Experiencia Educativa: Aprendizaje Máquina

Dr. David Martínez Galicia

Ingeniería en Ciberseguridad e Infraestructura de Cómputo
Facultad de Economía e Informática
Universidad Veracruzana

Presentación

- Ingeniero en Mecatrónica.
- Maestro en Inteligencia Artificial.
- Doctor en Inteligencia Artificial.

Información del curso

- **Objetivo de la experiencia:** Brindar los fundamentos para analizar grandes **conjuntos de datos**, identificar **anomalías** y **valores atípicos**, así como identificar rápidamente **tendencias** y **patrones**, con el **entramiento** de algoritmos de **aprendizaje automático**, en diferentes ámbitos asociados a la ciberseguridad.

- Miércoles (07:99 – 08:59) y jueves (07:00 – 08:59).
- Oportunidades de evaluación: ordinario, extraordinario y título (si aplica).
- Contacto: davidmartinez02@uv.mx

Criterios de evaluación

Criterio	Porcentaje
Exámenes	40%
Tareas	30%
Proyecto	30%
Total	100%

- Para la acreditación tanto en ordinario, extraordinario y título de suficiencia es necesario obtener un mínimo de 6 en todos los criterios de evaluación.
- Cumplir con los porcentajes de asistencia que indica el estatuto.

Saberes teóricos

Introducción	ST-01. Manejo de datos	ST-02. Aprendizaje supervisado
- Inteligencia Artificial (IA) - Definición de la IA - Prueba de Turing - Campos de la IA	- Estructurados y no estructurados - Etiquetados y no etiquetados - Inconsistencia de datos	- Redes neuronales - Redes bayesianas - Árboles de decisión
ST-03. Aprendizaje no supervisado	**ST-04. Aprendizaje por refuerzo**	**ST-05. Evaluación de modelos**
- K-means - Variantes de K-means	- Q-learning	- Precisión - Sensibilidad y especificidad - ROC - Validación cruzada - Distancia intra e inter clúster

Calendarización

Mes	Febrero			Marzo				Abril					Mayo				Jun.
Semana	2°	3°	4°	1°	2°	3°	4°	1°	2°	3°	4°	5°	1°	2°	3°	4°	1°
Introducción									SS								
ST-01																	
ST-02						18											
ST-03																	
ST-04																	
ST-05																	

Notación: SS = Semana Santa. Días de suspensión en blanco.

Nacimiento de la Inteligencia Artificial

- El concepto de Inteligencia Artificial (IA) se acuña en 1956.
- Surge como un campo de estudio en un taller en Dartmouth College.
  - *Cada aspecto de la inteligencia puede ser descrito con suficiente precisión que puede fabricarse una máquina para simularlo.*
- Muchas preguntas, pocas respuestas y aún más esperanzas.

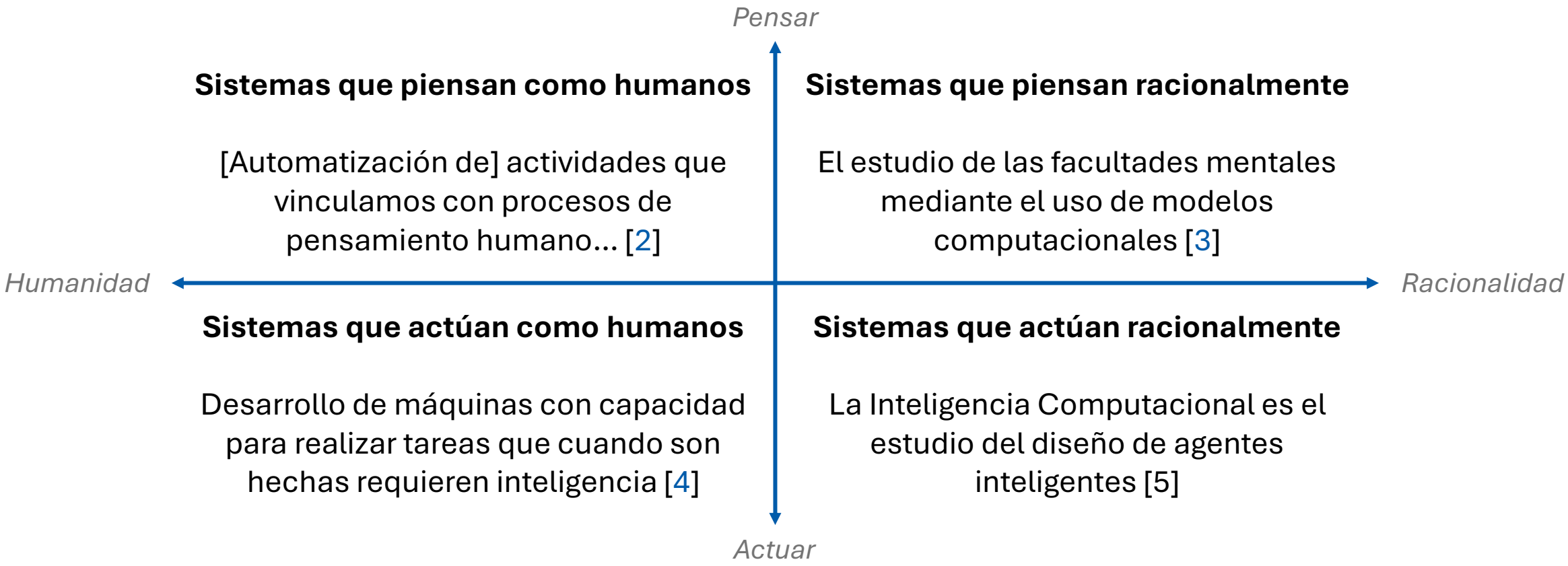
Objeto de estudio de la Inteligencia Artificial

