



UNIVERSIDAD  
CATÓLICA  
SEDES SAPIENTIAE

# Inteligencia Artificial

Ing. Juancarlos Santana Huamán  
[jsantana@ucss.edu.pe](mailto:jsantana@ucss.edu.pe)



# Análisis de componentes principales (Principal Component Analysis PCA)

- El PCA (Análisis de Componentes Principales) es una técnica de reducción de dimensionalidad utilizada en el análisis de datos y el aprendizaje automático. Permite reducir el número de características en un conjunto de datos, conservando la información más importante. Transforma las características originales en nuevas características; estas nuevas características no se superponen entre sí y las primeras conservan la mayoría de las diferencias importantes encontradas en los datos originales.
- El PCA se utiliza comúnmente para el preprocesamiento de datos con algoritmos de aprendizaje automático. Ayuda a eliminar la redundancia, mejorar la eficiencia computacional y facilitar la visualización y el análisis de datos, especialmente al trabajar con datos de alta dimensión.



# Cómo funciona el análisis de componentes principales

- El análisis de componentes principales (PCA) utiliza álgebra lineal para transformar los datos en nuevas características llamadas componentes principales. Estos se obtienen calculando vectores propios (direcciones) y valores propios (importancia) a partir de la matriz de covarianza. El PCA selecciona los componentes principales con los valores propios más altos y proyecta los datos sobre ellos para simplificar el conjunto de datos.

**Nota:** Prioriza las direcciones donde los datos varían más porque mayor variación = más información útil.

- Imagina que observas una nube desordenada de puntos de datos, como estrellas en el cielo, y quieres simplificarla. El análisis de componentes principales (PCA) te ayuda a encontrar los ángulos más importantes para observar esta nube y así no perderte los patrones principales.



# Cómo funciona el análisis de componentes principales

## Paso 1: Estandarizar los datos

- Diferentes características pueden tener unidades y escalas diferentes, como el salario frente a la edad. Para compararlas de forma justa, el PCA primero estandariza los datos, haciendo que cada característica tenga:
  - Una media de 0
  - Una desviación estándar de 1

$$Z = \frac{\text{incógnita} - \text{micras}}{\sigma}$$

- Dónde:
  - micras es la media de características independientes:  
 $\text{micras} = \{\text{micras}_1, \text{micras}_2, \dots, \text{micras}_{\text{metro}}\}$
  - $\sigma$  es la desviación estándar de las características independientes:  
 $\sigma = \{\sigma_1, \sigma_2, \dots, \sigma_{\text{metro}}\}$





# Cómo funciona el análisis de componentes principales

## Paso 2: Calcular la matriz de covarianza

- A continuación, el PCA calcula la matriz de covarianza para ver cómo se relacionan las características entre sí, independientemente de si aumentan o disminuyen juntas. La covarianza entre dos características...incógnita<sub>1</sub> y incógnita<sub>2</sub> es:

$$cov(x_1, x_2) = \frac{\sum_{y=1}^{norte} (x_{1i} - \bar{x}_1)(x_{2i} - \bar{x}_2)}{n - 1}$$

- Dónde:
  - $\bar{x}_1$  y  $\bar{x}_2$  son los valores medios de las características  $x_1$  y  $x_2$ .
  - $norte$  es el número de puntos de datos
- El valor de la covarianza puede ser positivo, negativo o cero.



# Cómo funciona el análisis de componentes principales

## Paso 3: Encuentra los componentes principales

- PCA identifica **nuevos ejes** donde los datos se dispersan más:
  - **Primer componente principal (CP1)**: La dirección de máxima varianza (mayor dispersión).
  - **2do componente principal (PC2)**: La siguiente mejor dirección, *perpendicular a PC1* y así sucesivamente.
- Estas direcciones provienen de los **vectores propios** de la matriz de covarianza y su importancia se mide mediante **los valores propios**. Para una matriz cuadrada A, un **vector propio** X (un vector distinto de cero) y su **valor propio** correspondiente  $\lambda$  satisfacen:

$$Una \ X = \lambda X$$

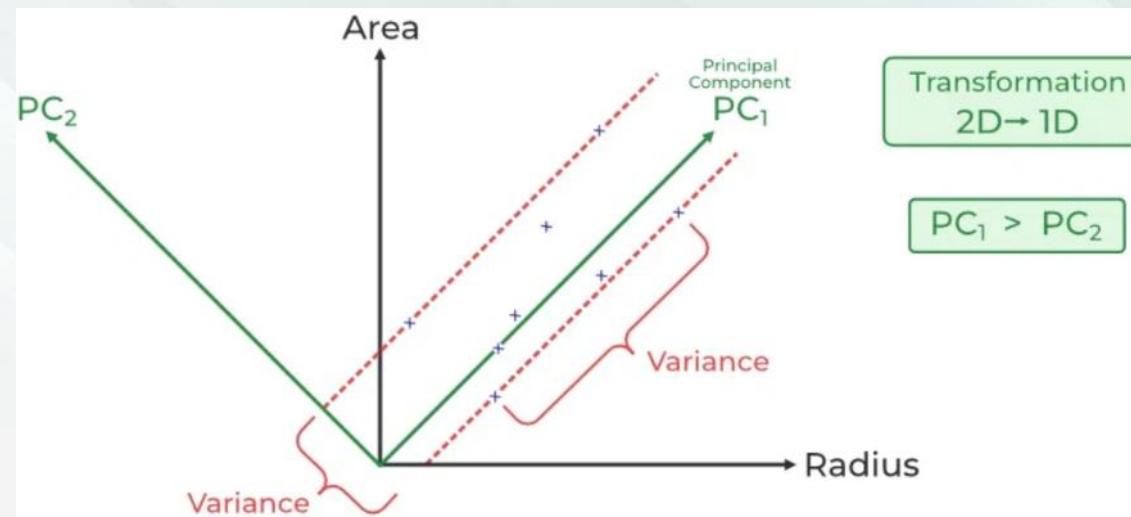
- Esto significa:
  - Cuando A actúa sobre X, solo estira o encoge X según el escalar  $\lambda$ .
  - La dirección de X permanece sin cambios, por lo tanto, los vectores propios definen "direcciones estables" de A.
- Los valores propios ayudan a clasificar estas direcciones por importancia.



# Cómo funciona el análisis de componentes principales

## Paso 4: Seleccione las direcciones principales y transforme los datos

- Tras calcular los autovalores y vectores propios, el análisis de componentes principales (PCA) los clasifica según la cantidad de información que capturan. A continuación:
- **Seleccione los k componentes principales** que capturan la mayor parte de la variación, como el 95 %.
- **Transforme el conjunto de datos original** proyectándolo sobre estos componentes superiores.
- Esto significa que reducimos el número de características (dimensiones) mientras mantenemos los patrones importantes en los datos.



# Cómo funciona el análisis de componentes principales

- En la imagen superior, el conjunto de datos original presenta dos características, "Radio" y "Área", representadas por los ejes negros. El análisis de componentes principales (PCA) identifica dos nuevas direcciones: **PC<sub>1</sub>** y **PC<sub>2</sub>**, que son los **componentes principales**.
- Estos nuevos ejes son versiones rotadas de los originales. **PC<sub>1</sub>** captura la máxima varianza de los datos, lo que significa que contiene la mayor cantidad de información, mientras que **PC<sub>2</sub>** captura la varianza restante y es perpendicular a **PC<sub>1</sub>**.
- La dispersión de datos es mucho mayor a lo largo de **PC<sub>1</sub>** que a lo largo de **PC<sub>2</sub>**. Por ello, se elige **PC<sub>1</sub>** para la reducción de dimensionalidad. Al proyectar los puntos de datos (cruces azules) en **PC<sub>1</sub>**, **transformamos eficazmente los datos 2D en 1D** y conservamos la mayor parte de la estructura y los patrones importantes.

# Implementación del análisis de componentes principales en Python

- Por lo tanto, el PCA utiliza una transformación lineal que se basa en preservar la máxima varianza en los datos utilizando el menor número de dimensiones. Implica los siguientes pasos:

## Paso 1: Importar las bibliotecas necesarias

- Importamos las bibliotecas necesarias como pandas , numpy , scikit learn , seaborn y matplotlib para visualizar los resultados.

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
```



# Implementación del análisis de componentes principales en Python

## Paso 2: Creación de un conjunto de datos de muestra

- Creamos un pequeño conjunto de datos con tres características: Altura, Peso, Edad y Género.

```

data = {
    'Height': [170, 165, 180, 175, 160, 172, 168, 177, 162, 158],
    'Weight': [65, 59, 75, 68, 55, 70, 62, 74, 58, 54],
    'Age': [30, 25, 35, 28, 22, 32, 27, 33, 24, 21],
    'Gender': [1, 0, 1, 1, 0, 1, 0, 1, 0, 0] # 1 = Male, 0 = Female
}
df = pd.DataFrame(data)
print(df)

```

	Height	Weight	Age	Gender
0	170	65	30	1
1	165	59	25	0
2	180	75	35	1
3	175	68	28	1
4	160	55	22	0
5	172	70	32	1
6	168	62	27	0
7	177	74	33	1
8	162	58	24	0
9	158	54	21	0



# Implementación del análisis de componentes principales en Python

## Paso 3: Estandarización de los datos

- Dado que las características tienen diferentes escalas (altura y edad), estandarizamos los datos. Esto hace que todas las características tengan una media de 0 y una desviación estándar de 1, de modo que ninguna característica predomine simplemente por sus unidades.

```
X = df.drop('Gender', axis=1)
```

```
y = df['Gender']
```

```
scaler = StandardScaler()
```

```
X_scaled = scaler.fit_transform(df)
```



# Implementación del análisis de componentes principales en Python

## Paso 4: Aplicación del algoritmo PCA

- Reducimos los datos de tres características a dos nuevas características llamadas componentes principales. Estos componentes capturan la mayor parte de la información original, pero en menos dimensiones.
- Dividimos los datos en un 70% de conjuntos de entrenamiento y un 30% de conjuntos de prueba.
- Entrenamos un modelo de regresión logística en los datos de entrenamiento reducidos y predecimos etiquetas de género en el conjunto de prueba.

```
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

X_train, X_test, y_train, y_test = train_test_split(X_pca, y, test_size=0.3, random_state=42)

model = LogisticRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
```



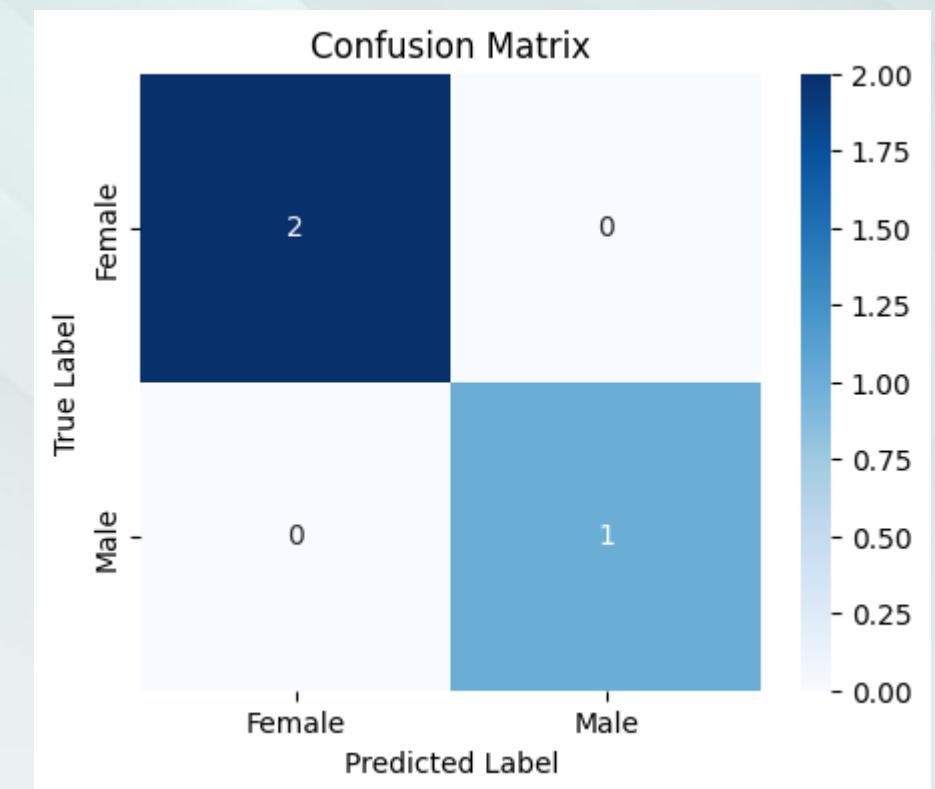
# Implementación del análisis de componentes principales en Python

## Paso 5: Evaluación con la Matriz de Confusión

- La matriz de confusión compara las etiquetas reales con las predichas. Esto facilita identificar dónde las predicciones fueron correctas o incorrectas.

```
cm = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(5,4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
xticklabels=['Female', 'Male'], yticklabels=['Female', 'Male'])
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Confusion Matrix')
plt.show()
```



# Implementación del análisis de componentes principales en Python

## Paso 6: Visualización del resultado del PCA

```
y_numeric = pd.factorize(y)[0]

plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
plt.scatter(X_scaled[:, 0], X_scaled[:, 1], c=y_numeric, cmap='coolwarm', edgecolor='k', s=80)
plt.xlabel('Original Feature 1')
plt.ylabel('Original Feature 2')
plt.title('Before PCA: Using First 2 Standardized Features')
plt.colorbar(label='Target classes')

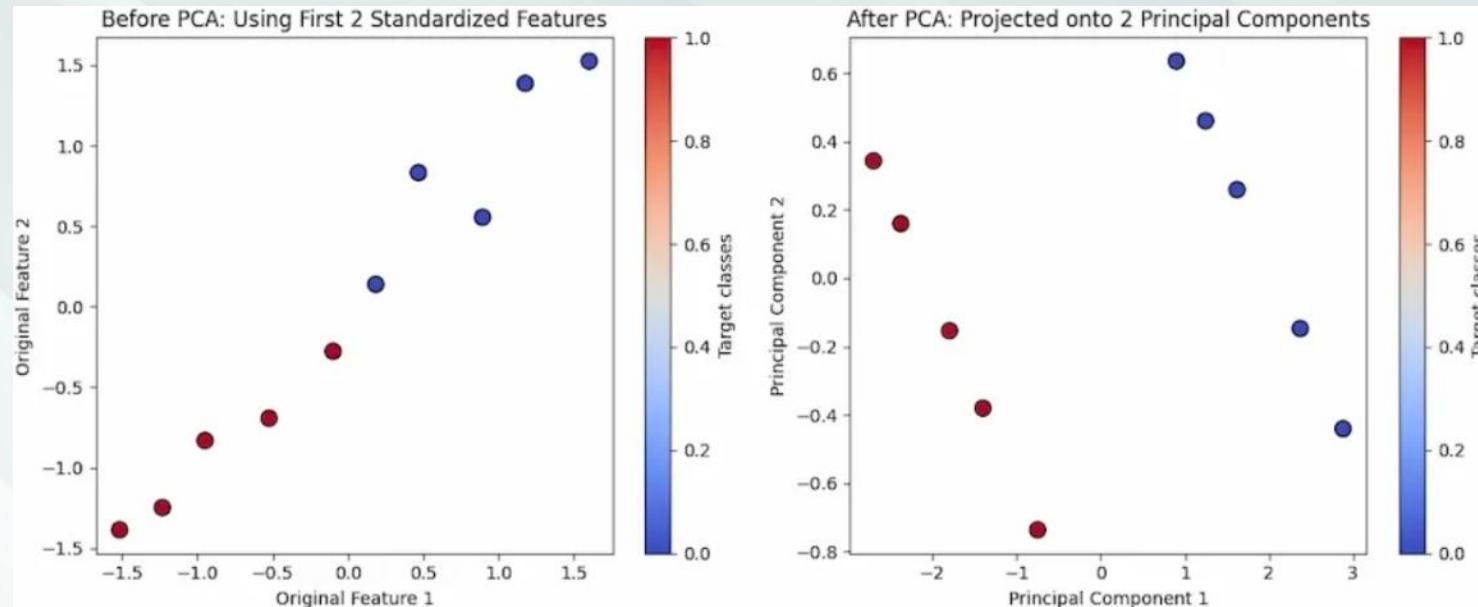
plt.subplot(1, 2, 2)
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=y_numeric, cmap='coolwarm', edgecolor='k', s=80)
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('After PCA: Projected onto 2 Principal Components')
plt.colorbar(label='Target classes')

plt.tight_layout()
plt.show()
```





# Implementación del análisis de componentes principales en Python



- **Gráfico izquierdo antes del PCA:** Muestra los datos estandarizados originales, graficados utilizando las dos primeras características. No se garantiza una separación clara entre clases, ya que se trata de dimensiones de entrada sin procesar.
- **Gráfico derecho después del PCA:** Muestra los datos transformados utilizando los dos componentes principales principales . Estos nuevos componentes capturan la varianza máxima, mostrando a menudo una mejor separación y estructura de clases, lo que facilita el análisis o modelado.



# Implementación del análisis de componentes principales en Python

## Ventajas del análisis de componentes principales

- Manejo de multicolinealidad:** crea nuevas variables no correlacionadas para abordar problemas cuando las características originales están altamente correlacionadas.
- Reducción de ruido:** elimina componentes con baja variación y mejora la claridad de los datos.
- Compresión de datos:** Representa datos con menos componentes que reducen las necesidades de almacenamiento y aceleran el procesamiento.
- Detección de valores atípicos:** identifica puntos de datos inusuales mostrando cuáles se desvían significativamente en el espacio reducido.





# Implementación del análisis de componentes principales en Python

## Desventajas del análisis de componentes principales

1. **Desafíos de interpretación:** Los nuevos componentes son combinaciones de variables originales que pueden ser difíciles de explicar.
2. **Sensibilidad de escalamiento de datos:** requiere un escalamiento adecuado de los datos antes de la aplicación o los resultados pueden ser engañosos.
3. **Pérdida de información:** al reducir las dimensiones se puede perder información importante si se mantienen muy pocos componentes.
4. **Supuesto de linealidad:** funciona mejor cuando las relaciones entre las variables son lineales y puede presentar dificultades con datos no lineales.
5. **Complejidad computacional:** puede ser lento y consumir muchos recursos en conjuntos de datos muy grandes.
6. **Riesgo de sobreajuste:** utilizar demasiados componentes o trabajar con un conjunto de datos pequeño puede generar modelos que no se generalicen bien.



# Análisis de componentes independientes (ICA)

- El Análisis de Componentes Independientes (ICA) es una técnica potente y versátil para el análisis de datos, que ofrece una perspectiva única para la exploración y extracción de patrones ocultos en conjuntos de datos complejos. En esencia, el ICA es un método de procesamiento de señales que busca separar un conjunto de señales mixtas en componentes estadísticamente independientes, lo que proporciona información valiosa y aplicaciones en diversos dominios.

## **La importancia del análisis de componentes independientes (ICA)**

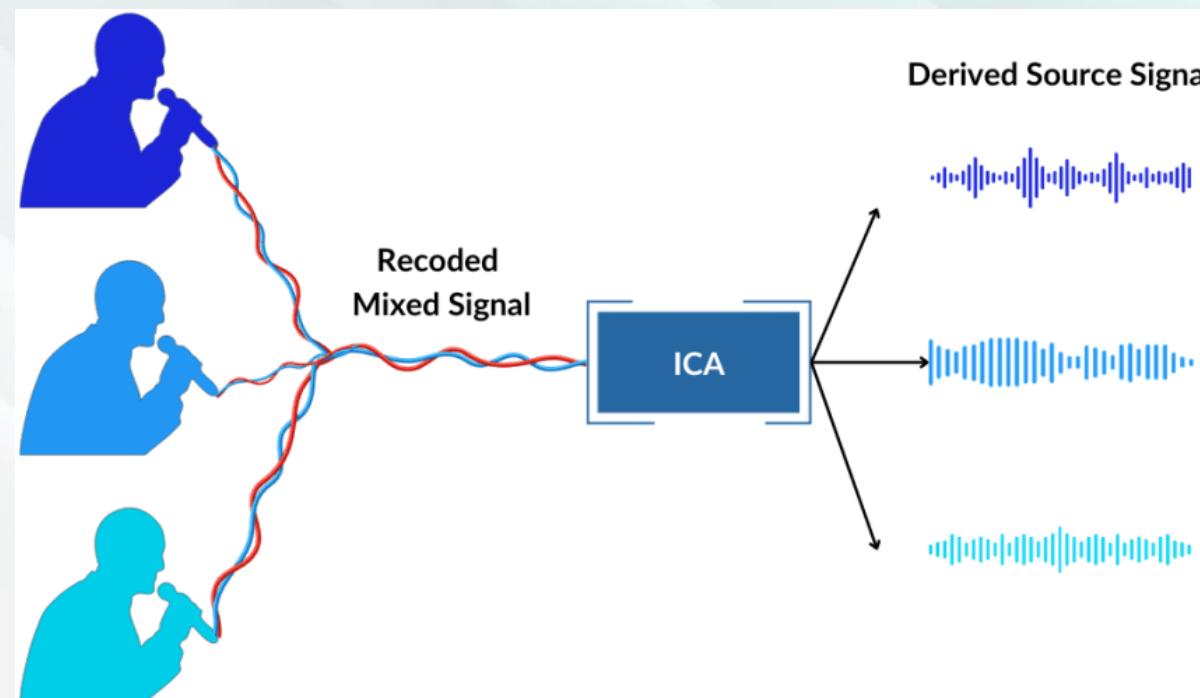
- En un mundo inundado de datos, la capacidad de descubrir información significativa oculta en un mar de ruido es fundamental. El ICA desempeña un papel fundamental en este esfuerzo, permitiéndonos identificar las fuentes subyacentes de los datos observados, incluso cuando no se comprende completamente el proceso de mezcla. Ya sea en el procesamiento de imágenes, el reconocimiento de voz, las finanzas o la investigación médica, el ICA es una herramienta fundamental para desentrañar estructuras de datos complejas.





# El problema de la separación ciega de fuentes

- Una característica distintiva del ICA es su enfoque en la separación ciega de fuentes. Este problema surge cuando solo tenemos acceso a las señales mixtas y necesitamos aplicar ingeniería inversa a las fuentes sin conocer previamente sus propiedades. Esta desafiante tarea es similar a resolver un rompecabezas con piezas que podrían haber sido reorganizadas o transformadas, lo que convierte al ICA en una esperanza para investigadores y analistas que se enfrentan a estos escenarios.



# Compreensión del análisis de componentes independientes (ICA)

## Asunción de la Independencia

- El Análisis de Componentes Independientes se basa en la suposición de que las señales observadas son una combinación lineal de señales fuente estadísticamente independientes. Esta suposición es crucial, ya que permite al ICA aprovechar las dependencias estadísticas presentes en los datos para recuperar las fuentes.

## Modelo de mezcla lineal

- El ICA se basa en el modelo de mezcla lineal, que se puede representar como  $X = AS$  Donde:
  - X son los datos observados (señales mixtas).
  - A es la matriz de mezcla, que describe cómo se combinan las fuentes.
  - S es la matriz fuente, que representa los componentes independientes que se pretenden extraer. El objetivo del ICA es estimar la matriz de mezcla A y recuperar la matriz fuente S.





# ICA vs. PCA: Diferencias clave

- Ortogonalidad vs. Independencia. El Análisis de Componentes Principales (ACP) y el Análisis de Componentes Independientes (ACI) se comparan a menudo, pero sus principios fundamentales difieren significativamente. El ACP busca componentes ortogonales (no correlacionados) en los datos, mientras que el ACI busca componentes estadísticamente independientes. Esta distinción hace que el ACI sea ideal para descubrir fuentes ocultas en datos mixtos.

## Separación ciega de fuentes

- El ICA se utiliza principalmente para la separación ciega de fuentes, lo que significa que puede gestionar situaciones en las que el número de fuentes y sus propiedades estadísticas se desconocen a priori. Por el contrario, el PCA se utiliza para la reducción de dimensionalidad y la decorrelación sin centrarse en la separación de fuentes.





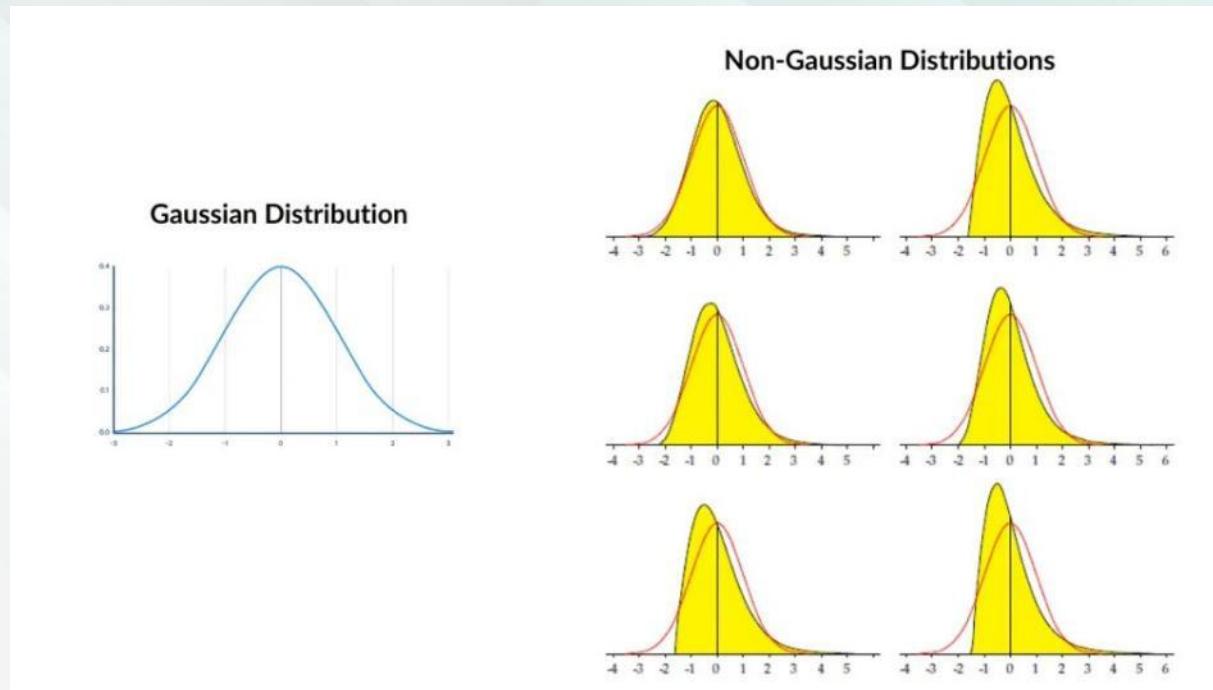
# Aplicaciones del análisis de componentes independientes (ICA)

- **Procesamiento de imágenes:** en el procesamiento de imágenes, se puede emplear ICA para separar una imagen en sus componentes subyacentes, como texturas, patrones o incluso las fuentes de ruido, lo que permite aplicaciones como mejora de imágenes, eliminación de ruido y reconocimiento de objetos.
- **Separación de Señales:** El ICA se utiliza ampliamente en el procesamiento de señales de audio para separar fuentes de audio mixtas en situaciones como el problema de las fiestas de cóctel. Permite recuperar voces o instrumentos individuales de una mezcla de señales de sonido.
- **Imágenes médicas:** La ICA es crucial en las imágenes médicas, en particular en la resonancia magnética funcional (fMRI). Ayuda a separar y analizar los patrones de activación cerebral, lo que facilita el estudio de los procesos cognitivos y los trastornos neurológicos.
- **Análisis de datos financieros:** en finanzas, el ICA se utiliza para identificar factores latentes que influyen en los movimientos de los precios de las acciones, lo que ayuda a los comerciantes y analistas a descubrir anomalías y tendencias ocultas del mercado.
- Comprender estos conceptos básicos y las diferencias entre el ICA y el PCA es esencial para comprender los fundamentos del Análisis de Componentes Independientes. En las siguientes secciones, exploraremos los fundamentos matemáticos del ICA y las técnicas utilizadas para extraer componentes independientes de datos mixtos.

# Fundamentos matemáticos del análisis de componentes independientes (ICA) - Distribuciones de probabilidad

## Distribuciones gaussianas y no gaussianas

- El Análisis de Componentes Independientes asume que las señales fuente son estadísticamente independientes. Para facilitar esta independencia, el ICA funciona mejor cuando las fuentes siguen distribuciones de probabilidad no gaussianas. La no gaussianidad es crucial porque implica falta de linealidad y permite al ICA aprovechar la independencia estadística.



# Fundamentos matemáticos del análisis de componentes independientes (ICA) - Distribuciones de probabilidad

## Estimación de máxima verosimilitud

- El objetivo de la ICA es encontrar la matriz de mezcla óptima A y la matriz de fuentes S que maximizan la probabilidad de los datos observados X, dadas las fuentes estimadas S. Esto se puede expresar como:
- $P(X|A,S)$
- El ICA busca encontrar A y S que maximicen esta función de verosimilitud. Esto se logra mediante diversos algoritmos y técnicas que refinan iterativamente las estimaciones de A y S para aumentar la verosimilitud de los datos observados.

# Fundamentos matemáticos del análisis de componentes independientes (ICA) - Distribuciones de probabilidad

## Encontrar componentes independientes

- **Blanqueo.** Uno de los pasos iniciales del ICA consiste en blanquear los datos observados. Este blanqueamiento transforma los datos en un espacio donde los componentes no están correlacionados y presentan varianzas unitarias. Esto simplifica el problema del ICA y lo hace más fácil de separar.
- **Algoritmo de deflación.** El algoritmo de deflación es un componente crucial del ICA. Extrae los componentes independientes uno a uno. Cada iteración estima y elimina un componente de los datos, lo que facilita la estimación de los componentes subsiguientes. El proceso continúa hasta que se extraen todos los componentes independientes.





# Los 3 principales algoritmos y técnicas de análisis de componentes independientes (ICA)

**FastICA.** Es un algoritmo ICA popular y eficiente que busca encontrar componentes independientes maximizando la no gaussianidad. Se basa en un esquema de iteración de punto fijo y puede trabajar con mezclas lineales y no lineales.

- Características principales
  - FastICA maximiza la negentropía, una medida de no gaussianidad, para separar componentes independientes.
  - Emplea un algoritmo de deflación para extraer componentes uno a uno.
  - FastICA es computacionalmente eficiente y se utiliza ampliamente en diversas aplicaciones, incluido el procesamiento de imágenes y voz.

**Infomax.** Es un algoritmo ICA inspirado en la teoría de la información. Busca maximizar la información mutua entre los componentes independientes estimados. Se utiliza frecuentemente en aplicaciones como la separación ciega de fuentes y el entrenamiento de redes neuronales.

- Características principales
  - Infomax maximiza la información mutua, que mide la independencia estadística de los componentes.
  - Se basa en arquitecturas de redes neuronales y reglas de aprendizaje para ajustar iterativamente la matriz de mezcla y estimar componentes independientes.





# Los 3 principales algoritmos y técnicas de análisis de componentes independientes (ICA)

**JADE (Diagonalización aproximada conjunta de automatrices).** Es un algoritmo ICA diseñado para funcionar con fuentes no gaussianas. Funciona diagonalizando conjuntamente las matrices cumulantes de cuarto orden de los datos observados. Este enfoque resulta conveniente en escenarios donde las fuentes presentan un comportamiento supergaussiano.

- Características principales
  - JADE es adecuado para separar fuentes con propiedades estadísticas de orden superior.
  - Proporciona una alternativa a los enfoques basados en curtosis y es robusto a la no gaussianidad.



# Comparación de diferentes algoritmos

- Los algoritmos de ICA varían en sus enfoques, ventajas y limitaciones. La elección del algoritmo adecuado depende de las características específicas de los datos y de los objetivos del análisis. Al seleccionar un algoritmo, se deben considerar aspectos como la distribución de la fuente, la dimensionalidad de los datos y la eficiencia computacional.
- En la práctica, es habitual experimentar con diferentes algoritmos de ICA y comparar su rendimiento en el conjunto de datos dado. La elección del algoritmo puede influir significativamente en la calidad de los componentes independientes extraídos y en el éxito del análisis.
- Comprender estos algoritmos y técnicas de ICA es esencial para quienes aplican el ICA al análisis de datos reales. La selección del algoritmo más adecuado debe basarse en las características específicas de los datos y los objetivos del análisis. En la siguiente sección, profundizaremos en la implementación práctica del ICA, incluyendo el preprocesamiento de datos, la interpretación de componentes y los errores comunes que se deben evitar.





# Preprocesamiento de datos

- **Centrado de datos:** Antes de aplicar el ICA, es fundamental centrar los datos restando la media de cada variable. Esto garantiza que la matriz de mezcla capture las relaciones entre las señales sin sesgos.
- **Blanqueamiento de datos:** El blanqueamiento de datos es un paso crucial que los transforma en un espacio donde los componentes no están correlacionados y presentan varianzas unitarias. El blanqueamiento simplifica el problema del ICA y lo hace más fácil de separar.
- **Reducción de dimensionalidad (opcional):** en conjuntos de datos de alta dimensión, las técnicas de reducción de dimensionalidad como el análisis de componentes principales (PCA) pueden reducir la complejidad computacional y al mismo tiempo preservar la información más relevante.



# Software y bibliotecas para el análisis de componentes independientes (ICA)

- Existen diversos paquetes de software y bibliotecas para ICA, lo que lo hace accesible a una amplia gama de usuarios. Algunas herramientas populares incluyen MATLAB, scikit-learn de Python y bibliotecas independientes específicas para ICA, como FastICA. Estos recursos proporcionan implementaciones listas para usar de algoritmos de ICA, lo que reduce la necesidad de que los usuarios desarrollen su código desde cero.
- La implementación práctica del ICA implica un preprocesamiento cuidadoso de los datos , la selección del número adecuado de componentes, la interpretación de los resultados y la solución de problemas comunes. Es fundamental adaptar la implementación a las características específicas de los datos y a los objetivos del análisis. En la siguiente sección, exploraremos aplicaciones reales del ICA, demostrando su versatilidad en diversos ámbitos.





# Implementar el análisis de componentes independientes (ICA) en Python

- Para realizar el Análisis de Componentes Independientes (ICA) en Python, se pueden usar diversas bibliotecas y paquetes que ofrecen implementaciones de ICA. Una de las bibliotecas más utilizadas para ICA en Python es `scikit-learn`, que ofrece una forma sencilla y práctica de realizarlo. Aquí tienes una guía paso a paso sobre cómo usar `scikit-learn` para aplicar ICA en Python:

`pip install scikit-learn`

