INSTITUTO POLITÉCTICO NACIONAL

ESCUELA SUPERIOR DE COMPUTO

INTELIGENCIA ARTIFICAL

EJERCICIO DE LABORATORIO 9. MÉTODOS DE VALIDACIÓN

PROFESOR:
GARCÍA FLORIANO ANDRÉS

ALUMNOS:

OLGUÍN REYES IVÁN ALEJANDRO

ORTEGA PRADO MAURICIO

RUIZ ALAMILLA MAURICIO EMILIANO

GRUPO: 6CV2

Ejercicio de laboratorio 9.

Introducción.

Hold Out estratificado.
El método Hold Out estratificado es una variante del método Hold Out estándar que se utiliza cuando se trabaja con conjuntos de datos desequilibrados o cuando se quiere asegurar que la distribución de clases en los conjuntos de entrenamiento y prueba sea representativa de la distribución en el conjunto de datos completo.

funcionamiento del método Hold Out estratificado:

1. División de datos estratificada: En lugar de realizar una división simple aleatoria de los datos en conjuntos de entrenamiento y prueba, se divide el conjunto de datos de forma estratificada. Esto significa que se mantiene la proporción de cada clase (o estrato) en ambos conjuntos.

2. Selección aleatoria dentro de cada estrato: Dentro de cada estrato, los datos se seleccionan aleatoriamente para formar tanto el conjunto de entrenamiento como el conjunto de prueba. La proporción de datos seleccionados de cada estrato se mantiene constante en ambos conjuntos.

3. Entrenamiento del modelo: Se utiliza el conjunto de entrenamiento estratificado para entrenar el modelo predictivo.

4. Evaluación del modelo: Se utiliza el conjunto de prueba estratificado para evaluar el rendimiento del modelo. Esto asegura que la evaluación del modelo refleje adecuadamente la capacidad de generalización del modelo en datos de todas las clases.

10-Fold Cross-Validation estratificado.
Es una técnica avanzada utilizada para evaluar el rendimiento de un modelo de manera más robusta y asegurando que la distribución de las clases se mantenga constante en cada pliegue. Este método es particularmente útil cuando se trabaja con conjuntos de datos desequilibrados.

División del conjunto de datos: El conjunto de datos completo se divide en 10 partes (pliegues) aproximadamente iguales. La división se hace de manera estratificada, lo que significa que cada pliegue tendrá aproximadamente la misma proporción de cada clase presente en el conjunto de datos original.

Proceso iterativo de validación cruzada:

- Iteración 1: Se toma el primer pliegue como el conjunto de prueba y los nueve pliegues restantes se combinan para formar el conjunto de entrenamiento.
- Entrenamiento: Se entrena el modelo usando el conjunto de entrenamiento (los nueve pliegues).
- Evaluación: Se evalúa el modelo usando el conjunto de prueba (el primer pliegue) y se registran las métricas de rendimiento.
- Iteración 2: Se toma el segundo pliegue como el conjunto de prueba y los otros nueve pliegues se usan para el entrenamiento. Se repiten los pasos de entrenamiento y evaluación.
- ...
- Iteración 10: El proceso se repite hasta que cada uno de los 10 pliegues ha sido usado una vez como conjunto de prueba.
- Promedio de los resultados: Después de completar las 10 iteraciones, se promedian las métricas de rendimiento obtenidas en cada iteración para obtener una estimación final del rendimiento del modelo.

Desarrollo.

- Hold Out 70/30 estratificado.

```
import pandas as pd
from sklearn.model_selection import train_test_split
import numpy as np
```

Se importan las bibliotecas necesarias: pandas para la manipulación de datos, train_test_split de scikit-learn para dividir los datos, y numpy para operaciones numéricas.

```
X = data[['sepal.length', 'sepal.width', 'petal.length',
'petal.width']].values
y = data['variety'].values
```

Se seleccionan las características (X) y la etiqueta (y) del conjunto de datos. X contiene las características numéricas de las flores y Y contiene las variedades de las flores.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
stratify=y, random_state=42)
```

Los datos se dividen en conjuntos de entrenamiento (70%) y prueba (30%) de manera estratificada, lo que asegura que las proporciones de las clases sean similares en ambos conjuntos. random_state se usa para asegurar reproducibilidad.

