INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO
INGENIERIA EN SISTEMAS COMPUTACIONALES

Ejercicio de laboratorio 7
"Clasificadores"

Presentan
Gómez Hernández, Alan Javier
Hernández Pérez, Juan Manuel
Jiménez Cruz, Daniel
Rivas Carrera, Diana Laura

Profesor
Andrés García Floriano

Noviembre 2023

Índice

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 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) sinoque pueda generalizar bien a nuevos datos no vistos.

Validación Holdout

El término "Hold-Out" se refiere a una técnica comúnmente utilizada en la validación de modelos de machine learning y estadísticas para evaluar el rendimiento de un modelo. Esta técnica implica dividir el conjunto de datos en dos subconjuntos: un conjunto de entrenamiento (70%) y un conjunto de prueba (30%).

- 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 losdatos
- 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 anuevos datos que no se vieron durante el entrenamiento

Validación Cruzada (K-Folds)

Es una técnica comúnmente utilizada en el aprendizaje automático y la estadísticapara 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.

Modelo Clasificador Euclidiano

El "Clasificador Euclidiano" es un método simple de clasificación en el campo de la estadística y el aprendizaje automático. Se basa en el cálculo de la distancia euclidiana entre un punto de datos a clasificar y los puntos de datos de entrenamiento pertenecientes a diferentes clases

Tiene algunas limitaciones. Por ejemplo, asume que todas las características (dimensiones) tienen igual importancia, lo que puede no ser cierto en todos los casos. Además, es sensible a la escala de las características y puede no funcionar bien en espacios de alta dimensionalidad.

Modelo 1NN (1 Nearest Neighbor)

El modelo 1NN (1 Nearest Neighbor o Vecino Más Cercano) es un algoritmo de clasificación en el campo del aprendizaje automático y la estadística que se basa en la idea de clasificar un punto de datos nuevo según la etiqueta de clase del punto de datos más cercano en el conjunto de entrenamiento. Es un enfoque simple pero efectivo para la clasificación de patrones

Tiene algunas limitaciones. Por ejemplo, asume que todas las características (dimensiones) tienen igual importancia, lo que puede no ser cierto en todos los casos. Además, es sensible a la escala de las características y puede no funcionar bien en espacios de alta dimensionalidad.

- Entrenamiento: Recopilación de ejemplos de datos de diferentes clases con etiquetas de clase. No se realiza un entrenamiento formal del modelo.
- Clasificación: Cuando se presenta un nuevo punto de datos que se debe clasificar:

    - Se calcula la distancia (generalmente la distancia euclidiana) entre el punto a clasificar y todos los puntos de datos de entrenamiento.
    - Se busca el punto de entrenamiento más cercano al punto a clasificar en función de la distancia calculada.
    - El punto de datos a clasificar se asigna a la misma clase que el punto de entrenamiento más cercano, adoptando su etiqueta de clase.

DESARROLLO

En el desarrollo de esta práctica utilizaremos el clasificador euclidiano para analizar los datasets de Iris y Diabetes, aplicando los métodos de validación cruzada K-folds y Holdout que revisamos en la práctica anterior. Nuestro objetivo será evaluar la precisión (accuracy) de este enfoque. Posteriormente, replicaremos el proceso utilizando el método 1NN (el vecino más cercano) para comparar los resultados y determinar la efectividad de ambos métodos en la clasificación de estos datasets.

I. Clasificador Euclidiano

A. Holdout

El código presentado implementa una versión simplificada de clasificación basada en la proximidad a centroides. Al iniciar, solicita al usuario el nombre de un conjunto de datos y la variable objetivo, para cargar archivos CSV correspondientes a los datos de entrenamiento y prueba. Luego, separa las características y las etiquetas de ambos conjuntos, calcula los centroides de las clases en el conjunto de entrenamiento y utiliza estos centroides para clasificar los puntos en el conjunto de prueba mediante la distancia euclidiana. Finalmente, evalúa la precisión de la clasificación comparando las predicciones con las etiquetas reales del conjunto de prueba y muestra el resultado en porcentaje.

