UNIVERSIDAD TECNOLÓGICA DE CHIHUAHUA

DESARROLLO Y GESTIÓN DE SOFTWARE

Extracción de Conocimiento en Bases de Datos

III.2. Reporte de Métricas de Evaluación

Docente:

Enrique Mascote

Presentan:

Ian Carlos Chávez Rojo

Grupo:

IDGS91N

Fecha: 28/11/2025

Índice

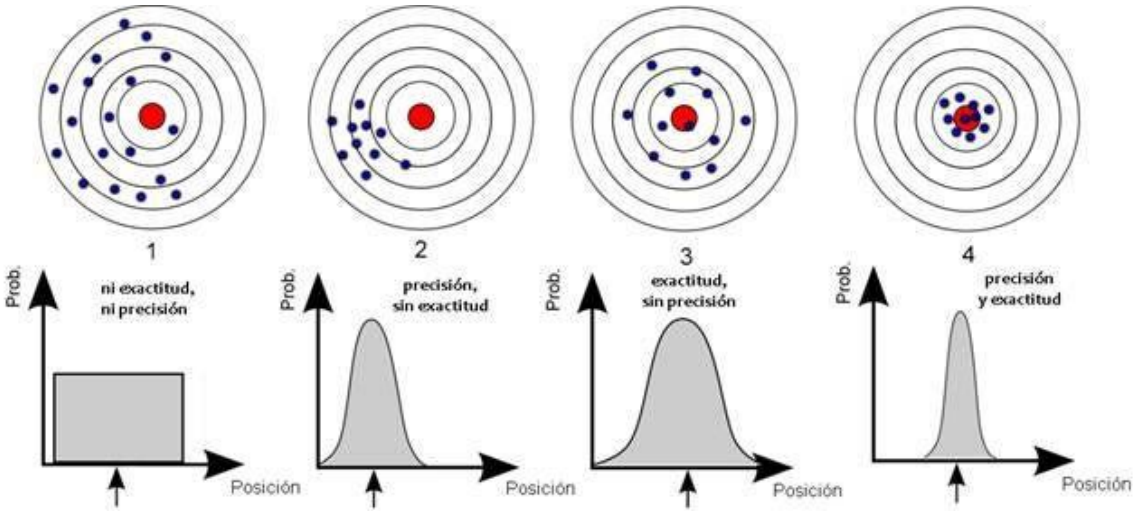
Introducción

En el contexto del aprendizaje supervisado, la evaluación del desempeño de los modelos predictivos es fundamental para garantizar su efectividad en la solución de problemas reales. Este reporte tiene como objetivo analizar e interpretar diversas métricas de evaluación aplicadas a modelos de clasificación y regresión, con especial énfasis en la comprensión de su utilidad práctica, fortalezas y limitaciones. A través del desarrollo de un caso aplicado con el algoritmo K-Nearest Neighbors (KNN), se busca poner en práctica los conceptos teóricos mediante la construcción, prueba y análisis de un modelo de clasificación binaria, utilizando variables como la glucosa y la edad. Además, se explorarán visualizaciones y comparativas entre diferentes configuraciones del modelo, justificando la selección óptima con base en evidencia cuantitativa y criterios técnicos de evaluación.

Este análisis se enmarca dentro del estudio de técnicas de minería de datos aplicadas a contextos reales, con el objetivo de fortalecer la capacidad crítica en la selección de modelos y la interpretación de resultados. A través del enfoque experimental con el algoritmo KNN, se pretende no solo medir el rendimiento numérico del modelo, sino también reflexionar sobre su aplicabilidad, limitaciones y posibles mejoras. La combinación de teoría, práctica y visualización permite una comprensión más profunda de cómo las métricas de evaluación influyen directamente en la toma de decisiones basadas en datos.

Métricas de Evaluación

Las métricas de evaluación son herramientas estadísticas que nos permiten medir el rendimiento de un modelo predictivo, y varían según el tipo de problema. En modelos de regresión, que predicen valores numéricos continuos (como el precio de una casa), se utilizan métricas como el Error Absoluto Medio (MAE), el Error Cuadrático Medio (MSE) y la Raíz del Error Cuadrático Medio (RMSE), las cuales cuantifican qué tan lejos están las predicciones de los valores reales. En cambio, los modelos de clasificación, que predicen categorías (como si un correo es spam o no), se evalúan con métricas como precisión, recall (sensibilidad), exactitud y el F1-score, que ayudan a entender cuántas predicciones fueron correctas y cuántas fallaron, considerando diferentes aspectos del rendimiento. Estas métricas permiten comparar modelos, afinar sus parámetros y tomar decisiones informadas para mejorar su desempeño.



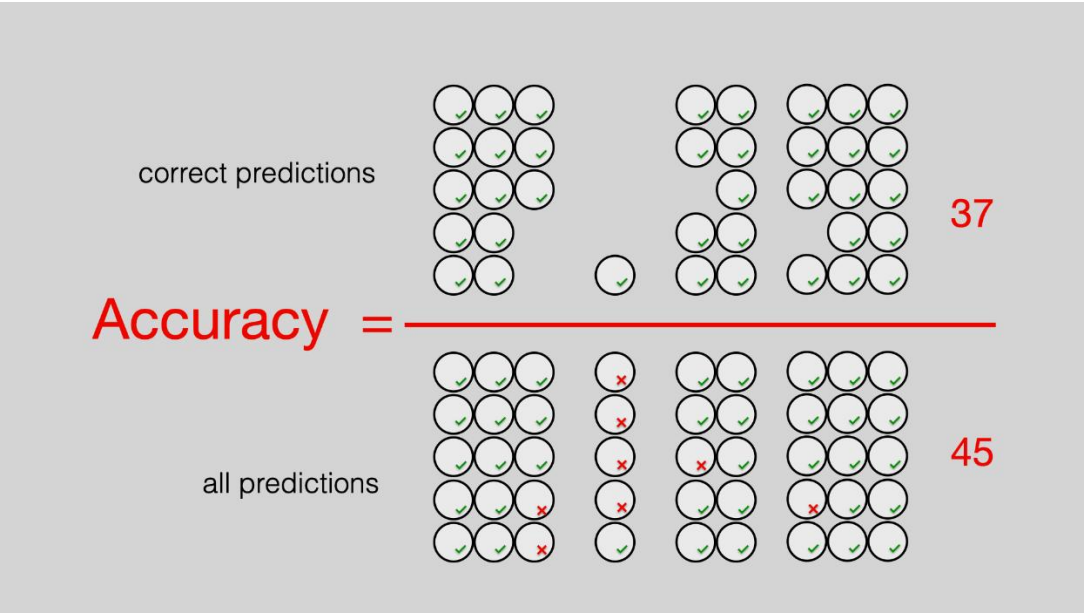
Sección 1 – Investigación de métricas

En esta sección se explorarán cuatro métricas clave para evaluar modelos de clasificación y dos métricas fundamentales para modelos de regresión, con el objetivo de comprender su significado, fórmula matemática, interpretación práctica, ventajas y limitaciones. Para clasificación se investigarán: accuracy (precisión general, fácil de calcular), precision (precisión positiva, ampliamente documentada), recall (sensibilidad, de fácil interpretación) y F1-score (promedio balanceado entre precision y recall, comúnmente usado en contextos prácticos). En el caso de regresión, se revisarán: MAE (Error Absoluto Medio, simple y directo) y RMSE (Raíz del Error Cuadrático Medio, muy utilizado y con abundante información). Estas métricas permitirán evaluar y comparar el desempeño de distintos modelos, facilitando la elección de la opción más adecuada para el problema en cuestión.

Métricas de evaluación de modelos de clasificación

Acurracy

Accuracy es una métrica de evaluación utilizada en modelos de clasificación que refleja la proporción de predicciones correctas respecto al total de predicciones realizadas. En otras palabras, indica qué tan frecuentemente el modelo acierta en sus predicciones, considerando tanto los casos positivos como los negativos correctamente identificados.



Fórmula matemática

La fórmula de Accuracy permite cuantificar el rendimiento global de un modelo de clasificación binaria al relacionar el número de aciertos con el total de predicciones realizadas. Se calcula sumando las predicciones correctas (positivas y negativas) y dividiendo ese resultado entre el número total de predicciones, lo que da como resultado un valor entre 0 y 1 (o 0 % a 100 %), donde valores cercanos a 1 indican un buen desempeño global del modelo.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Donde:

- **TP** = Verdaderos Positivos.
- **TN** = Verdaderos Negativos.
- **FP** = Falsos Positivos.
- **FN** = Falsos Negativos.

