INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

INGENIERIA EN SISTEMAS COMPUTACIONALES

Ejercicio de laboratorio 6

“Métodos de validación”

Presentan

Gómez Hernández, Alan Javier

Hernández Pérez, Juan Manuel

Jiménez Cruz, Daniel

Rivas Carrera, Diana Laura

Profesor

Andrés Floriano García

Octubre 2023

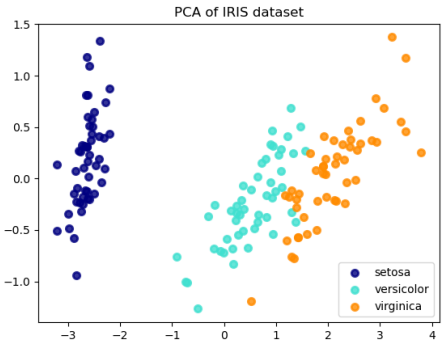
INTRODUCCIÓN

Conjunto de datos

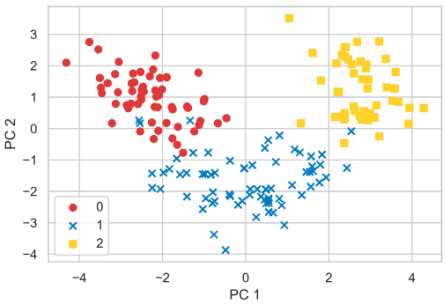
Un conjunto de datos (o "dataset") es una colección estructurada de datos. Normalmente, cuando nos referimos a un dataset en el contexto de machine learning o ciencia de datos, estamos hablando de una estructura tabular de datos donde cada **fila** representa una instancia, ejemplo o registro individual y cada **columna** representa una característica, atributo o variable de ese registro.

Muchos datasets están disponibles públicamente para investigación y desarrollo en diversas áreas del machine learning y la ciencia de datos. En esta práctica se usan los siguientes datasets ampliamente conocidos:

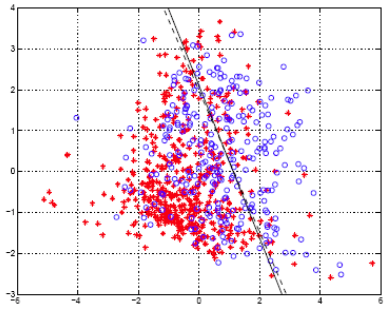
Dataset Iris: Contiene 150 observaciones de flores Iris divididas en 3 clases (setosa, versicolor y virginica), con 50 observaciones por clase. Cada observación tiene 4 características que representan medidas físicas de las flores: longitud del sépalo, ancho del sépalo, longitud del pétalo y ancho del pétalo. Es comúnmente utilizado para tareas de clasificación y clustering.



Dataset Wine: Contiene 178 muestras con 13 características químicas como el contenido de alcohol, la cantidad de magnesio, la intensidad del color, entre otros. Hay tres clases, cada una correspondiente a un tipo de vino. Se utiliza principalmente para clasificación y análisis de componentes principales (PCA).



Dataset Diabetes: Contiene 442 muestras con 10 características base como edad, sexo, índice de masa corporal, presión arterial y seis mediciones de suero sanguíneo. La variable de respuesta es una medida cuantitativa de la progresión de la enfermedad un año después de la línea base. Es ampliamente utilizado para tareas de clasificación binaria



Métodos de validación

Los métodos de validación son técnicas utilizadas para evaluar el rendimiento de un modelo en datos que no ha visto durante el entrenamiento. Esto es esencial para garantizar que el modelo no solo memorice los datos de entrenamiento (sobreajuste) sino que pueda generalizar bien a nuevos datos no vistos.

Conjunto de entrenamiento: Se utiliza un subset del dataset para entrenar modelos de machine learning. Durante este proceso, el modelo intenta aprender patrones en los datos.

Conjunto de validación y prueba: Otros subsets del dataset se utilizan para validar y probar el modelo, evaluando su rendimiento en datos que no se utilizaron durante el entrenamiento.

