UNIVERSIDAD TECNOLÓGICA DE CHIHUAHUA

Tecnologías de la información

Extracción de Conocimiento en Bases de Datos

III.2. Reporte de Métricas de Evaluación (50%)

Docente

ING. LUIS ENRIQUE MASCOTE CANO

Alumno

Erick Eduardo Maffiodo Delgado

IDGS 91N

Domingo, 30 de Noviembre del 2025

Índice

1. Introducción

En los proyectos de Machine Learning es fundamental evaluar objetivamente el desempeño de los modelos. Para ello se utilizan métricas de evaluación, las cuales cuantifican qué tan cerca están las predicciones del modelo de los valores reales. Este reporte tiene como objetivos: (1) investigar diversas métricas usadas para evaluar modelos de clasificación y de regresión, comprendiendo su definición, interpretación y limitaciones; y (2) aplicar dichas métricas en un caso práctico, entrenando un modelo clasificador K-Nearest Neighbors (KNN) con un conjunto de datos real y analizando sus resultados.

El caso práctico consiste en un problema de clasificación binaria. Se cuenta con una matriz de datos que incluye dos variables predictoras – glucosa en sangre y edad – y una etiqueta binaria que indica la pertenencia de cada registro a la clase positiva (valor 1) o negativa (valor 0). Un posible contexto de estos datos podría ser el diagnóstico de diabetes, donde glucosa y edad son factores utilizados para predecir si un paciente padece la condición (etiqueta = 1) o no (etiqueta = 0). Emplearemos un modelo KNN para esta tarea de clasificación. Primero, dividiremos los datos en entrenamiento y prueba, y normalizaremos las características. Luego entrenaremos el modelo con diferentes valores del hiperparámetro k (número de vecinos) y seleccionaremos el mejor basado en la métrica F1-score. Finalmente, evaluaremos el modelo seleccionado con diversas métricas de clasificación (exactitud, precisión, sensibilidad, F1, ROC-AUC), presentando la matriz de confusión resultante y la curva ROC correspondiente con su área bajo la curva (AUC). Con estos resultados, analizaremos el rendimiento obtenido y propondremos mejoras o alternativas de modelado.

2. Investigación de métricas de evaluación

En esta sección se describen algunas de las métricas más comunes para evaluar modelos de **clasificación** (se presentan 5 métricas) y de **regresión** (2 métricas). Para cada métrica se proporciona su definición formal (con fórmula matemática), su interpretación práctica y sus principales ventajas o limitaciones.

2.1. Métricas de clasificación

A continuación se abordan cuatro métricas clásicas de evaluación para modelos de clasificación binaria: **Exactitud (Accuracy)**, **Precisión (Precision)**, **Sensibilidad (Recall)** y **Puntuación F1 (F1-score)**. Además, se incluye la métrica **ROC-AUC** relacionada con la curva ROC. Todas estas métricas se derivan de la **matriz de confusión**, la cual resume las predicciones del modelo en términos de verdaderos positivos (TP), verdaderos negativos (TN), falsos positivos (FP) y falsos negativos (FN).

- **Exactitud (Accuracy):**

    - *Definición y fórmula:* la exactitud mide la proporción de predicciones correctas (tanto positivas como negativas) sobre el total de casos evaluados. Matemáticamente se define como el número de aciertos dividido entre el número total de ejemplos:

      Exactitud=TP+TNTP+TN+FP+FN\text{Exactitud} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}Exactitud=TP+TN+FP+FNTP+TN

      En otras palabras, es la fracción de ejemplos que el modelo clasifica correctamente. Un valor de exactitud = 1 (o 100%) indica un clasificador perfecto, mientras que una exactitud = 0 significa que el modelo falla en todas sus predicciones.

- *Interpretación:* en problemas balanceados (número similar de ejemplos en cada clase), la exactitud es una medida intuitiva del desempeño global del modelo. Por ejemplo, una exactitud de 0.90 (90%) indica que el modelo acierta en el 90% de las muestras (tanto positivas como negativas).

- *Ventajas:* es fácil de calcular y entender por audiencias no técnicas. Resume en un solo número la eficacia general del modelo cuando nos importan por igual ambas clases.

- *Limitaciones:* puede ser **engañosa en datos desequilibrados**. Si una clase minoritaria es muy poco frecuente, un modelo trivial que siempre prediga la clase mayoritaria puede tener una alta exactitud pero desempeño deficiente en la clase minoritaria. Por ello, la exactitud **no refleja** qué tipos de errores comete el modelo. En problemas con clases desbalanceadas se recomienda complementarla o reemplazarla con métricas que sí consideren ese desequilibrio.

- **Precisión (Precision):**

  - *Definición y fórmula:* la precisión positiva (también llamada valor predictivo positivo) es la proporción de ejemplos predichos como positivos por el modelo que realmente pertenecen a la clase positiva. Se calcula como:

    Precisio´n=TPTP+FP\text{Precisión} = \frac{\text{TP}}{\text{TP} + \text{FP}}Precisio´n=TP+FPTP

    donde TP es el número de verdaderos positivos y FP el número de falsos positivos. En esencia, la precisión responde a la pregunta: *"¿Qué fracción de las predicciones positivas del modelo son correctas?"*.

  - *Interpretación:* la precisión evalúa la **calidad de las predicciones positivas** del modelo. Un valor de precisión alto (cercano a 1) significa que **pocos de los casos**

señalados como positivos resultaron ser falsos positivos, es decir, el modelo es estricto a la hora de etiquetar positivos y suele acertar cuando lo hace. Por ejemplo, si un clasificador de spam tiene una precisión = 0.95, implica que el 95% de los correos que predijo como "spam" efectivamente eran spam real (5% fueron marcados incorrectamente).

