

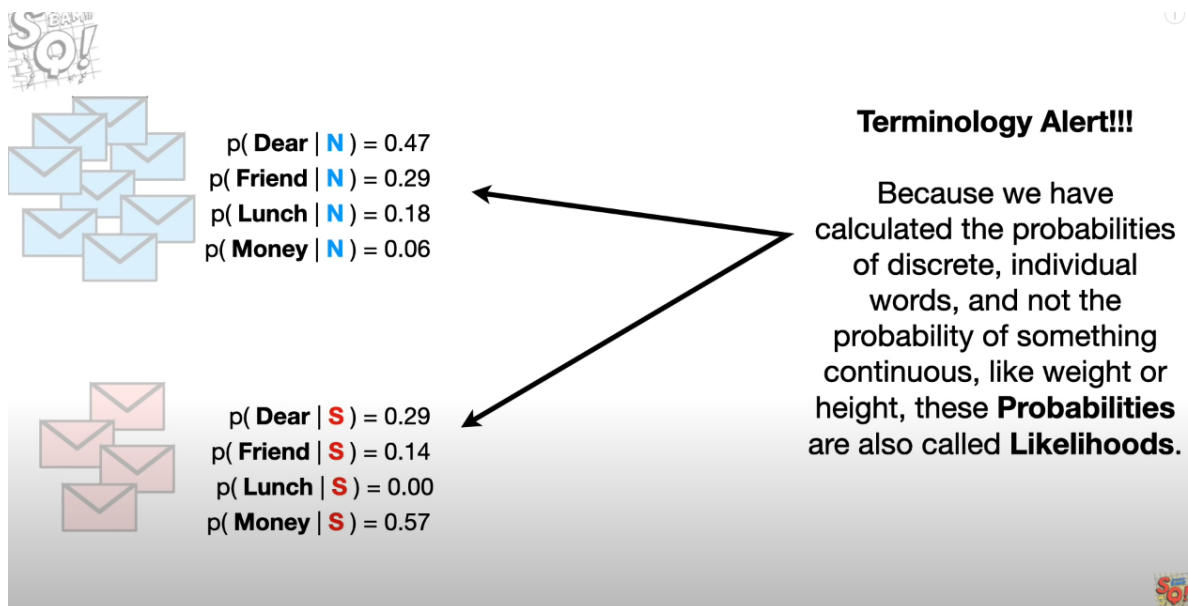
Nombre: Daniel Barandica

Apuntes

Naiva bayes

Se tiene un histograma con los mensajes obtenidos

Se le asigna una probabilidad a cada palabra de cada mensaje de spam y normales.



Se calcula la probabilidad de que sea un mensaje normal (prior)

06

$$p(N) = \frac{8}{8 + 4} = 0.67$$

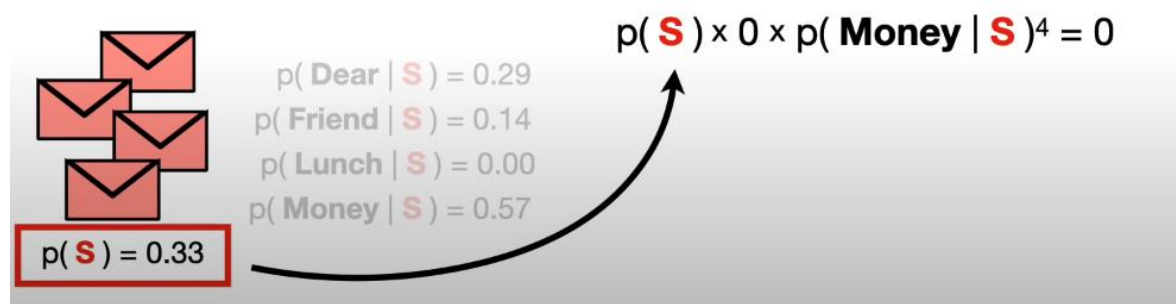
Para poder clasificar los mensajes, se debera multiplicar las probabilidades (prior y las de las palabras) y luego el que obtenga la mayor probabilidad, sera el mensaje correcto

$$p(\mathbf{N}) \times p(\mathbf{Dear} | \mathbf{N}) \times p(\mathbf{Friend} | \mathbf{N}) = 0.09$$