1. Implementación

```
import pandas as pd
import numpy as np


def classify_point(point, centroids):
    # Calcular la distancia euclidiana entre el punto y cada centroide
    # linalg = linear algebra, norm = norma, axis=1 para calcular la norma de
cada fila
    distancias = np.linalg.norm(centroids - point, axis=1)
    # Obtener el índice del centroide más cercano
    return centroids.index[distancias.argmin()]


# Comprobar que el conjunto de datos existe
try:
    dataset_name = input("¿Cuál es el nombre del conjunto de datos? ")
```

```
    # Cargar los datos desde el archivo CSV
    train_data = pd.read_csv(f"holdout/{dataset_name}_train.csv")
    test_data = pd.read_csv(f"holdout/{dataset_name}_test.csv")
    target = input("¿Cuál es la variable objetivo? ")
    # Obtener los conjuntos de entrenamiento y prueba
    X_train = train_data.drop(columns=[target])
    y_train = train_data[target]

    X_test = test_data.drop(columns=[target])
    y_test = test_data[target]
except FileNotFoundError:
    print(f"\n[x] No se encontraron los datos ingrsados.")
    exit()

# Entrenar el clasificador, es decir, calcular los centroides
centroids = X_train.groupby(y_train).mean()

print("\nCentroides:")
print(centroids)

# Calcular la distancia euclidiana entre cada ejemplo de prueba y cada centroide
y_pred = X_test.apply(classify_point, axis=1, centroids=centroids)

# Calcular la precisión del clasificador
accuracy = (y_pred == y_test).mean()
print(f"\nPrecisión: {accuracy * 100:.2f}%")
```

2. Resultados

a) Conjunto de datos Iris

Como podemos ver en la siguiente imagen, con el dataset **Iris**, dividiéndolo con la tecnica de **Holdout** y usando el clasificador **Euclideano** se obtuvo un *accuracy* del **95.56%**

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

Centroides:
            sepal_length  sepal_width  petal_length  petal_width
species
setosa          4.960000     3.408571      1.468571     0.245714
versicolor      5.894286     2.771429      4.271429     1.337143
virginica       6.434286     2.925714      5.408571     2.000000

Precisión: 95.56%
```

b) Conjunto de datos Diabetes

Con el dataset **Diabetes**, dividiéndolo con la técnica de **Holdout** y usando el clasificador **Euclidiano** se obtuvo un *accuracy* del **60.61%**

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

Centroides:
         Pregnancies     Glucose  ...  DiabetesPedigreeFunction        Age
Outcome                           ...
0           3.268571  110.077143  ...                  0.428360  31.377143
1           4.850267  142.663102  ...                  0.550615  37.262032

[2 rows x 8 columns]

Precisión: 60.61%
```

B. K-Folds

Este código ejecuta un procedimiento de validación cruzada en un clasificador basado en la distancia euclidiana a centroides. Pide al usuario que introduzca un nombre de conjunto de datos y una variable objetivo, cargando un conjunto de archivos CSV previamente divididos en pliegues (folds) para la validación cruzada. Utiliza todos menos uno de los pliegues para entrenar el clasificador, calculando los centroides de las clases, y el pliegue restante para probar la precisión del modelo. Este proceso se repite para cada pliegue, de modo que cada uno se utiliza una vez como conjunto de prueba. La función `classify_point` se utiliza para asignar cada punto del conjunto de prueba al centroide más cercano, y se calcula la precisión de las predicciones en cada iteración. Al final, se imprime la precisión promedio del clasificador a través de todos los pliegues de validación cruzada.

