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

Ejercicio de Laboratorio 9. Métodos de Validación

Hold Out 70/30 Estratificado

Iris

Código:

```python
# Autor: Cerda García Gustavo
# Practica 9: Métodos de Validación (Hold Out 70-30 IRIS)

# Programa que divide un dataset en dos subconjuntos de acuerdo a un porcentaje
# Se divide el dataset IRIS en 70% entrenamiento y 30% prueba

# Librerias
import numpy as np
import pandas as pd #Libreria para manejo de datos
from sklearn.model_selection import train_test_split #Libreria para dividir un dataset en
entrenamiento y prueba

# Cargamos el dataset IRIS
df = pd.read_csv('iris.csv', header=None)

# X y Y
X = df[df.columns[:-1]] # Todas las columnas menos la ultima
Y = df[df.columns[-1]] # La ultima columna

# Nombre: hold_out_estratificado
# Desc: Divide un dataset en dos subconjuntos de acuerdo a un porcentaje
# Entrada:
#  - X: DataFrame con los datos
#  - y: DataFrame con las etiquetas
# Salida:
#  - X_train: DataFrame con los datos de entrenamiento
#  - X_test: DataFrame con los datos de prueba
#  - y_train: DataFrame con las etiquetas de entrenamiento
#  - y_test: DataFrame con las etiquetas de prueba
def hold_out_estratificado(X, y):
    # Dividir en 70% entrenamiento y 30% prueba
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, stratify=y)
    return X_train, X_test, y_train, y_test

# Dividir en 70% entrenamiento y 30% prueba
X_train, X_test, y_train, y_test = hold_out_estratificado(X, Y)
print("HOLD OUT 70-30 IRIS")
print("Datos de entrenamiento")
print(X_train)
print(y_train)
print("Numero de datos de entrenamiento: ", len(X_train))
print()
print("Datos de prueba")
print(X_test)
print(y_test)
print("Numero de datos de prueba: ", len(X_test))
print()
```

Salida:

```
HOLD OUT 70-30 IRIS
Datos de entrenamiento
        0    1    2    3
144   6.7  3.3  5.7  2.5
74    6.4  2.9  4.3  1.3
148   6.2  3.4  5.4  2.3
47    4.6  3.2  1.4  0.2
64    5.6  2.9  3.6  1.3
..    ...  ...  ...  ...
63    6.1  2.9  4.7  1.4
107   7.3  2.9  6.3  1.8
81    5.5  2.4  3.7  1.0
113   5.7  2.5  5.0  2.0
142   5.8  2.7  5.1  1.9

[105 rows x 4 columns]
144     Iris-virginica
74     Iris-versicolor
148     Iris-virginica
47         Iris-setosa
64     Iris-versicolor
           ...
63     Iris-versicolor
107     Iris-virginica
81     Iris-versicolor
113     Iris-virginica
142     Iris-virginica
Name: 4, Length: 105, dtype: object
Numero de datos de entrenamiento:  105
44         Iris-setosa
70     Iris-versicolor
132     Iris-virginica
77     Iris-versicolor
5          Iris-setosa
128     Iris-virginica
25         Iris-setosa
65     Iris-versicolor
97     Iris-versicolor
130     Iris-virginica
95     Iris-versicolor
7          Iris-setosa
36         Iris-setosa
114     Iris-virginica
127     Iris-virginica
83     Iris-versicolor
111     Iris-virginica
99     Iris-versicolor
115     Iris-virginica
90     Iris-versicolor
56     Iris-versicolor
82     Iris-versicolor
17         Iris-setosa
80     Iris-versicolor
18         Iris-setosa
124     Iris-virginica
105     Iris-virginica
16         Iris-setosa
140     Iris-virginica
116     Iris-virginica
68     Iris-versicolor
48         Iris-setosa
Name: 4, dtype: object
Numero de datos de prueba:  45
```