- *Ventajas:* resulta muy útil cuando queremos minimizar los **falsos positivos**. Por ejemplo, en un diagnóstico médico o en un detector de fraudes, una alta precisión asegura que casi todos los casos detectados como positivos sean realmente válidos, evitando falsas alarmas costosas. La precisión es apropiada como métrica principal cuando el costo de una **alarma falsa** (FP) es alto y se requiere confianza en cada predicción positiva. Además, es intuitiva de entender: un modelo con precisión = 0.9 "acierta" el 90% de sus alertas positivas.

- *Limitaciones:* **ignora los falsos negativos**. Un modelo puede tener precisión del 100% si solo predice positivos en casos muy seguros, aunque pase por alto muchos positivos reales (FN). Por ello, la precisión por sí sola no refleja qué tantos positivos reales está dejando de identificar el modelo. En situaciones donde los falsos negativos importan (por ejemplo, no detectar una enfermedad cuando la hay), enfocarse solo en la precisión es insuficiente. En general, suele existir una **disyuntiva entre precisión y sensibilidad**: al exigir predicciones positivas muy confiables (alta precisión), el modelo puede volverse más conservador y omitirá algunos positivos (reduciendo la sensibilidad).

- **Sensibilidad (Recall):** (también conocida como **Exhaustividad** o **Tasa de Verdaderos Positivos, TPR**).

  - *Definición y fórmula:* la sensibilidad mide la proporción de casos positivos reales que el modelo logra identificar correctamente. Equivale matemáticamente a:

    Sensibilidad=TPTP+FN\text{Sensibilidad} = \frac{\text{TP}}{\text{TP} +

\text{FN}}Sensibilidad=TP+FNTP

donde FN son los falsos negativos. En otras palabras, responde a: *"¿Qué fracción del total de positivos existentes encuentra el modelo?"*.

- *Interpretación:* esta métrica evalúa la **capacidad del modelo para no dejar escapar positivos reales**. Un valor alto de sensibilidad (cercano a 1) indica que el clasificador encuentra casi todos los positivos en el conjunto de datos. Por ejemplo, una sensibilidad = 0.88 (88%) significa que el modelo logra detectar el 88% de los ejemplos positivos y solo un 12% se le escapan (falsos negativos). La sensibilidad es especialmente crítica en dominios donde perder un positivo tiene consecuencias graves (ejemplo: diagnósticos de enfermedades, detección de fraudes, etc.).

- *Ventajas:* es la métrica a priorizar cuando los **falsos negativos** son mucho más costosos que los falsos positivos. Por ejemplo, en una prueba médica de cáncer, una alta sensibilidad asegura que se detecten la mayoría de los enfermos (minimizando casos no diagnosticados). Es fundamental en aplicaciones donde es preferible generar algunas falsas alarmas con tal de *"no perder ningún caso positivo"*. Una sensibilidad perfecta (1.0) implica cero falsos negativos, es decir, el modelo no deja pasar ningún positivo real sin etiquetar.

- *Limitaciones:* **no toma en cuenta los falsos positivos**. Un modelo puede lograr sensibilidad = 1 si clasifica todo como positivo, pero en tal caso tendrá muchísimos falsos positivos y su precisión será muy baja. Es decir, por sí sola la sensibilidad puede llevar a modelos triviales (ejemplo: etiquetar todos los correos como spam da sensibilidad de 100% en detección de spam, pero es inútil debido a la cantidad de correos legítimos marcados incorrectamente). Por tanto, al igual que la precisión, la sensibilidad aislada no da una visión completa: un modelo con alta sensibilidad podría ser poco preciso. Suele ser necesario equilibrarla con la

precisión para obtener un desempeño adecuado.

- **Puntuación F1 (F1-Score):**

  - *Definición y fórmula:* la puntuación F1 es la **media armónica** de la precisión y la sensibilidad. Combina ambas métricas en un solo valor, dando una medida balanceada del rendimiento en la clasificación de la clase positiva. Su fórmula es:

    F1=2 · Precisio´n×SensibilidadPrecisio´n+Sensibilidad=2 TP2 TP+FP+FN\text{F1} = 2 \cdot \frac{\text{Precisión} \times \text{Sensibilidad}}{\text{Precisión} + \text{Sensibilidad}} = \frac{2\,\text{TP}}{2\,\text{TP} + \text{FP} + \text{FN}}F1=2 · Precisio´n+SensibilidadPrecisio´n×Sensibilidad=2TP+FP+FN2 TP

    donde se observa que F1 toma en cuenta tanto falsos positivos como falsos negativos en el denominador. El resultado oscila entre 0 y 1; un modelo perfecto tendría F1 = 1, mientras que desempeños muy pobres se reflejan en valores cercanos a 0.

  - *Interpretación:* el F1-score **equilibra la calidad y la cobertura** de las predicciones positivas. Es útil para resumir en un solo número el compromiso entre precisión y sensibilidad de un modelo. Un F1 alto significa que el modelo logra simultáneamente una precisión y una sensibilidad altas. Por ejemplo, si un clasificador tiene Precisión = 0.80 y Sensibilidad = 0.80, su F1 = 0.80; pero si una de ellas es mucho más baja que la otra, F1 reflejará esa disparidad cayendo hacia el valor menor. De esta forma, **F1 será alto solo si tanto la precisión como la sensibilidad son altas y balanceadas entre sí**.

  - *Ventajas:* es una métrica **más informativa que la exactitud en escenarios desbalanceados**, ya que penaliza desequilibrios entre la capacidad de encontrar

positivos y la de evitar falsos alarmas. El F1 es **especialmente útil cuando una clase es minoritaria** o cuando nos interesa un rendimiento robusto en la clase positiva. Al combinar precisión y sensibilidad, es apropiado para comparar modelos en los que un modelo pudiera tener mejor precisión y otro mejor sensibilidad – F1 identifica cuál tiene el mejor equilibrio global.