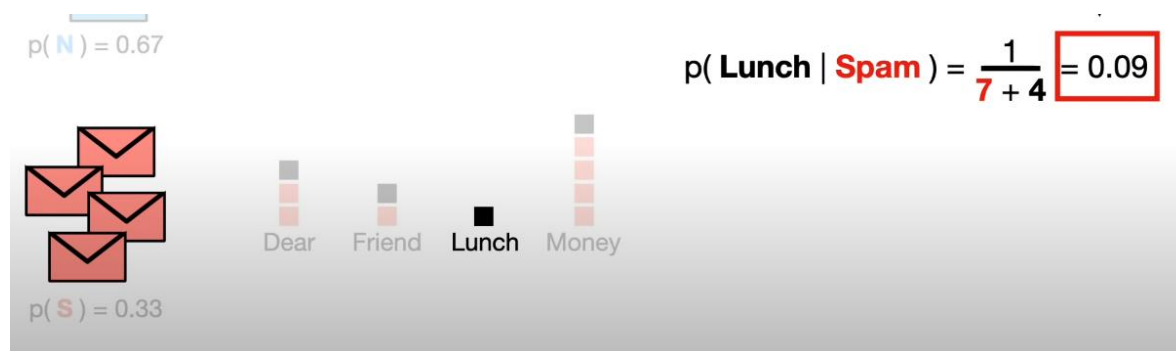
$$p(\mathbf{S}) \times p(\mathbf{Dear} | \mathbf{S}) \times p(\mathbf{Friend} | \mathbf{S}) = 0.01$$

»»

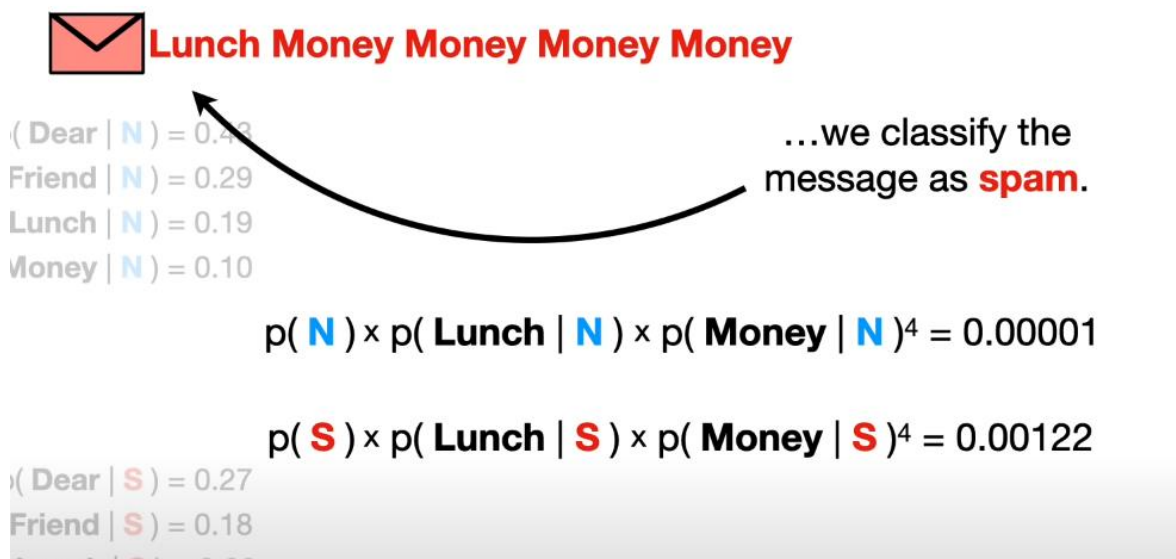
Si una no de las palabras no esta en el entrenamiento inicial, causaria error al tratar de clasificar




Se le debe agregar un alpha, con el fin de que la probabilidad de una palabra que no estuvo en el entrenamiento, no sea igual a cero.



Luego al calcular nuevamente, se obtiene una correcta clasificacion del mensaje



 **Lunch Money Money Money Money**

$p(\text{Dear} | \text{N}) = 0.43$
 $p(\text{Friend} | \text{N}) = 0.29$
 $p(\text{Lunch} | \text{N}) = 0.19$
 $p(\text{Money} | \text{N}) = 0.10$

