

Maestría en Inteligencia Artificial

Análisis de algoritmos

Dr. Efrén Juárez-Castillo



Objetivo

El alumno será capaz de **aplicar algoritmos de inteligencia artificial para dar solución a necesidades de** clasificación, predicción y aprendizaje en sistemas informáticos



Unidades temáticas

- 1. Introducción a algoritmos de IA
- 2. Machine Learning supervisado
- 3. Machine Learning no supervisado
- 4. Métodos de aprendizaje profundo

Repositorios de datos

- Un repositorio de datos para data mining es una **plataforma o colección en línea que almacena y proporciona acceso a conjuntos de datos** diversos y relevantes para realizar tareas de minería de datos.
- Estos repositorios permiten a los investigadores, científicos de datos y profesionales acceder a conjuntos de **datos preprocesados y listos para ser utilizados** en análisis y proyectos de minería de datos, lo que facilita la exploración, evaluación y aplicación de técnicas de extracción de conocimiento.

Repositorios comunes

- UCI Machine Learning Repository:
<http://archive.ics.uci.edu/ml>
- Kaggle: <https://www.kaggle.com/datasets>
- Data.gov: <https://www.data.gov/>
- Google Dataset Search:
<https://datasetsearch.research.google.com/>
- Amazon AWS Public Datasets:
<https://registry.opendata.aws/>
- INEGI: Instituto Nacional de Estadística y Geografía de México, que ofrece una amplia gama de datos estadísticos y geoespaciales para el país. <https://www.inegi.org.mx/>

Formatos de archivo para almacenar y representar conjuntos de datos

- CSV (Comma-Separated Values): Formato de valores separados por comas para datos tabulares.
- ARFF (Attribute-Relation File Format): Formato específico para conjuntos de datos utilizados en minería de datos y aprendizaje automático.
pip install liac-arff
- Excel (XLS, XLSX): Formato de hoja de cálculo ampliamente utilizado para almacenar y analizar datos tabulares.
- JSON (JavaScript Object Notation): Formato de texto basado en pares clave-valor utilizado para intercambio de datos.
- SQLite: Sistema de gestión de bases de datos relacional que permite almacenar y consultar datos estructurados en archivos compactos.

Metodologías Data mining y Aprendizaje automático

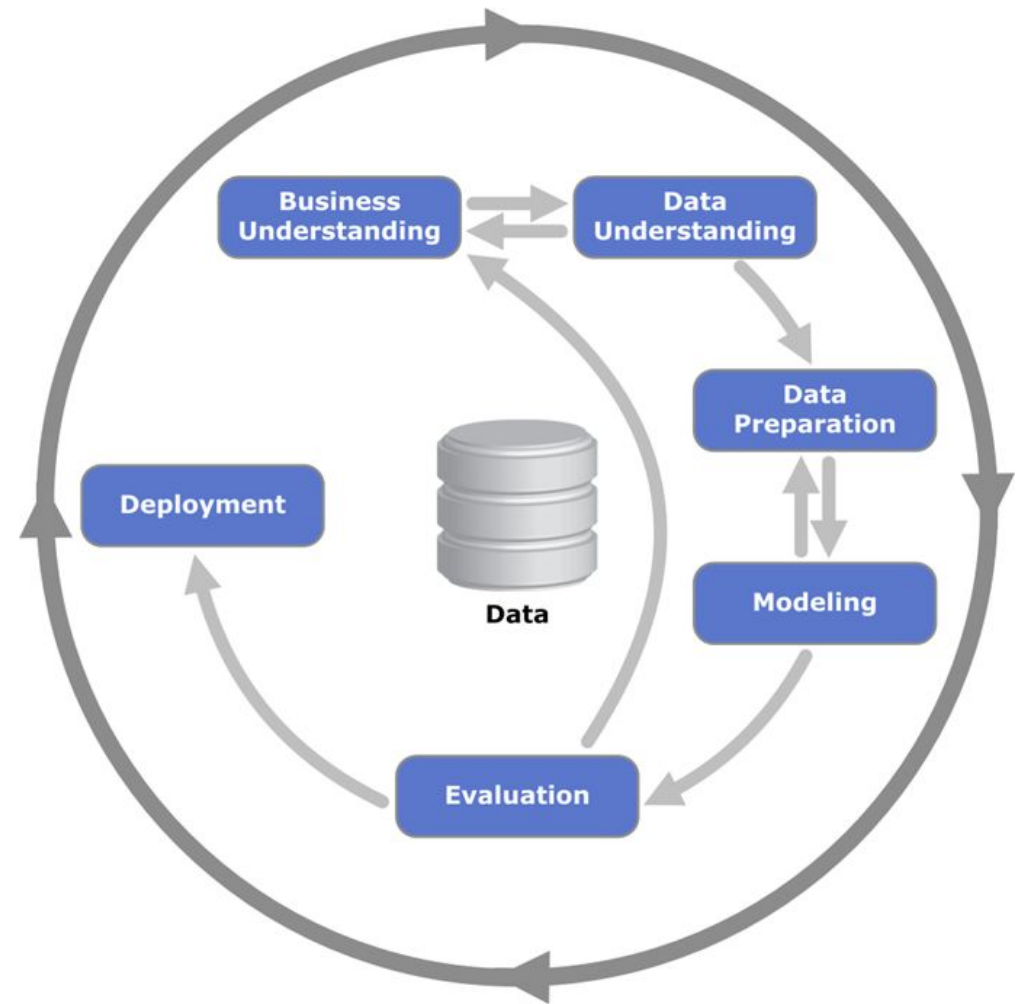
CRISP-DM (Cross-Industry Standard Process for Data Mining)

KDD Process Model

SEMMA (Sample, Explore, Modify, Model, Assess)

