

MACHINE LEARNING

TABLA DE CONTENIDO

1. QUÉ ES MACHINE LEARNING.....	4
1.1. OVERVIEW.....	4
1.2. INTRODUCCIÓN.....	7
1.3. PYTHON PARA MACHINE LEARNING.....	13
1.4. APRENDIZAJE SUPERVISADO VS NO SUPERVISADO.....	16
PARTE II. REGRESIÓN LINEAL.....	22
2.1. INTRODUCCIÓN A LA REGRESIÓN.....	22
2.2. REGRESIÓN LINEAL SIMPLE.....	26
2.3. EVALUACIÓN DEL MODELO EN REGRESIÓN LINEAL.....	31
2.4. MÉTRICAS DE EVALUACIÓN EN MODELOS DE REGRESIÓN.....	36
2.5. REGRESIÓN LINEAL MÚLTIPLE.....	38
2.6. REGRESIÓN NO LINEAL.....	42
PARTE 3. DIFERENTES MÉTODOS DE CLASIFICACIÓN.....	45
3.1. INTRODUCCIÓN A LA CLASIFICACIÓN.....	45
3.2. K-NEAREST NEIGHBORS (K VECINOS MÁS CERCANOS).....	48
3.3. MÉTRICAS DE EVALUACIÓN EN CLASIFICACIÓN.....	56
3.4. INTRODUCCIÓN A LOS ÁRBOLES DE DECISIÓN.....	61
3.5. CONSTRUYENDO ÁRBOLES DE DECISIÓN.....	64
3.6. INTRODUCCIÓN A LA REGRESIÓN LOGÍSTICA.....	71
3.7. REGRESIÓN LOGÍSTICA VS REGRESIÓN LINEAL.....	76
3.8. ENTRENAMIENTO EN REGRESIÓN LOGÍSTICA.....	85
3.9. SUPPORT VECTOR MACHINES (SVM).....	89
PARTE 4. CLUSTERING.....	95
4.1. INTRODUCCIÓN AL CLUSTERING.....	95
4.2. INTRODUCCIÓN A K-MEANS.....	101
4.3. MÁS SOBRE K-MEANS.....	109
4.4. INTRODUCCIÓN AL CLUSTER JERÁRQUICO.....	111
4.5. MAS SOBRE EL CLUSTERING JERÁRQUICO.....	114
4.6. DBSCAN.....	119
PARTE 5. SISTEMAS RECOMENDADORES.....	122
5.1. INTRODUCCIÓN.....	122
5.2. SISTEMAS RECOMENDADORES BASADOS EN CONTENIDO.....	125

5.3. FILTROS COLABORATIVOS.....	130
ANEXO. CÓDIGOS.....	137
1. REGRESIÓN.....	137
1.1. REGRESIÓN LINEAL SIMPLE.....	137
1.2. REGRESIÓN POLINÓMICA.....	139
1.3. REGRESIÓN NO LINEAL.....	141
2. CLASIFICACIÓN.....	144
2.1. K-NN.....	144
2.2. ARBOLES DE DECISION.....	146
2.3. SVM.....	148
2.4. REGRESIÓN LOGÍSTICA.....	151
3. CLUSTERING.....	153
3.1. K-MEANS.....	153
3.2. CLUSTERING JERÁRQUICO.....	155
3.3. DBSCAN.....	157

1. QUÉ ES MACHINE LEARNING

1.1. OVERVIEW

El machine learning ML tiene aplicaciones en muchos campos e industrias clave, por ejemplo:

1. La industria de la salud en la que se utiliza el ML para predecir si un cáncer es maligno o benigno.

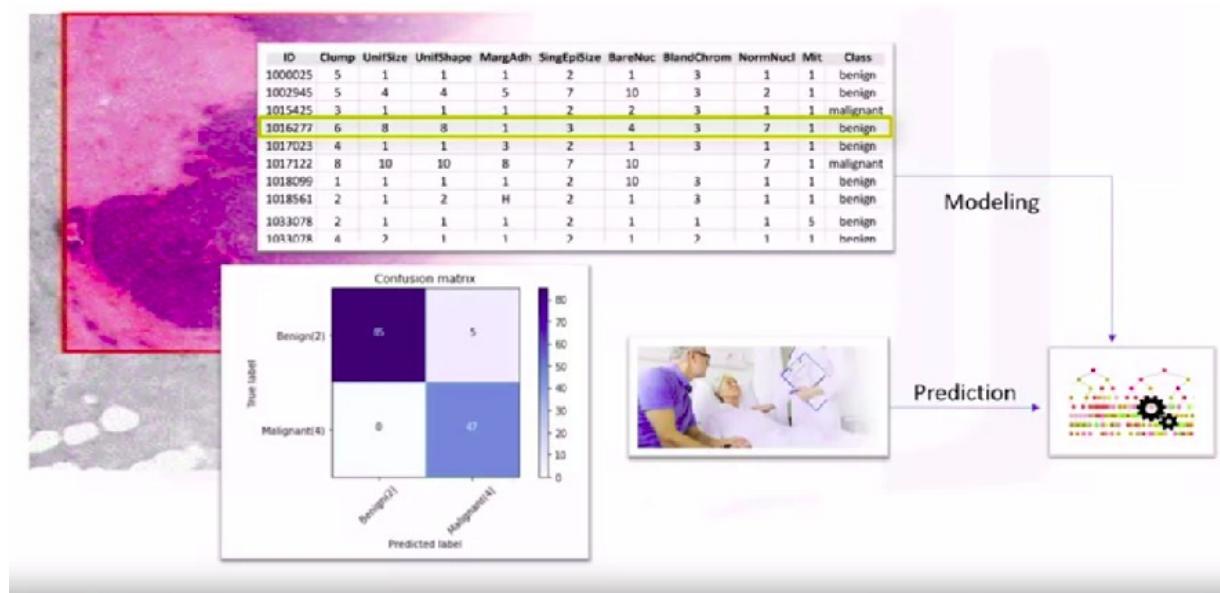


Figura. ML en medicina.

Aprenderemos el valor de los árboles de decisión, y cómo construir un buen árbol ayuda a los médicos a prescribir la medicina adecuada para cada uno de sus pacientes.

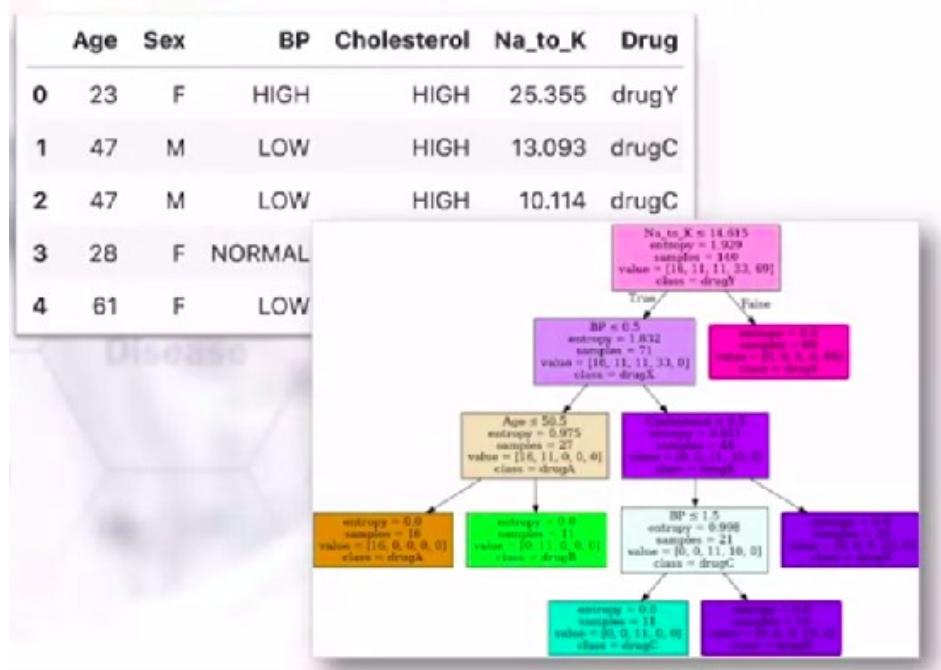


Figura. Árboles de decisión.

2. Los sistemas bancarios, que usan el ML para tomar decisiones sobre si aprobar solicitudes de préstamo y para realizar segmentaciones de clientes.

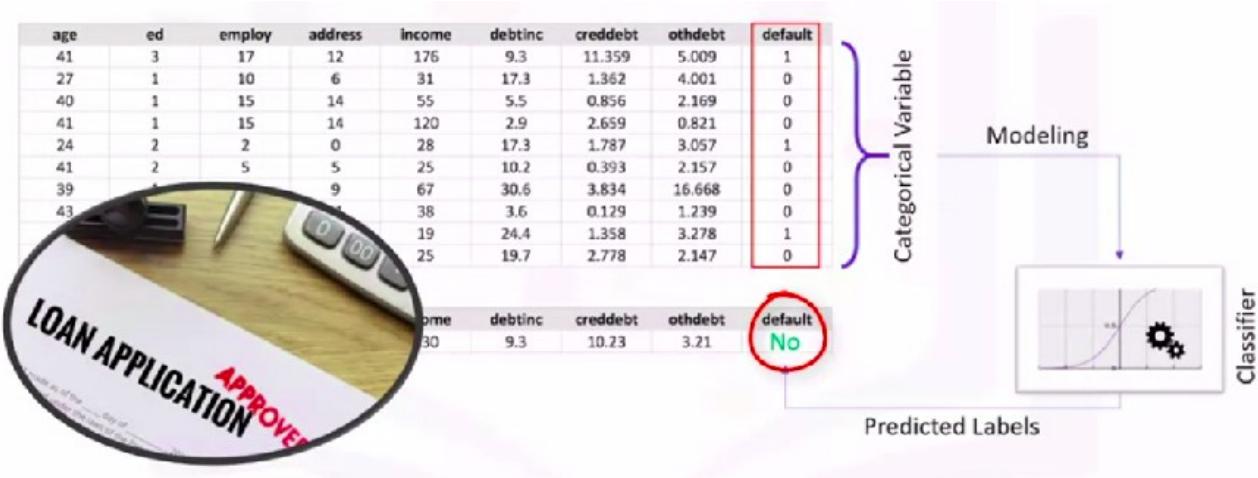


Figura. Toma de decisiones.

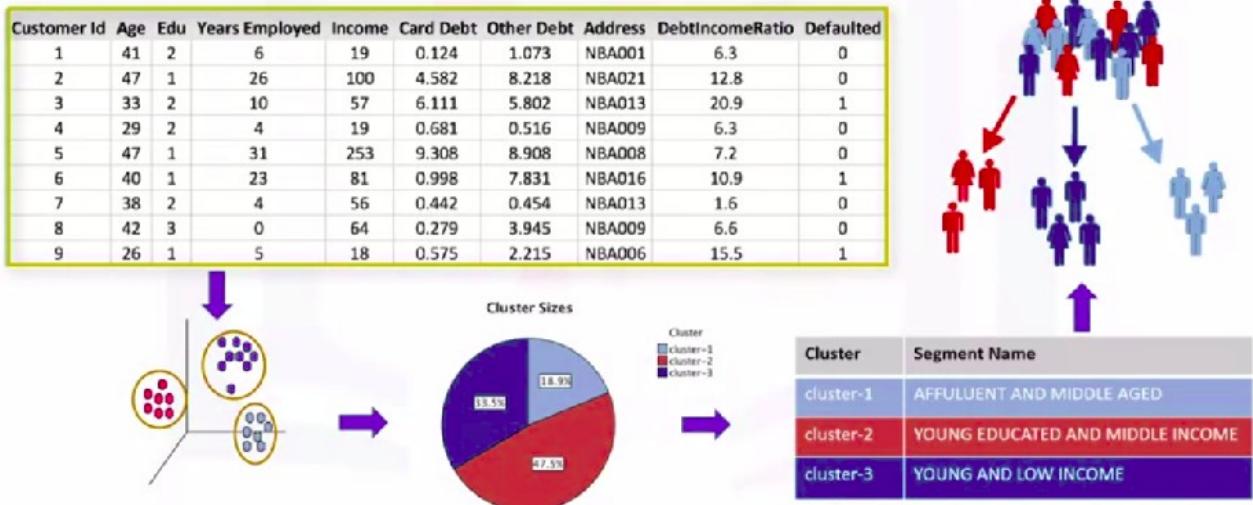


Figura. Segmentación de clientes.

3. Sitios como YouTube, Amazon o Netflix a desarrollare recomendaciones a sus clientes sobre diversos productos o servicios, cómo qué películas podrían estar interesados en ver o qué libros comprar.

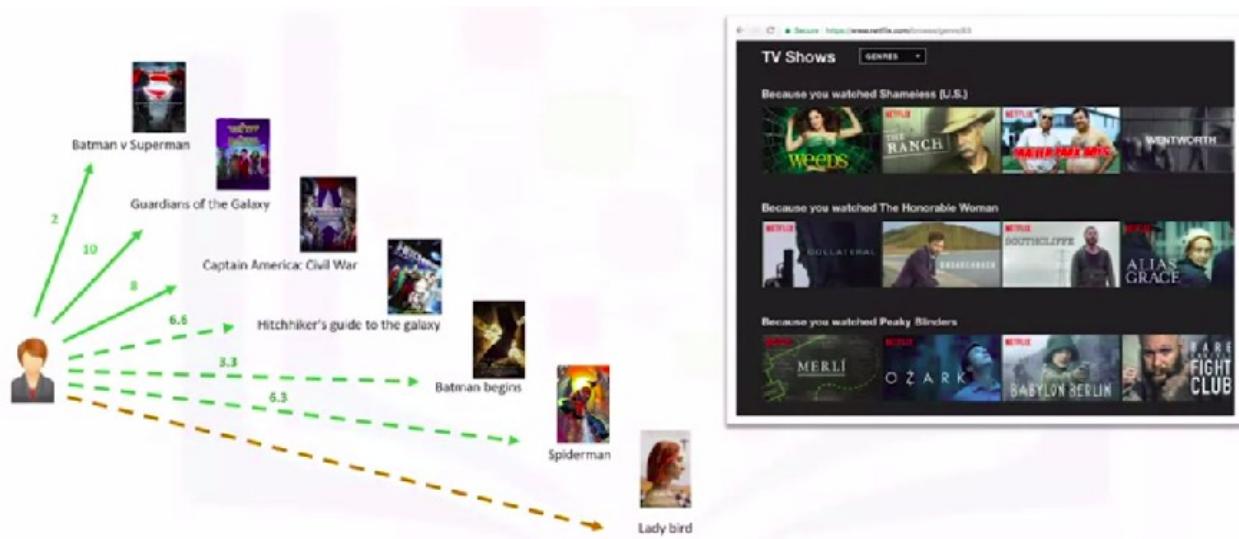


Figura. Sistemas recomendadores.

4. Las industrias de telecomunicaciones pueden predecir la pérdida de clientes.
 5. Muchos otros.

1.2. INTRODUCCIÓN

Veamos una introducción de alto nivel al ML.

En la figura siguiente se observa una muestra de células humanas extraída de un paciente junto con un grupo de características: por ejemplo su espesor de grupo es 6, su uniformidad de tamaño 1, su adhesión marginal 1, etc.

Una pregunta interesante sería:

Es una célula maligna o benigna?

A diferencia de un tumor benigno, uno maligno puede invadir el tejido circundante o diseminarse alrededor del cuerpo, y diagnosticarlo temprano podría ser la clave para la supervivencia del paciente.

Imagina que has obtenido un conjunto de datos que contiene las características de miles de muestras de células humanas extraídas de pacientes que se creía estaban en riesgo de desarrollar cáncer. El análisis de los datos originales mostró que muchas de las características diferían significativamente entre muestras benignas y malignas. Puede utilizar los valores de estas características celulares en muestras de otros pacientes para dar una indicación temprana de si una nueva muestra puede ser benigna o maligna.

Debe:

- Limpiar sus datos.
- Seleccionar un algoritmo adecuado para construir un modelo de predicción.
- Entrenar su modelo para comprender los patrones de células benignas o malignas dentro de los datos.

Una vez que el modelo ha sido entrenado revisando los datos de forma iterativa, se puede utilizar para predecir si una célula nueva o desconocida es benigna o maligna con cierta precisión. ¡ Esto es aprendizaje automático! . Es la forma en que un modelo de aprendizaje automático puede hacer la tarea de un médico o al menos ayudar a ese médico a hacer el proceso más rápido.

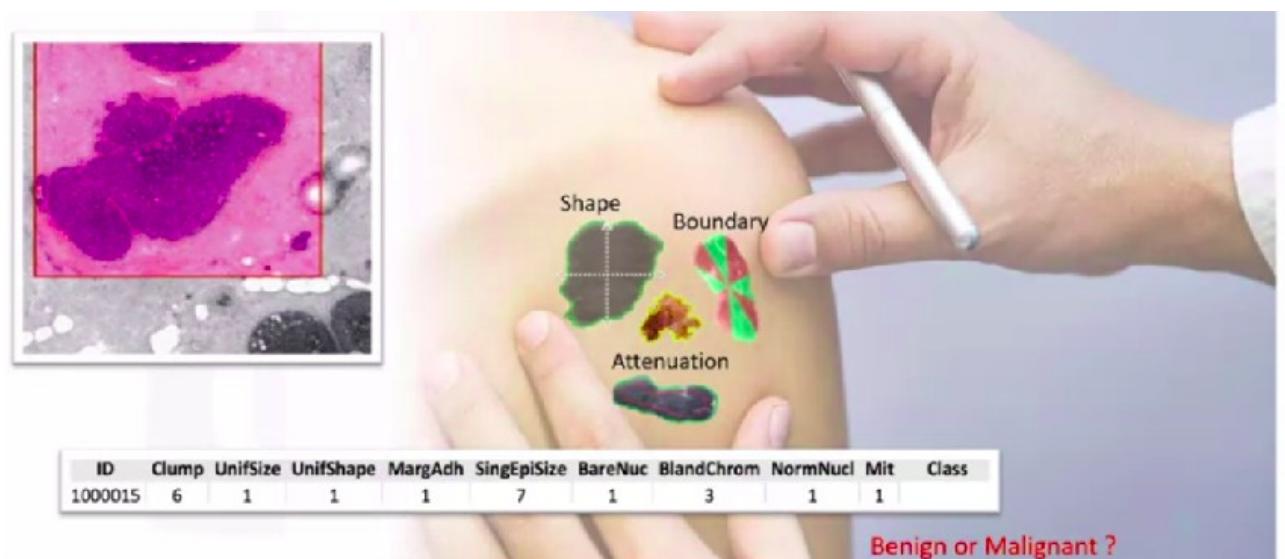


Figura. Aplicación del ML en medicina.

Demos ahora una definición formal de ML:

El **Machine Learning** es el subcampo de las ciencias de la computación que le da a “las computadoras la capacidad de aprender sin ser programadas explícitamente”.

Expliquemos a qué se refiere “sin ser programado explícitamente”.

Supongamos que tenemos un conjunto de datos de imágenes de animales como perros y gatos y desea tener una aplicación que pueda reconocerlos y diferenciarlos.

Lo primero que debe hacer es interpretar las imágenes como un conjunto de conjuntos de características. Por ejemplo, ¿la imagen muestra los ojos del animal? Si es así, ¿cuál es su tamaño? ¿ Tiene orejas? ¿ Qué tal una cola? ¿ Cuántas piernas? ¿ Tiene alas? .

Antes del ML, cada imagen se transformaría en un vector de entidades. Luego, tradicionalmente, teníamos que escribir algunas reglas o métodos para conseguir que los ordenadores fueran inteligentes y detectaran a los animales.

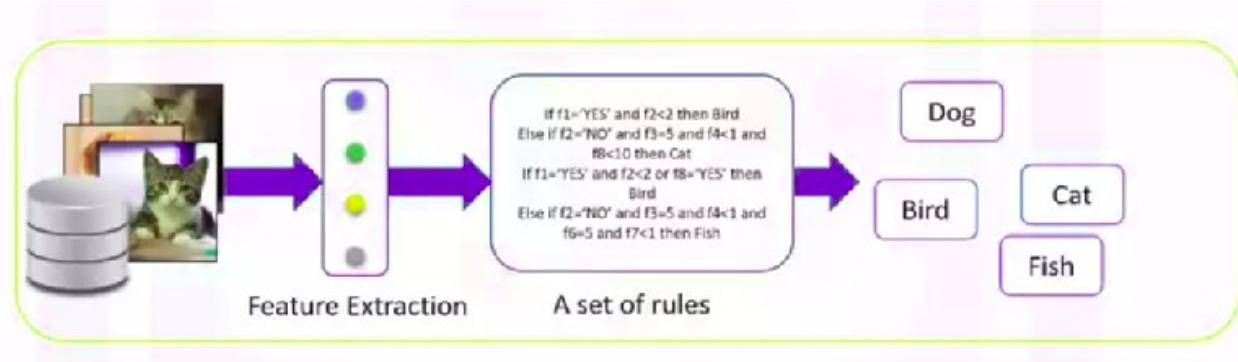


Figura. Reconocimiento de imágenes antes del ML.

Esto fue un fracaso. Por qué? Se necesitaban muchas reglas, altamente dependientes del conjunto de datos actual y no lo suficientemente generalizables como para detectar casos fuera de la muestra.

Aquí es cuando el ML entra en escena.

Usando el ML podemos construir un modelo que mira todos los conjuntos de características y su correspondiente tipo de animal y aprende el patrón de cada animal.

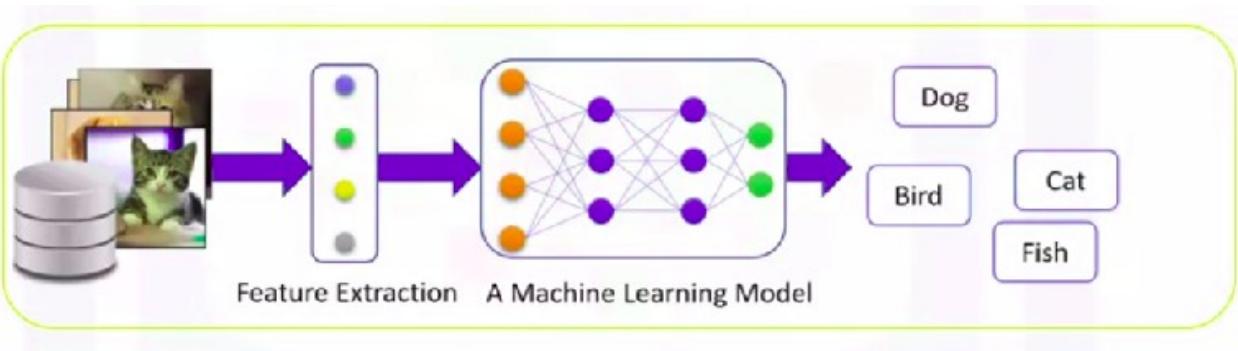


Figura. Reconocimiento con ML.

En esencia, el ML sigue el mismo proceso que un niño de 4 años utiliza para aprender, entender y diferenciar los animales. Así, los algoritmos de ML inspirados en el proceso de aprendizaje humano, aprenden iterativamente de los datos y permiten a las computadoras encontrar información oculta.

Aquí ejemplos de uso del ML en la vida real:

- Netflix y Amazon recomiendan videos, películas programas de televisión a sus usuarios usando ML.

- Esto es similar a cómo tus amigos podrían recomendarte un programa de TV, en función de su conocimiento de los tipos de programas que te gusta ver.
- La toma de decisiones de un banco de aprobar o no una solicitud de crédito.
 - Predicen la probabilidad de impago para cada solicitante y luego aprueban o rechazan la solicitud basándose en esa probabilidad.
- Las empresas de telecomunicaciones utilizan los datos demográficos de sus clientes para segmentarlos, o predecir si se darán de baja de su compañía el próximo mes.

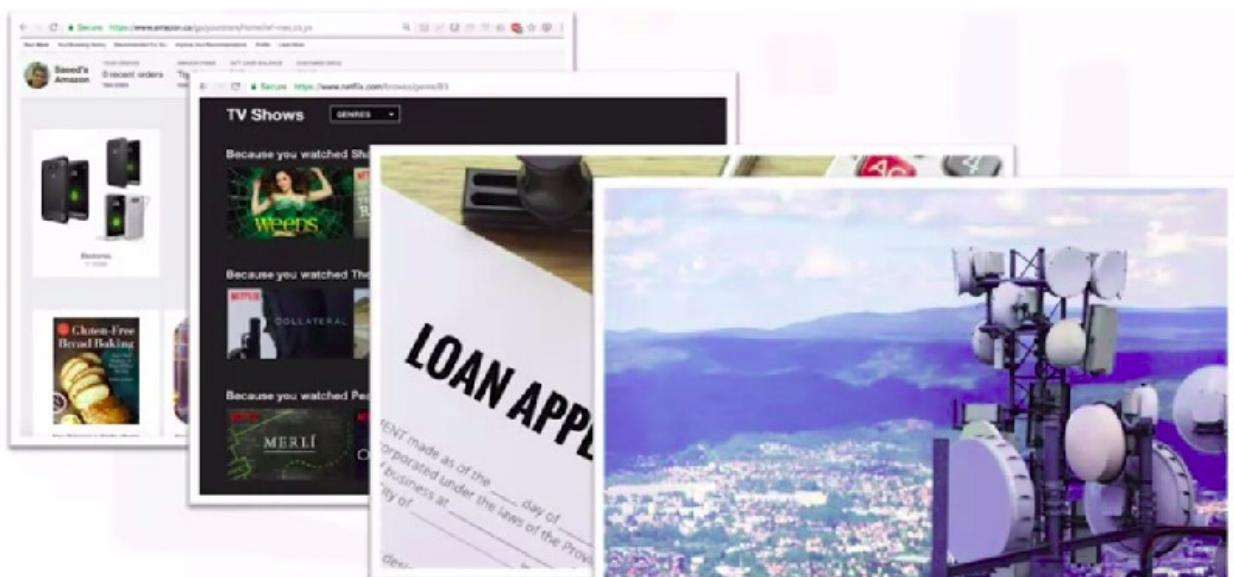


Figura. Aplicaciones del ML.

Hay muchas otras aplicaciones, como:

- chatbots
- Juegos de PC.
- Reconocimiento facial.

Examinemos algunas de las técnicas más populares:

- Regresión/estimación
 - Se utiliza para predecir un valor continuo.
 - Por ejemplo predecir cosas como el precio de una casa en función de sus características o estimar la emisión de CO₂ del motor de un auto.
- Clasificación

- Predecir la clase o categoría de un caso, por ejemplo, si una célula es benigna o maligna.
- Clustering
 - Es la agrupación de grupos de casos similares, por ejemplo, para encontrar pacientes similares o para la segmentación de clientes.
- Associations
 - La técnica de asociación se utiliza para encontrar artículos o eventos que a menudo co-ocurren, por ejemplo, artículos de comestibles que generalmente son comprados juntos por un cliente en particular.
- Detección de anomalías
 - Se utiliza para descubrir casos anormales o inusuales, por ejemplo la detección de fraude en tarjetas de crédito.
- Minería de secuencias
 - Se utiliza para predecir el próximo evento, por ejemplo, el flujo de clics en los sitios web.
- Reducción de dimensiones
 - Se utiliza para reducir el tamaño de los datos.
- Sistemas de recomendación
 - Asocian las preferencias de las personas con otras que tienen gustos similares y les recomienda nuevos artículos, como libros o películas.

Cuál es la diferencia entre:

- Inteligencia Artificial IA
- Machine Learning ML
- Deep Learning DL

IA intenta hacer que las computadoras sean inteligentes para imitar las funciones cognitivas de los humanos. Incluye:

- Computación visual.
- Procesamiento del lenguaje.
- Creatividad.

ML es la rama de la IA que cubre la parte estadística de la IA. Enseña a las computadoras a resolver problemas mirando cientos o miles de ejemplos, aprendiendo de

ellos y luego usando esa experiencia para resolver el mismo problema en nuevas situaciones.

El Deep Learning es un campo muy especial del ML donde las computadoras pueden aprender y tomar decisiones por sí mismas. El DL implica un nivel más profundo de automatización en comparación con la mayoría de los algoritmos de ML.

1.3. PYTHON PARA MACHINE LEARNING

Veremos cómo usar Python para ML.

Puede escribir sus algoritmos de ML usando Python directamente y funciona muy bien, sin embargo, hay muchos módulos y bibliotecas ya implementados que pueden hacer su vida mucho más sencilla.

1. NumPy

Es una biblioteca matemática para trabajar con matrices N-dimensionales. Permite realizar cálculos de manera eficiente y efectiva. Para trabajar con matrices, diccionarios, funciones, tipos de datos y trabajar con imágenes, necesita conocer NumPy.

2. SciPy

Es una colección de algoritmos numéricos y cajas de herramientas específicas que incluyen procesamiento de señales, optimización y estadística.

3. Matplotlib

Proporciona trazado 2D y 3D.

4. Pandas

Proporciona estructuras de datos fáciles de usar de alto rendimiento. Tiene muchas funciones para la importación de datos, manipulación y análisis. En particular, ofrece estructuras de datos y operaciones para manipular tablas numéricas y series temporales.

5. Scikit-Learn

Es una colección de algoritmos y herramientas para el ML.

Tiene ya implementados la mayoría de los algoritmos de clasificación, regresión y clustering y está diseñado para trabajar con NumPy y SciPy.

Tiene muy buena documentación.

Implementar modelos de ML es muy fácil.

La mayoría de las tareas que se deben realizar en un pipeline de ML ya están implementadas, incluyendo:

- Pre-procesamiento de datos.
- Selección de características.
- Extracción de características.

- División en conjuntos de test y training.
- Definición de algoritmos.
- Modelos de ajuste.
- Parámetros de ajuste.
- Predicción.
- Evaluación.
- Exportación de modelo.

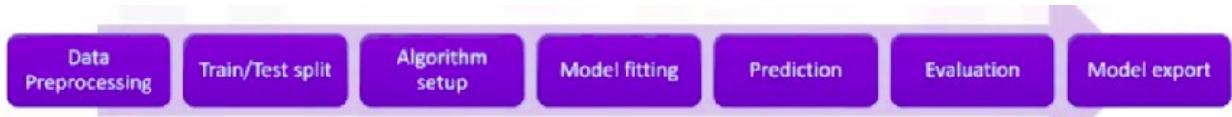


Figura. Pipeline típico de ML.

Veamos un ejemplo.

```

from sklearn import preprocessing
X = preprocessing.StandardScaler().fit(X).transform(X)

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33)

from sklearn import svm
clf = svm.SVC(gamma=0.001, C=100.)

clf.fit(X_train, y_train)

clf.predict(X_test)

from sklearn.metrics import confusion_matrix
print(confusion_matrix(y_test, yhat, labels=[1,0]))

import pickle
s = pickle.dumps(clf)

```

Analicemos un poco el código anterior:

- Los algoritmos de ML se benefician de la estandarización del conjunto de datos. Si hay valores atípicos o campos de escalas diferentes en el dataset, debe corregirlos. El paquete preprocessing proporciona varias funciones de utilidad

comunes y clases de transformadores para cambiar los vectores de entidades sin procesar en una forma adecuada de vector para el modelado.

- Debe dividir su conjunto de datos en conjuntos de entrenamiento y test para entrenar su modelo y luego probar la precisión del mismo por separado. Scikit-Learn puede hacer esto con una sola línea de código.
- En el ejemplo se creó un clasificador utilizando un algoritmo de clasificación de vectores de soporte (SVM). Llamamos a nuestra instancia estimadora CLF e inicializamos sus parámetros.
- Luego se entrena el modelo pasando el conjunto de entrenamiento al método de ajuste.
- Ahora podemos usar nuestro conjunto de test para realizar predicciones.
- Pueden usarse distintas métricas para evaluar la precisión del modelo, por ejemplo, una matriz de confusión para mostrar los resultados.
- Finalmente se guarda el modelo.

Si bien todo esto puede realizarse directamente con NumPy o SciPy, o mismo con Python puro, es mucho más sencillo con Scikit-Learn.

1.4. APRENDIZAJE SUPERVISADO VS NO SUPERVISADO

Hablemos del **aprendizaje supervisado**.

Una manera fácil de empezar a comprender el concepto de aprendizaje supervisado es mirando directamente las palabras que lo componen. Supervisar, significa observar y dirigir la ejecución de una tarea, proyecto o actividad. Obviamente no vamos a estar supervisando a una persona, en su lugar supervisaremos un modelo de aprendizaje automático que podría ser capaz de producir regiones de clasificación como vemos en la figura siguiente.

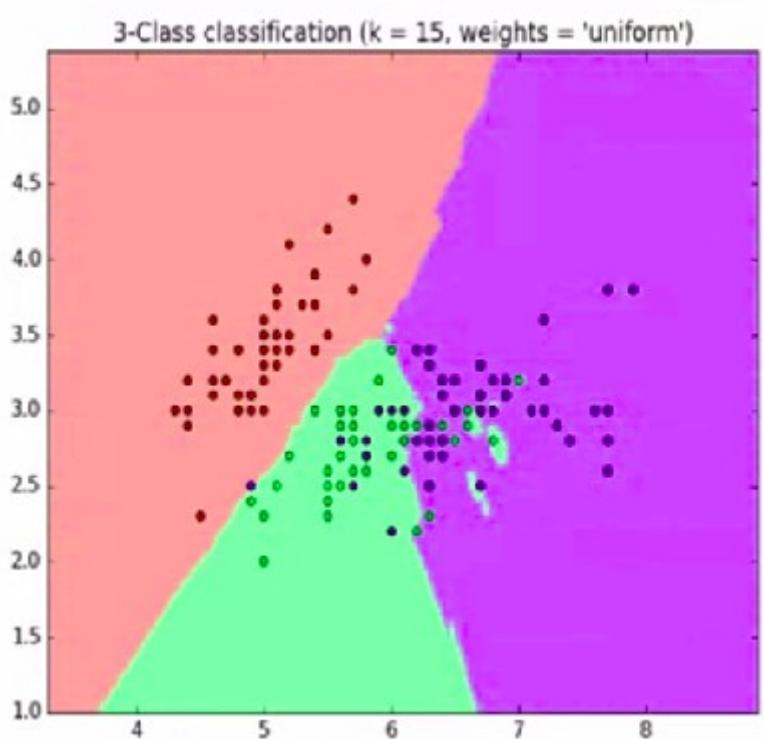


Figura. Aprendizaje supervisado.

Cómo supervisamos un modelo de ML?

Lo hacemos enseñándole al modelo, es decir, cargamos el modelo con conocimiento para que podamos predecir instancias futuras.

Entonces, cómo le enseñamos a un modelo?

Lo entrenamos con algunos datos de un **conjunto de datos etiquetado**.

Cómo es un conjunto de datos etiquetado?

Podría verse como se muestra en la figura siguiente. El ejemplo toma datos del conjunto de datos sobre el cáncer. Hay datos históricos de los pacientes y ya conocemos la clase de cada fila.

Los nombres de algunas de las columnas son:

- Espesor de grumos.
- Uniformidad del tamaño de las celdas.
- Uniformidad de la forma celular.
- Adhesión marginal.
- ...

Los nombres de las columnas se llaman **atributos**.

Las columnas se llaman **características**, que incluyen los datos.

Cada fila se llama **observación**.

ID	Clump	UnifSize	UnifShape	MargAdh	SingEpiSize	BareNuc	BlandChrom	NormNucl	Mit	Class
1000025	5	1	1	1	2	1	3	1	1	benign
1002945	5	4	4	5	7	10	3	2	1	benign
1015425	3	1	1	1	2	2	3	1	1	malignant
1016277	6	8	8	1	3	4	3	7	1	benign
1017023	4	1	1	3	2	1	3	1	1	benign
1017122	8	10	10	8	7	10		7	1	malignant
1018099	1	1	1	1	2	10	3	1	1	benign
1018561	2	1	2	H	2	1	3	1	1	benign
1033078	2	1	1	1	2	1	1	1	5	benign
1033078	4	2	1	1	2	1	2	1	1	benign

Figura. Datos etiquetados.

Observe la etiqueta (columna Class), su tipo de datos es categórico (benign o malignant). Esto es así porque este conjunto de datos está hecho para la clasificación.

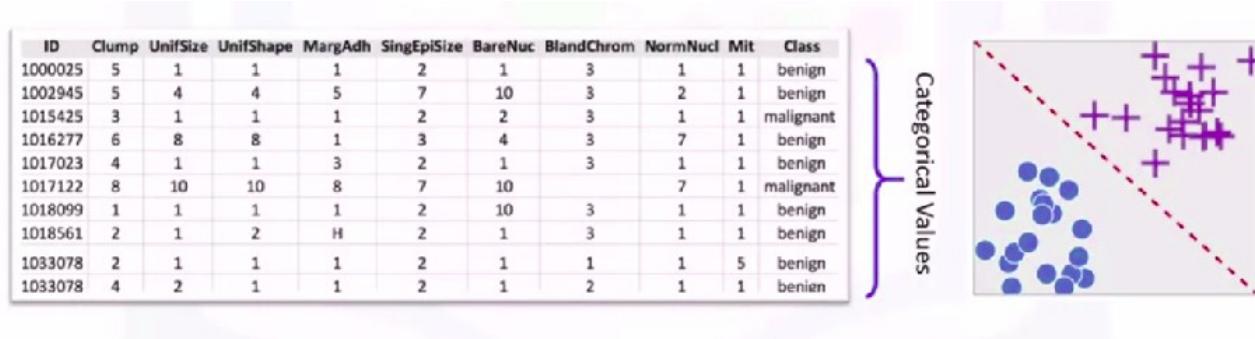


Figura. Ejemplo de clasificación.

Existen 2 tipos de técnicas de ML supervisadas:

- **Clasificación**
 - Es el proceso de predecir una etiqueta de clase discreta o categoría.
- **Regresión**
 - Es el proceso de predecir un valor continuo en lugar de uno discreto.

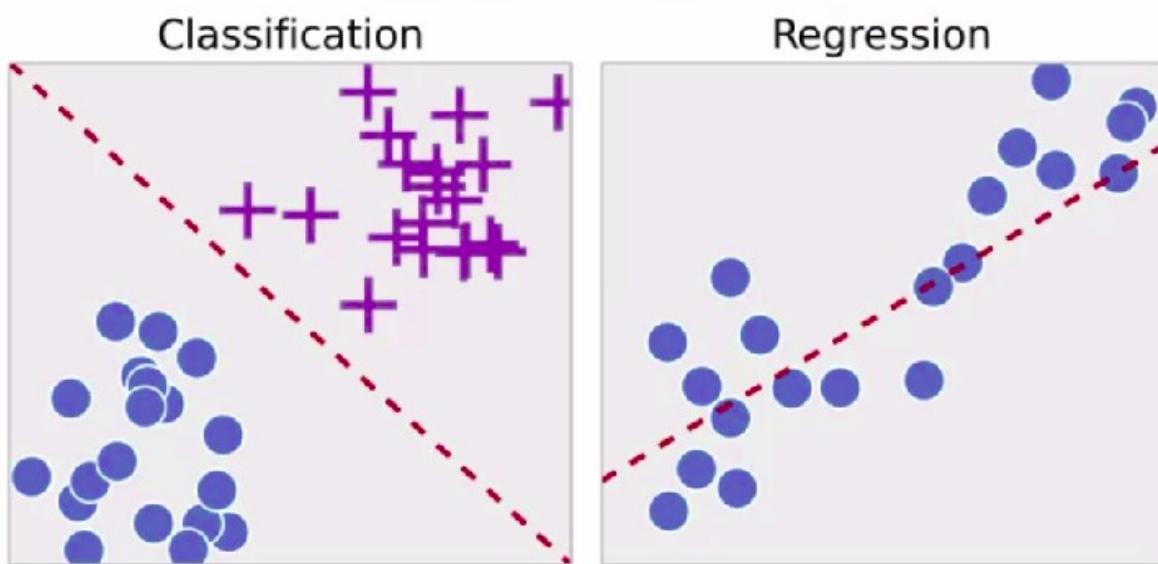


Figura. Tipos de aprendizaje supervisado.

Observe el conjunto de datos de la figura siguiente. Está relacionado con las emisiones de CO₂ de diferentes coches. Incluye: tamaño de motor, cilindros, consumo de combustible y emisión de CO₂ de varios modelos de autos. Dado este conjunto de datos, puede utilizar

la regresión para predecir la emisión de CO₂ de un coche nuevo utilizando otros campos como el tamaño del motor o el número de cilindros.

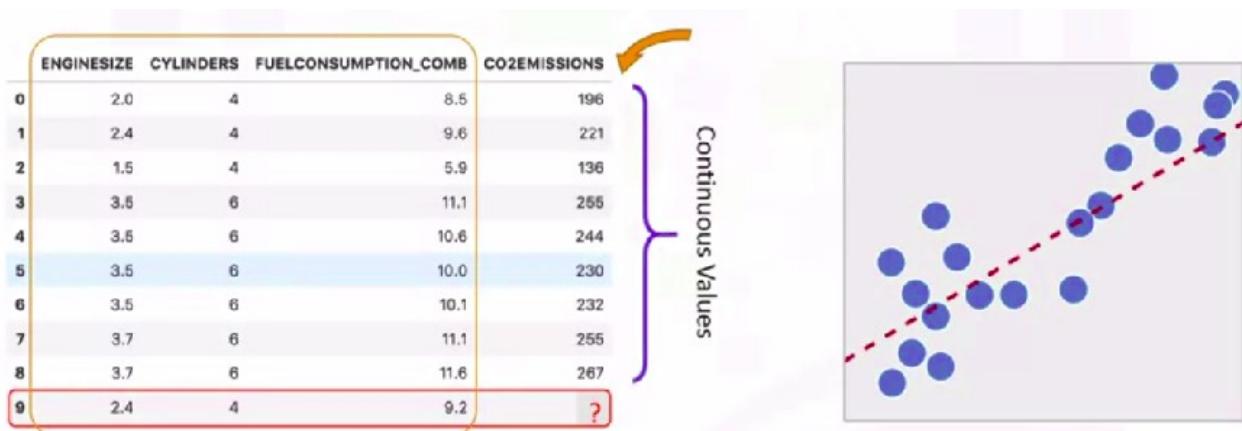


Figura. Ejemplo de regresión.

Veamos ahora el **aprendizaje no supervisado**.

No supervisamos el modelo, dejamos que el modelo funcione por sí solo para descubrir información que puede no ser visible para el ojo humano.