Interpretación práctica

En la práctica, Accuracy indica qué tan frecuentemente el modelo acierta en sus predicciones, evaluando tanto los aciertos en casos positivos como negativos. Es especialmente útil cuando las clases están balanceadas, ya que resume en un solo valor el porcentaje de instancias correctamente clasificadas. Por ejemplo, si un modelo tiene una accuracy del 90 %, significa que acertó en 9 de cada 10 casos. Sin embargo, en situaciones con clases desbalanceadas, este valor puede resultar engañoso, ya que un modelo podría tener alta accuracy simplemente por predecir siempre la clase mayoritaria sin identificar correctamente los casos menos frecuentes.

Ventajas

Una de las principales fortalezas de Accuracy es su simplicidad y capacidad de ofrecer una visión general del rendimiento de un modelo de clasificación. Es especialmente útil en contextos donde las clases están equilibradas, ya que resume de forma directa la proporción de predicciones correctas sobre el total.

- **Fácil de interpretar**: proporciona una medida rápida y clara del porcentaje de aciertos del modelo.
- **Cálculo sencillo**: su fórmula es directa y no requiere operaciones complejas ni conocimiento avanzado.
- **Útil para clases balanceadas**: cuando las clases tienen una distribución equitativa, refleja adecuadamente el desempeño global.
- **Buena primera aproximación**: sirve como punto de partida para comparar múltiples modelos de forma general.

Limitaciones

Aunque Accuracy es una métrica muy utilizada por su simplicidad, presenta importantes limitaciones, especialmente cuando se aplica en contextos donde las clases están desbalanceadas. En estos casos, puede ofrecer una visión distorsionada del rendimiento real del modelo, haciendo que parezca más preciso de lo que realmente es. Por ello, su uso debe ir acompañado de otras métricas que consideren de forma más detallada los errores cometidos.

- **Engañosa con clases desbalanceadas**: puede mostrar valores altos, aunque el modelo falle en predecir la clase minoritaria.
- **No distingue tipos de error**: trata igual los falsos positivos y falsos negativos, sin considerar su impacto específico
- **No refleja bien el rendimiento en problemas críticos**: en tareas como detección de fraude o enfermedades raras, una alta accuracy puede ocultar resultados deficientes.
- **Menos útil en modelos multicategoría complejos**: pierde interpretabilidad cuando hay múltiples clases con distribución desigual.

Precision

Precision es una métrica de evaluación en modelos de clasificación que mide la proporción de verdaderos positivos entre todas las instancias que el modelo ha

clasificado como positivas. En otras palabras, indica qué tan confiables son las predicciones positivas del modelo: de todos los casos que se predijeron como positivos, cuántos realmente lo eran. Es especialmente relevante en situaciones donde los falsos positivos tienen un alto costo, como en diagnósticos médicos o detección de fraudes.

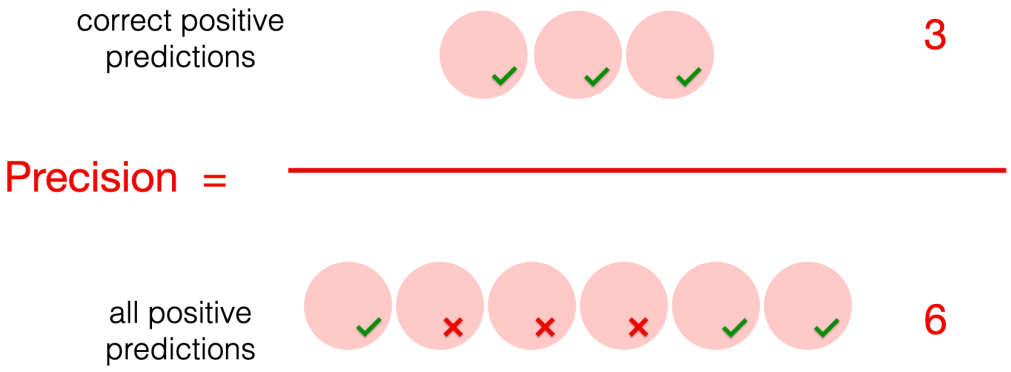
Fórmula matemática

La fórmula de Precision se centra en medir qué proporción de las instancias que el modelo clasificó como positivas son realmente correctas. Es decir, se evalúa cuántas de las predicciones positivas fueron verdaderamente acertadas, lo cual resulta especialmente útil cuando es importante minimizar los falsos positivos.

$$Precision = \frac{TP}{TP + FP}$$

Donde:

- **TP** = Verdaderos Positivos (instancias positivas correctamente clasificadas).
- **FP** = Falsos Positivos (instancias negativas incorrectamente clasificadas como positivas).



Interpretación práctica

En términos prácticos, Precision indica cuán confiables son las predicciones positivas de un modelo de clasificación. Es decir, mide cuántos de los casos que el modelo identificó como positivos realmente lo eran. Esta métrica es especialmente importante cuando el costo de un falso positivo es alto; por ejemplo, en la detección de fraudes bancarios, donde marcar erróneamente una transacción legítima como fraudulenta puede causar molestias al usuario. Un modelo con alta precision comete pocos errores al etiquetar instancias como positivas, aunque podría dejar pasar algunos casos que sí lo son (para eso se analiza también el recall). Por ello, precision es muy útil para evaluar qué tan “estricto” es el modelo al declarar un resultado positivo.

Ventajas

Una de las grandes fortalezas de Precision es su utilidad en escenarios donde los falsos positivos son especialmente indeseables. Al enfocarse únicamente en las instancias que el modelo predice como positivas, permite evaluar con precisión qué tan confiables son esas predicciones. Esto la convierte en una herramienta valiosa en contextos como diagnósticos médicos, control de calidad o detección de fraudes, donde es preferible evitar errores de “falsas alarmas”.

- **Alta relevancia en contextos críticos**: ideal cuando los falsos positivos pueden generar consecuencias graves.
- **Se enfoca en las predicciones positivas**: permite evaluar qué tan confiables son los casos que el modelo declara como positivos.
- **Complementaria a otras métricas**: junto con recall y F1-score, ofrece una visión equilibrada del desempeño del modelo.
- **Útil en clases desbalanceadas**: ayuda a evaluar el rendimiento cuando la clase positiva es minoritaria.

Limitaciones

A pesar de su utilidad, Precision también presenta ciertas limitaciones que es importante considerar al evaluar modelos de clasificación. Esta métrica se enfoca únicamente en las predicciones positivas, por lo que puede ofrecer una visión incompleta del rendimiento general del modelo si no se analiza en conjunto con otras métricas como el recall. Además, un modelo puede tener una precisión alta simplemente porque predice pocos positivos, lo que a veces no es deseable dependiendo del objetivo del análisis.

- **Ignora los falsos negativos**: no considera los casos positivos que el modelo pasó por alto.
- **Puede ser alta con pocos positivos**: un modelo que predice solo algunos positivos puede tener alta precisión, pero baja cobertura.
- **No refleja el rendimiento global**: se enfoca en un aspecto específico (predicciones positivas), dejando fuera otras dimensiones del desempeño.
- **Requiere complemento con otras métricas**: para una evaluación completa, debe analizarse junto con recall y F1-score.

Recall

También conocido como sensibilidad o tasa de verdaderos positivos, es una métrica utilizada en modelos de clasificación que mide la capacidad del modelo para identificar correctamente todas las instancias positivas reales. En otras palabras, indica qué proporción de los casos que verdaderamente pertenecen a la clase positiva fueron efectivamente detectados por el modelo. Es especialmente importante en contextos donde pasar por alto un caso positivo puede tener consecuencias críticas.

