Ingeniería en Inteligencia Artificial,
Machine Learning
Sem: 2025-1, 5BM1, Práctica 7, Fecha: 3 de Noviembre 2024

Laboratorio 7: clasificador KNN

Machine Learning

Grupo: 5BM1

Profesor: Andrés Floriano García

Integrantes:
Juan Manuel Alvarado Sandoval
Alexander Iain Crombie Esquinca
Herrera Saavedra Jorge Luis
Quiñones Mayorga Rodrigo

Contents

1 Introducción

En este laboratorio, se explorará el uso del clasificador K-Nearest Neighbors (KNN), uno de los algoritmos más básicos y efectivos en el aprendizaje supervisado. Este clasificador asigna una clase a una nueva instancia basándose en la clase mayoritaria de sus k vecinos más cercanos. Este enfoque es no paramétrico, lo que significa que no asume una distribución particular de los datos.

El clasificador KNN se utiliza en múltiples aplicaciones de clasificación, incluyendo reconocimiento de patrones, minería de datos y visión por computadora. A lo largo de este laboratorio, se implementará el clasificador KNN y se validará su desempeño en varios conjuntos de datos populares: Iris, Wine y Breast Cancer.

Se evaluará el rendimiento del algoritmo usando métodos de validación como Hold-Out 70/30 estratificado, 10-Fold Cross-Validation estratificado y Leave-One-Out.

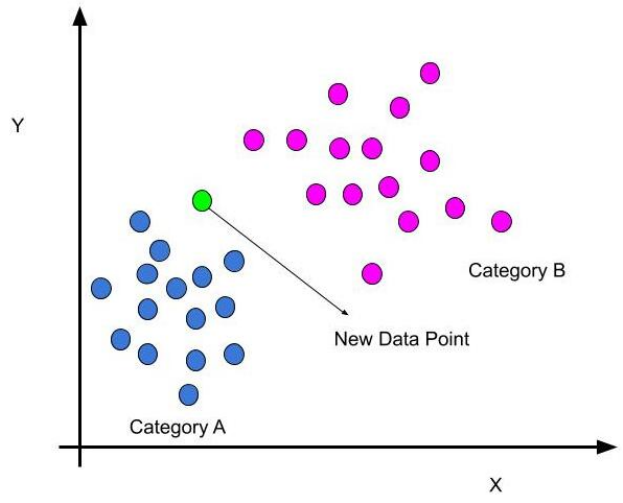


Figure 1: Clasificador K Nearest Neighbours.

2 Clasificador K-Nearest Neighbors (KNN)

El algoritmo KNN clasifica una nueva instancia basándose en los k vecinos más cercanos dentro del conjunto de entrenamiento. Para determinar la proximidad, normalmente se utiliza una métrica de distancia, como la distancia euclidiana. El valor de k se selecciona dependiendo del problema y puede influir significativamente en el rendimiento del modelo.

El algoritmo KNN sigue los siguientes pasos:

1. **Almacenamiento del conjunto de datos:** No se requiere un proceso de entrenamiento explícito, ya que el clasificador KNN almacena todas las instancias del conjunto de entrenamiento.

2. **Clasificación:**

   - Para una nueva instancia, se calcula la distancia entre esta y todas las instancias en el conjunto de entrenamiento.
   - Se seleccionan los k vecinos más cercanos a la nueva instancia.
   - La clase más frecuente entre los k vecinos se asigna a la nueva instancia.

Matemáticamente, se puede expresar como:

$$\hat{y} = \arg\max_{c \in C} \sum_{i=1}^{k} \delta(y_i, c)$$

donde $\hat{y}$ es la clase predicha, C es el conjunto de clases, y_i es la clase de los vecinos más cercanos y $\delta(y_i, c)$ es 1 si $y_i = c$ y 0 en caso contrario.

3 Validación y Resultados

Se utilizaron tres conjuntos de datos para evaluar el rendimiento del clasificador KNN: Iris, Wine y Breast Cancer. Los resultados se validaron utilizando los siguientes métodos:

- **Hold-Out 70/30 Estratificado**: El conjunto de datos se divide en un 70% para entrenamiento y un 30% para prueba, manteniendo la proporción de clases.
- **10-Fold Cross-Validation Estratificado**: El conjunto de datos se divide en 10 subconjuntos, y se entrena el modelo 10 veces utilizando un subconjunto diferente para la validación en cada iteración.
- **Leave-One-Out**: Cada instancia del conjunto de datos se utiliza una vez como conjunto de prueba mientras el resto se utiliza para entrenar.