```
from sklearn.decomposition import FastICA
import numpy as np
import matplotlib.pyplot as plt

# Generación de señales aleatoriamente
np.random.seed(0)
n_samples = 200
time = np.linspace(0, 8, n_samples)
s1 = np.sin(2 * time) # Señal 1
s2 = np.sign(np.sin(3 * time)) # Señal 2
s3 = np.random.randn(n_samples) # Señal 3
S = np.c_[s1, s2, s3] # Matriz combinada
A = np.array([[1, 1, 1], [0.5, 2, 1], [1.5, 1, 2]])
X = np.dot(S, A.T) # Señales combinadas
```

```
# Aplicación ICA
ica = FastICA(n_components=3)
independent_components = ica.fit_transform(X)

# Visualización de los componentes individuales
plt.figure(figsize=(12, 6))

plt.subplot(4, 1, 1)
plt.title("Señales Originales")
plt.plot(S)

plt.subplot(4, 1, 2)
plt.title("Señales Combinadas")
plt.plot(X)

plt.subplot(4, 1, 3)
plt.title("Componente 1")
plt.plot(independent_components[0])

plt.subplot(4, 1, 4)
plt.title("Componente 2")
plt.plot(independent_components[1])
```

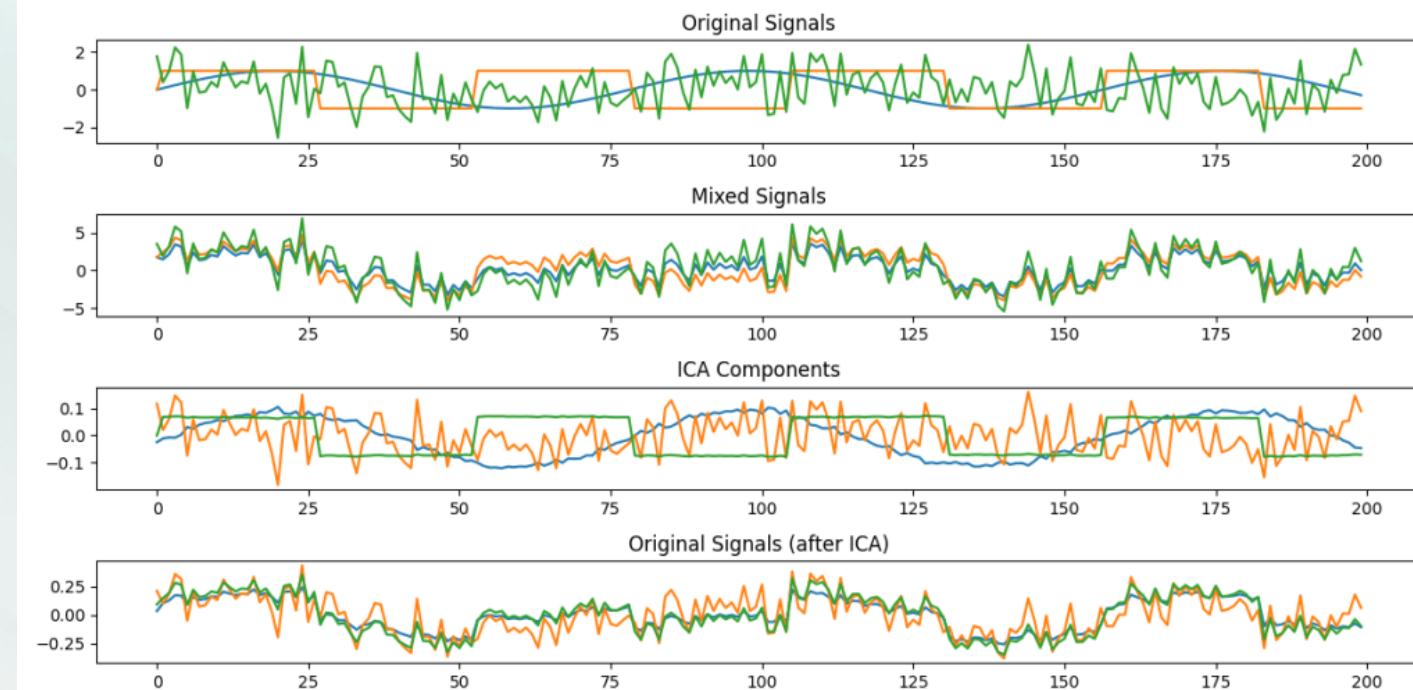


# Implementar el análisis de componentes independientes (ICA) en Python

```
plt.subplot(4, 1, 3)
plt.title("Componentes ICA")
plt.plot(independent_components)
```

```
plt.subplot(4, 1, 4)
plt.title("Señales Originales (Antes ICA)")
reconstructed_signals = np.dot(independent_components, A)
plt.plot(reconstructed_signals)
```

```
plt.tight_layout()
plt.show()
```





# Análisis de componentes independientes (ICA) en el aprendizaje automático

- **Extracción de características:** El ICA puede utilizarse para extraer características independientes y significativas de datos de alta dimensión. Al transformar los datos en un nuevo espacio de características, el ICA puede descubrir patrones subyacentes y reducir la dimensionalidad de los datos, haciéndolos más manejables para los algoritmos de aprendizaje automático. Esto puede ser especialmente útil en tareas como el reconocimiento de imágenes, donde ayuda a identificar características clave.
- **Separación de fuentes ciegas:** El ICA se utiliza para la separación ciega de fuentes, cuyo objetivo es separar señales mixtas en sus fuentes independientes. En el aprendizaje automático, esto puede ser beneficioso para el procesamiento de audio y voz. Por ejemplo, el ICA puede separar las voces de varios hablantes en una grabación de audio, lo que facilita el análisis y la transcripción del habla.
- **Preprocesamiento de datos:** El ICA puede emplearse como paso de preprocesamiento para mejorar la calidad de los datos utilizados en el aprendizaje automático. Ayuda a eliminar ruido y artefactos, mejorando la relación señal-ruido y el rendimiento de los algoritmos de aprendizaje automático posteriores. Esto es especialmente útil en tareas como el análisis de imágenes médicas, donde la eliminación de artefactos es crucial para un diagnóstico preciso.
- **Detección de anomalías:** El ICA puede utilizarse para la detección de anomalías mediante la identificación de desviaciones de los patrones esperados. Se puede aplicar para detectar eventos inusuales en los datos, como la detección de fraudes en transacciones financieras o la detección de fallos en los procesos de fabricación. La capacidad del ICA para identificar componentes independientes lo hace ideal para detectar anomalías que podrían no ajustarse a los patrones típicos.



# Análisis de componentes independientes (ICA) en el aprendizaje automático

- **Reducción de la multicolinealidad:** En el análisis de regresión y los modelos de aprendizaje automático, la multicolinealidad (alta correlación entre variables predictoras) puede afectar el rendimiento y la interpretabilidad del modelo. El ICA puede ayudar a reducir la multicolinealidad extrayendo componentes independientes que capturan información distinta de las características originales.
- **Mejorar la interpretabilidad de los datos:** El ICA puede facilitar la interpretación de los datos al extraer componentes más fáciles de comprender o analizar. Por ejemplo, en neurociencia, el ICA puede revelar fuentes cerebrales independientes asociadas con funciones cognitivas específicas, lo que facilita a los investigadores la interpretación de los resultados.
- **Clasificación y agrupamiento:** En ciertas aplicaciones, los datos transformados mediante ICA pueden incorporarse a algoritmos de aprendizaje automático para su clasificación o agrupamiento . Al reducir la dimensionalidad y centrarse en componentes independientes, ICA puede mejorar el rendimiento de estas tareas.

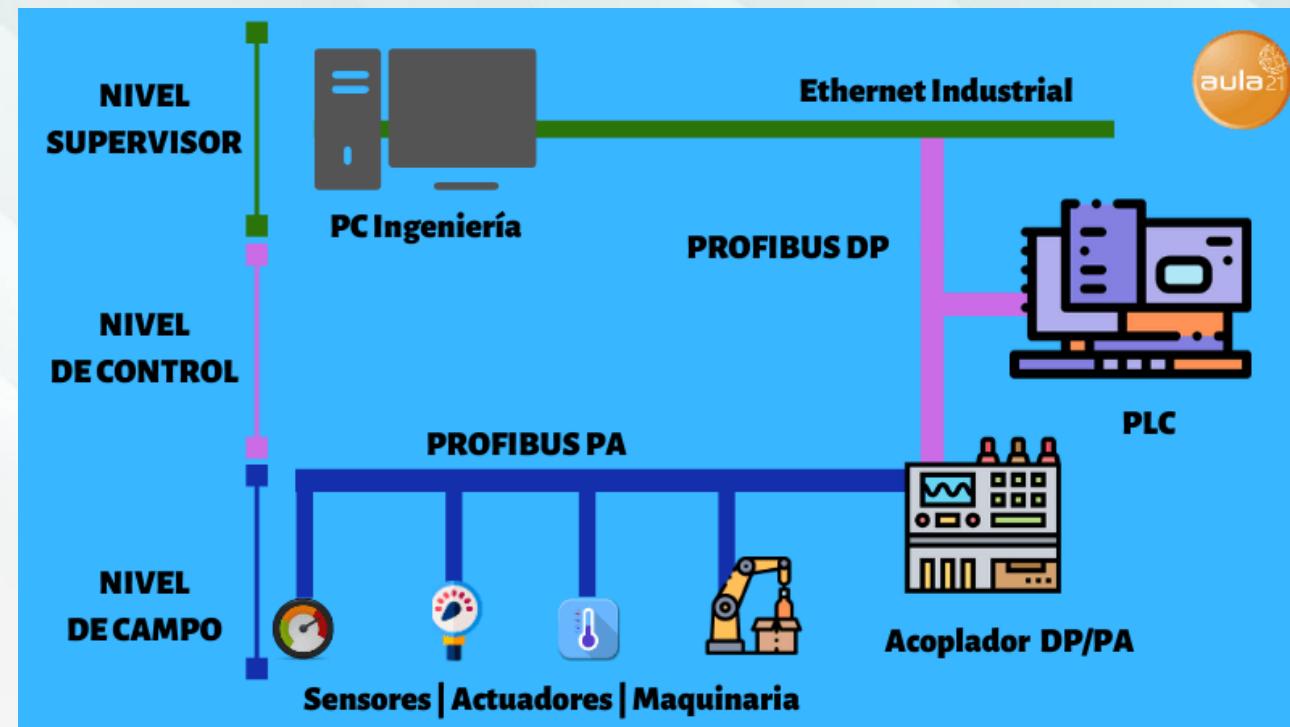
Es importante tener en cuenta que la decisión de usar ICA en el aprendizaje automático depende de la naturaleza de los datos y del problema específico en cuestión. Si bien ICA puede ser una herramienta potente para la extracción de características y el preprocessamiento de datos, no es una solución universal, y su idoneidad debe evaluarse en el contexto de la tarea de aprendizaje automático. Además, la interpretación de los componentes independientes suele ser específica de la aplicación y requiere conocimiento del dominio.





# Ejemplo: Señales de Sensores

- Imaginemos que tenemos 3 sensores que registran señales en una fábrica. Cada sensor recibe una mezcla de:
  - Una señal de vibración de una máquina (fuente 1)
  - Una señal de interferencia eléctrica (fuente 2)
  - Ruido aleatorio (fuente 3)
- Nuestro objetivo es separar estas señales originales.





# Usemos Python para Analizar

```
import numpy as np  
  
import matplotlib.pyplot as plt  
  
from sklearn.decomposition import PCA, FastICA  
  
# Configuración  
  
np.random.seed(42)  
n_muestras = 1000  
tiempo = np.linspace(0, 8, n_muestras)  
  
# Señales originales (fuentes independientes)  
s1 = np.sin(2 * tiempo) # Señal de máquina  
s2 = np.sign(np.sin(3 * tiempo)) # Interferencia (señal cuadrada)  
s3 = np.random.normal(0, 1, n_muestras) # Ruido aleatorio  
  
# Matriz de fuentes (cada fila es una fuente)  
S = np.c_[s1, s2, s3]  
  
# Mezclamos las señales (lo que captan los sensores)  
A = np.array([[0.5, 0.3, 0.2], # Matriz de mezcla  
              [0.4, 0.6, 0.1],  
              [0.2, 0.3, 0.8]])  
  
# Datos observados (lo que miden los sensores)  
X = np.dot(S, A.T)
```



## Visualización de las Señales

```
plt.figure(figsize=(15, 10))

# Señales originales
plt.subplot(3, 1, 1)
for i in range(3):
    plt.plot(tiempo, S[:, i] + i*3, label=f'Fuente {i+1}')
plt.title('Señales Originales (Fuentes)')
plt.legend()

# Señales mezcladas (lo que ven los sensores)
plt.subplot(3, 1, 2)
for i in range(3):
    plt.plot(tiempo, X[:, i] + i*3, label=f'Sensor {i+1}')
plt.title('Señales Mezcladas (Observaciones)')
plt.legend()

plt.tight_layout()
plt.show()
```

## Aplicación de PCA

```
# Aplicamos PCA
pca = PCA(n_components=3)
X_pca = pca.fit_transform(X)

print("Componentes principales (PCA):")
print(pca.components_)
print(f"Varianza explicada: {pca.explained_variance_ratio_}")
```

## Aplicación de ICA

```
# Aplicamos ICA
ica = FastICA(n_components=3, random_state=42)
X_ica = ica.fit_transform(X)

print("\nMatriz de separación (ICA):")
print(ica.components_)
```





```
plt.figure(figsize=(15, 12))

# Señales recuperadas por PCA
plt.subplot(3, 1, 1)
for i in range(3):
    plt.plot(tiempo, X_pca[:, i] + i*3, label=f'Componente PCA {i+1}')
plt.title('Señales Recuperadas por PCA')
plt.legend()

# Señales recuperadas por ICA
plt.subplot(3, 1, 2)
for i in range(3):
    plt.plot(tiempo, X_ica[:, i] + i*3, label=f'Componente ICA {i+1}')
plt.title('Señales Recuperadas por ICA')
plt.legend()

# Comparación con originales (normalizadas)
plt.subplot(3, 1, 3)
for i in range(3):
    # Normalizamos para comparar
    señal_norm = S[:, i] / np.std(S[:, i])
    plt.plot(tiempo, señal_norm + i*3, '--', alpha=0.7, label=f'Original {i+1}')
plt.title('Señales Originales (Referencia)')
plt.legend()

.... plt.tight_layout()
.... plt.show()
```

## Comparación de Resultados



```
# Métricas de evaluación
from sklearn.metrics import correlation_score

print("Correlación con señales originales:")
print("PCA vs Originales:")
for i in range(3):
    corr = np.corrcoef(X_pca[:, i], S[:, i])[0, 1]
    print(f"Componente {i+1}: {corr:.3f}")

print("\nICA vs Originales:")
for i in range(3):
    corr = np.corrcoef(X_ica[:, i], S[:, i])[0, 1]
    print(f"Componente {i+1}: {corr:.3f}")
```

## Análisis de Resultados





# Conclusiones

## PCA (Análisis de Componentes Principales)

- Reduce dimensionalidad manteniendo la máxima varianza
- Ordena componentes por importancia (varianza explicada)
- No separa perfectamente las fuentes originales
- Asume distribución gaussiana

## ICA (Análisis de Componentes Independientes)

- Separa fuentes independientes incluso si están mezcladas
- No requiere que las fuentes sean gaussianas
- Preserva la independencia estadística
- El orden de las componentes es arbitrario
- Más sensible al ruido

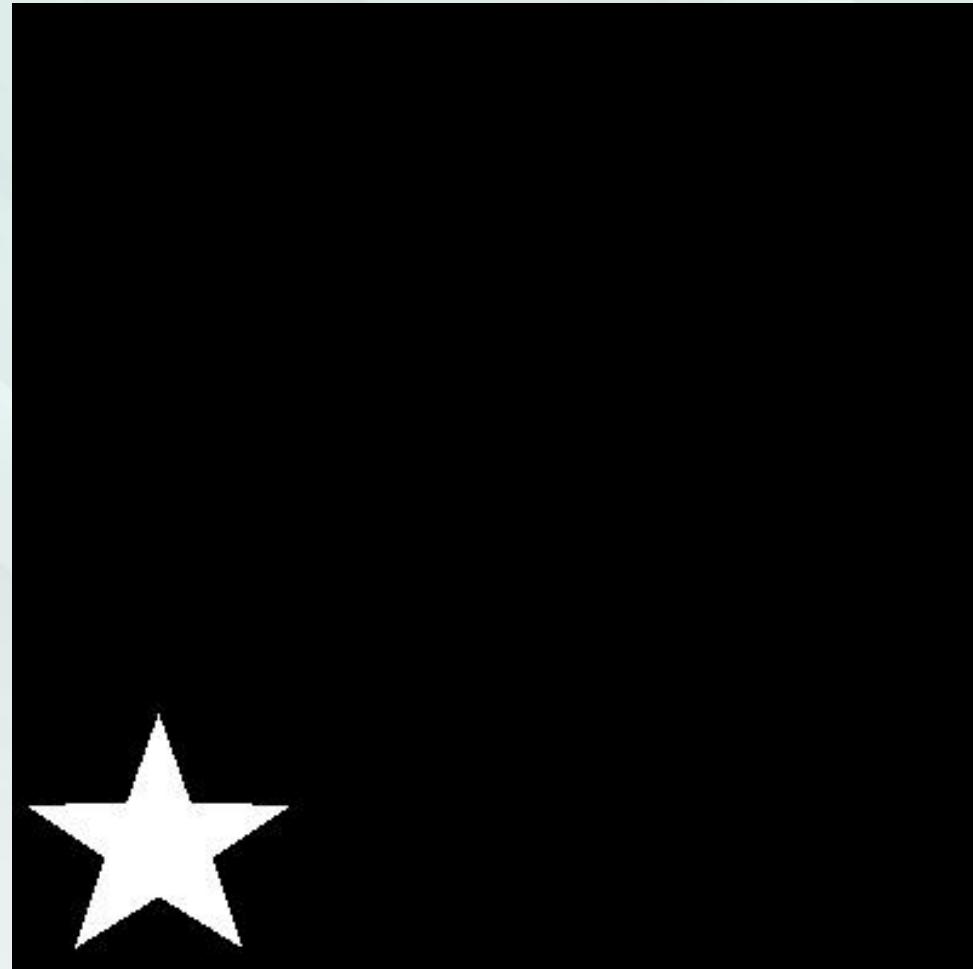
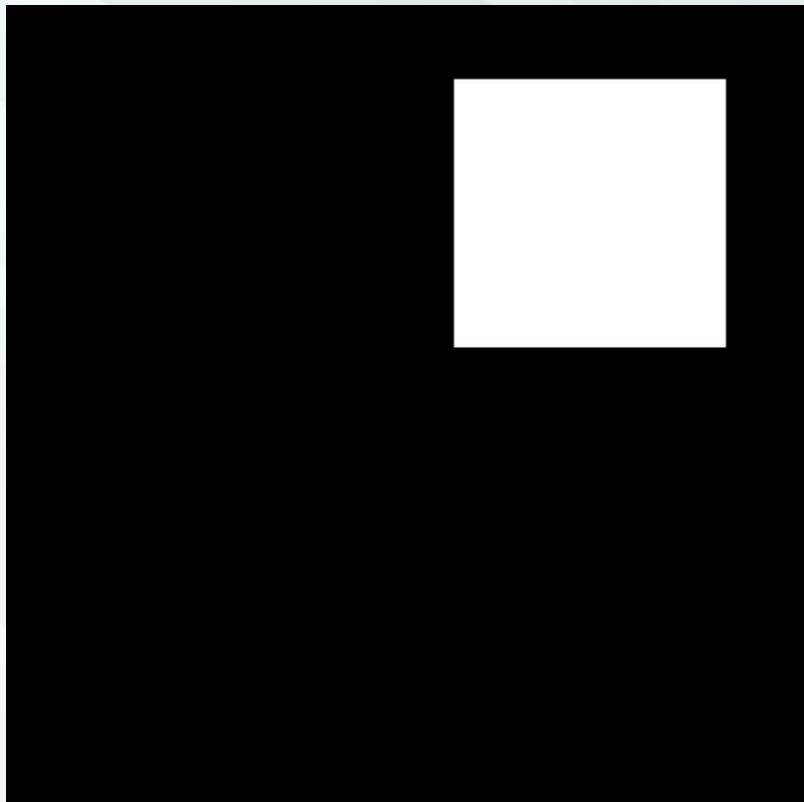


# Imágenes de Prueba:



UNIVERSIDAD  
CATÓLICA  
SEDES SAPIENTIAE

- Condición: Ambas imágenes deben de ser del mismo tamaño.



# Sumar Imágenes:

Se pueden añadir dos imágenes por la función OpenCV, cv2.add() o simplemente por la operación numpy. Ambas imágenes deben tener la misma profundidad y tipo, o la segunda imagen puede ser un valor escalar.  
`res = img1 + img2`

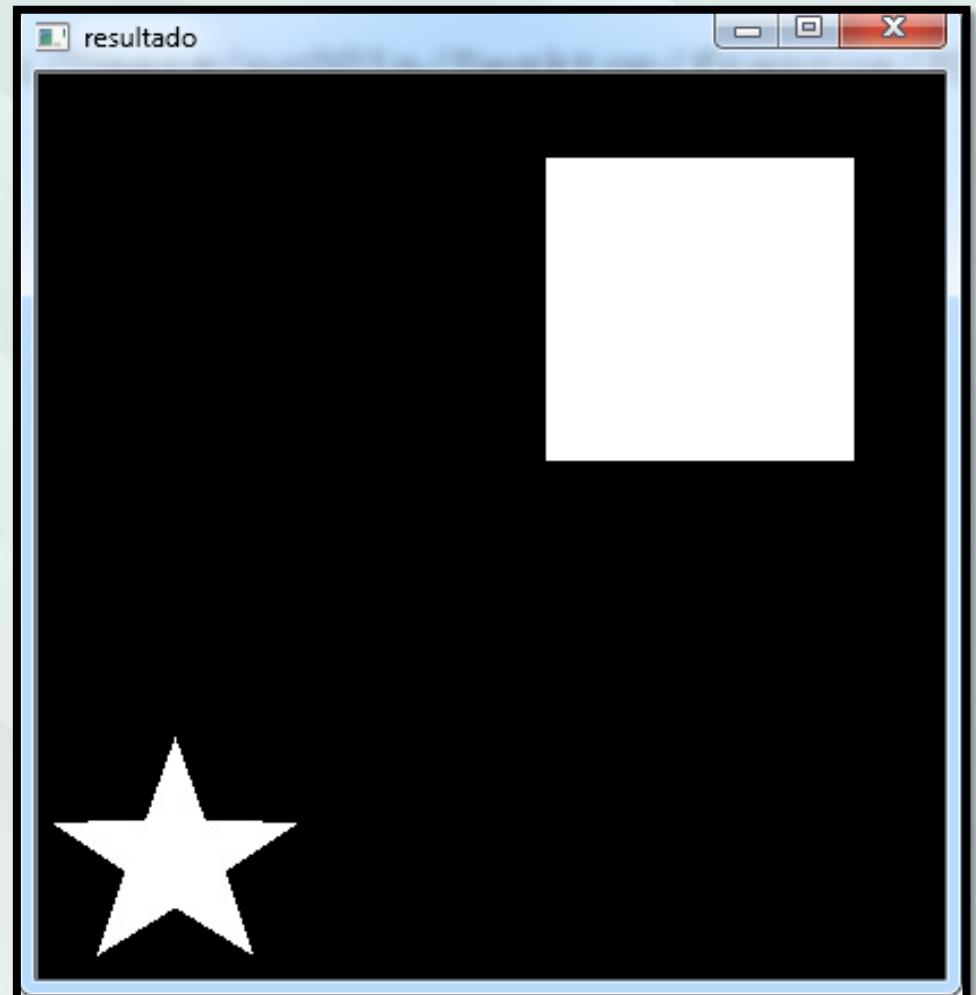
```
import numpy as np
import cv2

img1 = cv2.imread('cuadrado.jpg')
img2 = cv2.imread('estrella.jpg')

img3= cv2.add(img1, img2)

print(img1[0,0])
print(img2[0,0])
print(img3[0,0])

cv2.imshow('resultado', img3)
```



# Resultado de suma de imágenes:

```
import numpy as np
import cv2

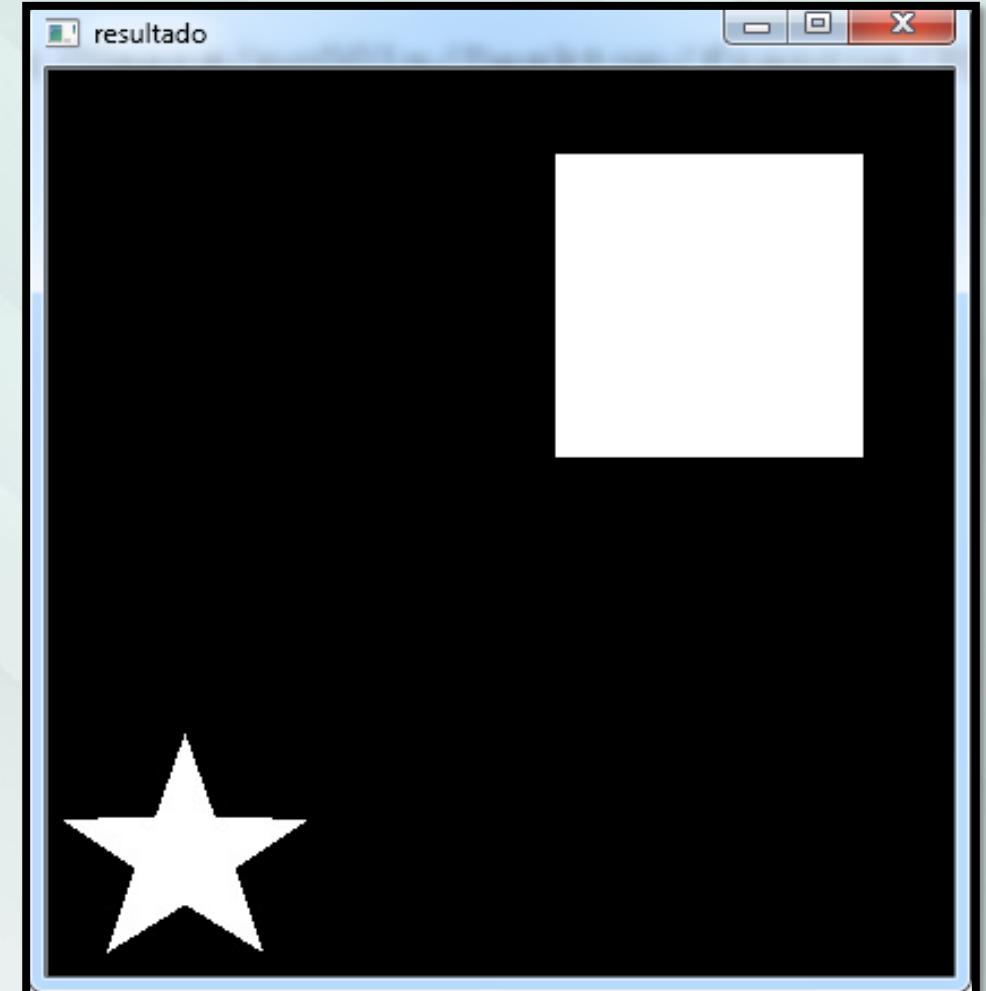
img1 = cv2.imread('cuadrado.jpg')
img2 = cv2.imread('estrella.jpg')

img3= cv2.add(img1,img2)

print(img1[0,0])
print(img2[0,0])
print(img3[0,0])

cv2.imshow('resultado',img3)
```

```
[ 29 229 181]
[ 29 229 181]
[ 58 255 255]
```





# Mezclar Imágenes:

Esto también es una adición de imagen, pero se otorgan diferentes pesos a las imágenes para que parezca una mezcla o transparencia. Las imágenes se agregan según la siguiente ecuación:

$$g(x) = (1 - \alpha)f_0(x) + \alpha f_1(x)$$

Al variar  $\alpha$  de  $0 \rightarrow 1$ , puede realizar una transición genial entre una imagen a otra.

Aquí tomé dos imágenes para combinarlas. La primera imagen recibe un peso de 0.7 y la segunda imagen recibe 0.3. cv2.addWeighted()aplica la siguiente ecuación en la imagen.

$$dst = \alpha \cdot img1 + \beta \cdot img2 + \gamma$$





# Mezclar Imágenes:

- Imágenes de prueba





# Resultado de Suma de Imágenes:

```
import numpy as np
import cv2

img1 = cv2.imread('img1.jpg')
img2 = cv2.imread('img2.jpg')

img3= cv2.addWeighted(img1,0.5,img2,0.5,0)

cv2.imshow('resultado',img3)
```

$$dst = 0.5 * img1 + 0.5 * img2 + 0$$





UNIVERSIDAD  
CATÓLICA  
SEDES SAPIENTIAE

# Resta de Imágenes:

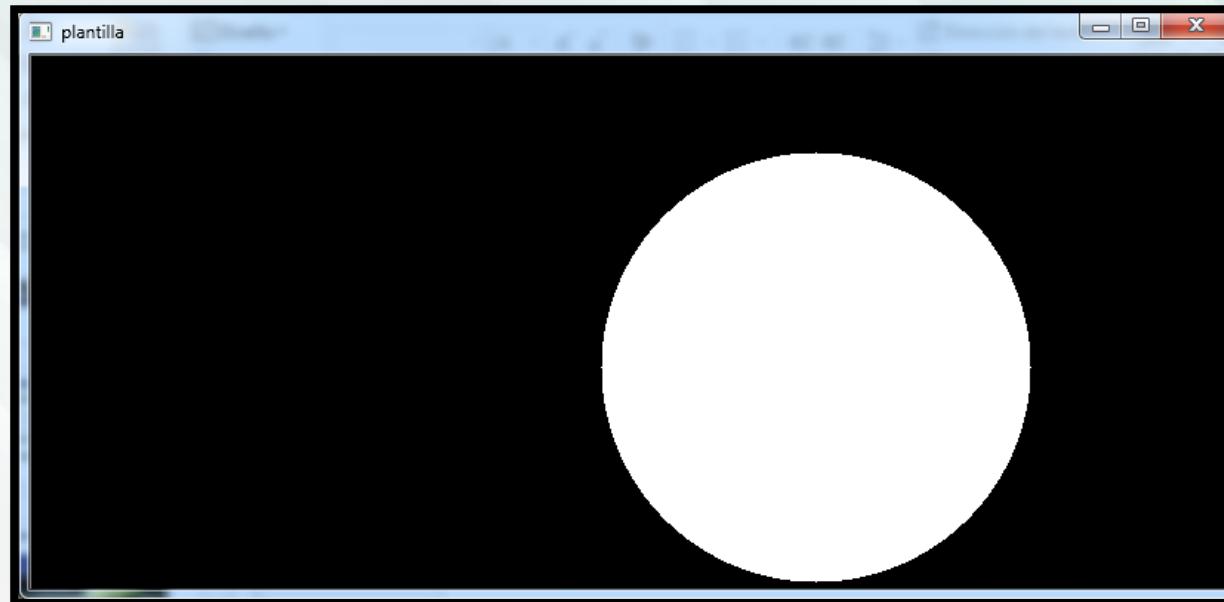




# Plantilla de Imagen:

```
import numpy as np
import cv2

img = np.zeros((360,810,3),np.uint8)
img = cv2.circle(img,(530,210),145,(255,255,255),-1)
cv2.imshow('plantilla',img)
```



# Resta de Imágenes:

```
import numpy as np
import cv2

img = np.zeros((360,810,3),np.uint8)
img = cv2.circle(img,(530,210),145,(255,255,255),-1)
cv2.imshow('plantilla',img)

img1 = cv2.imread('tenis.jpg')

img3= cv2.subtract(img,img1)

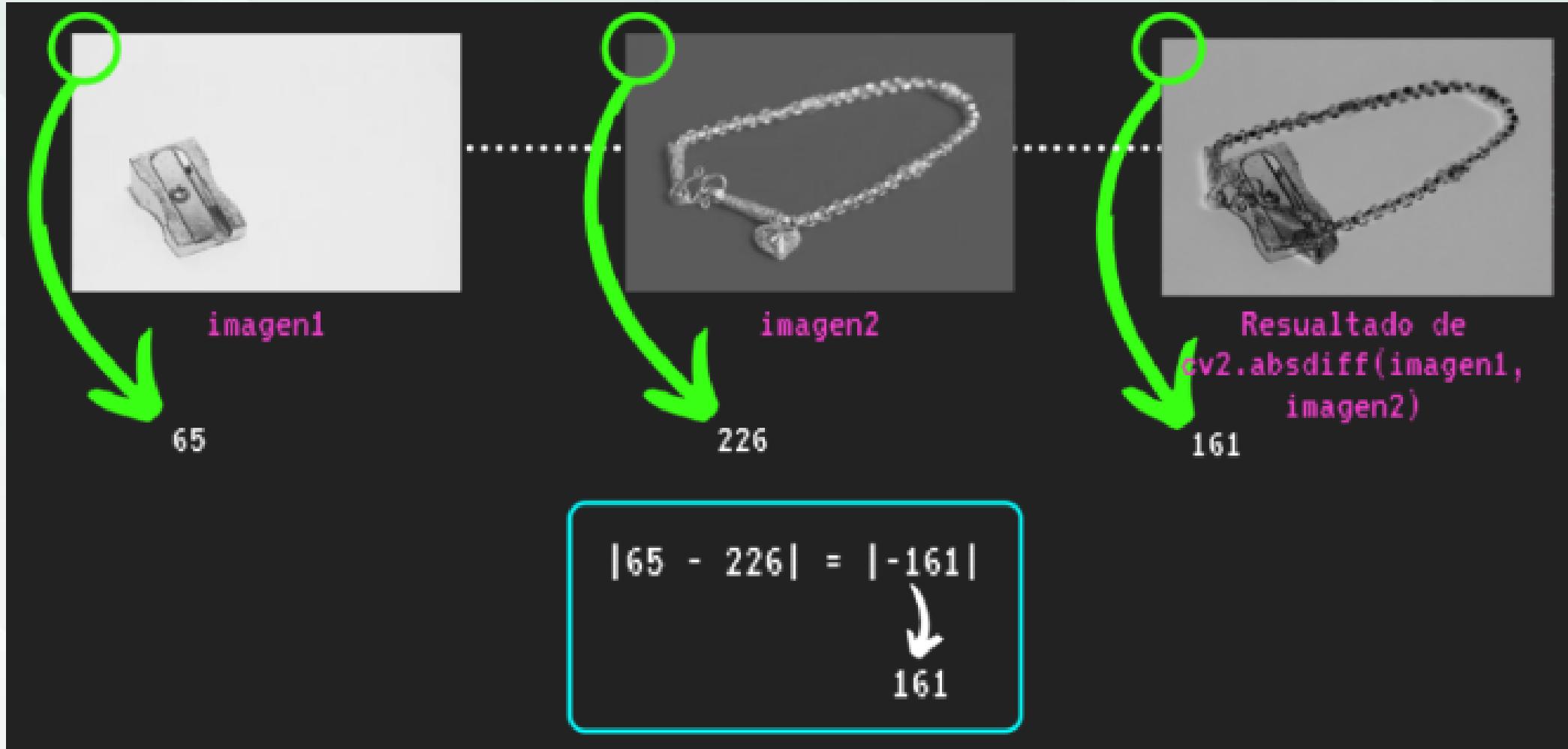
print(img1[0,0])
print(img3[0,0])

cv2.imshow('resultado',img3)
```





# Resta Absoluta de Imágenes (absdiff):





## Resta Absoluta de Imágenes:

- Para aplicar sustracción de imágenes OpenCV nos ofrece dos funciones: cv2.subtract y cv2.absdiff, vamos a ver como usar cada una de ellas y lo que hacen, ¡vamos por ello!.





# Resta Absoluta de Imágenes:

```
import numpy as np
import cv2

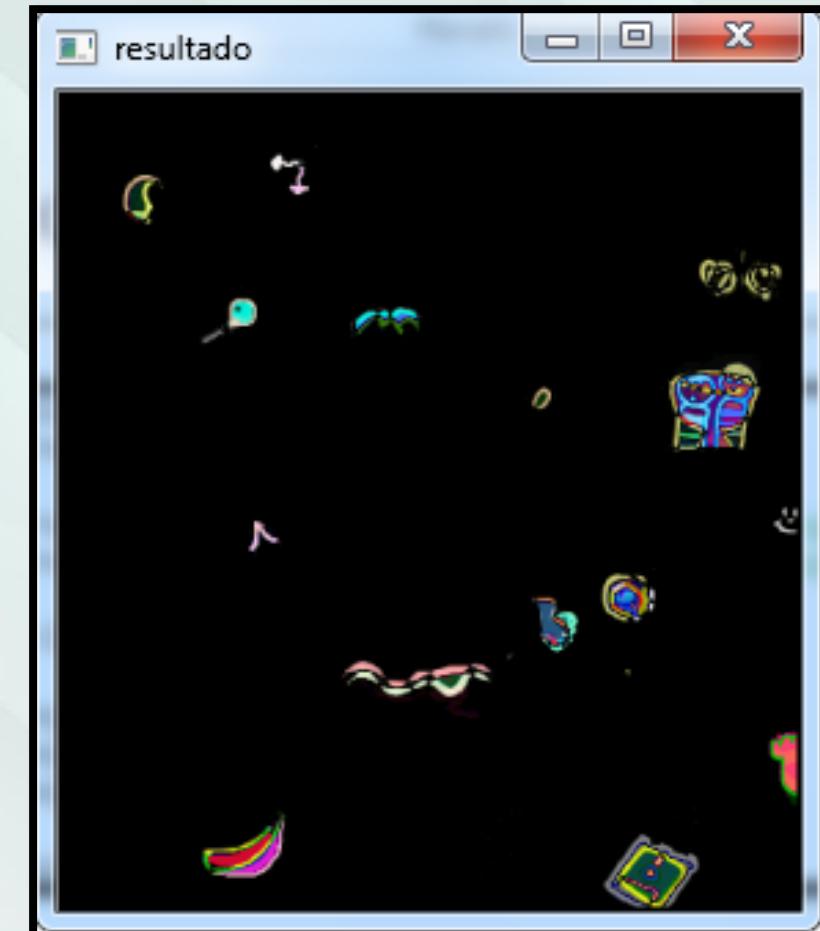
img1 = cv2.imread('Imagen1.png')
img2 = cv2.imread('Imagen2.png')

img3= cv2.absdiff(img1,img2)

print(img1[290,250])
print(img2[290,250])
print(img3[290,250])

cv2.imshow('resultado',img3)
```

```
[120 41 0]
[120 41 0]
[0 0 0]
```





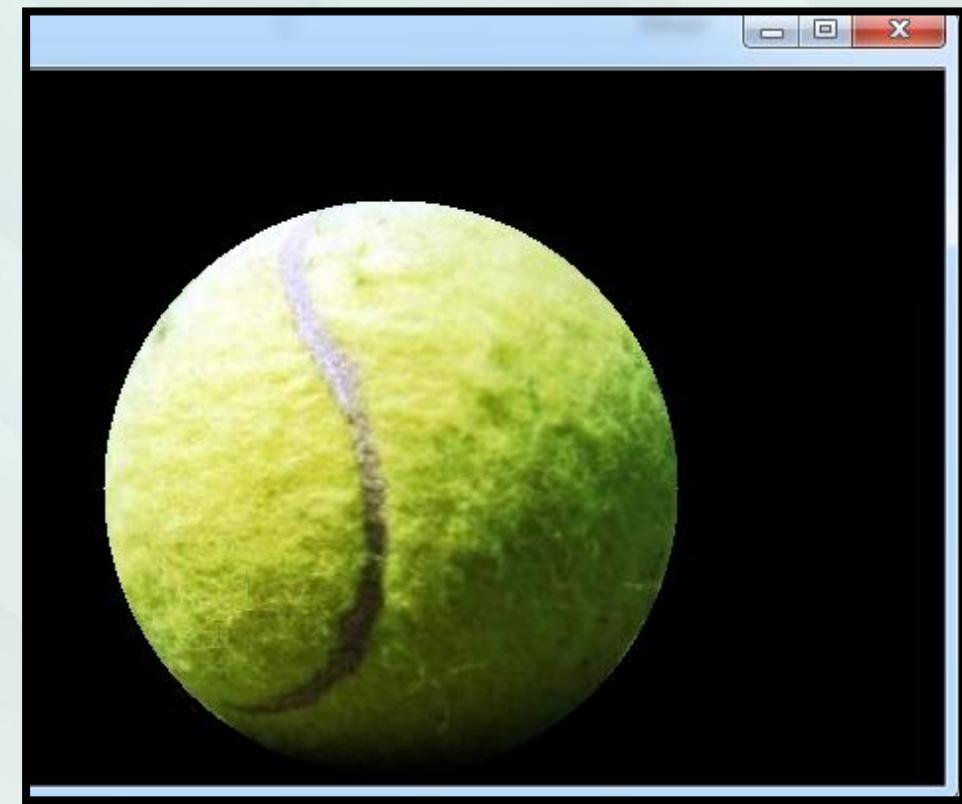
# Bitwise aplicado en Imágenes:

```
import numpy as np
import cv2

mask = np.zeros((360,810,3),np.uint8)
mask= cv2.circle(mask,(530,210),145,(255,255,255),-1)
img1 = cv2.imread('tenis.jpg')

img2= cv2.bitwise_and(img1,mask)

