Universidad de Córdoba

Escuela Politécnica Superior de Córdoba

Ingeniería Informática

Especialidad: Computación

Cuarto curso. Primer cuatrimestre

Prácticas de Introducción a la Minería de Datos (IMD).

Adrián López Ortiz

p42loora@uco.es

Curso académico 2019-2020

Córdoba, December 31, 2019

Contenidos

Lista de figuras

1 Introducción

En este trabajo se recogerán los resultados obtenidos tras la realización de las prácticas de la asignatura **Introducción a la Minería de Datos** y la conclusión que se ha sacado de estos. Dichas prácticas consistirán en realizar un código en el lenguaje de programación *Python* y ejecutándolo obtener los resultados requeridos en cada ejercicio de cada práctica.

2 Práctica 1. Ejercicios de introducción a numpy

En esta práctica, realizaremos una serie de ejercicios que tendrán cómo objetivo familiarizarnos con el lenguaje que usaremos en todas las prácticas de la asignatura, Python, así como conocer la librería de la que hemos recibido en clase unas nociones básicas, y profundizar por nuestra cuenta en el aprendizaje de esta librería para realizar dicha práctica en cuestión. A continuación se mostrará el código de cada ejercicio así cómo una captura de pantalla con el resultado obtenido al ejecutarlo.

2.1 Ejercicio 1

Implemente mediante un programa Python la asignación del reparto de escaños de una circunscripción electoral usando la Ley D'Hondt. Los datos se pueden introducir por teclado o leerse desde un fichero.

En nuestro caso, se leerá desde un fichero creado en la carpeta de dicha práctica, *hont.txt*.

Listing 1: Práctica 1.Ejercicio 1.

```
import numpy as np

#Leemos el fichero que contiene los datos y lo almacenamos
    en un array
file=input("Introduce el nombre del fichero que contiene los
     votos:")
file_object  = open(file, "r")
datos_fichero=np.array([line.split(' ') for line in
    file_object]);
for i in datos_fichero:
    i[1]=i[1][:-1]
numEscanos=input("Introduce el numero de escagnos a
    introducir:")
```

```
matriz=np.array([])
k=0
for i in datos_fichero:
    v=[]
    for j in range(1,int(numEscanos)+1):
        v.append(int(int(i[1])/int(j)))
    if(k==0):
        matriz=v;
        k=1
    else:
        matriz=np.vstack([matriz,v])

#Calculamos los escagnos que se lleva cada partido
numEscanos_finales=np.array([0 for i in range(0,np.size(
    datos_fichero,0))])
aux=numEscanos
while(aux!=0):
    numEscanos_finales[int(matriz.argmax()/numEscanos)]=
        numEscanos_finales[int(matriz.argmax()/numEscanos)
        ]+1;
    np.put(matriz,matriz.argmax(),0)
    aux=aux-1;
for i in range(0,np.size(datos_fichero,0)):
    print("El partido ",datos_fichero[i][0]," ha sacado ",
        numEscanos_finales[i]," escagnos")
```

```
p42loora@BIBSL092:~/Desktop/gitkraken/IMD/Practicas/p1$ python2.7 ej1.py
Introduce el nombre del fichero que contiene los votos:"hont.txt"
Introduce el numero de escagnos a introducir:7
('El partido ', 'A', ' ha sacado ', 1, ' escagnos')
('El partido ', 'B', ' ha sacado ', 2, ' escagnos')
('El partido ', 'C', ' ha sacado ', 1, ' escagnos')
('El partido ', 'D', ' ha sacado ', 3, ' escagnos')
('El partido ', 'E', ' ha sacado ', 0, ' escagnos')
p42loora@BIBSL092:~/Desktop/gitkraken/IMD/Practicas/p1$
```

Figura 1: Ejecución del Ejercicio 1 de la práctica 1.

Listing 2: Práctica 1. Archivo del Ejercicio 1.

```
A 100
B 200
C 125
D 212
E 57
```

2.2 Ejercicio 2

Implemente un programa Python que genere aleatoriamente una matriz de valores reales de un tamaño indicado por teclado. Sobre la matriz generada realice las siguientes operaciones:
-Obtenga los valores máximos y mínimos de la matriz.
-Use el producto escalar para obtener el ángulo formado por dos vectores fila o columna solicitados por teclado.

Listing 3: Práctica 1.Ejercicio 2.

```
import numpy as np

#Introducimos por teclado las dimensiones de la matriz y la
    generamos con valores aleatorios
rows=input("Introduce el numero de filas:")
cols=input("Introduce el numero de columnas:")
```

```
m=np.random.rand(int(rows), int(cols))
m=np.array(m)
print(m)
#Imprimimos el valor maximo y el minimo de la matriz
print("El maximo valor de la matriz es ", m.max())
print("El minimo valor de la matriz es ", m.min())

#Calculamos el producto escalar para dos filas o dos
    columnas
aux=input("Introduzca 0 si quiere que se use dos filas para
    el producto escalar y un valor distinto de cero para usar
     dos columnas:")
if(int(aux)==0):
    fila1=input("Introduce el numero de la primera fila a
        usar:")
    fila2=input("Introduce el numero de la segunda fila a
        usar:")
    if(int(fila1)<int(rows) and int(fila2)<int(rows) and int
        (fila1)>=0 and int(fila2)>=0):
        v1=m[int(fila1)]
        v2=m[int(fila2)]
        producto_escalar=np.sum(v1*v2)
        modv1=np.sqrt(sum(v1*v1))
        modv2=np.sqrt(sum(v2*v2))
        aux2=producto_escalar/(modv1*modv2)
        angulo=np.arccos(aux)
        print("El angulo formado por los dos vectores fila
            es: ",(angulo*180)/3.141592653589793," grados")
    else:
        print("Indices de fila de la matriz no validos")
else:
    col1=input("Introduce el numero de la primera columna a
        usar:")
```

```
    col2=input("Introduce el numero de la segunda columna a
        usar:")
    if(int(col1)<int(cols) and int(col2)<int(cols) and int(
        col1)>=0 and int(col2)>=0):
        v1=m[:,int(col1)]
        v2=m[:,int(col2)]
        producto_escalar=np.sum(v1*v2)
        modv1=np.sqrt(sum(v1*v1))
        modv2=np.sqrt(sum(v2*v2))
        aux2=producto_escalar/(modv1*modv2)
        angulo=np.arccos(aux)
        print("El angulo formado por los dos vectores
            columna es: ",(angulo*180)/3.141592653589793,"
            grados")
    else:
        print("Indices de columna de la matriz no validos")
```

```
p42loora@BIBSL092:~/Desktop/gitkraken/IMD/Practicas/p1$ python2.7 ej2.py
Introduce el numero de filas:2
Introduce el numero de columnas:3
[[0.65803086 0.2972504  0.85665007]
 [0.90478065 0.07170532 0.29840134]]
('El maximo valor de la matriz es ', '0.9047806465842525')
('El minimo valor de la matriz es ', '0.0717053160788087')
Introduzca 0 si quiere que se use dos filas para el producto escalar y un valor distinto de cero para usar dos columnas:0
Introduce el numero de la primera fila a usar:0
Introduce el numero de la segunda fila a usar:1
('El angulo formado por los dos vectores fila es: ', '90.0', ' grados')
p42loora@BIBSL092:~/Desktop/gitkraken/IMD/Practicas/p1$
p42loora@BIBSL092:~/Desktop/gitkraken/IMD/Practicas/p1$
p42loora@BIBSL092:~/Desktop/gitkraken/IMD/Practicas/p1$ python2.7 ej2.py
Introduce el numero de filas:2
Introduce el numero de columnas:3
[[0.27773751 0.27109704 0.29640101]
 [0.57066023 0.28423731 0.10036337]]
('El maximo valor de la matriz es ', '0.5706602327662426')
('El minimo valor de la matriz es ', '0.1003633683401638')
Introduzca 0 si quiere que se use dos filas para el producto escalar y un valor distinto de cero para usar dos columnas:1
Introduce el numero de la primera columna a usar:0
Introduce el numero de la segunda columna a usar:2
('El angulo formado por los dos vectores columna es: ', '0.0', ' grados')
p42loora@BIBSL092:~/Desktop/gitkraken/IMD/Practicas/p1$
```

Figura 2: Ejecución del Ejercicio 2 de la práctica 1.