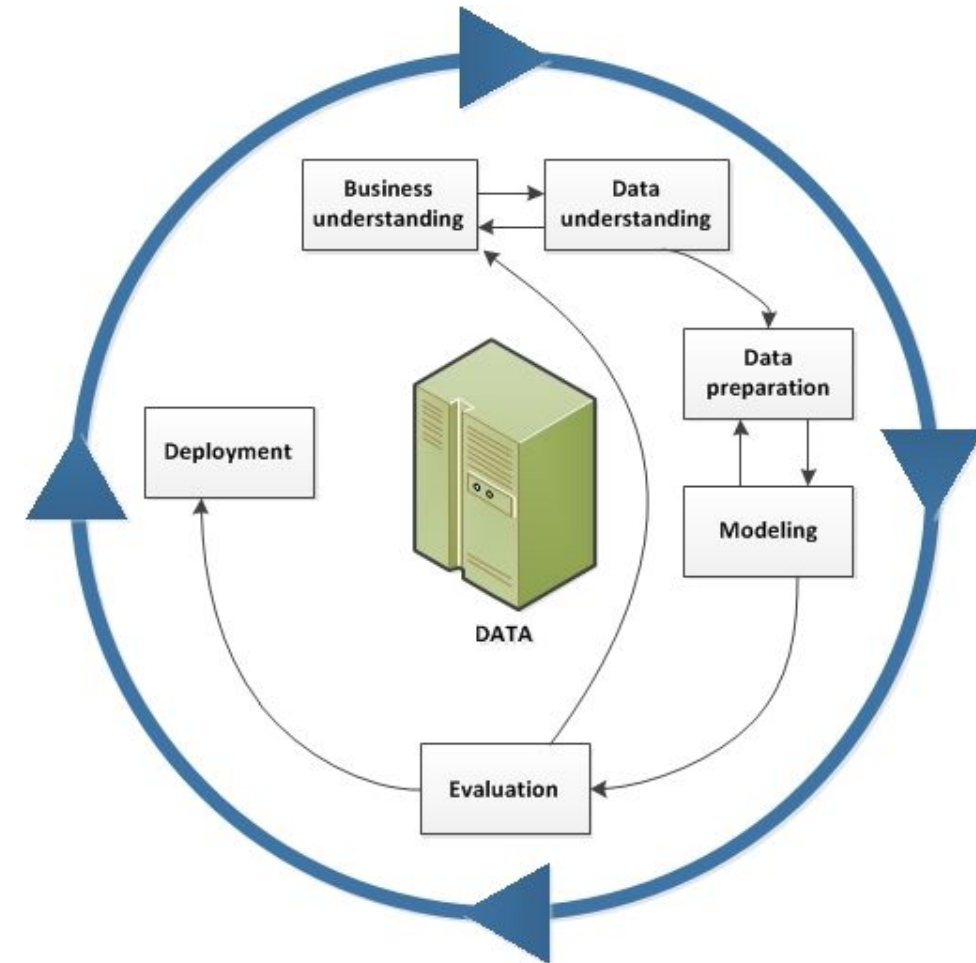
Metodología CRISP-DM (Cross Industry Process for Data Mining)

- Metodología estándar con seis fases: comprensión del negocio, comprensión de datos, preparación de datos, modelado, evaluación y despliegue.



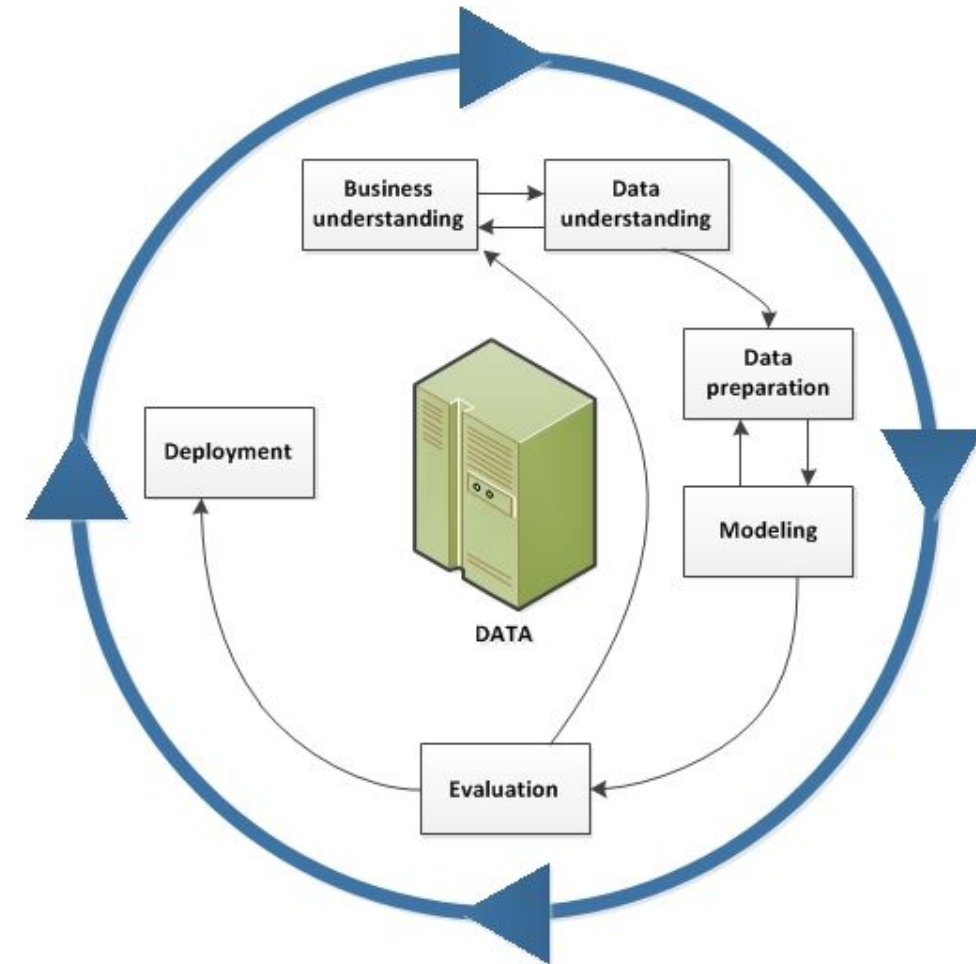
CRISP-DM (Cross-Industry Standard Process for Data Mining)

- Es una metodología popular que proporciona un marco estructurado para guiar un proyecto de minería de datos a través de sus varias fases.
- **Comprensión del negocio:** Esta es la primera fase del proceso, en la que se definen los objetivos del proyecto y se identifican los requisitos desde una perspectiva empresarial. El objetivo es entender el proyecto y planificar cómo abordarlo. Se establecen los criterios de éxito y se identifican los recursos disponibles.



