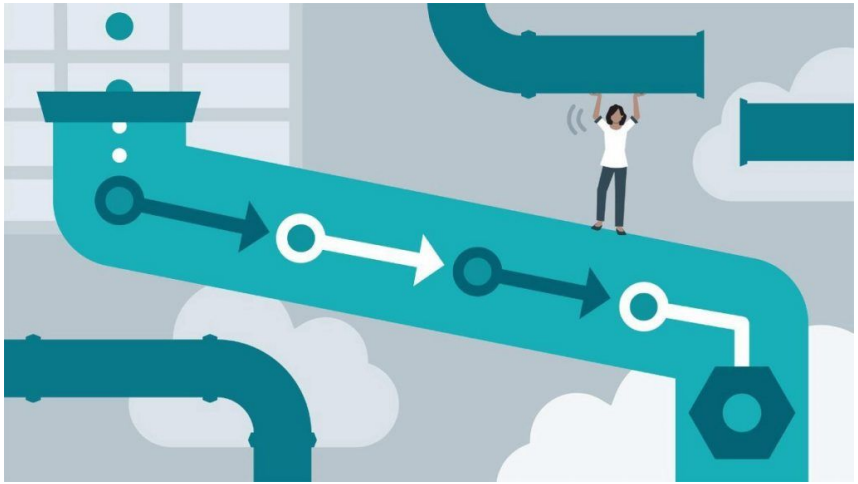
Clase: Del Dataset a la Red Neuronal – Un Viaje por el Aprendizaje Automático

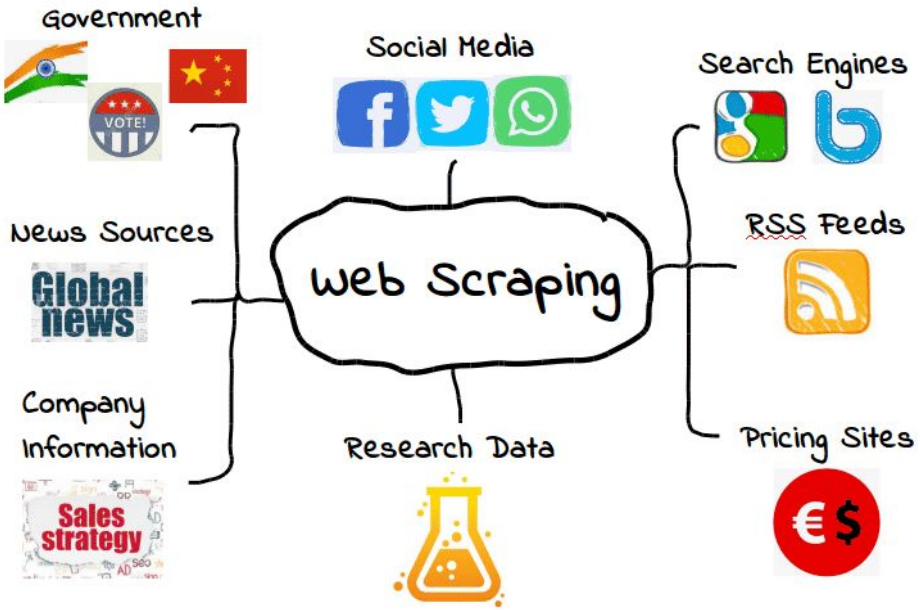
Objetivos

- Objetivo: Crear un pipeline completo desde la preparación de datos hasta el entrenamiento de un modelo profundo.



Construcción del Dataset

- Recolección desde CSV, APIs, scraping, etc.
- Limpieza: nulos, duplicados, outliers
- Visualización: distribuciones, correlaciones



Técnicas de Muestreo

- Las **técnicas de muestreo** en aprendizaje automático se utilizan para **modificar la distribución de clases en un dataset**, especialmente cuando hay un problema de **desequilibrio de clases** (por ejemplo, 95% clase A y 5% clase B). Estas técnicas ayudan a entrenar modelos más justos y precisos.

🎯 ¿Por qué son importantes?

