



APRENDIZAJE AUTOMÁTICO

Ing. Marcelo Monteros

AGENDA

- Introducción al aprendizaje automático y sus aplicaciones en el desarrollo de software.
- Algoritmos de aprendizaje supervisado (regresión, clasificación)
- Algoritmos de aprendizaje no supervisado (agrupamiento, reducción de dimensionalidad).
- Evaluación y selección de modelos de aprendizaje automático.



INTRODUCCIÓN: QUÉ ES EL APRENDIZAJE AUTOMÁTICO

Algoritmos que se ejecutan para aprender automáticamente en base a los datos proporcionados.

Un proceso de inducción del conocimiento.

MÉTODO INDUCTIVO

Se parte de observaciones para llegar a teorías.



Ejemplo:

Observación 1-Se observan los perros

Patrón-Los perros mueven la cola

Teoría-Todos los perros mueven la cola

CATEGORÍAS



Aprendizaje Supervisado:
Problemas de Regresión



Aprendizaje Supervisado:
Problemas de
Clasificación



Aprendizaje No
Supervisado: Problemas
de Agrupamiento



Aprendizaje No
Supervisado: Detección
de Anomalías



APRENDIZAJE SUPERVISADO

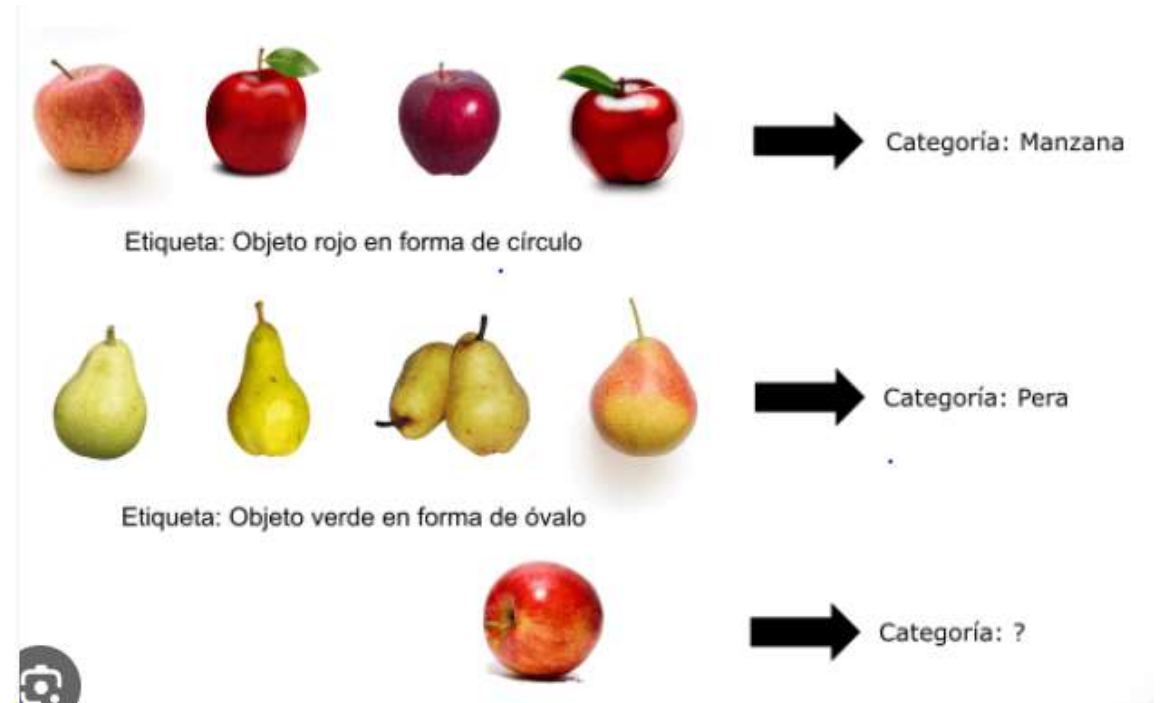


HISTORIA

- Surge a mediados de los años 80 con la aplicación de las redes de neuronas y los árboles de decisión.
- El aprendizaje automático tiene su auge con problemas de predicción complejos donde los modelos estadísticos clásicos no eran muy buenos como, por ejemplo:
 - Reconocimiento de voz
 - Imágenes,
 - Predicción de series temporales no lineales,
 - Predicción de los mercados financieros,
 - Reconocimiento de texto escrito, etc

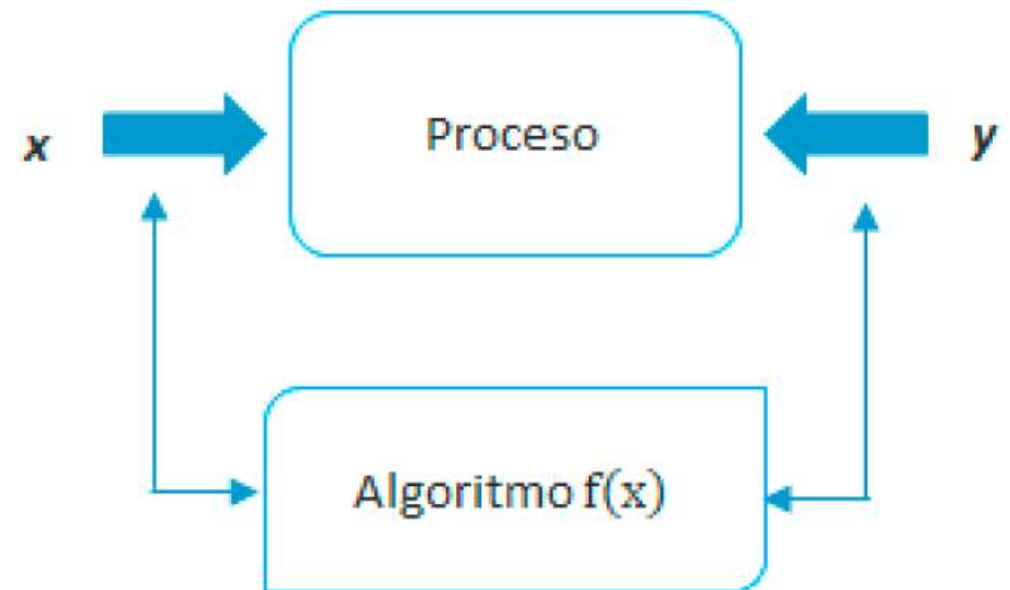
DEFINICIÓN

- El aprendizaje supervisado tiene como objetivo generalizar las respuestas sobre datos no observados, utilizando para ello ejemplos observados previamente

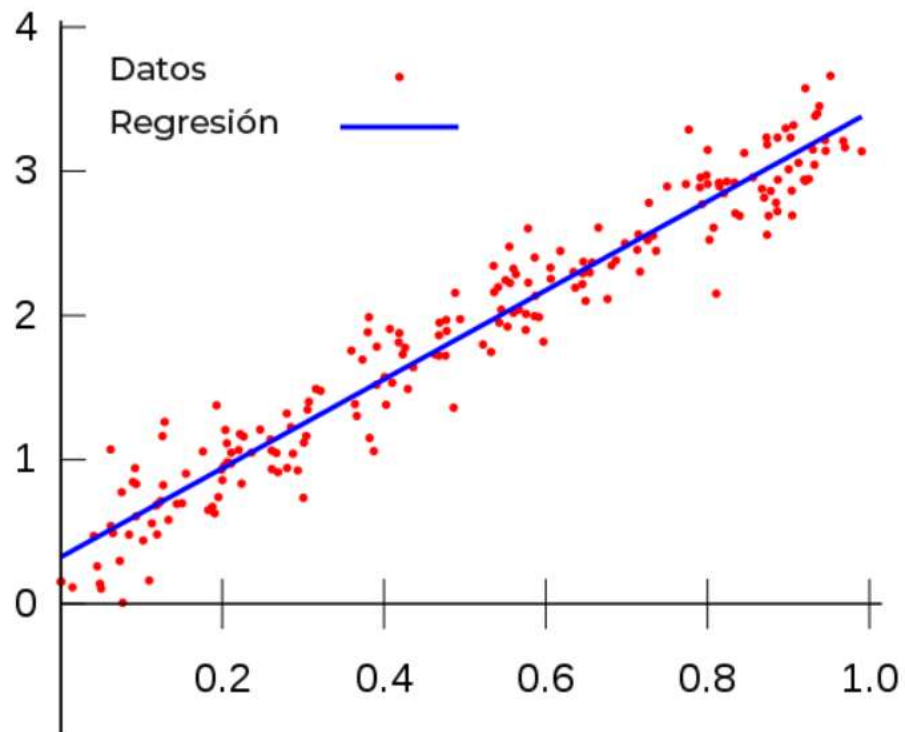


PROBLEMAS DE REGRESIÓN

- El Aprendizaje Automático busca el proceso que relaciona la entrada con la salida
- En el caso de los problemas de regresión la variable respuesta Y que se desea inferir es una variable cuantitativa (numérica continua)
- Se trata el mecanismo de generación de los datos como algo complejo y desconocido
- Es necesario buscar una función $f(x)$ que opere con los datos x para producir las respuestas Y .



PROBLEMAS DE REGRESIÓN



- Cada observación de las variables predictoras x existe una medida de la variable respuesta y .
- En los problemas de regresión la variable respuesta y del sistema que se desea inferir o generalizar es una variable cuantitativa (numérica continua)

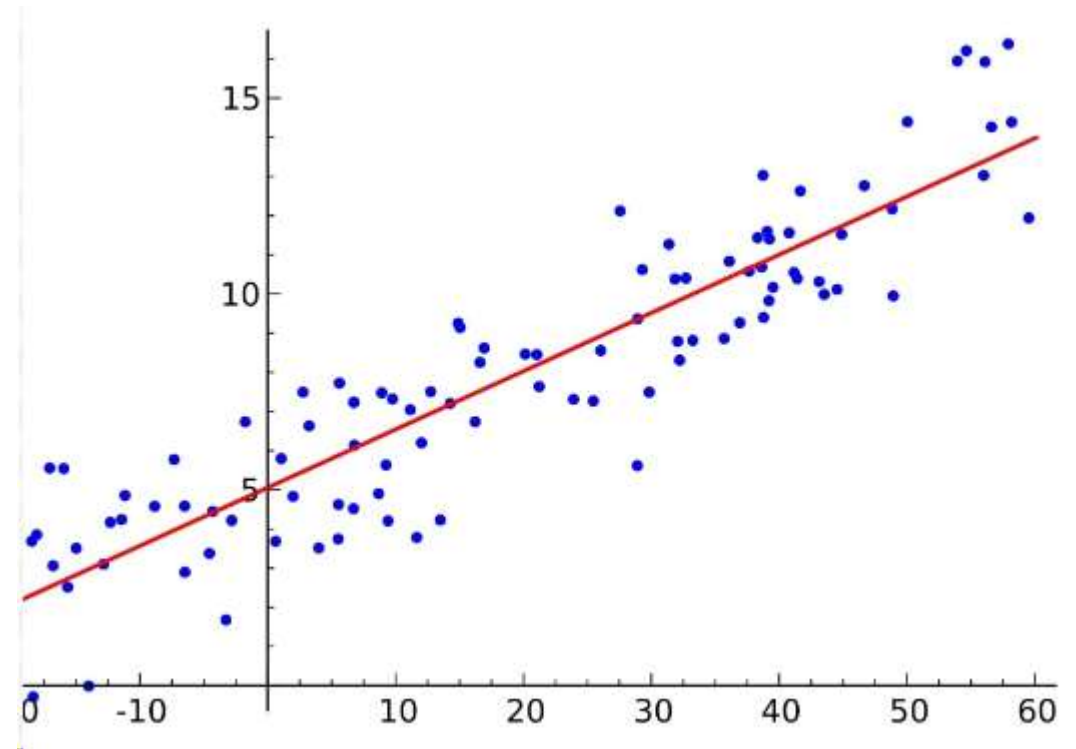
RECUERDE EL OBJETIVO



- Predecir las respuestas que habrá en el futuro con nuevas variables de entrada

EJEMPLOS REGRESIÓN

- Estimar una variable numérica
- Los ingresos de una persona, el precio de venta de un inmueble o algo tan dispar como la fuerza del cemento





APRENDIZAJE SUPERVISADO: PROBLEMAS DE CLASIFICACIÓN



PROBLEMAS DE CLASIFICACIÓN

- En el caso de los problemas de clasificación hay que elegir una de entre varias categorías por medio de ejemplos previamente etiquetados.



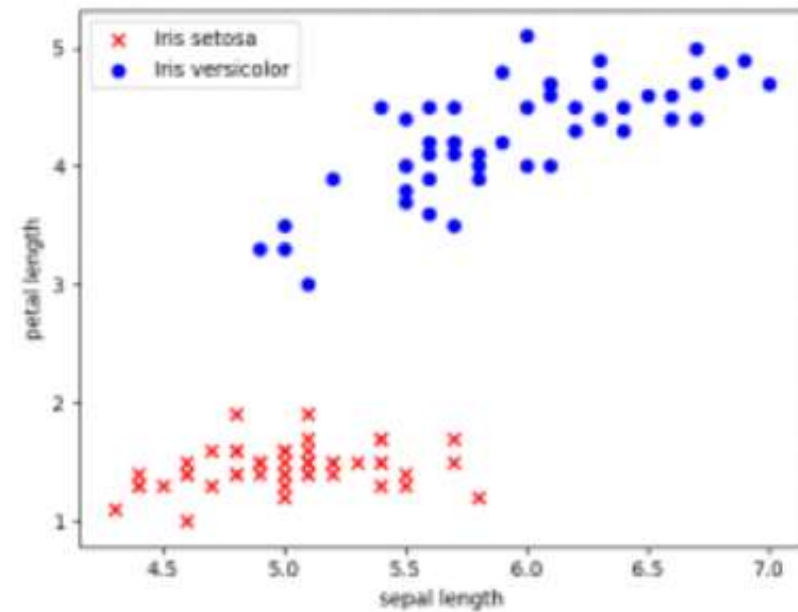
PROBLEMAS DE CLASIFICACIÓN

- Estos problemas se dividen en Clasificación Binaria o Clasificación Multi-Clase.



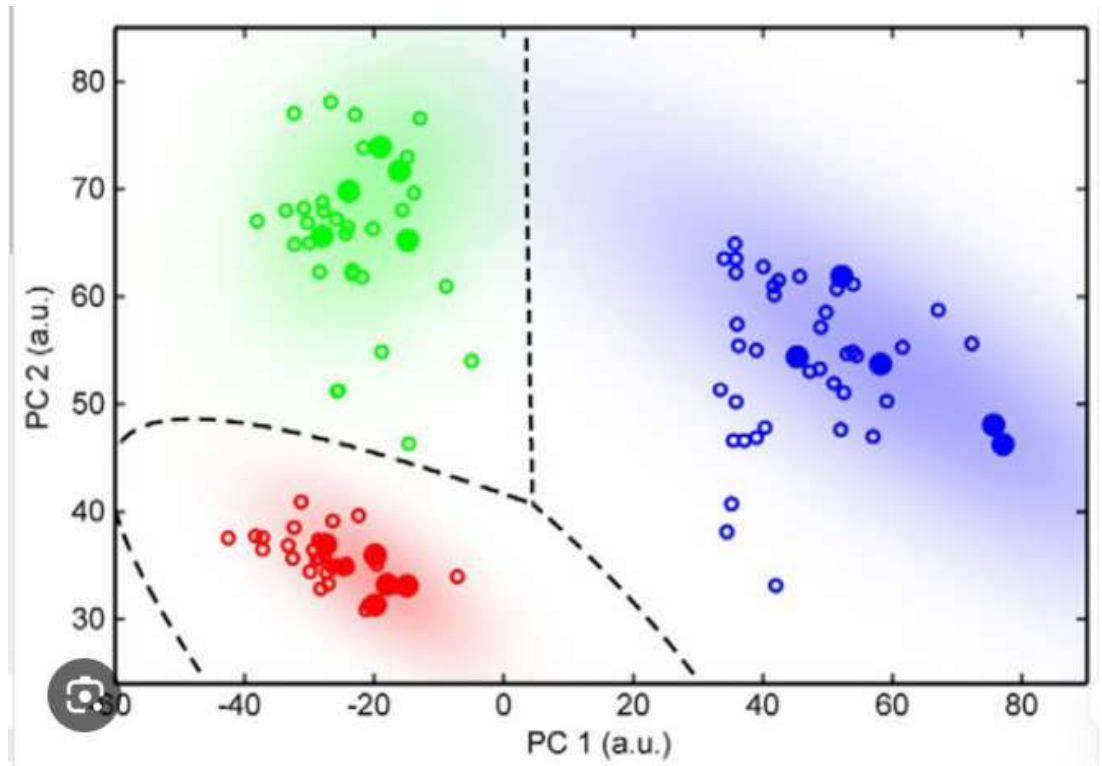
EJEMPLO CLASIFICACIÓN BINARIA

- Un ejemplo de clasificador binario sería un algoritmo para determinar la probabilidad de que un mensaje de correo electrónico sea spam y no spam
- Un algoritmo para predecir el impago de un crédito bancario



