UNIVERSIDAD TECNOLÓGICA DE CHIHUAHUA

TECNOLOGÍAS DE LA INFORMACIÓN

EXTRACCIÓN DE CONOCIMIENTO EN BASES DE DATOS

III.2. REPORTE DE MÉTRICAS DE EVALUACIÓN

IDGS91N

PRESENTA:

REGINA CHÁVEZ TAMAYO - 6521110019

DOCENTE:

LUIS ENRIQUE MASCOTE CANO

Chihuahua, Chih., 30 de noviembre de 2025

Contents

Introducción

En el aprendizaje supervisado, la correcta evaluación de los modelos es tan importante como su entrenamiento. Un modelo que aparenta tener buen rendimiento puede, en realidad, estar sobreajustado o ser poco útil en la práctica si se evalúa con métricas inadecuadas. Por ello, es fundamental comprender las distintas métricas de evaluación, su significado e implicaciones.

En este reporte se investiga un conjunto de métricas de evaluación para modelos de clasificación y modelos de regresión, incluyendo sus definiciones, fórmulas matemáticas, interpretación práctica, ventajas y limitaciones. Posteriormente, se aplican estas métricas en un caso práctico de clasificación binaria utilizando el algoritmo K-Nearest Neighbors (KNN) sobre una matriz de datos con variables predictoras (glucosa, edad) y una etiqueta binaria.

El objetivo general es comprender y aplicar métricas de evaluación de modelos para analizar el desempeño de un clasificador KNN, comparando distintos valores de *k* y extrayendo conclusiones sustentadas en los resultados obtenidos.

Investigación de métricas

Métricas de clasificación

En un problema de clasificación binaria se utilizan los siguientes conceptos básicos:

- TP (True Positives / Verdaderos positivos)
- TN (True Negatives / Verdaderos negativos)
- FP (False Positives / Falsos positivos)
- FN (False Negatives / Falsos negativos)

A partir de ellos se definen métricas clave.

Accuracy (Exactitud)

Definición y fórmula

La accuracy representa la proporción de ejemplos correctamente clasificados sobre el total de ejemplos.

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

Interpretación práctica

Indica qué porcentaje total de predicciones del modelo son correctas. Un valor cercano a 1 (o 100 %) sugiere que el modelo comete pocos errores en general.

Ventajas

- Fácil de entender y comunicar.
- Útil cuando las clases están balanceadas.

Limitaciones

- Puede ser engañosa en clases desbalanceadas: un modelo que siempre predice la clase mayoritaria puede tener alta accuracy pero ser inútil para detectar la clase minoritaria.

Precisión

Definición y fórmula

La precisión mide la proporción de verdaderos positivos entre todos los ejemplos que el modelo predijo como positivos.

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

Interpretación práctica

Responde a la pregunta: "De todo lo que el modelo dijo que era positivo, ¿qué fracción realmente lo era?" Es especialmente relevante cuando el costo de un falso positivo es alto (por ejemplo, marcar correos legítimos como spam).

Ventajas

- Útil cuando queremos minimizar los falsos positivos.
- Importante en contextos donde una acción errónea sobre un negativo es costosa.

Limitaciones

- No considera los falsos negativos.
- Debe analizarse junto con el recall.

Recall (Sensibilidad o tasa de verdaderos positivos)

Definición y fórmula

El recall mide la proporción de verdaderos positivos detectados entre todos los positivos reales.

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

Interpretación práctica

Responde a la pregunta: "De todos los positivos que había en realidad, ¿cuántos encontré?" Es fundamental en problemas donde es muy costoso no detectar un caso positivo (por ejemplo, detección de enfermedades).

Ventajas

- Útil cuando queremos minimizar los falsos negativos.
- Importante en detección de fraudes, diagnósticos médicos, etc.

Limitaciones

- No considera los falsos positivos.
- Un recall alto con precisión baja puede significar muchos falsos positivos.

F1-score

Definición y fórmula

El F1-score es la media armónica entre precisión y recall.

$$\text{F1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Interpretación práctica

Equilibra precisión y recall en un único valor. Es especialmente útil cuando las clases están desbalanceadas y se desea un balance entre evitar falsos positivos y falsos negativos.

