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**

Sesión 1

Mecanismos para mejorar el desempeño de un algoritmo

La optimización de modelos es un proceso fundamental para mejorar el rendimiento de algoritmos de Machine Learning. Exploraremos diversas técnicas para potenciar el desempeño de nuestros modelos, desde la ingeniería de características hasta el balanceo de datos.





Mecanismos para Mejorar el Desempeño

1 Feature Engineering

Proceso de transformar, crear y seleccionar variables de entrada para mejorar el rendimiento del modelo. Ayuda a capturar mejor las relaciones entre variables y facilita el aprendizaje.

2 Optimización de Hiperparámetros

Ajuste de las configuraciones que controlan el comportamiento del algoritmo antes del entrenamiento. Afecta directamente el rendimiento y la capacidad de generalización.

3 Regularización

Técnica para prevenir el sobreajuste penalizando la complejidad del modelo. Ayuda a que el modelo generalice mejor con datos nuevos no vistos durante el entrenamiento.

4 Balanceo de Datos

Estrategias para manejar clases desequilibradas en problemas de clasificación, evitando el sesgo hacia la clase mayoritaria y mejorando la predicción de clases minoritarias.

Feature Engineering

¿Qué es?

Es el proceso de transformar, crear y seleccionar variables de entrada para mejorar el rendimiento de un modelo. Ayuda a que el modelo capture mejor las relaciones entre las variables y la variable objetivo.

Pasos Principales

Incluye la creación de nuevas características a partir de las existentes, la selección de características relevantes eliminando las redundantes, y la transformación de datos mediante normalización, escalado o codificación de variables categóricas.

Ejemplos Prácticos

En predicción de precios de casas: crear "tamaño por habitación" dividiendo el tamaño entre el número de habitaciones. En clasificación de texto: usar TF-IDF para representar numéricamente el texto o calcular la frecuencia de palabras clave.



Hiperparámetros

Definición

Los hiperparámetros son configuraciones que se establecen antes de entrenar un modelo y no se aprenden durante el entrenamiento. A diferencia de los parámetros del modelo (como los coeficientes en regresión lineal), los hiperparámetros controlan cómo el algoritmo aprende.

Importancia

Afectan directamente el rendimiento del modelo y su capacidad de generalización. Controlan la complejidad del modelo, ayudando a evitar el sobreajuste. Una mala elección puede llevar a modelos que no generalizan bien con datos nuevos.

Ejemplos por Algoritmo

En árboles de decisión: `max_depth`, `min_samples_split`.
En regresión regularizada: `alpha`.
En K-NN: `n_neighbors`. En SVM: `C`, `kernel`. Cada algoritmo tiene hiperparámetros específicos que controlan su comportamiento.

Regularización

1

Concepto

La regularización es una técnica para prevenir el sobreajuste penalizando la complejidad del modelo. Añade un término de penalización a la función de costo que reduce la magnitud de los coeficientes, haciendo que el modelo sea más simple y generalice mejor.

2

Regularización L1 (Lasso)

Añade una penalización proporcional al valor absoluto de los coeficientes. Puede reducir algunos coeficientes exactamente a cero, funcionando como un selector de características automático. Útil cuando se sospecha que muchas características son irrelevantes.

3

Regularización L2 (Ridge)

Añade una penalización proporcional al cuadrado de los coeficientes. Reduce la magnitud de todos los coeficientes pero no los elimina completamente. Es útil cuando todas las características son potencialmente relevantes y se quiere reducir su impacto.

4

Elastic Net

Combina las penalizaciones L1 y L2, ofreciendo un equilibrio entre la selección de características y la reducción de la magnitud de los coeficientes. Es especialmente útil cuando hay muchas características correlacionadas.

Optimización de Parámetros

Definir el Espacio de Búsqueda

Identificar los hiperparámetros a optimizar y establecer los rangos de valores posibles para cada uno. Por ejemplo, para un Random Forest podríamos querer optimizar `n_estimators`, `max_depth` y `min_samples_split` con diferentes valores para cada uno.

Seleccionar la Estrategia de Búsqueda

Elegir entre Grid Search (exhaustivo pero costoso), Random Search (más eficiente para muchos hiperparámetros) u Optimización Bayesiana (más avanzada y eficiente). Cada estrategia tiene sus ventajas según el contexto y los recursos disponibles.

Implementar la Validación Cruzada

Utilizar técnicas como k-fold cross-validation para evaluar cada combinación de hiperparámetros de manera robusta, evitando el sobreajuste a un conjunto específico de datos de validación.

