A background network diagram consisting of numerous small blue nodes connected by thin, light blue lines, creating a complex web-like structure. The nodes are distributed across the entire frame, with a higher density in the upper right and lower right areas.

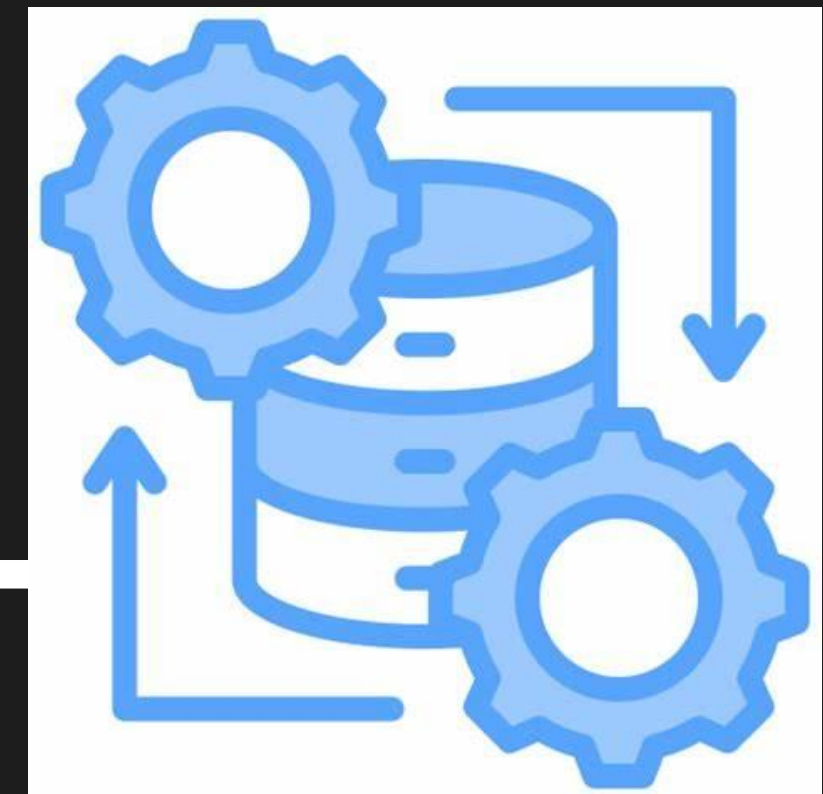
Aprendizaje de **Máquina Supervisado**

Sesión 3

El Concepto de Preprocesamiento

El preprocesamiento de datos es una etapa crítica en el flujo de trabajo de la ciencia de datos y el aprendizaje automático. Consiste en la preparación y transformación de los datos en bruto para que sean adecuados para su análisis y modelado. Los datos en bruto suelen estar incompletos, contener errores, ruido o inconsistencias que pueden afectar negativamente el rendimiento de los modelos.

En esta presentación, exploraremos los conceptos fundamentales del preprocesamiento y escalamiento de datos, incluyendo las técnicas más utilizadas y su implementación con la librería scikit-learn.



Importancia del Preprocesamiento

1 Mejora la calidad de los datos

Los datos en bruto suelen contener errores, valores faltantes, duplicados o inconsistencias. El preprocesamiento permite identificar y corregir estos problemas, lo que resulta en un conjunto de datos más limpio y confiable. Por ejemplo, en un dataset de ventas, algunos registros pueden tener valores nulos en la columna "Precio".

2 Reduce el ruido y la redundancia

Los datos pueden contener información irrelevante o redundante que no contribuye al análisis. El preprocesamiento ayuda a eliminar este ruido, lo que facilita la interpretación de los datos y mejora la eficiencia del modelo. Por ejemplo, en un dataset de clientes, puede haber columnas repetidas.

3 Mejora el rendimiento del modelo

Los modelos de Machine Learning funcionan mejor con datos limpios y normalizados. El preprocesamiento asegura que los datos estén en un formato adecuado para ser procesados por los algoritmos, lo que mejora la precisión y la eficiencia del modelo.

Tareas más frecuentes del preprocesamiento

Limpieza de Datos

La limpieza de datos es una de las tareas más importantes en el preprocesamiento. Incluye la eliminación de valores nulos, eliminación de duplicados y corrección de inconsistencias. Estas técnicas permiten obtener un conjunto de datos más confiable para el análisis posterior.