Ventajas

- Combina precisión y recall en una sola métrica.
- Útil en conjuntos desbalanceados.

Limitaciones

- No distingue explícitamente el costo relativo de FP y FN.
- No incorpora información sobre verdaderos negativos.

ROC-AUC (Área bajo la curva ROC)

Definición

La curva ROC (Receiver Operating Characteristic) representa la relación entre la tasa de verdaderos positivos (TPR) y la tasa de falsos positivos (FPR) para distintos umbrales de decisión. El AUC (Area Under the Curve) es el área bajo esta curva y resume el rendimiento global del modelo.

$$\text{TPR} = \frac{TP}{TP + FN}, \quad \text{FPR} = \frac{FP}{FP + TN}$$

Interpretación práctica

Un AUC cercano a 1.0 indica que el modelo separa muy bien las clases positivas y negativas. Un AUC de 0.5 indica un comportamiento similar al azar.

Ventajas

- Independiente del umbral de clasificación.
- Robusto ante cierto desbalance de clases.
- Permite comparar distintos modelos de manera global.

Limitaciones

- Puede dar una impresión demasiado optimista cuando solo interesa un rango específico de TPR/FPR.
- Menos intuitivo para usuarios no técnicos.

Métricas de regresión

En regresión se evalúa la calidad de las predicciones numéricas comparándolas con los valores reales.

MAE (Mean Absolute Error)

Definición y fórmula

El MAE mide el error absoluto promedio entre los valores reales y las predicciones:

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i|$$

Interpretación práctica

Indica, en promedio, cuántas unidades se equivoca el modelo. Se expresa en las mismas unidades de la variable objetivo.

Ventajas

- Fácil de interpretar.
- Menos sensible a valores atípicos que el MSE/RMSE.

Limitaciones

- No penaliza fuertemente los errores grandes.
- Menos adecuado cuando interesa castigar en mayor medida grandes desviaciones.

RMSE (Root Mean Squared Error)

Definición y fórmula

El RMSE es la raíz cuadrada del error cuadrático medio (MSE):

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Interpretación práctica

Resume el tamaño promedio de los errores, penalizando más fuertemente los errores grandes. También se expresa en las mismas unidades de la variable objetivo.

Ventajas

- Muy usado en práctica.
- Penaliza más los grandes errores, lo que puede ser deseable.

Limitaciones

- Sensible a valores atípicos.
- Una pequeña cantidad de errores muy grandes puede inflar la métrica.

Solución con KNN (preprocesamiento, entrenamiento, evaluación)

Descripción de la matriz de datos

Se utilizó la matriz de datos proporcionada (archivo Matriz.csv), con 30 observaciones y las siguientes columnas:

- glucosa: nivel de glucosa (numérica).
- edad: edad de la persona (numérica).
- etiqueta: variable binaria (0 = negativo, 1 = positivo).

El objetivo es clasificar cada registro como clase 0 o 1 utilizando un clasificador KNN.

Preprocesamiento y partición de datos

1. **Selección de variables**

    - Variables de entrada (predictoras): glucosa, edad.
    - Variable objetivo: etiqueta.

2. **División entrenamiento/prueba (70 % / 30 %)**

    Se realizó una división estratificada para conservar la proporción de clases en ambos conjuntos:

    - Entrenamiento: 70 % de los datos (21 observaciones).
    - Prueba: 30 % de los datos (9 observaciones).

3. **Escalado / normalización**

    Dado que KNN es un algoritmo basado en distancias, se aplicó escalado estándar (StandardScaler) a las variables glucosa y edad, de manera que queden con media 0 y desviación estándar 1.

Implementación del clasificador KNN

Se implementó KNN utilizando Python y scikit-learn, probando al menos tres valores de *k*: 3, 5 y 7.

Pasos:

1. Escalar los datos de entrenamiento y prueba.
2. Entrenar modelos KNN con distintos valores de k.
3. Generar predicciones sobre el conjunto de prueba.