Wine

Código:

```python
# Autor: Cerda García Gustavo
# Practica 9: Métodos de Validación (Hold Out 70-30 WINE)

# Programa que divide un dataset en dos subconjuntos de acuerdo a un porcentaje
# Se divide el dataset WINE en 70% entrenamiento y 30% prueba

# Librerias
import numpy as np
import pandas as pd #Libreria para manejo de datos
from sklearn.model_selection import train_test_split #Libreria para dividir un dataset en
entrenamiento y prueba

# Leer el archivo
df = pd.read_csv('wine.csv', header=None)

# X y Y
X = df[df.columns[1:]] # Todas las columnas menos la primera
Y = df[df.columns[0]] # La primera columna

print("X\n", X)
print("Y\n", Y)

# Nombre: hold_out_estratificado
# Desc: Divide un dataset en dos subconjuntos de acuerdo a un porcentaje
# Entrada:
#  - X: DataFrame con los datos
#  - y: DataFrame con las etiquetas
# Salida:
#  - X_train: DataFrame con los datos de entrenamiento#  - X_test: DataFrame con los datos de prueba
#  - y_train: DataFrame con las etiquetas de entrenamiento
#  - y_test: DataFrame con las etiquetas de prueba
def hold_out_estratificado(X, y):
    # Dividir en 70% entrenamiento y 30% prueba
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, stratify=y)
    return X_train, X_test, y_train, y_test

# Dividir en 70% entrenamiento y 30% prueba
X_train, X_test, y_train, y_test = hold_out_estratificado(X, Y)
print("HOLD OUT 70-30 WINE")
print("Datos de entrenamiento")
print(X_train)
print(y_train)
print("Numero de datos de entrenamiento: ", len(X_train))
print()
print("Datos de prueba")
print(X_test)
print(y_test)
print("Numero de datos de prueba: ", len(X_test))
print()
```

Salida:

```
HOLD OUT 70-30 WINE
Datos de entrenamiento
        1     2     3     4    5     6   ...     8     9     10    11    12   13
150  13.50  3.12  2.62  24.0  123  1.40  ...  0.22  1.25  8.60  0.59  1.30  500
101  12.60  1.34  1.90  18.5   88  1.45  ...  0.29  1.35  2.45  1.04  2.77  562
118  12.77  3.43  1.98  16.0   80  1.63  ...  0.43  0.83  3.40  0.70  2.12  372
40   13.56  1.71  2.31  16.2  117  3.15  ...  0.34  2.34  6.13  0.95  3.38  795
63   12.37  1.13  2.16  19.0   87  3.50  ...  0.19  1.87  4.45  1.22  2.87  420
..     ...   ...   ...   ...  ...   ...  ...   ...   ...   ...   ...   ...  ...
82   12.08  1.13  2.51  24.0   78  2.00  ...  0.40  1.40  2.20  1.31  2.72  630
125  12.07  2.16  2.17  21.0   85  2.60  ...  0.37  1.35  2.76  0.86  3.28  378
155  13.17  5.19  2.32  22.0   93  1.74  ...  0.61  1.55  7.90  0.60  1.48  725
116  11.82  1.47  1.99  20.8   86  1.98  ...  0.30  1.53  1.95  0.95  3.33  495
93   12.29  2.83  2.22  18.0   88  2.45  ...  0.25  1.99  2.15  1.15  3.30  290

[124 rows x 13 columns]
150    3
101    2
118    2
40     1
63     2
      ..
82     2
125    2
155    3
116    2
93     2
Name: 0, Length: 124, dtype: int64
Numero de datos de entrenamiento:  124
```

```
167    3
76     2
147    3
41     1
7      1
105    2
26     1
142    3
72     2
165    3
12     1
138    3
106    2
0      1
103    2
31     1
25     1
34     1
117    2
65     2
108    2
92     2
88     2
52     1
Name: 0, dtype: int64
Numero de datos de prueba:  54
```

