UNIDAD TEMÁTICA 2: Procesamiento previo de los datos

PRACTICOS DOMICILIARIOS INDIVIDUALES #3

En este ejercicio se desea practicar las técnicas de pre procesamiento y obtención estadísticas de un dataset utilizando funciones en Python.

1. Descargar el dataset “wine” del repositorio UCI y cargar el dataset en formato CSV.
2. Imprimir todas las columnas de las primeras 10 filas.
3. Convertir los valores numéricos en string a float, en caso de ser necesario
4. Obtener los valores mínimos y máximos correspondientes a cada columna.
5. Obtener la media de los valores de cada columna.
6. Obtener la desviación estándar de los valores de cada columna.
7. Normalizar los valores el dataset original.
8. Estandarizar los valores del dataset original.
9. Dividir el dataset en conjuntos de entrenamiento y testing.

Para esto se propone el siguiente código de referencia:

```
from csv import reader
from math import sqrt
from random import seed
from random import randrange

# Load a CSV file
def load_csv(filename):
     dataset = list()
     with open(filename, 'r') as file:
          csv_reader = reader(file)
          for row in csv_reader:
               if not row:
                    continue
               dataset.append(row)
     return dataset

# Convert string column to float
def str_column_to_float(dataset, column):
     for row in dataset:
          row[column] = float(row[column].strip())

# Find the min and max values for each column
def dataset_minmax(dataset):
     minmax = list()
     for i in range(len(dataset[0])):
          col_values = [row[i] for row in dataset]
          value_min = min(col_values)
```

```
          value_max = max(col_values)
          minmax.append([value_min, value_max])
     return minmax

# Rescale dataset columns to the range 0-1
def normalize_dataset(dataset, minmax):
     for row in dataset:
          for i in range(len(row)):
               row[i] = (row[i] - minmax[i][0]) /
(minmax[i][1] - minmax[i][0])

# calculate column means
def column_means(dataset):
     means = [0 for i in range(len(dataset[0]))]
     for i in range(len(dataset[0])):
          col_values = [row[i] for row in dataset]
          means[i] = sum(col_values) / float(len(dataset))
     return means

# calculate column standard deviations
def column_stdevs(dataset, means):
     stdevs = [0 for i in range(len(dataset[0]))]
     for i in range(len(dataset[0])):
          variance = [pow(row[i]-means[i], 2) for row in
dataset]
          stdevs[i] = sum(variance)
     stdevs = [sqrt(x/(float(len(dataset)-1))) for x in
stdevs]
     return stdevs

# standardize dataset
def standardize_dataset(dataset, means, stdevs):
     for row in dataset:
          for i in range(len(row)):
               row[i] = (row[i] - means[i]) / stdevs[i]

# Split a dataset into a train and test set
def train_test_split(dataset, split=0.60):
     train = list()
     train_size = split * len(dataset)
     dataset_copy = list(dataset)
     while len(train) < train_size:
          index = randrange(len(dataset_copy))
          train.append(dataset_copy.pop(index))
     return train, dataset_copy
```