- *Limitaciones:* asume igual importancia de falsos positivos y falsos negativos, lo cual podría no ser ideal si en el problema real un tipo de error es mucho más grave que el otro. Además, al ser un valor único, **no indica por separado qué tan precisas o exhaustivas son las predicciones** – es posible que dos modelos con el mismo F1 tengan perfiles de errores distintos (por ejemplo, uno con precisión alta y recall bajo, y otro al revés). Por último, F1 tampoco incluye a los verdaderos negativos en su cálculo, por lo que no refleja nada sobre el desempeño en la clase negativa. En resumen, aunque útil, **F1 pierde información detallada**, por lo que debe complementarse con la inspección de la matriz de confusión u otras métricas según el caso.

- **Curva ROC y AUC (ROC-AUC):**

  - *Definición:* la **curva ROC** (*Receiver Operating Characteristic*) es una representación gráfica del rendimiento de un clasificador binario al variar el umbral de decisión. La curva traza la **tasa de verdaderos positivos (TPR or Sensibilidad)** contra la **tasa de falsos positivos (FPR)** para todos los posibles umbrales de clasificación. Por su parte, **AUC** (*Area Under the Curve*) es el área bajo la curva ROC, resumida en un valor numérico entre 0 y 1. Cuanto más cerca de 1, mejor separa el modelo las dos clases en general.

  - *Interpretación:* el AUC puede interpretarse como la probabilidad de que el modelo asigne una puntuación más alta a un ejemplo positivo al azar que a un ejemplo negativo al azar. Un **AUC = 1.0** indica un modelo perfecto que siempre ordena por delante a los positivos sobre los negativos en probabilidad, mientras

que un **AUC = 0.5** corresponde a un modelo sin capacidad discriminativa (equivalente a adivinar al azar). Valores intermedios (p.ej. 0.8) indican que el modelo tiene un buen poder de discriminación global. A diferencia de métricas puntuales como la exactitud, la AUC refleja el desempeño promedio en **todos los umbrales**, proporcionando una visión más completa de la capacidad predictiva del modelo en diferentes condiciones de sensibilidad vs. precisión.

- *Ventajas:* ROC-AUC es muy informativa en problemas con datos desbalanceados, ya que **no depende de la proporción de clases** y evalúa la capacidad de ranking del modelo más que una tasa de error fija. Es útil para comparar modelos independientemente del umbral elegido para clasificar; dos modelos pueden tener igual exactitud en un umbral fijo, pero aquel con mayor AUC será superior globalmente porque ofrece mejor trade-off entre TPR y FPR en todos los umbrales. Además, la **curva ROC** permite visualizar claramente la relación entre verdaderos y falsos positivos, y escoger un umbral óptimo según el costo de cada error.

- *Limitaciones:* en escenarios con clases extremadamente desbalanceadas, la curva ROC puede dar una impresión demasiado optimista del modelo, ya que un pequeño aumento en verdaderos positivos puede no cambiar mucho el AUC si la clase negativa domina (en esos casos suele preferirse la curva Precision-Recall). Asimismo, la interpretación probabilística del AUC ("probabilidad de ranking correcto") puede no traducirse directamente a una métrica de error concreta en la aplicación. Por último, ROC-AUC **no informa sobre qué umbral específico usar**; un modelo con alta AUC podría requerir un umbral distinto a 0.5 para lograr la combinación deseada de precisión y recall. En resumen, es una métrica global, pero para decisiones operativas suele combinarse con otras métricas o análisis de umbral.

2.2. Métricas de regresión

Para modelos de **regresión** (que predicen valores numéricos continuos), se utilizan métricas de error que cuantifican cuán cerca están las predicciones de los valores reales. Dos de las métricas más comunes son el **Error Absoluto Medio (MAE)** y la **Raíz del Error Cuadrático Medio (RMSE)**. Ambas se basan en los *residuales* o errores individuales ($e_i = y_{\text{real},i} - y_{\text{pred},i}$) de las predicciones.

- **Error Absoluto Medio (MAE):**

    - *Definición y fórmula:* el MAE (por sus siglas en inglés *Mean Absolute Error*) es la media aritmética de los valores absolutos de los errores de predicción. Se calcula sumando las magnitudes de todos los errores y dividiéndolas entre el número de observaciones (N):

      MAE=1N∑i=1N∣yreal,i−ypred,i∣\text{MAE} = \frac{1}{N} \sum_{i=1}^{N} |\,y_{\text{real},i} - y_{\text{pred},i}\,|MAE=N1∑i=1N∣yreal,i−ypred,i∣

      Es decir, es el promedio de la distancia absoluta entre cada predicción del modelo y el valor real correspondiente..

    - *Interpretación:* el MAE indica en promedio **en cuánto se equivoca el modelo** en la misma unidad de la variable de salida. Por ejemplo, si se predicen precios de casas (en miles de dólares) y el MAE = 5, significa que, en promedio, las predicciones difieren del precio real en $5,000. Es una métrica fácil de interpretar: literalmente es el error promedio sin considerar si las predicciones se quedaron cortas o se excedieron (solo importa la magnitud del desvío).

    - *Ventajas:* es **sencilla e interpretable**, pues está en las mismas unidades del dato y comunica el error típico esperado. Además, **es menos sensible a valores atípicos** en comparación con métricas cuadráticas: al no elevar al cuadrado los errores, un error extremadamente grande impacta linealmente en el MAE en lugar de cuadráticamente. Esto la hace **más robusta ante outliers**, evitando que unos pocos errores muy grandes distorsionen excesivamente la métrica. También tiene

la ventaja de que trata todas las desviaciones por igual, lo cual es útil cuando queremos valorar por igual errores pequeños y grandes sin penalizar desproporcionadamente a estos últimos.

