Actividad: Laboratorio 7: clasificador KNN

Alumno:

Miguel Angel Ocampo

Gerardo Martinez Ayala

Maestro:

Andres Garcia Floriano

Explicación del código.

El código es un ejemplo completo y estructurado de cómo implementar y evaluar un modelo de clasificación basado en el algoritmo K-Nearest Neighbors (KNN) utilizando diversos enfoques de validación, como Hold-Out, 10-Fold Cross-Validation, y Leave-One-Out (LOO). El objetivo principal es aplicar este modelo a diferentes datasets estándar incluidos en la biblioteca sklearn y, a través de estas evaluaciones, analizar el rendimiento del clasificador con diferentes valores del hiperparámetro k (número de vecinos). Además, permite identificar el mejor valor de k para cada dataset, asegurando así un ajuste óptimo del modelo a los datos. A continuación, se explica cada sección del código con mayor profundidad.

El código comienza importando las bibliotecas necesarias, que incluyen herramientas clave para manipulación de datos, entrenamiento de modelos y evaluación de su desempeño. numpy se utiliza para operaciones matemáticas y cálculos estadísticos, mientras que pandas es útil en escenarios donde se requiera trabajar con estructuras de datos más complejas, aunque en este caso no se utiliza directamente en el código. La biblioteca sklearn proporciona todas las herramientas necesarias para construir y validar modelos. Por ejemplo, las funciones train_test_split y cross_val_score son utilizadas para dividir los datos y realizar validaciones cruzadas. Por otro lado, las métricas como accuracy_score y confusion_matrix sirven para medir el desempeño del modelo, y los datasets estándar como load_iris, load_wine y load_breast_cancer permiten probar fácilmente los algoritmos de clasificación sin necesidad de recopilar datos adicionales. Finalmente, el clasificador KNeighborsClassifier es el modelo principal que se entrena y evalúa en este script.

Los datasets disponibles en sklearn son cargados en un diccionario llamado datasets, donde las claves son nombres descriptivos como "Iris", "Wine" y "Breast Cancer", y los valores son los datos correspondientes. Cada dataset está compuesto por dos elementos principales: data, que contiene las características numéricas de las observaciones, y target, que representa las etiquetas de las clases a las que pertenecen dichas observaciones. Este diseño hace que sea sencillo iterar sobre los datasets y aplicar los mismos procesos de entrenamiento y validación a cada uno, manteniendo el código limpio y modular.

La primera función implementada en el código, hold_out_validation, aplica el método de validación conocido como Hold-Out. Este enfoque divide el dataset en dos subconjuntos: uno de entrenamiento (70% de los datos) y otro de prueba (30% de los datos). La división se realiza de manera estratificada, utilizando la función train_test_split, lo que garantiza que la proporción de clases en los conjuntos de entrenamiento y prueba sea la misma que en el dataset original. Este paso es importante en problemas de clasificación, especialmente cuando las clases están desbalanceadas, ya que evita que el modelo esté sesgado hacia la clase mayoritaria. Una vez dividido el dataset, se entrena un modelo

KNN con un número específico de vecinos k, y este modelo se evalúa utilizando el conjunto de prueba. El desempeño se mide en términos de la precisión del modelo, calculada como la proporción de predicciones correctas, y también mediante una matriz de confusión que muestra los aciertos y errores del clasificador en cada clase. Esta validación proporciona una visión rápida y directa del rendimiento del modelo, aunque su desventaja es que depende fuertemente de cómo se realice la división inicial de los datos.

La función cross_validation implementa un enfoque más robusto llamado 10-Fold Cross-Validation. En este método, el dataset se divide en 10 subconjuntos (o "folds") aproximadamente del mismo tamaño. En cada iteración, uno de estos folds se utiliza como conjunto de prueba, mientras que los demás se combinan para formar el conjunto de entrenamiento. Este proceso se repite 10 veces, de manera que cada fold se utiliza exactamente una vez como conjunto de prueba. Para garantizar que cada fold mantenga la distribución original de clases, se utiliza StratifiedKFold, una técnica de estratificación que preserva las proporciones de las clases en cada división. En cada iteración, se entrena un modelo KNN con un número específico de vecinos k, y se calcula la precisión del modelo en el fold de prueba. Al final, se promedian las precisiones de las 10 iteraciones, lo que resulta en una métrica más confiable del rendimiento del modelo en comparación con el método Hold-Out, ya que utiliza todo el dataset para entrenamiento y prueba. Este enfoque reduce el riesgo de obtener resultados sesgados debido a una única división de los datos.