- Cuando un modelo se entrena con un conjunto de datos desbalanceado, **tiende a favorecer la clase mayoritaria**, lo que genera una baja precisión para la clase minoritaria (la que muchas veces es más importante, cómo detectar fraudes o enfermedades).

📦 Tipos de técnicas de muestreo

1. Muestreo Estratificado (Stratified Sampling)

- Se utiliza al dividir los datos en conjuntos de entrenamiento y prueba.
- **Mantiene la proporción de clases** en ambos conjuntos.
- Asegura que la representación de cada clase sea la misma en cada subconjunto.

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y, test_size=0.3)
```

📦 Undersampling

- **Reduce el número de ejemplos de la clase mayoritaria** para equilibrar con la clase minoritaria.
- Es útil si tienes muchos datos, pero puede hacer que el modelo pierda información importante.

🟠 Ejemplo:

- Clase A: 950 muestras → se reducen a 50
- Clase B: 50 muestras → se mantiene igual

📦 Oversampling

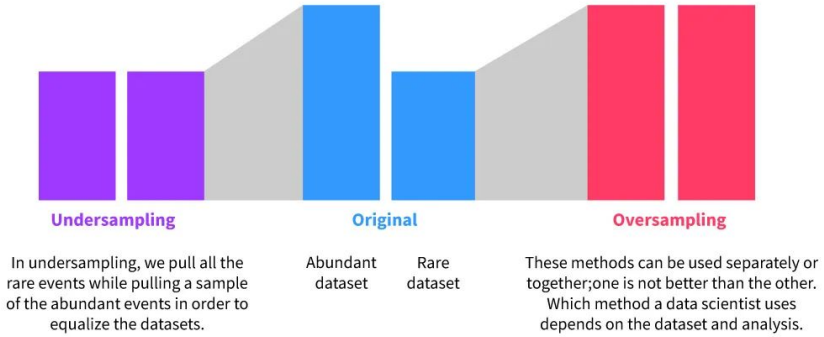
- **Duplica o genera nuevos ejemplos para la clase minoritaria**.
- Ayuda a mejorar el aprendizaje del modelo sobre esa clase.

✅ **Técnicas comunes:**

- **Random Oversampling**: copia aleatoriamente ejemplos de la clase minoritaria.
- **SMOTE (Synthetic Minority Over-sampling Technique)**: genera nuevos ejemplos sintéticos **interpolando entre muestras reales**.