4. Calcular las métricas de evaluación para cada *k*: accuracy, precision, recall, F1-score y ROC-AUC.
5. Seleccionar el mejor modelo según el F1-score y, en caso de empate, usar AUC como criterio adicional.

Resultados

Métricas para distintos valores de k

Sobre el conjunto de prueba (9 instancias), se obtuvieron los siguientes resultados:

Tabla 1. Métricas de evaluación por valor de k

k	Accuracy	Precision	Recall	F1-score	AUC ROC	Matriz de confusión (TN, FP / FN, TP)
3	0.78	0.75	0.75	0.75	0.93	[[4, 1], [1, 3]]
5	0.89	1.00	0.75	0.86	0.98	[[5, 0], [1, 3]]
7	0.89	1.00	0.75	0.86	1.00	[[5, 0], [1, 3]]

Nota: las métricas se redondearon a dos decimales.

Para k = 5 y k = 7 el F1-score es aproximadamente 0.86 en ambos casos. El AUC es ligeramente superior para k = 7 (AUC ≈ 1.00), por lo que, aunque el criterio principal es F1-score, se selecciona k = 7 como modelo final por su mejor capacidad de separación global entre clases.

Matriz de confusión del mejor modelo (k = 7)

La matriz de confusión para k = 7 (formato [[TN, FP], [FN, TP]]) es:

$$\begin{bmatrix} 5 & 0 \\ 1 & 3 \end{bmatrix}$$

Interpretación:

- **TN = 5**: 5 negativos correctamente clasificados como 0.
- **FP = 0**: ningún negativo clasificado erróneamente como positivo.
- **FN = 1**: 1 positivo clasificado erróneamente como negativo.
- **TP = 3**: 3 positivos correctamente clasificados como 1.

Esto muestra un modelo que no comete falsos positivos, pero sí al menos un falso negativo.

Curva ROC y AUC

Para k = 7, se calculó la curva ROC a partir de las probabilidades predichas por el modelo. La curva ROC se obtiene graficando TPR (recall) contra FPR para diferentes umbrales de decisión.

- El área bajo la curva (AUC) fue aproximadamente 1.00, lo que indica que, en este conjunto de prueba, el modelo separa muy bien las clases.
- En términos prácticos, el modelo asigna sistemáticamente mayores probabilidades a los ejemplos positivos que a los negativos.

Comparación de rendimiento entre valores de k

- **k = 3**
    - Métricas aceptables pero inferiores (accuracy ≈ 0.78, F1 ≈ 0.75).
    - Comete tanto FP como FN.
- **k = 5**
    - Mejora tanto en accuracy como en F1-score (≈ 0.89 y ≈ 0.86).
    - Elimina los falsos positivos (FP = 0), pero se mantiene 1 FN.
- **k = 7**
    - Mantiene accuracy y F1-score similares a k = 5.
    - Mejora el AUC hasta ≈ 1.00, lo que indica excelente discriminación entre clases.
    - Matriz de confusión idéntica a k = 5 (en este pequeño conjunto de prueba).

La elección final de k = 7 se justifica por el empate en F1-score con k = 5 y su mejor AUC.

Conclusiones y recomendaciones

En este trabajo se revisaron diversas métricas de evaluación para clasificación y regresión, destacando cómo cada una captura aspectos distintos del desempeño de un modelo. En la parte práctica, se aplicaron estas métricas a un problema de clasificación binaria utilizando KNN sobre un conjunto de datos con variables de glucosa y edad.

A partir de los resultados:

- Se observó que métricas como precision, recall, F1-score y ROC-AUC ofrecen una visión más completa que la *accuracy* por sí sola.
- El modelo KNN con k = 7 mostró un rendimiento sólido, sin falsos positivos y con un único falso negativo, alcanzando un F1-score ≈ 0.86 y AUC ≈ 1.00.
- El análisis de la matriz de confusión permitió identificar claramente el tipo de errores cometidos (principalmente falsos negativos).

Recomendaciones:

1. Ampliar el tamaño del conjunto de datos para obtener estimaciones más robustas de las métricas.
2. Probar otros algoritmos de clasificación (por ejemplo, regresión logística, árboles de decisión) y comparar su desempeño con KNN usando las mismas métricas.
3. Aplicar validación cruzada para reducir la dependencia de una sola partición entrenamiento/prueba.
4. Ajustar más valores de *k* y estudiar la estabilidad de las métricas a través de diferentes particiones.