- *Limitaciones:* al no elevar al cuadrado las diferencias, **no penaliza fuertemente los errores grandes**. Si en ciertos problemas los errores grandes son mucho más críticos que los pequeños, el MAE podría no reflejar adecuadamente esa preocupación (un gran error queda "promediado" con los demás). En términos de optimización matemática, MAE es menos conveniente que el error cuadrático para algunas técnicas (su función no es diferenciable en 0), por lo que muchos algoritmos usan MSE como función de pérdida, aunque esto afecta más al entrenamiento que a la evaluación. Finalmente, al ser una métrica de escala dependiente (expresada en las unidades de la variable), **no permite comparar directamente el error entre modelos de distintas escalas de variable** (cada variable puede requerir normalizar o usar métricas relativas porcentuales si se desea comparación entre problemas).

- **Raíz del Error Cuadrático Medio (RMSE):**

  - *Definición y fórmula:* el RMSE (*Root Mean Squared Error*) es la raíz cuadrada del promedio del cuadrado de los errores. Se obtiene primero calculando el **Error Cuadrático Medio (MSE)** y luego tomando su raíz:

    RMSE=1N∑i=1N(yreal,i−ypred,i)2 .\text{RMSE} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} \left(y_{\text{real},i} - y_{\text{pred},i}\right)^2 } \,.RMSE=N1∑i=1N(yreal,i−ypred,i)2.

    Equivalente a $(\text{MSE})^{1/2}$. Al igual que el MAE, sus unidades son las mismas que la variable objetivo (por la raíz cuadrada).

- *Interpretación:* el RMSE puede interpretarse como una especie de **desviación estándar de los errores** de predicción. Un RMSE = 5 (en miles de dólares, siguiendo el ejemplo anterior) indicaría que los errores típicos del modelo están en un rango de ±$5,000, dando más peso a los casos con error grande. Debido al cuadrado, los errores más grandes influyen más: por tanto **mientras más alto el RMSE, significa que existen algunos errores significativos**. Es una métrica muy utilizada para comparar modelos, ya que resume tanto la cantidad promedio de error como la variabilidad del mismo.

- *Ventajas:* a diferencia del MAE, el RMSE **penaliza fuertemente los errores grandes** (por el término al cuadrado). Esto es beneficioso cuando los outliers o grandes desviaciones son especialmente indeseables: el modelo es evaluado más estrictamente si comete pocos errores pero muy graves. Asimismo, al devolver la medida en la unidad original (tras la raíz cuadrada), conserva cierta interpretabilidad directa (aunque ligeramente menos intuitiva que MAE) y facilita comparaciones de desempeño. El RMSE es de uso casi estándar en muchos campos (ej. en optimización de modelos se suele minimizar el RMSE), y matemáticamente es conveniente (deriva del MSE, que es diferenciable y convexo).

- *Limitaciones:* **sensible a valores atípicos** – un solo error muy grande puede incrementar sustancialmente el RMSE debido al efecto cuadrático. Si los datos tienen outliers que no representan el desempeño típico, el RMSE puede sobredimensionar la impresión de mal rendimiento. Además, no es tan fácil de interpretar como el MAE en términos cotidianos; aunque está en la misma escala, la raíz cuadrática implica que no es un promedio lineal simple. Por ejemplo, un RMSE ligeramente mayor que el MAE sugiere presencia de algunos errores grandes. En resumen, **RMSE enfatiza los errores grandes**, lo que puede ser una ventaja o desventaja según el objetivo: si nos interesa medir variabilidad de error es útil, pero si queremos una métrica más “democrática” con todos los errores, preferiríamos MAE. Por último, al igual que MAE, es una métrica dependiente de

escala y no directamente comparable entre problemas con unidades distintas.

Comparativa MAE vs RMSE: En general, ambas métricas se complementan. El MAE nos habla del error medio típico y es fácil de interpretar, mientras que el RMSE nos informa si existen errores grandes (al ser mayor que el MAE cuando hay alta dispersión). Si el RMSE es significativamente mayor que el MAE, implica que la distribución de errores tiene una cola larga (algunos errores bastante grandes elevan el RMSE). En cambio, cuando RMSE ≈ MAE, los errores tienden a tener magnitudes similares sin outliers prominentes. La elección entre una y otra depende del contexto: **MAE** se prefiere si queremos una medida más robusta a outliers y fácilmente explicable (error promedio), mientras que **RMSE** se prefiere si queremos castigar duramente los errores grandes y tener una métrica relacionada con la varianza de los errores. En muchos reportes se informan ambas para tener un panorama completo del rendimiento de un modelo de regresión.

3. Solución con KNN

A continuación, se detalla el proceso de solución del caso práctico utilizando un **clasificador K-Nearest Neighbors (KNN)**. Este algoritmo de aprendizaje supervisado clasifica una nueva instancia según las clases de sus *k* vecinos más cercanos en el espacio de características. El flujo general consistió en preparar los datos, entrenar varios modelos KNN con diferentes valores de *k*, evaluar sus desempeños con las métricas investigadas, y finalmente analizar los resultados obtenidos.