EJEMPLO DE CLASIFICADOR MULTICLASE

- Algoritmo para predecir en qué país de doce posibles un cliente va a realizar la siguiente reserva



PROPÓSITO

- Adelantarte a situaciones futuras

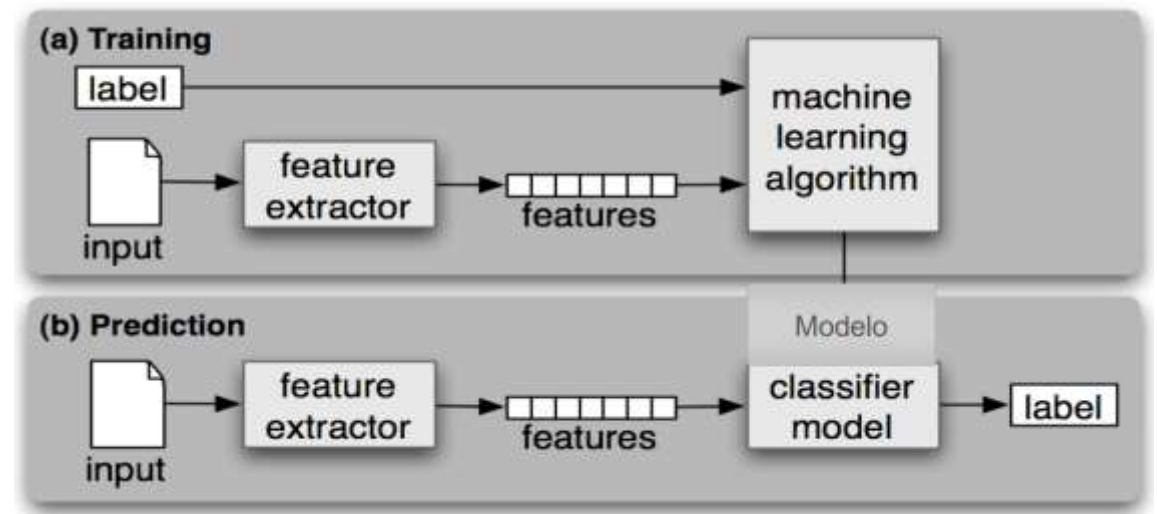


TERMINOLOGÍA

- Variable respuesta: también conocida como outcome, variable dependiente, variable objetivo, target, class, etc. Se suele denotar por y .
- Vector de p mediciones predictoras llamado x : también conocido como inputs, regressors, features, variables, variables independientes, etc.
- Se tienen datos de entrenamiento conocidos como training data: $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$ que son observaciones, ejemplos o instancias de cada una de las medidas de entrada.

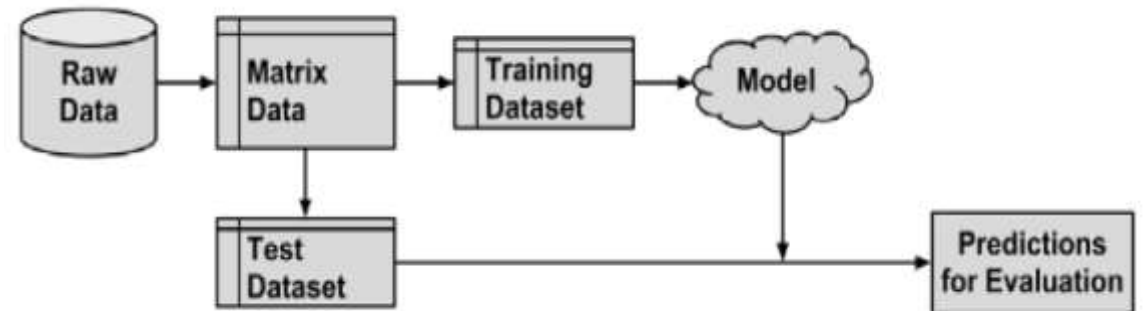
CONJUNTOS DE ENTRENAMIENTO

- **Clave:** Generalizar situaciones del futuro en función de los datos observados.
- Existen métricas de error de los modelos con datos de entrenamiento. (Optimistas)



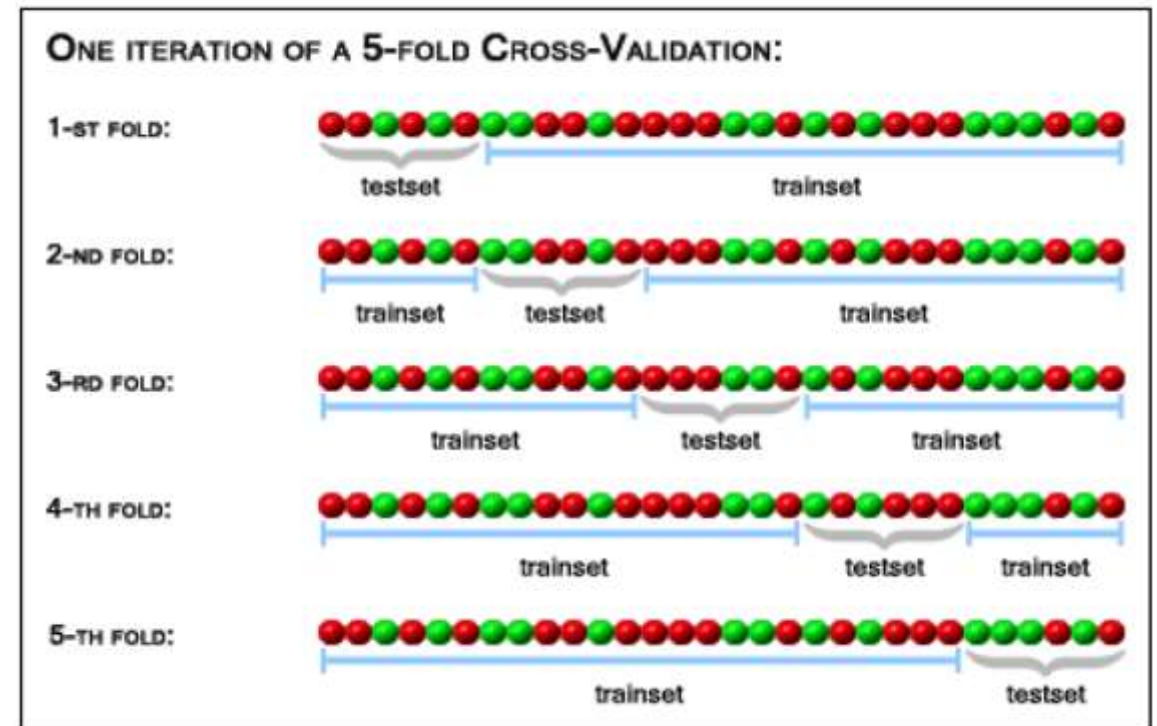
CONJUNTOS DE TEST

- Un modelo puede tener un error muy bajo en entrenamiento y no ser capaz de predecir bien los datos futuros
- Una métrica más realista y objetiva es obtenida con un conjunto de test
- Este procedimiento se conoce con el nombre de hold-out. Normalmente se utiliza un 80-20



VALIDACIÓN CRUZADA

- La repetición de la técnica de Hold-out es la base para la validación cruzada.
- Esta técnica también se conoce como K-Fold, donde K indica el número de conjuntos posibles





APRENDIZAJE NO SUPERVISADO



TÉCNICAS AGRUPAMIENTO APRENDIZAJE NO SUPERVISADO

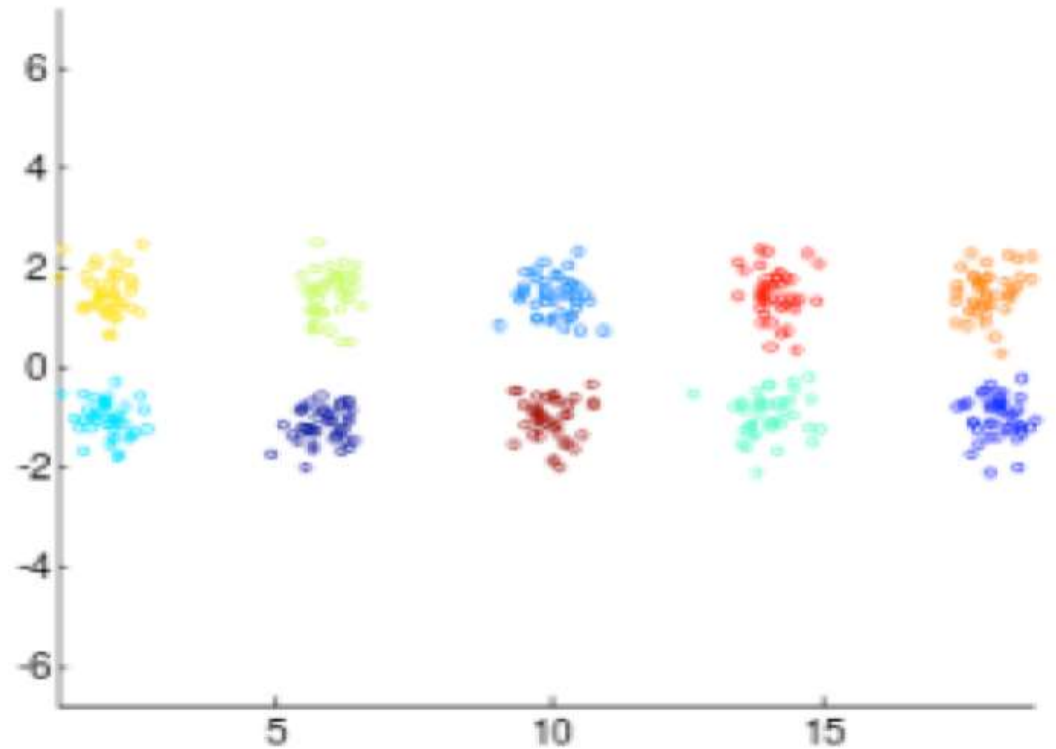
Una agrupación es un grupo de datos que son similares entre sí en función de su relación con los datos circundantes. El agrupamiento se utiliza para cosas tales como la ingeniería de características o el descubrimiento de patrones.



El uso de un algoritmo de agrupamiento significa que le darás al algoritmo una gran cantidad de datos de entrada sin etiquetas y te permitirá encontrar cualquier agrupación en los datos.

CUANDO SE USA

- Se desconoce la clase, etiqueta o respuesta de las instancias de entrenamiento.
- No se tiene conocimiento de las categorías de los datos y se desea buscar la estructura oculta en los datos.
- La evaluación de estas técnicas no es sencilla (falta de ground-truth)



TÉCNICAS DE DETECCIÓN DE ANOMALÍAS

- Una anomalía es una observación que es significativamente diferente del resto de observaciones.
- Es sospechosa de haber sido generada por un mecanismo diferente del resto de observaciones.
- Estos algoritmos también son conocidos con el nombre de detección de Outliers.
- Aplicación: fraude, detección de intrusos, detección defectos estructurales, problemas médicos, etc
- Detección de anomalías supervisada y no supervisada.



GRACIAS

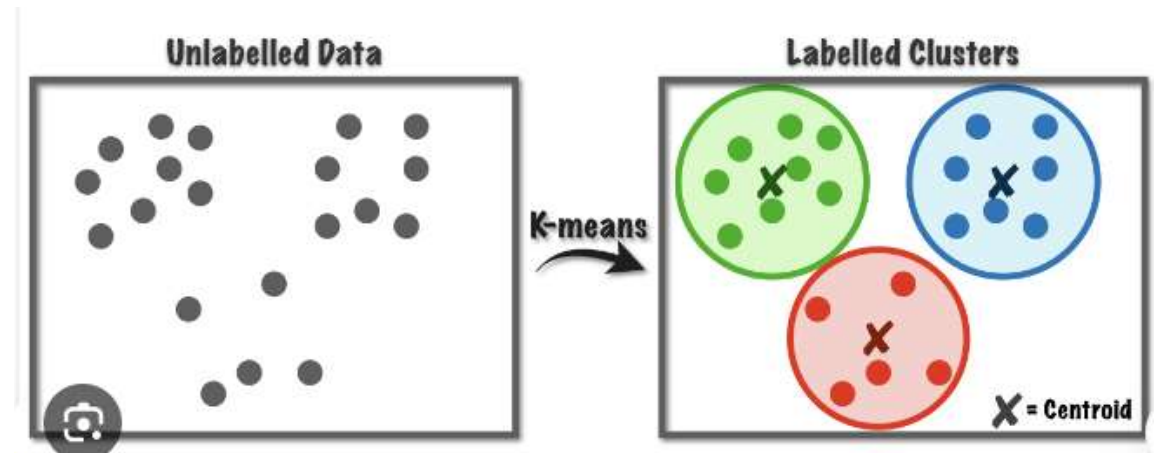
ALGUIEN@EJEMPLO.COM



APRENDIZAJE NO SUPERVISADO

INTRODUCCIÓN AL CLUSTERING

- El clustering es una técnica de aprendizaje automático que divide los datos en clusters o grupos similares
- Las instancias dentro de un grupo deben ser muy similares, pero muy distintas entre los grupos



APLICACIONES DEL CLUSTERING

- Segmentar Clientes entre grupos con patrones de compra similares
- Detectar comportamiento anómalo, identificando patrones que caen fuera de los clusters habituales
- Simplificar o resumir datasets muy grandes, agrupando usuarios con similares características



MEDIDAS DE SIMILITUD

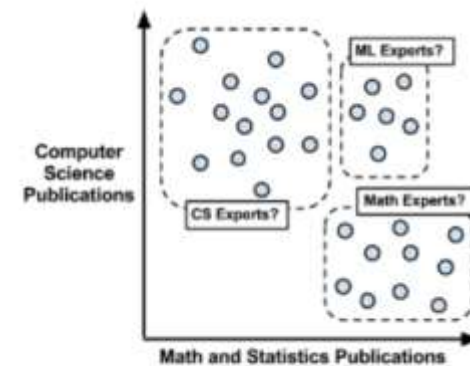
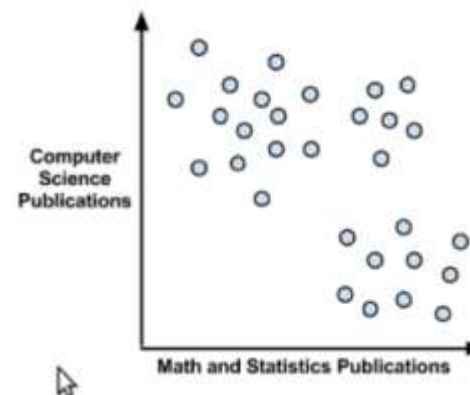
Names	Formula
Euclidean distance	$\ a - b\ _2 = \sqrt{\sum_i (a_i - b_i)^2}$
Squared Euclidean distance	$\ a - b\ _2^2 = \sum_i (a_i - b_i)^2$
Manhattan distance	$\ a - b\ _1 = \sum_i a_i - b_i $
maximum distance	$\ a - b\ _\infty = \max_i a_i - b_i $
Mahalanobis distance	$\sqrt{(a - b)^\top S^{-1} (a - b)}$ where S is the Covariance matrix

MEDIDAS DE SIMILITUD

- Investigar y explicar cada medida de similitud y su aplicación

EJEMPLO DE CLUSTERING

- En el clustering, instancias sin etiquetar son proporcionadas con el objetivo de inferir las relaciones en los datos



TIPOS DE ALGORITMOS DE CLUSTERING

Agrupación:

- Tienen definidos previamente un número de grupos. Son algoritmos iterativos que comienzan con una asignación inicial y se van modificando siguiendo un criterio de optimización.

