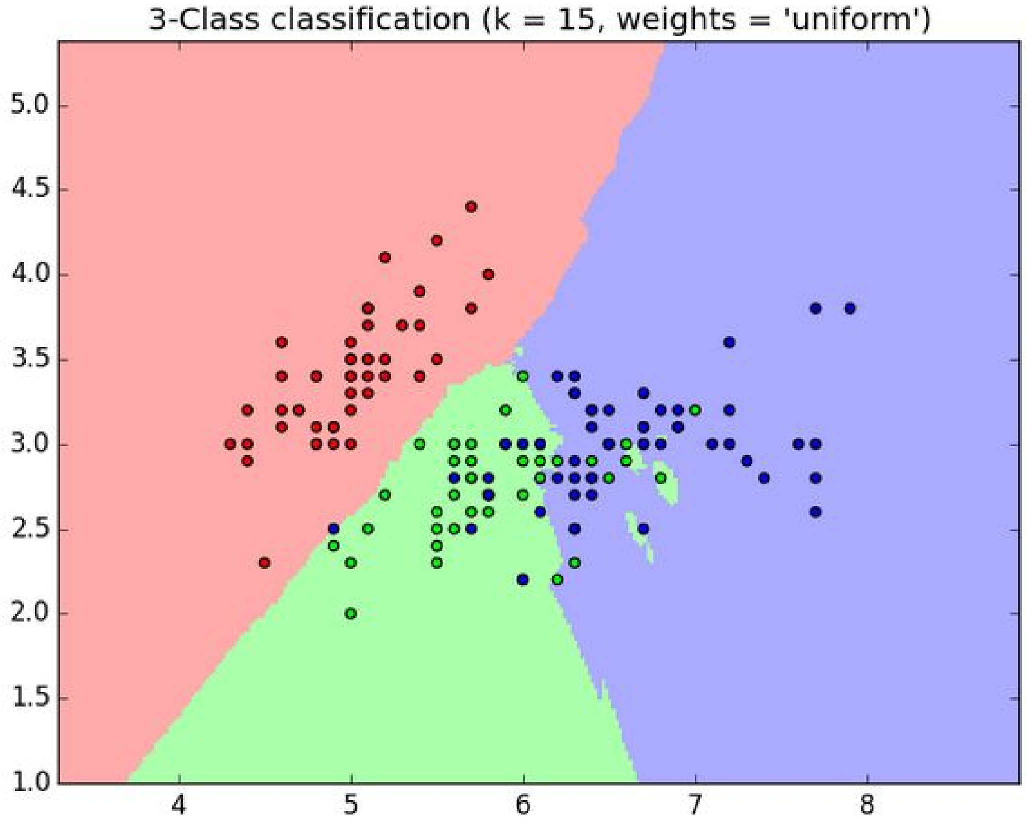
- Distancia Euclidiana
- Distancia de Manhattan
- Distancia de Minkowski
- Distancia de Hamming (funciona para variables discretas)

KNN - Ejemplo

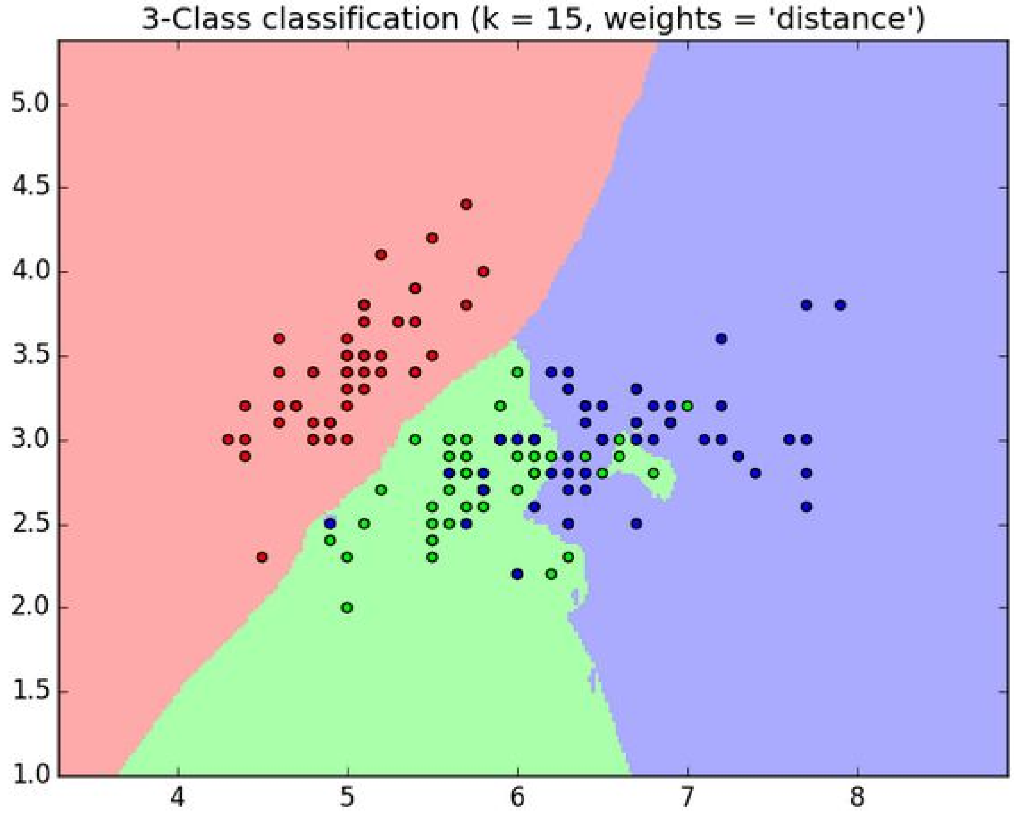
Problema: Responder si alguien una persona comprará una casa o no



