Práctica 4

Métodos de validación

Alumno:

Miguel Angel Ocampo Floriano

Gerardo Martinez Ayala

Maestro:

Andres Garcia Floriano

Introducción

La validación de modelos es un paso fundamental en el desarrollo de algoritmos de aprendizaje automático, ya que permite medir su desempeño y generalización en datos no vistos. Para este proyecto, utilizaremos al menos dos datasets provenientes del repositorio de UCI Machine Learning, asegurándonos de que al menos uno de ellos presente clases con tamaños desbalanceados. Esto permitirá evaluar el rendimiento de los métodos de validación en diferentes condiciones de los datos.

Descripción de las técnicas de validación:

1. Hold Out Validation: Esta técnica consiste en dividir el conjunto de datos en dos partes: un conjunto de entrenamiento y un conjunto de prueba, utilizando un porcentaje determinado para cada uno (por ejemplo, 70% para entrenamiento y 30% para prueba). Es una forma rápida de estimar el rendimiento de un modelo, aunque puede ser sensible a la manera en que los datos se dividen.
2. K-Fold Cross Validation: El conjunto de datos se divide en *K* subconjuntos (o "folds"). En cada iteración, un subconjunto diferente se utiliza como conjunto de prueba, mientras que los restantes *K-1* subconjuntos se utilizan como conjunto de entrenamiento. Este proceso se repite *K* veces, garantizando que cada dato haya sido utilizado para entrenamiento y prueba al menos una vez. Finalmente, se promedian los resultados de cada iteración para obtener una métrica más robusta.
3. Leave-One-Out Cross Validation (LOO): Similar a K-Fold, pero con el número de pliegues igual al número de observaciones en el conjunto de datos. En cada iteración, un solo dato actúa como conjunto de prueba, mientras que los demás datos forman el conjunto de entrenamiento. Es una técnica más exhaustiva, pero puede ser costosa en términos computacionales cuando el conjunto de datos es grande.

Conjuntos de datos utilizados:

- Iris: Un conjunto de datos clásico en Machine Learning, que contiene 150 observaciones de tres especies de flores de iris, con cuatro características numéricas para cada observación.
- Wine Quality: Un conjunto de datos más complejo que contiene 1599 observaciones de características físico-químicas de diferentes muestras de vino tinto, con el objetivo de predecir la calidad del vino (en una escala de 0 a 10).

Codigo

Este código utiliza diversas técnicas de validación para entrenar y evaluar modelos de aprendizaje automático en dos conjuntos de datos: **Iris**, que es muy utilizado en ejemplos de clasificación, y el **Wine Quality**, un conjunto de datos descargado desde el repositorio UCI, que contiene información sobre la calidad del vino. A continuación, te detallo cada parte del código con una explicación más extensa y detallada, sin mostrar el código como tal.

1. Importación de bibliotecas:

Para comenzar, se realiza la importación de las bibliotecas necesarias para llevar a cabo el análisis. Se usa `pandas`, una biblioteca para la manipulación y análisis de datos, la cual es fundamental para cargar y trabajar con el conjunto de datos de vino. Además, se importa `sklearn` (Scikit-learn), una biblioteca que incluye herramientas para tareas de aprendizaje automático, entre ellas, funciones para dividir los datos y aplicar técnicas de validación. Específicamente, se utilizan módulos como `train_test_split`, `KFold`, y `LeaveOneOut`, que implementan técnicas de validación cruzada. Finalmente, se importa `load_iris` para cargar el conjunto de datos Iris directamente desde Scikit-learn.

2. Cargar el conjunto de datos Iris:

El código carga el famoso conjunto de datos Iris, que contiene información sobre las características de tres tipos de flores. Este dataset se obtiene directamente desde la biblioteca Scikit-learn. Se carga en formato de `DataFrame`, que es una estructura de datos tabular ofrecida por `pandas`, y se separan las características (variables independientes) de las etiquetas objetivo (variable dependiente). En este caso, las características incluyen medidas como la longitud y anchura de los sépalos y pétalos, mientras que la variable objetivo indica a cuál de las tres especies de iris pertenece cada muestra.