La función leave_one_out implementa una validación aún más exhaustiva llamada Leave-One-Out (LOO). En este caso, el dataset se divide tantas veces como el número de observaciones. En cada iteración, una sola muestra se utiliza como conjunto de prueba, mientras que el resto de los datos se utiliza para entrenar el modelo. Esto significa que el modelo se entrena y evalúa tantas veces como el número total de observaciones. Aunque este método proporciona una evaluación extremadamente precisa, ya que utiliza casi todo el dataset para entrenamiento en cada iteración, es computacionalmente costoso, especialmente para datasets grandes. Al final, se calcula la precisión promedio del modelo en todas las iteraciones, lo que ofrece una visión detallada de su rendimiento.

La función find_best_k aborda un problema crucial en el uso del algoritmo KNN: seleccionar el mejor valor para el hiperparámetro k, que representa el número de vecinos a considerar durante la clasificación. La función itera sobre valores de k desde 1 hasta 20 y evalúa el modelo para cada valor utilizando dos posibles métodos: **Hold-Out** o **10-Fold Cross-Validation**. Dependiendo del método seleccionado, se calcula la precisión del modelo y se almacena el valor de k que produce la mayor precisión. Este proceso permite encontrar el valor óptimo de k para cada dataset, ajustando el modelo a las características específicas de los datos.

Finalmente, el código principal itera sobre los datasets cargados y aplica los métodos descritos. Para cada dataset, primero busca el mejor valor de k utilizando la validación cruzada estratificada de 10 folds. Luego, evalúa el modelo con el mejor valor de k utilizando los tres métodos de validación: Hold-Out, 10-Fold Cross-Validation y Leave-One-Out. Cada evaluación imprime resultados detallados, como la precisión promedio, la matriz de confusión y otras métricas relevantes. Esto proporciona una visión integral del rendimiento del modelo bajo diferentes enfoques de validación, permitiendo comparar las ventajas y desventajas de cada uno.

Explicación de los resultados

Dataset: Iris

Este dataset es uno de los más utilizados en clasificación, con tres clases balanceadas que representan diferentes tipos de flores. Los resultados muestran un desempeño notablemente alto del modelo KNN:

1. Selección del mejor K: Usando validación cruzada (10-Fold), se determinó que el mejor número de vecinos k es 17, alcanzando una precisión promedio de 98%. Esto sugiere que un modelo con mayor cantidad de vecinos logra una clasificación más robusta en este caso.
2. Hold-Out (70/30):
    - Precisión: Con el mejor valor de k (17), el modelo logró una precisión del 95.56% en el conjunto de prueba. Esto indica que el modelo generaliza bien, pero ligeramente por debajo de la validación cruzada.
    - Matriz de confusión: El modelo clasificó correctamente la mayoría de las observaciones:
        - La clase 1 fue perfectamente clasificada (15 instancias correctas).
        - En la clase 2 hubo un error donde una muestra fue confundida con la clase 3.
        - En la clase 3 también hubo un error donde una muestra fue confundida con la clase 2.
3. 10-Fold Cross-Validation: La precisión promedio de 98% confirma que el modelo es muy confiable y consistente, ya que mantiene un desempeño excelente en cada uno de los folds.
4. Leave-One-Out: La precisión promedio de 97.33% muestra un resultado similar a 10-Fold, aunque ligeramente menor debido a la naturaleza más exhaustiva de este método. Esto reafirma que el modelo clasifica de manera precisa incluso con una muestra de prueba muy limitada.

Conclusión para Iris: El modelo KNN se desempeña excepcionalmente bien en este dataset, lo cual es esperable dado que el dataset Iris es relativamente simple y bien balanceado. La alta precisión en todos los métodos de validación respalda la fiabilidad del clasificador.

