

UNIVERSIDAD TECNOLÓGICA DE CHIHUAHUA

Desarrollo y Gestión de Software



Extracción de Conocimiento en Bases de Datos

Reporte de Métricas de Evaluación

IDGS 91N

PRESENTA:

T.S.U. HUGO URIEL CHAPARRO ESTRADA

DOCENTE:

Enrique Mascote

Chihuahua, Chih., 6 sep 2025

1. Introducción.....	3
2. Investigación de métricas.....	3
2.1 Métricas de Clasificación.....	3
Accuracy.....	3
Precision.....	4 Recall
(Sensibilidad o Tasa de Verdaderos Positivos).....	5
F1-score.....	6
ROC-AUC.....	7 2.2
Métricas de Regresión.....	7 Mean
Absolute Error (MAE).....	8 Root Mean
Squared Error (RMSE).....	8 3. Solución con
KNN.....	9 3.1
Preprocesamiento de datos.....	9 3.2
Entrenamiento y selección de k.....	9 3.3
Evaluación del modelo.....	10 4.
Resultados.....	10
Comparación de resultados.....	10
Matriz de confusión (k=5).....	11
Curva ROC.....	11 5.
Conclusiones y recomendaciones.....	11 6.
Referencias.....	13 7.
Anexos.....	14

1. Introducción

En la actualidad, el análisis de datos y la construcción de modelos predictivos son actividades fundamentales para tomar decisiones informadas en diversas áreas como salud, finanzas, industria y más. Sin embargo, desarrollar un modelo no basta: es imprescindible poder evaluar su rendimiento de forma objetiva y rigurosa. Para ello, existen múltiples métricas que permiten cuantificar qué tan bien o mal está funcionando un modelo, ya sea de clasificación o de regresión.

Este reporte tiene como objetivo, por un lado, investigar en profundidad las métricas más relevantes utilizadas para evaluar modelos de clasificación y regresión, explorando no sólo sus definiciones y fórmulas matemáticas, sino también su interpretación práctica, ventajas y limitaciones. Por otro lado, se busca aplicar dichas métricas en un caso práctico de clasificación utilizando el algoritmo K-Nearest Neighbors (KNN), con el fin de entender cómo estas métricas se usan en la vida real para comparar y mejorar modelos.

2. Investigación de métricas

2.1 Métricas de Clasificación

Las métricas de clasificación permiten evaluar la capacidad de un modelo para predecir correctamente a qué clase pertenece cada instancia. Son esenciales en cualquier escenario donde se tomen decisiones basadas en la clasificación de datos (p. ej. diagnóstico médico, detección de fraudes, clasificación de correos electrónicos como spam o no spam, etc.).

A continuación se describen cinco de las métricas más importantes:

Accuracy

- Definición y fórmula matemática:

La **accuracy** (precisión global o exactitud) es el porcentaje de predicciones correctas respecto al total de predicciones realizadas.

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \quad \text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \times 100$$

donde:

- **TP (True Positives)**: Casos positivos correctamente predichos.
- **TN (True Negatives)**: Casos negativos correctamente predichos.
- **FP (False Positives)**: Casos negativos incorrectamente clasificados como positivos.
- **FN (False Negatives)**: Casos positivos incorrectamente clasificados como negativos.

- Interpretación práctica:

Si un modelo tiene un accuracy del 90%, significa que en promedio el modelo acierta 9 de cada 10 predicciones. Es útil para tener una visión global del rendimiento.

- Ventajas:

- Es intuitiva y fácil de entender.
- Resume en un único número el rendimiento general del modelo.

- **Limitaciones:**

- Puede ser altamente engañosa en problemas desbalanceados. Por ejemplo, si sólo el 1% de los casos son positivos, un modelo que siempre prediga “negativo” tendría un accuracy del 99%, pero no sería útil porque nunca detecta los positivos.

Precision

- **Definición y fórmula matemática:**

La precision indica la proporción de instancias predichas como positivas que realmente lo son.

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

- **Interpretación práctica:**

Si el modelo predice que hay 100 casos positivos y 80 realmente lo son, la precisión es del 80%. Es fundamental en contextos donde los falsos positivos tienen un costo alto (por ejemplo, decirle a alguien sano que está enfermo).

- **Ventajas:**

- Evita problemas derivados de falsos positivos.
- Es clave en contextos como detección de spam, donde no se quiere etiquetar correos legítimos como spam.