3.1. Preparación de los datos

División entrenamiento/prueba: Se partió de la matriz de datos proporcionada (Matriz.csv), la cual contiene 30 registros con las columnas **glucosa**, **edad** y **etiqueta** (clase binaria 0/1). Primero, se dividió el conjunto en dos subconjuntos: un **conjunto de entrenamiento** con el 70% de los datos (21 instancias) y un **conjunto de prueba** con el 30% restante (9 instancias). Esta separación se hizo de forma estratificada, es decir, manteniendo la proporción de etiquetas 0 y 1 similar en ambos grupos, para asegurar que la muestra de prueba sea representativa del problema. El propósito de esta división es entrenar el modelo en una porción de los datos y luego

evaluar su rendimiento en datos no vistos durante el entrenamiento (prueba), estimando así su capacidad de generalización.

Escalado de variables: Dado que KNN se basa en distancias (normalmente euclídeas) en el espacio de características, es importante que las variables estén en escalas comparables. En nuestro caso, la glucosa tiene valores numéricos (e.g. 85, 158, 183, etc.) y la edad también (21, 30, 65, etc.), pero sus rangos podrían diferir. Para evitar que una variable domine la distancia solo por su magnitud, se aplicó una **normalización estandarizada** a ambas características: se restó la media y dividió por la desviación estándar de cada variable (*Standardization*). De esta forma, glucosa y edad quedaron reescaladas con media 0 y desviación 1 en el conjunto de entrenamiento, aplicando luego la misma transformación al conjunto de prueba. Esto garantiza que **ambas características contribuyan equitativamente** al cálculo de los vecinos más cercanos, mejorando la eficacia del algoritmo KNN.

3.2. Entrenamiento del modelo

Se entrenaron clasificadores KNN con diferentes valores de k para encontrar el más adecuado. El hiperparámetro k (número de vecinos) controla la complejidad del modelo: valores bajos de k hacen el modelo más flexible pero potencialmente más susceptible a ruido (**overfitting**), mientras que valores altos de k lo vuelven más genérico pero podrían subestimar estructuras locales en los datos (**underfitting**).

En este experimento, se probaron **tres valores**: $k = 3$, $k = 5$ y $k = 7$. Para cada valor, se llevó a cabo el siguiente procedimiento:

- Se entrenó el modelo KNN usando el conjunto de entrenamiento (21 instancias). Entrenar un KNN esencialmente consiste en almacenar el conjunto de entrenamiento en memoria, ya que este algoritmo es de tipo *lazy learning* (no construye un modelo interno complejo, simplemente difiere el trabajo al momento de predecir).

- Se evaluó el modelo resultante sobre el conjunto de **prueba** (9 instancias) obteniendo las predicciones de clase para cada ejemplo de prueba.

- Se calculó la métrica **F1-score** de esas predicciones, considerando la clase positiva como la de etiqueta = 1. Se eligió F1 para la selección de modelo por ser una métrica balanceada entre precisión y sensibilidad, lo cual es valioso dado que ambas clases (0 y 1) están igualmente representadas en los datos. Además, el problema podría implicar costos tanto por falsos positivos como por falsos negativos, por lo que F1 ofrece un criterio combinatorio adecuado.

Tras entrenar y probar los 3 modelos, se compararon sus **F1-scores**. Los resultados (detallados en la siguiente sección) mostraron que los modelos con $k = 5$ y $k = 7$ obtuvieron el mejor desempeño (empate) en términos de F1, superior al modelo con $k = 3$. Se decidió seleccionar **k = 5** como el hiperparámetro óptimo para el modelo final, ya que ofreció un excelente compromiso y por simplicidad del modelo (menos vecinos a considerar reduce ligeramente el costo de cómputo frente a $k = 7$, además de que comúnmente se prefiere un k impar para evitar empates en votación de clases).

En resumen, el modelo final elegido fue **KNN con k = 5 vecinos**. Vale mencionar que, en una aplicación real, podría ser conveniente usar un conjunto de **validación cruzada** para escoger k de forma más robusta, dada la baja cantidad de datos. Sin embargo, para este caso ilustrativo, la selección basada en el conjunto de prueba es suficiente para comparar los resultados de los distintos k.

3.3. Evaluación del modelo

Con el modelo KNN (k=5) entrenado, se procedió a evaluar su rendimiento sobre el conjunto de prueba utilizando las métricas de clasificación estudiadas en la Sección 2. En concreto, se calcularon las siguientes métricas con respecto a las predicciones de prueba: **Exactitud**, **Precisión**, **Sensibilidad (Recall)**, **Puntuación F1** y **ROC-AUC**. Además, se generó la **Matriz de Confusión** para visualizar el desglose de aciertos y errores, y se trazó la **Curva ROC** correspondiente junto con el valor AUC.

El proceso de evaluación fue el siguiente:

- Se usó el modelo entrenado para predecir la etiqueta de cada instancia en el conjunto de prueba. Dado que KNN es un método determinista (para un *k* fijo), las predicciones de clase son reproducibles.

- Con las etiquetas predichas y las verdaderas, se construyó la **matriz de confusión** de 2x2. En esta matriz, las filas corresponden a la clase real (0 = Negativo, 1 = Positivo) y las columnas a las predicciones del modelo. A partir de la matriz, se identificaron los conteos de TP, TN, FP, FN.

- Usando los valores de TP, TN, FP, FN, se calcularon:

    - **Exactitud = (TP+TN)/(TP+TN+FP+FN)**,

    - **Precisión = TP/(TP+FP)**,

    - **Sensibilidad = TP/(TP+FN)**,

    - **F1 =** media armónica de precisión y sensibilidad (fórmula dada previamente).

- Se obtuvo el **puntaje AUC** calculando primero las probabilidades estimadas de clase positiva para cada instancia de prueba (en KNN, la probabilidad puede derivarse de la proporción de vecinos positivos entre los k vecinos, o equivalentemente la decisión de voto con algún peso si se usa). Luego, con las puntuaciones continuas, se integró el área bajo la curva ROC.