```
from imblearn.over_sampling import SMOTE
smote = SMOTE()
X_resampled, y_resampled = smote.fit_resample(X, y)
```

🧠 Cual usar ?

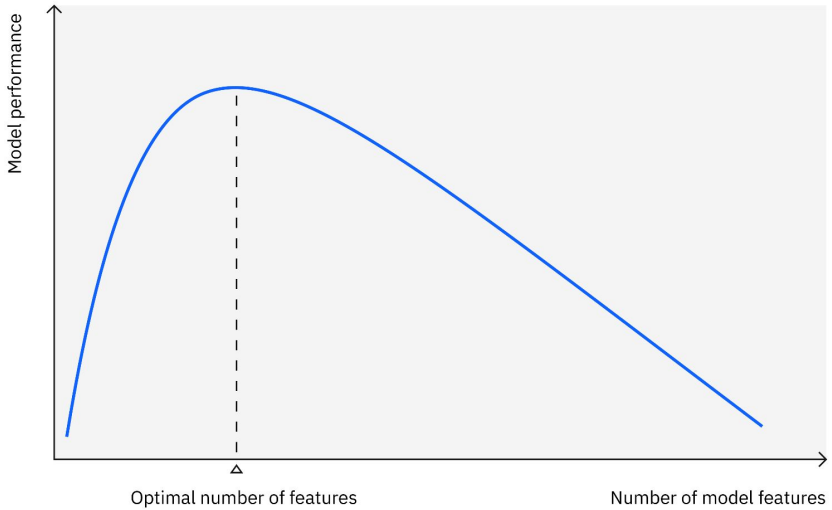


Técnica	Pros	Contras
Muestreo estratificado	Fácil, conserva proporción	No corrige el desbalance en sí
Undersampling	Simple y rápido	Puede perder datos importantes
Oversampling / SMOTE	Mejora representación de clases	Puede sobreajustar o agregar ruido

Reducción de dimensionalidad

La **reducción de dimensionalidad** es una técnica que se usa cuando trabajamos con datasets que tienen **muchas variables (features)**. El objetivo es simplificar los datos, manteniendo la mayor cantidad de información posible. Esto ayuda a:

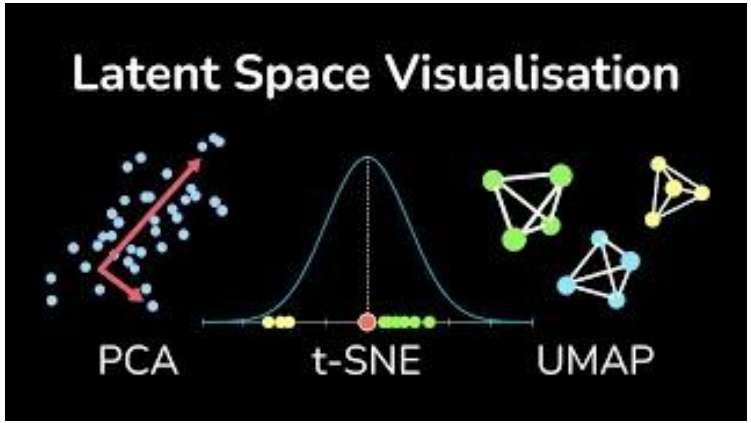
- Acelerar el entrenamiento de modelos.
- Eliminar el “ruido” de variables irrelevantes.
- Visualizar datos en 2D o 3D (cuando originalmente tienen muchas dimensiones).



🎯 ¿Para qué sirve?

- Visualizar **clústeres naturales** en los datos.
- Comprobar si las clases están **bien separadas**.
- Explorar cómo se distribuyen los datos antes de aplicar un modelo.

https://vimeo.com/340677521



🔧 Técnicas comunes

PCA (Análisis de Componentes Principales)

- Es una técnica **lineal**.
- Busca las **direcciones de máxima varianza** en los datos y las proyecta en un nuevo espacio de menor dimensión.
- Ideal para mantener **relaciones globales** en los datos.

🔍 Ejemplo: Reducir un dataset de 30 columnas a solo 2 componentes principales para visualizarlo en 2D.

```
from sklearn.decomposition import PCA

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

🔧 Técnicas comunes

t-SNE (t-distributed Stochastic Neighbor Embedding)

- Técnica **no lineal**, ideal para **visualización**.
- Preserva bien las **relaciones locales** (qué puntos están cerca de cuáles).
- No sirve para entrenamiento, pero sí para entender cómo se agrupan los datos.

```
from sklearn.manifold import TSNE

tsne = TSNE(n_components=2)
X_tsne = tsne.fit_transform(X)
```

🔧 Técnicas comunes

UMAP (Uniform Manifold Approximation and Projection)

- Similar a t-SNE, pero más rápido y escalable.
- También es **no lineal** y muy usado para explorar estructuras complejas en los datos.
- Preserva tanto **relaciones locales como globales** mejor que t-SNE en muchos casos.

```
import umap

reducer = umap.UMAP(n_components=2)
X_umap = reducer.fit_transform(X)
```