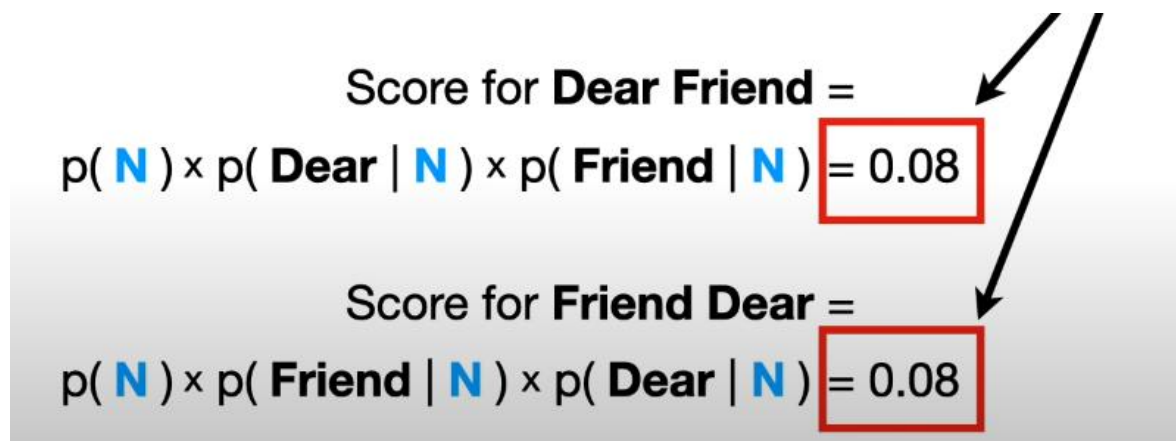
...we classify the message as **spam**.

$$p(\text{N}) \times p(\text{Lunch} | \text{N}) \times p(\text{Money} | \text{N})^4 = 0.00001$$

$$p(\text{S}) \times p(\text{Lunch} | \text{S}) \times p(\text{Money} | \text{S})^4 = 0.00122$$

$p(\text{Dear} | \text{S}) = 0.27$
 $p(\text{Friend} | \text{S}) = 0.18$
 $p(\text{Lunch} | \text{S}) = 0.09$
 $p(\text{Money} | \text{S}) = 0.00$

En naive bayes no importa el orden de las palabras, al final tendra la misma probabilidad



Score for **Dear Friend** =

$$p(\text{N}) \times p(\text{Dear} | \text{N}) \times p(\text{Friend} | \text{N}) = 0.08$$

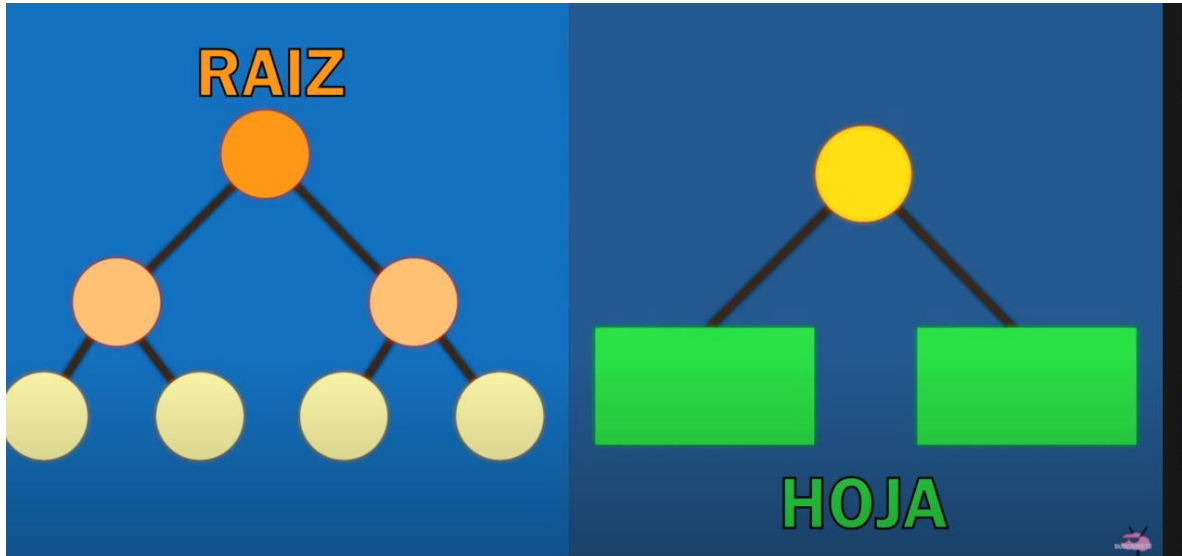
Score for **Friend Dear** =

$$p(\text{N}) \times p(\text{Friend} | \text{N}) \times p(\text{Dear} | \text{N}) = 0.08$$