1

2

Normalización

La normalización es el proceso de escalar los datos para que estén en un rango específico, como $[0, 1]$ o $[-1, 1]$. Esto es especialmente útil cuando las variables tienen diferentes escalas, lo que puede afectar el rendimiento de los modelos que dependen de la distancia.

3

Transformación de Datos

La transformación de datos implica aplicar funciones matemáticas a los datos para mejorar su distribución o reducir el impacto de valores extremos. Algunas transformaciones comunes incluyen la logarítmica, la raíz cuadrada y la transformación de Box-Cox.

Codificación de Variables Categóricas

Label Encoding

El Label Encoding asigna un número único a cada categoría. Es útil cuando las variables categóricas tienen un orden natural o una jerarquía intrínseca. Por ejemplo, para la variable "Tamaño" con valores ["Pequeño", "Mediano", "Grande"], después de aplicar Label Encoding, los valores se transformarían en [0, 1, 2].

Ventajas	- Simplicidad: Es fácil de implementar y no aumenta la dimensionalidad de los datos.
	- Eficiencia: Es útil cuando las categorías tienen un orden natural, como en el caso de niveles educativos o rangos de edad.
Desventajas	- Relaciones numéricas artificiales: Al asignar números a las categorías, el modelo podría interpretar una relación numérica que no siempre es deseable.
	- No es adecuado para categorías sin orden: Si las categorías no tienen un orden natural (por ejemplo, colores), puede introducir sesgos en el modelo.

```
1 from sklearn.preprocessing import LabelEncoder
2
3 # Ejemplo de Label Encoding
4 le = LabelEncoder()
5 datos = ['Pequeño', 'Mediano', 'Grande', 'Mediano']
6 datos_transformados = le.fit_transform(datos)
7 print(datos_transformados) # Salida: [2 1 0 1]
```

Codificación de Variables Categóricas

One-Hot Encoding

El One-Hot Encoding convierte cada categoría en una columna binaria (0 o 1). Para cada categoría única en la variable original, se crea una nueva columna en el dataset. Si la categoría está presente en una fila, se marca con un 1; de lo contrario, se marca con un 0.

Ventajas	- Elimina relaciones numéricas artificiales: Convierte cada categoría en una columna binaria, evitando interpretaciones erróneas sobre relaciones numéricas.
	- Adecuado para categorías sin orden: Ideal para variables categóricas sin un orden natural, como colores o tipos de productos.
Desventajas	- Aumento de la dimensionalidad: Muchas categorías únicas pueden incrementar significativamente el número de columnas, afectando el rendimiento del modelo.
	- Dispersión de los datos: El dataset resultante puede contener muchas columnas con valores 0, dificultando el procesamiento en algunos casos.

```
1  from sklearn.preprocessing import OneHotEncoder
2  import numpy as np
3
4  # Ejemplo de One-Hot Encoding
5  datos = np.array([["Pequeño"], ["Mediano"], ["Grande"]])
6  ohe = OneHotEncoder()
7  datos_transformados = ohe.fit_transform(datos).toarray()
8  print(datos_transformados)
9  # Salida:
10 # [[0. 0. 1.]
11 #  [0. 1. 0.]
12 #  [1. 0. 0.]
```

Implementación con Scikit-Learn

Scikit-Learn proporciona clases como `LabelEncoder` y `OneHotEncoder` para transformar variables categóricas en valores numéricos. Estas herramientas son fáciles de usar y se integran perfectamente con el flujo de trabajo de Machine Learning.

```
1  from sklearn.preprocessing import LabelEncoder, OneHotEncoder
2  import numpy as np
3
4  # Paso 1: Crear un array de datos categóricos
5  datos_categoricos = np.array(["bajo", "medio", "alto", "medio"])
6  print("Datos categóricos originales:\n", datos_categoricos)
7  # Datos categóricos originales:
8  # ['bajo' 'medio' 'alto' 'medio']
9
10 # Paso 2: Aplicar Label Encoding
11 le = LabelEncoder()
12 datos_numericos = le.fit_transform(datos_categoricos)
13 print("Label Encoding:", datos_numericos) # Salida: [0 1 2 1]
14
15 # Aplicar One-Hot Encoding a los datos categóricos
16 # Nota: reshape(-1, 1) es necesario porque OneHotEncoder espera una matriz 2D
17 ohe = OneHotEncoder()
18 datos_transformados = ohe.fit_transform(datos_categoricos.reshape(-1, 1)).toarray()
19 print("One-Hot Encoding:\n", datos_transformados)
20 # Salida:
21 # One-Hot Encoding:
22 # [[0. 1. 0.]
23 #  [0. 0. 1.]
24 #  [1. 0. 0.]
25 #  [0. 0. 1.]]
```

Variables Dummy

Variables Dummy

Las variables dummy son una técnica de codificación similar al One-Hot Encoding, pero implementada directamente en DataFrames de Pandas. Convierten cada categoría única en una nueva columna binaria (0 o 1), donde un valor de 1 indica la presencia de la categoría y un valor de 0 indica su ausencia.