Al dividir el dataset en conjuntos de entrenamiento, validación y prueba, se puede ayudar a garantizar que el modelo no solo memorice los datos de entrenamiento, sino que también generalice bien a nuevos datos.

Estratificación

En machine learning, se refiere específicamente a la técnica utilizada para garantizar que los subconjuntos de datos (como los conjuntos de entrenamiento y prueba) tengan aproximadamente la misma proporción de ejemplos de cada clase como el conjunto de datos original.

Esto es especialmente importante en situaciones donde las clases están desequilibradas, es decir, cuando una clase tiene muchos más ejemplos que otra. Sin estratificación, podríamos terminar con un conjunto de prueba que tiene muy pocos (o ninguno) ejemplos de la clase minoritaria, lo que podría dar una representación engañosa del rendimiento del modelo.

Validación Holdout

El "holdout" es una técnica comúnmente utilizada en el aprendizaje automático y la estadística para evaluar el rendimiento de un modelo predictivo, como un modelo de regresión o clasificación. La idea detrás del holdout es dividir el conjunto de datos en dos partes: un conjunto de entrenamiento (training set) y un conjunto de prueba (test set).

- El conjunto de entrenamiento se utiliza para poder entrenar el modelo, lo que permite que el algoritmo de aprendizaje se ajuste a los parámetros y se adapte a los datos
- El conjunto de prueba es para evaluar el rendimiento del modelo después de que este se ha entrenado. Se simula para saber cómo se comportaría el modelo frente a nuevos datos que no se vieron durante el entrenamiento

Validación Cruzada (K-Folds)

Es una técnica comúnmente utilizada en el aprendizaje automático y la estadística para evaluar el rendimiento de un modelo predictivo y mitigar los sesgos de la división de datos en conjuntos de entrenamiento y prueba mediante el "holdout" tradicional. El objetivo principal de la validación cruzada de k-fold es obtener una estimación más robusta y precisa del rendimiento del modelo, ya que se evalúa en múltiples conjuntos de prueba y conjuntos de entrenamiento diferentes. Esto es especialmente útil cuando se dispone de un conjunto de datos limitado, ya que permite utilizar todos los datos tanto para entrenar como para evaluar el modelo.

DESARROLLO

Se llevaron a cabo métodos de validación como **Holdout** y **K-Folds** sobre los conjuntos de datos de *Iris*, *Wine* y *Diabetes*. Para la manipulación y carga de datos tabulares, se empleó la biblioteca "pandas". Además, se hizo uso de la biblioteca estándar "os", que ofrece funciones para interactuar con el sistema operativo.

