

Lead University

BCD Bachillerato Ingeniería en Ciencia de Datos

2025- I BCD7213 Minería de datos avanzada

Profesor:

PhD: Juan Murillo Morera

**Caso 1**

Estudiantes:

Marla Gómez

Carolina Salas

Fecha de Entrega: Marzo 23, 2025

**Resumen**

El caso de estudio consiste en desarrollar una aplicación gráfica interactiva utilizando Streamlit, la cual permita al usuario seleccionar entre técnicas de clasificación y regresión para aplicar modelos de minería de datos a conjuntos de datos específicos. La herramienta integrará métodos de preprocesamiento, limpieza y selección de atributos, y realizará benchmarking de modelos a través de la validación cruzada, la probabilidad de corte y el uso de métricas como el AUC para clasificación y el RMSE para regresión, permitiendo comparar el rendimiento de diferentes algoritmos tanto dentro de la misma familia como entre familias distintas. Además, se incorporará un módulo de series temporales para el modelado y predicción, complementando así las secciones de agrupamiento, clasificación y regresión, y ofreciendo una solución integral para determinar el modelo óptimo en función del dataset analizado.

**Clasificación de los algoritmos, métricas y balance de clases**

La clasificación es un tipo de aprendizaje supervisado en el cual el algoritmo aprende a asignar etiquetas o categorías a nuevas instancias, clasificando los datos de entrada en clases predefinidas (1). Es decir, a partir de ejemplos etiquetados de entrenamiento, el modelo aprende patrones que le permiten predecir la clase (categoría discreta) correspondiente a datos nunca antes vistos (1). Por ejemplo, un clasificador puede aprender a diagnosticar si un paciente tiene diabetes (Sí/No) en base a variables médicas, o distinguir correos spam de no spam.

Existen diversos algoritmos de clasificación, cada uno con sus características. Entre los más utilizados se encuentran la regresión logística, los árboles de decisión y sus ensambles (p. ej. bosque aleatorio), las máquinas de vectores de soporte (SVM), los métodos de vecinos más cercanos (k-NN) y los clasificadores Bayesianos ingenuos, entre otros. Cada técnica tiene fortalezas y debilidades dependiendo de la naturaleza de los datos (linealidad, tamaño, dimensionalidad, etc.). A continuación, la Tabla 1 resume algunas características de algoritmos comunes de clasificación:

**Tabla 1.** Comparación de algoritmos de clasificación comunes.

|  |  |
| --- | --- |
| Algoritmo | Características principales |
| Regresión Logística | Modelo lineal para clasificación binaria que estima la probabilidad de la clase positiva. Es sencillo, **rápido** y proporciona probabilidades bien calibradas; funciona mejor si la relación es aproximadamente lineal. |
| Decisition tree | Modelo basado en reglas jerárquicas if-then; divide recursivamente los datos según características. Captura relaciones **no lineales** y es **interpretable**, pero puede sobreajustar si no se poda adecuadamente. |
| Random Forest | Ensamble de múltiples árboles de decisión entrenados con aleatoriedad (bagging). Mejora la **precisión** y generalización reduciendo el sobreajuste de un solo árbol ([Clasificación en machine learning: Guía para principiantes |
| Máquina de Vectores de Soporte (SVM) | Encuentra el hiperplano óptimo que separa las clases con el mayor margen posible. Puede usar **kernels** para clasificar de forma no lineal. Es efectivo en alta dimensión, pero requiere ajustar sus hiperparámetros (como C y kernel) y no provee probabilidades directamente. |
| k-Neighbours Más Cercanos (k-NN) | Clasifica según las **instancias más cercanas** en el espacio de características. Es un método **no paramétrico** y simple, útil cuando las fronteras de decisión son complejas. Sin embargo, puede ser ineficiente con muchos datos y sensible a la escala de las variables. |