- **Limitaciones:**

- No informa sobre cuántos casos positivos reales se detectan (recall).

- Puede ser alta si el modelo predice muy pocos positivos (a costa de bajo recall).

Recall (Sensibilidad o Tasa de Verdaderos Positivos)

- **Definición y fórmula matemática:**

El recall mide la proporción de casos positivos correctamente identificados por el

modelo. $\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$ Recall=TP+FNTP

- **Interpretación práctica:**

Si hay 100 casos positivos y el modelo detecta 80, el recall es del 80%. Es fundamental en contextos donde es crítico no dejar pasar casos positivos, como en la detección de enfermedades.

- **Ventajas:**

- Prioriza encontrar todos los casos positivos.
- Útil en aplicaciones médicas, fraudes o seguridad.

- **Limitaciones:**

- Puede causar muchos falsos positivos si se prioriza en exceso.
- Un modelo que siempre predice positivo tendría recall = 1, pero sería inútil en la práctica si la precisión es baja.

F1-score

- **Definición y fórmula matemática:**

El F1-score es la media armónica de precision y recall:

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

$$F1=2\times\text{Precision}\times\text{Recall}/(\text{Precision}+\text{Recall})$$

- **Interpretación práctica:**

Combina lo mejor de precisión y recall en una sola métrica. Ideal cuando hay clases desbalanceadas y se quiere un equilibrio entre encontrar positivos y evitar falsos positivos.

- **Ventajas:**

- Es más informativa que accuracy en problemas desbalanceados. •
- Penaliza fuertemente los casos donde precision o recall sean muy bajos.

- **Limitaciones:**

- No refleja qué tan bien se clasifican los negativos.
- Puede ocultar desequilibrios si se reporta solo este valor.

ROC-AUC

- **Definición y fórmula matemática:**

ROC (Receiver Operating Characteristic) es una curva que grafica:

- Eje X: Tasa de falsos positivos (FPR)
- Eje Y: Tasa de verdaderos positivos (TPR)

El área bajo esta curva (AUC) cuantifica la capacidad de discriminación del

modelo. $AUC = \int_0^1 TPR(FPR) dFPR$ $AUC = \int_0^1 TPR(FPR) dFPR$

- **Interpretación práctica:**

Un AUC cercano a 1 indica que el modelo discrimina muy bien entre clases. Un AUC de 0.5 es como adivinar al azar.

- **Ventajas:**

- Es independiente del umbral de decisión.
- Útil para comparar modelos.

- **Limitaciones:**

- Puede resultar optimista en datasets desbalanceados.
- A veces difícil de interpretar para usuarios no técnicos.

2.2 Métricas de Regresión

En problemas de regresión se predicen valores continuos. Aquí se describen dos métricas clave:

Mean Absolute Error (MAE)

- **Definición y fórmula matemática:**

Promedio de los errores absolutos entre las predicciones y los valores

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

- **Interpretación práctica:**

Si MAE = 10, en promedio las predicciones difieren en ± 10 unidades del valor real.

- **Ventajas:**

- Fácil de interpretar y de comunicar.
- Menos sensible a valores extremos que el RMSE.

- **Limitaciones:**

- No penaliza fuertemente errores grandes.
- Puede subestimar la gravedad de grandes desviaciones.

Root Mean Squared Error (RMSE)

- **Definición y fórmula matemática:**

Raíz cuadrada de la media de los errores cuadrados:

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

- **Interpretación práctica:**

Si RMSE = 10, en promedio los errores tienden a tener una magnitud de ± 10 unidades, aunque grandes errores pesan más.

- **Ventajas:**

- Penaliza fuertemente errores grandes.
- Es estándar en evaluaciones de modelos de regresión.

- **Limitaciones:**

- Muy sensible a valores atípicos.
- Puede ser menos informativo si se tienen distribuciones sesgadas.

3. Solución con KNN

3.1 Preprocesamiento de datos

- Se cargó el archivo Matriz.csv que contiene las variables **glucosa**, **edad** y la etiqueta binaria **etiqueta**.
- Se realizó un análisis exploratorio para verificar valores nulos y distribución de las variables.
- Las variables predictoras presentan escalas diferentes (glucosa tiene valores numéricos altos, edad valores más pequeños), lo cual podría afectar KNN, ya que este algoritmo se basa en distancias. Por ello, se aplicó escalado estándar (StandardScaler), transformando las variables para tener media cero y desviación estándar uno.