Código: Holdout

El primer código se enfoca en la división de datos en conjuntos de entrenamiento y prueba. Se carga el dataset y se especifica la variable objetivo de este. Posteriormente, se procede a aleatorizarlos, lo que garantiza que no haya sesgos en la selección de los conjuntos de entrenamiento y prueba. Los datos se dividen en conjuntos de entrenamiento y prueba siguiendo una proporción predeterminada (80% para entrenamiento y 20% para prueba). Se realizan verificaciones para garantizar que los conjuntos generados sean disjuntos y tengan la misma proporción de clases. Finalmente, los conjuntos de entrenamiento y prueba se guardan en archivos separados.

```
import pandas as pd
import os

dataset_name = input("¿Cuál es el nombre del conjunto de datos? ")
# Comprobar que el conjunto de datos existe
while not os.path.exists(f"{dataset_name}.csv"):
    print(f"\n[x] El conjunto de datos '{dataset_name}' no existe.")
    dataset_name = input("¿Cuál es el nombre del conjunto de datos? ")

# Cargar los datos desde el archivo CSV
data = pd.read_csv(f"{dataset_name}.csv")
# Asegurarse de que los datos estén aleatorizados
data = data.sample(frac=1, random_state=0)
# Definir la proporción para la división
train_ratio = 0.8  # 70% para entrenamiento, 30% para prueba
# Declaramos los conjuntos de entrenamiento y prueba
train_data = pd.DataFrame()
test_data = pd.DataFrame()

target = input("¿Cuál es la variable objetivo? ")
# Comprobar que la variable objetivo existe en el conjunto de datos
```

```
while target not in data.columns:
    print(f"\n[x] La variable objetivo '{target}' no existe en el
conjunto de datos.")
    target = input("¿Cuál es la variable objetivo? ")

# Dividir los datos en conjuntos de entrenamiento y prueba según la
proporción
for _, group_data in data.groupby(target):
    # Obtener el número de ejemplos para cada especie
    n = len(group_data)
    # Calcular el número de ejemplos para cada conjunto
    n_train = int(n * train_ratio)
    n_test = n - n_train
    # Dividir los datos en conjuntos de entrenamiento y prueba
    train_data = pd.concat([train_data, group_data.iloc[:n_train]])
    test_data = pd.concat([test_data, group_data.iloc[n_train:]])

# Comprobar que los conjuntos generados son disjuntos
intersection = set(train_data.index) & set(test_data.index)
if len(intersection) == 0:
    print("\n[*] Los conjuntos de entrenamiento y prueba son disjuntos.")
else:
    print("\n[x] Los conjuntos de entrenamiento y prueba NO son
disjuntos.")

# Comprobar que los conjuntos generados tienen la misma proporción de
clases
train_class_distribution = train_data[target].value_counts()
test_class_distribution = test_data[target].value_counts()

# Mostrar la distribución de clases en el conjunto de entrenamiento en
porcentaje y en número de ejemplos

print("\nDistribución de clases en el conjunto de entrenamiento:")
for class_name, count in train_class_distribution.items():
    print(f"{class_name}: {count} ({count / len(train_data) *
100:.2f}%)")

print("\nDistribución de clases en el conjunto de prueba:")
```

```python
for class_name, count in test_class_distribution.items():
    print(f"{class_name}: {count} ({count / len(test_data) * 100:.2f}%)")

# Guardar los conjuntos de entrenamiento y prueba en archivos CSV
train_data.to_csv(f"{dataset_name}_train.csv", index=False)
test_data.to_csv(f"{dataset_name}_test.csv", index=False)
```

Pruebas de funcionamiento: Holdout

- **Iris Dataset**

```
¿Cuál es el nombre del conjunto de datos? iris
¿Cuál es la variable objetivo? species

[*] Los conjuntos de entrenamiento y prueba son disjuntos.

Distribución de clases en el conjunto de entrenamiento:
setosa: 40 (33.33%)
versicolor: 40 (33.33%)
virginica: 40 (33.33%)

Distribución de clases en el conjunto de prueba:
setosa: 10 (33.33%)
versicolor: 10 (33.33%)
virginica: 10 (33.33%)
```

- **Diabetes dataset**

```
¿Cuál es el nombre del conjunto de datos? diabetes
¿Cuál es la variable objetivo? Outcome

[*] Los conjuntos de entrenamiento y prueba son disjuntos.

Distribución de clases en el conjunto de entrenamiento:
0: 400 (65.15%)
1: 214 (34.85%)

Distribución de clases en el conjunto de prueba:
0: 100 (64.94%)
1: 54 (35.06%)
```

- **Wine dataset**

```
¿Cuál es el nombre del conjunto de datos? wine
¿Cuál es la variable objetivo? Wine

[*] Los conjuntos de entrenamiento y prueba son disjuntos.

Distribución de clases en el conjunto de entrenamiento:
2: 56 (39.72%)
1: 47 (33.33%)
3: 38 (26.95%)

Distribución de clases en el conjunto de prueba:
2: 15 (40.54%)
1: 12 (32.43%)
3: 10 (27.03%)

Process finished with exit code 0
```