**Métricas de evaluación**

Para evaluar el desempeño de un modelo de clasificación se utilizan métricas derivadas de la matriz de confusión, como exactitud (accuracy), precisión, recuperación (recall), puntaje F1, entre otras (2). Una métrica particularmente importante en muchas aplicaciones es el área bajo la curva ROC (AUC). El AUC (Area Under the Curve) se refiere al área bajo la curva ROC (Receiver Operating Characteristic) y cuantifica la capacidad del modelo para separar correctamente las clases positivas de las negativas en diferentes umbrales de decisión (3). Un AUC de 1.0 indica un clasificador perfecto, mientras que un AUC de 0.5 equivale a un rendimiento azaroso (2) En términos probabilísticos, el AUC representa la probabilidad de que el modelo asigne una puntuación más alta a un ejemplo positivo aleatorio que a uno negativo (3). Cabe destacar que el AUC es independiente del umbral de clasificación utilizado; por ello, es útil para comparar modelos sin importar el punto de corte elegido (3). No obstante, la ROC/AUC asume clases relativamente balanceadas; cuando los datos están desequilibrados, a menudo es más informativo emplear curvas de precisión-exhaustividad (precision-recall) y su área bajo la curva correspondiente (3)

**Problemas balanceados vs. Desbalanceados**

En un problema balanceado, las clases están representadas en proporciones similares; en cambio, en la clasificación desbalanceada (desequilibrada) una clase minoritaria representa mucho menos casos que la mayoritaria (2). Por ejemplo, en un conjunto de diagnóstico médico podría haber 90% de casos negativos y 10% positivos, lo que constituye un dataset desbalanceado. Este desequilibrio puede causar que los clasificadores convencionales parezcan performar bien simplemente prediciendo siempre la clase mayoritaria, obteniendo alta exactitud pero ignorando la clase minoritaria. De hecho, modelos como árboles de decisión o regresión logística tienden a sesgarse hacia la clase mayoritaria si no se toman medidas (2). Por ello, en escenarios desbalanceados es preferible utilizar métricas como recall de la clase minoritaria, AUC-PR (precisión vs. recall) o el AUC-ROC con precaución, además de técnicas especiales. Para abordar el desbalance se emplean enfoques como muestreo (submuestreo de la clase mayoritaria o sobremuestreo de la minoritaria) o algoritmos sensibles al coste que penalizan más los errores sobre la clase minoritaria (2,3). Por ejemplo, en la detección de fraude (casos de fraude muy escasos) o en diagnóstico de enfermedades raras (2), es crucial manejar el desequilibrio para que el modelo detecte la clase importante aun si es minoritaria.

**Regresión (modelos, RMSE y aplicaciones)**

La regresión es una técnica de aprendizaje supervisado utilizada para predecir valores numéricos continuos en función de una o múltiples variables de entrada (1). A diferencia de la clasificación, donde se asigna una clase discreta, en regresión el objetivo es estimar una variable cuantitativa. Por ejemplo, predecir el precio de una vivienda dado su tamaño y ubicación, o pronosticar las ventas futuras de un producto según tendencias pasadas, son problemas típicos de regresión. Al igual que en clasificación, el modelo de regresión aprende una función a partir de datos de entrenamiento (con valores continuos conocidos) y luego generaliza para predecir nuevos casos (1).

Con respecto a los modelos de regresión más utilizados destaca la regresión lineal (que ajusta una relación lineal entre las características y la variable objetivo). También se emplean extensiones polinómicas (regresión polinómica) para capturar curvaturas, árboles de regresión y ensambles como bosques aleatorios de regresión (Random Forest Regressor), así como métodos de máquinas de soporte para regresión (SVR) y redes neuronales en casos más complejos. La selección del modelo depende de la naturaleza de la relación a modelar (lineal vs no lineal), la cantidad de datos y la interpretabilidad deseada, entre otros factores. Por ejemplo, los árboles y bosques pueden captar interacciones no lineales y variables categóricas fácilmente, mientras que la regresión lineal es más simple de interpretar y entrenar.