1. Implementación

```
import pandas as pd
import numpy as np

def classify_point(point, centroids):
    # Calcular la distancia euclidiana entre el punto y cada centroide
    # linalg = linear algebra, norm = norma, axis=1 para calcular la norma de
cada fila
    distancias = np.linalg.norm(centroids - point, axis=1)
    # Obtener el índice del centroide más cercano
    return centroids.index[distancias.argmin()]


def cross_validate(folds, target):
    accuracies = []
    num_folds = len(folds)
    for i in range(num_folds):
        # Obtener los conjuntos de entrenamiento y prueba
        # El conjunto de entrenamiento es la unión de todos los folds excepto el
i-ésimo
        try:
            train_data = pd.concat([folds[j] for j in range(num_folds) if j !=
i])
            X_train = train_data.drop(columns=[target])
```

```
            y_train = train_data[target]

            # El conjunto de prueba es el i-ésimo fold
            test_data = folds[i]
            X_test = test_data.drop(columns=[target])
            y_test = test_data[target]
        except KeyError:
            print(f"\n[x] La variable objetivo {target} no existe en los
datos.")
            exit()

        # Entrenar el clasificador, es decir, calcular los centroides
        centroids = X_train.groupby(y_train).mean()
        # Imprimir los centroides
        print("\nCentroides - Fold", i + 1)
        print(centroids)

        # Calcular la distancia euclidiana entre cada ejemplo de prueba y cada
centroide
        y_pred = X_test.apply(classify_point, axis=1, centroids=centroids)

        # Calcular la precisión del clasificador
        accuracy = (y_pred == y_test).mean()
        accuracies.append(accuracy)

    return np.mean(accuracies)


# Comprobar que el conjunto de datos existe
try:
    dataset_name = input("¿Cuál es el nombre del conjunto de datos? ")
    # Cargar los datos desde el archivo CSV
    train_data =
[pd.read_csv(f"folds/{dataset_name}_folds/{dataset_name}_fold_{i}.csv") for i in
range(1, 11)]
except FileNotFoundError:
    print(f"\n[x] No se encontraron los datos ingrsados.")
    exit()
```

```python
target = input("¿Cuál es la variable objetivo? ")

# Calcular la precisión del clasificador
accuracy = cross_validate(train_data, target)
print(f"\nPrecisión: {accuracy * 100:.2f}%")
```

2. Resultados

a) Conjunto de datos Iris

Como podemos ver en la siguiente imagen, con el dataset **Iris**, dividiéndolo con la técnica de **K-Folds** y usando el clasificador **Euclidiano** se obtuvo un *accuracy* del **92.0%**

```
Centroides - Fold 8
            sepal_length  sepal_width  petal_length  petal_width
species
setosa          4.986667     3.411111      1.464444     0.240000
versicolor      5.922222     2.762222      4.277778     1.331111
virginica       6.493333     2.957778      5.466667     2.022222

Centroides - Fold 9
            sepal_length  sepal_width  petal_length  petal_width
species
setosa          4.993333     3.413333      1.464444     0.246667
versicolor      5.928889     2.784444      4.266667     1.331111
virginica       6.568889     2.966667      5.535556     2.020000

Centroides - Fold 10
            sepal_length  sepal_width  petal_length  petal_width
species
setosa          5.002222     3.422222      1.466667     0.246667
versicolor      5.924444     2.764444      4.244444     1.324444
virginica       6.582222     2.960000      5.542222     2.015556

Precisión: 92.00%
```

b) Conjunto de datos Iris

Como podemos ver en la siguiente imagen, con el dataset **Diabetes**, dividiéndolo con la técnica de **K-Folds** y usando el clasificador **Euclidiano** se obtuvo un *accuracy* del **63.41%**

```
Centroides - Fold 8
         Pregnancies     Glucose  ...  DiabetesPedigreeFunction        Age
Outcome                           ...
0           3.284444  109.453333  ...                  0.427747  31.397778
1           4.863071  141.995851  ...                  0.546834  37.078838

[2 rows x 8 columns]

Centroides - Fold 9
         Pregnancies     Glucose  ...  DiabetesPedigreeFunction        Age
Outcome                           ...
0           3.273333  109.911111  ...                  0.427609  31.204444
1           4.855372  140.756198  ...                  0.554694  37.289256

[2 rows x 8 columns]

Centroides - Fold 10
         Pregnancies     Glucose  ...  DiabetesPedigreeFunction        Age
Outcome                           ...
0           3.313333  110.651111  ...                  0.432778  31.113333
1           4.888430  142.128099  ...                  0.551174  37.132231

[2 rows x 8 columns]

Precisión: 63.41%
```