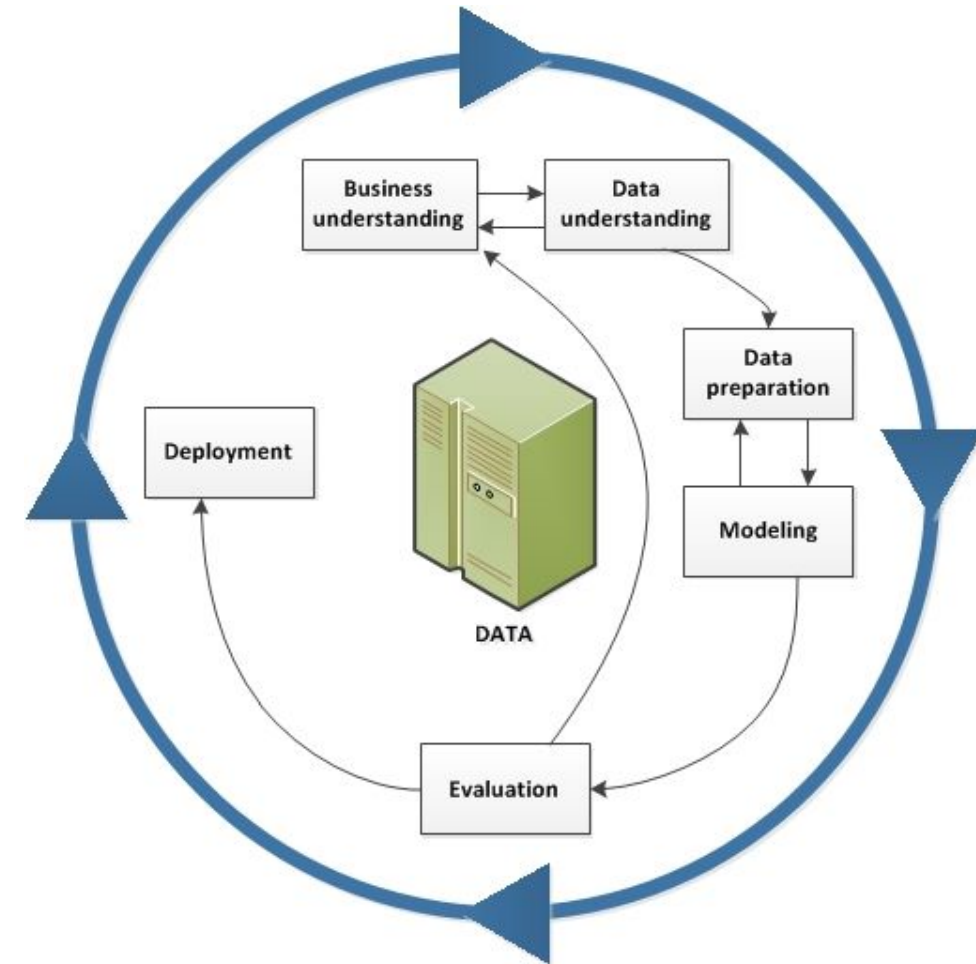
CRISP-DM (Cross-Industry Standard Process for Data Mining)

- **Comprensión de los datos:** En esta fase, se recopilan los datos y se familiarizan con ellos utilizando diversas herramientas para identificar la calidad de los datos, detectar problemas iniciales, descubrir primeras intuiciones sobre los datos y determinar las hipótesis interesantes que se podrían confirmar con los datos.
- **Preparación de los datos:** Aquí los datos son limpiados y transformados en un formato apropiado para la minería de datos. Esto podría implicar la limpieza de datos (tratamiento de datos perdidos, eliminación de ruido o errores), la selección de variables, la transformación de variables y la creación de nuevas variables.



