INSTITUTO POLITECNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO
NOMBRE: Cerda García Gustavo
Materia: Inteligencia Artificial

Ejercicio de Laboratorio 10. Clasificador Euclidiano

Código

Función `euclidean_classifier (X_train, y_train, X_test)`

Calcula las distancias Euclidianas entre cada punto de prueba y los puntos de entrenamiento, y asigna la etiqueta de la instancia más cercana.

```python
# Nombre: euclidean_classifier
# Entradas:
# - X_train: características de entrenamiento
# - y_train: etiquetas de entrenamiento
# - X_test: características de prueba
# Salidas:
# - y_pred: etiquetas predichas
def euclidean_classifier(X_train, y_train, X_test):
    predictions = [] # Lista para almacenar las predicciones
    # Por cada punto de prueba
    for test_point in X_test:
        distances = distance.cdist([test_point], X_train, 'euclidean') # Calculamos las distancias
        nearest_neighbor = np.argmin(distances) # Obtenemos el índice del vecino más cercano
        predictions.append(y_train[nearest_neighbor]) # Añadimos la etiqueta del vecino más cercano a las
predicciones
    return np.array(predictions)
```

Función `hold_out_validation (X, y)`

Divide el dataset en un conjunto de entrenamiento (70%) y uno de prueba (30%), también calcula la precisión del clasificador en el conjunto de prueba.

```python
# Nombre: hold_out_validation
# Entradas:
# - X: características
# - y: etiquetas
# Salidas:
# - accuracy: precisión
def hold_out_validation(X,y):
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, stratify=y) #
Dividimos el dataset
    y_pred = euclidean_classifier(X_train, y_train, X_test) # Realizamos las predicciones
    accuracy = np.mean(y_pred == y_test) # Calculamos la precisión
    return accuracy
```

Función `cross_validation (X, y)`

Aquí se divide el dataset en 10 subconjuntos estratificados, en cada iteración, entrena el clasificador en 9 pliegues y lo prueba en el pliegue restante, al final calcula el promedio y la desviación estándar de las precisiones obtenidas.

```python
# Nombre: cross_validation
# Entradas:
# - X: características
# - y: etiquetas
# Salidas:
# - accuracy: precisión
# - std: desviación estándar
def cross_validation(X,y):
    skf = StratifiedGroupKFold(n_splits=10) # Creamos el objeto StratifiedGroupKFold
    accuracies = [] # Lista para almacenar las precisiones
    # Por cada partición
    for train_index, test_index in skf.split(X, y, groups=df.index):
        X_train, X_test = X[train_index], X[test_index] # Dividimos las características
        y_train, y_test = y[train_index], y[test_index] # Dividimos las etiquetas
        y_pred = euclidean_classifier(X_train, y_train, X_test) # Realizamos las predicciones
        accuracy = np.mean(y_pred == y_test) # Calculamos la precisión
        accuracies.append(accuracy) # Añadimos la precisión a la lista
    return np.mean(accuracies), np.std(accuracies) # Devolvemos la precisión promedio
```

DataSet `iris.csv`

Código

```python
# Autor: Cerda Garcia Gustavo
# Practica 10: Clasificación Euclidiana (Hold-out y Cross Validation)

# Programa que implementa un clasificador euclidiano y lo evalúa mediante hold-out y cross-validation
# Se utiliza el dataset IRIS

# Librerias
import pandas as pd # Libreria para manejo de datos
import numpy as np # Libreria para manejo de arreglos
from sklearn.model_selection import train_test_split, StratifiedGroupKFold # Libreria para dividir un dataset en entrenamiento y prueba
from sklearn.preprocessing import LabelEncoder # Libreria para codificar etiquetas
from scipy.spatial import distance # Libreria para calcular distancias

# Cargamos el dataset
df = pd.read_csv("iris.csv")

# Separamos las características y las etiquetas
X = df.iloc[:, :-1].values
y = df.iloc[:, -1].values

# Codificamos las etiquetas
le = LabelEncoder()
y = le.fit_transform(y)

# Realizamos la validación hold-out
hold_out_accuracy = hold_out_validation(X,y)
print(f"Hold-out validation accuracy: {hold_out_accuracy}")

# Realizamos la validación cruzada
cross_val_accuracy, std = cross_validation(X, y)
print(f"Cross-validation accuracy: {cross_val_accuracy} +/- {std}")
```

Salida

```
PS C:\Users\Gus\Desktop\ESCOM\6 Semestre\Inteligencia Artificial\Practicas\IA-6CV2-GCG\Practica-10>   python .\E1-CE-Iris.py
Hold-out validation accuracy: 0.9777777777777777
Cross-validation accuracy: 0.9528571428571431 +/- 0.060175480350173755
```

DataSet `wine.csv`

Código

```python
# Autor: Cerda García Gustavo
# Practica 10: Clasificación Euclidiana (Hold-out y Cross Validation)

# Programa que implementa un clasificador euclidiano y lo evalúa mediante hold-out y cross-validation
# Se utiliza el dataset WINE

# Librerias
import pandas as pd # Libreria para manejo de datos
import numpy as np # Libreria para manejo de arreglos
from sklearn.model_selection import train_test_split, StratifiedGroupKFold # Libreria para dividir un dataset en entrenamiento y prueba
from sklearn.preprocessing import LabelEncoder # Libreria para codificar etiquetas
from scipy.spatial import distance # Libreria para calcular distancias

# Cargamos el dataset
df = pd.read_csv("wine.csv")

# Separamos las caracteristicas y las etiquetas
# X es el conjunto de caracteristicas
X = df.iloc[:, 1:].values
# Y es el conjunto de etiquetas (primera columna)
y = df.iloc[:, 0].values

# Codificamos las etiquetas
le = LabelEncoder()
y = le.fit_transform(y)

# Realizamos la validación hold-out
hold_out_accuracy = hold_out_validation(X,y)
print(f"Hold-out validation accuracy: {hold_out_accuracy}")

# Realizamos la validación cruzada
cross_val_accuracy, std = cross_validation(X, y)
print(f"Cross-validation accuracy: {cross_val_accuracy} +/- {std}")
```

Salida

```
PS C:\Users\Gus\Desktop\ESCOM\6 Semestre\Inteligencia Artificial\Practicas\IA-6CV2-GCG\Practica-10> python .\E2-CE-Wine.py
Hold-out validation accuracy: 0.7777777777777778
Cross-validation accuracy: 0.784967320261438 +/- 0.08735704285805815
```