UNIVERSIDAD COMPLUTENSE MADRID

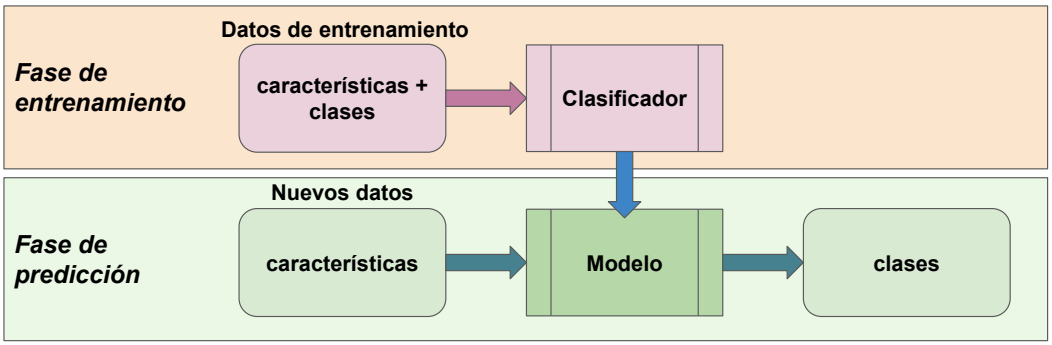
Desarrollo de Aplicaciones y Sistemas Inteligentes (DASI)

Clasificación

Juan Pavón Mestras
Dep. Ingeniería del Software e Inteligencia Artificial UCM

Clasificación

- Una de las aplicaciones comunes del aprendizaje supervisado
  - **Observaciones ⇒ Categorías**
    - Observaciones son las **características** (***features***) o **variables predictivas**
    - Las categorías con **etiquetas** o **clases**
  - El modelo de clasificación se genera con **entrenamiento** para que aprenda de las características y los objetivos de las muestras de entrenamiento
  - Posteriormente, con nuevos datos, el modelo entrenado será capaz de determinar las clases a las que pertenecen



Tipos de clasificación

- **Binaria**
  - Dos clases (mutuamente excluyentes)
  - Ejemplo: clasificador de correo spam
- **Multi-clase**
  - Más de dos clases posibles (mutuamente excluyentes)
  - Ejemplo: reconocimiento de dígitos (0 a 9)
- **Multi-etiqueta**
  - Varias etiquetas posibles a la vez
  - Ejemplo: una película que se puede considerar de varios géneros (acción, aventura, ciencia ficción)
  - A veces se puede tratar como un conjunto de clasificaciones binarias, una por cada posible clase

DASI

Clasificación con Naïve-Bayes

Naïve-Bayes

- En **aprendizaje supervisado**, se dispone de una muestra (x_i, y_i) con $i=1..n$, y el objetivo es aprender la función que relaciona el vector de características $\boldsymbol{x}_i$ con la clase a la que pertenece pertenece $\boldsymbol{y}_i \in 1..m$ siendo m el número de clases posibles
- Un clasificador **probabilístico** proporciona una forma de relacionar las entradas con las salidas, pero, además, permite determinar las probabilidades p(y|x) a tenor de las observaciones disponibles
- Pueden ser
  - **Modelos discriminativos**: tratan de aprender directamente las
  - probabilidades $\boldsymbol{p(y|x)}$
    - Ejemplo: regresión logística
  - **Modelos generativos**: tratan de aprender las probabilidades condicionales para cada clase de $\boldsymbol{y}$, $\boldsymbol{p(y|x)}$, y las probabilidades a priori de cada clase $\boldsymbol{p(y)}$, aplicando la regla de Bayes
    ⇒ Se llaman generativos porque a partir de ellos podemos generar vectores de características x para cada clase y
    - Ejemplo: Naïve-Bayes

Naïve-Bayes

- Clasificador probabilístico
  - **Bayes**: mapea la probabilidad de las características de entrada observadas dada una clase posible con la probabilidad de la clase dada teniendo en cuenta las evidencias usando el Teorema de Bayes
    - **Teorema de Bayes**:

$$p(y|\boldsymbol{x}) = \frac{p(\boldsymbol{x} \wedge y)}{p(\boldsymbol{x})} = \frac{p(\boldsymbol{x}|y)p(y)}{\sum_{y'=1}^{m} p(\boldsymbol{x}|\boldsymbol{y}')p(y')}$$