```
unique, counts_train = np.unique(y_train, return_counts=True)
unique, counts_test = np.unique(y_test, return_counts=True)
print("Proporciones en el conjunto de entrenamiento:", counts_train /
len(y_train))
print("Proporciones en el conjunto de prueba:", counts_test / len(y_test))
```

Se verifican las proporciones de las clases en los conjuntos de entrenamiento y prueba para asegurar que la división estratificada ha mantenido las proporciones originales de las clases.

```
train_df = pd.DataFrame(X_train, columns=['sepal.length', 'sepal.width',
'petal.length', 'petal.width'])
train_df['variety'] = y_train

test_df = pd.DataFrame(X_test, columns=['sepal.length', 'sepal.width',
'petal.length', 'petal.width'])
test_df['variety'] = y_test
```

Se crean DataFrames a partir de los conjuntos de entrenamiento y prueba, incluyendo tanto las características como las etiquetas.

```
train_df.to_csv('train_data.csv', index=False)
test_df.to_csv('test_data.csv', index=False)
```

Los DataFrames se guardan en archivos CSV llamados train_data.csv y test_data.csv respectivamente, sin incluir los índices.

Pruebas:

1.- La primera prueba se realizó con el conjunto de datos (dataset) flor iris, este conjunto de datos. El dataset consta de 150 observaciones de flores de iris de tres especies diferentes: Iris setosa, Iris versicolor e Iris virginica. Cada observación incluye cuatro características:

- Longitud del sépalo (Sepal length)
- Ancho del sépalo (Sepal width)
- Longitud del pétalo (Petal length)
- Ancho del pétalo (Petal width)

Para el correcto funcionamiento en nuestro programa tenemos que especificar la ruta del archivo .csv y las características como las clases o etiquetas:

```
data = pd.read_csv(r'C:\Users\mauri\Documents\7MO SEMESTRE\INTELIGENCIA ARTIFICIAL\Practica 9\iris.csv')

# Columnas 'sepal.length', 'sepal.width', 'petal.length', 'petal.width'
# Etiqueta/clase en una columna llamada 'variety'
X = data[['sepal.length', 'sepal.width', 'petal.length', 'petal.width']].values
y = data['variety'].values
```

Después de estas especificaciones, podemos compilar y ejecutar el programa, el programa imprimirá el siguiente mensaje:

```
Proporciones en el conjunto de entrenamiento: [0.33333333 0.33333333 0.33333333]
Proporciones en el conjunto de prueba: [0.33333333 0.33333333 0.33333333]
Conjuntos de entrenamiento y prueba guardados en 'train_data.csv' y 'test_data.csv' respectivamente.
```

y podremos abrir y observar los conjuntos generados que se adjuntan en la entrega.

2.- La segunda prueba se realizó con el conjunto wine contiene información química sobre diferentes muestras de vino cultivadas en la misma región de Italia, pero derivadas de tres diferentes cultivares (variedades de uva).
Cada observación en el dataset incluye las siguientes 13 características químicas:

- Alcohol
- Ácido málico (Malic acid)
- Ceniza (Ash)
- Alcalinidad de la ceniza (Ash alkalinity)
- Magnesio
- Fenoles totales (Total phenols)
- Flavonoides
- Fenoles no flavonoides (Nonflavonoid phenols)
- Proantocianidinas (Proanthocyanidins)
- Intensidad del color (Color intensity)
- Tono (Hue)
- OD280/OD315 de vinos diluidos
- Prolina

Se dividen en tres clases: 1, 2 ,3 y son un total de 178 muestras.
Para el correcto funcionamiento en nuestro programa tenemos que especificar la ruta del archivo .csv y las características como las clases o etiquetas:

```
X = data[['Alcohol', 'Malic.acid', 'Ash', 'Acl', 'Mg', 'Phenols',
          'Flavanoids', 'Nonflavanoid.phenols', 'Proanth', 'Color.int', 'Hue',
          'OD', 'Proline']].values
y = data['Wine'].values
```

Después de estas especificaciones, podemos compilar y ejecutar el programa, el programa imprimirá el siguiente mensaje:

```
Proporciones en el conjunto de entrenamiento: [0.33064516 0.40322581
0.26612903]
Proporciones en el conjunto de prueba: [0.33333333 0.38888889 0.27777778]
Conjuntos de entrenamiento y prueba guardados en 'train_data_wine.csv' y
'test_data_wine.csv' respectivamente.
```

y podremos abrir y observar los conjuntos generados que se adjuntan en la entrega.

3.- El tercer dataset que se eligió para las pruebas fue el diabetes. El dataset proviene del Instituto Nacional de Diabetes y Enfermedades Digestivas y Renales de los Estados Unidos. Los datos fueron recolectados por el Servicio de Salud de los Estados Unidos de las mujeres de la tribu Pima, en Arizona, una población que tiene una alta incidencia de diabetes.

Características del Dataset:

Número de muestras: 768
Número de características (atributos): 8
Número de clases: 2 (diabetes ‘1’ o no diabetes ‘0’)

Para el correcto funcionamiento en nuestro programa tenemos que especificar la ruta del archivo .csv y las características como las clases o etiquetas:

```
X = data[['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness',
'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age']].values
y = data['Outcome'].values
```

Después de estas especificaciones, podemos compilar y ejecutar el programa, el programa imprimirá el siguiente mensaje:

```
Proporciones en el conjunto de entrenamiento: [0.65176909 0.34823091]
Proporciones en el conjunto de prueba: [0.64935065 0.35064935]
Conjuntos de entrenamiento y prueba guardados en 'train_data_diabetes.csv' y
'test_data_diabetes.csv' respectivamente.
```

y podremos abrir y observar los conjuntos generados que se adjuntan en la entrega.

- 10-Fold Cross-Validation estratificado.

```
import pandas as pd
from sklearn.model_selection import StratifiedKFold
import numpy as np
```

Se importan las bibliotecas necesarias: pandas para la manipulación de datos, StratifiedkFold de scikit-learn para dividir los datos, y numpy para operaciones numéricas.

```
# Cargar datos desde un archivo CSV
data = pd.read_csv(r'C:\Users\Mauricio\Documents\7 SEMESTRE\INTELIGENCIA ARTIFICIAL\Ejercicio Lab 9\iris.csv')
# Imprimir los nombres de las columnas para verificar
print(data.columns)
# Separar características y etiquetas utilizando los nombres exactos de las columnas
X = data[['sepal.length', 'sepal.width', 'petal.length', 'petal.width']].values
y = data['variety'].values
```

Se carga el archivo “.csv” indicando la ruta de dicho archivo, se imprime el nombre de las columnas para verificar si son correctos o hacer algún cambio. Una vez verificando los nombres de las columnas podemos separar entre las características como las clases.

```
# Configurar StratifiedKFold con 10 folds
skf = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)

# Lista para almacenar los índices de cada fold
fold_indices = list(skf.split(X, y))

# Verificar que las proporciones de las clases son similares en cada fold
for i, (train_index, test_index) in enumerate(fold_indices):
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]

    unique_train, counts_train = np.unique(y_train, return_counts=True)
    unique_test, counts_test = np.unique(y_test, return_counts=True)
```

Se configura el StratifiedKFold con 10 folds, en una lista se guardaran los índices de cada fold, posteriormente se verifican las proporciones de las clases para que sean similares en cada fold.