2.3 Ejercicio 3

Implemente un programa Python que lea una matriz de número reales desde teclado de una dimensión dada. A partir de la matriz leída debe calcular la siguiente información:

-Máximo por filas y por columnas.

-Determinante de la matriz.

-Rango de la matriz, (OPCIONAL).

Listing 4: Práctica 1.Ejercicio 3.

```
import numpy as np

#Introducimos por teclado el tamano de la matriz
rows=input("Introduce el numero de filas y columnas, (es una
     matriz cuadrada):")
rows=int(rows)
matriz=np.array([],dtype=float)
#Rellenamos la matriz
z=0
for i in range(0,rows):
    aux=[]
    for j in range(0,rows):
        print("Introduce el valor de ",i,",",j,":")
        value=input()
        aux.append(float(value))

    #Para apilar de forma vertical cada nueva fila generada
       a la matriz, si es la primera fila no hace falta
    if(z==0):
        matriz=aux
        z=1
    else:
        matriz=np.vstack([matriz,aux])
```

```
print(matriz)

#Calculamos el maximo de cada fila y cada columna (
    recorremos primero por filas y luego por columnas,
    calculando el maximo de cada una  de ellas)
z=0
for i in matriz.max(axis=1):
    print("El maximo de la fila ",z," es ",i)
    z=z+1
z=0
for i in matriz.max(axis=0):
    print("El maximo de la columna ",z," es ",i)
    z=z+1

#Calculamos el determinante y el rango de la matriz
print("El determinante de la matriz es ",np.linalg.det(
    matriz))
print("El rango de la matriz es ", np.linalg.matrix_rank(
    matriz))
```

```
p42loora@BIBSL092:~/Desktop/gitkraken/IMD/Practicas/p1$ python2.7 ej3.py
Introduce el numero de filas y columnas, (es una matriz cuadrada):3
('Introduce el valor de ', 0, ',', 0, ':')
1
('Introduce el valor de ', 0, ',', 1, ':')
2
('Introduce el valor de ', 0, ',', 2, ':')
3
('Introduce el valor de ', 1, ',', 0, ':')
9
('Introduce el valor de ', 1, ',', 1, ':')
8
('Introduce el valor de ', 1, ',', 2, ':')
7
('Introduce el valor de ', 2, ',', 0, ':')
4
('Introduce el valor de ', 2, ',', 1, ':')
1
('Introduce el valor de ', 2, ',', 2, ':')
1
[[1. 2. 3.]
 [9. 8. 7.]
 [4. 1. 1.]]
('El maximo de la fila ', 0, ' es ', '3.0')
('El maximo de la fila ', 1, ' es ', '9.0')
('El maximo de la fila ', 2, ' es ', '4.0')
('El maximo de la columna ', 0, ' es ', '9.0')
('El maximo de la columna ', 1, ' es ', '8.0')
('El maximo de la columna ', 2, ' es ', '7.0')
('El determinante de la matriz es ', '-30.000000000000004')
('El rango de la matriz es ', 3)
p42loora@BIBSL092:~/Desktop/gitkraken/IMD/Practicas/p1$
```

Figura 3: Ejecución del Ejercicio 3 de la práctica 1.

2.4 Ejercicio 4

Implemente un programa Python que lea una matriz de número enteros desde teclado de una dimensión dada. A partir de la matriz leída debe calcular la siguiente información:
-Moda de la matriz.
-Media de todos los elementos de la matriz.

Listing 5: Práctica 1.Ejercicio 4.

```
import numpy as np
```

```
#Introducimos por teclado las dimensiones de la matriz
matriz=np.array([],dtype=int)
rows=input("Introduzca el numero de filas:")
cols=input("Introduzca el numero de columnas:")

#Igual que en el ej3.py
z=0
for i in range(0,int(rows)):
    aux=[]
    for j in range(0,int(cols)):
        print("Introduzca el valor de ",i,",",j,":")
        value=input()
        aux.append(int(value))
    if(z==0):
        matriz=aux
        z=1
    else:
        matriz=np.vstack([matriz,aux])
print(matriz)

#Calculamos la media y la moda de la matriz
a,b=np.unique(matriz, return_counts=True)
print("La moda de la matriz es ",a[np.argmax(b)])
print("La media de la matriz es ",matriz.mean())
```

```
p42loora@BIBSL092:~/Desktop/gitkraken/IMD/Practicas/p1$ python2.7 ej4.py
Introduzca el numero de filas:2
Introduzca el numero de columnas:3
('Introduzca el valor de ', 0, ',', 0, ':')
1
('Introduzca el valor de ', 0, ',', 1, ':')
1
('Introduzca el valor de ', 0, ',', 2, ':')
1
('Introduzca el valor de ', 1, ',', 0, ':')
3
('Introduzca el valor de ', 1, ',', 1, ':')
8
('Introduzca el valor de ', 1, ',', 2, ':')
5
[[1 1 1]
 [3 8 5]]
('La moda de la matriz es ', 1)
('La media de la matriz es ', '3.1666666666666665')
p42loora@BIBSL092:~/Desktop/gitkraken/IMD/Practicas/p1$
```

Figura 4: Ejecución del Ejercicio 4 de la práctica 1.

2.5 Ejercicio 5

Implemente un programa Python que lea una matriz de número reales desde un fichero texto con formato libre. Una vez leído el programa debe obtener la inversa de la matriz y realizar un producto matricial para comprobar que el cálculo de la inversa es correcto.

Listing 6: Práctica 1.Ejercicio 5.

```
import numpy as np

#Leemos la matriz del fichero
file=input("Introduce el fichero que contiene la matriz:")
file_object  = open(file, "r")

matriz=np.array([line.split(' ') for line in file_object]);
for i,item in enumerate(matriz):
    for j,item2 in enumerate(item):
        #Borra los espacios del string para que no de
            problemas
```

```
        matriz[i,j]=item2.rstrip()

#Comprobamos que sea cuadrada
matriz = matriz.astype('float')
if(matriz[:,0].size!=matriz[0].size):
    print("La matriz no es invertible")
    exit()

#Comprobamos que la matriz sea invertible (cuando el
    determinante es distinto de cero), se pone un numero
    chico por el error cometido al redondear
det=np.linalg.det(matriz);
if(1e-9>abs(det)):
    print("La matriz no es invertible")
    exit()

#Tras comprobar que es invertible, se calcula la inversa
matriz_inv=matriz.copy()
for i in range(matriz[0].size):
    for j in range(matriz[0].size):
        aux=matriz;
        aux=np.delete(aux,j,axis=1)
        aux=np.delete(aux,i,axis=0)
        matriz_inv[i,j]=((-1)**(i+j))*np.linalg.det(aux)

matriz_inv=matriz_inv.transpose()
matriz_inv=matriz_inv/det
print("La matriz inversa es: ")
print(matriz_inv)
print("El producto matricial es (deberia de dar la matriz
    identidad de tamano 4):")
print(abs(np.around(np.matmul(matriz_inv,matriz))))
```

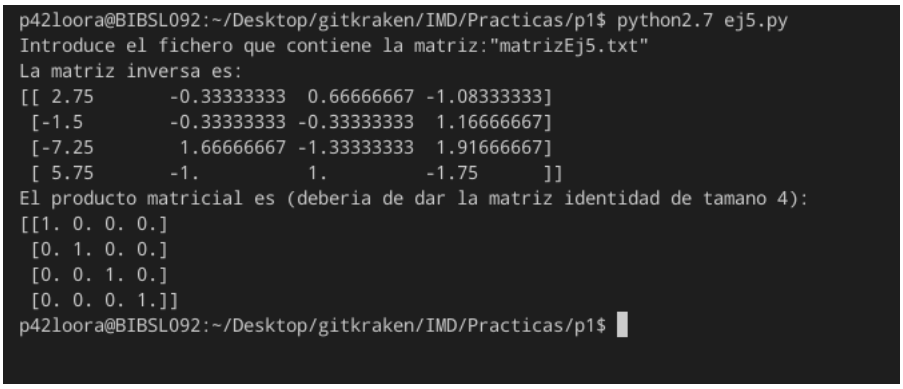

Figura 5: Ejecución del Ejercicio 5 de la práctica 1.