Ventajas de Variables Dummy

Elimina relaciones numéricas artificiales, facilita la interpretación de los datos y es compatible con diversos modelos de Machine Learning como regresión lineal, árboles de decisión y redes neuronales.

Implementación Variables Dummy

Con la librería Pandas podemos crear variables dummy.

```
1  import pandas as pd
2
3  # Crear un DataFrame con una columna categórica
4  df = pd.DataFrame({"Color": ["Rojo", "Azul", "Verde", "Azul", "Rojo"]})
5
6  # Aplicar variables dummy
7  df_dummies = pd.get_dummies(df, columns=["Color"])
8
9  print(df_dummies)
```

#Salida:

#	Color_Azul	Color_Rojo	Color_Verde
#0	False	True	False
#1	True	False	False
#2	False	False	True
#3	True	False	False
#4	False	True	False

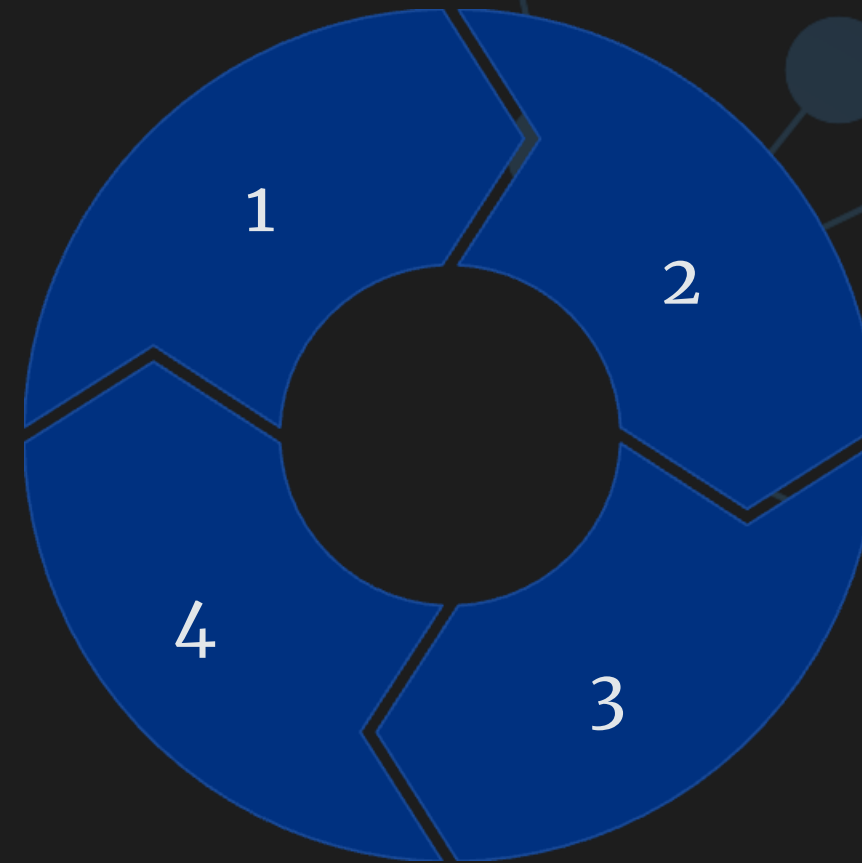
Escalamiento de Datos: El Concepto de Distancia

Importancia del Escalamiento

El escalamiento asegura que todas las características contribuyan equitativamente al cálculo de la distancia, mejorando el rendimiento del modelo y acelerando la convergencia en algoritmos iterativos.

Distancia Minkowski

Generalización de las distancias Manhattan y Euclidiana, con un parámetro p que determina la forma de la distancia.