Referencias

Google Developers. (2025). *Classification: Accuracy, precision, and recall*. Machine Learning Crash Course. https://developers.google.com/machine-learning/crash-course/classification/accuracy-precision-recall (Google for Developers)

IBM. (2024). *What is model performance in machine learning?* IBM Think. https://www.ibm.com/think/topics/model-performance (ibm.com)

GeeksforGeeks. (2025). *Regression Metrics in Machine Learning*. GeeksforGeeks. https://www.geeksforgeeks.org/machine-learning/regression-metrics/ (GeeksforGeeks)

Pedregosa, F., Varoquaux, G., Gramfort, A., et al. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research, 12*, 2825–2830. http://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html (scikit-learn.org)

Coursera. (2025). *What Is ROC Curve in Machine Learning?* Coursera. https://www.coursera.org/articles/what-is-roc-curve (Coursera)

Anexos

Anexo A. Script completo en Python (KNN + métricas)

Script completo

```python
import pandas as pd
import numpy as np

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,
    roc_auc_score,
    confusion_matrix,
    roc_curve
)
import matplotlib.pyplot as plt

# 1. Carga de datos
data = pd.read_csv("Matriz.csv")  # columnas: glucosa, edad, etiqueta

X = data[["glucosa", "edad"]]
y = data["etiqueta"]

# 2. División 70% entrenamiento, 30% prueba (estratificada)
X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.3,
    random_state=42,
    stratify=y
)

# 3. Escalado de variables (muy importante para KNN)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# 4. Entrenamiento y evaluación para varios k
ks = [3, 5, 7]
resultados = []

for k in ks:
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train_scaled, y_train)

    y_pred = knn.predict(X_test_scaled)
    y_proba = knn.predict_proba(X_test_scaled)[:, 1]

    acc = accuracy_score(y_test, y_pred)
    prec = precision_score(y_test, y_pred, zero_division=0)
    rec = recall_score(y_test, y_pred, zero_division=0)
    f1 = f1_score(y_test, y_pred, zero_division=0)
    auc = roc_auc_score(y_test, y_proba)
    cm = confusion_matrix(y_test, y_pred)

    resultados.append({
        "k": k,
        "accuracy": acc,
        "precision": prec,
        "recall": rec,
        "f1": f1,
        "auc": auc,
        "confusion_matrix": cm
    })
```

```python

# Impresión de resultados
for r in resultados:
    print(f"k = {r['k']}")
    print("  Accuracy :", round(r["accuracy"], 3))
    print("  Precision:", round(r["precision"], 3))
    print("  Recall   :", round(r["recall"], 3))
    print("  F1-score :", round(r["f1"], 3))
    print("  AUC      :", round(r["auc"], 3))
    print("  Matriz de confusión:\n", r["confusion_matrix"])
    print("-" * 40)

# 5. Selección del mejor k (por F1; en caso de empate, por AUC)
mejor = max(resultados, key=lambda r: (r["f1"], r["auc"]))
print("Mejor modelo:")
print("k =", mejor["k"], "F1 =", round(mejor["f1"], 3), "AUC =", round(mejor["auc"], 3))

# 6. Curva ROC del mejor modelo
mejor_k = mejor["k"]
knn_mejor = KNeighborsClassifier(n_neighbors=mejor_k)
knn_mejor.fit(X_train_scaled, y_train)
y_proba_mejor = knn_mejor.predict_proba(X_test_scaled)[:, 1]

fpr, tpr, thresholds = roc_curve(y_test, y_proba_mejor)

plt.figure()
plt.plot(fpr, tpr, label=f"ROC KNN (k={mejor_k})")
plt.plot([0, 1], [0, 1], linestyle="--", label="Azar")
plt.xlabel("Tasa de falsos positivos (FPR)")
plt.ylabel("Tasa de verdaderos positivos (TPR)")
plt.title("Curva ROC")
plt.legend()
plt.grid(True)
plt.show()
```