Decision tree y Random forest

Decision tree

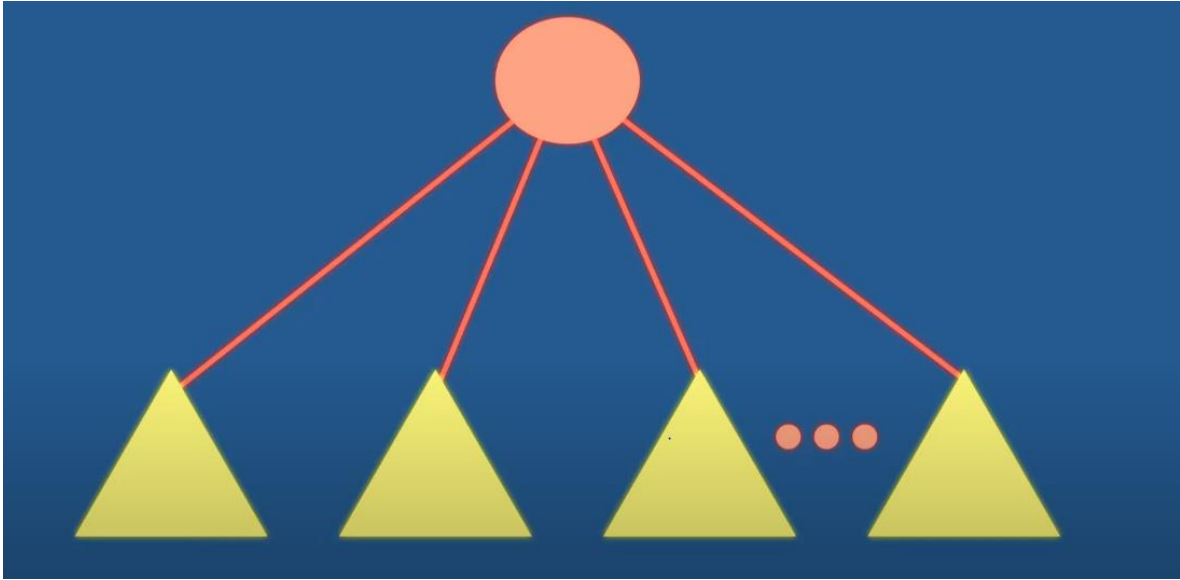
En decision tree se tiene una raíz que es de donde parte el algoritmo y luego los nodos que son llamados hoja



Se debe evaluar que característica puede predecir mejor la etiqueta. (Para esto los datasets debe estar completos, es decir, sin huecos en sus características)

Existe un método Gini, se calcula un coeficiente de Gini, y la característica con menor coeficiente se convierte en el nodo raíz y se vuelve a recalcular para los demás nodos faltantes.

Random forest



Bootstrap dataset, se escogen una cantidad determinada de características y sobre eso se crea una decision tree. El procedimiento se repite n veces.

El problema de random forest es que crea muchos modelos matemáticos, pero para poder seleccionar el mejor, será complicado. Por otro lado, decisión tree te simplifica este problema.

Siempre se inicia por decision tree y luego se pasa a random forest.

Tabla hallada con Matlab 2020b

Método	Accuracy	PCA Accuracy (3/4)
Fine Tree	96.7%	93.3%
Medium Tree	96.7%	93.3%
Coarse Tree	96.7%	92.7%
Linear Discriminant	98.0%	98.7%
Quadratic Discriminant	98.0%	98.0%
Gaussian Naive Bayes	94.7%	92.0%
Kernel Naive Bayes	96.0%	91.3%
Linear SVM	95.3%	98.7%
Quadratic SVM	96.7%	97.3%
Cubic SVM	95.3%	94.7%
Fine Gaussian SVM	94.0%	94.0%
Medium Gaussian SVM	96.7%	97.3%
Coarse Tree	96.0%	95.3%
Fine KNN	95.3%	90.7%
Medium KNN	95.3%	88.0%
Coarse KNN	65.3%	71.3%
Cosine KNN	83.3%	83.3%
Cubic KNN	94.7%	87.3%
Weighted KNN	96.0%	91.3%
Boosted Trees	33.3%	33.3%
Bagged Trees	94.7%	96.7%
Subspace Discriminant	95.3%	96.0%
Subspace KNN	94.0%	94.7%
Rusboosted Trees	33.3%	33.3%