KNN - Fronteras de decisión y pesos



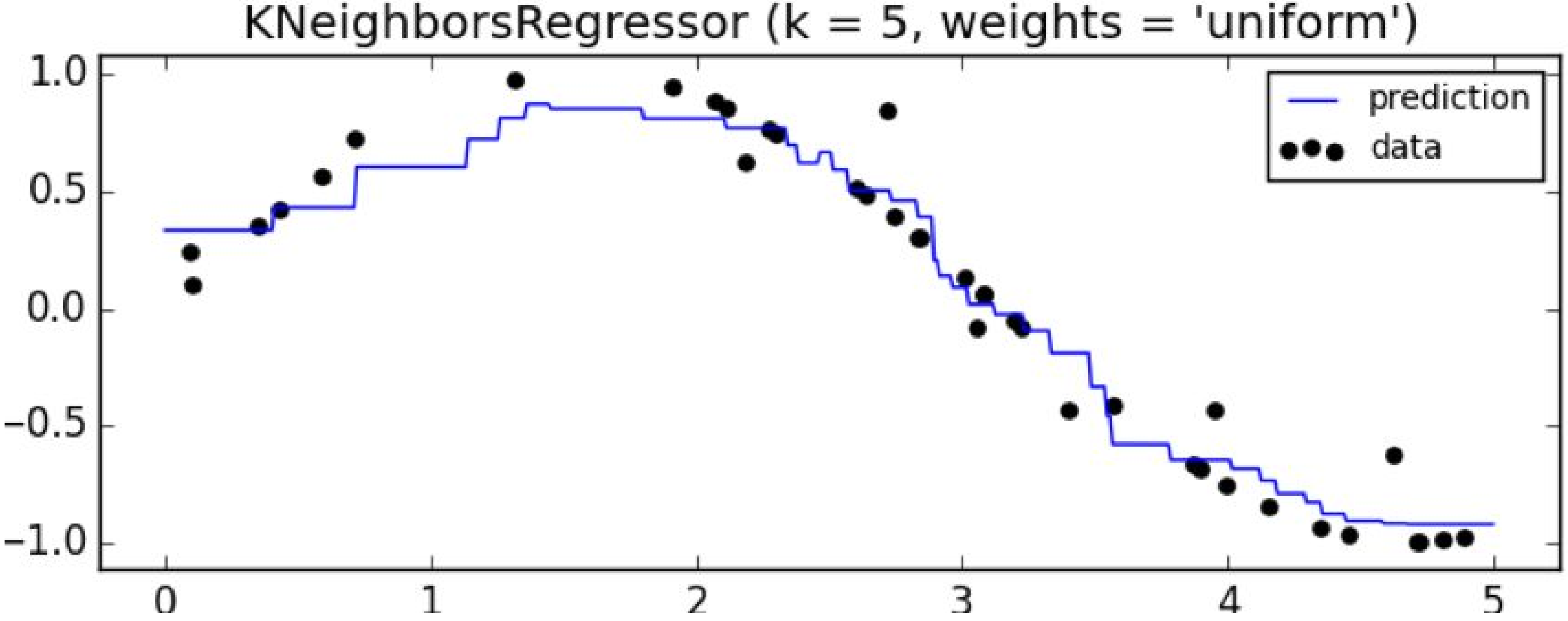
KNN - Fronteras de decisión y pesos



KNN - Regresión

- Muy parecido al caso de clasificación
- Se escogen los **k**-vecinos más cercanos
- El valor de regresión es igual a la media aritmética de las ***k*** variables de salida ***y*** de los vecinos más cercanos.

KNN Regresión - Ejemplo



Estructura de datos para KNN

Para encontrar los k-vecinos más cercanos, uno puede aplicar fuerza bruta.

- Da la respuesta correcta, solo que si *n* es grande, la búsqueda tomaría siglos.

Sin embargo existe várias estructuras de datos que logran hacer una búsqueda más eficiente por ejemplo:

- K-D tree
- Ball tree
- Vantage-point tree

KNN - Python sklearn.

- Implementación y documentación de K-nearest Neighbors en Python:

  http://scikit-learn.org/stable/modules/neighbors.html