- Rusell y Norvig [1] analizaron ocho libros de texto.
- Encontraron cuatro enfoques sobre cuál debería ser el objeto de estudio.
- Concuerdan en que la IA debe de estudiar sistemas inteligentes.
- ¿Qué características deben tener para ser inteligentes?

Cognición	Comportamiento
Pensar (crear)	Humano
Actuar (simular)	Racional

[1] Russell, S. J., y Norvig, P. (2021). *Inteligencia artificial: un enfoque moderno* (4.ª ed.). Pearson.

Definición de la Inteligencia Artificial



[2] Bellman, R. E. (1978). *An introduction to artificial intelligence: Can computers think?* Boyd & Fraser.
[3] Charniak, E., & McDermott, D. (1985). *Introduction to artificial intelligence*. Addison-Wesley.
[4] Kurzweil, R. (1990). *The age of intelligent machines*. MIT Press.
[5] Poole, D. L., et al. (1998). *Computational intelligence: A logical approach*. Oxford University Press.

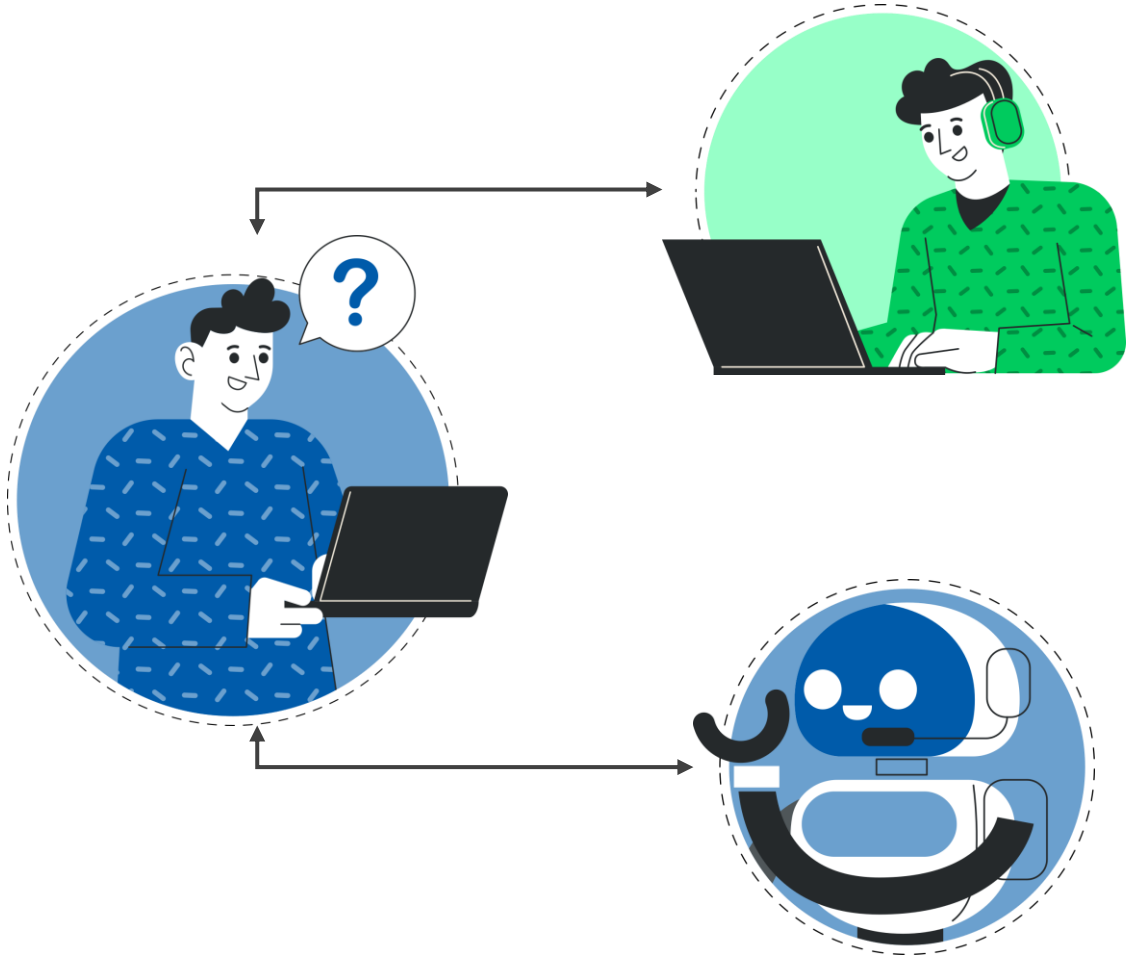
Simulación del comportamiento humano

- La Prueba de Turing [6] fue propuesta por Alan Turing (1950).
- Se diseñó para proporcionar una definición de inteligencia.
- No proporciona una lista de cualidades para obtener inteligencia.
- Se basa en la incapacidad de diferenciar entre entidades inteligentes.
- La prueba se supera si no se distingue entre una persona y un sistema.

[6] Turing, A. M. (1950). *Computing machinery and intelligence*. Mind, 59(236), 433–460.

La prueba de Turing

Habitación 1:
Evaluador humano
Cuenta con dos terminales de texto para comunicarse.



Habitación 2:
Ser humano