3. **Cargar el conjunto de datos Wine Quality:

El segundo conjunto de datos se refiere a la **calidad del vino tinto**, y se descarga desde un repositorio en línea (UCI Machine Learning Repository). Este dataset incluye varias características que describen las propiedades físico-químicas del vino (como el pH, el ácido cítrico, el contenido de alcohol, entre otros), y una variable objetivo que representa la **calidad del vino** en una escala del 1 al 10. En este punto, se utiliza `pandas` para cargar el archivo CSV que contiene los datos, y luego se separan las características de la columna de calidad, que será la variable objetivo. Esto es crucial para entrenar modelos de predicción más adelante.

4. Validación Hold Out:

Se implementa una función que realiza la técnica de validación conocida como Hold Out. En esta técnica, el conjunto de datos se divide en dos partes: un conjunto de entrenamiento y un conjunto de prueba. La proporción de esta división se define por un parámetro (por ejemplo, se puede utilizar el 70% de los datos para el entrenamiento y el 30% restante para la prueba). Esta técnica es común y relativamente sencilla, pero tiene la limitación de que los resultados dependen de una única división de los datos, lo que podría no representar adecuadamente la variabilidad inherente al conjunto completo. En este caso, el código hace uso de una semilla aleatoria (`random_state`) para garantizar que la división sea reproducible cada vez que se ejecuta.

5. Validación K-Fold Cross Validation:

La segunda técnica implementada es la **validación cruzada K-Fold**, que es una metodología más robusta para evaluar modelos. En K-Fold, el conjunto de datos se divide en `K` subconjuntos o "folds". Durante el proceso de validación, se realizan `K` iteraciones, donde en cada iteración un fold se utiliza como conjunto de prueba y los `K-1` folds restantes se utilizan como conjunto de entrenamiento. Al finalizar todas las iteraciones, se promedian los resultados obtenidos en cada fold, lo que ofrece una mejor estimación del rendimiento del modelo al haber usado cada instancia del conjunto de datos tanto para entrenar como para probar el modelo. Esta técnica es especialmente útil porque reduce la varianza asociada a la división de los datos y proporciona una evaluación más estable del modelo.

El código incluye un parámetro para especificar cuántos folds (divisiones) se desean realizar, y también utiliza una semilla aleatoria para asegurar que el proceso sea repetible. En este caso, el código está configurado para realizar una validación con cinco folds (`K=5`), lo cual es una configuración bastante común, ya que ofrece un buen equilibrio entre el costo computacional y la fiabilidad de los resultados.

6. Validación Leave-One-Out Cross Validation (LOOCV):

El código también incluye la implementación de una técnica de validación llamada Leave-One-Out Cross Validation (LOOCV). Esta es una variante extrema de K-Fold, donde el número de folds es igual al número de observaciones en el conjunto de datos. En otras palabras, se crean tantos folds como instancias de datos, y en cada iteración solo una observación se utiliza como conjunto de prueba, mientras que el resto se utiliza para entrenar el modelo. Esto asegura que cada observación sea probada una vez, y cada iteración es extremadamente precisa en términos de la evaluación de la capacidad del modelo para generalizar. Sin embargo, LOOCV puede ser muy costoso en términos computacionales, ya que se requieren tantas iteraciones como el número de observaciones en el conjunto de datos. Debido a esto, el código limita el número de

iteraciones a cinco para evitar una salida extensa y pesada al momento de imprimir los resultados.

7. Aplicación de Hold Out en el dataset Iris:

La primera aplicación práctica de las técnicas de validación es sobre el conjunto de datos Iris utilizando la metodología Hold Out. El código divide los datos en un 70% para el entrenamiento y un 30% para la prueba. Luego, imprime el tamaño de cada conjunto, lo que permite ver cuántas muestras se utilizan en cada parte. Esta metodología es muy sencilla de implementar y rápida de ejecutar, pero no ofrece una evaluación tan robusta como otras técnicas debido a que la división depende de una única partición de los datos.