Jerárquicos:

- En cada iteración solo un objeto cambia de grupo y los grupos están anidados en los de pasos anteriores. Si un objeto ha sido asignado a un grupo ya no se vuelve a cambiar



KMEDIAS ALGORITMO



K-MEDIAS



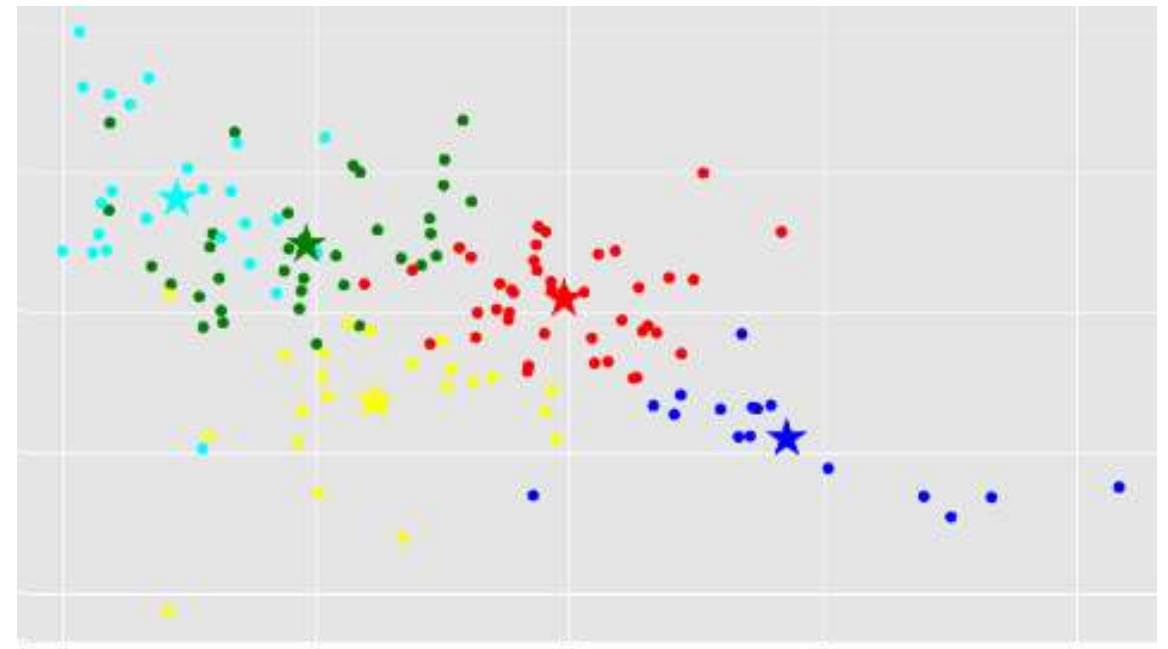
Es el algoritmo de clustering más utilizado



Entendiendo su funcionamiento podemos entender casi cualquier algoritmo de clustering utilizando hoy en día

K-MEDIAS

- El algoritmo consiste en asignar cada uno de los n ejemplos a uno de los k clusters, donde k es un número definido previamente.
- El objetivo es minimizar las diferencias entre los grupos de cada cluster y maximizar las diferencias entre clusters.
- A menos que k y n sean extremadamente pequeños no es factible calcular los grupos óptimos entre todas las combinaciones posibles de ejemplos. En su lugar el algoritmo utiliza un proceso heurístico para calcular la solución óptima



K-MEDIAS



El Algoritmo comprende 2 fases



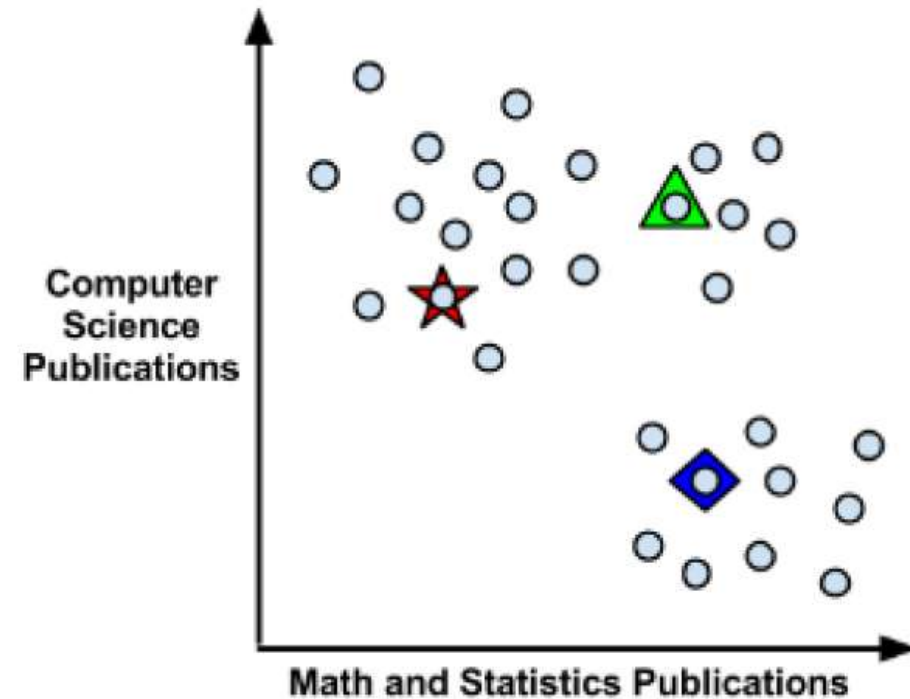
Primero asigna ejemplos a un conjunto inicial de k clusters. Después actualiza las asignaciones ajustando los límites de los grupos de acuerdo con los ejemplos de cada cluster.



Este proceso de asignación y actualización ocurre varias veces hasta que los cambios no proporcionan mejoras en los clusters

K-MEDIAS

- Debido a la naturaleza heurística del algoritmo los resultados pueden ser distintos en función de la inicialización del algoritmo.
- K-Medias considera que los valores de las variables son coordenadas en un espacio multi-dimensional
- El algoritmo empieza eligiendo los k puntos iniciales.
- Para los puntos iniciales se suelen elegir 3 ejemplos al azar.



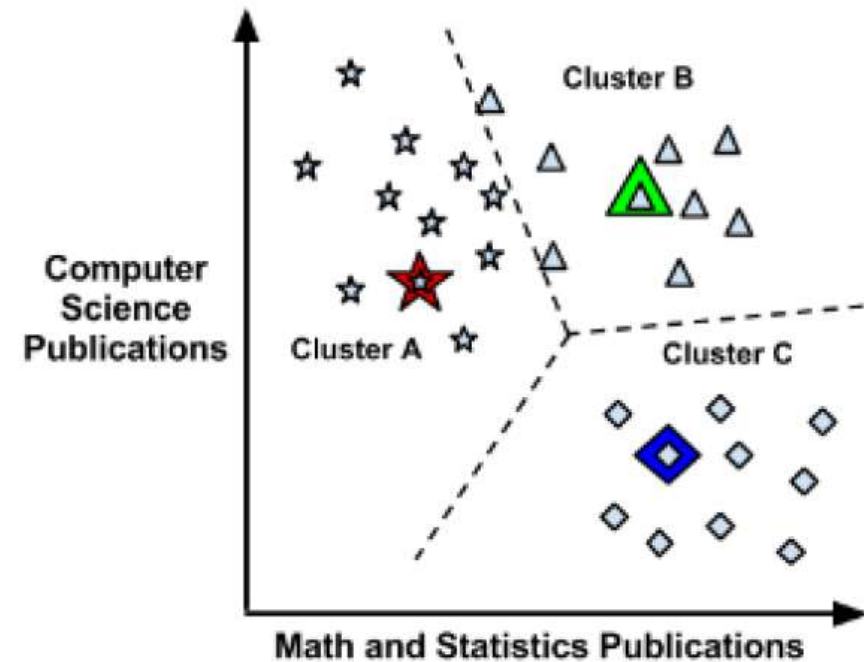
CENTROIDE

- Después de elegir los puntos iniciales, los otros ejemplos se asignan al centroide del cluster más cercano de acuerdo a la función de distancia. Esta función suele ser la distancia Euclídea.

$$\text{dist}(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

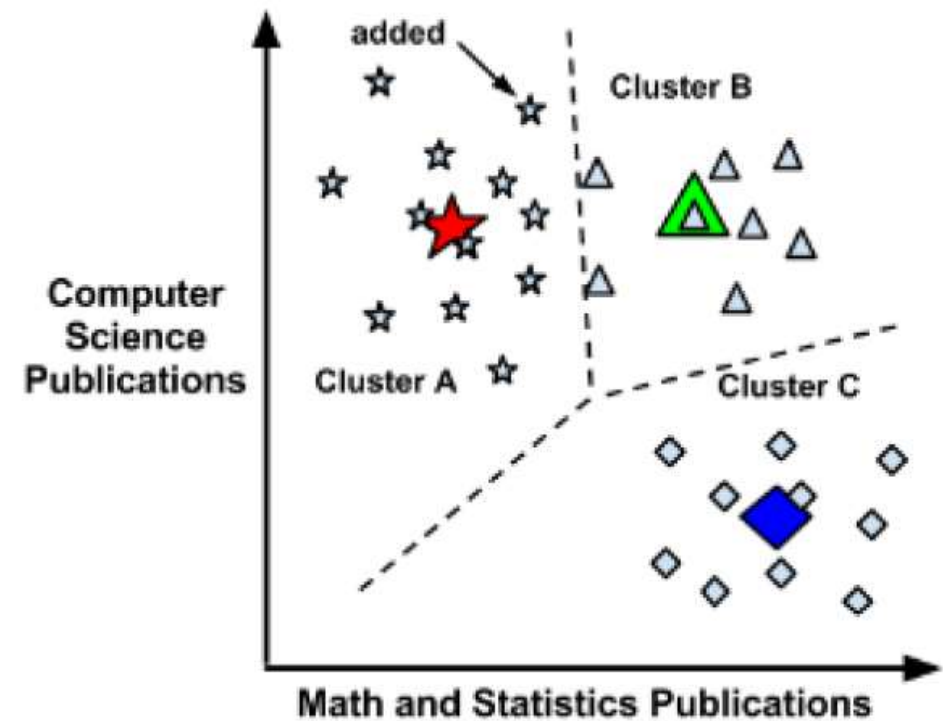
K-MEDIAS

- Debido a que estamos utilizando distancias, todos los puntos deben de ser numéricos y normalizados antes de
- utilizarlos.



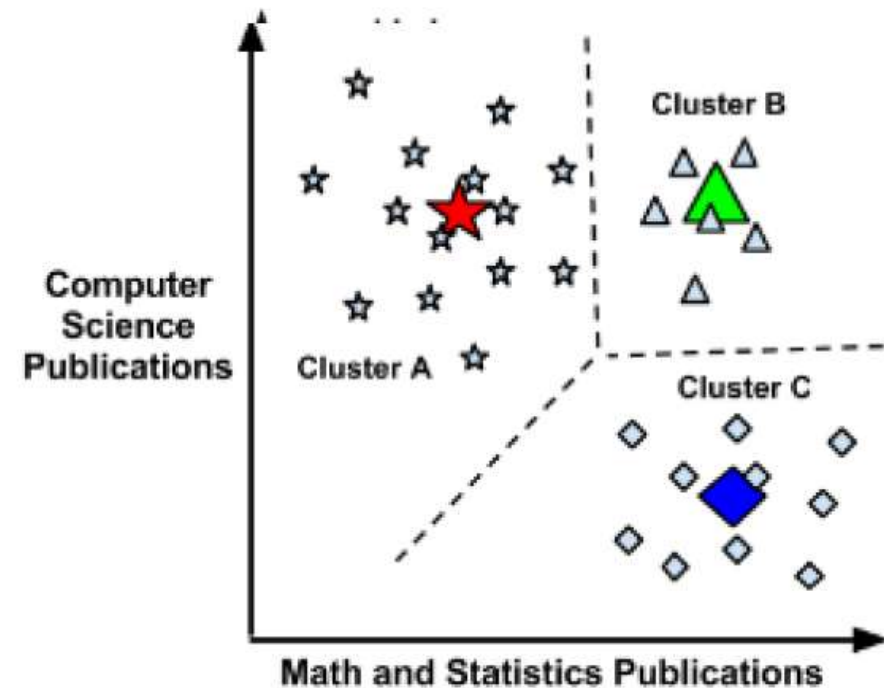
K-MEDIAS (ACTUALIZACIÓN)

- El primer paso de la actualización conlleva mover los centroides iniciales a una nueva posición, calculada como la media de los puntos asignados al cluster



K-MEDIAS

- Como los límites de los centroides se han modificado es muy posible que haya que reasignar instancias a otros clusters.



K-MEDIAS: ELEGIR K

- Elegir un buen valor requiere de cierto balance, Un número muy grande mejora la homogeneidad pero a la vez sobre-ajusta los datos
- Idealmente existe un conocimiento a-priori sobre el número de grupos apropiado. Por ejemplo, si estamos agrupando películas se pueden agrupar por los géneros existentes. Otras veces el número de clusters viene dado por los requisitos de negocio.
- Sin conocimiento apriori se suele elegir $K = \sqrt{n/2}$

EJERCICIO

- Aplicar e interpretar los resultados de un algoritmo de Clustering aplicado sobre el conjunto de datos. Se recomienda la aplicación del algoritmo K-MEANS
- **Definición de columnas del conjunto de datos**
 1. FRESH: annual spending (m.u.) on fresh products (Continuous)
 2. MILK: annual spending (m.u.) on milk products (Continuous);
 3. GROCERY: annual spending (m.u.) on grocery products (Continuous);
 4. FROZEN: annual spending (m.u.) on frozen products (Continuous)
 5. DETERGENTS_PAPER: annual spending (m.u.) on detergents and paper products (Continuous)
 6. DELICATESSEN: annual spending (m.u.) on and delicatessen products (Continuous);
 7. CHANNEL: customers Channel - Horeca (Hotel/Restaurant/Café) or Retail channel (Nominal)
 8. REGION: customers Region Lisbon, Oporto or Other (Nominal)

METODOLOGÍA

```
# importar librerías necesarias
import pandas as pd
import numpy as np

import warnings
warnings.filterwarnings("ignore")

from sklearn.cluster import KMeans
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import pairwise_distances_argmin_min

import pydot
from prettytable import PrettyTable

import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
from mpl_toolkits.mplot3d import Axes3D
plt.rcParams['figure.figsize'] = (16, 9)
plt.style.use('ggplot')
```

RECUPERACIÓN DE DATOS PASO I

```
datos = pd.read_csv('Wholesale customers data.csv')
```

```
datos.head()
```

	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
0	2	3	12669	9656	7561	214	2674	1338
1	2	3	7057	9810	9568	1762	3293	1776
2	2	3	6353	8808	7684	2405	3516	7844
3	1	3	13265	1196	4221	6404	507	1788
4	2	3	22615	5410	7198	3915	1777	5185

ANÁLISIS DESCRIPTIVO PASO 2

- El análisis descriptivo se lo realiza usando las funciones estadísticas de percentiles, mediana y desviación estándar

```
# Se realiza el análisis descriptivo de los datos, se verifica un valor elevado del valor máximo respecto a la media
# Se establece que los canales son datos categóricos entre 1 y 2. La región también se considera datos categóricos
#entre 1 y 3.
datos.describe()
```

	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
count	440.000000	440.000000	440.000000	440.000000	440.000000	440.000000	440.000000	440.000000
mean	1.322727	2.543182	12000.297727	5796.265909	7951.277273	3071.931818	2881.493182	1524.870455
std	0.468052	0.774272	12647.328865	7380.377175	9503.162829	4854.673333	4767.854448	2820.105937
min	1.000000	1.000000	3.000000	55.000000	3.000000	25.000000	3.000000	3.000000
25%	1.000000	2.000000	3127.750000	1533.000000	2153.000000	742.250000	256.750000	408.250000
50%	1.000000	3.000000	8504.000000	3627.000000	4755.500000	1526.000000	816.500000	965.500000
75%	2.000000	3.000000	16933.750000	7190.250000	10655.750000	3554.250000	3922.000000	1820.250000
max	2.000000	3.000000	112151.000000	73498.000000	92780.000000	60869.000000	40827.000000	47943.000000

ANÁLISIS DESCRIPTIVO PASO 3

```
# Verificar valores missing, se verifica que no hay existencias
print(datos.isnull().any())
```

```
Channel      False
Region       False
Fresh        False
Milk         False
Grocery      False
Frozen       False
Detergents_Paper  False
Delicassen   False
dtype: bool
```

```
# Visualizamos los datos acorde al canal y región que son los clientes Channel - Horeca (Hotel/Restaurant/Caf  ) or Retail ch
# Regi  n que son Lisbon, Oporto or Other (Nominal)

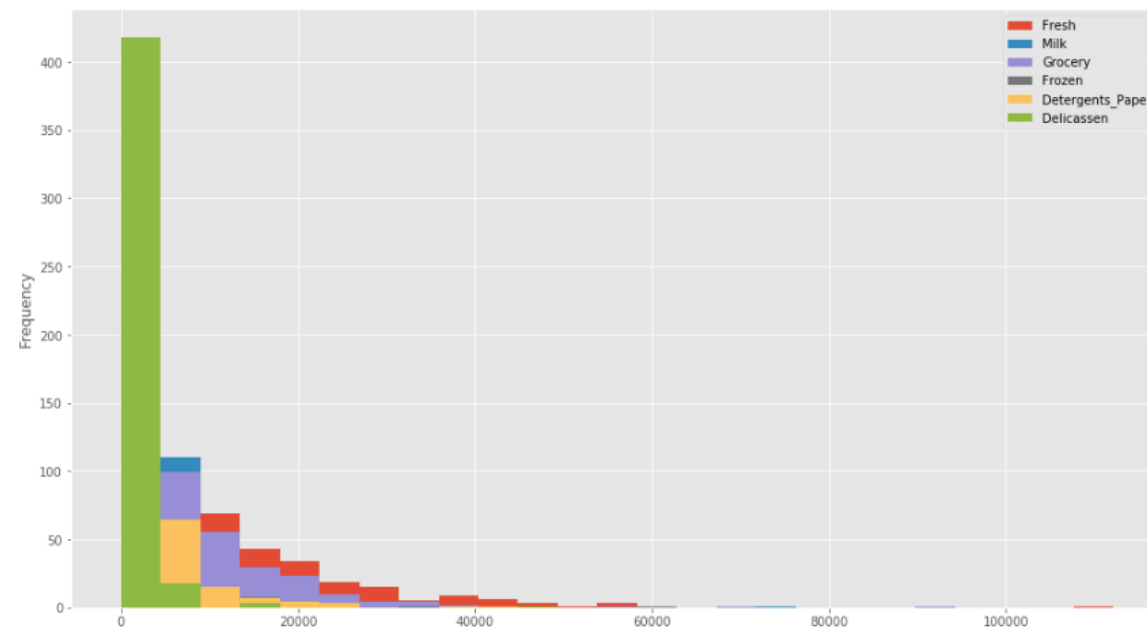
datos.groupby(['Channel','Region']).size()
```

```
5]: Channel  Region
1          1      59
          2      28
          3     211
2          1      18
          2      19
          3     105
dtype: int64
```