Para evaluar la calidad de las predicciones en regresión, una métrica fundamental es el Error Cuadrático Medio (MSE) y su raíz cuadrada, la Raíz del Error Cuadrático Medio (RMSE). El MSE calcula el promedio de los cuadrados de las diferencias entre los valores predichos por el modelo y los valores reales (5). Al elevar al cuadrado las diferencias, penaliza fuertemente los errores grandes. La fórmula del MSE para $n$ observaciones $y\_i$ con predicciones $\hat{y}\_i$ es:

MSE=1n∑i=1n(yi−y^i)2.MSE = \frac{1}{n}\sum\_{i=1}^{n} (y\_i - \hat{y}\_i)^2.

Dado que el MSE está en unidades al cuadrado de la variable objetivo (lo que dificulta su interpretación directa), se utiliza a menudo su raíz cuadrada. La RMSE se define como:

RMSE=MSE.RMSE = \sqrt{MSE}.

La RMSE tiene la ventaja de regresar a las unidades originales de la variable de salida, lo que facilita la interpretación del error promedio (5). En términos prácticos, la RMSE representa aproximadamente la magnitud promedio de los errores de predicción del modelo. Un valor de RMSE más bajo indica mejor ajuste (menor error promedio), mientras que valores altos revelan una mayor discrepancia entre predicciones y valores reales (5). Cabe notar que tanto MSE como RMSE son sensible(s) a valores atípicos, ya que unos pocos errores grandes pueden incrementar significativamente el valor al cuadrado (5). A pesar de ello, RMSE es ampliamente utilizada por su interpretabilidad y porque en muchos métodos de optimización (p. ej. mínimos cuadrados) se minimiza naturalmente el MSE.

Otras métricas de regresión incluyen el Error Absoluto Medio (MAE), el Coeficiente de Determinación ($R^2$), entre otras (5), pero el RMSE suele preferirse cuando se quiere penalizar fuertemente los grandes errores y expresar la precisión en las mismas unidades. Por ejemplo, si un modelo de regresión lineal se usa para predecir los niveles de glucosa en sangre de pacientes, una RMSE = 15 mg/dL indicaría que, en promedio, las predicciones difieren del valor real en 15 unidades de la medida de glucosa.

La regresión se aplica en numerosos campos: en finanzas para predecir precios de acciones o tipos de cambio, en inmobiliaria para estimar precios de casas (como en el ejemplo de bienes raíces donde las características de la propiedad predicen su valor (1), en salud para estimar parámetros clínicos (p. ej., tiempo de recuperación de un paciente según su tratamiento (1), en energía para pronosticar consumo eléctrico futuro (1), entre otros. En el contexto del caso de estudio, si se considerara un modelo de regresión, podría ser para predecir una medida cuantitativa derivada de los datos de pacientes (por ejemplo, nivel de riesgo o progresión esperada de una enfermedad en escala continua).

Dado que clasificación y regresión abordan problemas diferentes (categóricos vs continuos) también difieren sus métricas de evaluación principales. A modo de comparación resumida, la Tabla 2 contrasta las características de la métrica AUC (típica en clasificación) con la RMSE (típica en regresión):

**Tabla 2.** Comparación entre las métricas AUC y RMSE.

|  |  |  |
| --- | --- | --- |
| Aspecto | AUC (Área bajo ROC) | RMSE (Raíz del MSE) |
| Tipo de problema | Clasificación (binaria o multiclase binarizada) | Regresión (variable continua) |
| Definición | Probabilidad de que el modelo ordene aleatoriamente un positivo por encima de un negativo ([Clasificación: ROC y AUC | Machine Learning |
| Rango de valores | 0.5 (azar) a 1.0 (perfecto) en clasificación binaria; valores más altos indican mejor discriminación de clases ([Clasificación: ROC y AUC | Machine Learning |
| Interpretación | Independiente del umbral: resume la capacidad global de distinguir clases. Útil para comparar modelos, especialmente si las clases están balanceadas ([Clasificación: ROC y AUC | Machine Learning |
| Uso principal | Selección/comparación de clasificadores y ajuste de umbral. Especialmente relevante en problemas de clasificación con probabilidades de salida, donde se requiere una métrica global de desempeño. | Medir la precisión de predicciones de modelos de regresión. Útil para optimizar modelos durante el entrenamiento (minimizando MSE) y para comunicar el error esperado en términos intuitivos (unidad original). |