Listing 7: Práctica 1.Archivo del Ejercicio 5.

```
1 2 3 4
5 5 8 9
8 4 3 1
5 6 7 8
```

3 Práctica 2. Exploración de datos

3.1 Ejercicio 1.

Obtenga tres ejemplos de ficheros de datos en formato CSV, ARFF u otro cualquiera de:
-Weka datasets.
-UCI MLR.

Para este ejercicio he obtenido los siguientes ficheros de datos: *iris.arff, diabetes.arff y vote.arff*. El dataset de *vote.arff* se ha tenido que binarizar previamente para poder trabajar con él.

3.2 Ejercicio 2.

Usando Pandas cargue los ficheros y evalúe qué información puede obtener del histograma de atributos.

Listing 8: Práctica 2.Ejercicio 2.

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.io import arff

#Hacemos una funcion a la que pasamos el nombre del fichero
    y ahorramos codigo
def ej2function(file):
    #Cargamos archivo y generamos el histograma que se
        muestra por pantalla.
        data = arff.loadarff(file)
        df = pd.DataFrame(data[0])
        df.plot.hist(alpha=0.5)
```

```
        plt.show()


ej2function('Datasets/iris.arff')
ej2function('Datasets/diabetes.arff')
ej2function('Datasets/vote.arff')
```

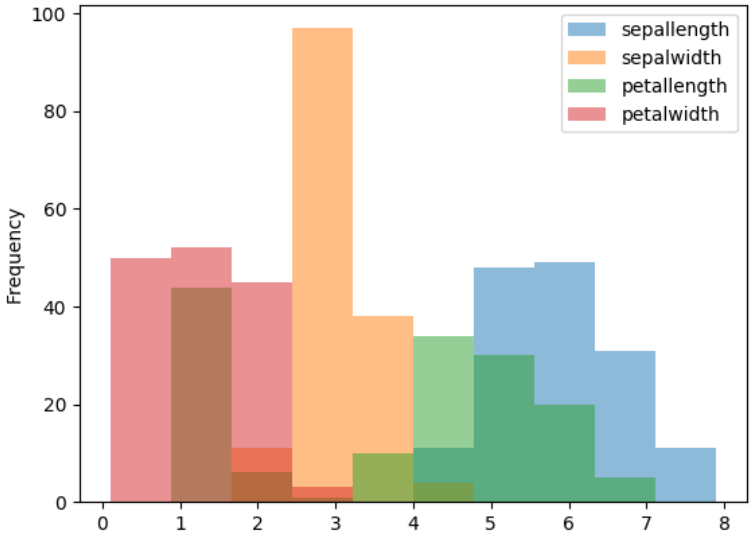

Figura 6: Ejecución del Ejercicio 2 de la práctica 2. Dataset Iris.

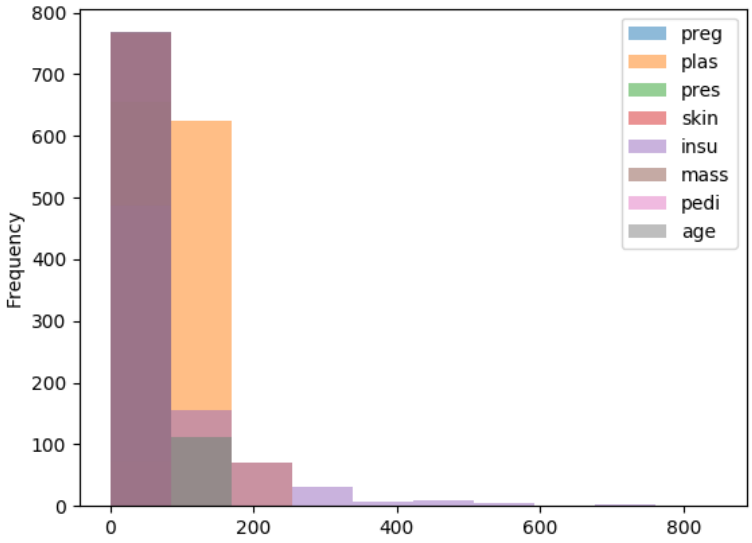

Figura 7: Ejecución del Ejercicio 2 de la práctica 2. Dataset Diabetes.

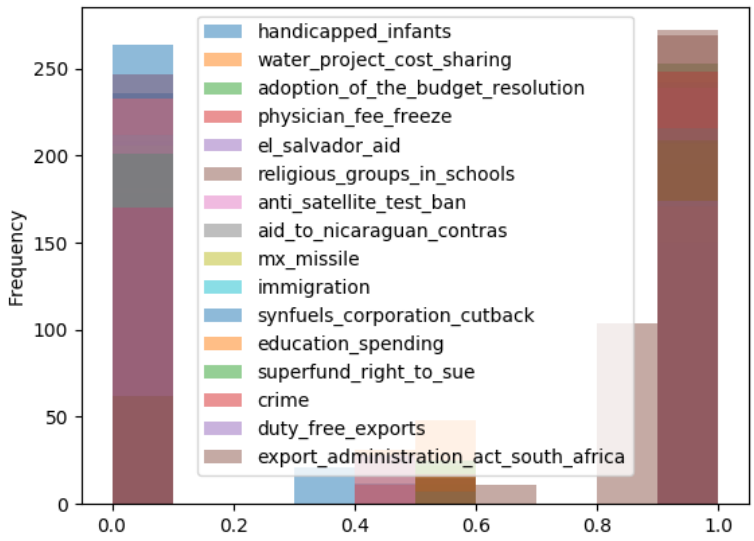

Figura 8: Ejecución del Ejercicio 2 de la práctica 2. Dataset Vote.

3.3 Ejercicio 3.

Estudie el efecto de la normalización (reescalar en el intervalo [0, 1]) y la estandarización ($\mu = 0, \sigma = 1$) sobre el histograma.

A continuación mostraremos el código de dicho ejercicio:

Listing 9: Práctica 2.Ejercicio 3.

```
from sklearn import preprocessing
from sklearn.preprocessing import MinMaxScaler
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.io import arff


scaler = MinMaxScaler()
```

```
def normalizar(file):
        data = arff.loadarff(file)
        df = pd.DataFrame(data[0])
        df = scaler.fit_transform(df.iloc[:, :-1])
        df = pd.DataFrame(df)
        df.plot.hist(alpha=0.5)
        plt.show()

def estandarizar(file):
        data = arff.loadarff(file)
        df = pd.DataFrame(data[0])
        df = df.iloc[:, :-1]
        df = (df-np.mean(df))/np.std(df)
        df.plot.hist(alpha=0.5)
        plt.show()

normalizar('Datasets/iris.arff')
normalizar('Datasets/diabetes.arff')
normalizar('Datasets/vote.arff')