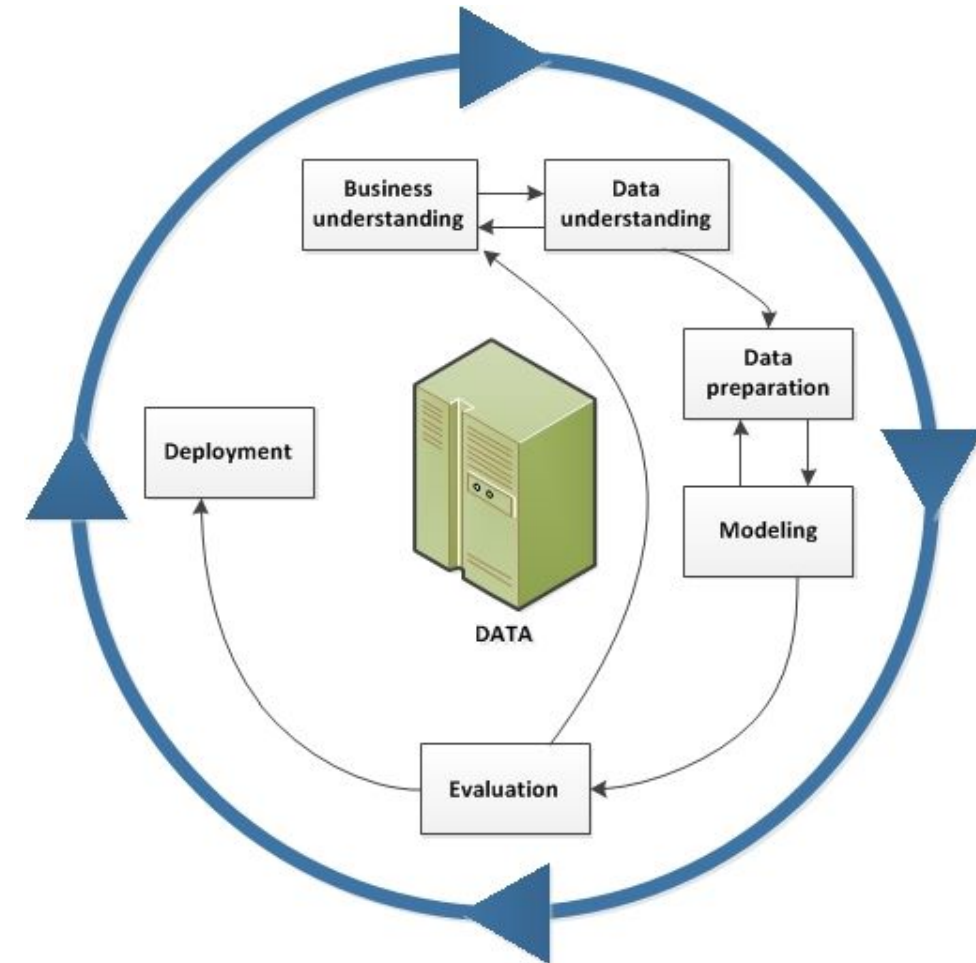
CRISP-DM (Cross-Industry Standard Process for Data Mining)

- **Modelado:** En esta fase, se seleccionan y aplican varias técnicas de modelado y algoritmos de minería de datos para los datos preparados. Se selecciona el algoritmo de minería de datos adecuado y se crean los modelos de datos. Este proceso puede requerir la vuelta al paso de preparación de datos si es necesario.
- **Evaluación:** Aquí se evalúan los modelos generados para determinar su calidad y eficacia. Los modelos se prueban y se revisan para determinar si satisfacen los objetivos empresariales definidos en la primera fase. Se revisan los resultados obtenidos para decidir si el modelo es suficientemente bueno para el despliegue.



CRISP-DM (Cross-Industry Standard Process for Data Mining)

- **Despliegue:** En esta última fase, se implementa el modelo en el entorno de producción y se monitoriza para asegurar que está produciendo los resultados esperados. Esto podría implicar la generación de informes, la puesta en marcha de un proceso de minería de datos repetible o la implementación de un modelo en una aplicación de datos en tiempo real.



Comprensión del negocio

- **Definición de los objetivos del negocio:** Antes de empezar a recoger o analizar datos, es importante tener una comprensión clara de lo que el negocio espera lograr con el proyecto de minería de datos. Esto podría implicar identificar un problema que el negocio quiere resolver, una oportunidad que quieren aprovechar, o una pregunta que quieren responder.
- **Evaluación de la situación actual:** Esto podría implicar una revisión de los recursos disponibles (incluyendo los datos, las herramientas y las habilidades), una evaluación de las limitaciones y restricciones, y una revisión de los procesos y las políticas relevantes del negocio.
- **Determinación de los objetivos de la minería de datos:** Con base en los objetivos del negocio, se determinan los objetivos específicos de la minería de datos. Estos objetivos deberían ser medibles y específicos, y deberían alinearse con los objetivos del negocio.
- **Creación de un plan de proyecto:** Esto podría implicar la definición de un cronograma del proyecto, la asignación de roles y responsabilidades, y la identificación de los pasos necesarios para lograr los objetivos de la minería de datos.

Comprensión de los datos

Es el paso donde se recogen, describen, exploran y verifican la calidad de los datos que se van a utilizar en el proyecto de minería de datos

1. **Recopilación de datos:** Esto puede implicar la recopilación de datos de varias fuentes, como bases de datos, archivos de texto, APIs de Internet, entre otros.
2. **Exploración de datos:** Esto puede implicar el uso de estadísticas descriptivas para entender la distribución de los datos, la identificación de posibles correlaciones entre las variables, y la visualización de los datos para entender las relaciones y las tendencias.
3. **Verificación de la calidad de los datos:** Esto puede implicar la identificación y el manejo de valores perdidos o erróneos, la detección de posibles outliers, y la verificación de la coherencia y la fiabilidad de los datos.

Tipos de atributos

Tipos de datos

- **Datos nominales:** Este es el tipo más simple de datos. Son datos que pueden ser divididos en múltiples categorías pero que no tienen ningún orden o prioridad. Ejemplos: género (masculino, femenino), color (rojo, azul, verde), país de origen (EE.UU., México, Canadá).
- **Datos ordinales:** Al igual que los datos nominales, los datos ordinales se pueden dividir en múltiples categorías, pero estas categorías tienen un orden o jerarquía específicos. Sin embargo, la distancia entre las categorías no es necesariamente uniforme. Ejemplos: niveles de satisfacción (muy insatisfecho, insatisfecho, neutral, satisfecho, muy satisfecho), grados educativos (primaria, secundaria, universidad).
- **Datos discretos / Intervalo:** Este es un tipo de datos numéricos que pueden tomar un número contable de valores. A menudo son el resultado de contar. Ejemplos: número de hijos en una familia, número de estudiantes en una clase.
- **Datos continuos:** Este es otro tipo de datos numéricos, pero a diferencia de los datos discretos, los datos continuos pueden tomar cualquier valor dentro de un rango específico. Los datos continuos a menudo son el resultado de la medición. Ejemplos: altura, peso, temperatura, edad.