Significa que el algoritmo no supervisado se entrena en el conjunto de datos, y saca conclusiones sobre los datos sin etiquetar.

En términos generales, el aprendizaje no supervisado tiene algoritmos más difíciles que el aprendizaje supervisado, ya que sabemos poca o ninguna información sobre los datos, o los resultados que se esperan.

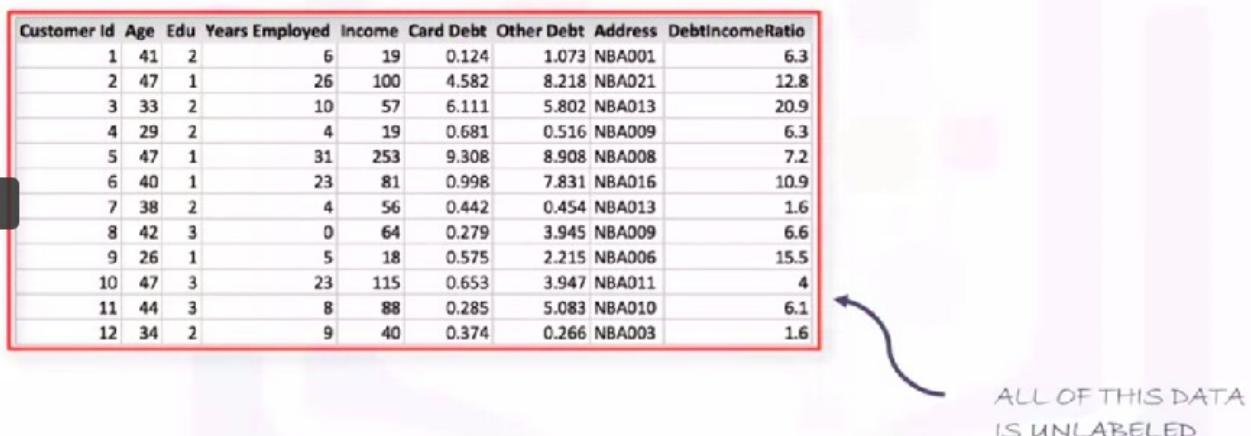


Figura. Aprendizaje no supervisado.

Las técnicas de aprendizaje no supervisado más utilizadas son:

- Reducción de la dimensionalidad.
 - Reduce las funciones redundantes para facilitar la clasificación.
- Estimación de densidad
 - Se utiliza para explorar los datos y encontrar alguna estructura dentro de ellos.
- Análisis de cesta de mercado (Market basket analysis)
 - Es una técnica de modelado basada en la teoría de que si usted compra un determinado grupo de artículos, es más probable que compre otro grupo de artículos.
- Clustering (agrupación)
 - Se usa para agrupar puntos de datos u objetos que de alguna manera son similares.
 - Se utiliza principalmente para:
 - Descubrir estructuras.
 - Summarization
 - Detección de anomalías.
 - Algunas aplicaciones del análisis de clústeres son:
 - Segmentación de clientes en un banco.
 - Ayudar a un individuo a organizar en grupo su música favorita.

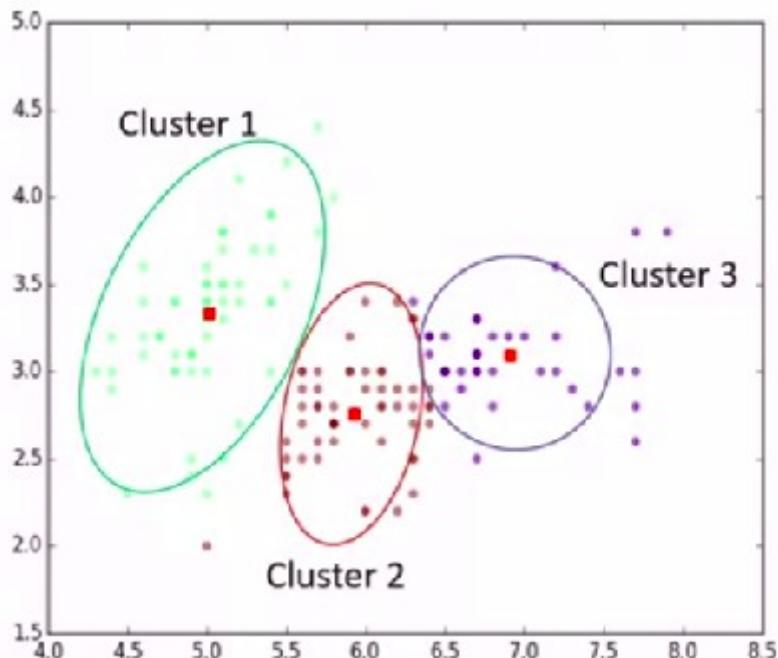


Figura. Clustering.

Comparemos:

APRENDIZAJE SUPERVISADO	APRENDIZAJE NO SUPERVISADO
<ul style="list-style-type: none">• Se ocupa de datos etiquetados.• Tenemos algoritmos de ML para clasificación y regresión.	<ul style="list-style-type: none">• Trata con datos sin etiqueta.• Tenemos métodos como el clustering.• Tiene menos modelos y métodos de evaluación que el aprendizaje supervisado.

Figura. Comparación entre los tipos de aprendizaje.

PARTE II. REGRESIÓN LINEAL

2.1. INTRODUCCIÓN A LA REGRESIÓN

Observe el siguiente dataset.

	ENGINESIZE	CYLINDERS	FUELCONSUMPTION_COMB	CO2EMISSIONS
0	2.0	4	8.5	196
1	2.4	4	9.6	221
2	1.5	4	5.9	136
3	3.5	6	11.1	255
4	3.5	6	10.6	244
5	3.5	6	10.0	230
6	3.5	6	10.1	232
7	3.7	6	11.1	255
8	3.7	6	11.6	267
9	2.4	4	9.2	?

Figura. Dataset de ejemplo.

El dataset pertenece a las emisiones de CO2 de diferentes autos. Incluye tamaño del motor, número de cilindros, consumo de combustibles y emisiones de CO2 de varios modelos.

La pregunta es:

dado este dataset, podemos predecir la emisión de CO2 de un auto utilizando otros campos como tamaño del motor o número de cilindros?

Asumamos que tenemos datos históricos de diferentes autos y que un auto tal como el de la fila 9 no ha sido fabricado aún, pero queremos estimar su emisión de CO2 aproximada luego de su producción. Es posible?

Podemos usar métodos de regresión para predecir valores continuos, como la emisión de CO2 usando otras variables. De hecho, la regresión es el proceso de predecir valores continuos.

En la regresión hay 2 **tipos de variables**:

- La **variable dependiente**
 - Es el estado, el target (objetivo), la variable que intentamos predecir.
- Una o más **variables independientes**
 - También se conocen como variables explicativas y puede verse como la causa de tales estados.

Las variables independientes se muestran convencionalmente como X y las dependientes como Y. Un modelo de regresión relaciona una variable independiente Y con una o más variables independientes X.

El punto clave en la regresión es que **la variable dependiente debe ser continua**; no puede ser discreta. Sin embargo, la variable independiente puede ser medida en una escala continua o discreta.

Lo que queremos aquí es usar los datos históricos de algunos autos usando una o más de sus características y a partir de ello construir un modelo. Utilizaremos la regresión para construir un modelo de estimación; el modelo será utilizado para predecir la emisión esperada de CO₂ para un auto nuevo o desconocido.

	X: Independent variable			Y: Dependent variable
	ENGINESIZE	CYLINDERS	FUELCONSUMPTION_COMB	CO2EMISSIONS
0	2.0	4	8.5	196
1	2.4	4	9.6	221
2	1.5	4	5.9	136
3	3.5	6	11.1	255
4	3.5	6	10.6	244
5	3.5	6	10.0	230
6	3.5	6	10.1	232
7	3.7	6	11.1	255
8	3.7	6	11.6	267
9	2.4	4	9.2	?

Figura. Variables típicas en la regresión.

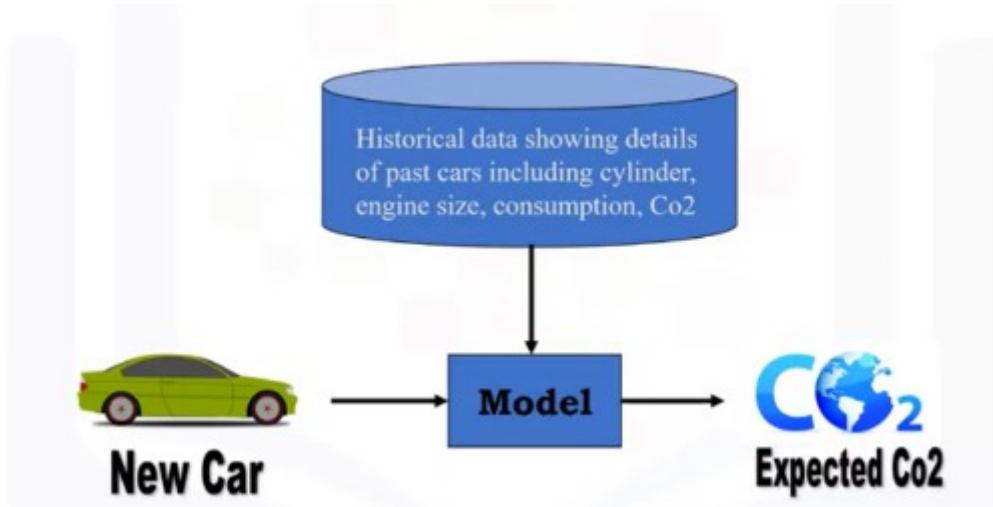


Figura. Modelo de regresión.

Básicamente, hay 2 **tipos de modelos de regresión**:

- **Simple**
 - Se utiliza 1 variable independiente para estimar la dependiente.
 - Puede ser lineal o no lineal.
 - Por ejemplo, predecir la emisión de CO2 usando el tamaño del motor.
- **Múltiple**
 - Hay más de una variable independiente presente.
 - Por ejemplo, predecir la emisión de CO2 usando el tamaño del motor y el número de cilindros.
 - Puede ser lineal o no lineal.

La linealidad de la regresión se basa en la naturaleza de la relación entre la variable independiente y la(s) independiente(s).

Veamos algunas aplicaciones de la regresión:

- Pronóstico de ventas
 - Predecir las ventas anuales de una persona a partir de variables independientes como la edad, educación y años de experiencia laboral.
- En psicología
 - Para determinar la satisfacción individual basado en factores demográficos y psicológicos.

- Predecir el precio de una casa en un área, basado en su tamaño, el número de dormitorios, etc.
- Ingreso de una persona a partir de variables independientes como horas de trabajo, educación, sexo, etc.

Hay muchos algoritmos de regresión, algunos de los cuales trataremos en este texto.

2.2. REGRESIÓN LINEAL SIMPLE

Tomemos como ejemplo el siguiente dataset de emisiones de CO2.

	ENGINESIZE	CYLINDERS	FUELCONSUMPTION_COMB	CO2EMISSIONS
0	2.0	4	8.5	196
1	2.4	4	9.6	221
2	1.5	4	5.9	136
3	3.5	6	11.1	255
4	3.5	6	10.6	244
5	3.5	6	10.0	230
6	3.5	6	10.1	232
7	3.7	6	11.1	255
8	3.7	6	11.6	267
9	2.4	4	9.2	?

Figura. Dataset de ejemplo.

La pregunta es:

podemos predecir el CO2 emisión de un coche utilizando otro campo como el tamaño del motor ?

Sí, y una forma es utilizar la regresión lineal.

La **regresión lineal** es la aproximación de un modelo lineal utilizado para describir la relación entre dos o más variables.

En regresión lineal simple hay 2 variables, una dependiente y una independiente. El punto clave de la regresión es que nuestro valor dependiente debe ser continuo, cualquiera sea su escala de medida.

Hay 2 tipos de modelos de regresión lineal:

- Simple
 - Se utiliza una variable independiente para estimar una variable dependiente.
 - Por ejemplo, la predicción de emisiones de CO2 utilizando la variable de tamaño del motor.
- Múltiple
 - Hay más de una variable independiente presente.
 - Por ejemplo, la predicción de emisiones de CO2 utilizando el tamaño del motor y cilindros de coches

Aquí estudiaremos la regresión lineal simple.

Volvamos a nuestro conjunto de datos.

En la figura siguiente tomamos el tamaño del motor como variable independiente y la variable objetivo son las emisiones de CO₂.

La gráfica de dispersión muestra que hay una relación entre las variables. Mediante la regresión lineal se puede ajustar una recta para los datos. Puede verse que a medida que el tamaño del motor aumenta, aumentan las emisiones.

	ENGINESIZE	CYLINDERS	FUELCONSUMPTION_COMB	CO2EMISSIONS
0	2.0	4	8.5	196
1	2.4	4	9.6	221
2	1.5	4	5.9	136
3	3.5	6	11.1	255
4	3.5	6	10.6	244
5	3.5	6	10.0	230
6	3.5	6	10.1	232
7	3.7	6	11.1	255
8	3.7	6	11.6	267
9	2.4	4	9.2	?

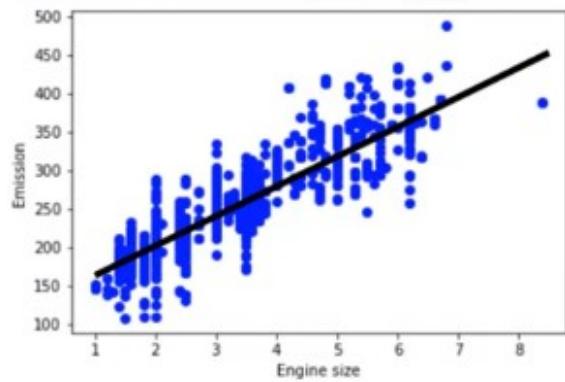


Figura. Gráfico de dispersión y regresion lineal.

Podemos usar esta recta, para realizar predicciones, por ejemplo, para un coche con tamaño del motor de 2.4 la emisión que se predice es de 214.

	ENGINESIZE	CYLINDERS	FUELCONSUMPTION_COMB	CO2EMISSIONS
0	2.0	4	8.5	196
1	2.4	4	9.6	221
2	1.5	4	5.9	136
3	3.5	6	11.1	255
4	3.5	6	10.6	244
5	3.5	6	10.0	230
6	3.5	6	10.1	232
7	3.7	6	11.1	255
8	3.7	6	11.6	267
9	2.4	4	9.2	?

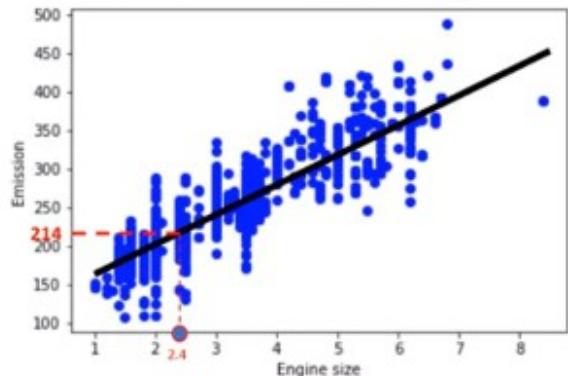


Figura. Prediciendo valores.

Encontremos la recta de ajuste.

Sean:

- Y: la variable dependiente o target (la emisión de CO₂ en este caso)
- X₁: la variable independiente (tamaño del motor en este caso)

La recta de ajuste es un polinomio, donde:

- \hat{y} : es la variable independiente
- X₁: es la variable independiente
- θ_0 y θ_1 : son los parámetros de la recta que debemos ajustar. También se llaman coeficientes de la ecuación.
 - θ_1 es la pendiente o gradiente.
 - θ_0 es la intercepción.

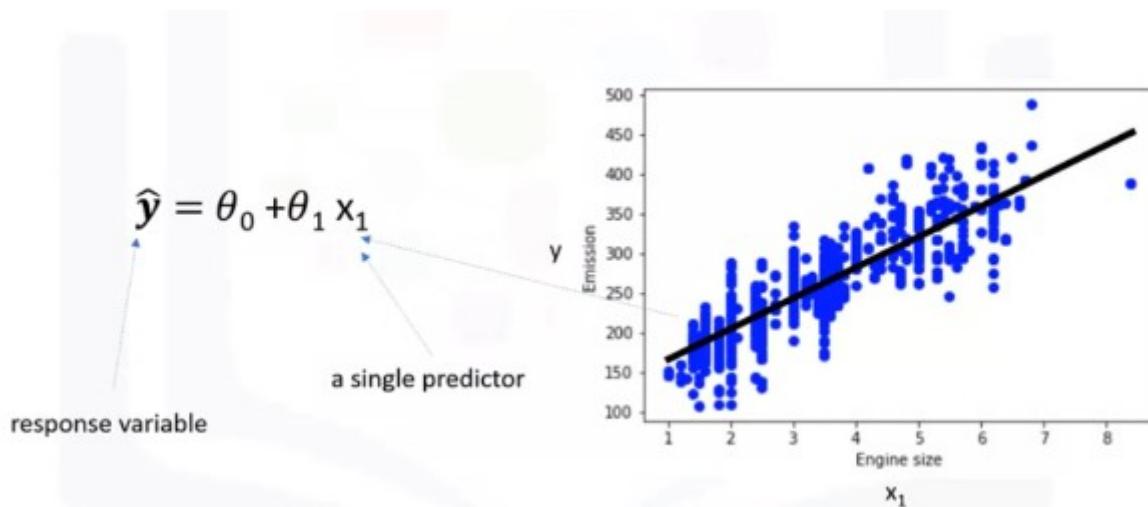


Figura. Recta de ajuste.

La pregunta ahora es cómo obtener la recta de mejor ajuste, es decir, cuáles son los parámetros que definen la recta que mejor se ajusta a los datos.

Por un momento supongamos que encontramos la recta de mejor ajuste. Para ver qué tan bien ajusta nuestra recta comparamos el valor predicho con el valor real. La diferencia se conoce como **error residual**.

$x_1 = 5.4$ independent variable
 $y = 250$ actual Co2 emission of x_1

$$\hat{y} = \theta_0 + \theta_1 x_1$$

$$\hat{y} = 340 \text{ the predicted emission of } x_1$$

$$\begin{aligned}\text{Error} &= y - \hat{y} \\ &= 250 - 340 \\ &= -90\end{aligned}$$

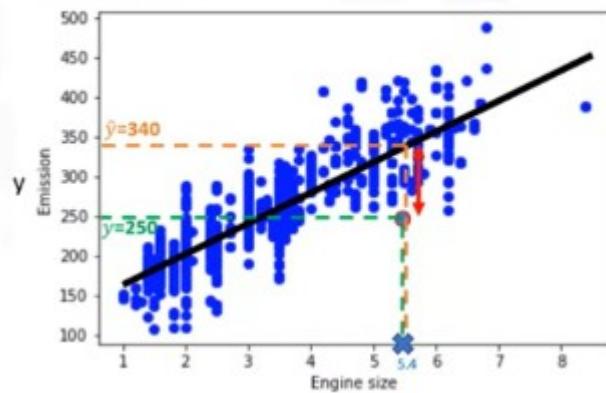


Figura. Error residual.

Podemos decir que el error es la distancia desde el punto de datos a la recta de ajuste; la media de todos los errores residuales muestra qué tan bien encaja la recta con el conjunto de datos. Esto, está dado matemáticamente por el MSE (Minimun Square Error). Nuestro objetivo es minimizar el MSE; es decir, debemos hallar los parámetros (θ_0 y θ_1 en este caso) que minimizan el MSE.

$x_1 = 5.4$ independent variable
 $y = 250$ actual Co2 emission of x_1

$$\begin{aligned}\hat{y} &= \theta_0 + \theta_1 x_1 \\ \hat{y} &= 340 \text{ the predicted emission of } x_1\end{aligned}$$

$$\begin{aligned}\text{Error} &= y - \hat{y} \\ &= 250 - 340 \\ &= -90\end{aligned}$$

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

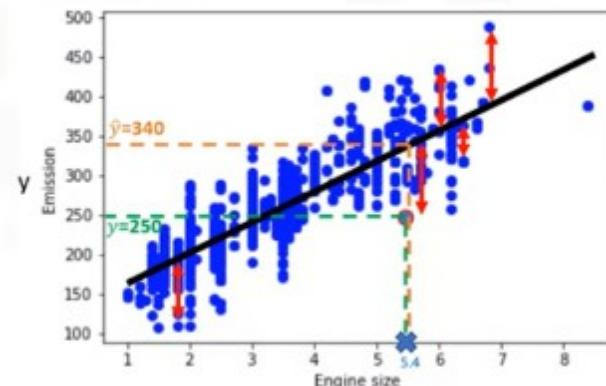


Figura. MSE.

Los parámetros pueden obtenerse matemáticamente, no obstante las bibliotecas de ML de Python lo hacen automáticamente.

Obtenida la ecuación de la recta, podemos utilizarla para predecir nuevos valores.

Las ventajas de la regresión lineal son:

- Es rápida.
- No requiere ajuste de parámetros.
- Es fácil de entender y altamente interpretable.

2.3. EVALUACIÓN DEL MODELO EN REGRESIÓN LINEAL

Vamos a estudiar la evaluación de modelos.

El objetivo de la regresión es construir un modelo para predecir con precisión un caso desconocido. Para ello, tenemos que realizar una evaluación de regresión después de la construcción del modelo y así saber qué tan bueno es.

Veremos **2 enfoques**:

- **Entrenar y probar en el mismo conjunto de datos.**
- **Train test/split** (división en conjuntos de prueba y entrenamiento)

Veamos cómo sería el enfoque de **entrenar y probar en el mismo conjunto de datos**.

Seleccionamos una parte de nuestro conjunto de datos para pruebas. Por ejemplo, supongamos que tenemos 10 registros en nuestro conjunto de datos.

- Utilizamos todo el conjunto de datos para el entrenamiento y construimos un modelo usándolo.
- Seleccionamos una pequeña parte del conjunto de datos, por ejemplo de las filas 6 a la 9, pero sin etiquetar.
 - Este se llama conjunto de pruebas, que tiene las etiquetas, pero las mismas no se utilizan para la predicción, sólo como verdad de fondo.
 - Las etiquetas se denominan valores reales del conjunto de prueba.
- Pasamos el conjunto de características de la parte de prueba a nuestro modelo construido y predecimos los valores objetivo.
- Comparamos los valores predichos por nuestro modelo con los valores reales en el conjunto de pruebas; esto indica cuán preciso es nuestro modelo.

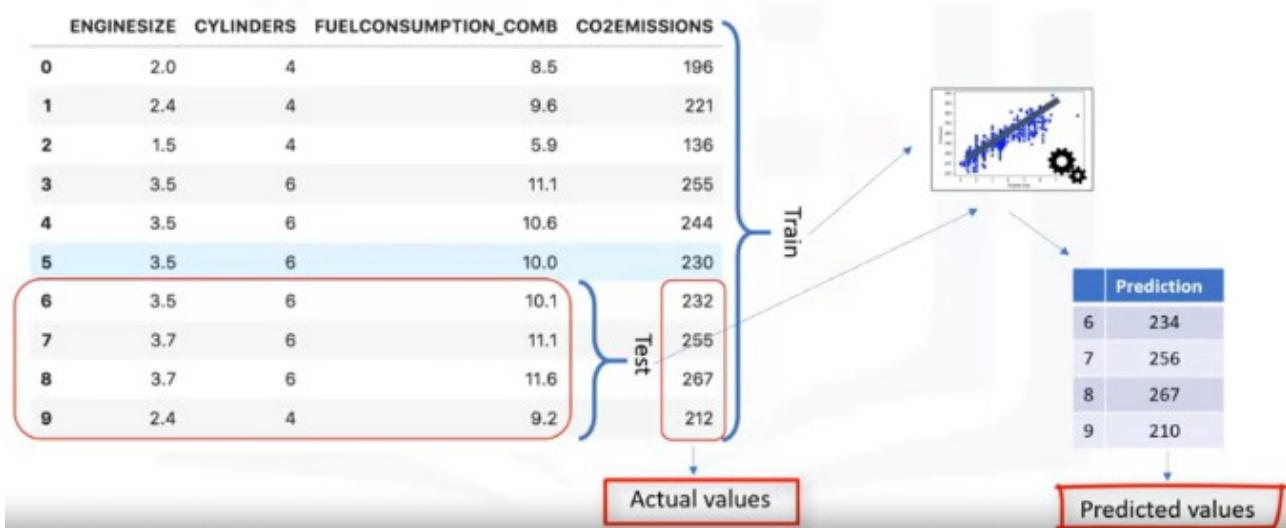


Figura. Entrenar y probar en el mismo conjunto de datos.

Existen diferentes métricas para informar la precisión del modelo. Una de las más simples es la diferencia media entre los valores previstos y reales para todas las filas.

$$\text{Error} = \frac{(232 - 234) + (255 - 256) + \dots}{4}$$

$$\text{Error} = \frac{1}{n} \sum_{j=1}^n |y_j - \hat{y}_j|$$

Figura. Métrica de evaluación del modelo.

Así, este enfoque, tal cual lo dice su nombre, entrena el modelo con todo el conjunto de datos y luego lo prueba con una parte del mismo conjunto de datos.

Este enfoque probablemente tendría una alta precisión de entrenamiento y una baja precisión fuera de la muestra, ya que el modelo conoce todos los puntos de datos de la prueba.

La **precisión del entrenamiento** es el porcentaje de predicciones correctas que realiza el modelo al usar el conjunto de datos de prueba. Una alta precisión de entrenamiento no es necesariamente algo bueno, por ejemplo puede resultar en overfitting (el modelo está excesivamente entrenado para el conjunto de datos y puede capturar ruido y producir un modelo que no generalice bien).

La **precisión fuera de muestra** es el porcentaje de predicciones correctas que el modelo hace sobre los datos en los que el modelo no ha sido entrenado.

Es importante que nuestros modelos tengan una alta precisión fuera de la muestra porque el propósito de nuestro modelo es, por supuesto, hacer predicciones correctas sobre datos desconocidos.

Entonces, ¿cómo podemos mejorar la precisión fuera de la muestra?.

Una forma es utilizar otro enfoque llamado **división entrenamiento/test**. En este enfoque:

- Seleccionamos una parte de nuestro conjunto de datos para el entrenamiento, por ejemplo las fila de la 0 a la 5 y el resto se usa para test.
- El modelo se basa en el conjunto de entrenamiento.
- Luego, el conjunto de funciones de prueba se pasa al modelo para la predicción.
- Finalmente, los valores previstos para el conjunto de pruebas se comparan con los valores reales del conjunto de pruebas.

La división entrenamiento/test implica dividir el conjunto de datos en conjuntos de entrenamiento y pruebas respectivamente, que son mutuamente excluyentes. Después de lo cual, entrena con el conjunto de entrenamiento y prueba con el conjunto de pruebas. Esto proporcionará una evaluación más precisa de la precisión fuera de la muestra porque el conjunto de datos de prueba no forma parte del conjunto de datos que se ha utilizado para entrenar los datos.

Es más realista para los problemas del mundo real. Esto significa que conocemos el resultado de cada punto de datos en el conjunto de datos, por lo que es ideal para probar con. Dado que estos datos no se han utilizado para entrenar el modelo, el modelo no tiene conocimiento del resultado de estos puntos de datos. Entonces, en esencia, es realmente una prueba fuera de la muestra.

Sin embargo, asegúrese de entrenar su modelo con el conjunto de pruebas después, ya que no desea perder datos potencialmente valiosos.

El problema con la división entrenamiento/test es que depende en gran medida de los conjuntos de datos en los que se entrenaron y probaron los datos. La variación de esto hace que la división entrenamiento/test tenga una mejor predicción fuera de la muestra que la capacitación y las pruebas en el mismo conjunto de datos, pero todavía tiene algunos problemas debido a esta dependencia.

Otro modelo, llamado **validación cruzada del pliegue K (K-folds)** resuelve la mayoría de estos problemas.

¿Cómo soluciona una variación alta que resulta de una dependencia? Bueno, lo promedias.

Expliquemos un poco este método.

- Todo el conjunto de datos está representado por los puntos de la imagen en la parte superior izquierda de la figura siguiente.
- En el ejemplo, dividimos el conjunto de datos en K=4 pliegues.
- En el primer pliegue, por ejemplo, usamos el primer 25 por ciento del conjunto de datos para las pruebas y el resto para la capacitación. El modelo se construye utilizando el conjunto de entrenamiento y se evalúa utilizando el conjunto de pruebas.
- Luego, en la siguiente ronda o en el segundo pliegue, el segundo 25 por ciento del conjunto de datos se usa para probar y el resto para entrenar el modelo. Una vez más, se calcula la precisión del modelo.
- Continuamos por todos los pliegues. Por último, se promedió el resultado de las cuatro evaluaciones.
 - Es decir, la precisión de cada pliegue se promediará, teniendo en cuenta que cada pliegue es distinto, donde no se utilizan datos de entrenamiento en un pliegue en otro.



Figura. K-folds.

La validación cruzada del pliegue K en su forma más simple realiza múltiples divisiones de entrenamiento/prueba, utilizando el mismo conjunto de datos y donde cada división es diferente. Luego, el resultado es promedio para producir una precisión fuera de la muestra más consistente.

2.4. MÉTRICAS DE EVALUACIÓN EN MODELOS DE REGRESIÓN

Las métricas de evaluación se utilizan para explicar el rendimiento de un modelo.

Básicamente, podemos comparar los valores reales y los valores predecidos, para calcular la precisión de nuestro modelo de regresión.

Las métricas de evaluación proporcionan un papel clave en el desarrollo de un modelo, ya que proporciona información sobre las áreas que requieren mejoras.

Necesitamos definir qué es realmente un error.

En el contexto de la regresión, el error del modelo es la diferencia entre los puntos de datos y la línea de tendencia generada por el algoritmo.

Dado que hay varios puntos de datos, un error puede determinarse de varias maneras.

- El error absoluto medio MAE es la media del valor absoluto de los errores. Esta es la más fácil de entender de las métricas, ya que es solo el error promedio.
- El error cuadrado medio MSE es la media del error cuadrado.
 - Es más popular que un error absoluto medio porque el enfoque está orientado más hacia errores grandes.
 - Esto se debe a que el término cuadrado aumenta exponencialmente errores más grandes en comparación con los más pequeños.
- El RMSE es la raíz cuadrada del MSE.
 - Esta es una de las métricas de evaluación más populares porque el error cuadrado medio raíz es interpretable en las mismas unidades que el vector de respuesta o unidades y, por lo que es fácil relacionar su información.
- El RAE (Relative Absolute Error) toma el error absoluto total y lo normaliza dividiéndolo por el error absoluto total del predictor simple.
- El RSE es similar al RAE, pero es ampliamente adoptado por la comunidad científica de datos, ya que se utiliza para calcular R^2 .
- R^2 no es un error per se, pero es una métrica popular para la precisión del modelo.
 - Representa qué tan cerca están los valores de datos de la línea de regresión ajustada.
 - Cuanto más alto sea el R^2 , mejor se adaptará el modelo a sus datos.

Cada una de estas métricas se puede utilizar para cuantificar su predicción. La elección de la métrica depende de:

- El tipo de modelo.
- El tipo de datos.

- El dominio de conocimiento.

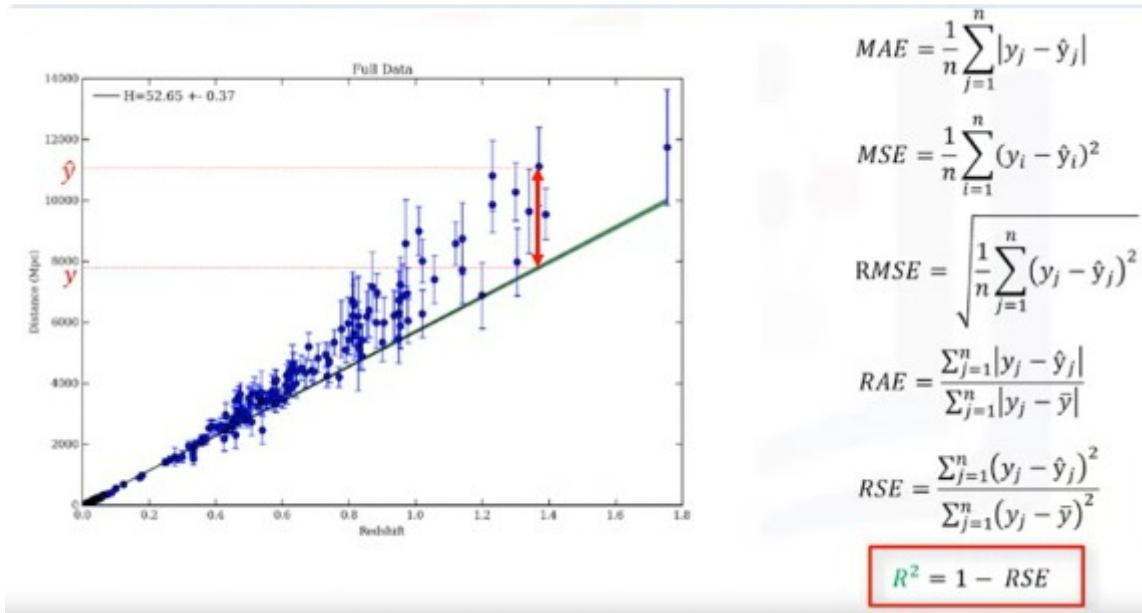


Figura. Métricas para el error.

2.5. REGRESIÓN LINEAL MÚLTIPLE

Veremos la **regresión lineal múltiple**.

Existen múltiples variables que predicen la emisión de CO₂. Cuando hay varias variables independientes presentes, el proceso se denomina regresión lineal múltiple. Por ejemplo, predecir las emisiones CO₂ utilizando el tamaño del motor y el número de cilindros en el motor del automóvil.

La regresión lineal múltiple es la extensión del modelo de regresión lineal simple.

Básicamente, hay dos **aplicaciones para regresión lineal múltiple**:

- El efecto de las variables independientes en la predicción.
 - Por ejemplo, ¿el tiempo de revisión, la ansiedad de las pruebas, la asistencia a las conferencias y el género tienen algún efecto en el desempeño de los exámenes de los estudiantes?
- Predecir el impacto de los cambios.
 - Entender cómo cambia la variable dependiente cuando cambiamos las variables independientes.
 - Por ejemplo, si estuviéramos revisando los datos de salud de una persona, una regresión lineal múltiple puede indicarle cuánto sube o baja la presión arterial de esa persona por cada unidad de aumento o disminución en el índice de masa corporal de un paciente manteniendo otros factores constantes.

Como es el caso de la regresión lineal simple, la regresión lineal múltiple es un método para predecir una variable continua.

La regresión lineal múltiple utiliza múltiples variables llamadas variables independientes o predictores que mejor predicen el valor de la variable objetivo que también se llama variable dependiente. El valor objetivo Y es una combinación lineal de variables independientes X. Por ejemplo, puede predecir cuánto CO₂ puede admitir un automóvil debido a variables independientes como el tamaño del motor del automóvil, el número de cilindros y el consumo de combustible.

La regresión lineal múltiple es muy útil porque puede examinar qué variables son predictores significativos de la variable de resultado. Además, puede averiguar cómo afecta cada entidad a la variable de resultado.

Como es el caso de la regresión lineal simple, si logra construir un modelo de regresión de este tipo, puede usarlo para predecir la cantidad de emisión de un caso desconocido.

Matemáticamente, la ecuación puede mostrarse en forma vectorial, utilizando el vector de parámetros y el del conjunto de entidades.

La ecuación se escribe como $Y = \theta^T X$, donde theta se conoce como vector de pesos y X es el conjunto de características, que en este caso representa el coche (por ejemplo X1 para el tamaño del motor, X2 para los cilindros, etc). El primer elemento del conjunto de entidades se establece en 1.

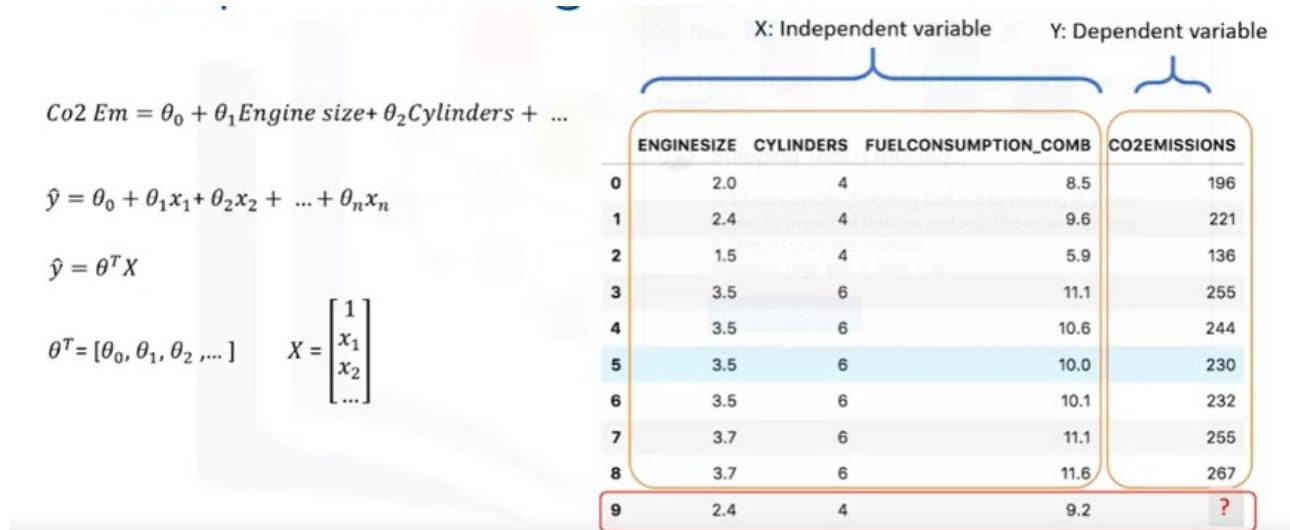


Figura. Regresión Lineal Múltiple.

En un espacio unidimensional, la ecuación anterior es la de una recta, que es la que usamos en regresión lineal simple. En dimensiones más altas es un plano o un hiperplano. Así, la idea es encontrar el hiperplano que mejor se ajuste a los datos.

Para ello debemos estimar los valores para el vector theta que mejor predice el valor del campo objetivo en cada fila. Para lograr este objetivo, tenemos que minimizar el error de la predicción.

¿Cómo encontramos los parámetros óptimos?

Una forma es minimizar el MSE, los métodos más comunes son:

- Mínimos cuadrados
 - Los mínimos cuadrados ordinarios tratan de estimar los valores de los coeficientes minimizando el error cuadrado medio.
 - Este enfoque utiliza los datos como una matriz y utiliza operaciones de álgebra lineal para estimar los valores óptimos para el theta.

- El problema con esta técnica es la complejidad temporal del cálculo de las operaciones de matriz, ya que puede tardar mucho tiempo en terminarse. Cuando el número de filas del conjunto de datos es menor que 10.000, puede pensar en esta técnica como una opción. Sin embargo, para valores mayores, debe probar otros enfoques más rápidos.
- Enfoque de optimización
 - Es decir, puede utilizar un proceso de optimización de los valores de los coeficientes minimizando iterativamente el error del modelo en sus datos de entrenamiento.
 - Por ejemplo, puede usar el descenso de gradiente que comienza la optimización con valores aleatorios para cada coeficiente, luego calcula los errores e intenta minimizarlo a través del cambio y de los coeficientes en múltiples iteraciones.
 - El descenso de degradado es un enfoque adecuado si tiene un conjunto de datos grande.

Existen también otros enfoques.

Después de encontrar los mejores parámetros para su modelo, puede ir a la fase de predicción que es tan simple como resolver la ecuación para un conjunto específico de entradas.

La regresión lineal múltiple estima la importancia relativa de los predictores, en el ejemplo de la figura siguiente el cilindro tiene un mayor impacto en las cantidades de emisiones CO₂ en comparación con el tamaño del motor (tiene un coeficiente mayor).

	ENGINESIZE	CYLINDERS	FUELCONSUMPTION_COMB	CO2EMISSIONS	
0	2.0	4	8.5	196	
1	2.4	4	9.6	221	
2	1.5	4	5.9	136	
3	3.5	6	11.1	255	
4	3.5	6	10.6	244	
5	3.5	6	10.0	230	
6	3.5	6	10.1	232	
7	3.7	6	11.1	255	
8	3.7	6	11.6	267	
9	2.4	4	9.2	?	

$$\hat{y} = \theta^T X$$

$$\theta^T = [125, 6.2, 14, \dots]$$

$$\hat{y} = 125 + 6.2x_1 + 14x_2 +$$

$$Co2Em = 125 + 6.2 \text{EngSize} + 14 \text{Cylinders} + \dots$$

$$Co2Em = 125 + 6.2 \times 2.4 + 14 \times 4 + \dots$$

$$Co2Em = 214.1$$

Figura. Regresión Lineal Múltiple.

¿ La adición de variables independientes a un modelo de regresión lineal múltiple siempre aumenta la precisión del modelo?

Básicamente, agregar demasiadas variables independientes sin ninguna justificación teórica puede dar lugar al overfitting.

El overfitting es un problema real porque el modelo es demasiado complicado para su conjunto de datos y no generaliza bien como para ser utilizado para la predicción.

Hay varias formas de evitar el overfitting.

¿ Las variables independientes deben ser continuas?

Las variables categóricas independientes se pueden incorporar en un modelo de regresión convirtiéndolas en variables numéricas. Por ejemplo, dada una variable binaria como el tipo de coche, el código ficticio cero para manual y uno para automóviles automáticos.

Como último punto, recuerde que la regresión lineal múltiple es un tipo específico de regresión lineal. Por lo tanto, debe haber una relación lineal entre la variable dependiente y cada una de sus variables independientes. Hay una serie de formas de comprobar la relación lineal. Por ejemplo, puede utilizar gráficos de dispersión y, a continuación, comprobar visualmente la linealidad. Si la relación mostrada en el gráfico de dispersión no es lineal, entonces debe usar regresión no lineal.

2.6. REGRESIÓN NO LINEAL

Vamos a cubrir los conceptos básicos de regresión no lineal.

Los datos de la figura siguiente corresponden al producto interno bruto o PIB de China de 1960-2014. La primera columna son los años y la segunda es el ingreso interno bruto anual correspondiente de China en dólares estadounidenses para ese año. También se grafican los datos.