Seleccionar el Mejor Modelo

Comparar el rendimiento de las diferentes configuraciones según métricas relevantes (precisión, recall, F1-score) y seleccionar la combinación óptima de hiperparámetros para el modelo final.

Grilla de Parámetros

Definición de Hiperparámetros 1

Identificar los hiperparámetros clave que queremos optimizar. Por ejemplo, en un Random Forest podríamos seleccionar `n_estimators`, `max_depth` y `min_samples_split` como los hiperparámetros a ajustar.

2

Especificación de Valores

Para cada hiperparámetro, definir un conjunto de valores posibles a probar. Por ejemplo: `n_estimators`: [100, 200, 300], `max_depth`: [None, 10, 20], `min_samples_split`: [2, 5, 10].

Creación de la Grilla 3

Generar todas las combinaciones posibles de los valores de hiperparámetros. En nuestro ejemplo, tendríamos $3 \times 3 \times 3 = 27$ combinaciones diferentes para evaluar, formando nuestra grilla de búsqueda.

4

Evaluación y Selección

Entrenar y evaluar el modelo con cada combinación de la grilla, utilizando validación cruzada para obtener estimaciones robustas del rendimiento. Seleccionar la combinación que proporcione el mejor resultado según las métricas elegidas.



Balanceo de Datos



Problema del Desbalanceo

En problemas de clasificación, cuando una clase tiene muchas más muestras que otra, el modelo puede sesgarse hacia la clase mayoritaria. Esto resulta en un rendimiento pobre para la clase minoritaria, que suele ser la de mayor interés (como detección de fraude).



Submuestreo (Undersampling)

Reduce el número de muestras de la clase mayoritaria para equilibrar las clases. Es simple y rápido, pero puede perder información valiosa al descartar muestras. Útil cuando se tiene una gran cantidad de datos de la clase mayoritaria.



Sobremuestreo (Oversampling)

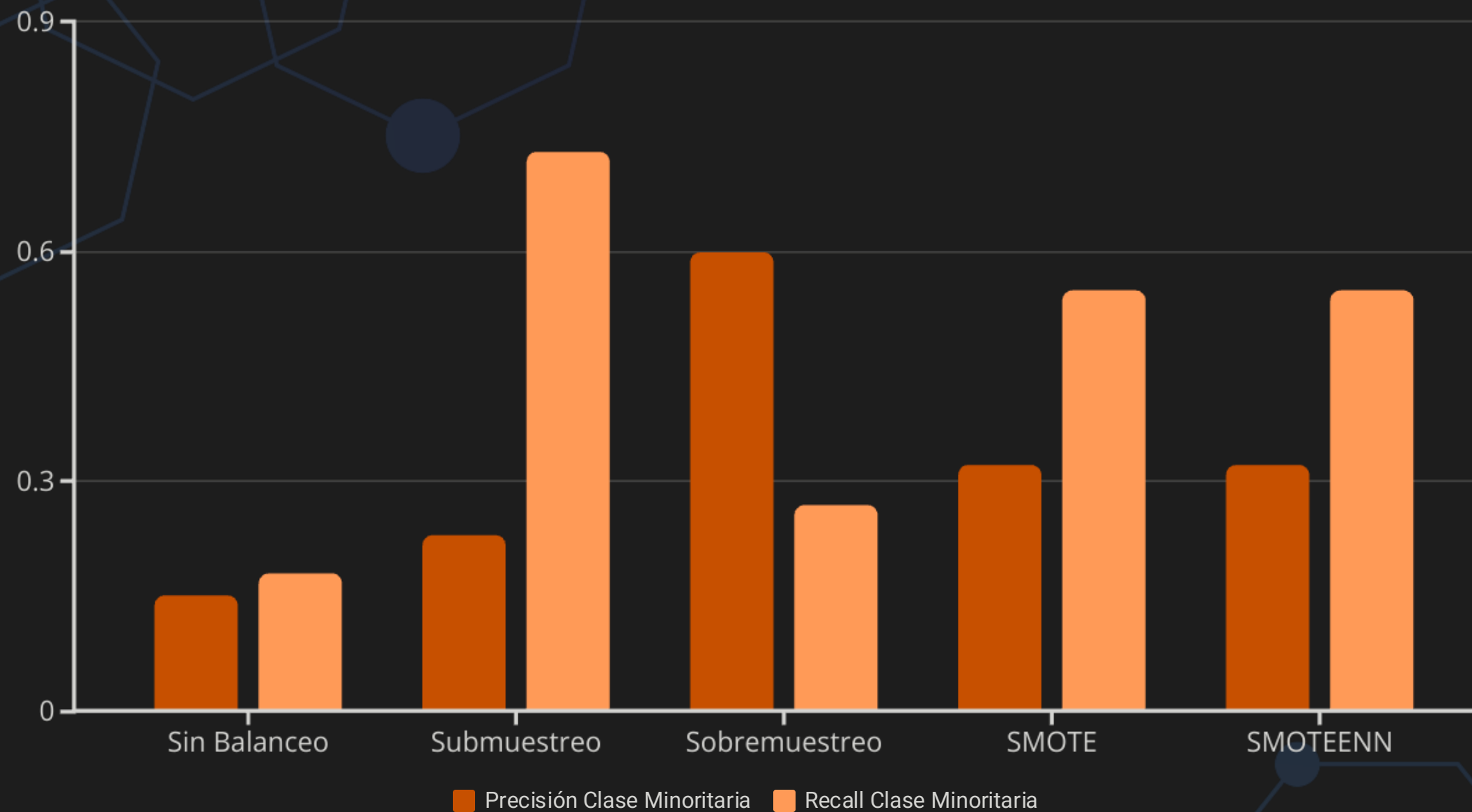
Aumenta el número de muestras de la clase minoritaria duplicándolas o generando nuevas. No pierde información pero puede causar sobreajuste si se duplican exactamente las mismas muestras. RandomOverSampler es una implementación común.