Código: K-Folds

El siguiente código solicita al usuario el nombre de un archivo CSV y la variable objetivo dentro de ese archivo. Luego, carga el conjunto de datos, lo mezcla aleatoriamente y divide estratificadamente el conjunto en 10 pliegues (folds) basándose en la variable objetivo. La estratificación asegura que cada pliegue tenga una distribución similar de clases. Finalmente, muestra la distribución de clases en cada pliegue y guarda cada uno de ellos en archivos CSV separados dentro de un directorio llamado "folds".

```
import pandas as pd
import os

# Entrada para el nombre del conjunto de datos
dataset_name = input("¿Cuál es el nombre del conjunto de datos? ")
# Comprobar que el conjunto de datos existe
while not os.path.exists(f"{dataset_name}.csv"):
    print(f"\n[x] El conjunto de datos '{dataset_name}' no
existe.")
    dataset_name = input("¿Cuál es el nombre del conjunto de
datos? ")
# Cargar los datos desde el archivo CSV
```

```
data = pd.read_csv(f"{dataset_name}.csv")
# Asegurarse de que los datos estén aleatorizados
data = data.sample(frac=1, random_state=0)

# Entrada para la variable objetivo
target = input("¿Cuál es la variable objetivo? ")
# Comprobar que la variable objetivo existe en el conjunto de
datos
while target not in data.columns:
    print(f"\n[x] La variable objetivo '{target}' no existe en el
conjunto de datos.")
    target = input("¿Cuál es la variable objetivo? ")

# Número de pliegues
k = 10
# Crear directorio para guardar los pliegues
if not os.path.exists("folds"):
    os.makedirs("folds")

# Lista para almacenar los pliegues
folds = [pd.DataFrame()] * k

# Dividir cada grupo (especie) en k pliegues estratificados
for _, group in data.groupby(target):
    group_size = len(group)
    fold_size = group_size // k
    remainder = group_size % k
    fold_indices = []
    start_index = 0

    for j in range(k):
        end_index = start_index + fold_size + (1 if j < remainder
else 0)
        fold_indices.append((start_index, end_index))
        start_index = end_index

    # Añadir las muestras de la especie actual a los pliegues
correspondientes
    for j, (start, end) in enumerate(fold_indices):
```

```
        folds[j] = pd.concat([folds[j], group.iloc[start:end]])

# Función para mostrar la distribución de clases en un conjunto de datos
def display_class_distribution(data, target, set_name):
    class_distribution = data[target].value_counts()
    proportions = [f"{class_name}: {count} ({count / len(data) * 100:.2f}%)" for class_name, count in class_distribution.items()]
    print(f"{set_name}:  {'  '.join(proportions)}")

# Mostrar la distribución de clases en cada fold
for i, fold_data in enumerate(folds):
    display_class_distribution(fold_data, target, f"Fold {i+1}")

# Guardar los pliegues en archivos CSV
for i, fold_data in enumerate(folds):
    fold_data.to_csv(f"folds/{dataset_name}_fold_{i + 1}.csv", index=False)

print(f"\n[*] {k} pliegues se han guardado en archivos CSV en el directorio 'folds'.")
```

Pruebas de funcionamiento: K-folds

- **Iris dataset**

```
¿Cuál es el nombre del conjunto de datos? iris
¿Cuál es la variable objetivo? species
Fold 1:  setosa: 5 (33.33%)  versicolor: 5 (33.33%)  virginica: 5 (33.33%)
Fold 2:  setosa: 5 (33.33%)  versicolor: 5 (33.33%)  virginica: 5 (33.33%)
Fold 3:  setosa: 5 (33.33%)  versicolor: 5 (33.33%)  virginica: 5 (33.33%)
Fold 4:  setosa: 5 (33.33%)  versicolor: 5 (33.33%)  virginica: 5 (33.33%)
Fold 5:  setosa: 5 (33.33%)  versicolor: 5 (33.33%)  virginica: 5 (33.33%)
Fold 6:  setosa: 5 (33.33%)  versicolor: 5 (33.33%)  virginica: 5 (33.33%)
Fold 7:  setosa: 5 (33.33%)  versicolor: 5 (33.33%)  virginica: 5 (33.33%)
Fold 8:  setosa: 5 (33.33%)  versicolor: 5 (33.33%)  virginica: 5 (33.33%)
Fold 9:  setosa: 5 (33.33%)  versicolor: 5 (33.33%)  virginica: 5 (33.33%)
Fold 10:  setosa: 5 (33.33%)  versicolor: 5 (33.33%)  virginica: 5 (33.33%)

[*] 10 pliegues se han guardado en archivos CSV en el directorio 'folds'.

Process finished with exit code 0
```