Nos preguntamos:

¿Se puede predecir el PIB sobre la base del tiempo?

¿Podemos usar una regresión lineal simple para modelar?

Si los datos muestran una tendencia curvada , la regresión lineal no produciría resultados muy precisos en comparación con una regresión no lineal. Simplemente porque, como su nombre lo indica, la regresión lineal supone que los datos son lineales.

La gráfica de dispersión muestra que parece haber una fuerte relación entre el PIB y el tiempo, pero la relación no es lineal. Como se puede ver, el crecimiento comienza lentamente y , a partir de 2005, el crecimiento es muy significativo. Por último, se desacelera ligeramente en la década de 2010. Parece una función logística o exponencial. Por lo tanto, requiere un método especial de estimación del procedimiento de regresión no lineal.

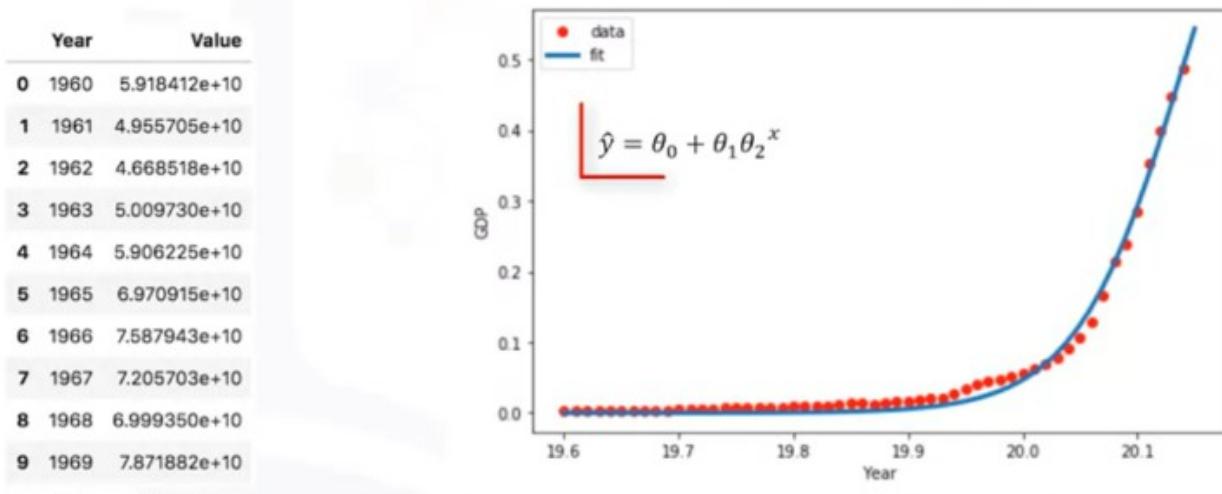


Figura. Ejemplo donde debe aplicarse la regresión no lineal.

Existen muchas regresiones diferentes que se pueden usar para adaptarse a cualquier aspecto del conjunto de datos. Se puede ver una línea de regresión cuadrática y cúbica

aquí, y puede seguir y seguir en grados infinitos. En esencia, podemos llamar a todas estas **regresiones polinómicas**, donde la relación entre la variable independiente X y la variable dependiente Y se modela como un polinomio de grado n en X.

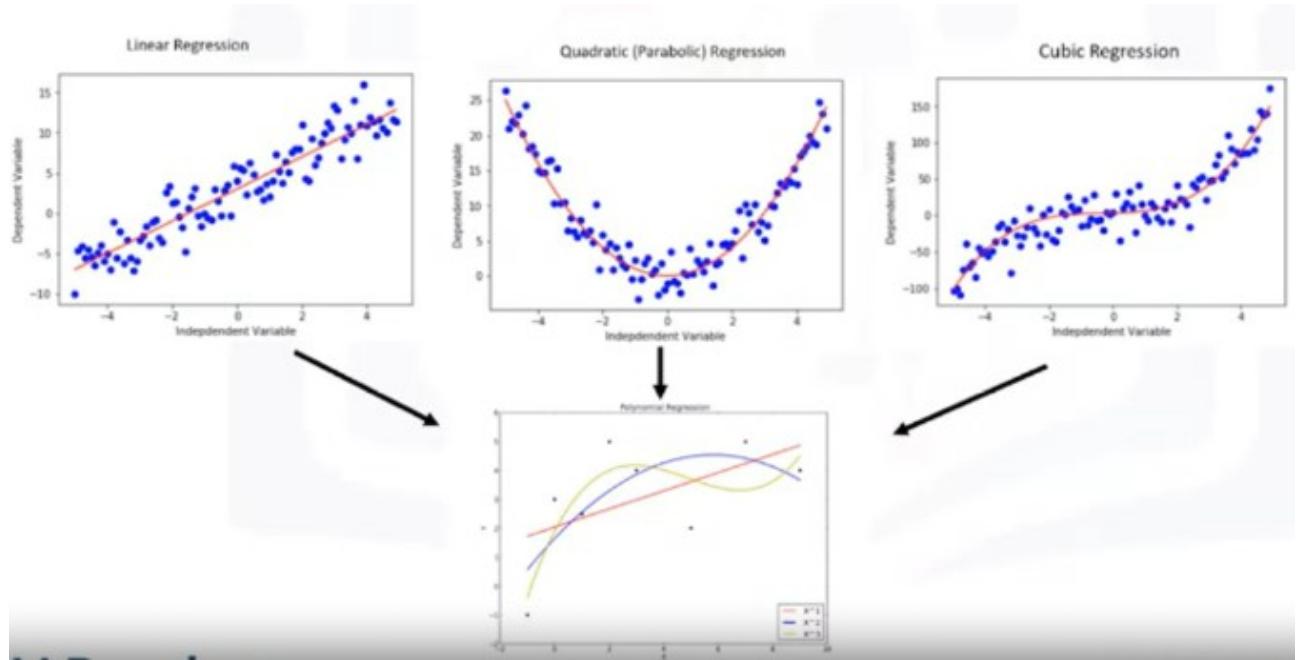


Figura. Regresión polinómica.

Es importante elegir una regresión que se ajuste mejor a los datos.

Un modelo de regresión polinómica se puede expresar como una regresión lineal.

Por ejemplo, la siguiente ecuación:

$$\hat{y} = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$$

puede escribirse, mediante un cambio de variables, como:

$$\begin{aligned}x_1 &= x \\x_2 &= x^2 \\x_3 &= x^3\end{aligned}$$

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$$

El nuevo modelo es lineal en los parámetros a estimar, por lo tanto para su resolución (obtener los parámetros theta) podemos usar los métodos ya conocidos, como mínimos cuadrados.

Entonces, la regresión no lineal es un método para modelar una relación no lineal entre la variable dependiente y un conjunto de variables independientes.

Para que un modelo se considere no lineal, Yhat debe ser una función NO lineal de los parámetros θ , no necesariamente de las características X.

Una ecuación no lineal, puede ser la forma de exponencial, logarítmico y logístico, o de muchos otros tipos.

¿Cómo puedo saber si un problema es lineal o no lineal?

- Podemos averiguar visualmente si la relación es o no lineal.
 - Lo mejor es trazar gráficos bivariados de variables de salida con cada variable de entrada.
- Se puede calcular el coeficiente de correlación entre variables independientes y dependientes, y si, para todas las variables, es 0.7 o superior, existe una tendencia lineal y, por lo tanto, no es apropiado ajustar una regresión no lineal.
- No podemos modelar con precisión la relación con parámetros lineales.

¿Cómo debería modelar mis datos si muestra no lineal en una gráfica de dispersión?

Puede:

- Utilizar regresión polinómica.
- Utilizar regresión no lineal.
- Transformar los datos.

PARTE 3. DIFERENTES MÉTODOS DE CLASIFICACIÓN

3.1. INTRODUCCIÓN A LA CLASIFICACIÓN

En ML, la **clasificación** es un enfoque de aprendizaje supervisado que puede ser considerado como un medio de categorizar o clasificar algunos elementos desconocidos en un conjunto discreto de clases.

La clasificación intenta aprender la relación entre un conjunto de variables de entidad y una variable objetivo de interés. El atributo objetivo en la clasificación es una variable categórica con valores discretos.

¿Cómo funcionan las clasificaciones y los clasificadores?

Dado un conjunto de puntos de datos de entrenamiento junto con las etiquetas de destino, la clasificación determina la etiqueta de clase para un caso de prueba sin etiqueta.

Veamos un ejemplo.

Una buena muestra de clasificación es la predicción de impago de un préstamo.

Supongamos que un banco está preocupado por la posibilidad de que los préstamos no sean reembolsado. Si los datos anteriores de impago del préstamo se pueden usar para predecir qué clientes tienen problemas para pagar préstamos, a ellos se les puede rechazar la solicitud o brindarle soluciones alternativas.

El predictor de impago de préstamo usa datos de impago de préstamos existentes, que tiene información sobre los clientes, como edad, ingresos, educación, etc para construir un clasificador.

Luego se le pasa un nuevo cliente y éste es etiquetado, por ejemplo con 0 o 1 (se le puede brindar préstamo o no).

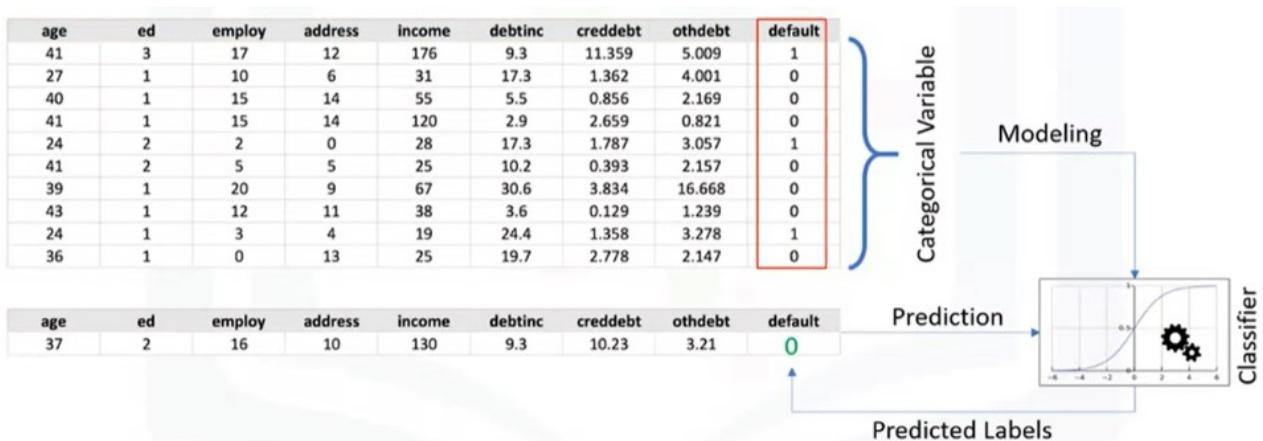


Figura. Ejemplo de clasificación.

Tenga en cuenta que este ejemplo específico se refería a un clasificador binario con dos valores. También podemos construir modelos de clasificador tanto para la clasificación binaria como para la clasificación multiclase. Por ejemplo, imagine que ha recopilado datos sobre un conjunto de pacientes, todos los cuales sufrieron la misma enfermedad. Durante su tratamiento, cada paciente respondió a uno de los tres medicamentos. Puede utilizar este dataset etiquetado con un algoritmo de clasificación para crear un modelo de clasificación. Luego puede usarlo para averiguar qué medicamento podría ser apropiado para un futuro paciente con la misma enfermedad. Como puede ver, es una muestra de clasificación multiclase.



Figura. Clasificación multiclase.

La clasificación también tiene diferentes casos de uso comercial, por ejemplo:

- Categoría de un cliente.
- Detección de churn (predecir si un cliente cambiará de proveedor o marca)
- Predecir si un cliente responde o no a una campaña publicitaria concreta.

La clasificación de datos tiene varias aplicaciones en una amplia variedad de industrias. Básicamente, muchos problemas se pueden expresar como asociaciones entre variables de entidad y objetivo, especialmente cuando se dispone de datos etiquetados. Esto proporciona una amplia gama de aplicabilidad para la clasificación, por ejemplo:

- Filtrado de correo electrónico.
- Reconocimiento de voz.

- Reconocimiento de escritura a mano.
- Identificación biométrica.
- Clasificación de documentos y mucho más.

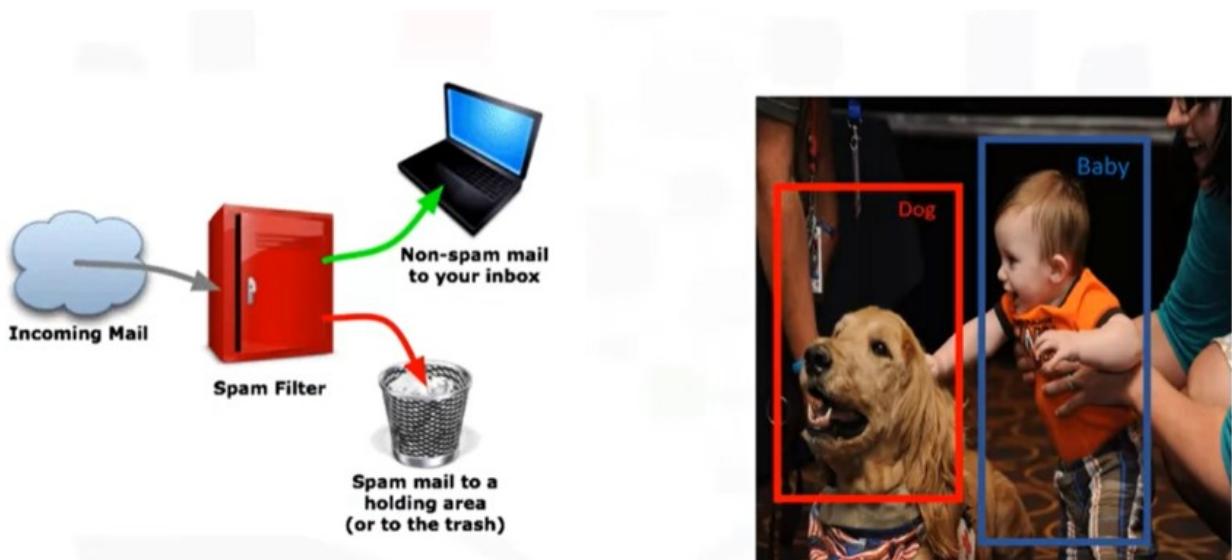


Figura. Aplicaciones de la clasificación.

Los algoritmos de clasificación y aprendizaje automático incluyen:

- Árboles de decisión.
- Naive Bayes.
- Discriminante lineal.
- K-vecinos más cercanos.
- Regresión logística.
- Redes neuronales.
- Máquinas vectoriales de soporte.

3.2. K-NEAREST NEIGHBORS (K VECINOS MÁS CERCANOS)

Veremos el algoritmo de K-Nearest Neighbors.

Imagine que un proveedor de telecomunicaciones ha segmentado su base de clientes por patrones de uso de servicios, clasificando a los clientes en cuatro grupos. Si se pueden utilizar datos demográficos para predecir la pertenencia a un grupo, la empresa puede personalizar las ofertas para sus clientes.

Este es un problema de clasificación, es decir, dado el conjunto de datos con etiquetas predefinidas, necesitamos construir un modelo para ser utilizado para predecir la clase de un caso nuevo o desconocido.

El ejemplo se centra en el uso de datos demográficos, como la región, la edad y el estado civil para predecir patrones de uso.

El campo de destino denominado custcat (Custom Category) tiene cuatro valores posibles que corresponden a los cuatro grupos de clientes de la siguiente manera:

- Servicio básico.
- Servicio E.
- Servicio Plus.
- Servicio total.

Nuestro objetivo es construir un clasificador.

	X: Independent variable											Y: Dependent variable	
	region	age	marital	address	income	ed	employ	retire	gender	reside	custcat		
0	2	44	1	9	64	4	5	0	0	2	1		
1	3	33	1	7	136	5	5	0	0	6	4		
2	3	52	1	24	116	1	29	0	1	2	3		
3	2	33	0	12	33	2	0	0	1	1	1		
4	2	30	1	9	30	1	2	0	0	4	3		
5	2	39	0	17	78	2	16	0	1	1	3		
6	3	22	1	2	19	2	4	0	1	5	2		
7	2	35	0	5	76	2	10	0	0	3	4		
8	3	50	1	7	166	4	31	0	0	5	?		

Figura. Ejemplo de problema de clasificación.

Usaremos un tipo específico de clasificación llamado K-Nearest Neighbor.

Sólo por el bien de la demostración, usemos solo dos campos como predictores específicamente: edad e ingresos.

Ahora, digamos que tenemos un nuevo cliente con una edad e ingresos conocidos: ¿Cómo podemos encontrar la clase de este cliente?

¿Podemos encontrar uno de los casos más cercanos y asignar la misma etiqueta de clase a nuestro nuevo cliente?

¿Podemos decir también que la clase de nuestro nuevo cliente como en el ejemplo de la figura siguiente es probablemente el grupo cuatro (el servicio total) porque su vecino más cercano es también de clase cuatro?

Sí, podemos. De hecho, es el primer vecino más cercano.

	region	age	marital	address	income	ed	employ	retire	gender	reside	custcat
0	2	44	1	9	64	4	5	0	0	2	1
1	3	33	1	7	136	5	5	0	0	6	4
2	3	52	1	24	116	1	29	0	1	2	3
3	2	33	0	12	33	2	0	0	1	1	1
4	2	30	1	9	30	1	2	0	0	4	3
5	2	39	0	17	78	2	16	0	1	1	3
6	3	22	1	2	19	2	4	0	1	5	2
7	2	35	0	5	76	2	10	0	0	3	4
8	3	60	1	7	166	4	31	0	0	5	?

1-NN → 4: Total Service

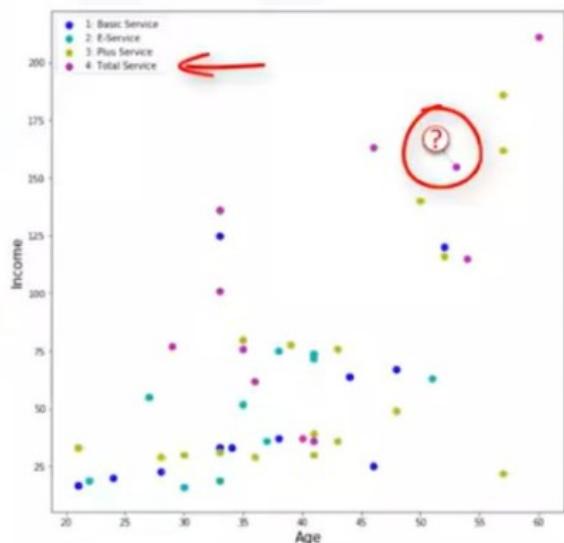


Figura. 1-NN

Ahora, la pregunta es:

¿Hasta qué punto podemos confiar en nuestro juicio que se basa en el primer vecino más cercano?

Podría ser un mal juicio, especialmente si el primer vecino más cercano es un caso muy específico o un valor atípico.

Ahora, en lugar de elegir al primer vecino más cercano, ¿qué tal si elegimos a los cinco vecinos más cercanos y votamos por mayoría entre ellos para definir la clase de nuestro nuevo cliente?

En este caso, veríamos que tres de cada cinco vecinos más cercanos nos dicen que vayamos a la clase tres, que es el Servicio Plus.

¿ Esto no tiene más sentido?

Sí, lo tiene. En este caso, el valor de K en el algoritmo K-Nearest Neighbors es cinco.

	region	age	marital	address	income	ed	employ	retire	gender	reside	custcat
0	2	44	1	9	64	4	5	0	0	2	1
1	3	33	1	7	136	5	5	0	0	6	4
2	3	52	1	24	116	1	29	0	1	2	3
3	2	33	0	12	33	2	0	0	1	1	1
4	2	30	1	9	30	1	2	0	0	4	3
5	2	39	0	17	78	2	16	0	1	1	3
6	3	22	1	2	19	2	4	0	1	5	2
7	2	35	0	5	76	2	10	0	0	3	4
8	3	50	1	7	166	4	31	0	0	5	?

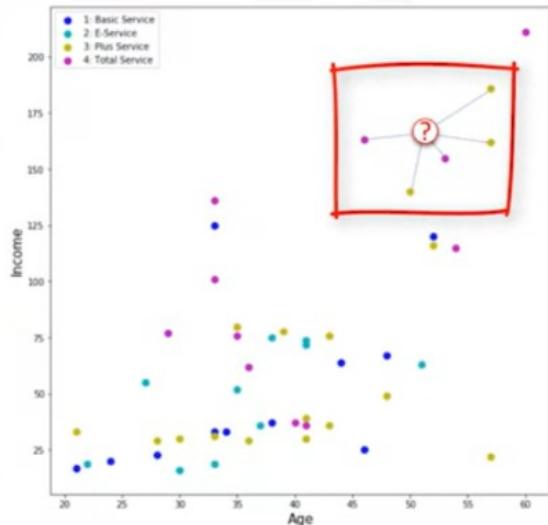


Figura. 5-NN

El ejemplo anterior resalta la intuición detrás del algoritmo K-NN.

Ahora definamos a los K vecinos más cercanos.

El algoritmo K-Nearest Neighbors es un algoritmo de clasificación que toma un montón de puntos etiquetados y los utiliza para aprender a etiquetar otros puntos. Este algoritmo clasifica los casos en función de su similitud con otros casos. En K-Nearest Neighbors, se dice que los puntos de datos que están cerca unos de otros son vecinos. K-Veinos más cercanos se basa en este paradigma: casos similares con las mismas etiquetas de clase están cerca uno del otro.

Así, la distancia entre dos casos es una medida de su disimilitud.

Hay diferentes formas de calcular la similitud o por el contrario, la distancia o disimilitud de dos puntos de datos. Por ejemplo, esto se puede hacer usando la distancia euclídea.

En un problema de clasificación, **el algoritmo K-Nearest Neighbors funciona de la siguiente manera:**

- Se elige un valor para K.

- Se calcula la distancia desde el nuevo caso a cada uno de los casos del conjunto de datos.
- Se buscan las K observaciones en los datos de entrenamiento que están más cerca de las mediciones del punto de datos desconocido.
- Se predice la respuesta del punto de datos desconocido usando el valor de respuesta más popular de los K vecinos más cercanos.

Hay 2 partes en este algoritmo que requieren ahondar un poco más:

- Cómo seleccionar el valor correcto de K.
- Cómo calcular la similitud entre los casos.

Empecemos por el segundo punto, ¿cómo podemos calcular la similitud entre dos puntos de datos?

Podemos usar por ejemplo un tipo específico de distancia de Minkowski, la distancia euclídea.



Customer 1

Age

34



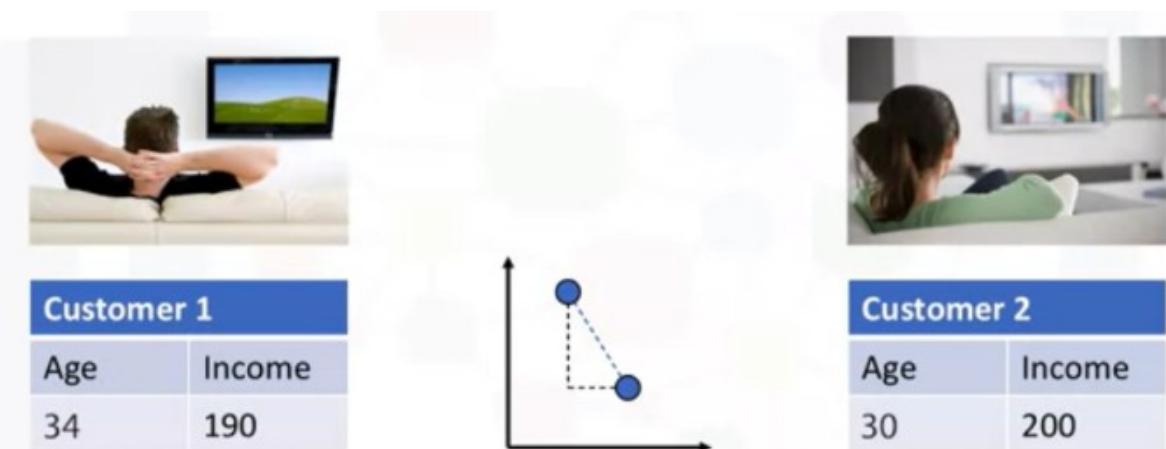
Customer 2

Age

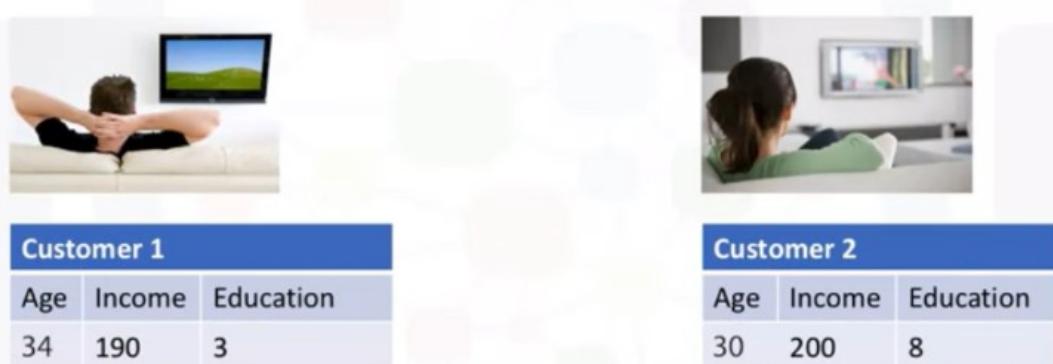
30

$$\text{Dis } (x_1, x_2) = \sqrt{\sum_{i=0}^n (x_{1i} - x_{2i})^2}$$

$$\text{Dis } (x_1, x_2) = \sqrt{(34 - 30)^2} = 4$$



$$\begin{aligned} \text{Dis } (x_1, x_2) &= \sqrt{\sum_{i=0}^n (x_{1i} - x_{2i})^2} \\ &= \sqrt{(34 - 30)^2 + (190 - 200)^2} = 10.77 \end{aligned}$$



$$\begin{aligned} \text{Dis } (x_1, x_2) &= \sqrt{\sum_{i=0}^n (x_{1i} - x_{2i})^2} \\ &= \sqrt{(34 - 30)^2 + (190 - 200)^2 + (3 - 8)^2} = 11.87 \end{aligned}$$

Figura. Distancia euclídea para 1, 2 y 3 características.

K es el número de vecinos más cercanos a tomar en cuenta y debe ser especificado por el usuario.

¿Cómo elegimos la K correcta?

Supongamos que queremos encontrar la clase del cliente anotado como signo de interrogación en el gráfico de la figura siguiente.

¿Qué sucede si elegimos un valor muy bajo de K? Digamos que K = 1.

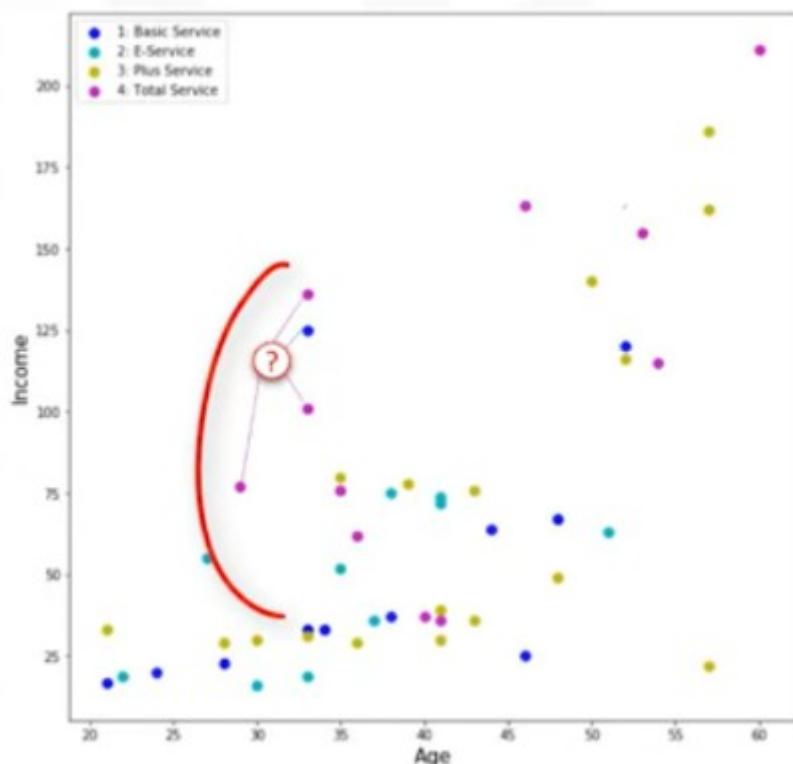


Figura. Ejemplo donde K=1 brinda una mala predicción.

El primer punto más cercano sería el azul, que es la clase uno. Esta sería una mala predicción, ya que más de los puntos a su alrededor son magenta o clase cuatro. De hecho, dado que su vecino más cercano es azul podemos decir que capturamos el ruido en los datos o elegimos uno de los puntos que era una anomalía en los datos.

Un valor bajo de K también causa un modelo altamente complejo, lo que podría resultar en un sobreajuste del modelo, lo que significa que el proceso de predicción no está lo suficientemente generalizado como para ser utilizado en casos fuera de la muestra. Los datos fuera de la muestra son datos que están fuera del conjunto de datos utilizado para entrenar el modelo. En otras palabras, no se puede confiar en que se utilice para la predicción de muestras desconocidas. Es importante recordar que el sobreajuste es malo, ya que queremos un modelo general que funcione para cualquier dato, no solo para los datos utilizados para el entrenamiento.

Ahora, en el lado opuesto del espectro, si elegimos un valor muy alto de K como K es igual a 20, entonces el modelo se generaliza excesivamente.

Entonces, ¿cómo podemos encontrar el mejor valor para K?

- Se reserva una parte de los datos para probar la precisión del modelo.
- Se elige $K = 1$.
- Se utiliza la parte de entrenamiento para modelar y luego se calcula la precisión con el conjunto de pruebas.
- Se aumenta K y se repite el proceso.
- Se toma el K que brinda la mayor precisión.

En nuestro caso, como puede verse en la figura siguiente, $K=4$ brinda la mayor precisión.

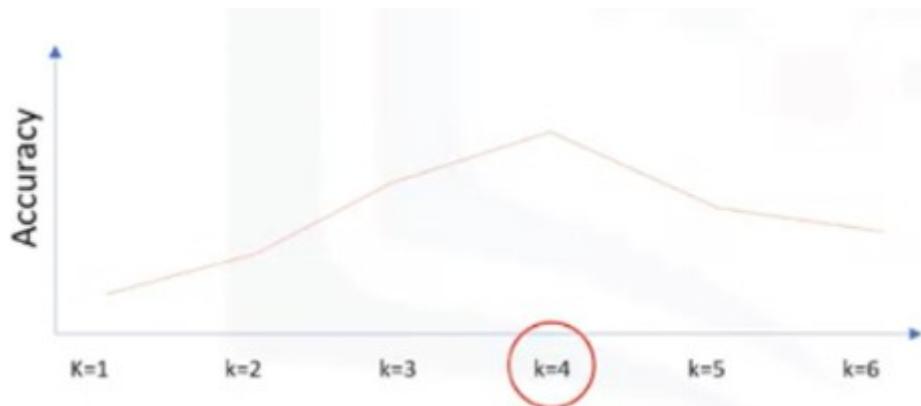


Figura. $K = 4$ brinda la mayor precisión.

El análisis de vecinos más cercanos también se puede utilizar para calcular valores de un objetivo continuo. En esta situación, se utiliza el valor objetivo medio o mediana de los vecinos más cercanos para obtener el valor previsto para el nuevo caso.

Por ejemplo, suponga que está prediciendo el precio de una vivienda en función de su conjunto de características, como el número de habitaciones , metros cuadrados, el año en que se creó, etc. Usted puede encontrar fácilmente las tres casas vecinas más cercanas, por supuesto, no sólo en base a la distancia, sino también en base a todos los atributos y luego predecir el precio de la casa como el promedio de los vecinos.

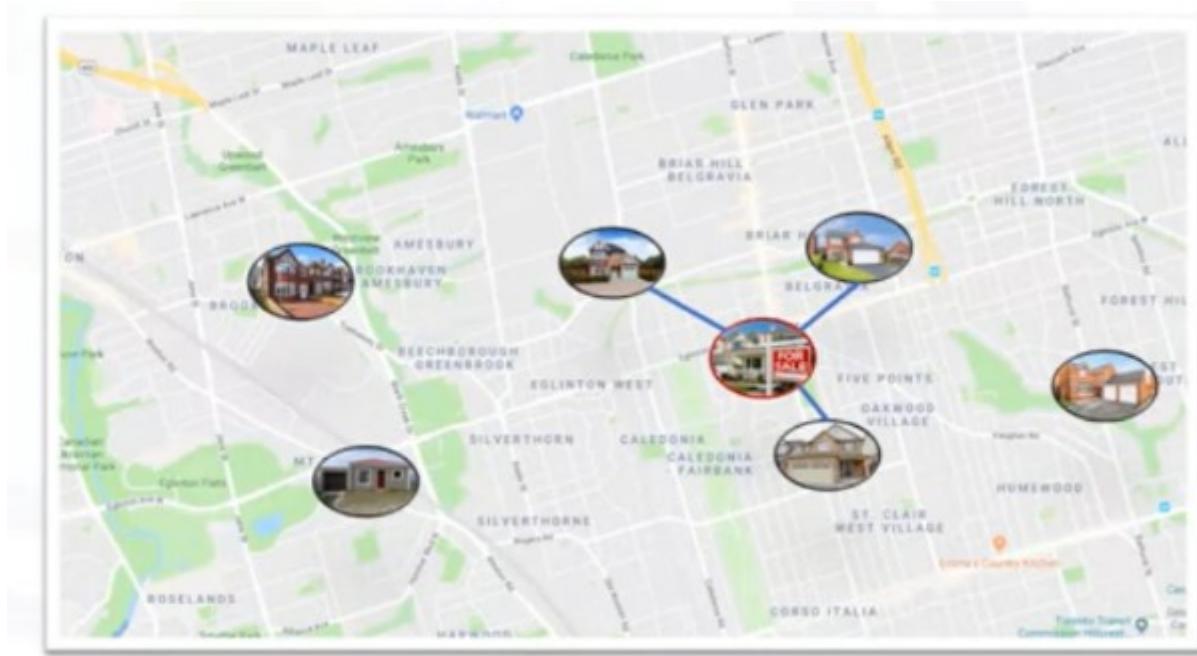


Figura. Uso de K-NN en regresión.

3.3. MÉTRICAS DE EVALUACIÓN EN CLASIFICACIÓN

Vamos a cubrir las métricas de evaluación de los clasificadores. Las métricas de evaluación explican el rendimiento de un modelo.

Imagine que tenemos un conjunto de datos histórico que muestra el churn del cliente para una empresa de telecomunicaciones. Hemos entrenado el modelo, y ahora queremos calcular su precisión usando el conjunto de pruebas. Pasamos el conjunto de pruebas a nuestro modelo, y encontramos las etiquetas predecidas. Ahora la pregunta es:

«¿Qué tan preciso es este modelo?»

Básicamente, comparamos los valores reales en el conjunto de pruebas con los valores predecidos por el modelo, para calcular la precisión del modelo.

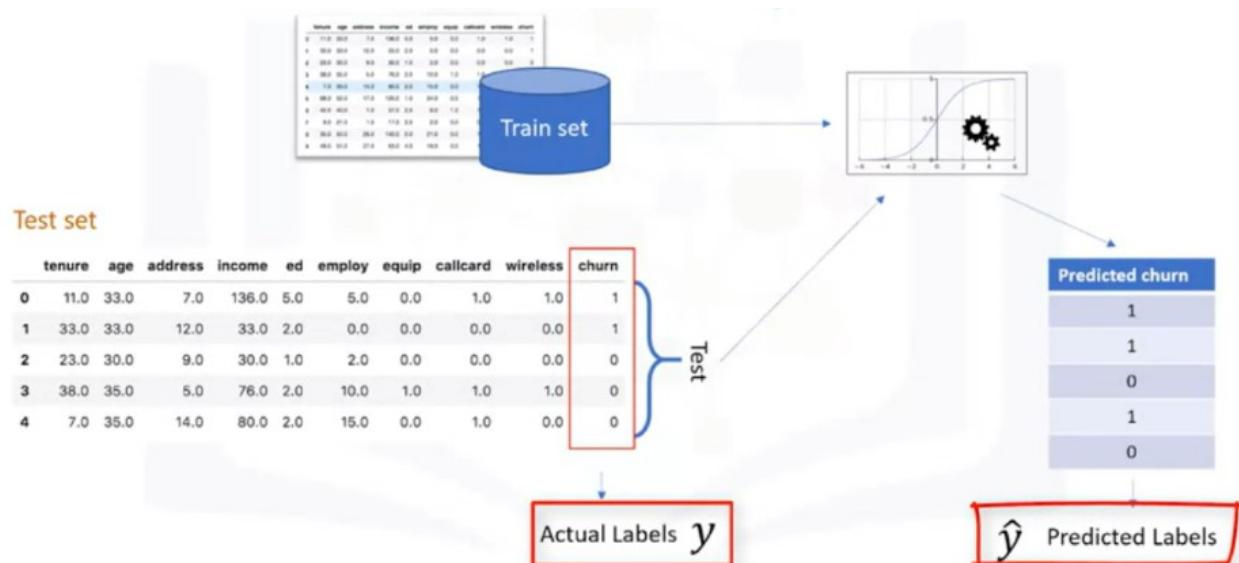


Figura. Ejemplo de clasificación.

Las métricas de evaluación desempeñan un papel clave en el desarrollo de un modelo, ya que proporcionan información sobre las áreas que podrían requerir mejoras.

Hay diferentes **métricas de evaluación de modelos**, aquí hablaremos de 3 de ellas:

- Índice de Jaccard.
- F1-score
- Log Loss.

Comencemos con el **índice Jaccard**, también conocido como coeficiente de similitud de Jaccard.

Digamos que y muestra las etiquetas verdaderas del dataset de churn e \hat{y} muestra los valores predecidos por nuestro clasificador. Entonces podemos definir Jaccard como el tamaño de la intersección dividido por el tamaño de la unión de dos conjuntos de etiquetas.

Por ejemplo, para un conjunto de pruebas de tamaño 10, con 8 predicciones correctas u 8 intersecciones, la precisión por el índice Jaccard sería 0.66.

Si todo el conjunto de etiquetas previstas para una muestra coincide estrictamente con el conjunto verdadero de etiquetas, entonces la precisión del subconjunto es 1.0; en el caso opuesto, donde no hay ninguna coincidencia, es 0.0.

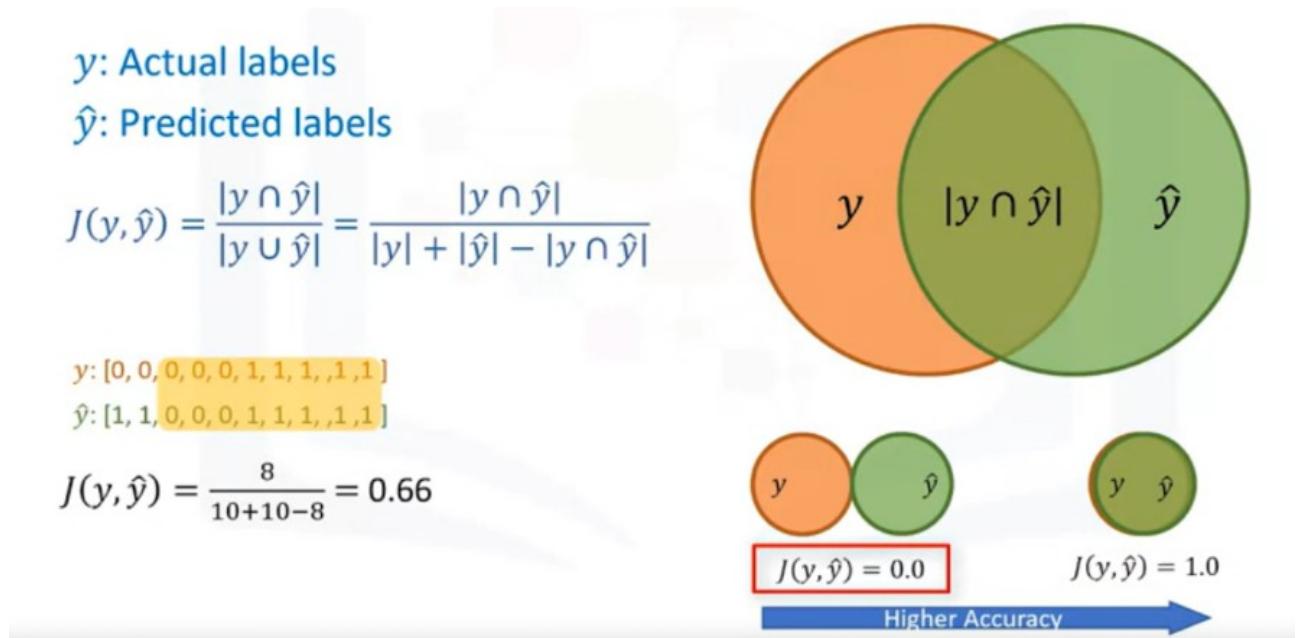


Figura. Índice Jaccard.

Otra forma de ver la precisión de los clasificadores es mirar una **matriz de confusión**.

Por ejemplo, supongamos que nuestro conjunto de pruebas tiene solo 40 filas y consideremos la matriz de la figura siguiente.

Esta matriz muestra las predicciones correctas e incorrectas, en comparación con las etiquetas reales.

Cada fila de matriz de confusión muestra las etiquetas Actual/True del conjunto de prueba, y las columnas muestran las etiquetas predecidas por clasificador.

La primera fila es para clientes cuyo valor real de churn en el conjunto de prueba es 1. Como puede calcular, de 40 clientes, el valor de churn de 15 de ellos es 1. Y de estos 15, el clasificador predijo correctamente 6 de ellos como 1, y 9 de ellos como 0. Esto significa que para 6 clientes, el valor de churn real era 1, en el conjunto de prueba, y el clasificador también predijo correctamente los como 1. Sin embargo, mientras que la etiqueta real de 9 clientes era 1, el clasificador predijo que eran 0, lo cual no es muy bueno. Podemos considerar esto como un error del modelo para la primera fila.