SMOTE y Técnicas Híbridas

SMOTE genera muestras sintéticas de la clase minoritaria interpolando entre muestras existentes. Las técnicas híbridas como SMOTEENN combinan sobremuestreo y submuestreo para un mejor equilibrio. Reducen el riesgo de sobreajuste.

Implementación en Scikit-Learn



La gráfica muestra el rendimiento de diferentes técnicas de balanceo de datos aplicadas a un conjunto desbalanceado (95% clase mayoritaria, 5% minoritaria). Observamos que el submuestreo mejora significativamente el recall pero sacrifica precisión, mientras que el sobremuestreo mejora la precisión pero reduce el recall. SMOTE y SMOTEENN ofrecen un equilibrio entre ambas métricas.

Estas técnicas se implementaron utilizando las bibliotecas scikit-learn e imbalanced-learn, que proporcionan herramientas robustas para la optimización de modelos de machine learning.

Ejercicio Guiado

Requisitos:

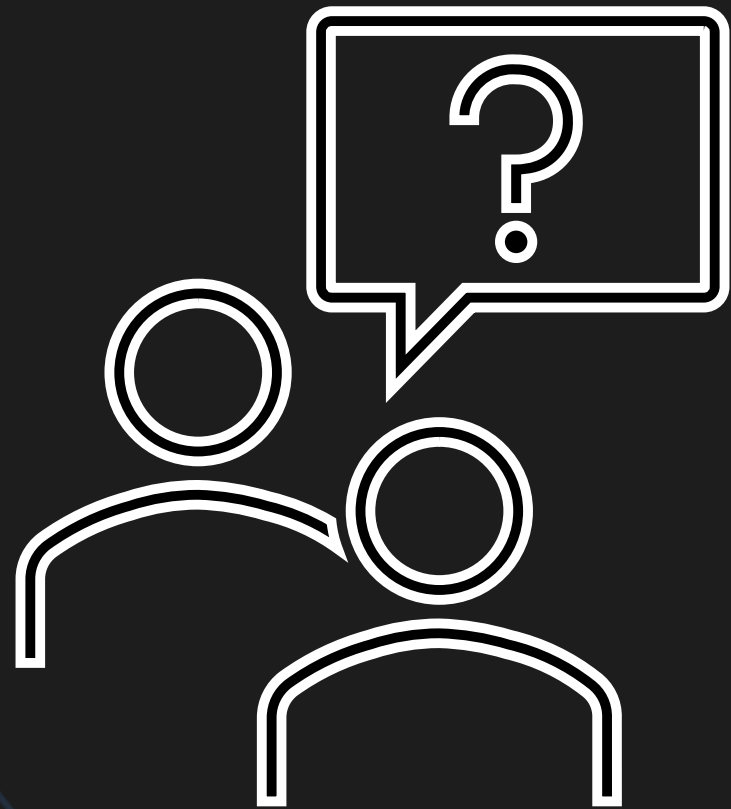
1. Importa las librerías.
2. Carga los Datos
3. Generación de datos desbalanceados
4. División de datos
5. Aplicación de las técnicas de balanceo:
 - o Submuestreo (Undersampling)
 - o Sobremuestreo (Oversampling)
 - o Combinación de Submuestreo y Sobremuestreo (SMOTEENN)
 - o Generación de Muestras Sintéticas (SMOTE)
6. Entrenamiento y evaluación del modelo