```
# Crear DataFrames para cada fold y guardarlos en archivos CSV
for fold, (train_index, test_index) in enumerate(fold_indices):
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]

     train_df = pd.DataFrame(X_train, columns=['sepal.length', 'sepal.width',
'petal.length', 'petal.width'])
    train_df['variety'] = y_train

      test_df = pd.DataFrame(X_test, columns=['sepal.length', 'sepal.width',
'petal.length', 'petal.width'])
    test_df['variety'] = y_test

    train_df.to_csv(f'train_data_fold_{fold+1}.csv', index=False)
    test_df.to_csv(f'test_data_fold_{fold+1}.csv', index=False)

print("Conjuntos de entrenamiento y prueba para cada fold guardados en
archivos CSV.")
```

Por último se crean los dataframes de cada Fold y son guardados en archivos “.csv”

Pruebas:

1.- La primera prueba se realizó con el dataset de iris.csv, este fue el resultado:

```
old 1
Proporciones en el conjunto de entrenamiento: [0.33333333 0.33333333
0.33333333]
Proporciones en el conjunto de prueba: [0.33333333 0.33333333 0.33333333]

Fold 2
Proporciones en el conjunto de entrenamiento: [0.33333333 0.33333333
0.33333333]
Proporciones en el conjunto de prueba: [0.33333333 0.33333333 0.33333333]

Fold 3
Proporciones en el conjunto de entrenamiento: [0.33333333 0.33333333
0.33333333]
```

```
Proporciones en el conjunto de prueba: [0.33333333 0.33333333 0.33333333]

Fold 4
Proporciones en el conjunto de entrenamiento: [0.33333333 0.33333333
0.33333333]
Proporciones en el conjunto de prueba: [0.33333333 0.33333333 0.33333333]

Fold 5
Proporciones en el conjunto de entrenamiento: [0.33333333 0.33333333
0.33333333]
Proporciones en el conjunto de prueba: [0.33333333 0.33333333 0.33333333]

Fold 6
Proporciones en el conjunto de entrenamiento: [0.33333333 0.33333333
0.33333333]
Proporciones en el conjunto de prueba: [0.33333333 0.33333333 0.33333333]

Fold 7
Proporciones en el conjunto de entrenamiento: [0.33333333 0.33333333
0.33333333]
Proporciones en el conjunto de prueba: [0.33333333 0.33333333 0.33333333]

Fold 8
Proporciones en el conjunto de entrenamiento: [0.33333333 0.33333333
0.33333333]
Proporciones en el conjunto de prueba: [0.33333333 0.33333333 0.33333333]

Fold 9
Proporciones en el conjunto de entrenamiento: [0.33333333 0.33333333
0.33333333]
Proporciones en el conjunto de prueba: [0.33333333 0.33333333 0.33333333]

Fold 10
Proporciones en el conjunto de entrenamiento: [0.33333333 0.33333333
0.33333333]
Proporciones en el conjunto de prueba: [0.33333333 0.33333333 0.33333333]

Conjuntos de entrenamiento y prueba para cada fold guardados en archivos CSV.
```

Los archivos de cada fold se adjuntan en la entrega de la práctica.

2.- Resultados con el dataset wine.csv:

```
Fold 1
Proporciones en el conjunto de entrenamiento: [0.33125 0.4     0.26875]
Proporciones en el conjunto de prueba: [0.33333333 0.38888889 0.27777778]

Fold 2
Proporciones en el conjunto de entrenamiento: [0.33125 0.4     0.26875]
Proporciones en el conjunto de prueba: [0.33333333 0.38888889 0.27777778]

Fold 3
Proporciones en el conjunto de entrenamiento: [0.33125 0.4     0.26875]
Proporciones en el conjunto de prueba: [0.33333333 0.38888889 0.27777778]

Fold 4
Proporciones en el conjunto de entrenamiento: [0.33125 0.4     0.26875]
Proporciones en el conjunto de prueba: [0.33333333 0.38888889 0.27777778]

Fold 5
Proporciones en el conjunto de entrenamiento: [0.33125 0.4     0.26875]
Proporciones en el conjunto de prueba: [0.33333333 0.38888889 0.27777778]

Fold 6
Proporciones en el conjunto de entrenamiento: [0.33125 0.4     0.26875]
Proporciones en el conjunto de prueba: [0.33333333 0.38888889 0.27777778]