¿Qué pasa con los clientes con un valor de churn 0?

Veamos la segunda fila.

Parece que había 25 clientes cuyo valor de churn era 0. El clasificador predijo correctamente 24 de ellos como 0, y uno de ellos predijo erróneamente como 1. Por lo tanto, ha hecho un buen trabajo al predecir los clientes con un valor de churn de 0.

Algo bueno sobre la matriz de confusión es que muestra la capacidad del modelo para predecir o separar correctamente las clases.

En el caso específico de un clasificador binario, como este ejemplo, podemos interpretar estos números como el recuento de:

- TP: verdaderos positivos.
- FN: falsos negativos.
- TN: verdaderos negativos.
- FP: falsos positivos.

Basándonos en el recuento de cada sección, podemos calcular la precisión y la recuperación (recall) de cada etiqueta.

La precisión es una medida de la precisión, siempre que se haya predicho una etiqueta de clase, se define por:

$$\text{precisión} = \text{Verdadero Positivo}/(\text{Verdadero Positivo} + \text{Falso Positivo}).$$

Y Recall es la verdadera tasa positiva, se define como:

$$\text{Recall} = \text{Verdadero Positivo}/(\text{Verdadero Positivo} + \text{Falso Negativo}).$$

Ahora estamos en condiciones de calcular las puntuaciones de F1 para cada etiqueta, en función de la precisión y la recuperación de esa etiqueta.

La **puntuación F1** es el promedio armónico de la precisión y el recall, donde una puntuación F1 alcanza su mejor valor en 1 (que representa precisión perfecta y recuerdo) y su peor en 0. Es una buena manera de mostrar que un clasificador tiene un buen valor tanto para la recuperación como para la precisión.

En el ejemplo, la puntuación F1 para la clase 0 (es decir, churn=0), es 0.83, y la puntuación F1 para la clase 1 (es decir, churn=1), es 0.55.

La precisión promedio para este clasificador es el promedio de la puntuación F1 para ambas etiquetas, que es 0.72 en nuestro caso.

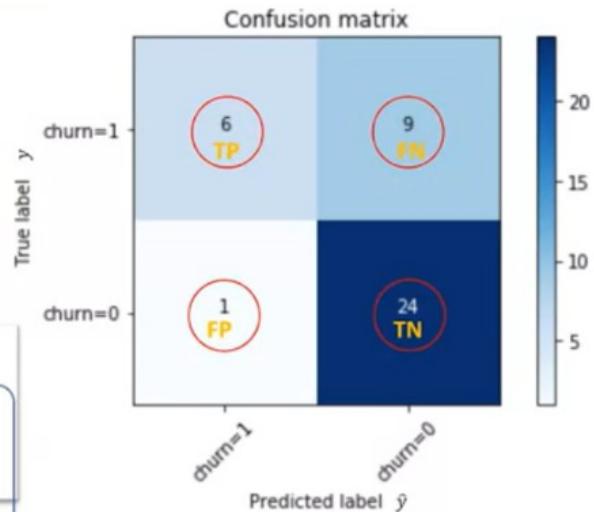
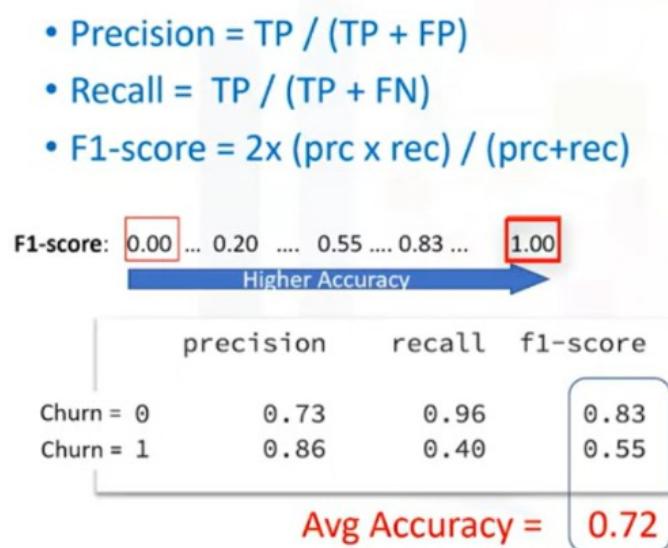


Figura. F1-score.

Tanto Jaccard como F1-score también se pueden utilizar para clasificadores multiclas.

Ahora veamos otra métrica de precisión para los clasificadores.

A veces, la salida de un clasificador es la probabilidad de una etiqueta de clase, en lugar de la etiqueta. Por ejemplo, en regresión logística, la salida puede ser la probabilidad de pérdida del cliente, es decir, sí (o igual a 1). Esta probabilidad es un valor entre 0 y 1.

La pérdida logarítmica (también conocida como **Log loss**) mide el rendimiento de un clasificador donde la salida prevista es un valor de probabilidad entre 0 y 1.

Entonces, por ejemplo, predecir una probabilidad de 0,13 cuando la etiqueta real es 1, sería malo y daría como resultado una alta pérdida de registro.

Podemos calcular la pérdida de log para cada fila usando la ecuación de log loss, que mide cuán lejos está cada predicción, de la etiqueta real. A continuación, calculamos la pérdida promedio de registro en todas las filas del conjunto de pruebas. Es obvio que los clasificadores ideales tienen valores progresivamente más pequeños de pérdida de registro. Por lo tanto, el clasificador con menor pérdida de registro tiene una mayor precisión.

value between 0 and 1.

Test set

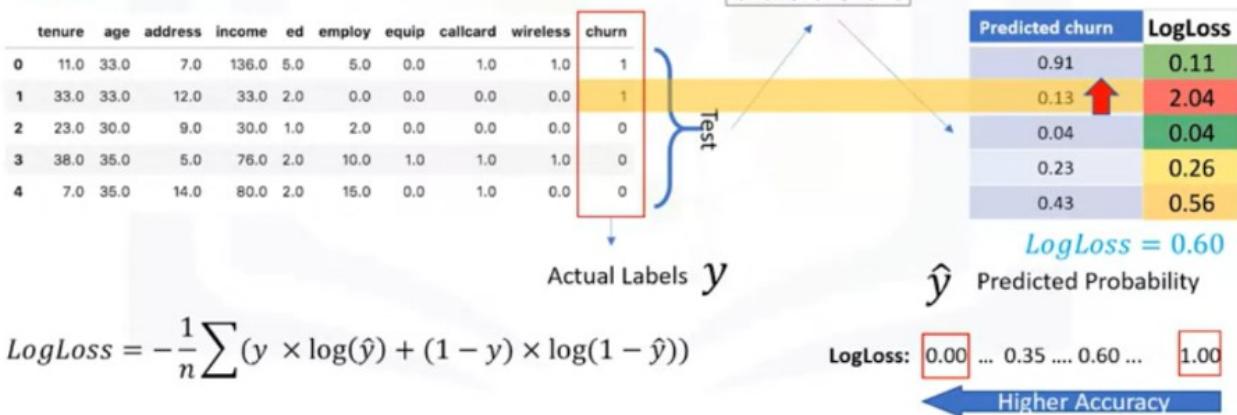


Figura. Log loss.

3.4. INTRODUCCIÓN A LOS ÁRBOLES DE DECISIÓN

Veremos los árboles de decisión y responderemos las siguientes preguntas:

¿Qué es un árbol de decisión?

¿Cómo los usamos para ayudarnos a clasificar?

¿Cómo puedo crear mi propio árbol de decisiones?

Imagina que eres un investigador médico recopilando datos para un estudio. Ya ha recopilado datos sobre un conjunto de pacientes que sufrían de la misma enfermedad. Durante su tratamiento, cada paciente respondió a uno de los dos medicamentos, a los cuales llamamos drug A y drug B.

Parte de su trabajo es construir un modelo para averiguar qué medicamento podría ser apropiado para un futuro paciente con la misma enfermedad. Los conjuntos de características de este conjunto de datos son la edad, el género, la presión arterial y el colesterol de nuestro grupo de pacientes y el objetivo es el medicamento al que respondió cada paciente. Es una muestra de clasificadores binarios, y puede usar la parte de entrenamiento del conjunto de datos para construir un árbol de decisiones y luego usarlo para predecir la clase de un paciente desconocido, en esencia, para llegar a una decisión sobre qué medicamento prescribir a un nuevo paciente.

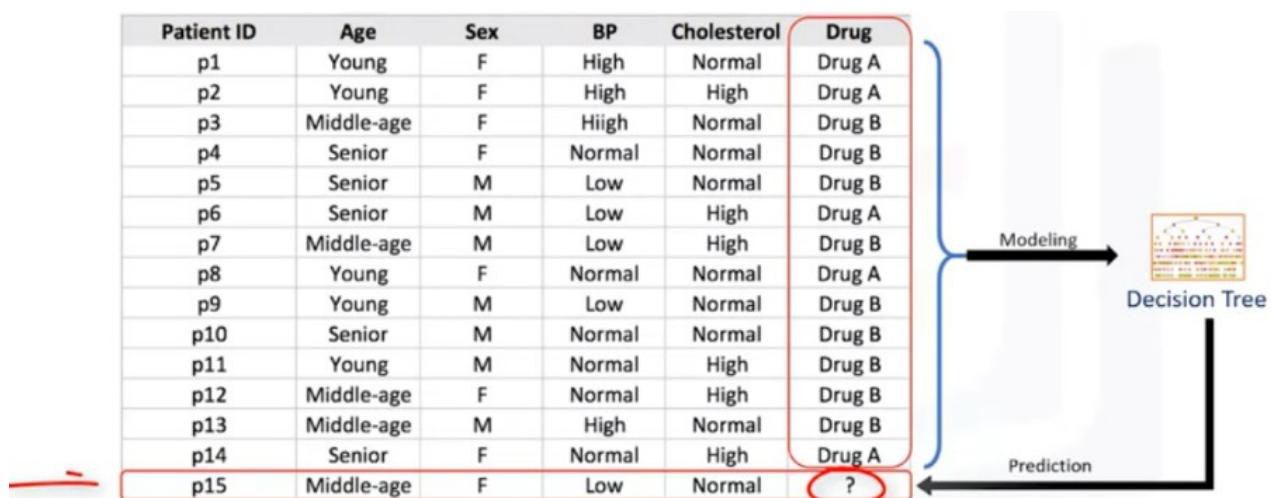


Figura. Árboles de decisión.

Veamos cómo se construye un árbol de decisiones para este conjunto de datos.

Los árboles de decisión se crean dividiendo el conjunto de entrenamiento en nodos distintos, donde un nodo contiene toda o la mayoría de una categoría de los datos.

Si miramos el diagrama de la figura siguiente, podemos ver que es el clasificador de un paciente.

Queremos recetar un medicamento a un nuevo paciente, pero la decisión de elegir el medicamento A o B estará influenciada por la situación del paciente.

Comenzamos con la edad, que puede ser joven, de mediana edad o mayor. Si el paciente es de mediana edad, entonces definitivamente vamos a tomar el medicamento B. Por otro lado, si tiene un paciente joven o de edad avanzada, necesitará más detalles para ayudarnos a determinar qué medicamento recetar. Las variables de decisión adicionales pueden ser cosas como los niveles de colesterol, sexo o presión arterial. Por ejemplo, si el paciente es femenino, entonces recomendaremos el medicamento A, pero si el paciente es masculino, entonces irá por el medicamento B.

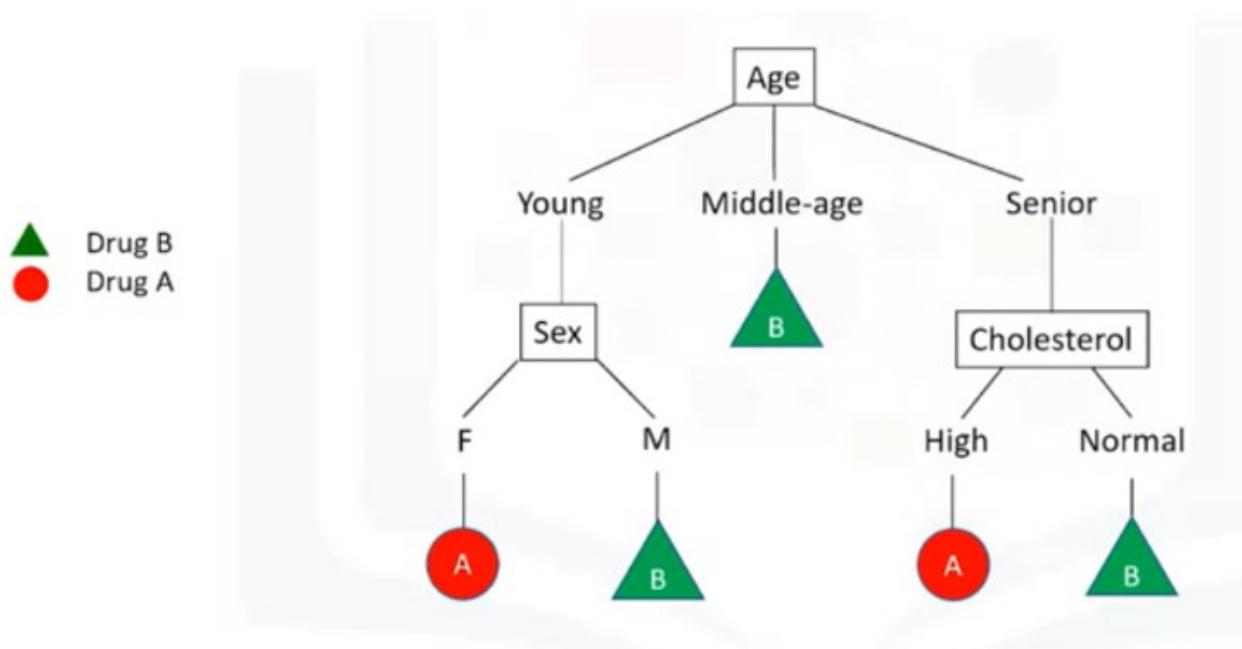


Figura. Árbol de decisión.

Los árboles de decisión son acerca de probar un atributo y ramificar los casos basados en el resultado de la prueba.

- Cada nodo interno corresponde a una prueba.
- Cada rama corresponde a un resultado de la prueba.

- Cada nodo hoja asigna un paciente a una clase.

¿Cómo podemos construir un árbol de decisiones?

Un árbol de decisión se puede construir considerando los atributos uno por uno.

1. Elija un atributo del conjunto de datos.
2. Calcule la importancia del atributo en la división de los datos.
3. Divida los datos en función del valor del mejor atributo.
4. Vaya a cada rama y repita para el resto de los atributos

Después de construir este árbol, puede usarlo para predecir la clase de casos desconocidos; en nuestro caso, el medicamento adecuado para un nuevo paciente basado en sus características.

3.5. CONSTRUYENDO ÁRBOLES DE DECISIÓN

Cubriremos el proceso de construcción de árboles de decisión

Consideré el conjunto de datos de drogas de nuevo.

La pregunta es:

¿Cómo construimos un árbol de decisiones basado en ese conjunto de datos?

Patient ID	Age	Sex	BP	Cholesterol	Drug
p1	Young	F	High	Normal	Drug A
p2	Young	F	High	High	Drug A
p3	Middle-age	F	Hiigh	Normal	Drug B
p4	Senior	F	Normal	Normal	Drug B
p5	Senior	M	Low	Normal	Drug B
p6	Senior	M	Low	High	Drug A
p7	Middle-age	M	Low	High	Drug B
p8	Young	F	Normal	Normal	Drug A
p9	Young	M	Low	Normal	Drug B
p10	Senior	M	Normal	Normal	Drug B
p11	Young	M	Normal	High	Drug B
p12	Middle-age	F	Normal	High	Drug B
p13	Middle-age	M	High	Normal	Drug B
p14	Senior	F	Normal	High	Drug A
p15	Middle-age	F	Low	Normal	?

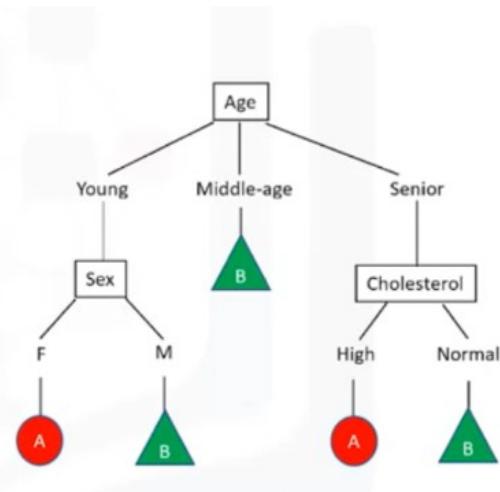


Figura. Dataset de drogas y árbol de decisión.

Los árboles de decisión se crean utilizando particiones recursivas para clasificar los datos.

Digamos que tenemos 14 pacientes en nuestro conjunto de datos, el algoritmo elige la característica más predictiva para dividir los datos. Así, lo importante es determinar qué atributo es el mejor o el más predictivo para dividir los datos.

Digamos que elegimos el colesterol como el primer atributo para dividir los datos, dividirá nuestros datos en dos ramas. Como puede ver, si el paciente tiene colesterol alto no podemos decir con alta confianza que el medicamento B podría ser adecuado para él. Además, si el colesterol del paciente es normal, todavía no tenemos suficiente evidencia o información para determinar si el medicamento A o el medicamento B es de hecho adecuado. Es una muestra de mala selección de atributos para dividir datos.

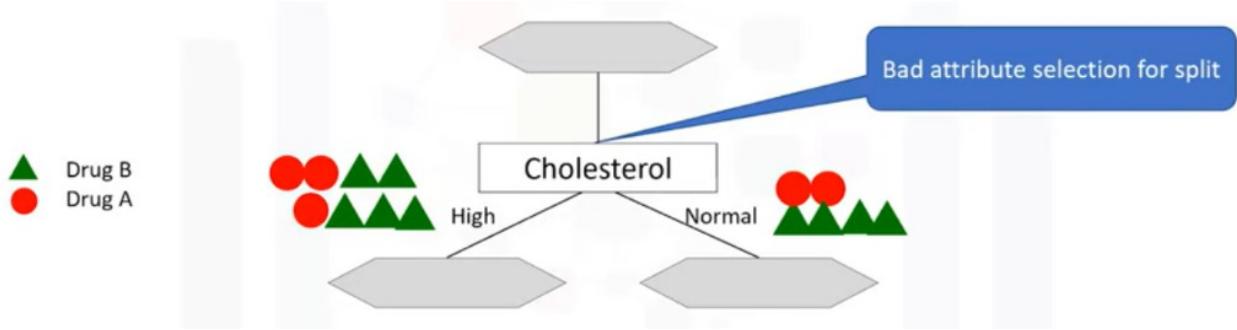


Figura. El colesterol no es un buen atributo para la división inicial.

Entonces, intentemos otro atributo. Una vez más, tenemos nuestros 14 casos, esta vez elegimos el atributo sexual de los pacientes. Dividirá nuestros datos en dos ramas, masculina y femenina. Como puede ver, si la paciente es femenina, podemos decir que la droga B podría ser adecuada para ella con alta certeza. Pero si el paciente es masculino, no tenemos suficiente evidencia o información para determinar si el medicamento A o el medicamento B son adecuados. Sin embargo, sigue siendo una mejor opción en comparación con el atributo colesterol porque el resultado en los nodos son más puros (son principalmente de un solo tipo). Por lo tanto, podemos decir que el atributo sexual es más significativo que el colesterol, o en otras palabras, es más predictivo que los otros atributos.

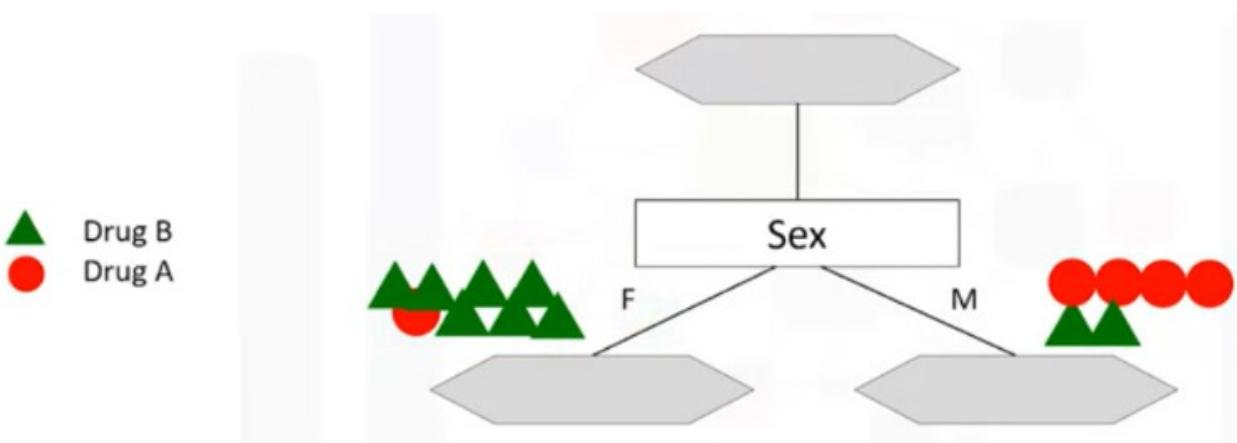


Figura. El sexo es un mejor atributo para la división.

La predictividad se basa en la disminución de la impureza de los nodos.

Estamos buscando la mejor característica para disminuir la impureza de los pacientes en las hojas, después de dividirlos en función de esa característica. Por lo tanto, la característica sexual es un buen candidato en el siguiente caso porque casi encontró a los pacientes puros.

Vamos un paso más allá. Para la rama del paciente masculino, nuevamente probamos otros atributos para dividir el sub-árbol. Volvemos a probar el colesterol aquí, ya que puede ver que resulta en hojas aún más puras. Así que podemos tomar fácilmente una decisión aquí. Por ejemplo, si un paciente es masculino y su colesterol es alto, sin duda podemos recetar el medicamento A, pero si es normal, podemos recetar el medicamento B con alta confianza.

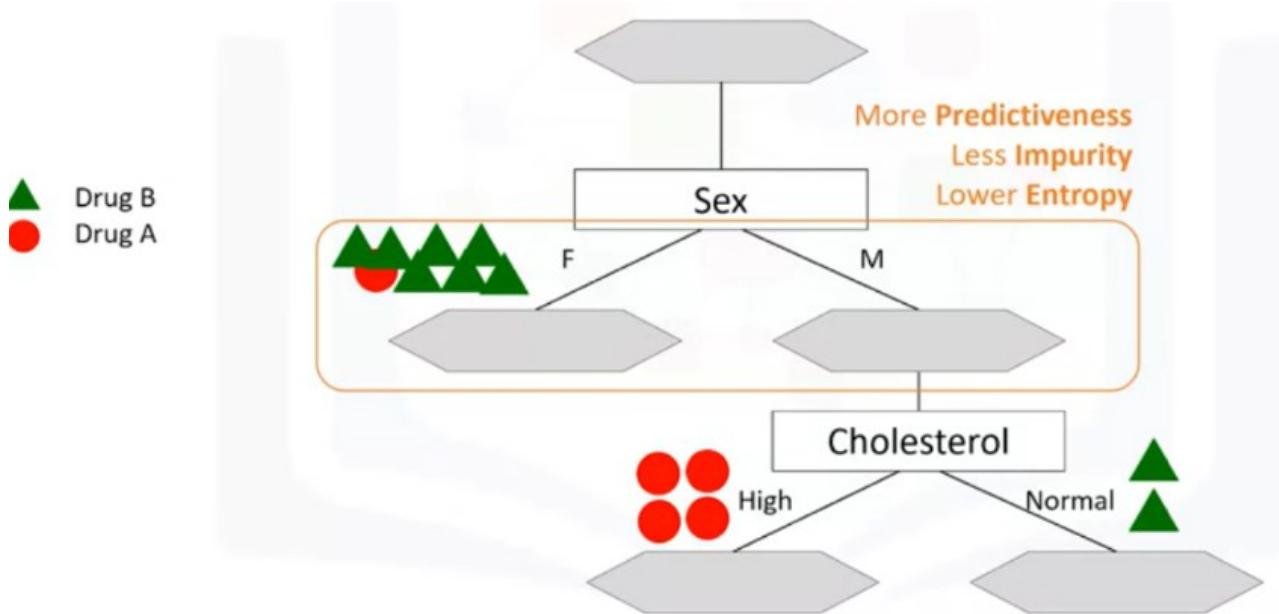


Figura. Profundizamos el árbol.

Como se puede notar, la elección del atributo para dividir los datos es muy importante y se relaciona con la pureza de las hojas después de la división.

Un nodo en el árbol se considera puro si en el 100 por ciento de los casos, los nodos caen en una categoría específica del campo de destino.

De hecho, el método utiliza particiones recursivas para dividir los registros de entrenamiento en segmentos minimizando la impureza en cada paso. La impureza de los nodos se calcula mediante la entropía de los datos en el nodo.

¿Qué es la entropía?

La **entropía** es la cantidad de trastorno de la información o la cantidad de aleatoriedad en los datos.

La entropía en el nodo depende de cuántos datos aleatorios hay en ese nodo y se calcula para cada nodo.

En los árboles de decisión, estamos buscando árboles que tengan la entropía más pequeña en sus nodos.

La entropía se utiliza para calcular la homogeneidad de las muestras en ese nodo. Si las muestras son completamente homogéneas, la entropía es cero y si las muestras se dividen por igual tiene una entropía de uno.

Esto significa que si todos los datos en un nodo son drogas A o drogas B, entonces la entropía es 0, pero si la mitad de los datos son drogas A y la otra mitad son B entonces la entropía es 1.

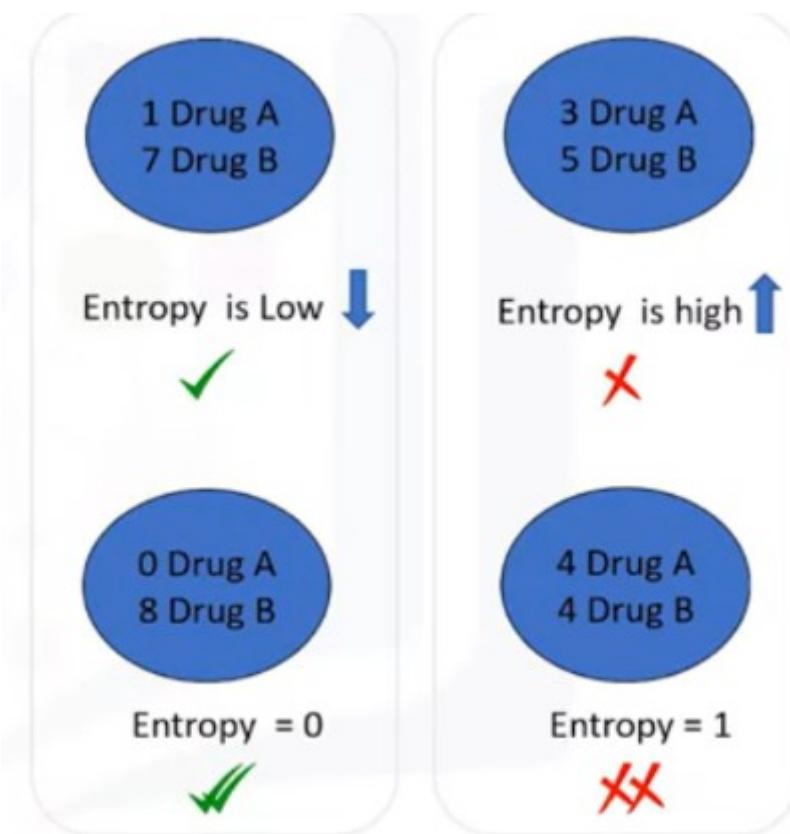


Figura. Entropía.

Puede calcular fácilmente la entropía de un nodo utilizando la tabla de frecuencias del atributo a través de la fórmula de entropía donde P es la proporción de una categoría, como el medicamento A o B.

$$\text{entropía} = p(A) \log(p(A)) - p(B) \log(p(B))$$

Vamos a calcular la entropía del conjunto de datos antes de dividirlo. Tenemos nueve ocurrencias de la droga B y cinco de la droga A. Usando la fórmula vemos que vale 0.94. ¿Cuál es la entropía después de la división?

Ahora, podemos probar diferentes atributos para encontrar el que tiene más predictividad. Primero seleccionemos el colesterol del paciente y veamos cómo se dividen los datos en función de sus valores. Por ejemplo, cuando es normal tenemos seis para la droga B, y dos para la droga A. Podemos calcular la entropía de este nodo en función de la distribución de la droga A y B que es 0,8 en este caso. Pero, cuando el colesterol es alto, los datos se dividen en tres para el medicamento B y tres para el medicamento A. Calculando su entropía, podemos ver que sería 1.0.

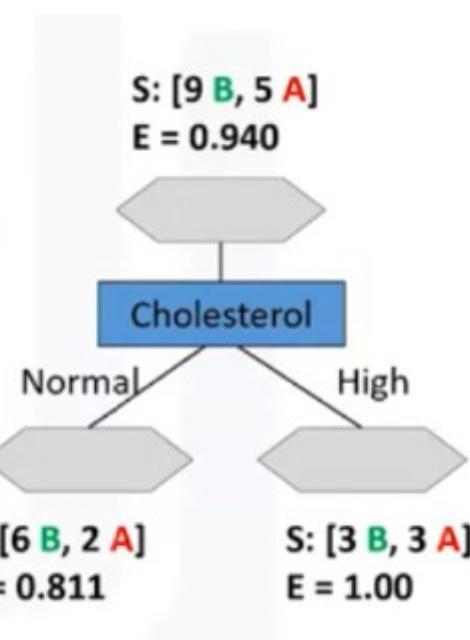


Figura. Entropía eligiendo el colesterol como primer atributo.

Deberíamos revisar todos los atributos y calcular la entropía después de la división y luego elegir el mejor atributo.

Intentemos con otro campo, esta vez el sexo.

Como puede ver, cuando usamos el atributo de sexo para dividir los datos, cuando su valor es femenino, tenemos tres pacientes que respondieron al medicamento B y cuatro pacientes que respondieron al medicamento A. La entropía para este nodo es de 0,98, lo que no es muy prometedor. Sin embargo, en el otro lado de la rama, cuando el valor del atributo sexual es masculino, el resultado es más puro con sexo para la droga B y solo uno para la droga A. La entropía para este grupo es 0.59.

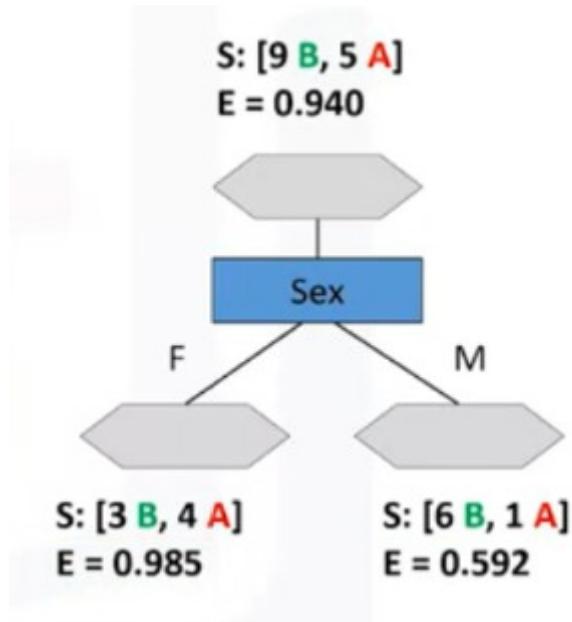


Figura. Entropía eligiendo el sexo como primer atributo.

Entre los atributos colesterol y sexo:

¿Cuál es una mejor opción? ¿ Cuál es mejor en el primer atributo para dividir el conjunto de datos en dos ramas? O en otras palabras, ¿qué atributo da como resultado nodos más puros para nuestros medicamentos? ¿ O en qué árbol tenemos menos entropía después de la división en lugar de antes de la división?

El atributo sexo tiene entropía de 0,98 y 0,59 y el atributo colesterol tiene entropía de 0,81 y 1,0 en sus ramas.

La respuesta es el árbol con la mayor ganancia de información después de la división.

La **ganancia de información** es la información que puede aumentar el nivel de certeza después de la división. Es la entropía de un árbol antes de la división menos la entropía ponderada después de la división por un atributo.

Podemos pensar en la ganancia de información y la entropía como opuestos. A medida que la entropía o la cantidad de aleatoriedad disminuye, la ganancia de información o la cantidad de certeza aumenta y viceversa.

Por lo tanto, construir un árbol de decisión se trata de encontrar atributos que devuelvan la mayor ganancia de información.

La ganancia de información se calcula como:

$$\text{ganancia de información} = \text{entropía antes de la división} - \text{entropía ponderada luego de la división}$$

Para el atributo sexo es 0.151, para el colesterol, 0.048. Así, el atributo sexo es más adecuado y debemos elegirlo como primer divisor.

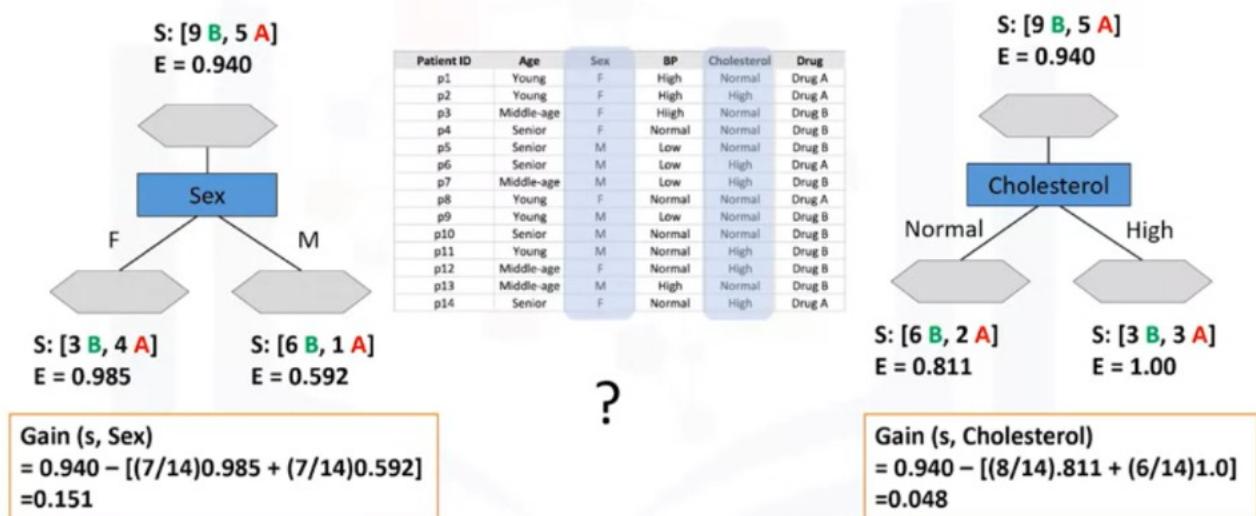


Figura. El atributo sexo tiene mayor ganancia de información.

¿Cuál es el siguiente atributo después de ramificar por el atributo sexo?

Se debe repetir el proceso para cada rama y probar cada uno de los otros atributos para seguir alcanzando las hojas más puras.

3.6. INTRODUCCIÓN A LA REGRESIÓN LOGÍSTICA

Veremos la regresión logística y responderemos 3 preguntas:

¿Qué es la regresión logística?

¿Qué tipo de problemas se pueden resolver por regresión logística?

¿En qué situaciones utilizamos la regresión logística?

La **regresión logística** es una técnica estadística y de aprendizaje automático para clasificar registros de un conjunto de datos basado en los valores de los campos de entrada.

Digamos que tenemos un conjunto de datos de telecomunicaciones que nos gustaría analizar para entender qué clientes podrían dejarnos el próximo mes. Se trata de datos históricos del cliente en los que cada fila representa a un cliente. Imagina que eres analista en esta compañía y tienes que averiguar quién se va y por qué?

Utilizará el conjunto de datos para crear un modelo basado en registros históricos y lo usará para predecir el cambio futuro dentro del grupo de clientes.

El conjunto de datos incluye información sobre los servicios a los que se ha inscrito cada cliente, información de cuenta de cliente, información demográfica sobre clientes como género y rango de edad, y también clientes que han abandonado la empresa durante el último mes (la columna se llama churn).

Podemos usar la regresión logística para construir un modelo para predecir el churn del cliente usando las características dadas.

En regresión logística, utilizamos una o más variables independientes como tenencia, edad e ingreso para predecir un resultado, como el churn, que llamamos la variable dependiente que representa si los clientes dejarán de usar el servicio.

	Independent variables									Dependent variable
	tenure	age	address	income	ed	employ	equip	callcard	wireless	churn
0	11.0	33.0	7.0	136.0	5.0	5.0	0.0	1.0	1.0	Yes
1	33.0	33.0	12.0	33.0	2.0	0.0	0.0	0.0	0.0	Yes
2	23.0	30.0	9.0	30.0	1.0	2.0	0.0	0.0	0.0	No
3	38.0	35.0	5.0	76.0	2.0	10.0	1.0	1.0	1.0	No
4	7.0	35.0	14.0	80.0	2.0	15.0	0.0	1.0	0.0	?

Continuous/Categorical variables

Categorical Vari

Figura. Regresión Logística para predecir el churn.

La regresión logística es análoga a la regresión lineal, pero intenta predecir un campo objetivo categórico o discreto en lugar de uno numérico.

En regresión lineal, podríamos tratar de predecir un valor continuo de variables como el precio de una casa, la presión arterial de un paciente o el consumo de combustible de un automóvil.

En regresión logística, predecimos una variable que es binaria como sí/no, verdadero/falso, exitoso o no exitoso, embarazada/no embarazada, y así sucesivamente, todo lo cual puede ser codificado como cero o uno.

En regresión logística, las variables independientes deben ser continuas. Si son categóricos, deben ser ficticios o indicadores codificados. Esto significa que tenemos que transformarlos a algún valor continuo.

Tenga en cuenta que la regresión logística se puede utilizar tanto para la clasificación binaria como para la clasificación multiclasa, aquí nos centraremos en la clasificación binaria.

Como se mencionó, la regresión logística es un tipo de algoritmo de clasificación, por lo que se puede utilizar en diferentes situaciones, por ejemplo:

- Predecir la probabilidad de que una persona tenga un ataque cardíaco dentro de un período de tiempo especificado, según nuestro conocimiento de la edad, el sexo y el índice de masa corporal de la persona.
- Predecir la probabilidad de mortalidad en un paciente lesionado.

- Predecir si un paciente tiene una enfermedad determinada, como la diabetes, según las características observadas de ese paciente, tales como peso , altura, presión arterial, y los resultados de varios análisis de sangre, etc.
- En un contexto de marketing, podemos utilizarlo para predecir la probabilidad de que un cliente compre un producto o detenga una suscripción, como hemos hecho en nuestro ejemplo de churn.
- Predecir la probabilidad de fallo de un proceso, sistema o producto dado.
- Predecir la probabilidad de que un dueño de casa no pague una hipoteca.

Observe que en todos estos ejemplos no solo predecimos la clase de cada caso, también medimos la probabilidad de que un caso pertenezca a una clase específica.

¿Cuándo deberíamos utilizar la regresión logística?

- Cuando el campo de destino en sus datos es categórico.
 - Como cero/uno, sí/no, churn o no churn, positivo/negativo y así sucesivamente.
- Necesita la probabilidad de su predicción.
 - Por ejemplo, si desea saber cuál es la probabilidad de que un cliente compre un producto. La regresión logística devuelve una puntuación de probabilidad entre cero y uno para una muestra dada de datos. De hecho, la regresión logística predice la probabilidad de esa muestra y asignamos los casos a una clase discreta basada en esa probabilidad.
- Si sus datos son linealmente separables.
 - Aquí el límite de decisión de la regresión logística es una línea o un plano o un hiperplano. Un clasificador clasificará todos los puntos de un lado del límite de decisión como pertenecientes a una clase y todos los del otro lado como pertenecientes a la otra clase.
- Debe comprender el impacto de una característica.
 - Puede seleccionar las mejores entidades en función de la significación estadística de los coeficientes o parámetros del modelo de regresión logística.
 - De hecho, nos permite comprender el impacto que una variable independiente tiene sobre la variable dependiente mientras se controlan otras variables independientes.

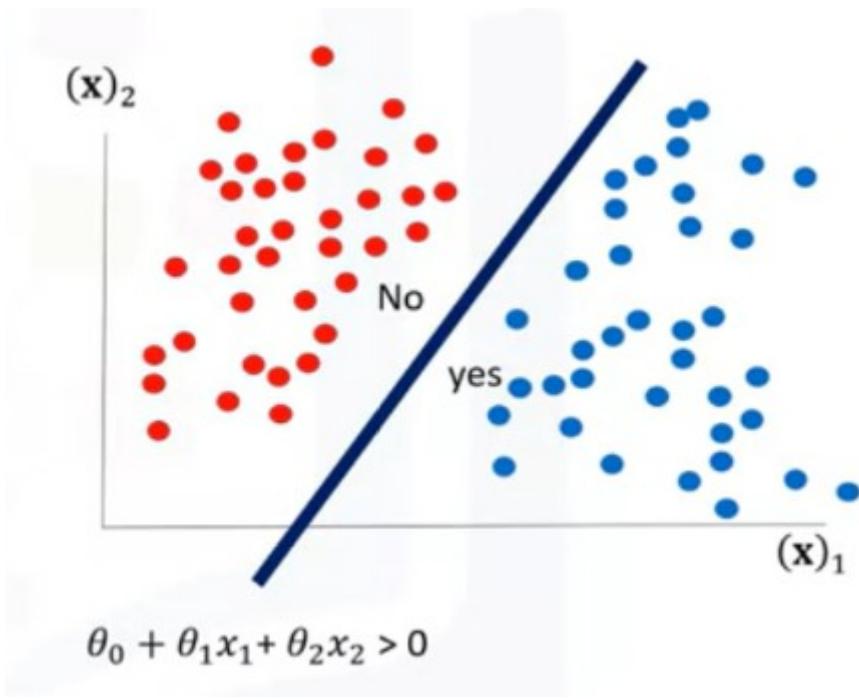


Figura. Ejemplo de aplicación de la regresión logística.

Veamos nuestro conjunto de datos de nuevo.