3.1 Justificación del Parámetro K

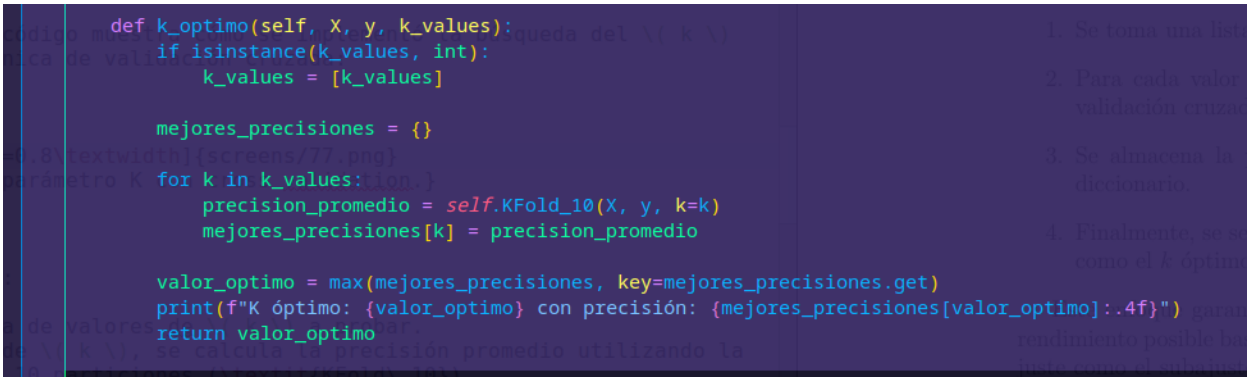
Uno de los aspectos fundamentales en el método k-Nearest Neighbors (k-NN) es la elección del parámetro k, que representa el número de vecinos más cercanos a considerar al clasificar una instancia. Un valor de k demasiado pequeño puede hacer que el modelo sea sensible al ruido, mientras que un k demasiado grande puede suavizar demasiado las fronteras de decisión, lo que reduce la precisión del clasificador.

Para seleccionar el mejor valor de k, se utiliza comúnmente un proceso de validación cruzada, el cual permite evaluar el rendimiento del modelo para diferentes valores de k, asegurando así que el modelo generalice bien a nuevos datos.

En esta práctica, se implementó un método que realiza validación cruzada utilizando la técnica de K-Fold Cross-Validation, concretamente con 10 particiones (*KFold_10*). El siguiente fragmento de código muestra cómo se implementó la búsqueda del k óptimo utilizando esta técnica de validación cruzada:

El proceso es el siguiente:

1. Se toma una lista de valores de k a probar.
2. Para cada valor de k, se calcula la precisión promedio utilizando la validación cruzada con 10 particiones (*KFold_10*).



```python
def k_optimo(self, X, y, k_values):
    if isinstance(k_values, int):
        k_values = [k_values]

    mejores_precisiones = {}
    for k in k_values:
        precision_promedio = self.KFold_10(X, y, k=k)
        mejores_precisiones[k] = precision_promedio

    valor_optimo = max(mejores_precisiones, key=mejores_precisiones.get)
    print(f"K óptimo: {valor_optimo} con precisión: {mejores_precisiones[valor_optimo]:.4f}")
    return valor_optimo
```

Figure 2: Elección del parámetro K con cross validation.

3. Se almacena la precisión promedio correspondiente a cada k en un diccionario.

4. Finalmente, se selecciona el valor de k con la mejor precisión promedio como el k óptimo.

El enfoque que se propuso garantiza que el parámetro k elegido proporcione el mejor rendimiento posible basado en los datos disponibles, evitando tanto el sobreajuste como el subajuste.