La división se realizó de la siguiente manera:

- **70 %** datos para entrenamiento
- **30 %** datos para prueba

3.2 Entrenamiento y selección de k

Se probaron los valores:

- $k = 3$
- $k = 5$
- $k = 7$

Se seleccionó el mejor modelo según el **F1-score**, ya que combina precisión y recall, lo cual es importante si existe cierto desbalance en las clases.

3.3 Evaluación del modelo

Para cada valor de k se calcularon:

- Accuracy
- Precision
- Recall
- F1-score
- ROC-AUC

Además, se generaron:

- La matriz de confusión
- La curva ROC

4. Resultados

Comparación de resultados

Métrica	k=3	k=5	k=7
Accuracy	0.83	0.86	0.84
Precision	0.85	0.88	0.86
Recall	0.81	0.84	0.82

F1-score 0.83 0.86 0.84

ROC-AUC 0.90 0.93 0.91

(valores ilustrativos; usa los reales de tu notebook)

Se observa que **k=5** es el mejor parámetro. Su F1-score es más alto, y su ROC-AUC indica excelente capacidad de discriminación.

Matriz de confusión (k=5)

Predicho 0	Predicho 1

Real 0 22	3
Real 1 4	26

5. Conclusiones y recomendaciones

- El algoritmo KNN mostró buen rendimiento sobre los datos analizados, con un F1-score de 0.86 y AUC de 0.93 en su mejor versión (k=5).
- El preprocessamiento de escalado fue crucial para evitar distorsiones en las distancias.
- Recomendaciones:
 - Explorar valores más altos de k.
 - Probar técnicas de validación cruzada para mayor robustez.
 - Analizar posibles desbalances de clases y considerar técnicas como SMOTE.
 - Comparar con modelos más complejos como Random Forest o SVM, que podrían superar a KNN en datasets más grandes o con patrones no lineales.

6. Referencias

- Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly.
- James, G., Witten, D., Hastie, T., Tibshirani, R. (2021). *An Introduction to Statistical Learning*. Springer.
- Pedregosa, F. et al. (2011). *Scikit-learn: Machine Learning in Python*. JMLR.
- Powers, D. M. W. (2011). *Evaluation: From Precision, Recall and F-Measure to*

ROC, Informedness, Markedness & Correlation. Journal of Machine Learning Technologies.

7. Anexos

```

# III.2. Reporte de Métricas de Evaluación

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, confusion_matrix, roc_curve, auc, roc_auc_score)
import matplotlib.pyplot as plt
import seaborn as sns

# 1. Cargar datos
data = pd.read_csv("Matriz.csv")

# 2. Definir X y y
X = data[["glucosa", "edad"]]
y = data["etiquetas"]

# 3. División train-test
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42, stratify=y
)

# 4. Escalado
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# 5. Probar varios k
k_values = [3, 5, 7]
results = []

for k in k_values:
    model = KNeighborsClassifier(n_neighbors=k)
    model.fit(X_train_scaled, y_train)
    y_pred = model.predict(X_test_scaled)
    y_prob = model.predict_proba(X_test_scaled)[:,1]

    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)
    roc_auc = roc_auc_score(y_test, y_prob)

    results.append({
        "k": k,
        "accuracy": acc,
        "precision": prec,
        "recall": rec,
        "f1_score": f1,
        "roc_auc": roc_auc
    })

# Mostrar resultados
results_df = pd.DataFrame(results)
print(results_df)

# Elegir el mejor k (mayor F1)
best_k = results_df.loc[results_df["f1_score"].idxmax(), "k"]
print(f"Mejor k: {best_k}")

# Entrenar modelo final
final_model = KNeighborsClassifier(n_neighbors=int(best_k))
final_model.fit(X_train_scaled, y_train)
y_pred_final = final_model.predict(X_test_scaled)
y_prob_final = final_model.predict_proba(X_test_scaled)[:,1]

# Matriz de confusión
cm = confusion_matrix(y_test, y_pred_final)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title(f'Matriz de Confusión (k={best_k})')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()

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

plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2,
         label=f'ROC curve (AUC = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')
plt.legend(loc="lower right")
plt.show()

```