Definimos las variables independientes como X y la variable dependiente como Y.

En aras de la simplicidad podemos codificar los valores objetivo o dependientes a cero o uno.

El objetivo de la regresión logística es construir un modelo para predecir la clase de cada muestra, que en este caso es un cliente, así como la probabilidad de que cada muestra pertenezca a una clase.

- X es nuestro conjunto de datos en el espacio de números reales de m por n. Es decir, de m dimensiones o entidades y n registros.
- Y es la clase que queremos predecir, que puede ser cero o uno.

Idealmente, un modelo de regresión logística, \hat{Y} , puede predecir que la clase del cliente es una, dadas sus características X.

También se puede mostrar que la probabilidad de que un cliente esté en la clase cero puede calcularse como uno menos la probabilidad de que la clase del cliente es uno.

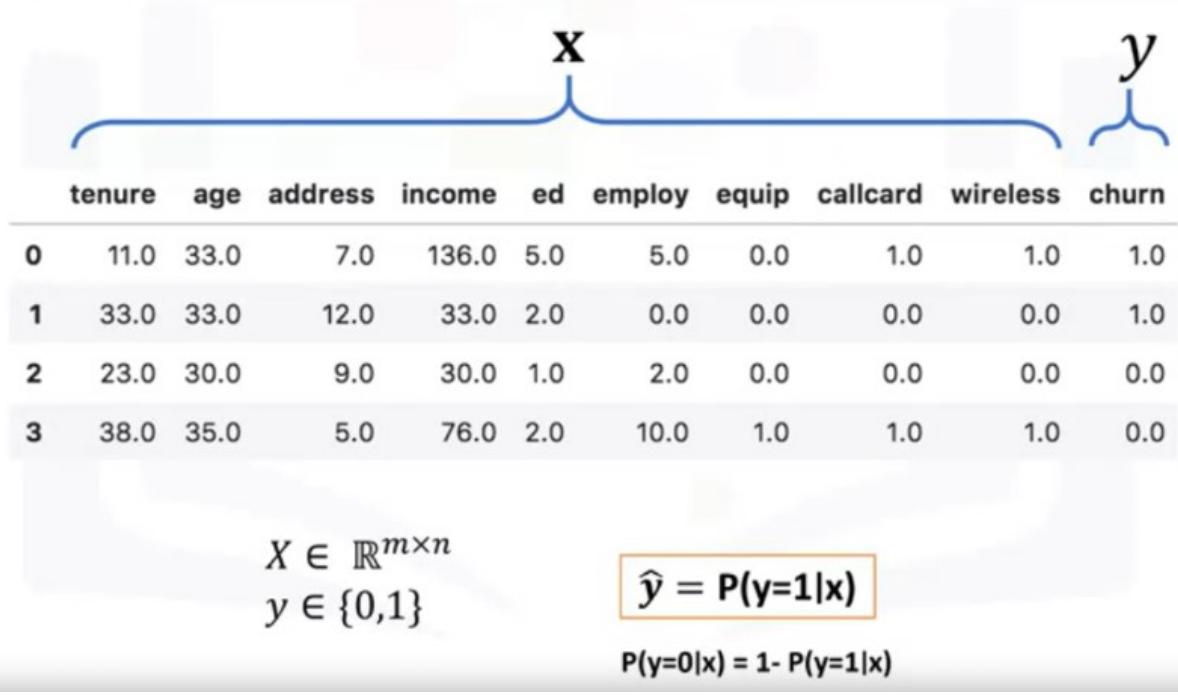


Figura. Modelo de regresión logística.

3.7. REGRESIÓN LOGÍSTICA VS REGRESIÓN LINEAL

Vamos a aprender la diferencia entre la regresión lineal y la regresión logística. Revisaremos la regresión lineal y veremos por qué no se puede usar correctamente para algunos problemas de clasificación binaria. Estudiaremos la función sigmoide, que es la parte principal de la regresión logística.

Veamos el conjunto de datos de telecomunicaciones de nuevo.

El objetivo de la regresión logística es construir un modelo para predecir la clase de cada cliente y también la probabilidad de que cada muestra pertenezca a una clase. Idealmente, queremos construir un modelo, y_{hat} , que pueda estimar la clase de un cliente dadas su características X .

y es el vector de etiquetas e y_{hat} es el vector de los valores pronosticados por nuestro modelo.

Asignando las etiquetas de clase a números enteros:

¿Podemos usar regresión lineal para resolver este problema?



	X										
	tenure	age	address	income	ed	employ	equip	callcard	wireless	churn	
0	11.0	33.0	7.0	136.0	5.0	5.0	0.0	1.0	1.0	Yes	
1	33.0	33.0	12.0	33.0	2.0	0.0	0.0	0.0	0.0	Yes	
2	23.0	30.0	9.0	30.0	1.0	2.0	0.0	0.0	0.0	No	
3	38.0	35.0	5.0	76.0	2.0	10.0	1.0	1.0	1.0	No	
4	7.0	35.0	14.0	80.0	2.0	15.0	0.0	1.0	0.0	No	

$$\hat{y} = P(y=1|x)$$

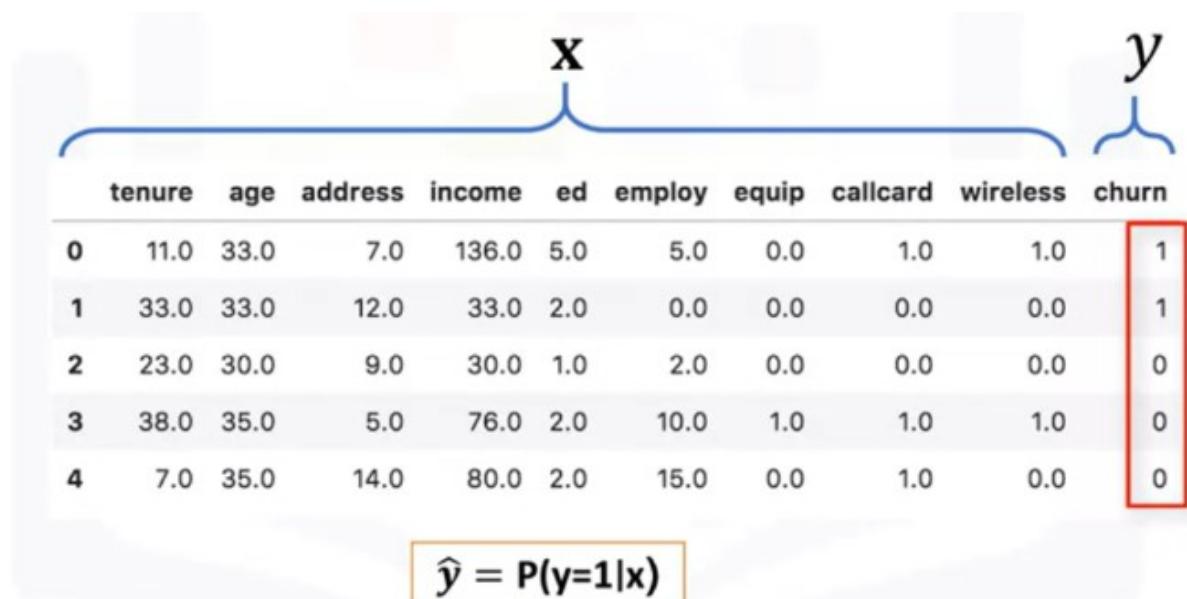


Figura. Codificamos las etiquetas como 1 y 0.

Recordemos cómo funciona la regresión lineal para comprender mejor la regresión logística.

Dejemos de lado la predicción de churn por un minuto y asumamos que nuestro objetivo es predecir los ingresos de los clientes en el conjunto de datos. Esto significa que en lugar de predecir el churn, que es un valor categórico, vamos a predecir el ingreso, que es un valor continuo.

¿Cómo podemos hacer esto?

Vamos a seleccionar una variable independiente como la edad del cliente y predecir la variable dependiente como el ingreso. Por supuesto podemos tener más características, pero por simplicidad utilizaremos solo una.

Podemos graficar y mostrar la edad como una variable independiente y el ingreso como el valor objetivo que nos gustaría predecir. Con la regresión lineal, puede ajustar una línea o un polinomio a través de los datos. Podemos encontrar esta línea entrenando nuestro modelo o calculándolo matemáticamente a partir de los conjuntos de muestras. Diremos que esta es una línea recta a través del conjunto de muestras. Ahora, utilice esta línea para predecir el valor continuo, y. Es decir, utilice esta línea para predecir los ingresos de un cliente desconocido en función de su edad.

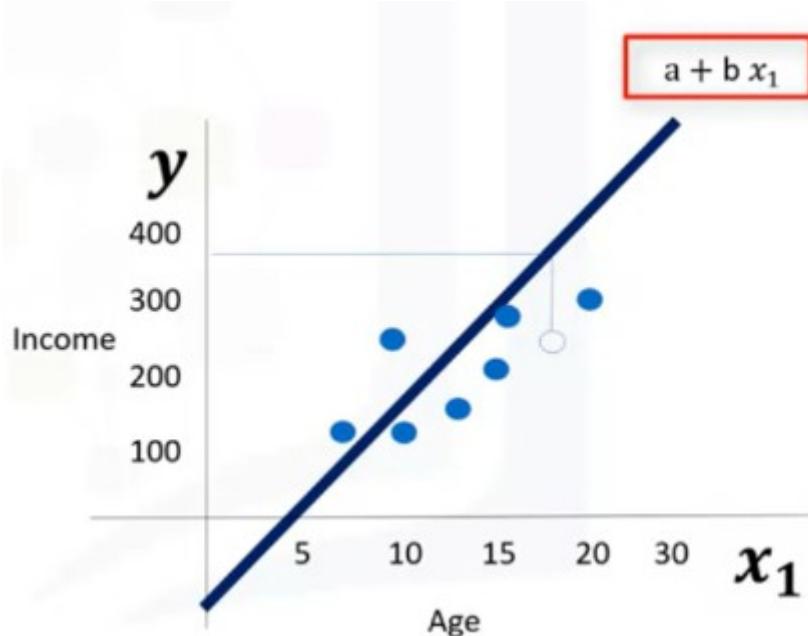


Figura. Regresión lineal para predicción del ingreso en función de la edad.

¿ Y si queremos predecir el churn?

¿ Podemos usar la misma técnica para predecir un campo categórico como churn?

Veamos.

Digamos, se nos dan datos sobre el churn de los clientes y nuestro objetivo esta vez es predecir el churn de los clientes en función de su edad.

Tenemos una característica, la edad denotada como x_1 , y una característica categórica, churn, con dos clases:

- churn es sí
- churn es no.

Como se mencionó, podemos asignar sí y no a valores enteros cero y uno.

¿Cómo podemos modelarlo ahora?

Gráficamente, podríamos representar nuestros datos con una gráfica de dispersión, pero esta vez, sólo tenemos dos valores para el eje Y. En esta gráfica, la clase cero se denota en rojo y la clase uno se denota en azul.

Nuestro objetivo aquí es hacer un modelo basado en los datos existentes para predecir si un nuevo cliente es rojo o azul.

Vamos a hacer la misma técnica que usamos para la regresión lineal aquí para ver si podemos resolver el problema para un atributo categórico como churn.

Con la regresión lineal, de nuevo puede ajustar un polinomio a través de los datos. Luego se realizan predicciones, por ejemplo, para el punto p1 de la figura siguiente se obtiene 0.3. Ahora, definimos un umbral, por ejemplo 0.5, para definir la clase. Así, si $\theta^T X$ es menor que 0.5 se decide es clase 0, y, si es mayor, se decide es clase 1. Entonces, en el ejemplo lo clasificaríamos en la clase 0.

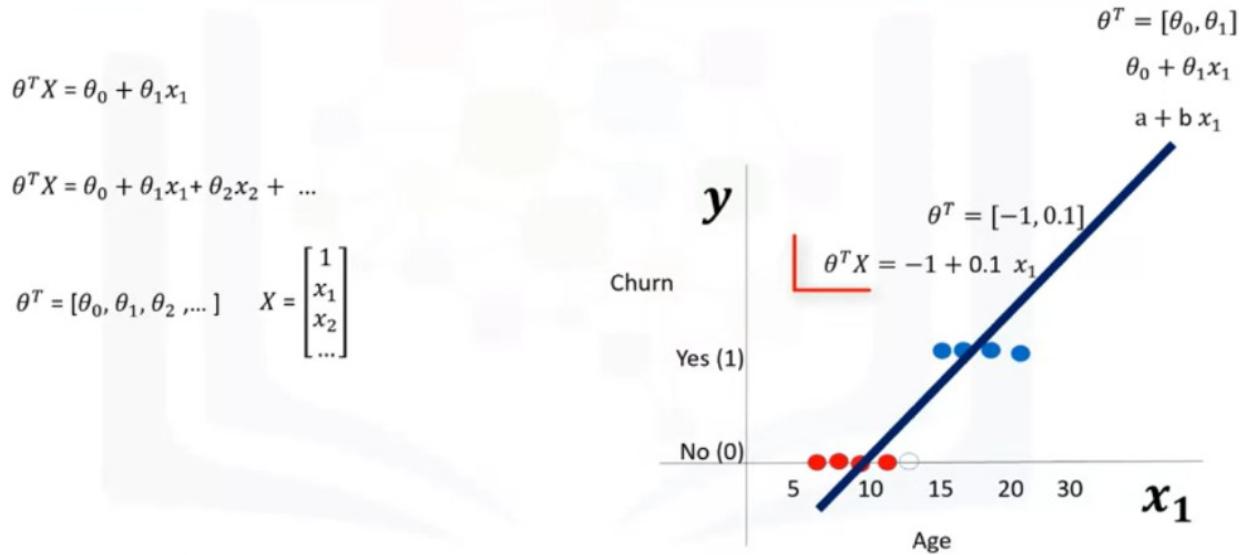


Figura. Regresión lineal aplicada en el problema del churn.

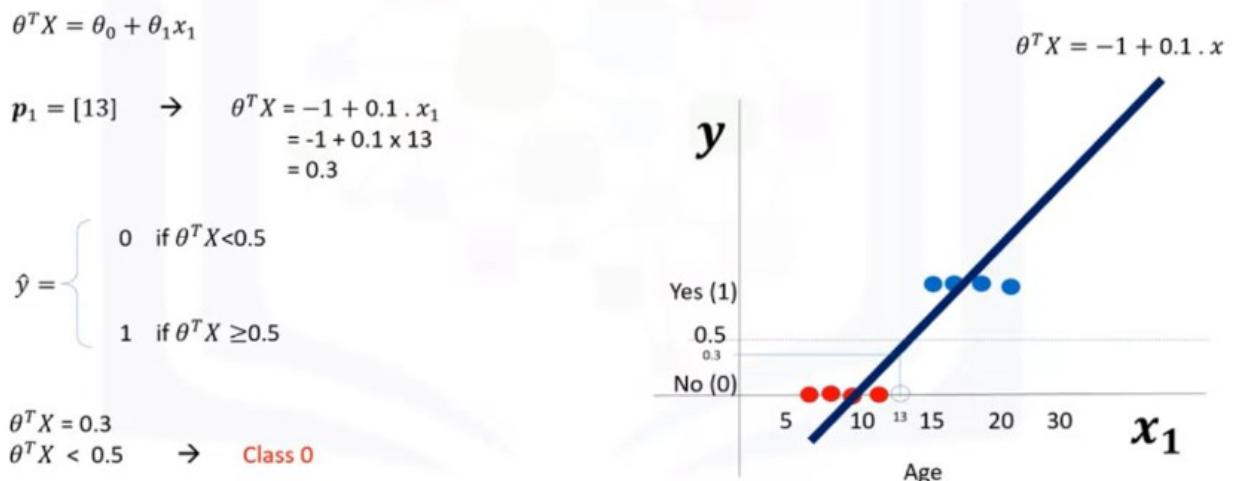


Figura. En el ejemplo decidimos la clase es la 0.

Pero hay un problema aquí:

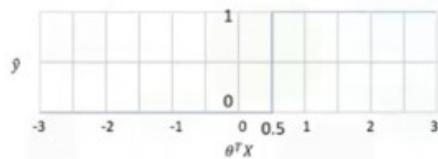
¿Cuál es la probabilidad de que este cliente pertenezca a la clase cero?

Como puede ver, no es el mejor modelo para resolver este problema.

Además, hay algunos otros problemas que verifican que la regresión lineal no es el método adecuado para los problemas de clasificación.

Así, como se mencionó, si usamos la línea de regresión para calcular la clase de un punto, siempre devuelve un número como -2 o 3. Luego, debemos usar un umbral, por ejemplo, 0.5, para asignar ese punto a cualquiera de las clases de cero o uno. Este umbral funciona como una función escalón, que genera cero o uno independientemente de lo grande o pequeño, positivo o negativo que sea la entrada. Por lo tanto, usando el umbral, podemos encontrar la clase de un registro. Observe que en la función escalón del ejemplo, no importa cuán grande sea el valor, siempre y cuando sea mayor que 0.5, simplemente equivale a uno y viceversa. Independientemente de lo pequeño que sea el valor y, la salida sería cero si es menor que 0.5. En otras palabras, no hay diferencia entre un cliente que tiene un valor de uno o 1.000, el resultado sería uno en ambos casos.

$$\theta^T X = \theta_0 + \theta_1 x_1 + \dots$$



$$\hat{y} = \begin{cases} 0 & \text{if } \theta^T X < 0.5 \\ 1 & \text{if } \theta^T X \geq 0.5 \end{cases}$$

Figura. Función escalón del ejemplo anterior.

En lugar de tener esta función escalón, ¿no sería bueno si tuviéramos una línea más suave, una que proyectara estos valores entre cero y uno?

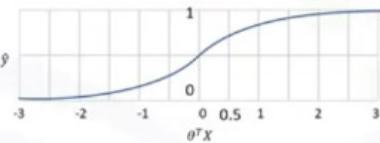
De hecho, el método existente no nos da realmente la probabilidad de que un cliente pertenezca a una clase. Necesitamos un método que pueda darnos la probabilidad de caer en la clase también.

¿Cuál es la solución?

En lugar de usar $\theta^T X$ utilizamos la función sigmoide aplicada a ella, lo que nos da la probabilidad de que un punto pertenezca a una clase en lugar del valor de y directamente.

Devuelve la probabilidad de que $\theta^T X$ sea muy grande o muy pequeña, su valor está entre 0 y 1.

$$\sigma(\theta^T X) = \sigma(\theta_0 + \theta_1 x_1 + \dots)$$



$$\hat{y} = \sigma(\theta^T X)$$

$$P(y=1|x)$$

Figura. Aplicamos la sigmoide en lugar del escalón.

¿Cuál es la función sigmoide?

La función sigmoide, también llamada función logística, tiene la siguiente ecuación:

$$\sigma(\theta^T X) = \frac{1}{1 + e^{-\theta^T X}}$$

Cuando $\theta^T X$ se hace muy grande, el valor de la sigmoide se acerca a 1, y cuando se hace muy pequeño se acerca a 0.

Que la salida de la sigmoide siempre esté entre 0 y 1 hace que sea apropiado interpretar los resultados como probabilidades.

$$\sigma(\theta^T X)$$

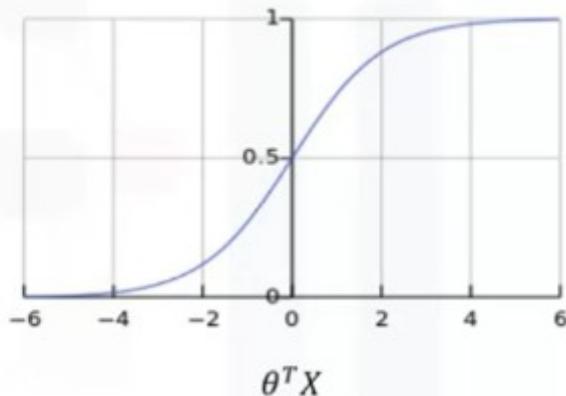


Figura. Sigmoide.

Cuando el resultado de la función sigmoide se acerca a 1, la probabilidad de $y=1$ dada x sube.

Cuando el valor sigmoide está más cerca de 0, la probabilidad de $y=1$ dado x es muy pequeña.

¿Cuál es la salida de nuestro modelo cuando usamos la función sigmoide?

La probabilidad de que una entrada x pertenezca a la clase predeterminada $y = 1$ se escribe como:

$$P(y=1|X)$$

La probabilidad de que Y pertenezca a la clase 0 es:

$$P(y=0|X) = 1 - P(y=1|X)$$

En nuestro ejemplo:

$$P(\text{churn}=1 | \text{income}, \text{age}) = 0.8$$

$$P(\text{churn}=0 | \text{income}, \text{age}) = 1 - 0.8 = 0.2$$

$$\sigma(\theta^T X) \longrightarrow P(y=1|x)$$

$$1 - \sigma(\theta^T X) \longrightarrow P(y=0|x)$$

Ahora nuestro trabajo es entrenar el modelo para establecer sus valores de parámetros de modo que resulte ser bueno.

Veamos cuál es el **proceso de entrenamiento**.

1. Inicializamos el vector θ con valores aleatorios.
2. Calculamos la salida del modelo, que es la sigmoide aplicada a $\theta^T X$: $y\hat{=} = \sigma(\theta^T X)$.
 1. La salida de esta ecuación es el valor de predicción, es decir, la probabilidad de que el cliente pertenezca a la clase 1.

3. Comparamos la salida de nuestro modelo, y_{hat} , con la etiqueta real y registramos la diferencia como nuestro error.
 1. En nuestro ejemplo, es el error para un único cliente.
4. Calcule el error para todos los clientes como hicimos en los pasos anteriores y sume estos errores.
 1. El error total es el coste del modelo y se calcula mediante la **función de coste** de los modelos.
 1. La función de costo, básicamente representa cómo calcular el error del modelo que es la diferencia entre los valores reales y los modelos predecidos. Por lo tanto, el costo muestra lo mal que el modelo está estimando las etiquetas de los clientes. Por lo tanto, cuanto menor sea el costo, mejor será el modelo para estimar correctamente las etiquetas de los clientes. Así, lo que queremos hacer es tratar de minimizar este costo.
5. Debido a que los valores iniciales de Theta fueron elegidos al azar, es muy probable que la función de costo sea muy alta, por lo que cambiamos el Theta de tal manera que esperamos reducir el costo total.
6. Después de cambiar los valores de Theta, volvemos al paso dos, luego comenzamos otra iteración y calculamos el costo del modelo nuevamente. Seguimos haciendo esos pasos una y otra vez, cambiando los valores de Theta cada vez hasta que el costo sea lo suficientemente bajo.

Esto genera 2 preguntas:

¿Cómo podemos cambiar los valores de Theta para que el costo se reduzca a través de iteraciones?

Hay diferentes formas, pero una de las formas más populares es el **descenso del gradiente**.

¿Cuándo deberíamos detener las iteraciones?

Cuando acorde a los criterios la precisión es satisfactoria.

$$\theta = [-1, 2]$$

$$\hat{y} = \sigma([-1, 2] \times [2, 5]) = 0.7$$

$$\text{Error} = 1 - 0.7 = 0.3$$

$$Cost = J(\theta)$$

$$\theta_{new}$$

Figura. Ejemplo de proceso de entrenamiento.

3.8. ENTRENAMIENTO EN REGRESIÓN LOGÍSTICA

El objetivo principal del entrenamiento es cambiar los parámetros del modelo, para mejorar la estimación.

Primero tenemos que mirar la función de coste, y ver cuál es la relación entre la función de coste y los parámetros (theta en este caso).

Por lo tanto, debemos formular la función de costo. Luego, usando la derivada de la función de costo podemos encontrar cómo cambiar los parámetros para reducir el costo.

Busquemos primero la ecuación de función de costo para un caso de muestra. Para hacer esto, podemos usar uno de los clientes en el problema de churn. Normalmente hay una ecuación general para calcular el costo. La función de coste es la diferencia entre los valores reales de y y nuestro modelo de salida \hat{y} . Esta es una regla general para la mayoría de las funciones de coste en el aprendizaje automático. Podemos mostrar esto como el costo de nuestro modelo comparándolo con las etiquetas reales, que es la diferencia entre el valor predicho de nuestro modelo y el valor real del campo objetivo, donde el valor predicho de nuestro modelo es la sigmoide aplicada a $\theta^T X$. Se eleva al cuadrado por la posibilidad de que haya valores negativos y se agrega el factor $\frac{1}{2}$ por conveniencia.

Ahora, podemos escribir la función de coste para todas las muestras en nuestro conjunto de entrenamiento. Por ejemplo, para todos los clientes podemos escribirlo como la suma media de las funciones de costo de todos los casos. También se denomina error cuadrado medio y como es una función de un vector theta.

$$Cost(\hat{y}, y) = \frac{1}{2} (\sigma(\theta^T X) - y)^2$$

$$J(\theta) = \frac{1}{m} \sum_{i=1}^m Cost(\hat{y}_i, y_i)$$

Figura. Función de costo.

¿Cómo podemos encontrar o establecer los mejores pesos o parámetros que minimicen esta función de costo?

Debemos calcular el punto mínimo de esta función de costo.

No hay una manera fácil de encontrar el punto mínimo global para tal ecuación. Dada esta complejidad, describir cómo alcanzar el mínimo global para esta ecuación está fuera del alcance de esta sección.

¿Cuál es la solución?

Debemos encontrar otra función de costo, que tenga el mismo comportamiento, pero sea más fácil de minimizar.

Vamos a graficar la función de costo deseable para nuestro modelo.

Nuestro valor real es y que equivale a cero o uno, y nuestro modelo intenta estimarlo como queremos encontrar una función de costo simple para nuestro modelo.

Por un momento supongamos que nuestro valor deseado para y es uno. Esto significa que nuestro modelo es mejor si estima y es igual a uno. En este caso, necesitamos una función de coste que devuelva cero si el resultado de nuestro modelo es uno, que es el mismo que la etiqueta real. Y el costo debe seguir aumentando a medida que el resultado de nuestro modelo se aleja de uno. Y el costo debe ser muy grande si el resultado de nuestro modelo es cercano a cero.

Así que si el valor real es uno y el modelo también predice uno, la función de registro menos devuelve costo cero. Pero si la predicción es menor que uno, la función de registro menos devuelve un valor de costo mayor. Por lo tanto, podemos utilizar la función $-\log(\hat{y})$ para calcular el costo de nuestro modelo de regresión logística.

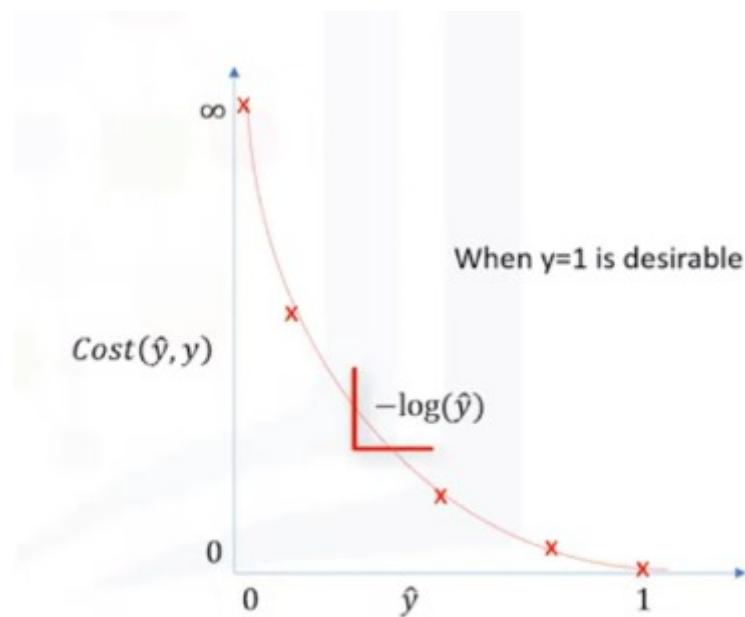


Figura. Función de costo logarítmica.

Para tal función es mucho más sencillo calcular el costo:

$$Cost(\hat{y}, y) = \begin{cases} -\log(\hat{y}) & \text{if } y = 1 \\ -\log(1 - \hat{y}) & \text{if } y = 0 \end{cases}$$
$$J(\theta) = -\frac{1}{m} \sum_{i=1}^m y^i \log(\hat{y}^i) + (1 - y^i) \log(1 - \hat{y}^i)$$

Como puede verse, penaliza situaciones en las que la clase es cero y la salida del modelo es uno, y viceversa.

Recuerde que \hat{y} no devuelve una clase como salida, es un valor que debe interpretarse como una probabilidad.

Ahora, podemos utilizar fácilmente esta función para encontrar los parámetros de nuestro modelo de tal manera que se minimice el costo.

Recapitulemos:

Nuestro objetivo era encontrar un modelo que sea el que estime mejor.

Encontrar el mejor modelo significa encontrar los mejores parámetros θ para ese modelo.

¿Cómo encontramos los mejores parámetros para nuestro modelo?

Encontrando y minimizando la función de costo.

¿Cómo minimizamos la función de costo?

Usando un enfoque de optimización, uno de los más famosos y efectivos es el descenso del gradiente.

¿Qué es el descenso de gradiente?

Es un enfoque iterativo para encontrar el mínimo de una función. Específicamente en nuestro caso es una técnica para utilizar la derivada de una función de costo para cambiar los valores de los parámetros para minimizar el costo o error. El objetivo principal del descenso de gradiente es cambiar los valores de los parámetros para minimizar el costo.

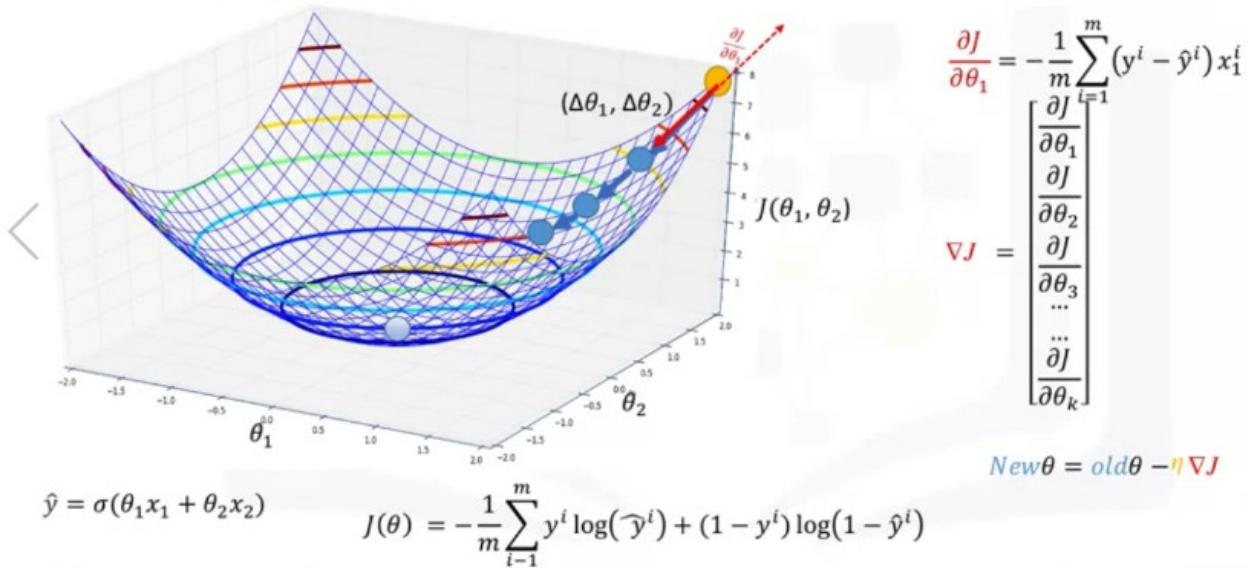


Figura. Descenso del gradiente.

Recapitulemos:

1. Inicializamos los parámetros con valores aleatorios.
2. Alimentamos la función de costo con el conjunto de entrenamiento y calculamos el costo.
 1. Esperamos una alta tasa de error ya que los parámetros se establecen aleatoriamente.
3. Calculamos el gradiente de la función de coste teniendo en cuenta que tenemos que utilizar una derivada parcial.
 1. Por lo tanto, para calcular el vector de gradiente necesitamos todos los datos de entrenamiento para alimentar la ecuación de cada parámetro.
 2. Esta es una parte costosa del algoritmo, pero hay algunas soluciones.
4. Actualizamos los pesos con nuevos valores de parámetros.
5. Volvemos al paso dos y alimentamos la función de costo nuevamente, que tiene nuevos parámetros.
 1. Esperamos menos errores a medida que bajamos por la superficie del error.
 2. Continuamos este bucle hasta que alcancemos un valor corto de costo o un número limitado de iteraciones.
6. Se obtienen los parámetros luego de cierto número de iteraciones. Ahora el modelo está listo.

3.9. SUPPORT VECTOR MACHINES (SVM)

Estudiaremos un método llamado SVM (Support Vector Machine), que se utiliza para la clasificación.

Imagine que ha obtenido un conjunto de datos que contiene las características de miles de muestras de células humanas extraídas de pacientes que se creía que estaban en riesgo de desarrollar cáncer.

El análisis de los datos originales mostró que muchas de las características diferían significativamente entre muestras benignas y malignas. Puede utilizar los valores de estas características celulares en muestras de otros pacientes, para dar una indicación temprana de si una nueva muestra puede ser benigna o maligna.

Puede utilizar Support Vector Machine, o SVM, como clasificador para entrenar su modelo para comprender los patrones dentro de los datos.

Una vez que el modelo ha sido entrenado, se puede utilizar para predecir su célula nueva o desconocida con una precisión bastante alta.



Figura. SVM para la clasificación de células en benignas o malignas.

Una **máquina de vectores de soporte** es un algoritmo supervisado que puede clasificar los casos mediante la búsqueda de un separador.

SVM funciona mapeando primero los datos a un espacio de entidades de alta dimensión para que los puntos de datos se puedan categorizar, incluso cuando los datos no se puedan separar linealmente. A continuación, se estima un separador para los datos. Los

datos deben transformarse de tal manera que un separador pueda dibujarse como un hiperplano.

Por ejemplo, considere la siguiente figura, que muestra la distribución de un pequeño conjunto de celdas únicamente en función del tamaño de la unidad y el grosor de la agrupación. Como puede ver, los puntos de datos se dividen en dos categorías diferentes. Representa un conjunto de datos linealmente no separable. Las dos categorías se pueden separar con una curva pero no con una línea. Es decir, representa un conjunto de datos linealmente no separable, que es el caso de la mayoría de los conjuntos de datos del mundo real.

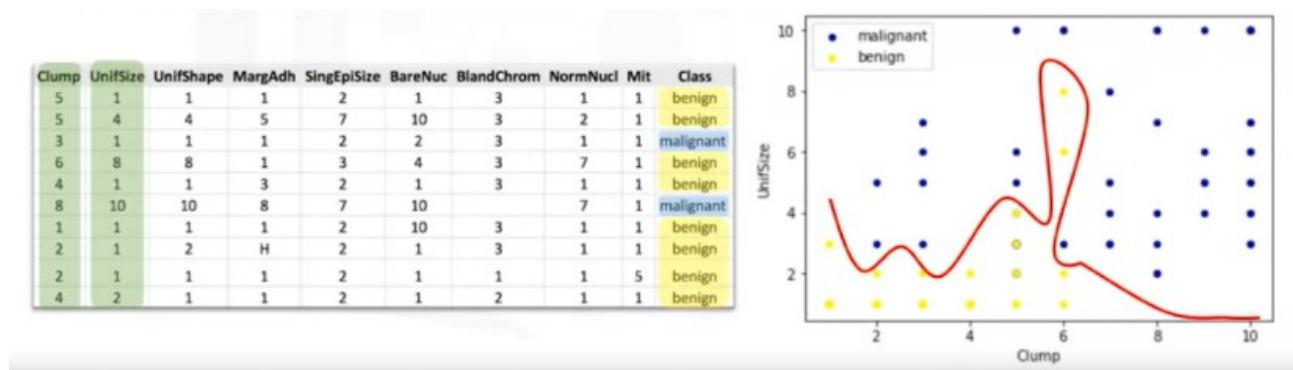


Figura. Conjunto de datos que NO es linealmente separable.

Podemos transferir estos datos a un espacio de dimensiones superiores, por ejemplo, mapeándolo a un espacio tridimensional. Después de la transformación, el límite entre las dos categorías se puede definir mediante un hiperplano. Como estamos ahora en el espacio tridimensional, el separador se muestra como un plano, que se puede utilizar para clasificar casos nuevos o desconocidos.

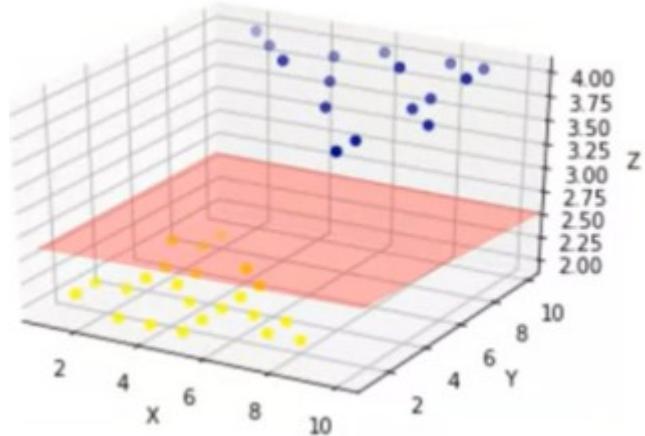


Figura. Mapeamos a un espacio de dimensión superior donde podemos utilizar un plano para separar los datos.

Así, el algoritmo SVM genera un hiperplano óptimo que categoriza nuevos ejemplos.

Hay 2 preguntas a considerar:

¿Cómo transferimos datos de tal manera que un separador pueda dibujarse como un hiperplano?

¿Cómo podemos encontrar el mejor separador de hiperplanos u optimizado después de la transformación?

Veamos primero la transformación de datos para ver cómo funcionan.

En aras de la simplicidad, imagine que nuestro conjunto de datos es unidimensional. Esto significa que sólo tenemos una característica x . Asumamos que no es separable linealmente. Entonces, ¿qué podemos hacer aquí? Bueno, podemos transferirlo a un espacio bidimensional. Por ejemplo, puede aumentar la dimensión de los datos asignando x a un nuevo espacio utilizando una función con salidas x y x^2 . Ahora los datos son linealmente separables.

Observe que como estamos en un espacio bidimensional, el hiperplano es una línea que divide un plano en dos partes donde cada clase se encuentra a cada lado. Ahora podemos usar esta línea para clasificar nuevos casos.

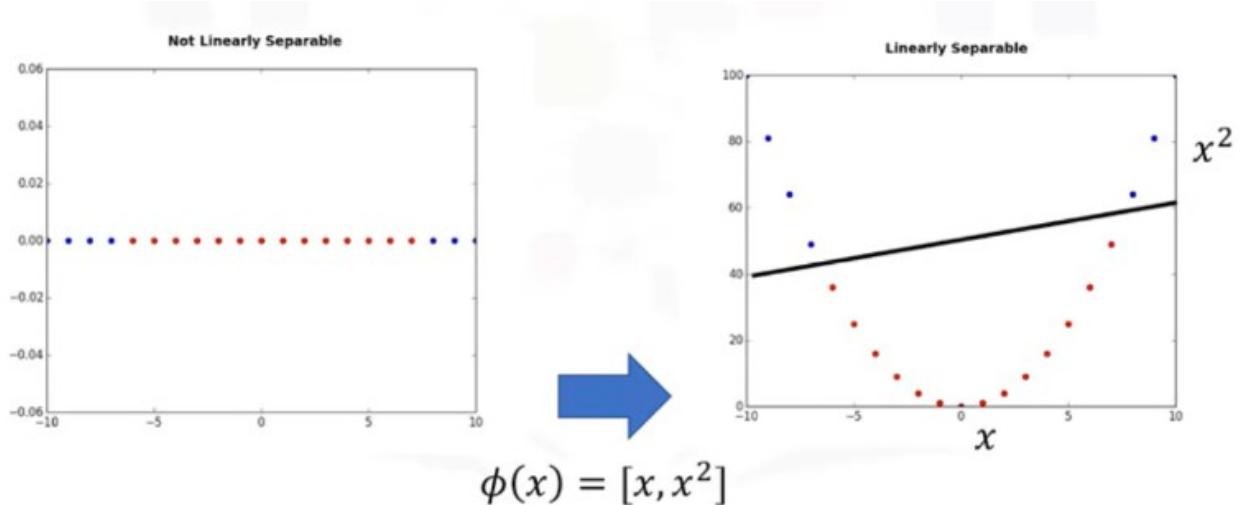


Figura. Transformación de datos.

El mapeo de datos en un espacio de dimensiones superiores se llama, **kernelling**. La función matemática utilizada para la transformación se conoce como la **función del núcleo**, y puede ser de diferentes tipos, como lineal, polinomial, función de base radial o RBF, y sigmoide. Cada una de estas funciones tiene sus propias características, sus pros y sus contras, y su ecuación.

Como no hay una manera fácil de saber qué función funciona mejor con cualquier conjunto de datos dado, generalmente elegimos diferentes funciones a su vez y comparamos los resultados.

Ahora, ¿cómo encontramos el separador correcto u optimizado después de la transformación?

Básicamente, las SVM se basan en la idea de encontrar un hiperplano que divida mejor un conjunto de datos en dos clases, como se muestra en la figura siguiente.

Como estamos en un espacio bidimensional, puedes pensar en el hiperplano como una línea que separa linealmente los puntos azules de los puntos rojos. Una opción razonable como el mejor hiperplano es la que representa la mayor separación o margen entre las dos clases. Así que el objetivo es elegir un hiperplano con un margen lo más grande posible. Los ejemplos más cercanos al hiperplano son los **vectores de soporte**. Es intuitivo que solo los vectores de soporte importan para lograr nuestro objetivo. Tratamos de encontrar el hiperplano de tal manera que tenga la distancia máxima a los vectores de soporte.

El hiperplano se aprende de los datos de entrenamiento mediante un procedimiento de optimización que maximiza el margen. Al igual que muchos otros problemas, este problema de optimización también se puede resolver mediante el descenso de gradiente. La salida del algoritmo son los valores w y b para la línea. Puede realizar clasificaciones utilizando esta línea estimada. Es suficiente conectar los valores de entrada en la ecuación de línea. A continuación, puede calcular si un punto desconocido está por encima o por debajo de la línea. Si la ecuación devuelve un valor mayor que 0, entonces el punto pertenece a la primera clase que está por encima de la línea, y viceversa.