Recibe preguntas y responde por texto.

Habitación 3:
Sistema de IA
Recibe las mismas preguntas y responde por texto.

Capacidades de la prueba de Turing

El sistema debería poseer las siguientes capacidades:

- **Procesamiento de lenguaje natural** que le permita comunicarse en inglés.
- **Representación del conocimiento** para almacenar lo que se conoce.
- **Razonamiento automático** para responder preguntas usando información.
- **Aprendizaje máquina** para adaptarse a circunstancias y problemas.

La prueba global de Turing

- La prueba original evitó la interacción física entre evaluador y sistema.
- La prueba global [7] también evalúa la percepción y la manipulación física.

- Para superar la prueba global de Turing la computadora debe incluir:
  - **Visión computacional** para percibir objetos.
  - **Robótica** para manipular y mover objetos.

- Estas seis capacidades constituyen los campos de la IA actual.

[7] Harnad, S. (1989). *Minds, machines and Searle*. Journal of Experimental & Theoretical Artificial Intelligence, 1(1), 5–25.

Aprendizaje máquina

- El aprendizaje se refiere a un espectro de situaciones en las cuales un agente incrementa su conocimiento para cumplir una tarea.

- Según Tom Mitchell [8], un programa de computadora aprende de la **experiencia E**, con respecto a alguna clase de **tareas T** y una **medida de desempeño P**, si su desempeño en las tareas de T, medido por P, mejora con la experiencia E.

[8] Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill.