TRATAMIENTO DE LOS DATOS PASO 4

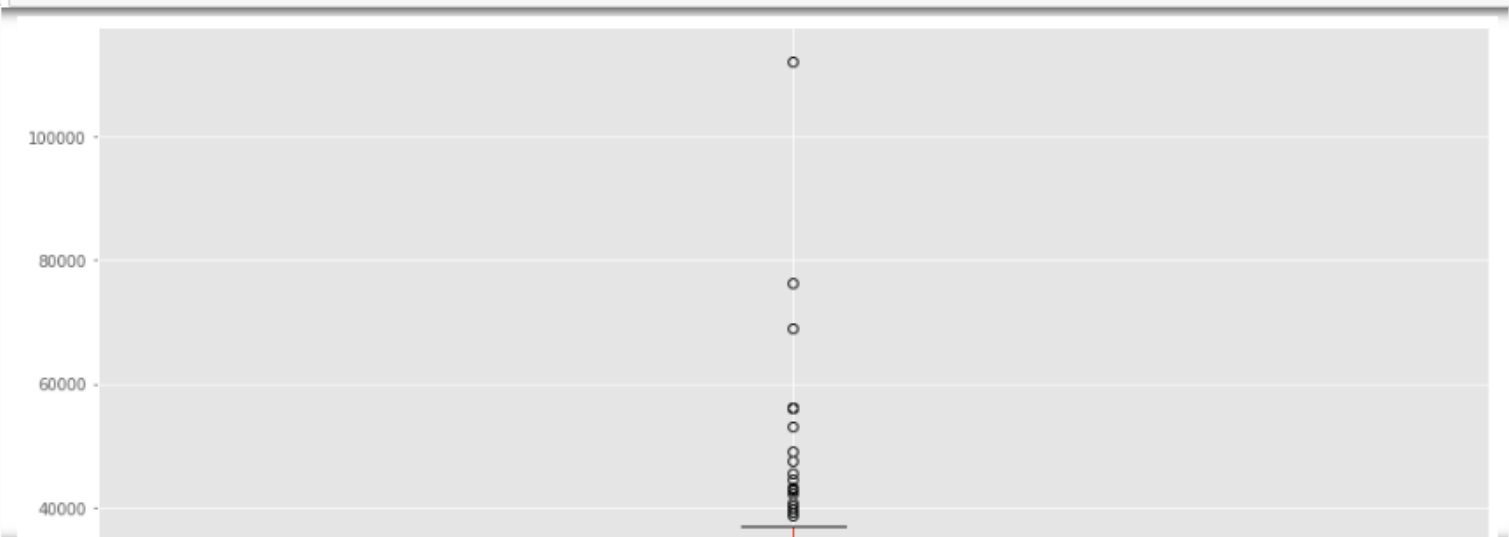
```
#Se analiza la dispersión de los datos y se observa que tienen  
# poca dispersión  
datos.drop(['Channel', 'Region'], axis=1).plot.hist(bins=25)
```

```
: <matplotlib.axes._subplots.AxesSubplot at 0x199d3192f88>
```



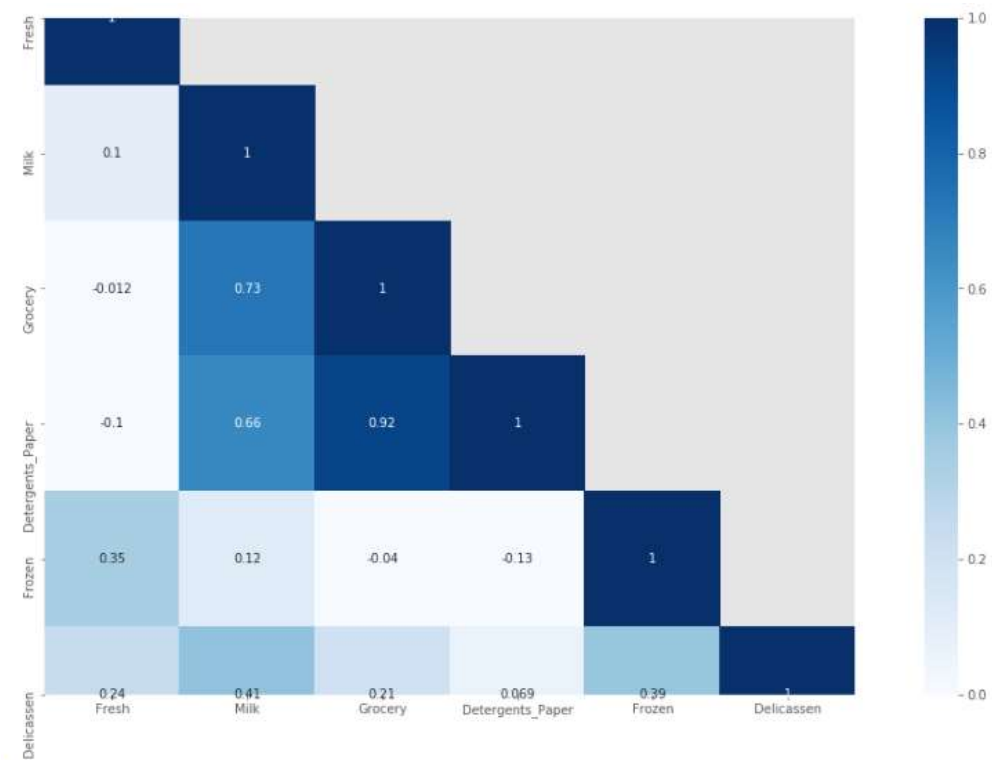
TRATAMIENTO DE LOS DATOS PASO 4

```
# Se verifica mediante boxplot los datos con graficas de bloque para observar outliers,  
#si bien existen datos alejados de la mediana, no se los considera outliers para ser eliminados  
variables_numericas=['Fresh','Milk','Grocery','Detergents_Paper','Frozen','Delicassen' ]  
  
for columna in variables_numericas:  
    plt.figure()  
    datos.boxplot(column=columna)  
    plt.show()
```



TRATAMIENTO DE LOS DATOS PASO 4

```
# Construir matriz de correlacion, se observa una fuerte correlación entre Detergents_paper y Grocery
matrix = datos[variables_numericas].corr()
heat = np.array(matrix)
heat[np.tril_indices_from(heat)] = False
# Construir mapa de calor
fig,ax=plt.subplots()
fig.set_size_inches(20,10)
sns.heatmap(matrix, mask=heat,vmax=1.0, vmin=0.0, square=True,annot=True, cmap="Blues")
```



TRATAMIENTO DE LOS DATOS PASO 4

```
# Se define eliminar las variable correlacionada, Detergents_paper. No consideramos Grocery como variable candidata a ser #eliminada
```

```
datos_procesados=datos.drop(['Detergents_Paper'],axis=1)  
print(datos_procesados)
```

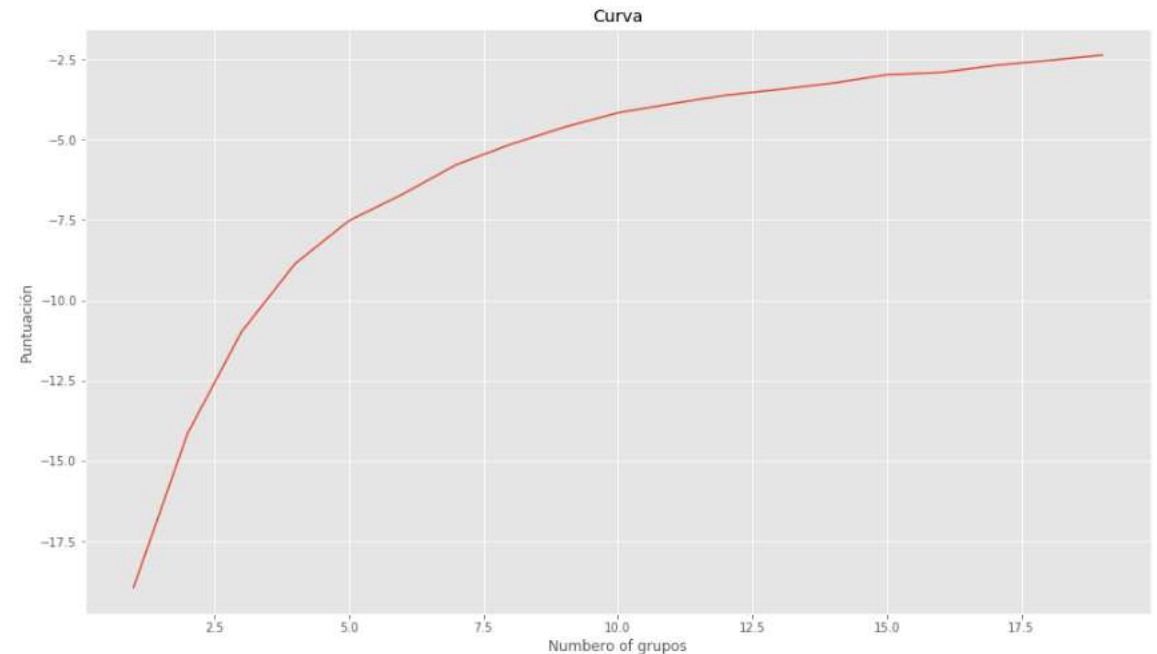
	Channel	Region	Fresh	Milk	Grocery	Frozen	Delicassen
0	2	3	12669	9656	7561	214	1338
1	2	3	7057	9810	9568	1762	1776
2	2	3	6353	8808	7684	2405	7844
3	1	3	13265	1196	4221	6404	1788
4	2	3	22615	5410	7198	3915	5185
..
435	1	3	29703	12051	16027	13135	2204
436	1	3	39228	1431	764	4510	2346
437	2	3	14531	15488	30243	437	1867
438	1	3	10290	1981	2232	1038	2125
439	1	3	2787	1698	2510	65	52

```
[440 rows x 7 columns]
```

DETERMINA EL TAMAÑO DE K, UTILIZANDO LA TÉCNICA DEL CODO PASO 5

```
# Se determina el tamaño de K, utilizando la técnica del codo,  
# aquí se considera la distancia media entre los puntos de datos y su centroid.  
# Acorde a lo indicado y a la grafica se toma el valor de 3 clusters
```

```
Nc = range(1, 20)  
kmeans = [KMeans(n_clusters=i) for i in Nc]  
#kmeans  
score = [kmeans[i].fit(X).score(X) for i in range(len(kmeans))]  
#score  
plt.plot(Nc,score)  
plt.xlabel('Numero of grupos')  
plt.ylabel('Puntuación')  
plt.title('Curva')  
plt.show()
```



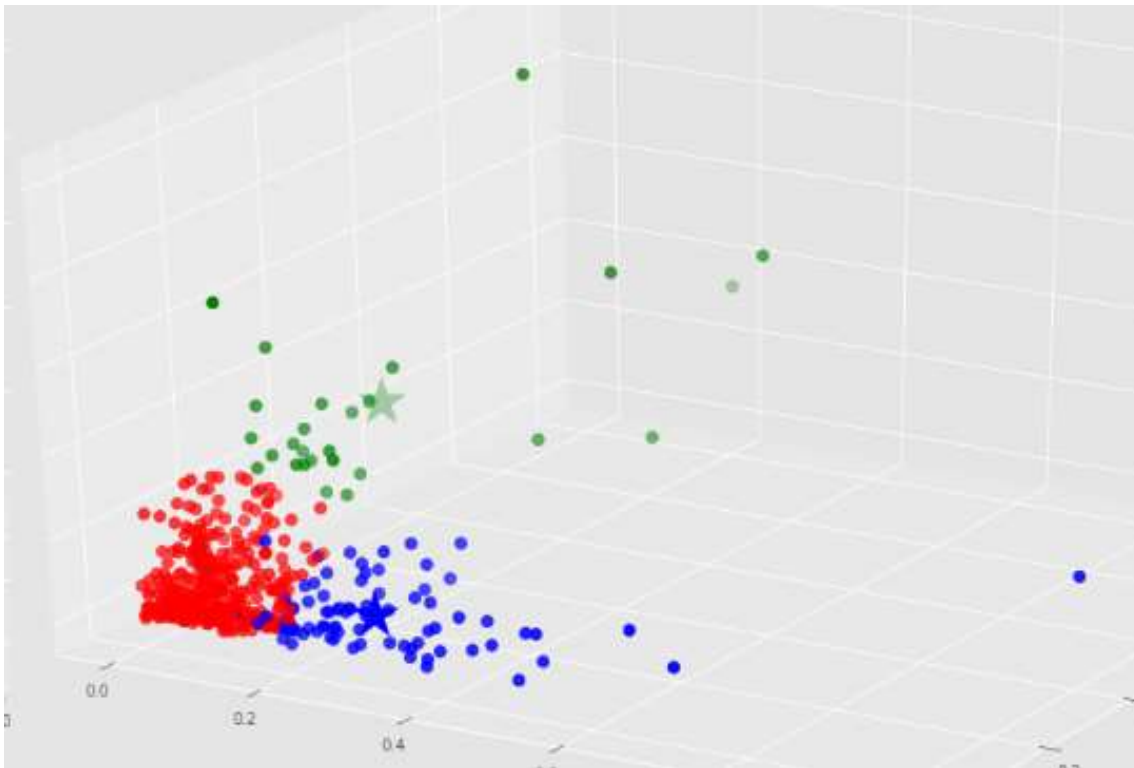
PREDICCIÓN PASO 6

```
▶ # Se realiza las predicciones para establecer los grupos
labels = kmeans.predict(X)
# Getting the cluster centers
C = kmeans.cluster_centers_
```

```
array([[0.06323635, 0.05963813, 0.06949032, 0.03580441, 0.02231855],
       [0.1114767 , 0.35357932, 0.36244216, 0.06158677, 0.10275962],
       [0.27685015, 0.0622449 , 0.06008198, 0.10228246, 0.04584078]])
```

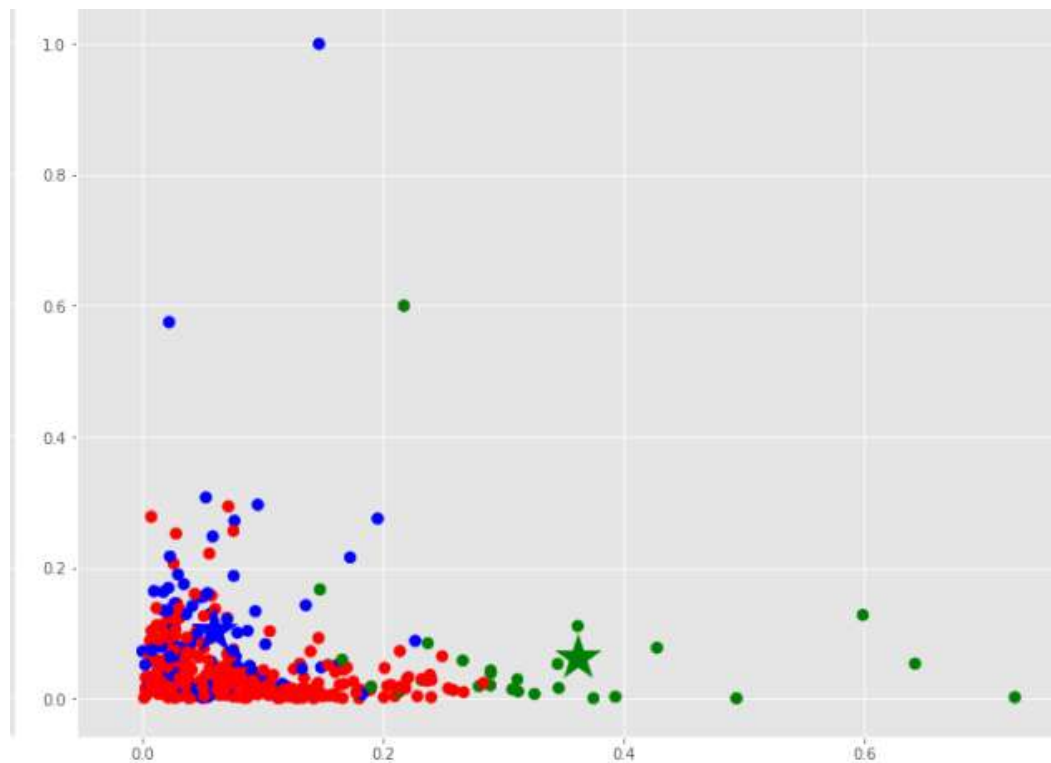
```
datos_numericos=['Fresh','Milk','Grocery','Frozen','Delicassen']
datos_categoria=['Channel','Region']
```