- **Diabetes dataset**

```
¿Cuál es el nombre del conjunto de datos? diabetes
¿Cuál es la variable objetivo? Outcome
Fold 1:  0: 50 (64.94%)  1: 27 (35.06%)
Fold 2:  0: 50 (64.94%)  1: 27 (35.06%)
Fold 3:  0: 50 (64.94%)  1: 27 (35.06%)
Fold 4:  0: 50 (64.94%)  1: 27 (35.06%)
Fold 5:  0: 50 (64.94%)  1: 27 (35.06%)
Fold 6:  0: 50 (64.94%)  1: 27 (35.06%)
Fold 7:  0: 50 (64.94%)  1: 27 (35.06%)
Fold 8:  0: 50 (64.94%)  1: 27 (35.06%)
Fold 9:  0: 50 (65.79%)  1: 26 (34.21%)
Fold 10:  0: 50 (65.79%)  1: 26 (34.21%)

[*] 10 pliegues se han guardado en archivos CSV en el directorio 'folds'.

Process finished with exit code 0
```

- **Wine dataset**

```
¿Cuál es el nombre del conjunto de datos? wine
¿Cuál es la variable objetivo? Wine
Fold 1:  2: 8 (42.11%)  1: 6 (31.58%)  3: 5 (26.32%)
Fold 2:  2: 7 (38.89%)  1: 6 (33.33%)  3: 5 (27.78%)
Fold 3:  2: 7 (38.89%)  1: 6 (33.33%)  3: 5 (27.78%)
Fold 4:  2: 7 (38.89%)  1: 6 (33.33%)  3: 5 (27.78%)
Fold 5:  2: 7 (38.89%)  1: 6 (33.33%)  3: 5 (27.78%)
Fold 6:  2: 7 (38.89%)  1: 6 (33.33%)  3: 5 (27.78%)
Fold 7:  2: 7 (38.89%)  1: 6 (33.33%)  3: 5 (27.78%)
Fold 8:  2: 7 (38.89%)  1: 6 (33.33%)  3: 5 (27.78%)
Fold 9:  2: 7 (41.18%)  1: 6 (35.29%)  3: 4 (23.53%)
Fold 10:  2: 7 (43.75%)  1: 5 (31.25%)  3: 4 (25.00%)

[*] 10 pliegues se han guardado en archivos CSV en el directorio 'folds'.

Process finished with exit code 0
```

CONCLUSIÓN

Durante esta práctica se abordó la importancia y la aplicación de métodos de validación en el aprendizaje automático, destacando su papel crucial para evaluar el rendimiento de un modelo predictivo en datos que no se utilizaron durante el entrenamiento. Utilizando conjuntos de datos representativos en el campo, como Iris, Wine y Diabetes, se aplicaron dos técnicas de validación principales: Holdout y K-Folds. Es fundamental que estos métodos se apliquen adecuadamente para garantizar que los modelos de machine learning sean capaces de generalizar y no solo memorizar el conjunto de datos con el que fueron entrenados. La estratificación es una consideración esencial en este proceso, asegurando que haya una representación equitativa de las clases en los diferentes conjuntos generados, evitando sesgos y representaciones erróneas del rendimiento del modelo.

REFERENCIAS

DataScientist (2023, 25 de septiembre) *What is a dataset? How do I work with it?* Recuperado el 24 de octubre de 2023, de: https://datascientest.com/en/what-is-a-dataset-how-do-i-work-with-it

Meon, S. (2020, 6 de diciembre) *Stratified sampling in Machine Learning.* Recuperado el 24 de octubre de 2023, de: https://medium.com/analytics-vidhya/stratified-sampling-in-machine-learning-f5112b5b9cfe

Joby, A. (2021, 21 de julio). *What Is Cross-Validation? Comparing Machine Learning Models.* Recuperado el 24 de octubre de 2023, de: https://learn.g2.com/cross-validation

Devil, K (2023, 14 de agosto). *Understanding Hold-Out Methods for Training Machine Learning Models.* Recuperado el 24 de octubre de 2023, de: https://www.comet.com/site/blog/understanding-hold-out-methods-for-training-machine-learning-models