  - **Naïve**: Simplifica el cálculo de la probabilidad asumiendo que las características predictivas son mutuamente independientes

Teorema de Bayes

- **Ejemplo**: Se han obtenido los siguientes datos en un estudio sobre los tests del covid con 10.000 personas:

	Covid	No covid	Total
Test positivo	80	900	980
Test negativo	20	9000	9200
Total	**100**	**9900**	**10000**

  - La probabilidad de tener Covid (C) siendo el test positivo (TP) es:

    P (C | TP) = (P (TP | C) * P (C)) / P(TP)

    = (0,8 * 0,01) / 0,098 = 8,16% *(esto es mucho más que P(C)=1%, sin hacer el test)*

Naïve Bayes

- Si se tienen múltiples características (variables $x_1, x_2, \ldots, x_n$), entonces se tendría:

$$P(y_k \mid x) = \frac{P(x \mid y_k)\, P(y_k)}{P(x)}$$

siendo:

  - ***P(y_K)*** la distribución de probabilidad de las clases (esto es, sin tener en cuenta las características) ⇒ probabilidad a priori ***(prior)***
  - ***P(y_K\|x)*** la probabilidad dada la observación de las características x ⇒ ***posterior***
  - ***P(x\|y_K)***, o ***P(x_1, x_2,..., x_n \|y_K)*** la distribución conjunta de las n características cuando un elemento pertenece a la clase y_K ⇒ ***likelihood***
    ⇒ Esto es costoso de calcular cuando n aumenta (si xi son n variables binarias, habría que calcular 2^{n+1}-1 probabilidades para todas las combinaciones de valores de x e y, es decir, $P(y \wedge x_1 \wedge \cdots \wedge x_n)$
    ⇒ Naïve Bayes asume que las variables $x_1, x_2, \ldots, x_n$ son **independientes,** entonces se hace más fácilmente:

$$P(x_1 \wedge \cdots \wedge x_n \mid y) = P(x_1 \mid y) * P(x_2 \mid y) * \cdots * P(x_n \mid y)$$

  - ***P(x)*** la ***evidencia***, depende solo de la distribución de las características x_i, que no es específica de las clases, luego es una constante normalizada. Por tanto:

$$P(y_k \mid x) \propto P(x \mid y_k) P(y_k) = P(x_1 \mid y_k) * P(x_2 \mid y_k) * \ldots * P(x_n \mid y_k)$$

Clasificador Naïve Bayes

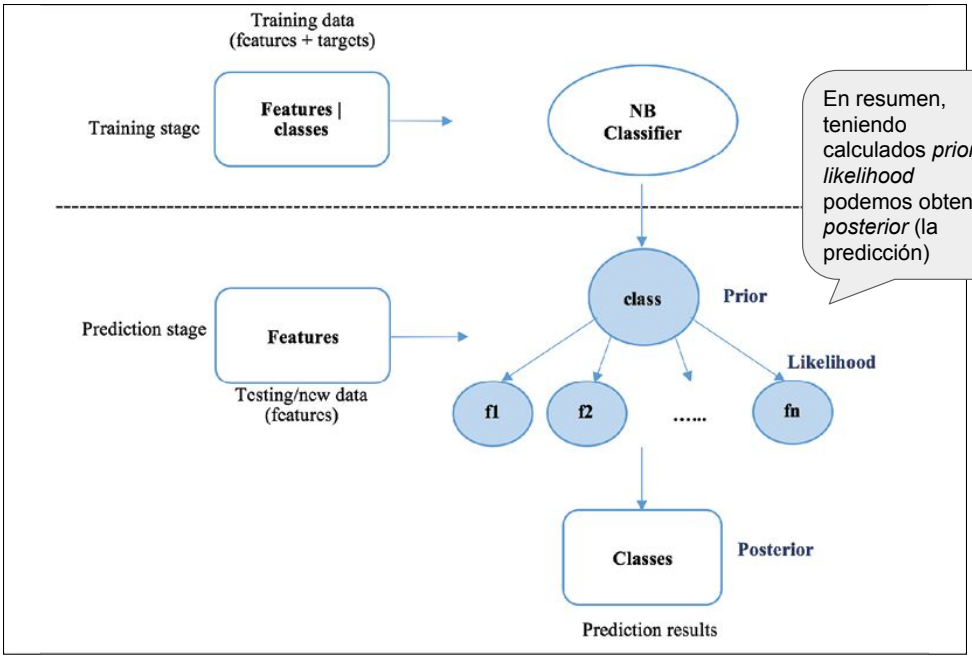


Figure 2.6: Training and prediction stages in Naïve Bayes classification, del libro Yuxi (Hayden) Liu: *Python Machine Learning By Example, Third Edition*. Packt Publishing, 2020