**Benchmarking de modelos de aprendizaje automático**

En minería de datos avanzada es común realizar benchmarking de modelos, es decir, una evaluación comparativa del rendimiento de distintos algoritmos o configuraciones de modelo sobre un mismo conjunto de datos. El objetivo del benchmarking es identificar qué modelo produce los mejores resultados según métricas definidas (por ejemplo, el AUC en clasificación o el RMSE en regresión), así como entender las diferencias de comportamiento entre ellos.

En la práctica, el benchmarking implica entrenar y probar múltiples modelos bajo condiciones equivalentes (misma partición de datos o mediante validación cruzada) y luego contrastar sus métricas. Es útil distinguir entre comparar algoritmos de una misma familia versus algoritmos de familias distintas. Comparar modelos de la misma familia suele significar evaluar variantes del mismo algoritmo con diferentes hiperparámetros o ajustes; por ejemplo, comparar varios árboles de decisión con distinta profundidad máxima, o comparar un bosque aleatorio con diferente número de árboles. Esto permite afinar el algoritmo elegido encontrando la configuración óptima. Por otro lado, comparar modelos de distintas familias implica enfrentar enfoques completamente diferentes (por ejemplo, comparar una regresión logística vs. un SVM vs. un árbol) para ver cuál tipo de modelo se adapta mejor a los datos. En este caso, el benchmarking ayuda en la selección del algoritmo más adecuado para el problema en cuestión.

Un buen benchmarking evalúa no solo la métrica principal de precisión predictiva, sino también otros factores como la variabilidad (intervalos de confianza de la métrica, evaluados quizá con validación cruzada), el tiempo de entrenamiento/predicción, el uso de recursos computacionales y la interpretabilidad de cada modelo. Por ejemplo, puede encontrarse que un modelo X tiene un AUC apenas 1% superior a otro modelo Y, pero si X toma el doble de tiempo en entrenar y es mucho más complejo, quizás Y sea preferible dependiendo de las restricciones del proyecto.

En el contexto de este caso de estudio, se podría realizar un benchmarking comparando, por ejemplo, un modelo de bosque aleatorio y uno de red neuronal para la tarea de clasificación de diabetes, evaluando cuál obtiene mejor AUC y sensibilidad. Igualmente, dentro de la misma familia de árboles, se podría comparar un árbol de decisión simple frente al ensamble de bosque aleatorio para cuantificar la ganancia en rendimiento. Esta evaluación comparativa sistemática de los modelos brinda sustento para justificar la selección del modelo final en el reporte.

**Validación cruzada (k-fold cross-validation)**

Al entrenar y evaluar modelos, es fundamental emplear técnicas robustas que estimen su desempeño de forma confiable. La validación cruzada es una técnica ampliamente usada para este fin, pues garantiza que los resultados no dependan excesivamente de una única partición de datos (4). En lugar de entrenar el modelo una sola vez con unos datos fijos de entrenamiento y evaluar en un conjunto de prueba fijo (método hold-out), la validación cruzada repite el entrenamiento y prueba del modelo múltiples veces con diferentes particiones, promediando luego los resultados (4). Esto proporciona una medida más estable y general del rendimiento esperado.

El tipo más común es la validación cruzada de $k$ pliegues (k-fold). En un esquema k-fold, los datos disponibles se dividen en $k$ subconjuntos (folds) aproximadamente iguales (4). Luego se realizan $k$ iteraciones de entrenamiento/validación: en cada iteración, uno de los $k$ folds se reserva como conjunto de validación (prueba) y los restantes $k-1$ folds se utilizan como \*\*conjunto de entrenamiento】 (4). De esta forma, el modelo se entrena y evalúa $k$ veces, usando un fold distinto como prueba en cada ocasión. Finalmente, se calcula el promedio de la métrica de desempeño a través de las $k$ iteraciones para obtener una estimación única de rendimiento (4).