8. Aplicación de Hold Out en el dataset Wine Quality:

La siguiente aplicación de Hold Out es sobre el conjunto de datos de calidad del vino. De manera similar al conjunto de datos Iris, los datos se dividen en un 70% para el entrenamiento y un 30% para la prueba, y el tamaño de cada conjunto se imprime. Dado que este conjunto de datos es más grande que el de Iris, es interesante ver cuántas muestras hay en cada parte. En este caso, la validación Hold Out también puede resultar menos confiable debido a la posible variabilidad en la partición de los datos.

9. Aplicación de K-Fold en el dataset Wine Quality:

La validación K-Fold se aplica posteriormente al conjunto de datos de calidad del vino. Aquí se utiliza K=5, por lo que se generan cinco folds. En cada iteración, un fold se usa como conjunto de prueba y el resto como conjunto de entrenamiento, y se imprime el tamaño de los conjuntos en cada fold. Esta metodología es mucho más robusta que Hold Out, ya que reduce el riesgo de una partición sesgada de los datos, proporcionando una evaluación más representativa del rendimiento general del modelo.

10. Aplicación de Leave-One-Out en el dataset Iris:

Finalmente, se aplica la técnica LOOCV al conjunto de datos Iris. Dado que este conjunto de datos es pequeño, LOOCV es manejable en términos computacionales. Sin embargo, debido a que LOOCV genera una gran cantidad de iteraciones (una por cada observación

en el conjunto de datos), el código solo imprime las primeras cinco iteraciones para evitar que la salida sea demasiado extensa. Esto permite ver cómo se dividen los datos en cada iteración, donde una sola muestra se usa como prueba y todas las demás como entrenamiento.

Carga de librerías y de datasets

```
import pandas as pd
from sklearn.model_selection import train_test_split, KFold, LeaveOneOut
from sklearn.datasets import load_iris

# Cargar el dataset Iris
iris = load_iris(as_frame=True)
X_iris = iris.data
y_iris = iris.target

# Cargar el dataset Wine Quality desde UCI
wine = pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv', sep=';')
X_wine = wine.drop('quality', axis=1)  # Todas las características menos la columna 'quality'
y_wine = wine['quality']  # Columna objetivo
```

Aplicación de los métodos de validación

```
# Función para Hold Out Validation
def hold_out_validation(X, y, r):
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=r, random_state=42)
    return X_train, X_test, y_train, y_test

# Función para K-Fold Cross Validation
def k_fold_validation(X, y, K):
    kf = KFold(n_splits=K, shuffle=True, random_state=42)
    for train_index, test_index in kf.split(X):
        X_train, X_test = X.iloc[train_index], X.iloc[test_index]
        y_train, y_test = y.iloc[train_index], y.iloc[test_index]
        yield X_train, X_test, y_train, y_test

# Función para Leave-One-Out Cross Validation
def leave_one_out_validation(X, y):
    loo = LeaveOneOut()
    for train_index, test_index in loo.split(X):
        X_train, X_test = X.iloc[train_index], X.iloc[test_index]
        y_train, y_test = y.iloc[train_index], y.iloc[test_index]
        yield X_train, X_test, y_train, y_test
```

Impresión de resultados

```python
# Uso con el dataset Iris (Hold Out)
X_train, X_test, y_train, y_test = hold_out_validation(X_iris, y_iris, 0.3)
print("Hold Out - Iris dataset:")
print(f"Training set size: {len(X_train)}")
print(f"Test set size: {len(X_test)}\n")

# Uso con el dataset Wine Quality (Hold Out)
X_train, X_test, y_train, y_test = hold_out_validation(X_wine, y_wine, 0.3)
print("Hold Out - Wine Quality dataset:")
print(f"Training set size: {len(X_train)}")
print(f"Test set size: {len(X_test)}\n")

# Uso con el dataset Wine Quality (K-Fold)
print("K-Fold - Wine Quality dataset:")
for i, (X_train, X_test, y_train, y_test) in enumerate(k_fold_validation(X_wine, y_wine, 5), 1):
    print(f"Fold {i}: Training set size: {len(X_train)}, Test set size: {len(X_test)}")

# Uso con Leave-One-Out (Iris dataset)
print("\nLeave-One-Out - Iris dataset (first 5 iterations):")
for i, (X_train, X_test, y_train, y_test) in enumerate(leave_one_out_validation(X_iris, y_iris)):
    if i >= 5:  # Mostrar solo las primeras 5 iteraciones para no saturar la salida
        break
    print(f"Iteration {i + 1}: Training set size: {len(X_train)}, Test set size: {len(X_test)}")
```