Tipos de aprendizaje

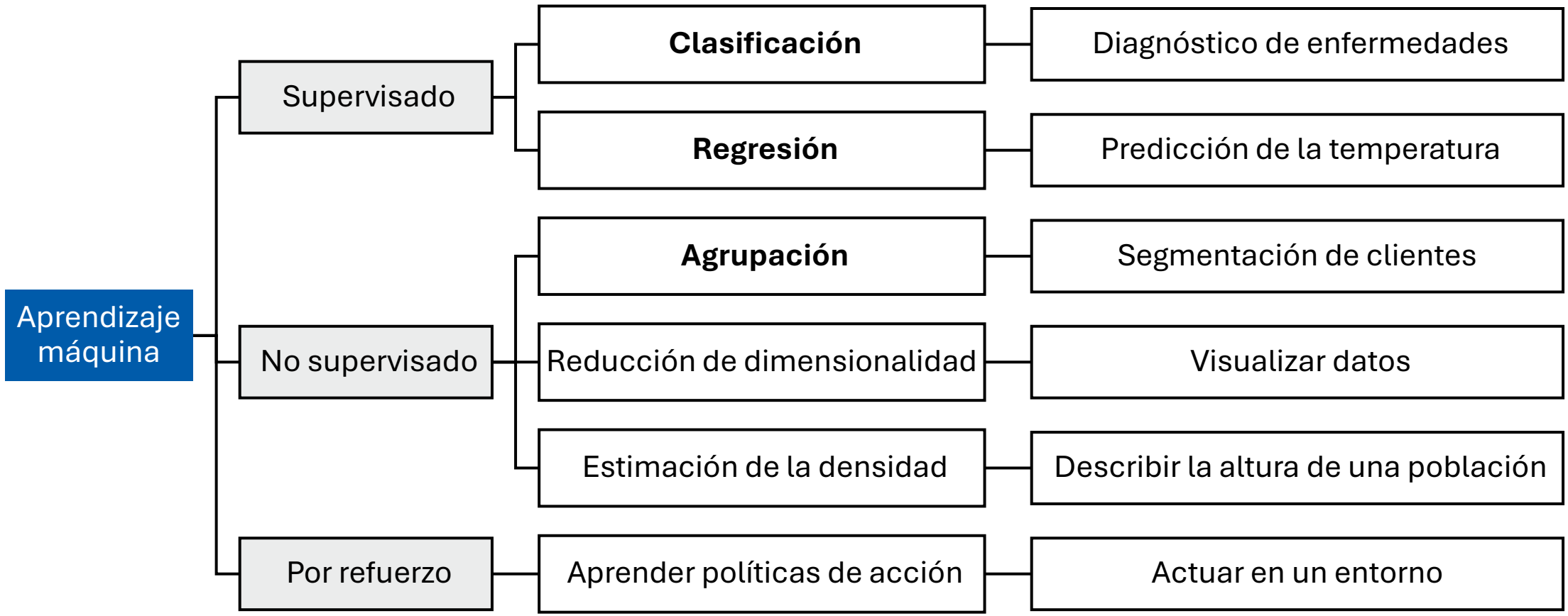
- Según el tipo de experiencia, se distinguen tres tipos de aprendizaje:
  - El **aprendizaje supervisado** consiste en aprender una función a partir de ejemplos de sus entradas y sus salidas.
  - El **aprendizaje no supervisado** consiste en aprender a partir de patrones de entradas que no tienen valores de salidas.
  - El **aprendizaje por refuerzo** consiste en aprender una forma de actuar usando un refuerzo (recompensa o penalización).

Ejemplos de aprendizaje

- Un sistema puede usar varios tipos de aprendizaje para distintas tareas.

SISTEMA TAXISTA		
Supervisado	**No supervisado**	**Por refuerzo**
Cuando el instructor de conducción grite “frena”, el agente puede aprender una regla condición-acción sobre cuándo frenar.	Desarrollar gradualmente los conceptos “día de tráfico bueno” y “día de tráfico malo”, sin que se le haya dado una solución.	La falta de propina al final del viaje da al agente algunas indicaciones de que su comportamiento no es el deseable.

Tareas de aprendizaje



Introducción al manejo de datos

- La calidad de los datos determina la calidad del modelo.
- Sin una preparación adecuada de los datos, no hay aprendizaje efectivo
- Principio fundamental: "Garbage in, garbage out".
- Transformamos datos crudos en patrones para la toma de decisiones.
- Datos limpios detectan amenazas reales en ciberseguridad.
- Se clasifican los datos por su estructura, etiquetas y consistencia.

Datos estructurados

- Datos organizados en formato tabular con esquema definido.

Características:

- Formato predecible (filas y columnas).
- Tipos de datos definidos (enteros, flotantes, categóricos).
- Fáciles de almacenar y consultar en bases de datos relacionales.