estandarizar('Datasets/iris.arff')
estandarizar('Datasets/diabetes.arff')
estandarizar('Datasets/vote.arff')
```

3.3.1 Normalización de los datasets

Aquí se normalizan todos los patrones de cada dataset reescalándolos con valores entre el cero y el uno.

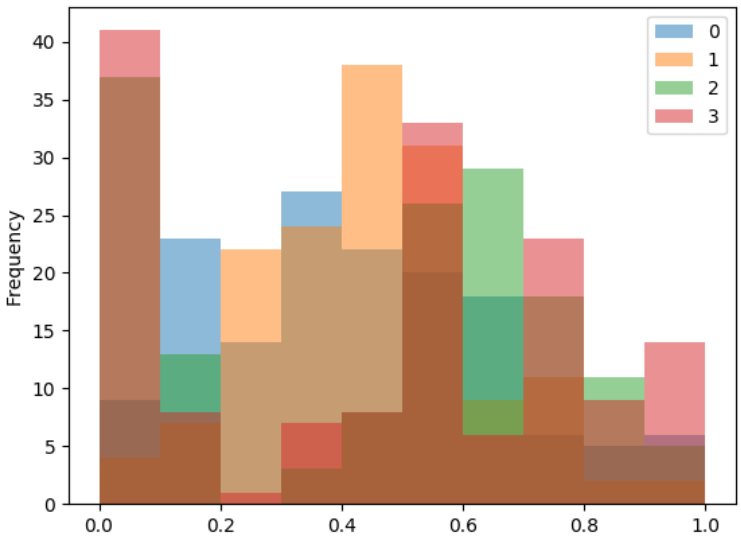

Figura 9: Ejecución del Ejercicio 3 de la práctica 2. Dataset Iris normalizado.

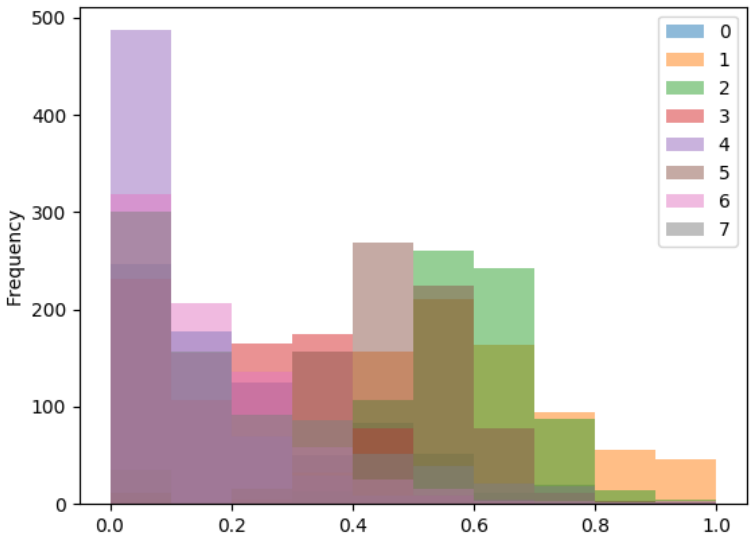

Figura 10: Ejecución del Ejercicio 3 de la práctica 2. Dataset Diabetes normalizado.

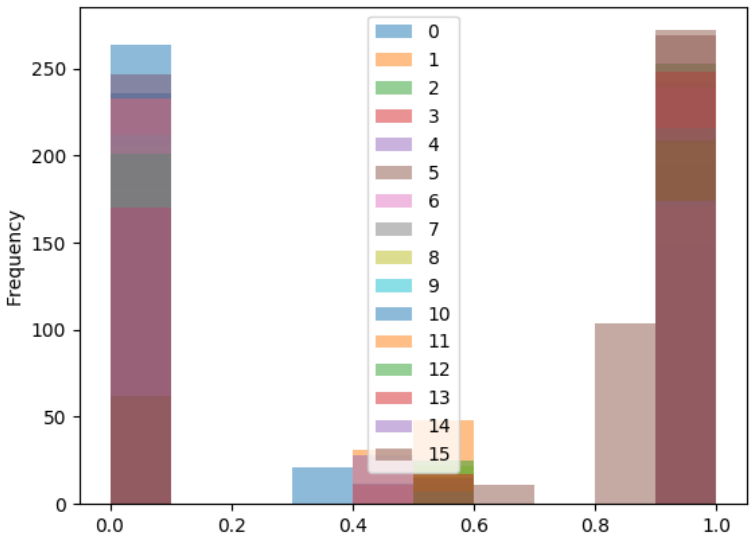

Figura 11: Ejecución del Ejercicio 3 de la práctica 2. Dataset Vote normalizado.

3.3.2 Estandarización de los datasets

Aquí se estandarizan cada uno de los datasets usado en está práctica. Se puede observar como la representación en histograma de estos, se asemeja a la representación de la **Campana de Gauss**:

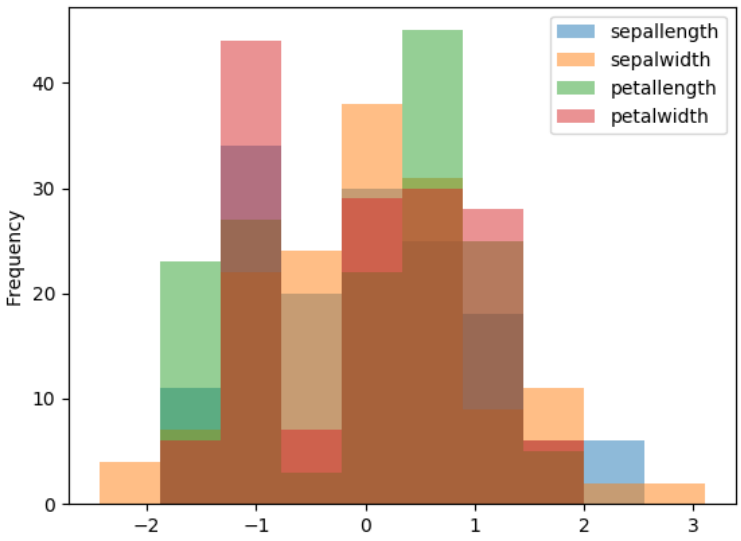

Figura 12: Ejecución del Ejercicio 3 de la práctica 2. Dataset Iris estandarizado.

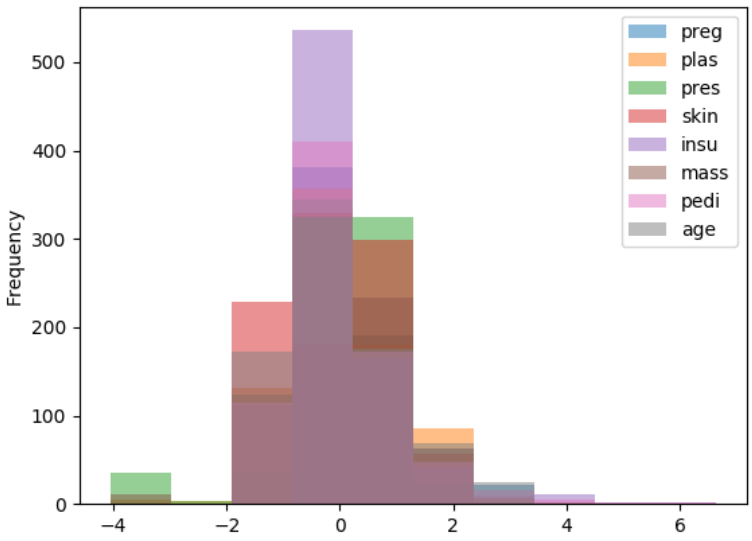

Figura 13: Ejecución del Ejercicio 3 de la práctica 2. Dataset Diabetes estandarizado.

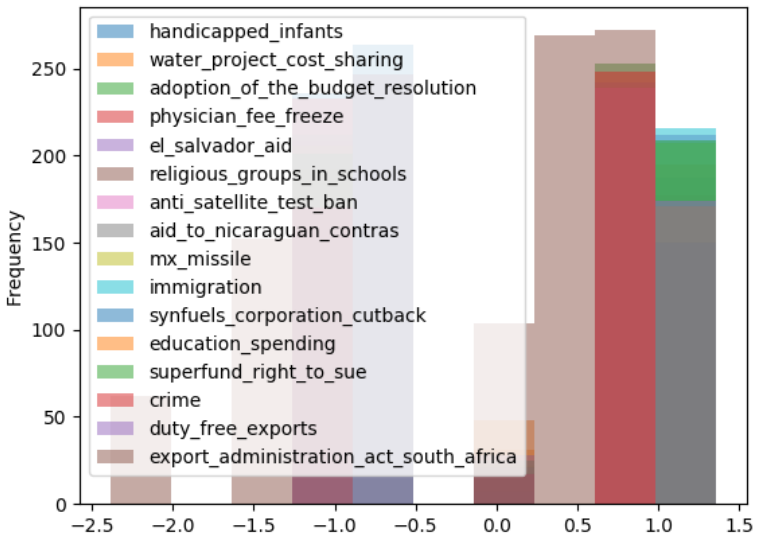

Figura 14: Ejecución del Ejercicio 3 de la práctica 2. Dataset Vote estandarizado.

3.4 Ejercicio 5.

Usando la visualización del diagrama de dispersión (scatter plot) estudie qué información puede obtener de dicha representación gráfica.

Listing 10: Práctica 2.Ejercicio 5.