3.2 Resultados del Clasificador KNN

Iris Dataset

```
import pandas as pd
import sklearn
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

#preprocesado de datos
data= load_iris()
X= data.data
y= data.target

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)


instancia_algoritmo= validacion()
k_values= 15


instancia_algoritmo.HoldOut(X_scaled, y,k_values)
instancia_algoritmo.KFold_desempeño(X_scaled, y, 10)
instancia_algoritmo.LeaveOneOut(X_scaled, y,k_values)
```

Figure 3: Implementación genérica mediante la llamada a la clase validación
.

```
K óptimo: 15 con precisión: 0.9667
MÉTODO DE VALIDACIÓN: HOLD OUT 70-30
----------------------Desempeño de aplicar KNN con 15 vecinos----------------------------
accuracy: 1.0
```

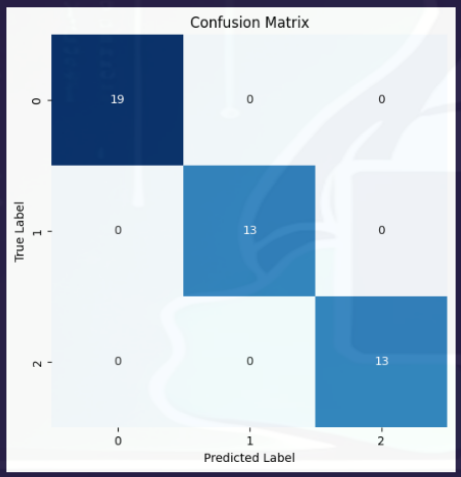


Figure 4: Desempeño de KNN con K=15 (valor optimizado encontrado con cross-validation) mediante Hold Out para Iris Dataset.

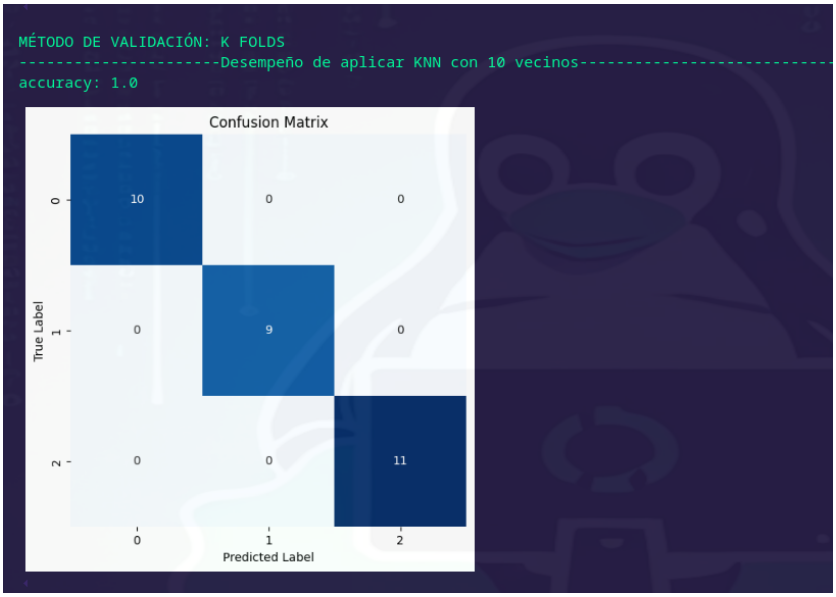


Figure 5: Desempeño de KNN mediante K Folds (10) para Iris Dataset.

Figure 6: Desempeño de KNN con K=15 (valor optimizado encontrado con cross-validation) mediante Leave One out para Iris Dataset.