Ejemplo

- Considérese un ejemplo de recomendación de películas
  - Entrenado con 4 usuarios
    - peli1, peli2, peli3 son características (features)
    - pelix es la clase objetivo
    - 0: no gusta, 1: sí gusta
  - Se trata de ver para un quinto usuario si le gustará la pelix

	Usuario	peli1	peli2	peli3	**pelix ?**
Datos de entrenamiento	usuario1	0	1	1	**Y**
	usuario2	0	0	1	**N**
	usuario3	0	0	0	**Y**
	usuario4	1	1	0	**Y**
Caso de prueba	*usuario5*	*1*	*1*	*0*	***?***

Ejemplo adaptado del capítulo 2, del libro Yuxi (Hayden) Liu: *Python Machine Learning By Example, Third Edition*. Packt Publishing, 2020

Ejemplo

- Las probabilidades serán las frecuencias relativas que se obtienen contando las observaciones del conjunto de entrenamiento

$$P(x_k = x_{ki} \,|y = y_i) = \frac{\#(x_k = x_{ki} \wedge y = y_i)}{\#(y = y_i)}$$

- Calcular la probabilidad de que le guste la pelix al usuario5, P(Y), será calcular:
  - `P(Y|x)` siendo `x = (1, 1, 0)`
    - p_1, p_2, p_3 indican si al usuario le gusta o no la película correspondiente
      - En este caso (1, 1, 0)
    - Habrá que calcular el likelihood:
      $P(p_1=1|S)$, $P(p_2=1|S)$, $P(p_3=0|S)$, y $P(p_1=1|N)$, $P(p_2=1|N)$, $P(p_3=0|N)$
    - Como en el conjunto de entrenamiento $P(p_1=1|N) = 0$ se tiene que

      $P(N|x) \sim P(p_1=1|N)*P(p_2=1|N) = 0 \Rightarrow$ Siempre saldrá que le gustará la peli

Ejemplo

- Para evitar el factor de multiplicación por 0 como en este caso se puede aplicar el ***alisado de Laplace***
  - Se incrementar la probabilidad de 0 a algún valor positivo (el factor de alisado, *l*, *normalmente 1*) a todas las características

$$P(x_k = x_{ki} \,|y = y_i) = \frac{\#(x_k = x_{ki} \wedge y = y_i) + l}{\#(y = y_i) + l|x_k|}$$

  - Así se tendria entonces:

0 veces que gusta la peli1 dada la clase N, y 1 vez

Hay una vez que no gusta la pelix, y son 2 las opciones posibles

$P(p_1=1|N) = (0+1) / (1+2) = 1/3$

$P(p_1=1|Y) = (1+1) / (3+2) = 2/5$

$P(p_2=1|N) = (0+1) / (1+2) = 1/3$ $\quad$ $P(p_2=1|Y) = (2+1) / (3+2) = 3/5$

$P(p_3=1|N) = (0+1) / (1+2) = 1/3$ $\quad$ $P(p_3=1|Y) = (2+1) / (3+2) = 3/5$

  - Ahora el cálculo sale mejor:

$$\frac{P(N\,|x)}{P(Y\,|x)} \propto \frac{P(N) * P(f_1 = 1|N) * P(f_2 = 1\,|N) * P(f_3 = 0\,|N)}{P(Y) * P(f_1 = 1\,|Y) * P(f_2 = 1\,|Y) * P(f_3 = 0\,|Y)} = \frac{125}{1458}$$

  - Se tiene finalmente que

    P(Y|x) = 1458*100 / 1583 **= 92%**

Implementación

Con Python, en Google Colab:

https://colab.research.google.com/drive/1PC_nVmqgyeS1xbmaZMJN8RHHv36Irgza?usp=sharing

Evaluación del rendimiento

- **Matriz de confusión *(Confusion matrix)***: resume cómo han sido los resultados obtenidos con los ejemplos de prueba