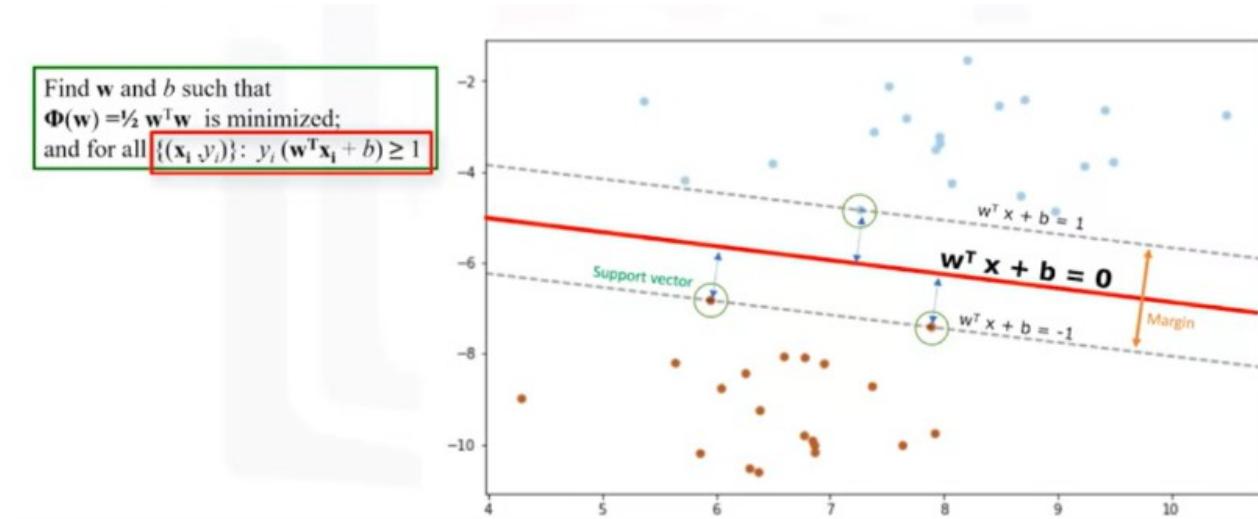


Figura. Encontrando el separador correcto.

Las ventajas de SVM son:

- Alta precisión en espacios de alta dimensión.
- Son eficientes desde el punto de vista de la memoria.

Desventajas de SVM:

- El algoritmo es propenso a sobreajuste si el número de entidades es mucho mayor que el número de muestras.
- No proporcionan directamente estimaciones de probabilidad, que son deseables en la mayoría de los problemas de clasificación.
- No son muy eficientes computacionalmente si su conjunto de datos es muy grande, como cuando tiene más de 1.000 filas.

¿En qué situación debería usar SVM?

- Es bueno para tareas de análisis de imágenes, como clasificación de imágenes y reconocimiento de dígitos escritos a mano.
- es muy eficaz en tareas de minería de texto, particularmente debido a su efectividad en el tratamiento de datos de alta dimensión.
 - Por ejemplo, se utiliza para detectar spam, asignación de categorías de texto y análisis de opinión.
- Clasificación de datos de expresión génica, de nuevo, debido a su poder en la clasificación de datos de alta dimensión.
- Regresión, detección de valores atípicos y clustering.

PARTE 4. CLUSTERING

4.1. INTRODUCCIÓN AL CLUSTERING

Daremos una introducción de alto nivel al clustering, sus aplicaciones y los diferentes tipos.

Imagine que tiene un conjunto de datos de clientes y que necesita aplicar la segmentación de clientes en base a los mismos.

La segmentación de clientes es la práctica de dividir una base de clientes en grupos que tengan características similares. Es una estrategia significativa, ya que permite que el negocio se dirija a grupos específicos de clientes, con el fin de asignar más eficazmente los recursos de marketing.

Por ejemplo, un grupo puede contener clientes con altos beneficios y bajo riesgo. Es decir, es más probable que compre productos o se suscriba a un servicio. Conocer esta información permite a una empresa dedicar más tiempo y atención a retener a estos clientes. Otro grupo podría incluir clientes de organizaciones sin fines de lucro, etc.

Un proceso general de segmentación no suele ser factible para grandes volúmenes de datos, por lo que necesita un enfoque analítico para derivar segmentos y grupos de grandes datasets.

Los clientes se pueden agrupar en función de varios factores, como:

- Edad
- Sexo
- Hábitos de gasto
- Otros

El requisito importante es utilizar los datos disponibles para comprender e identificar cómo los clientes son similares entre sí.

Vamos a aprender cómo dividir un conjunto de clientes en categorías, en función de las características que comparten. Uno de los enfoques más adoptados que se pueden utilizar para la segmentación de clientes es el **clustering**.

La agrupación en clústeres solo puede agrupar datos sin supervisión, en función de la similitud de los clientes entre sí.

- Se dividirá a sus clientes en grupos mutuamente excluyentes.
 - Por ejemplo, en tres clústeres.
 - Los clientes de cada clúster son similares demográficamente.

- Ahora podemos crear un perfil para cada grupo, teniendo en cuenta las características comunes de cada clúster.
 - Por ejemplo, el primer grupo formado por clientes ricos y de mediana edad. El segundo está formado por clientes jóvenes, educados y de ingresos medios, y el tercer grupo incluye clientes jóvenes y de bajos ingresos.
- Finalmente, podemos asignar cada individuo en nuestro conjunto de datos a uno de estos grupos o segmentos de clientes.

Ahora imagine que se une a este conjunto de datos segmentado con el conjunto de datos del producto o servicios que los clientes compran a su empresa. Esta información realmente ayudaría a entender y predecir las diferencias y preferencias individuales de los clientes y sus comportamientos de compra en varios productos. De hecho, tener esta información permitiría a su empresa desarrollar experiencias altamente personalizadas para cada segmento.

La segmentación de clientes es uno de los usos más populares del clustering.

Clustering significa buscar clústeres en un conjunto de datos, sin supervisión.

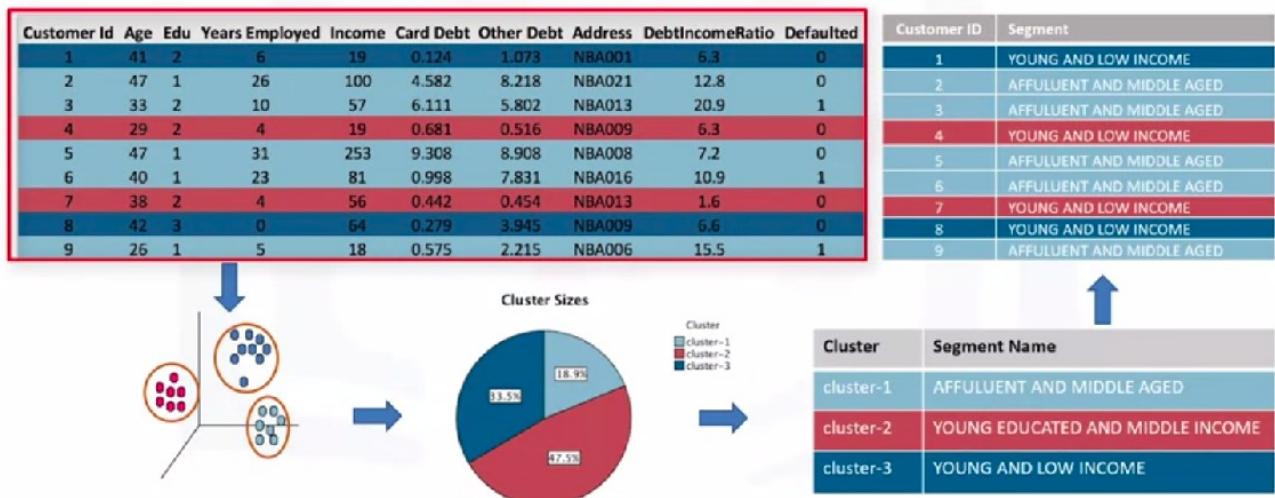


Figura. Clustering para segmentación de clientes.

Un **clúster** es un grupo de puntos de datos u objetos de un dataset que son similares a otros objetos del grupo y que no son similares a los puntos de datos de otros clústeres.

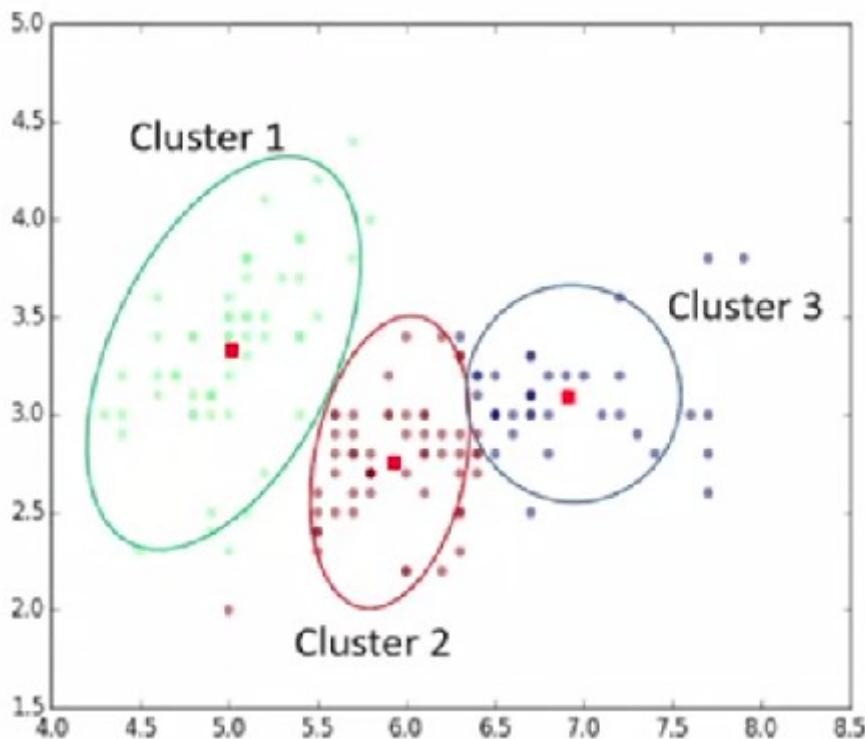


Figura. Qué es un clúster.

¿Qué es diferente entre clustering y clasificación?

Echemos un vistazo a nuestro conjunto de datos de clientes de nuevo.

Los algoritmos de clasificación predicen etiquetas clasificadas categóricas. Esto significa asignar instancias a clases predefinidas como predeterminadas o no predeterminadas.

Por ejemplo, si una analista desea analizar los datos de los clientes para saber qué clientes pueden no pagar sus préstamos, utiliza un conjunto de datos etiquetado como datos de entrenamiento y utiliza enfoques de clasificación como un árbol de decisiones, SVM, o regresión logística, para predecir el valor predeterminado para un cliente nuevo o desconocido. En términos generales, la clasificación es un aprendizaje supervisado en el que cada instancia de datos de formación pertenece a una clase en particular.

En clústeres, los datos no están etiquetados y el proceso no está supervisado. Por ejemplo, podemos usar un algoritmo de clustering como k-means para agrupar clientes similares como se mencionó, y asignarlos a un clúster, en función de si comparten atributos similares , como; edad, educación, etc.

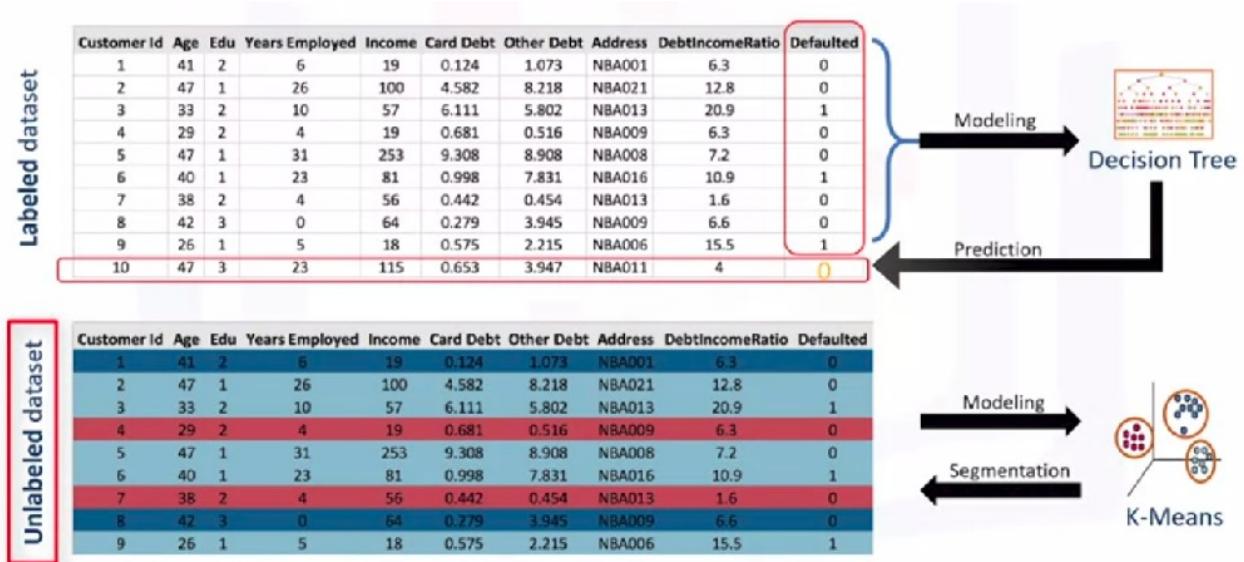


Figura. Diferencias entre clasificación y clustering.

Veamos algunas aplicaciones del clustering:

- En la industria minorista, la agrupación se utiliza para encontrar asociaciones entre clientes en función de sus características demográficas y utilizar esa información para identificar patrones de compra de varios grupos de clientes.
- se puede utilizar en sistemas de recomendación para encontrar un grupo de artículos similares o usuarios similares y utilizarlo para el filtrado colaborativo, para recomendar cosas como libros o películas a los clientes.
- En la banca, los analistas encuentran agrupaciones de transacciones normales para encontrar los patrones de uso fraudulento de tarjetas de crédito.
- Identificar clústeres de clientes.
 - Por ejemplo, para encontrar clientes leales frente a clientes que puedan querer dejar la empresa.
- En el sector de los seguros, la agrupación se utiliza para la detección de fraudes en el análisis de siniestros, o para evaluar el riesgo de seguros de ciertos clientes en función de sus segmentos.

- En los medios de publicación, el clustering se utiliza para categorizar automáticamente las noticias en función de su contenido o para etiquetar noticias, luego agruparlas para recomendar artículos de noticias similares a los lectores.
- En medicina, se puede usar para caracterizar el comportamiento del paciente, con el fin de identificar terapias médicas exitosas para diferentes enfermedades.
- En biología, el agrupamiento se utiliza para agrupar genes con patrones de expresión similares o para agrupar marcadores genéticos para identificar lazos familiares.
- Muchas otras.

Generalmente la agrupación se puede utilizar para uno de los siguientes propósitos:

- Análisis de datos exploratorios.
- Generación de resumen o reducción de la escala.
- Detección de valores atípicos.
 - Especialmente para ser utilizado para detección de fraude o eliminación de ruido.
- Encontrar duplicados y conjuntos de datos.
- Preprocesamiento para la predicción.

Veamos los **diferentes algoritmos de agrupamiento** y sus características.

- La **agrupación basada en particiones** es un grupo de algoritmos de agrupamiento que produce clústeres como esferas, como; K-Means, K-Medians o C-medias difusas.
 - Estos algoritmos son relativamente eficientes y se utilizan para bases de datos de tamaño mediano y grande.
- Los **algoritmos de agrupamiento jerárquico** producen árboles de clústeres, como algoritmos aglomerativos y divisivos.
 - Este grupo de algoritmos son muy intuitivos y generalmente son buenos para su uso con conjuntos de datos de pequeño tamaño.
- Los **algoritmos de agrupamiento basados en densidad** producen clústeres de forma arbitraria.
 - Son especialmente buenos cuando se trata de clústeres espaciales o cuando hay ruido en el conjunto de datos.

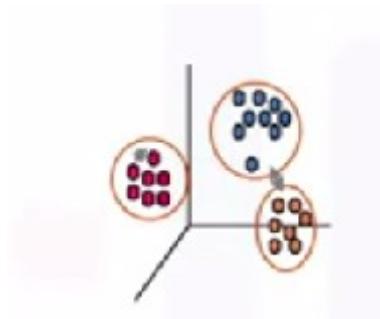


Figura. Clúster basado en particiones.

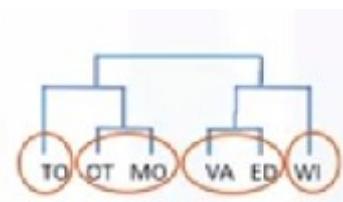


Figura. Clúster jerárquico.

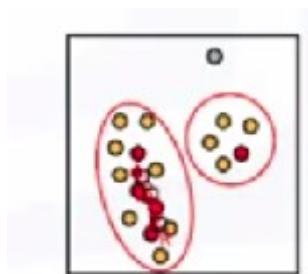


Figura. Clúster basado en densidades.

4.2. INTRODUCCIÓN A K-MEANS

Vamos a estar cubriendo **k-means** clustering.

Imagine que tiene un conjunto de datos de clientes y que necesita aplicar la segmentación. Uno de los algoritmos que se puede utilizar para ello es K-means.

k-means es un tipo de agrupamiento por partición que divide los datos en K subconjuntos o clústeres no superpuestos sin ninguna estructura interna de clúster. Los objetos dentro de un clúster son muy similares y los objetos a través de diferentes clústeres son muy diferentes.

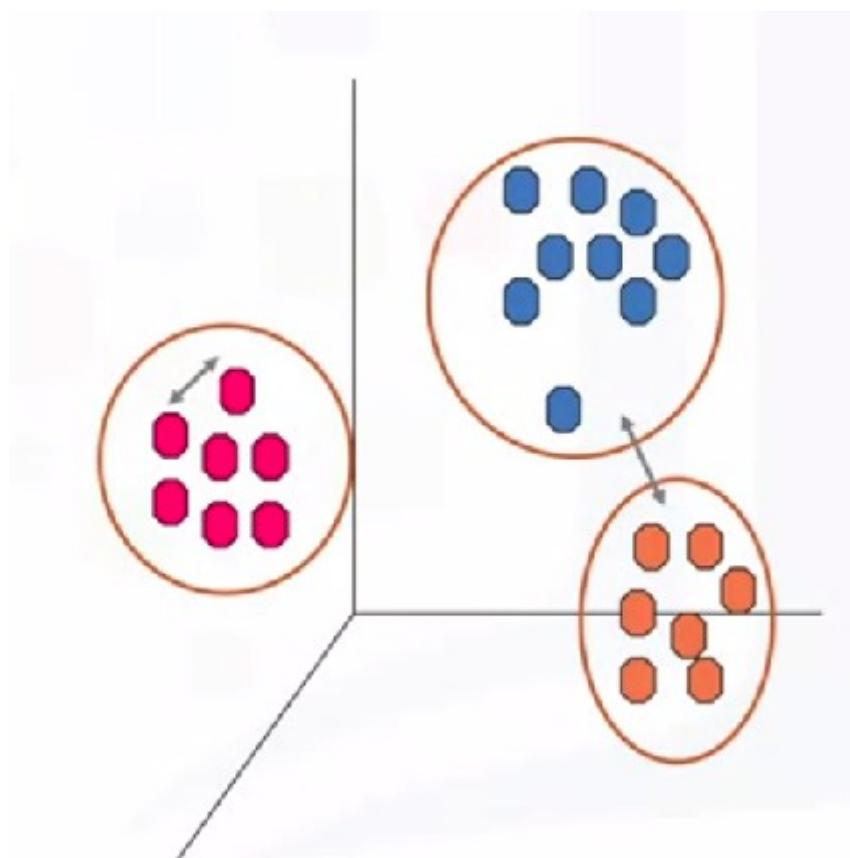


Figura. K-means.

Para el uso de k-means tenemos que encontrar muestras similares, por ejemplo clientes similares.

Nos enfrentamos a la siguiente interrogante:

¿ Cómo podemos encontrar la similitud de las muestras en la agrupación?

En nuestro ejemplo:

¿ Cómo medimos cuán similares son dos clientes con respecto a su demografía?

Aunque el objetivo de k-means es formar clústeres de tal manera que las muestras similares entren en un clúster y muestras diferentes caigan en diferentes grupos, se puede mostrar que en lugar de una métrica de similitud podemos usar métricas de disimilitud; en otras palabras, la distancia de muestras entre sí se usa para dar forma a los clústeres.

Podemos decir que **k-means intenta minimizar las distancias entre clústeres y maximizar las distancias entre clústeres**.

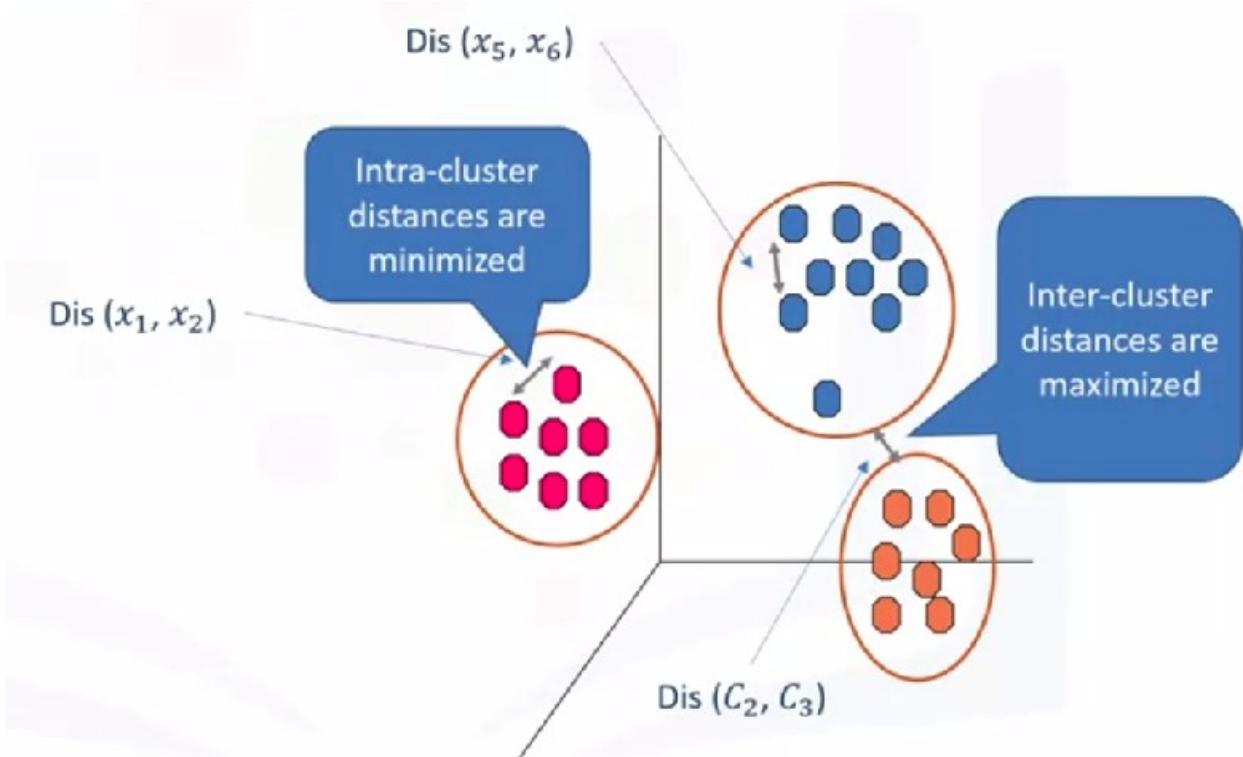


Figura. K-means.

Cómo podemos calcular la distancia entre 2 clientes?

Podemos usar por ejemplo la distancia euclídea.

Hay otras medidas; qué utilizar depende del tipo de datos y del dominio en el que se realiza la agrupación, por lo que se recomienda conocer bien los mismos.

Veamos cómo funciona k-means.

En aras de la simplicidad vamos a suponer que nuestro conjunto de datos tiene sólo dos características: la edad y el ingreso de los clientes. De esta forma, podemos mostrar la distribución de los clientes usando una gráfica de dispersión.

Intentaremos agrupar el dataset de clientes en distintos grupos o clusters basándonos en estas 2 dimensiones.

1. Determinar el número de clústeres.

El concepto clave del algoritmo de k-means es que selecciona aleatoriamente un centro para cada clúster.

Debemos inicializar K, que representa el número de clusters. Esto suele ser un problema complejo y lo trataremos luego; por ahora pongamos K = 3.

Es como si tuviéramos 3 puntos representativos para nuestros clústeres. Estos punto se llaman **centroides** y deben ser del mismo tamaño de características que nuestro conjunto de características.

Hay 2 enfoques para elegir los centroides:

- Elegimos 3 observaciones al azar y las utilizamos como las means iniciales.
- Creamos 3 puntos aleatorios como centroides de los clusters, que es la opción que seleccionamos en este ejemplo (puntos rojos de la figura siguiente).

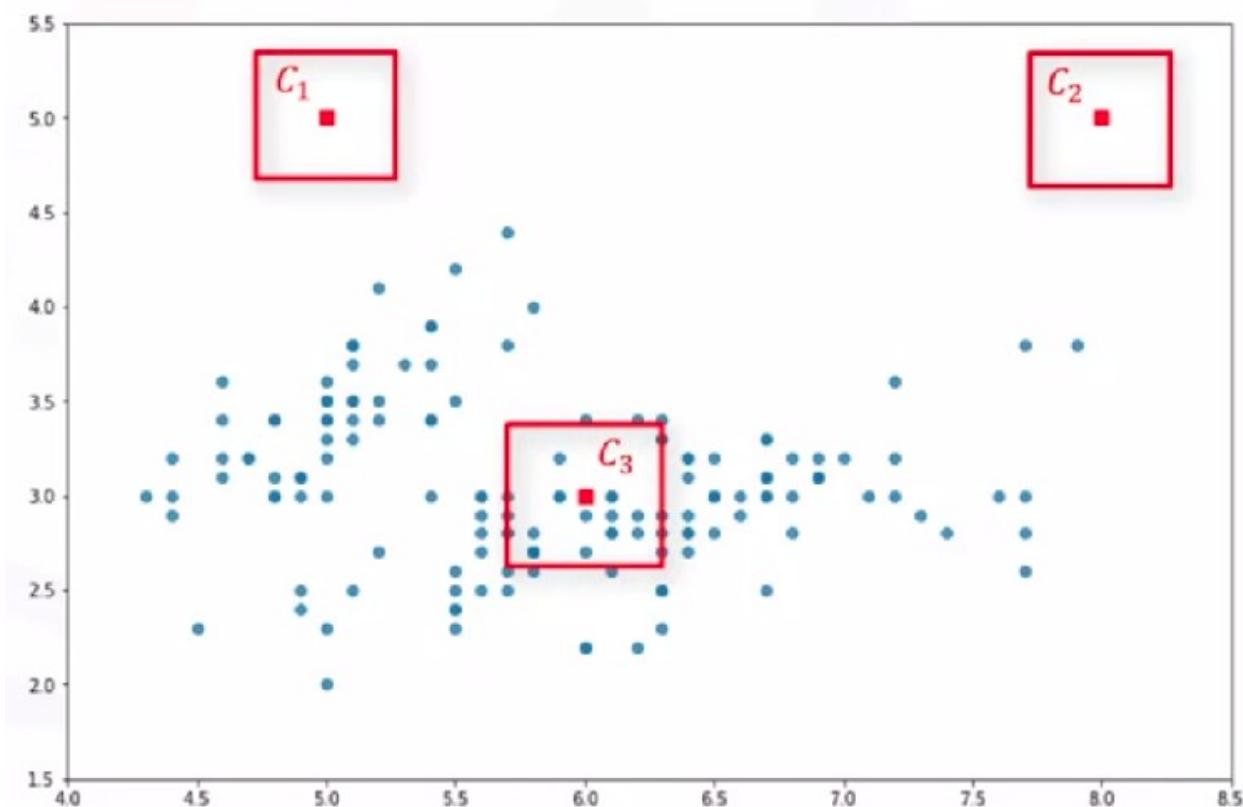


Figura. Creamos puntos aleatorios como centroides.

Paso 2. Cálculo de la distancia y asignación de cada punto a su centroide.

Luego de la inicialización, que fue definir el centroide de cada clúster, debemos asignar a cada cliente al centro más cercano.

Para este propósito debemos calcular la distancia de cada punto de datos o en nuestro caso de los clientes al centroide.

Dependiendo de la naturaleza de los datos y el propósito para el cual se crea el cluster se utilizan diferentes distancias.

Entonces, se forma una matriz donde cada fila representa la distancia de un cliente a cada centroide, llamada **Matriz de distancia**.

El objetivo principal de K-means es minimizar la distancia de los puntos de datos al centroide del cluster y maximizar la distancia a los otros centroides. Así, en este paso, debemos encontrar el centroide más cercano a cada punto de datos. Para ello usamos la matriz de distancia.

Al encontrar los centroides más cercanos, asignamos cada punto de datos a un cluster.

En otras palabras, todos los clientes van a caer en un cluster dependiendo de su distancia a los centroides.

Esto no resultará en buenos clusters porque los centroides fueron elegidos al azar; así, el modelo tendrá un error alto.

Aquí el error es la distancia total de cada punto a su centroide. Puede verse como la suma de errores cuadráticos inter-cluster. Intuitivamente, intentamos reducir este error, esto significa que debemos darle a los clusters una forma tal que la distancia al centroide sea minimizada.

Cómo mejoramos los clusters?

Movemos los centroides.

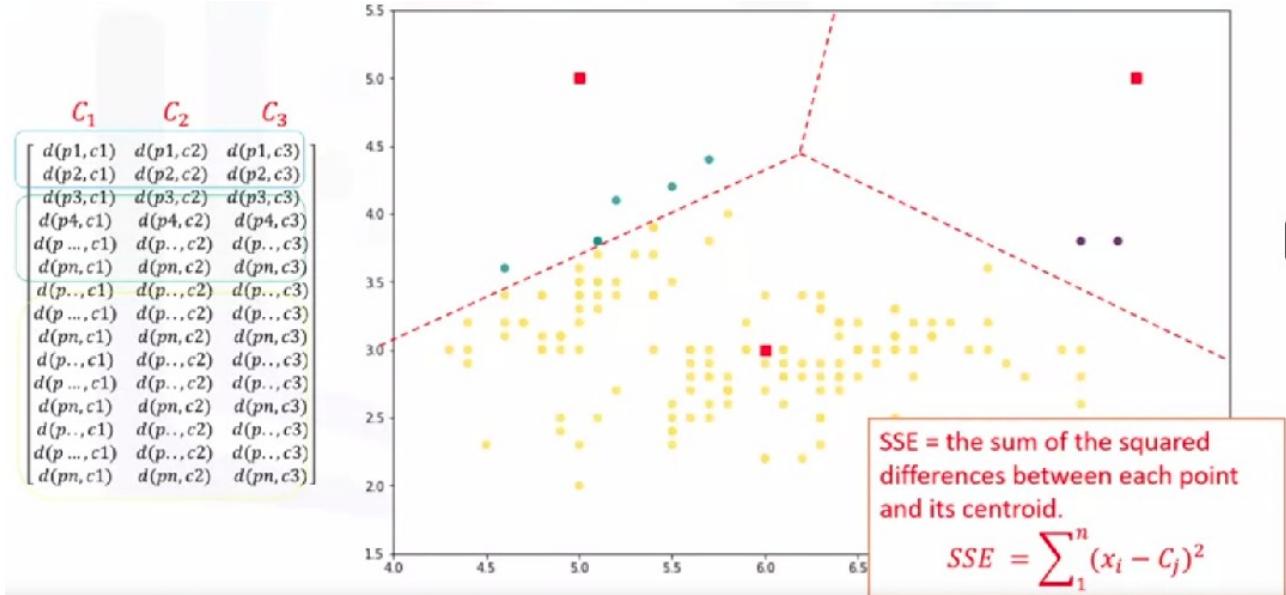


Figura. Asignamos cada punto al centroide más cercano.

Paso 3. Computamos los nuevos centroides.

Cada centro de cluster será actualizado para que sea la media de los puntos de datos en su cluster. De hecho, cada centroide se mueve acorde a sus miembros.

Por ejemplo, si se tienen 2 puntos en el cluster y:

- Las coordenadas del punto A son: (7.4, 3.6)
- Las coordenadas del punto B son: (7.8, 3.8)

el nuevo centroide de este cluster es el promedio de ellos

$$(7.6, 3.7)$$

Ahora tenemos nuevos centroides.

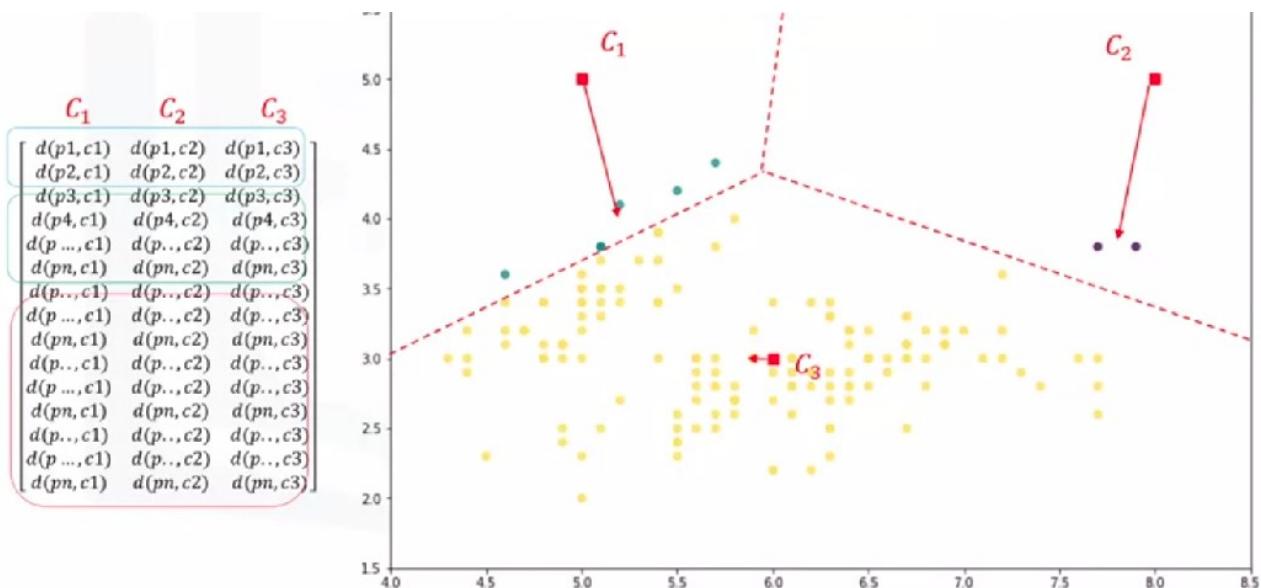


Figura. Computamos los nuevos centroides.

Paso 4. Repetimos hasta la convergencia.

Luego el proceso se repite hasta que los centroides ya NO se mueven.

Observe que cada vez que un centroide se mueve, cada distancia al centroide debe ser medida otra vez.

K-means es un algoritmo iterativo y debemos repetir los pasos hasta que el algoritmo converja.

En cada iteración:

- Mueve los centroides.
- Calcula la distancia desde los centroides.
- Asigna puntos de datos a los centroides más cercanos.

Esto resulta en clusters con mínimo error o más densos.

Sin embargo, es un algoritmo heurístico, no hay garantía de que converja al óptimo global y el resultado puede depender de los clusters iniciales. Para solucionar este problema es común ejecutar el proceso completo múltiples veces con diferentes condiciones iniciales. Como el algoritmo es muy rápido, esto no suele ser un problema.

4.3. MÁS SOBRE K-MEANS

En el algoritmo de k-means:

1. Se colocan aleatoriamente k centroides, uno para cada clúster. Cuanto más separados estén los clusters mejor.
2. Se calcula la distancia de cada punto de datos a cada centroide. Hay diferentes distancias, la euclídea es una de las más populares.
3. Se asigna cada punto de datos a su centroide más cercano, creando de esta forma un grupo.
4. Se vuelve a calcular la posición de los centroides. La nueva posición del centroide se determina por la media de todos los puntos del grupo.
5. Esto continúa hasta que los centroides ya no se mueven

¿ Cómo podemos evaluar la bondad de los clusters formados ?

En otras palabras, ¿cómo calculamos la precisión de la agrupación?

Una forma es comparar con la verdad, si está disponible. Sin embargo, debido a que K-means es un algoritmo no supervisado, generalmente no tenemos la verdad en los problemas del mundo real para ser utilizados.

Por suerte hay otras opciones. El promedio de las distancias de los puntos de datos desde sus centroides de clúster se puede utilizar como una métrica de error para el algoritmo de agrupamiento.

Determinar el número de clústeres en un conjunto de datos, K en el caso de K-means, es un problema frecuente en la agrupación de datos.

La elección correcta de K suele ser ambigua porque depende mucho de la forma y escala de la distribución de puntos en un dataset. Existen algunos enfoques para abordar este problema, pero una de las técnicas que se utiliza comúnmente es ejecutar la agrupación en los diferentes valores de K y buscar una métrica de precisión para la agrupación. Luego, mirando el cambio de esta métrica, podemos encontrar el mejor valor para K . Pero el problema es que con el aumento del número de clústeres, la distancia de los centroides a los puntos de datos siempre se reducirá. Esto significa que aumentar K siempre disminuirá el error. Por lo tanto, el valor de la métrica como una función de K se grafica y el punto del codo se determina donde la tasa de disminución cambia bruscamente. Es la K correcta para agrupar. Este método se denomina **método del codo**.

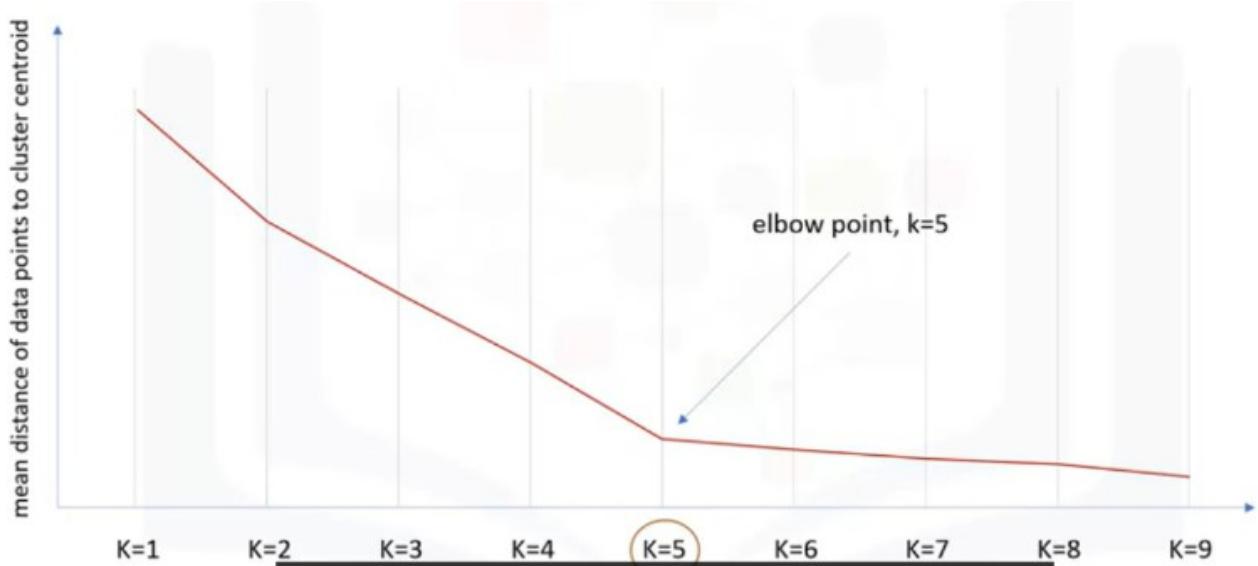


Figura. Método del codo para determinar el valor de K.

Recapitulemos: K-means es un algoritmo de clustering basado en particiones que:

- Es relativamente eficiente en conjuntos de datos de tamaño mediano y grande.
- Produce clústeres tipo esfera, porque los clústeres tienen forma esférica alrededor de los centroides.
- Su inconveniente es que debemos pre-especificar el número de clústeres, y esta no es una tarea fácil.

4.4. INTRODUCCIÓN AL CLUSTER JERÁRQUICO

Hablaremos del **cluster jerárquico**.

Miremos la figura siguiente.

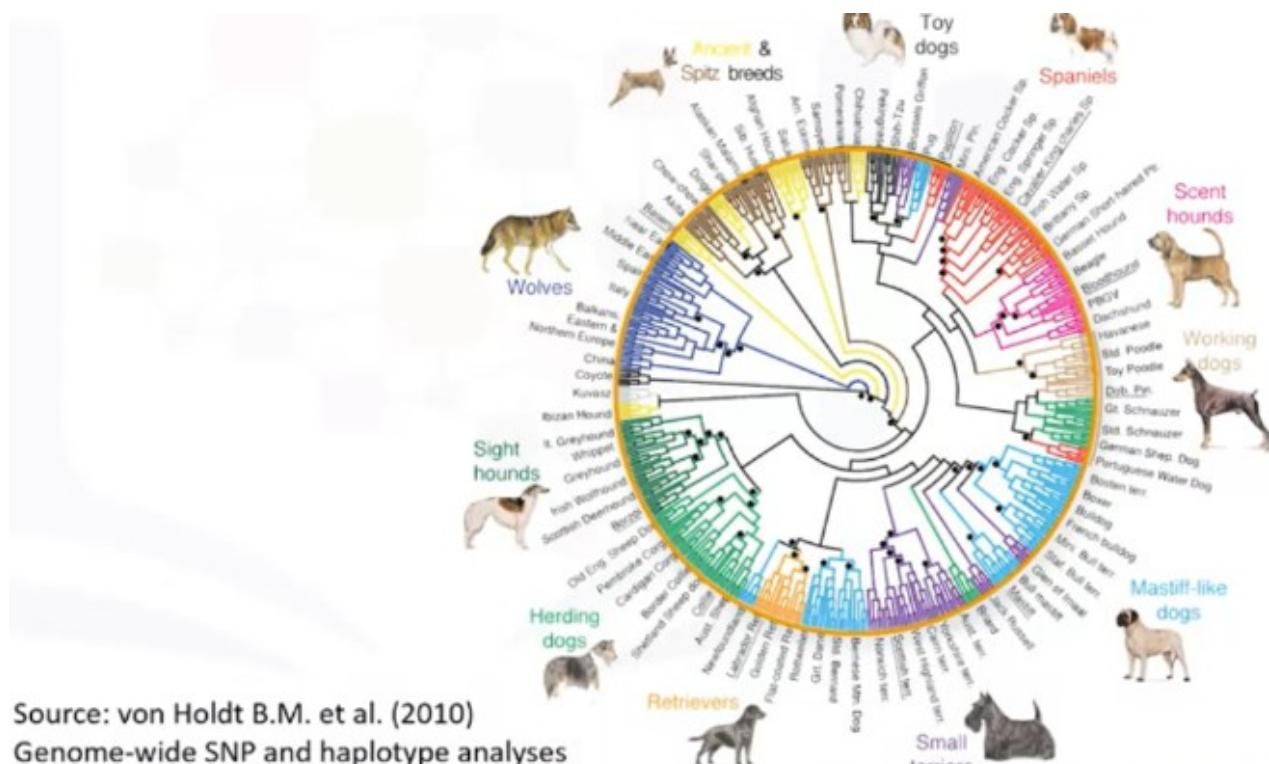


Figura. Datos genéticos de razas de perros y lobos.