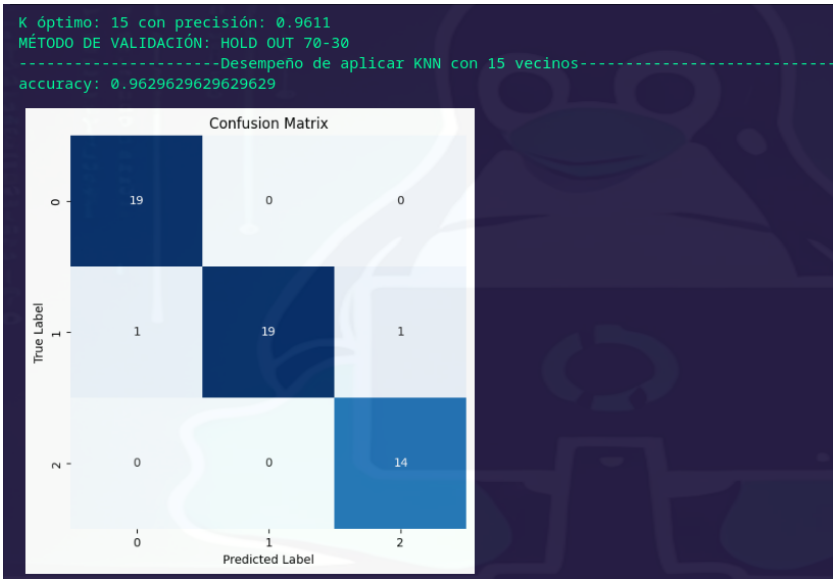


Figure 7: Desempeño de KNN con K=15 (valor optimizado encontrado con cross-validation) mediante Hold Out para Wine Dataset.

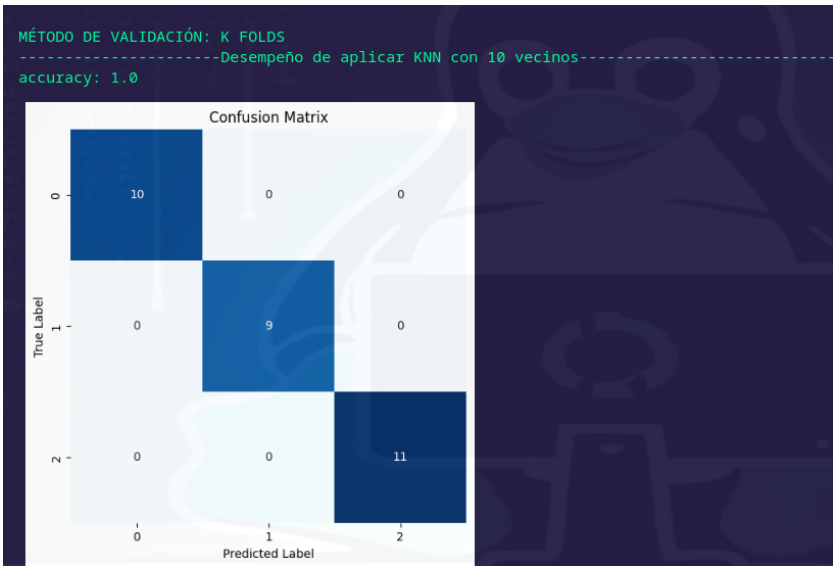


Figure 8: Desempeño de KNN mediante K Folds (10) para Wine Dataset.

Figure 9: Desempeño de KNN con K=15 (valor optimizado encontrado con cross-validation) mediante Leave One out para Wine Dataset.

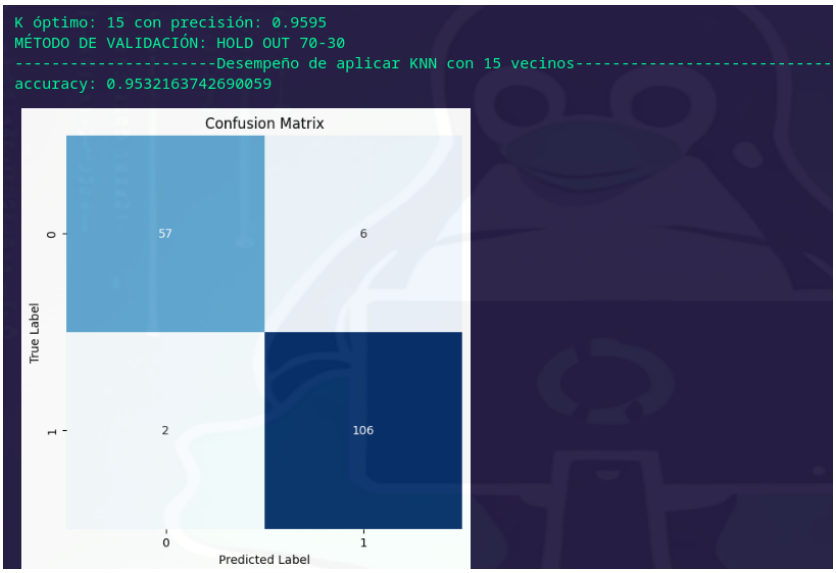


Figure 10: Desempeño de KNN con K=15 (valor optimizado encontrado con cross-validation) mediante Hold Out para Breast Cancer Dataset.