II. Clasificador Vecino Más Cercano (1NN)

A. Holdout

Este código implementa el algoritmo K-Vecinos Más Cercanos (K-NN), en particular, el vecino más cercano (1NN), para clasificar puntos de datos. Comienza solicitando al usuario el nombre del conjunto de datos y la variable objetivo. Luego intenta cargar los datos de entrenamiento y prueba de los archivos CSV correspondientes. Con los datos cargados, el código separa las características (X) de la etiqueta objetivo (y) para ambos conjuntos. Utiliza la función `knn_classify` para predecir la etiqueta de cada punto en el conjunto de prueba basándose en las 'k' observaciones más cercanas del conjunto de entrenamiento (en este caso, k se establece en 1, lo que significa que se considera el vecino más cercano). La función knn_classify calcula las distancias euclidianas de un punto de prueba a todos los puntos de entrenamiento, identifica las 'k' más pequeñas, obtiene sus etiquetas y asigna la etiqueta más frecuente a ese punto de prueba. Finalmente, el código evalúa la precisión de las predicciones comparándolas con las etiquetas reales del conjunto de prueba y muestra esta precisión como un porcentaje.

1. Implementación

```
import pandas as pd
import numpy as np

def knn_classify(k, X_train, y_train, X_test):
    y_pred = []
    for test_point in X_test.iterrows():
        distances = np.linalg.norm(X_train - test_point[1], axis=1)
        nearest_indices = np.argsort(distances)[:k]
        nearest_labels = y_train.iloc[nearest_indices]
        unique, counts = np.unique(nearest_labels, return_counts=True)
        predicted_label = unique[np.argmax(counts)]
        y_pred.append(predicted_label)
    return y_pred

def cross_validate(folds, target, k):
    accuracies = []
    num_folds = len(folds)
    for i in range(num_folds):
        # Obtener los conjuntos de entrenamiento y prueba
```

```
        # El conjunto de entrenamiento es la unión de todos los folds excepto el
i-ésimo
        try:
            train_data = pd.concat([folds[j] for j in range(num_folds) if j !=
i])
            X_train = train_data.drop(columns=[target])
            y_train = train_data[target]

            # El conjunto de prueba es el i-ésimo fold
            test_data = folds[i]
            X_test = test_data.drop(columns=[target])
            y_test = test_data[target]
        except KeyError:
            print(f"\n[x] La variable objetivo {target} no existe en los
datos.")
            exit()

        # Entrenar el clasificador KNN
        y_pred = knn_classify(k, X_train, y_train, X_test)

        # Calcular la precisión del clasificador
        accuracy = (y_pred == y_test).mean()
        accuracies.append(accuracy)
    return np.mean(accuracies)

# Comprobar que el conjunto de datos existe
try:
    dataset_name = input("¿Cuál es el nombre del conjunto de datos? ")
    # Cargar los datos desde el archivo CSV
    train_data =
[pd.read_csv(f"folds/{dataset_name}_folds/{dataset_name}_fold_{i}.csv") for i in
range(1, 11)]
except FileNotFoundError:
    print(f"\n[x] No se encontraron los datos ingresados.")
    exit()

target = input("¿Cuál es la variable objetivo? ")
k = 1
```

```
# Calcular la precisión del clasificador
accuracy = cross_validate(train_data, target, k)
print(f"\nPrecisión (K={k}): {accuracy * 100:.2f}%")
```

2. Resultados

a) Conjunto de datos Iris

Como podemos ver en la siguiente imagen, con el dataset **Iris**, dividiéndolo con la técnica de **K-Folds** y usando el clasificador **1NN** se obtuvo un *accuracy* del **95.56%**

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

Precisión: 95.56%
```

b) Conjunto de datos Diabetes

Como podemos ver en la siguiente imagen, con el dataset **Diabetes**, dividiéndolo con la técnica de **K-Folds** y usando el clasificador **1NN** se obtuvo un *accuracy* del **68.83%.**

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

Precisión: 68.83%
```

B. K-Folds

El código implementa el algoritmo de aprendizaje automático K-Vecinos Más Cercanos (K-NN) con una rutina de validación cruzada para evaluar su rendimiento. Solicita al usuario el nombre de un conjunto de datos y una variable objetivo, carga los datos de entrenamiento divididos en varios pliegues desde archivos CSV, y realiza validación cruzada al entrenar el clasificador K-NN en cada pliegue y probarlo en los restantes. Utiliza la distancia euclidiana para identificar los vecinos más cercanos y predecir la clase de cada muestra de prueba. Finalmente, calcula y reporta la precisión media del clasificador sobre todos los pliegues.