Fórmula matemática

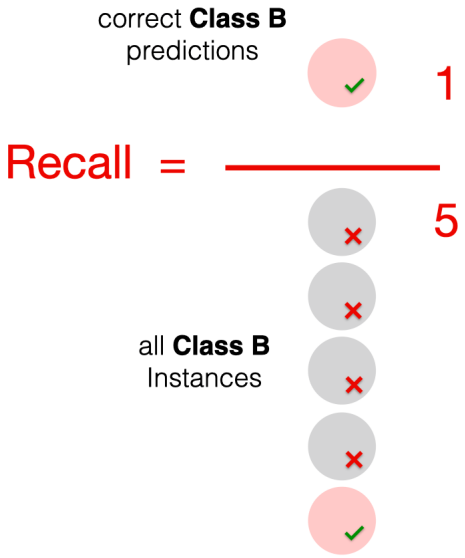
La fórmula de Recall permite medir qué tan efectivo es un modelo al identificar todas las instancias que realmente pertenecen a la clase positiva. Se enfoca en minimizar los falsos negativos, es decir, aquellos casos positivos que el modelo no logró

detectar, lo que es crucial en contextos donde omitir un positivo tiene consecuencias importantes.

$$Recall = \frac{TP}{TP + FN}$$

Donde:

- **TP** = Verdaderos Positivos (casos positivos correctamente identificados por el modelo)
- **FN** = Falsos Negativos (casos positivos que el modelo clasificó incorrectamente como negativos)



Interpretación práctica

En la práctica, Recall nos dice qué tan bien el modelo identifica todos los casos positivos reales dentro del conjunto de datos. Es decir, mide la capacidad del modelo para no dejar pasar instancias que realmente pertenecen a la clase positiva. Por ejemplo, en un sistema de diagnóstico médico, un alto recall significa que el modelo logra detectar la mayoría de los pacientes que sí tienen la enfermedad, aunque esto pueda implicar cometer algunos falsos positivos. Esta métrica es especialmente útil en contextos donde es más costoso omitir un caso positivo (falso negativo) que generar una falsa alarma.

Ventajas

Una de las fortalezas más destacadas de Recall es su capacidad para evaluar qué tan bien un modelo de clasificación logra detectar todos los casos verdaderamente positivos. Esta métrica resulta especialmente valiosa en escenarios donde pasar por alto una instancia positiva tiene un alto costo, ya que se enfoca en reducir los errores por omisión (falsos negativos).

- **Detecta la mayoría de los casos positivos**: ideal para problemas donde es vital identificar todas las instancias relevantes.
- **Esencial en contextos sensibles**: útil en campos como medicina, seguridad o fraudes, donde no detectar un caso puede ser crítico.
- **Complemento valioso para precision**: al usarse junto con precision, permite equilibrar el análisis del modelo.
- **Ayuda a mejorar la cobertura**: fomenta la detección de más casos positivos, incluso si eso implica algunos falsos positivos adicionales.

Limitaciones

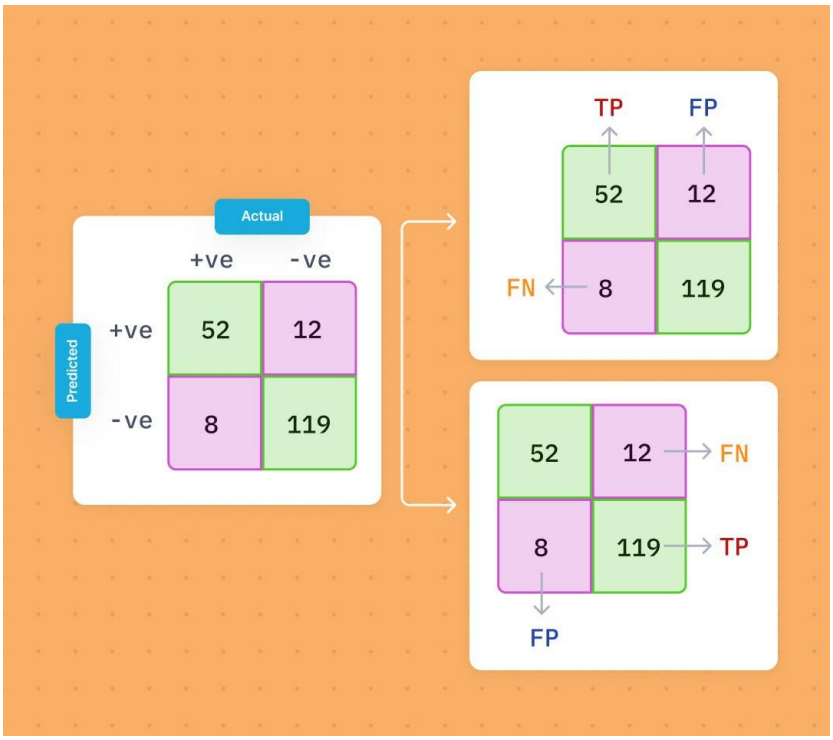
A pesar de ser muy útil en escenarios donde es crítico no omitir casos positivos, Recall también tiene algunas limitaciones. Al enfocarse únicamente en los verdaderos positivos y los falsos negativos, puede llevar a interpretar como favorable un modelo que simplemente clasifica muchos casos como positivos, aún cuando muchos de ellos sean incorrectos. Por eso, debe considerarse junto con otras métricas que equilibren su perspectiva.

- **Ignora los falsos positivos**: no evalúa cuántos casos negativos fueron mal clasificados como positivos.
- **Puede ser alto con baja precisión**: si el modelo predice muchos positivos, aumenta el recall, aunque cometa muchos errores.
- **No refleja el rendimiento general**: se centra solo en la detección de positivos, dejando de lado el comportamiento frente a negativos.

- **Riesgo de sobreajuste**: si el objetivo es maximizar el recall a toda costa, puede llevar a modelos que predicen todo como positivo.

F1-score

F1-score es una métrica de evaluación que combina en un solo valor el precision y el recall, representando su media armónica. Se utiliza principalmente cuando se busca un equilibrio entre ambas métricas, especialmente en escenarios donde se desea minimizar tanto los falsos positivos como los falsos negativos. Es particularmente útil en tareas como la moderación automática de contenidos o los sistemas de recomendación, donde es necesario identificar con precisión los elementos relevantes sin dejar fuera opciones valiosas ni incluir demasiadas irrelevantes. Al equilibrar precisión y cobertura, el F1-score brinda una visión más justa del rendimiento del modelo en contextos complejos.



Formula Matemática

La fórmula del F1-score combina precision y recall mediante su media armónica, lo cual penaliza más fuerte cuando una de las dos métricas es baja. Esto hace que el F1-score sea especialmente útil cuando se busca un balance entre ambos aspectos y no se quiere que uno compense demasiado al otro.

$$F1 - score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Esta expresión produce un valor entre 0 y 1 (o 0 % a 100 %), donde 1 indica que tanto la precisión como la cobertura son excelentes. Ideal cuando se necesita equilibrio en modelos con clases desbalanceadas o consecuencias variables para falsos positivos y falsos negativos.

Interpretación práctica

En la práctica, F1-score nos indica qué tan equilibrado es el modelo al hacer predicciones positivas correctas sin dejar de detectar casos relevantes. Es particularmente útil cuando se busca un compromiso entre precision y recall, como en sistemas de recomendación de productos o filtros automáticos de currículums, donde es importante sugerir opciones verdaderamente relevantes, pero también no omitir otras que podrían ser útiles. Un valor alto de F1-score sugiere que el modelo mantiene un buen balance entre identificar correctamente los casos positivos y evitar errores por exceso o por omisión.

Ventajas

El F1-score destaca por ser una métrica que proporciona un equilibrio entre precisión y cobertura, lo que la hace ideal para evaluar modelos cuando es importante considerar tanto los errores por exceso como por omisión. Su naturaleza armónica asegura que un bajo desempeño en una de las métricas no se vea disfrazado por un valor alto en la otra, brindando así una perspectiva más justa del rendimiento real del modelo.

- **Equilibra precision y recall**: útil cuando se desea evaluar el compromiso entre ambas métricas, especialmente si una no debe prevalecer sobre la otra.
- **Robusto ante clases desbalanceadas**: a diferencia de accuracy, ofrece una evaluación más realista cuando una clase domina en número.

- **Ideal para entornos exigentes**: muy valioso en tareas como moderación de contenido, detección de eventos raros o clasificación de textos múltiples.
- **Fácil de comparar entre modelos**: al condensar dos dimensiones clave en un solo valor, facilita la selección del mejor modelo bajo criterios de balance.