VISUALIZACIÓN



```
#Se grafica en 3D los grupos que contengan  
colores=['red','green','blue']  
asignar=[]  
for row in labels:  
    asignar.append(colores[row])  
  
fig = plt.figure()  
ax = Axes3D(fig)  
ax.scatter(X[:, 0], X[:, 1], X[:, 2], c=asignar)  
ax.scatter(C[:, 0], C[:, 1], C[:, 2], marker='x')
```

VISUALIZACIÓN



```
#Se grafican 'Grocery' y 'Frozen' en 2D  
plt.scatter(X[:, 2], X[:, 3], c=asignar, s=60)  
plt.scatter(C[:, 2], C[:, 3], marker='*', c=colores, s=1000)
```




GRACIAS

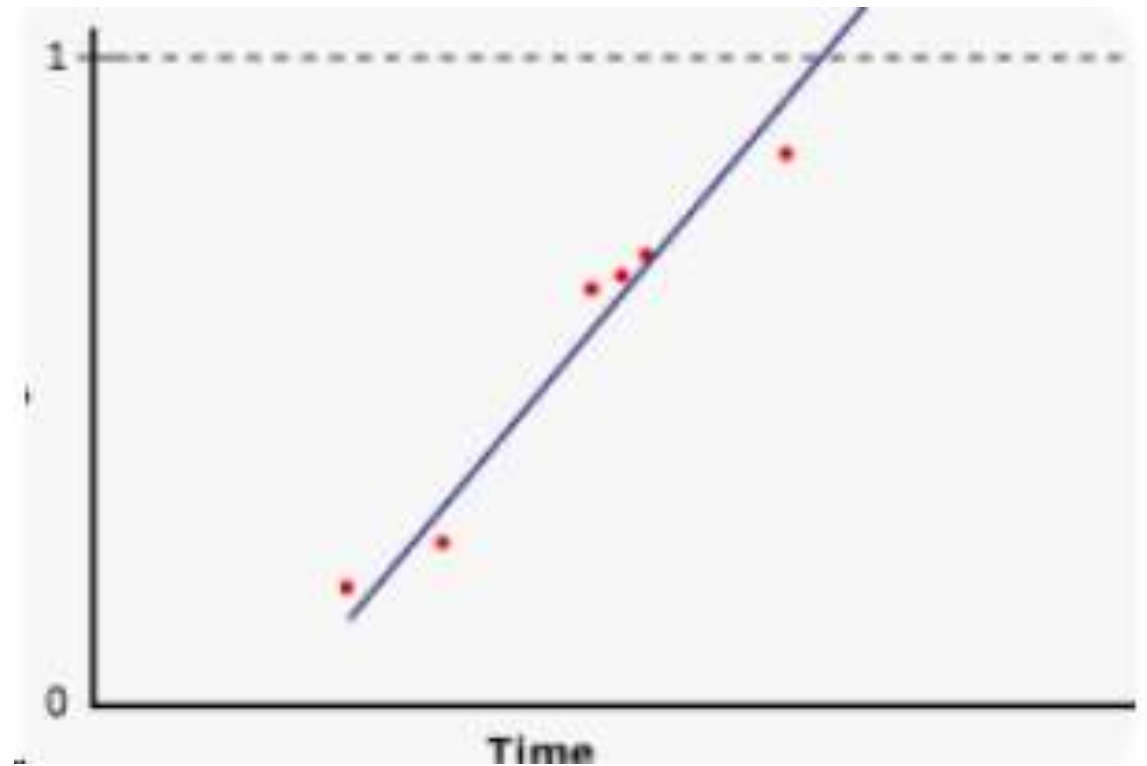
ALGUIEN@EJEMPLO.COM



ALGORITMOS DE REGRESIÓN

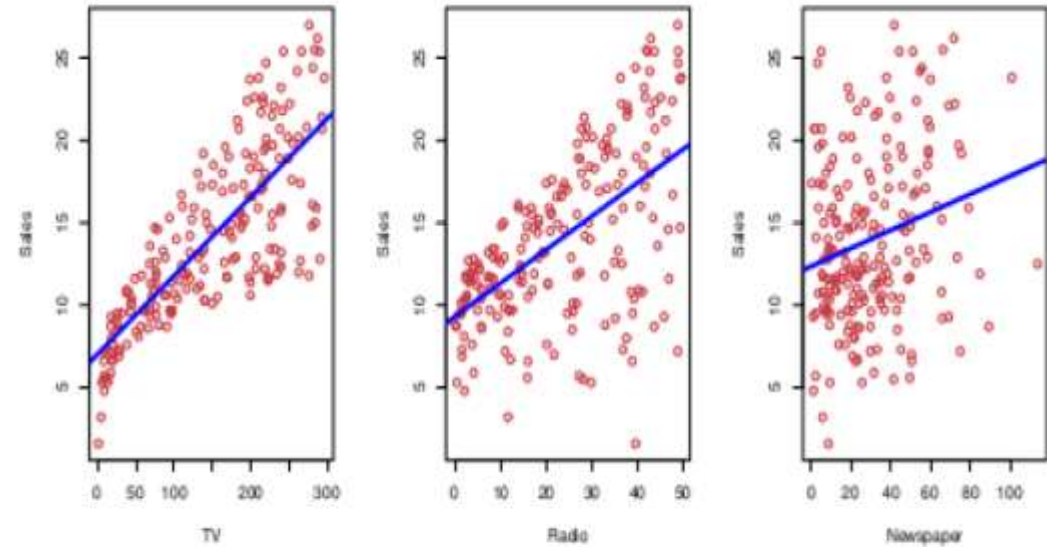
DEFINICIÓN

Los algoritmos de regresión son un tipo concreto de algoritmos de aprendizaje supervisado y consisten en realizar una predicción de una variable numérica o cuantitativa



PREDICCIÓN DE VENTAS

- El gráfico muestra la distribución de las ventas en función del gasto publicitario en TV, radio o prensa escrita.



$$\text{Sales} \approx f(\text{TV}, \text{Radio}, \text{Newspaper})$$

$$X = \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix} \quad Y = f(X) + \epsilon$$

REGRESIÓN: F(X) IDEAL

- ▶ ¿Cuál es el valor de Y cuando X=4?

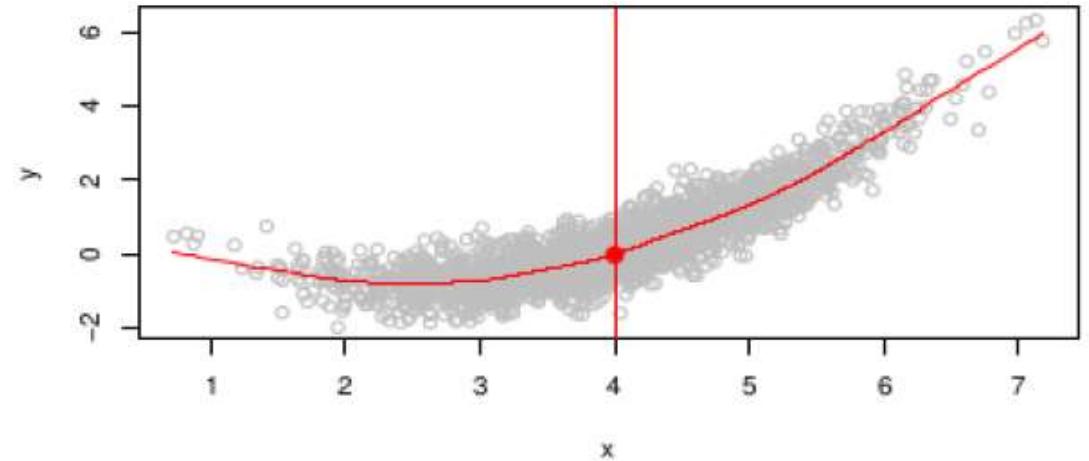
$$f(4) = E(Y|X = 4)$$

- ▶ Al f(x) ideal se le conoce como función de regresión

$$f(x) = E(Y|X = x)$$

- ▶ También esta definida para un vector de **X**

$$f(x) = f(x_1, x_2, x_3) = E(Y|X_1 = x_1, X_2 = x_2, X_3 = x_3)$$

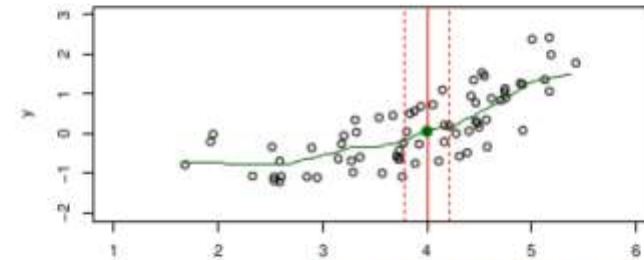


CÓMO ESTIMAR EL $F(X)$ IDEAL?

- Normalmente tendremos pocos o ningún punto en $X=4$
- Por tanto, no se puede calcular $E(Y|X=x)$
- La solución es relajar la definición:
 - P : número características
 - N : número de ejemplos

$$\hat{f}(x) = \text{Ave}(Y|X \in \mathcal{N}(x))$$

where $\mathcal{N}(x)$ is some *neighborhood* of x .



Nota: Estimar los puntos utilizando **Nearest-neighbor averaging** funciona bien para problemas con p pequeña (i.e. $p < 5$) y N muy grande. (Curse of dimensionality)

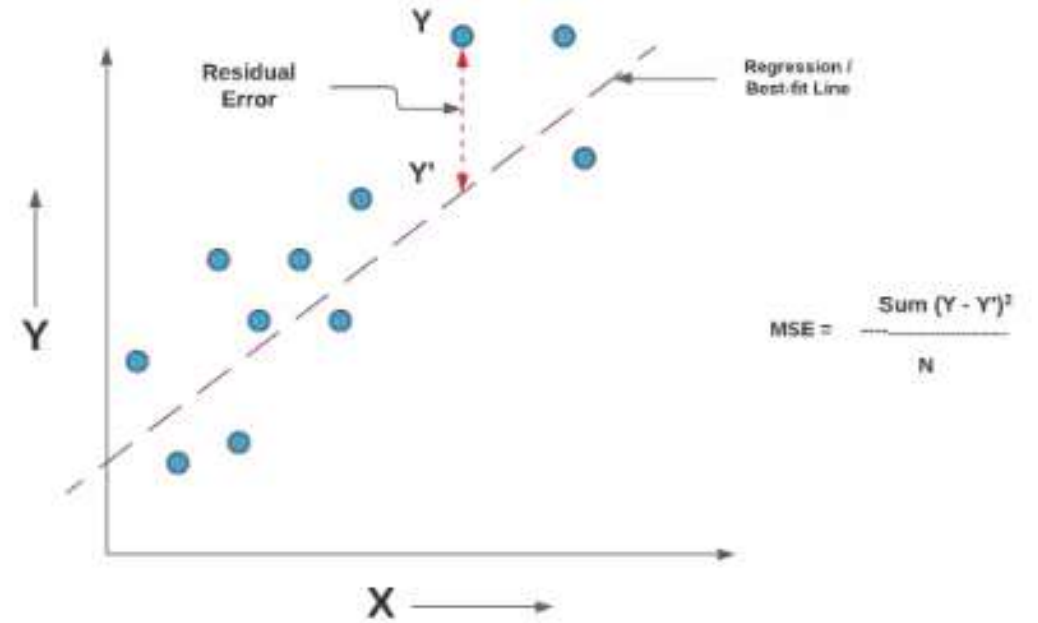


MÉTRICAS DE ERROR



Se define como la media de la diferencia entre el valor real y el valor predicho o estimado al cuadrado

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$



MEAN SQUARE ERROR

MEAN ABSOLUTE ERROR

- Se define como la diferencia en valor absoluto entre el valor real
- y el valor predicho.

$$MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$$

ROOT MEAN SQUARE ERROR (RMSE)

- Se define como la raíz cuadrada de la media de la diferencia entre el valor real y el valor predicho o estimado al cuadrado
- Comparada con el error absoluto medio (MAE), amplifica y penaliza los errores grandes.

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}$$

ROOT MEAN SQUARE LOGARITHMIC ERROR (RMSE)

- Logaritmo de la raíz del error cuadrático medio, Root Mean Logarithmic Square Error (RMLSE):
- Esta métrica penaliza una under-prediction más que una over-prediction

$$RMLSE = \sqrt{1/n \sum_{i=1}^n (\log(y_i + 1) - \log(\hat{y}_i + 1))^2}$$

UNDER PREDICTION (SUBPREDICCIÓN)

Definición

- La subpredicción ocurre cuando un modelo predice un valor más bajo que el valor real

Ejemplos

- **En finanzas:** Un modelo que predice ingresos futuros de una empresa y estima un valor mucho más bajo de lo que realmente será.
- **En diagnóstico médico:** Un modelo que predice una baja probabilidad de enfermedad para un paciente que en realidad tiene una alta probabilidad de padecerla.
- **En pronósticos meteorológicos:** Un modelo que subestima la cantidad de lluvia que caerá

OVER PREDICTION (SOBREPREDICCIÓN)

Definición

La sobrepredicción, por otro lado, ocurre cuando un modelo predice un valor más alto que el valor real

Ejemplos

En finanzas: Un modelo que predice ingresos futuros de una empresa y estima un valor mucho más alto de lo que realmente será, lo que puede llevar a decisiones de inversión erróneas

En diagnóstico médico: Un modelo que predice una alta probabilidad de enfermedad para un paciente que en realidad tiene una baja probabilidad de padecerla, lo que puede llevar a pruebas y tratamientos innecesarios

n pronósticos meteorológicos: Un modelo que sobrestima la cantidad de lluvia que caerá, lo que puede llevar a preparaciones excesivas

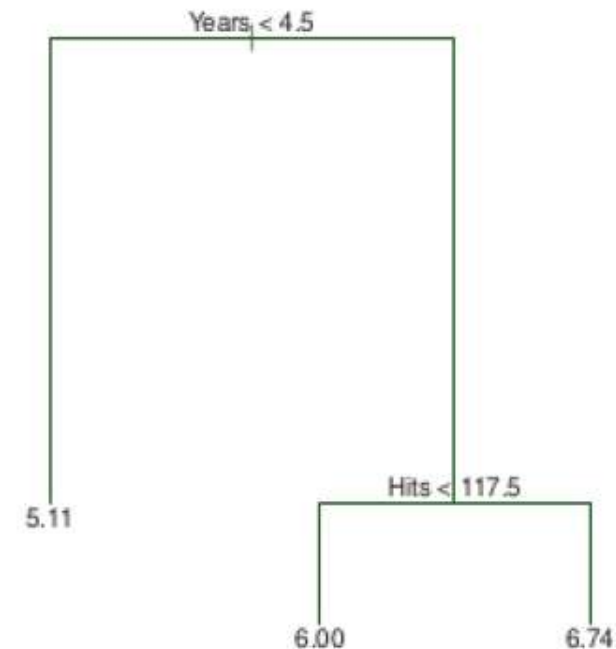


REGRESIÓN Y CLASIFICACIÓN CON ÁRBOLES DE DECISIÓN



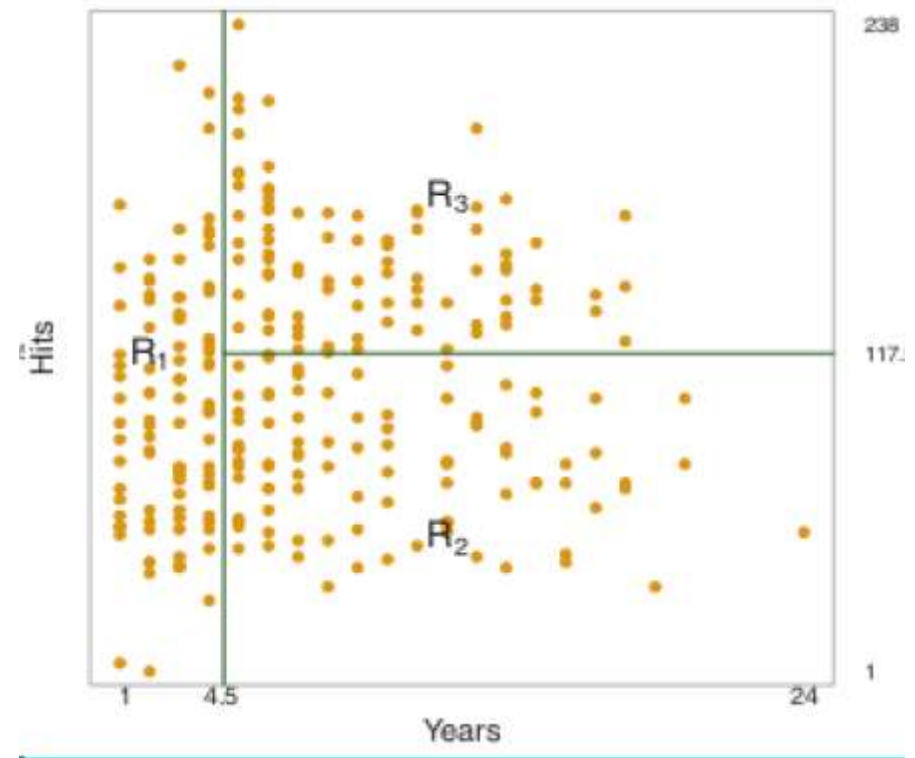
INTRODUCCIÓN

- Una de las técnicas más populares en data mining: permiten resolver problemas de regresión y clasificación.
- Dividen o segmentan el espacio de las variables predictoras en una serie de regiones.
- Para predecir una observación se utiliza la media o la moda de las observaciones que pertenecen a esa región
- Las reglas para separar las variables predictoras se resumen en forma de árbol, y se les conoce como árboles de decisión
- Se trata de métodos simples y fáciles de interpretar