print(img1[0,0])
print(mask[0,0])
cv2.imshow('resultado',img2)
```





UNIVERSIDAD  
CATÓLICA  
SEDES SAPIENTIAE



UNIVERSIDAD  
CATÓLICA  
SEDES SAPIENTIAE

# Inteligencia Artificial

Ing. Juancarlos Santana Huamán  
[jsantana@ucss.edu.pe](mailto:jsantana@ucss.edu.pe)



# ¿Qué es la gestión de datos de IA?

- La gestión de datos de IA es la práctica de emplear inteligencia artificial (IA) y machine learning (ML) en el ciclo de vida de la gestión de datos. Los ejemplos incluyen la aplicación de IA para automatizar u optimizar la recopilación de datos, la limpieza de datos, el análisis de datos, la seguridad de datos y otros procesos de gestión de datos.
- Tanto la IA tradicional basada en reglas como los modelos de IA generativa más avanzados pueden ayudar con la gestión de datos.
- Las empresas modernas poseen grandes cantidades de datos sobre todo, desde transacciones financieras e inventario de productos hasta registros de empleados y preferencias de los clientes. Las organizaciones que emplean estos datos para informar la toma de decisiones e impulsar iniciativas comerciales pueden obtener beneficios significativos sobre sus competidores.

# ¿Qué es la gestión de datos de IA?

- Sin embargo, el desafío proviene de hacer que estos grandes conjuntos de datos sean lo suficientemente precisos, confiables y accesibles para que las personas los utilicen en la práctica.
- Las herramientas de IA y ML pueden ayudar a las organizaciones a emplear sus datos optimizando tareas como la integración de fuentes de datos, la limpieza de datos y la recuperación de datos. Como resultado, las empresas pueden tomar más decisiones basadas en datos.
- La gestión de datos de IA también ayuda a las organizaciones a construir las canalizaciones de datos de alta calidad que necesitan para entrenar y desplegar sus propios modelos de IA y algoritmos de machine learning.





# Herramientas de gestión de datos de IA

- Muchos tipos de herramientas de administración de datos—como soluciones de almacenamiento de datos, herramientas de integración de datos, herramientas de master data management, soluciones de gobierno y otras—ahora incorporan capacidades de ML e IA. Estas herramientas pueden utilizar tanto algoritmos de IA tradicionales como sistemas de IA generativa.
  - **Los sistemas tradicionales** de IA realizan tareas específicas basadas en reglas—por ejemplo, un sistema de gestión de bases de datos que categoriza automáticamente los datos según criterios predefinidos.
  - **Los sistemas de IA generativa**, como Microsoft Copilot, Llama de Meta e IBM Granite™, responden al lenguaje natural y crean contenido original. Por ejemplo, un sistema de gestión de bases de datos con un modelo de lenguaje extensos (LLM) integrado puede crear resúmenes de datos y aceptar consultas en inglés sencillo en lugar de SQL.



# Casos de uso de gestión de datos de IA

## Descubrimiento de datos

- Las organizaciones de hoy en día trabajan con una gran cantidad de datos, que llegan al negocio de múltiples fuentes diferentes, en múltiples formatos. Estos datos son manejados por varios usuarios y terminan dispersos en nubes públicas y privadas, sistemas de almacenamiento on-premises e incluso endpoints personales de los empleados.
- Puede ser difícil realizar un seguimiento y administrar centralmente todos estos datos, lo que plantea dos problemas.
- En primer lugar, una organización no puede emplear un conjunto de datos si no sabe que el conjunto de datos existe.
- En segundo lugar, estos "datos ocultos" no descubiertos ni gestionados plantean riesgos de seguridad. Según el Informe del costo de una filtración de datos de IBM, un tercio de las violaciones de datos implican datos ocultos. Estas filtraciones cuestan 5.27 millones de dólares de promedio—un 16% más que el costo promedio global de las filtraciones.
- IA y ML pueden automatizar muchos aspectos del descubrimiento de datos, otorgando a las organizaciones más visibilidad y control sobre todos sus activos de datos.



# Casos de uso de gestión de datos de IA

- Ejemplos de IA en el descubrimiento de datos
  - Las herramientas de descubrimiento de datos impulsadas por IA pueden escanear automáticamente dispositivos de red y repositorios de almacenamiento de datos, indexando nuevos datos casi en tiempo real.
  - Las herramientas automatizadas de clasificación de datos pueden etiquetar nuevos datos en función de reglas predefinidas o modelos de machine learning. Por ejemplo, la herramienta podría clasificar cualquier número de nueve dígitos en el formato XXX-XX-XXXX como un número de seguro social de Estados Unidos.
  - Los LLMs y otras herramientas de procesamiento de lenguaje natural pueden extraer datos estructurados de fuentes de datos no estructurados, como por ejemplo, extraer los datos de contacto y la experiencia previa de candidatos a un puesto de trabajo a partir de currículos en documentos de texto con distintos formatos.



# Casos de uso de gestión de datos de IA

## Data quality

- Los datos erróneos pueden causar más problemas que la falta total de datos. Si los datos de una organización son incompletos o inexactos, entonces las iniciativas comerciales y los modelos de IA creados sobre esos datos también serán deficientes.
- Las herramientas de IA y ML pueden ayudar a identificar y corregir errores en los datos de la organización, lo que significa que los usuarios no necesitan hacer el largo trabajo de limpieza manual de datos. La IA también puede trabajar más rápido y detectar más errores que un usuario humano.





# Casos de uso de gestión de datos de IA

- Ejemplos de IA en la limpieza de datos

- Las herramientas de preparación de datos habilitadas para IA pueden realizar comprobaciones de validación y marcar o corregir errores, como formato inadecuado y valores irregulares. Algunas herramientas de preparación de datos impulsadas por IA también pueden convertir los datos al formato adecuado, como convertir notas de reuniones no estructuradas en tablas estructuradas.
- Los generadores de datos sintéticos pueden proporcionar missing values y llenar otros huecos en los conjuntos de datos. Estos generadores pueden emplear modelos de machine learning para identificar patrones en los datos existentes y generar puntos de datos sintéticos de gran precisión.
- Algunas herramientas de master data management (MDM) pueden emplear IA y ML para detectar y corregir errores y duplicados en registros críticos. Por ejemplo, fusionar dos registros de clientes con el mismo nombre, dirección y detalles de contacto.
- Las herramientas de observabilidad de los datos impulsadas por IA pueden generar automáticamente registros de linaje de datos para que las organizaciones puedan rastrear quién usa los datos y cómo cambian con el paso del tiempo.





# Casos de uso de gestión de datos de IA

## Accesibilidad a los datos

- Los silos de datos impiden que muchas organizaciones aprovechen todo el valor de sus datos. IA y ML pueden optimizar los esfuerzos de integración de datos, reemplazando los repositorios de silos con estructuras de datos unificadas. Los usuarios de toda la organización pueden acceder a los activos de datos que necesitan cuando los necesitan.
- Ejemplos de IA en el acceso a datos
  - Las herramientas de integración de datos habilitadas para IA pueden detectar automáticamente las relaciones entre diferentes conjuntos de datos, lo que permite a la organización conectarlos o fusionarlos.
  - Las herramientas de gestión de metadatos con capacidades de IA pueden ayudar a automatizar la creación de catálogos de datos al generar descripciones de activos de datos basadas en etiquetado y clasificación.
  - Las bases de datos y los catálogos de datos con interfaces impulsadas por LLM pueden aceptar y procesar comandos de lenguaje natural, lo que permite a los usuarios encontrar activos de datos y productos sin escribir código personalizado o SQL queries. Algunas interfaces impulsadas por LLM también pueden ayudar a los usuarios a refinar consultas, enriquecer conjuntos de datos o sugerir puntos de datos relacionados.
  - Los motores de consulta habilitados para IA pueden emplear algoritmos de machine learning para mejorar el rendimiento de la base de datos mediante el análisis de patrones de carga de trabajo y la optimización de la ejecución de consultas.



# Casos de uso de gestión de datos de IA

## Data security

- Existe un caso comercial que justifica priorizar la seguridad de los datos. La filtración de datos promedio le cuesta a una organización 4.88 millones de dólares entre pérdida de negocios, tiempo de inactividad del sistema, daño a la reputación y esfuerzos de respuesta, según el Informe del costo de una filtración de datos.
- IA y ML pueden ayudar a aplicar políticas de seguridad, detectar filtraciones y bloquear actividades no autorizadas.
- Ejemplos de IA en la seguridad de datos
  - Las herramientas de prevención de pérdida de datos impulsadas por IA pueden detectar automáticamente información de identificación personal (PII) y otros datos confidenciales, aplicar controles de seguridad y marcar o bloquear el uso no autorizado de esos datos.
  - Las herramientas de detección de amenazas basadas en anomalías, tales como el análisis del comportamiento de usuarios y entidades (UEBA) y la detección y respuesta de endpoints (EDR), emplean algoritmos de IA y ML para monitorear la actividad de la red. Detectan desviaciones sospechosas de la norma, como una gran cantidad de datos que se trasladan repentinamente a una nueva ubicación.
  - Los LLMs pueden ayudar a las organizaciones a generar e implementar políticas de gobernanza de datos. Por ejemplo, en un sistema de control de acceso basado en roles (RBAC), un LLM puede ayudar al equipo de seguridad a describir los diferentes tipos de roles y sus permisos. El LLM también podría ayudar a convertir estas descripciones de roles en reglas para un sistema de gestión de identidad y acceso.
  - Las herramientas de detección de fraude habilitadas para IA pueden emplear IA y ML para analizar patrones y detectar transacciones anormales.



# Beneficios de la gestión de datos de IA

## Aprovechar todo el valor del big data para las empresas

- En el Reporte de gestión de la IA y la información de AvePoint, el 64% de las organizaciones encuestadas dijeron que gestionaban al menos un petabyte de datos.<sup>1</sup> En perspectiva, eso equivale a aproximadamente 9 cuatrillones de bits de información. Y gran parte viene en formatos no estructurados, como archivos de texto, imágenes y video.
- Todos estos datos pueden ser una bendición para los científicos de datos, pero es imposible gestionar manualmente datos tan complejos en cantidades tan masivas. Las herramientas de IA y ML pueden hacer que estos datos sean utilizables mediante la automatización de tareas críticas como el descubrimiento, la integración y la limpieza.
- Cuando los datos están limpios y son accesibles, las organizaciones pueden emplearlos para proyectos avanzados de analytics de datos, como una iniciativa de análisis predictivos que emplea datos históricos para pronosticar tendencias futuras en el gasto de los consumidores.
- Las tecnologías de IA también pueden hacer que los datos sean más accesibles para los usuarios sin experiencia en ciencia de datos. Los catálogos de datos fáciles de usar con interfaces de bases de datos impulsadas por LLM y visualizaciones automatizadas permiten que más usuarios de toda la empresa empleen datos para fundamentar sus decisiones.



# Beneficios de la gestión de datos de IA

## Impulsando las iniciativas de IA

- El 59% de los directores ejecutivos (CEOs) encuestados por IBM Institute for Business Value creen que el beneficio competitivo de una organización en el futuro depende de tener la IA generativa más avanzada. Para crear y desplegar esos modelos de IA, las organizaciones necesitan flujos constantes de datos buenos y limpios.
- Al agilizar la gestión de datos, las herramientas de IA ayudan a crear los canales de datos fiables y de alta calidad que las organizaciones necesitan para capacitar sus propios modelos de IA y ML. Y como estos modelos se pueden entrenar con los datos de la empresa, se pueden entrenar para realizar tareas y resolver problemas específicos de la empresa y sus clientes.





# Beneficios de la gestión de datos de IA

## Usar los datos sin dejar de cumplir

- Las herramientas de seguridad y gobernanza basadas en IA ayudan a prevenir ciberataques y filtraciones de datos, que pueden resultar costosos. También permiten a las empresas emplear los datos que tienen cumpliendo con las regulaciones de privacidad y protección de datos como GDPR y el Estándar de Seguridad de Datos de la Industria de Tarjetas de Pago (PCI-DSS).
- Según el Institute for Business Value, el 57% de los CEOs afirma que la seguridad de datos es una barrera para adoptar IA generativa. El 45% dice que la privacidad de los datos también es una barrera. Estas barreras pueden ser especialmente desafiantes en industrias altamente reguladas, como la atención médica y las finanzas.
- La gestión de datos habilitada por IA puede ayudar aplicando automáticamente las protecciones adecuadas y las políticas de uso de datos. De esa manera, solo los usuarios autorizados pueden acceder a los datos y solo pueden usarlos de la manera que lo permitan las regulaciones de la industria y la política de la empresa.
- Los generadores de datos sintéticos también pueden ayudar generando conjuntos de datos que reflejen con precisión las tendencias generales, al tiempo que eliminan datos personales confidenciales que una organización podría no tener permitido usar de ciertas maneras.

# ¿Qué es el Decision Tree o árbol de decisiones en IA?

- Un decision tree (o árbol de decisión) es una técnica de modelado predictivo en la cual se utiliza una estructura de árbol para representar decisiones y sus consecuencias. El árbol de decisión divide un conjunto de datos en subconjuntos más pequeños en función de las características de los datos. Cada nodo del árbol representa una característica (atributo), cada rama representa una decisión basada en esa característica y cada hoja representa el resultado de la decisión tomada.
- Los árboles de decisión se utilizan en la ciencia de datos y el aprendizaje automático como una técnica de clasificación y predicción. En la clasificación, los árboles de decisión se utilizan para asignar una etiqueta o clasificación a un conjunto de datos en función de las características de los mismos. En la predicción, se utilizan para predecir valores numéricos de una variable objetivo. Los algoritmos de árboles de decisión son a menudo utilizados en problemas de clasificación y se han utilizado en numerosos campos, como medicina, finanzas, marketing, entre otros.
- Es una técnica popular debido a que es fácil de entender y se puede visualizar de manera gráfica, lo que facilita la interpretación de los resultados del análisis. En resumen, el decision tree es una técnica en el aprendizaje automático que permite la creación de modelos predictivos utilizando una estructura de árbol de decisiones para la clasificación y predicción de datos.



# ¿Cómo funcionan los árboles de decisión en machine learning?

- Los árboles de decisión son un tipo de aprendizaje automático supervisado destinado a realizar predicciones en función de un conjunto de preguntas a las que el sistema ha de responder y realizar una predicción.

## ¿Cómo funcionan los árboles de decisión?

- Podemos asemejar el funcionamiento de un árbol de decisión al de la simple mecánica del juego infantil “veo, veo”.
- Imagina que queremos que un ordenador aprenda a distinguir entre diferentes tipos de frutas, como manzanas y naranjas. El árbol de decisión empezaría con una pregunta, como “¿Es la fruta redonda?” Si la respuesta es sí, entonces la computadora podría preguntar “¿Es la fruta roja?” Si la respuesta es sí de nuevo, podría decidir que la fruta es una manzana. Si la respuesta es no, podría ser una naranja. Si la fruta no es redonda, el sistema formularía otra pregunta a raíz de otra característica, como “¿Es la fruta de color naranja?”
- El árbol de decisión sigue haciendo preguntas y tomando decisiones basadas en las respuestas hasta que clasifica correctamente todas las frutas en grupos.





# ¿Cómo funcionan los árboles de decisión?

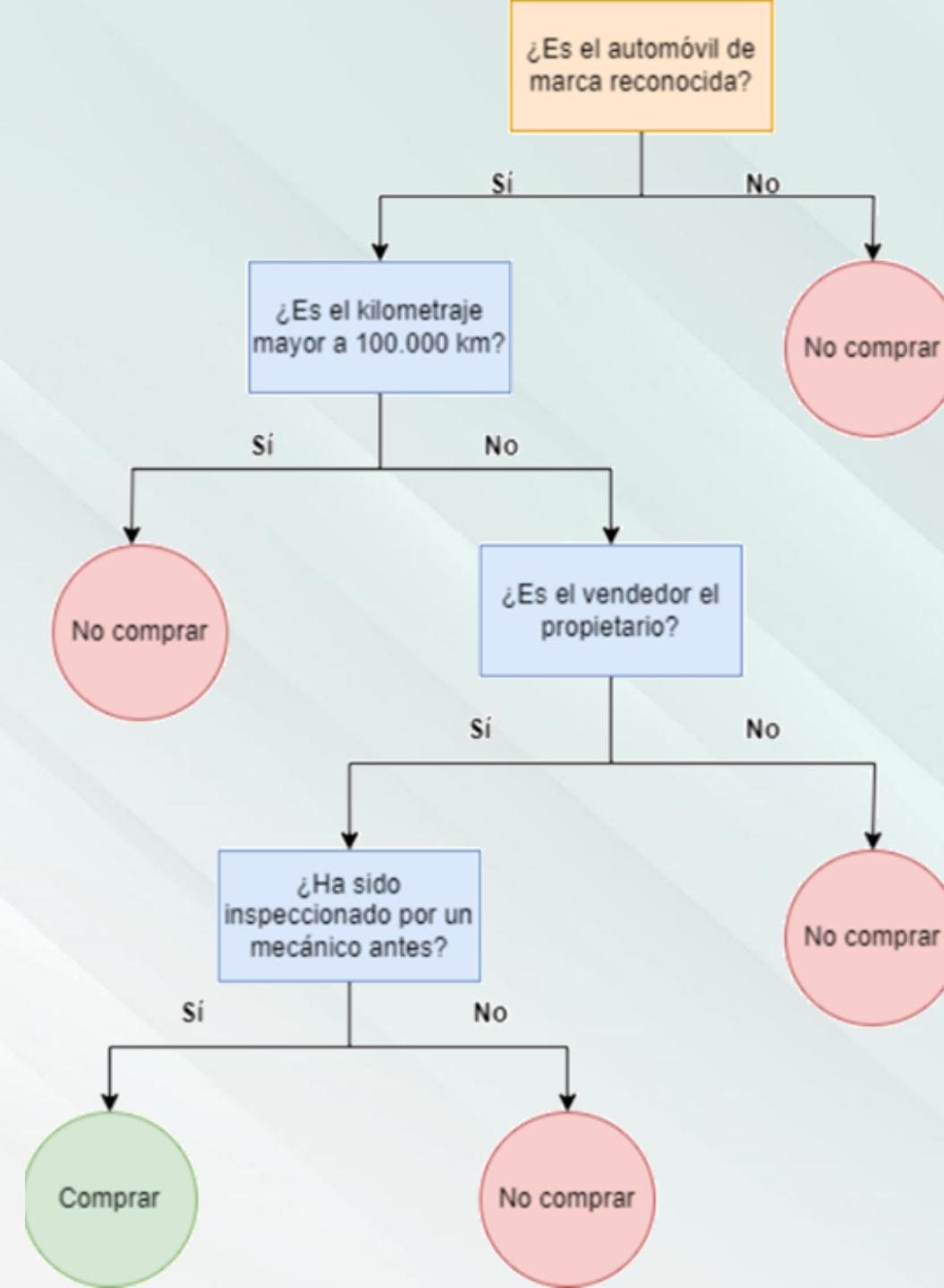
- **Nodos raíz y nodos de decisión.** Así pues, un árbol de decisión es parecido a un árbol. A la base del árbol se le llama nodo raíz, a partir del cual nacen los nodos de decisión, que representan posibles preguntas por las que se quiere segmentar toda la muestra. La formulación de estas preguntas dependerá de las variables que hayamos identificado y que queramos utilizar para segmentar el conjunto de datos.
- **Divisiones y subnodos.** A estos nodos de decisión les corresponden una, dos o más respuestas, llamadas divisiones o particiones. Cuantas más particiones o respuestas posibles tengan un nodo de decisión, más subnodos de decisión se crearán, y mayor será la complejidad del árbol.
- A medida que avance al árbol y sus ramificaciones, el número de nodos de decisión se irán reduciendo y el conjunto de datos se irán clasificando en subconjuntos más homogéneos basándose en ciertos criterios.
- **Hojas.** El objetivo del árbol de decisión es que el sistema de inteligencia artificial vaya cribando poco a poco el conjunto de datos y los categorice con precisión y rigor. Así pues, el árbol de decisión termina en hojas, es decir, respuestas únicas sin particiones para todos los nodos de decisión.
- **Poda.** A medida que va ramificándose el árbol en particiones, es necesario ir filtrando nodos para evitar que crezca en exceso con información que no nos es relevante para nuestros propósitos. Esta depuración del modelo de inteligencia artificial, también llamada en inglés “pruning”, se realiza durante el diseño del árbol de decisiones, así como después de completar todo el árbol. Es una forma de simplificar el modelo, hacerlo más útil y evitar que se sobreajuste a los datos del entrenamiento.





# ¿Para qué sirven los árboles de decisión?

- Los árboles de decisión en la Inteligencia Artificial (IA) se utilizan para diversas aplicaciones, aquí te menciono algunas de las más comunes:
  - **Clasificación de datos:** Los árboles de decisión pueden clasificar datos en diferentes categorías o clases basándose en ciertos criterios.
  - **Predicciones:** Su aplicación más útil y extendida es la de hacer predicciones sobre valores numéricos. Por ejemplo, pueden predecir el precio de una casa basándose en características como su tamaño, ubicación, número de habitaciones, etc.
  - **Detección de anomalías:** En ciberseguridad, se pueden utilizar para identificar actividades sospechosas (patrones de uso irregular) en una red, como intentos de piratería o intrusiones.
  - **Aprendizaje automático supervisado:** Los árboles de decisión son un tipo de aprendizaje automático supervisado destinado a realizar predicciones en función de un conjunto de preguntas a las que el sistema ha de responder y realizar una predicción.
  - **Resolución de problemas de regresión o de clasificación:** Un árbol de decisión es una técnica que emplea algoritmos en forma de árbol para enseñar a las máquinas a tomar decisiones y, por tanto, a resolver problemas de regresión o de clasificación.
  - Creación de modelos predictivos precisos y fiables: Como resultado, obtenemos modelos predictivos precisos y fiables.
- En resumen, los árboles de decisión son una herramienta poderosa en el campo de la IA que permite una representación clara y fácil de entender de la toma de decisiones.





# Ventajas y desventajas de los árboles de decisión

- Los árboles de decisión en machine learning son una técnica muy útil y popular por las siguientes razones:
  - **No requieren mucho tiempo de preprocesado y limpieza de datos:** A diferencia de algunos otros algoritmos de aprendizaje automático, los árboles de decisión pueden manejar datos con diferentes tipos de variables (categóricas y numéricas) sin necesidad de una preparación extensa. Trabajar con árboles de decisión es muy eficiente, ya que no se ha de perder mucho tiempo antes de elaborar uno.
  - **No importa si el dataset tiene lagunas:** Los árboles de decisión pueden manejar conjuntos de datos incompletos, es decir, conjuntos de datos que tienen valores faltantes o lagunas.
  - **Es sencillo de explicar visualmente a otras personas:** Los árboles de decisión son fáciles de entender y visualizar, incluso para personas que no tienen formación específica en el área. Puedes representar un árbol de decisión como un diagrama que muestra las decisiones tomadas en cada nodo y cómo se llega a una conclusión para facilitar que otras personas comprendan el modelo que estás entrenando.



# Ventajas y desventajas de los árboles de decisión

- Sin embargo, debemos ser conscientes de que los árboles de decisión tienen sus limitaciones y desventajas, como las siguientes:
  - **Inestabilidad:** Los cambios en los datos o la presencia de ruido pueden hacer que la estructura del árbol varíe enormemente. En esencia, los árboles son sensibles a pequeñas variaciones en los datos de entrenamiento o a la presencia de ruido. Es decir, si los datos cambian un poco o si hay errores en los datos, la estructura del árbol resultante puede verse muy alterada, haciendo que la toma de decisiones sea menos rigurosa según las predicciones.
  - **Lleva más tiempo entrenar un modelo usando un árbol de decisión:** Aunque los árboles de decisión son relativamente rápidos en la etapa de predicción, pueden llevar más tiempo entrenar el modelo, especialmente si el conjunto de datos es grande o si el árbol crece demasiado. Esto se debe a que el algoritmo necesita considerar todas las posibles divisiones y características para construir el árbol, lo que puede ser computacionalmente costoso.
  - **Sobreajuste (overfitting):** Los árboles de decisión tienen una alta capacidad para adaptarse a los datos de entrenamiento, lo que conduce a una situación de overfitting. El sobreajuste ocurre cuando el modelo se ajusta demasiado a los datos de entrenamiento y captura el ruido o las características específicas de esos datos en lugar de aprender patrones generales que se puedan generalizar a nuevos datos. Así pues, cuando a un modelo sobreentrenado lo alimentamos con información de un conjunto de datos diferentes, ofrece muchas respuestas incorrectas y un rendimiento ínfimo.



# Tipos de árboles de decisión

- Diferenciamos los siguientes árboles de decisión según su enfoque:

## De clasificación

- Este tipo de árbol se utiliza cuando la variable objetivo es categórica, es decir, pertenece a un conjunto discreto de clases o categorías. El árbol de clasificación divide el conjunto de datos en función de las características para clasificar las instancias en diferentes categorías. Por ejemplo, puede clasificar correos electrónicos como spam o no spam, pacientes como enfermos o sanos, o clientes como compradores o no compradores.

## De regresión

- Un árbol de regresión se utiliza cuando la variable objetivo es numérica o continua, es decir, puede tomar cualquier valor dentro de un rango específico. El árbol de regresión divide el conjunto de datos en función de las características para predecir un valor numérico. Por ejemplo, puede predecir el precio de una casa en función de su tamaño, la cantidad de habitaciones, la ubicación, etc. o predecir la temperatura máxima diaria en función de variables meteorológicas como la humedad, la presión atmosférica y la velocidad del viento.





# ¿Cómo se crea el Decision Tree o árbol de decisiones de IA?

- Crear un árbol de decisiones en Inteligencia Artificial (IA) implica varios pasos. Aquí te dejo un resumen de cómo se puede hacer:
  - **División de los datos:** La técnica comienza dividiendo los datos en subconjuntos más pequeños. La variabilidad se reduce al mismo tiempo que aumenta la homogeneidad dentro de cada subgrupo.
  - **Selección de variables:** En este paso, el modelo selecciona automáticamente las mejores variables para realizar la partición utilizando medidas como entropía e índice Gini.
  - **Continuación del proceso:** El proceso continúa hasta que todos los nodos sean puros o hasta alcanzar alguna condición límite previa establecida por el usuario (por ejemplo, profundidad del árbol).
  - **Aplicación del modelo:** Cuando el modelo está entrenado, puede aplicarse a nuevos conjuntos de datos entrantes para hacer predicciones precisas.
  - **Poda:** A medida que va ramificándose el árbol en particiones, es necesario ir filtrando nodos para evitar que crezca en exceso con información que no nos es relevante para nuestros propósitos. Esta depuración del modelo de inteligencia artificial, también llamada en inglés “pruning”, se realiza durante el diseño del árbol de decisiones, así como después de completar todo el árbol.
  - **Validación y evaluación del modelo:** Una vez que el árbol de decisiones ha sido construido, es importante validar y evaluar su rendimiento. Esto puede implicar el uso de técnicas como la validación cruzada.
  - **Implementación del modelo:** Finalmente, una vez que estás satisfecho con el rendimiento del modelo, puedes implementarlo para hacer predicciones en nuevos datos.
- Es importante mencionar que existen varias bibliotecas y herramientas que pueden ayudarte a crear árboles de decisiones en IA, como Scikit-learn en Python, rpart en R, y otros.



# Crear un árbol de decisiones usando el algoritmo ID3 con un ejemplo resuelto

## ¿Qué es el algoritmo ID3?

- El algoritmo ID3 (Dicotomizador Iterativo 3) es uno de los más antiguos y utilizados para crear árboles de decisión a partir de un conjunto de datos. Emplea el concepto de entropía y ganancia de información para seleccionar el mejor atributo para dividir los datos en cada nodo. La entropía mide la incertidumbre o aleatoriedad de los datos, y la ganancia de información cuantifica la reducción de la incertidumbre lograda al dividir los datos en un atributo específico. El algoritmo ID3 divide recursivamente el conjunto de datos según los atributos con mayor ganancia de información hasta que se cumple un criterio de parada, lo que genera un árbol de decisión que puede utilizarse para tareas de clasificación.

## Comprensión del algoritmo ID3:

- El algoritmo ID3 utiliza el concepto de entropía y ganancia de información para construir un árbol de decisión. La entropía mide la incertidumbre o aleatoriedad de un conjunto de datos, mientras que la ganancia de información cuantifica la reducción de entropía lograda al dividir los datos en un atributo específico. El atributo con la mayor ganancia de información se selecciona como nodo de decisión del árbol.





# Pasos para crear un árbol de decisiones utilizando el algoritmo ID3:

- **Paso 1: Preprocesamiento de datos:** Limpiar y preprocesar los datos. Gestionar los valores faltantes y convertir las variables categóricas en representaciones numéricas si es necesario.
- **Paso 2: Selección del nodo raíz:** Calcule la entropía de la variable objetivo (etiquetas de clase) con base en el conjunto de datos. La fórmula para la entropía es:

$$Entropy(S) = - \sum (p_i \times \log_2(p_i))$$

- Donde:
  - $p_i$  es la proporción de instancias que pertenecen a la clase  $i$ .





# Pasos para crear un árbol de decisiones utilizando el algoritmo ID3:

- **Paso 3: Cálculo de la ganancia de información:** Para cada atributo del conjunto de datos, calcule la ganancia de información al dividir el conjunto de datos en función de dicho atributo. La fórmula para la ganancia de información es:

$$\text{Information Gain}(S, A) = \text{Entropy}(S) - \sum \left( \left( |S_v|/|S| \right) \times \text{Entropy}(S_v) \right)$$

- Donde:
  - $S_v$  ∈ es el subconjunto de instancias para cada valor posible del atributo A y  $|S_v|$  ∈ es el número de instancias en ese subconjunto.
- **Paso 4: Seleccionar el mejor atributo:** elija el atributo con la mayor ganancia de información como nodo de decisión para el árbol.
- **Paso 5: Dividir el conjunto de datos:** divida el conjunto de datos según los valores del atributo seleccionado.
- **Paso 6: Repetir el proceso:**  
Repetir recursivamente los pasos 2 a 5 para cada subconjunto hasta que se cumpla un criterio de detención (por ejemplo, la profundidad del árbol alcanza un límite máximo o todas las instancias de un subconjunto pertenecen a la misma clase).



# Ejemplo resuelto:

- Ilustremos el algoritmo ID3 con un ejemplo sencillo de clasificación para jugar al tenis según las condiciones meteorológicas. Consideremos el siguiente conjunto de datos:

Clima	Temperatura	Humedad	Ventoso	¿Juegas al tenis?
Soleado	Caliente	Alto	FALSO	No
Soleado	Caliente	Alto	Verdadero	No
Nublado	Caliente	Alto	FALSO	Sí
Lluvioso	Leve	Alto	FALSO	Sí
Lluvioso	Fresco	Normal	FALSO	Sí
Lluvioso	Fresco	Normal	Verdadero	No
Nublado	Fresco	Normal	Verdadero	Sí
Soleado	Leve	Alto	FALSO	No
Soleado	Fresco	Normal	FALSO	Sí
Lluvioso	Leve	Normal	FALSO	Sí
Soleado	Leve	Normal	Verdadero	Sí
Nublado	Leve	Alto	Verdadero	Sí
Nublado	Caliente	Normal	FALSO	Sí
Lluvioso	Leve	Alto	Verdadero	No

# Ejemplo resuelto:

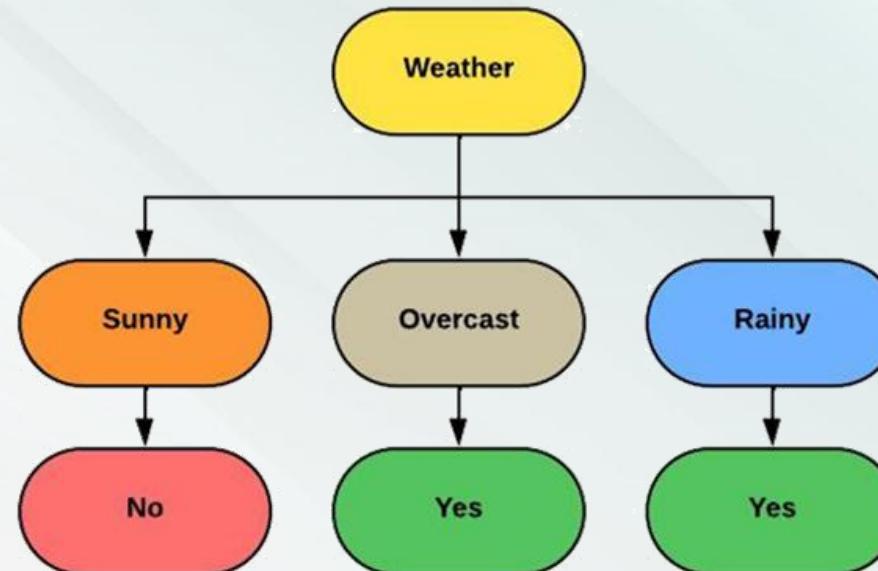
- **Paso 1: Preprocesamiento de datos:** el conjunto de datos no requiere ningún preprocesamiento, ya que está en un formato adecuado.
- **Paso 2: Cálculo de la entropía:** para calcular la entropía, primero determinamos la proporción de instancias positivas y negativas en el conjunto de datos:
  - Casos positivos (Jugar al tenis = Sí): 9
  - Instancias negativas (Jugar al tenis = No): 5
$$Entropy(S) = - \sum (p_i \times \log_2(p_i))$$
  - Entropía(S) =  $-(9/14) * \log_2(9/14) - (5/14) * \log_2(5/14) \approx 0,940$
- **Paso 3: Cálculo de la ganancia de información:** Calculamos la ganancia de información para cada atributo (clima, temperatura, humedad, viento) y elegimos el atributo con la mayor ganancia de información como nodo raíz.
  - Ganancia de información (S, Clima) = Entropía (S) – [(5/14) \* Entropía (Soleado) + (4/14) \* Entropía (Nublado) + (5/14) \* Entropía (Lluvioso)]  $\approx 0,246$
  - Ganancia de información (S, Temperatura) = Entropía (S) – [(4/14) \* Entropía (Caliente) + (4/14) \* Entropía (Suave) + (6/14) \* Entropía (Fría)]  $\approx 0,029$
  - Ganancia de información (S, Humedad) = Entropía (S) – [(7/14) \* Entropía (Alta) + (7/14) \* Entropía (Normal)]  $\approx 0,152$
  - Ganancia de información (S, Windy) = Entropía (S) – [(8/14) \* Entropía (Falso) + (6/14) \* Entropía (Verdadero)]  $\approx 0,048$





# Ejemplo resuelto:

- **Paso 4: Seleccionar el mejor atributo:** El atributo “Clima” tiene la mayor ganancia de información, por lo que lo seleccionamos como el nodo raíz de nuestro árbol de decisiones.
- **Paso 5: División del conjunto de datos:** Dividimos el conjunto de datos en función de los valores del atributo “Clima” en tres subconjuntos (Soleado, Nublado, Lluvioso).
- **Paso 6: Repetir el proceso:** Dado que el atributo "Clima" no tiene valores repetidos en ningún subconjunto, dejamos de dividir y etiquetamos cada nodo hoja con la clase mayoritaria de ese subconjunto. El árbol de decisión se verá como se muestra a continuación:





# Código Python para crear un árbol de decisiones utilizando el algoritmo ID3:

```
import pandas as pd
import numpy as np
import random

# Define the dataset
data = {
    'Weather': ['Sunny', 'Sunny', 'Overcast', 'Rainy', 'Rainy', 'Rainy', 'Overcast', 'Sunny', 'Sunny', 'Rainy', 'Sunny', 'Overcast', 'Overcast',
    'Rainy'],
    'Temperature': ['Hot', 'Hot', 'Hot', 'Mild', 'Cool', 'Cool', 'Cool', 'Mild', 'Mild', 'Mild', 'Mild', 'Hot', 'Mild'],
    'Humidity': ['High', 'High', 'High', 'High', 'Normal', 'Normal', 'Normal', 'High', 'Normal', 'Normal', 'Normal', 'High', 'Normal', 'High'],
    'Windy': [False, True, False, False, False, True, True, False, False, False, True, True, False, True],
    'Play Tennis': ['No', 'No', 'Yes', 'Yes', 'Yes', 'No', 'Yes', 'No', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes']
}

df = pd.DataFrame(data)

def entropy(target_col):
    elements, counts = np.unique(target_col, return_counts=True)
    entropy_val = -np.sum([(counts[i] / np.sum(counts)) * np.log2(counts[i] / np.sum(counts)) for i in range(len(elements))])
    return entropy_val
```





```
def information_gain(data, split_attribute_name, target_name):
    total_entropy = entropy(data[target_name])
    vals, counts= np.unique(data[split_attribute_name], return_counts=True)
    weighted_entropy = np.sum([(counts[i] / np.sum(counts)) * entropy(data.where(data[split_attribute_name]==vals[i]).dropna()[target_name]) for i in range(len(vals))])
    information_gain_val = total_entropy - weighted_entropy
    return information_gain_val

def id3_algorithm(data, original_data, features, target_attribute_name, parent_node_class):
    # Base cases
    if len(np.unique(data[target_attribute_name])) <= 1:
        return np.unique(data[target_attribute_name])[0]
    elif len(data) == 0:
        return np.unique(original_data[target_attribute_name])[np.argmax(np.unique(original_data[target_attribute_name], return_counts=True)[1])]
    elif len(features) == 0:
        return parent_node_class
    else:
        parent_node_class = np.unique(data[target_attribute_name])[np.argmax(np.unique(data[target_attribute_name], return_counts=True)[1])]
        item_values = [information_gain(data, feature, target_attribute_name) for feature in features]
        best_feature_index = np.argmax(item_values)
        best_feature = features[best_feature_index]
        tree = {best_feature: {}}
        features = [i for i in features if i != best_feature]
        for value in np.unique(data[best_feature]):
            value = value
            sub_data = data.where(data[best_feature] == value).dropna()
            subtree = id3_algorithm(sub_data, data, features, target_attribute_name, parent_node_class)
            tree[best_feature][value] = subtree
        return tree
```



```
def predict(query, tree, default = 1):
    for key in list(query.keys()):
        if key in list(tree.keys()):
            try:
                result = tree[key][query[key]]
            except:
                return default
            result = tree[key][query[key]]
        if isinstance(result, dict):
            return predict(query, result)
        else:
            return result

def train_test_split(df, test_size):
    if isinstance(test_size, float):
        test_size = round(test_size * len(df))
    indices = df.index.tolist()
    test_indices = random.sample(population=indices, k=test_size)
    test_df = df.loc[test_indices]
    train_df = df.drop(test_indices)
    return train_df, test_df
```