Fold 7
Proporciones en el conjunto de entrenamiento: [0.33125 0.4     0.26875]
Proporciones en el conjunto de prueba: [0.33333333 0.38888889 0.27777778]

Fold 8
Proporciones en el conjunto de entrenamiento: [0.33125 0.4     0.26875]
Proporciones en el conjunto de prueba: [0.33333333 0.38888889 0.27777778]

Fold 9
Proporciones en el conjunto de entrenamiento: [0.32919255 0.39751553
0.27329193]
Proporciones en el conjunto de prueba: [0.35294118 0.41176471 0.23529412]

Fold 10
Proporciones en el conjunto de entrenamiento: [0.33540373 0.39130435
0.27329193]
Proporciones en el conjunto de prueba: [0.29411765 0.47058824 0.23529412]
```

```
Conjuntos de entrenamiento y prueba para cada fold guardados en archivos CSV.
```

3.- Resultados con el dataset diabetes.csv:

```
Fold 1
Proporciones en el conjunto de entrenamiento: [0.6512301 0.3487699]
Proporciones en el conjunto de prueba: [0.64935065 0.35064935]

Fold 2
Proporciones en el conjunto de entrenamiento: [0.6512301 0.3487699]
Proporciones en el conjunto de prueba: [0.64935065 0.35064935]

Fold 3
Proporciones en el conjunto de entrenamiento: [0.6512301 0.3487699]
Proporciones en el conjunto de prueba: [0.64935065 0.35064935]

Fold 4
Proporciones en el conjunto de entrenamiento: [0.6512301 0.3487699]
Proporciones en el conjunto de prueba: [0.64935065 0.35064935]

Fold 5
Proporciones en el conjunto de entrenamiento: [0.6512301 0.3487699]
Proporciones en el conjunto de prueba: [0.64935065 0.35064935]

Fold 6
Proporciones en el conjunto de entrenamiento: [0.6512301 0.3487699]
Proporciones en el conjunto de prueba: [0.64935065 0.35064935]

Fold 7
Proporciones en el conjunto de entrenamiento: [0.6512301 0.3487699]
Proporciones en el conjunto de prueba: [0.64935065 0.35064935]

Fold 8
Proporciones en el conjunto de entrenamiento: [0.6512301 0.3487699]
Proporciones en el conjunto de prueba: [0.64935065 0.35064935]

Fold 9
Proporciones en el conjunto de entrenamiento: [0.65028902 0.34971098]
Proporciones en el conjunto de prueba: [0.65789474 0.34210526]

Fold 10
Proporciones en el conjunto de entrenamiento: [0.65028902 0.34971098]
```

```
Proporciones en el conjunto de prueba: [0.65789474 0.34210526]

Conjuntos de entrenamiento y prueba para cada fold guardados en archivos CSV.
```

Conclusión.

En la evaluación de modelos de aprendizaje automático, el método Hold Out 70/30 estratificado y la validación cruzada estratificada de 10 pliegues ofrecen diferentes ventajas y desventajas. El método Hold Out 70/30 es sencillo y rápido de implementar, dividiendo los datos en una sola partición de entrenamiento y prueba. Esto es beneficioso en términos de tiempo y recursos, especialmente cuando se trabaja con grandes conjuntos de datos. Sin embargo, este método puede sufrir de alta varianza en los resultados y dependencia de cómo se realiza la partición inicial, lo que puede llevar a una evaluación menos precisa y representativa del modelo.

Por otro lado, la validación cruzada estratificada de 10 pliegues proporciona una estimación más robusta y estable del rendimiento del modelo al promediar los resultados obtenidos de diez particiones diferentes. Este método hace un uso más completo de los datos y asegura que cada pliegue tiene una representación proporcional de cada clase, lo cual es crucial para conjuntos de datos con clases desequilibradas. Aunque es más costoso en términos de tiempo y recursos computacionales, la precisión y fiabilidad adicionales que ofrece suelen justificar su uso.