Ejemplos:

- Tablas SQL.
- Hojas de cálculo Excel/CSV.
- Logs de sistemas con formato fijo.

Datos no estructurados

- Datos sin formato predefinido ni organización tabular.

Características:

- Sin esquema fijo.
- Requieren procesamiento previo para extraer información.
- Representan ~80% de los datos empresariales.

Ejemplos:

- Texto (correos, código fuente, documentos, páginas web).
- Imágenes y videos.
- Audio y grabaciones.

Datos etiquetados

- Conjuntos de datos donde cada ejemplo tiene una etiqueta/clase asociada.

Características:

- Requieren proceso manual o semi-automático de etiquetado.
- Esenciales para entrenar modelos supervisados.
- Costosos de obtener y validar.

Ejemplos:

- Correos etiquetados como "spam" o "no spam".
- Imágenes de rostros con nombre de persona.
- Transacciones marcadas como "fraudulentas" o "legítimas".

Datos no etiquetados

- Datos sin categorías o respuestas conocidas.

Características:

- Abundantes y fáciles de recolectar.
- Requieren técnicas de descubrimiento de patrones.
- Útiles para detección de anomalías.

Ejemplos:

- Tráfico de red.
- Comportamientos de usuario.
- Código malicioso.

Repositorio de datos

- Los repositorios son bancos de datos públicos y organizados.
- Proporcionan conjuntos de datos para entrenar y validar modelos.
- Ofrecen datos etiquetados y documentados para distintos problemas.
- Son esenciales para la investigación y el desarrollo en aprendizaje máquina.
- Ejemplos: Kaggle, UCI, OpenML, gubernamentales y académicos.
- Su uso acelera el desarrollo y facilita la reproducibilidad.

Formatos de datos de texto

- Los formatos de texto permiten almacenar y procesar datos eficientemente.
- Son los más comunes para cargar en la mayoría de las herramientas.

  - CSV: formato tabular simple y universalmente compatible.
  - TSV: similar al CSV, pero con tabulaciones como delimitador.
  - JSON: ligero y flexible para datos estructurados y anidados.
  - XML: formato estructurado y autodescriptivo, aunque verboso.
  - ARFF: incluye metadatos, usado comúnmente en Weka.
  - Texto plano: sin formato, para datos sin estructura o documentos.

Limpieza y preprocesamiento de datos

- Los datos del mundo real son inexactos, inconsistentes e incompletos.
- El preprocesamiento de datos asegura la integridad de los datos.
- Permite generar análisis confiables y conclusiones válidas.
- Evita decisiones incorrectas o modelos ineficaces.

Flujo de trabajo de limpieza y preprocesamiento

- Recopilación: Obtener datos de bases de datos, APIs o scraping.
- Limpieza: Corregir errores, eliminar duplicados y manejar valores faltantes.
- Integración: Unir datos de múltiples fuentes, resolviendo inconsistencias.
- Transformación: Codificar, normalizar y crear nuevas variables.
- Reducción: Seleccionar características para simplificar el análisis.

Ejemplo de un flujo de trabajo

Recopilación	Limpieza	Integración	Transformación	Reducción
Fuentes comunes: Bases de datos, APIs, archivos CSV, scraping web.	Eliminar duplicados y observaciones irrelevantes. Manejar valores faltantes: Eliminar, imputar o marcar.	Alinear columnas y resolver conflictos entre fuentes.	Codificar variables categóricas ("Sí/No" a 1/0) Normalizar características numéricas (escalar entre 0 y 1)	Eliminar variables irrelevantes. Reducir dimensionalidad combinando variables.
Datos de ventas de múltiples tiendas.	Corregir errores tipográficos en nombres de productos.	Unir datos de clientes y ventas.	Crear características derivadas (edad a partir de fecha de nacimiento).	Usar solo 10 de 50 características para un modelo predictivo.

Python para el aprendizaje máquina

- **Python:** Lenguaje preferido por su facilidad y bibliotecas especializadas.

- Bibliotecas importantes:
  - Pandas: Manipulación de datos.
  - NumPy: Cálculos numéricos.
  - SciPy: Cómputo científico, estadística y señales.
  - Matplotlib/Seaborn: Visualización de datos.
  - Scikit-learn: Machine learning y preprocesamiento.