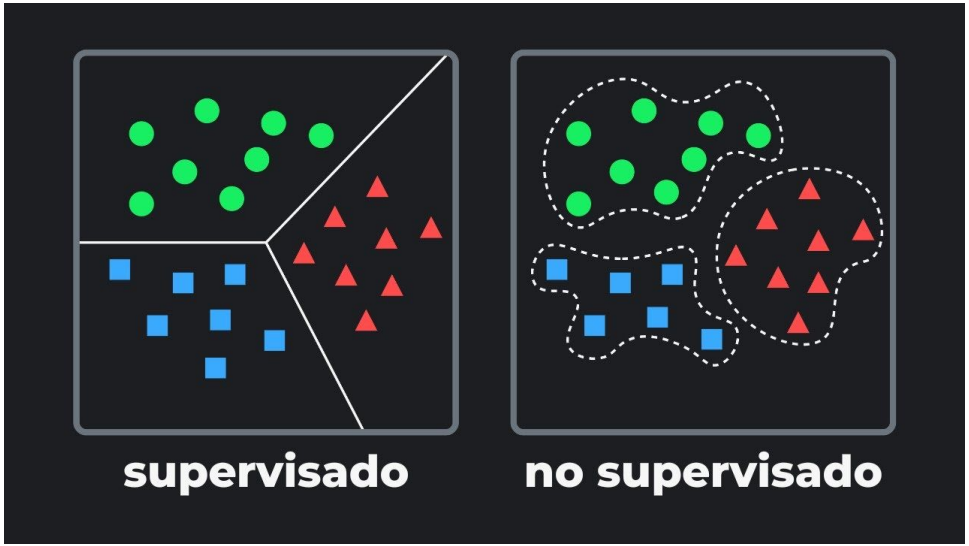
Análisis con Aprendizaje No Supervisado

¿Qué es el Aprendizaje No Supervisado?

Es un tipo de aprendizaje automático donde **no tenemos etiquetas o clases**. El modelo trata de **descubrir patrones ocultos** en los datos sin que le digamos qué buscar.

Se usa principalmente para:

- **Agrupación de datos (clustering)**
- **Reducción de dimensionalidad**



📦 Técnicas más comunes

Clustering (agrupamiento)

El objetivo es **agrupar observaciones similares entre sí**, sin conocer las etiquetas.

Algoritmo	Características clave
K-Means	Divide los datos en K grupos, optimiza la distancia media al centroide.
DBSCAN	Detecta grupos de densidad, útil para formas no esféricas y ruido.
Hierarchical	Agrupa en forma de árbol (dendrograma), útil para exploración jerárquica.

```
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters=2, random_state=42)
clusters = kmeans.fit_predict(X_pca)
```

📦 Técnicas más comunes

Métricas de evaluación (sin etiquetas)

Como no hay "verdad", usamos métricas internas:

- **Silhouette Score**: mide cuán bien separado está un clúster del resto.
- **Davies-Bouldin Index**: compara compactación vs separación entre clústeres.

```
from sklearn.metrics import silhouette_score
score = silhouette_score(X_pca, clusters)
```

Silhouette Score

Ejemplo práctico:

Si aplicas KMeans con `n_clusters=2` y obtienes:

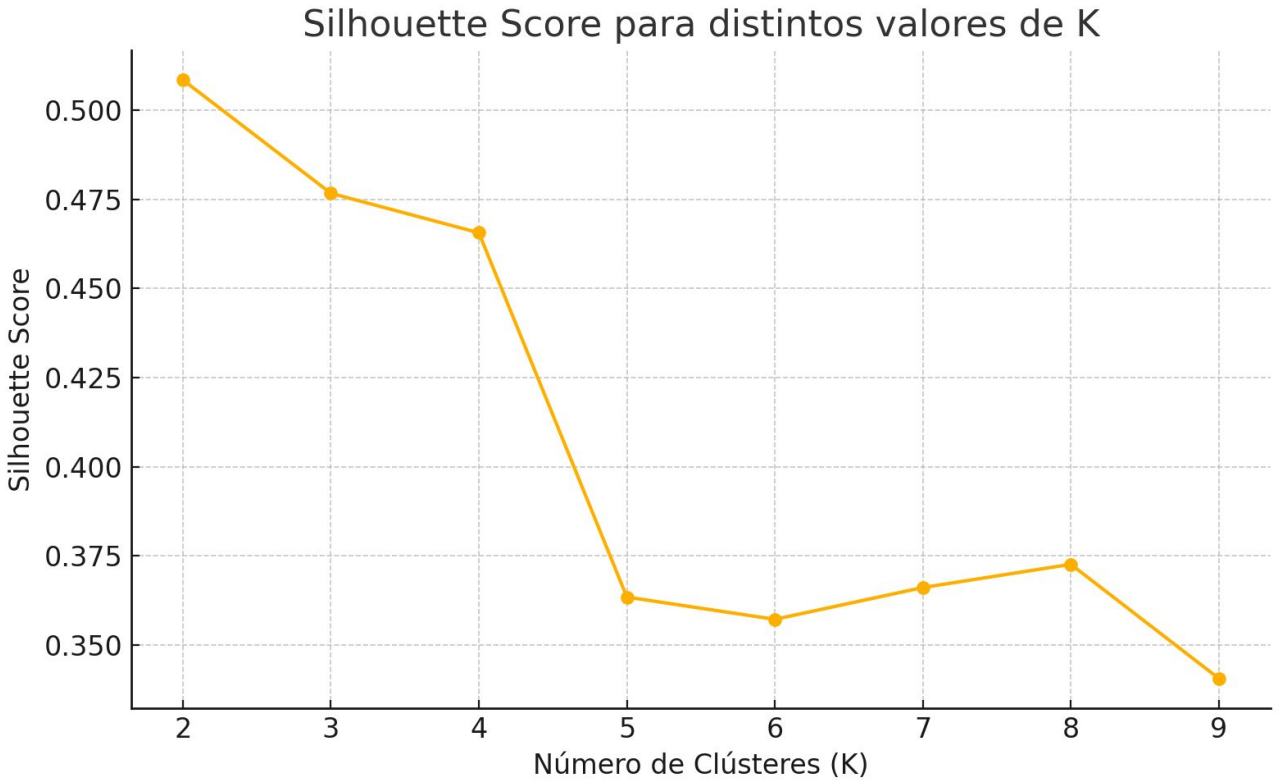
- **Silhouette Score = 0.65** → buena separación
- **Silhouette Score = 0.10** → probablemente no hay estructura real

Consejos:

- Prueba con distintos valores de `n_clusters` y elige el que dé el **mayor Silhouette Score**.
- El Silhouette Score **no dice si los clústeres tienen sentido semántico**, solo mide su separación geométrica.

Valor del Silhouette Score	Interpretación
≈ 1.0	Excelente separación entre clústeres
0.5 – 0.8	Buena estructura, separación razonable
0.2 – 0.5	Estructura débil, puede haber solapamiento
< 0.2	Clústeres mal definidos o sin estructura
< 0.0	Posibles errores: puntos mal asignados

Relación cluster x Silhouette Score



Centroide y outlier

¿Qué es un centroide?

El **centroide** de un clúster es el **punto promedio** de todos los elementos que pertenecen a ese grupo. Es como el “centro de gravedad” del clúster.

En términos técnicos:
Es el **vector promedio** de todas las observaciones (puntos) dentro de un clúster.

Supón que estás agrupando clientes según:

- Edad
- Ingresos

Y el centroide del clúster 1 tiene:

- Edad = 25
- Ingresos = 30000

Entonces, podrías decir que **ese grupo representa jóvenes de ingresos bajos**.

Centroide y outlier

¿Qué es un outlier?