Types of Attributes

Attribute Type	Description	Examples	Operations
Nominal	The values of a nominal attribute are just different names, i.e., nominal attributes provide only enough information to distinguish one object from another. ($=$, \neq)	zip codes, employee ID numbers, eye color, sex: $\{male, female\}$	mode, entropy, contingency correlation, χ^2 test
Ordinal	The values of an ordinal attribute provide enough information to order objects. ($<$, $>$)	hardness of minerals, $\{good, better, best\}$, grades, street numbers	median, percentiles, rank correlation, run tests, sign tests
Interval	For interval attributes, the differences between values are meaningful, i.e., a unit of measurement exists. ($+$, $-$)	calendar dates, temperature in Celsius or Fahrenheit	mean, standard deviation, Pearson's correlation, t and F tests
Ratio	For ratio variables, both differences and ratios are meaningful. ($*$, $/$)	temperature in Kelvin, monetary quantities, counts, age, mass, length, electrical current	geometric mean, harmonic mean, percent variation

Properties of Attribute Values

- The type of an attribute depends on which of the following properties it possesses:
 - Distinctness: $= \neq$
 - Order: $< >$
 - Addition: $+ -$
 - Multiplication: $* /$
 - Nominal attribute: distinctness
 - Ordinal attribute: distinctness & order
 - Interval attribute: distinctness, order & addition
 - Ratio attribute: all 4 properties

Datos numéricos

```
graph TD; A[Datos numéricos] --> B(Continua); A --> C(Discreta); B --> B1[altura, peso, salario, temperatura, tipos de interés.]; B --> B2[23.45, 45.76, 89.26]; C --> C1[unidades vendidas, número de idiomas hablados, número de estudiantes.]; C --> C2[33, 56, 78, 12];
```

Continua

altura, peso, salario,
temperatura,
tipos de interés.

23.45, 45.76, 89.26

Discreta

unidades vendidas,
número de idiomas
hablados, número de
estudiantes.

33, 56, 78, 12

Discrete and Continuous Attributes

- Discrete Attribute
 - Has only a finite or countably infinite set of values
 - Examples: zip codes, counts, or the set of words in a collection of documents
 - Often represented as integer variables.
 - Note: binary attributes are a special case of discrete attributes
- Continuous Attribute
 - Has real numbers as attribute values
 - Examples: temperature, height, or weight.
 - Practically, real values can only be measured and represented using a finite number of digits.
 - Continuous attributes are typically represented as floating-point variables.

¿Cuáles son categóricos?

Datos categóricos

Son aquellos que están divididos en distintas categorías o grupos

1. **Datos nominales:** Estos son datos categóricos que no tienen un orden o jerarquía inherente. Por ejemplo, el color del pelo (rubio, moreno, pelirrojo) es un atributo nominal porque no hay un orden lógico en el que estas categorías puedan ser organizadas.
2. **Datos ordinales:** Estos son datos categóricos que tienen un orden o jerarquía específica. Por ejemplo, las calificaciones en una encuesta de satisfacción (insatisfecho, neutral, satisfecho) son un atributo ordinal porque las categorías tienen un orden lógico (satisfecho es mejor que neutral, que a su vez es mejor que insatisfecho).

Por otro lado, **los datos discretos y continuos** son tipos de datos numéricos, **no categóricos**.

Otras clasificación para tipos de datos

- **Datos numéricos:** Estos son datos cuantitativos que se pueden medir en una escala numérica. Pueden ser continuos (como la altura o el peso) o discretos (como el número de hijos).
- **Datos categóricos:** Estos son datos cualitativos que describen una característica o cualidad de una observación y que generalmente se dividen en categorías o grupos. Los datos categóricos pueden ser nominales (como el color de los ojos o el país de origen) o ordinales (como un rango de satisfacción en una encuesta) y pueden estar representados por números; clase 1, clase 2.
- **Datos binarios:** Son un tipo especial de datos categóricos que sólo tienen dos categorías (0/1, Verdadero/Falso, Sí/No).
- **Datos de texto:** Estos son datos no estructurados que consisten en palabras, frases o incluso párrafos enteros de texto.
- **Datos de fecha y hora:** Estos datos representan puntos específicos en el tiempo o intervalos de tiempo.
- **Datos de secuencia:** Son datos ordenados en una secuencia específica, como una serie de eventos que ocurren en un orden específico.
- **Datos geoespaciales:** Estos datos representan ubicaciones geográficas, como coordenadas de latitud y longitud.
- **Datos de imagen, audio y video:** Estos son datos no estructurados que pueden requerir técnicas especiales de procesamiento y análisis.

Dataset Titanic

Identificar
los
tipos de
datos

```
df.head()  
df.describe()  
df.info()  
df['Embarked'].unique()  
df['Embarked'].value_counts()
```

The screenshot shows the Kaggle website interface. On the left is a sidebar with navigation links: Create, Home, Competitions (highlighted), Datasets, Code, Discussions, Courses, and More. Below these are 'Your Work' and a 'RECENTLY VIEWED' section showing 'Titanic - Machine Le...'. The main content area features a search bar at the top. Below it is a large banner for the 'Titanic - Machine Learning from Disaster' competition, with the text 'Start here! Predict survival on the Titanic and get familiar with ML basics' and 'Kaggle · 13,792 teams · Ongoing'. A navigation bar below the banner includes links for Overview, Data, Code, Discussion, Leaderboard, Rules, Team, My Submissions, and a Submit Predictions button. The 'Overview' section is active, showing a 'Description' tab. The description text reads: 'Ahoy, welcome to Kaggle! You're in the right place. This is the legendary Titanic ML competition – the best, first challenge for you to dive into ML competitions and familiarize yourself with how the Kaggle platform works. The competition is simple: use machine learning to create a model that predicts which passengers survived the Titanic shipwreck.'