Pandas para la manipulación de datos

- Funcionalidades :
  - Filtrado, ordenamiento, agregación y fusión de datos.
  - Lectura y escritura de archivos (CSV, Excel, etc.).

```python
# Uso de datos desde un archivo
import pandas as pd
df = pd.read_csv('datos.csv')
df.head()
```

```python
# Uso de datos precargados
import pandas as pd
import seaborn as sns
df = sns.load_dataset('iris')
df.head()
```

- Uso típico: Limpieza, transformación y análisis exploratorio

NumPy para cálculos numéricos

- Funcionalidades:
  - Arreglos multidimensionales y operaciones matemáticas eficientes.
  - Base para muchas operaciones en Pandas y Scikit-learn.

```
import numpy as np                          # Carga de la biblioteca
arr = np.array([1, 2, 3, 4, 5])             # Definición de arreglos
arr2 = arr * 2                              # Multiplicación vectorial
```

- Uso típico: Cálculos numéricos y transformaciones de datos.

SciPy para cómputo científico y estadístico

- Contiene una amplia colección de algoritmos y funciones:
  - stats: pruebas estadísticas, distribuciones, medidas descriptivas.
  - signal: procesamiento de señales.
  - optimize: optimización y ajuste de curvas.
  - interpolate: interpolación de datos.
  - linalg: álgebra lineal avanzada.
- Se integra con NumPy (estructuras de datos) y Pandas (análisis de datos).

Matplotlib para la visualización de datos

- Funcionalidades:
  - Creación de gráficos (líneas, barras, dispersión, etc.).
  - Diagnóstico de problemas de calidad de datos.

```
import matplotlib.pyplot as plt          # Carga de la biblioteca
plt.plot([1, 2, 3, 4, 5])                # Creación de una gráfica
plt.title("Gráfico de línea simple")     # Título de la gráfica
plt.show()                               # Visualización de la gráfica
```

- Uso típico: Análisis exploratorio y visualización de datos.

Seaborn para la visualización estadística

- Funcionalidades:
  - Gráficos atractivos e informativos con pocas líneas de código.
  - Integración con Pandas para análisis rápido.

```
import seaborn as sns                          # Carga de la biblioteca
df = sns.load_dataset('tips’)                  # Uso de datos precargados
sns.histplot(df['total_bill’])                 # Creación de un histograma
plt.title("Histograma de Total Bill")          # Título de la gráfica
plt.show()                                     # Visualización de la gráfica
```

- Uso típico: Visualización avanzada y análisis estadístico.

Caso de estudio

- Un conjunto de datos ampliamente conocido y utilizado para practicar la limpieza y preprocesamiento de datos es el `Titanic Dataset`.
- Este conjunto contiene información sobre los pasajeros del Titanic, como edad, género, clase del boleto, tarifa, puerto de embarque y si sobrevivieron.

```
import pandas as pd
import seaborn as sns

df = sns.load_dataset('titanic’)          # Cargar Titanic desde Seaborn
df.head()                                 # Mostrar las primeras 5 filas
```

Análisis exploratorio de datos

***Exploratory Data Analysis* (EDA)**

- El análisis exploratorio es el primer acercamiento sistemático a los datos.
- Su objetivo es comprender la estructura y detectar problemas.
- Permite identificar patrones, tendencias y relaciones entre variables.
- Orienta las decisiones sobre limpieza, transformación y modelado.
- Un EDA sólido evita conclusiones erróneas y modelos mal especificados.
- Se apoya en estadísticas descriptivas y visualizaciones.

Dimensiones y tipos de datos I

- Es importante conocer la forma del conjunto: número de filas y columnas.
- Identificar el tipo de cada variable (numérica, categórica, texto, fecha).
- Verificar que los tipos sean adecuados para el análisis.

```
# Dimensiones del conjunto de datos
df.shape
print("Filas:", df.shape[0], "Columnas:", df.shape[1])

# Tipos de datos en cada variable
print(df.dtypes)

# Resumen de información
df.info()
```

Tipos e información del conjunto

```
print(df.dtypes)

survived          int64
pclass            int64
sex              object
age             float64
sibsp             int64
parch             int64
…
adult_male         bool
deck           category
embark_town      object
alive            object
alone              bool
dtype: object
```

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   survived    891 non-null    int64
 1   pclass      891 non-null    int64
 …
 13  alive       891 non-null    object
 14  alone       891 non-null    bool
dtypes: bool(2), category(2),
float64(2), int64(4), object(5)
memory usage: 80.7+ KB
```

Estadísticos descriptivos básicos

- Resumen de variables categóricas: frecuencias y proporciones.
- Para variables continuas: media, mediana, mínimo, máximo, cuartiles.
- Permite detectar rangos anómalos y escala de las variables.