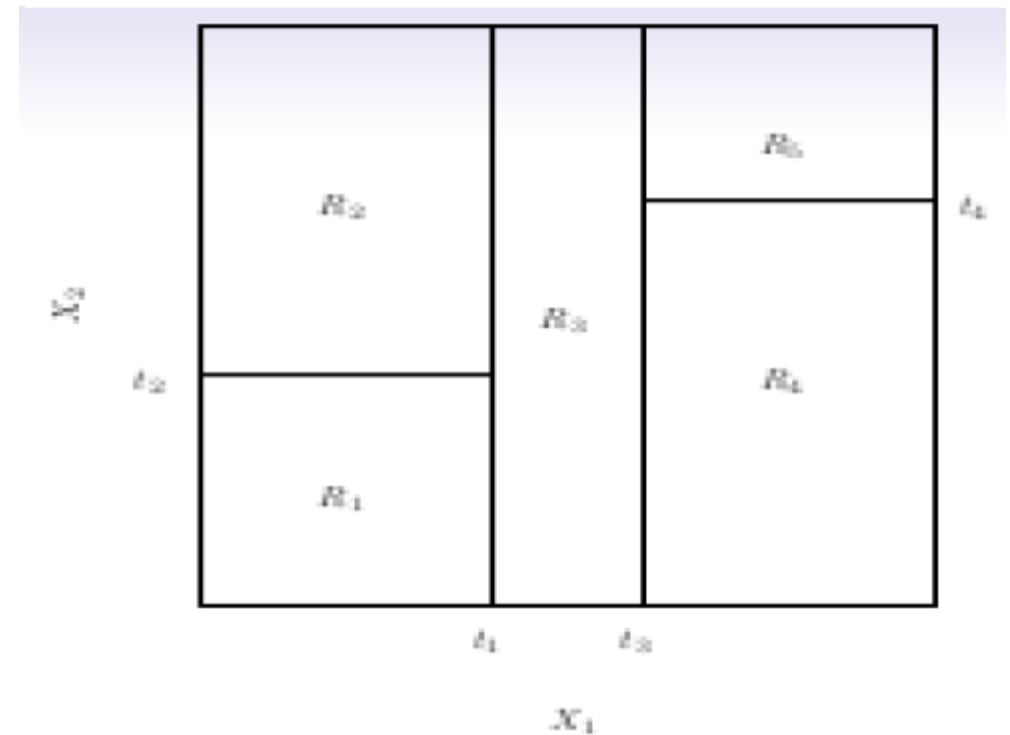
ÁRBOLES DE DECISIÓN

- Dividen el espacio en rectángulos en high-dimensión o boxes que minimizan el error de la predicción
- Es computacionalmente imposible considerar cualquier posible partición del espacio de entrada.
- Se utiliza un enfoque top-down greedy conocido como recursive binary splitting.



ALGORITMO

- En el nodo raíz se elige aquella variable que es más predictiva del target. Los ejemplos se dividen en grupos con distintos valores para esta clase
- El algoritmo continúa dividiendo los nodos con la elección de la mejor variable hasta que se alcance el criterio de parada:
 - Todos (casi todos) los ejemplos del nodo son de la misma clase.
 - No existen variables para distinguir entre los ejemplos.
 - El árbol ha alcanzado un tamaño predefinido.



MEJOR CORTE

- Sí los segmentos de una división contienen valores de una sola clase se consideran que es una división pura.
- Existen varias métricas de pureza para identificar los criterios de corte
- **Entropía.** Un valor de 0 indica que la muestra es completamente homogénea, mientras 1 indica desorden completo.

$$\text{Entropy}(S) = \sum_{i=1}^c -p_i \log_2(p_i)$$

MEJOR CORTE

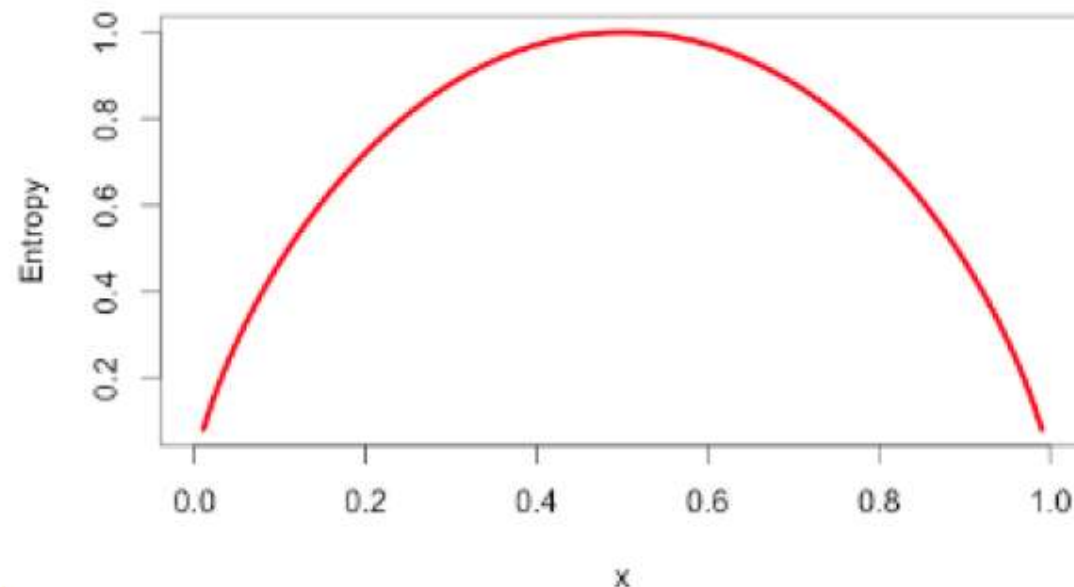
Por ejemplo si tenemos una partición de los datos en dos clases 60% y 40%.

```
> -0.60 * log2(0.60) - 0.40 * log2(0.40)
```

```
[1] 0.9709506
```

Si conocemos que la proporción de una clase es x , se puede examinar la entropía de todas las posibles divisiones de dos clases utilizando la función `curve()`

```
> curve(-x * log2(x) - (1 - x) * log2(1 - x), col = "red", xlab = "x", ylab = "Entropy", lwd = 4)
```



MEJOR CORTE

- Con esta medida de pureza el algoritmo tiene que decidir con que variable hacer el corte. Se utiliza la entropía para calcular el cambio resultante de hacer el corte en esa variable.
- Se calcula la Information Gain (IG) que es la diferencia entre la entropía en el segmento antes de hacer el split (S_1) y la partición resultante de hacer el split (S_2).
- Después de un split los datos se pueden dividir en más de una partición, por tanto se considera la entropía a lo largo de las N particiones ponderando la entropía de cada partición con el número de instancias de esa partición

$$\text{InfoGain}(F) = \text{Entropy}(S_1) - \text{Entropy}(S_2)$$

$$\text{Entropy}(S) = \sum_{i=1}^n w_i \text{Entropy}(P_i)$$



El proceso top-down greedy puede generar buenas predicciones en el training set, pero también sufre de overfitting.



Esto es porque el árbol puede ser muy complejo. Un árbol más pequeño puede dar lugar a una menor varianza y mejor interpretación con el coste de un pequeño bias(sesgo).

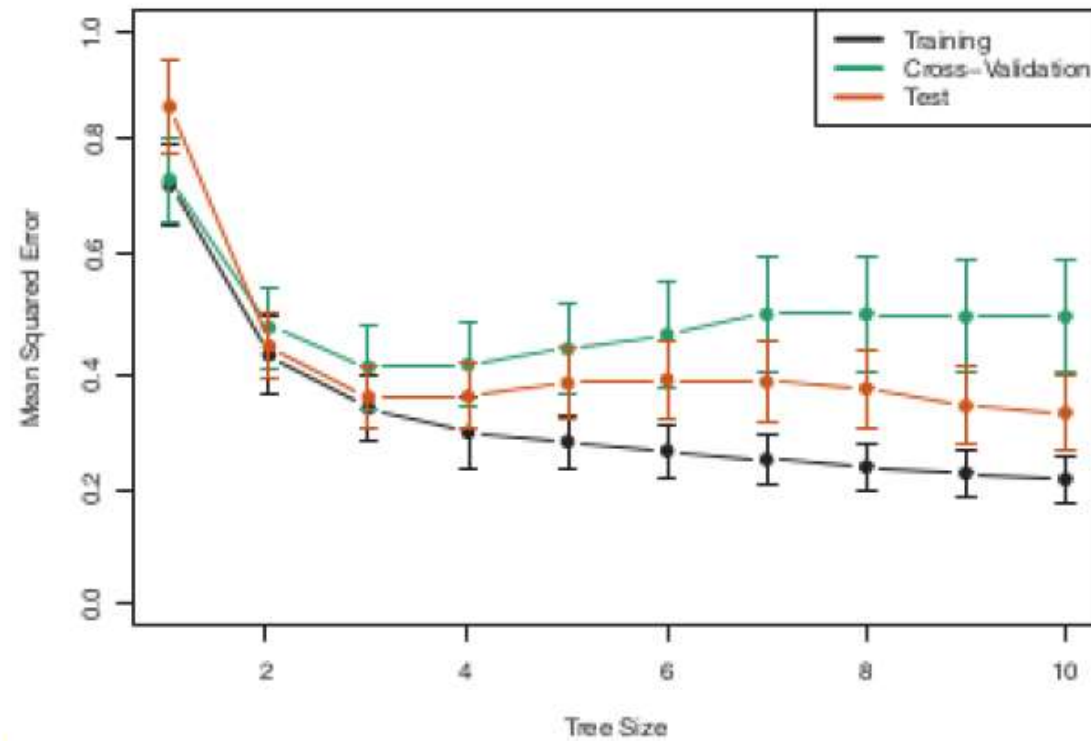


La estrategia suele ser: generar un árbol muy grande y luego podarlo para obtener un sub-árbol. (post-pruning)



¿Como seleccionamos el sub-árbol?. Utilizando aquel que proporcione un menor error de test con cross-validation o bien en un conjunto de validación.

MEAN SQUARED ERROR





En árboles de regresión la predicción corresponde a la media de las observaciones de ese nodo terminal. En

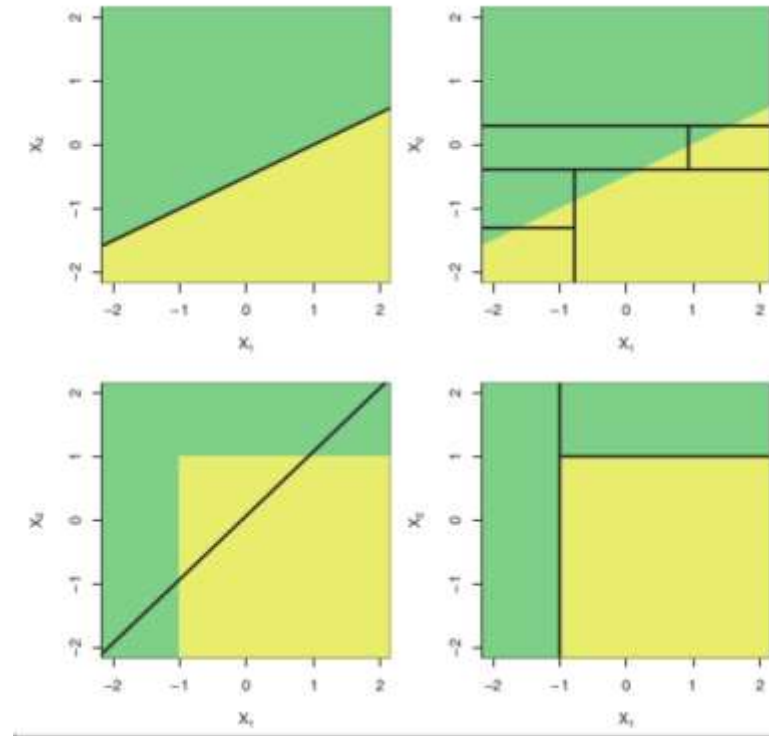


clasificación con la clase que tiene mayor número de ocurrencias.

RESULTADO

ÁRBOLES VS. MODELOS LINEALES QUE MODELO ES MEJOR?

- Depende. Si las relaciones entre las variables y la variable respuesta se pueden aproximar bien por un modelo lineal, una regresión lineal funciona bien y supera a un árbol de regresión (que no puede modelar esta estructura).
- Sin embargo, sí el problema es no-lineal y con relaciones complejas entre las variables, los árboles funcionan mejor.



FORTALEZAS Y DEBILIDADES

- Clasificador de propósito general que se comporta bien en la mayoría de los problemas
- Utiliza solo las variables más importantes
- Se puede utilizar con pocos o muchos datos de entrenamiento
- Da como resultado un modelo que se puede interpretar sin conocimientos matemáticos
- Suelen estar sesgados a splits en variables que tienen muchos niveles
- Pequeños cambios en los datos de entrenamiento dan lugar a grandes cambios en la lógica de decisión

EJERCICIO

Definición de columnas del conjunto de datos

- instant: índice de registro
- dteday : fecha
- season : estación (1: invierno, 2: primavera, 3: verano, 4: otoño)
- yr : año (0: 2011, 1:2012)
- mnth : mes (1 to 12)
- hr : hora (0 to 23)
- holiday : día feriado (extracted from [Web Link])
- weekday : día de la semana
- workingday : si el día no es fin de semana ni feriado entonces es 1 caso contrario 0.
- weathersit :
 - 1: despejado, pocas nubes, parcialmente nublado, parcialmente nublado
 - 2: Niebla + Nublado, Niebla + Nubes rotas, Niebla + Pocas nubes, Niebla
 - 3: nieve ligera, lluvia ligera + tormenta eléctrica + nubes dispersas, lluvia ligera + nubes dispersas
 - 4: lluvia intensa + paletas de hielo + tormenta eléctrica + niebla, nieve + niebla
- temp : Temperatura normalizada en grados Celsius. Los valores se derivan a través de $(t - t_{\min}) / (t_{\max} - t_{\min})$, $t_{\min} = -8$, $t_{\max} = + 39$ (solo en escala por hora)
- atemp: Temperatura de sensación normalizada en grados Celsius. Los valores se derivan a través de $(t - t_{\min}) / (t_{\max} - t_{\min})$, $t_{\min} = -16$, $t_{\max} = + 50$ (solo en escala por hora)
- hum: Humedad normalizada. Los valores se dividen en 100 (máx.)
- windspeed: Velocidad del viento normalizada. Los valores se dividen en 67 (máx.)
- casual: recuento de usuarios ocasionales
- registered: recuento de usuarios registrados
- cnt: recuento del total de bicicletas de alquiler, incluidas las casuales y las registradas

LIBRERIAS NECESARIAS

```
import pandas as pd
import numpy as np

import warnings
warnings.filterwarnings("ignore")

from sklearn.tree import DecisionTreeRegressor, DecisionTreeClassifier
from sklearn.ensemble import RandomForestRegressor, RandomForestClassifier
from sklearn import metrics
from sklearn.model_selection import train_test_split
import pydot
from prettytable import PrettyTable
from sklearn.tree import export_graphviz
import graphviz
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

RECUPERACIÓN DE DATOS PASO I

Recuperación de datos

```
▶ datos = pd.read_csv('hour.csv')
```

```
▶ datos.head()
```

ANÁLISIS DESCRIPTIVO PASO 2

- Para realizar analisis descriptivo separamos las columnas en categóricas y numéricas
- Se definen la columnas de variables (features)
- Se define la columna de clases (objetivo)

```
▶ # columnas con valores categoricos
variables_categoricas = ['season', 'yr', 'mnth', 'hr', 'holiday', 'weekday', 'workingday', 'weathersit']
# columnas con valores numericos
variables_numericas = ['temp', 'atemp', 'hum', 'windspeed']
# lista de columnas de valores
variables = variables_categoricas + variables_numericas
# definicion de clases o variable objetivo
clases = ['cnt']
variables_clases = variables + clases
```

```
▶ # Análisis de variables numéricas
print(datos[variables_numericas].describe())
```

ANÁLISIS DESCRIPTIVO PASO 2

```
# Análisis de variables categóricas  
print(datos[variables_categoricas].astype('category').describe())
```

```
# Verificar valores missing  
print(datos.isnull().any())
```

```
# Análisis de clases  
print(datos[clases].describe())
```

TRATAMIENTO DE DATOS PASO 3

Al analizar la variable a predecir cnt vemos que existe una distancia bastante grande entre el máximo y su cuartil 75, por lo que procederemos a realizar una eliminación de outliers