Dataset: Wine

El dataset Wine presenta mayor complejidad, con características más variadas y clases que no siempre están perfectamente separadas. Esto se refleja en los resultados, que muestran un desempeño moderado del modelo KNN:

1. Selección del mejor K: En este caso, el mejor valor de k fue 1, con una precisión promedio de 73% en 10-Fold Cross-Validation. Esto indica que considerar solo el vecino más cercano produce los mejores resultados, lo cual podría sugerir una alta variabilidad en las características entre muestras de la misma clase.
2. Hold-Out (70/30):
    - Precisión: En el conjunto de prueba, la precisión fue 70.37%, ligeramente inferior a la validación cruzada. Esto sugiere que la división inicial de los datos puede haber influido en el desempeño.
    - Matriz de confusión: Se observan errores en las tres clases:
        - La clase 1 tuvo 3 errores (clasificaciones incorrectas en las clases 2 y 3).
        - La clase 2 presentó 6 errores, principalmente confundida con la clase 3.
        - La clase 3 tuvo 6 errores, principalmente confundida con la clase 2. Esto indica que las clases 2 y 3 son más difíciles de separar, lo cual puede deberse a características que no diferencian suficientemente estas clases.
3. 10-Fold Cross-Validation: La precisión promedio de 73% muestra que el modelo tiene un desempeño relativamente consistente en los diferentes folds, aunque lejos de ser perfecto. Esto sugiere que el dataset presenta desafíos para la clasificación.
4. Leave-One-Out: La precisión promedio aumentó a 76.97%, lo que puede deberse a que LOO entrena el modelo con casi todo el dataset en cada iteración, mejorando ligeramente los resultados.

Conclusión para Wine: El rendimiento del modelo en el dataset Wine es moderado, con precisión máxima en torno al 76%. Los errores en la matriz de confusión sugieren que

las clases 2 y 3 son las más difíciles de distinguir. Este resultado refleja la naturaleza más compleja del dataset.

Dataset: Breast Cancer

El dataset Breast Cancer tiene características más definidas para separar las clases, lo que se traduce en un rendimiento alto del modelo KNN:

1. Selección del mejor K: El mejor valor de k fue 12, con una precisión promedio de 93.85% en 10-Fold Cross-Validation. Esto indica que un valor de k intermedio permite al modelo aprovechar mejor la estructura del dataset.
2. Hold-Out (70/30):
    - Precisión: La precisión en el conjunto de prueba fue 93.57%, muy cercana a la de la validación cruzada, lo que demuestra la capacidad del modelo para generalizar bien en este dataset.
    - Matriz de confusión: El modelo tuvo pocos errores:
        - La clase 0 tuvo 7 errores, donde las muestras fueron clasificadas como clase 1.
        - La clase 1 tuvo 4 errores, clasificadas incorrectamente como clase 0. Esto sugiere que la mayoría de los errores ocurren en los límites entre las dos clases.
3. 10-Fold Cross-Validation: La precisión promedio de 93.85% reafirma la consistencia del modelo, indicando que el rendimiento es estable en diferentes particiones del dataset.
4. Leave-One-Out: La precisión promedio fue 93.67%, casi idéntica a los otros métodos, lo que muestra que el modelo se desempeña bien incluso en configuraciones de validación más estrictas.

Conclusión para Breast Cancer: El modelo KNN se desempeña muy bien en el dataset Breast Cancer, con una precisión superior al 93% en todos los métodos de validación. Los errores son mínimos y están distribuidos de manera razonable entre las dos clases, lo que demuestra que el dataset es adecuado para este tipo de modelo.

Evidencias.

