INSTITUTO POLITECNICO NACIONAL

Escuela Superior de Computo

Materia: Machine Learning

Grupo: 5BV1

Práctica 7: Clasificador KNN

Introducción:

El algoritmo K Nearest Neighbors (KNN) es uno de los métodos de clasificación más simples y efectivos para problemas de clasificación y regresión. KNN es un algoritmo basado en instancias, lo que significa que no tiene un proceso de entrenamiento explícito, sino que almacena todos los ejemplos de entrenamiento y hace predicciones con base en la similitud entre las instancias. Para clasificar un punto nuevo, el algoritmo busca los k vecinos más cercanos en el conjunto de datos y decide la clase a partir de la mayoría de los votos de estos vecinos. La elección del valor de k es fundamental, ya que un valor pequeño puede llevar a un modelo sensible al ruido (overfitting), mientras que un valor grande puede diluir la información relevante (underfitting).

Desarrollo:

Importación de Librerías

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split, cross_val_score, StratifiedKFold, LeaveOneOut
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.datasets import load_wine
import seaborn as sns
import matplotlib.pyplot as plt
```

Las librerías importadas cumplen funciones específicas y fundamentales:

- **numpy y pandas**: Librerías usadas para manejar y procesar datos tabulares. numpy facilita las operaciones matemáticas y pandas permite trabajar con tablas de datos de manera eficiente.
- **sklearn.model_selection**: Proporciona herramientas como train_test_split para dividir conjuntos de datos y realizar validaciones cruzadas mediante StratifiedKFold y LeaveOneOut.
- **sklearn.preprocessing.StandardScaler**: Se usa para normalizar las características, asegurando que todas tengan una media de cero y una desviación estándar de uno. Esto es importante en KNN porque se basa en distancias.
- **sklearn.neighbors.KNeighborsClassifier**: Se utiliza para implementar el clasificador KNN.
- **sklearn.metrics**: Incluye métricas para evaluar el rendimiento del modelo, como accuracy_score y confusion_matrix.

- **seaborn y matplotlib**: Se utilizan para la visualización, especialmente para graficar la matriz de confusión y proporcionar una comprensión visual de los resultados.

Carga del Dataset de Calidad del Vino

```
def load_wine_quality():
    url = "https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv"
    df = pd.read_csv(url, sep=';')
    X = df.drop('quality', axis=1)
    y = df['quality']
    return X, y
```

La función **load_wine_quality** descarga el dataset de calidad del vino desde una fuente pública, lo carga en un DataFrame de pandas, y lo divide en X (características del vino) e y (calificación de la calidad). Esta estructura permite trabajar con las características y las etiquetas de manera separada.

Definición de los Datasets a Usar

```
wine = load_wine()
wine_quality_X, wine_quality_y = load_wine_quality()
seeds_url = "https://archive.ics.uci.edu/ml/machine-learning-databases/00236/seeds_dataset.txt"
seeds_columns = ["area", "perimeter", "compactness", "length_kernel", "width_kernel", "asymmetry_coef", "groove_length", "type"]
seeds = pd.read_csv(seeds_url, sep='\s+', names=seeds_columns)

# Cargar datos de Iris (de scikit-learn)
iris = load_wine()
iris_X, iris_y = iris.data, iris.target
```

En esta sección, se cargan tres datasets:

- **Dataset wine**: Utiliza el dataset integrado en scikit-learn relacionado con características del vino.
- **Dataset wine_quality**: Cargado mediante la función load_wine_quality, se usa para evaluar la calidad del vino según distintas características.
- **Dataset seeds**: Este dataset incluye características de semillas de diferentes tipos y se carga manualmente con nombres de columnas proporcionados.
- **Dataset iris**: Aunque se utiliza incorrectamente load_wine(), debería cargarse el dataset de Iris que contiene datos de clasificación de flores. Esta parte requiere una corrección.

Función Principal (main()) para Ejecutar los Métodos de Validación

