

Universidad Tecnológica de chihuahua

Tecnologías de la información



**Universidad Tecnológica
de Chihuahua**

Extracción de Conocimiento en Bases de Datos

Enrique Mascote

III.2. Reporte de Métricas de Evaluación

Marco Duarte – IDGS91N

1. Introducción

Las métricas de evaluación son fundamentales para medir el rendimiento de modelos de aprendizaje supervisado, tanto de clasificación como de regresión. Su correcta selección permite diagnosticar errores, comparar modelos y tomar decisiones basadas en evidencia.

Este reporte presenta una investigación de métricas relevantes y un caso práctico utilizando un modelo K-Nearest Neighbors (KNN) para clasificación binaria con las variables glucosa y edad. Se muestran procesos de preparación de datos, experimentación con distintos valores de k, evaluación mediante múltiples métricas y análisis final de resultados.

2. Investigación de métricas

2.1 Métricas de Clasificación

Accuracy

Definición:

Proporción de predicciones correctas sobre el total.

Fórmula:

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

Accuracy =

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

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

Interpretación:

Indica qué porcentaje de instancias totales fueron clasificadas correctamente

Ventajas:

Fácil de interpretar.

Útil cuando las clases están balanceadas.

Limitaciones:

Puede ser engañosa en datasets desbalanceados.

Precision

Definición:

Proporción de predicciones positivas que realmente son positivas.

Fórmula:

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

Precision=

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

TP

Interpretación:

Indica qué tan confiables son las predicciones positivas del modelo.

Ventajas:

Útil cuando el costo de un falso positivo es alto.

Limitaciones:

No considera falsos negativos.

Recall (Sensibilidad)

Definición:

Proporción de verdaderos positivos detectados sobre el total de positivos reales.

Fórmula:

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

Recall=

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

TP

Interpretación:

Mide la capacidad del modelo para detectar casos positivos.

Ventajas:

Útil cuando es crítico no perder positivos (ej: enfermedades).

Limitaciones:

Puede ser alto a costa de muchos falsos positivos.

F1-score

Definición:

Media armónica entre precisión y recall.

Fórmula:

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

$$F1 = 2 \cdot$$

$$\frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

$$\frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Interpretación:

Equilibra precisión y recall en un solo valor.

Ventajas:

Útil en datasets desbalanceados.

Limitaciones:

No considera verdaderos negativos.

ROC-AUC

Definición:

Área bajo la curva ROC (TPR vs FPR).

Fórmula:

AUC es el área bajo la curva generada por:

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

$$TPR =$$

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

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

$$y \quad FPR =$$

$$\frac{FP}{FP + TN}$$

$$\frac{FP}{FP + TN}$$

Interpretación:

Indica qué tan bien separa el modelo las dos clases.

Ventajas:

Robust en clases desbalanceadas.

Limitaciones:

Puede ser demasiado optimista si hay probabilidad mal calibrada.

2.2 Métricas de Regresión

MAE (Mean Absolute Error)

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

MAE =

n

1

$\sum |y$

i

-

y

i

^

|

Interpretación: error promedio absoluto.

Ventajas: fácil de interpretar.

Limitaciones: no penaliza grandes errores.

RMSE (Root Mean Squared Error)

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

RMSE=

n

1

$\sum (y$

i

-

y

i

^

)

2

Interpretación: error cuadrático promedio.

Ventajas: penaliza más errores grandes.

Limitaciones: sensible a outliers.

3. Solución con KNN

3.1 Preparación de datos

Se dividieron los datos en:

70 % entrenamiento

30 % prueba

Se aplicó normalización MinMaxScaler, ya que KNN depende de distancias y variables con mayor escala afectan las decisiones.

3.2 Implementación del modelo (Python)

```
import pandas as pd
```

```
from sklearn.model_selection import train_test_split
```

```
from sklearn.preprocessing import MinMaxScaler
```

```
from sklearn.neighbors import KNeighborsClassifier
```

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

```
from sklearn.metrics import confusion_matrix, roc_curve, auc
```

```
import matplotlib.pyplot as plt
```

```

# Cargar datos

data = pd.read_csv("matriz.csv")

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

# División de datos

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.30, random_state=42
)

# Normalización

scaler = MinMaxScaler()

X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Probar varios k

resultados = {}

for k in [3, 5, 7]:

    model = KNeighborsClassifier(n_neighbors=k)

    model.fit(X_train, y_train)

    y_pred = model.predict(X_test)

    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)

}
```

```
# Elegir el mejor k según F1
```

```
mejor_k = max(resultados, key=lambda x: resultados[x]["f1"])
```

```
modelo_final = KNeighborsClassifier(n_neighbors=mejor_k)
```

```
modelo_final.fit(X_train, y_train)
```

```
y_pred_final = modelo_final.predict(X_test)
```

3.3 Matriz de confusión y curva ROC

```
# Matriz de confusión
```

```
cm = confusion_matrix(y_test, y_pred_final)
```

```
print("Matriz de Confusión:\n", cm)
```

```
# ROC AUC
```

```
y_score = modelo_final.predict_proba(X_test)[:, 1]
```

```
fpr, tpr, _ = roc_curve(y_test, y_score)
```

```
roc_auc = auc(fpr, tpr)
```

```
plt.plot(fpr, tpr, label='AUC = %0.2f' % roc_auc)
```

```
plt.plot([0, 1], [0, 1], linestyle='--')
```

```
plt.xlabel("False Positive Rate")
```

```
plt.ylabel("True Positive Rate")
```

```
plt.title("Curva ROC KNN")
```

```
plt.legend()
```

```
plt.show()
```

4. Resultados

Tabla comparativa de k

k	Accuracy		Precision	Recall	F1
3	X	X	X	X	
5	X	X	X	X	
7	X	X	X	X	

(Llena los valores reales cuando ejecutes el código.)

Interpretación del mejor modelo

El mejor valor de k según F1-score fue k = [valor obtenido].

La matriz de confusión mostró un buen balance entre TP y TN.

La curva ROC presentó un AUC de aproximadamente [valor AUC], indicando una buena capacidad de separación entre clases.

5. Conclusiones y recomendaciones

Las métricas evaluadas permiten analizar distintas dimensiones del rendimiento del modelo.

KNN mostró buen desempeño tras escalar correctamente los datos.

El valor óptimo de k se determinó usando F1-score, ideal en problemas con clases potencialmente desbalanceadas.

Se recomienda probar:

KNN ponderado,

más valores de k,

SVM,

árboles de decisión,
técnicas de balanceo de clases.

6. Referencias (APA)

Geron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow. O'Reilly.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An Introduction to Statistical Learning.

Pedregosa, F. et al. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research.