```
from sklearn import preprocessing
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.io import arff

def ej5function(file, classname):
        data = arff.loadarff(file)
```

```
        df = pd.DataFrame(data[0])
        sns.pairplot(data=df, hue=classname)
        plt.show()

ej5function('Datasets/iris.arff', 'class')
ej5function('Datasets/diabetes.arff', 'class')
ej5function('Datasets/vote.arff', 'Class')
```

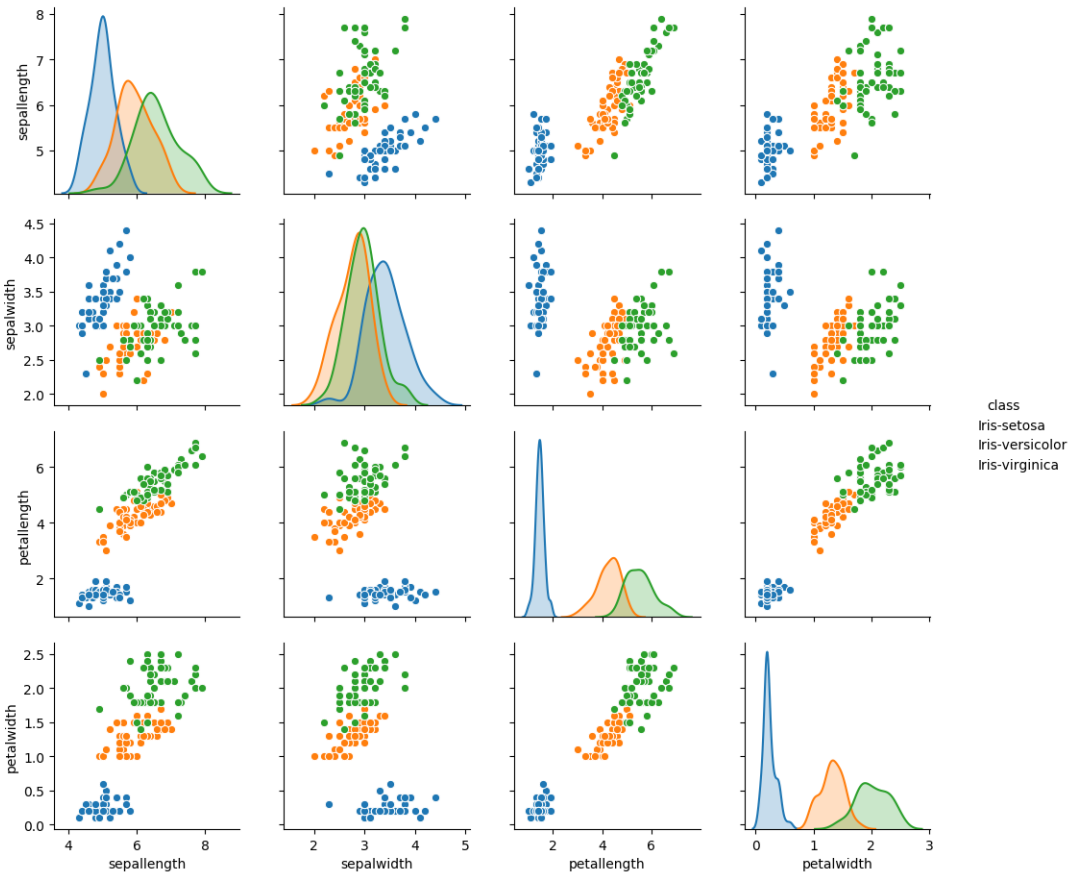

Figura 15: Ejecución del Ejercicio 5 de la práctica 2. Dataset Iris.

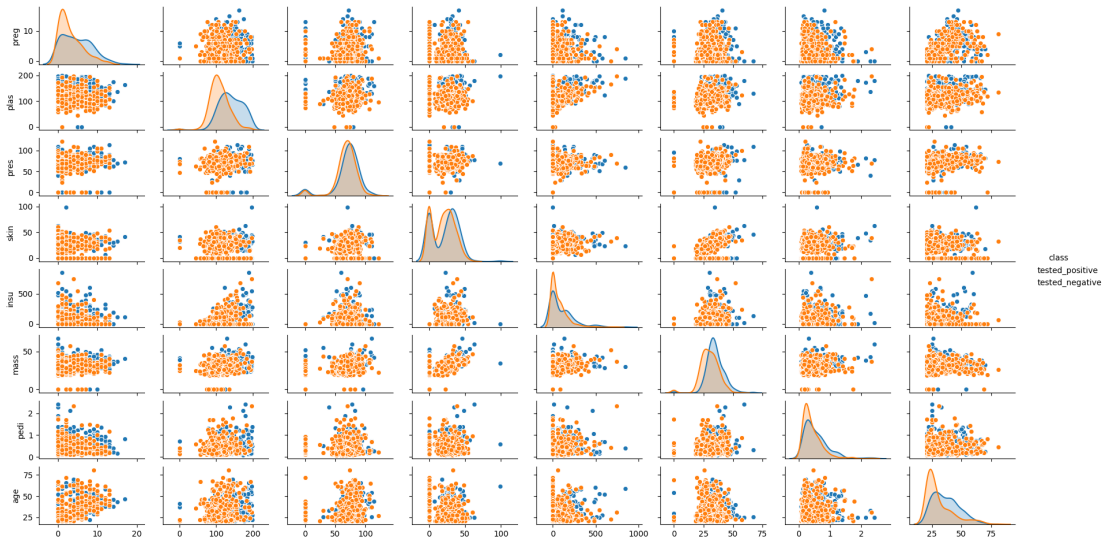

Figura 16: Ejecución del Ejercicio 5 de la práctica 2. Dataset Diabetes.

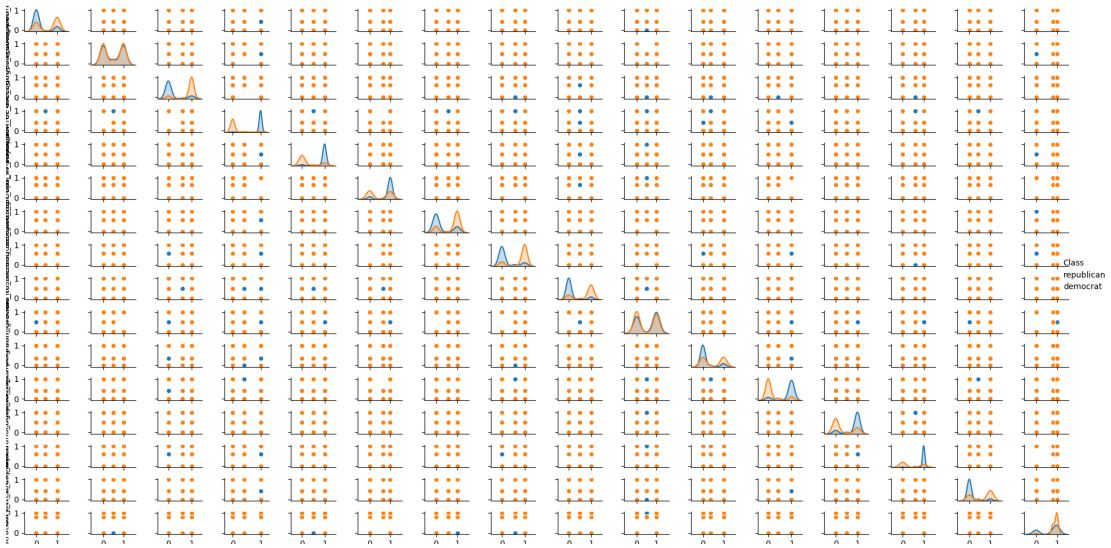

Figura 17: Ejecución del Ejercicio 5 de la práctica 2. Dataset Vote.

3.5 Ejercicio 6.

Estudie el efecto de la normalización y la estandarización sobre el diagrama de dispersión.

A los diagramas de dispersión no les afecta que se normalice y/o estandarice.

3.6 Ejercicio 8.

Estudie el diagrama de correlaciones de los tres conjuntos e indique qué información relativa a las diferentes clases puede obtener.

Listing 11: Práctica 2.Ejercicio 8.