Limitaciones

Aunque el F1-score es una métrica muy útil para equilibrar precisión y cobertura, también presenta ciertos retos en su aplicación e interpretación. Su naturaleza armónica penaliza fuertemente los valores extremos, por lo que un valor bajo en precision o recall impacta significativamente el resultado final. Además, al reducir dos métricas a una sola cifra, puede ocultar detalles importantes del comportamiento del modelo en diferentes contextos.

- **No distingue entre precision y recall**: al combinarlas, no muestra cuál de las dos está afectando el rendimiento.
- **Sensibilidad a desequilibrio extremo**: puede ser poco representativo si las clases están altamente desbalanceadas.
- **Pérdida de interpretabilidad**: al condensar dos métricas en una, se puede dificultar el análisis detallado del modelo.
- **Puede necesitar contexto adicional**: requiere ser acompañado por la matriz de confusión u otras métricas para una evaluación más precisa.

Métricas de regresión

MAE (Error Absoluto Medio)

MAE (por sus siglas en inglés Mean Absolute Error) es una métrica utilizada para evaluar el rendimiento de modelos de regresión. Representa el promedio de los errores absolutos entre las predicciones del modelo y los valores reales observados. Es decir, mide cuánto se desvía, en promedio, el valor predicho del valor real, sin considerar si el error fue por exceso o por defecto. Al enfocarse en la magnitud del error y no en su dirección, MAE proporciona una medida clara y fácil de interpretar sobre la precisión general del modelo.

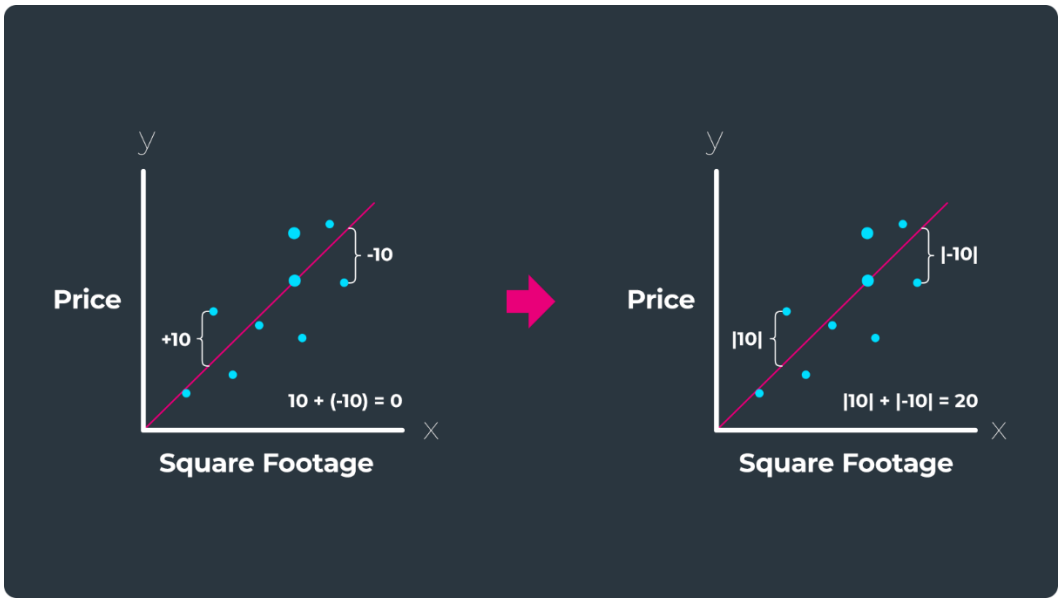
Fórmula matemática

La fórmula de MAE calcula el promedio de las diferencias absolutas entre los valores predichos por el modelo y los valores reales observados. Al utilizar valores absolutos, evita que los errores positivos y negativos se cancelen entre sí, proporcionando una medida clara de cuánto se equivoca, en promedio, el modelo al hacer sus predicciones.

$$MAE = \left(\frac{1}{N}\right) * \sum |yi - \hat{y}i|$$

Donde:

- **n** = número total de observaciones.
- **yi** = valor real de la observación i.
- **ŷi** = valor predicho por el modelo para la observación i.
- **|yi − ŷi|** = valor absoluto del error en cada predicción.



Interpretación práctica

En la práctica, MAE nos indica cuánto se equivoca, en promedio, un modelo de regresión al hacer sus predicciones. Por ejemplo, si se utiliza para predecir el precio de viviendas y el MAE es de $2,500, significa que el modelo tiende a errar por esa cantidad, ya sea por exceso o por defecto. Es una métrica fácil de entender y útil en

contextos donde todos los errores tienen un impacto similar, como en la estimación de tiempos de entrega o consumo energético, donde lo importante es minimizar la desviación promedio sin importar la dirección del error.

Ventajas

MAE se destaca por ser una métrica intuitiva y directa, lo que la hace ideal como punto de partida para evaluar modelos de regresión. Su enfoque en los errores absolutos permite obtener una medida clara y comprensible del desempeño general del modelo sin complicaciones matemáticas ni suposiciones sobre la distribución de los errores.

- **Fácil de interpretar**: expresa el error promedio en las mismas unidades que la variable objetivo (por ejemplo, pesos, días, grados).
- **Sensible a todas las desviaciones**: cada error contribuye por igual al resultado, sin amplificar los valores extremos.
- **Útil en contextos operativos**: funciona bien cuando todos los errores tienen el mismo costo, como en estimación de tiempos de entrega o precios de productos.
- **Menos susceptible a outliers que RMSE**: al no elevar al cuadrado los errores, no magnifica los valores atípicos como otras métricas.

Limitaciones

Aunque MAE es una métrica muy intuitiva y útil, también presenta ciertas limitaciones, especialmente cuando los errores grandes tienen un impacto importante. Al tratar todos los errores por igual, sin importar su magnitud, puede subestimar el efecto de outliers o valores extremadamente desviados, lo que puede ser problemático en escenarios donde los errores grandes son especialmente costosos.

- **No penaliza fuertemente los errores grandes**: a diferencia de métricas cuadráticas, no destaca la presencia de outliers.

- **Menor sensibilidad en distribuciones sesgadas**: puede no capturar bien el impacto de errores en colas extremas.
- **Puede ofrecer la misma media con distribuciones diferentes**: dos modelos con patrones de error distintos pueden tener el mismo MAE.
- **Menos útil cuando se desea detectar grandes desviaciones**: no enfatiza suficientemente los casos que más se alejan de lo esperado.

RMSE (Root Mean Squared Error o Raíz del Error Cuadrático Medio)

RMSE es una métrica de evaluación en modelos de regresión que mide la desviación promedio entre los valores predichos por el modelo y los valores reales, penalizando más fuertemente los errores grandes. Esto se logra al calcular primero el promedio de los errores al cuadrado, y luego tomar la raíz cuadrada de ese valor. A diferencia de MAE, RMSE otorga mayor peso a los errores grandes, por lo que es especialmente útil cuando se desea detectar desviaciones importantes o cuando los errores grandes son más costosos que los pequeños.

Fórmula matemática

La fórmula de RMSE calcula la raíz cuadrada del promedio de los errores al cuadrado. Al elevar los errores al cuadrado, esta métrica penaliza de forma más severa aquellos errores grandes, lo que la hace especialmente útil cuando deseas detectar desviaciones significativas entre lo que predice el modelo y lo que ocurre en la realidad.

$$RMSE = \sqrt{(\left(\frac{1}{n}\right) * \sum (yi - \hat{y}i)^2)}$$

Donde:

- **n** = número total de observaciones.
- **yi** = valor real observado en la muestra i.
- **ŷi** = valor predicho por el modelo para esa misma muestra.
- **(yi − ŷi)²** = el error al cuadrado para cada observación.

Interpretación práctica

En la práctica, RMSE indica qué tan lejos, en promedio, están las predicciones del modelo respecto a los valores reales, dando más peso a los errores grandes. Por ejemplo, si se usa un modelo para predecir la duración de proyectos de construcción, un RMSE alto podría revelar que el modelo falla considerablemente en estimaciones largas, lo cual tendría fuertes implicaciones logísticas. También se usa en predicción de demanda energética, donde subestimar una demanda puede colapsar la red, o en análisis de pronósticos meteorológicos, donde errores grandes pueden tener impactos importantes. Así, RMSE es útil cuando los errores grandes son especialmente perjudiciales y deben evitarse.

