Laboratorio 8: Clasificador Naive Bayes

Alumno:

Miguel Angel Ocampo

Gerardo Martinez Ayala

Maestro:

Andres Garcia Floriano

Explicación de las técnicas

Las técnicas de validación implementadas en el código (*Hold-Out estratificado*, *10-Fold Cross-Validation* estratificado y *Leave-One-Out*) son fundamentales para evaluar el desempeño de un modelo de aprendizaje supervisado, como el clasificador **Naïve Bayes**. A continuación, se explican en detalle:

1. Hold-Out 70/30 Estratificado

Esta técnica divide el dataset en dos subconjuntos:

- **70% para entrenamiento**: Se utiliza para ajustar los parámetros del modelo.
- **30% para prueba**: Se utiliza para evaluar la capacidad del modelo de generalizar a datos no vistos.

La **estratificación** asegura que la proporción de clases (categorías) se mantenga igual tanto en el conjunto de entrenamiento como en el de prueba. Esto es crucial en datasets desbalanceados, donde algunas clases tienen muchas más observaciones que otras.

Ventajas:

- Fácil de implementar.
- Rápido y eficiente, especialmente en datasets grandes.

Desventajas:

- La evaluación puede ser sensible a cómo se hace la división (dependencia del azar).
- Utiliza solo una parte del dataset para evaluación, lo que puede introducir variabilidad.

2. 10-Fold Cross-Validation Estratificado

En esta técnica, el dataset se divide en 10 partes aproximadamente iguales (**folds**). Luego, se realizan 10 iteraciones donde:

- En cada iteración, se utilizan 9 folds para entrenar el modelo y 1 fold diferente para evaluar.
- Este proceso asegura que todas las observaciones del dataset se usen tanto para entrenamiento como para evaluación, pero en diferentes iteraciones.

La **estratificación** garantiza que cada fold mantenga la proporción original de las clases.

Ventajas:

- Más robusto que Hold-Out, ya que promedia el desempeño en diferentes particiones del dataset.
- Reduce la dependencia de una única división del conjunto de datos.

Desventajas:

- Más costoso computacionalmente que Hold-Out, ya que el modelo debe entrenarse 10 veces.
- El número de folds elegido puede influir en los resultados (aunque 10 es un estándar ampliamente aceptado).

3. Leave-One-Out (LOO)

Es un caso extremo de validación cruzada donde:

- Se usa **todo el dataset menos una instancia** para entrenar el modelo.
- La única instancia restante se utiliza como prueba.
- Este proceso se repite tantas veces como observaciones tenga el dataset, cambiando la instancia de prueba en cada iteración.

Al final, se promedian las métricas obtenidas en cada iteración.

Ventajas:

- Máxima utilización de datos para entrenamiento en cada iteración.
- Proporciona una evaluación muy exhaustiva, especialmente útil en datasets pequeños.

Desventajas:

- Muy costoso computacionalmente, ya que requiere entrenar el modelo tantas veces como observaciones tenga el dataset.
- Puede ser susceptible a overfitting en datasets con ruido o valores atípicos, ya que cada instancia tiene un peso desproporcionado.

Comparación de las técnicas

Técnica	Uso típico	Eficiencia computacional	Robustez	Mejor para
Hold-Out Estratificado	Evaluación inicial, exploración rápida	Alta (una sola división)	Baja a moderada	Datasets grandes
10-Fold Cross-Validation	Evaluación intermedia, preferida en la práctica	Moderada (10 entrenamientos)	Alta	Datasets de tamaño moderado o grande
Leave-One-Out	Evaluación exhaustiva en datasets pequeños	Baja (muy costoso)	Muy alta	Datasets pequeños

Explicación del código

Este código utiliza varias técnicas de validación para evaluar el desempeño del modelo Naïve Bayes Gaussiano sobre tres conjuntos de datos de ejemplo proporcionados por la biblioteca scikit-learn: Iris, Wine, y Breast Cancer. Las técnicas utilizadas incluyen Hold-Out, Validación Cruzada con 10 particiones estratificadas (10-Fold Cross-Validation), y Leave-One-Out (LOO). A continuación, se explica en detalle cada parte del código:

Importación de bibliotecas

El código comienza importando las bibliotecas necesarias:

- numpy: Para realizar operaciones matemáticas y de manipulación de datos (aunque en este caso no se usa directamente).
- scikit-learn: Proporciona herramientas para dividir datos, realizar validaciones cruzadas, calcular métricas de rendimiento, crear modelos de machine learning y cargar conjuntos de datos de prueba.