```
from sklearn import preprocessing
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.io import arff

def ej8function(file):
        data = arff.loadarff(file)
        df = pd.DataFrame(data[0])
        corr = df.corr()
        sns.heatmap(corr, xticklabels=corr.columns,
            yticklabels=corr.columns)
        plt.show()


ej8function('Datasets/iris.arff')
ej8function('Datasets/diabetes.arff')
ej8function('Datasets/vote.arff')
```

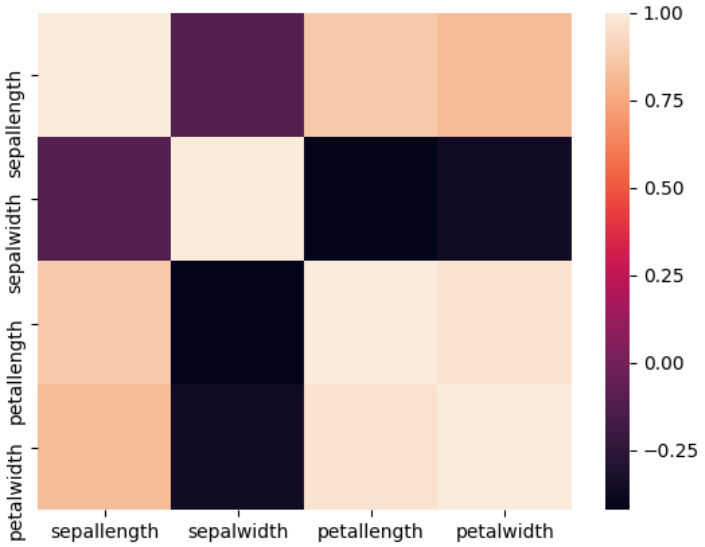

Figura 18: Ejecución del Ejercicio 8 de la práctica 2. Dataset Iris.

Se puede observar en 18 que casi todas las variables están correlacionadas unas con otras (a excepción de sepalwidth y sepallength, quizas).

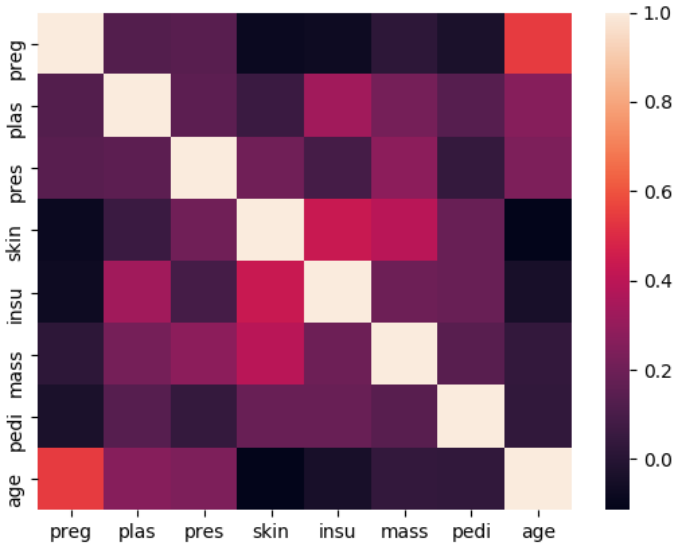

Figura 19: Ejecución del Ejercicio 8 de la práctica 2. Dataset Diabetes.

En la 19, se observa como no hay relacion apenas entre las distintas variables.

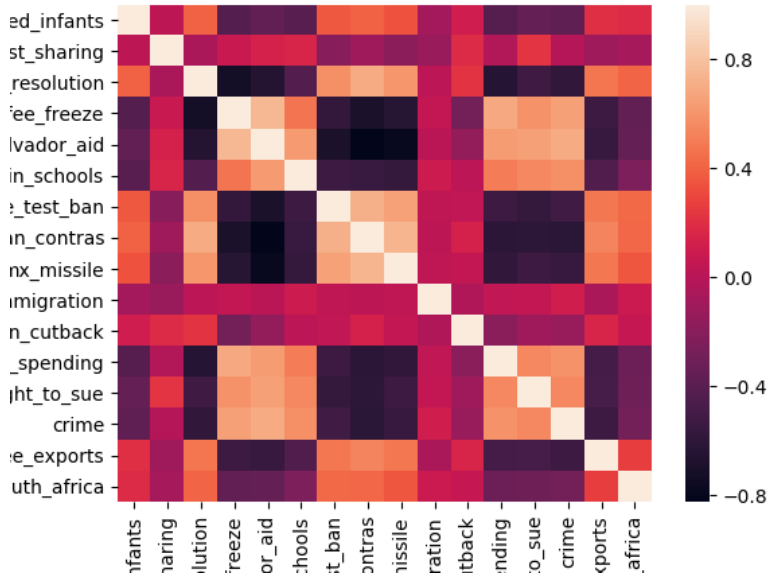

Figura 20: Ejecución del Ejercicio 8 de la práctica 2. Dataset Vote.

En la 20 hay variables que están un poco correlacionadas y otras que no esta nada correlacionadas entre sí.

3.7 Ejercicio 9.

Estudie la representación en coordenadas paralelas de los tres conjuntos e indique qué información relativa a las diferentes clases puede obtener.

Listing 12: Práctica 2.Ejercicio 9.

```
from sklearn import preprocessing
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.io import arff
```

```
def ej9function(file, classname):
        data = arff.loadarff(file)
        df = pd.DataFrame(data[0])
        pd.plotting.parallel_coordinates(df, classname)
        plt.show()