Este procedimiento asegura que cada observación del dataset complete se utilice exactamente una vez para validación y $k-1$ veces para entrenamiento. Como ventaja, la validación k-fold aprovecha mejor los datos (no descarta gran parte para testing como el hold-out simple) y la estimación de desempeño es más precisa y estable, al estar promediada sobre $k$ experimentos diferentes (4). El principal inconveniente es el costo computacional: se entrena el modelo $k$ veces en lugar de una, lo cual puede ser lento si el modelo es complejo o el conjunto de datos muy grande (4). En la práctica, valores típicos son $k=5$ o $k=10$ (5-fold o 10-fold cross-validation), siendo 10-fold muy popular ya que ofrece buen balance entre sesgo y varianza de la estimación (4).

La validación cruzada es crucial durante el benchmarking de modelos y la selección de hiperparámetros, ya que ayuda a detectar sobreajuste: si un modelo tiene alto rendimiento en entrenamiento, pero consistentemente peor en las iteraciones de validación, indica que está sobreajustando a los datos de entrenamiento (4). Al usar cross-validation, nos aseguramos de que el modelo generaliza y podemos elegir el modelo con mejor promedio de desempeño en validación. En este caso de estudio, se podría usar, por ejemplo, una validación cruzada de 5-fold para comparar los modelos candidatos en la predicción de diabetes, promediando su AUC en las 5 particiones. Así, la evaluación final reportada sería más confiable que la de una sola partición de entrenamiento/prueba.

**Umbral de probabilidad (threshold) en clasificación**

En los modelos de clasificación binaria que predicen una probabilidad para la clase positiva (como la regresión logística, bosques aleatorios, redes neuronales, etc.), es necesario elegir un umbral de corte para traducir esa probabilidad en una decisión categórica (5). Por defecto, suele emplearse un umbral de 0.5 (es decir, predicción positiva si $P(\text{clase}=1) \ge 0.5$), asumiendo ambas clases igual de probables y con costos similares de error. Sin embargo, este threshold es ajustable y su elección puede tener un gran impacto en las métricas de rendimiento, especialmente en conjuntos de datos desbalanceados.

El umbral de probabilidad determina la sensibilidad vs. especificidad del clasificador: si se reduce el umbral (por ejemplo, de 0.5 a 0.3), más instancias serán clasificadas como positivas (aumenta la recuperación o tasa de verdaderos positivos), a costa de incluir más falsos positivos (baja la precisión). Por el contrario, un umbral más estricto (ej. 0.7) dará menos positivos previstos (más conservador, reduciendo falsos positivos, pero aumentando falsos negativos). La selección óptima depende de las prioridades del problema, que a veces vienen dictadas por los costos asociados a errores. Por ejemplo, en un sistema de detección de fraude, quizá se tolere revisar manualmente algunos casos (falsos positivos) con tal de atrapar la mayoría de los fraudes (minimizar falsos negativos); eso implicaría usar un umbral bajo para ser muy sensible. En cambio, para un diagnóstico médico donde un falso positivo podría someter a un paciente sano a tratamientos innecesarios costosos o riesgosos, quizá se prefiera un umbral más alto, sacrificando algo de sensibilidad para ganar especificidad.

La curva ROC precisamente muestra el trade-off de verdaderos positivos vs falsos positivos para todos los posibles umbrales. Una técnica común es buscar el umbral que maximiza cierta métrica de interés en el conjunto de validación (por ejemplo, maximizar la F1 o la curva Precision-Recall). También es útil observar la ROC para elegir un punto de operación adecuado. Por ejemplo, en la Figura ROC hipotética con A, B y C que indica diferentes umbrales (5), si los falsos positivos son muy costosos se elegiría un umbral correspondiente a un punto como A (FPR baja, aunque implique menor TPR) (3). Si, por el contrario, es crítico capturar prácticamente todos los positivos (falsos negativos costosos), se optaría por un umbral que produzca alta TPR como el punto C (aun si aumenta FPR) (3). En muchos casos, un balance intermedio (punto B) puede ser adecuado cuando los costos de ambos errores son comparables (3).