```
train_data, test_data = train_test_split(df, test_size=0.2)

def fit(df, target_attribute_name, features):
    return id3_algorithm(df, df, features, target_attribute_name, None)

def get_accuracy(df, tree):
    df["classification"] = df.apply(predict, axis=1, args=(tree, 'Yes'))
    df["classification_correct"] = df["classification"] == df["Play Tennis"]
    accuracy = df["classification_correct"].mean()
    return accuracy

tree = fit(train_data, 'Play Tennis', ['Weather', 'Temperature', 'Humidity', 'Windy'])
accuracy = get_accuracy(test_data, tree)
print("Decision Tree:")
print(tree)
print("Accuracy:", accuracy)
```



# Limitaciones del algoritmo ID3 y los árboles de decisión:

- Si bien los árboles de decisión y el algoritmo ID3 ofrecen varias ventajas, también tienen algunas limitaciones que deben tenerse en cuenta antes de usarlos en ciertos escenarios:
  - **Sobreajuste:** Los árboles de decisión son propensos al sobreajuste, especialmente cuando se vuelven demasiado profundos o complejos. El sobreajuste ocurre cuando el árbol captura ruido o fluctuaciones aleatorias en los datos de entrenamiento, lo que resulta en un rendimiento deficiente con datos no visualizados.
  - **Inestabilidad:** Pequeños cambios en los datos pueden generar diferentes estructuras de árbol, lo que hace que los árboles de decisión sean menos estables. Una pequeña variación en los datos podría causar una división en un atributo o umbral diferente, lo que podría afectar a todo el árbol.
  - **Incapacidad para capturar relaciones lineales:** Los árboles de decisión no son adecuados para capturar relaciones lineales entre variables. Dividen los datos en regiones distintas, lo que dificulta la representación de patrones lineales.
  - **Sesgo hacia atributos con más niveles:** Los atributos con más niveles o categorías tienden a generar mayor ganancia de información simplemente por tener más divisiones posibles. Esto puede sesgar el árbol de decisión hacia dichos atributos, incluso si no son los más informativos.
  - **Falta de robustez al ruido:** los árboles de decisión pueden ser sensibles a los datos ruidosos, ya que podrían crear divisiones basadas en ruido o valores atípicos que no se generalizan bien a datos nuevos.

# Limitaciones del algoritmo ID3 y los árboles de decisión:

- **Dificultad para manejar variables continuas:** El algoritmo ID3 y los árboles de decisión básicos están diseñados para manejar variables categóricas. Para las variables continuas, se requiere preprocessamiento para convertirlas en intervalos discretos o utilizar otros algoritmos como CART (árboles de clasificación y regresión).
- **Crecimiento exponencial del tamaño del árbol:** Los árboles de decisión pueden crecer rápidamente, especialmente al trabajar con grandes conjuntos de datos o un gran número de características. Esto puede generar árboles complejos y difíciles de interpretar.
- **Expresividad limitada:** si bien los árboles de decisión pueden representar límites de decisión simples, pueden tener dificultades para capturar relaciones complejas en los datos.
- **Dificultad para gestionar valores faltantes:** El algoritmo ID3 no gestiona adecuadamente los valores faltantes. Es necesario utilizar métodos de imputación u otros algoritmos para gestionar los datos faltantes.
- **Desequilibrio de clases:** los árboles de decisión pueden tener dificultades para lidiar con distribuciones de clases desequilibradas en los datos, potencialmente favoreciendo a la clase mayoritaria y teniendo un desempeño deficiente en la clase minoritaria.
- A pesar de estas limitaciones, los árboles de decisión y el algoritmo ID3 siguen siendo ampliamente utilizados y aún pueden ser eficaces en diversos escenarios, especialmente al combinarse con técnicas como la poda, métodos de conjunto (p. ej., bosques aleatorios, potenciación de gradiente) o al emplearse como parte de procesos de aprendizaje automático más sofisticados. Comprender las fortalezas y debilidades de los árboles de decisión ayuda a los científicos de datos y profesionales del aprendizaje automático a tomar decisiones informadas sobre su uso en diferentes aplicaciones.

# Clasificación J48 (algoritmo C4.5) en pocas palabras

- Vivimos en un mundo de tecnología, e internet ha abierto las puertas a un vasto conocimiento e investigación al alcance de todos. La innovación y los avances continuos han abierto muchas puertas, lo que resulta en una enorme cantidad de datos útiles y significativos en todos los aspectos de nuestra vida. La pregunta es qué hacemos con estos datos, cómo los aprovechamos y cómo profundizamos en ellos para comprender su significado. Aquí es donde entra en juego el poder del aprendizaje automático y la inteligencia artificial. Estas técnicas ofrecen alternativas inteligentes al análisis de grandes volúmenes de datos. Mediante el desarrollo de algoritmos rápidos y eficientes, y modelos basados en datos para el procesamiento de datos en tiempo real, el aprendizaje automático puede producir resultados y análisis precisos.
- El algoritmo J48 es uno de los algoritmos de aprendizaje automático más utilizados para examinar datos de forma categórica y continua. El algoritmo C4.5 (J48) se utiliza principalmente en diversos campos para la clasificación de datos, por ejemplo, la interpretación de datos clínicos para el diagnóstico de enfermedades coronarias, la clasificación de datos de gobernanza electrónica y muchos más.





# Clasificación J48 y su árbol de decisiones

## Algoritmo C4.5/J48

- El algoritmo C4.5 es un algoritmo de clasificación que genera árboles de decisión basados en la teoría de la información. Es una extensión del algoritmo ID3 de Ross Quinlan, también conocido en Weka como J48 (J de Java). Los árboles de decisión generados por C4.5 se utilizan para la clasificación, por lo que a menudo se le denomina clasificador estadístico.
- La implementación J48 del algoritmo C4.5 ofrece numerosas funciones adicionales, como la contabilización de valores faltantes, la poda de árboles de decisión, rangos continuos de valores de atributos, la derivación de reglas, etc. En la herramienta de minería de datos WEKA, J48 es una implementación Java de código abierto del algoritmo C4.5. J48 permite la clasificación mediante árboles de decisión o reglas generadas a partir de ellos.
- Este algoritmo construye árboles de decisión a partir de un conjunto de datos de entrenamiento, de forma similar al algoritmo ID3, utilizando el concepto de entropía de la información. Los datos de entrenamiento son un conjunto  $S=\{s_1, s_2, \dots\}$  de muestras ya clasificadas. Cada muestra si consiste en un vector p-dimensional  $(x_1, i, x_2, i, \dots, x_p, i)$ , donde  $x_j$  representa los valores o características de los atributos de la muestra correspondiente, así como la clase a la que pertenece. Para obtener la máxima precisión de clasificación, el mejor atributo para dividir es aquel con la mayor información.



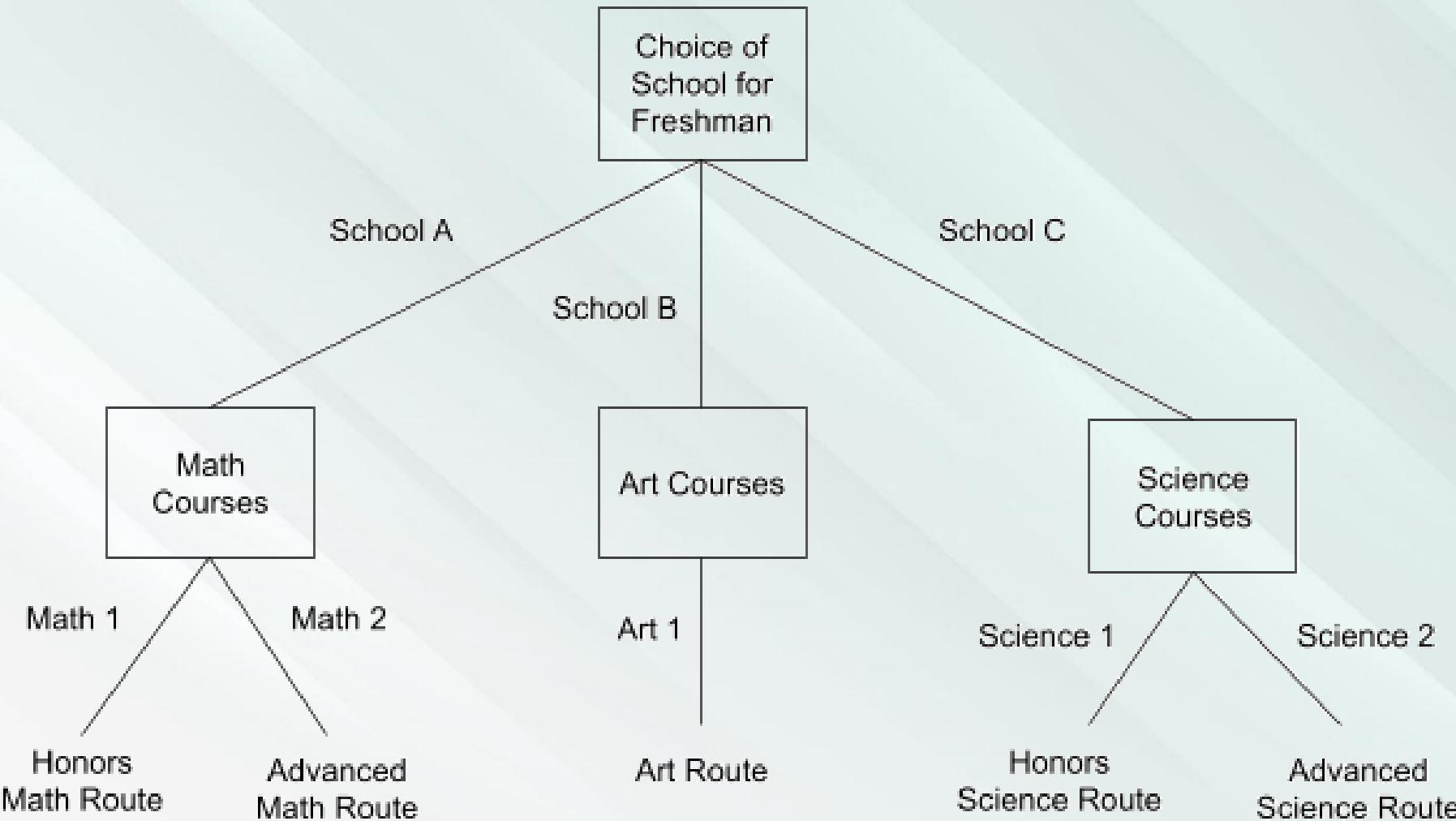


# Clasificación J48 y su árbol de decisiones

- En cada nodo del árbol, el algoritmo C4.5 selecciona el atributo de los datos que divide mejor su conjunto de muestras en subconjuntos, enriquecidos en una u otra clase. El criterio de división es la ganancia de información normalizada, que se calcula a partir de la diferencia de entropía. Se selecciona el atributo con la mayor ganancia de información normalizada para tomar la decisión. A continuación, el algoritmo C4.5 recurre sobre las sublistas particionadas utilizando un enfoque de "divide y vencerás" y crea un árbol de decisión basado en el algoritmo voraz.
- En un ejemplo más concreto y de alto nivel, el algoritmo trabaja con el árbol de decisión. Para ello, analiza los datos de los estudiantes que ya han cursado dichos cursos y los utiliza para construir un modelo que predice qué cursará un futuro estudiante según su elección de universidad al comenzar su primer año. Tomamos nuestra lista de estudiantes y la dividimos aleatoriamente en conjuntos de datos. Con cada conjunto de datos, generamos un conjunto de ponderaciones que predicen la trayectoria del estudiante y, a continuación, seleccionamos el conjunto de datos que predice con mayor precisión qué cursará.



# Clasificación J48 y su árbol de decisiones





# Clasificación J48 y su árbol de decisiones

## El árbol de decisiones

- Los nodos internos de este árbol de decisión representan los diferentes atributos, y las ramas entre los nodos indican los posibles valores que estos atributos pueden tener en las muestras observadas, mientras que los nodos terminales indican el valor final (clasificación) de la variable dependiente. El atributo predicho es la variable dependiente y los demás atributos del árbol son las variables independientes del conjunto de datos.
- Al construir un árbol, J48 ignora los valores faltantes; el valor de ese elemento puede predecirse a partir de los valores de los atributos de los demás registros. La idea básica es dividir los datos en rangos según los valores de los atributos presentes en la muestra de entrenamiento.
- Algunos casos de uso básicos de este algoritmo son:
  - Este algoritmo crea una lista para todas las muestras de una clase.
  - Evalúa características para evaluar cualquier ganancia de información, si no es posible ninguna ganancia, entonces este algoritmo crea un nodo más arriba en el árbol utilizando el valor esperado de la clase.
  - Si este algoritmo encuentra una clase desconocida que no ha visto hasta ahora, entonces crea un nodo más arriba en el árbol utilizando el valor esperado.



# Clasificación J48 y su árbol de decisiones

- Los árboles de decisión se utilizan para delinear los procesos de toma de decisiones. Se trata de un clasificador que actúa como un diagrama de flujo, similar a la construcción de un árbol, para representar modelos de asociación. Los árboles de decisión se utilizan para categorizar instancias, ordenándolas desde el origen hasta un pequeño nodo hoja. Cada nodo especifica un análisis de la instancia, y cada división corresponde a uno de los posibles beneficios de este atributo.
- Divide un conjunto de datos en subconjuntos cada vez más pequeños y se desarrolla de forma incremental. El resultado final es un árbol con nodos de decisión y nodos hoja. Cada nodo de decisión tiene dos o más divisiones, y el nodo hoja representa una asociación o decisión. El nodo de decisión superior del árbol, que corresponde al mejor predictor, se denomina nodo de origen.





# Clasificación J48 y su árbol de decisiones

## Poda de árboles

- El sobreajuste es un riesgo importante con estos árboles de decisión, por lo que es vital validar el modelo antes de implementarlo en el conjunto de prueba. A medida que este árbol de decisión crece, tiende naturalmente a sobreajustarse a los datos. La poda es un proceso mediante el cual se selecciona el árbol más grande y más generalizable, y se eliminan todas las ramas por debajo de ese nivel. Esto mejora significativamente el rendimiento con nuevos datos.
- La herramienta WEKA para J48 ofrece varias opciones para la poda de árboles. En caso de un posible sobreajuste, la poda puede utilizarse como herramienta de resumen. En otros algoritmos, la clasificación se realiza recursivamente hasta que cada hoja sea pura; es decir, la clasificación de los datos debe ser lo más perfecta posible. Este algoritmo genera las reglas a partir de las cuales se genera la identidad de esos datos. El objetivo es la generalización progresiva de un árbol de decisión hasta alcanzar un equilibrio entre flexibilidad y precisión.





# Clasificación J48 y su árbol de decisiones

## Condiciones previas a la poda y de terminación

- El problema con la poda es que es bastante ineficiente, ya que se realiza después de construir el árbol de decisión, pudiendo estar sobreajustado. Por lo tanto, la siguiente mejora de este algoritmo sería podar el árbol durante su construcción. El concepto de prepoda consiste en finalizar el algoritmo antes de que el árbol de decisión sea demasiado grande o se convierta en un árbol completamente desarrollado. Esto evita el riesgo de sobreajuste y evita que los árboles de decisión se vuelvan demasiado complejos. Los objetivos principales de la prepoda de un árbol son detener el algoritmo si todas las muestras pertenecen a la misma clase o si todos los valores de los atributos son iguales. Sin embargo, estos dos puntos no son comunes y no abordan todos los aspectos del sobreajuste. Por lo tanto, existen tres condiciones de finalización que deben aplicarse al construir un árbol de decisión: la primera es detener el algoritmo si el tamaño de la muestra es demasiado pequeño, la segunda es limitar la profundidad del árbol para evitar que crezca demasiado, y la tercera es no dividir un nodo en ramas si no se reduce lo suficiente el error de clasificación. Con estas tres condiciones de terminación, el algoritmo J48 se vuelve mucho más eficiente y preciso ya que evita el sobreajuste del árbol de decisión.



# Limitaciones

- Limitaciones de los árboles de decisión
- Si bien estos árboles de decisión son muy útiles para analizar y clasificar los datos, también presentan algunas desventajas asociadas con la poda. La poda reduce la complejidad del clasificador final y, por lo tanto, mejora la precisión predictiva al reducir el sobreajuste.
- Algunas de las cuestiones asociadas a estos árboles de decisión son:
  - El árbol en sí es computacionalmente costoso en cada nodo; cada candidato que divide el conjunto de datos debe ordenarse antes de poder encontrar su mejor nivel.
  - Además, los algoritmos de poda pueden resultar costosos, ya que es necesario industrializar y comparar innumerables subárboles candidatos.
  - La medición de la entropía consume muchos recursos.
  - El algoritmo del árbol de decisión produce un árbol de tamaño colosal.
  - Los tamaños de ejemplo minúsculos de un conjunto de entrenamiento plantean un desafío importante para los árboles de decisión, ya que la cantidad de instancias de entrenamiento obtenibles disminuye exponencialmente a medida que el árbol se divide.
  - Los criterios de decisión son rígidos en el sentido de que al final sólo se puede seleccionar un nodo.
  - Los árboles de decisiones profundos que abarcan innumerables niveles también pueden abarcar innumerables atributos, lo que genera valores de atributos atípicos.





# Limitaciones

## Limitaciones del algoritmo J48

- El algoritmo J48 se basa en árboles de decisión, por lo que presenta algunos de los problemas asociados a estos. A continuación, se presentan algunas deficiencias de este algoritmo:
  - **Ramas vacías:** La construcción de árboles con valores significativos es uno de los pasos importantes para la generación de reglas mediante el algoritmo J48. Sin embargo, estos valores no contribuyen a la creación de clases para las tareas de clasificación, sino que amplían y complican el árbol.
  - **Ramas insignificantes:** El número de atributos distintos seleccionados producen el mismo número de divisiones potenciales para construir un árbol de decisión. Sin embargo, no todas son significativas para las tareas de clasificación. Estas ramas insignificantes no solo reducen la usabilidad de los árboles de decisión, sino que también generan el problema del sobreajuste.
  - **Sobreajuste:** El sobreajuste ocurre cuando la visualización de un algoritmo obtiene información con atributos excepcionales. Esto causa muchas fragmentaciones, es decir, nodos estadísticamente poco importantes, en la distribución del proceso. Normalmente, el algoritmo J48 construye árboles y hace crecer sus ramas "lo suficientemente profundo como para clasificar perfectamente los ejemplos de entrenamiento", ya que este enfoque funciona mejor con datos sin ruido; sin embargo, la mayoría de las veces esta estrategia sobreajusta los ejemplos de entrenamiento con datos ruidosos. Actualmente, hay dos estrategias que se utilizan ampliamente para evitar este sobreajuste en el aprendizaje de árboles de decisión. La primera es que, si un árbol crece, se detenga su crecimiento antes de que alcance el punto máximo de clasificación precisa de los datos de entrenamiento. La segunda estrategia es dejar que el árbol se sobreajuste a los datos de entrenamiento y luego podarlo posteriormente.



# Optimizaciones

## Mejoras de J48 utilizando un mejor esquema de entropía

- El punto clave en el ensamblaje de árboles de decisión es la elección del mejor atributo para determinar el nivel del nodo actual. Para cada valor del conjunto de entrenamiento, se toma la mejor entropía calculada para determinar la ubicación del nodo. Los conjuntos de entrenamiento en general presentan algunas limitaciones, como: el conjunto de entrenamiento contiene una o más muestras que pertenecen a la misma clase, el conjunto de entrenamiento no contiene muestras y el conjunto de entrenamiento contiene muestras que pertenecen a una combinación de clases. Debido a estas limitaciones, el árbol de decisión resultante puede no representar con precisión la generalización; por lo tanto, se han ideado diferentes maneras de calcular la ganancia de entropía con mayor precisión. El algoritmo J48 puede mejorarse con un mejor cálculo de la ganancia de entropía.





# Optimizaciones

## Mejoras de J48 mediante el uso de clasificaciones híbridas no lineales

- La forma más común de construir árboles de decisión es mediante partición descendente. El conjunto de entrenamiento se utiliza recursivamente para encontrar una división lineal que maximice la ganancia de entropía. El árbol obtenido mediante este proceso suele ser demasiado grande y sobreajusta los datos, por lo que se poda examinando cada nodo intermedio y evaluando la utilidad de reemplazarlo con una hoja.
- Tras la poda, el conjunto local de casos de entrenamiento para un nodo hoja puede ser bastante amplio, y tomar solo la clase mayoritaria puede generalizar demasiado el árbol. La lógica J48 puede extenderse a partir del algoritmo básico de aprendizaje del árbol de decisión, de modo que, en lugar de la regla de la mayoría, se pueda utilizar un tipo de modelo diferente en cualquiera de las hojas. La decisión de reemplazar una hoja simple por un modelo alternativo se toma durante la pospoda. Los clasificadores alternativos de hojas se introducen principalmente durante la fase de pospoda. Las estimaciones de error se comparan antes de reemplazar un nodo hoja. Los algoritmos híbridos resultantes combinan los sesgos de la inducción descendente de árboles de decisión basada en la entropía y los respectivos modelos alternativos de hojas.



# Optimizaciones

## Mejoras de J48 mediante Meta Learning

- En la minería de datos, generalmente es necesario establecer los parámetros que utiliza el algoritmo para obtener el mejor modelo y los mejores resultados posibles. Los experimentos muestran un aumento sustancial en la precisión cuando se utilizan los parámetros correctos. Sin embargo, existe un problema asociado al ajuste de los parámetros de la mayoría de los algoritmos de minería de datos. Esta tarea puede implicar un alto costo computacional para encontrar los parámetros óptimos o, de lo contrario, correr el riesgo de basarse en suposiciones que podrían sesgar los resultados.
- La mayoría de los algoritmos y herramientas de minería de datos actuales requieren configuración previa. En otras palabras, los usuarios deben proporcionar valores adecuados para los parámetros con antelación para obtener buenos resultados o modelos; por lo tanto, deben poseer cierta experiencia para encontrar la configuración correcta. Para resolver este problema, la minería de datos permite aprender de ejecuciones anteriores de los algoritmos y así mejorar la selección futura de parámetros según el comportamiento previo del algoritmo.
- El metaaprendizaje es el estudio de métodos basados en principios que aprovechan el metaconocimiento para obtener modelos y soluciones eficientes mediante la adaptación del aprendizaje automático y la minería de datos. El modelo de árbol de decisión tiene parámetros que influyen en el grado de poda. Al podar árboles, se puede optimizar la eficiencia computacional y la precisión de clasificación del modelo.



# Ejercicio 1: Clasificación de Frutas (D43)

- Clasificar frutas según su color y tamaño
- Datos de Entrenamiento (NO aleatorios):

Fruta	Color (1=Rojo, 2=Verde, 3=Amarillo)	Tamaño (1=Pequeño, 2=Mediano, 3=Grande)	Tipo
Manzana	1	2	Manzana
Manzana	2	2	Manzana
Manzana	1	3	Manzana
Naranja	3	2	Naranja
Naranja	3	3	Naranja
Plátano	3	3	Plátano
Plátano	3	2	Plátano
Uva	1	1	Uva
Uva	2	1	Uva



# Resolución en Python

```
from sklearn.tree import DecisionTreeClassifier  
from sklearn import tree  
import matplotlib.pyplot as plt
```

# Datos estructurados correctamente

```
X = [  
    [1, 2], [2, 2], [1, 3], # Manzanas  
    [3, 2], [3, 3],      # Naranjas  
    [3, 3], [3, 2],      # Plátanos  
    [1, 1], [2, 1]       # Uvas  
]
```

```
y = [  
    'Manzana', 'Manzana', 'Manzana',  
    'Naranja', 'Naranja',  
    'Plátano', 'Plátano',  
    'Uva', 'Uva'  
...]  
:
```

```
# Crear y entrenar el modelo D43  
modelo = DecisionTreeClassifier(criterion='entropy',  
                                 max_depth=3)  
modelo.fit(X, y)  
  
# Visualizar el árbol  
plt.figure(figsize=(12, 8))  
tree.plot_tree(modelo,  
               feature_names=['Color', 'Tamaño'],  
               class_names=['Manzana', 'Naranja', 'Plátano', 'Uva'],  
               filled=True)  
plt.show()  
  
# Predecir una nueva fruta  
nueva_fruta = [[3, 1]] # Color amarillo, tamaño pequeño  
prediccion = modelo.predict(nueva_fruta)  
print(f"Predicción: {prediccion[0]}")
```



# Ejercicio 2: Diagnóstico Médico (J48/C4.5)

- Objetivo: Diagnosticar enfermedades basado en síntomas

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Datos médicos realistas (no aleatorios)
datos_médicos = {
    'fiebre': [1, 1, 0, 1, 0, 1, 0, 1, 0, 1],
    'tos': [1, 1, 0, 1, 0, 1, 1, 0, 0, 1],
    'dolor_cabeza': [0, 1, 0, 1, 1, 0, 1, 0, 0, 1],
    'edad': [25, 45, 30, 60, 35, 50, 28, 42, 33, 55],
    'enfermedad': ['Gripe', 'Gripe', 'Sano', 'COVID', 'Migraña',
                   'COVID', 'Migraña', 'Sano', 'Sano', 'COVID']
}

df = pd.DataFrame(datos_médicos)
print("Datos médicos:")
print(df)

....
```



# Ejercicio 2: Diagnóstico Médico (J48/C4.5)

```
# Preparar datos
X = df[['fiebre', 'tos', 'dolor_cabeza', 'edad']]
y = df['enfermedad']

# Dividir en entrenamiento y prueba (80-20)
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Modelo J48 (C4.5) - Usando entropía
modelo_j48 = DecisionTreeClassifier(
    criterion='entropy', # Esto implementa C4.5/J48
    max_depth=4,
    min_samples_split=2,
    min_samples_leaf=1
)

# Entrenar
modelo_j48.fit(X_train, y_train)

# Evaluar
y_pred = modelo_j48.predict(X_test)
precision = accuracy_score(y_test, y_pred)
print(f"Precisión del modelo: {precision:.2f}")

# Hacer predicción
nuevo_paciente = [[1, 1, 0, 40]] # Fiebre, tos, sin dolor
cabeza, 40 años
diagnostico = modelo_j48.predict(nuevo_paciente)
print(f"Diagnóstico probable: {diagnostico[0]}")
```



UNIVERSIDAD  
CATÓLICA  
SEDES SAPIENTIAE



UNIVERSIDAD  
CATÓLICA  
SEDES SAPIENTIAE

# Inteligencia Artificial

Ing. Juancarlos Santana Huamán  
[jsantana@ucss.edu.pe](mailto:jsantana@ucss.edu.pe)



# ¿Qué es el Perceptrón? | La red neuronal artificial más simple

- El **perceptrón** es una de las arquitecturas **de redes neuronales artificiales** más simples , introducida por Frank Rosenblatt en 1957. Se utiliza principalmente para **la clasificación binaria** .
- En aquella época, se utilizaban comúnmente métodos tradicionales como el aprendizaje automático estadístico y la programación convencional para realizar predicciones. A pesar de ser una de las formas más simples de redes neuronales artificiales, el modelo del perceptrón demostró ser muy eficaz para resolver problemas específicos de clasificación, sentando las bases para los avances en IA y aprendizaje automático.





# ¿Qué es el Perceptrón?

- **El perceptrón** es un tipo de red neuronal que realiza una clasificación binaria que asigna características de entrada a una decisión de salida, generalmente clasificando los datos en una de dos categorías, como 0 o 1.
- El perceptrón consta de una sola capa de nodos de entrada completamente conectados a una capa de nodos de salida. Es especialmente eficaz en el aprendizaje **de patrones linealmente separables**. Utiliza una variante de neuronas artificiales denominada **Unidades Lógicas de Umbral (TLU)**, introducidas por primera vez por McCulloch y Walter Pitts en la década de 1940. Este modelo fundamental ha desempeñado un papel crucial en el desarrollo de redes neuronales más avanzadas y algoritmos de aprendizaje automático.
- **Tipos de perceptrón**
  - **El perceptrón de una sola capa** es un tipo de perceptrón limitado al aprendizaje de patrones linealmente separables. Es eficaz en tareas donde los datos se pueden dividir en distintas categorías mediante una línea recta. Si bien su simplicidad lo hace potente, presenta dificultades en problemas más complejos donde la relación entre entradas y salidas no es lineal.
  - **Los perceptrones multicapa** poseen capacidades de procesamiento mejoradas ya que constan de dos o más capas, capaces de manejar patrones y relaciones más complejas dentro de los datos.





