

—NAIVE BAYES ALGORITHM—

Naive Bayes es un algoritmo de ML supervisado utilizado para clasificar datos en múltiples categorías.

BAYES THEOREM: Describe la probabilidad de un evento basándose en conocimiento (BT) previo relevante para este evento.

Imagínate que vimos a una persona pasar por la oficina pero no sabemos si fue Adrián o Jimena. Por lo tanto: $P(\text{Adrián}) = P(\text{Jimena}) = 0.5$.

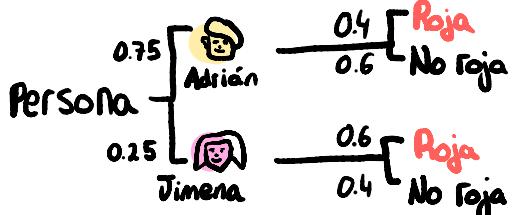
↳ A esta predicción se le denomina prior.

Según los días que : ↳ Adrián 3 veces a la semana: $P(\text{Adrián}) = 0.75$. vienen a las oficinas Jimena 1 vez a la semana: $P(\text{Jimena}) = 0.25$.

Y también nos fijamos que la persona llevaba sudadera roja:

- Adrián la lleva 2 veces a la semana: $P(R|A) = 0.4$.
- Jimena la lleva 3 veces a la semana: $P(R|J) = 0.6$.

↳ A esta predicción se le denomina posterior.



$$P(A|R) = \frac{P(A) \cdot P(R|A)}{P(A) \cdot P(R|A) + P(B) \cdot P(R|B)}$$

← La probabilidad que queremos
← Normalización

$$P(B|R) = \frac{P(B) \cdot P(R|B)}{P(A) \cdot P(R|A) + P(B) \cdot P(R|B)}$$

BAYESIAN LEARNING: Para aplicar el BT al ML sólo hay que repetirlo.

Vamos a poner el ejemplo de un clasificador de spam que clasifica mensajes en Spam(S) y ham(̄S, no spam):



$$P(\text{Palabra}) = \frac{\text{num_palabras}}{\text{num_mensajes}}$$

- Para "easy" (E): $P(E|S) = 1/3$; $P(E|\bar{S}) = 1/5$.
- Para "money" (M): $P(M|S) = 2/3$; $P(M|\bar{S}) = 1/5$.

Naive Bayes Algorithm: Vamos a hacer dos asunciones ingenuas (naive):

- Probabilidad de que dos eventos pasen a la vez: $P(A \cap B) = P(A) \cdot P(B)$.
Sob podemos usar esta fórmula cuando ambos eventos son independientes.
Así que asumimos que todos los eventos son independientes.
- Fórmula de la probabilidad condicional: $P(A|B) \cdot P(B) = P(A) \cdot P(B|A)$.
El truco es que nos olvidamos de $P(B)$: $P(A|B) \propto P(A) \cdot P(B|A)$.
El resultado será una probabilidad proporcional. Para normalizarla dividimos la probabilidad entre la suma de todas las probabilidades.

Ej.: Prob. condicional: $P(\text{spam} | \text{'easy'}, \text{'money'}) \propto \underbrace{P(\text{'easy'}, \text{'money'} | \text{spam}) \cdot P(\text{spam})}$

Naive assumption: $P(\text{'easy'} | \text{spam}) \cdot P(\text{'money'} | \text{spam})$

Así que tenemos:

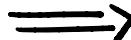
$$P(\text{spam} | \text{'easy'}, \text{'money'}) \propto \frac{1}{12} \rightarrow P(\text{spam} | \text{'easy'}, \text{'money'}) = \frac{\frac{1}{12}}{\frac{1}{12} + \frac{1}{40}} = \frac{40}{40+12} = \frac{40}{52} = \frac{10}{13}$$
$$P(\text{ham} | \text{'easy'}, \text{'money'}) \propto \frac{1}{40} \rightarrow P(\text{ham} | \text{'easy'}, \text{'money'}) = \frac{3}{13}$$

Bag of words (BoW): La mayoría de algoritmos de ML utilizan datos numéricos.
Si tenemos un dataset de texto deberemos pasarlo a números utilizando una BoW.



Cada mensaje será una fila y cada token una columna.

['Hello, how are you!',
'Win money, win from home.',
'Call me now',
'Hello, Call you tomorrow?']



	are	call	from	hello	home	how	me	money	now	tomorrow	win	you
0	1	0	0	1	0	1	0	0	0	0	0	1
1	0	0	1	0	1	0	0	1	0	0	2	0
2	0	1	0	0	0	0	1	0	1	0	0	0
3	0	1	0	1	0	0	0	0	0	1	0	1

Para tokenizar un string, darle un ID a cada token, contar el número de veces que aparece y normalizar el string podemos utilizar el método CountVectorizer de sklearn:

```
class sklearn.feature_extraction.text.CountVectorizer(*, input='content', encoding='utf-8', decode_error='strict',
strip_accents=None, lowercase=True, preprocessor=None, tokenizer=None, stop_words=None, token_pattern='(?u)\b\w+\b',
ngram_range=(1, 1), analyzer='word', max_df=1.0, min_df=1, max_features=None, vocabulary=None, binary=False, dtype=<class
'numpy.int64'>)  
Ignores all punctuation
```

[source]