En datasets desequilibrados, ajustar el umbral es particularmente importante. Un modelo entrenado para optimizar AUC o accuracy puede aún otorgar bajas probabilidades a los casos positivos minoritarios; si nos quedamos con 0.5, podríamos no detectar casi ninguno. Es común entonces bajar el threshold para aumentar la detección de positivos. Adicionalmente, existen métodos como ajustar la función de costos o aplicar calibración de probabilidades para mejorar la toma de decisiones con umbral.

**Streamlit**

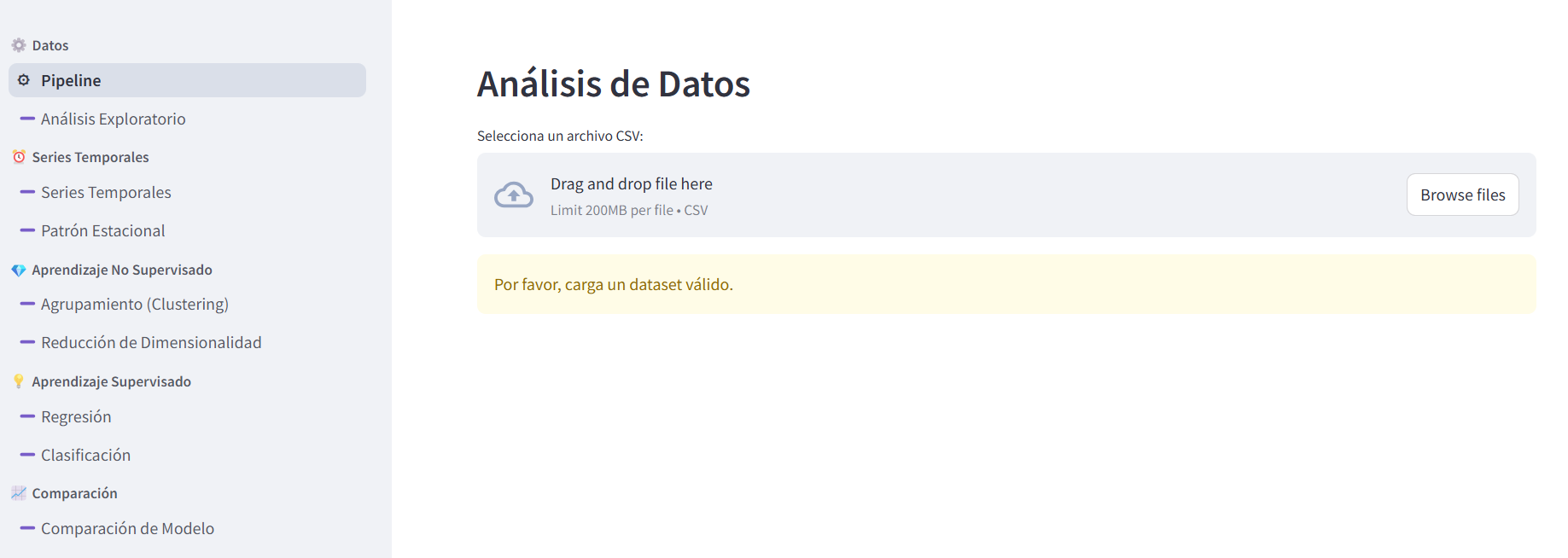
Streamlit es un marco de trabajo de Python de código abierto diseñado para construir aplicaciones web interactivas basadas en datos de manera rápida y sencilla (6). Orientado a científicos de datos y desarrolladores de *machine learning*, Streamlit permite convertir scripts de Python en aplicaciones web interactivas con tan solo agregar unas pocas líneas de código, sin necesidad de conocimientos profundos de desarrollo web (6). Esto lo hace muy adecuado para crear demos de modelos de ML, paneles de visualización y prototipos de herramientas de ciencia de datos.