1. Implementación

```
import pandas as pd
import numpy as np

def knn_classify(k, X_train, y_train, X_test):
```

```
    y_pred = []
    for test_point in X_test.iterrows():
        distances = np.linalg.norm(X_train - test_point[1], axis=1)
        nearest_indices = np.argsort(distances)[:k]
        nearest_labels = y_train.iloc[nearest_indices]
        unique, counts = np.unique(nearest_labels, return_counts=True)
        predicted_label = unique[np.argmax(counts)]
        y_pred.append(predicted_label)
    return y_pred

def cross_validate(folds, target, k):
    accuracies = []
    num_folds = len(folds)
    for i in range(num_folds):
        # Obtener los conjuntos de entrenamiento y prueba
        # El conjunto de entrenamiento es la unión de todos los folds excepto el
i-ésimo
        try:
            train_data = pd.concat([folds[j] for j in range(num_folds) if j !=
i])
            X_train = train_data.drop(columns=[target])
            y_train = train_data[target]

            # El conjunto de prueba es el i-ésimo fold
            test_data = folds[i]
            X_test = test_data.drop(columns=[target])
            y_test = test_data[target]
        except KeyError:
            print(f"\n[x] La variable objetivo {target} no existe en los
datos.")
            exit()

        # Entrenar el clasificador KNN
        y_pred = knn_classify(k, X_train, y_train, X_test)

        # Calcular la precisión del clasificador
        accuracy = (y_pred == y_test).mean()
        accuracies.append(accuracy)
    return np.mean(accuracies)
```

```
# Comprobar que el conjunto de datos existe
try:
    dataset_name = input("¿Cuál es el nombre del conjunto de datos? ")
    # Cargar los datos desde el archivo CSV
    train_data =
[pd.read_csv(f"folds/{dataset_name}_folds/{dataset_name}_fold_{i}.csv") for i in
range(1, 11)]
except FileNotFoundError:
    print(f"\n[x] No se encontraron los datos ingresados.")
    exit()

target = input("¿Cuál es la variable objetivo? ")
k = 1

# Calcular la precisión del clasificador
accuracy = cross_validate(train_data, target, k)
print(f"\nPrecisión (K={k}): {accuracy * 100:.2f}%")
```

2. Resultados

a) Conjunto de datos Iris

Como podemos ver en la siguiente imagen, con el dataset **Iris**, dividiéndolo con la técnica de **K-Folds** y usando el clasificador **1NN** se obtuvo un *accuracy* del **96.0%**

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

Precisión (K=1): 96.00%
```

b) Conjunto de datos Diabetes

Como podemos ver en la siguiente imagen, con el dataset **Diabetes**, dividiéndolo con la técnica de **K-Folds** y usando el clasificador **1NN** se obtuvo un *accuracy* del **66.92%**

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

Precisión (K=1): 66.92%
```

CONCLUSIÓN

En esta práctica de laboratorio, se han explorado dos métodos de clasificación supervisada basados en la proximidad: el Clasificador basado en Distancia Euclidiana y el Clasificador 1NN (Vecino más Cercano). Mediante el método Holdout y K-Folds, ambos clasificadores se evaluaron en su capacidad de predecir la variable objetivo de conjuntos de datos particulares. El Clasificador Euclidiano se centró en calcular y utilizar centroides para determinar la clase de puntos de datos, mientras que el Clasificador 1NN dependió del punto más cercano para esta tarea.

Los resultados demostraron que, aunque ambos métodos son válidos para el enfoque de clasificación, el clasificador 1NN superó al clasificador basado en distancia euclidiana en términos de precisión promedio, especialmente cuando se aplicó el procedimiento K-Folds.

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 MachineLearning Models.* Recuperado el 24 de octubre de 2023, de: https://www.comet.com/site/blog/understanding-hold-out-methods-for-training- machine-learning-models

Sahani, G (2020, 24 de julio). *Euclidean and Manhattan distance metrics in Machine Learning.* Recuperado el 30 de octubre, de: https://medium.com/analytics-vidhya/euclidean-and-manhattan-distance-metrics-in-machine-learning-a5942a8c9f2f