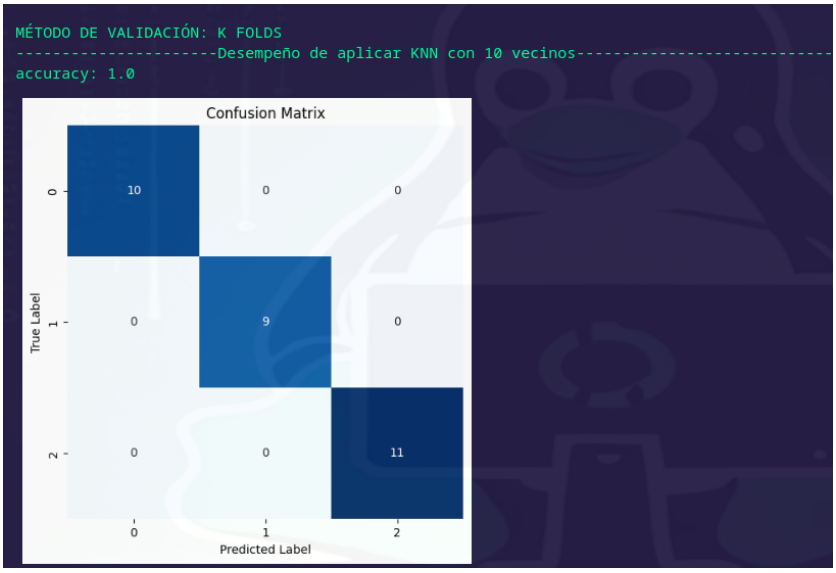


Figure 11: Desempeño de KNN mediante K Folds (10) para Breast Cancer Dataset.

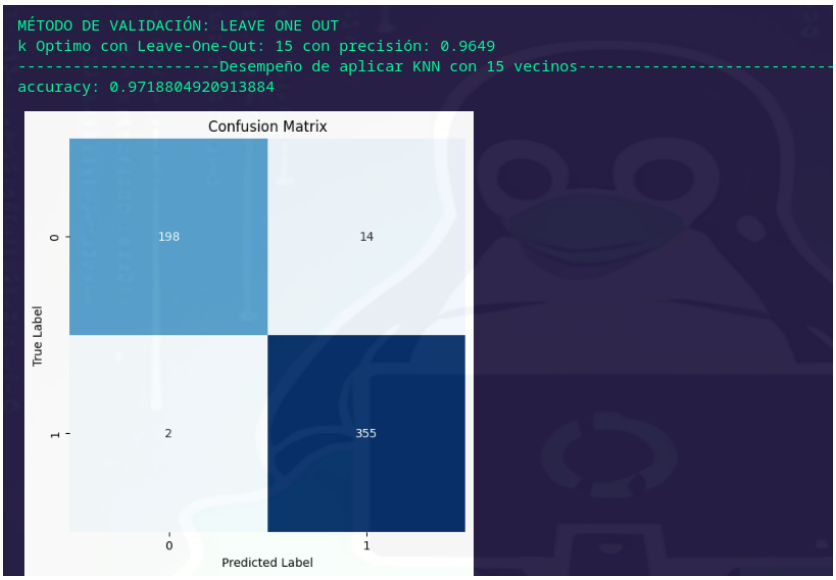


Figure 12: Desempeño de KNN con K=15 (valor optimizado encontrado con cross-validation) mediante Leave One out para Breast Cancer Dataset.

4 Conclusión

El clasificador KNN es un algoritmo poderoso debido a su simplicidad y efectividad en muchas tareas de clasificación. A través de este laboratorio, se demostró que, a pesar de ser un método simple, puede ser competitivo si se selecciona adecuadamente el valor de k y se preprocesan correctamente los datos.

Los resultados en los conjuntos de datos Iris, Wine y Breast Cancer reflejan que el clasificador KNN puede obtener buenos resultados, aunque es sensible a los valores de k y a la escala de las características. Los métodos de validación aplicados proporcionan una evaluación adecuada del rendimiento del modelo.

5 Enlace al repositorio

Haz click para seguir el enlace

6 Referencias

- Scikit-learn developers. (2024). *Scikit-learn: Machine Learning in Python*. Recuperado de: https://scikit-learn.org/stable/