Distancia Manhattan

Suma las diferencias absolutas de coordenadas, como moverse en una cuadrícula de calles. Fórmula: $d(A,B) = |x_2 - x_1| + |y_2 - y_1|$

Distancia Euclidiana

La distancia más corta entre dos puntos en un espacio Euclidiano. Fórmula: $d(A,B) = \sqrt{[(x_2 - x_1)^2 + (y_2 - y_1)^2]}$

Técnicas más usadas de Escalamientos

1

Escalador Min-Max

Normaliza los datos a un rango específico, generalmente [0, 1]. Fórmula: $X' = (X - X_{\min}) / (X_{\max} - X_{\min})$. Es fácil de implementar y mantiene la forma de la distribución original de los datos, pero es sensible a valores atípicos.

$$X_{\text{scaled}} = \frac{X - X_{\min}}{X_{\max} - X_{\min}}$$

2

Escalador Estándar

Transforma los datos restando la media y dividiendo por la desviación estándar, resultando en una distribución con media 0 y desviación estándar 1. Fórmula: $X' = (X - \mu) / \sigma$. Es menos sensible a valores atípicos en comparación con Min-Max Scaling.

$$X_{\text{scaled}} = \frac{X - \mu}{\sigma}$$

Comparación

El Escalador Min-Max es útil cuando se desea un rango específico y los datos no tienen valores atípicos significativos. El Escalador Estándar es preferible cuando los datos tienen una distribución normal o cuando hay valores atípicos que podrían distorsionar el rango.

Escalamiento con Scikit-Learn

MinMaxScaler

Normaliza los datos a un rango específico, generalmente [0, 1].

```
1 from sklearn.preprocessing import MinMaxScaler
2 import numpy as np
3
4 # Datos de ejemplo
5 datos = np.array([[10], [20], [30], [40]])
6
7 # Aplicar Min-Max Scaling
8 scaler = MinMaxScaler()
9 datos_escalados = scaler.fit_transform(datos)
10
11 print("Min-Max Scaling:\n", datos_escalados)
```

StandardScaler

Transforma los datos restando la media y dividiendo por la desviación estándar, resultando en una distribución con media 0 y desviación estándar 1.

```
1 from sklearn.preprocessing import StandardScaler
2 import numpy as np
3
4 # Datos de ejemplo
5 datos = np.array([[10], [20], [30], [40]])
6
7 # Aplicar Standard Scaling
8 scaler = StandardScaler()
9 datos_escalados = scaler.fit_transform(datos)
10
11 print("Standard Scaling:\n", datos_escalados)
```

Actividad Práctica Guiada

Implementación de Escalamiento

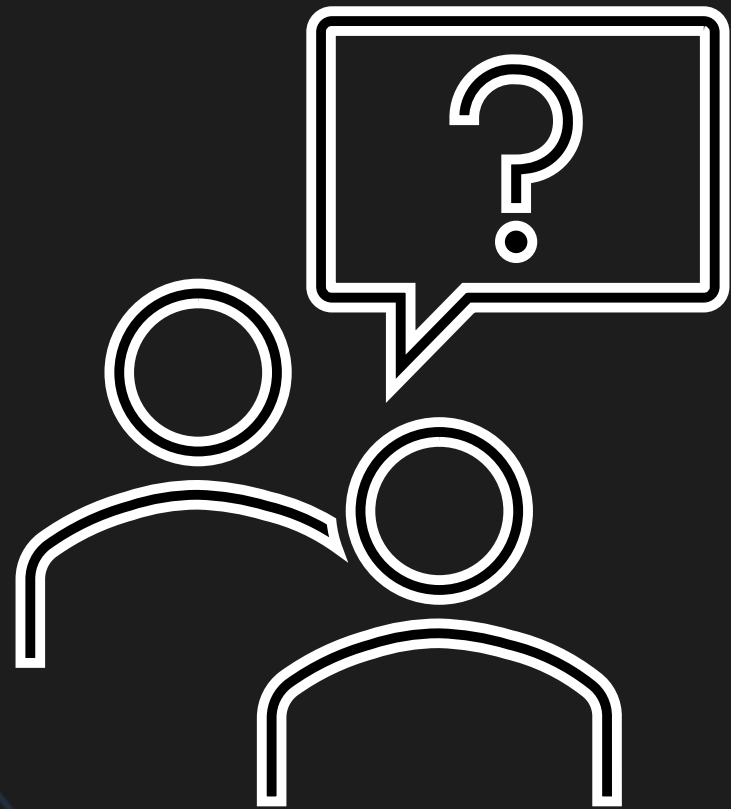
Requisitos:

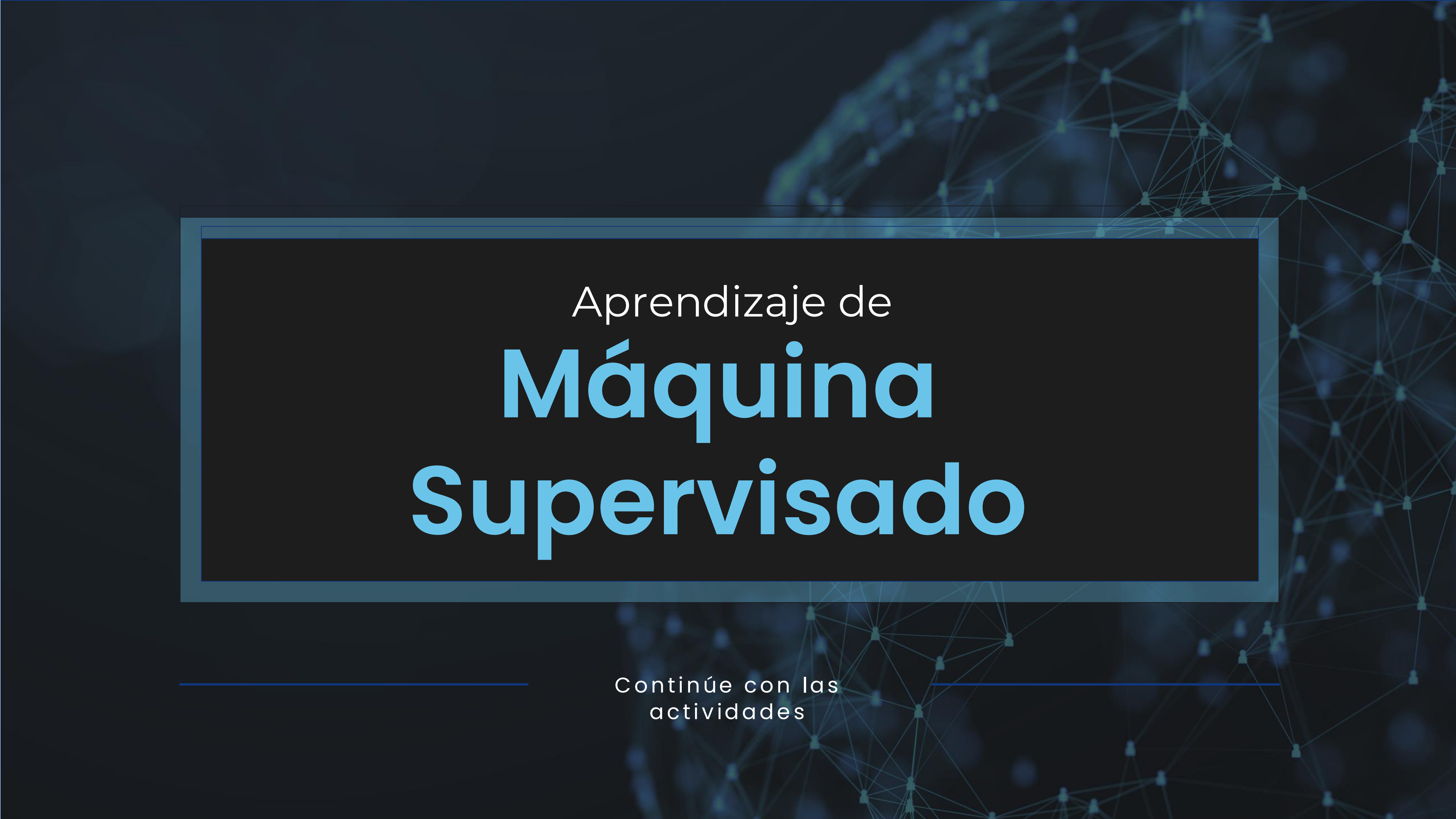
1. Importar las librerías y crear el dataset (datos en la guía de estudio).
2. Realiza el escalamiento con MinMaxScaler.
3. Luego, utiliza StandardScaler para transformar los datos restando la media y dividiendo por la desviación estándar.

El detalle de la actividad se encuentra en la guía de estudio de la sesión.

Preguntas

Sección de preguntas



The background of the slide features a complex network diagram with numerous nodes connected by lines, creating a web-like structure. The nodes are small squares, and the lines are thin and light blue. The overall color scheme is dark blue with lighter blue accents.

Aprendizaje de **Máquina Supervisado**

Continúe con las
actividades