<https://www.kaggle.com/c/titani>

Dataset Titanic



	Tipo	Nulos
0 PassengerId		
1 Survived		
2 Pclass		
3 Name		
4 Sex		
5 Age		
6 SibSp		
7 Parch		
8 Ticket		
9 Fare		
10 Cabin		
11 Embarked		

Dataset Titanic

- 0 PassengerId Discreto
- 1 Survived Nominal
- 2 Pclass Ordinal
- 3 Name Nominal
- 4 Sex Nominal
- 5 Age Continuo
- 6 SibSp Discreto
- 7 Parch Discreto
- 8 Ticket Nominal
- 9 Fare Continuo
- 10 Cabin Nominal
- 11 Embarked Nominal

Valores faltantes

Missing data

Valores faltantes

- La limpieza de datos es el proceso de encontrar y corregir los datos inexactos/incorrectos que están presentes en el conjunto de datos. Uno de esos procesos necesarios es hacer algo con los valores que faltan en el conjunto de datos. En la vida real, muchos conjuntos de datos tendrán muchos valores faltantes, por lo que tratar con ellos es un paso importante.
- **¿Por qué necesita completar los datos que faltan?** Porque la mayoría de los modelos de aprendizaje automático que desea utilizar generarán un error si le pasa valores de NaN. La forma más fácil es simplemente llenarlos con 0, pero esto puede reducir significativamente la precisión de su modelo.
- Para completar los valores faltantes, hay muchos métodos disponibles. Para elegir el mejor método, debe comprender el tipo de valor faltante y su significado, antes de comenzar a completar/eliminar los datos.

¿Cómo saber si los datos tienen valores faltantes?

- Los valores faltantes generalmente se representan en forma de **Nan** o **null** o **None** en el conjunto de datos.
- **df.info()** La función se puede utilizar para dar información sobre el conjunto de datos. Esto proporciona los nombres de las columnas junto con la cantidad de valores no nulos en cada columna.
- **isnull()** Para saber cuántos nulos: **print(df.isnull().sum())**

Métodos para tratar valores nulos

- Rellenar los datos faltantes con un valor – Imputación
- Imputación con una columna adicional
- Eliminar las columnas con datos faltantes
- Eliminar las filas con datos faltantes
- Llenar con un modelo de regresión

Rellenar los datos faltantes con un valor – Imputación

- Completar los datos faltantes con el promedio (mean) o la mediana (median) si es una variable numérica.
`df['SepalLengthCm'].mean()`
- Rellenar los datos que faltan con la moda si es un valor categórico. `df['Species'].mode()[0]`
- Rellenar un valor numérico con **0** o **-999**, o algún otro número que no aparecerá en los datos. Esto se puede hacer para que se pueda reconocer que los datos no son reales o son diferentes.
- Rellenar el valor categórico con un **nuevo tipo** para los valores faltantes.

Rellenar los datos faltantes con un valor – Imputación

- `updated_df = df`
- `updated_df['Age']=updated_df['Age'].fillna(updated_df['Age'].mean())`
- `df['SepalLengthCm'].mean()`
- `df['Species'].mode()[0]`

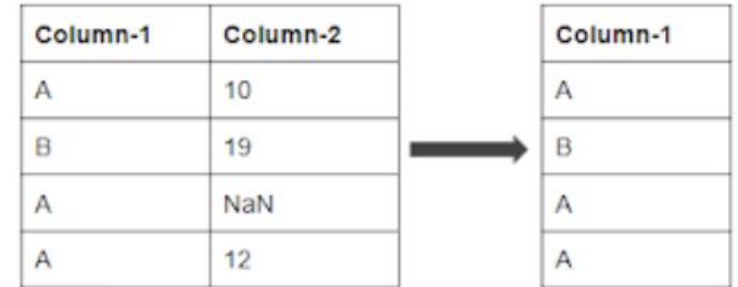
Imputación con una columna adicional

Column-1	Column-2		Column-1	Column-2	Column-3
A	10		A	10	False
B	19		B	19	False
A	NaN		A	12	True
A	12		A	12	False

- `updated_df = df`
- `updated_df['Ageismissing'] = updated_df['Age'].isnull()`
- `from sklearn.impute import SimpleImputer`
- `my_imputer = SimpleImputer(strategy = 'median')`
- `data_new = my_imputer.fit_transform(updated_df)`
- `updated_df.info()`

Eliminar las columnas con datos faltantes

- `df.drop(['Age'], axis=1)`
- `updated_df = df.dropna(axis=1)`
- `axis=1` para tratar columnas, `axis=0` para filas



The diagram illustrates the process of dropping a column with missing data. On the left, a table with two columns, 'Column-1' and 'Column-2', contains four rows. The third row has a missing value (NaN) in 'Column-2'. An arrow points to the right, where a new table is shown with only the 'Column-1' column, indicating that the column with missing data has been removed.

Column-1	Column-2
A	10
B	19
A	NaN
A	12

Column-1
A
B
A
A

- Solo debe usarse si hay demasiados valores nulos.
- El problema con este método es que **podemos perder información valiosa sobre esa columna**, ya que la hemos eliminado por completo debido a algunos valores nulos.

Eliminar las filas con datos faltantes

- `updated_df = df.dropna(axis=0)`
- `axis=1` para tratar columnas, `axis=0` para filas
- No es muy recomendable, pueden perderse muchos registros

Llenar con un modelo de regresión

- En este caso, los valores nulos en una columna se completan ajustando un modelo de regresión usando otras columnas en el conjunto de datos (el modelo de regresión contendrá todas las columnas excepto Age en X y Age en Y. Luego, después de completar los valores en la columna Edad, usaremos la regresión logística para calcular la precisión.
- Revisar el método en el siguiente enlace:
- <https://www.analyticsvidhya.com/blog/2021/05/dealing-with-missing-values-in-python-a-complete-guide/>



Codificación de Variables Categóricas


Codificación de Variables Categóricas

- Muchos algoritmos de machine learning requieren que las entradas sean numéricas. Esto significa que las variables categóricas deben ser transformadas en una forma que el algoritmo pueda entender.