- Se generó la **gráfica de la curva ROC**, marcando el punto correspondiente al umbral usado (0.5 para decisión final) y visualizando la forma de la curva. Se indicó el valor AUC en la leyenda de la gráfica.

- Finalmente, se compararon los resultados de las métricas para los diferentes valores de *k* explorados, para tener una visión completa del impacto de *k* en el rendimiento.

Toda esta evaluación se realizó utilizando librerías estándar (*scikit-learn*) para asegurar cálculos correctos y evitar errores manuales. A continuación, en la sección de Resultados, se presentan los valores obtenidos de las métricas y las visualizaciones mencionadas, junto con la discusión correspondiente.

4. Resultados

A continuación, se resumen los resultados obtenidos del experimento con KNN. En la **Tabla 1** se presentan las métricas de desempeño en el conjunto de prueba para cada valor de *k* evaluado (3, 5 y 7). Posteriormente, se muestra la matriz de confusión y la curva ROC del mejor modelo seleccionado (*k = 5*).

Tabla 1. *Desempeño del modelo KNN en prueba para distintos valores de k.* Se reportan la Exactitud, Precisión, Sensibilidad y F1-score para cada valor. El mejor resultado en F1 (métrica de selección) se obtuvo con *k = 5* (empatado con k = 7).

k	Exactitud	Precisión	Sensibilidad	F1-score
3	0.778	0.75	0.75	0.750
5	0.889	1.00	0.75	0.857
7	0.889	1.00	0.75	0.857

Del cuadro anterior se desprenden varias observaciones: con *k = 3* el modelo obtuvo un F1 de 0.75, mientras que al aumentar a *k = 5* el F1 mejoró a ~0.857 (precisión perfecta de 1.0 y

sensibilidad 0.75). Incrementar a $k = 7$ no cambió el rendimiento, manteniendo F1 en 0.857. Dado que $k = 5$ y $k = 7$ rindieron igual según estas métricas, se eligió $k = 5$ por simplicidad. Es interesante notar que la **precisión** para k=5 (y k=7) fue 1.0, indicando **cero falsos positivos** en la muestra de prueba, mientras que la **sensibilidad** fue 0.75 (el modelo no logró encontrar 1 de cada 4 positivos). Esto sugiere que el clasificador KNN con k≥5 fue conservador al marcar positivos (de ahí su alta precisión), a costa de dejar escapar algunos (sensibilidad moderada). Con k=3, en cambio, tuvo un equilibrio exacto entre precisión y recall (ambos 0.75), pero inferior en precisión comparado con k=5. Estos patrones se entienden mejor examinando la matriz de confusión del modelo seleccionado.

Figura 1. Matriz de confusión del modelo KNN seleccionado ($k = 5$) en el conjunto de prueba. Las filas indican la clase verdadera (0 = Negativo, 1 = Positivo) y las columnas la predicción del modelo. Se observa que de 5 casos verdaderamente negativos, el modelo clasificó correctamente los 5 (TN = 5, FP = 0). De 4 casos verdaderamente positivos, el modelo identificó 3 correctamente (TP = 3) y cometió 1 falso negativo (FN = 1) clasificándolo como negativo. Esta matriz refleja la alta **precisión** (no hubo falsos positivos) y una **sensibilidad** del 75% (detectó 3 de 4 positivos) del modelo.

Figura 2. Curva ROC para el modelo KNN ($k = 5$) en el conjunto de prueba. La curva (línea naranja) muestra la relación entre TPR (sensibilidad) y FPR a medida que varía el umbral de decisión. En este caso, el modelo logró un **AUC = 1.00**, lo cual indica una separación perfecta de las clases en términos de ranking de probabilidad: es posible trazar un umbral donde el modelo obtiene TPR = 1 y FPR = 0. (La diagonal punteada negra representa el desempeño de un clasificador aleatorio, AUC = 0.5). Cabe aclarar que un AUC = 1 con sensibilidad < 1 en el punto de corte estándar sugiere que los ejemplos positivos obtuvieron sistemáticamente puntuaciones de probabilidad más altas que cualquier ejemplo negativo, permitiendo una separación teórica perfecta con otro umbral. En resumen, la curva ROC refleja un excelente desempeño general del modelo, consistente con las métricas calculadas.

Los resultados cuantitativos y visuales confirman que el modelo KNN con $k=5$ tuvo un rendimiento **muy bueno** en la muestra de prueba. Presentó una exactitud del ~88.9%, que en este caso refleja bien el rendimiento ya que las clases estaban balanceadas (recordemos, 15 positivos

y 15 negativos en total). Más importante, obtuvo **Precisión = 100%**, indicando que no hubo falsos positivos en prueba (cada paciente marcado como positivo realmente era positivo), y **Recall = 75%**, es decir, detectó 3 de cada 4 casos positivos. El **F1-score** de ~0.86 resume ese balance alto entre precisión y recall. Por último, el **AUC = 1.0** sugiere que las distribuciones de puntuaciones para las clases estaban completamente separadas (en efecto, al examinar las probabilidades devueltas por KNN, todos los ejemplos positivos recibieron puntuaciones mayores que todos los negativos, permitiendo esa área perfecta). Esto puede deberse a la naturaleza del conjunto de datos tan pequeño y limpio; en general, AUC = 1 es raro y podría indicar que el modelo "memoriza" los datos. En nuestro caso, KNN con pocos vecinos probablemente clasificó por vecindad inmediata haciendo que ningún negativo estuviera rodeado de positivos ni viceversa, resultando en una separación ideal.