Las funciones específicas importadas son:

1. train_test_split: Para dividir los datos en conjuntos de entrenamiento y prueba.
2. StratifiedKFold: Realiza validación cruzada asegurando que las clases estén equilibradas en cada partición.

3. LeaveOneOut: Divide los datos dejando una muestra para prueba en cada iteración.
4. cross_val_score: Calcula métricas de rendimiento usando validación cruzada.
5. accuracy_score: Calcula la precisión del modelo.
6. confusion_matrix: Genera una matriz de confusión.
7. GaussianNB: Implementa el clasificador Naïve Bayes Gaussiano.
8. load_iris**,** load_wine**,** load_breast_cancer: Proporcionan conjuntos de datos de prueba para machine learning.

Carga de conjuntos de datos

Se utilizan tres conjuntos de datos: **Iris**, **Wine** y **Breast Cancer**. Cada uno contiene características y etiquetas predefinidas para clasificación supervisada.

- datasets almacena los tres datasets en una lista para iterar sobre ellos más adelante.

Iteración sobre los conjuntos de datos

El bucle for recorre cada conjunto de datos y realiza las siguientes acciones:

1. **Extracción de características y etiquetas**:
    - X representa las características (inputs).
    - y representa las etiquetas (outputs o clases).
    - dataset_name extrae el nombre del dataset de su descripción (DESCR).
2. **Definición del modelo**:
    - GaussianNB es el modelo de Naïve Bayes Gaussiano. Este clasificador asume que las características siguen una distribución normal.

Validación del modelo

Se emplean tres técnicas para evaluar el modelo:

1. **Hold-Out (70/30 estratificado)**:
    - Se divide el dataset en un 70% para entrenamiento y un 30% para prueba, respetando la proporción de las clases con stratify=y.

- El modelo se entrena con los datos de entrenamiento (model.fit).
- Se hacen predicciones con los datos de prueba (model.predict).
- La precisión se calcula con accuracy_score.
- La matriz de confusión se genera con confusion_matrix, mostrando el rendimiento del modelo en términos de verdaderos positivos, verdaderos negativos, falsos positivos y falsos negativos.

2. **Validación cruzada con 10 particiones estratificadas (10-Fold Cross-Validation)**:
    - Con StratifiedKFold(n_splits=10), el conjunto de datos se divide en 10 particiones, asegurando una distribución balanceada de clases en cada división.
    - cross_val_score calcula la precisión promedio del modelo a través de las 10 particiones.
3. **Leave-One-Out (LOO)**:
    - Con LeaveOneOut, cada muestra se utiliza como conjunto de prueba una vez, mientras que el resto sirve para entrenamiento.
    - Es una técnica intensiva en cómputo, especialmente para datasets grandes.
    - cross_val_score calcula la precisión promedio considerando cada partición de LOO.

Almacenamiento y presentación de resultados

Los resultados para cada conjunto de datos se almacenan en la lista results como diccionarios, conteniendo:

- Nombre del dataset.
- Precisión y matriz de confusión obtenidas con **Hold-Out**.
- Precisión promedio con **10-Fold Cross-Validation**.
- Precisión promedio con **Leave-One-Out**.

Finalmente, los resultados se imprimen para cada dataset en un formato claro y estructurado.