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split, cross_val_score, StratifiedKFold, LeaveOneOut
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.neighbors import KNeighborsClassifier
from sklearn.datasets import load_iris, load_wine, load_breast_cancer
```

```python
# Cargar los datasets
datasets = {
    "Iris": load_iris(),
    "Wine": load_wine(),
    "Breast Cancer": load_breast_cancer()
}
```

```python
# Hold-Out 70/30 Estratificado
def hold_out_validation(data, k):
    X, y = data.data, data.target
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, stratify=y, random_state=42)

    # Clasificador KNN
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train, y_train)

    # Predicción y métricas
    y_pred = knn.predict(X_test)
    acc = accuracy_score(y_test, y_pred)
    cm = confusion_matrix(y_test, y_pred)

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

```python
# 10-Fold Cross-Validation Estratificado
def cross_validation(data, k):
    X, y = data.data, data.target
    skf = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)

    # Clasificador KNN
    knn = KNeighborsClassifier(n_neighbors=k)
    scores = cross_val_score(knn, X, y, cv=skf)

    print("Accuracy promedio en 10-Fold:", np.mean(scores))
```

```python
# Leave-One-Out (LOO)
def leave_one_out(data, k):
    X, y = data.data, data.target
    loo = LeaveOneOut()

    # Clasificador KNN
    knn = KNeighborsClassifier(n_neighbors=k)
    scores = cross_val_score(knn, X, y, cv=loo)

    print("Accuracy promedio en LOO:", np.mean(scores))
```

```python
# Elección del mejor valor de K
def find_best_k(data, method="cross_val"):
    X, y = data.data, data.target
    best_k, best_score = 0, 0

    for k in range(1, 21):
        if method == "cross_val":
            skf = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)
            knn = KNeighborsClassifier(n_neighbors=k)
            score = np.mean(cross_val_score(knn, X, y, cv=skf))
        elif method == "hold_out":
            X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, stratify=y, random_state=42)
            knn = KNeighborsClassifier(n_neighbors=k)
            knn.fit(X_train, y_train)
            score = knn.score(X_test, y_test)

        if score > best_score:
            best_k, best_score = k, score

    print(f"Mejor valor de K: {best_k} con Accuracy: {best_score}")
    return best_k
```

```python
for name, data in datasets.items():
    print(f"\n--- Dataset: {name} ---")

    # Buscar mejor K usando Cross-Validation
    print("\nBuscando mejor K con 10-Fold Cross Validation:")
    best_k = find_best_k(data, method="cross_val")

    # Hold-Out Validation
    print("\nValidación Hold-Out 70/30 Estratificado:")
    hold_out_validation(data, best_k)

    # 10-Fold Cross-Validation
    print("\nValidación 10-Fold Cross-Validation Estratificado:")
    cross_validation(data, best_k)

    # Leave-One-Out
    print("\nValidación Leave-One-Out:")
    leave_one_out(data, best_k)
```

```
--- Dataset: Iris ---

Buscando mejor K con 10-Fold Cross Validation:
Mejor valor de K: 17 con Accuracy: 0.9800000000000001

Validación Hold-Out 70/30 Estratificado:
Accuracy: 0.9555555555555556
Matriz de Confusión:
 [[15  0  0]
 [ 0 14  1]
 [ 0  1 14]]

Validación 10-Fold Cross-Validation Estratificado:
Accuracy promedio en 10-Fold: 0.9800000000000001

Validación Leave-One-Out:
Accuracy promedio en LOO: 0.9733333333333334

--- Dataset: Wine ---

Buscando mejor K con 10-Fold Cross Validation:
Mejor valor de K: 1 con Accuracy: 0.7300653594771241

Validación Hold-Out 70/30 Estratificado:
Accuracy: 0.7037037037037037
Matriz de Confusión:
 [[14  3  1]
 [ 1 15  5]
 [ 1  5  9]]
```

```
Validación 10-Fold Cross-Validation Estratificado
Accuracy promedio en 10-Fold: 0.7300653594771241

Validación Leave-One-Out:
Accuracy promedio en LOO: 0.7696629213483146

--- Dataset: Breast Cancer ---

Buscando mejor K con 10-Fold Cross Validation:
```

```
Mejor valor de K: 12 con Accuracy: 0.9385025062656641

Validación Hold-Out 70/30 Estratificado:
Accuracy: 0.935672514619883
Matriz de Confusión:
 [[ 57   7]
 [  4 103]]

Validación 10-Fold Cross-Validation Estratificado:
Accuracy promedio en 10-Fold: 0.9385025062656641

Validación Leave-One-Out:
Accuracy promedio en LOO: 0.9367311072056239
```