En cuanto al efecto de *k*, la comparación indicó que con $k = 3$ hubo un caso de falso positivo (precisión < 1) y quizá algún patrón de clasificación más arriesgado. Al aumentar a $k = 5$, el modelo se volvió más conservador (ningún falso positivo) manteniendo buena sensibilidad, mejorando así F1. Con $k = 7$ no varió, lo que sugiere que a partir de 5 vecinos adicionales no aportaron mejora tangible en este dataset. Esto suele pasar en conjuntos pequeños: una vez que *k* alcanza un tamaño comparable a la clase minoritaria, puede haber un punto de saturación en beneficio.

5. Conclusiones

En este reporte se exploraron las **métricas de evaluación** más utilizadas tanto en clasificación como en regresión, entendiendo su significado y aplicabilidad. En particular, se destacó que en clasificación, métricas como la *precisión* y la *sensibilidad* ofrecen información más detallada que la exactitud, especialmente cuando las clases están desbalanceadas o ciertos errores importan más que otros. La *puntuación F1* se mostró valiosa para combinar precisión y recall en un solo indicador equilibrado, y la métrica *ROC-AUC* permitió evaluar el rendimiento global del clasificador independientemente del umbral. En regresión, se revisaron MAE y RMSE, notando cómo cada una aporta una perspectiva ligeramente distinta del error (promedio absoluto vs. penalización de errores grandes).

En la **solución con KNN** al problema planteado, el modelo logró un desempeño elevado. Con solo dos variables predictoras (glucosa y edad), el KNN ($k = 5$) alcanzó una exactitud cercana al 89% en la prueba, identificando correctamente la mayoría de los casos. El análisis detallado mostró que el modelo cometió **muy pocos errores**: no marcó como positivo ningún caso que no lo fuera (0 falsos positivos) y solamente se le escapó un caso positivo (1 falso negativo). Este comportamiento se reflejó en una precisión perfecta (100%) y una sensibilidad aceptable (75%). El F1-score, que balancea ambos aspectos, resultó alto (~0.86) indicando que el modelo mantiene un buen equilibrio. La curva ROC evidenció la excelente separación entre clases que logró el modelo en este conjunto, con un AUC de 1.0.

Comparando los valores de k, se pudo concluir que usar $k = 5$ *vecinos* fue mejor que $k = 3$ en términos de F1, ya que redujo los falsos positivos sin perder demasiados verdaderos positivos. Incrementar a $k = 7$ no mejoró más el rendimiento, por lo que $k = 5$ fue una elección adecuada. Esto sugiere que para este conjunto de datos relativamente limpio, considerar algo más de

contexto (5 vecinos) ayudó a tomar decisiones más estables que con muy pocos vecinos, pero demasiados vecinos diluyen la información local sin aportar beneficio adicional.

Como **posibles mejoras o trabajos futuros**, se proponen las siguientes ideas:

- **Validación más robusta:** Dado el tamaño pequeño del dataset (30 instancias), los resultados pueden variar si se cambia la partición. Sería recomendable utilizar *validación cruzada* (por ejemplo, 5-fold cross-validation) para estimar de forma más confiable el desempeño promedio del modelo y la mejor selección de *k*. Esto reduciría la varianza asociada a una sola separación entrenamiento/prueba.

- **Tuning de parámetros y variantes de KNN:** Además de probar valores adicionales de *k*, podría explorarse el uso de **pesos en los vecinos** (por ejemplo, dar más peso al voto de los vecinos más cercanos que a los más lejanos) o probar diferentes métricas de distancia (Euclídea vs Manhattan) para ver si mejora la performance. En este problema, dado que las escalas ya se normalizaron, la distancia euclídea estándar funcionó bien.

- **Más características predictoras:** Solo se usaron glucosa y edad. Si el contexto es predecir diabetes, típicamente intervienen otras variables (presión sanguínea, índice de masa corporal, etc.). Incluir más características relevantes podría mejorar la capacidad predictiva del modelo. Sin embargo, habría que tener cuidado de no agregar atributos irrelevantes, ya que KNN sufre la *maldición de la dimensionalidad* (muchas dimensiones pueden diluir la noción de vecindad). Una opción intermedia es aplicar técnicas de **selección de características** para quedarse solo con las más informativas.

- **Balance de precisión vs. recall:** En nuestro resultado, obtuvimos precisión máxima pero a costa de una recall moderada. Dependiendo del objetivo, podría interesar aumentar la sensibilidad incluso si cae algo la precisión. Con KNN, esto podría lograrse disminuyendo *k* ligeramente (haciendo el modelo más flexible para captar positivos dudosos) o ajustando un **umbral de decisión** distinto a la mayoría simple (por ejemplo, exigir solo 2 de 5 vecinos positivos para clasificar como positivo en lugar de >2). También se podría considerar un método de ensamble o ajustar manualmente la

propensión a predecir la clase positiva (p. ej., a través de *threshold moving*).

- **Comparación con otros modelos:** Si bien KNN funcionó muy bien en este dataset, conviene probar otros clasificadores más avanzados (como un árbol de decisión, un modelo de regresión logística, o métodos basados en *boosting*). Estos podrían ofrecer ventajas, por ejemplo, en interpretabilidad (árboles) o en rendimiento con datos más grandes. Evaluar múltiples modelos con las métricas estudiadas daría una perspectiva más amplia de cuál es la mejor alternativa para este problema.