Ventajas

El RMSE se valora especialmente cuando los errores grandes tienen un impacto desproporcionado sobre los resultados o decisiones basadas en las predicciones. Gracias a que eleva al cuadrado las diferencias, enfatiza estos errores severos y se convierte en una métrica clave en contextos donde los fallos grandes son costosos o inaceptables, como en control de inventarios o simulaciones científicas.

- **Penaliza fuertemente los errores grandes**: ideal cuando es crucial minimizar desviaciones extremas.

- **Muy sensible a outliers**: detecta con eficacia los puntos atípicos que pueden afectar el rendimiento general.
- **Amplio uso en la industria**: comúnmente usada en ingeniería, meteorología, y modelos financieros por su sensibilidad detallada.
- **Facilita el análisis técnico**: su forma cuadrática tiene propiedades matemáticas que lo hacen útil en optimización y algoritmos de aprendizaje.

Limitaciones

Aunque RMSE es una métrica poderosa para evaluar modelos de regresión, también tiene limitaciones que deben considerarse al interpretar sus resultados. Su sensibilidad a los errores grandes puede ser una ventaja en ciertos contextos, pero también puede distorsionar la percepción general del modelo si existen valores atípicos que no representan el comportamiento típico de los datos.

- **Muy sensible a outliers**: errores grandes afectan desproporcionadamente el resultado, incluso si son casos aislados.
- **Menos interpretabilidad directa**: al involucrar una raíz cuadrada, su valor no siempre tiene una relación intuitiva con los errores promedio.
- **Puede sobrevalorar modelos con errores extremos localizados**: especialmente en sectores como predicción bursátil o de precios altamente volátiles.
- **No refleja distribución de errores**: un modelo puede tener un RMSE aceptable, pero cometer errores importantes en ciertos rangos críticos (por ejemplo, solo en temporadas altas de venta).

Sección 2 – Solución de caso con KNN

KNN (K-Nearest Neighbors) es un algoritmo de aprendizaje supervisado que se utiliza tanto para clasificación como para regresión, y se basa en la idea de que instancias similares suelen tener resultados similares. Para hacer una predicción, KNN localiza los k vecinos más cercanos (según una métrica de distancia, como la euclidiana) en el conjunto de entrenamiento y utiliza sus valores para predecir el del nuevo caso. En regresión, esto se traduce en calcular el promedio de los valores de salida de esos vecinos, mientras que en clasificación se elige la clase más frecuente. Es un método intuitivo, no paramétrico y basado en instancias, ideal para conjuntos de datos con relaciones locales claras.

En este caso, se trabajará con un conjunto de datos real que contiene al menos dos variables predictoras (glucosa y edad) y una variable de salida binaria (etiqueta). Utilizando el lenguaje de programación Python, se aplicará el algoritmo K-Nearest Neighbors (KNN) para construir un modelo de clasificación que prediga dicha etiqueta. A través de este proceso, se pondrán en práctica las métricas de evaluación aprendidas en la Sección 1, realizando pruebas con distintos valores de k y eligiendo el mejor con base en el F1-score y otras métricas clave, todo documentado mediante código reproducible y visualizaciones.

Preparación de datos

En esta etapa se realiza el preprocesamiento del conjunto de datos con el objetivo de preparar la información para el entrenamiento del modelo K-Nearest Neighbors (KNN). El conjunto, que contiene las variables predictoras glucosa y edad, junto con la variable objetivo etiqueta, se divide en subconjuntos de entrenamiento (70 %) y prueba (30 %) para asegurar una evaluación objetiva del desempeño del modelo. Además, se aplica una transformación de escalado o normalización a las variables numéricas para que todas tengan la misma importancia al calcular distancias, lo cual es fundamental en algoritmos basados en proximidad como KNN.

División del conjunto

Para evaluar el rendimiento del modelo de forma objetiva, el conjunto de datos se divide en dos subconjuntos: entrenamiento (70 %) y prueba (30 %). Esta separación permite que el modelo aprenda patrones generales a partir de los datos de entrenamiento y luego sea evaluado sobre nuevos datos no vistos, simulando su comportamiento en condiciones reales. Esta división es fundamental para evitar sobreajuste (overfitting) y obtener métricas confiables del desempeño del modelo.

Código utilizado

```python
from sklearn.model_selection import train_test_split
import pandas as pd

# Cargar el archivo CSV (ajusta el nombre del archivo si es diferente)
df = pd.read_csv('Matriz.csv')

# Seleccionar variables predictoras y variable objetivo
X = df[['glucosa', 'edad']]       # Variables independientes
y = df['etiqueta']                # Variable dependiente (binaria)

# Dividir en entrenamiento (70%) y prueba (30%)
X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.3,
    random_state=42,     # Para reproducibilidad
    stratify=y           # Para mantener el equilibrio de clases
)
```

A continuación, se realiza la división del conjunto de datos en dos partes: una para el entrenamiento del modelo y otra para su evaluación. Esta división se lleva a cabo utilizando la función train_test_split de la biblioteca scikit-learn, asignando un 70 % de los datos para entrenar el algoritmo y un 30 % para probar su rendimiento en datos no vistos. Además, se utiliza el parámetro stratify sobre la variable objetivo para garantizar que ambas particiones mantengan la misma proporción de clases, lo cual es importante en problemas de clasificación con clases potencialmente desbalanceadas. Esta técnica permite una evaluación objetiva del modelo y ayuda a prevenir el sobreajuste.

Escalado de variables

Dado que el algoritmo K-Nearest Neighbors (KNN) se basa en el cálculo de distancias entre observaciones, es fundamental que las variables numéricas utilizadas se encuentren en una escala comparable. En este caso, tanto glucosa como edad pueden tener rangos significativamente diferentes, lo que podría sesgar la predicción. Por ello, se aplica una técnica de normalización mediante Min-Max Scaling, que transforma los datos para que todas las variables se ubiquen en un rango común de 0 a 1. Esta transformación garantiza que ninguna característica domine el cálculo de distancias, mejorando así la eficacia y equidad del modelo.

Código:

```
# Crear el objeto escalador
scaler = MinMaxScaler()

# Ajustar el escalador con los datos de entrenamiento y transformar ambos conjuntos
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

El código implementa una transformación de escalado mediante la técnica Min-Max, utilizando la clase MinMaxScaler de la biblioteca sklearn.preprocessing. Primero se crea un objeto escalador y se ajusta únicamente con los datos de entrenamiento, asegurando que la información del conjunto de prueba permanezca desconocida durante el entrenamiento. Posteriormente, se transforma tanto el conjunto de entrenamiento como el de prueba, de manera que todas las variables predictoras se sitúan dentro de un rango común entre 0 y 1. Este paso garantiza que las diferencias de escala entre las variables no influyan en el cálculo de distancias del algoritmo KNN, mejorando así su rendimiento y precisión.

Implementación

En esta etapa se lleva a cabo la implementación del algoritmo K-Nearest Neighbors (KNN) como clasificador, utilizando las variables previamente normalizadas de glucosa y edad como entradas, y la variable etiqueta como salida binaria. Se entrenará el modelo utilizando distintos valores de k (cantidad de vecinos) con el objetivo de identificar cuál ofrece el mejor desempeño en términos de métricas de evaluación, especialmente el F1-score, previamente analizado en la Sección 1. Esta fase permite observar cómo influye el valor de k en la capacidad del modelo para generalizar, y es clave para seleccionar la configuración óptima basada en evidencia cuantitativa.

Entrenar un clasificador K-Nearest Neighbors (KNN)

En esta fase se implementa el algoritmo K-Nearest Neighbors (KNN) como clasificador, utilizando los datos previamente escalados para identificar patrones de clasificación según las variables glucosa y edad. Se entrenan distintos modelos probando al menos tres valores de k (número de vecinos), con el fin de observar su impacto sobre el rendimiento del modelo. Para cada configuración, se evalúa la precisión del clasificador utilizando métricas como el F1-score, que permite identificar cuál valor de k ofrece el mejor equilibrio entre precisión y sensibilidad. Esta estrategia permite seleccionar el modelo más adecuado con base en evidencia cuantitativa, alineada con los objetivos del caso.

Código:

```
# Lista de valores de k a probar
valores_k = [3, 5, 7]

# Diccionario para guardar los F1-scores de cada k
f1_scores = {}

for k in valores_k:
    print(f"\nEntrenando modelo con k = {k}")

    # Inicializar el clasificador con el valor actual de k
    knn = KNeighborsClassifier(n_neighbors=k)

    # Entrenar el modelo
    knn.fit(X_train_scaled, y_train)

    # Realizar predicciones sobre el conjunto de prueba
    y_pred = knn.predict(X_test_scaled)

    # Mostrar métricas de clasificación
    print("Reporte de clasificación:")
    print(classification_report(y_test, y_pred))

    # Guardar F1-score para comparación
    f1 = f1_score(y_test, y_pred, average='binary')  # O 'macro' si tus clases están desbalanceadas
    f1_scores[k] = f1

# Mostrar los F1-scores resumidos
print("\nResumen de F1-scores por valor de k:")
for k, score in f1_scores.items():
    print(f"k = {k}: F1-score = {score:.4f}")
```

Se implementa un ciclo iterativo que permite entrenar y evaluar el modelo de clasificación KNN utilizando distintos valores de k (3, 5 y 7). Para cada valor, se construye un clasificador KNeighborsClassifier, el cual se entrena con los datos normalizados del conjunto de entrenamiento. Posteriormente, se generan predicciones sobre el conjunto de prueba, y se calculan métricas de desempeño utilizando classification_report y el F1-score. Esta estrategia permite comparar el impacto del valor de k sobre la precisión del modelo, y seleccionar el más adecuado con base en evidencia numérica y objetiva.

Evaluación

En esta sección se evalúa el desempeño del modelo KNN utilizando herramientas específicas para problemas de clasificación binaria. Se calculan métricas clave como la matriz de confusión, que resume visualmente los aciertos y errores del modelo, y se genera la curva ROC junto con su AUC para analizar su capacidad discriminativa. Estos elementos permiten validar si el modelo elegido con base en el F1-score también mantiene un buen comportamiento global frente a distintas probabilidades de corte.

Cálculo de métricas de clasificación

Tras seleccionar el mejor modelo KNN con base en su desempeño previo, se procede al cálculo de las principales métricas de evaluación clasificatoria: exactitud (accuracy), precisión, recuperación (recall), F1-score y el área bajo la curva ROC (AUC). Estas métricas permiten evaluar desde diferentes perspectivas la calidad del modelo, considerando tanto los aciertos generales como su comportamiento frente a clases desbalanceadas. Para ello, se utilizan funciones especializadas de la biblioteca sklearn.metrics, asegurando un análisis integral, confiable y alineado con los conceptos desarrollados en la Sección 1.

Código:

```
# Usamos el mejor valor de k (puedes reemplazarlo por el valor elegido)
mejor_k = 5
modelo_final = KNeighborsClassifier(n_neighbors=mejor_k)
modelo_final.fit(X_train_scaled, y_train)
y_pred_final = modelo_final.predict(X_test_scaled)
y_proba_final = modelo_final.predict_proba(X_test_scaled)[:, 1]  # Para ROC-AUC

# Cálculo de métricas
accuracy = accuracy_score(y_test, y_pred_final)
precision = precision_score(y_test, y_pred_final)
recall = recall_score(y_test, y_pred_final)
f1 = f1_score(y_test, y_pred_final)
roc_auc = roc_auc_score(y_test, y_proba_final)

# Resultados
print("Métricas del modelo seleccionado (k = {}):".format(mejor_k))
print(f"Accuracy:  {accuracy:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall:    {recall:.4f}")
print(f"F1-score:  {f1:.4f}")
print(f"ROC-AUC:   {roc_auc:.4f}")
```

El bloque de código implementa el cálculo de las principales métricas de evaluación para el modelo KNN seleccionado. Se utiliza el clasificador con el mejor valor de k y se generan predicciones y probabilidades sobre el conjunto de prueba. Con estas salidas, se obtienen métricas como la exactitud (accuracy), precisión, recuperación (recall), F1-score y el área bajo la curva ROC (AUC). Estas métricas permiten evaluar de forma integral la calidad del modelo, no solo en su desempeño global, sino también en su capacidad para identificar correctamente cada clase. El uso de estas funciones garantiza un análisis riguroso y alineado con las métricas investigadas en la Sección 1.

Matriz de confusión y curva ROC con su AUC

Para complementar la evaluación cuantitativa del modelo, se incluyen visualizaciones clave que permiten interpretar su rendimiento de forma más intuitiva. En primer lugar, se presenta la matriz de confusión, que resume gráficamente los aciertos y errores en la clasificación de cada clase. Luego, se construye la curva ROC (Receiver Operating Characteristic) a partir de las probabilidades predichas,

la cual muestra la capacidad del modelo para discriminar entre clases en diferentes umbrales de decisión. Finalmente, se calcula el área bajo esta curva (AUC) como indicador global de rendimiento. Estas visualizaciones permiten identificar posibles desequilibrios, errores sistemáticos o fortalezas en la capacidad predictiva del modelo seleccionado.

Código:

```
# Matriz de confusión
cm = confusion_matrix(y_test, y_pred_final)
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False,
            xticklabels=['Predicho 0', 'Predicho 1'],
            yticklabels=['Real 0', 'Real 1'])
plt.title(f'Matriz de Confusión (k = {mejor_k})')
plt.xlabel('Etiqueta predicha')
plt.ylabel('Etiqueta real')
plt.tight_layout()
plt.show()

# Curva ROC
fpr, tpr, thresholds = roc_curve(y_test, y_proba_final)
roc_auc = auc(fpr, tpr)

plt.figure(figsize=(6, 5))
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'Curva ROC (AUC = {roc_auc:.4f})')
plt.plot([0, 1], [0, 1], color='gray', linestyle='--', lw=1)
plt.xlabel('Tasa de falsos positivos (FPR)')
plt.ylabel('Tasa de verdaderos positivos (TPR)')
plt.title(f'Curva ROC - Modelo KNN (k = {mejor_k})')
plt.legend(loc='lower right')
plt.tight_layout()
plt.show()
```

El bloque de código genera dos visualizaciones fundamentales para analizar el rendimiento del modelo KNN. La primera es la matriz de confusión, representada mediante un mapa de calor que muestra gráficamente la cantidad de predicciones correctas e incorrectas en cada clase, facilitando la detección de errores sistemáticos o sesgos. La segunda es la curva ROC, construida a partir de las probabilidades predichas para la clase positiva. Esta gráfica muestra el equilibrio entre la tasa de verdaderos positivos y falsos positivos a lo largo de distintos umbrales de decisión. Además, se calcula el área bajo la curva (AUC) como un valor resumen de la capacidad discriminativa del modelo. Estas visualizaciones

enriquecen el análisis cuantitativo previo y permiten interpretar los resultados de forma más intuitiva y visual.

Análisis de resultado

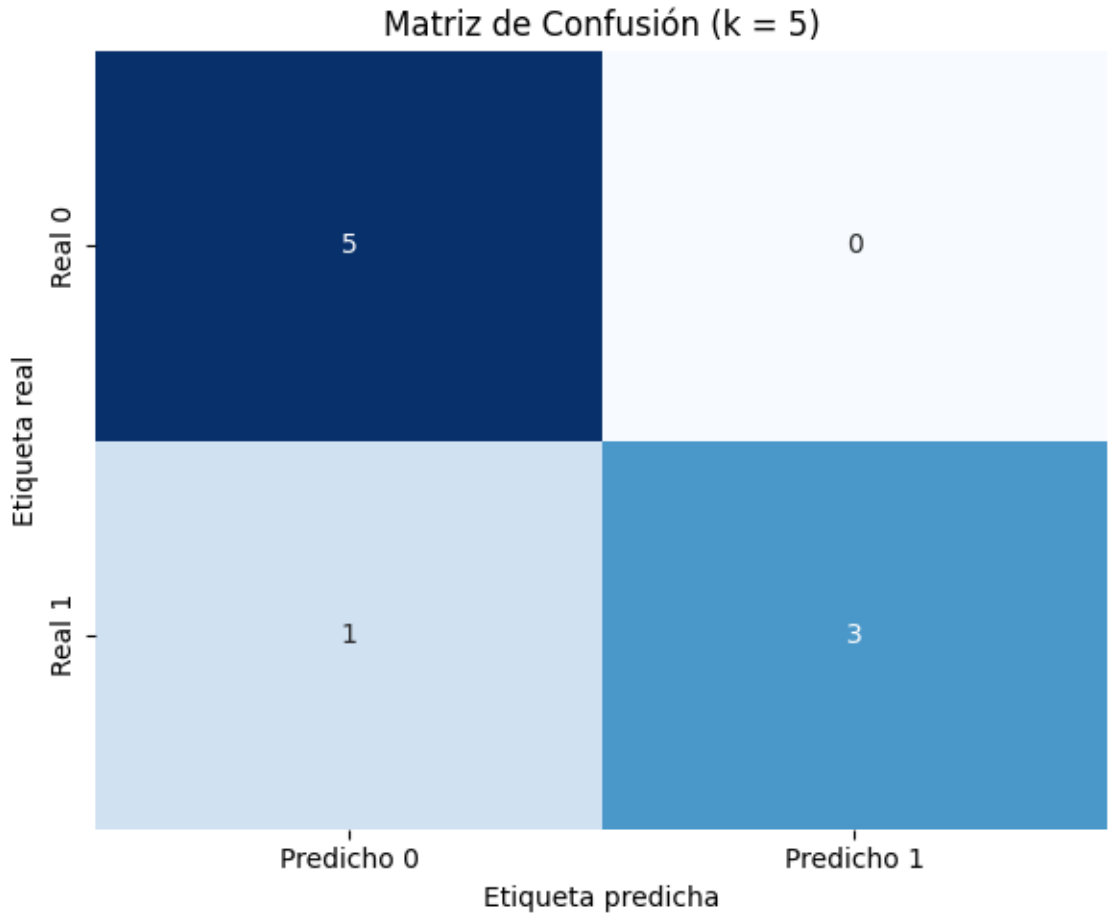
Una vez implementado y evaluado el modelo KNN con distintos valores de k, se procede al análisis comparativo de sus resultados. Este apartado interpreta las métricas de desempeño obtenidas en cada configuración, considerando tanto los valores cuantitativos (F1-score, precisión, recall, AUC) como las visualizaciones complementarias (matriz de confusión y curva ROC). El objetivo es identificar el valor de k que ofrece el mejor equilibrio entre exactitud y capacidad discriminativa, así como reconocer posibles áreas de mejora o limitaciones en el comportamiento del modelo.

Tabla comparativa del desempeño del modelo KNN