```
# Estadísticos para variables numéricas
df.describe()

# Estadísticos para variables categóricas
df['sex'].value_counts()
df['embarked'].value_counts(dropna=False)
```

Distribuciones y frecuencias

- Analizar cómo se distribuyen los valores de una variable.
- Histogramas (*histograms*) para variables continuas.
- Diagramas de barras (*bar plots*) para variables categóricas.

```
# Histograma de edad
sns.histplot(df['age'].dropna(), bins=30, kde=True)
plt.title('Distribución de edad')
plt.show()

# Frecuencia de clase de pasajero
sns.countplot(x='class', data=df)
plt.title('Pasajeros por clase')
plt.show()
```

Correlaciones entre variables

- Mide la relación lineal entre dos variables numéricas.
- Tiene valores entre -1 y 1. La cercanía a ±1 indica fuerte asociación.
- Útil para detectar redundancias y relaciones relevantes.
- No implica causalidad.

```
# Matriz de correlación
corr = df[['age', 'fare', 'pclass', 'sibsp', 'parch']].corr()
print(corr)

# Mapa de calor
sns.heatmap(corr, annot=True, cmap='coolwarm')
plt.title('Correlaciones numéricas')
plt.show()
```

Visualización inicial de datos

- Gráficos simples que resumen múltiples aspectos de los datos.
- Diagramas de dispersión (*scatter plots*) para pares de variables.
- Matriz de dispersión (*pair plots*) para una vista general multivariada.
- Diagramas de caja (*box plots*) para comparar distribuciones entre categorías.

```
# Dispersión: edad vs tarifa coloreada por supervivencia
sns.scatterplot(x='age', y='fare', hue='survived', data=df)
plt.title('Relación edad-tarifa según supervivencia')
plt.show()

# Pairplot (solo algunas columnas)
sns.pairplot(df[['age', 'fare', 'survived']].dropna(),hue='survived')
plt.show()
```

Identificación de valores faltantes

- Identificar valores faltantes puede ser complejo en datos reales.
- Los datos faltantes pueden tomar varias formas.
- Celdas vacías, marcadores como “N/A” o “-999” o datos mal ingresados.

```
df.isnull()                          # Presencia de valores faltantes
faltantes = df.isnull().sum()        # Conteo de valores faltantes
faltantes

df['age'].isnull()                   # Valores faltantes por columna
df[df['age'].isnull()]               # Registros con valores faltantes
df[df['deck'].isnull()]
```

Identificación de valores atípicos I

- También son conocidos por el anglicismo *outliers*.
- Valores inusuales que difieren significativamente del resto de los datos.
- **Causas:** errores de medición, errores de procesamiento y anomalías reales.

```
# Carga de librerías
from scipy import stats
import numpy as np

# Cálculo de puntajes Z, cuidado con los valores faltantes
z_scores = np.abs(stats.zscore(df2['fare']))
df2[z_scores > 3]
```

Identificación de valores atípicos II

- **IQR:** Detecta valores fuera de 1.5 veces el rango intercuartil (Q1 - Q3).

```
# Carga de librerías
import numpy as np

# Cálculo de cuartiles y el rango
Q1 = np.percentile(df['fare'], 25)
Q3 = np.percentile(df['fare'], 75)
IQR = Q3 - Q1

# Detección
df[(df['fare'] < (Q1 - 1.5 * IQR)) | (df['fare'] > (Q3 + 1.5 * IQR))]
```

Identificación de valores atípicos III

- Visualización de valores atípicos con gráficos:

```
# Carga de librerías
import matplotlib.pyplot as plt

# Gráfico de caja
plt.boxplot(df['fare'])
plt.show()

# Gráfico de dispersión
plt.scatter(range(len(df['fare'])), df['fare'])
plt.show()
```

Limpieza de datos

- Los datos rara vez vienen limpios y listos para usar.
- Problemas comunes que afectan la calidad:
  - Valores faltantes: Datos incompletos que pueden sesgar análisis.
  - *Outliers*: Valores atípicos que distorsionan resultados.
  - Formato inconsistente: Datos mal estructurados o inconsistentes.
  - Duplicados: Registros repetidos que hacen crecer el conjunto de datos.
  - Datos redundantes: Información que no aporta valor.
- Estos problemas pueden sesgar el análisis y generar modelos erróneos.

Preservación de datos originales

- Las técnicas para manejar datos faltantes modifican el conjunto de datos.
- Si el resultado no es satisfactorio, no se puede volver al punto de partida.
- La solución es crear una copia de trabajo antes de cualquier manipulación.