Resultados

Los datos que proporcionas se refieren a los resultados de aplicar tres técnicas de validación diferentes (Hold Out, K-Fold y Leave-One-Out) a dos conjuntos de datos: Iris y Wine Quality. A continuación, analizo cada conjunto de resultados, destacando las características relevantes de cada técnica y sus implicaciones.

1. Hold Out en el conjunto de datos Iris**

- Training set size: 105 (70% de los datos)

- Test set size: 45 (30% de los datos)

El conjunto de datos Iris contiene 150 observaciones en total. En la validación Hold Out, el 70% de los datos se asigna al conjunto de entrenamiento y el 30% al conjunto de prueba, resultando en 105 muestras para entrenar el modelo y 45 muestras para probarlo. Esta es una división estándar en validación Hold Out, y es efectiva cuando el

conjunto de datos no es muy pequeño. Sin embargo, puede ser vulnerable a la variabilidad en los resultados si la división particular no es representativa de todo el conjunto de datos.

2. Hold Out en el conjunto de datos Wine Quality

- Training set size: 1119 (70% de los datos)

- Test set size: 480 (30% de los datos)

El conjunto de datos Wine Quality contiene 1599 observaciones en total. Al aplicar la técnica Hold Out, también se usa una división 70%-30%, lo que deja 1119 muestras para entrenar el modelo y 480 para probarlo. El mayor tamaño de este conjunto de datos implica que la evaluación será más estable que en el caso del conjunto de datos Iris, debido a que más datos están involucrados tanto en el entrenamiento como en la prueba. Sin embargo, como con cualquier validación Hold Out, los resultados pueden depender significativamente de cómo se realizó la partición específica, ya que solo se realiza una división única.

3. K-Fold en el conjunto de datos Wine Quality (K = 5)

- Cada fold tiene aproximadamente 1280 observaciones para entrenar y 320 para probar.

En este caso, se ha utilizado una validación K-Fold con **5 folds**. Esto significa que el conjunto de datos se ha dividido en 5 partes (folds), y cada fold se utiliza como conjunto de prueba una vez, mientras que los otros 4 folds se usan para el entrenamiento en esa iteración.

- En 4 folds, el tamaño del conjunto de entrenamiento es 1279 (4/5 del total de 1599 observaciones), mientras que el conjunto de prueba tiene 320 muestras.

- En 1 fold (Fold 5), el conjunto de prueba tiene 319 muestras y el conjunto de entrenamiento tiene 1280.

La pequeña diferencia en el tamaño del fold 5 se debe a que el total de observaciones (1599) no es divisible exactamente por 5, lo que provoca una ligera variación en el último fold.

Este enfoque es más robusto que Hold Out porque utiliza todas las observaciones tanto para entrenamiento como para prueba en algún momento, lo que reduce el riesgo de depender de una única división de los datos. Además, como el conjunto de prueba se cambia en cada iteración, proporciona una evaluación más confiable del rendimiento del modelo en general.

4. Leave-One-Out en el conjunto de datos Iris**

- En cada iteración, el conjunto de entrenamiento tiene 149 observaciones y el conjunto de prueba tiene solo 1.

Leave-One-Out Cross Validation (LOOCV) es una versión extrema de la validación cruzada, en la que en cada iteración solo “una observación” se utiliza como conjunto de prueba, mientras que todas las demás se usan para entrenar el modelo. Dado que el conjunto de datos Iris tiene 150 observaciones, se realizan 150 iteraciones en total, cada una con un tamaño de conjunto de prueba de 1 y un tamaño de conjunto de entrenamiento de 149.