Entre las características principales de Streamlit está su sencillez: el flujo de una aplicación Streamlit sigue el flujo natural del código Python, actualizándose automáticamente cada vez que cambia alguna entrada (en modo interactivo). Los desarrolladores pueden insertar fácilmente elementos de interfaz como textos, encabezados, tablas y gráficos, usando funciones como st.title(), st.write(), st.line\_chart(), etc. Asimismo, ofrece una variedad de widgets interactivos (con funciones st.button(), st.slider(), st.selectbox(), por ejemplo) que permiten al usuario ingresar datos o ajustar parámetros en la aplicación. Estos widgets facilitan construir formularios de entrada, filtros y controles en pocos pasos.

Streamlit provee una biblioteca de componentes preconstruidos para visualización de datos y creación de interfaces, lo que simplifica significativamente el proceso de desarrollo (6) Por ejemplo, integrarse con librerías de Python populares es directo: se pueden mostrar dataframes de pandas con st.dataframe(), graficar con Matplotlib, Plotly o Altair simplemente pasando las figuras u objetos de gráficos a funciones de Streamlit, e incluso mostrar mapas o imágenes con comandos dedicados. También soporta la organización del layout mediante columnas y una barra lateral (st.sidebar) donde se pueden agrupar controles.

Otra ventaja es que las apps creadas con Streamlit son fácilmente desplegables y compartibles. Al ser aplicaciones web, pueden alojarse en un servidor o en servicios en la nube (Streamlit Share, etc.) para que otros usuarios interactúen con ellas mediante el navegador. Esto resulta muy útil para compartir resultados de proyectos de ciencia de datos de forma interactiva, permitiendo a stakeholders explorar los modelos y datos por sí mismos.

En el contexto del caso de estudio, Streamlit se utilizará para desarrollar una interfaz interactiva que permita cargar nuevos datos y obtener la predicción del modelo de diabetes en tiempo real, además de visualizar métricas, gráficas ROC o distribuciones. Sus componentes de texto ayudarán a mostrar explicaciones y conclusiones, mientras que los widgets permitirán probar diferentes umbrales o seleccionar distintos modelos entrenados para comparar sus desempeños en vivo. De este modo, Streamlit sirve como puente para llevar los resultados del análisis de datos desde un *notebook* o script hasta una aplicación accesible e intuitiva.



**Resultados para dataset de diabetes**

*+ screens cuando se analice el dataset de diabetes*

**Referencias**

[1] Hewlett Packard Enterprise, “¿Qué es el aprendizaje automático supervisado? - Glosario,” *HPE LAMERICA*, [Online]. Available: <https://www.hpe.com/lamerica/es/what-is/supervised-machine-learning.html>. (accessed Mar. 21, 2025).

[2] DataCamp, “Clasificación en machine learning: Guía para principiantes,” *DataCamp Blog*, 2022. [Online]. Available: <https://www.datacamp.com/es/blog/classification-machine-learning>. (accessed Mar. 21, 2025).

[3] Google, “Clasificación: ROC y AUC,” *Machine Learning Crash Course*, Google for Developers, [Online]. Available: <https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc?hl=es-419>. (accessed Mar. 21, 2025).

[4] Amazon Web Services, “Validación cruzada,” in *Amazon Machine Learning – Developer Guide*. [Online]. Available: <https://docs.aws.amazon.com/es_es/machine-learning/latest/dg/cross-validation.html>. (accessed Mar. 21, 2025).

[5] CertiDevs, “Tutorial ScikitLearn: Métricas de Regresión,” *CertiDevs.com*, Oct. 2023. [Online]. Available: <https://certidevs.com/tutorial-scikit-learn-regresion-metricas-de-error>. (accessed Mar. 21, 2025).

[6] S. Navarro, “Crear una interfaz en Streamlit,” *KeepCoding Tech Blog*, 12 abr 2024. [Online]. Available: <https://keepcoding.io/blog/crear-una-interfaz-en-streamlit/>. (accessed Mar. 21, 2025).