En conclusión, el ejercicio mostró la importancia de elegir métricas de evaluación adecuadas para interpretar correctamente el desempeño de un modelo. Aplicando KNN, se obtuvo un clasificador eficaz para distinguir las dos clases con los datos dados. Las métricas calculadas – exactitud, precisión, sensibilidad, F1 y AUC – brindaron juntas una comprensión completa de los aciertos y errores del modelo. Al comparar diferentes configuraciones (*k* vecinos) mediante el F1-score, se seleccionó un modelo bien equilibrado. Las visualizaciones (matriz de confusión y curva ROC) complementaron este análisis. Como recomendación general, siempre es útil observar múltiples métricas en conjunto, ya que cada una revela un aspecto del rendimiento, y ajustar el modelo según las prioridades del problema (ya sea minimizar falsos positivos, maximizar detección de positivos, etc.). Con un enfoque iterativo de evaluación y mejora, se puede lograr un modelo final robusto y alineado con los objetivos del proyecto.

6. Referencias

- Friedman, Y. (2024). *Understanding F1 Score, Accuracy, ROC-AUC, and PR-AUC Metrics for Models*. **Deepchecks Blog** (13 de junio de 2024)deepchecks.comdeepchecks.com. *(Explica las definiciones de exactitud, precisión, recall, F1 y AUC, junto con ejemplos y advertencias sobre desequilibrio de clases).*

- Google Developers. (s.f.). *Clasificación: Exactitud, recuperación, precisión y métricas relacionadas*. **Curso intensivo de AA de Google**developers.google.comdevelopers.google.com. *(Documento en español del Machine Learning Crash Course de Google que define formalmente las métricas de clasificación y brinda recomendaciones de uso para cada una.)*

- Sabbha, M. (2024). *Understanding MAE, MSE, and RMSE: Key Metrics in Machine Learning*. **DEV Community** (16 de agosto de 2024)dev.todev.to. *(Artículo que describe*

MAE, MSE y RMSE con fórmulas, ejemplos de cálculo y discusión sobre cuándo usar cada métrica.)

- Ultralytics. (2025). *Área Bajo la Curva (AUC)* – Explicación del Glosario Ultralyticsultralytics.comultralytics.com. *(Entrada de glosario que describe en español la métrica AUC, su relación con la curva ROC y la interpretación de sus valores extremos.)*

- Codificando Bits. (2021). *Precision, recall y F-score para la clasificación binaria*codificandobits.comcodificandobits.com. *(Blog educativo en español que discute las limitaciones de la exactitud en conjuntos desbalanceados y presenta las métricas precision, recall y F con fórmulas y ejemplos prácticos.)*

(Se incluyen enlaces a las fuentes para ver detalles adicionales. Las referencias están formateadas en estilo APA básico.)

7. Anexos

A. Código en Python para entrenamiento y evaluación

A continuación se muestra un extracto del código utilizado para dividir los datos, entrenar el modelo KNN y calcular las métricas y gráficos presentados en el reporte. Este código fue implementado en Python empleando la biblioteca scikit-learn para las operaciones de modelado y métricas.

```
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

from sklearn.metrics import confusion_matrix, roc_auc_score, RocCurveDisplay

import matplotlib.pyplot as plt


# 1. Cargar los datos
```

```
df = pd.read_csv('Matriz.csv')

X = df[['glucosa', 'edad']]

y = df['etiqueta']


# 2. Dividir en entrenamiento (70%) y prueba (30%), estratificado por etiqueta

X_train, X_test, y_train, y_test = train_test_split(

    X, y, test_size=0.3, stratify=y, random_state=42

)


# 3. Escalar características (normalización estándar)

scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)

X_test_scaled = scaler.transform(X_test)


# 4. Entrenar KNN con diferentes k y evaluar F1-score en prueba

k_values = [3, 5, 7]

best_k = None

best_f1 = -1

for k in k_values:
```

```
    model = KNeighborsClassifier(n_neighbors=k)
    model.fit(X_train_scaled, y_train)
    y_pred = model.predict(X_test_scaled)
    f1 = f1_score(y_test, y_pred)
    print(f"k={k}, F1-score={f1:.3f}")
    if f1 > best_f1:
        best_f1 = f1
        best_k = k

print(f"Mejor k según F1-score: {best_k} (F1={best_f1:.3f})")

# 5. Entrenar modelo final con k óptimo
best_model = KNeighborsClassifier(n_neighbors=best_k)
best_model.fit(X_train_scaled, y_train)
y_pred = best_model.predict(X_test_scaled)

# 6. Calcular métricas de clasificación
acc = accuracy_score(y_test, y_pred)
prec = precision_score(y_test, y_pred)
rec = recall_score(y_test, y_pred)
```

```
f1 = f1_score(y_test, y_pred)

print(f"Accuracy={acc:.3f},                Precisión={prec:.3f},
Recall={rec:.3f}, F1={f1:.3f}")


# 7. Matriz de confusión

cm = confusion_matrix(y_test, y_pred)

print("Matriz de confusión:")

print(cm)


# 8. Curva ROC y AUC

y_scores = best_model.predict_proba(X_test_scaled)[:, 1]

auc_value = roc_auc_score(y_test, y_scores)

print(f"ROC AUC = {auc_value:.3f}")


RocCurveDisplay.from_estimator(best_model,         X_test_scaled,
y_test)

plt.plot([0, 1], [0, 1], 'k--', label="Clasificador aleatorio")

plt.title(f"Curva ROC (modelo KNN k={best_k})")

plt.legend(loc="lower right")

plt.show()
```

En la salida impresa por este script se obtuvieron los valores reportados en la sección de Resultados (por ejemplo, *k=5 con F1 ≈ 0.857*, etc.). La gráfica de la curva ROC y la matriz de confusión se generaron usando las funciones de scikit-learn; dichas figuras se incluyeron en el reporte como la Figura 1 y 2. El código puede adaptarse fácilmente para probar otros valores de *k* o para realizar validación cruzada si se contara con más datos. Asimismo, la semilla (random_state=42) se fijó para garantizar reproducibilidad en la división de datos.