Este método tiene la ventaja de utilizar la mayor cantidad posible de datos para el entrenamiento en cada iteración, lo que tiende a maximizar la capacidad del modelo para aprender de los datos. Sin embargo, es computacionalmente más costoso, ya que requiere tantas iteraciones como el número total de observaciones. Además, la variabilidad en las particiones de prueba puede ser alta, lo que puede hacer que los resultados en una iteración no representen con precisión el rendimiento general del modelo.

Para evitar imprimir una salida demasiado extensa, el código solo muestra las primeras 5 iteraciones de LOOCV. En cada una de estas iteraciones, el conjunto de prueba contiene 1 observación y el conjunto de entrenamiento los 149 restantes.

Comparación entre los enfoques:

- Hold Out es la técnica más simple y rápida, pero depende de una única partición de los datos, lo que puede no representar bien la diversidad del conjunto de datos completo.

- K-Fold es más robusto, ya que realiza múltiples particiones, utilizando todo el conjunto de datos tanto para entrenamiento como para prueba en diferentes momentos. El tamaño constante de los conjuntos de entrenamiento y prueba en cada fold asegura una evaluación más equilibrada.

- Leave-One-Out maximiza el uso de los datos para el entrenamiento, pero puede ser costoso en cuanto a tiempo de cómputo. Además, aunque cada observación se prueba una vez, este método puede ser propenso a sobreajustes en conjuntos de datos pequeños.

En el caso del conjunto de datos Iris, **LOOCV** es una opción viable debido a su pequeño tamaño, mientras que **K-Fold** puede ser más práctico en el conjunto de datos más grande como Wine Quality.

Resultados impresos

```
Hold Out - Iris dataset:
Training set size: 105
Test set size: 45

Hold Out - Wine Quality dataset:
Training set size: 1119
Test set size: 480

K-Fold - Wine Quality dataset:
Fold 1: Training set size: 1279, Test set size: 320
Fold 2: Training set size: 1279, Test set size: 320
Fold 3: Training set size: 1279, Test set size: 320
Fold 4: Training set size: 1279, Test set size: 320
Fold 5: Training set size: 1280, Test set size: 319

Leave-One-Out - Iris dataset (first 5 iterations):
Iteration 1: Training set size: 149, Test set size: 1
Iteration 2: Training set size: 149, Test set size: 1
Iteration 3: Training set size: 149, Test set size: 1
Iteration 4: Training set size: 149, Test set size: 1
Iteration 5: Training set size: 149, Test set size: 1
```

Conclusiones

En conclusión, el análisis comparativo de las técnicas de validación aplicadas a los conjuntos de datos Iris y Wine Quality destaca las fortalezas y limitaciones de cada enfoque en función del tamaño del conjunto de datos y la complejidad de la evaluación.

La técnica Hold Out, aunque simple y eficiente, puede ser insuficiente para capturar la variabilidad del rendimiento del modelo, ya que se basa en una única partición de los datos. Esto es particularmente relevante en conjuntos de datos más pequeños como Iris, donde la división puede no ser representativa. Sin embargo, en conjuntos más grandes como Wine Quality, el tamaño del conjunto de datos de prueba y entrenamiento en Hold Out ofrece más estabilidad en los resultados.

La validación K-Fold proporciona una evaluación más robusta al dividir los datos en múltiples folds, lo que garantiza que cada observación se utilice tanto para el entrenamiento como para la prueba. Este enfoque reduce la varianza en los resultados y es adecuado para conjuntos de datos de tamaño intermedio, como Wine Quality, donde los datos son lo suficientemente grandes como para justificar múltiples particiones sin un aumento considerable en el costo computacional.

Por otro lado, Leave-One-Out Cross Validation (LOOCV) maximiza el uso de los datos para el entrenamiento, lo que es ventajoso en conjuntos de datos pequeños como Iris. Aunque es computacionalmente más costoso, este método evalúa el modelo de manera exhaustiva al usar cada observación como prueba una vez. No obstante, en conjuntos de datos grandes, LOOCV puede ser impráctico debido al elevado número de iteraciones necesarias y al potencial de sobreajuste en modelos muy ajustados a los datos.