# ¿Qué es el algoritmo de aprendizaje del perceptrón?

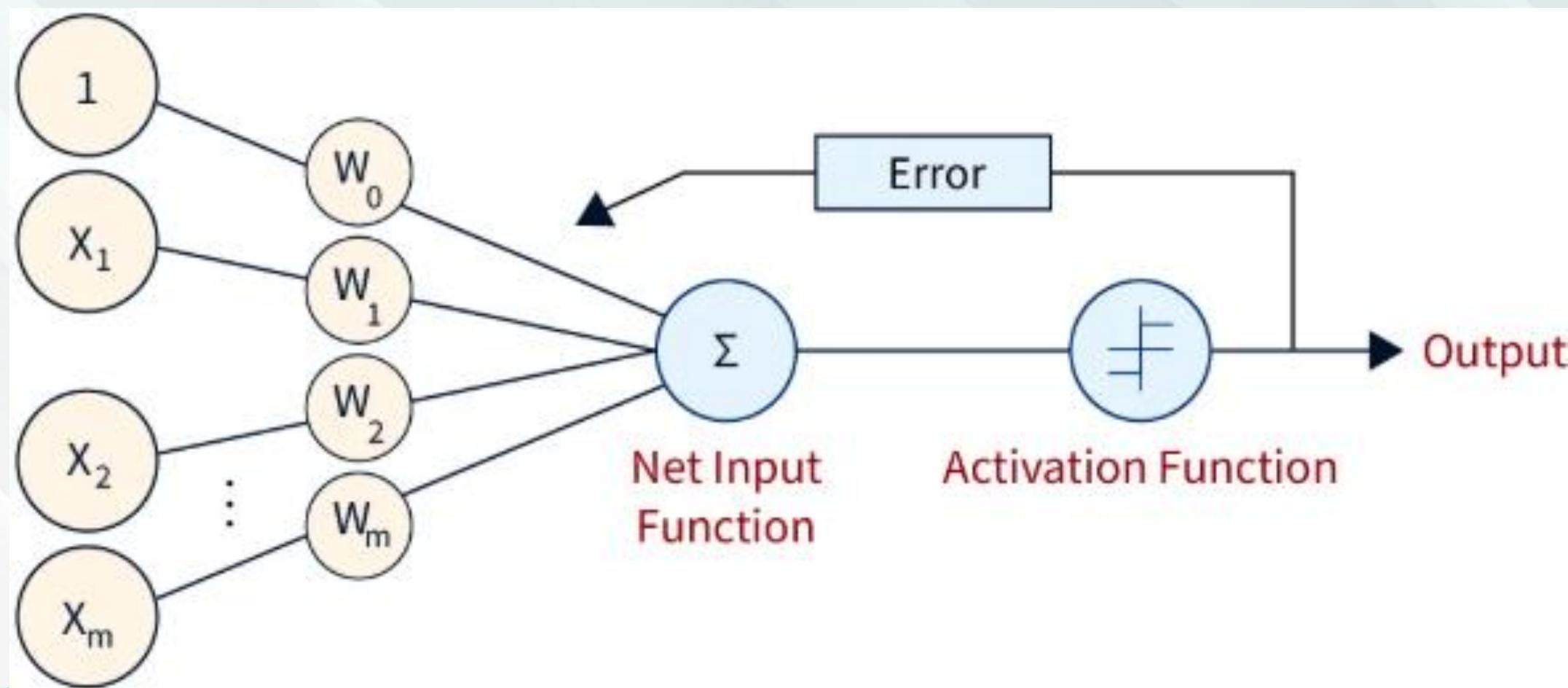
**Hay cuatro pasos importantes en un algoritmo de aprendizaje de perceptrón:**

1. Primero, multiplique todos los valores de entrada por sus ponderaciones correspondientes y luego súmelos para determinar la suma ponderada. Matemáticamente, podemos calcular la suma ponderada de la siguiente manera: $\sum w_i * \text{incógnita}_i = \text{incógnita}_1 * w_1 + \text{incógnita}_2 * w_2 + \dots + w_{\text{norte}} * \text{incógnita}_{\text{norte}}$  Agregue otro término esencial llamado sesgo 'b' a la suma ponderada para mejorar el rendimiento del modelo. $\sum w_i * \text{incógnita}_i + b$ .
2. A continuación, se aplica una función de activación a esta suma ponderada, produciendo una salida binaria o de valor continuo. $Y = F(\sum w_i * \text{incógnita}_i + b)$
3. A continuación, se calcula la diferencia entre esta salida y el valor objetivo real para obtener el término de error, E , generalmente expresado como error cuadrático medio. Los pasos hasta este punto constituyen la parte de propagación hacia adelante del algoritmo. $m_i = (Y - Y_{\text{adoeltúayo}})^2$
4. Optimizamos este error (función de pérdida) mediante un algoritmo de optimización. Generalmente, se utiliza algún tipo de algoritmo de descenso de gradiente para encontrar los valores óptimos de los hiperparámetros como la tasa de aprendizaje , el peso , el sesgo , etc. Este paso constituye la parte de propagación hacia atrás del algoritmo.





- En la siguiente figura se ilustra una descripción general de este algoritmo:





- En una notación más estandarizada, el algoritmo de aprendizaje del perceptrón es el siguiente:

```
P <- inputs with label 1
N <- inputs with label 0
Initialise w randomly;
while !converge do:
    $ \hspace{2em} $ Pick random x $\in P \cup N;
    $ \hspace{2em} $ if x $\in$ P and w.x $<$ 0 then
        $ \hspace{3em} $ w = w+x
    $ \hspace{2em} $ end
    $ \hspace{3em} $ if x $\in$ N and w.x $\geq$ 0 then
        $ \hspace{3em} $ w = w-x
    $ \hspace{2em} $ end
end
```

- Nuestro objetivo es encontrar el vector  $w$  que clasifique perfectamente las entradas positivas y negativas en un conjunto de datos.  $w$  se inicializa con un vector aleatorio. A continuación, iteramos las muestras positivas y negativas (PUN). Ahora bien, si una entrada  $x$  pertenece a  $P$ ,  $wx$  debe ser mayor o igual a 0. Y si  $x$  pertenece a  $N$ ,  $wx$  debe ser menor o igual a 0. Solo cuando no se cumplen estas condiciones, actualizamos las ponderaciones.

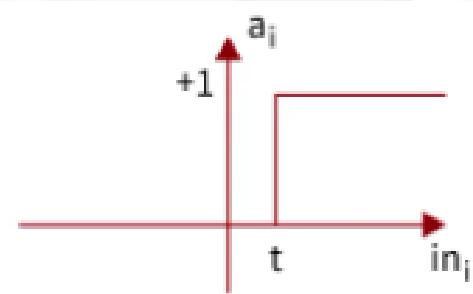
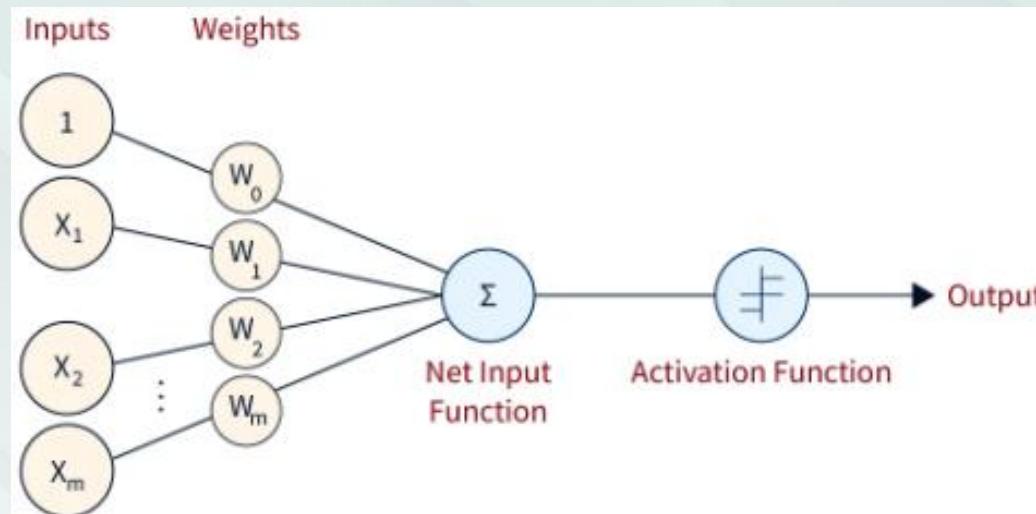


# Componentes básicos del perceptrón

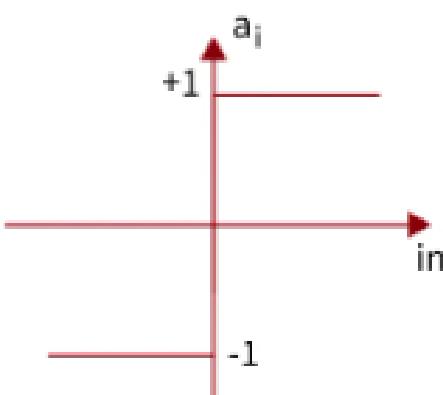
- Frank Rosenblatt inventó el algoritmo de aprendizaje del perceptrón.
- Es un clasificador binario y consta de tres componentes principales:
  - **Nodos de Entrada o Capa de Entrada:** Componente principal del algoritmo de aprendizaje del Perceptrón, que acepta los datos de entrada iniciales en el modelo. Cada nodo de entrada contiene un valor real.
  - **Peso y sesgo:** El parámetro de peso representa la fuerza de la conexión entre las unidades. El sesgo puede considerarse como la línea de intersección en una ecuación lineal.
  - **Función de activación:** Los componentes finales y esenciales ayudan a determinar si la neurona se activará. La función de activación puede considerarse principalmente una función escalonada. Existen varios tipos de funciones de activación utilizadas en un algoritmo de aprendizaje percepitrón. Algunas de ellas son la función signo, la función escalonada, la función sigmoidea, etc.



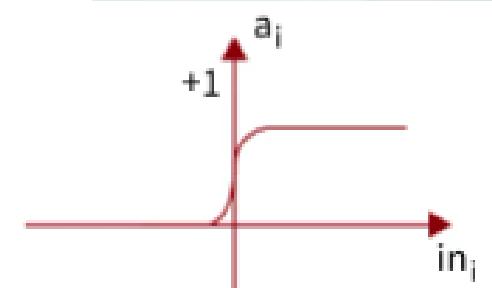
# Componentes básicos del perceptrón



Step Function



Sign Function



Sigmoid Function



# Tipos de modelos de perceptrón

- **Según el número de capas, los perceptrones se clasifican en dos categorías principales:**
- **Modelo de perceptrón de capa única:**

Es el modelo de red neuronal artificial (RNA) más simple. Un modelo de perceptrón de capa única consiste en una red de propagación hacia adelante e incluye una función de transferencia de umbral para la umbralización de la salida. El objetivo principal del modelo de perceptrón de capa única es clasificar datos linealmente separables con etiquetas binarias.

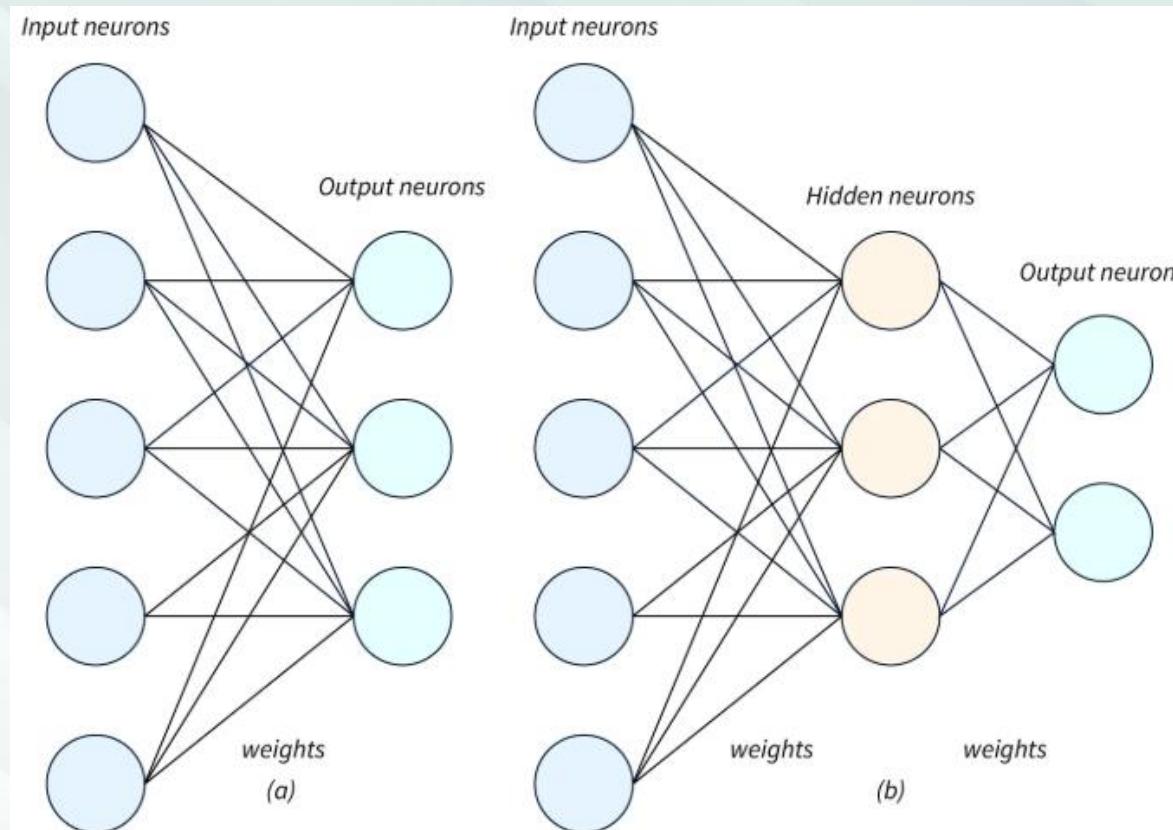
- **Modelo de perceptrón multicapa:**

El algoritmo de aprendizaje del perceptrón multicapa tiene la misma estructura que un perceptrón monocapa, pero consta de una o más capas ocultas adicionales, a diferencia de un perceptrón monocapa, que consta de una sola capa oculta. La distinción entre estos dos tipos de modelos de perceptrón se muestra en la figura a continuación.





# Tipos de modelos de perceptrón



(a) Architecture of a single layer perceptron. The architecture consists of a layer on input neurons fully connected to a single layer of output neurons.

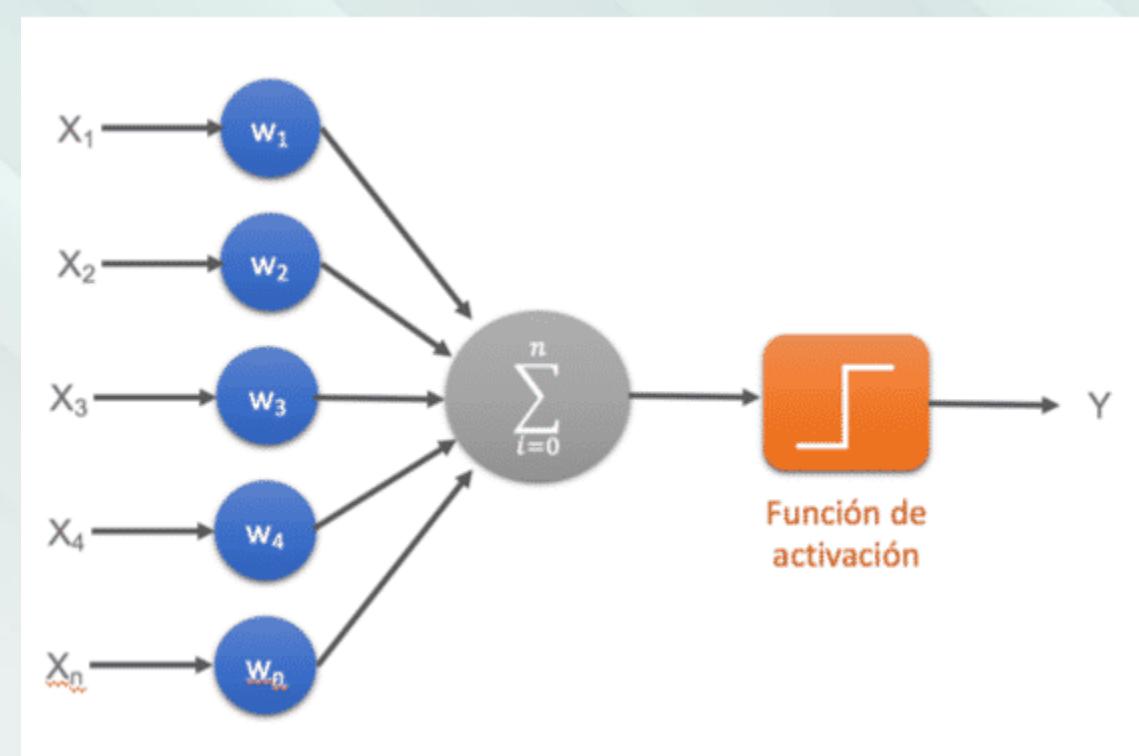
(b) Extension to a multi-layer perceptron including more than one layer of trainable weights. In this example, the network includes 3 layers: input, hidden and output layer. Each connection between two neurons is given by a certain weight.



# Clasificación de redes neuronales artificiales

## Red neuronal Monocapa – Perceptrón simple

- La red neuronal monocapa se corresponde con la red neuronal más simple, está compuesta por una capa de neuronas que proyectan las entradas a una capa de neuronas de salida donde se realizan los diferentes cálculos.





# Algoritmo de aprendizaje del perceptrón: implementación de la puerta AND

Los pasos para esta implementación son los siguientes:

- Importar todas las bibliotecas necesarias:

```
#import required library
import tensorflow as tf
```

- Definir variables vectoriales para entrada y salida:

```
#input1, input2 and bias
train_in = [
    [1., 1., 1.],
    [1., 0., 1.],
    [0., 1., 1.],
    [0., 0., 1.]]
```

```
#output
train_out = [
    [1.],
    [0.],
    [0.],
    [0.]]
```





# Algoritmo de aprendizaje del perceptrón: implementación de la puerta AND

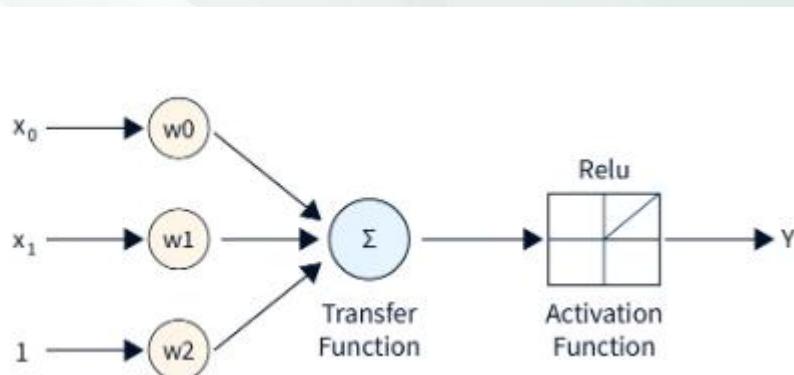
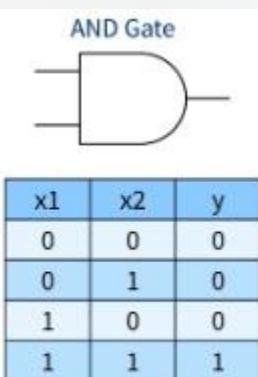
- Defina la variable Peso:

```
#weight variable initialized with random values
using random_normal()
w = tf.Variable(tf.random_normal([3, 1], seed=12))
```

- Definir marcadores de posición para entrada y salida:

```
#Placeholder for input and Output
x = tf.placeholder(tf.float32, [None, 3])
y = tf.placeholder(tf.float32, [None, 1])
```

- Calcular la función de salida y activación:



```
#calculate output
output = tf.nn.relu(tf.matmul(x, w))
```



# Algoritmo de aprendizaje del perceptrón: implementación de la puerta AND

- Calcular el coste o error:

```
#Mean Squared Loss or Error
loss = tf.reduce_sum(tf.square(output - y))
```

- Minimizar error:

```
#Minimize loss using GradientDescentOptimizer with a
learning rate of 0.01
optimizer = tf.train.GradientDescentOptimizer(0.01)
train = optimizer.minimize(loss)
```

- Inicializar todas las variables:

```
#Initialize all the global variables
init = tf.global_variables_initializer()
sess = tf.Session()
sess.run(init)
```





# Algoritmo de aprendizaje del perceptrón: implementación de la puerta AND

- Entrenamiento del algoritmo de aprendizaje del Perceptrón en Iteraciones:

```
training_epochs = 1000

#Compute cost w.r.t to input vector for 1000 epochs

for epoch in range(training_epochs):
    sess.run(train, {x:train_in,y:train_out})
    cost = sess.run(loss,feed_dict={x:train_in,y:train_out})
    if i > 990:
        print('Epoch--',epoch,'--loss--',cost)
```





# Perceptrón con Scikit-Learn

- El algoritmo de aprendizaje del perceptrón está disponible en la biblioteca de aprendizaje automático de Python scikit-learn a través de la clase Perceptron. Algunos de los parámetros configurables importantes para esta clase son: la tasa de aprendizaje ( eta0 ), cuyo valor predeterminado es 1.0 , y las épocas de entrenamiento ( max\_iter ), cuyo valor predeterminado es 1000. La detención anticipada, cuyo valor predeterminado es "False" , y el tipo de regularización (penalización), cuyo valor predeterminado es "Ninguno" y cuyos valores pueden ser " l2 ", " l1 " y " elastic net ".
- **Un ejemplo de llamada de este algoritmo perceptrón es el siguiente:**

*Modelo=Percepción ( eta0=0.1 ,máximo=1000 )*

- Ahora, demostraremos la implementación del algoritmo de aprendizaje del Perceptrón con un ejemplo práctico. Para ello, generaremos un conjunto de datos de clasificación sintética. Usaremos la función make\_classification() para crear un conjunto de datos con 1000 ejemplos, cada uno con 20 variables de entrada.





# Perceptrón con Scikit-Learn

```
# Evaluate a perceptron model on a synthetic dataset
from numpy import mean
from numpy import std
from sklearn.datasets import make_classification
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.linear_model import Perceptron
# Define the dataset
X, y = make_classification(n_samples=1000, n_features=10, n_informative=10,
n_redundant=0, random_state=1)
# Define the model
model = Perceptron()
# Define model evaluation method
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
# Evaluate the model
scores = cross_val_score(model, X, y, scoring='accuracy', cv=cv, n_jobs=-1)
# Summarize the results
print('Mean Accuracy: %.3f (%.3f)' % (mean(scores), std(scores)))
```





# Limitaciones del modelo de perceptrón

**Un modelo de perceptrón tiene las siguientes limitaciones:**

- La salida de un perceptrón solo puede ser un número binario ( 0 o 1 ) debido a la función de transferencia de límite rígido. Por lo tanto, es difícil de usar en problemas distintos a la clasificación binaria, como la regresión o la clasificación multiclas.
- El perceptrón solo puede utilizarse para clasificar conjuntos de vectores de entrada linealmente separables. Si los vectores de entrada no son lineales, no es fácil clasificarlos correctamente.

**El futuro del perceptrón**

- Los perceptrones tienen un futuro prometedor, ya que son un modelo muy intuitivo e interpretable que facilita la interpretación de los datos. Las neuronas artificiales constituyen la base de los perceptrones y representan el futuro de los modelos de redes neuronales de vanguardia y muy populares. Por lo tanto, con la creciente popularidad de la inteligencia artificial y las redes neuronales, los algoritmos de aprendizaje perceptrónico desempeñan un papel fundamental.





# Ejercicios

- **Caso:** Clasificación de flores Iris (**Setosa vs. No Setosa**)
- Iris (150 muestras, 3 especies, 4 características por flor).
- **Objetivo:** Clasificar si una flor es *Setosa* (1) o *No Setosa* (0) usando solo dos características: **Longitud del pétalo (cm)** y **Ancho del pétalo (cm)**.

# Ejercicios

- **Caso:** Identificación del estado de ánimo de una persona.
- **Objetivo:** Clasificar el estado de una persona si se encuentra:
  - Enojado
  - Disgustado
  - Miedoso
  - Feliz
  - Triste
  - Sorprendida
  - Neutral



UNIVERSIDAD  
CATÓLICA  
SEDES SAPIENTIAE



UNIVERSIDAD  
CATÓLICA  
SEDES SAPIENTIAE

# Inteligencia Artificial

Ing. Juancarlos Santana Huamán  
[jsantana@ucss.edu.pe](mailto:jsantana@ucss.edu.pe)



# Backpropagation en Inteligencia Artificial

## ¿Qué es Backpropagation o retropropagación en IA?

- Backpropagation es un algoritmo de aprendizaje supervisado en inteligencia artificial utilizado para **entrenar redes neuronales artificiales** (ANNs). El objetivo de Backpropagation es ajustar los pesos y sesgos de una red neuronal para minimizar la diferencia entre las salidas de la red y las salidas deseadas o esperadas durante el entrenamiento.
- El algoritmo utiliza una técnica de optimización matemática llamada gradiente descendente, que permite ajustar los pesos y sesgos de la red para minimizar el error. El proceso de ajuste se realiza en dos fases: en la **propagación hacia adelante** (forward propagation), se calcula la salida de la red para una entrada dada, y en la **propagación hacia atrás** (back propagation) se ajustan los pesos y sesgos de la red para minimizar el error.

# ¿Cómo funciona el algoritmo Backpropagation (retropropagación)?

- El algoritmo de retropropagación, también conocido como Backpropagation, es un método utilizado para entrenar redes neuronales supervisadas. Este algoritmo funciona en varias etapas:
  - **Primero**, se aplica un vector de entrada a la red y se calcula la salida correspondiente. Este proceso se conoce como “forward pass” o “feed forward” porque se introducen los inputs a la red, “de izquierda hacia la derecha”, realizando todos los cálculos hasta obtener la primera estimación de las salidas de la red.
  - **Segundo**, se compara la salida obtenida con la salida deseada y se calcula el error. Este error es la diferencia entre las salidas de la red y las salidas deseadas o esperadas durante el entrenamiento.
  - **Tercero**, el algoritmo de retropropagación utiliza el error calculado para ajustar los pesos sinápticos en cada una de las capas intermedias. Este proceso se lleva a cabo desde la última capa hasta la primera capa, propagando el error hacia atrás y actualizando los pesos sinápticos a lo largo del camino.
  - **Finalmente**, para actualizar los pesos sinápticos de las neuronas, se utiliza un método conocido como descenso del gradiente (*gradient descent*). El objetivo es encontrar los valores óptimos para los pesos que minimizan el error en la red neuronal. Para lograr esto, se calcula la derivada parcial del error con respecto a cada peso y se actualiza en consecuencia.
- Es importante mencionar que a pesar de ser uno de los algoritmos más populares en el campo de las redes neuronales supervisadas, existen algunos **problemas asociados con el algoritmo de retropropagación**.
- El más comunes es el **sobreajuste (overfitting)**, que ocurre cuando una red neuronal se ajusta demasiado bien a los datos específicos utilizados para entrenarla y no generaliza bien a nuevos datos.

# Funciones de Backpropagation en el entrenamiento de redes neuronales

- El algoritmo de retropropagación, también conocido como Backpropagation, tiene varias funciones clave en el entrenamiento de redes neuronales artificiales:
  - **Minimizar los errores:** El objetivo principal de Backpropagation es minimizar los errores en el proceso de aprendizaje automático de una máquina. Trata de imitar el funcionamiento de las redes de neuronas biológicas.
  - **Ajustar los parámetros:** Backpropagation ajusta los parámetros (pesos y sesgos) para reducir la tasa de error. Después de cada paso hacia adelante a través de una red, Backpropagation realiza un paso hacia atrás mientras ajusta los parámetros del modelo (pesos y sesgos).
  - **Detectar responsabilidades:** Lo que hace Backpropagation es detectar cuánta responsabilidad tiene cada uno de los nodos de la red en los posibles errores y corregir la configuración de los parámetros de esos nodos, que luego influirán en otros nodos y capas posteriores.
  - **Mejorar la precisión de las predicciones:** El objetivo del algoritmo Backpropagation es minimizar los errores en el proceso de aprendizaje y mejorar la precisión de las predicciones ajustando los pesos y sesgos de cada nodo de la red.
- En resumen, Backpropagation es un algoritmo que se emplea para entrenar redes neuronales artificiales con el objetivo de minimizar los errores en el proceso de aprendizaje automático de una máquina.