```
▶ # Rango intercuartilico 25
q1 = datos.cnt.quantile(0.25)
# Rango intercuartilico 75
q3 = datos.cnt.quantile(0.75)
# Diferencia entre rangos quartilicos
iqr = q3 - q1

# Definir el limite inferior
limite_inferior = q1 -(1.5 * iqr)
# Definir el limite superior
limite_superior = q3 +(1.5 * iqr)

# Filtrar los datos de acuerdo a limites inferior y superior
datos_preprocesados = pd.DataFrame(datos[variables_clases].loc[(datos.cnt >= limite_inferior) & (datos.cnt <= limite_superior)])
print("Ejemplos del grupo de entrenamiento con outliers: {}".format(len(datos)))
print("Ejemplos del grupo de entrenamiento sin outliers: {}".format(len(datos_preprocesados)))

# Desplegar la grafica
sns.distplot(datos.cnt)
sns.distplot(datos_preprocesados.cnt)
```

ANÁLISIS DE LA CORRELACIÓN DE VARIABLES PASO 3

```
# Construir matriz de correlacion
matrix = datos[variables_numericas + clases].corr()
heat = np.array(matrix)
heat[np.tril_indices_from(heat)] = False
# Construir mapa de calor
fig,ax= plt.subplots()
fig.set_size_inches(20,10)
sns.heatmap(matrix, mask=heat,vmax=1.0, vmin=0.0, square=True,annot=True, cmap="Blues")
```


ENTRENAMIENTO Y EVALUACIÓN

```
# Conjunto de valores
valores = datos_preprocesados[variables_procesar]
nombre_columnas_valores = datos_preprocesados[variables_procesar].columns.values

# Conjunto de Clases
valores_clases = datos_preprocesados[clases]

# Dividir en Conjuntos de Valores y Clases en Conjuntos de Entrenamiento y Test
X_train, X_test, y_train, y_test = train_test_split(valores, valores_clases, test_size=0.20, random_state=0)

# Se define la profundidad del árbol basándose en el error cuadrático medio,
max_profundidad = 15

# Se Crea el modelo utilizando la entropía para la definición del split
Modelo = DecisionTreeRegressor(max_depth=max_profundidad)

# Evaluar el modelo
evaluar_metrica_regresion(Modelo, X_train, y_train, X_test, y_test)
```

Máxima profundidad definida: 14

Model	RMSE	RMSLE	R ² score
DecisionTreeRegressor	53.24	0.44	0.88

EVALUACIÓN

```
def evaluar_metriza_regresion(estimador, X_train, y_train, X_test, y_test):  
    """  
    Función Para evaluar las métricas en algoritmos de regresión  
    Inputs:  
        estimador: Modelo estimador a evaluar  
        X_train: Conjunto de Valores de Entrenamiento  
        y_train: Conjunto de Clases de Entrenamiento  
        X_test: Conjunto de Valores de Test  
        y_test: Conjunto de Clases de Test  
    """  
  
    # Construir una Tabla  
    table = PrettyTable()  
    # Definir nombre de columnas de la tabla  
    table.field_names = ["Model", "RMSE", "RMSLE", "R2 score"]  
    # Entrenar el modelo  
    estimador.fit(X_train, y_train)  
    # Realizar predicción  
    y_pred = estimador.predict(X_test)  
    # Obtener Métrica RMSE  
    rmse = np.sqrt(metrics.mean_squared_error(y_test, y_pred))  
    # Obtener Métrica Score  
    score = estimador.score(X_test, y_test)  
    # Obtener Métrica RMSLE  
    rmsle = np.sqrt(metrics.mean_squared_log_error(y_test, y_pred))  
    # Agregar a la tabla los valores de las métricas  
    table.add_row([type(estimador).__name__, format(rmse, '.2f'), format(rmsle, '.2f'), format(score, '.2f')])  
    # imprimir resultados  
    print(table)  
  
    # Grafica Diagrama de Dispersion x=reales, y=prediccion  
    plt.figure(figsize=(10,8))  
    plt.scatter(x=y_test, y=y_pred, marker='o', c='b', edgecolors=(0, 0, 0), alpha=0.5)  
    plt.title('Dispersion {}'.format(type(estimador).__name__))  
    plt.ylabel('Predichos')  
    plt.xlabel('Reales')  
    plt.legend()  
    plt.show()
```

PRÁCTICA

- Generar una función para determinar el número de árboles óptimo



REGRESIÓN CON **RANDOM FOREST**

INTRODUCCIÓN

- Uno de los inconvenientes de los árboles de decisión o regresión (poca precisión) se puede solventar utilizando un **ensemble de árboles**.
- Los árboles se pueden combinar utilizando las técnicas de Bagging o Boosting (selección de variables aleatoria)

INTRODUCCIÓN



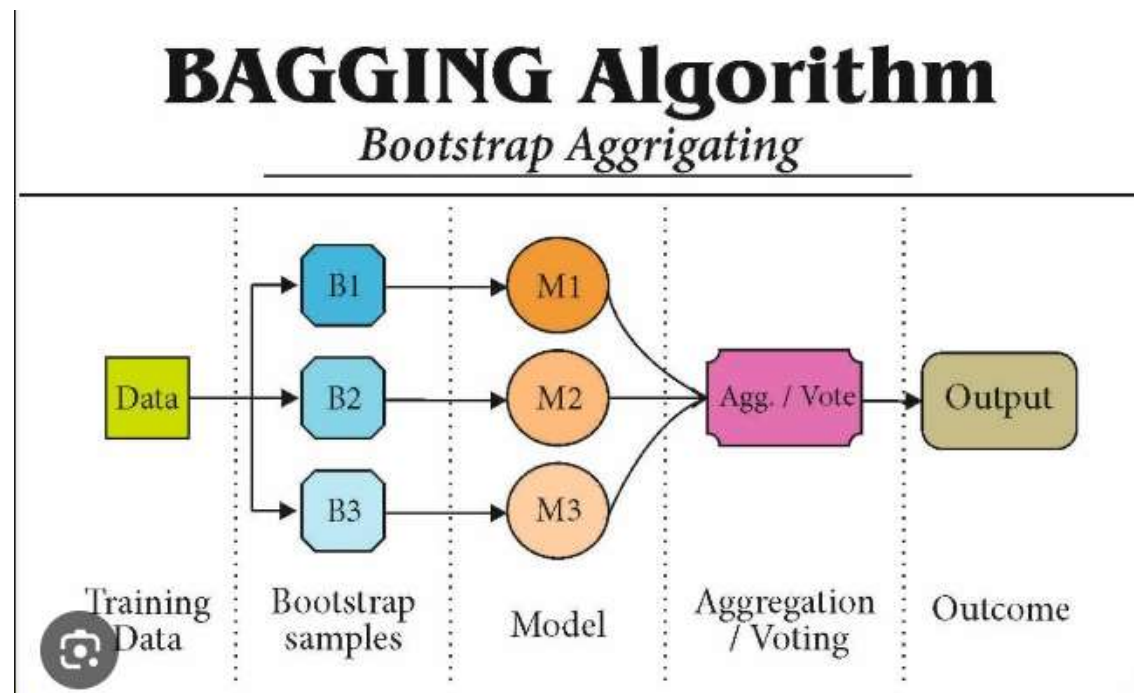
Para problemas de clasificación: en cada observación de test se almacena la clase predicha por cada uno de los B árboles y se obtiene un voto de la mayoría. La predicción global es la clase que más veces ocurre a lo largo de las B predicciones.



En el caso de los problemas de regresión, la predicción global es la media de las predicciones de cada uno de los árboles

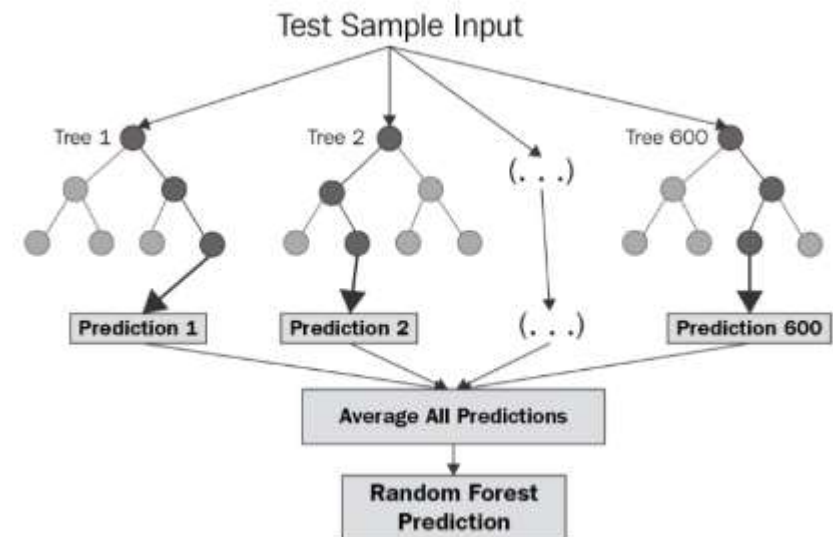
RANDOM FORESTS

- El modelo de random forests combina los principios de bagging con selección de variables aleatorias para añadir diversidad a los árboles de decisión. Una vez es generado el ensemble de árboles (forest) el modelo utiliza el mecanismo de votación o la media para generar las predicciones



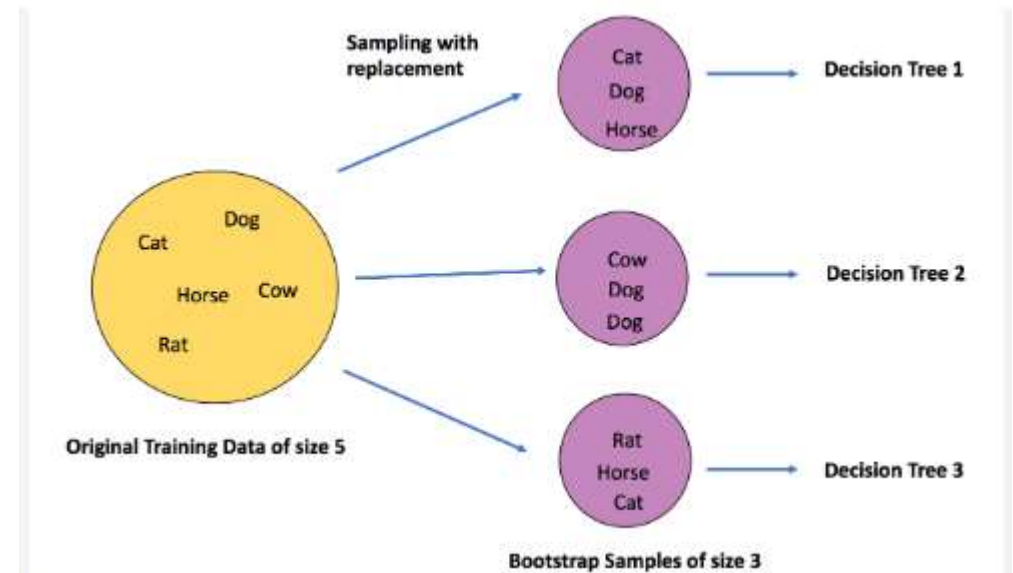
RANDOM FORESTS

- Este modelo se basa en la utilización de un gran número de árboles. La razón de utilizar un gran número de árboles es para que cada variable, de entre todas las posibles, tenga la oportunidad de aparecer en varios modelos



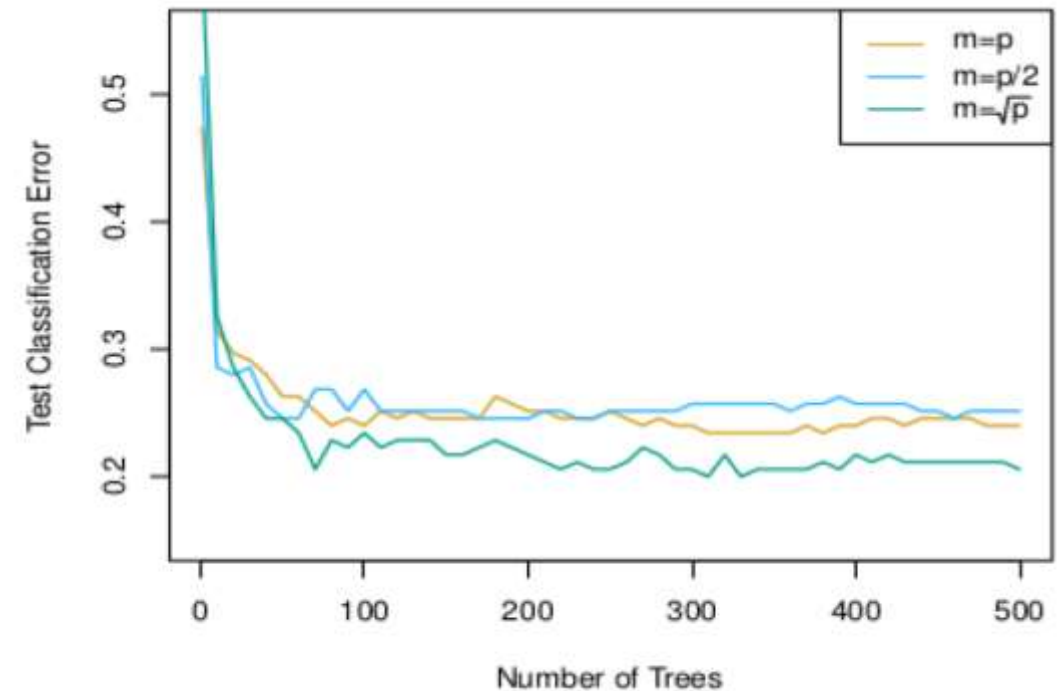
ERROR OUT OF BAG (OOB)

- Existe una forma sencilla de estimar el error de test en un modelo bagged.
- Se puede demostrar que en media cada bagged tree hace uso de $2/3$ de las observaciones.
- El $1/3$ restante de las observaciones que no se utilizan para ajustar un bagged tree se llaman observaciones out-of-bag (OOB).
- Para cada observación se puede predecir la respuesta de los bagged trees donde era OOB. Esto proporciona
- alrededor de $B/3$ predicciones para la observación i -ésima.
- Si B es grande, esta estimación es equivalente al Leave One Out cross-validation error



NÚMERO DE ÁRBOLES E IMPORTANCIA DE VARIABLES

- Los modelos basados en random forest tiene dos parámetros para optimizar
 - Número de árboles
 - Número de variables que se evalúan en cada split



PRACTICA



Calidad del vino

Donado el 6/10/2009

Se incluyen dos conjuntos de datos relacionados con muestras de vino verde tinto y blanco del norte de Portugal. El objetivo es modelar la calidad del vino basándose en pruebas fisicoquímicas (véase [Cortez et al., 2009],...

Características del conjunto de datos

Multivariante

Área temática

Negocio

Tareas asociadas

Clasificación, Regresión

Tipo de característica

Real

Instancias

4898

Características

11



GRACIAS

ALGUIEN@EJEMPLO.COM



ALGORITMOS DE CLASIFICACIÓN

DIFERENCIAS ENTRE UN PROBLEMA DE CLASIFICACIÓN BINARIA Y UN PROBLEMA DE CLASIFICACIÓN MULTICLASE



clasificador binario: Algoritmo para determinar la probabilidad de que un mensaje de correo electrónico sea spam y no spam



clasificador multiclase: Algoritmo que busque predecir en qué país de doce posibles un cliente va a realizar la siguiente reserva

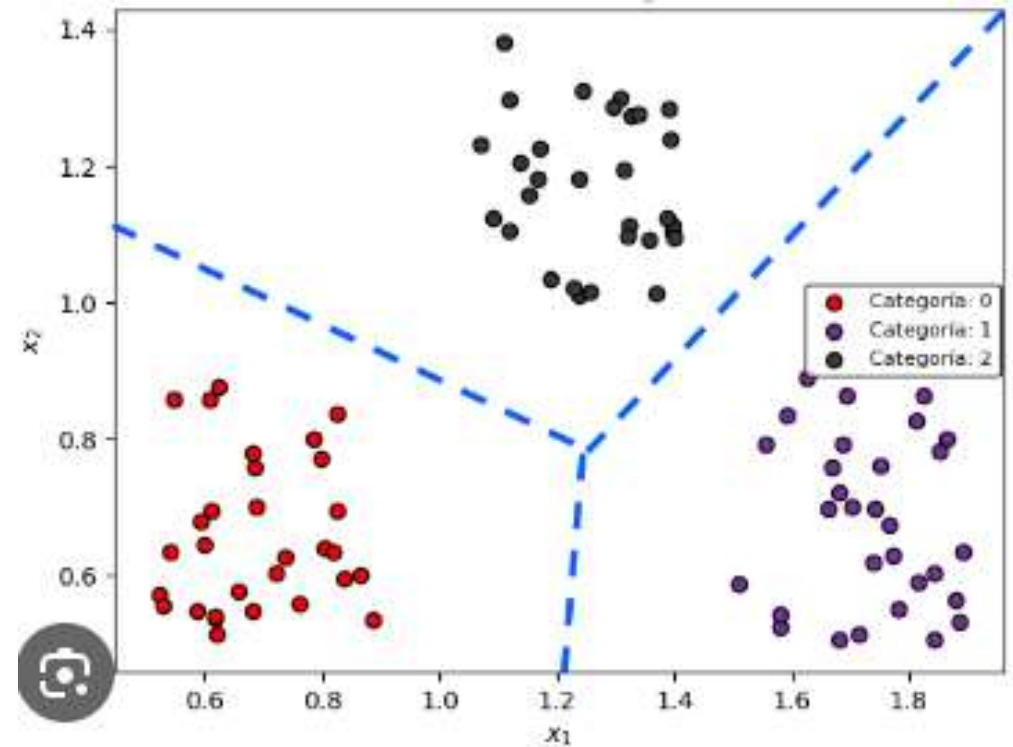
VENTAJA COMPETITIVA

- Capacidad para adelantarte a situaciones futuras de la forma más adecuadas y gestionarlas mejor
- Cualquier evaluación corre el riesgo de ser subjetiva, por tanto, es necesario establecer los criterios de evaluación más objetivos y rigurosos posibles



PROPÓSITO

- Objetivo: Obtener la clase más probable para cada una de sus instancias
- Se pueden utilizar para predecir o estimar la probabilidad de pertenencia de una clase de entre 2 posibles (Clasificación binaria)
- Predecir o estimar la probabilidad de una clase de entre varias posibles También esta definida para un vector de (Clasificación multiclase)



EVALUANDO EL RENDIMIENTO



Cualquier evaluación corre el riesgo de ser subjetiva



Es necesario establecer los métodos de evaluación más objetivos en cualquier escenario



Los algoritmos de los diferentes modelos pueden tener fortalezas y debilidades y es necesario disponer de la posibilidad de evaluarlos en las mismas condiciones

EVALUANDO EL RENDIMIENTO

- La mejor métrica de rendimiento de un clasificador es ver si un clasificador tiene éxito para su propósito
- Es muy útil utilizar una serie de medidas a partir de la matriz de confusión
- Existen diferentes tipos de datos que se pueden utilizar para evaluar un clasificador: (1) los valores reales de las clases, (2) los valores predichos y (3) la probabilidad estimada de la predicción



EN LA PRÁCTICA



Lo habitual es mantener dos vectores de datos: (1) contiene la verdad de la clasificación y (2) contiene los valores predichos. Ambos vectores deben tener la misma longitud y los datos en el mismo orden



La clase verdadera se conoce de antemano y es la variable a predecir del conjunto de datos



Los valores predichos se obtienen por medio de la aplicación del modelo. En la mayoría de los paquetes de ML de Python con la función `predict()` sobre un objeto de un modelo entrenado y un `data.frame` de test.

EN LA PRÁCTICA

Incluso en el caso de los clasificadores que predicen una sola clase, es útil saber con que certeza o confianza predicen cada clase. Por ejemplo: un mensaje puede ser Spam con una prob de 0.9 ó de 0.51.



Si dos modelos cometen los mismos errores pero uno es más capaz de tener en cuenta la incertidumbre, es mejor modelo.

- Es ideal encontrar un modelo que tenga mucha confianza en las predicciones correctas y sea temeroso en aquellas dudosas

EJEMPLO: CLASIFICACIÓN BINARIA

- Hay que tener cuidado en usar la probabilidad correcta para la categoría. En una clasificación binaria es común utilizar solo una de las clases