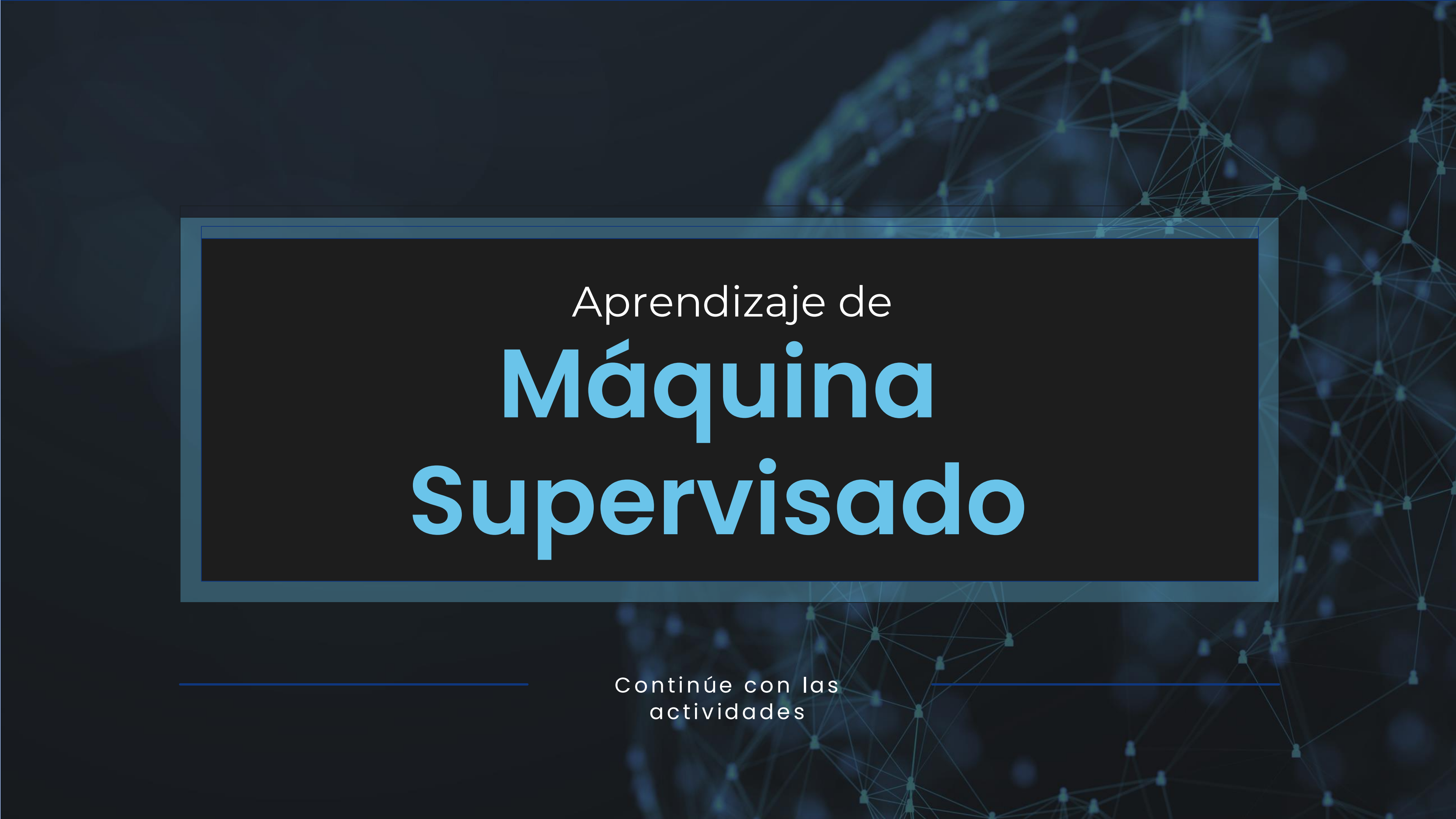


El detalle de la actividad se encuentra en guía de estudio de la sesión

Preguntas

Sección de preguntas



The background of the slide features a complex network diagram with numerous nodes and connecting lines, rendered in a light blue color against a dark blue background. The nodes are small squares, and the lines are thin and interconnected, creating a web-like structure that fills the entire slide.

Aprendizaje de **Máquina** **Supervisado**

Continúe con las
actividades