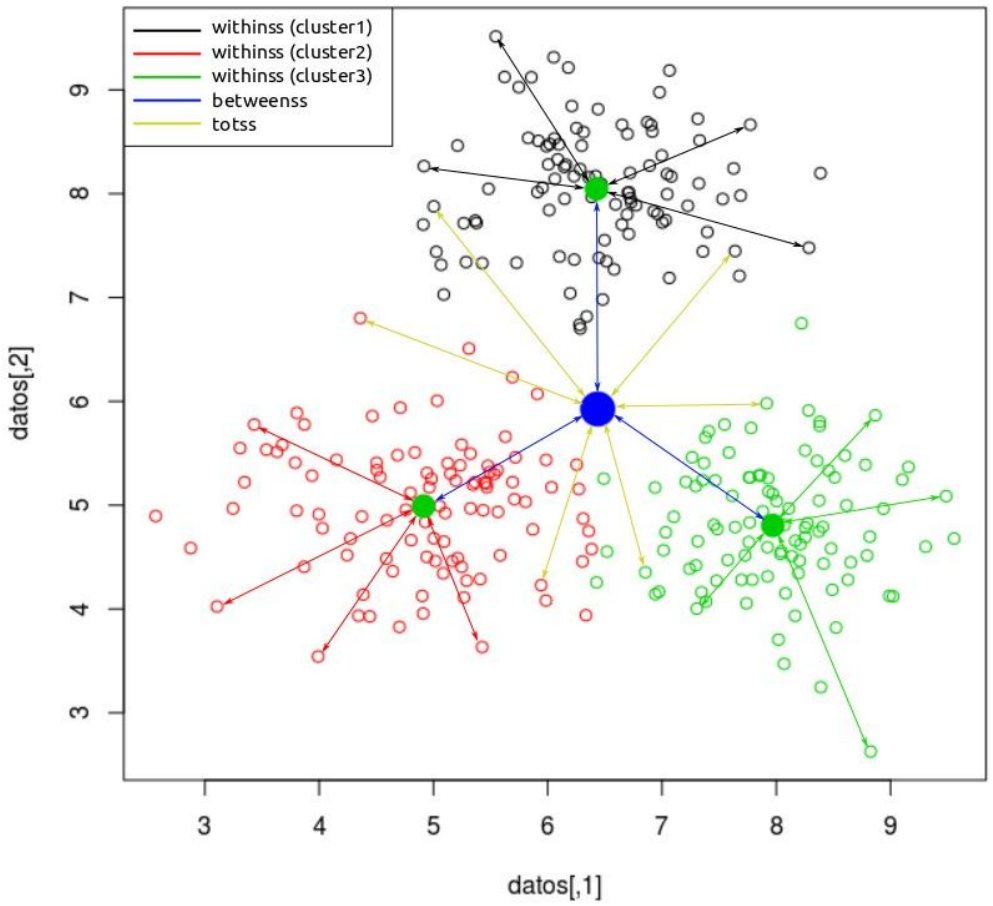
Un **outlier** es una observación que **se desvía significativamente** del resto de los datos. Puede ser:

- Un error de medición.
- Un caso raro.
- Una observación legítima pero poco común.

Un outlier en un clúster es un punto que:

- Fue asignado a un clúster porque **no había otra opción mejor**.
- Está **lejos del centroide** del clúster.
- Puede **afectar negativamente** la calidad del agrupamiento.

Ejemplo


withinss (cluster1)
withinss (cluster2)
withinss (cluster3)
betweenss
totss
datos[,2]
datos[,1]
3
4
5
6
7
8
9

Ejemplo

https://drive.google.com/file/d/17eJmfXVE4VkBIP0O0xm1RY6Fd96cjJ2V/view?usp=sharing

Centroide:
https://drive.google.com/file/d/1uoWE9WnFFD4sTUPzq89_xq3O7y8zMBZg/view?usp=sharing

🎨 Visualización

Generalmente, usamos reducción de dimensionalidad (como **PCA**, **t-SNE** o **UMAP**) para representar los datos en 2D antes de agruparlos.

Esto permite ver:

- Si los clústeres están claramente definidos.
- Si hay superposición.
- La estructura del dataset.

🧪 Actividad Práctico (flujo completo - 45 min) :

1. Cargar un dataset sin etiquetas claras (sklearn wine).
2. Aplicar reducción de dimensionalidad (PCA).
3. Usar `KMeans` para agrupar los datos.
4. Visualizar los clústeres.
5. Medir la calidad con Silhouette Score.

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.datasets import load_wine
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA

df = load_wine(as_frame=True)
df = df.frame
df.head()
```

🧪 Actividad Práctico 2 (flujo completo - 45 min) :

1. Cargar un dataset sin etiquetas claras diferente a lo disponible en sklearn.
2. Aplicar reducción de dimensionalidad (PCA).
3. Usar `KMeans` para agrupar los datos.
4. Visualizar los clústeres.
5. Medir la calidad con Silhouette Score.
6. Presenta el centroide de gravedad y los outliers con tolerancia del 5%.