Un equipo internacional de científicos liderados por biólogos de la UCLA utilizó este dendrograma para reportar datos genéticos de más de 900 perros de 85 razas, y más de 200 lobos grises salvajes en todo el mundo, incluyendo poblaciones de América del Norte , Europa, Oriente Medio y Asia Oriental. Utilizaron técnicas genéticas moleculares para analizar más de 48.000 marcadores genéticos.

Este diagrama muestra la agrupación jerárquica de estos animales en función de la similitud en sus datos genéticos.

Los algoritmos de agrupación jerárquica crean una jerarquía de clústeres donde cada nodo es un clúster que consta de los clústeres de sus nodos secundarios.

Las **estrategias para la agrupación jerárquica** se dividen en dos tipos:

- **Divisorias**
 - Es de arriba hacia abajo (top down), por lo que comienza con todas las observaciones en un grupo grande y lo divide en pedazos más pequeños.

- Piense en la división como dividir el clúster.

- **Agglomerativas**

- Es lo opuesto a la división.
- Es de abajo hacia arriba (bottom up), donde cada observación comienza en su propio clúster y los pares de clústeres se fusionan a medida que se mueven hacia arriba en la jerarquía.
- Significa acumular o recoger cosas

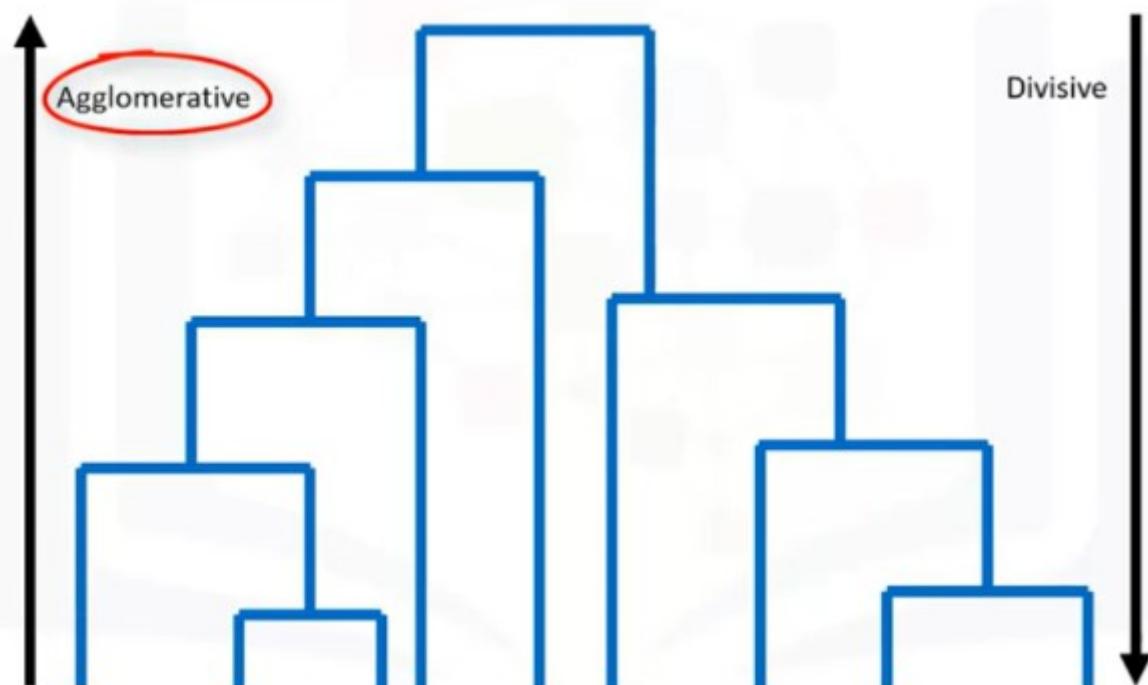


Figura. Estrategias para la agrupación jerárquica.

El enfoque aglomerativo es más popular entre los científicos de datos.

La agrupación jerárquica en el enfoque aglomerativo suele visualizarse como un **dendrograma**.

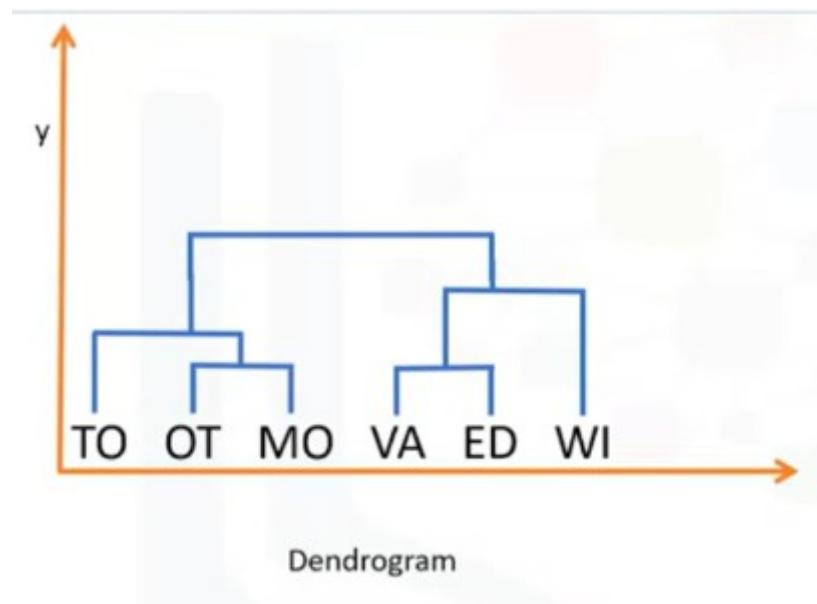


Figura. Dendograma.

Cada combinación está representada por una línea horizontal.

Al pasar de la capa inferior al nodo superior, un dendrograma nos permite reconstruir el historial de fusiones que dieron lugar a la agrupación representada.

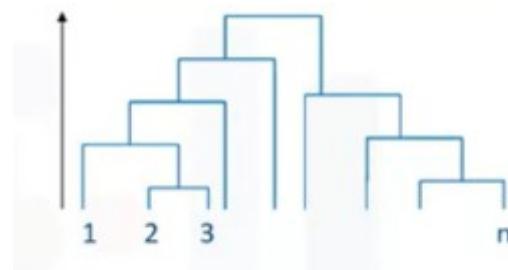
Básicamente, la agrupación jerárquica no requiere un número prespecificado de clústeres. Sin embargo, en algunas aplicaciones, queremos una partición de clústeres disjuntos al igual que en clústeres planos. En esos casos, la jerarquía necesita ser cortada en algún momento.

4.5. MAS SOBRE EL CLUSTERING JERÁRQUICO

Veamos el algoritmo aglomerativo para agrupamiento jerárquico, que es un enfoque bottom-up.

Digamos que nuestro conjunto de datos tiene n puntos, entonces:

1. Creamos n clústeres, uno para cada punto de datos. Cada punto se asigna como un clúster.
2. Calculamos la matriz de proximidad de distancia, que será $n \times n$.
3. Ejecutamos iterativamente los siguientes pasos hasta que se alcance el número de clusteres especificado o hasta que quede solo un clúster.
 1. Combinamos los 2 clústeres más cercanos; las distancias ya se obtuvieron en la matriz de proximidad.
 2. Se actualiza la matriz de proximidad con los nuevos valores.



$$\begin{bmatrix} 0 & & & & \\ d(2,1) & 0 & & & \\ d(3,1) & d(3,2) & 0 & & \\ \vdots & \vdots & \vdots & & \\ d(n,1) & d(n,2) & \dots & \dots & 0 \end{bmatrix}$$

Figura. Algoritmo aglomerativo.

Así que en la matriz de proximidad tenemos que medir las distancias entre los clústeres y a su vez debemos fusionar los clústeres que estén más cercanos.

Es clave el cálculo de la proximidad entre los clústeres con un punto y también clústeres con múltiples puntos de datos.

Hay preguntas que deben ser respondidas:

¿ Cómo medimos las distancias entre estos clústeres?

¿ Cómo definimos el más cercano entre los clústeres?

¿Qué puntos usamos?

Primero, veamos cómo calcular la distancia entre dos clústeres con un punto cada uno.

Supongamos que tenemos un conjunto de datos de pacientes y queremos agruparlos usando agrupamiento jerárquico. Así que nuestros puntos de datos son pacientes con un conjunto de por ejemplo 3 dimensiones, sean éstas edad, índice de masa corporal IMC y presión arterial para fijar ideas. Podemos utilizar diferentes distancias para calcular la matriz de proximidad, por ejemplo la euclídea. Por lo tanto, si tenemos un conjunto de datos de n pacientes, podemos construir una matriz de distancia de disimilitud n x n. Esto nos dará la distancia de los clústeres con un punto de datos.

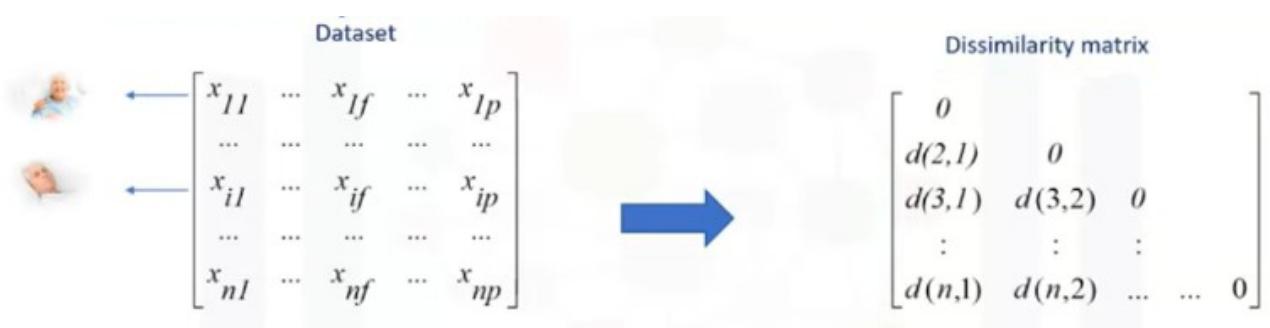


Figura. Distancia entre clústeres con un punto de datos.

Ahora la pregunta es:

¿Cómo podemos calcular la distancia entre clústeres cuando hay múltiples pacientes en cada grupo?

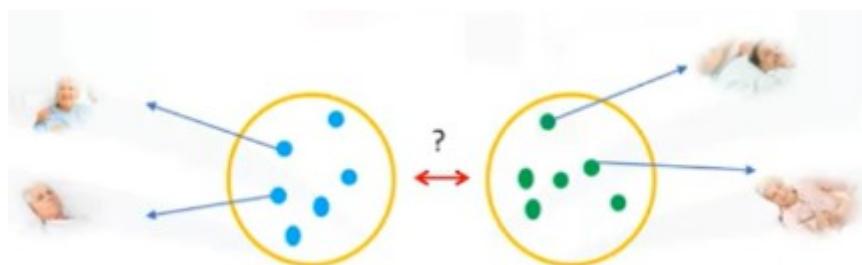


Figura. ¿Cómo podemos calcular la distancia entre clústeres cuando hay múltiples pacientes en cada grupo?

Podemos usar diferentes criterios para encontrar los clústeres más cercanos y fusionarlos.

En general, depende de:

- El tipo de datos.
- La dimensionalidad de los datos.
- El conocimiento del dominio del conjunto de datos.

Diferentes enfoques para definir la distancia entre clústeres distinguen los diferentes algoritmos.

Hay varias maneras en que podemos hacer esto.

- Agrupación de enlaces únicos.
 - El enlace único se define como la distancia más corta entre dos puntos en cada clúster.
- Agrupación completa de enlaces.
 - Buscamos la distancia más larga entre los puntos de cada clúster.
- Agrupamiento de enlaces promedio.
 - Buscamos la distancia promedio de cada punto de un clúster a cada punto de otro clúster.
- Agrupación de enlaces centroides.
 - El centroide es el promedio de los conjuntos de entidades de puntos de un clúster. Este enlace tiene en cuenta el centroide de cada clúster al determinar la distancia mínima.

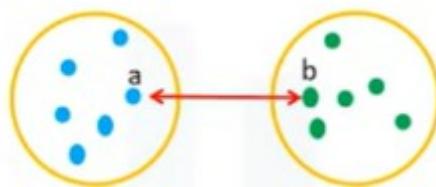


Figura. Agrupación de enlaces únicos.

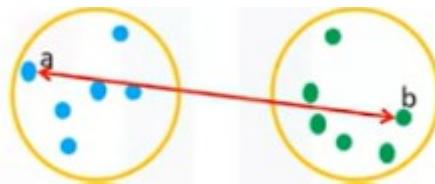


Figura. Agrupación completa de enlaces.

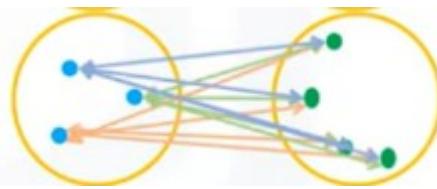


Figura. Agrupamiento de enlaces promedio.

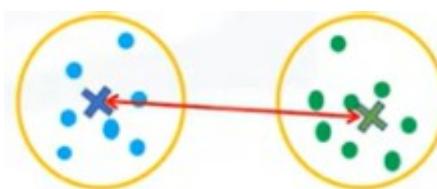


Figura. Agrupación de enlaces centroide.

El uso de clústeres jerárquicos tiene tres ventajas principales:

- No es necesario especificar el número de clústeres necesarios para el algoritmo.
- Es fácil de implementar.
- El dendrograma producido es muy útil para entender los datos.

El uso de clústeres jerárquicos tiene las siguientes desventajas:

- El algoritmo nunca puede deshacer ningún paso anterior
 - Si por ejemplo se agrupan 2 puntos y luego vemos que el enlace no era bueno no es posible deshacer ese paso.
- Puede demorar mucho en comparación con K-means.
- Si tenemos un gran conjunto de datos, puede resultar difícil determinar el número correcto de clústeres mediante el dendrograma.

Comparemos ahora el agrupamiento jerárquico con K-means.

K-MEANS	CLUSTERING JERÁRQUICO
Es más eficiente para grandes conjuntos de datos.	Puede demorar para datasets grandes.

Requiere se especifique el número de clusteres.	No requiere que se especifique el número de clústeres.
Brinda solamente una partición de los datos basado en el número predefinido de clusters.	Da más de una partición dependiendo de la resolución.
Devuelve potencialmente diferentes clusters cada vez debido a la inicialización aleatoria de los centroides.	Siempre genera los mismos clusters.

4.6. DBSCAN

Vamos a cubrir el escaneo de base de datos

Puntualmente el algoritmo de agrupamiento basado en densidad, que es apropiado para usar al examinar datos espaciales.

La mayoría de las técnicas tradicionales de clustering como K-Means, jerárquica y agrupación difusa se pueden utilizar para agrupar datos de una manera no supervisada, sin embargo, cuando se aplican a tareas con clústeres de forma arbitraria o clústeres dentro de clústeres, es posible que no puedan lograr buenos resultados: es decir, los elementos del mismo clúster no comparten suficiente similitud o el rendimiento puede ser deficiente.

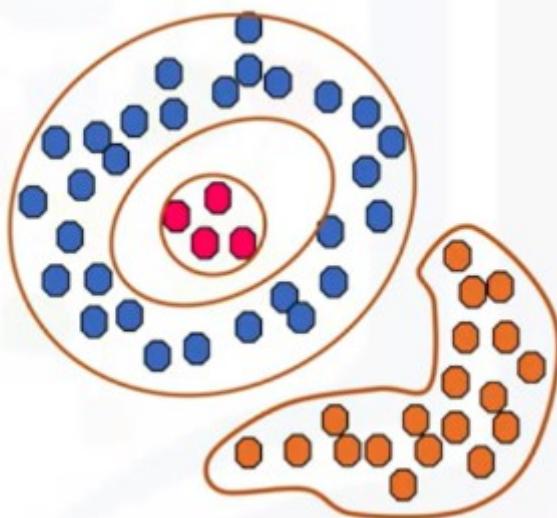


Figura. Clusters de forma arbitraria.

Además, aunque los algoritmos basados en particiones como K-Means pueden ser fáciles de entender e implementar en la práctica, el algoritmo no tiene noción de valores atípicos, es decir, todos los puntos se asignan a un clúster incluso si no pertenecen a ninguno. En el dominio de detección de anomalías, esto causa problemas ya que los puntos anómalos se asignarán al mismo clúster que los puntos de datos normales.

La agrupación basada en densidad localiza regiones de alta densidad separadas entre sí y regiones de baja densidad. La **densidad** en este contexto se define como el número de puntos dentro de un radio especificado. Un tipo específico y muy popular de clustering basado en densidad es DBSCAN. **DBSCAN** es particularmente eficaz para tareas como la identificación de clases en un contexto espacial.

Los maravillosos atributos del algoritmo DBSCAN es que puede encontrar cualquier clúster de forma arbitraria sin verse afectado por el ruido.

Por ejemplo, el mapa de la figura siguiente muestra la ubicación de las estaciones meteorológicas en Canadá. DBSCAN se puede utilizar aquí para encontrar el grupo de estaciones que muestran las mismas condiciones climáticas. Como puede ver, no solo encuentra diferentes clústeres de forma arbitraria, sino que puede encontrar la parte más densa de las muestras centradas en datos ignorando áreas o ruidos menos densos.

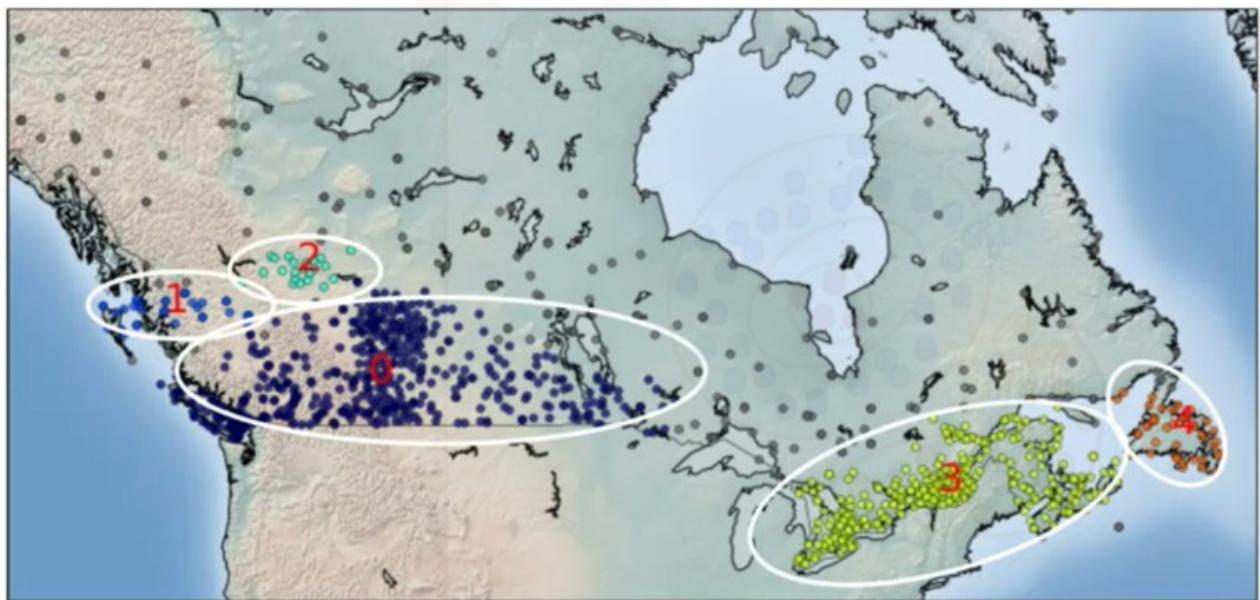


Figura. Estaciones meteorológicas en Canadá.

DBSCAN:

- Significa Clustering espacial basado en densidad de aplicaciones con ruido.
- Es uno de los algoritmos de clustering más comunes que funciona en función de la densidad del objeto.
- Trabaja con la idea de que si un punto en particular pertenece a un clúster, debería estar cerca de muchos otros puntos en ese clúster.
- Funciona en base a dos parámetros:
 - Radio R
 - Determina un radio especificado que si incluye suficientes puntos dentro de él, lo llamamos un área densa.
 - Puntos mínimos M

- Determina el número mínimo de puntos de datos que queremos en una vecindad para definir un clúster.

En DBSCAN **cada punto de nuestro conjunto de datos puede ser del tipo:**

- **Núcleo o central**
 - Un punto de datos es un punto de núcleo si dentro de nuestra vecindad del punto hay al menos M puntos.
- **Borde**
 - Un punto de datos es un punto de frontera si su vecindad contiene menos de M puntos de datos o si es accesible desde algún punto central. Aquí, la accesibilidad significa que está a nuestra distancia desde un punto central.
- **Outlier (punto atípico)**
 - Un valor atípico es un punto que no es un punto central y tampoco está lo suficientemente cerca como para ser accesible desde un punto central.

La idea detrás del algoritmo DBSCAN es visitar cada punto y encontrar su tipo primero, para luego agrupar puntos como clústeres basados en estos tipos.

Así, visitamos todos los puntos del conjunto de datos y los etiquetamos como núcleo, borde o valor atípico. El siguiente paso es conectar los puntos centrales que son vecinos y colocarlos en el mismo clúster. Por lo tanto, un clúster se forma como al menos un punto central más todos los puntos centrales accesibles más todos sus bordes.

Repasemos, DBSCAN:

- Puede encontrar clústeres de forma arbitraria. Incluso puede encontrar un clúster completamente rodeado por un clúster diferente.
- Tiene una noción de ruido y es robusto para los valores atípicos.
- Es muy práctico para su uso en muchos problemas del mundo real porque no requiere que se especifique el número de clústeres como K en K-means.

PARTE 5. SISTEMAS RECOMENDADORES

5.1. INTRODUCCIÓN

Veremos una rápida introducción a los sistemas recomendadores.

Aunque los gustos de las personas pueden variar, generalmente siguen patrones. Es decir, a la gente tienden a gustar cosas en la misma categoría o cosas que comparten las mismas características. Por ejemplo, si recientemente ha comprado un libro sobre Machine Learning en Python y ha disfrutado leerlo, es muy probable que también disfrute leyendo un libro sobre Visualización de datos. La gente también tiende a tener gustos similares a los de las personas a las que están cerca en sus vidas.

Los sistemas de recomendación intentan capturar estos patrones y comportamientos similares, para ayudar a predecir qué más le gustaría.

Los sistemas de recomendación suelen utilizarse en muchos sitios web. Por ejemplo, sugerir libros en Amazon y películas en Netflix. De hecho, todo lo que hay en el sitio web de Netflix está impulsado por la selección de los clientes. Si una película determinada se ve con la suficiente frecuencia, el sistema de recomendación de Netflix garantiza que esa película reciba un número cada vez mayor de recomendaciones.

Otro ejemplo se puede encontrar en una aplicación móvil de uso diario, donde se usa un motor de recomendación para recomendar cualquier cosa, desde dónde comer o a qué trabajo aplicar.

En las redes sociales, sitios como Facebook o LinkedIn, regularmente recomiendan amistades.

Los sistemas de recomendación se utilizan incluso para personalizar su experiencia en la web. Por ejemplo, cuando vaya a un sitio web de una plataforma de noticias, un sistema de recomendación tomará nota de los tipos de historias en las que hizo clic y hará recomendaciones sobre qué tipos de historias podría estar interesado en leer en el futuro. Una de las principales ventajas de utilizar sistemas de recomendación es que los usuarios obtienen una mayor exposición a muchos productos diferentes en los que podrían estar interesados. Esta exposición anima a los usuarios a un uso continuo o a la compra de su producto. Esto no sólo proporciona una mejor experiencia para el usuario, sino que también beneficia al proveedor de servicios, con mayores ingresos potenciales y una mejor seguridad para sus clientes.

Generalmente hay dos **tipos principales de sistemas de recomendación:**

- **Filtrado basado en contenido.**

- El paradigma principal de un sistema de recomendaciones basado en contenido está impulsado por la declaración: «Muéstrame más de lo que me ha gustado antes».
- Los sistemas basados en contenido intentan averiguar cuáles son los aspectos favoritos de un elemento de un usuario y, a continuación, hacer recomendaciones sobre los elementos que comparten esos aspectos.
- **Filtrado colaborativo.**
 - Se basa en un usuario que dice: «Dime lo que es popular entre mis vecinos porque a mí también me gusta».
 - Las técnicas de filtrado colaborativo encuentran grupos de usuarios similares y proporcionan recomendaciones basadas en gustos similares dentro de ese grupo. En resumen, se supone que un usuario podría estar interesado en lo que los usuarios similares están interesados.

La principal diferencia entre cada uno, puede así resumirse por el tipo de declaración que un consumidor podría hacer.

Además, hay sistemas de recomendación híbridos, que combinan varios mecanismos.

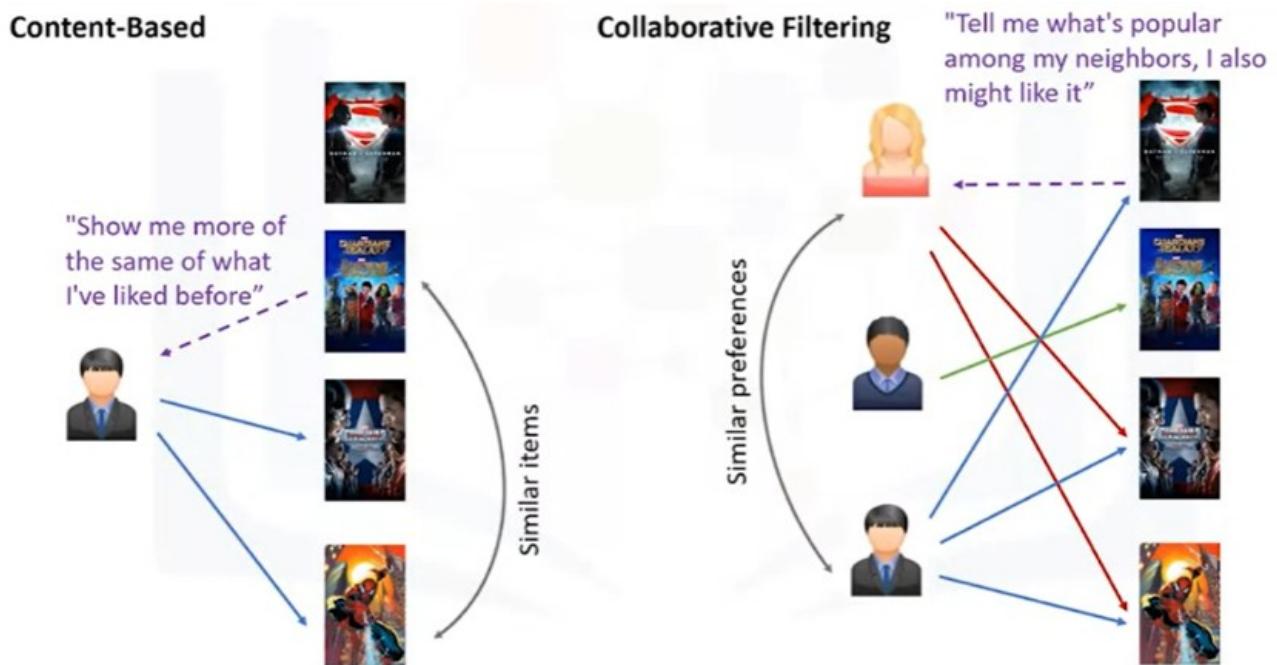


Figura. Tipos de sistemas de recomendación.

En cuanto a la **implementación de sistemas de recomendación, hay dos tipos**:

- **Basados en memoria.**
 - Utilizamos todo el conjunto de datos de usuario de elementos para generar un sistema de recomendaciones.
 - Utiliza técnicas estadísticas.
 - Ejemplos de estas técnicas incluyen: Correlación de Pearson, Similaridad de Cosenos y Distancia Euclidiana.
- **Basados en modelos.**
 - Utiliza técnicas estadísticas para aproximar usuarios o elementos.
 - Se desarrolla un modelo de usuarios en un intento de aprender sus preferencias. Los modelos se pueden crear utilizando técnicas de aprendizaje automático como regresión, agrupación, clasificación, etc.

5.2. SISTEMAS RECOMENDADORES BASADOS EN CONTENIDO

Un sistema de recomendaciones basado en contenido intenta recomendar elementos a los usuarios en función de su perfil.

El perfil del usuario:

- Gira en torno a las preferencias y gustos del mismo.
- Se forma en función de las calificaciones del mismo, incluyendo el número de veces que ha hecho clics en diferentes elementos o de si indicó el que le hayan gustado.

El proceso de recomendación se basa en la similitud entre esos temas.

La similitud o cercanía de los artículos se mide en función de la similitud en el contenido de los mismos.

Cuando decimos contenido estamos hablando de cosas como:

- Categoría de elementos.
- Etiqueta
- Genéro.
- Otras.

Por ejemplo, si tenemos cuatro películas, y si al usuario le gusta o califica los dos primeros elementos, y si el Item 3 es similar al Item 1, el motor recomendará el Item 3 al usuario. En esencia, esto es lo que hacen los motores del sistema de recomendación basados en contenido.

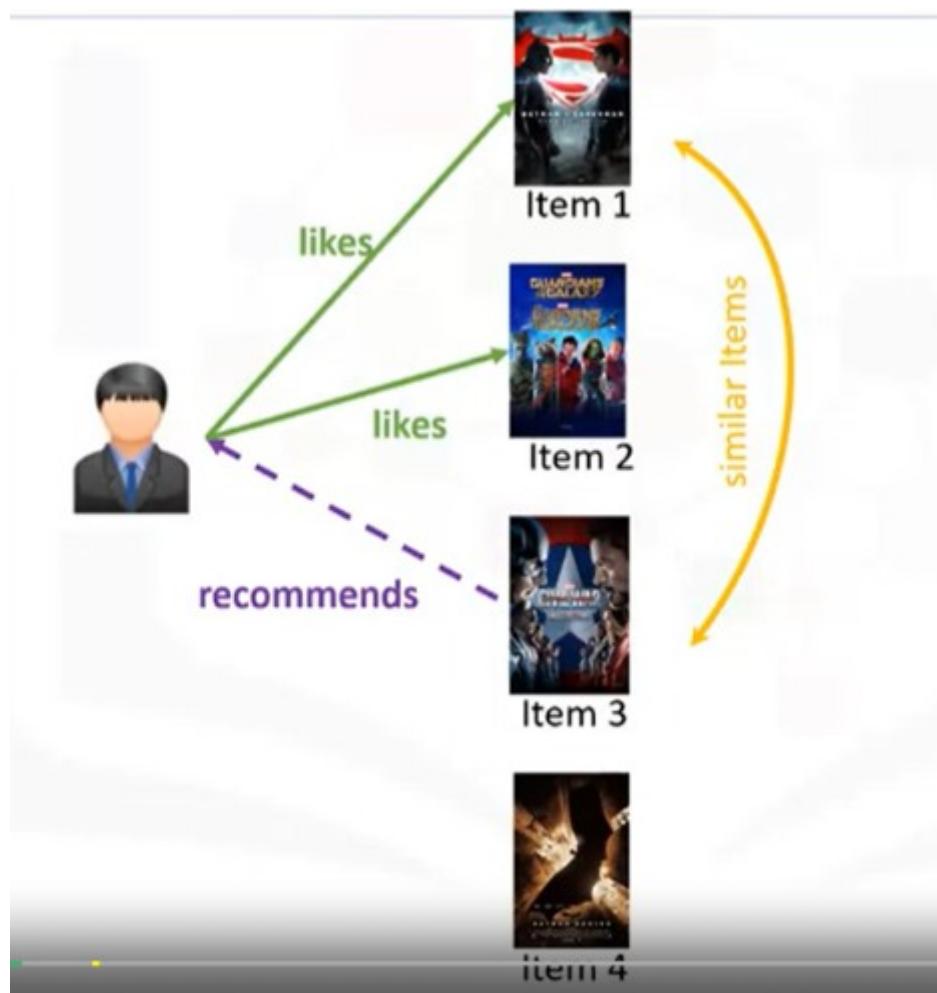


Figura. Sistema de recomendación basado en contenido.

Ahora, vamos a sumergirnos en un sistema de recomendación basado en contenido para ver cómo funciona.

Supongamos que tenemos un conjunto de datos de sólo seis películas.

Este conjunto de datos muestra las películas que nuestro usuario ha visto y también el género de cada una de las películas. Por ejemplo, Batman contra Superman está en el género Aventura, Super Hero y Guardianes de la Galaxia está en los géneros Comedia, Aventura, Super Héroe y Ciencia Ficción.

Supongamos que el usuario ha visto y clasificado tres películas hasta el momento y ha dado una calificación de dos de cada 10 a la primera película, 10 de 10 a la segunda película y ocho de 10 a la tercera.

La tarea del motor de recomendación es recomendar una de las tres películas candidatas a este usuario.

Para lograr esto, tenemos que construir el perfil de usuario.

- Primero, creamos un vector para mostrar las calificaciones de los usuarios para las películas que ya ha visto. Lo llamamos Input User Ratings.
- Luego, codificamos las películas a través del enfoque de codificación one-hot. Género de películas se utilizan aquí como un conjunto de características. Utilizamos las tres primeras películas para hacer esta matriz, que representa la matriz del conjunto de características de la película.
- Si multiplicamos estas dos matrices podemos obtener el conjunto de funciones ponderadas para las películas.
 - Esta matriz también se llama matriz de género ponderado y representa los intereses del usuario para cada género basándose en las películas que ha visto.
- Ahora, dada la Matriz de Género ponderada, podemos dar forma al perfil de nuestro usuario activo.
 - Esencialmente, podemos agregar los géneros ponderados y luego normalizarlos para encontrar el perfil de usuario.

En este ejemplo, que se muestra en la figura siguiente, puede verse que al usuario le gustan las películas de superhéroes más que de otros géneros.

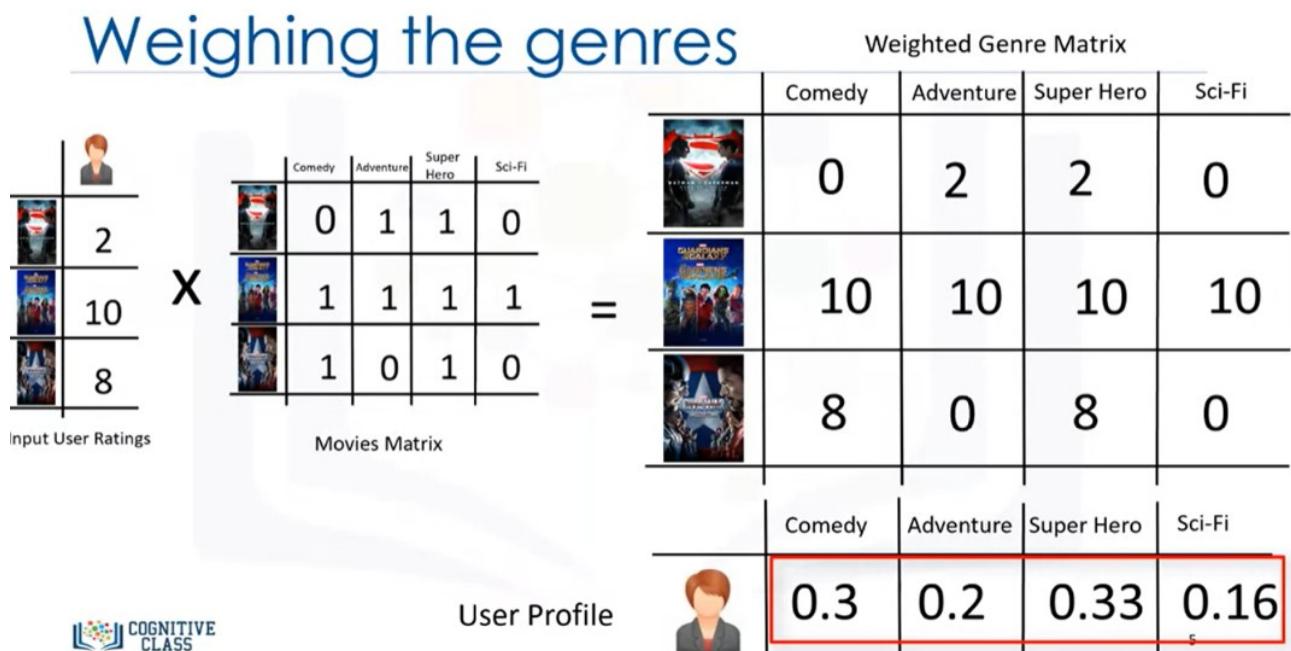


Figura. Creación del perfil de usuario.

Utilizamos este perfil para averiguar qué película es adecuada para recomendar a este usuario.

Recordemos que también teníamos tres películas candidatas para recomendación que no han sido vistas por el usuario, también las codificamos.

Ahora estamos en la posición en la que tenemos que averiguar cuál de ellos es el más adecuado para ser recomendado al usuario. Para ello, simplemente multiplicamos la matriz de perfil de usuario por la matriz de película candidata, lo que resulta en la matriz de películas ponderadas, que muestra el peso de cada género respecto al perfil de usuario. Si agregamos estas calificaciones ponderadas, obtenemos el posible nivel de interés del usuario activo en estas tres películas.

En esencia, son nuestras listas de recomendaciones, que podemos ordenar para clasificar las películas y recomendarlas al usuario. Por ejemplo, podemos decir que la Guía del Autoestopista para la Galaxia tiene la puntuación más alta en nuestra lista, y es apropiado recomendar al usuario. Ahora, puede volver y llenar las calificaciones previstas para el usuario.





Figura. Encontrando la recomendación.

Por lo tanto, para recapitular lo que hemos discutido hasta ahora, la recomendación en un sistema basado en contenido se basa en el gusto del usuario y el contenido o elementos del conjunto de características.

Tal modelo es muy eficiente, sin embargo, en algunos casos, no funciona. Por ejemplo, supongamos que tenemos una película en el género dramático, que el usuario nunca ha visto, entonces, este género no estaría en su perfil. Por lo tanto, sólo obtendrá recomendaciones relacionadas con géneros que ya están en su perfil y el motor de recomendación puede nunca recomendar ninguna película dentro de otros géneros. Este problema se puede resolver con otros tipos de sistemas de recomendación, como el filtrado colaborativo.

5.3. FILTROS COLABORATIVOS

El filtrado colaborativo se basa en el hecho de que existen relaciones entre los productos y los intereses de las personas.

Muchos sistemas de recomendación utilizan el filtrado colaborativo para encontrar estas relaciones y dar una recomendación precisa de un producto que al usuario le pueda interesar.

El filtrado colaborativo tiene básicamente dos enfoques:

- **Basado en usuarios.**
 - Se basa en la similitud o vecindad del usuario.
- **Basado en elementos.**
 - Se basa en la similitud entre los elementos.

Veamos primero la intuición detrás del **enfoque basado en el usuario**.

En él tenemos un usuario activo al que se dirige la recomendación.

El motor de filtrado colaborativo primero busca usuarios similares, es decir, aquellos que comparten los patrones de clasificación.

El filtrado colaborativo basa esta similitud en cosas como:

- El historial
- Las preferencias
- Las opciones que los usuarios toman al comprar, ver o disfrutar algo.

Un ejemplo serían las películas que usuarios similares han calificado altamente.

Luego utiliza las calificaciones de estos usuarios similares para predecir las posibles calificaciones del usuario activo para una película que no ha visto previamente. Por ejemplo, si 2 usuarios son similares o son vecinos en términos de sus películas interesadas, podemos recomendar una película al usuario activo que su vecino ya haya visto.

- User-based collaborative filtering



Figura. Filtro basado en usuario.

Ahora vamos a sumergirnos en el algoritmo.

Supongamos que:

- Tenemos una matriz de elementos de usuario simple, que muestra las calificaciones de 4 usuarios para 5 películas diferentes.
- Nuestro usuario activo ha visto y calificado 3 de estas 5 películas.

Averiguemos cuál de las 2 películas que nuestro usuario no ha visto le debería ser recomendada.

El primer paso es descubrir cuán similar es el usuario activo a los demás usuarios.

¿Cómo hacemos esto?

Se puede hacer con varias técnicas estadísticas y vectoriales diferentes, tales como mediciones de distancia o similitud, incluyendo distancia euclídea, correlación de Pearson y similitud de coseno.

Para calcular el nivel de similitud entre dos usuarios, utilizamos las tres películas que ambos usuarios han valorado en el pasado.

Independientemente de lo que utilicemos para la medición de similitud, digamos, por ejemplo, que la similitud podría ser 0.7, 0.9 y 0.4 entre el usuario activo y otros usuarios.

Estos números representan pesos de similitud o proximidad del usuario activo a otros usuarios del conjunto de datos.