```
def main():
    print("Elige el método de validación:")
    print("1. Hold-Out 70/30 Estratificado")
    print("2. 10 Fold Cross-Validation Estratificado")
    print("3. Leave-One-Out Cross-Validation")
    opcion = int(input("Ingresa el número de la opción deseada: "))

    datasets = [
        (wine_quality_X, wine_quality_y, "Wine Quality"),
        (seeds.drop('type', axis=1), seeds['type'], "Seeds"),
        (iris_X, iris_y, "Iris")
    ]
```

- **Selección del Método de Validación**: Se da al usuario la opción de elegir entre tres métodos de validación.

- **Lista de Datasets**: Agrupa los tres datasets que se usarán. Cada uno contiene las características (X), las etiquetas (y) y el nombre del dataset para referencia.

Procesamiento de Cada Dataset

```
    for X, y, dataset_name in datasets:
        print(f"\nProcesando dataset: {dataset_name}")

        # Escalar las características
        scaler = StandardScaler()
        X = scaler.fit_transform(X)

        # Crear el clasificador KNN
        knn = KNeighborsClassifier(n_neighbors=5)

        if opcion == 1:
            # Hold-Out 70/30 Estratificado
                X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
stratify=y, random_state=42)
            knn.fit(X_train, y_train)
            y_pred = knn.predict(X_test)
            accuracy = accuracy_score(y_test, y_pred)
            cm = confusion_matrix(y_test, y_pred)
            print(f"Accuracy: {accuracy}")
            print("Matriz de Confusión:")
```

```
print(cm)
sns.heatmap(cm, annot=True, cmap='Blues', fmt='g')
plt.title(f'Matriz de Confusión - Hold-Out ({dataset_name})')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```

- **Estandarización de los Datos**: Se aplica StandardScaler para normalizar los datos antes de aplicar el clasificador KNN, lo cual mejora el rendimiento del algoritmo.

- **Crear Clasificador KNN**: Se define el clasificador con k=5, aunque no hay un análisis previo para seleccionar este valor.

- **Validación Hold-Out (70/30 Estratificado)**:

    - Se divide el dataset en 70% para entrenamiento y 30% para prueba, asegurando la estratificación.

    - Se entrena el clasificador con los datos de entrenamiento y se calculan las predicciones sobre los datos de prueba.

    - Se imprime la precisión (accuracy) y la matriz de confusión. Esta última se muestra usando seaborn para una mejor comprensión visual.

Validación Cruzada (10-Fold y Leave-One-Out)

```
elif opcion == 2:
    # 10 Fold Cross-Validation Estratificado
    skf = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)
    scores = cross_val_score(knn, X, y, cv=skf)
    print(f"Accuracy promedio con 10 Fold Cross-Validation: {scores.mean()}")
    print(f"Desviación estándar de la accuracy: {scores.std()}")

elif opcion == 3:
    # Leave-One-Out Cross-Validation
    loo = LeaveOneOut()
    scores = cross_val_score(knn, X, y, cv=loo)
            print(f"Accuracy promedio con Leave-One-Out Cross-Validation: {scores.mean()}")
```

- **10 Fold Cross-Validation Estratificado**: Se usa StratifiedKFold para dividir el dataset en 10 particiones estratificadas. Se calcula la precisión promedio y la desviación estándar.

- **Leave-One-Out Cross-Validation**: Utiliza LeaveOneOut() para evaluar el modelo en cada observación. Esto puede ser muy costoso computacionalmente, pero proporciona una medida precisa del rendimiento del modelo.

Finalización

```
    else:
        print("Opción de validación no válida.")
        return

if __name__ == "__main__":
    main()
```

- **Verificación de la Opción de Validación**: Si el usuario selecciona una opción no válida, se muestra un mensaje de error.

- **Ejecución del Código**: Se llama a la función main() cuando el script se ejecuta directamente.

Conclusión:

El algoritmo KNN es un método de clasificación efectivo cuando se selecciona el valor adecuado de k y se normalizan las características del dataset. Este laboratorio permitió implementar KNN utilizando tres métodos distintos de validación, cada uno ofreciendo una perspectiva diferente del rendimiento del modelo. Aca vimos la importancia de la validación cruzada para si obtener el rendimiento mas confiable y la utilidad de ver mejor los errores travez de las matrices de confusión.