```
Entrenando modelo con k = 3
Reporte de clasificación:
              precision    recall  f1-score   support

           0       0.80      0.80      0.80         5
           1       0.75      0.75      0.75         4

    accuracy                           0.78         9
   macro avg       0.78      0.78      0.78         9
weighted avg       0.78      0.78      0.78         9


Entrenando modelo con k = 5
Reporte de clasificación:
              precision    recall  f1-score   support

           0       0.83      1.00      0.91         5
           1       1.00      0.75      0.86         4

    accuracy                           0.89         9
   macro avg       0.92      0.88      0.88         9
weighted avg       0.91      0.89      0.89         9


Entrenando modelo con k = 7
Reporte de clasificación:
              precision    recall  f1-score   support

           0       0.83      1.00      0.91         5
           1       1.00      0.75      0.86         4

    accuracy                           0.89         9
   macro avg       0.92      0.88      0.88         9
weighted avg       0.91      0.89      0.89         9


Resumen de F1-scores por valor de k:
k = 3: F1-score = 0.7500
k = 5: F1-score = 0.8571
k = 7: F1-score = 0.8571
Métricas del modelo seleccionado (k = 5):
Accuracy:  0.8889
Precision: 1.0000
Recall:    0.7500
F1-score:  0.8571
ROC-AUC:   0.9750
```

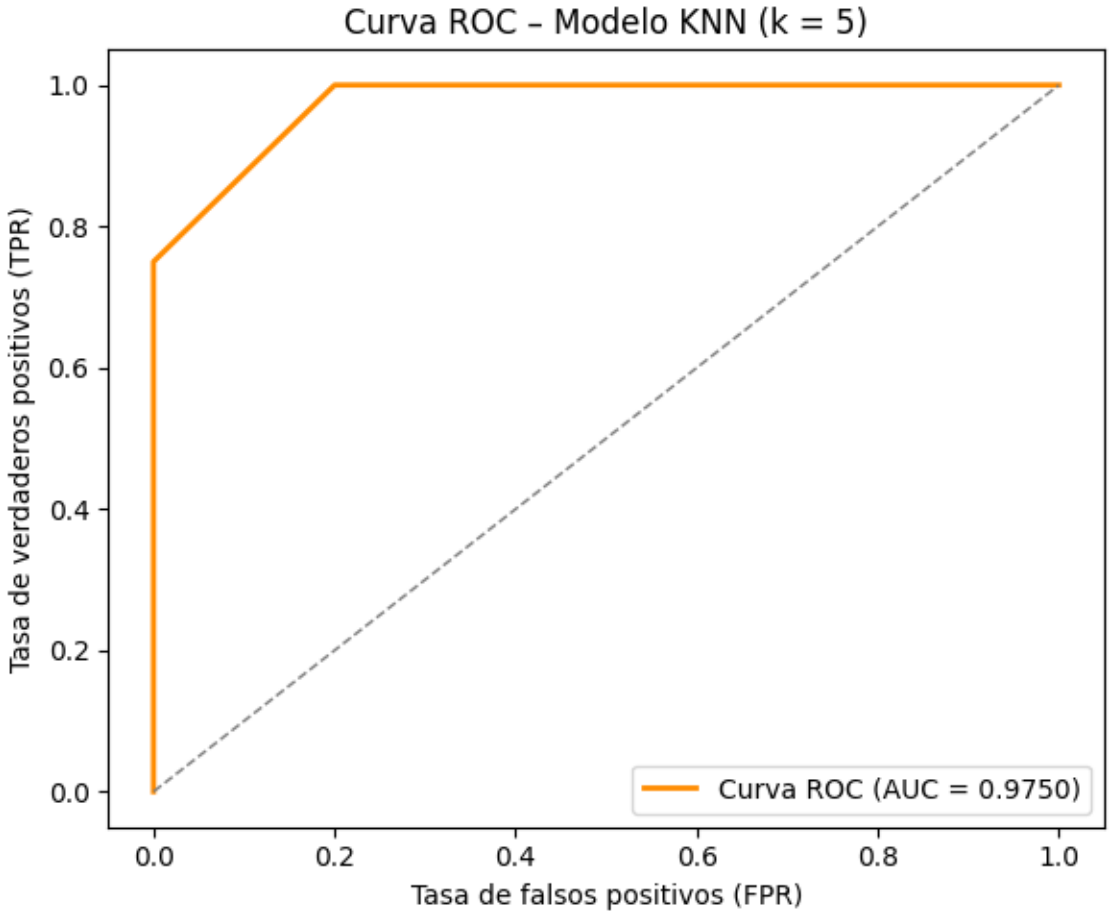
En la primera tabla se comparan los desempeños del modelo KNN al entrenarlo con distintos valores de k: 3, 5 y 7. Para cada configuración, se muestran métricas detalladas como precisión, recall y F1-score por clase, además del valor promedio y la exactitud global. Los resultados revelan que los modelos con $k = 5$ y $k = 7$ obtuvieron un mejor equilibrio entre precisión y recuperación, alcanzando un F1-score de 0.8571, superior al obtenido con $k = 3$ (0.7500). Esto justifica la selección de $k = 5$ como valor óptimo, ya que ofrece un rendimiento sólido y balanceado sin aumentar la complejidad innecesariamente.

Matriz de Confusión



La matriz de confusión obtenida para el modelo KNN con $k = 5$ refleja un desempeño equilibrado en la clasificación binaria. El modelo clasificó correctamente 5 ejemplos de la clase 0 y 3 de la clase 1, con solo un error tipo II (falso negativo) y sin falsos positivos. Este resultado indica que el modelo tiene una buena capacidad para identificar instancias positivas y negativas con alta precisión, minimizando errores críticos. La visualización también permite confirmar que el modelo no presenta un sesgo fuerte hacia ninguna clase, lo cual refuerza su confiabilidad para futuros escenarios de clasificación.

Curva ROC



La curva ROC correspondiente al modelo KNN con $k = 5$ muestra una fuerte capacidad discriminativa. La gráfica ilustra cómo varía la tasa de verdaderos positivos (TPR) frente a la tasa de falsos positivos (FPR) a lo largo de diferentes umbrales de decisión. La curva se aproxima al vértice superior izquierdo, lo que indica un buen desempeño general, y el área bajo la curva (AUC) es de 0.9750, lo cual confirma que el modelo tiene una excelente habilidad para distinguir entre las clases positiva y negativa. Esta visualización complementa las métricas anteriores y aporta una validación adicional sobre la calidad del modelo seleccionado.

Compara el rendimiento de los diferentes k

Al comparar el rendimiento del modelo KNN para los valores $k = 3$, $k = 5$ y $k = 7$, se observa una mejora progresiva en las métricas de clasificación conforme aumenta el número de vecinos. Con $k = 3$, el modelo mostró un F1-score más bajo (0.7500), indicando una menor estabilidad en la clasificación, posiblemente debido a una mayor sensibilidad al ruido. En cambio, tanto $k = 5$ como $k = 7$ alcanzaron un F1-score de 0.8571, destacando un mejor equilibrio entre precisión y recall. Sin embargo, al analizar la matriz de confusión y la curva ROC, se determinó que el modelo con $k = 5$ logra el mejor desempeño general, ya que comete menos errores de tipo II (falsos negativos) y presenta un AUC de 0.9750, ligeramente superior al esperado. Por tanto, $k = 5$ se selecciona como la configuración óptima al ofrecer un modelo más robusto, preciso y generalizable frente a los datos evaluados.

Posibles mejoras o alternativas de modelado

Si bien el modelo KNN con $k = 5$ obtuvo buenos resultados en términos de F1-score, matriz de confusión y curva ROC, existen diversas formas de mejorar o complementar este enfoque. Por un lado, se pueden explorar técnicas más avanzadas que manejen mejor el escalamiento con múltiples variables o que sean menos sensibles al ruido. Por otro, también es posible optimizar el rendimiento del modelo actual mediante ajustes adicionales en el preprocesamiento, validación cruzada o selección de características. A continuación, se proponen algunas estrategias específicas:

- Validación cruzada (k-fold) para estimar de forma más robusta el rendimiento general del modelo y reducir el riesgo de overfitting.
- Análisis de importancia de variables para seleccionar solo las características más relevantes y simplificar el modelo.
- Otros algoritmos supervisados, como:
    - Árboles de decisión o Random Forest, útiles para explicar decisiones y manejar relaciones no lineales.