10-Fold-Cross-Validation Estratificado

Iris

Código:

```python
# Autor: Cerda García Gustavo
# Practica 9: Metodos de Validación (10-Fold Cross Validation IRIS)

# Programa que divide un dataset en 10 subconjuntos de acuerdo a un porcentaje
# Se divide el dataset IRIS en 10 subconjuntos

# Librerias
import numpy as np
import pandas as pd #Libreria para manejo de datos
from sklearn.model_selection import StratifiedKFold #Libreria para dividir un dataset en 10 subconjuntos

# Cargamos el dataset IRIS
df = pd.read_csv('iris.csv', header=None)

# Nombre: 10_fold_cross_validation
# Desc: Divide un dataset en 10 subconjuntos de acuerdo a un porcentaje
# Entrada:
#  - X: DataFrame con los datos
#  - y: DataFrame con las etiquetas
# Salida:
#  - X_train: DataFrame con los datos de entrenamiento
#  - X_test: DataFrame con los datos de prueba
#  - y_train: DataFrame con las etiquetas de entrenamiento
#  - y_test: DataFrame con las etiquetas de prueba
def fold_cross_validation(X,y,modelo):
    # Inicializa el generador de los 10 subconjuntos
    skf = StratifiedKFold(n_splits=10)

    # Inicializa las listas para guardar los resultados
    resultados = []

    # Itera sobre los 10 subconjuntos
    for train_index, test_index in skf.split(X, y):
        # Divide los datos en entrenamiento y prueba
        X_train, X_test = X.iloc[train_index], X.iloc[test_index]
        y_train, y_test = y.iloc[train_index], y.iloc[test_index]

        # Entrena el modelo
        modelo.fit(X_train, y_train)

        # Evalua el modelo
        score = modelo.score(X_test, y_test)
        resultados.append(score)

    return resultados

# X y Y
# Todas menos la ultima columna
X = df[df.columns[:-1]]
# La ultima columna
Y = df[df.columns[-1]]

# Modelo
from sklearn.tree import DecisionTreeClassifier

modelo = DecisionTreeClassifier()

resultados = fold_cross_validation(X,Y,modelo)

print("Resultados de 10-Fold Cross Validation IRIS")
print(resultados)
```

Salida:

```
Resultados de 10-Fold Cross Validation IRIS
[1.0, 0.9333333333333333, 1.0, 0.9333333333333333, 0.9333333333333333, 0.8666666666666667, 0.9333333333333333, 1.0, 1.0, 1.0]
```

WineW

Código:

```python
# Autor: Cerda García Gustavo
# Practica 9: Métodos de Validación (10-Fold Cross Validation WINE)

# Programa que divide un dataset en 10 subconjuntos de acuerdo a un porcentaje
# Se divide el dataset WINE en 10 subconjuntos

# Librerias
import numpy as np
import pandas as pd #Libreria para manejo de datos
from sklearn.model_selection import StratifiedKFold #Libreria para dividir un dataset en 10
subconjuntos

# Cargamos el dataset IRIS
df = pd.read_csv('wine.csv', header=None)

# Nombre: 10_fold_cross_validation
# Desc: Divide un dataset en 10 subconjuntos de acuerdo a un porcentaje
# Entrada:
#  - X: DataFrame con los datos
#  - y: DataFrame con las etiquetas
# Salida:
#  - X_train: DataFrame con los datos de entrenamiento
#  - X_test: DataFrame con los datos de prueba
#  - y_train: DataFrame con las etiquetas de entrenamiento
#  - y_test: DataFrame con las etiquetas de prueba
def fold_cross_validation(X,y,modelo):
    # Inicializa el generador de los 10 subconjuntos
    skf = StratifiedKFold(n_splits=10)

    # Inicializa las listas para guardar los resultados
    resultados = []

    # Itera sobre los 10 subconjuntos
    for train_index, test_index in skf.split(X, y):
        # Divide los datos en entrenamiento y prueba
        X_train, X_test = X.iloc[train_index], X.iloc[test_index]
        y_train, y_test = y.iloc[train_index], y.iloc[test_index]

        # Entrena el modelo
        modelo.fit(X_train, y_train)

        # Evalua el modelo
        score = modelo.score(X_test, y_test)
        resultados.append(score)

    return resultados

# X y Y
X = df[df.columns[1:]] # Todas las columnas menos la primera
Y = df[df.columns[0]] # La primera columna

# Modelo
from sklearn.tree import DecisionTreeClassifier

modelo = DecisionTreeClassifier()

resultados = fold_cross_validation(X,Y,modelo)

print("Resultados de 10-Fold Cross Validation WINE")
print(resultados)
```

Salida:

```
Resultados de 10-Fold Cross Validation WINE
[0.8888888888888888, 0.8888888888888888, 0.7222222222222222, 0.8888888888888888
, 0.8333333333333334, 0.8333333333333334, 1.0, 0.9444444444444444, 0.9411764705
882353, 0.7058823529411765]
```