Codificación One-Hot

- Crea una columna por categoría y pone un 1 en la columna de la categoría correspondiente y 0 en las demás.


```
df_encoded = pd.get_dummies(df, columns=['Embarked'],  
dtype=int)
```



Size			
M	0	1	0
L	1	0	0
S	0	0	1
M	0	1	0

Codificación Ordinal

- Asigna a cada categoría única un valor entero único.



workclass	workclass
State-gov	0
Self-emp-not-inc	1
Private	2
Private	2
Private	2

```
from sklearn.preprocessing import  
OrdinalEncoder
```

```
encoder = OrdinalEncoder()
```

```
df['Cabin'] = encoder.fit_transform(df[['Cabin']])
```


Codificación de etiquetas (y)

- 'LabelEncoder' es una utilidad de la biblioteca scikit-learn en Python que convierte **etiquetas** categóricas en números enteros, lo cual es útil para algoritmos de machine learning que requieren entradas numéricas. Se utiliza a menudo para codificar la variable objetivo, transformando categorías en valores numéricos correspondientes.

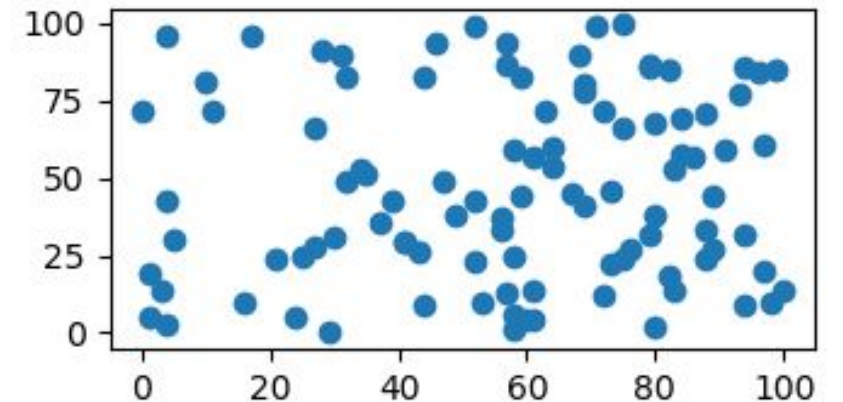
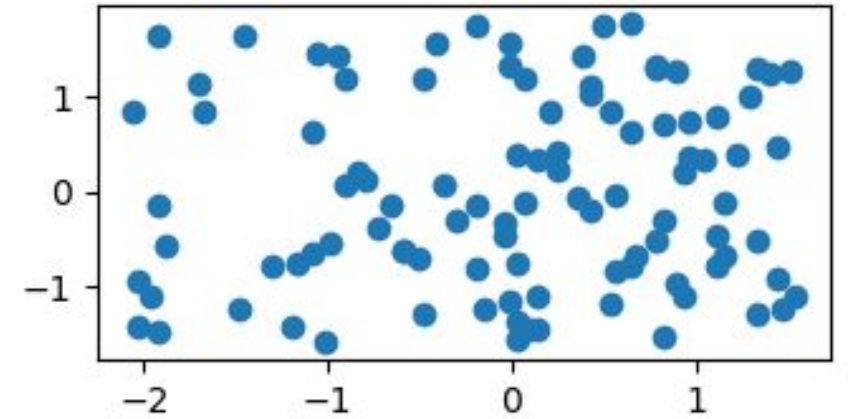
```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
df = pd.read_csv('iris.csv')
le = LabelEncoder()
df['species'] = le.fit_transform(df['species'])
```



Escalado de Características

Escalado de Características

- Algunos algoritmos de machine learning no se comportan bien cuando las características numéricas de entrada tienen escalas muy diferentes. Para solucionar este problema, es común escalar las características para que todas tengan aproximadamente el mismo rango



Normalización (Min-Max Scaling)

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

- Normalizar es el proceso de ajustar los valores de las características (o variables) en un dataset para que caigan dentro de un rango específico, comúnmente entre 0 y 1. Este proceso es útil cuando las características tienen diferentes escalas y se requiere que todas tengan la misma importancia en el análisis o modelado

```
from sklearn.preprocessing import  
MinMaxScaler  
  
scaler = MinMaxScaler()  
  
df['sepal length'] =  
scaler.fit_transform(df[['sepal length']])
```

Estandarización (Standard Scaling)

- Es una técnica de escalado en la que se ajustan los valores de las características para que tengan una media de 0 y una desviación estándar de 1. Esto significa que los valores se distribuyen de manera que el 68% de los valores se encuentran dentro de una desviación estándar de la media.

```
from sklearn.preprocessing import StandardScaler  
scaler = StandardScaler()  
data_scaled = scaler.fit_transform(data)
```

$$X' = \frac{X - \mu}{\sigma}$$

Model $f()$

X_{train}

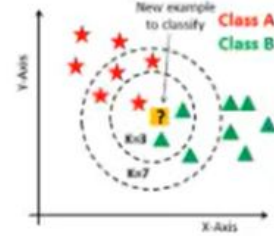
Features

Sepal length	Sepal width	Petal length	Petal width
5.1	3.5	1.4	0.2
4.9	3.0	1.4	0.2
7.0	3.2	4.7	1.4
6.4	3.2	4.5	1.5
6.3	3.3	6.0	2.5
5.8	3.3	6.0	2.5

Y_{train}

Labels

Species
Iris setosa
Iris setosa
Iris versicolor
Iris versicolor
Iris virginica



Model Training

Prediction

X_{test}

Features

Sepal length	Sepal width	Petal length	Petal width
5.1	3.5	1.4	0.2
4.9	3.0	1.4	0.2

