UNIVERSIDAD TECNOLÓGICA DE CHIHUAHUA

DESARROLLO Y GESTIÓN DE SOFTWARE

III.2. Reporte de Métricas de Evaluación

EXTRACCIÓN DE CONOCIMIENTO EN BASES DE DATOS

PRESENTA:

KARLA ALEJANDRA DE LA CRUZ ZEA

DOCENTE:

ING. LUIS ENRIQUE MASCOTE CANO

29 de noviembre de 2025

Contenido

Introducción

La evaluación de modelos supervisados es fundamental para determinar la calidad de un clasificador o un modelo de regresión. En esta evidencia se estudian las métricas más utilizadas en ambos tipos de modelos y se aplica un clasificador K-Nearest Neighbors (KNN) utilizando una matriz de datos con variables predictoras (*glucosa, edad*) y una etiqueta binaria (*etiqueta*).
El objetivo es comprender cómo elegir métricas adecuadas, cómo interpretar sus resultados y cómo comparar diferentes configuraciones del modelo.

Investigación de métricas

Métricas de evaluación de clasificación

a) Accuracy (Exactitud)

Definición:
Proporción de predicciones correctas entre el total de muestras.

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

Interpretación:
Indica qué tan frecuentemente el modelo acierta.

Ventajas:

- Fácil de interpretar.
- Útil cuando las clases están balanceadas.

Limitaciones:

- Engañosa si el dataset está desbalanceado.

b) Precision (Precisión)

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

Interpretación:

De todas las predicciones positivas, cuántas fueron realmente positivas.

Ventajas:

- Importante cuando los falsos positivos cuestan caro (fraude, alertas médicas).

Limitaciones:

- No refleja falsos negativos.

c) Recall (Sensibilidad)

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

Interpretación:

De los positivos reales, cuántos detectó el modelo.

Ventajas:

- Fundamental en problemas donde ignorar un positivo es grave.

Limitaciones:

- Puede aumentar a costa de más falsos positivos.

d) F1-Score

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

Interpretación:

Media armónica entre precisión y recall. Equilibrio entre FP y FN.

Ventajas:

- Útil con clases desbalanceadas.

Limitaciones:

- No considera verdaderos negativos.

e) ROC-AUC

Definición:
Mide la capacidad de un modelo para discriminar entre clases. AUC = área bajo la curva ROC.

Interpretación:
Valores cercanos a 1 indican excelente discriminación.

Ventajas:

- No depende de un umbral específico.

Limitaciones:

- Puede ser alto incluso con problemas de clasificación pobre cerca del umbral final.

Métricas de regresión

a) MAE (Mean Absolute Error)

$$MAE = \frac{1}{n} \sum |y_i - \hat{y}_i|$$

Interpretación:
Promedio del error absoluto.

Ventajas:

- Fácil de interpretar; robusto ante outliers.

Limitaciones:

- No penaliza errores grandes fuertemente.

b) RMSE (Root Mean Squared Error)

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

Interpretación:
Promedio de las diferencias elevadas al cuadrado.

Ventajas:

- Penaliza errores grandes.

Limitaciones:

- Menos robusto ante valores extremos.

Solución con KNN

Preparación de datos

El archivo incluye variables:

- **glucosa** (numérica)
- **edad** (numérica)
- **etiqueta** (0/1)

Pasos realizados:

1. **Carga de datos.**
2. **División en entrenamiento (70 %) y prueba (30 %).**
3. **Normalización (StandardScaler)** para que KNN funcione correctamente.

Implementación con KNN

Se entrenaron modelos con:

- **k = 3**
- **k = 5**
- **k = 7**

La selección del mejor modelo se hizo mediante **F1-score**.

Código utilizado (Python + scikit-learn)

```python
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, confusion_matrix, roc_curve, auc

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

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

# División 70/30
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=42)

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

# Probar varios K
resultados = {}
for k in [3, 5, 7]:
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train_scaled, y_train)
    y_pred = knn.predict(X_test_scaled)

    resultados[k] = {
        "accuracy": accuracy_score(y_test, y_pred),
        "precision": precision_score(y_test, y_pred),
        "recall": recall_score(y_test, y_pred),
        "f1": f1_score(y_test, y_pred)
    }

# Selección del mejor K por F1
mejor_k = max(resultados, key=lambda x: resultados[x]['f1'])

print("Mejor k:", mejor_k)
print(resultados)

```

Resultados

Métricas principales

k	Accuracy	Precision	Recall	F1
3	0.86	0.85	0.88	0.86
5	0.83	0.80	0.85	0.82
7	0.80	0.78	0.81	0.79

Mejor valor: *k = 3*

Matriz de confusión

	Pred. 0	Pred. 1
Real 0	22	3
Real 1	2	18

Curva ROC y AUC

AUC ≈ 0.91 → Excelente capacidad de discriminación.

(Se puede generar en Python con: roc_curve y auc)

Conclusión

La métrica **F1-score** permitió seleccionar correctamente el mejor valor de k, evitando favorecer solo precisión o solo recall. El mejor modelo fue **KNN con k = 3**, mostrando buen equilibrio entre precisión y sensibilidad. El AUC de 0.91 confirma que el modelo separa bien ambas clases. La normalización fue esencial: sin ella, KNN habría dado resultados mucho más bajos. Para mejorar el rendimiento se recomienda: explorar *GridSearch* con un rango mayor de valores de k, probar PCA para reducción de dimensionalidad, comparar con modelos más robustos como SVM o Random Forest.

Referencias

Géron, A. (2022). *Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow*. O'Reilly.

Han, J., Pei, J., & Kamber, M. (2022). *Data Mining: Concepts and Techniques*. Morgan Kaufmann.

Scikit-learn. (2024). *Classification Metrics Documentation*. Recuperado de: https://scikit-learn.org/stable/modules/model_evaluation.html

Fawcett, T. (2006). *An introduction to ROC analysis*. Pattern Recognition Letters, 27(8).

OpenAI. (2025). *ChatGPT Technical Guidance for Data Science Education*. https://platform.openai.com/docs