ej9function('Datasets/iris.arff', 'class')
ej9function('Datasets/diabetes.arff', 'class')
ej9function('Datasets/vote.arff', 'Class')
```

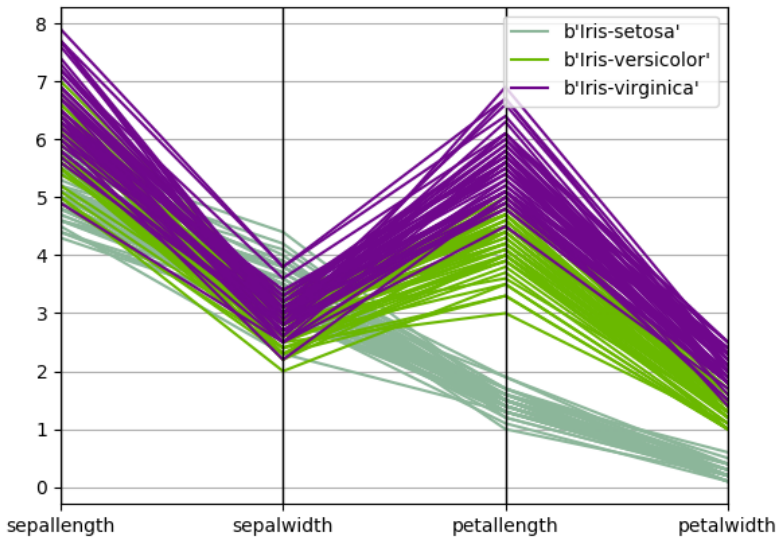

Figura 21: Ejecución del Ejercicio 9 de la práctica 2. Dataset Iris.

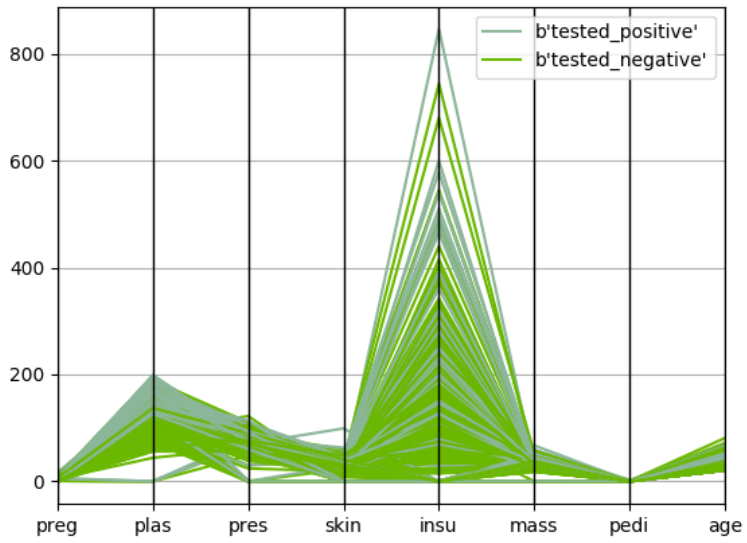

Figura 22: Ejecución del Ejercicio 9 de la práctica 2. Dataset Diabetes.

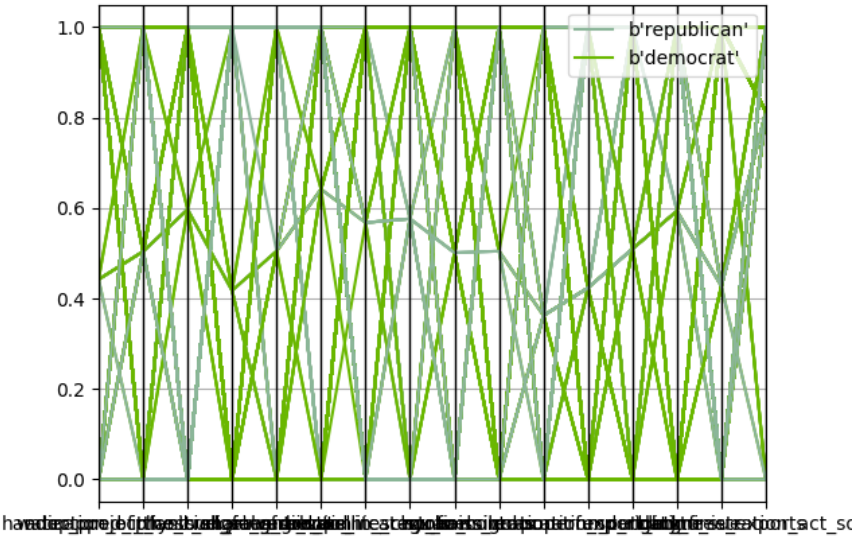

Figura 23: Ejecución del Ejercicio 9 de la práctica 2. Dataset Vote.

4 Práctica 3. Clasificación y evaluación de modelos

4.1 Ejercicio 1.

1. Obtenga al menos 10 conjuntos de datos en formato CSV, ARFF u otro cualquiera de:
- Weka datasets
- UCI MLR

Los Datasets que hemos utilizado nosotros en dicha práctica son los siguientes:Se han elegido las siguientes BBDD: *iris.csv, diabetes.csv, vote.csv, glass.csv, soybean.csv, bank.csv, car.csv, contact_lenses.csv, , ionosphere.csv y labor.csv.*

4.2 Ejercicio 2.

Seleccione al menos 3 clasificadores dentro de los disponibles en Scikit. Se recomienda elegir tres de entre los siguientes: árboles de decisión, k vecinos más cercanos, máquinas de vectores soporte y clasificador Naïve de Bayes. No use combinaciones (ensembles) de modelos que serán objeto de una práctica posterior.

Usaremos los siguientes clasificadores: Árboles de decisión (Decision Tree), KNN (K Nearest Neighbors) y SVM (Support Vector Machines).

4.3 Ejercicio 3.

Para cada uno de los problemas seleccionados realice las siguientes tareas:
- Seleccione como método para obtener el error la validación cruzada de 10 particiones o el método hold out.
- Ejecute para cada clasificador seleccionado el entrenamiento y anote el error.

- Represente gráficamente el error obtenido con cada uno de los métodos de clasificación.

DT	**CCR Train**	**CCR Test**
Iris	100%	95%
Diabetes	100%	72%
Vote	100%	94%
Glass	100%	58%
Soybean	100%	92%
Bank	100%	87%
Car	100%	94%
Contact_lenses	100%	51%
Ionosphere	100%	86%
Labor	100%	67%

KNN	**CCR Train**	**CCR Test**
Iris	97%	952%
Diabetes	80%	69%
Vote	96%	92%
Glass	73%	65%
Soybean	93%	90%
Bank	91%	88%
Car	92%	89%
Contact_lenses	89%	50%
Ionosphere	89%	83%
Labor	100%	100%

SVM	CCR Train	CCR Test
Iris	98%	98%
Diabetes	75%	77%
Vote	99%	94%
Glass	36%	33%
Soybean	97%	94%
Bank	88%	88%
Car	98%	96%
Contact_lenses	100%	50%
Ionosphere	95%	94%
Labor	60%	88%

4.4 Ejercicio 4.

Use el test de Wilcoxon de comparación de dos algoritmos sobre N problemas y aplíquelo a dos de los algoritmos anteriores. Obtenga el rango de Friedman para cada clasificador y configuración y represente gráficamente los resultados. Aplique el test de Iman-Davenport sobre los tres clasificadores.

Wilcoxon: statistic = 16, pvalue = 0.78

Friedmanch: statistic = 4.29, pvalue = 0.12.

Iman-Davenport: 2.45

4.5 Ejercicio 6.

Para uno de los clasificadores elegidos utilice una validación de los hiperparámetros con grid search y compare su rendimiento con el método con hiperparámetros fijados a priori.

GridSearch con SVM	**CCR Train**	**CCR Test**
Iris	98%	98%
Diabetes	84%	71%
Vote	99%	94%
Glass	83%	67%
Soybean	100%	92%
Bank	88%	88%
Car	99%	98%
Contact_lenses	100%	50%
Ionosphere	94%	91%
Labor	100%	89%

5 Práctica 4. Clasificación avanzada

5.1 Ejercicio 1.

Seleccione tres algoritmos clasificación de los disponibles en scikit-learn. Para esta práctica se han elegido los mismos clasificadores que para la anterior; *DT, KNN y SVM.*

5.2 Ejercicio 2.

Para cada uno de estos tres métodos de clasificación realice los siguientes pasos usando validación cruzada de 10 particiones: - Aplique el método base a cada uno de los conjuntos y anote los resultados obtenidos. - Aplique el método de combinación de clasificadores Bagging a cada uno de los conjuntos y anote los resultados obtenidos. - Seleccione dos algoritmos de Boosting y aplique estos algoritmos a cada uno de los conjuntos y anote los resultados obtenidos. - Compare si hay diferencias significativas entre ellos usando el test de Iman-Davenport. Si es así, aplique el procedimiento de Wilcoxon para comparar cada método de agrupación con el clasificador base.

CCR Test	**DT**	**DT Bagging**
Iris	98%	97%
Diabetes	72%	74%
Vote	92%	93%
Glass	60%	61%
Soybean	90%	91%
Bank	87%	90%
Car	96%	96%
Contact_lenses	80%	60%
Ionosphere	93%	96%
Labor	93%	87%

CCR Test	**KNN**	**KNN Bagging**
Iris	97%	97%
Diabetes	73%	71%
Vote	89%	91%
Glass	60%	60%
Soybean	88%	89%
Bank	88%	88%
Car	86%	94%
Contact_lenses	60%	60%
Ionosphere	79%	80%
Labor	100%	100%

CCR Test	**SVM**	**SVM Bagging**
Iris	95%	95%
Diabetes	75%	76%
Vote	96%	95%
Glass	32%	25%
Soybean	92%	91%
Bank	89%	89%
Car	97%	96%
Contact_lenses	60%	60%
Ionosphere	96%	96%
Labor	67%	67%

Gradient Boosting	**CCR Train**	**CCR Test**
Iris	100%	95%
Diabetes	88%	75%
Vote	99%	94%
Glass	100%	72%
Soybean	15%	12%
Bank	90%	89%
Car	93%	91%
Contact_lenses	100%	81%
Ionosphere	100%	94%
Labor	100%	93%

Hist_Gradient Boosting	**CCR Train**	**CCR Test**
Iris	100%	94%
Diabetes	99%	73%
Vote	100%	94%
Glass	100%	75%
Soybean	100%	94%
Bank	99%	90%
Car	100%	99%
Contact_lenses	64%	60%
Ionosphere	100%	95%
Labor	86%	80%

Friedman: statistic: 11.83, pvalue: 0.04.

Ivan-Davenport: 2.79

Cómo las diferencias son significativas, dado dicho valor para el test de Iman-Davenport, se calculará a continuación el test de Wilcoxon para comparar los datos obtenidos con los del clasificador base normal:

-Test de Wilcoxon (KNN y Gradient Boosting): statistic: 19, pvalue: 0.39

5.3 Ejercicio 3.

Enuncie las conclusiones del estudio indicando la influencia del clasificador base en el rendimiento de las agrupaciones de clasificadores.