# Usos de Backpropagation en la actualidad

- El algoritmo de retropropagación, o Backpropagation, tiene una amplia gama de aplicaciones en diversos campos. Se utiliza en el reconocimiento óptico de caracteres, la **conversión de texto escrito a voz**, el procesado de imágenes médicas y la inspección automática de defectos.
- Además, Backpropagation puede ser una herramienta útil en la **identificación de patrones y tendencias en grandes conjuntos de datos de tráfico web y consumo de contenido**, y en la optimización de la experiencia de usuario y la generación de contenido innovador y relevante para una audiencia específica.
- En general, el algoritmo de retropropagación se emplea para **entrenar redes neuronales artificiales** con el objetivo de minimizar los errores en el proceso de aprendizaje automático de una máquina. Por lo tanto, cualquier aplicación que implique el uso de redes neuronales puede beneficiarse del uso de Backpropagation.

# Aplicaciones del algoritmo Backpropagation en Content marketing y SEO

- Backpropagation tiene aplicaciones en Content marketing y SEO, algunas de ellas pueden ser:
  - **Análisis de palabras clave:** Al entrenar una red neuronal utilizando Backpropagation con grandes cantidades de datos de palabras clave, se puede identificar patrones y tendencias en el comportamiento del usuario y las preferencias de búsqueda relacionadas con ciertos temas o palabras clave. Esto puede ayudar a las empresas de SEO a diseñar estrategias de optimización de búsqueda más efectivas.
  - **Personalización de publicidad digital:** Utilizando la propagación hacia adelante, se puede predecir el comportamiento del usuario y personalizar anuncios digitales en función de las necesidades y preferencias de cada usuario. Esto puede mejorar la eficacia de la publicidad digital y aumentar el retorno de inversión (ROI) de las campañas de publicidad.
  - **Análisis y predicción de tendencias:** Utilizando Backpropagation para entrenar una red neuronal con grandes cantidades de datos de tráfico web y consumo de contenido, se pueden identificar patrones y tendencias en el comportamiento del usuario. Esto puede ayudar a las empresas de Content marketing a diseñar estrategias de contenido más efectivas y a adaptar su contenido a las necesidades e intereses de su audiencia.
- En general, Backpropagation puede ser una herramienta útil en la identificación de patrones y tendencias en grandes conjuntos de datos de tráfico web y consumo de contenido, y en la optimización de la experiencia de usuario y la generación de contenido innovador y relevante para una audiencia específica.

# Pases hacia adelante y hacia atrás en redes neuronales

- Para entrenar una red neuronal, hay 2 pasos (fases):
  - Adelante
  - Hacia atrás
- En el paso hacia adelante, comenzamos propagando las entradas de datos a la capa de entrada, pasamos por las capas ocultas, medimos las predicciones de la red desde la capa de salida y, finalmente, calculamos el error de la red en función de las predicciones que hizo la red.
- Este error de red mide la distancia a la red para realizar la predicción correcta. Por ejemplo, si la salida correcta es 4 y la predicción de la red es 1,3, el error absoluto de la red es  $4 - 1,3 = 2,7$ . Cabe destacar que el proceso de propagación de las entradas desde la capa de entrada a la capa de salida se denomina **propagación hacia adelante**. Una vez calculado el error de red, la fase de propagación hacia adelante finaliza y comienza la propagación hacia atrás.

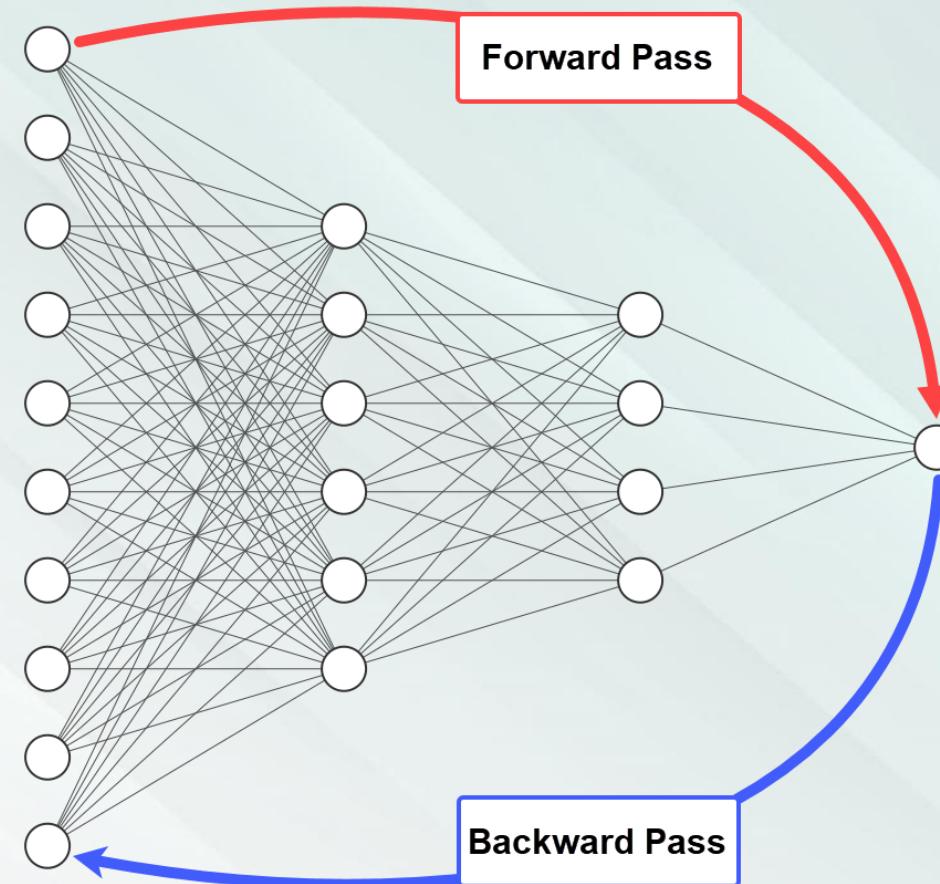


# Pases hacia adelante y hacia atrás en redes neuronales

- La siguiente figura muestra una flecha roja que apunta en la dirección de la propagación hacia adelante.
- En el paso hacia atrás, el flujo se invierte, de modo que comenzamos propagando el error a la capa de salida hasta llegar a la capa de entrada, pasando por las capas ocultas. El proceso de propagación del error de red desde la capa de salida a la capa de entrada se denomina **propagación hacia atrás** o **retropropagación simple**. El algoritmo de retropropagación es el conjunto de pasos que se utilizan para actualizar los pesos de la red y reducir el error de red.
- En la siguiente figura, la flecha azul apunta en la dirección de propagación hacia atrás.
- Las fases de avance y retroceso se repiten a partir de algunas épocas. En cada época, ocurre lo siguiente:
  - Las entradas se propagan desde la capa de entrada a la de salida.
  - Se calcula el error de red.
  - El error se propaga desde la capa de salida a la capa de entrada.



# Pases hacia adelante y hacia atrás en redes neuronales





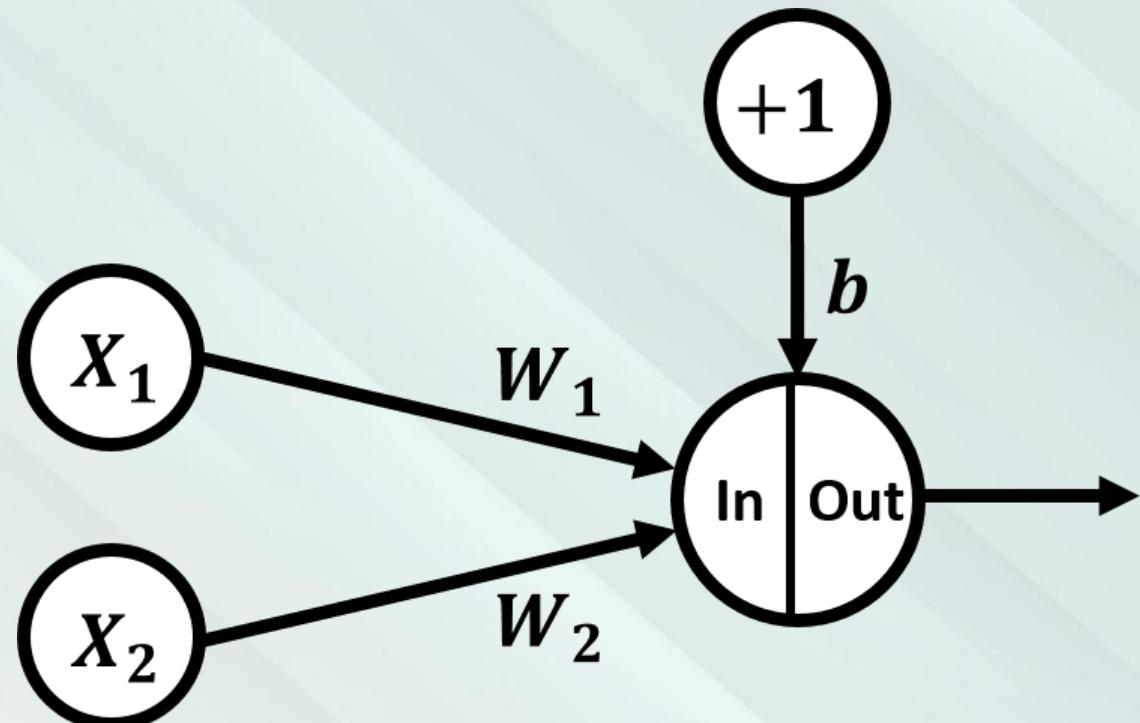
# ¿Por qué utilizar el algoritmo de retropropagación?

- Anteriormente, explicamos que la red se entrena mediante dos pasadas: una hacia adelante y otra hacia atrás. Al final de la pasada hacia adelante, se calcula el error de la red, que debe ser lo más bajo posible.
- Si el error actual es alto, la red no aprendió correctamente de los datos. ¿Qué significa esto? Significa que el conjunto actual de ponderaciones no es lo suficientemente preciso como para reducir el error de red y realizar predicciones precisas. Por lo tanto, debemos actualizar las ponderaciones de la red para reducir el error.
- El algoritmo de retropropagación es uno de los encargados de actualizar los pesos de la red con el objetivo de reducir el error de red. Es muy importante.
- Estas son algunas de las ventajas del algoritmo de retropropagación:
  - Es eficiente en el uso de memoria al calcular las derivadas, ya que utiliza menos memoria que otros algoritmos de optimización, como el algoritmo genético. Esta es una característica muy importante, especialmente en redes grandes.
  - El algoritmo de retropropagación es rápido, especialmente para redes pequeñas y medianas. A medida que se añaden más capas y neuronas, se vuelve más lento a medida que se calculan más derivadas.
  - Este algoritmo es lo suficientemente genérico para funcionar con diferentes arquitecturas de red, como redes neuronales convolucionales, redes generativas antagónicas, redes totalmente conectadas y más.
  - No hay parámetros para ajustar el algoritmo de retropropagación, por lo que la sobrecarga es menor. Los únicos parámetros del proceso están relacionados con el algoritmo de descenso de gradiente, como la tasa de aprendizaje.



# Cómo funciona el algoritmo de retropropagación

- El funcionamiento del algoritmo se explica mejor con una red simple, como la que se muestra en la siguiente figura. Solo tiene una capa de entrada con dos entradas ( $X_1$  y  $X_2$ ) y una capa de salida con una salida. No hay capas ocultas.
- Los pesos de las entradas son  $W_1$  y  $W_2$ , respectivamente. El sesgo se considerá una nueva neurona de entrada para la neurona de salida, la cual tiene un valor fijo +1 y un peso  $b$ . Tanto los pesos como los sesgos podrían denominarse **parámetros**.





# Cómo funciona el algoritmo de retropropagación

- Supongamos que la capa de salida utiliza la función de activación sigmoidea definida por la siguiente ecuación:

$$f(s) = \frac{1}{1 + e^{-s}}$$

- Donde  $s$  es la suma de productos (SOP) entre cada entrada y su peso correspondiente:

$$s = X_1 \times W_1 + X_2 \times W_2 + b$$

- Para simplificar, en este ejemplo se utiliza una sola muestra de entrenamiento. La siguiente tabla muestra la muestra de entrenamiento con la entrada y su correspondiente salida deseada (es decir, correcta). En la práctica, se utilizan más ejemplos de entrenamiento.

$X_1$	$X_2$	Salida deseada
0.1	0.3	0.03



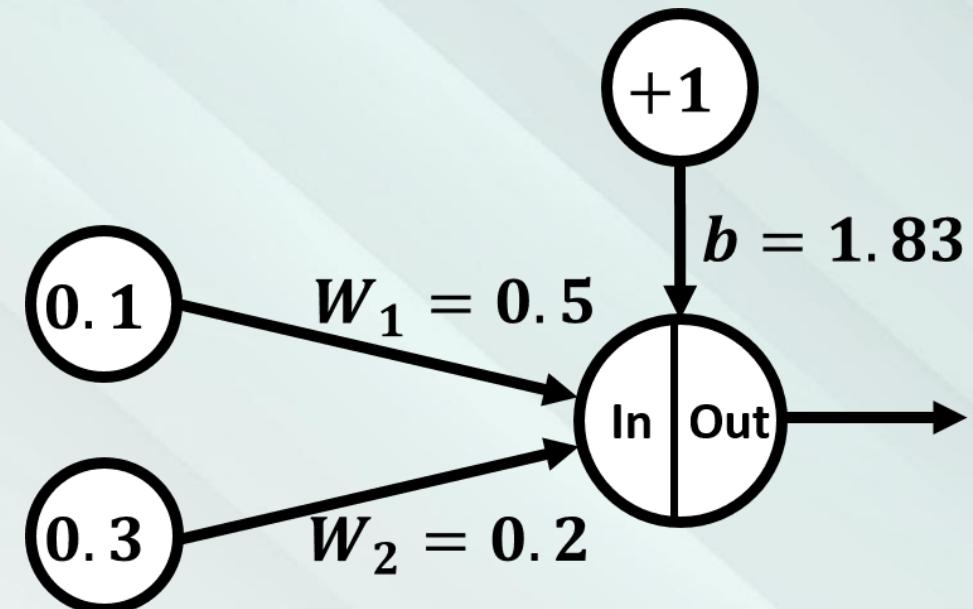


# Cómo funciona el algoritmo de retropropagación

- Supongamos que los valores iniciales tanto para los pesos como para el sesgo son como en la siguiente tabla.

$W_1$	$W_2$	b
0.5	0.2	1.83

- Para simplificar, los valores de todas las entradas, pesos y sesgos se agregarán al diagrama de red.
- Ahora, entrenemos la red y veamos cómo la red predecirá la salida de la muestra en función de los parámetros actuales.
- Como comentamos anteriormente el proceso de entrenamiento tiene 2 fases, avance y retroceso.





# Pase hacia adelante

- La entrada de la función de activación será el SOP entre cada entrada y su peso. El SOP se suma al sesgo para obtener la salida de la neurona:

$$s = X_1 * W_1 + X_2 * W_2 + b$$

$$s = 0,1 * 0,5 + 0,3 * 0,2 + 1,83$$

$$s = 1,94$$

- Luego se aplica el valor 1,94 a la función de activación (sigmoidea), lo que da como resultado el valor 0,874352143.

$$f(s) = \frac{1}{1 + e^{-s}}$$

$$f(s) = \frac{1}{1 + e^{-1.94}}$$

$$f(s) = \frac{1}{1 + 0.143703949} \quad f(s) = 0.874352143$$





# Pase hacia adelante

- La salida de la función de activación de la neurona de salida refleja la salida predicha de la muestra. Es obvio que existe una diferencia entre la salida deseada y la esperada. Pero ¿por qué? ¿Cómo podemos acercar la salida predicha a la deseada? Responderemos estas preguntas más adelante. Por ahora, veamos el error de nuestra red basado en una función de error.
- Las funciones de error indican la proximidad entre los resultados previstos y los deseados. El valor óptimo de error es **cero**, lo que significa que no hay error alguno y que tanto los resultados deseados como los previstos son idénticos. Una de estas funciones es la **función de error al cuadrado**, definida en la siguiente ecuación:
- Tenga en cuenta que el valor 12 multiplicado por la ecuación es para simplificar el cálculo de las derivadas utilizando el algoritmo de retropropagación.



# Pase hacia adelante

- Basándonos en la función de error, podemos medir el error de nuestra red de la siguiente manera:

$$E = \frac{1}{2}(desired - predicted)^2$$

$$E = \frac{1}{2}(0.03 - 0.874352143)^2$$

$$E = \frac{1}{2}(-0.844352143)^2$$

$$E = \frac{1}{2}(0.712930542)$$

$$E = 0.356465271$$

- El resultado muestra que hay un error, y uno grande: (~0.357). Este error simplemente nos da una indicación de la distancia entre los resultados previstos y los deseados.
- Sabiendo que hay un error, ¿qué debemos hacer? Debemos minimizarlo. Para minimizar el error de red, debemos cambiar algo en ella. Recuerde que los únicos parámetros que podemos cambiar son los pesos y los sesgos. Podemos probar diferentes pesos y sesgos y luego probar nuestra red.
- Calculamos el error, luego finaliza el pase hacia adelante y debemos comenzar el **pase hacia atrás** para calcular las derivadas y actualizar los parámetros.
- Para sentir prácticamente la importancia del algoritmo de retropropagación, intentemos actualizar los parámetros directamente sin usar este algoritmo.





# Ecuación de actualización de parámetros

- Los parámetros se pueden cambiar según la siguiente ecuación:
- $W_{(n+1)} = W(n) + \eta[d(n) - Y(n)]X(n)$
- Dónde:
- n: Paso de entrenamiento (0, 1, 2, ...).
- W(n): Parámetros del paso de entrenamiento actual.  $W_n = [b_n, W_1(n), W_2(n), W_3(n), \dots, W_m(n)]$
- $\eta$  : Tasa de aprendizaje con un valor entre 0,0 y 1,0.
- d(n): Salida deseada.
- Y(n): Salida prevista.
- X(n): Entrada actual en la que la red realizó una predicción falsa.
- Para nuestra red, estos parámetros tienen los siguientes valores:
  - n: 0
  - W(n): [1,83, 0,5, 0,2]
  - $\eta$  : Debido a que es un hiperparámetro, podemos elegirlo 0,01, por ejemplo.
  - d(n): [0,03].
  - Y(n): [0,874352143].
  - X(n): [+1, 0,1, 0,3]. El primer valor (+1) es para el sesgo.





# Ecuación de actualización de parámetros

- Podemos actualizar nuestros parámetros de red de la siguiente manera:
  - $W_{(n+1)} = W(n) + \eta [d(n) - Y(n)] X(n)$
  - $=[1,83, 0,5, 0,2] + 0,01[0,03 - 0,874352143][+1, 0,1, 0,3]$
  - $=[1,83, 0,5, 0,2] + 0,01[-0,844352143][+1, 0,1, 0,3]$
  - $=[1,83, 0,5, 0,2] + -0,00844352143[+1, 0,1, 0,3]$
  - $=[1,83, 0,5, 0,2] + [-0,008443521, -0,000844352, -0,002533056]$
  - $=[1.821556479, 0.499155648, 0.197466943]$
- Los nuevos parámetros se enumeran en la siguiente tabla:

W1nuevo	W2new	nuevo
0,197466943	0,499155648	1.821556479





# Ecuación de actualización de parámetros

- Con base en los nuevos parámetros, recalcularemos la salida predicha. Esta nueva salida predicha se utiliza para calcular el nuevo error de red. Los parámetros de red se actualizan según el error calculado. El proceso continúa actualizando los parámetros y recalculando la salida predicha hasta alcanzar un valor aceptable para el error.
- Aquí, actualizamos correctamente los parámetros sin usar el algoritmo de retropropagación. ¿Aún necesitamos ese algoritmo? Sí. Ya verás por qué.
- La ecuación de actualización de parámetros depende únicamente de la tasa de aprendizaje para actualizar los parámetros. Cambia todos los parámetros en dirección opuesta al error.
- Pero, usando el algoritmo de retropropagación, podemos saber cómo se correlaciona cada peso con el error. Esto nos indica el efecto de cada peso en el error de predicción. Es decir, ¿qué parámetros aumentamos y cuáles disminuimos para obtener el menor error de predicción?
- Por ejemplo, el algoritmo de retropropagación podría brindarnos información útil, como que aumentar el valor actual de  $W_1$  en 1,0 aumenta el error de red en 0,07. Esto nos indica que un valor menor de  $W_1$  es mejor para minimizar el error.





# Derivada parcial

- Una operación importante en el paso hacia atrás es el cálculo de derivadas. Antes de analizar el cálculo de derivadas en el paso hacia atrás, podemos empezar con un ejemplo sencillo para simplificar las cosas.
- Para una función multivariante como  $Y=X_2Z+H$ , ¿cuál es el efecto en la salida Y dado un cambio en la variable X? Podemos responder esto mediante **derivadas parciales**, como se indica a continuación:

$$\frac{\partial Y}{\partial X} = \frac{\partial}{\partial X}(X^2Z + H) \quad \frac{\partial Y}{\partial X} = 2XZ + 0 \quad \frac{\partial Y}{\partial X} = 2XZ$$

- Tenga en cuenta que todo excepto X se considera constante. Por lo tanto, H se reemplaza por 0 después de calcular una derivada parcial. Aquí,  $\partial X$  significa un cambio mínimo en la variable X. De igual manera,  $\partial Y$  significa un cambio mínimo en Y. El cambio en Y es el resultado de cambiar X. Al hacer un cambio muy pequeño en X, ¿cuál es el efecto en Y?
- El pequeño cambio puede ser un aumento o una disminución de un valor minúsculo. Sustituyendo los diferentes valores de X, podemos determinar cómo cambia Y con respecto a X.
- Se puede seguir el mismo procedimiento para comprender cómo cambia el error de predicción de la NN con respecto a los cambios en los pesos de la red. Por lo tanto, nuestro objetivo es calcular  $\partial E/W_1$  y  $\partial E/W_2$ , ya que solo tenemos dos pesos  $W_1$  y  $W_2$ .





UNIVERSIDAD  
CATÓLICA  
SEDES SAPIENTIAE



UNIVERSIDAD  
CATÓLICA  
SEDES SAPIENTIAE

# Inteligencia Artificial

Ing. Juancarlos Santana Huamán  
[jsantana@ucss.edu.pe](mailto:jsantana@ucss.edu.pe)



# Breve historia de los datos y las dificultades de administración asociadas

- Alrededor de 2,5 trillones (2 500 000 000 000 000) de bytes de datos se crean a diario, o aproximadamente 1,7 megabytes por segundo y persona en el planeta.
- Los datos no se pierden por arte de magia. Son los usuarios quienes los pierden. Sin embargo, en los entornos de trabajo actuales, distribuidos y centrados en la nube (cloud-first), la mayoría de los departamentos de TI tienen poca visibilidad de las pérdidas de datos provocadas por las personas y ninguna capacidad para gestionarlas. Este libro electrónico analiza cinco **fugas de datos** reales para descubrir cómo ocurrieron, las consecuencias para la empresa, y cómo podrían haberse evitado. Descubrirá:
  - Cómo hacer frente a los riesgos de apps de terceros con acceso OAuth
  - Qué hacer con los usuarios comprometidos
  - Cómo detectar y corregir los errores de configuración de los servidores
  - Las ventajas de un enfoque centrado en las personas de la prevención de la pérdida de datos



# ¿Qué es la clasificación de datos?

- La clasificación de datos es un método para definir y categorizar los archivos y otra información empresarial clave. Se usa principalmente en grandes organizaciones para crear sistemas de seguridad que siguen estrictas normativas de cumplimiento, pero también se puede usar en entornos reducidos. El uso más importante de la clasificación de datos es comprender la sensibilidad de la información almacenada para crear las herramientas adecuadas de ciberseguridad, controles de acceso y monitorización.
- La clasificación es el proceso de categorizar los activos de datos basándose en la sensibilidad de la información. Mediante la clasificación de datos, las organizaciones pueden determinar dos elementos clave:
  - Quién debe tener autorización para acceder a esta.
  - Qué políticas de protección aplicar al almacenarla y transferirla.
- La clasificación también ayuda a determinar los estándares normativos aplicables para proteger los datos. En general, la clasificación ayuda a las organizaciones a gestionar mejor sus datos para propósitos de privacidad, cumplimiento y ciberseguridad.



# Motivos para realizar clasificación de datos

- Todas las organizaciones deben clasificar los datos que crean, gestionan y almacenan. Pero es aún más importante para entornos de grandes empresas. Esto es porque las grandes empresas tienen activos de datos distribuidos entre múltiples ubicaciones, incluyendo en la nube.
- Los administradores deben hacer seguimiento y auditar esta información para garantizar que tenga los controles de acceso y autenticación adecuados. La clasificación de datos permite a los administradores identificar las ubicaciones en donde se almacenan datos delicados y determinar cómo se debe acceder y compartir estos datos.
- La clasificación es un primer paso fundamental para cumplir con casi cualquier normativa de conformidad de datos. HIPAA, RGPD, FERPA y otros organismos reguladores gubernamentales exigen que los datos se etiqueten para que los controles de seguridad y autenticación puedan limitar el acceso. El etiquetado de datos ayuda a organizar y proteger la información. El ejercicio también reduce la duplicación innecesaria de datos, reduce los costes de almacenamiento, incrementa el rendimiento y hace posible registrarlos cuando se comparte.
- La clasificación de datos es la base de unas políticas eficaces para protección de datos y reglas para la prevención de pérdida de datos (DLP). Para tener unas reglas de DLP eficaces, primero se deben clasificar sus datos para asegurarse de conocer bien los datos almacenados en cada archivo.



# Tipos de clasificación de datos

- Cualquier dato almacenado se puede clasificar en categorías. Para clasificar sus datos, es necesario formular diversas preguntas a medida que se va descubriendo y revisando. Use las siguientes preguntas de ejemplo al revisar cada sección de sus datos:
  - ¿Qué información almacena para sus clientes, empleados y proveedores?
  - ¿Qué tipos de datos crea la organización al generar un nuevo registro?
  - ¿Qué tan sensibles son los datos en una escala numérica (¿p. ej. 1-10, donde 1 indica la mayor sensibilidad?)
  - ¿Quién debe acceder a estos datos para seguir operando con productividad?
- Usando estas preguntas, es posible definir categorías generales para sus datos, por ejemplo:
- **Alta sensibilidad:** Estos datos deben ser protegidos y monitorizados para protegerlos contra agentes de amenaza. Con frecuencia quedan sujetos a las normativas de cumplimiento como información que requiere de estrictos controles de acceso y que también minimice la cantidad de usuarios que pueden acceder a los datos.
- **Sensibilidad intermedia:** Los archivos y datos que no se pueden revelar al público, pero que si sufriesen de una filtración no implicarían un nivel de riesgo significativo, se podrían considerar de riesgo intermedio. Requieren de controles de acceso, al igual que los datos altamente sensibles, pero una gama más amplia de usuarios puede acceder a estos.
- **Baja sensibilidad:** Estos datos típicamente constan de información pública que no requiere de mucha seguridad para protegerla de una filtración de datos.



# Técnicas de clasificación de datos

- La clasificación de datos colabora estrechamente con otras tecnologías para proteger y gobernar mejor los datos. Si la organización sufre una vulneración de datos, la clasificación de datos ayuda a los administradores a identificar datos perdidos y potencialmente ayuda a rastrear al cibercriminal.
- Aquí mencionamos algunas técnicas de clasificación de datos:
  - **Gestión de acceso a la identidad (IAM):** Las herramientas de IAM permiten a los administradores determinar quién y qué puede acceder a los datos. Los usuarios con permisos similares se pueden agrupar. Los grupos reciben niveles de autorización y se gestionan como una unidad individual. Cuando un usuario se va, al usuario se le puede eliminar del grupo, lo que elimina todos los permisos de dicho usuario. Este tipo de reagrupación y organización optimiza la gestión de permisos en toda la red.
  - **Cifrado de datos:** Ciertos activos de datos deben estar cifrados tanto en reposo como en movimiento. Los datos “en reposo” se almacenan en cualquier dispositivo de almacenamiento, típicamente en un disco duro. Los datos “en movimiento” son aquellos que se transfieren a través de una red. El **cifrado de datos** los vuelve ilegibles para cualquier atacante que logre interceptarlos.
  - **Automatización:** La automatización funciona mano a mano con herramientas de monitorización para hallar, clasificar y etiquetar datos para su revisión administrativa. Algunas herramientas integran la inteligencia artificial (IA) y el aprendizaje automático (ML) para detectar, etiquetar y clasificar datos automáticamente. Las tecnologías también pueden ayudar a identificar amenazas que podrían usarse para robarla. Con los datos etiquetados, los administradores pueden usar la IAM para aplicar permisos y evitar que ciertas amenazas específicas obtengan acceso a datos almacenados.
  - **Análisis forense de datos:** El análisis forense es el proceso de identificar qué fue lo que salió mal y quién vulneró a la red. Después de una filtración de datos, el análisis forense de datos se encarga de recopilar y preservar evidencia para continuar con las investigaciones. El análisis forense de datos generalmente es un proceso de dos partes. Primero, las herramientas de automatización recopilan los datos; después, un analista humano identifica las anomalías y las investiga.