```
> head(predicted_prob)
      no      yes
1 0.0808272 0.9191728
2 1.0000000 0.0000000
3 0.7064238 0.2935762
4 0.1962657 0.8037343
5 0.8249874 0.1750126
6 1.0000000 0.0000000
```

EJEMPLO: CLASIFICACIÓN BINARIA

- Cuando los valores predichos son de la clase ham, los valores de spam son muy cercanos a 0. Cuando los valores predichos son Spam el valor es 1. Esto sugiere que el modelo tiene mucha confianza en sus clasificaciones.

```
> head(sms_results)
  actual_type predict_type   prob_spam
1         ham          ham 2.560231e-07
2         ham          ham 1.309835e-04
3         ham          ham 8.089713e-05
4         ham          ham 1.396505e-04
5        spam          spam 1.000000e+00
6         ham          ham 3.504181e-03
```

¿QUÉ OCURRE CUANDO LAS PREDICCIONES DIFIEREN?

- Observe que los valores son menos extremos. Por ejemplo la instancia 73 tiene un 35% de ser ham y se ha clasificado como Spam

```
> head(subset(sms_results, actual_type != predict_type))
```

	actual_type	predict_type	prob_spam
53	spam	ham	0.0006796225
59	spam	ham	0.1333961018
73	spam	ham	0.3582665350
76	spam	ham	0.1224625535
81	spam	ham	0.0224863219
184	spam	ham	0.0320059616







MÉTRICAS DE ERROR












MATRIZ DE CONFUSIÓN

- Una de las dimensiones de la tabla hace referencia a las categorías de los valores predichos, la otra dimensión a las categorías reales
- Las instancias clasificadas correctamente caen en la diagonal de la matriz. Los valores fuera de la diagonal indican las instancias clasificadas incorrectamente

Two Classes

		Predicted Class	
		A	B
Actual Class	A		
	B		

Three Classes

		Predicted Class		
		A	B	C
Actual Class	A			
	B			
	C			

MATRIZ DE CONFUSIÓN

		Condition (as determined by "Gold standard")			
		Condition Positive	Condition Negative		
Test Outcome	Test Outcome Positive	True Positive	False Positive (Type I error)	Positive predictive value = $\frac{\Sigma \text{ True Positive}}{\Sigma \text{ Test Outcome Positive}}$	
	Test Outcome Negative	False Negative (Type II error)	True Negative	Negative predictive value = $\frac{\Sigma \text{ True Negative}}{\Sigma \text{ Test Outcome Negative}}$	
		Sensitivity = $\frac{\Sigma \text{ True Positive}}{\Sigma \text{ Condition Positive}}$	Specificity = $\frac{\Sigma \text{ True Negative}}{\Sigma \text{ Condition Negative}}$		

- 1. Tasa de verdaderos positivos (true positives): se trata de las clasificaciones correctas de las instancias que corresponden a la clase positiva.
- 2. Tasa de falsos positivos (false positives): se trata de las clasificaciones de la clase negativa que han sido incorrectamente clasificadas como clase positiva.
- 3. Tasa de verdaderos negativos (true negatives): se trata de las clasificaciones correctas de las instancias que corresponden a la clase negativa.
- 4. Tasa de falsos negativos (false negatives): se trata de las clasificaciones de la clase positiva que han sido incorrectamente clasificadas como clase negativa.

ACCURACY / ERROR RATE

- Esta métrica también se conoce como el ratio de éxito.
- Representa la proporción del número de predicciones correctas entre el número total de predicciones.
- El ratio de Error indica la proporción de predicciones incorrectas

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\text{error rate} = \frac{FP + FN}{TP + TN + FP + FN} = 1 - \text{accuracy}$$

SENSIVITY / SPECIFICITY

- La sensibilidad de un modelo es el ratio de los ejemplos positivos correctamente clasificados.
- La especificidad de un modelo indica la proporción de los ejemplos negativos correctamente clasificados.
- Estas métricas tienen valores de 0 a 1, siendo 1 lo más deseable

$$\text{specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$

$$\text{sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

PRECISION / RECALL

- Precision (Positive Predictive Value PPV): Indica la proporción de ejemplos que son verdaderamente positivos. Cuando un modelo predice la clase positiva, ¿cuántas veces está en lo cierto?
- Recall: Indica que tan completos son los resultados (== sensivity)

$$\text{precision} = \frac{TP}{TP + FP}$$

$$\text{recall} = \frac{TP}{TP + FN}$$

F-MEASURE

- También conocida como F1, es una métrica que combina precisión y recall
- Se utiliza con mucha frecuencia puesto que reduce el rendimiento de un clasificador con una solo métrica.
- La métrica estándar asume que el precision y recall tienen la misma importancia, pero es posible ponderarlos para dar mayor peso a uno u otro.

$$F - \text{measure} = \frac{2 \times \text{precision} \times \text{recall}}{\text{recall} + \text{precision}} = \frac{2 \times TP}{2 \times TP + FP + FN}$$



CLASIFICACIÓN CON NÄIVE BAYES



INTRODUCCIÓN

- El clasificador Näive Bayes utiliza las probabilidades a-priori para estimar la probabilidad de los eventos futuros
- Utiliza datos de entrenamiento para calcular la probabilidad observada de cada evento en función del vector de características.
- Cuando el clasificador es utilizado con datos sin etiquetar, utiliza las probabilidades observadas para estimar la clase más probable.
- Ejemplo: Utilizar la frecuencia de las palabras en los correos spam para identificar nuevos correos spam en el futuro.



INTRODUCCIÓN

- Se utiliza principalmente para: clasificar texto, detección de intrusos en redes, diagnósticos médicos, etc.
- Si todos los eventos fueran independientes, sería imposible predecir ningún evento con los datos observados por el otro
- Los eventos dependientes son la base del modelado predictivo.
- Ejemplo, la presencia de nubes suele ser un evento predictivo de un día lluvioso. La presencia de la palabra viagra suele ser un evento predictivo de spam



NÄIVE BAYES

- La relación entre los eventos dependientes se puede describir utilizando el teorema de Bayes: $P(A|B) = \{P(B|A)P(A)\} / P(B)$
- Supongamos que queremos estimar la probabilidad de que un mensaje sea spam. Sin tener evidencias adicionales es un 0.2 (probabilidad a priori).
- Si tenemos la evidencia que el mensaje contiene la palabra viagra. La probabilidad de que viagra haya usado en mensajes de spam previos se llama verosimilitud (likelihood) y la probabilidad de que aparezca en cualquier mensaje se llama verosimilitud marginal (marginal likelihood).
- Aplicando el teorema de bayes, se puede calcular la probabilidad a-posteriori (posterior probability). Si es mayor que 0.5, es más probable que sea spam que ham y se debería filtrar.

$$P(\text{spam} | \text{Viagra}) = \frac{P(\text{Viagra} | \text{spam}) P(\text{spam})}{P(\text{Viagra})}$$

Diagram illustrating the Naïve Bayes formula for spam classification:

- $P(\text{spam} | \text{Viagra})$ is labeled as the **posterior probability**.
- $P(\text{Viagra} | \text{spam})$ is labeled as the **likelihood**.
- $P(\text{spam})$ is labeled as the **prior probability**.
- $P(\text{Viagra})$ is labeled as the **marginal likelihood**.

VIDEO

NAIVE

BAYES

TEORÍA

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

Es naïve porque trata a todas las variables como independientes e igualmente importantes.

Esto no es así en el mundo real. Por ejemplo, la persona que envía el email puede ser más importante que el propio texto.

Además, la aparición de las palabras no es independientes. Si aparece la palabra viagra, es muy probable que la palabra droga aparezca cerca.

Sin embargo, aunque no se tengan en cuenta estas situaciones, el clasificador funciona relativamente bien.

CARACTERÍSTICAS

EJEMPLO SPAM

- ▶ Para obtener cada uno de los componentes hay que construir una tabla de frecuencias, que indica el número de veces que la palabra viagra ha aparecido en los mensajes de spam. Esta tabla de frecuencias se puede utilizar para calcular una tabla de verosimilitud.

Frequency	Viagra		Total
	Yes	No	
spam	4	16	20
ham	1	79	80
Total	5	95	100

Likelihood	Viagra		Total
	Yes	No	
spam	4 / 20	16 / 20	20
ham	1 / 80	79 / 80	80
Total	5 / 100	95 / 100	100

- ▶ Para calcular la probabilidad posteriori, $P(\text{Spam}|\text{Viagra}) = (4/20) * (20/100) / (5/100) = 0.8$
- ▶ La probabilidad de que un mail que contenga la palabra viagra sea spam es del 0.8.

EJERCICIO APRENDIZAJE SUPERVISADO CON NAIVE BAYES

- Analizar el archivo house-votes-84.NAMES, donde se indica que existen 435 instancias correspondientes a las votaciones de EEUU, las cuales están conformadas por 267 demócratas y 168 republicanos con un total de 435 registros. Estas votaciones están clasificadas por 16 atributos que corresponde a los sectores de los Votantes más “Class Name”, que corresponde a la clase que se desea.
- Para el análisis de la Data, se recupera el archivo house-votes-84.DATA y se incorpora las columnas correspondientes a los atributos obtenidos de house-votes-84.NAMES

ANÁLISIS DE DATOS PASO I

```
#Paso 1: Revisión de datos
# Importar la libreria Pandas
import pandas as pd

#Recupero el archivo de data etiquetada
atributos = ['Class-Name','handicapped-infants','water-project-cost-sharing',
            'adoption-of-the-budget-resolution', 'physician-fee-freeze',
            'el-salvador-aid', 're-groups-in-schools', 'anti-satellite-test-ban',
            'aid-to-nicaraguan-contras', 'mx-missile', 'immigration',
            'synfuels-corporation-cutback', 'education-spending', 'superfund-right-to-sue',
            'crime', 'duty-free-exports','export-administration-act-south-africa']
df_datos=pd.read_csv('house-votes-84.data',sep=',',index_col='Class-Name', names=atributos)

print(df_datos.head(6))
```

ANÁLISIS DE DATOS PASO 2

```
[ ] #Paso 2
    #Reemplazo de datos correspondientes de y,n y ?. y significa que votaron por ese candidato (1)
    #n significa que no votaron por el candidato(0)
    #? significa que votaron en blanco (3), en el texto se indica que (?) no significa desconocido, signifca otro valor,
    df_datos = df_datos.replace('n','0')
    df_datos = df_datos.replace('y','1')
    df_datos = df_datos.replace('?', '3')

    df_datos.describe()
```


ENTRENAMEINTO PASO 3

```
#Paso 3
#Se Determina el conjunto de modelización y el de validación, para esto se utiliza train_test_split indicando que
# preserve las proporciones de Dataframe original
#El bloque de entrenamiento es el 75% de los registros, y al bloque de pruebas el 25% restante
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB

X_train, X_test, y_train, y_test = train_test_split(df_datos, df_datos.index, random_state=1, stratify = df_datos.index)

#Se verifica los tamaños de los datos originales, de entrenamiento y test
print("\nNumero de datos total", df_datos.shape[0])
print("Numero de datos para Entrenamiento 75% ", X_train.shape[0])
print("Numero de datos para Test 25%", X_test.shape[0])

#Se analiza que se mantengan los porcentajes de clasificación de lo datos del originar, entrenamiento y test agrupado por Cla:
print("_____Porcetnaje Original_____")
print(df_datos.groupby("Class-Name").count()/len(df_datos))
print("_____Porcentaje entrenamiento_____")
print(X_train.groupby("Class-Name").count()/len(X_train))
print("_____Porcentaje Test_____")
print(X_test.groupby("Class-Name").count()/len(X_test))
```

PASO 4 PREDICCIÓN


```
#Paso 5: Predicción
## Se usa MultinomialNB por ser valores discretos

naive_bayes = MultinomialNB()
naive_bayes.fit(X_train,y_train)

#Probar con predicciones
predictions = naive_bayes.predict(X_test)
predictions
```

MÉTRICAS DE EVALUACIÓN

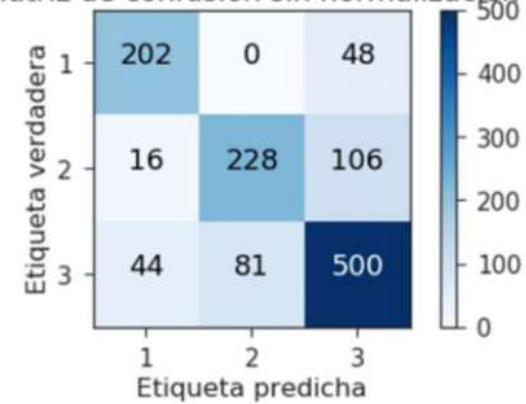
```
[ ] ##Paso 6 Evaluar Algoritmo
    ## Se calculan las siguientes métricas y matriz de confusión.
    from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
    print('Accuracy score: ', format(accuracy_score(y_test, predictions)))
```

 Accuracy score: 0.8623853211009175

EJERCICIO

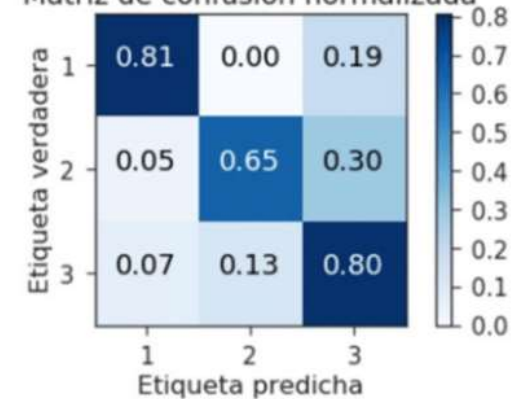
- Realizar la matriz confusión

Matriz de confusión sin normalización



(a)

Matriz de confusión normalizada



(b)



GRACIAS

ALGUIEN@EJEMPLO.COM