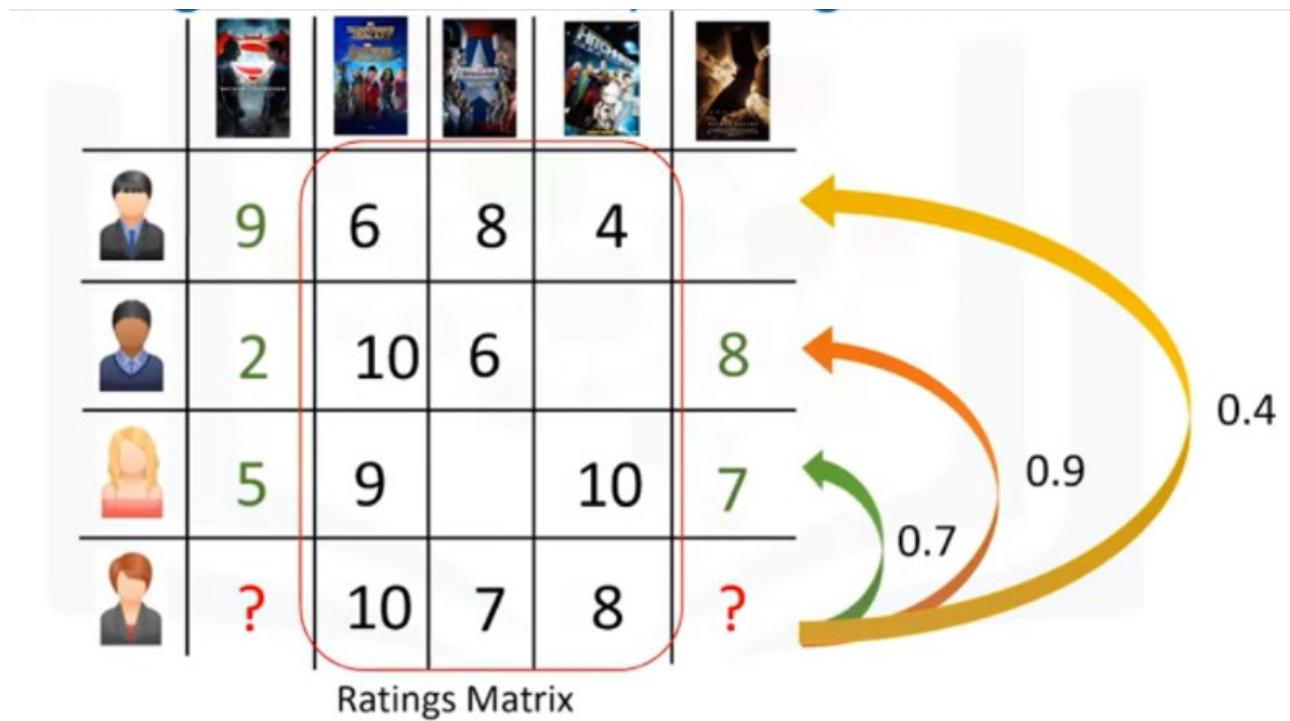


Figura. Ejemplo.

El siguiente paso es crear una matriz de clasificación ponderada.

Acabamos de calcular la similitud de los usuarios con nuestro usuario activo en la diapositiva anterior, ahora, podemos usarlo para calcular la posible opinión del usuario activo sobre nuestras dos películas objetivo, lo que se logra multiplicando los pesos de similitud con las calificaciones de los usuarios.

Se traduce en una matriz de calificaciones ponderadas, que representa la opinión de los vecinos del usuario sobre son dos películas candidatas para la recomendación. De hecho, incorpora el comportamiento de otros usuarios y da más peso a las calificaciones de aquellos usuarios que son más similares al usuario activo.

Ahora, podemos generar la matriz de recomendaciones agregando todas las tasas ponderadas.

Como tres usuarios calificaron la primera película potencial y dos usuarios calificaron la segunda, tenemos que normalizar los valores de clasificación ponderados, hacemos esto dividiéndolo por la suma del índice de similitud para los usuarios. El resultado es la

calificación potencial que nuestro usuario activo dará a estas películas en función de su similitud con otros usuarios.

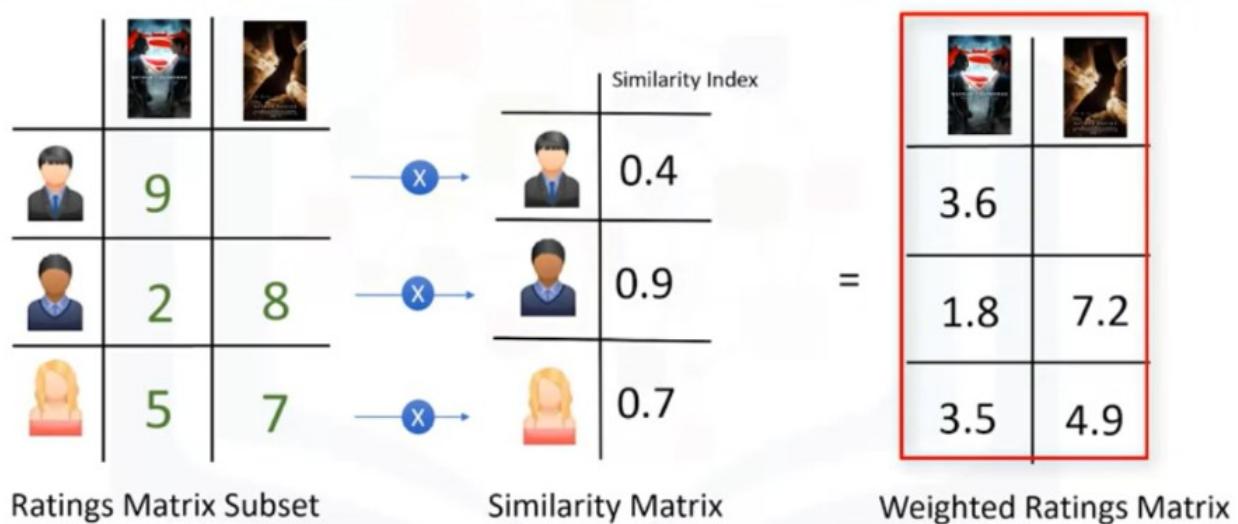


Figura. Generando la matriz de ratings ponderada.

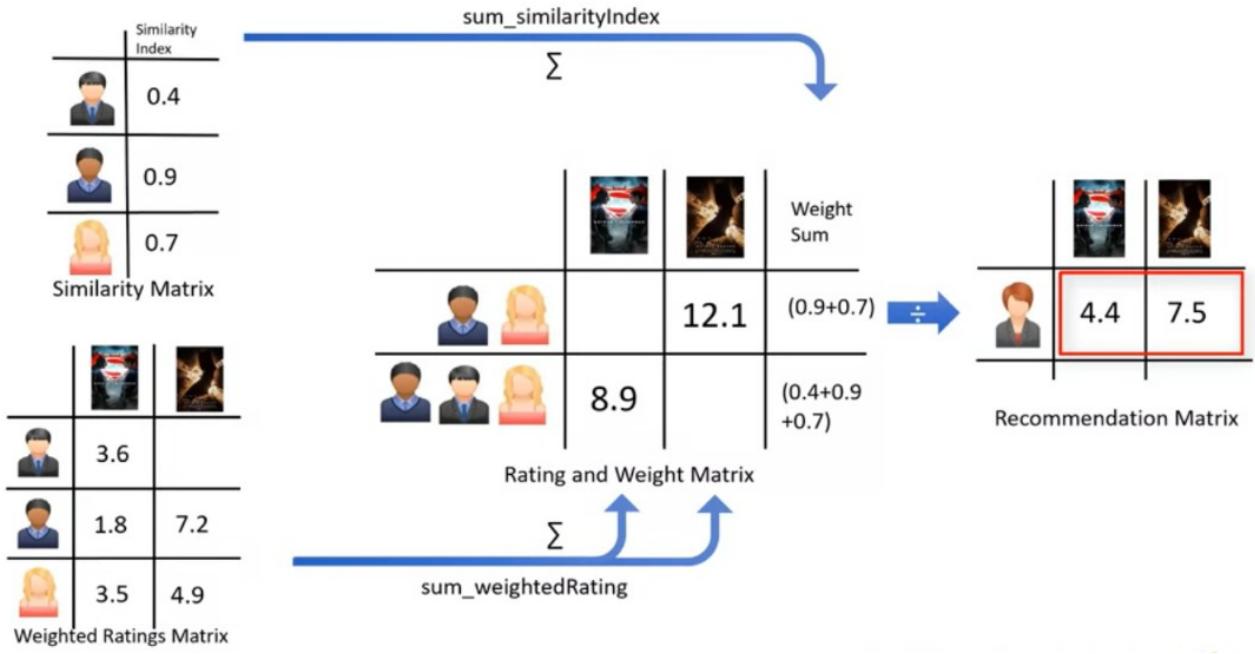


Figura. Generando la matriz de recomendaciones.

Podemos usar esto para clasificar las películas para proporcionar recomendaciones a nuestro usuario activo.

Ahora, vamos a examinar qué es diferente entre el filtrado colaborativo basado en usuarios y basado en elementos.

En el enfoque basado en el usuario:

- La recomendación se basa en usuarios con los que comparte preferencias.
 - Por ejemplo, como al Usuario 1 y al Usuario 3 les gustaba el Item 3 y el Item 4, los consideramos como usuarios similares o vecinos, y recomendamos el Item 1, que es positivamente clasificado por el Usuario 1 al Usuario 3 (figura siguiente).
- En el enfoque basado en elementos:
 - Elementos similares construyen vecindarios sobre el comportamiento de los usuarios.
 - No se basa en su contenido.
 - Por ejemplo, el ítem 1 y el ítem 3 se consideran vecinos, ya que fueron calificados positivamente por el usuario 1 y el usuario 2. Por lo tanto, el ítem 1 puede ser recomendado al usuario 3, ya que ya ha mostrado interés en el ítem 3 (figura siguiente).

- Las recomendaciones aquí se basan en los elementos de la vecindad que un usuario podría preferir.

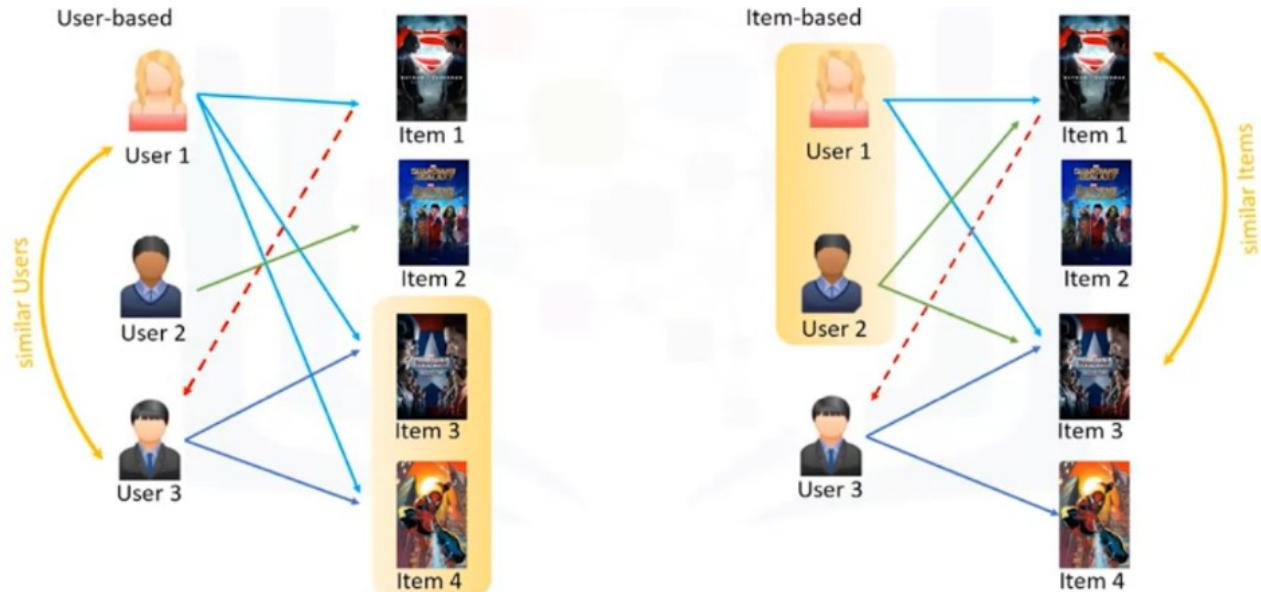


Figura. Filtrado colaborativo basados en usuarios vs basado en ítems.

El filtrado colaborativo es un sistema de recomendaciones muy eficaz, sin embargo, también hay algunos **desafíos** con él:

- Escasez de datos.
 - Ocurre cuando se tiene un gran conjunto de datos de usuarios que generalmente califican sólo un número limitado de elementos.
 - Como se mencionó, los recomendadores basados en colaboración solo pueden predecir la puntuación de un elemento si hay otros usuarios que lo han calificado. Debido a la escasez, es posible que no tengamos suficientes clasificaciones en el conjunto de datos de elementos de usuario, lo que hace imposible proporcionar recomendaciones adecuadas.
- Inicio en frío (Cold Start)
 - Se refiere a la dificultad que tiene el sistema de recomendaciones cuando hay un nuevo usuario, y como tal no existe todavía un perfil para ellos.
 - También puede ocurrir cuando tenemos un nuevo artículo que no ha recibido una calificación.

- Escalabilidad.
 - A medida que aumenta el número de usuarios o elementos y la cantidad de datos se expande, los algoritmos de filtrado colaborativo comenzarán a sufrir caídas en el rendimiento, simplemente debido al crecimiento y al cálculo de similitud

Existen algunas soluciones para cada uno de estos desafíos, como el uso de sistemas de recomendación basados en híbridos, pero están fuera del alcance de este curso.

ANEXO. CÓDIGOS

1. REGRESIÓN

1.1. Regresión Lineal Simple

```
import matplotlib.pyplot as plt
import pandas as pd
import pylab as pl
import numpy as np
import urllib.request
from sklearn import linear_model
from sklearn.metrics import r2_score
%matplotlib inline

# Descargo los datos
url = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDriverSkillsNetwork-ML0101EN-SkillsNetwork/labs/Module%202/data/FuelConsumptionCo2.csv'
filename = 'FuelConsumption.csv'
urllib.request.urlretrieve(url, filename)

# Lo leo en un dataframe
df = pd.read_csv("FuelConsumption.csv")

# Divido en conjuntos de entrenamiento y test
msk = np.random.rand(len(df)) < 0.8
train = cdf[msk]
test = cdf[~msk]

# Modelo
regr = linear_model.LinearRegression()
# ajuste
train_x = np.asarray(train[['ENGINESIZE']])
train_y = np.asarray(train[['CO2EMISSIONS']])
regr.fit (train_x, train_y)
# Valores de los coeficientes
print ('Coefficients: ', regr.coef_)
print ('Intercept: ',regr.intercept_)

# Graficamos (gráfico de dispersión junto con recta de regresión)
```

```
plt.scatter(train.ENGINESIZE, train.CO2EMISSIONS, color='blue')
plt.plot(train_x, regr.coef_[0][0]*train_x + regr.intercept_[0], '-r')
plt.xlabel("Engine size")
plt.ylabel("Emission")

# Evaluación del modelo
test_x = np.asarray(test[['ENGINESIZE']])
test_y = np.asarray(test[['CO2EMISSIONS']])
test_y_ = regr.predict(test_x)
print("Mean absolute error: %.2f" % np.mean(np.absolute(test_y_ - test_y)))
print("Residual sum of squares (MSE): %.2f" % np.mean((test_y_ - test_y) ** 2))
print("R2-score: %.2f" % r2_score(test_y , test_y_) )
```

1.2. Regresión Polinómica

```
import matplotlib.pyplot as plt
import pandas as pd
import pylab as pl
import numpy as np
import urllib.request
from sklearn.preprocessing import PolynomialFeatures
from sklearn import linear_model
from sklearn.metrics import r2_score
%matplotlib inline

# Descargo los datos
url = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDriverSkillsNetwork-ML0101EN-SkillsNetwork/labs/Module%202/data/FuelConsumptionCo2.csv'
filename = 'FuelConsumption.csv'
urllib.request.urlretrieve(url, filename)

# Leo los datos en un dataframe
df = pd.read_csv("FuelConsumption.csv")

# Seleccionamos algunas características
cdf = df[['ENGINESIZE','CYLINDERS','FUELCONSUMPTION_COMB','CO2EMISSIONS']]

# Divido en conjuntos de test y de entrenamiento
msk = np.random.rand(len(df)) < 0.8
train = cdf[msk]
test = cdf[~msk]

#
train_x = np.asarray(train[['ENGINESIZE']])
train_y = np.asarray(train[['CO2EMISSIONS']])
test_x = np.asarray(test[['ENGINESIZE']])
test_y = np.asarray(test[['CO2EMISSIONS']])

# Uso de Polynomial Features y fit transform
poly = PolynomialFeatures(degree=2)
```

```

train_x_poly = poly.fit_transform(train_x)

# Ahora podemos aplicar regresión lineal
clf = linear_model.LinearRegression()
train_y_ = clf.fit(train_x_poly, train_y)

# Los coeficientes
print ('Coefficients: ', clf.coef_)
print ('Intercept: ',clf.intercept_)

# Graficamos
plt.scatter(train.ENGINESIZE, train.CO2EMISSIONS, color='blue')
XX = np.arange(0.0, 10.0, 0.1)
yy = clf.intercept_[0]+ clf.coef_[0][1]*XX+ clf.coef_[0][2]*np.power(XX, 2)
plt.plot(XX, yy, '-r' )
plt.xlabel("Engine size")
plt.ylabel("Emission")

# Evaluamos
test_x_poly = poly.fit_transform(test_x)
test_y_ = clf.predict(test_x_poly)
print("Mean absolute error: %.2f" % np.mean(np.absolute(test_y_ - test_y)))
print("Residual sum of squares (MSE): %.2f" % np.mean((test_y_ - test_y) ** 2))
print("R2-score: %.2f" % r2_score(test_y_ , test_y) )

```

1.3. REGRESIÓN NO LINEAL

```
import numpy as np
import matplotlib.pyplot as plt
import urllib.request
from scipy.optimize import curve_fit
%matplotlib inline

# Polinomio de grado 3
x = np.arange(-5.0, 5.0, 0.1)
y = 1*(x**3) + 1*(x**2) + 1*x + 3
y_noise = 20 * np.random.normal(size=x.size)
ydata = y + y_noise
plt.plot(x, ydata, 'bo')
plt.plot(x,y, 'r')
plt.ylabel('Dependent Variable')
plt.xlabel('Independent Variable')
plt.show()

# Función cuadrática
x = np.arange(-5.0, 5.0, 0.1)
y = np.power(x,2)
y_noise = 2 * np.random.normal(size=x.size)
ydata = y + y_noise
plt.plot(x, ydata, 'bo')
plt.plot(x,y, 'r')
plt.ylabel('Dependent Variable')
plt.xlabel('Independent Variable')
plt.show()

# Función exponencial
X = np.arange(-5.0, 5.0, 0.1)
Y= np.exp(X)
plt.plot(X,Y)
plt.ylabel('Dependent Variable')
plt.xlabel('Independent Variable')
plt.show()
```

```
# Función logarítmica
X = np.arange(-5.0, 5.0, 0.1)
Y = np.log(X)
plt.plot(X,Y)
plt.ylabel('Dependent Variable')
plt.xlabel('Independent Variable')
plt.show()
```

```
# Sigmoidal
X = np.arange(-5.0, 5.0, 0.1)
Y = 1/(1+np.power(3, X-2))
plt.plot(X,Y)
plt.ylabel('Dependent Variable')
plt.xlabel('Independent Variable')
plt.show()
```

"" Ejemplo de reg. no lineal """

Descargamos los datos

```
url = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-ML0101EN-SkillsNetwork/labs/Module%202/data/china_gdp.csv'
filename = 'china_gdp.csv'
urllib.request.urlretrieve(url, filename)
```

Leemos los datos en un dataframe

```
df = pd.read_csv("china_gdp.csv")
```

Graficamos

```
plt.figure(figsize=(8,5))
x_data, y_data = (df["Year"].values, df["Value"].values)
plt.plot(x_data, y_data, 'ro')
plt.ylabel('GDP')
plt.xlabel('Year')
plt.show()
```

```

# Construimos el modelo (usaremos la sigmoide en este caso)

def sigmoid(x, Beta_1, Beta_2):
    y = 1 / (1 + np.exp(-Beta_1 * (x - Beta_2)))
    return y

# Parámetros iniciales
beta_1 = 0.10
beta_2 = 1990.0

# Función logística
Y_pred = sigmoid(x_data, beta_1, beta_2)

# Normalizamos
xdata = x_data/max(x_data)
ydata = y_data/max(y_data)

# Obtenemos los mejores parámetros
popt, pcov = curve_fit(sigmoid, xdata, ydata)

# Imprimimos los parámetros
print(" beta_1 = %f, beta_2 = %f" % (popt[0], popt[1]))

# Graficamos
x = np.linspace(1960, 2015, 55)
x = x/max(x)
plt.figure(figsize=(8,5))
y = sigmoid(x, *popt)
plt.plot(xdata, ydata, 'ro', label='data')
plt.plot(x, y, linewidth=3.0, label='fit')
plt.legend(loc='best')
plt.ylabel('GDP')
plt.xlabel('Year')
plt.show()

```

2. CLASIFICACIÓN

2.1. K-NN

```
# Importo las librerías necesarias
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import urllib.request
from sklearn import preprocessing
from sklearn.neighbors import KNeighborsClassifier
%matplotlib inline

url = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-ML0101EN-SkillsNetwork/labs/Module%203/data/teleCust1000t.csv'
filename = 'teleCust1000t.csv'
urllib.request.urlretrieve(url, filename)

# Normalización
X = preprocessing.StandardScaler().fit(X).transform(X.astype(float))

# Divido en conjuntos de entrenamiento y test
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split( X, y, test_size=0.2, random_state=4)
# Chequeo las dimensiones
print ('Train set:', X_train.shape, y_train.shape)
print ('Test set:', X_test.shape, y_test.shape)

# Pruebo con varios valores para obtener el mejor valor de K (al menos en ese rango)
Ks = 10
mean_acc = np.zeros((Ks-1))
std_acc = np.zeros((Ks-1))
for n in range(1,Ks):
    # Entrenamiento y predicción
    neigh = KNeighborsClassifier(n_neighbors = n).fit(X_train,y_train)
    yhat=neigh.predict(X_test)
    mean_acc[n-1] = metrics.accuracy_score(y_test, yhat)
    std_acc[n-1]=np.std(yhat==y_test)/np.sqrt(yhat.shape[0])
```

```

# Obtengo el mejor valor de k junto con su índice; recuerdo en numpy los índices comienzan en 0
maxValue = np.amax(mean_acc)
ind = np.where(mean_acc == maxValue)
indice = ind[0][0] + 1
Kmax = indice

# mejor modelo
neigh = KNeighborsClassifier(n_neighbors = Kmax).fit(X_trainK,y_trainK)
yhat = neigh.predict(X_test)

# métricas
print("Train set Accuracy: ", metrics.accuracy_score(y_train, neigh.predict(X_train)))
print("Test set Accuracy: ", metrics.accuracy_score(y_test, yhat))

# Grafico
plt.plot(range(1,Ks),mean_acc,'g')
plt.fill_between(range(1,Ks),mean_acc - 1 * std_acc,mean_acc + 1 * std_acc, alpha=0.10)
plt.fill_between(range(1,Ks),mean_acc - 3 * std_acc,mean_acc + 3 * std_acc, alpha=0.10,color="green")
plt.legend(['Accuracy ','+/- 1xstd','+/- 3xstd'])
plt.ylabel('Accuracy ')
plt.xlabel('Number of Neighbors (K)')
plt.tight_layout()
plt.show()

```

2.2. ARBOLES DE DECISION

```
# Importo las librerías necesarias
import pandas as pd
import urllib.request
from sklearn.tree import DecisionTreeClassifier
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn import metrics
from io import StringIO
import pydotplus
import matplotlib.image as mpimg
from sklearn import tree
%matplotlib inline

# Descargo los datos
url = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDriverSkillsNetwork-ML0101EN-SkillsNetwork/labs/Module%203/data/drug200.csv'
filename = 'drug200.csv'
urllib.request.urlretrieve(url, filename)

# Leo los datos en un CSV
my_data = pd.read_csv("drug200.csv", delimiter=",")

# Selecciono algunas características
X = my_data[['Age', 'Sex', 'BP', 'Cholesterol', 'Na_to_K']].values

# Transformo variables categóricas en numéricas
# Sexo
le_sex = preprocessing.LabelEncoder()
le_sex.fit(['F', 'M'])
X[:, 1] = le_sex.transform(X[:, 1])

# BP
le_BP = preprocessing.LabelEncoder()
le_BP.fit(['LOW', 'NORMAL', 'HIGH'])
X[:, 2] = le_BP.transform(X[:, 2])

# Etiquetas
y = my_data['Drug'].values
```

```

y = my_data["Drug"]

# Divido en conjuntos de entrenamiento y test
X_trainset, X_testset, y_trainset, y_testset = train_test_split(X, y, test_size=0.3, random_state=3)

# Creo la instancia
drugTree = DecisionTreeClassifier(criterion="entropy", max_depth = 4)

# Entreno el modelo
drugTree.fit(X_trainset,y_trainset)

# Predicción
predTree = drugTree.predict(X_testset)

# Evaluación
print("DecisionTrees's Accuracy: ", metrics.accuracy_score(y_testset, predTree))

# Grafico el árbol
dot_data = StringIO()
filename = "drugtree.png"
featureNames = my_data.columns[0:5]
targetNames = my_data["Drug"].unique().tolist()
out=tree.export_graphviz(drugTree,feature_names=featureNames, out_file=dot_data, class_names=
np.unique(y_trainset), filled=True, special_characters=True,rotate=False)
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png(filename)
img = mpimg.imread(filename)
plt.figure(figsize=(100, 200))
plt.imshow(img,interpolation='nearest')

```

2.3. SVM

```
import pandas as pd
import pylab as pl
import numpy as np
import urllib.request
import scipy.optimize as opt
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
%matplotlib inline
import matplotlib.pyplot as plt
from sklearn import svm
from sklearn.metrics import classification_report, confusion_matrix
import itertools
from sklearn.metrics import f1_score

# Descargo los datos
url = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDriverSkillsNetwork-ML0101EN-SkillsNetwork/labs/Module%203/data/cell_samples.csv'
filename = 'cell_samples.csv'
urllib.request.urlretrieve(url, filename)

# Leo los datos en un dataframe
cell_df = pd.read_csv("cell_samples.csv")

# Hay un tipo object, lo cambio a numérico (veo los tipos con dtypes)
cell_df = cell_df[pd.to_numeric(cell_df['BareNuc'], errors='coerce').notnull()]
cell_df['BareNuc'] = cell_df['BareNuc'].astype('int')
cell_df.dtypes

# Características
feature_df = cell_df[['Clump', 'UnifSize', 'UnifShape', 'MargAdh', 'SingEpiSize', 'BareNuc', 'BlandChrom', 'NormNucl', 'Mit']]
X = np.asarray(feature_df)

# Etiquetas
cell_df['Class'] = cell_df['Class'].astype('int')
```

```

y = np.asarray(cell_df['Class'])

# Divido en conjuntos de entrenamiento y test
X_train, X_test, y_train, y_test = train_test_split( X, y, test_size=0.2, random_state=4)

# Modelo
clf = svm.SVC(kernel='rbf')
clf.fit(X_train, y_train)

```

Predicción

```
yhat = clf.predict(X_test)
```

Evaluación

Matriz de confusión

```
def plot_confusion_matrix(cm, classes,
                         normalize=False,
                         title='Confusion matrix',
                         cmap=plt.cm.Blues):
```

"""

This function prints and plots the confusion matrix.

Normalization can be applied by setting `normalize=True`.

"""

```

if normalize:
    cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
    print("Normalized confusion matrix")
else:
    print('Confusion matrix, without normalization')

print(cm)
plt.imshow(cm, interpolation='nearest', cmap=cmap)
plt.title(title)
plt.colorbar()
tick_marks = np.arange(len(classes))
plt.xticks(tick_marks, classes, rotation=45)
plt.yticks(tick_marks, classes)
fmt = '.2f' if normalize else 'd'
thresh = cm.max() / 2.
```

```

for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
    plt.text(j, i, format(cm[i, j], fmt),
              horizontalalignment="center",
              color="white" if cm[i, j] > thresh else "black")
plt.tight_layout()
plt.ylabel('True label')
plt.xlabel('Predicted label')

# Compute confusion matrix
cnf_matrix = confusion_matrix(y_test, yhat, labels=[2,4])
np.set_printoptions(precision=2)
print(classification_report(y_test, yhat))

# Plot non-normalized confusion matrix
plt.figure()
plot_confusion_matrix(cnf_matrix, classes=['Benign(2)', 'Malignant(4)'], normalize=False, title='Confusion matrix')

# Métricas

# F1-score
f1_score(y_test, yhat, average='weighted')

```

2.4. REGRESIÓN LOGÍSTICA

```
import pandas as pd
import pylab as pl
import numpy as np
import scipy.optimize as opt
from sklearn import preprocessing
%matplotlib inline
import matplotlib.pyplot as plt
import urllib.request
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
from sklearn.metrics import jaccard_score
from sklearn.metrics import log_loss

# Descargamos los datos
url = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDriverSkillsNetwork-ML0101EN-SkillsNetwork/labs/Module%203/data/ChurnData.csv'
filename = 'ChurnData.csv'
urllib.request.urlretrieve(url, filename)

# Leemos los datos en un dataframe
churn_df = pd.read_csv("ChurnData.csv")

# Seleccionamos algunas características
churn_df = churn_df[['tenure', 'age', 'address', 'income', 'ed', 'employ', 'equip', 'callcard', 'wireless', 'churn']]

# Cambiamos a entero ya que lo requiere el algoritmo de scikit-learn en vez de yes o no será 0 o 1
churn_df['churn'] = churn_df['churn'].astype('int')

# Características
X = np.asarray(churn_df[['tenure', 'age', 'address', 'income', 'ed', 'employ', 'equip']])
# Etiquetas
y = np.asarray(churn_df['churn'])
```

```

# Escalado
X = preprocessing.StandardScaler().fit(X).transform(X)

# Dividimos en conjuntos de entrenamiento y test
X_train, X_test, y_train, y_test = train_test_split( X, y, test_size=0.2, random_state=4)
print ('Train set:', X_train.shape, y_train.shape)
print ('Test set:', X_test.shape, y_test.shape)

# Modelo
LR = LogisticRegression(C=0.01, solver='liblinear').fit(X_train,y_train)

# Predicción
yhat = LR.predict(X_test)

# Estimados para todas las clases
yhat_prob = LR.predict_proba(X_test)

# Métricas

# Jaccard
jaccard_score(y_test, yhat)

# Log Loss
log_loss(y_test, yhat_prob)

```

3. CLUSTERING

3.1. K-MEANS

```
# Importamos las librerías necesarias
import random
import numpy as np
import matplotlib.pyplot as plt
import urllib.request
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.datasets.samples_generator import make_blobs
from mpl_toolkits.mplot3d import Axes3D
%matplotlib inline

# Descargo los datos
url = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDriverSkillsNetwork-ML0101EN-SkillsNetwork/labs/Module%204/data/Cust_Segmentation.csv'
filename = 'Cust_Segmentation.csv'
urllib.request.urlretrieve(url, filename)

# Leemos los datos en un dataframe
cust_df = pd.read_csv("Cust_Segmentation.csv")

# Eliminamos la variable Address que es categórica
df = cust_df.drop('Address', axis=1)

# Escalamos
X = df.values[:,1:]
X = np.nan_to_num(X)
Clus_dataSet = StandardScaler().fit_transform(X)
Clus_dataSet

# Modelo
clusterNum = 3
k_means = KMeans(init = "k-means++", n_clusters = clusterNum, n_init = 12)
k_means.fit(X)
labels = k_means.labels_
```

```

print(labels)

# Asignamos etiquetas a cada fila en el dataframe
df["Clus_km"] = labels

# Podemos chequear los valores de los centroides promediando las características en cada cluster.
df.groupby('Clus_km').mean()

# Distribución de clientes basado en edad e ingreso
area = np.pi * (X[:, 1])**2
plt.scatter(X[:, 0], X[:, 3], s=area, c=labels.astype(np.float), alpha=0.5)
plt.xlabel('Age', fontsize=18)
plt.ylabel('Income', fontsize=16)
plt.show()

fig = plt.figure(1, figsize=(8, 6))
plt.clf()
ax = Axes3D(fig, rect=[0, 0, .95, 1], elev=48, azim=134)
plt.cla()
ax.set_xlabel('Education')
ax.set_ylabel('Age')
ax.set_zlabel('Income')
ax.scatter(X[:, 1], X[:, 0], X[:, 3], c=labels.astype(np.float))

```

3.2. CLUSTERING JERÁRQUICO

```
# Importamos las librerías necesarias
import numpy as np
import pandas as pd
import urllib.request
from scipy import ndimage
from scipy.cluster import hierarchy
from scipy.spatial import distance_matrix
from matplotlib import pyplot as plt
from sklearn import manifold, datasets
from sklearn.cluster import AgglomerativeClustering
from sklearn.datasets.samples_generator import make_blobs
%matplotlib inline

# Descargamos los datos
url = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDriverSkillsNetwork-ML0101EN-SkillsNetwork/labs/Module%204/data/cars_clus.csv'
filename = 'cars_clus.csv'
urllib.request.urlretrieve(url, filename)

# Leemos el csv en un dataframe de Pandas
pdf = pd.read_csv(filename)
print ("Shape of dataset: ", pdf.shape)

# Eliminamos las filas que contienen valores null
print ("Shape of dataset before cleaning: ", pdf.size)
pdf[['sales', 'resale', 'type', 'price', 'engine_s',
      'horsepow', 'wheelbas', 'width', 'length', 'curb_wgt', 'fuel_cap',
      'mpg', 'Insales']] = pdf[['sales', 'resale', 'type', 'price', 'engine_s',
      'horsepow', 'wheelbas', 'width', 'length', 'curb_wgt', 'fuel_cap',
      'mpg', 'Insales']].apply(pd.to_numeric, errors='coerce')
pdf = pdf.dropna()
pdf = pdf.reset_index(drop=True)
print ("Shape of dataset after cleaning: ", pdf.size)

# Selección de características
featureset = pdf[['engine_s', 'horsepow', 'wheelbas', 'width', 'length', 'curb_wgt', 'fuel_cap', 'mpg']]
```

```

# Normalización
from sklearn.preprocessing import MinMaxScaler
x = featureset.values #returns a numpy array
min_max_scaler = MinMaxScaler()
feature_mtx = min_max_scaler.fit_transform(x)

#
dist_matrix = distance_matrix(feature_mtx,feature_mtx)

#
agglom = AgglomerativeClustering(n_clusters = 6, linkage = 'complete')
agglom.fit(feature_mtx)
agglom.labels_

# Agregamos nuevo campo al dataframe para mostrar el cluster de cada fila
pdf['cluster_'] = agglom.labels_

# Número de casos en cada grupo
pdf.groupby(['cluster_','type'])['cluster_'].count()

# Características de cada cluster
agg_cars = pdf.groupby(['cluster_','type'])[['horsepow','engine_s','mpg','price']].mean()

# Graficamos
plt.figure(figsize=(16,10))
for color, label in zip(colors, cluster_labels):
    subset = agg_cars.loc[(label),]
    for i in subset.index:
        plt.text(subset.loc[i][0]+5, subset.loc[i][2], 'type=' + str(int(i)) + ', price=' + str(int(subset.loc[i][3])) + 'k')
    plt.scatter(subset.horsepow, subset.mpg, s=subset.price*20, c=color, label='cluster'+str(label))
plt.legend()
plt.title('Clusters')
plt.xlabel('horsepow')
plt.ylabel('mpg')

```

3.3. DBSCAN

```
import numpy as np
import urllib.request
import csv
import pandas as pd
from sklearn.cluster import DBSCAN
from sklearn.datasets.samples_generator import make_blobs
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
from mpl_toolkits.basemap import Basemap
import matplotlib.pyplot as plt
from pylab import rcParams
import sklearn.utils
%matplotlib inline

# Descargamos los datos
url = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDriverSkillsNetwork-ML0101EN-SkillsNetwork/labs/Module%204/data/weather-stations20140101-20141231.csv'
filename = 'weather-stations20140101-20141231.csv'
urllib.request.urlretrieve(url, filename)

# Leemos el csv en un dataframe
pdf = pd.read_csv(filename)

# Removemos filas que no tengan ningún valor en Tm
pdf = pdf[pd.notnull(pdf["Tm"])]
pdf = pdf.reset_index(drop=True)

# Visualización
rcParams['figure.figsize'] = (14,10)
llon=-140
ulon=-50
llat=40
ulat=65
pdf = pdf[(pdf['Long'] > llon) & (pdf['Long'] < ulon) & (pdf['Lat'] > llat) &(pdf['Lat'] < ulat)]
my_map = Basemap(projection='merc',
resolution = 'l', area_thresh = 1000.0,
```

```

llcrnrlon=llon, llcrnrlat=llat, #min longitude (llcrnrlon) and latitude (llcrnrlat)
urcrnrlon=ulon, urcrnrlat=ulat) #max longitude (urcrnrlon) and latitude (urcrnrlat)

my_map.drawcoastlines()
my_map.drawcountries()
# my_map.drawmapboundary()
my_map.fillcontinents(color = 'white', alpha = 0.3)
my_map.shadedrelief()

# To collect data based on stations

xs,ys = my_map(np.asarray(pdf.Long), np.asarray(pdf.Lat))

pdf['xm']= xs.tolist()
pdf['ym'] =ys.tolist()

#Visualization1

for index,row in pdf.iterrows():

    my_map.plot(row.xm, row.ym,markerfacecolor =([1,0,0]), marker='o', markersize= 5, alpha = 0.75)

plt.show()

# Clustering de estaciones basado en su ubicación (latitud y longitud)

sklearn.utils.check_random_state(1000)

Clus_dataSet = pdf[['xm','ym']]

Clus_dataSet = np.nan_to_num(Clus_dataSet)

Clus_dataSet = StandardScaler().fit_transform(Clus_dataSet)

# Compute DBSCAN

db = DBSCAN(eps=0.15, min_samples=10).fit(Clus_dataSet)

core_samples_mask = np.zeros_like(db.labels_, dtype=bool)

core_samples_mask[db.core_sample_indices_] = True

labels = db.labels_

pdf["Clus_Db"]=labels

realClusterNum=len(set(labels)) - (1 if -1 in labels else 0)

clusterNum = len(set(labels))

# Una muestra de los clusters

pdf[['Stn_Name','Tx','Tm','Clus_Db']].head(5)

# Visualización del clustering basado en ubicación

rcParams['figure.figsize'] = (14,10)

my_map = Basemap(projection='merc',
    resolution = 'I', area_thresh = 1000.0,
    llcrnrlon=llon, llcrnrlat=llat, #min longitude (llcrnrlon) and latitude (llcrnrlat)
    urcrnrlon=ulon, urcrnrlat=ulat) #max longitude (urcrnrlon) and latitude (urcrnrlat)

```

```

my_map.drawcoastlines()
my_map.drawcountries()
#my_map.drawmapboundary()
my_map.fillcontinents(color = 'white', alpha = 0.3)
my_map.shadedrelief()
# To create a color map
colors = plt.get_cmap('jet')(np.linspace(0.0, 1.0, clusterNum))

#Visualization1

for clust_number in set(labels):
    c=([0.4,0.4,0.4]) if clust_number == -1 else colors[np.int(clust_number)]
    clust_set = pdf[pdf.Clus_Db == clust_number]
    my_map.scatter(clust_set.xm, clust_set.ym, color =c, marker='o', s= 20, alpha = 0.85)
    if clust_number != -1:
        cenx=np.mean(clust_set.xm)
        ceny=np.mean(clust_set.ym)
        plt.text(cenx,ceny,str(clust_number), fontsize=25, color='red',)
        print ("Cluster "+str(clust_number)+', Avg Temp: '+ str(np.mean(clust_set.Tm)))

```

```

# Clustering basado en ubicación, máxima, media y mínima temperatura
sklearn.utils.check_random_state(1000)
Clus_dataSet = pdf[['xm','ym','Tx','Tm','Tn']]
Clus_dataSet = np.nan_to_num(Clus_dataSet)
Clus_dataSet = StandardScaler().fit_transform(Clus_dataSet)

# Compute DBSCAN
db = DBSCAN(eps=0.3, min_samples=10).fit(Clus_dataSet)
core_samples_mask = np.zeros_like(db.labels_, dtype=bool)
core_samples_mask[db.core_sample_indices_] = True
labels = db.labels_
pdf["Clus_Db"]=labels
realClusterNum=len(set(labels)) - (1 if -1 in labels else 0)
clusterNum = len(set(labels))

# A sample of clusters
pdf[["Stn_Name","Tx","Tm","Clus_Db"]].head(5)

```

```

# Visualización del clustering anterior
rcParams['figure.figsize'] = (14,10)
my_map = Basemap(projection='merc',

```

```

resolution = 'I', area_thresh = 1000.0,
llcrnrlon=llon, llcrnrlat=llat, #min longitude (llcrnrlon) and latitude (llcrnrlat)
urcrnrlon=ulon, urcrnrlat=ulat) #max longitude (urcrnrlon) and latitude (urcrnrlat)

my_map.drawcoastlines()
my_map.drawcountries()
#my_map.drawmapboundary()
my_map.fillcontinents(color = 'white', alpha = 0.3)
my_map.shadedrelief()

# To create a color map
colors = plt.get_cmap('jet')(np.linspace(0.0, 1.0, clusterNum))

#Visualization1

for clust_number in set(labels):
    c=(([0.4,0.4,0.4]) if clust_number == -1 else colors[np.int(clust_number)])
    clust_set = pdf[pdf.Clus_Db == clust_number]
    my_map.scatter(clust_set.xm, clust_set.ym, color =c, marker='o', s= 20, alpha = 0.85)
    if clust_number != -1:
        cenx=np.mean(clust_set.xm)
        ceny=np.mean(clust_set.ym)
        plt.text(cenx,ceny,str(clust_number), fontsize=25, color='red')
        print ("Cluster "+str(clust_number)+', Avg Temp: '+ str(np.mean(clust_set.Tm)))

```