# Niveles de clasificación de datos

- Mediante estos niveles, podrá clasificar mejor sus datos. La clasificación de datos se suele dividir en cuatro categorías:
  - **Datos públicos.** Los datos están disponibles al público, ya sea localmente o por internet. Los datos públicos requieren un nivel bajo de seguridad, porque su revelación no vulneraría los requisitos de cumplimiento.
  - **Datos de uso interno únicamente.** Memorandos, propiedad intelectual y mensajes de correo electrónico son algunos ejemplos de datos que deben estar restringidos a empleados internos.
  - **Datos confidenciales.** La diferencia entre los datos de uso interno únicamente y los confidenciales es que los confidenciales requieren de autorización para su acceso. Usted puede asignarles autorización a empleados específicos o a proveedores externos autorizados.
  - **Datos restringidos.** Los datos restringidos suelen estar relacionados con información gubernamental a la que solo pueden acceder individuos autorizados. La revelación de datos restringidos puede causar daños irreparables a la reputación e ingresos corporativos.



# Alineación con una lista de activos

- Antes de comenzar con una revisión de clasificación de datos, deben estar en la misma página. Al comienzo de la revisión, crean una lista de activos para definir sus categorías de negocio. Por ejemplo, a lo mejor tiene archivos en los que almacena datos de tecnología, financieros y de los clientes. La definición de categorías alinea sus requerimientos de seguridad con sus datos.
- Este paso también implica el aplicar los niveles de clasificación de datos definidos en la sección anterior. Para cada categoría, es probable que tenga diferentes niveles de datos para cada grupo de archivos. Este primer paso crea una base para todo el proceso de clasificación de datos.



# Proceso de clasificación de datos

- Cuando decida que es momento de clasificar datos para cumplir con los estándares de conformidad, el primer paso es implementar procedimientos para asistir con la ubicación y clasificación de datos, así como para determinar las medidas de ciberseguridad adecuadas. La ejecución de cada procedimiento depende de los estándares de conformidad de su organización y de la infraestructura que mejor proteja a los datos. Los pasos generales para clasificar datos son:
  - **Ejecutar una evaluación de riesgos:** Una evaluación de riesgos determina la sensibilidad de los datos e identifica cómo un atacante podría vulnerar las defensas de la red.
  - **Desarrollar políticas y estándares de clasificación:** Si se generan datos adicionales en el futuro, una política de clasificación permite optimizar y volver repetible al proceso, facilitando así el trabajo al personal y a la vez minimizando los errores en el proceso.



# Proceso de clasificación de datos

- **Categorización de datos:** Ya con políticas establecidas y con una evaluación de riesgo realizada, categorice sus datos basándose en su sensibilidad, quién debe tener acceso y si las penalizaciones de conformidad deben ser hechas públicas.
- **Hallar la ubicación de almacenamiento de sus datos:** Antes de implementar las defensas de ciberseguridad adecuadas, usted debe saber en dónde se almacenan los datos. Identificar las ubicaciones de almacenamiento de datos indica el tipo de ciberseguridad necesario para proteger los datos.
- **Identificar y clasificar sus datos:** Con los datos identificados, ya puede clasificarlos. El software externo le ayuda con este paso para facilitarle la clasificación y rastreo de los datos.
- **Implementar controles:** Los controles empleados deberían solicitar autenticación y autorización a cada usuario y recurso que necesite acceso a los datos. Ese acceso debe otorgarse “solo si es necesario”, lo que significa que los usuarios solamente reciben acceso si necesitan ver los datos para ejecutar una función laboral.
- **Monitorización de acceso y datos:** La monitorización de datos es un requisito de cumplimiento y para la privacidad de sus datos. Sin la monitorización, un atacante podría disponer de meses enteros para **exfiltrar datos** de la red. Los controles de monitorización adecuados detectan anomalías y reducen el tiempo necesario para detectar, mitigar y erradicar una amenaza de la red.



# Optimizar el proceso de clasificación de datos

- Si bien es posible optimizar el proceso de clasificación de datos y hasta automatizarlo parcialmente, el proceso requiere de elementos de revisión humana y procedimientos manuales.
- Los sistemas automatizados sugieren etiquetas y clasificaciones, pero es mediante una revisión humana que se determina si estas etiquetas son correctas. Los objetivos y estándares deben ser esbozados y definidos, cosa que precisa de revisores humanos y personal de TI.
- Las herramientas automatizadas señalan activos para que los revisen seres humanos. La lista muestra los objetos (tales como los datos de un cierto cliente) y las reglas (como HIPAA o PCI-DSS) que se aplican a cada uno. Algunas herramientas de automatización pueden indexar objetos. (La indexación es el proceso de clasificar y organizar datos para permitir búsquedas rápidas y eficientes en la red).
- Otras políticas también se aplican durante el proceso de clasificación de datos. El **reglamento general de protección de datos** (RGPD) es una normativa de la UE que les da a los consumidores el derecho a que sus datos sean eliminados. Las organizaciones deben cumplir con esto al almacenar datos de los consumidores en la UE. Algunas herramientas de clasificación de datos indexan objetos de manera tal que puedan eliminarse rápidamente cuando los clientes así lo soliciten.



# Ejemplos de clasificación de datos

- Uno de los pasos más complejos de la clasificación de datos es comprender los riesgos. Si bien los estándares de conformidad supervisan la mayor parte de los datos privados delicados, las organizaciones deben adherirse a las normativas de conformidad aplicables a diferentes datos almacenados en archivos y bases de datos. La clasificación de datos ayuda a proteger datos y a garantizar la conformidad. Es fundamental para cumplir con los requisitos del RGPD. (De hecho, las organizaciones deben indexar los datos de los consumidores en la UE, para que estos puedan borrarse si así se solicita).
- El RGPD también exige la protección de información personal secundaria de los clientes, como el origen étnico, las opiniones políticas, su raza y sus creencias religiosas. Para ello, las organizaciones deben clasificar estos datos y determinar los permisos adecuados entre los diversos activos digitales. La clasificación es la que determina quiénes pueden acceder a estos datos para que no se les dé un uso indebido. Solo entonces pueden evitar el revelar información privada de los consumidores, así como costosas filtraciones de datos.



# Ejemplos de clasificación de datos

- Tres pasos para clasificar para el RGPD serían:
  - **Localizar y auditar los datos.** Antes de la clasificación, los administradores deben identificar dónde se almacenan los datos y qué reglas los afectan.
  - **Crear una política de clasificación.** Para garantizar el cumplimiento, es necesario crear estándares y procedimientos de clasificación de datos para definir cómo su organización almacena y transfiere datos delicados.
  - **Organizar y asignar prioridades a los datos.** Con la priorización, su organización puede determinar la clasificación de datos y los permisos necesarios para acceder a esta.
- Aquí indicamos algunos ejemplos de sensibilidad de datos que podrían categorizarse como alta, media y baja.
  - **Alta sensibilidad:** Imagínese que su empresa registra números de tarjetas de crédito como método de pago de los clientes que le compran. Estos datos deben tener controles de autorización estrictos, auditorías para detectar solicitudes de acceso y cifrado aplicado a los datos almacenados y transmitidos. Una filtración de datos probablemente causaría daños tanto al cliente como a la organización, así que deben clasificarse como altamente sensible con estrictos controles de ciberseguridad.
  - **Sensibilidad intermedia:** Para cada proveedor externo, usted tiene un contrato con firmas que ejecutan un acuerdo. Estos datos no causarían daño a los clientes, pero aun así son información delicada que describe detalles del negocio. Estos archivos se considerarían de sensibilidad intermedia.
  - **Sensibilidad baja:** Los datos para consumo del público pueden considerarse de baja sensibilidad. Por ejemplo, el material de marketing publicado en su web no necesitaría de controles estrictos, dado que está disponible al público y ha sido creado para una audiencia general.



# Uso de la inteligencia artificial (IA) para clasificación de datos

- La clasificación de datos precisa de interacción humana, pero buena parte del proceso se puede automatizar. Para añadir la automatización a las capacidades de toma de decisiones. La automatización por IA garantiza que las organizaciones puedan identificar, clasificar y proteger sus documentos continuamente, lo que implica que el motor continuamente escanea y revisa nuevos documentos a medida que se agregan al entorno.
- El módulo de Aprendizaje Activo ingiere alrededor de 20 documentos por categoría para comenzar el proceso y mejorar su precisión. El motor de clasificación de datos emplea modelos de aprendizaje automático para identificar patrones. Cada grupo de archivos debe ser lo suficientemente diverso como para que los algoritmos de aprendizaje automático puedan volverse más precisos.
- Los modelos de aprendizaje automático predicen las etiquetas para los documentos y determinan la exactitud de las predicciones. Al revisor se le muestra un “nivel de confianza” para reevaluar los datos del modelo para otra ronda de clasificación de información. Si el modelo indica que la exactitud es baja, los examinadores humanos pueden actualizar los modelos para tener conjuntos de archivos más diversos que permitan incrementar la exactitud. El motor se “entrena” a sí mismo aprovechando la nueva información para producir resultados mejores y óptimos, así que asigna permisos de acceso a usuarios solamente en los archivos necesarios para realizar sus funciones laborales.
- El software de clasificación de datos basada en IA reduce buena parte de los gastos generales de un proceso que podría llevar meses. Escanea todos sus archivos de forma automática, identifica su contenido, asigna las categorías y niveles de datos correctos y después le permite determinar la seguridad de salvaguarda adecuada.



# Importancia de la clasificación de datos

- El “nivel de sensibilidad” de los datos determina cómo los procesa y los protege. Incluso si usted sabe que los datos son importantes, es necesario evaluar sus riesgos. El proceso de clasificación de datos le ayuda a descubrir potenciales amenazas e implementar las soluciones de ciberseguridad más beneficiosas para su empresa.
- Al asignar niveles de sensibilidad y categorizar los datos, usted comprende las reglas de acceso para los datos clave. Usted puede monitorizar mejor los datos para hallar potenciales filtraciones de datos y, más importante aún, mantener un alto nivel de conformidad. Las normativas de conformidad le ayudan a determinar cuáles son los controles de ciberseguridad adecuados, pero antes debe ejecutar una evaluación de riesgos y clasificar los datos. Las organizaciones suelen requerir de ayuda externa para la clasificación de datos, de modo que la implementación de la ciberseguridad se pueda ejecutar más eficientemente.
- La exactitud de la clasificación de datos es fundamental para futuras estrategias de DLP; por tanto, muchas organizaciones, grandes y pequeñas, han optado por usar automatización basada en IA. La inteligencia artificial aprovecha los modelos de aprendizaje automático para determinar el nivel de clasificación y categoría adecuados.



# Buenas prácticas de clasificación de datos

- Observar buenas prácticas de clasificación de datos hace mucho más eficiente tanto a la creación de políticas como al proceso en general. Las buenas prácticas definen los pasos a seguir para indexar y etiquetar correctamente los activos digitales para que ninguno se pase por alto o se gestione mal.
- Las organizaciones deben seguir estas buenas prácticas:
  - Identificar cuidadosamente en dónde se ubican todos los datos delicados, incluyendo la propiedad intelectual, en todas las ubicaciones de almacenamiento.
  - Definir categorías de datos, para que los datos delicados se puedan etiquetar y configurar con los permisos adecuados. Las categorías deben ser granulares, para que los permisos también lo sean. Las categorías también deben permitir a los administradores categorizar los datos en grupos.
  - Identificar los datos más sensibles y claves. Después, las herramientas de automatización pueden etiquetarlos con la clasificación y mandatos normativos correctos.
  - Capacitar a los empleados para que comprendan cómo manejar información delicada. Darles las herramientas necesarias para proteger datos delicados y seguir buenas prácticas de ciberseguridad.
  - Examinar todos los estándares normativos de modo que se cumpla con las reglas y se eviten sanciones.
  - Crear políticas que permitan a los usuarios identificar datos mal clasificados o sin clasificar, para arreglar el problema.
  - Usar IA en donde se pueda mejorar la exactitud y acelerar el proceso de clasificación de datos.





# Clasificación de sensibilidad de datos

- La clasificación de datos requiere que evalúe el nivel de sensibilidad de los datos en toda su organización. Estos niveles generalmente varían de alto a medio a bajo y se correlacionan con lo dañino que sería si esos datos se perdieran, robaran o comprometerán.
- Clasificar los datos de esta manera ayuda a las organizaciones a entender dónde enfocar sus esfuerzos de mitigación de riesgos. Cuanto más sensibles sean los datos, más necesita su organización centrarse en protegerlos.

¿Qué es  
un dato  
“sensible”?

«Dato personal que requiere **especial protección** ya que su tratamiento puede entrañar **importantes riesgos** para los derechos y las libertades fundamentales.»





# Datos de baja sensibilidad

- Los datos de baja sensibilidad son datos que tendrían poco o ningún impacto si se vieran comprometidos, perdidos o destruidos (aunque una organización aún puede implementar controles de seguridad para protegerse contra daños). Los datos de baja sensibilidad son para uso público y no requieren ninguna protección de confidencialidad. Generalmente se etiquetan como datos no restringidos o públicos, dependiendo de su modelo de clasificación.
- Ejemplos de datos de baja sensibilidad incluyen:
  - Información pública y páginas web, como ofertas de trabajo, publicaciones de blog, etc.
  - Comunicados de prensa
  - Directorio de empleados



# Datos de sensibilidad media

- Los datos de sensibilidad media son datos que no tendrían un impacto catastrófico si fueran comprometidos, perdidos o destruidos, pero resultarían en algún riesgo para una organización. Por lo tanto, estos datos solo deben ser accesibles para el personal interno al que se le haya concedido acceso y comúnmente se etiquetan como internos o privados.
- Ejemplos de datos de sensibilidad media incluyen:
  - Correos electrónicos o documentos internos que no contienen datos confidenciales
  - Contratos con proveedores
  - Información de gestión de servicios de TI o telecomunicaciones



# Datos de alta sensibilidad

- Los datos de alta sensibilidad son datos que, si se comprometen, pierden o destruyen, tendrían un impacto catastrófico en una organización. Por lo tanto, las organizaciones deben implementar los controles de acceso más estrictos para los datos de alta sensibilidad. Dado que el acceso está limitado según la necesidad de saber, los datos de alta sensibilidad usualmente se etiquetan como datos confidenciales o restringidos.
- Ejemplos de datos de alta sensibilidad incluyen:
  - Registros financieros, como números de tarjetas de crédito
  - Datos médicos y biométricos, incluida la información de salud protegida (PHI)
  - Registros de empleados, incluida la información de identificación personal (PII) como números de Seguridad Social
  - Datos de autenticación, como credenciales de inicio de sesión



# Modelos y esquemas de clasificación de datos

- Un modelo y esquema de clasificación de datos define cómo una organización identifica y categoriza sus activos de datos. Típicamente, éstos definen de tres a cinco niveles basados en la criticidad y sensibilidad de los datos para ayudar a determinar los controles de seguridad apropiados.
- Las organizaciones deben diseñar sus propios modelos y esquemas de clasificación de datos según su necesidad de proteger datos propietarios, empresariales y/o de usuarios con diferentes niveles de sensibilidad y para cumplir con los requisitos de cumplimiento y regulación. Sin embargo, pueden comenzar con o basar sus modelos en diferentes modelos y esquemas de clasificación desarrollados por gobiernos y organizaciones comerciales.
- Por ejemplo, el gobierno de los EE.UU. utiliza un esquema de clasificación de tres niveles para datos basado en el impacto potencial en la seguridad nacional si se divulga:
  - **Confidencial:** La divulgación no autorizada de esta información probablemente causaría daño a la seguridad nacional.
  - **Secreto:** La divulgación no autorizada de esta información probablemente causaría un daño grave a la seguridad nacional.
  - **Altamente secreto:** La divulgación no autorizada de esta información probablemente causaría un daño excepcionalmente grave a la seguridad nacional.



# Modelos y esquemas de clasificación de datos

- NIST (Instituto Nacional de Estándares y Tecnología) desarrolló un esquema de categorización de tres niveles basado en el impacto potencial no solo en la confidencialidad, sino también en la integridad y disponibilidad de la información y los sistemas de información aplicables a la misión de una organización:
  - **Bajo:** La divulgación no autorizada de esta información tendría un efecto adverso limitado en las operaciones de la organización, los activos de la organización o los individuos.
  - **Moderar:** La divulgación no autorizada de esta información tendría un efecto adverso serio en las operaciones de la organización, los activos de la organización o los individuos.
  - **Alto:** La divulgación no autorizada de esta información tendría un efecto adverso severo o catastrófico en las operaciones de la organización, los activos de la organización o los individuos.
- Las organizaciones pueden usar etiquetas secundarias dentro de estos niveles para especificar diferentes activos de datos y procedimientos de manejo o requisitos de cumplimiento y regulación. Por ejemplo, una organización que solo recopila registros financieros puede clasificarlos como “datos confidenciales”, pero una organización que recopila registros médicos puede clasificarlos de manera más específica como “información de salud protegida” para indicar que los requisitos de HIPAA se aplican a esos datos.



# Ejemplos de clasificación de datos

- Aunque el esquema de clasificación de datos del NIST es ampliamente reconocido como un esquema de clasificación adecuado en certificaciones sectoriales, nacionales e internacionales, las organizaciones deben desarrollar sus propios esquemas de clasificación según sus necesidades únicas de gestión organizacional y de riesgos.
- Para inspirarnos, veremos algunos ejemplos de organizaciones y el modelo de clasificación y esquema que han implementado.

## **UW-Madison**

- UW-Madison clasifica los datos en cuatro categorías, que se utilizan para determinar cómo proporcionar acceso a los datos a las personas. Las categorías son:
  - **Público:** La divulgación, alteración o destrucción no autorizada de estos datos resultaría en poco o ningún riesgo para la Universidad y sus afiliados. Cualquier dato mostrado en sitios web o publicado sin restricciones de acceso debe clasificarse como público.
  - **Interno:** La divulgación, alteración o destrucción no autorizada de estos datos podría resultar en algún riesgo para la Universidad y sus afiliados. Por defecto, cualquier dato que no esté clasificado explícitamente en las otras tres categorías debe ser clasificado como interno.
  - **Sensible:** La divulgación, alteración, pérdida o destrucción no autorizada de estos datos podría causar un nivel moderado de riesgo para la Universidad, sus afiliados o proyectos de investigación.
  - **Restringido:** La divulgación, alteración, pérdida o destrucción no autorizada de esos datos podría causar un nivel significativo de riesgo para la Universidad, sus afiliados o proyectos de investigación. Si la protección de los datos es requerida por ley o regulación o si UW-Madison está obligada a informar al gobierno y/o notificar al individuo en caso de acceso inapropiado a los datos, entonces deben clasificarse como restringidos.



# Ejemplos de clasificación de datos

## Harvard

- Harvard clasifica los datos en cinco niveles:

- **N1:** N1 se refiere a información pública. La Universidad proporciona intencionadamente esta información al público. Investigaciones publicadas, catálogos de cursos, hallazgos regulatorios y legales, informes anuales publicados, patentes liberadas y políticas universitarias son todos ejemplos.
- **N2:** N2 se refiere a información confidencial de bajo riesgo. La Universidad elige mantener esta información privada dentro de la comunidad de Harvard, pero su divulgación fuera de la comunidad no causaría daño material. Políticas y procedimientos del departamento, materiales de formación de Harvard, borradores de trabajos de investigación y solicitudes de patentes y subvenciones son todos ejemplos.
- **N3:** N3 se refiere a información confidencial de riesgo medio. La Universidad intenta compartir esta información solo con aquellos que tienen una “necesidad comercial de saber” y la divulgación más allá de los destinatarios previstos podría causar un daño material a las personas o a la Universidad. Información no directorio de estudiantes, información no publicada de profesores y personal, información de transacciones presupuestarias/financieras y la información especificada como confidencial por contratos de proveedores y acuerdos de confidencialidad son todos ejemplos.
- **N4:** N4 se refiere a información confidencial de alto riesgo. La Universidad tiene estrictos controles para esta información y la divulgación más allá de los destinatarios especificados probablemente causaría un daño grave a las personas o a la Universidad. Contraseñas y PINs, credenciales del sistema y claves de cifrado privadas son todos ejemplos.
- **N5:** N5 está reservado solo para datos de investigación, según lo determinado por el IRB o el Acuerdo de Uso de Datos. Los datos que, si se divulgaran, podrían poner al sujeto en un grave riesgo de daño o los datos con requisitos contractuales para medidas de seguridad excepcionales deben clasificarse como N5.





# Ejemplos de clasificación de datos

## AWS

- AWS recomienda comenzar con un enfoque de clasificación de datos de tres niveles. Tanto las organizaciones públicas como comerciales que han adoptado la nube de AWS han podido satisfacer adecuadamente sus necesidades y requisitos de clasificación de datos utilizando el enfoque a continuación.

Data classification tier	System security categorization	Cloud deployment model options
Unclassified	Low to High	Accredited public cloud
Official	Moderate to High	Accredited public cloud
Secret and above	Moderate to High	Accredited private/hybrid/community cloud/public cloud



# Procesamiento Básico de Datos con Python

- Los requisitos de programación en la ciencia de datos exigen un lenguaje muy versátil pero flexible que sea simple para escribir el código pero que pueda manejar un procesamiento matemático altamente complejo. Python es más adecuado para estos requisitos, ya que ya se ha establecido como un lenguaje para la informática general y científica.
- Además, se está actualizando continuamente en forma de una nueva adición a su gran cantidad de bibliotecas destinadas a diferentes requisitos de programación.
- Los siguientes, son fragmentos de Python que pueden ser útiles para principiantes para algunas tareas de procesamiento de datos diferentes.
- Las tareas del procesamiento de datos van desde procesamiento de texto y listas básico a procesamiento de conjunto de datos con Pandas.





# Concatenar varios archivos de texto

- Comencemos con la concatenación de varios archivos de texto. Si tienes varios archivos de texto en un solo directorio que necesitas concatenar en un solo archivo, este código Python lo hará.
- Primero obtenemos una lista de todos los archivos txt en la ruta; luego leemos en cada archivo y escribimos su contenido en el nuevo archivo de salida; finalmente, volvemos a leer el nuevo archivo e imprimimos su contenido en la pantalla para verificarlo.

```
import glob
# Cargar todos los archivos txt en la ruta
files = glob.glob ('/ path / to / files / *.
txt')
# Concatenar archivos a un nuevo archivo
with open('2020_output.txt', 'w') as out_file:
    for file_name in files:
        with open(file_name) as in_file:
            out_file.write(in_file.read())
# Leer archivo e imprimir
with open('2020_output.txt', 'r') as new_file:
    lines = [line.strip() for line in new_file]
for line in lines:
    print(line)
```

# Concatenar varios archivos CSV en un marco de datos

- Siguiendo con el tema de la concatenación de archivos, esta vez abordemos la concatenación de varios archivos de valores separados por comas en un solo marco de datos de Pandas.
- Primero obtenemos una lista de los archivos CSV en nuestra ruta; luego, para cada archivo en la ruta, leemos el contenido en su propio marco de datos; luego, combinamos todos los marcos de datos en un solo marco; finalmente, imprimimos los resultados para inspeccionar.

```
import glob
# Cargar todos los archivos csv en la ruta
files = glob.glob ('/ path/to/files/ *. csv')
# Crear una lista de dataframe, una serie por
CSV
fruit_list = []
for file_name in files:
    df = pd.read_csv (file_name, index_col =
None, header = None)
    fruit_list.append (df)
# Crear un marco combinado a partir de la lista
de marcos individuales
fruit_frame = pd.concat (fruit_list, axis = 0,
ignore_index = True)
print (fruit_frame)
```



# Aplanar listas

- Quizás tengas una situación en la que estés trabajando con una lista de listas, es decir, una lista en la que todos sus elementos también sean listas. Este fragmento tomará esta lista de listas incrustadas y la acoplará a una lista lineal.
- Primero crearemos una lista de listas para usar en nuestro ejemplo; luego usaremos listas por comprensión para aplanar la lista de una manera Pythonic; finalmente, imprimimos la lista resultante en la pantalla para su verificación.

```
# Creación de lista de listas (una lista donde todos sus elementos son listas)
list_of_lists = [['manzana', 'pera', 'plátano', 'uvas'], ['cebra', 'burro', 'elefante', 'vaca'], ['vainilla', 'chocolate'], ['princesa', 'Príncipe']]

# Aplanar la lista de listas en una sola lista
flat_list = [element for sub_list in list_of_lists for element in sub_list]

# Imprime ambos para comparar
print(f'List of lists:\n{list_of_lists}')
print(f'Flattened list:\n{flat_list}')
```



# Ordenar lista de tuplas

- Este fragmento considerará la idea de ordenar tuplas según el elemento especificado. Las tuplas son una estructura de datos de Python que a menudo se pasa por alto y son una excelente manera de almacenar datos relacionados sin usar un tipo de estructura más complejo.
- En este ejemplo, primero crearemos una lista de tuplas de tamaño 2 y las llenaremos con datos numéricos; a continuación clasificaremos los pares, por separado por el primer y segundo elemento, imprimiendo los resultados de ambos procesos de clasificación para inspeccionar los resultados; finalmente, ampliaremos esta clasificación a elementos de datos alfanuméricos mixtos.

```
# Algunos datos emparejados
pares = [(1, 10.5), (5, 7.), (2, 12.7), (3, 9.2), (7, 11.6)]
# Ordenar pares por primera entrada
sorted_pairs = sorted(pares, clave = lambda x: x [0])
print (f'Ordenado por elemento 0 (primer elemento): \n {sorted_pairs} ')
# Ordenar pares por segunda entrada
sorted_pairs = sorted(pares, clave = lambda x: x [1])
print (f'Ordenado por elemento 1 (segundo elemento): \n {sorted_pairs} ')
# Extiende esto a tuplas de tamaño n y entradas no numéricas
pares = [('banana', 3), ('manzana', 11), ('pera', 1), ('sandía', 4), ('fresa', 2), ('kiwi', 12) ]
sorted_pairs = sorted(pares, clave = lambda x: x [0])
print (f'Apareces alfanuméricos ordenados por elemento 0 (primer elemento): \n {sorted_pairs} ')
```



# Procesamiento de datos con Pandas

- Al cargar nuestros datos, podemos ver una serie de tipos de características únicas. Tenemos características categóricas como “Employee\_Name” y “Position”. Tenemos funciones binarias como “MarriedID”. Tenemos características continuas como “PayRate” y “EmpSatisfaction”.
- Tenemos funciones discretas como “DaysLateLast30” y finalmente tenemos funciones de fecha como “LastPerformanceReview\_Date”.

```
import numpy as np
import pandas as pd
import datetime
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.pipeline import make_pipeline
from feature_engine import missing_data_imputers as mdi
from feature_engine import categorical_encoders as ce
from sklearn.model_selection import train_test_split
%matplotlib inline
with open('HRDataset.csv') as f:
    df = pd.read_csv(f)
f.close()
df.head()
df.info()
```



# Variabilidad de características alta o baja

- El primer paso que suelo dar es revisar el recuento único de valores por característica para determinar si alguna característica se puede eliminar rápidamente debido a una variabilidad muy alta o muy baja.
- En otras palabras, ¿tenemos características que tengan tantos valores únicos como la longitud del conjunto de datos o características que tengan un solo valor único?

```
for col in df.columns:  
    print(col, df[col].nunique(), len(df))
```

- Podemos eliminar con seguridad “Employee\_Name”, “Emp\_ID”, “DOB” ya que la mayoría, si no todos, los valores son únicos para cada función. Además, podemos eliminar “DaysLateLast30” ya que esta característica solo contiene un valor único.

```
df.drop(['Employee_Name'], axis=1,  
        inplace=True)  
df.drop(['EmpID'], axis=1, inplace=True)  
df.drop(['DOB'], axis=1, inplace=True)  
df.drop(['DaysLateLast30'], axis=1,  
        inplace=True)
```





# Funciones duplicadas

- A continuación, al examinar el libro de códigos, que contiene las definiciones de cada función, podemos ver que tenemos muchas funciones duplicadas.
- Por ejemplo, “MarriedStatusID” es una función numérica que produce el código que coincide con los estatutos de casados en la función “MaritalDesc”. Podemos eliminar estas características.

```
df.drop(['MaritalStatusID', 'EmpStatusID',
'DeptID'], axis=1, inplace=True)
df.drop(['GenderID'], axis=1, inplace=True)
df.drop(['PerformanceScore'], axis=1,
inplace=True)
df.drop(['MarriedID'], axis=1,
inplace=True)
```





UNIVERSIDAD  
CATÓLICA  
SEDES SAPIENTIAE