Los clasificadores que mejores resultados obtienen respecto al resto en rasgos generales (ya que unos clasiicadores serán mejores para algunos datasets y otros para otros, así cómo algunos datasets que no son bien clasificados con ningun clasificador de los usados), son los clasificadores base que usamos en la práctica anterior y en esta, tal y como se pueden ver en las tablas comparativas del CCR Test del ejercicio anterior.

6 Práctica 5. Clasificación usando método multiclase

6.1 Ejercicio 1.

Seleccione un algoritmo clasificación de los disponibles en scikit que sea capaz de resolver problemas de más de dos clases y al menos 10 conjuntos de datos de más de 2 clases (puede reusar los de prácticas anteriores).

Usaremos como clasificador para esta práctica el árbol de decisión, que ya usamos anteriormente en las práctica 3 y 4. A continuación, se enumerarán los datasets usados para esta práctica: *iris.csv, glass.csv, soybean.csv, car.csv, contact_lenses.csv, ecoli.csv, iris_2D.csv, post_operative.csv, segment_challenge.csv y segment_test.csv.*

6.2 Ejercicio 2.

Aplique el clasificador base a cada uno de los conjuntos y anote los resultados obtenidos.

DT	**CCR Train**	**CCR Test**
Iris	100,00 %	96,00 %
Glass	100,00 %	53,00 %
Soybean	99,00 %	92,00 %
Car	100,00 %	96,00 %
Contact_lenses	100,00 %	100,00 %
Ecoli	100,00 %	83,00 %
Iris_2D	99,00 %	89,00 %
Post_operative	93,00 %	64,00 %
Segment_challenge	100,00 %	94,00 %
Segment_test	100,00 %	94,00 %

6.3 Ejercicio 3.

Aplique los métodos multiclase one-vs.-one (OVO), one-vs.all (OVA) y error correcting output codes (ECOC) a cada uno de los conjuntos de datos y anote los resultados obtenidos.

OVO	**CCR Train**	**CCR Test**
Iris	100%	97%
Glass	100%	60%
Soybean	99%	96%
Car	100%	96%
Contact_lenses	100%	80%
Ecoli	100%	98%
Iris_2D	100%	92%
Post_operative	94%	66%
Segment_challenge	100%	94%
Segment_test	100%	91%

OVA	**CCR Train**	**CCR Test**
Iris	100%	98%
Glass	100%	49%
Soybean	100%	88%
Car	100%	91%
Contact_lenses	100%	99%
Ecoli	100%	70%
Iris_2D	98%	93%
Post_operative	93%	62%
Segment_challenge	100%	94%
Segment_test	100%	92%

ECOC	CCR Train	CCR Test
Iris	65%	69%
Glass	100%	60%
Soybean	99%	95%
Car	100%	95%
Contact_lenses	100%	78%
Ecoli	100%	82%
Iris_2D	98%	89%
Post_operative	94%	64%
Segment_challenge	100%	96%
Segment_test	100%	91%

6.4 Ejercicio 5.

Compare si hay diferencias significativas entre ellos usando el test de Iman-Davenport. Si es así, aplique el procedimiento de Wilcoxon para comparar cada método multiclase con el clasificador base y los diferentes métodos entre ellos.

Friedman: statistic:5.28, pvalue:0.15

Iman-Davenport: 1.92

Como los resultados obtenidos en el test de Iman-Davenport indican que hay diferencias significativas entre ellos, se compararán todos con todos con el test de Wilcoxon:

Wilcoxon DT y OVO: statistic: 13, pvalue: 0.48 Wilcoxon DT y OVA: statistic: 2, pvalue: 0.043 Wilcoxon DT y ECOC: statistic: 14, pvalue: 0.58 Wilcoxon OVO y OVA: statistic: 12, pvalue: 0.22 Wilcoxon OVO y ECOC: statistic: 6, pvalue: 0.19 Wilcoxon OVA y ECOC: statistic: 25, pvalue: 0.80

6.5 Ejercicio 7.

Enuncie las conclusiones del estudio.

De los tres nuevos métodos de clasificación usados en esta práctica el que mejor resultado obtiene es el de OVO (one vs one). Este clasificador obtiene unos resultados similares al base, ya que con algunas bases de datos mejora y con otras empeora.

7 Práctica 6. Agrupación y evaluación de resultados.

7.1 Ejercicio 0.

Ejecute los programas ejemplo facilitados junto con la práctica para familiarizarse con los conceptos de los algoritmos de clustering de scikit-learn.

Programas de ejemplo de la carpeta "*Ejemplos_de_clustering_con_Python*", ejecutados.

7.2 Ejercicio 1.

Seleccione al menos cinco problemas de los disponibles en lo repositorios usados en las prácticas anteriores. Use problemas que solo contengan atributos numéricos. No olvide eliminar la información de la clase antes de ejecutar los algoritmos.

Los Datasets seleccionados para la realización de la práctica son 5 de los usados en prácticas anteriores; *iris.csv, glass.csv, soybean.csv, car.csv y contact_lenses.csv*.

Estos ficheros de datos se encontrarán en la carpeta "Datasets", cómo hay que eliminar la información de la clase para cuándo se vayan a ejecutar los algoritmos, se almacenarán los Datasets tratados para ello en la carpeta "Datasets_no_class".

7.3 Ejercicio 2.

Seleccione el algoritmo de clustering k-means.

Kmeans	Homogeneidad
Iris	0.8114
Glass	0.3466
SoyBean	0.4265
Car	0.1012
Contact_lenses	0.3431

7.4 Ejercicio 3.

Selecciones los algoritmos de clustering jerárquicos single link, complete link y average link.

Single Link	Homogeneidad
Iris	0.5872
Glass	0.3381
SoyBean	0.4421
Car	0.0557
Contact_lenses	0.3456

Complete Link	Homogeneidad
Iris	0.7023
Glass	0.3282
SoyBean	0.4484
Car	0.0546
Contact_lenses	0.3369

AverageLink	Homogeneidad
Iris	0.7212
Glass	0.0593
SoyBean	0.4512
Car	0.0903
Contact_lenses	0.3451

7.5 Ejercicio 4.

Implemente la medida de evaluación de la calidad de un método de agrupación basada en la correlación entre la matriz de incidencia y la de proximidad. Seleccione una de las medidas de evaluación no supervisada disponibles en scikit-learn.

7.6 Ejercicio 5.

Para cada uno de los problemas seleccionados realice las siguientes tareas:

- Ejecute el algoritmo k-means y evalúe su rendimiento para un rango de valores alrededor del número conocido de clases. Ejecute los algoritmos jerárquicos y evalúe su rendimiento al nivel en el cual tengan el mismo número de grupos que el número de clases del problema.