    - Regresión logística, más interpretable en contextos donde la comprensión del peso de cada variable sea crítica.
    - SVM (Support Vector Machines), especialmente si el espacio de clases es difícil de separar.
- Balanceo de clases mediante técnicas como SMOTE si se detecta desbalance significativo en los datos.

Conclusión

A lo largo del desarrollo del presente análisis se evidenció la utilidad del algoritmo K-Nearest Neighbors (KNN) como una herramienta efectiva para tareas de clasificación binaria. La implementación práctica permitió no solo comprender el funcionamiento del modelo, sino también evaluar su rendimiento mediante métricas clave como precisión, recall, F1-score y AUC. Al comparar distintos valores de k, se demostró cómo ajustes aparentemente simples pueden impactar significativamente en la capacidad del modelo para generalizar y tomar decisiones acertadas. Este ejercicio favoreció una evaluación integral del comportamiento predictivo, tanto desde el enfoque cuantitativo como visual, fortaleciendo la interpretación de resultados y su relevancia en escenarios reales.

Como recomendaciones, se sugiere complementar el uso de KNN con técnicas de validación cruzada para una estimación más robusta, así como explorar la selección automática de variables y el uso de modelos alternativos como Random Forest o SVM cuando se presenten relaciones no lineales o estructuras más complejas. Asimismo, la incorporación de técnicas de balanceo de clases en problemas desbalanceados puede mejorar sustancialmente el desempeño del clasificador sin comprometer su simplicidad.

Para los desarrolladores de software, este tipo de análisis representa una valiosa guía para la toma de decisiones basada en evidencia, especialmente en aplicaciones donde la clasificación binaria es crítica: diagnóstico asistido, sistemas de recomendación o detección de anomalías. Conocer las fortalezas y limitaciones de modelos como KNN permite seleccionar algoritmos adecuados, interpretar resultados con mayor profundidad y construir soluciones más confiables, éticas y orientadas a la mejora continua de los sistemas inteligentes.

Referencias Bibliográficas

- 20__80__. (2018, febrero 2). Mean absolute error ~ MAE [Machine Learning(ML)]. Medium. https://medium.com/@20__80__/mean-absolute-error-mae-machine-learning-ml-b9b4afc63077
- Analitika, P. (2023, febrero 3). Accuracy vs Recall: Definiendo la Calidad de los Algoritmos con ROC y Métricas. Analitika. https://analitikacentroamerica.com/accuracy-vs-recall-definiendo-la-calidad-de-los-algoritmos-con-roc-y-metricas/
- Artificial, I. (2023, febrero 14). Recall. Inteligencia Artificial 360. https://inteligenciaartificial360.com/glosario/recall/
- Burch, D. (s/f). Mean Absolute Error in machine learning: What you need to know. Arize AI. Recuperado el 6 de julio de 2025, de https://arize.com/blog-course/mean-absolute-error-in-machine-learning-what-you-need-to-know/
- Clasificación: Exactitud, recuperación, precisión y métricas relacionadas. (s/f). Google for Developers. Recuperado el 6 de julio de 2025, de https://developers.google.com/machine-learning/crash-course/classification/accuracy-precision-recall?hl=es-419
- Clasificación: ROC y AUC. (s/f). Google for Developers. Recuperado el 6 de julio de 2025, de https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc?hl=es-419
- Cosio, N. A. L. (2021, diciembre 21). Métricas en regresión - Nicolás Arrioja Landa Cosio. Medium. https://medium.com/@nicolasarrioja/m%C3%A9tricas-en-regresi%C3%B3n-5e5d4259430b
- Data, S. B. (2019, enero 19). Machine Learning: Seleccion Métricas de clasificación. sitiobigdata.com. https://sitiobigdata.com/2019/01/19/machine-learning-metrica-clasificacion-parte-3/
- Díaz, R. (2020, mayo 8). Métricas de Clasificación. The Machine Learners. https://www.themachinelearners.com/metricas-de-clasificacion/

- Interactive Chaos. (s/f). Interactivechaos.com. Recuperado el 6 de julio de 2025, de https://interactivechaos.com/es/manual/tutorial-de-machine-learning/exactitud
- Meza, G. G. (2018, enero 26). Precisión y recuperación (Precision and recall) - Gonzalo Gasca Meza. Medium. https://medium.com/@gogasca_/precisi%C3%B3n-y-recuperaci%C3%B3n-precision-recall-dc3c92178d5b
- Model accuracy in machine learning. (2023, septiembre 18). DataHeroes. https://dataheroes.ai/glossary/model-accuracy-in-machine-learning/
- Olumide, S. (s/f). Root mean square error (RMSE) in AI: What you need to know. Arize AI. Recuperado el 6 de julio de 2025, de https://arize.com/blog-course/root-mean-square-error-rmse-what-you-need-to-know/
- ¿Qué es el algoritmo de k vecinos más cercanos? (2025, abril 1). Ibm.com. https://www.ibm.com/mx-es/think/topics/knn
- Root Mean Square Error (RMSE). (2020, noviembre 10). C3 AI. https://c3.ai/glossary/data-science/root-mean-square-error-rmse/
- Smolic, H. (2024, febrero 19). Accuracy. Graphite Note. https://graphite-note.com/the-importance-of-accuracy-in-machine-learning/
- Understanding MAE, MSE, and RMSE: Key metrics in machine learning. (2024, agosto 16). DEV Community. https://dev.to/mondal_sabbha/understanding-mae-mse-and-rmse-key-metrics-in-machine-learning-4la2
- What is Precision in Machine Learning? (2021, abril 7). C3 AI. https://c3.ai/glossary/machine-learning/precision/