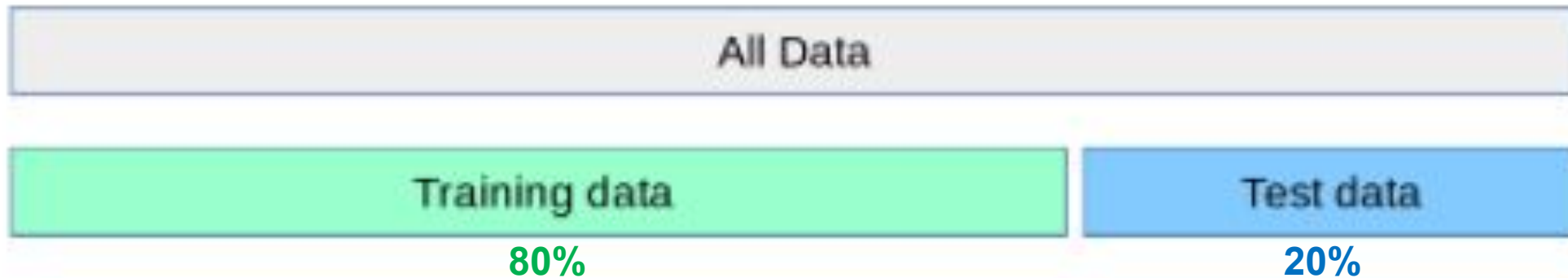


Clasificación

All Data = Training Data + Test Data

My_Collection

Insect ID	Abdomen Length	Antennae Length	Insect Class
1	2.7	5.5	Grasshopper
2	8.0	9.1	Katydid
3	0.9	4.7	Grasshopper
4	1.1	3.1	Grasshopper
5	5.4	8.5	Katydid
6	2.9	1.9	Grasshopper
7	6.1	6.6	Katydid
8	0.5	1.0	Grasshopper
9	8.3	6.6	Katydid
10	8.1	4.7	Katydid



Leave one out

```
1 from sklearn.model_selection import train_test_split as tts
2 x = df.drop(columns = 'price_range')
3 y = df.price_range
4 x_train, x_test, y_train, y_test = tts(
5     x, y.values.reshape(-1,1),
6     train_size = 0.8,
7     random_state = 1234,
8     shuffle = True)
9 print('Features x_train : ' + str(x_train.shape)+ ' x_test: ' + str(x_test.shape))
10 print('Features y_train : ' + str(y_train.shape)+ ' y_test: ' + str(y_test.shape))
```

```
y = df["price_range"].values
x=df.drop(["price_range"],axis=1)
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=1)
```

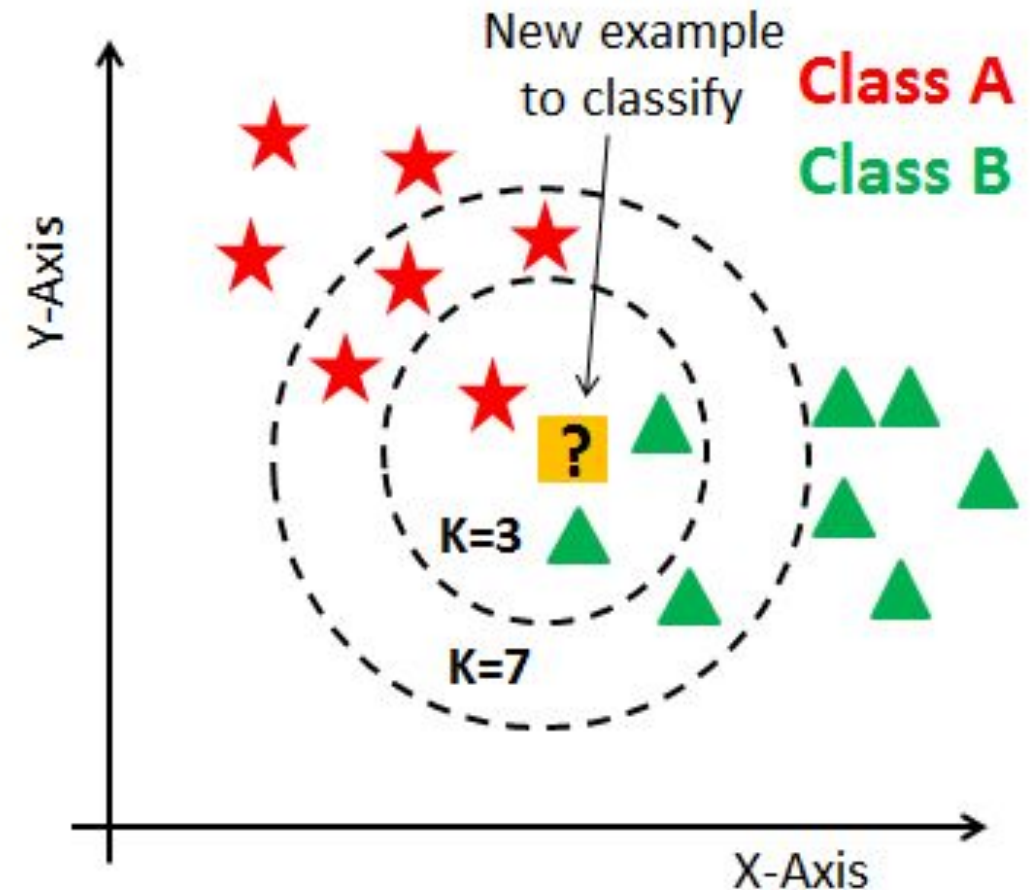
El parámetro `random_state` simplemente fija una semilla para el generador de números aleatorios, lo que permite reproducir la función.

$$d(P, Q) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

K-Nearest Neighbor (KNN)

$$d(P, Q) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

Es un algoritmo de clasificación supervisada que se basa en la similitud de los datos. Cuando se introduce un nuevo ejemplo a clasificar (representado con un signo de interrogación en la imagen), el algoritmo revisa los 'k' vecinos más cercanos a ese ejemplo en el espacio de características.





Preguntas