```
# Copia del conjunto original
df2 = df.copy(deep = True)
```

Técnicas para manejar valores faltantes I

- **Eliminación de registros:** Se usa solo si los valores faltantes constituyen una pequeña fracción del conjunto de datos.

```
# Eliminación de datos faltantes
df2.dropna(inplace=True)
# Comprobación de datos faltantes
df2.isnull().sum( )
```

- **Desventaja:** Puede llevar a la pérdida de información valiosa.

Técnicas para manejar valores faltantes II

- **Sustituir datos faltantes con valores estadísticos:**
  - Media – Para datos continuos.
  - Mediana – Para datos ordinales.
  - Moda – Para datos categóricos.

```
# Imputación de datos
df2.fillna({'age': df2['age'].mean()}, inplace = True)
df2.fillna({'age': df2['age'].median()}, inplace = True)
df2.fillna({'age': df2['age’].mode()}, inplace = True)
```

- **Desventaja:** Puede reducir la varianza y afectar la correlación.

Técnicas para manejar valores faltantes III

- **Sustituir datos faltantes por un valor constante:** Útil cuando se puede hacer una suposición razonable sobre el valor.

```
# Imputación de datos
df2.fillna({'age': 0}, inplace = True)
```

- **Desventaja:** Puede introducir sesgo si el valor elegido no es representativo.

Técnicas para manejar datos atípicos I

- **Eliminación:** Eliminar valores atípicos solo si están claramente mal registrados o medidos.

```
# Carga de librerías
from scipy import stats
import numpy as np

# Cálculo de puntajes Z, cuidado con los valores faltantes
z_scores = np.abs(stats.zscore(df2['fare']))
df2[z_scores <= 3]
```

- **Desventaja:** Puede causar pérdida de información si el valor no es un error.

Técnicas para manejar datos atípicos II

- **Transformación:** Las transformaciones pueden reducir el efecto de los valores extremos. Por ejemplo: Transformación logarítmica.

```
# Carga de librerías
import matplotlib.pyplot as plt
import numpy as np

# Transformación de datos
df['fare'] = np.log(df['fare'])

# Gráfico de dispersión
plt.scatter(range(len(df['fare'])), df['fare'])
plt.show()
```

- **Desventaja:** Puede alterar la interpretación de los datos.

Formato inconsistente

- Los datos categóricos suelen presentar variaciones en su escritura.
- Mayúsculas/minúsculas, espacios, abreviaciones, tipeo erróneo.
- Deben unificarse para que una misma categoría no sea tratada como varias.

```
# Estandarizar texto (minúsculas, sin espacios)
df['sex'] = df['sex'].str.lower().str.strip()

# Unificar categorías
df['embark_town'] = df['embark_town'].replace({'S': 'Southampton',
                                               'C': 'Cherbourg',
                                               'Q': 'Queenstown'})
```

Datos duplicados

- Registros idénticos (o casi idénticos) que aparecen más de una vez.

```
# Identificar filas duplicadas
duplicados = df.duplicated()
print("Filas duplicadas:", duplicados.sum())

# Mostrar duplicados (si existen)
df[duplicados]

# Eliminar duplicados (conservar el primero)
df.drop_duplicates(inplace=True)

# Eliminar duplicados considerando solo algunas columnas
df.drop_duplicates(subset=['pclass', 'sex', 'age'], keep='last',
inplace=True)
```

Datos redundantes

- Variables que expresan la misma información de forma diferente.

```
# Identificar redundancias mediante correlación o inspección
df[['survived', 'alive']].head()

# Eliminar variable redundante
df.drop('alive', axis=1, inplace=True)

# Otras redundancias comunes:
# - 'class' (First, Second, Third) y 'pclass' (1,2,3)
# - 'adult_male' derivado de 'age' y 'sex'
df.drop('class', axis=1, inplace=True)
```