Código

Cargamos librerías y datasets, así mismo comenzamos con los procedimientos

```python
import numpy as np
from sklearn.model_selection import train_test_split, StratifiedKFold, LeaveOneOut, cross_val_score
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.naive_bayes import GaussianNB
from sklearn.datasets import load_iris, load_wine, load_breast_cancer

# Cargar datasets
datasets = [load_iris(), load_wine(), load_breast_cancer()]
results = []

for dataset in datasets:
    X, y = dataset.data, dataset.target
    dataset_name = dataset.DESCR.split("\n")[0]

    # Modelo Naïve Bayes
    model = GaussianNB()

    # 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)
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    acc_holdout = accuracy_score(y_test, y_pred)
    conf_matrix_holdout = confusion_matrix(y_test, y_pred)
```

Terminamos con los procedimientos y almacenamos nuestros resultados

```python
# 10-Fold Cross-Validation estratificado
skf = StratifiedKFold(n_splits=10)
acc_kfold = cross_val_score(model, X, y, cv=skf).mean()

# Leave-One-Out
loo = LeaveOneOut()
acc_loo = cross_val_score(model, X, y, cv=loo).mean()

# Almacenar resultados
results.append({
    "Dataset": dataset_name,
    "Accuracy Hold-Out": acc_holdout,
    "Confusion Matrix Hold-Out": conf_matrix_holdout,
    "Accuracy 10-Fold": acc_kfold,
    "Accuracy LOO": acc_loo
})
```

Imprimir los resultados

```
# Mostrar resultados
for result in results:
    print(f"Dataset: {result['Dataset']}")
    print(f"Accuracy Hold-Out: {result['Accuracy Hold-Out']}")
    print(f"Confusion Matrix Hold-Out:\n{result['Confusion Matrix Hold-Out']}")
    print(f"Accuracy 10-Fold: {result['Accuracy 10-Fold']}")
    print(f"Accuracy LOO: {result['Accuracy LOO']}")
    print("-" * 50)
```

Resultados

1. Desempeño del modelo por dataset

- **Hold-Out 70/30 estratificado**:
  - Este método mide la precisión del modelo en un conjunto de datos separados específicamente para prueba.
  - Resultados típicos muestran una mayor variabilidad en comparación con los métodos más robustos debido a la aleatoriedad en la división. Sin embargo, al ser estratificado, se mantiene representatividad en las proporciones de clases.
- **10-Fold Cross-Validation estratificado**:
  - Este método proporciona un resultado más estable, ya que se promedian las métricas de 10 iteraciones. Al ser estratificado, asegura que las clases están equilibradas en cada subconjunto.
  - Generalmente, es considerado más confiable que Hold-Out debido a la reducción de sesgos provocados por una única división.
- **Leave-One-Out (LOO)**:
  - Proporciona una evaluación muy exhaustiva al entrenar y validar el modelo en prácticamente todos los datos menos una instancia en cada iteración.
  - Debido a que utiliza casi toda la información del dataset para entrenar en cada paso, los resultados suelen ser los más optimistas, especialmente en datasets pequeños.

2. Comparación de las métricas entre métodos

- En general, se espera que la precisión calculada con 10-Fold Cross-Validation sea una estimación intermedia, mientras que la precisión de LOO puede ser ligeramente más alta debido a la maximización de los datos de entrenamiento.
- El Hold-Out, dependiendo del dataset, puede ser más bajo o comparable, pero es menos consistente.

3. Interpretación de la matriz de confusión

- La matriz de confusión del método Hold-Out permite observar la distribución de los errores de clasificación entre las clases. Esto puede dar pistas sobre:
  - Clases desbalanceadas.
  - Casos donde las clases se confunden con más frecuencia (posibles mejoras en el modelo o necesidad de transformación de características).

4. Impacto del dataset

- Datasets más simples, como *Iris* (líneas divisorias claras entre clases), suelen mostrar una precisión más alta en todos los métodos.
- Datasets más complejos, como *Wine* o *Breast Cancer* (especialmente con clases no linealmente separables), podrían dar lugar a un menor desempeño.
- El impacto de Naïve Bayes es mayor en datasets con características independientes; si estas están correlacionadas, el modelo puede ser menos efectivo.

5. Conclusión

- El método más adecuado de validación dependerá del tamaño del dataset:
  - 10-Fold Cross-Validation es generalmente preferido por su equilibrio entre robustez y costo computacional.
  - LOO puede ser útil en datasets muy pequeños.

- La precisión de Naïve Bayes debe ser comparada con otros modelos para evaluar su adecuación al problema específico.
- Si hay un sesgo sistemático en la matriz de confusión, pueden ser necesarias estrategias adicionales, como transformar características o usar modelos más complejos.