		Predicted	
		Negative	Positive
Actual	Negative	TN	FP
	Positive	FN	TP

TN = True Negative
FP = False Positive
FN = False Negative
TP =True Positive

- La ***exactitud*** de la clasificación (***accuracy***) es la tasa de aciertos:

$$\frac{TN + TP}{TN + TP + FP + FN}$$

Evaluación del rendimiento

- **Precisión**: mide la fracción de ejemplos positivos correctos

$$\frac{TP}{TP + FP}$$

- **Exhaustividad *(Recall)*** o **sensibilidad**: mide la fracción de ejemplos positivos que se han detectado sobre el total de casos

$$\frac{TP}{TP + FN}$$

- **F1 score**: media armónica de los dos anteriores:

$$f_1 = 2 * \frac{precision * recall}{precision + recall}$$

  - Más adecuada que las anteriores porque es más sensible a que uno de los valores sea muy pequeño

Evaluación del rendimiento

- Dependerá del problema que nos interese más una medida u otra
- Ejemplo: clasificador binario para determinar si una persona está enferma (Sí | No)
  - Si se tienen tres clasificadores con las siguientes medidas:

clasificador1:	Precisión(enfermo)=0.96	*Recall*(enfermo)=0.50
clasificador2:	Precisión(enfermo)=0.55	*Recall*(enfermo)=1
clasificador3:	Precisión(enfermo)=0.80	*Recall*(enfermo)=0.92

    - ➢ ¿Cuál es más útil para detectar una enfermedad letal y contagiosa?
    - ➢ ¿Cuál será el más indicado para una leve que se cura con un medicamento?
    - ➢ ¿Y para otro problema de clasificación como “buen-pagador/moroso” para concederle un préstamo?

⇒ Al entrenar el modelo (fase de aprendizaje) habrá que tener en cuenta la medida que más nos interese (precisión, *recall*, F1)

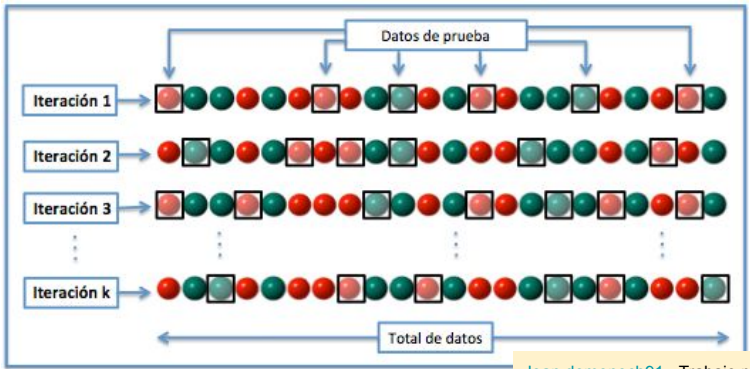
Área bajo la curva

- *Area under the curve* (**AUC**) of the *receiver operating characteristic* (**ROC**)
  - ROC es una curva de la tasa de TP vs. la tasa de FP con varios umbrales de probabilidad entre 0 y 1
    - La TP es equivalente al recall, y la FP a la fracción de negativos que se han identificado incorrectamente como positivos

⇒ Ver código

Ajuste del modelo con validación cruzada

- Usaremos *k-fold cross validation* (visto en el tema anterior)
  - Se dividen los datos originales en subconjuntos de k elementos, preservando la proporción entre clases objetivo
    - Si el conjunto de entrenamiento es pequeño se suele tomar k = 5 o 10
    - Si el conjunto de entrenamiento es grande se suele tomar k = 3 o 4
  - Cada uno de estos conjuntos se utilizará de prueba para evaluar el modelo, y en cada vuelta el resto se utilizarán para entrenamiento
  - Al final se toma la media para generar el resultado global



Joan.domenech91 - Trabajo propio, Wikimedia commons

DASI

Bibliografía

Bibliografía

- Sridhar Alla & Suman Kalyan Adari: *Beginning MLOps with MLFlow. Deploy Models in AWS SageMaker, Google Cloud, and Microsoft Azure*. Apress, 2021.
- Stuart Russell, Peter Norvig: *Artificial Intelligence: A Modern Approach, 3rd edition*. Prentice Hall, 2016.Teik Toe Teoh & Zheng Rong: Artificial Intelligence with Python. Springer Nature, 2022.
- Yuxi (Hayden) Liu: *Python Machine Learning By Example, Third Edition*. Packt Publishing, 2020.
- Documentación en línea de las distintas herramientas